UNIVERSITY OF SOUTHAMPTON

Optimisation Classification on the Web of Data using Linked Data. A study case: Movie Popularity Classification

by

Gunawan Budiprasetyo

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

March 2019

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

Optimisation Classification on the Web of Data using Linked Data. A study case: Movie Popularity Classification

by Gunawan Budiprasetyo

Data mining algorithms have been widely used to solve various types of prediction models in movie domain. Classification problems especially to predict the future success of movies have attracted many researchers in order to find efficient ways to address them. However, movie popularity classification has become very complicated as it has too many parameters with different degrees. In this thesis, we review a broad range of literature on (1) movie prediction domain and identify related data, a main data source, and additional data sources to address these problems; (2) on data mining algorithms to build more robust classification models to predict movie popularity. To obtain the robust movie popularity classification model, three experiments were conducted. The first experiment examined five single classifiers (Artificial Neural Network (ANN), Decision Tree (DT), k-NN, Rule Induction (RI) and SVM Polynomial) to develop classification models to predict the future success of movie. The second experiment assessed the use of wrapper-type feature selection algorithms to develop classification models of movie popularity. The last one scrutinized two ensemble methods, bagging and boosting in classifying movie popularity. Based upon the finding and analysis, this thesis contributes in four areas: (1) it demonstrates the capabilities of linked data to get external movie-related data sources and shows how additional attributes fom external data sources can be used to improve performances of the classification model based on a single data source; (2) it presents the use of Grid Search to get a set of optimal hyper-parameters of Artificial Neural Network (ANN), Decision Tree (DT), Rule Induction (RI) and SVM Polynomial classifiers so as to get more robust classification model; (3) it proves the use of wrapper-type feature selection using Genetic Algorithm suited to those classifiers either using default or optimized parameters in order to get the robust classification model and (4) it establishes the use of ensemble methods (bagging and boosting) to those classifiers either using default or optimized parameters in order to get the model in question.

Contents

List of Figures

List of Tables

Authorship

I, Gunawan Budiprasetyo, declare that the thesis entitled Optimisation Classification on the Web of Data using Linked Data and Supervised Learning Techniques, the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given.

  With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;

Signed:

Date: 19 January 2018

Acknowledgements

First of all, I would like to express my gratitude to my supervisors: Professor Dame Wendy Hall and Dr Thanassis Tiropanis for their excellent and continued supports, guidance and patience during my study and towards the completion of this thesis.

I would like to thank my external examiner Professor Kalina Bontcheva and my internal examiner Dr David Millard for their excellent exhorts, detailed review and revision during and after the viva.

I gratefully acknowledge the funding sources that supported my PhD. I was fully funded by Indonesian Endowment Fund for Education Ministry of Finance, the Republic of Indonesia for the four years scholarship.

I would like to thank to PhD student friends in Building 32 ECS who have made my time was really enjoyable. Additionally, my PPI Soton friends: Ahsanu, Belfrit, Hanri, Niken, and Nizam, deserve special thanks for their help, support, and friendship.

I would not have finished this thesis if not for my parents, and patience of my beautiful wife, Eka Lilis Suciyati and my beloved children Fadlan Arfa Gunawan and Faeyza Ahmad Gunawan. Being with you, life seems so beautiful.

Gunawan Budiprasetyo

University of Southampton

January 2019

Chapter 1

Introduction

Web evolution has occurred through a network of unlinked documents transforming into our modern web of interlinking documents, data, people and organisations in various and often unexpected ways Hall and Tiropanis (2012). Activities on the web produce an abundance of either structured or unstructured data. While unstructured data is generated by the Web 1.0 and the Web 2.0, structured data is generated by the Web 3.0. The web 1.0 can be considered as being the “read-only” web, whereas the Web 2.0 is the “read-write” web. While examples of the web 1.0 include mp3.com, Britannica Online, content management systems, etc., the web 2.0 encompasses Social Media websites, such as Flickr, BitTorrent, etc Tim (2005). Subsequently, the web 3.0 is a semantic web Hall and Tiropanis (2012). This data is composed using a standard model for data interchange on the web called RDF, clearly structuring the data. Examples of the web 3.0 are DBpedia, linkedmdb, Eurostat, CIA Factbook, etc. The web is increasingly asserting its role as the main globally distributed data source. The World Wide Web, the largest data and knowledge base Che et al. (2013), is a potential gold-mine with its complex and heterogeneous content, offering hidden knowledge and insights into various domains. Moreover, various resources on the web can potentially enrich information, and aids the user in avoiding making isolated decisions Trujillo and Maté (2011).

Data mining has been widely used to identify, discover, extract and analyse large quantities of data by automatic or semi-automatic means to get knowledge or insights Turban et al. (2011) Tan et al. (2013). There has been a shift in orientation in the implementation of data mining techniques in real life, moving from descriptive toward predictive analysis. While descriptive analysis employs unsupervised data mining Berkhin (2006), predictive analysis leverages supervised data mining Kotsiantis (2007). Descriptive analysis is focused on the intrinsic structure, relationship and interconnectedness of data whereas predictive analysis is centralised on supporting decisions which turns data into valuable, actionable information. Classification is a supervised learning technique leveraged to turn data into knowledge by learning a training dataset which contains attributes input and a target output Kotsiantis (2007). The main objective of a classification algorithm is to maximise the predictive accuracy of classification models.

This thesis investigates how to discover knowledge or insights from the web by employing classification techniques in a real-world scenario. By leveraging the related literature, it will develop effective prediction models from the openness of the web data.

1.1 Classification Problems With Web Data

Problems with classification can be considered as optimisation problems, where the goal is to find the best function (model) of representation of the predictive relationships in the data Otero et al. (2012). There are some critical steps to take before employing classification algorithms to get the best model. Fayyad et al. (1996) described the steps as data pre-processing which comprises of a data selection, filtering, cleaning, and transformation. Existing classification techniques were designed to access relevant data at a centralised location, and the data is in structured formats. It is arguably not pertinent to use data mining techniques with data on the web, as its complexities far exceed the nature of any traditional text documents Han and Chang (2002).

Moreover, the data on the web is massive, distributed and heterogeneous. A large number of resources on the web that potentially make up the data have many different types and forms of representation; online data is considerably interconnected, interrelated and inconsistently represented. For instance, the Internet Movie Database (IMDb) [1] represents a movie title as "Furious 7", whereas MovieMeter [2] writes this as "Furious Seven", whilst the Czech website, ČSFD [3] the Czech spelling of "Rychle a zběsile 7". These special characteristics of data on the web, with its many relations, are non-iid (independent and identically distributed) Yang and Wu (2006). For specific data mining purposes, there are no resources which provide high quality, complete, and up-to-date data in one place, and requires the researcher to find more data sources to obtain complete information. Therefore, there is a need to link and integrate multiple data sources. In the example of movie data, one of the possible link factors amongst different movie web resources is a movie title. The challenge of movie title matching in different representations is in selecting a proper string of algorithms and specific record matching techniques Hassanzadeh and Consens (2009). However, the selected algorithms and techniques do not guarantee the obtaining of maximum accuracy Bhattacharya and Getoor (2007) Raimond et al. (2008).

Real-life datasets, especially data on the web, are not ready to use classification techniques for prediction purposes. As reported by Svátek et al. (2014), datasets on the web

[1]http://www.imdb.com/
[2]https://www.moviemeter.nl/
[3]http://www.csfd.cz/

have heterogeneity and an uneven degree of completeness and reliability. Even structured data such as Linked Open Data has mistakes and incompatibilities in consequence of the inherently heterogeneous nature of interlinked datasets coming from very different sources Beek et al. (2014). Real-life data is frequently imperfect, erroneous, incomplete, uncertain and vague Grzymala-Busse and Hu (2000). Therefore, mining data on the web requires more effective pre-processing steps, such as data filtering and integration Che et al. (2013).

As a result, various data mining applications encounter many difficulties when they work on web data. Mining imperfect datasets require constructing a specific, more complex, multi-model classification system Obradovic and Vucetic (2004). A single classification model would most likely not result in satisfactory mining results. Managing the heterogeneity of datasets on the web with attributes that are noisy, collected with different resolutions, or even missing, is a critical task of knowledge discovery on the web.

1.2 Obtaining Better Classification Models

According to existing literature, there is a variety of possible factors to improve the performance of classification techniques, such as: (1) adding background knowledge, (2) selecting appropriate algorithms, (3) attributes selection, (4) employing ensemble (meta-learning) classifiers, and (5) tuning parameters of the algorithm. In this research, we intend to investigate and apply these factors to derive the best prediction model.

Background knowledge can potentially benefit data mining to identify novel, valid and interesting patterns in data, leading to better results in producing more accurate predictive models Paulheim et al. (2014) Ristoski and Paulheim (2016). Both of them advocate leveraging Linked Data Bizer et al. (2009) as the background knowledge because it has grown into an extensive, global, accessible collection of machine interpretable data sets, following a standard such as RDF.

The selection of which specific learning algorithm to use for classification tasks is a critical step. We employ four data mining algorithms which have been selected as the Top Ten Algorithms in Data Mining Wu et al. (2008). The selected algorithms are Decision Tree, k-NN (k-Nearest Neighbour), and SVM (Support Vector Machine). Further, we also use ANN (Artificial Neural Network) and Rule Induction. Decision Tree and Rule Induction are logical (symbolic) learning methods. The ANN is based on the notion of the perceptron Rosenblatt (1962). k-NN comes under instance-based learning. The last is SVM, the newest supervised learning technique Vapnik (1995).

Many existing classification techniques may not perform well if they attempt to cope with datasets containing highly correlated attributes and redundant information. Feature Selection is a process to select an optimal subset of features according to an objective function Holder et al. (2005). To obtain an optimal subset, feature selection removes independent features that may be strongly correlated to one another, and keeps independent features that may be strongly correlated to the predicted feature. Therefore, there is a necessity to combine classification algorithms with feature selection techniques to improve the classification performance Liu and Motada (1998) Guyon and Elisseeff (2003) Liu and Yu (2005). Feature Selection mostly benefits the process as follows: (1) avoids over-fitting (training on highly-related features rather than contingent ones) and improves model performance, (2) gains a deeper insight into the underlying process that generates data, and (3) makes it easier to interpret the outcome of modelling Fodor (2002) Saeys et al. (2007).

The ensemble method produces one strong classifier by combining many weak classifiers. Many researchers have demonstrated in various fields that employing ensemble methods can improve classification performance significantly compared to a single classifier Bauer and Kohavi (1999) Lee and Cho (2010) Graczyk et al. (2010). The ensemble method constructs an ensemble of individual classifiers that are diverse, yet accurate, to improve the classification performance of any constituent classifiers Dietterich (2000). The ensemble method also improves generalisability or robustness, to minimise overfitting and overcome the modelling bias of individual models dos Santos et al. (2009) Gaber and Bader-El-Den (2012). We employ two different types of ensemble methods: bagging and boosting.

It is quite common to not achieve satisfactory results from supervised data mining. One of the possible reasons is that the selected classification algorithm is not in tune with appropriate parameters Kotsiantis (2007). It is imperative to choose appropriate classification algorithms as well as adjust their parameters to obtain a satisfactory predictive model Leung et al. (2003) Díaz-Uriarte and de Andrés (2006). However, parameter tuning manually can be a labour-intensive task especially when the learning algorithm has many parameters Friedrichs and Igel (2004). In this research, we employ a Grid Search algorithm to obtain proper parameters of the selected classification algorithm.

1.3 Motivations

In thesis, we do not propose or develop new algorithms, but are motivated to investigate and analyse how to improve data mining results by using two approaches. The first approach focuses on the level of data. The second explores how various existing data mining techniques can be used to get optimal data mining results. For the sake of simplicity, we limit our research to applying a supervised data mining technique: classification.

We are concerned how to enrich the existing data source, therefore producing more classification models. Subsequently, we compare the uses of multiple data sources to build classification models to give better results compared with models built with a single data source. We leverage the Linked Data technology Berners-Lee (2006) to get potential data sources on the Web, as this offers the availability of integrated access to data from a wide range of distributed and heterogeneous data sources Bizer et al. (2009). We utilise Semantic Web Berners-Lee et al. (2001) to integrate the selected data sources, which presents the data in the RDF data model, allowing both humans and machines to understand the data Dimou et al. (2015). For data mining purposes, SPARQL Endpoint is a tool to access the RDF data store. Mining data from distributed data sources and complex data formats are highly difficult, especially if the classification performance is a critical factor. We evaluate the leveraging of linked data to support data mining processes, and explore how far the effectivity of combining multiple data sources can be maintained, or outperform classification results from a single data source with a complete data feature. We aim to use this approach to improve data mining results.

We evaluate five popular classification algorithms: Artificial Neural Network (ANN), Decision Tree (DT), k-Nearest Neighbour (k-NN), Rule Induction (RI) and Support Vector Machines (SVM), to result in an optimal classification model. The first attempt relies on default parameters of these, and we then attempt to optimize their parameters. This approach may contribute to achieving better classification performance. We evaluate a Grid Search algorithm to tune the parameters of the two selected classification learners (Decision Tree and SVM).

It is undeniable that not all attributes of the dataset will contribute to achieving successful data mining. Therefore, feature selection is an indispensable component Guyon and Elisseeff (2003) de Amorim (2009). This serves to improve learning accuracy, and to lead to better model comprehensibility by reducing the number of attributes, and removing irrelevant, redundant, or noisy attributes Liu et al. (2010). In this thesis, we employ Genetic Algorithm and Particle Swarm Optimization (PSO) to perform a feature selection process. Beside the feature selection, ensemble classifiers (Bagging and boosting) may also contribute to improve classification performance.

1.4 Research Objectives

The aim of this research is to address the following research questions:

- How can linked data technology be used to construct datasets for classifying movie ratings? How is the linked data used to create a data set namely IMDb which comes from a single data source? How is the linked data able to combine a single data source (IMDb) and external data sources to create a data set called IMDb++? Also, How does the linked data technology support to find related data sources to create different datasets, such as: FilmAffinity, MovieMeter and RottenTomatoes?

- Does the use of classifiers (ANN, DT, RI, and SVM Polynomial) with parameter optimization using Grid Search outperform classifiers (ANN, DT, k-NN, RI, and SVM Polynomial) with default parameters when applied to all datasets (IMDB, IMDb++, FilmAffinity, MovieMeter, and RottenTomates)? Is the use of IMDb++ dataset able to improve the results of IMDb dataset when they are classified using the classifiers either using default or optimized parameters? What is the best approach to build more robust classification model to predict movie ratings?

- What is the best wrapper-type feature selection algorithm to select the most important attributes of the IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets to build more robust classification model to predict movie popularity? Is the use of (1) the wrapper-type feature selection using Genetic Algorithm (GA) suited to the classifiers (ANN, DT, k-NN, RI, and SVM Polynomial) with default parameters (2) the wrapper-type feature selection using Genetic Algorithm (GA) suited to the classifiers (ANN, DT, RI, and SVM Polynomial) with optimized parameters (3) the wrapper-type feature selection using Particle Swarm Optimization (PSO) suited to the classifiers (ANN, DT, k-NN, RI, and SVM Polynomial) with default parameters, and (4) the wrapper-type feature selection using Particle Swarm Optimization (PSO) suited to the classifiers (ANN, DT, k-NN, RI, and SVM Polynomial) with optimized parameters? Is the use of the reduced-attributes IMDb++ dataset able to improve the results of the reduced-attributes IMDb dataset when they are classified using the classifiers either using default or optimized parameters?

- What is the best ensemble methods to build more robust movie popularity classification model based on the IMDb, IMDb++, FilmAffinity, MovieMeter and RottenTomatoes datasets with full and reduced attributes? Is the use of (1) a bagging or boosting ensemble method to the classifiers (ANN, DT, k-NN, RI, and SVM Polynomial) with default parameters, and (2) a bagging or boosting ensemble method with the base classifiers (ANN, DT, RI, and SVM Polynomial) with optimized parameters? What is the best approach to generate more robust movie ratings classification model?

1.5 Research Contributions

We present the outputs of interdisciplinary research combining semantic web technology and data mining techniques, specifically by establishing predictive models of the semantic web data. Our key contributions can be summarised as follows:

- To propose the use of linked data to broadly explore data sources on the web, gaining more relevant data sources for classification tasks.
- To apply and analyse whether the use of multiple data sources will give better classification results compared to the use of a single data source.
- To apply and analyse the performance of the five classification algorithms (ANN, Decision Tree, k-NN, Rule Induction and SVM Polynomial) to classify semantic web data coming from multiple data sources.
- To implement and analyse the use of a Grid Search Algorithm to tune parameters of the following classifiers: ANN, Decision Tree, Rule Induction and SVM Polynomial.
- To apply and analyse the use of Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) to improve classification results by performing a feature selection process on the semantic web data.
- To apply and analyse the use of ensemble classifiers (bagging & boosting) to boost the classification performance of the semantic web data.

1.6 Thesis Structure

The remaining chapters of this thesis are structured as follows:
Chapter 2 provides related literature review of this thesis. We start by outlining a brief review of the Semantic Web and Linked Data. Subsequently, we elaborate about data mining techniques. Various techniques in data mining such as classification algorithms, parameter optimization, ensemble methods and feature selection are described in this chapter.
Chapter 3 presents research methodology. This chapter illustrates the overall methods included to address the research questions of this thesis.
Chapter 4 describes data collection and preparation. We used a Linked Data technology to get relevance data sources. It is applied a broad range of techniques (SPARQL Query, API, and Web Scraping) to extract data from various distributed data sources on the web. The following data is linked and integrated into a semantic data warehouse. Based on the existing semantic web data, it is prepared to build classification models.

Chapter 5 explains the implementation of the five single classification algorithms on the semantic web data. It is evaluated whether the combination of multiple data sources contributes to better classification results.
Chapter 6 is about the leveraging of feature selection to refine attributes from the noisy ones before they include to deliver classification models. We employ the Genetic algorithm to perform feature selections tasks.
Chapter 7 is about the implementation of ensemble classifiers (bagging & boosting) to improve the classification accuracy on the Semantic Web Data. The bagging and boosting methods are applied to get better accuracy of five single classifiers: ANN, Decision Tree, k-NN, Rule Induction and SVM. Subsequently, it is about the use of a Grid Search Algorithm to improve the existing classification results. It is done by optimizing parameters of Decision Tree and SVM classifiers.
Chapter 8 presents a final summary of our research represented in this thesis. We discuss and summary the following topics: (1) linked data to support classification process; (2) feature selection; (3) ensemble classifiers; (4) DT and SVM with parameter optimization; and (5) comparisons all supervised learning methods.
Chapter 9 describes conclusions and future works.

Chapter 2

Literature Review

2.1 Movie Prediction Domain

A number of researches have been conducted in the movie domain to make movie predictions. Researchers apply various data mining techniques to extract interesting patterns and trends to get hidden knowldege which can be useful in predicting movies for different purposes. According the purpose of prediction, the researchs in the movie prediction domain may be categorized into three types. The first type is the prediction of popularity success of movies (e.g. Saraee et al. (2004) Asad et al. (2012) Apala et al. (2013) Latif and Afzal (2016)). The second one is the prediction of financial success of movies (e.g. Sharda and Delen (2006) Delen and SHARDA (2010) Ghiassi et al. (2015)). The last one is the prediction of movie contents (e.g. Changkaew and Kongkachandra (2010) Kabinsingha et al. (2012)).

2.1.1 The Prediction of Popularity Success of Movies

According to Canet et al. (2016), Saraee et al. (2004) had already begun to analyse user rating in the IMDb data using data mining techniques. Also, in a paper of Kabinsingha et al. (2012), it has been stated that Saraee et al. (2004) proposed the use of data mining method to predict movie popularities. Saraee et al. (2004) downloaded the IMDb plain text files from [1] and developed a Java application to parse those files. They introduced the transformation method to calculate a numerical rating for the directors, actors and actresses. They argued that it was imperative to eliminate the bias of an unknown movie with a few high votes. Therefore, they only calculated the average of the films which had at least one thousand votes. With numerical ratings for the actors and actresses, Saraee et al. (2004) applied a sum function of all the ratings for each actor and actress involved in a particular film, to get an "Actor Rank" and "Actress Rank"

[1] http://www.imdb.com/interfaces

for that film. To get a "Director Rank" from the numerical ratings of directors, Saraee et al. (2004) applied an average function, as they argued that more high-rated directors working on a film will not assure it better than one or two directors will. Finally, Saraee et al. (2004) generated a classification model using the three following input attributes: director rank, actor rank and actress rank. The input attributes were used to predict an output attribute which was the rating of a movie. Saraee et al. (2004) classified the rating into levels as excellent, average, poor, and terrible. In terms of data training and testing, Saraee et al. (2004) only considered movies whose budget was in US dollars and movies due for release later 2004 or in 2005-2006

Asad et al. (2012) continued the succes work of Saraee et al. (2004) to predict movie popularities. They did not only using the IMDb plain text files but also introducing a new additional data source (BoxOfficeMojo[2]). To minimize the effort of data preparation, Asad et al. (2012) employed an aided tool name JMDB [3] to import the IMDb plain text files into a MySQL database. From the IMDb database, Asad et al. (2012) selected the five following attributes: director, actor, actress, votes (number of votes as given by IMDb users), and rating (IMDb user ratings as generated by IMdb) to consider as the input attributes of the prediction model. Asad et al. (2012) adopted the Saraee et al. (2004) procedures to count ranks for directors, actors, and actresses as well as to consider an output attribute. The output attribute is classified into classes as excellent, average, poor, and terrible. Besides the IMDb database, Asad et al. (2012) also considered the BoxOfficeMojo as the additional data source to collect movie financial data, such as budget. The web scraping technique had been implemented by Asad et al. (2012) to obtain the budget data from the BoxOfficeMojo website. Finally, the classification model of Asad et al. (2012) comprised six input attributes(director rank, actor rank, actress rank, votes, and rating). For the experimentation data, they put into four filters. The first filter is movies budget in USD. The second one is movies release year >2000 and <2011. The third one is movies language in English. The last one is movies country in USA. To mine actionable information between the input attributes and the output attribute, Asad et al. (2012) selected Decision Tree (C4.5) and Decision Rules (PART) as the classifiers. Both classifiers were implemented using a well-known data mining tool namely WEKA [4] with a default setting 10-fold cross validation. When applied to 820 instances, Decision Tree classifier can correctly classify 77.3562% instances and incorrectly classify 22.6438%, while Decision Rules with PART can correctly classify 77.7234% instances and incorrectly classify 22.2766%.

Apala et al. (2013) proposed a model to predict potential audiences reception of a movie using social media data (YouTube and Twitter). They classified the prediction in three classes: hit, neutral, and flop. The eight following attributes (genre, popularities (director, actor, and actress), view & comment count, sequel movie, and sentiment

[2]`http://www.boxofficemojo.com`
[3]`http://www.jmdb.de`
[4]`https://www.cs.waikato.ac.nz/ml/weka/`

analysis) were set as inputs for the prediction model. The model was constructed by using 35 US movies data to be released between May and July 2013. Those movies were identifed from the website [5]. Based on a list of those movies, it was tracked the trailer released for the corresponding movie from the website [6] to get the YouTube video id number, trailer name, and release date. Apala et al. (2013) collected the YouTube and Twitter data for a one month period from February 28, 2013 to March 28, 2013. They extracted the view count and total number of comments from YouTube on daily basis. For the "Genre" input attribute, they measured the values based on the frequency weight distribution of the movie genres. It was tallied from the 50 top listed movies between years 2010 and 2011 based on the ratings from IMDb. For the input attributes of Actor, Actress, and Directory popularity, Apala et al. (2013) assigned the values by counting their followers on Twitter. For the "View and Comment Count" input attribute, they counted the total number of views and comments for the corresponding movie official trailer from YouTube. Apala et al. (2013) classified the "Sequel Movie" input attribute into two values "Yes" and "No" and obtained the sequel list from IMDb. They assigned a numerical value 10 to "Yes" and 1 to "No". For the last input attribute "Sentiment Analysis", Apala et al. (2013) performed sentiment analysis to classify the whole set of comments extracted from YouTube into three distinct values: positive, neutral, and negative. Apala et al. (2013) divided their experiments into three steps. Firstly, they applied a min-max algorithm to normalize the experiment data. Secondly, they employeed a K-Means algorithm to cluster the data. They clustered it into three classes: hit, neutral, and flop. In clustering, they applied two methods to assign weights to input attributes, such as: Uniform Weighted Approach and Non-Uniform Weighted Approach. Thirdly, they generated a predictive model. The clustering result was used to label each data instances to create a training set to generate a predictive model, as well as to create a training set to test the predictive model. Apala et al. (2013) employed two classifiers (Decision Tree J48 and Naïve Bayesian) to generate two predictive models validated by 3-fold cross validation.

A recent work in the prediction of movie popularities had been done by Latif and Afzal (2016)). They employed web scraping techniques to extract movies on Wikipedia film that were released from year 2004 to 2014. These processes resulted in specific ULR's to the IMDb website. Based on those URL's, Latif and Afzal (2016)) performed the extraction process on the IMDb web pages by employing web scraping methods. Only those movies were extracted that were English movies and released in USA. By applying those filters were obtained 2000 data points. As the data were obtained from multiple sources on the Web (i.e. IMDb and Wikidata film). The data were more likely having inconsistent, missing, and noisy. To deal with missing values for attributes, Latif and Afzal (2016) applied central tendency as it had been done by the previous researcher Nithin et al. (2014). In their predictive model, they included the eight following independent

[5]http://www.comingsoon.net/
[6]http://themoviebox.net/

variables (MPAA Rating, Genre, Awards, Number of Screens, Opening week business, Budget, Metascore, and Number of votes) and determined a dependent variable (rating) to predict. The dependent variable was discretized into four classes (Excellent, Average, Poor, & Terrible) using the same approach as used by Saraee et al. (2004) Asad et al. (2012). Similar to the previous work by Sharda and Delen (2006), Latif and Afzal (2016)) also included the following independent variables: MPAA Rating, Genre, and Number of Screens in their prediction model. Ericson and Grodman (2013). They used WEKA to perform supervised learning as well as selected some linear and non-linear classifiers. Six different classifiers (Logistic Regression, Simple Logistic, Neural Network, Decision Tree (J48), Naïve Bayes, and Decision Rules (PART)). were used to train and test classification models. As the previous researcher Asad et al. (2012), they tested and validated classification results using 10-fold cross validation. Latif and Afzal (2016)) achieved the best two accuracies with Simple Logistic and Logistic Regression 84.34% and 84.15%, respectively. The other classifiers J48, Naïve Bayes, PART, and Neural Network yielded accuracies 82.42%, 79.66%, 79.52%, and 79.07%, respectively.

2.1.2 The Prediction of Financial Success of Movies

Sharda and Delen (2006) have done a work to propose a classification model to predict box-office revenues. They used seven different types of independent variables and a dependent variable. While the indpendent variables were MPAA Rating, Competition, Star Value, Genre, Special Effects, Sequel, and Number of Screens, the dependent variable was Box-Office Gross Revenues. The dependent variable was discretized into nine classess (1 to 9) from a "flop" to a "blockbuster". The class breakpoints (range in \$ millions) were 1 (< 1), 2 (> 1 and < 10), 3 (> 10 and < 20), 4 (> 20 and < 40), 5 (> 40 and < 65), 6 (> 65 and < 100), 7 (> 100 and < 150), 8 (> 150 and < 200), and 9 (> 200). They used the sample data that was purchased from ShowBiz Data, Inc [7]. They limited the sample data as much as 834 movies which were released between 1998 and 2002. To train, test, and validate the classification model, Sharda and Delen (2006) employed a Neural Network (NN) classifier. A stratified 10-fold cross validation was used to estimate the performance of NN classifier. The stratified 10-fold cross validation was selected rather than regular 10-fold cross validation because it tends to generate results with lower bias and lower variance Kohavi (1995). The NN classifier was implemented using a commercial data mining tool called NeuroSolutions [8]. After experimental runs, it was foud that NeuroSolutions generated the NN that can correctly classify 36.9% instances and incorrectly classify 63.1%.

Delen and SHARDA (2010) attempted to improve the previous work by Sharda and Delen (2006) in predicting movie movies box-office gross revenues. They used the same independent variables and the dependent variable as used by the previous ones. They

[7] http://www.showbizdata.com/
[8] http://www.neurosolutions.com//

added a number of experimentation data as much as 2632 movies that were released between 1998 and 2006. These data were obtained from public movie databases and partially purchased from commercial movie databases. Delen and SHARDA (2010) used three single classifiers (Neural Network, Decision trees, and Support Vector Machine (SVM)) and three ensemble classifiers (Random Forest Breiman (2001), Boosted TreesHastie et al. (2001), and Information Fusion Elder (2003)) to build classification models. Subsequently, they splitted the experimental data into two groups based on movies release years. The first group contained movies released between 1998 and 2005 which were used to train and validate classification models, while the second one was only movies released in 2006 to test the classification models. After experimental runs. it were achieved accuracy results as follows: Information Fusion (56.07%), SVM (55.49%). Random Forest (54.62%), Boosted Tree (54.05%), Neural Network (52.60%), and Decision Tree (40.46%). The best classification model was achieved by applying a classification method, Information Fusion. It can correctly classify 56.07% instances and incorrectly classify 43.93%. Amongst the single classifiers, SVM yielded the highest accuracy by 55.49%. It can not correctly classify 44.51%. Compared to the previous work (Sharda and Delen (2006)), Delen and SHARDA (2010) achieved better accuracy results. The improved results achieved by applying two approaches. Firsly, Delen and SHARDA (2010) added the number of experimental data. Secondly, they applied both single and ensemble classifiers.

According to Ghiassi et al. (2015), Delen and SHARDA (2010) have achieved the most promising results in predicting movies box-office revenues. Therefore, they used the work of Delen and SHARDA (2010) as a baseline to measure how well their models perform againts a leading. Ghiassi et al. (2015) conducted two types of experiment. The first experiment was to establish approach by using the same attributes and methodologies. The second one was to propose an alternative prediction model by adding production budgets, pre-release advertising expenditures, runtime, and seasonality to the independent variables. In the first experiment, Ghiassi et al. (2015) reproduced 1758 of the 2632 movie records (67%) employed by Delen and SHARDA (2010). They procurred the experimental data from a variety of publicly data sources, primarily IMDb and the-numbers [9]. To generate a classification model to predict box-office revenues, Ghiassi et al. (2015) employed a classifier namely Dynamic Artificial Neural Network (DAN2) Ghiassi and Saidane (2005). Out of 1758 movies data, Ghiassi et al. (2015) used 80% of the data to train the classification model, while the remaining 20% was used to test the model. Following experimental runs, the DAN2 can correctly classify 74.40%. There was an improvement of 18.33% compared to the previous result by Delen and SHARDA (2010).

For the second experiment, Ghiassi et al. (2015) remained the same for the DAN2 to classify 354 films data which were released between 1999–2010. The data was purchased

[9]`https://www.the-numbers.com/`

from Kantar, a media research company [10]. Based upon the general practises employed in ANN modeling, they divided for the training data 80% (283) and the testing data 20% (71). Compared to the classification model of Delen and SHARDA (2010), Ghiassi et al. (2015) included the following atttributes: production budget, pre-release advertising expenditures, runtime, and seasonality as the independent variables and removed the following independent attributes, such as: competition, star value, genre, and special effects. Therefore, the final set of independent variabels were MPAA Rating, Sequel, Number of Screens, Production Budget (dollars), Pre-Release Advertising Expenditures (dollars), Runtime (minutes), and Seasonality. Subsequently, Ghiassi et al. (2015) implemented new class breakpoints (range in \$ millions) as follows: I (< 1), H (1-10), G (10-20), F (20-30), E (30-40), D (40-60), C (60-80), B (80-100), and A (100-200) (range of classes in millions). Additionally, they considered F_1 Score as a measuremet to measure the performance of ANN classifier. F_1 score is a measure of the weighted harmonic mean of the precision and recall of a test's accuracy Manning et al. (2008). The best F_1 values achieved for training and testinge were 94.10% and 92.70%, respectively.

2.1.3 The Prediction of Movie Contents

Changkaew and Kongkachandra (2010) employed image processing and text processing to classify movie contents. While the image processing was used to extract colors in the scenes to measure the violence values, the text processing was applied to estimate the bad words in the video movies. They developed the system to classify three video movie ratings: 13+, 15+, and 18+. The system also included two independent attributes, which is the violence value and the language value. Generally, they conducted four main steps in their work. Firstly, they selected 26 the Hollywood's movies from the year 2008 to 2009. The composition was 8 movies for rate 13+, 15+, and 18+, respectively, while 1 movie for rate general and 20-. Secondly, it was a preparation to extract features from the movie files. They used two softwares. The first software was BoilSoft Video Splitter used to trim out the title and credit parts. The second one was FramShots Video Capture. It was used to capture sequentially the image size 66 x 50 pixels in every 0.5 second from the beginning until the end of the movie file. Thirdly, it was to define rules, key image and key words. The rules were divided into the violence rules and the language rules. The violence rules were V0 (not found violence), V1 (found low violence), V2 (found medium violence), and V3 (found high violence). Whilst, the language rules were L0 (not found bad words), L1 (found low number of bad words), L2 (found medium number of bad words), and L3 (found high number of bad words). As the goal was to assign the violence value, so that, a blood picture with size of 20x16 pixels had been chosen as the key image. The key image will be matched pixel by pixel with the spesific image of the movie. The key words were defined to obtain the language

[10] https://www.kantarmedia.com/

value. The key words constitued a group of bad words which were obtained from [11]. The language value was obtained by counting the number of bad words found in movie subtitle according to the reference of the group of bad word. Finally, Changkaew and Kongkachandra (2010) applied a classification algorithm and yielded accuracy around 66.67%. In their paper, they did not clearly explain what classification algorithm used in their experiment.

Kabinsingha et al. (2012) studied a range of attributes in order to classify movie contents. They proposed the use of movie reviews to classify movie ratings according to Motion Picture Association of America (MPAA)'s film rating system. The movie ratings classification was objectived as a guide for parents. Kabinsingha et al. (2012) collected experimental data from IMDb and filtered only films with MPAA ratings: PG, PG-13, and R. They considered film attributes, such as: genre and short description. They employed text processing techniques to generate keywords from the short description attributes. The text processing comprised three main tasks. Firstly, it was required to remove stop words, such as: how, to, and, etc. Secondly, it was needed to transform back nouns and verbs to the original forms by removing –s, -es,- ing,-ed,-er,-ied, etc. Thirdly, it was to counting the word's frequency and to grouping words into word groups. They categorized the word groups into: bad words, sexual, terror, religion, imprison, violence and drug. Fourthly, it levelized the number of word's frequency by using two approaches: T-score and mid-point. While T-score approach was to give the frequency value with the score itself, mid-point was to calculate the mid-point for each word group for each rating. After the applying of text processing techniques, it was obtained the final set of independent attributes which were Genre, G_Badword, G_Sexual, G_Terror, G_Religion, G_Imprison, G_Violence, and G_Drug. The Dependent attribute contained three distinct values of MPAA ratings (PG, PG-13, and R).

2.2 Semantic Web

In its present form, the web may be seen as a collection of documents, connections between which can be made from multiple hyperlinks. Web development has been such that it can be used as an application portal – for example, webpages can be used to access information about movies. The search for a particular movie may involve completing a search form, in response to which the user receives a list of webpages. This list is compiled from text and web page links according to the way the documents behind the webpages have been marked up. Berners-Lee et al. (2001) describes the web as it presently is as a medium by which computers can automatically process not information but documents.

[11]http://www.noswearing.com/list.php

Berners-Lee et al. (2001) also made popular the Semantic Web, the primary goal of which is to create a generic infrastructure for Web content that can be processed by machine. In 2004, the World Wide Consortium [12] said, "The Semantic Web is a vision: the idea of having data on the Web defined and linked in a way that it can be used by machine not just display purposes, but for automation, integration and reuse of data across various applications". The Semantic Web can be seen as a network of data that is not only linked but is also available for machines to understand, reuse and interpret it. Stumme et al. (2006) sees the way to achieve those goals as:

- Making available a common syntax for statements understandable by machine.f
- Establishing vocabularies that are universally understood and used.
- Agreeing on a universally understood and used logical language.
- • Exchanging proofs by means of the agreed logical language.

Figure 2.1 shows the layer structure that Berners-Lee[13] suggested for the Semantic Web which are compliant with the above steps.

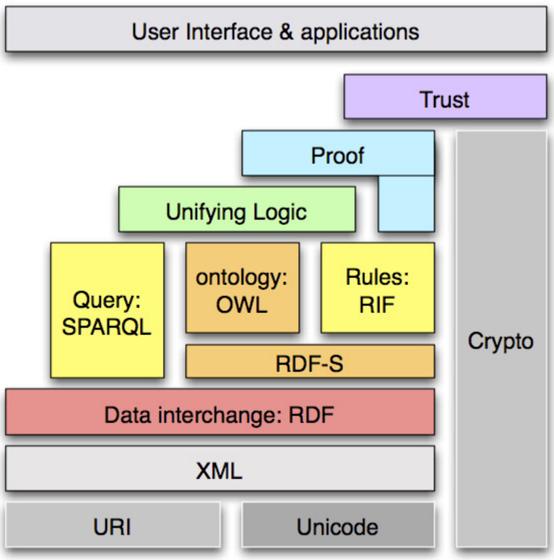


FIGURE 2.1: The Semantic Web Architecture [13]

Each of the first two layers has three blocks (URI, Unicode & XML) which provide a common syntax. URI (Uniform Resource Identifiers) identify resources on the web; Unicode is a standard for encoding the main characters in the majority of languages; and XML (Extensible Markup Language) defines the rules by which documents can be encoded in a human- and machine-readable format, thus providing a notation to describe labelled trees. XML Schema uses the definitions of grammars to validate XML

[12]http://www.w3.org
[13]http://www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html#(14)

documents. A further use of XML is the exchange of data between applications Otto et al. (2001). It should be noted that what XML formalises is the structure and not the content of a document.

Above the XML layer is the RDF (Resource Description Framework), the first layer that transforms information into a format understandable by machines. RDF comprises entities of three types: resources, properties, and statements. A resource may be anything a user wishes to describe, and is identified in RDF by URIs. A property may be characteristics, attributes, or relationships that describe resources. Statements consist of a resource and a property, and the value of a statement may be a literal, a resource, or another statement.

RDF-S is an RDF schema that defines a simple modelling language sitting on top of RDF and including kinds and classes of resources together with properties specific to them. Their use is for semantic annotation, but also to denote classes and properties that are expected to be used in conjunction with each other Manola and Miller (2004). After RDF-S comes the ontology:OWL layer. Gruber (1993) defines an ontology as " an explicit specification of a conceptualization". An ontology it's an explicit representation of what terms in a vocabulary mean and of the relationships that exist between those terms. The Ontology Web Language (OWL) builds on XML, RDF and RDF-S to facilitate the interpretation of web content by machine through the provision of additional vocabulary and formal semantics McGuinness and van Harmelen (2004). OWL can be written by using the RDF/XML syntax Gandon and Schreiber (2014).

The Rule Interchange Format (RIF) is a standard through which rules can be exchanged between heterogeneous rule systems, and particularly between Web rule engines Kif (2013). XML is the language used to define a rule, with the result that the rules can be executed by machines. SPARQL (Simple Protocol and RDF Query Language) is a standardised query language for handling RDF data in quantity Prud'hommeaux and Seaborne (2008). It makes it possible to execute queries over a diversity of data sources on the Semantic Web and operates both for data stored natively as RDF and for data requiring middleware to allow it to be viewed as RDF. A Berners-Lee presentation, "The Semantic Web" [14], defined the status of the unifying logic layer as follows:

- It provides a universal language for monotonic logic
- Any rule system can export, generally cannot import
- No one standard engine - inference capabilities differ
- Many engines exist (SQL to KIF, Cycl, etc)
- Any system can validate proofs

[14] https://www.w3.org/2002/Talks/04-sweb/slide20-0.html

- Web assumptions different from closed world.

Incorporation of this layer means that what begins as a limited declarative language is transformed into a complete logical language, as defined by Turing, possessing functions and inferences.

The remaining layers are: Proof; Trust; and Crypto. In the proof layer, statements made in the Semantic Web are checked for validity according to an agreed set of protocols. This validation is extended and confirmed in the trust layer, while the crypto layer is designed to use encryption techniques to protect all layers below the trust layer.

2.3 The Linked Data

Data published on the web is structured in such a way as to make it sufficiently flexible for linking with other external data sets as well as making it machine readable Bizer et al. (2009). It allows a wide variety of data, both in type and in distribution, to be linked. Previously isolated data and autonomously generated and managed data become open and interconnected. Linked data is presented on the web by means of RDF (Resource Description Framework); the format is RDF Triples (subject-predicate-object triples) Manola and Miller (2004). Representation of a triple's subject, predicate and object are through the use of URIs (Uniform Resource Identifiers). The purpose of subject and object it is to identify resource, URI and string literal, while the predicate indicates the relationship between subject and object.

At the heart of linked data is the use of two technologies fundamental to the web's operation: URIs (Uniform Resource Identifiers) and HTTP (HyperText Transfer Protocol), supplemented by RDF, which provides generic data based on a graph model enabling the structuring and linking of data describing things that exist in the world Bizer et al. (2009). An interesting feature of RDF is its ability to move from data source to data source, discovering additional data. Using URIs to name entities and RDF to link them together by the use of using predicates allows data from multiple sources to be integrated Sahoo et al. (2009). Linked data uses RDF stores and a SPARQL query language to store and query large amounts of RDF data Prud'hommeaux and Seaborne (2008).

Berners-Lee (2006) proposed four rules for the publication and connection of data on the web:

- Use URIs as names for things.
- Use HTTP URIs to enable those names to be looked up.
- When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).

- Include RDF statements that link to other URIs so that they can discover related things.

2.4 Wikidata

Wikidata [15] is a platform for collaboratively acquiring and maintaining structured data. The object is the provision of well-maintained, high-quality data Vrandecic (2012). Moreover, Wikidata is the community-created knowledge base of Wikipedia as well as its the central data management platform Erxleben et al. (2014). Wikidata offers rich data with flexible access. It can be accessed in two ways Vrandečić and Krötzsch (2014). The first is the leveraging by client applications of Wikidata's API to browse the data, to query it, and to edit it. The second is through regular Wikidata downloads (dumps). This research accesses Wikidata online through the Wikidata Toolkit [16].

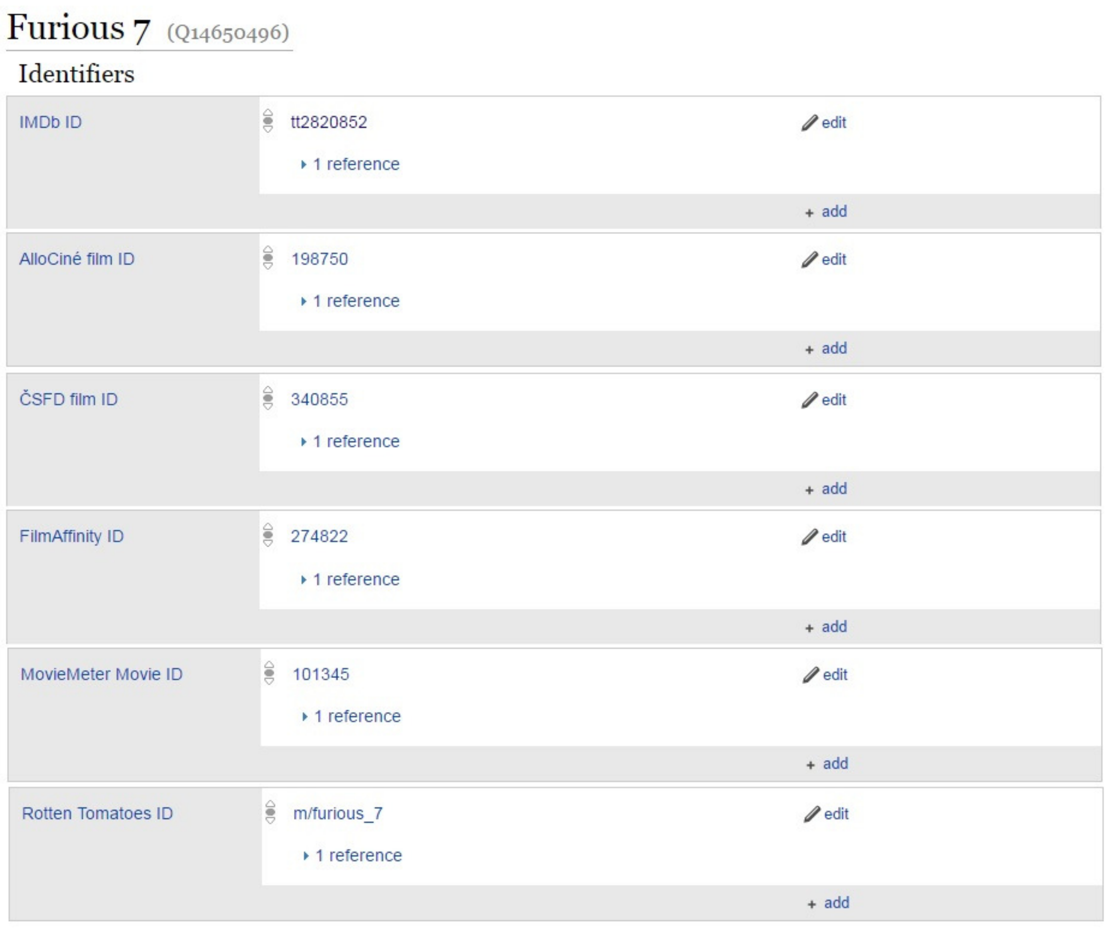
Wikidata's page organisation is similar to that of Wikipedia. Each Wikidata subject that contains structured data is an entity, and every entity includes a page Erxleben et al. (2014). Wikidata recognises two types of entity: items and properties. Item identifiers start with "Q", while property identifiers start with "P." In either case, the letter is followed by a number. So, for example, "Q14650496" is an item for "Furious 7", while "P345" is a property for IMDB (Internet Movie Database) id. There are five main parts to each page: label, description, aliases, statements, and site links. The richest part of the data is the statement, which contains properties; site links inform related external resources. For example, the address of the "Furious 7" movie item page is [17]; Figure 2.2 shows an excerpt.

The identifier part of the address is a property and value pair. As shown in Figure 2.2, property and value pairs may be summarised as in Table 2.1. Each property's value in the identifier part gives a URL that is the address of a movie rating page. In the Furious 7 case, there are some twenty-three properties concerning movie links in the identifiers part, not all of which address our criteria. A manual inspection revealed six websites giving ratings of movies according to user preference.

Table 2.1: The Identifiers Property of the item Q14650496

Identifiers	**Proper ties**	**Values**
AlloCiné film ID	P1265	http://www.allocine.fr/film/fichefilm_gen_cfilm=198750.html
ČSFD film ID	P2529	http://www.csfd.cz/film/340855
FilmAffinity ID	P480	https://www.filmaffinity.com/en/film274822.html
IMDb ID	P345	http://www.imdb.com/title/tt2820852/
MovieMeter Movie ID	P1970	http://www.moviemeter.nl/film/101345
Rotten Tomatoes ID	P1258	https://www.rottentomatoes.com/m/furious_7

[15] https://www.wikidata.org/
[16] https://www.mediawiki.org/wiki/Wikidata_Toolkit
[17] https://www.wikidata.org/wiki/Q14650496



Furious 7 (Q14650496)

Identifiers

IMDb ID	tt2820852	edit
	1 reference	
		+ add
AlloCiné film ID	198750	edit
	1 reference	
		+ add
ČSFD film ID	340855	edit
	1 reference	
		+ add
FilmAffinity ID	274822	edit
	1 reference	
		+ add
MovieMeter Movie ID	101345	edit
	1 reference	
		+ add
Rotten Tomatoes ID	m/furious_7	edit
	1 reference	
		+ add

FIGURE 2.2: The Wikidata Page

2.5 Web Scraping

Its ability to provide access to globally distributed data sources makes the web a source of an enormous amount of information, but not all web resources make available user-friendly data consumption tools. Web scraping, a popular method of extracting textual information from webpages, is still a good way to meet demands for data. It enables new information to be found by the automatic extraction of information from a range of data sources Berry (2003).

One way of extracting information from the web is by using a web crawler – an automated script or software that carries out a methodical and automatic browsing of the web Kobayashi and Takeda (2000). Web scrapers are a more recent development of crawlers and are designed to look for particular kinds of information before aggregating it into new webpages Adams and McCrindle (2008). Using web scraping techniques enables unstructured data in HTML documents to be changed into text in a format useful for particular purposes. It also eliminates the need to copy and paste.

The following steps generally take place in web scraping:

- Site access: communication is established with the target website by means of HTTP. Web servers generally investigate the type of program accessing their content and, in particular, whether it is a browser or a robot. User-Agent is a request header sent by the user through which the server identifies the program, and it is therefore necessary to have the program appear to be a browser. This is done by setting the user-Agent parameters to the name of a browser – Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1), AppleWebKit/537.36 (KHTML, like Gecko), Chrome/32.0.1700.107, Safari/537.36, etc.
- HTML parsing contents extraction: the contents of interest are extracted in one of two ways. One is a regular expression matching which may be simple regular expression or may use additional logic. The other is to use HTML parsing libraries.
- Output building: the extracted text is transformed to a structured format allowing ease of analysis and storage.

In some circumstances, the use of web scrapping may be against the term of use of the site. For the purpose of this thesis, we do not take into account legal issues on applying the web scrapping technique to extract contents from the sites.

2.6 REST Web Service

Web service is the management of two separate electronic devices that allows them to communicate with each other over the web. Leymann (2003) describes web service as a virtual component that can be accessed through a variety of formats and protocols. Web services are loosely coupled, distributed, and heterogeneous. Two technologies are available for building a web service: Simple Object Access Protocol (SOAP); and Representational State Transfer (REST) Serme et al. (2012).

A web service is an API (application programming interface) that hosts the data request service, allowing intercommunication over the web between server and client applications. Examples of REST web service include MovieMeter, and RottenTomatoes. As a light weight approach, a REST does not commit communication between client and server to any specific protocol. As its primary transfer protocol, it uses HTTP (Hyper Text Terminal Protocol). Access to the APIs of the two REST web services just mentioned in order to execute the service in line with the request is through HTTP. The two services also use the Open Authorization (OAuth) 2.0 protocol, which allows users to authorise a client application to access their web resources without being required to share login information. To implement OAuth, API users must apply for access to the APIs key, which operates as the access token during authorisation. Authorisation is complete when the API has inserted authorisation information into the access token and has signed it. Access tokens are reusable but expire.

2.7 Data Mining

Data mining combines a number of statistical, mathematical and AI (artificial intelligence) techniques to extract useful information from large data repositories Hand et al. (2001).

2.7.1 Types of Data Mining

Data mining may be supervised or unsupervised Roiger and Geatz (2003). Supervised data mining uses datasets that contain predefined classes from which it builds classification models before using input attributes to predict output attribute values.

Data mining can also be categorised as directed and directed Berry and Linoff (2000). Directed data mining uses available data to build a model that identifies one variable as a target; examples of the target value are contained in the rest of the data. Classification, prediction and estimation are all techniques used by directed data mining for predictive model construction. Undirected data mining, on the other hand, is designed to establish a relationship among all variables and does not need one variable to be identified as the target. Undirected data mining includes: clustering; rules of association; description; and visualisation Berry and Linoff (2000).

Then again, it is possible to specify three categories of data mining: EDA (Exploratory Data Analysis); Descriptive Modelling; and Predictive Modelling Hand et al. (2001) Tan et al. (2013). EDA sets out to explore the data with no preconception of what is being looked for. Common EDA techniques include interaction and visualisation. In the case of descriptive modelling, the main features of the data are presented with the intention of deriving clusters, correlations, patterns, trajectories, and anomalies concerning underlying relationships in the data. Both of these categories use clustering techniques. The purpose of a predictive model is to predict a specific attribute's value on the basis of the relationship between input and output attributes. Predictive modelling includes techniques such as Classification and regression.

Directed, supervised, and predictive modelling all belong to the same category, which we may describe as supervised learning. It can also be said that unsupervised learning, undirected data mining and descriptive modelling describe similar techniques that can be referred to as unsupervised learning. The focus of this research is supervised learning and the algorithms used for classification.

2.7.2 Algorithms used for classification

The purpose of classification is to start with labelled training data, from there to deduce a relationship or function based on the data, and then to map new and unlabelled data

Kotu and Deshpande (2015). A prediction model can then be created representing the relationship between attribute values and a class value; the prediction model is used to predict unknown data's class label. Classification is the attempt to resolve a problem of optimisation in order to find the model that gives the best representation of the data's predictive relationships Otero et al. (2012).

Classification problems can be solved in a number of ways, and methods have been developed to address them. Each has its own way of extracting relationships from data. The most popular methods can be summarised briefly as follows:

- Artificial neural networks resemble a biological neural network, with neurones acting as nerve cells connecting with each other to build the network. This network can solve a number of complex data problems and help build a good classification model.
- Decision trees are algorithms that seek to resolve the classification problems by purifying input variables into more pure subsets of the data. Those variables that contribute most go into the roof of the tree and the tree model that results is able to predict new unlabelled data.
- k-Nearest Neighbour (k-NN) is an attempt to simplify construction of a prediction model, with particular emphasis on elimination of the problem of extraction of complex relationships from data. It memorises the whole of the training data set as a model and treats relationships in the data as generalised.
- Rule induction infers IF-THEN rules from a data set or decision tree. The rules help data miners to understand relationships between attributes and labels in the data set.
- Support vector machine was originally developed for optical character recognition but has been adapted to solve a number of classification problems. It identifies boundaries between patterns, placing any given data input either within a boundary (that is, belonging to a specific class) or outside the boundary (that is, not belonging to that class).
- Ensemble learners. A single algorithm will not always be enough to provide a good classification model, and ensemble learners make use of "meta" models combined from a number of different models. This approach is used to obtain a good predictive performance when no such performance can be obtained from any single classification method Opitz and Maclin (1999) Polikar (2006) Rokach (2010).

Single classifiers used in this research are: Artificial Neural Network (ANN), Decision Tree, k-Nearest Neighbour (k-NN), Rule Induction and Support Vector Machine (SVM).

2.7.2.1 Artificial Neural Network

The neural network method operates by developing a mathematical explanation closely resembling a neuron's biological process as a way of modelling relationships between input and output variables. A neural network consists of a group of interconnected artificial neurons processing information through a connectionist approach Medler (1998). Each neuron has six basic functional properties: an input device, an integrated device, a conductive device, an output device, a computational device and a representational device Dudai (1989). Implementing the neural network involves multi-layer perception through a model using a forward-feeding neural network trained by a backward-propagating algorithm.

Forward-feeding neural networks are artificial neural networks in which information moves only in a forward direction from input nodes through any hidden nodes that might exist to output nodes. The network does not contain loops. Backward-propagating algorithms fall into the supervised learning category and comprised two phases: propagation and weight. These phases work repeatedly to maintain good performance from the network. Multilevel perception models this neural network by mapping input data sets to an appropriate output set. It makes use of backward propagation to train the network. Key steps in development of an artificial neural network are: (1) determine topology and activation function; (2) initiation; (3) calculate error; (4) adjust weight.

Advantages of neural networks include: the fact that it is data-driven and self-adapting; the universality of functional approximators; the ability to generate non-linear models; and the ability to estimate posterior probabilities Zhang (2000). Neural networks are able to self-adjust to the data in respect of the underlying model, whatever the nature of the data, the function or the form of distribution. They are also able to achieve accuracy in the approximation of any function Hornik (1991). Neural networks provide a degree of flexibility that allows them to model the sort of complex relationship found in real world situations. The final advantage of neural networks is their ability to estimate posterior probabilities resulting from the building of classification rules and the performance of statistical analysis Steppe and Bauer Jr. (1997).

2.7.2.2 Decision Tree

The decision tree is, perhaps, the most popular data mining technique. Its advantages include ease of use such that even new users find it fairly easy to understand. It is robust in its imperviousness to noise, flexible in dealing with redundant attributes, and generates models at low computational cost Barros et al. (2012). Model induction means that its ability to generalise is good Han and Kamber (2001) Tan et al. (2013). There are, however, also disadvantages to the decision tree. Classification accuracy is on the low side when the volume of training data is large Anyanwu and Shiva (2009). When implemented for datasets with a large number of categories, accuracy of classification falls away alarmingly and it is more difficult to find rules based on a combination of variables Omand and Gupta (2014).

The goal of the decision tree is to build a tree structure through a “divide-and-conquer” approach Quinlan (1986). Trees are good at classifying instances after sorting them on the basis of values Kotsiantis (2007) and, since the structure of the tree comprises nodes and leaves, it makes a good model for learning predictive relationships between input values and the desired output Geurts et al. (2009) Barros et al. (2012) Otero et al. (2012). Each inner node tests the value of a specific attribute, while each leaf shows the decision for a particular target class. The decision tree moves down the tree till it reaches a leaf in order to discover a new instance.

This top-down, recursive, divide and conquer approach takes place while a decision tree is being constructed. The process has three steps. The first is selection of the root node’s attributes. Then comes the creation of a branch for each possible attribute value, at which time the tree is also splitting instances into subsets on the basis of possible values so that each branch extending from the node is concerned with a single subset. The third and final step is the recursive repetition or reiteration of the steps for each branch, this time using only the instances that extend that branch. When all instances have the same class, the process comes to an end.

A number of decision tree algorithms exist; they include: Quinlan (1986), C4.5 Quinlan (1993), CART Breiman et al. (1984), M5 Quinlan (1992), REPTree Witten and Frank (2005), OCI Murthy et al. (1993). These algorithms have been widely used for the solution of supervised data mining problems. In the literature for decision tree construction, C4.5 is the most popular algorithm Kotsiantis (2007). Impurity measures for selection of the best attribute when creating a node exist in a number of variants. They include: information gain Quinlan (1986), the gain ratio Quinlan (1993), Gini index Breiman et al. (1984), and distance-based measures de Mántaras (1991). C4.5 uses information gain and gain ratio to find the best attribute and is popular as a means of comparing new classification algorithms Witten and Frank (2005).

2.7.2.3 k-Nearest Neighbours (k-NN)

Generalised relationships in a dataset are used by predicting algorithms to predict new and as yet unseen data. Two types of learner can be seen in the process of prediction: eager learner and lazy learner. Eager learners carry out learning processes in order to find the best approximation of the relationship that actually exists between input and target variables. Lazy learners, on the other hand, use the training dataset as a lookup table with the aid of which they can match input variables to achieve the desired outcome. Such learners do not need to learn from a training dataset.

k-NN is a lazy learner and is a variant of the Nearest Neighbours classification tool Sarma et al. (2013). It works by measuring the similarity of data with the same target class labels in the same region in n-dimensional space Cover and Hart (1967). The whole of the training dataset is memorised and the attributes of any unlabelled records that need to be classified are compared with the whole dataset and the closest match is selected. The label class of the nearest training record forms the prediction of the unseen test record. k-NN is non-parametric and able to work even when the distribution data is little-known or not known at all Altman (1992). It can perform consistently well in comparison with supervised learning algorithms Islam et al. (2010). There is no an explicit model provided in the K-NN and therefore has the potential to throw up difficult problems for solution, especially in the presence of irrelevant attributes and where no precise match of a test data record with the training data is available.

The k in k-NN is the number of close training samples needing to be considered when predicting a new query instance. Where, for example, k=1, the model we will try to find the first nearest sample and will adopt as the prediction value of the first nearest training sample the neighbourhood classification. When $k = 3$, the nearest three training samples are considered. In a two-class problem, the chosen value of k will usually be an odd number, because the prediction value of the query instance is evaluated by voting Peterson (2009). The k-NN algorithm's key task of working effectively is dependent on determining the degree of similarity or dissimilarity possessed by a query instance in comparison with the memorised training sample. A number of techniques exist to quantify similarity between two records. Selection of the distance measure depends on the data Grabusts (2011). All inputs of k-NN will frequently have different measures and units, making comparison between the impossible and requiring that all attributes be normalised in order that the inputs may be re-scaled into a particular range. Decimal Scaling, Min-Max normalisation, Z-scores normalisation and logarithmic transformation are all suitable normalisation methods Roiger and Geatz (2003).

2.7.2.4 Rule Induction

Rule Induction is a commonly used algorithm to handle classification problems in data mining Clark and Niblett (1989) Cohen (1995). The object is the induction of a set of rules capturing all knowledge capable of being generalised from the data while at the same time keeping it as small as possible van den Bosch (2000). Two methods exist of rule induction from data during the learning phase – the direct and indirect method. The first is direct extraction of rules by means of one of a number of rules-based algorithms Kotsiantis (2007). The second is to derive rules from decision trees by splitting the data at every node leading to the leaf in which the class is identified Quinlan (1993).

In the direct method, a rule set is built by sequential covering, an iterative procedure by which rules are extracted from the data in an effort to find every rule in the data, class by class. A specific implementation of sequential covering is the RIPPER (Repeated Incremental Pruning to Produce Error Reduction) Cohen (1995). The RIPPER comprises five sequential steps: class selection; rule development; learn-one-rule; next rule; and development of rule set Tan et al. (2013). The RIPPER produces a set of optimised rules capable of illustrating patterns in the data and constituting the predictive classification model Saian and Ku-Mahamud (2011).

Advantages of rule induction include its ability to classify unknown data and to produce if-then rules in a simple form easily understood by non-expert users. It's also true that rule learning systems provide better solutions of a number of problems than decision trees Cohen (1995). On the other hand, this method is less successful with large samples and with noisy data.

2.8 Feature Selection

Feature selection is important in the preparation of a dataset for data mining. It reduces the amount of noise from the data and excludes irrelevant features as far as that can be done. Efficient, robust selection of features selects from the original dataset the best features to work on. A good selection tool has the power to discriminate, to improve the efficiency with which tasks are learned as well as predictive accuracy performance, and to make the results more comprehensible Yu and Liu (2003) Chen et al. (2012) Seal et al. (2014).

Feature selection falls into two categories: filter method and wrapper method Das (2001). In the filter method, features are chosen on the basis of the training data's general characteristics with no reliance on knowledge of the algorithm that will be used. In the wrapper method, feature selection is built around the learning algorithm that will be used and uses the performance of that algorithm to evaluate features and decide which to choose. It requires that a hypothesis or classifier better learned for each new feature subset and thus has a tendency to find those features that are best suited to the learning algorithm which has already been chosen. The wrapper method is generally more computationally intensive than the filter method Langley (1994) Saeys et al. (2007).

Feature selection techniques search in the feature subset space and must take a position on four issues at the heart of the search process Langley (1994):

- Starting point.

  Choosing a starting point in the feature subset space affects the search's direction and is therefore vital. Three options exist: forward selection; backward elimination; and a random position. In forward selection, the search beginners with no features selected and adds attributes as it moves forward. Backward elimination works in the opposite way, beginning at the end with all features included and removing them as it works backwards. The random option begins at some fairly central point, from which it moves outwards.

- Search organization.

  The simplest search method is to search every possibility in the search space. A dataset containing N features will have a search space of 2N. Where the number of features is large, exhaustive searches are simply not possible. Heuristic search is more feasible for large searches than an exhaustive search but cannot guarantee producing optimal results.

- Evaluation strategy.

  How effective feature subsets are should be evaluated and two strategies exist. They are, as with feature selection, filter and wrapper. The filter method involves the use of a function, while the wrapper method evaluates feature subsets on the basis of the performance of the classifier. Kohavi and John (1997) proposed a wrapper model that used the accuracy of the classifier is the measure of performance. It has been concluded by a number of researchers that the predictive accuracy of the classifier is the factor of greatest importance in what the model is seeking to do is to reduce the classifier's error rate to a minimum, but this would only hold true if the cost of measurement for all features is the same. A wrapper method usually takes longer to complete than a filter method but will normally perform better, having been optimised for the learning algorithm the researchers intend to use Hall and Holmes (2003).

- Stopping criterion.

  A feature selection algorithm that selects features, there must be a criterion that decides when iteration of the search should cease. One example would be the ending of a feature selector or the removal of some features when, after a number of iterations, no improvement has been seen in the classifier's performance.

Many techniques exist for the selection of features, but only two algorithms are used for this research: Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). Both will be analysed in greater detail in Sub Sections 2.8.1 and 2.8.2.

2.8.1 Genetic Algorithms (GA)

The GA was conceived by Holland (1975) with the initial purpose of establishing whether a computer program can be adaptive in the Darwinian sense Eiben and Smith (2003). GAs are efficient search methods Paz (1998) which are categorized as stochastic search algorithms and work on the basis of genetics and natural selection. They are used not

Table 2.2: The General Scheme of a Genetic Algorithm in Pseudocode

```
BEGIN
    INITIALIZE population    with random candidate solutions;
    EVALUATE                 each candidate;
    REPEAT UNTIL             ( TERMINATION CONDITION   is satisfied ) DO
        1   SELECT           parents;
        2   RECOMBINE        pairs of parents;
        3   MUTATE           the resulting offspring;
        4   EVALUATE         new candidates;
        5   SELECT           individuals for the next generation;
    OD
END
```

only for searching but also in the optimisation of problem functions. They perform well in finding optimal and near-optimal solutions Krishna and Murty (1999) and are robust when searching in large spaces and independently of domain. In the view of Malhotra et al. (2011), they are a good match for a number of problems of optimisation where the objective functions are such that the standard optimisation algorithms cannot handle them. The objective functions in question are: discontinuous; non-differentiable; stochastic; and highly non-linear. GAs are, for this reason, used successfully in the search for solutions to problems in fields as varied as business, engineering and science Goldberg (1994).

In order to obtain the best solution to a problem, GAs manipulate a number of candidate solutions. They produce the next generation to reproducing and mating of the best candidates. Over a number of generations, the percentage of good candidates increases and there is a matching increase in the quality of solutions. Three rules are applied for the creation of the next generation:

- *Selection rules*: this rule selects individuals to act as parents in the population.
- *Crossover rules*: this rule brings together two parents to form children for the next generation.
- *Mutation rules*: this rule applies random changes to individual parents to create children.

The general scheme showing how the GAs work Mitchell (1998) can be shown in Table 2.2:

2.8.2 Particle Swarm Optimization (PSO)

Eberhart and Kennedy (1995) suggested a population-based stochastic search technique, PSO (Particle Swarm Optimization). The concept was drawn from the behaviour of birds in a flock or fish in a school up Huang and Dun (2008) describe PSO as a powerful, easy to implement and computationally efficient way of optimising optimal feature subsets and this has been supported by Schuh et al. (2012) and Seal et al. (2014) who described PSO as effective and flexible in obtaining optimum characteristics through local as well as global iterative search in the feature search space. PSO allows near-optimal solutions to be found in search spaces containing local minima Tjiong and Monteiro (2011).

PSO envisages a group of random particles, each of which represents a possible solution to the problem. The particles swarm around the problem's solution space by iterating in the search for the optimum solution and continue until candidate solutions converge. Starting location and velocity are assigned at random on the basis of the following equations:

$$\upsilon_{i,j}^{t+1} = W\upsilon_{i,j}^{t} + c_1 r_1(\rho_{i,j} - \chi_{i,j}^{t}) + c_2 r_2(\rho_{g,j} - \chi_{i,j}^{t}); \tag{2.1}$$

$$\chi_{i,j}^{t+1} = \chi_{i,j}^{t} + \nu_{i,j}^{t+1}, \tag{2.2}$$

in which χ and ν are the position and velocity of the particle i for dimension j, respectively, at time t. W is the inertial weight and represents how much of the previous velocity is retained while exploring. Parameters c_1 and c_2, and r_1 and r_2 are, respectively, the weighting and random factor associated with the local best particle $p_{i,\ j}$ and the global best particle $p_{g,\ j}$.

A particle's local best is, in a sense, PSO's cognitive aspect; its global best becomes its social aspect. PSO "flies" over the local minima towards the global optimum. Nevertheless, PSO is essentially a stochastic algorithm which means that – dependent on the search's initial conditions – particle evolution may vary and the global optimum may not be revealed.

2.9 Summary

In this chapter, we have reviewed literatures in the movie prediction domain, related-technogies to collect and integrate data (Semantic Web, Linked Data, Wikidata, Web Scraping, and REST Web Service), single classifers (Artificial Neural Network, Decision Tree, k-Nearest Neighbour, Rule Induction, and Support Vector Machines), two feature seletion algorithms (Genetic Algorithm and Particle Swarm Optimization), ensemble classifiers (Bagging and Boosting).

Few researches in the movie prediction domain are focused to predict the popularity of movies as well as to identify the contributing factors of movie success in the future. This is caused by an excessive number of parameters of distinctive degrees are connected and finding an appropriate way to speak to all the data identified with a motion picture in a single occasion is an impractical task Asad et al. (2012). There may be only two researchers (Saraee et al. (2004) Asad et al. (2012)) performing transformation to calculate a numerical rating for the actors, actresses, and directors in their prediction modeles of movie ratings. Moreover, they only considered the four following input attributes (Actor Rank, Actress Rank, Director Rank, and budget) and used data source IMDb. Also, while Saraee et al. (2004) generated their model using Excel, Asad et al. (2012) employed two classification algorithms (Decision Tree and Decision Rules). Asad et al. (2012) obtained the best classification accuracy 77.3562% using Decision Tree (C4.5), while it was achieved the best classification result by 77.7234% using Decision Rules (PART).

Existing researches in the movie popularities classification offer challenges to identify more input attributes to the future success of movies and approaches to improve the classification results. This research investigate the use of Linked Data tecnology in constructing data sets that can improve the movie popularities classifation process. Also, a range of supervised learning techniques will be reviewed to get a suitable algorithm for classifying movie populaties. The single classifiers with default parameters, the feature selection algorithms, an algorithm (Grid Search) to optimize parameters for the two classifiers (Decision Tree and Support Vector Machine), and the ensemble classifiers will be applied to train and test different data sets constructed using the Linked Data technology. The next chapter explains the research methodology to perform to achieve the goals of this research.

Chapter 3

Research Methodology

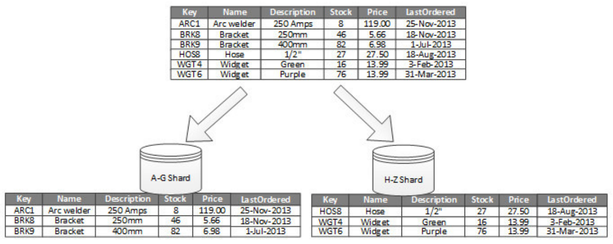
This chapter details datasets and the overarching methodology by which the research is performed. Firstly, datasets used in this research is described, as well as providing an explanation of how each dataset was generated. Secondly, a diagram of the research methodology is used to explain the approach taken to answering the research question mentioned in Chapter 1.

3.1 Introduction

Web content is a gold mine of data available to collect, analyse, and study. According to Lesser et al. (2000), the abundance of information available today on the Web has great potential to improve the quality of decision-making processes. For this specific domain, Asad et al. (2012) argued that the vast amount of movie data across the web makes it an obvious candidate for machine learning and knowledge discovery. However, a single data source often contains insufficient information for a knowledge discovery task Yin and Han (2005). Additionally, it is nearly impossible to get a single data source on the web which contains high quality, complete and up-to-date data Aldarra and Muñoz (2015). Alternately, adding background knowledge from external data sources leads to better results of knowledge discovery Paulheim et al. (2014). The adding process can improve the accuracy level for predictive models and can reveal more interesting findings for descriptive models. Therefore, this research aims to investigate the impact of additional data sources in addressing classification problems, especially performing classification tasks from multiple data sources on the web.

However, the massive size, semantic heterogeneity, autonomy, and distributed nature of the data sources present significant barriers in obtaining useful knowledge from the available data Caragea et al. (2005). According to existing literature, there are several approaches to deal with data mining on multiple data sources. Firstly, the approaches

Table 3.1: Types of Partitioning Data

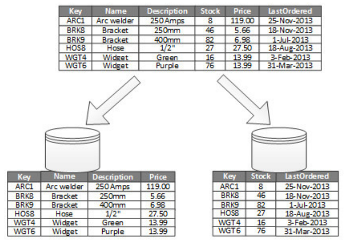


Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc welder	250 Amps	8	119.00	25-Nov-2013
BRK8	Bracket	250mm	46	5.66	18-Nov-2013
BRK9	Bracket	400mm	82	6.98	1-Jul-2013
HOS8	Hose	1/2"	27	27.50	18-Aug-2013
WGT4	Widget	Green	16	13.99	3-Feb-2013
WGT6	Widget	Purple	76	13.99	31-Mar-2013

Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc welder	250 Amps	8	119.00	25-Nov-2013
BRK8	Bracket	250mm	46	5.66	18-Nov-2013
BRK9	Bracket	400mm	82	6.98	1-Jul-2013

Key	Name	Description	Stock	Price	LastOrdered
HOS8	Hose	1/2"	27	27.50	18-Aug-2013
WGT4	Widget	Green	16	13.99	3-Feb-2013
WGT6	Widget	Purple	76	13.99	31-Mar-2013

Horizontally partitioned data
Docs (2016)



Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc welder	250 Amps	8	119.00	25-Nov-2013
BRK8	Bracket	250mm	46	5.66	18-Nov-2013
BRK9	Bracket	400mm	82	6.98	1-Jul-2013
HOS8	Hose	1/2"	27	27.50	18-Aug-2013
WGT4	Widget	Green	16	13.99	3-Feb-2013
WGT6	Widget	Purple	76	13.99	31-Mar-2013

Key	Name	Description	Price
ARC1	Arc welder	250 Amps	119.00
BRK8	Bracket	250mm	5.66
BRK9	Bracket	400mm	6.98
HOS8	Hose	1/2"	27.50
WGT4	Widget	Green	13.99
WGT6	Widget	Purple	13.99

Key	Stock	LastOrdered
ARC1	8	25-Nov-2013
BRK8	46	18-Nov-2013
BRK9	82	1-Jul-2013
HOS8	27	18-Aug-2013
WGT4	16	3-Feb-2013
WGT6	76	31-Mar-2013

Vertically partitioned data
Docs (2016)

can be categorised according to the availability of direct access to data. Massive size of data, access restrictions, or bandwidth requirements can obstruct such access. In such settings, Caragea et al. (2005) Koul et al. (2008) Koul et al. (2010) proposed a statistical-queries technique to perform knowledge discovery without direct access to data sources. When in a setting where direct access to data is available, there are three approaches that can be used to perform knowledge discovery on multiple data sources. The first approach is the traditional way: to integrate the data sources and then to apply the adequate data mining algorithms Rahm and Bernstein (2001) Reinoso et al. (2003) Dhamankar et al. (2004). This is is the traditional way to extract knowledge from multiple data sources. Before applying data mining algorithms, it is required to integrate multiple data sources into a single data source. It is common to face problems of efficiency and privacy concerns during this approach.

The second approach is distributed data mining (DDM) Kantarcioglu and Clifton (2004) Liu et al. (2011) Zeng et al. (2012) which aims to discover knowledge from a data set that is distributed at different sites. However, DDM is not able to handle heterogeneous relational databases, but only considers a homogeneous data set (a single table or a set of transactions) that is distributed to multiple sites Yin and Han (2005) Zeng et al. (2012). Data can be categorised into types of distribution as shown in Table 3.1. The first type is horizontally partitioned data, where each site has a set of complete transactions Kantarcioglu and Clifton (2004). The second type is vertically partitioned data, where each site contains some attributes of a transaction Vaidya and Clifton (2002). The DDM works on a well-structured data table distributed at different sites.

The third approach is cross-database data mining, which aims to explore knowledge in multiple heterogeneous relational databases Yin and Han (2005) Mehenni and Moussaoui (2012), needed to obtain better data mining results. To build data mining models, the main idea of this approach is to create bridges across the multiple heterogeneous databases with some useful links. It is imperative to apply this approach to locate the most useful links in the databases. As the database location may be far or physically separated, this approach is quite often used to face the efficiency problem, although the communication cost is expensive. However, in this research, we are not focused to solve the efficiency problems, but focus on how the heterogeneity data can be exploited to boost the classification results.

In this research, we build classification models based on multiple heterogeneous data sources (as shown in Figure 3.1), including multiple freely open data sources on the web. We explore how the classification of multiple data sources gives better results compared to the classification on a single data source, a question posed in chapter 1. According to the three options mentioned above, we choose the best integration approach to conduct in this research Firstly, this is due to the data not being homogeneous, so that it is not appropriate to apply the Distribution Data Mining approach in this context. As the data sources are from the Web, the data formats are unstructured. It is required that we treat the data in order for it to be used in the data mining processes. Secondly, the availability of direct access to data sources means that we do not apply the statistical queries to build the classification models.

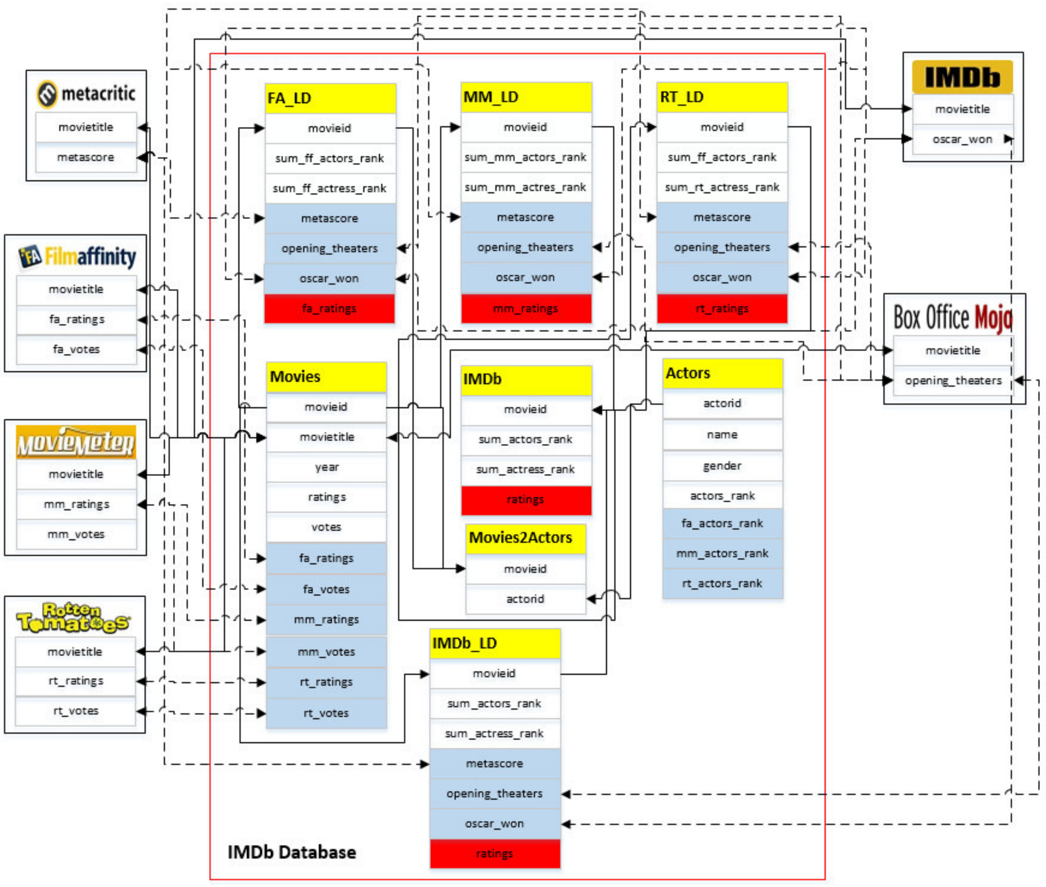


FIGURE 3.1: Movies Relational Data Sources

To address the research questions, we detail the research steps taken in the two following sections. Firstly, we categorise the data sets, to clarify the comparison results between data mining on multiple data sources and a single data source. Secondly, all the classification algorithms used on each of data set types are explained. This will ease the understanding of the impact of adding data sources to improve the classification results.

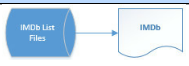
Also, we evaluate whether building classifiers based on multiple data sources will give better accuracy when applied with various data mining techniques.

3.2 Datasets

Figure 3.1 indicates the diagram of relational data sources to build classification models. The primary data source is IMDb [1] which constitutes of downloadable flat text files. As the original files are in an unstructured-format, they require pre-processing steps to be converted into a single database. The additional data sources are websites (BoxOfficeMojo [2], FilmAffinity [3], IMDb [4], Metacritic [5], MovieMeter [6] and RottenTomatoes [7]). We will leverage the existing data sources to readily generate data sets for building classification models. We categorise the data sets into four types, as shown in Table 3.2. The first type of data set is DS1. In figure 3.1, DS1 is represented by the *IMDb* entity in the red box "IMDb database". The IMDb entity contains two different coloured boxes, white and red, respectively. While the white colour shows input attributes, the red one represents an output attribute. Both the inputs and the output are generated from the IMDb database. The goal of classification is to build an accurate classifier for predicting the class labels of the output attribute instances.

The *IMDB_LD* entity in Figure 3.1 represents DS2 which constitutes the second type of data set as described in Table 3.2. The IMDB_LD has a new blue box which denotes input attributes gathered from external data sources. For instance (as in Figure 3.1), the IMDB_LD has the "metacritic" attribute gathered from the Metacritic [5] website. Subsequently, the BoxOfficeMojo [2] contributes to the "opening_theaters" attribute and the "oscar_won" attribute is gathered from the IMDb [4]. The integration process is needed to obtain additional attributes from the external data sources as described in Table 3.2. The adding of new attributes from external sources is expected to enhance the quality and richness of the existing data. The target attribute of the IMDb_LD data set is the same with the IMDb data set which is retrieved from the IMDb database.

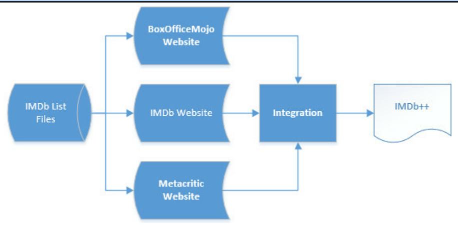
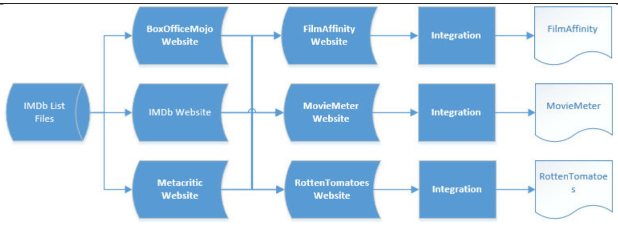
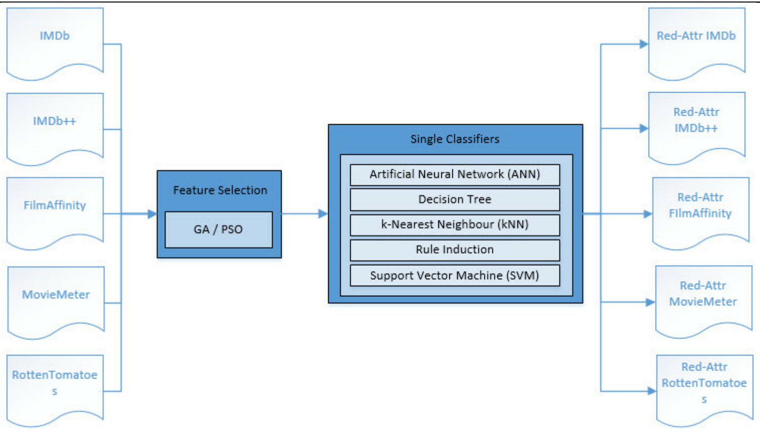
Table 3.2: Types of Data set

Type	Generation Process	Name
1		DS1

Continued on next page

[1]http://http://www.imdb.com/interfaces/
[2]http://www.boxofficemojo.com/
[3]https://www.filmaffinity.com/
[4]http://http://www.imdb.com/
[5]http://www.metacritic.com/
[6]https://www.moviemeter.nl/
[7]https://www.rottentomatoes.com/

Table 3.2 – continued from previous page

Type	Generation Process	Name
2	 IMDb List Files BoxOfficeMojo Website IMDb Website Metacritic Website Integration IMDb++	DS2
3	 IMDb List Files BoxOfficeMojo Website IMDb Website Metacritic Website FilmAffinity Website MovieMeter Website RottenTomatoes Website Integration Integration Integration FilmAffinity MovieMeter RottenTomatoes	DS3
4	 IMDb IMDb++ FilmAffinity MovieMeter RottenTomatoes Feature Selection GA / PSO Single Classifiers Artificial Neural Network (ANN) Decision Tree k-Nearest Neighbour (kNN) Rule Induction Support Vector Machine (SVM) Red-Attr IMDb Red-Attr IMDb++ Red-Attr FIlmAffinity Red-Attr MovieMeter Red-Attr RottenTomatoes	DS4

The FA_LD, MM_LD, and RT_LD found in Figure 3.1 are examples of the third type of data set (DS3) as illustrated in Table 3.2. The total number of attributes of DS3 is the same with DS2. DS3 differs from DS2 in the source of its target attribute. While the target attribute of DS2 is from the IMDb database, the DS3 target attribute is from external data sources. Moreover, in the process of the DS3 data set generation, attribute values gathered from other external data sources are not intended to add new features, however, they are only used as reference values to substitute and recalculate the existing attributes from DS2. For instance, we have a primary data source to generate DS1, consisting of information, such as movies, actresses, and IMDb ratings, as the illustration shows in Figure 3.2. For the purpose of the DS1 generation, it is required to calculate average ranks of actresses with the simple equation $\overline{x} = \frac{\sum X}{n}$. Therefore, the rank value of actress Cindy Hogan in DS1 and DS2 will be $\frac{(7.2+6.2+6.7+5.9)}{4} = 6.5$. Both DS1 and DS2 calculate female cast ranks based on IMDb ratings. In the generation of the DS3 data sets, we explore external data sources providing movie ratings instead of IMDb ratings. For example, there are two other data sources, MovieMeter and RottenTomatoes which have movie rating features. By using their values, we generate the new data sets, namely, DS3i and DS3ii, respectively. In DS3i, we calculate the rank for Cindy Hogan based on the value of MovieMeter rating as follows $\frac{(6.6+5.64+4.3+3.66)}{4} = 5.05$. Further, we calculate Cindy Hogan's rank using the RottenTomatoes rating for DS3ii $\frac{(7.6+6.8+6.8+6)}{4} = 6.8$.

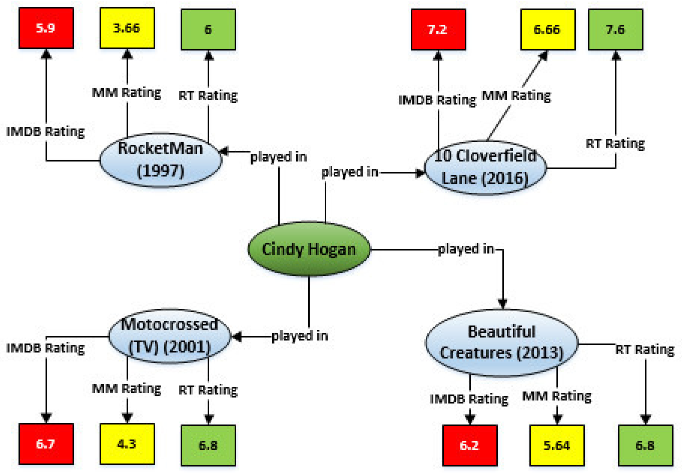


FIGURE 3.2: Sample Actress Rating

The last type of data sets to use in this research is DS4. We intend to obtain compact data sets by purifying the previous type of data sets with feature selection algorithms. Adding new features into an existing data set may have positive contributions to improve classification processes. However, more dimensions in data may pose some problems: the 'curse of dimensionality' as coined by Bellman (1957). Firstly, attributes can become uncorrelated to the outcome of interest. This can lead to classification problems, such as: overfitting, accuracy and reliability. Secondly, superfluous attributes may increase costs relating with the collection and processing of those attributes. Therefore, it is imperative to include a feature selection process to generate the DS4 data set. We employ Genetic Algorithm and Particle Swarm Optimisation algorithms to select the most related attributes to the outcome of interest. To get an optimal solution means obtaining not only improvement accuracy but also minimising the number of attributes. Thus, we wrap the feature selection around the classification algorithms (ANN, DT, k-NN, RI & SVM) to be used, applying cross-validation to predict the advantages of adding or removing feature subsets used. We apply the feature selection system to the data sets DS1, DS2, and DS3, respectively, in order to have a full-set of reduced data sets from each of those.

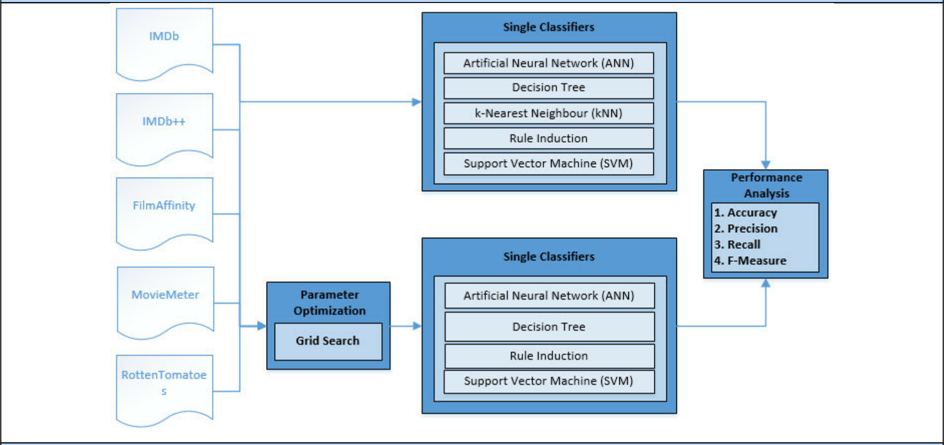
3.3 Methods

This section discusses the methods used within this research, in order to provide answers to the research questions generated in Chapter 1, Section 1.4. A method to answer the

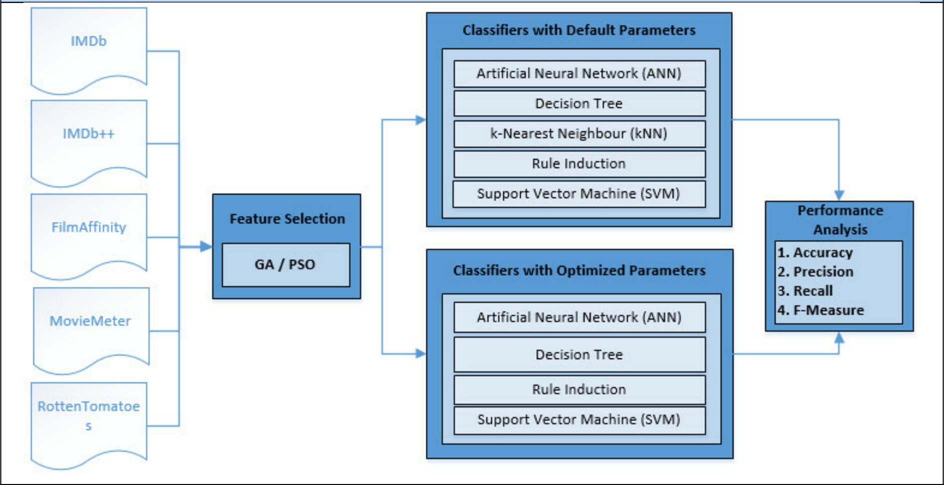
RQ1 shapes a research design as shown in Figure 3.3, while the others method to address the RQ2, RQ3, and RQ4 form the others research design as indicated in Table 3.3.

Table 3.3: Overview of the method used within this research

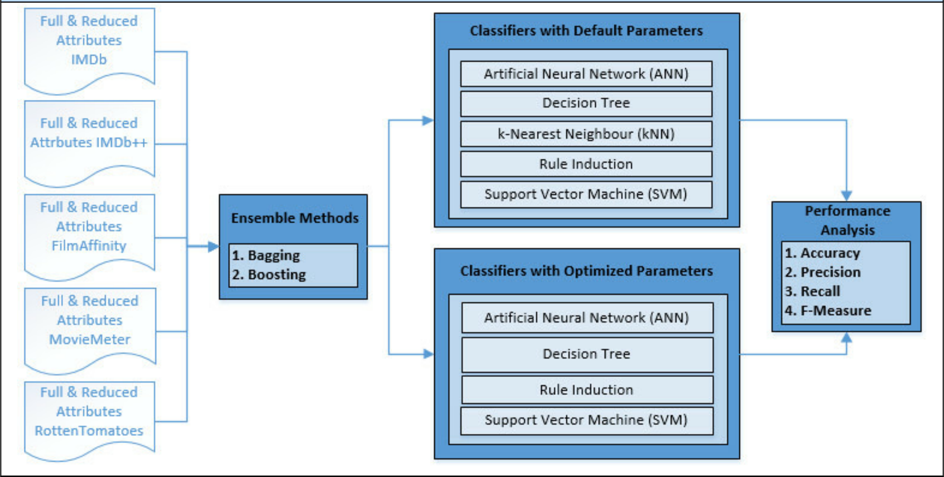
Research Question 2



Research Question 3


IMDb
IMDb++
FilmAffinity
MovieMeter
RottenTomatoes
Feature Selection
GA / PSO
Classifiers with Default Parameters
Artificial Neural Network (ANN)
Decision Tree
k-Nearest Neighbour (kNN)
Rule Induction
Support Vector Machine (SVM)
Classifiers with Optimized Parameters
Artificial Neural Network (ANN)
Decision Tree
Rule Induction
Support Vector Machine (SVM)
Performance Analysis
1. Accuracy
2. Precision
3. Recall
4. F-Measure

Research Question 4


Full & Reduced Attributes IMDb
Full & Reduced Attrbutes IMDb++
Full & Reduced Attributes FilmAffinity
Full & Reduced Attributes MovieMeter
Full & Reduced Attributes RottenTomatoes
Ensemble Methods
1. Bagging
2. Boosting
Classifiers with Default Parameters
Artificial Neural Network (ANN)
Decision Tree
k-Nearest Neighbour (kNN)
Rule Induction
Support Vector Machine (SVM)
Classifiers with Optimized Parameters
Artificial Neural Network (ANN)
Decision Tree
Rule Induction
Support Vector Machine (SVM)
Performance Analysis
1. Accuracy
2. Precision
3. Recall
4. F-Measure

3.3.1 Method to Answer RQ1

RQ1 was primarily framed to demonstrate the ability of linked data to get a data source and to explore more related data sources. Furthermore, this thesis explores the use of linked data to support data mining processes. How can linked data be used to integrate multiple data sources into an integrated database. Therefore, linked data can be useful to prepare and create readily datasets for knowledge discovery processes. To address the questions of RQ1, we design a method to collect and prepare the research data as shown in Figure 3.3.

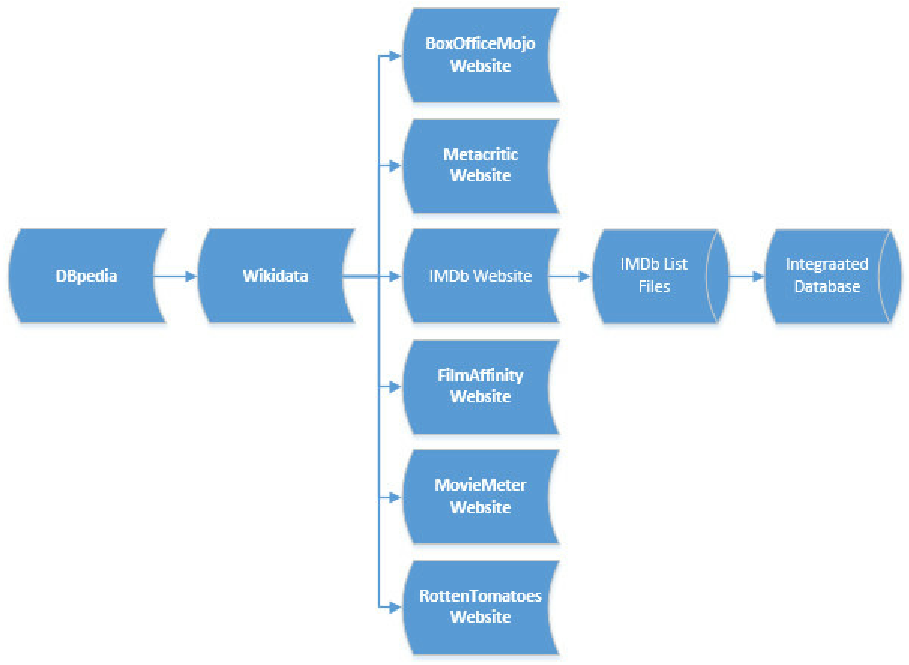


FIGURE 3.3: Overview of the method to collect and prepare the research data

3.3.2 Method to Answer RQ2

RQ2 was primarily framed to indicate that in the specific context, classifiers (ANN, DT, RI, and SVM Polynomial) with optimized parameters attain better classification results than the ones with default parameters. RQ2 also demonstrated that single classifiers perform better with multiple data sources than a single data source. To prove that claim, we design an experiment scenario as shown in the First Part of Table 3.3, consisting of two scenarios:

- Building classification models using the classifiers (ANN, DT, k-NN, RI, and SVM Polynomial) with default parameters based on the IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets.
- Building classification models using the classifiers (ANN, DT, RI, and SVM Polynomial) with optimized parameters based on the IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets.

  We apply a parameter optimization algorithm called Grid Search to get a set of optimal hyper-parameters of ANN, DT, RI, and SVM Polynomial classifiers.

To evaluate the performance of classification models, we use accuracy, precision, recall and F-Measure. Accuracy refers to the percentage of correct predictions made by the model when compared to the actual classifications in the test data, whilst F Measure indicates the harmonic average of precision and recall. According to results of the experiment above, it can be derived insights whether the use of Grid Search is able to improve the classification performance or not. As described in Section 3.2, the IMDb dataset is generated from a single data source, while the IMDb++ dataset is built by including additional attributes from external data sources into the existing IMDb dataset. By comparing the classifiaction results based on the IMDb and IMDb++ datasets, this also can give a conclusion whether classification on multiple data sources is better than a single data source or not.

3.3.3 Method to Answer RQ3

RQ3 was primarily arranged to demonstrate the approach to improve classification performances, not only classification on a single data source but also classification on multiple data sources by employing the feature selection algorithms. Also, it is aimed to convey understanding of the application of the feature selection process to pre-processing tasks to gain better classification results. We employ Genetic Algorithm and Particle Swarm Optimization to obtain optimal features, and details steps to answer RQ3 as follows:

- Applying wrapper-type feature selection suited to the ANN, DT, k-NN, RI, and SVM Polynomial classifiers with default parameters to select the most important attributes of the IMDb, IMDb++, FilmAffinity, MovieMeter and RottenTomatoes datasets.
- Applying wrapper-type feature selection suited to the ANN, DT, RI, and SVM Polynomial classifiers with optimized parameters to select the most important attributes of the IMDb, IMDb++, FilmAffinity, MovieMeter and RottenTomatoes datasets.

We use a set of optimal parameter of the ANN, DT, RI and SVM Polynomial classifiers that have been generated by applying the Grid Search algorithm.

3.3.4 Method to Answer RQ4

RQ4 was primarily intended to evaluate the approach of improving classification tasks by employing ensemble classifiers. Ensemble learners are also called meta learners, optimise classification problems by combining classification models from single classifiers to form an aggregate hypothesis or model. We use the Bagging and Boosting techniques in building an ensemble model. We also apply the ensemble technique to all types of dataset, so that, we can evaluate whether ensemble learners perform better not only on a single data source but also multiple data sources. To answer RQ4, we:

- Apply a bagging ensemble method to the base classifiers (ANN, DT, k-NN, RI, and SVM Polynomial) with default parameters using the full and reduced attributes IMDb, IMDb++, FIlmAffinity, MovieMeter, and RottenTomaotes datasets.
- Apply a bagging ensemble method to the base classifiers (ANN, DT, RI, and SVM Polynomial) with optimized parameters using the IMDb, IMDb++, FIlmAffinity, MovieMeter, and RottenTomaotes datasets with full and reduced attributes.
- Apply a boosting ensemble method to the base classifiers (ANN, DT, k-NN, RI, and SVM Polynomial) with default parameters using the IMDb, IMDb++, FIlmAffinity, MovieMeter, and RottenTomaotes datasets with full and reduced attributes.
- Apply a boosting ensemble method to the base classifiers (ANN, DT, RI, and SVM Polynomial) with optimized parameters using the full and reduced attributes IMDb, IMDb++, FIlmAffinity, MovieMeter, and RottenTomaotes datasets.

3.4 Research Ethics

This section discusses ethical issues relating to data sources on the web to use in this research as well as methods to collect those data for experimentations. The discussion focused on the terms of service of data sources and the use of a file called robots.txt to control robots or crawlers to crawl web sites. The terms of service of the use of web sites can be mentioned as the agreement of web site visitors to obey rules provided by web site owners. The owners use the terms of service as legal purposes to define user or visitor rights and responsibilities. Therefore, the web site owners usually dedicate one of their web pages containing of the terms of service. The web sites user is not only human but also machines of computer programs. Chakrabarti (2002) defines a web crawler as a computer program that automatically is able to download a web page, extract hyperlinks from that page, and crawl those URLs.

Koster (1993) introduces Robots Exclusion Protocol also known as robots.txt and describes it as the principal set of rules for how web crawlers behave ethically and politely. The robots.txt file is placed under the root directory of the web server (e.g. https://www.imdb.com/robots.txt) and is used to control the access and to regulate activities of crawlers or robots on web sites . In their paper, Pant et al. (2004) explains that the Robots Exclusion Protocol can be used to communicate file access policies of web server administrators to web crawlers which files may be accessed and those can not be accessed. According to Thelwall and Stuart (2006), by following the Koster (1996) suggestions, the robots.txt file can address two basic issues. Firstly, it was to protect site owners from web crawlers to use their server and network resources heavily. Secondly, it was to avoid the less desirable-uses by stopping web search engines from indexing undesired content. Giles et al. (2010) argued that the robots.txt file can regulate web crawler activites not only to identify which parts of web sites must not to visit but also to define a minimum time interval between visits. They also proposed, even though, robots.txt gives just an unenforced warning to crawlers, infringing it may prompt serious privacy and security concerns. Hendler and Berners-Lee (2010) argued that the a straightforward accountability mechanism to assess the ethics of web crawler can be achieved by applying the combination of robots.txt and web site logging. While robots.txt provides a mechanism of web crawlers to avoid pages not to crawl, web site logging gives a mechanism to track which web crawlers reached what pages.

The following sub sections details about the terms of service of data souces involved in this research, such as: DBpedia, Wikidata, IMDb, BoxOfficeMojo, FilmAffinity, Metacritic, MovieMeter, and RottenTomatoes. We reviewed each of terms of service of those data sources as appeared in their web sites. Also, it is discussed the robots.txt file each of the following data sources: IMDb, BoxOfficeMojo, FilmAffinity, Metacritic, and RottenTomatoes. We only considered those data sources because we employed web crawler technigue to collect the data. We attempted to behave ethics in crawling the web sites by considering very carefully each of their robots.txt. We only crawled web pages that allow to visit and avoided as much as possible web pages that can not be accessed as defined in the robots.txt file.

3.4.1 IMDb Data Source

In this research, we used two data types of IMDb. The first one is the IMDb list files (plain data text files) which can be downloaded from the IMDb interfaces site [8]. The second one is data displayed on the IMDb web site. The explanation of the terms of

[8]http://www.imdb.com/interfaces

service both the IMDb list files and the IMDb web site can be found in 3.4.1.1, while the explanation of robots.txt of IMDb web site is in 3.4.1.2.

3.4.1.1 The Terms of Service of IMDb

The terms of service of the IMDb list files are written in the header and footer of each text files and it is the same for all files. The terms of service express that the IMDb list files are freely available to anyone, however, it remains Copyright 2005 Internet Movie Database Limited. The use of IMDb list files to construct on-line database is allowed only for individual personal use. The source data in this research may not be allowed to further distribution.

The terms of service of the IMDb web site can be found in IMDb Conditions of Use [9]. It is declared that IMDb.com may not allow to use robots and crawlers, except with their written consent. They only allow the limited use of robots and crawlers for non-personal and commercial use with their express written permission.

3.4.1.2 The Robots.txt of IMDb Web Site

IMDb web server administrators provide access policies for robots or crawlers in the IMDb robots.txt [10]. Details of the robots.txt can be seen in Appendix L, Figure L.1. A User-agent value of the IMDb robots.txt is "*". It means for any robots or crawlers need to obey rules in the robots.txt. "/title/tt*/mediaviewer/rm*/tr" is the only Disallow value of the IMDb robots.txt that indicates a restriction rule for crawlers to access spesific sub directories under the title directory. Spesifically, the rule prohibited for any crawlers or robots to visit any pages or to access any files in the directory of any of its sub directories "/title/tt*/mediaviewer/rm*/tr". We paid attention to this rule because our attributes target to crawl in the IMDb web site is under the "title" directory as shown in Table 3.4. Fortunately, the attributes target to crawl is outside of the prohibit directory or sub directories. For example, to get movie information of Crocodile Dundee II, it is required to visit a web page with the following URL [11] to reach its original title , while the other [12] is the URL of web page to get information relating with awards achieved by Crocodile Dundee II the movie. Therefore, a crawler developed in this research is highly polite to to avoid pages that it is requested not to crawl in IMDb web site.

[9]https://www.imdb.com/conditions
[10]https://www.imdb.com/robots.txt
[11]https://www.imdb.com/title/tt0092493/
[12]https://www.imdb.com/title/tt0092493/awards

TABLE 3.4: The IMDb Attributes to Crawl

NO	ATTRIBUTE	DIRECTORY
1	Original Title	/title/tt*/
2	Number of GG Nominated	/title/tt*/awards
3	Number of GG Win	/title/tt*/awards
4	Number of Oscar Nominated	/title/tt*/awards
5	Number of Oscar Win	/title/tt*/awards

3.4.2 Box Office Mojo Data Source

Box Office Mojo is an online data source of information related to movies financial. Information, such as movies production budget, movies box office revenues, etc can be found in Box Office Mojo web site. However, they do not provide mechanism in accessing data, such as providing downloadable files and API for automated queries. Therefore, the most possible way to perform automatica queries is by employing web crawlers. The two issues to consider in applying web crawlers are the terms of service and a robot exclusion protocol. The two following sub sections discussed both issues.

3.4.2.1 The Terms of Service of Box Office Mojo

The terms of service of the Box Office Mojo web site can be found in Box Office Mojo Conditions of Use [13]. As Box Office Mojo owned and operated by IMDb.com, they have the same terms of service. Principally, Box Office Mojo does not allow to extract information from their web site using web crawlers or related extraction tools. It is allowed for web crawlers only for limited use and with Box Office Mojo express written permission.

3.4.2.2 The Robots.txt of Box Office Mojo Web Site

BoxOfficeMojo web server administrators provide access policies for robots or crawlers in the Box Office Mojo robots.txt [14]. Details of the robots.txt can be seen in Appendix L, Figure L.2. Box Office Mojo enforce any crawlers or robots to follow rules in the robots.txt. This is indicated by by User-agent value of the robots.txt which is "*". The first rule is any useer-agents would be instructed not to visit a page "default.movies.htm " in the movies directory. The rule number two is to instruct any crawlers or robots not to access a page "buy.php" in the showtimes directory. The rules number three until seven are to order any user-agents not to visit any page in the directories (forums, derbygame, grades, moviehangman, and users, respectively) or any of its sub directories. Table 3.5 summarizes attributes to crawl in the Box Office Mojo web site. As indicated in Table 3.5, all attributes target found in web pages are in the movies directory, while

[13] https://www.boxofficemojo.com/about/termsofuse.htm

[14] https://www.boxofficemojo.com/robots.txt

Table 3.5: The BoxOfficeMojo Attributes to Crawl

NO	ATTRIBUTE	DIRECTORY
1	Title	/movies/?id=*.htm
2	Release Date	/movies/?id=*.htm
3	Production Budget	/movies/?id=*.htm
4	Opening Gross	/movies/?id=*.htm
5	Opening Theaters	/movies/?id=*.htm

there are no restriction defined in the robots.txt to access pages in the movies directory or any of its sub directories. For example, to get information (title release date, production budget, opening gross, and opening theaters) of the Crocodile Dundee II movie, it is required to visit a web page with the following URL [15]. Therefore, it can be mentioned that we did not break the rules defined in the robots.txt in crawling information from the Box Office Mojo web site.

3.4.3 FilmAffinity Data Source

FilmAffinity is a web site that dedicates to give movie recommendations. Recommendations can be in the forms of ratings and reviewing. Registered users can not only rate but also write reviews to a selected movie. We selected this data source as we need information relating with movie ratings. To follow a code of conduct the use of FilmAffinity web site as data source in this research, we consider the terms of service and robots.txt.

3.4.3.1 The Terms of Service of FilmAffinity

FilmAffinity web site administrators displays the terms of service on Filmaffinity Privacy Policy's page [16]. The privacy policy page does not contain restictions to use web crawlers or other extraction data tools on FilmAffinity web site.

3.4.3.2 The Robots.txt of FilmAffinity Web Site

FilmAffinity web server administrators placed a series of instructions to crawlers in the FilmAffinity robots.txt [17]. Details of the robots.txt can be seen in Appendix L, Figure L.3. FilmAffinity does not ban the whole site, but only spesific areas (directories) from being crawled. First and second, the administrators instructed any crawlers not to crawl any pages in the root directory with suffixs ?FASID and &FASID. Thirdly, it is prohibited to any crawlers for accessing any pages or files in the sharerating directory.

[15] https://www.boxofficemojo.com/movies/?id=crocodiledundee2.htm
[16] https://www.filmaffinity.com/en/private.php
[17] http://www.filmaffinity.com/robots.txt

Table 3.6: The FilmAffinity Attributes to Crawl

NO	ATTRIBUTE	DIRECTORY
1	Title	/en/film*.html
2	Release Date	/en/film*.html
3	User Rating	/en/film*.html
4	Number of Votes	/en/film*.html

This directory is located under any directories in the root directory. Lastly, any crawlers would be informed not to access a file called rats.swf in the flash sub directory. According to table 3.6, all pages would be crawled are located in the en sub directory and the pages name prefixed with film. It is clear that our crawler strictly obeys a series of instructions of robots.txt.

3.4.4 Metacritic Data Source

Metacritic is another web site that allows registered users to rate their favourite movies, however, those users could not write reviews because. Only someone considered as a movie expert can write reviews for movies. Moreover, Metacritic does not only review movies but also different media products, such as: music albums, video games, tv shows, and books. As we extracted information from Metacritic web site using a web crawler, then, we discussed the terms of service and robots.txt of Metacritic.

3.4.4.1 The Terms of Service of Metacritic

The terms of service of Metacritic is on CBS Interactive Terms of Use [18]. The terms of service do not allow for unauthorized crawlers or use any other unauthorized extraction data tools on Metacritic web site.

3.4.4.2 The Robots.txt of Metacritic Web Site

Metacritic web server administrator placed a file called robots.txt [19] to control the access of crawlers on the web site, so that, user-agents (crawlers or robots) could behave polite to avoid pages that it is requested not to crawl. Details of the robots.txt can be seen in Appendix L, Figure L.4. Metacritic does not ban the whole site, but only restricts seven directories from being crawled by any user-agents. The restrictions are applied to instruct any crawlers not to visit any pages or or to access any files in the following directories (search, signup, login, user, jl, 8264, and 7336) or any of its sub directories. As shown in Table 3.7, all attributes target found in web pages are in the movie directory, while there are no restriction defined in the robots.txt to access pages in

[18] https://www.filmaffinity.com/en/private.php

[19] http://www.metacritic.com/robots.txt

Table 3.7: The Metacritic Attributes to Crawl

NO	ATTRIBUTE	DIRECTORY
1	Title	/movie/*
2	Release Date	/movie/*
3	Metascore	/movie/*
4	Number of Critics	/movie/*

the movie directory or any of its sub directories. For example, to get information (title release date, metascore, and number of critics) of the Crocodile Dundee II movie, it can be found in a web page with the following URL [20]. It is clear that we did not break the rules defined in the robots.txt in crawling information from the Metacritic web site.

3.5 The Selection of Supervised Learning Techniques

There are three considerations in selecting algorithms to build movie popularities classification models.Fistly, it is found in previous works that an algorithm used to build prediction models in the movie domain. Secondly, a classifier is belonged to the top 10 data mining algorithms as presented in the Wu et al. (2008). Thirdly, a classifier which has robustness to classify real life data on the web. As they are often imperfect, erroneous, incomplete, uncertain and vague Grzymala-Busse and Hu (2000). Moreove, heterogeneity and an uneven degree of completeness and reliability which is typically the nature of data on the web Svátek et al. (2014).

The three criterions aforementioned may be addressed by the following supervised learning techniques: classifiers (Artificial Neural Network, Decision Tree, k-Nearest Neighbour, Rule Induction, and Support Vector Machine), ensemble learners (bagging and boosting), feature selection algorithms (Genetic Algorithm and Particle Swarm Optimization), and a parameter optimisation algorithm (Grid Search).

3.5.1 Classifiers

It has been found that previous researchers used neural network (NN) in predicting movie revenues (Sharda and Delen (2006); Delen et al. (2007); Lee and Chang (2009); Zhang et al. (2009); Delen and SHARDA (2010); Ghiassi et al. (2015); Lash and Zhao (2016); Rhee and Zulkernine (2016); Quader et al. (2017b); Quader et al. (2017a)). For a slight different purpose, Neural Network were used to predict movie popularities (Hsu et al. (2014); Ahmed et al. (2015); Latif and Afzal (2016)). Marovic et al. (2011) employed NN to build a recommender system to determinie a possible user rating for a spesific movie. Although, there is no Neural Network classifier in the list of top 10 data mining algorithms. Neural Network or Artificial Neural Network is highly sophisticated

[20] http://www.metacritic.com/movie/crocodile-dundee-ii

classification algorithm, powerful to model extremely complex non-linear functions (Delen and SHARDA (2010)). According to Hastie et al. (2009), Neural Networ derives features by extracting linear combinations of the inputs and then model the output as a non-linear functions of these features. Therefore, NN is considered to use in this research, as it is popular in predicting the movie domain and has some advantages to use as a classifier.

It is noticable that decision tree is popular to use in the movie prediction domain. While decision tree is popular in predicting movies revenue performance (Sharda and Delen (2006); Delen et al. (2007); Lee and Chang (2009); Delen and SHARDA (2010); Apala et al. (2013); Lash and Zhao (2016); Quader et al. (2017b)), it is also a powerful algorithm to predict the future success of movies rating (Shijia et al. (2010); Asad et al. (2012); Kabinsingha et al. (2012); Ahmed et al. (2015); Latif and Afzal (2016)). Li and Yamada (2004) used decision tree to build a movie recommender system. In terms of the algorithm popularity, Wu et al. (2008) listed decision tree in the top 10 data mining algorithms. Decision tree is a popular classification algorithm because it has some advantages. Firstly, decision tree is robust to missing data, so that, it does not require data imputation Bogard et al. (2010). Secondly, it depicts a more transparent model structure and explicitly explains the logical process related with outcomes Delen (2010). Thirdly, the decision tree models are non-parametric and possible to capture non-linear relationships and complex interactions between independent attributes and a dependent attribute Kovačić (2010).

k-Nearest Neighbor (k-NN) is also a popular algorithm to use in the movie prediction domain. k-NN have been used by (Zhang and Skiena (2009); Kim et al. (2015)) in predicting movie gross, while Marovic et al. (2011) employed k-NN to build a recommender system to determinie a possible user rating for a spesific movie. Moreover, Pampalk et al. (2005) have used k-NN in classifying music genres. Zhou et al. (2010) employed k-NN to classify movie genres based on scene classification of movie trailers. In fact, k-NN is belonged to the top 10 data mining algorithms according to the survey of Wu et al. (2008). k-NN becomes popular as it offers some advantages. Firstly, the process is transparent, so that, it is easy to understand and implement (Cunningham and Delany (2007); Bhatia and Vandana (2010)). Secondly, k-NN has robustness to noisy training data as well as the effectiveness of large training data (Jiang et al. (2007); Bhatia and Vandana (2010)). Thirdly, a classifier that can work well in many situations (Wu et al. (2008)). Lastly, k-NN is a classifier that can classify well data sets with multi class labels (Wu et al. (2008)).

Rule Induction may not has been found in the movie domain prediction. Also, it did not been identied through the study Top 10 algorithms in data mining (Wu et al. (2008)). However, there were some considerations to use Rule Induction to classify our research data sets. Firstly, it is able to explain an inherent relationship between input attributes and an output attribute in the data set in the form of simple if-then rules. This allows general users to understand the rules easily. Secondly, it benefits not only as a predictive model but also a descriptive model. Thirdly, it has a remarkable property to induce decision rules that have high predictablilty or reliability from a dataset as small as possible at the same time (Kotsiantis (2007)). Lastly, Rule Induction outperforms decision tree classifier on many problems (Cohen (1995)).

Support Vector Machine (SVM) have been used in predicting movie box-office success by the following researchers (Delen and SHARDA (2010); Lash and Zhao (2016); Rhee and Zulkernine (2016); Quader et al. (2017b); Quader et al. (2017a)). Ahmed et al. (2015) combined sentiment analysis and SVM in performing multi-class classification to predict movies popularity, while Tsutsumi et al. (2007) and Liu et al. (2012) used sentiment analysis and SVM in performing binary classification to predict movie ratings. SVM is one of the most frequently used algorithms for supervised learning as identified through the study Top 10 algorithms in data mining (Wu et al. (2008)). According to Delen and SHARDA (2010), SVM may not suffer form multiple local minima, its solution is global and unique. Moreover, SVM has a simple geometric interpretation and provides a sparse solution. Liu et al. (2012) argued that one remarkable property of SVM is that its ability to learn well even with a large number of attributes and less training data. This is also confirmed by Yu and Kim (2012) where two special properties of SVM are that they achieve high generalization by maximizing the margin, and support an efficient learning of nonlinear functions using kernels.

Ensemble classifiers combine multiple classifiers to improve the error rate and overcome the modeling bias of a single classifier. A range of ensemble classifiers, such as Random Forest, Bootstrap Aggregation (Bagging), LogitBoost, and Adaptive Boosting (AdaBoost) have been found to use in prediction in the movie domain. Random Forest is is a type of ensemble classification algorithm called bagging. LogitBoost is a variant of AdaBoost which is a type of ensemble classifiers called boosting. Lash and Zhao (2016) employed Random Forest and LogitBoost to predict movie success of profitability, while Random Forest and AdaBoost were applied in predicting movie box office success (Delen and SHARDA (2010); Quader et al. (2017a)). We considered applying bagging and boosting in the modeling of classification with some considerations. Bagging is considered as it is able to improve the stability of single classifiers (Graczyk et al. (2010)). The single classifiers such as decision tree and neural network can be unstable when the training data changes even a slight change. On the other hand, there were two considerations to use boosting, spesifically, AdaBoost. Firstly, it is a general method for improving the accuracy of a weak single classifier Das (2001). Secondly, it has been identified through the study top 10 data mining algorithms (Wu et al. (2008)).

3.5.2 Feature Selection

Feature selection is a part of the data pre-processing step, therefore, it did not been identied through the study Top 10 algorithms in data mining by Wu (Wu et al. (2008)). The aim of feature selection is to discover the most discriminate features to the target attribute and help reduce the attribute of the data (Bell and Wang (2000)). This improves a classification algorithm performance by minimalizing the effects of the curse of dimensionality and by removing noise due to irrelevant and redundant features. There are three basic techniques to evaluate features, such as: filter, wrapper, and embedded approach. We selected the wrapper method to apply in this research, as it usually perform better because it is optimised for the particular classification algorithms (Hall and Holmes (2003)). There are various algorithms for applying the wrapper method to obtain a set of optimal attributes. In this research, we considered Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) with two reasons. The first reason is their involment in predicting the movie domain. The second one is literatures that explain their ability to yield an optimal model for improving prediction results. In the practical of movie domain, Kim et al. (2015) used a Genetic Algorithm (GA) based attribute selection (Cho and Hermsmeier (2002); Jarvis and Goodacre (2005)) to obtain the optimal set of input attributes. In terms of the powerful of feature selection algorithms, Malhotra et al. (2011) argued that GA improves the performance of standar feature selection algorithms which have difficulties to face discontinlus, non-differentiable, stochastic, or highly non-linear data. On the other hand, PSO is an evolutionary computation technique that is easy to implement, powerful as well as effective and efficient to address the search space of a problem (Huang and Dun (2008); Schuh et al. (2012)).

3.5.3 Parameter Optimization

Parameter optimization is a technique to optimize the performance of existing classifiers, so that, it was not involved through the study top 10 algorithms in supervised learning. Most of supervised learners may not perform well if they do not have a correct setup in terms of their configuration paramters. However, the parameter optimization is not a trivial task. Indeed, it is a very time consuming task if it is done manually especially when the classifer has many parameters (Rossi et al. (2008)). In this research, we optimized the parameters of Decision Tree and SVM classifiers, as their performances can vary widely if the parameter settings changes even on the same dataset. The two previous works (Camilleri and Neri (2014); Neri and Camilleri (2014)) showed the results of a decision tree learner when its parameters change, while different researchers (Chang and Lin (2011); Hric et al. (2011); Subasi (2013)) confirmed the effects of changing parameter settings of a SVM learner. We then considered Grid Search as an algorithm to find a set of optimal parameters for Decision Tree and SVM learners. The choice of Grid Search as the optimizer is based on a literature that it is one of the most widely used strategies for hyper-parameter optimization (Bergstra and Bengio (2012)). A previous work (Larochelle et al. (2007)) used Grid Search to optimise the two parameters of SVM (C and γ), while another previous researcher (Camilleri and Neri (2014)) employed Grid Search to find a set of optimal decision tree parameters.

As noted by Hsu et al. (2003), the parameter C represents the error penalty that controls the margin of misclassification allowed within the prediction model, while the γ parameter is used to tune the the mapping between input space and feature space. If the value of C is too large will result in the hyperplane being tuned to classify every training example correctly, thus risking overfitting the data causing low generalizability. If the value of C is too small will result in underfitting causing an error prone prediction model. For the γ parameter, if its value is too large will result in over-fitting, while if it is too small will lead to under-fitting (Pardo and Sberveglieri (2005)).

3.6 Summary

This chapter presents the datasets and experimentation methodology used to address the research questions. Additionally, this chapter presents research ethics as well as the considerations to select supervised learning techniques used in this research.

The following four chapters describe data preparations an the outcome of the experimentations described in this chapter. Chapter four presents data collection and preparation. Chapter five presents the first experimentation, which explored movie popularity classification models built with single classifiers. Chapter six presents the second experimentation, which explored movie popularity classification models built with features selection. Chapter seven presents the third experimentation, which explored movie popularity classification models built with ensemble classifiers.

Chapter 4

Data Collection and Preparation

This chapter details the three initial steps of the Knowledge Discovery Process (KDD): data selection, data pre-processing, and data transformation (Fayyad et al. (1996)). These important parts provide datasets ready for data mining processes. This chapter covers how to generate all types of dataset as described in Table 3.2.

4.1 Introduction

Firstly, we determined all related data sources used in this thesis. Subsequently, we decided what preparations are needed to prepare a dataset for the data mining processes. When the datasets are ready, we can employ classification algorithms to build classification models. This also enables us to evaluate whether classification on multiple heterogeneous data sources gives better results rather than classification with a single data source.

Notably, data selection and preparation are crucial steps in this thesis in order to answer the research questions, defined in Chapter 1. The organisation of this chapter is detailed as follows: Section 2 covers data selection, Section 3 details accessing the data, and Section 4 discusses data preparation. Section 2 describes what data sources are involved and how they can be gathered. Section 4 explains what attributes are involved to build classification models, and how they are prepared.

4.2 Data Selection

To deal with multiple data sources distributed on the web, we constructed an integrated database, in order to unlock hidden knowledge. In this thesis, we focus on data available from the web, the main data source here to provide movie-related data being IMDb

TABLE 4.1: Information to Extract from Websites

Websites	**Extracted Information**
IMDb	Display Title, Original Title, AKA Titles, Release Date, Directors, Writers, The number of Golden Globe (GG) nominated, The number of GG win, The number of oscar nominee, The number of oscar win
BoxOfficeMojo	Title, Release Date, Production Budget, Opening Gross and Opening Theaters
FilmAffinity	Title, Release Date, User Rating and The number of votes
Metacritic	Title, Release Date, Metascore, and The number of critics
MovieMeter	Title, Release Date, User Rating and The number of votes
RottenTomatoes	Title, Release Date, User Rating and The number of votes

[1]. The IMDb website provides a collection of downloadable list-files, and it can be installed in a local computer. The other resources are BoxOfficeMojo[2], FilmAffinity[3], Metacritic[4], MovieMeter[5], and RottenTomatoes[6] – details of information extracted from those resources are listed in Table 4.1. The schema used to integrate multiple data sources into an integrated database can be shown in Figure 4.1. It indicates that there are two data sources related IMDb. The first one is IMDb list files and the other one is IMDb website. It is considered to use IMDb list files because they provide information such as actor, actress, cinematographer, composer, costume designer, director, distributor, editor, production company, production designer, producer, writer, movie rating, etc that can be installed to a local database. Those information are also available in the IMDb website. As the IMDb website does not provide aided tools to access its data, therefore, the possible way to access it is by using web scraping techniques. We avoid as much as possible to use web scraping techniques because it is possibly to distruct the IMDb website traffic. However, we can not avoid to use the web scraping to get data such as Golden Globe Nominee, Golded Globe Win, Oscar Nominee, and Oscar Win. Furthermore, those data are not available in IMDb list files. This makes our consideration to include both IMDb list files and IMDb website as the data sources in this thesis.

4.2.1 IMDb List Files Extraction and Transformation

IMDb is an excellent resource to provide free, up-to-date movie information online. It is claimed to be the earth's biggest movie database. The explanations of how to access the IMDb data set can be found in the IMDb alternate interface site [7]. The original IMDb data set is arguably not suitable for data mining without the additional use of natural language processing techniques for extracting the data (Peralta (2007a)). Before

[1]http://www.imdb.com/
[2]http://www.boxofficemojo.com/
[3]https://www.filmaffinity.com/
[4]http://www.metacritic.com/
[5]https://www.moviemeter.nl/
[6]https://www.rottentomatoes.com/
[7]http://www.imdb.com/interfaces

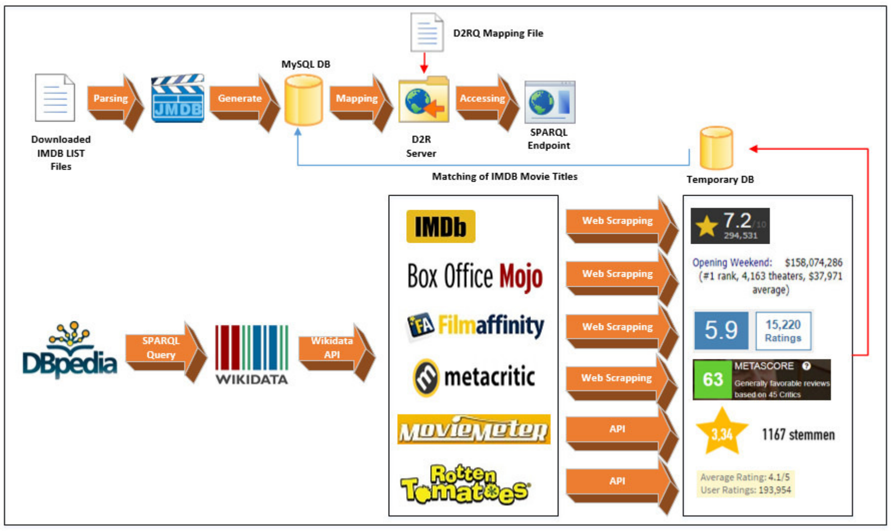


FIGURE 4.1: The Schema to Generate an Integrated Movie Database

the local installation of data in a database, we downloaded all 49 text files in an ad-hoc format (called lists) from [8]. We utilised a free Java-based application, namely JMDB [9], which has capabilities to parse all the IMDB flat text files and convert each one to a table in a database. JMDB automatically imported the IMDb text files (also called IMDb list files) into a MySQL database system and created a database with a total 48 tables, each having 'movie id' as the primary key and linked with a table, namely 'movies'.

4.2.2 DBpedia Data Extraction

As our study case is concerned in the movie domain, we extracted all film data from DBpedia on 29th April 2017. First of all, we needed to count the total number of DBPedia movies. DBpedia represents movie data as instances of classes in the Film ontology. Therefore, we wrote the SPARQL query to count the total number of movies as shown in Figure 4.2. The query returned 106369 films.

The DBpedia server is limited, as it cannot return a large number of results in one call; it is required to perform multiple queries to list all DBpedia movies. Therefore, we employed the pair of limit and offset modifiers. While the limit modifier restricts the number of solutions, the offset controls where the solutions start from in the overall sequence of solutions (Prud'hommeaux and Seaborne (2008)). We determined the results consisting of 10,000 rows. For determining the number of iterations, we divided the total, 106,369 films, by the 10,000 row limit. Thus, we performed 11 queries to DBpedia to extract all movie URI's and titles. Subsequently, the offset value is obtained by multiplying the step of iteration with the limit. The values of limit and offset are used

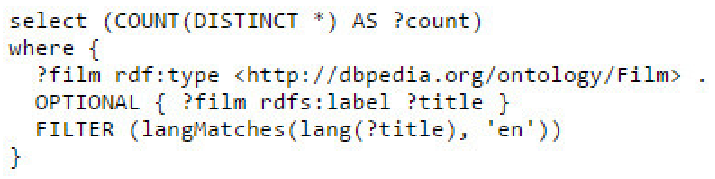
[8]ftp://ftp.fu-berlin.de/
[9]http://www.jmdb.de/

TABLE 4.2: The General Scheme of Film Data Retrieval and Integration from DBPedia

```
BEGIN
1:    NumberOfFilm = SPARQL to count all DBPedia film data
2:    NumberOfLoops = NumberOfFilm / 10000
3:    for step = 0 to NumberOfLoops
4:        Offset = step * 10000
5:        results = SPARQL to list film URI's & film title's (limit=10000, offset=Offset)
6:        while results have records do
7:            dbpedia = a film URI from the resultant SPARQL query
8:            title = a film title from the resultant SPARQL query
9:            wikidata = SPARQL to obtain a wikidata link (film URI)
10:           imdb = acquiring a IMDb link by using Wikidata API (wikidata)
11:           bom = acquiring a BoxOfficeMojo link by using Wikidata API (wikidata)
12:           fa = acquiring a FilmAffinity link by using Wikidata API (wikidata)
13:           mc = acquiring a Metacritic link by using Wikidata API (wikidata)
14:           mm = acquiring a MovieMeter link by using Wikidata API (wikidata)
15:           rt = acquiring a RottenTomatoes link by using Wikidata API (wikidata)
16:           Insert dbpedia, title, wikidata, imdb, bom, fa, mc, mm and rt
                          into the movie_links_290417_step1 table
17:       endwhile
18:   endfor
END
```

as parameters for the SPARQL query in the fifth step of the algorithm in Table 4.2. The complete SPARQL query to extract movie URI's and movie titles from DBpedia is shown in Figure 4.3.

Every DBpedia entity has a property of owl:sameAS [10] holding links to the corresponding external web resources. We used every movie URI as the result of the query in Figure 4.3 as a query parameter to extract the corresponding Wikidata page. Also, we filtered out the literal values only consisting of *wikidata.org/entity*. Therefore, the complete query to extract the corresponding Wikidata page can be formulated as seen in Figure 4.4. Every Wikidata page has the possibility of containing links to IMDB, BoxOfficeMojo, FilmAffinity, Metacritic, MovieMeter, and RottenTomatoes pages for the corresponding movie in the section named "Identifiers". The details of the process of extracting Wikidata are discussed below, in Sub Section 4.2.3.



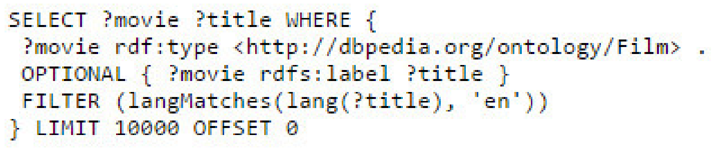
```
select (COUNT(DISTINCT *) AS ?count)
where {
  ?film rdf:type <http://dbpedia.org/ontology/Film> .
  OPTIONAL { ?film rdfs:label ?title }
  FILTER (langMatches(lang(?title), 'en'))
}
```

FIGURE 4.2: SPARQL Query to count all movies from DBpedia

4.2.3 Extraction of Wikidata Data

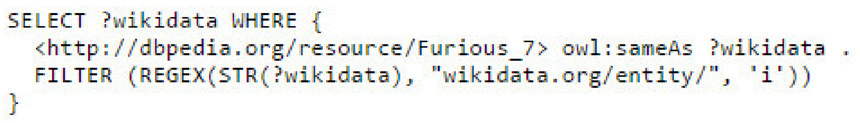
Wikidata's data is not only very rich and complex, but also highly interlinked and connected to many other data sets. This fact advocated us to combine it with the

[10] http://www.w3.org/2002/07/owl#sameAs



```
SELECT ?movie ?title WHERE {
 ?movie rdf:type <http://dbpedia.org/ontology/Film> .
 OPTIONAL { ?movie rdfs:label ?title }
 FILTER (langMatches(lang(?title), 'en'))
} LIMIT 10000 OFFSET 0
```

FIGURE 4.3: SPARQL query for extracting film URI's and film titles from DBpedia



```
SELECT ?wikidata WHERE {
  <http://dbpedia.org/resource/Furious_7> owl:sameAs ?wikidata .
  FILTER (REGEX(STR(?wikidata), "wikidata.org/entity/", 'i'))
}
```

FIGURE 4.4: SPARQL query for extracting the corresponding the Wikidata link

DBpedia data to obtain useful, relevant links to conduct knowledge discoveries on the web. In this research, we use the Wikidata toolkit [16] to read live data from the Wikidata website without having to download a dump file beforehand. The resulting wikidata link obtained by the SPARQL Query 4.4 allowed us to get a unique identifier of the Wikidata data set, by parsing the link and removing useless characters. For example, The Dbpedia gives the link "https://www.wikidata.org/wiki/Q14650496". To work with the Wikidata Toolkit, we must obtain the identifier from it. The unique identifier of this is identified by Q14650496.

Once the identifier is obtained, we can use it to explore the specified wikidata page programmatically by using the Wikidata Toolkit. The richest part of the Wikidata page is the "Statements" element. There are two common types of statement part, *Statements* and *Identifiers*. We focused on the Identifiers part, comprising of properties informing useful links. Based on the identifier Q14650496, we could potentially receive five properties from the Identifiers part that are may inform related data sources. The properties are IMDb id, BoxOfficeMojo ID, FilmAffinity ID, Metacritic ID, MovieMeter Movie ID, and Rotten Tomatoes ID. To this end, we successfully found the six data sources being used in this research. For identification purposes and to remain in-line with automatic data extracting using the Wikidata Toolkit, we use an ID property instead of a label property. For the properties aforementioned, they have the following ids: P345, P1237, P480, P1712, P1970, and P1258, respectively.

To link and obtain data from DBpedia and Wikidata, we implemented the data extraction scheme in Table 4.2 using the Java programming language. The Java code uses the Jena library [11] to access the DBPedia SPARQL Endpoint [12] and to work with the Linked Data. To work with Wikidata, we leveraged the Wikidata Toolkit [13]. The statistics of all the data extracted are given in Table 4.3., with the processing times to extract the data being 58:45:20.850s.

[11]https://jena.apache.org/
[12]http://dbpedia.org/sparql
[13]https://www.mediawiki.org/wiki/Wikidata_Toolkit

TABLE 4.3: Statistic Results of DBpedia and Wikidata Extraction

Data Sources	No. of Records Generated
DBpedia	Records generated for 106.363 movies
Wikidata	Records generated for 106.244 movies
IMDb	Records generated for 88.048 movies
FilmAffinity	Records generated for 23.701 movies
MovieMeter	Records generated for 46.240 movies
RottenTomatoes	Records generated for 35.209 movies

TABLE 4.4: The General Scheme of to Extract Information from the IMDB Website Using Web Scraping

```
BEGIN
1:  SQL to list DBpedia, and IMDb URI's where IMDB URI's exist
    \\returns 88048 records
2:  while results have records do
3:      if IMDb URI exits then
4:          Extracting display title, original title, release date, directors, and writers
            by using the Web Scrapping (IMDb URI)
5:          Extracting AKA titles by using the Web Scrapping (IMDb URI + "\releaseinfo")
6:          Extracting the number of GG nominated, GG win, Oscar nominee, & Oscar win
            by using the Web Scrapping (IMDb URI + "\awards")
7:          Updating display title, original title, aka titles, release date, directors, writers,
            the number of GG nominated, GG win, Oscar nominee, & Oscar win
            into the previous results of the process in Table 4.2
8:      endif
9:  endwhile
END
```

4.2.4 Data Extraction from the IMDb Website

This section explains how to extract the values described in Table 4.1 from the IMDb website. As the IMDb administrator does not provide a Web API, the possible solution for this is to apply a web scraping technique. The scenario to scrap the information from the IMDb website is based on an IMDb URI parameter gathered from processes in sub-section 4.2.3. Scheme 4.4 outlines the data extraction process by using a web scraping technique.

As the IMDb URI was already known, the web scraper was able to immediately extract the specified values (Display Title, Original Title, Release Date, Directors and Writers) from the web page. To get AKA titles, we required to add "\releaseinfo" to suffix of the IMDb URI. The other values (the number of Golden Globe (GG) nominated, the number of GG win, the number of oscar nominee, and the number of oscar win) are located in a different web page. We needed to add "\awards" to suffix of the IMDb URI to get information related to the respective movie awards.

Table 4.5: The General Scheme of to Extract Information from the BoxOfficeMojo Website Using Web Scraping

```
BEGIN
1:  SQL to list DBpedia, and BOM URI's where BOM URI's exist
2:  while results have records do
3:      if BOM URI exits then
4:          Extracting Title, Release Date, Production Budget, Opening Gross
            and Opening Theaters by using the Web Scrapping (BOM URI)
5:          Updating Title, Release Date, Production Budget, Opening Gross and Opening
            Theaters, Oscar nominee, & Oscar win into the previous results
            of the process in Table 4.2
6:      endif
7:  endwhile
END
```

Table 4.6: The General Scheme of Linking IMDB and FilmAffinity Data Using Web Scraping

```
BEGIN
1:  SQL to list DBpedia, IMDB TITLE's and FilmAffinity URI's where IMDB URI's exist
    \\returns 88048 records
2:  while results have records do
3:      if FilmAffinity URI exits then
4:          Pulling year, title, votes and notes by using the Web Scrapping (FilmAffinity URI)
5:          Updating year, title, vote and note into the previous results
            of the process 4.2
6:      endif
7:  endwhile
END
```

4.2.5 Data Extraction from the BoxOfficeMojo Website

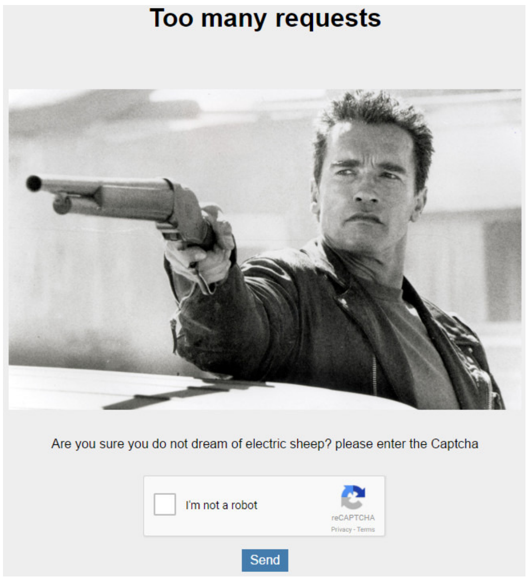
As there was a limitation to automatically extract data from the BoxOfficeMojo, we applied the same techniques as the data extraction of the IMDb website, using a web scraping technique. As we already have a BoxOfficeMojo URI, the web scraper can immediately extract the following information from the web page: Title, Release Date, Production Budget, Opening Gross and Opening Theaters. The Scheme used to extract the BoxOfficeMojo according to the specific URI is outlined in Schema 4.5.

4.2.6 Data Extraction from the FilmAffinity Website

This section illustrates the process to pull out useful values (URI, title, year, rating, and vote) from the FilmAffinity website. The extraction process was based on the gathered values (FA URI, IMDb title and IMDb year) from process 4.2., and was conducted by applying a web scraping technique, as the FA administrator does not provide a Web API for accessing the data. If the FA URI was available, then, the web scraper directly went to dig the goal values (FA title, year, rating, and vote) from the FA website. To clarify, the scheme of extraction process can be shown in figure 4.6.

reCAPTCHA is a new version of CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart). Acquired by Google in September 2009, reCAPTCHA was originally developed to give the World Wide Web users a challenge-response test to determine whether a user is a human or a computer Von Ahn et al. (2008). It was . The FA web administrators apply reCAPTCHA [14] to protect their website from spam and abuse by preventing users sending too many requests in a given amount of time (”rate limiting”). The existence of reCAPTCHA can insist a user to attest whether they are human or robot. This occurs if a user has exceeded his account’s throttle limit, or he has sent requests at a rate above the limit. The FA web administrator blocks his IP, and reCAPTCHA will pop up a form dialog as shown in Figure 4.5. In the form dialog, users are asked to just click on a checkbox and to press the send button. After that, a form appears (Figure 4.6) consisting of a collection of images, and ask users to make distinctions that might be tough for bots to decipher.

Obviously, standard web scraping will not work to extract data from the FA website. We have to calculate how long to wait before making a new request. Generally, there is no waiting time from one request to another in standard web scrapping. However, to deal with the reCAPTCHA, we must make a new request every five seconds; this approach succeeded in avoiding screening from reCAPTCHA. As a result, the process of web scraping can run continuously without hassle.


Too many requests

Are you sure you do not dream of electric sheep? please enter the Captcha

FIGURE 4.5: The Spam and Abuse protection of FilmAffinity

[14]https://www.google.com/recaptcha/

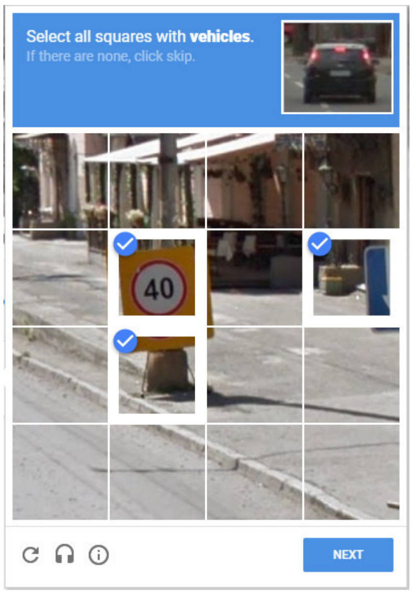

Select all squares with **vehicles**.
If there are none, click skip.
40
NEXT

FIGURE 4.6: Confirmation that They are not robot

TABLE 4.7: The General Scheme of to Extract Information from the Metacritic Website Using Web Scrapping

```
BEGIN
1:  SQL to list DBpedia, and MC URI's where MC URI's exist
2:  while results have records do
3:      if MC URI exits then
4:          Extracting Title, Release Date, Metascore, and The number of critics
            by using the Web Scrapping (MC URI)
5:          Updating Title, Release Date, Metascore, and The number of critics
            into the previous results
            of the process in Table 4.2
6:      endif
7:  endwhile
END
```

4.2.7 Data Extraction from the Metacritic Website

Similarly to the IMDb and BoxOfficeMojo websites, a Metacritic administrator does not provide friendly accessing tools for users to use information from their website automatically. Therefore, we employ a web scrapping technique to extract information from the respective movie page based on a Metacritic URI. The web scrapper can immediately go to the web page to extract the following information: Title, Release Date, Metascore, and The number of critics. The Scheme used to extract the Metacritic according to the specific URI is outlined in Table 4.7.

TABLE 4.8: The General Scheme of Linking IMDB and MovieMeter (MM) Data Using The MM API

```
BEGIN
1:   results = SQL to list DBpedia, IMDB URI's and MM URI's where IMDB URI's exist
     \\returns 88048 records
2:   while results have records do
3:      IMDb ID = Parsing the IMDB URI
4:      if MM URI exits then
5:         MM ID = Parsing the MM URI
6:         Pulling url, year, title, votes_count and average by using the MM API (MM ID)
7:      else
8:         Pulling url, year, title, votes_count and average by using the MM API (IMDb ID)
9:      endif
10:     Updating url, year, title, votes_count and average
        into the previous results of the process 4.2
11:  endwhile
END
```

4.2.8 Data Extraction from the MovieMeter Website Using the MM API

In this section we outline the process of linking IMDb and MovieMeter by utilising the MM API. The algorithm of the process of data linking can be seen in Table 4.8. The MM administrator provides an official Web API to allow MM clients retrieve information from the MM website. To work with the API, the clients mandatorily register their applications to get a valid API key. There are three methods to retrieve film information in the MM website using the API. Firstly, the searching method uses an MM id. The second method is to supply an IMDb id. The last one is used to search film information based on a movie title. The MM API endpoint is [15].

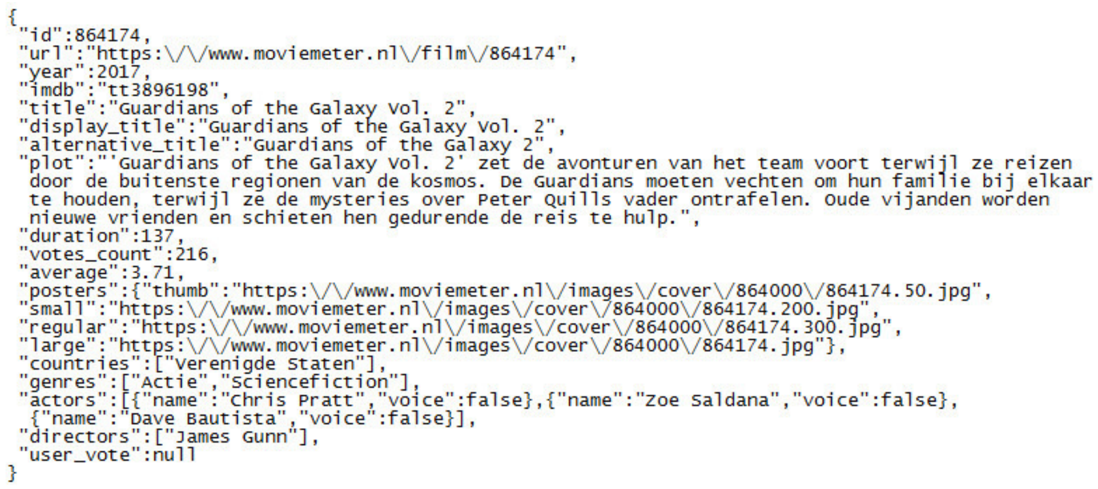
According to the results of process 4.2, 88,048 IMDB URIs and 46,240 MM URIs are generated. Out of 46,240, there are 7 MM URIs which do not contain IMDB URIs. For an IMDb URI which already has a pair of MM URIs we leveraged this to obtain MM detail attributes such as: URI, title, year, votes_count and average; otherwise we performed a searching method to obtain an MM URI based on the IMDb URI. To perform movie searching based on an id, the MM API requires two mandatory parameters to supply. The first parameter is an ID of a film. The second one is an API key. Given a specific IMDb URI (e.g. tt3896198), we can use this to pull a relevant MM URI by passing the following URI [16] to the MM web server. If we want to perform the film searching based on an MM ID, the complete URI will be [17]. The MM server returns the search results in the format of JSON data – the JSON format generated by the MovieMeter API is shown in Figure 4.7. Our target values from the JSON data are year, title, votes_count and average. MovieMeter and RottenTomatoes apply the 1-to-5 rating scale, therefore, we multiplied the average values by 2 to convert them to

[15] `http://www.moviemeter.nl/api/film/`
[16] `http://www.moviemeter.nl/api/film/tt3896198?api_key=1255b5a948a9f4d7125989742c2c3ba9`
[17] `http://www.moviemeter.nl/api/film/864174?api_key=1255b5a948a9f4d7125989742c2c3ba9`

Table 4.9: The General Scheme of Linking IMDB and RottenTomatoes (RT) Data Using the OMDb API

```
BEGIN
1:  results = SQL to list DBpedia and IMDB URI's where IMDB URI's exist
    \\returns 88048 records
2:  while results have records do
3:      IMDB ID = Parsing the IMDB URI
4:      Pulling IMDB titles, years and RottenTomatoes (RT) URI's
        by using the OMDb API (IMDB ID)
5:      Extracting RT titles, year, average ratings and user ratings
        by using the Web Scrapping (RT URI)
6:      Updating IMDB title, RT titles, years, average ratings and user ratings
        into the previous results of the process 4.2
7:  endwhile
END
```

the 1-to-10 rating scale. The results of process 4.8 added attributes such as: MM URIs, MM titles, MM years, MM votes_counts and MM averages .



```
{
 "id":864174,
 "url":"https:\/\/www.moviemeter.nl\/film\/864174",
 "year":2017,
 "imdb":"tt3896198",
 "title":"Guardians of the Galaxy Vol. 2",
 "display_title":"Guardians of the Galaxy Vol. 2",
 "alternative_title":"Guardians of the Galaxy 2",
 "plot":"'Guardians of the Galaxy Vol. 2' zet de avonturen van het team voort terwijl ze reizen
  door de buitenste regionen van de kosmos. De Guardians moeten vechten om hun familie bij elkaar
  te houden, terwijl ze de mysteries over Peter Quills vader ontrafelen. Oude vijanden worden
  nieuwe vrienden en schieten hen gedurende de reis te hulp.",
 "duration":137,
 "votes_count":216,
 "average":3.71,
 "posters":{"thumb":"https:\/\/www.moviemeter.nl\/images\/cover\/864000\/864174.50.jpg",
 "small":"https:\/\/www.moviemeter.nl\/images\/cover\/864000\/864174.200.jpg",
 "regular":"https:\/\/www.moviemeter.nl\/images\/cover\/864000\/864174.300.jpg",
 "large":"https:\/\/www.moviemeter.nl\/images\/cover\/864000\/864174.jpg"},
 "countries":["Verenigde Staten"],
 "genres":["Actie","Sciencefiction"],
 "actors":[{"name":"Chris Pratt","voice":false},{"name":"Zoe Saldana","voice":false},
  {"name":"Dave Bautista","voice":false}],
 "directors":["James Gunn"],
 "user_vote":null
}
```

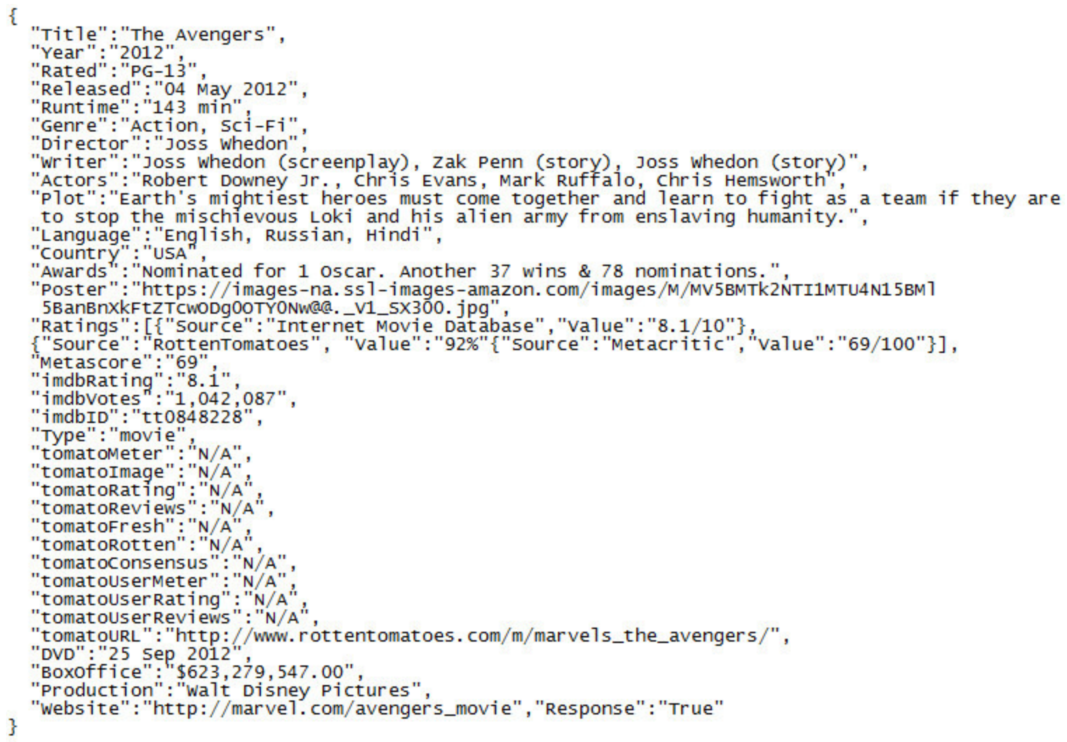
Figure 4.7: The JSON format generated by the MovieMeter API

4.2.9 Data Extraction from the RottenTomatoes Website Using the OMDb API

In this section, we describe a strategy to link IMDB and RottenTomatoes data by leveraging the OMDb API [18], a free web service to obtain movie information. The results of process 4.2 as shown in Table 4.3 generated 88,048 rows of IMDB URIs and 35,209 rows of RT URIs. Out of 35209, 21 RT URIs which do not contain IMDB URIs. Every single row of the IMDB URI is used as a parameter to use in the OMDb API to retrieve a IMDB title, IMDB year and RottenTomatoes URI. The algorithm of the process of data retrieval in this section can be seen in Table 4.9.

[18] http://www.omdbapi.com/

Firstly, the URL that we are going to pull the data from is needed, known as the API endpoint. The endpoint is http://www.omdbapi.com. To return the values of title, year and RT URI, the OMDb API requires the four following parameters: (1) *i*: a valid IMDb ID (e.g. tt0848228); (2) *type*: type of result to return (movie, series, episode); (3) *tomatoes*: whether result returns a RT URI or not (true, false); and (4) *r*: the data type to return (json, xml). The resultant IMDB URI's are in the full URI format (e.g. http://www.imdb.com/title/tt0848228/), whereas the API only needs the id (e.g. tt0848228). Therefore, it is urgent to parse it to meet the requirements. The following URL [19] is a valid format to pull the data via the OMDb API and the result is sent to clients as JSON. JSON stands for JavaScript Object Notion which is considered as a simple, human-readable, easily parsable data interchange format. Thus, many web APIs use it. The example of the resultant JSON of the OMDb API is shown in Figure 4.8.



```
{
  "Title":"The Avengers",
  "Year":"2012",
  "Rated":"PG-13",
  "Released":"04 May 2012",
  "Runtime":"143 min",
  "Genre":"Action, Sci-Fi",
  "Director":"Joss Whedon",
  "Writer":"Joss Whedon (screenplay), Zak Penn (story), Joss Whedon (story)",
  "Actors":"Robert Downey Jr., Chris Evans, Mark Ruffalo, Chris Hemsworth",
  "Plot":"Earth's mightiest heroes must come together and learn to fight as a team if they are
   to stop the mischievous Loki and his alien army from enslaving humanity.",
  "Language":"English, Russian, Hindi",
  "Country":"USA",
  "Awards":"Nominated for 1 Oscar. Another 37 wins & 78 nominations.",
  "Poster":"https://images-na.ssl-images-amazon.com/images/M/MV5BMTk2NTI1MTU4N15BMl
   5BanBnXkFtZTcwODg0OTY0Nw@@._V1_SX300.jpg",
  "Ratings":[{"Source":"Internet Movie Database","Value":"8.1/10"},
  {"Source":"RottenTomatoes", "Value":"92%"{"Source":"Metacritic","Value":"69/100"}],
  "Metascore":"69",
  "imdbRating":"8.1",
  "imdbVotes":"1,042,087",
  "imdbID":"tt0848228",
  "Type":"movie",
  "tomatoMeter":"N/A",
  "tomatoImage":"N/A",
  "tomatoRating":"N/A",
  "tomatoReviews":"N/A",
  "tomatoFresh":"N/A",
  "tomatoRotten":"N/A",
  "tomatoConsensus":"N/A",
  "tomatoUserMeter":"N/A",
  "tomatoUserRating":"N/A",
  "tomatoUserReviews":"N/A",
  "tomatoURL":"http://www.rottentomatoes.com/m/marvels_the_avengers/",
  "DVD":"25 Sep 2012",
  "BoxOffice":"$623,279,547.00",
  "Production":"Walt Disney Pictures",
  "Website":"http://marvel.com/avengers_movie","Response":"True"
}
```

FIGURE 4.8: The JSON format generated by the OMDb API

To extract the values of IMDB title, year and RottenTomatoes URI from the JSON data, it is imperative to perform JSON parsing. There are various JSON libraries in Java, and we employed the Jackson libray [20] to work with JSON data. The RT URI values of the results of the JSON parsing will be used to explore deeper into the RottenTomatoes website to get average ratings and user ratings. As the administrator of RottenTomatoes no longer provides a free open web API, we performed web scraping to retrieve the target values.

[19] `http://www.omdbapi.com/?i=tt0848228&tye=movie&&tomatoes=true&&r=json`
[20] `https://github.com/FasterXML/jackson-databind/`

As explained in Chapter 2 Section 2.5, the first step to performing web scrapping is to open a connection with the web sites through the HTTP protocol. Once the HTML document is retrieved, we are able to extract its content. For this purpose, we combined the use of Regular Expression techniques and an open source Java HTML parser, JSoup [21]. To produce the expected outputs from the RottenTomatoes website, web scraping is not only to extract and to transform contents into an expected data format but also to perform value conversion. The RottenTomatoes applies a 1-to-5 rating scale, while the IMDB adopts a 1-to-10 rating scale. Therefore, as aforementioned, we converted the values of average ratings into the IMDB rating standard. The results of process 4.9 added attributes such as: IMDB titles, IMDB years, RT titles, RT years, RT average ratings and RT user ratings.

4.2.10 Linking IMDb database and DBpedia

To exploit the potentiality of the IMDb downloadable files and extensive linked data, it is imperative to link the IMDb database and DBpedia. The IMDb database is generated from the process in sub-section 4.2.1, while the DBpedia data is obtained based on the process in sub-section 4.2.2. We used three types of movie title of IMDb website instead of DBpedia movie titles to link with movie titles in the IMDb database. The title types were display, original, and AKA (Also Known As). Figure 4.9 illustrates an example of movie which has a display title as well as an original title. While the display one was Antonia, the original one was Antônia. Moreover, Antonia the movie also has AKA titles as shown in Figure 4.10. A movie is possible to have more than one AKA title.

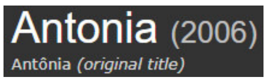
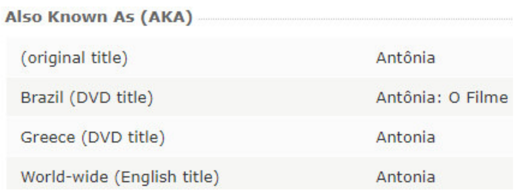

Antonia (2006)
Antônia *(original title)*

FIGURE 4.9: Example of display and original title for Antonia the movie


Also Known As (AKA)

(original title)	Antônia
Brazil (DVD title)	Antônia: O Filme
Greece (DVD title)	Antonia
World-wide (English title)	Antonia

FIGURE 4.10: Example of AKA titles for Antonia the movie

We linked 88,048 instances from DBpedia (containing links to Wikidata which has links to IMDb website) with a ”movies” table in the IMDb database. The movies table contains 3,754,328 movies. Both, IMDb website and movies table contain not only movies but also videos, tv series, tv mini-series, tv movies, tv shorts, tv specials, and tv episodes. Each of them has different identifers.

[21] https://jsoup.org/

Both, IMDb website and IMDb database have the same manner in building movie titles. They present ad-hoc identifiers for movies by concatenating title and running year (between brackets). For instance, they identify a movie with title 10 Cloverfield Lane released in year 2016 by 10 Cloverfield Lane (2016). As movie titles are not universal identifiers, so that, they may have the following drawbacks: abbreviation, different languages, and insufficiency of writing standardization Peralta (2007b). It has been proved by finding the discepancies of movie title in both datasets as illustrated in Table 4.10.

Table 4.10: Examples of Movie Title Discrepancy

NO	IMDb Website	movies Table	Causes
1	Crocodile Dundee II (1988)	'Crocodile' Dundee II (1988)	Using Original Title
	'38 - Vienna Before the Fall (1986)	'38 (1986)	
2	3 Generations (2015)	About Ray (2015)	Using AKA Title
	Antonia (2006)	Antônia: O Filme (2006)	
3	10 Minutes (2013)	10 Minutes (2013/II)	Display Title + Running Year + Additional Characters
	Artificial (2012)	Artificial (2012/III)	
4	Aftermath (2014)	Aftermath (2012/II)	Display Title + Different Running Year + Additional Characters
	Innocence (2013)	Innocence (2014/V)	
5	Alex & Eve (2016)	Alex & Eve (2015)	Display Title + Different Running Year
	New Blood (1999)	New Blood (2000)	
	American Exorcist (2018)	American Exorcist (2016)	
	City of Dead Men (2014)	City of Dead Men (2016)	
	A United Kingdom (2016)	A United Kingdom (????)	
6	Allegiance (2012)	Recalled (2012/I)	Original Title + Running Year + Additional Characters
	At the Devil's Door (2014)	Home (2014/VI)	
7	A Bottle in the Gaza Sea (2010)	Une bouteille à la mer (2011)	Original Title + Different Running Year
	Gladiators of Rome (2012)	Gladiatori di Roma (2013)	
8	The Last Survivors (2014)	The Well (2014/IV)	AKA Title + Running Year + Additional Characters
	Spark: A Space Tail (2016)	Spark (2016/I)	
9	Blood Brother (2017)	Brothers' Blood (2016)	AKA Title +

Continued on next page

Table 4.10 – continued from previous page

NO	IMDb Website	movies Table	Causes
	Manifest: The Chryzinium Era (2017)	Chryzinium (????)	Different Running Year

IMDb builds identifiers with different manner for videos than movies. IMDb website identifies movies by concatenating title and running year (between brackets), while videos are identified by concatenating title and brackets which contains a keyword video and running year. For instance, IMDb website identifies a video with title 2-Headed Shark Attack released in 2012 by 2-Headed Shark Attack (Video 2012). Furthermore, IMDb database differs in writing video titles than IMDb website. IMDb writes it as 2-Headed Shark Attack (2012) (V). Similar to the movie cases, it has been found the discepancies of video title between IMDb website and IMDb database as illustrated in Table 4.11.

Table 4.11: Examples of Video Title Discrepancy

NO	IMDb Website	movies Table	Causes
1	Medieval Fleshpots 2: Hot Wenches (Video 2003)	Lash of the Scorpion (2003) (V)	Using Original Title
	Time to Die (Video 2005)	Aika tappaa (2005) (V)	
2	The Blair Bitch Project (Video 1999)	The Blair Bitch Project starring Linda Blair (1999) (V)	Using AKA Title
	Joy Ride 3: Road Kill (Video 2014)	Joy Ride 3 (2014) (V)	
3	The Amityville Curse (Video 1990)	The Amityville Curse (1989) (V)	Display Title + Different Running Year
	Barbie in a Mermaid Tale 2 (Video 2011)	Barbie in a Mermaid Tale 2 (2012) (V)	
	Debbie Does Dallas Again (Video 2005)	Debbie Does Dallas Again (2007) (V)	
4	Tortured (Video 2008)	Tortured (2008/I) (V)	Display Title + Running Year + Additional Characters
5	Nine Inch Nails: Broken (Video 1993)	Broken (1992) (V)	AKA Title + Different Running Year

IMDb builds identifiers with different manner for TV's (TV Series, TV Mini-Series, TV Movies, TV Shorts, TV Specials, and TV Episodes) than movies and videos. Table 4.12 presents the comparison of TV Identifiers are presented in the IMDb website and the IMDb Database.

Table 4.12: The Comparison of TV Identifiers Between IMDb Website and IMDb Database

NO	**IMDb Website**	**movies Table**	**Identifier**
1	Cedric the Entertainer Presents (TV Series 2002–2003)	"Cedric the Entertainer Presents" (2002)	TV
	Cinéast(e)s (TV Series 2013–)	"Cinéast(e)s" (2013)	Series
2	Flying: Confessions of a Free Woman (TV Mini-Series 2006–2008)	"Flying: Confessions of a Free Woman" (2006)	TV
	Lego Marvel Super Heroes: Maximum Overload (TV Mini-Series 2013–)	"Lego Marvel Super Heroes: Maximum Overload" (2013)	Mini-Series
3	Tales from the Organ Trade (TV Movie 2013)	Tales from the Organ Trade (2013) (TV)	TV
	Donald Trump's The Art of the Deal: The Movie (TV Movie 2016)	Donald Trump's The Art of the Deal: The Movie (2016) (TV)	Movie
4	Lego Indiana Jones and the Raiders of the Lost Brick (TV Short 2008)	Lego Indiana Jones and the Raiders of the Lost Brick (2008) (TV)	TV
	The Smurfs: The Legend of Smurfy Hollow (TV Short 2013)	The Smurfs: The Legend of Smurfy Hollow (2013) (TV)	Short
5	Look, Up in the Sky! The Amazing Story of Superman (2006)	Look, Up in the Sky! The Amazing Story of Superman (2006) (TV)	TV
	Vampire Secrets (2006)	Vampire Secrets (2006) (TV)	Special
6	"30-Second Bunny Theatre" The Shining (TV Episode 2004)	"30-Second Bunny Theatre" (2004) {The Shining}	TV
	"Style Exposed" Baring It All (TV Episode 2011)	"Style Exposed" (2011) {Baring It All}	Episode

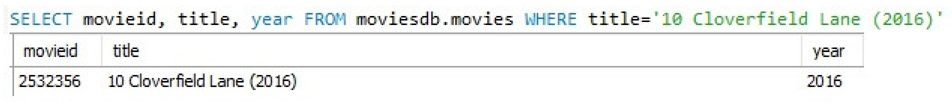
To deal with the unavailaibility of universal identifier, we applied several strategies to match the titles (movie, video, tv serie, tv mini-serie, tv movie, tv short, tv special, and tv episode) between IMDb website and the movies table. The titles matching process involved 25 steps, as presented in Table 4.13. Each step attempts to join the titles in IMDb website not matched by previous steps. Each step performs a different strategy to build candidate titles for unmatched titles and compares them with titles in movies table. The followings are the details of stategy performed to match the titles:

Table 4.13: Steps of the Title Matching Process

Steps	Matched	Total	%	Steps	Matched	Total	%
Match1	64,557	64,557	73.32	Match14	18	86,601	98.36
Match2	17,352	81,909	93.03	Match15	1	86,602	98.36
Match3	84	81,993	93.12	Match16	1	86,603	98.36
Match4	1,513	83,506	94.84	Match17	39	86,642	98.40
Match5	16	83,522	94.86	Match18	40	86,682	98.45
Match6	915	84,437	95,90	Match19	642	87,324	99.18
Match7	86	84,523	96.00	Match20	40	87,364	99.22
Match8	402	84,925	96.45	Match21	28	87,392	99.25
Match9	5	84,930	96.46	Match22	124	87,516	99.40
Match10	14	84,944	96.47	Match23	379	87,895	99.83
Match11	1,569	86,513	98.26	Match24	64	87,959	99.90
Match12	66	86,579	98.33	Match25	89	88,048	100
Match13	4	86,583	98.34				

1. Matching Movie Titles using IMDb website display titles (Match1)

   We extracted display titles from the IMDb website and kept them in the original forms to join them with titles written in movies table. For instance, IMDb display title of Cloverfield Lane the movie [22] released in 2016 is 10 Cloverfield Lane (2016). To match it with the one in the movies table, we run a SQL query as shown in Figure 4.11 and it also indicated the query result. It may some movies having the same titles and running years. Therefore, we performed two validations to test the candidate title. We used information both director and writer from IMDb website as illustrated in Figure 4.12 to validate the candidate title.



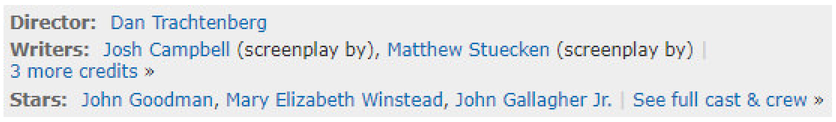
```
SELECT movieid, title, year FROM moviesdb.movies WHERE title='10 Cloverfield Lane (2016)'
```

```
movieid   title                          year
2532356   10 Cloverfield Lane (2016)     2016
```

FIGURE 4.11: A SQL for matching a IMDb display title

According to figure 4.12, Cloverfield Lane the movie was directed by Dan Tracht-enberg and one of its writers was Josh Campbell. We used those information and

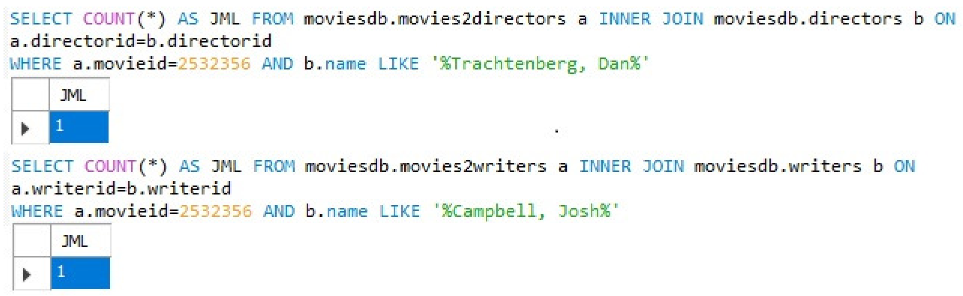
[22] http://www.imdb.com/title/tt1179933/



Director: Dan Trachtenberg
Writers: Josh Campbell (screenplay by), Matthew Stuecken (screenplay by) | 3 more credits »
Stars: John Goodman, Mary Elizabeth Winstead, John Gallagher Jr. | See full cast & crew »

FIGURE 4.12: Example of director and writers for Cloverfield Lane the movie

movieid 2532356 as indicated in Figure 4.11 to validate the candidate title. We wrote two SQL queries as shown in Figure 4.13 to perform validation processes. If the result of both queries indicated greater than zero, than, it can be said that the candidate title was valid. This strategy obtained 64,557 matches (73.32%).



```
SELECT COUNT(*) AS JML FROM moviesdb.movies2directors a INNER JOIN moviesdb.directors b ON
a.directorid=b.directorid
WHERE a.movieid=2532356 AND b.name LIKE '%Trachtenberg, Dan%'
```

	JML
▶	1

```
SELECT COUNT(*) AS JML FROM moviesdb.movies2writers a INNER JOIN moviesdb.writers b ON
a.writerid=b.writerid
WHERE a.movieid=2532356 AND b.name LIKE '%Campbell, Josh%'
```

	JML
▶	1

FIGURE 4.13: A SQL for validating the movie matched using director

2. Matching Movie Titles using IMDb website original titles (Match2)

   As illustrated in Figure 4.9, IMDb website informs not only display titles but also original titles. We considered to match titles in movies table with IMDb original titles instead of display titles. This strategy was used to address the first problem as shown in Table 4.10. As a result, we extracted original titles from IMDb website based on unmatched DBpedia movies obtaining candidate titles. Then, we linked them with titles in movies table. Lastly, we performed validation processes to candidate titles as the same as Match1 using directors and writers. It was obtained 17352 new matches, totalizing 81,909 matches (93.03%).

3. Matching MovieTitles using IMDb website AKA titles (Match3)

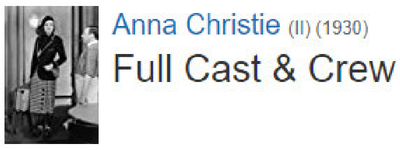
   Besides display titles and original titles, IMDb webiste also provides information of AKA titles. As illustrated in the second case in Table 4.10, AKA titles may cause discrepancies in building movie identifiers. We obtained AKA titles by adding a suffix releaseinfo into IMDb movie URL's. For example, the URL of Antonia (2006) the movie is [23]. To obtain the AKA titles of it, we have to visit the following URL: [24]. An IMDb movie could have more than one AKA title. To match IMDb AKA titles with titles in movies table, we implemented some steps. The first one was to query unmatched DBpedia movies and to retrieve the corresponding IMDb URL's. Secondly, We added a suffix releaseinfo to each IMDb URL and used it as a landing page to extract AKA titles. Thirdly, we concatenated each AKA

[23]http://www.imdb.com/title/tt0492869/
[24]http://www.imdb.com/title/tt0492869/releaseinfo

title and release year (between brackets), then we linked it with titles in movies table. Lastly, we validated the candidate titles using directors and writers, as we did in Match1 and Match2. This strategy obtained 84 matches, totalizing 81,993 matches (93.12%).

4. Matching MovieTitles using IMDb website display titles with additional characters (Match4)

   This strategy was intended to cope the third case caused by additional characters after running year as illustrated in Table 4.10. For instance, IMDb displays Anna Christie the movie as Anna Christie (1930). If we inspected manually into the movies table, the Anna Christie movie was stored in the movies table as Anna Christie (1930/II). There was additional charaters "/II" after the running year. Obviously, we can not perform the normal matching process by linking movie title. Our first step in this strategy was to select the rest of unmatched DBpedia movies from the the process in sub-section 4.2.2 to get IMDb URL's. Subsequently, we added them with a suffix "fullcredits" and used them to extract movie titiles. For instance, the original of Anna Christie's is [25]. The new complete URL would be [26]. This URL would bring us to a web page that provides an IMDb display title including additional characters as shown in Figure 4.14 as Anna Christie (II) (1930). We extracted and parsed the title, so that, we obtained a candidate title as Anna Christie (1930/II). Than, we used it to link with a title in movies table. As performed in the three previous strategies, we also used two validation processess (directors and writers) in this strategy. For the linking process did not yield a result. We gave a marker to indicate this strategy. The marker would be used to the subsequent strategy. The results of this strategy ware 1,513 new matches, totalizing 83,506 movies (94.84%).



Anna Christie (II) (1930)

Full Cast & Crew

FIGURE 4.14: Anna Christie - The Movie

5. Matching MovieTitles using IMDb website display titles with different running years and additional characters (Match5)

   This strategy contributed to solve the fourth problem as indicated in Table 4.10. We started this strategy by querying unmatched DBpedia movies with markers as given in Match4. For each movie, we linked a title without a running year to movies table. This linking may return more than one results. To get the most fit candidate, we validated each movie by adding two parameters: director and

[25] http://www.imdb.com/title/tt0020642/

[26] http://www.imdb.com/title/tt0020642/fullcredits

writer. The validation processes were the same as performed in Match1 to Match4. It was obtained 16 new matches by running this strategy, totalizing 83,522 movies (94.86%).

6. Matching MovieTitles using IMDb website display titles with different running years (Match6)

   As depicted in the fifth case in Table 4.10, movie titles may embed different years in brackets. Our strategy is to replace original running years with new formed running years. We specified -5 for a lower limit and 5 for an upper limit. We queried unmatched DBpedia movies. For each movie, we extracted titles and ignored years. We used years to form a new running year. We iterated from lower limit to upper limit and added iteration value with an original running year to form new running years. Then, we concatenated the title and the new running year in brackets, then we linked it with titles in movies table. If the iteration already reached an upper limit and there was no math found, then we applied a new formulasim which was the title concatenated with {????}. Lastly, we applied two validation processes (director and writer) as we did to all previous matching processes. As a result, it was obtained 915 new matches, totalizing 84,437 movies (95.90%).

7. Matching MovieTitles using IMDb website original titles with additional characters (Match7)

   This strategy was focused to solve problems as illustrated in the sixth case of Table 4.10. It was the same with the strategy to fix problems number three. The distinguish thing was the use of original titles instead of display titles. By applying this strategy, it was obtained 86 new matches, totalizing 84,523 movies (96.00%).

8. Matching MovieTitles using IMDb website original titles with different running years (Match8)

   We intended to use this strategy to cope with problems number seven as shown in Table 4.10. It was the same with the strategy to fix problems number five. The distinguish thing was the use of original titles instead of display titles. By applying this strategy, it was obtained 402 new matches, totalizing 84,925 movies (96.45%).

9. Matching MovieTitles using IMDb website AKA titles with additional characters (Match9)

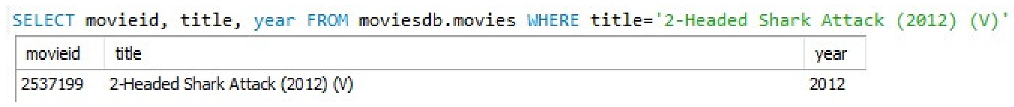
   This strategy was focused to solve problems as illustrated in the eighth case of Table 4.10. It was the same with the strategy to fix problems number three and six. The distinguish thing was the use of AKA titles instead of display titles and original titles. By applying this strategy, it was obtained 5 new match, totalizing 84,930 movies (96.46%).

10. Matching MovieTitles using IMDb website AKA titles with different running years (Match10)

    We intended to use this strategy to cope with problems number nine as shown in Table 4.10. It was the same with the strategy to fix problems number five and seven. The distinguish thing was the use of AKA titles instead of display titles and original title. By applying this strategy, it was obtained 14 new matches, totalizing 84,944 movies (96.47%).

11. Matching Video Titles using IMDb website display titles (Match11)

    We extracted video display titles by filtering titles which only contain a keyword video. For each video, we extracted title, running year and removed keyword video. We built a candidate title by concatenating display title, running year in brackets, and a letter V in brackets. For instance, a video called 2-Headed Shark Attack [27] released in 2012 has an IMDB display title as 2-Headed Shark Attack (Video 2012). Based on it, we obtained a candidate title, as follow: 2-Headed Shark Attack (2012) (V). To match it with the one in movies table, we run a SQL query as indicated in Figure 4.15. It also indicated the query result. To get the most accuracy result, we performed two validations (director and writer) as performed in the movie titles matching. This strategy obtained 1,569 new matches, totalising 86,513 movies (98.26%).



```
SELECT movieid, title, year FROM moviesdb.movies WHERE title='2-Headed Shark Attack (2012) (V)'
```

movieid	title	year
2537199	2-Headed Shark Attack (2012) (V)	2012

FIGURE 4.15: A SQL for matching a video title using na IMDb display title

12. Matching Video Titles using IMDb website original titles (Match12)

    As depicted in the first case of Table 4.11, there were some video titles in movies table written using original titles. This strategy was to build candidate titles by concatenating original titles instead of display titles and a letter V in brackets. Similar to the previous steps, we also performed directors and writers validation to get exact matches. This strategy was succesful to match 66 records. Therefore, the new total matches were 86,579 records (98.33%).

13. Matching Video Titles using IMDb website AKA titles (Match13)

    This strategy was focused to solve problems as illustrated in the second case of Table 4.11. This strategy was to build candidate titles by concatenating AKA titles instead of display titles and original titles and a letter V in brackets. Similar to the previous steps, it were also performed directors and writers validation to get exact matches. This strategy obtained 4 new matches, totalising 86,583 records (98.34%).

[27]http://www.imdb.com/title/tttt2043757/

14. Matching Video Titles using IMDb video display titles with different running years (Match14)

    As depicted in the third case in Table 4.11, video titles may embed different years in brackets. Our strategy is to replace original running years with new formed running years. We specified -5 for a lower limit and 5 for an upper limit. We queried unmatched DBpedia movies. For each video, we extracted titles and ignored years. We used years to form a new running year. We iterated from lower limit to upper limit and added iteration value with an original running year to form new running years. Then, we concatenated the title, the new running year in brackets as well as a letter V in brackets, then we linked it with titles in movies table. If the iteration already reached an upper limit and there was no math found, then we applied a new pattern which was the title concatenated with {????}. Lastly, we applied two validation processes (director and writer) as we did to all previous matching processes. This strategy obtained 18 new matches, totalising 86,601 records (98.34%).

15. Matching Video Titles using IMDb Video Display Titles with Additional Character (Match15)

    This strategy was intended to cope the fourth case caused by additional characters after running year as illustrated in Table 4.11. Our first step in this strategy was to select the rest of unmatched DBpedia movies from the the process in sub-section 4.2.2 to get IMDb URL's. Subsequently, we added them with a suffix "fullcredits". This URL would bring us to a web page that provides an IMDb video display title including additional characters, running year, and a keyword Video. We extracted and parsed those attributes, so that, we can build a candidate title by concatenating the title, the running year followed by a slash and an additional character in brackets, and a letter V in brackets. Then, we used it to link with a title in movies table. As performed in the the previous strategies, we also used two validation processess (directors and writers) in this strategy. This strategy obtained 1 new matches, totalising 86,602 records (98.36%).

16. Matching Video Titles using IMDb Video AKA titles with Different Running Years (Match16)

    This strategy was intended to solve the fifth case in Table 4.11. To get an IMDb Video AKA title, it was queried unmatched DBpedia movies and retrieved the corresponding IMDb URL. Secondly, it was added a suffix releaseinfo to the IMDb URL and used it as a landing page to extract AKA titles. This strategy was to build candidate titles by concatenating AKA titles instead of display titles and original titles, a running year in brackets as well as a letter V in brackets. The apporach to form a running year was similar to the Match14 strategy. Similar to the previous steps, it were also performed directors and writers validation to get exact matches.This strategy obtained 1 new matches, totalising 86,603 records (98.36%).

17. Matching TV Series Titles using IMDb TV Series Display Titles (Match17)

    As depicted in the first example in Table 4.12, IMDb website builds an identifier for TV Series by concatenating a title and a keyword TV Series followed by a running year in brackets, while movies Table constructs an TV Series identifier by concatenating a title in quotes and a running year in brackets. To match TV Series titles between IMDb website and IMDb database, firstly, it was queried the rest of unmatched DBpedia movies from the the process in sub-section 4.2.2 to get IMDb URL's. For each the IMDb URL, we used it to extract a display title and a running year. We only extracted information that contained a TV Series keyword. Secondly, we built a candidate title by concatenating the display title (between quotes) and the running year (between brackets). Thirdly, we linked the candidate title with a title in movies table. Lastly, we performed directors and writers validation as done in the previous steps. This strategy obtained 39 new matches, totalising 86,642 records (98.40%).

18. Matching TV Mini-Series Titles using IMDb TV Mini-Series Display Titles (Match18)

    As portrayed in the second example in Table 4.12, IMDb website builds an identifier for TV Mini-Series by concatenating a title and a keyword TV Mini-Series followed by a running year in brackets, while movies Table constructs an TV Mini-Series identifier the same as an TV Series identifier. To match TV Mini-Series titles between IMDb website and IMDb database, firstly, it was queried the rest of unmatched DBpedia movies from the the process in sub-section 4.2.2 to get IMDb URL's. For each the IMDb URL, we used it to extract a display title and a running year. We only extracted information that contained a TV Mini-Series keyword. The steps 2 to 4 were the same as the Match17 steps. This strategy obtained 40 new matches, totalising 86,682 records (98.45%).

19. Matching TV Movie Titles using IMDb TV Movie Display Titles (Match19)

    As depicted in the third example in Table 4.12, IMDb website builds an identifier for TV Movie by concatenating a title and a keyword TV Movie followed by a running year in brackets, while movies Table constructs an TV Movie identifier by concatenating a title and a TV keyword in brackets. To match TV Movies titles between IMDb website and IMDb database, firstly, it was queried the rest of unmatched DBpedia movies from the process in sub-section 4.2.2 to get IMDb URL's. For each the IMDb URL, we used it to extract a display title and a running year. We only extracted information that contained a TV Movie keyword. Secondly, we built a candidate title by concatenating the display title, the running year (between brackets) and the TV keyword (between brackets). The steps 3 to

4 were the same as the Match17 steps. This strategy obtained 642 new matches, totalising 87,324 records (99.18%).

20. Matching TV Short Titles using IMDb TV Short Display Titles (Match20)

    As presented in the fourth example in Table 4.12, IMDb website builds an identifier for TV Movie by concatenating a title and a keyword TV Short followed by a running year in brackets, while movies Table constructs an TV Short identifier the same as an TV Movie identifier. To match TV Short titles between IMDb website and IMDb database, firstly, it was queried the rest of unmatched DBpedia movies from the the process in sub-section 4.2.2 to get IMDb URL's. For each the IMDb URL, we used it to extract a display title and a running year. We only extracted information that contained a TV Short keyword. The steps 2 to 4 were the same as the Match19 steps. This strategy obtained 40 new matches, totalising 87,364 records (99.22%).

21. Matching TV Special Titles using IMDb TV Special Display Titles (Match21)

    As figured in the fifth example in Table 4.12, IMDb website builds an identifier for TV Movie by concatenating a title and a running year in brackets, while movies Table constructs an TV Special identifier the same as an TV Movie and TV Short identifiers. A TV Special identifier resembles a movie identifier, however, there is a clear difference in the corresponding page of IMDb website that mentions the particular movie is a TV Special. To match TV Special titles between IMDb website and IMDb database, firstly, it was queried the rest of unmatched DBpedia movies from the the process in sub-section 4.2.2 to get IMDb URL's. For each the IMDb URL, we used it to extract a display title and a running year. We only extracted information that contained a TV Special keyword. The steps 2 to 4 were the same as the Match19 and Match20 steps. This strategy obtained 28 new matches, totalising 87,392 records (99.25%).

22. Matching TV Episode Titles using IMDb TV Episode Display Titles (Match22)

    As depicted in the last example in Table 4.12, IMDb website builds an identifier for TV Episode by concatenating a title of TV Series in quotes, a title of TV Episode and a keyword TV Episode followed by a running year in brackets, while movies Table constructs an TV Episode identifier by concatenating a title of TV Series in quotes, a running year in brackets and a title of TV Episode in curly brackets. To match TV Episode titles between IMDb website and IMDb database, firstly, it was queried the rest of unmatched DBpedia movies from the the process in sub-section 4.2.2 to get IMDb URL's. For each the IMDb URL, we used it to extract a display title of TV Episode, a display title of TV Series and a running year. Secondly, we built a candidate title by concatenating the TV Series title (between quotes), the running year (between brackets), and the TV Episode title in curly brackets. The steps 3 to 4 were the same as the Match17 steps. This strategy obtained 124 new matches, totalising 87,516 records (99.40%).

23. Matching Movie Titles using Approximate Matching String (Match23)

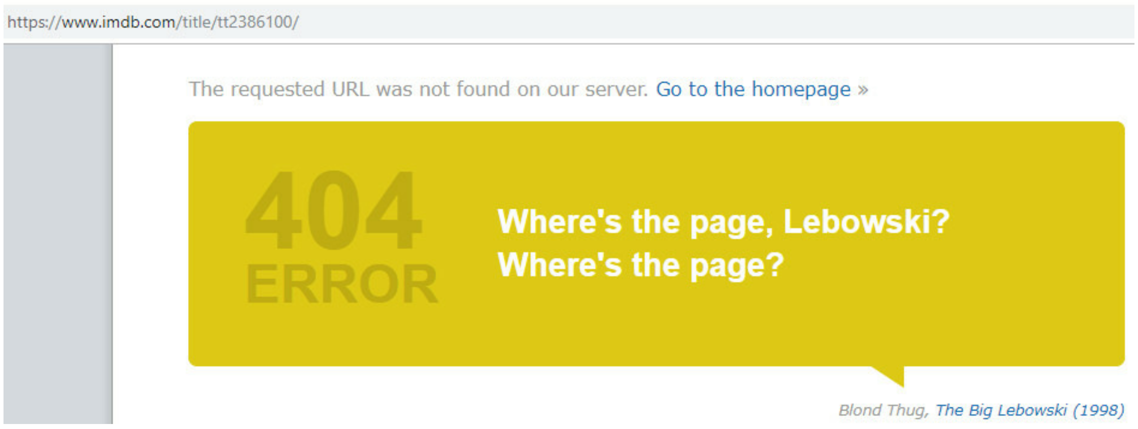
    Firstly, we selected the movies Table as the base source and the table generated from the process in sub-section 4.2.2 as the the query source. We queried all data (3,754,328 records) from the movies Table and wrote the query results into a text file. Secondly, we queried the rest of unmatched DBpedia movies from the query source to get IMDb URL's. For each the IMDb URL, we used it to extract a display title and a running year. Thirdly, we built a candidate title by concatenating the display title and the running year in brackets. Fourthly, we linked the candidate title with a movie title in the text file and measured their similarity. We employed Jaccard Distance Similarity algorithm to measure the similarity of two movie titles and set a threshold θ by 0.5. If the similarity of the candidate title is greater that or equal tho the θ, then, we performed directors and writers validation as indicated in the Match1. If the directors and writers validation result in the score more than zero, then, the candidat title is valid. This strategy obtained 379 new matches, totalising 87,895 records (99.83%). Table 4.14 exhibites examples of the similarity measurement results movie titles between the IMDb website and the IMDb database.

Table 4.14: Examples of The Similarity Measurement between Movie Titles of the IMDb Website and Movie Titles of the movies Table

NO	IMDb Website	movies Table	Similarity
1	Zoi jeung gong woo (1990)	Choi saan gong woo (1990)	0.56
2	Jai-Vijay (1977)	Jay-Vejay: Part - II (1977)	0.61
3	Margaret Cho: I'm the One That I Want (2000)	I'm the One That I Want (2000)	0.71
4	Genomu hazâdo: Aru tensai kagakusha no 5-kakan (2013)	Genom Hazard: aru tensai kagakusha no itsukakan (2013)	0.82
5	Slaughter on 10th Avenue (1957)	Slaughter on Tenth Avenue (1957)	0.90
6	Peace, Love & Misunderstanding (2011)	Peace, Love, & Misunderstanding (2011)	1.00

24. Matching to Identify Broken Links (Match24)

    We queried the rest of unmatched DBpedia movies from the query source to get IMDb URL's. For each the IMDb URL, we used it to check the active status of the link. If the link brought us to a web page as indicated in Figure 4.16. It means that the link is not active or broken. By applying this approach, we identified 64 broken links, totalising 87,959 records (99.90%).


https://www.imdb.com/title/tt2386100/

The requested URL was not found on our server. Go to the homepage »

404 ERROR

Where's the page, Lebowski? Where's the page?

Blond Thug, The Big Lebowski (1998)

FIGURE 4.16: Page is Not Found

25. Matching to Identify the Availaiblity of Movie Titles in the movies Table (Match25)

    Firstly, we queried the rest of unmatched DBpedia movies from the query source to get IMDb URL's. For each the IMDb URL, we used it to extract a display title and a running year. Thirdly, we built a candidate title by concatenating the display title and the running year in brackets. Fourthly, we queried the movies Table or the base source and applied the candidate title as a querying parameter. If the query did not return any results, then, we performed manually inspection to assure whether the candidate title is available or not in the base source. This strategy found 89 movie titles that was not available in the base source, totalising 88,048 records (100.00%).

4.3 Data Accessing

After performing processes in section 4.2, we integrated movie data stored in the MySQL database. The database is comprised of:

- Data transformation from 48 list files.
- Linking to the following external sources: DBpedia, Wikidata, and movie-related websites (IMDb, BoxOfficeMojo, FilmAffinity, Metacritic, MovieMeter, and RottenTomatoes).
- Having data from the external data sources as described in Table 4.1.

We used two different ways to access the data for data mining purposes.

- SQL.

  We access the data using SQ statements to build classification models which are intended to demonstrate building the models from a single data source.

- SPARQL

  The data accessing by using SPARQL queries are intended to exhibit the use of multiple data sources to build classification models.

4.3.1 Publishing the Database Content on the Web

Besides the use of SQL queries to access the integrated movie database, we also facilitated to bringing the database into the Web of Linked Data. The linked data is commonly used to integrate data from different sources on the web. Furthermore, Linked Data is presented in the RDF data model which is capable to understand both humans and machines. To enable the integrated movie database transformed into an RDF format, we used the D2R server Bizer and Cyganiak (2006).

D2R Server has two important features. Firstly, it brings through two types of browser (HTML and RDF) to explore the content of related databases. Secondly, it provides a SPARQL endpoint, so client applications are able to query the database using SPARQL Queries. To allow the D2R Server to perform on-the-fly translation from SPARQL queries to SQL queries, we prepared a D2RQ mapping file (Bizer and Seaborne (2004)). By using the mapping between RDBMS schemas and non-RDBMS schemas (RDFS or OWL ontologies), the D2RQ mapping file can identify resources and generate property values from the content of databases.

To map our movie database to RDF, we firstly needed to define database connection by using a database object in D2RQ to map a particular database. The D2RQ statement used to define the research database can be seen in Appendix B, Figure B.1. Secondly, we to mapped a table in the database by using a ClassMap object of D2RQ. This object is used to map an entity defined in the database, to a class of RDF resource. For instance, we aimed to map a movies table in the database to RDF. To do so, it is required to specify a primary key of the table and map it to the property uriPattern of the ClassMap object. We assigned the uriPattern property with the following values “movies/@@movies.movieid@@”. The mapping result would be identified by an URI such as http://localhost:2020/page/movies/2531289, where 2531289 is the primary key. Figure B.2 in Appendix B demonstrates how to map a movies table to RDF.

Thirdly, we mapped a field in the table of the database to a property of RDF. Each ClassMap has a set of PropertyBridges, used to attach a field in the table to a RDF property. The value of belongsToClassMap property of the PropertyBridge indicates the link between a ClassMap object and a PropertyBridges object. A PropertyBridge allows to attach properties, such as: dc:title, dc:description, owl:sameAs, dc:creator, foaf:Person, foaf:Name, foaf:mbox, rdfs:label etc to each resource, which their values are taken from fields in the database. Furthermore, a PropertyBridge also permits researchers to attach values from a different table via a join. Figures B.3 and B.6 in Appendix B indicates how to attach properties of dc:title and rdfs:label, respectively to a RDF resource, whose value is taken from the value of a field of the movies table, namely title.

4.4 Data Preparation

This section provides an appropriate approach along with notable factors to be considered for developing readily datasets for movie popularity classification models. We classified datasets into three types. The first type of dataset is called IMDb. The second one is namely IMDb++. The last one contains three datasets which are FilmAffinity, MovieMeter, and RottenTomatoes. The IMDb dataset is a reference for the others dataset in determining the selected movies for training data. All datasets involved 6,287 movies for analysis and classification purposes. Those movies are gathered from the integrated movie database by applying 5 filters. The first filter is movies released between 1990 and 2017. As argued by Asad et al. (2012), movies released after year 2000 fell around the same time which delighted in the product of innovation. However, it is hard to find movies in the low ratings in these periods. Therefore, we made the range of movie released years wider. We believe that the wider the year range is the more possibilites to get movies with the high, medium and low ratings. The second and third ones are English movies released in the USA. As suggested by Asad et al. (2012), IMDB list files potentially contain more movie data that based on English language and released in the USA. The fourth one is only movies receiving more than 1,000 user votes. As mentioned by the previous researcher (Saraee et al. (2004)), the fourth filter was applied to elliminate the bias of an unknown movie with a few high votes. The last filter is only movies listed in DBpedia, IMDb list files (movies.lst) and IMDb website. The reason to apply this filter is because DBpedia provides links to Wikidata. Subsequently, Wikidata provides links to the other resources, such as the BoxOfficeMojo, FilmAffinity, IMDb, Metacritic, MovieMeter, and RottenTomatoes websites. Therefore, the consideration to use only movies provided in DBpedia as the data training is for the ease of data integration. Details of the generation of all dataset types will be explained in the following sub sections: (i) Generating the IMDb Dataset, (ii) Generating the IMDB++ Dataset and (iii) Generating the FilmAffinity, MovieMeter, and RottenTomatoes Datasets.

4.4.1 Generating the IMDb Dataset

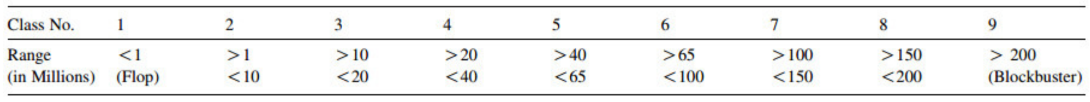
The IMDb dataset is generated only from IMDb list files. To build a dataset for classification purposes, it is required to define input attributes and an output attribute. The input attributes are factors possibly recognised as the success factor of a movie to get a high rating from users.

4.4.1.1 The Input Attributes of the IMDb Dataset

According to previous studies, useful attributes that are considered in the movie domain prediction may include:

- Budget.

  Budget is the amount of money allocated for a whole film-making purpose. A film budget may include costs for the following purposes: production, advertising and marketing (Simonoff and Sparrow (2000); Sharda and Delen (2006)). The budget can vary from a budget of just several hundred dollars to one of millions of dollars (Saraee et al. (2004)). For instance, Tarnation was produced in 2004 on a budget of just 220 dollars; in contrast, Pirates of the Caribbean: At World's End (2007) had a budget of 300 million dollars. The budget variable was mentioned by (Saraee et al. (2004); Asad et al. (2012); Latif and Afzal (2016)) in their studies to predict the rating of a film. Budget is a continuous attribute, so we have to discretise this into nine classes, following the guidance from (Sharda and Delen (2006)) as shown in Figure 4.17.



Class No.	1	2	3	4	5	6	7	8	9
Range (in Millions)	< 1 (Flop)	> 1 < 10	> 10 < 20	> 20 < 40	> 40 < 65	> 65 < 100	> 100 < 150	> 150 < 200	> 200 (Blockbuster)

FIGURE 4.17: 9 Classes Discretization

- Competition.

  (Simonoff and Sparrow (2000); Sharda and Delen (2006)) included the time of release, as an indication of the level of competition. Competition is considered to bring a significant impact to a success factor of future movies. In the (Sharda and Delen (2006)) study, they categorise the competition into three groups: high, medium and low. Movies released in June and November are categorised as being high competition. The medium competition comprises of movies with the release months of May, July, and December. Low competition represents the movies released in the remaining months.

- Genre.

A commonly used attribute in predicting the future success of a movie is content category, or genre. (Sharda and Delen (2006)) involved this as one of the input attributes for predicting the financial success of a movie, while (Latif and Afzal (2016)) used it to predict the popularity of a movie. Genre represent the type of content present in a movie. A movie may fall into a single genre or many genres. For instance, 100 Tears, a movie released in 2007, is categorised as a single genre: horror. #Horror, a movie released in 2015, has 4 genres: drama, horror, mystery and thriller. Our integrated movie database includes 20 genres: action, adventure, thriller, biography, crime, drama, horror, comedy, fantasy, animation, mystery, music, war, documentary, romance , Scientific-fiction, western, family, sport, and short.

- MPAA Rating.

  Another popular attribute used to predict the future success of a movie is MPAA rating. A movie rating will be assigned by the Motion Picture Association of America (MPAA) based on its content. Earlier studies from (Simonoff and Sparrow (2000); Sharda and Delen (2006); Latif and Afzal (2016)) have included this attribute in their studies to investigate the future success factor of a movie. These ratings represent violence, sexual content, and language in a movie. There are 5 categories for each of the movies, mainly R, PG, PG13, G and NR.

- Actors and Actress Rating .

  (Saraee et al. (2004)) argued that ratings of the inherent movie attributes, such as: actors, actress, and directors would have considerable impact on the future popularity of a movie. They introduced a transformation method to calculate a numerical rating for casts (actors and actress), and directors. (Saraee et al. (2004)) used the sum function, to sum up all ratings for each cast member to get an Actor Rating and an Actress Rating for that film. This approach has also been followed by (Asad et al. (2012)). Both included the attributes of actors rating and actress rating in their movie popularity classification models.

- Directors Rating .

  To get a director rating for a film, (Saraee et al. (2004)) took the average rating of the directors involved. They considered that having more two excellent-rated directors working on a film is not a guarantee to make it better than having less directors working on it. (Asad et al. (2012)) also adopted this technique, and involved this attribute in their classification models for movie popularity.

Other inherent movie attributes from the IMDb list files that may be useful to predict the future popularity of a movie are cinematographers, composers, costume designers, distributors, editors, production companies, product designers, and writers. To the best our knowledge, those attributes have not been investigated by other researchers as the consideration factors to the future success of movies. We used the sum function only to get a distributor rating, while the other attributes were used to gauge the average function to get the ratings. We believe more distributors involved in marketing a movie will increase that movie's likelihood of popularity. Table 4.15 presents the summary of calculation methods to transform numerical rating of movie inherent attributes.

Table 4.15: Summary of Calculation Methods Applied to Transform Numerical Rating of Movie Attributes

NO	ATTRIBUTE NAMES	CALCULATION METHODS
1	Actor Rank	Sum
2	Actress Rank	Sum
3	Cinematographer Rank	Average
4	Composer Rank	Average
5	Costume Designer Rank	Average
6	Director Rank	Average
7	Distributor Rank	Sum
8	Editor Rank	Average
9	Production Company Rank	Average
10	Production Designer Rank	Average
11	Producer Rank	Average
12	Writer Rank	Average

Table M.1 in Appendix M indicates the values and number of input attributes used to generate the IMDb dataset, used to predict the future of popularity of a movie. The values in red fonts in the Attribute Name column highlights the input attributes that have not not been used in any of the previous work. As a movie may have more than a single genre, the final IMDb dataset contains 16,008 instances. Meanwhile, table N.1 in Appendix N presents a full-example of the Hacksaw Ridge the Movie with its values based on the IMDb dataset.

4.4.1.2 The Output Attributes of IMDb Dataset

This attribute serves as a dependent attribute or a class in prediction. The data is based on user rating as provided by IMDb. IMDb provides a rating scale between 0 and 10, and every IMDb user can vote their favourite movie. Movie rating is a weighted average vote, and the value is continuous. To complete the aim of analysis and classification, we generalised the continuous numeric value of the average of voting into four categories, as shown in Table 4.16. This approach has been used by previous researchers Saraee et al. (2004) Asad et al. (2012) Latif and Afzal (2016).

Table 4.16: The Class Output of The IMDb Dataset

Class	Rating
Excellent	7.5 - 10
Average	5 - 7.4
Poor	2.5 - 4.9
Terrible	1 - 2.4

4.4.2 Generating the IMDb++ Dataset

The IMDb++ dataset constitutes an extension of the IMDb dataset (the first type of dataset). There is only one dataset that is categorised into the second type of dataset: IMDb++. For the second type of dataset, we added another seven new input attributes into the existing input attributes, as described in Table M.1 in Appendix M. The new input attributes come from three different websites (BoxOfficeMojo, IMDb, and Metacritic). The additional input attributes used to build classification models for movie popularity are described as follows:

- Awards (Golden Globe Nominee, Golden Globe Win, Oscar Nominee, and Oscar Win)

  Academy awards, also known as Oscars, and Golden Globe awards are recognised for excellence in movie achievements. Many movies are awarded as winners and many others are awarded as nominees, and it is still a big achievement to even get nominated within the competitive film industry. Simonoff and Sparrow (2000) Latif and Afzal (2016) mentioned these attributes in previous studies. Those attributes are found in the IMDb website.

- Metascore.

  Metascore is a single number derived from the average value of movie review scores by a large group of famous critics. The condition of a film gets a metascore if it gets at least four critic reviews. Metascore indicates positive reviews for a particular film and can be obtained in the Metacritic website. Previous researchers Schaible et al. (2015) Latif and Afzal (2016) discussed this attribute in their studies for predicting the future of a movie.

- Opening Theatres.

  Opening theatres is the number of screenings of a movie scheduled to show at its opening. Previous researchers Simonoff and Sparrow (2000) Sharda and Delen (2006) Latif and Afzal (2016) indicated close correlations between the future of a movie and opening theatres. We added this attribute into the existing input attributes and represented it as continuous values. We collected the values of opening theatres from the BoxOfficeMojo website.

- Opening Gross.

  Opening gross is a domestic gross revenue of a movie counted from its first week release. Simonoff and Sparrow (2000) argued that opening gross may contribute to a quarter of the total domestic gross. Earlier studies from Simonoff and Sparrow (2000) Latif and Afzal (2016) have included this variable in their studies to investigate the success factor of a movie in the future. The opening gross is a continuous attribute, so we discretized this attribute into 9 buckets as performed by Sharda and Delen (2006).

We summarised the input attributes of the IMDb++ dataset in Table M.2 in Appendix M. The input attributes with the the Native values in the Note column indicates those from the IMDb list files. The ones with the Derived values are the attributes from external data sources (the BoxOfficeMojo, IMDb, and Metacritic websites). The output attribute of the IMDb++ dataset is the same as the ones from the IMDb dataset, as shown in Table 4.16. Meanwhile, table N.2 in Appendix N presents full example of the Hacksaw Ridge the Movie with its values based on the IMDb++ dataset. The values of SumActorsRank, SumActressRank, Budget, AvgCinematographerRank, Competition, AvgComposerRank, AvgCostumeDesignerRank, AvgDirectorRank, SumDistributorRank, AvgEditorRank, Genre, MPAA, AvgProductionCompanyRank, AvgProductionDesignerRank, AvgProducerRank, and AvgWriterRank attributes of the IMDb++ dataset are the same with those values of the IMDb dataset. Also, the value of output attribute of the IMDb++ are the same with the output attribute value of the IMDb dataset.

4.4.3 Generating FilmAffinity, MovieMeter, and RottenTomatoes Datasets

The third type of dataset contains three datasets (FilmAffinity, MovieMeter, and RottenTomatoes). The dataset names indicate the original source of each dataset. The structure of input attributes and an output attribute of the third type of dataset is the same as the second type of dataset. The third type of dataset has 23 input attributes and a single output attribute. The difference between the second and third type datasets is how the ranks of the input attributes are calculated, and how the class of the output attribute is determined. The second type of dataset uses movie ratings from IMDb list files to determine the output class and calculate the ranks of the following input attributes: actor, actress, cinematographer, composer, costume designer, director, distributor, editor, production company, production designer, producer, and writer. However, the third type of dataset uses movie ratings from the following websites: FilmAffinity, MovieMeter, and RottenTomatoes. For instance, the FilmAffinity dataset comprises the ranks calculated from movie ratings of the FilmAffinity website. Likewise, for the output attribute, movie ratings displayed in the FilmAffinity website will be used to determine the output class.

Table M.3 in Appendix M summarises the input attributes of the FilmAffinity dataset. We used movie ratings from the FilmAffinity website to calculate the movie inherent attributes (actor, actress, cinematographer, composer, costume designer, director, distributor, editor, production company, production designer, producer, and writer). If the movie rating for a particular movie is not available in the FilmAffinity website, then, the movie rating will be substitued with the one from the IMDb database. It can be noted that the output attribute always use the movie rating from the FilmAffinity website. The values of GoldenGlobeNominee, GoldenGlobeWon, Metascorre, Opening-Gross, OpeningTheaters, OscarNominee, and OscarWon attributes in the FilmAffinity dataset is the same with those values in the IMDb++ dataset. Table N.3 in Appendix N presents a full example of the Hacksaw Ridge the Movie with its values based on the FilmAffinity dataset.

Table M.4 in Appendix M recaps the input attributes of the MovieMeter dataset. We used movie ratings from the MovieMeter website to calculate the movie inherent attributes. If the movie rating for a particular movie is not available in the MovieMeter website, then, the movie rating will be substitued with the one from the IMDb database. It can be noted that the output attribute always use the movie rating from the MovieMeter website. The values of GoldenGlobeNominee, GoldenGlobeWon, Metascorre, OpeningGross, OpeningTheaters, OscarNominee, and OscarWon attributes in the MovieMeter dataset is the same with those values in the IMDb++ dataset. Meanwhile, table N.4 in Appendix N illustrates a full example of the Hacksaw Ridge the Movie with its values based on the MovieMeter dataset.

Table M.5 in Appendix M sums up the input attributes of the RottenTomatoes dataset. The movie ratings from the RottenTomatoes website are used to calculate the movie inherent attributes. If the movie rating for a particular movie is not available in the RottenTomatoes website, then, the movie rating will be substitued with the one from the IMDb database. It can be highlighted that the output attribute always use the movie rating from the RottenTomatoes website. The values of GoldenGlobeNominee, GoldenGlobeWon, Metascorre, OpeningGross, OpeningTheaters, OscarNominee, and OscarWon attributes in the RottenTomatoes dataset is the same with those values in the IMDb++ dataset. Table N.5 in Appendix N demonstrates a full example of the Hacksaw Ridge the Movie with its values based on the RottenTomatoes dataset.

4.5 Summary

In this chapter, we discourse the data and data sources that relates to addressing movie popularity classification challenges. The first discussion is about what data sources are involved and how they can be gathered. In this thesis, we used downloadable IMDb list files and installed them into a single MySQL database. At the side of the IMDb database, we also used other external data sources (BoxOfficeMojo, FilmAffinity, Metacritic, MovieMeter, and RottenTomatoes websites). We extracted some additional attributes from those data sources and integrated into the IMDb database. We leveraged linked data technologies to assist the integration between the IMDb database and those external data sources. There are two data souces (DBpedia and Wikidata) that can be used to aid the integration process. We used SPARQL queries to extract all movies data from DBpedia. For each movie, we employed a SPARQ query to extract the corresponding Wikidata URL. Based on it, it is possible to get URL links to BoxOfficeMojo, FilmAffinity, IMDb, Metacritic, MovieMeter, and RottenTomatoes websites. The last step of the integration process is to link movie titles in the IMDb database and IMDb website.The second discussion is about the data accessing. In this thesis, there is no process to generate files for data mining processes, but, classficiation algorithms are able directly to access the integration database using SQL or SPARQL queries. D2R Server is used to transform the IMDb database into an RDF format. Therefore, it can be accessed using SPARQL queries. The last discussion is about what attributes are involved to build classification models, and how they are prepared. In this thesis, we use three types of datasets. The first type is the IMDb dataset that is solely based on a single data source (the IMDb list files transformed into the IMDb database). The second type is the IMDb++ dataset that is combined with additional attributes from the BoxOfficeMojo, IMDb, Metacritic websites. The last type comprises three datasets (FilmAffinity, MovieMeter, and RottenTomatoes). In the next Chapter 5, all those datasets will be used by classification algorithms either with default or optimized parameters to build more robust movie popularity classification models.

Chapter 5

Basic Classifiers for Movie Classification Models

In this chapter, we investigate the performance of five basic classifiers (ANN, Decision Tree, kNN, Rule Induction and SVM) when implemented to build classification models on a single data source and multiple data sources. Subsequently, we evaluate their results based on classification accuracy and execution time.

5.1 Classification System Design

In this section, we outline the design of the classification system, shown in Figure 5.1.

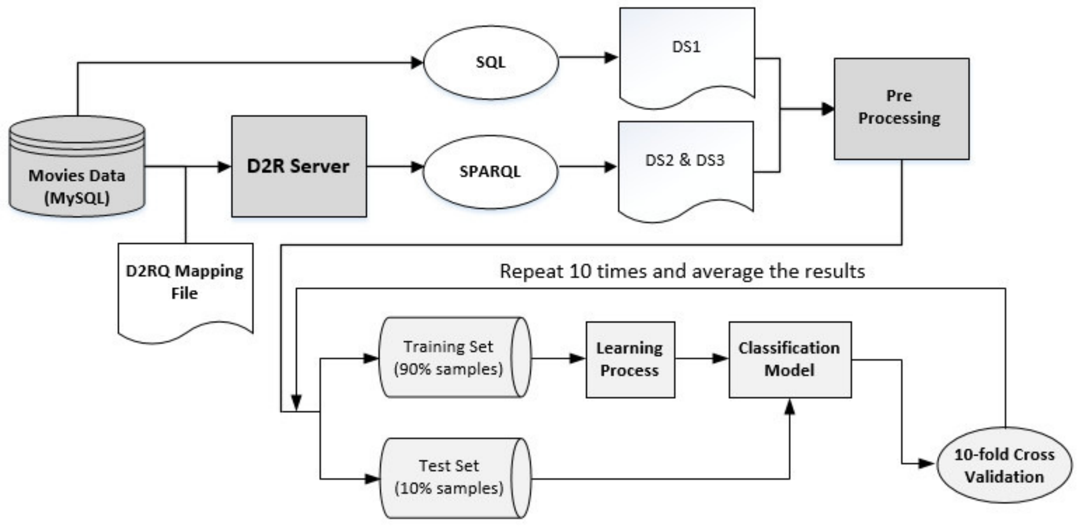


FIGURE 5.1: The Design of Classification System

The details of the design is explained as follows:

1. The system will access data stored in the MySQL database by using SQL queries to generate the first type of dataset DS1.
2. For the purpose of transforming MySQL data into a semantic data format (RDF), we create a D2RQ mapping file and then register it to the D2R server. To generate dataset DS2 and DS3, we access the data by using SPARQL queries via a SPARQL endpoint registered in the D2R server.
3. The system will process DS1, DS2 & DS3 datasets in the pre-processing stage before they will be handled in the data mining process.
4. In the data mining process, we employ a 10-fold Cross Validation technique to evaluate classification models by partitioning each of the datasets (DS1, DS2, & DS3) into a training set to train the model, and a test set to evaluate it.
5. In 10-fold cross-validation, the original datasets (DS1, DS2, & DS3) are randomly partitioned into 10 equal size subsamples. Out of the 10 subsamples, one subsample is kept as the validation data for testing the model, and the further 9 subsamples are leveraged as training data.
6. The cross-validation process will repeat 10 times and be validated in each iteration using the validation data. To produce a single estimation, the 10 results from the folds can then be averaged (or otherwise combined). The advantage of this method is that all the examples data in the datasets (DS1, DS2, & DS3) are used for both training and validation, and each iteration is used for validation exactly once.

5.1.1 Design Implementations

This sub section details the implementation of the classification system as shown in Figure 5.1 into RapidMiner [1], a data science software platform. RapidMiner provides an integrated environment for data preparation, machine learning, deep learning, text mining, and predictive analytics. Moreover, it provides a powerful visual work flow designer for rapidly building a science data work flows.

For the sake of simplicity, we classify workflows into levels. The main workflow, Level 0, will consist of three sub-process operators (Data Collection, Data Preparation, & Data Mining). The sub-process operator introduces a process within another process. Figure 5.2 indicates level 0 of workflow as well as displaying the order of operator execution. Firstly, the Data Collection operator is responsible for extracting data from data sources, and outputting a single example set. Secondly, the output of the previous process feeds into the Data Preparation operator to perform tasks that are specific to enabling data ready for modelling. Lastly, the Data Mining operator uses the data from the previous process to result in a classification model.

[1] https://rapidminer.com/

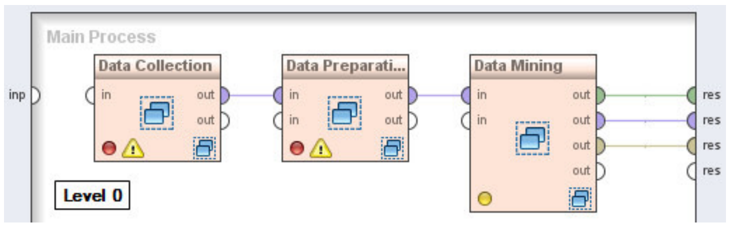


FIGURE 5.2: Level 0 - The Work Flow of Classification System

5.1.1.1 Data Collection

The Data Collection operator contains two types of workflow. Firstly, the workflow presents data extraction from the MySql databae using an SQL query. This workflow outputs an example set to assembly the DS1 dataset. The other workflow is designed to extract semantic data from the D2R Server by using SPARQL queries. This workflow results in a data table for the purpose of generating DS1, DS2 & DS3.

Figure 5.3 shows a workflow for accessing data from the MySQL database. This workflow is comprised of two operators: Read Database and Materialize Data, respectively. The two most important parameters of Read Database operator are connection, used to connect to a specified database, and query, which specifies a statement that is used to select required data from the database. A value for the query parameter is shown in Appendix A Figure A.1. The next operator in workflow 5.3 is Materialise Data. It is intended to create a fresh and clean copy of the data in the memory. The output of this workflow is an example set.

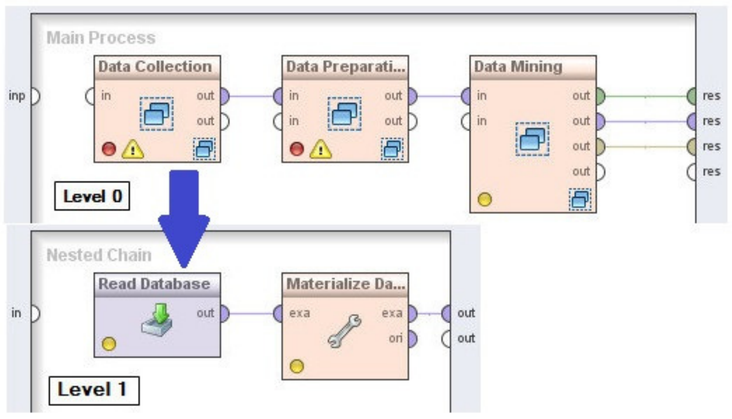


FIGURE 5.3: Data Extraction using a SQL Query Workflow

The workflow to access the semantic data is shown in Figure 5.4, and is more complex compared with the workflow in Figure 5.3, due to having to break down a single large sparql query into several queries to avoid a query time out. The workflow in Figure 5.4 consists of four SPARQL Data Importer operators, three Join operators and a single Materialise Data operator. The SPARQL Data Importer operator has two parameters:

SPARQL connection and SPARQL query. The Join operator has two important parameters: join type (inner, left, outer, right) and key attributes, and is used to join two ExampleSets using one or more attributes of the input ExampleSets as key attributes.

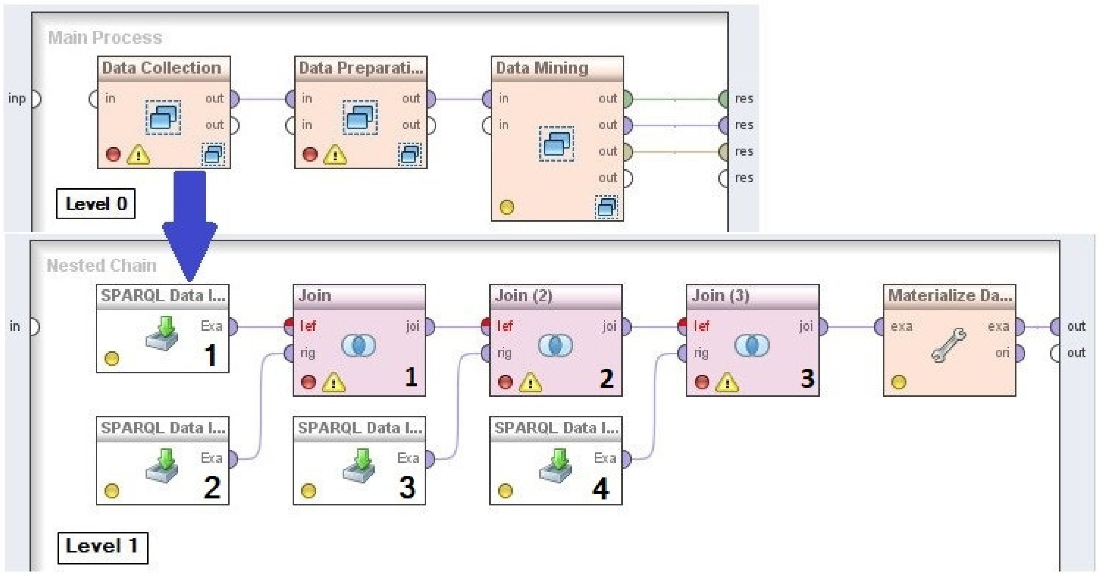


FIGURE 5.4: Data Extraction using SPARQL Queries Workflow

The availability of four operators to import SPARQL data indicates that there will be four different SPARQL queries to assemble the data. The first SPARQL operator collects the data such as: sum_actors_rank, sum_actress_rank, avg_cinematgrs_rank etc. The parameter value will be different for each dataset. For instance, to create an IMDB++ dataset, the SPARQL query of SPARQL operator 1 will be shown in Appendix A Figure A.4. The FilmAffinity, MovieMeter and RottenTomatoes datasets will have SPARQL queries as shown in Appendix A Figures A.6, A.8, and A.10, respectively. The second SPARQL operator is leveraged to retrieve movie budget data. The query is shown in Appendix A Figure A.2 and will be the same for IMDB++, FilmAffinity, MovieMeter and RottenTomatoes datasets.

The first Join operator joins two example sets generated from SPARQL operators 1 & 2. A key attribute to join those example sets is movieid and the join type is left. The left type is chosen because to keep the row number of joining results the same as the number of example sets from the first SPARQL operator. The third SPARQL operator is used to display movie ranks data and its value is different for each dataset. On the other hand, the value for the fourth SPARQL operator is the same for all datasets. This operator serves to list movie genre data. A summary of operators and their parameter values can be shown in Table 5.1.

Table 5.1: Summary of Operators and Their Parameter Values

OPERATOR	#	QUERY	TYPE	KEY ATTRIBUTE LEFT	KEY ATTRIBUTE RIGHT
IMDB++					
SPARQL Data Importer	1	Appendix A Figure A.4			
SPARQL Data Importer	2	Appendix A Figure A.2			
Join	1		left	movieid	movieid
SPARQL Data Importer	3	Appendix A Figure A.5			
Join	2		inner	movieid	movieid
SPARQL Data Importer	4	Appendix A Figure A.3			
Join	3		inner	movieid	movieid
FilmAffinity					
SPARQL Data Importer	1	Appendix A Figure A.6			
SPARQL Data Importer	2	Appendix A Figure A.2			
Join	1		left	movieid	movieid
SPARQL Data Importer	3	Appendix A Figure A.7			
Join	2		inner	movieid	movieid
SPARQL Data Importer	4	Appendix A Figure A.3			
Join	3		inner	movieid	movieid
MovieMeter					
SPARQL Data Importer	1	Appendix A Figure A.8			
SPARQL Data Importer	2	Appendix A Figure A.2			
Join	1		left	movieid	movieid
SPARQL Data Importer	3	Appendix A Figure A.9			
Join	2		inner	movieid	movieid
SPARQL Data Importer	4	Appendix A Figure A.3			
Join	3		inner	movieid	movieid
RottenTomatoes					
SPARQL Data Importer	1	Appendix A Figure A.10			
SPARQL Data Importer	2	Appendix A Figure A.2			
Join	1		left	movieid	movieid
SPARQL Data Importer	3	Appendix A Figure A.11			
Join	2		inner	movieid	movieid
SPARQL Data Importer	4	Appendix A Figure A.3			
Join	3		inner	movieid	movieid

5.1.1.2 Data Preparation

Virtually all of the data mining algorithms would require a single table with cleaned data as input. Preparing the dataset to suit a data mining task is one of steps to get better data mining results. In this research, we design a data preparation workflow as shown in Figure 5.5. The workflow contains some spesific tasks to output a readily example set for data mining processes.

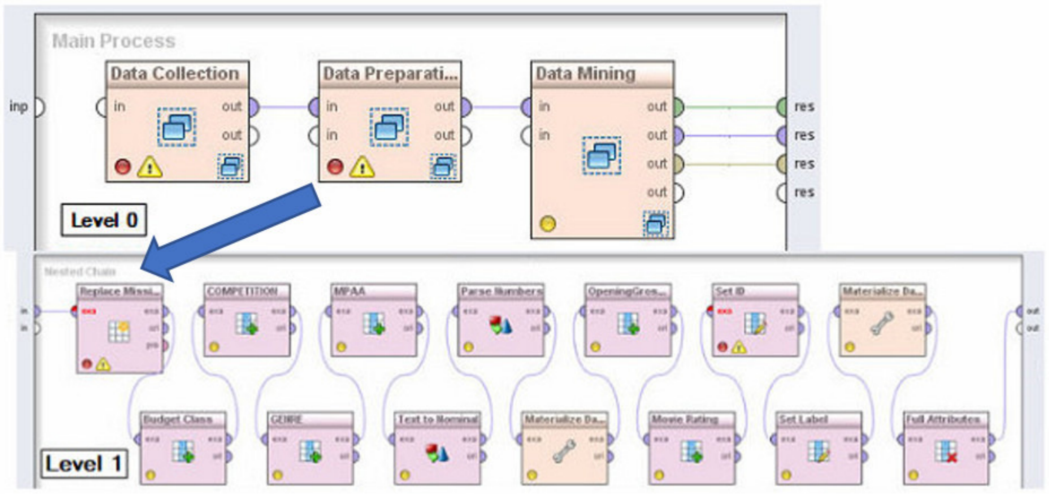


FIGURE 5.5: Data Preparation Workflow

The specific tasks involved in the data preparation workflow is detailed as follows:

1. **Replace Missing Values**. This task replaces missing values in the following selected attributes: (avg_cinematgrs_rank, avg_composers_rank, avg_costdesigners_rank, avg_directors_rank, avg_editors_rank, avg_prodcompanies_rank, avg_proddes_rank, avg_producers_rank, avg_writers_rank, budget, opening_gross, opening_theaters, sum_actors_rank, sum_actress_rank & sum_distributors_rank) and sets the replenishment value with 0.

2. **Discretization of the Budget Attribute**. This task converts the original value of movie budget in a continuous variable into a limited number of nine classess, shown in Figure 4.17. The process of discretization is to construct a new attribute from the budget attribute of the input example set and arbitrary constants using mathematical expressions. The mathematical expression to discretize the budget attribute is shown in Figure 5.6. To implement this task in RapidMiner, we use the Generate Attributes operator.

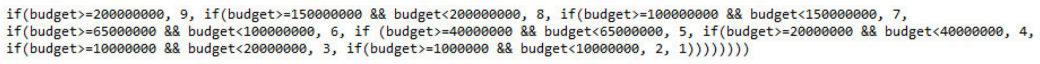
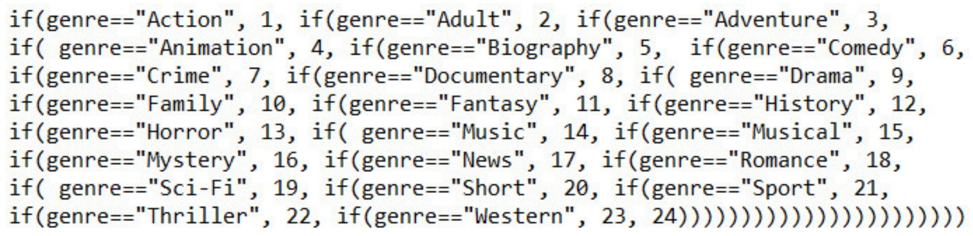

```
if(budget>=200000000, 9, if(budget>=150000000 && budget<200000000, 8, if(budget>=100000000 && budget<150000000, 7,
if(budget>=65000000 && budget<100000000, 6, if (budget>=40000000 && budget<65000000, 5, if(budget>=20000000 && budget<40000000, 4,
if(budget>=10000000 && budget<20000000, 3, if(budget>=1000000 && budget<10000000, 2, 1))))))))
```

FIGURE 5.6: The Mathematical Epression for Discretting the Budget Attribute

3. **Deriving New Competition Attribute**. This task computes polynomial indicator attributes (values 1, 2 &3) to determine whether a competition is associated with high competiton, medium competition or low competiton. The Generate Attributes operator in RapidMiner with if-expression (whose second argument is returned if the first argument, a boolean condition, is true, while the third argument is returned otherwise) is used to accomplish the task. The following if-expression *"if(competition=="High Competition", 3, if(competition=="Medium Competition", 2, 1))"* is used to create the polynomial attribute competition.

4. **Deriving New Genre Attribute**. This task tranforms a string value of the genre attribute into polynomial indicator attributes (values 1 to 24). To accomplish this task, we use the Generate Attributes operator and assign the if-expression as shown in Figure 5.7 to create a new attribute genre.



```
if(genre=="Action", 1, if(genre=="Adult", 2, if(genre=="Adventure", 3,
if( genre=="Animation", 4, if(genre=="Biography", 5,  if(genre=="Comedy", 6,
if(genre=="Crime", 7, if(genre=="Documentary", 8, if( genre=="Drama", 9,
if(genre=="Family", 10, if(genre=="Fantasy", 11, if(genre=="History", 12,
if(genre=="Horror", 13, if( genre=="Music", 14, if(genre=="Musical", 15,
if(genre=="Mystery", 16, if(genre=="News", 17, if(genre=="Romance", 18,
if( genre=="Sci-Fi", 19, if(genre=="Short", 20, if(genre=="Sport", 21,
if(genre=="Thriller", 22, if(genre=="Western", 23, 24)))))))))))))))))))))))
```

Figure 5.7: The if-expression for transforming the Genre Attribute

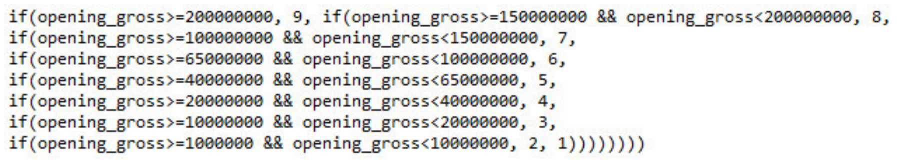
5. **Deriving New MPAA Attribute**. This task constructs a new MPAA attribute by using the Generate Attributes operator with if-expression. The new attribute will have a range of values between 1 and 6, gathered by transforming a string value of the existing MPPA attribute into a polynomial value by implementing the following if-expression *"if(mpaa=="G", 1, if(mpaa=="NC-17", 2, if(mpaa=="PG", 3, if(mpaa=="PG-13", 4, if(mpaa=="R", 5, 6)))))"*.

6. **Text to Nominal**. This task ensures that opening_gross and opening_theaters have corresponding nominal values. To accomplish this task, text attributes of opening_gross and opening_theaters are converted to nominal attributes by using the Text to Nominal attribute.

7. **Parsing Numbers**. This task converts the nominal attributes of opening_gross and opening_theaters to numeric types. We employ the Parse Numbers operators to accomplish this, which will convert the type of opening_gross and opening_theaters attributes and map all values of these attributes to numeric values by parsing the numbers if possible.

8. **Discretization the opening_gross attribute**. This task converts the original value of opening_gross in a continuous variable into a limited number of nine classes. The discretization of those nine classes can be shown in Figure 4.17. The process of discretization is in order to construct a new attribute from the

opening_gross attribute of the input example set and arbitrary constants using mathematical expressions; the expression used to discretize the budget attribute is shown in Figure 5.8. To implement this task in RapidMiner, we use the Generate Attributes operator.



```
if(opening_gross>=200000000, 9, if(opening_gross>=150000000 && opening_gross<200000000, 8,
if(opening_gross>=100000000 && opening_gross<150000000, 7,
if(opening_gross>=65000000 && opening_gross<100000000, 6,
if(opening_gross>=40000000 && opening_gross<65000000, 5,
if(opening_gross>=20000000 && opening_gross<40000000, 4,
if(opening_gross>=10000000 && opening_gross<20000000, 3,
if(opening_gross>=1000000 && opening_gross<10000000, 2, 1))))))))
```

FIGURE 5.8: The Mathematical Expression for Discretizing the Opening Gross Attribute

9. **Grouping the rank attribute**. This task groups a continuous numeric value of the rating attribute into four categories (Excellent, Average, Poor & Terrible) as shown in Table 4.16. We construct a new variable rating from the existing rank attribute using the Generate Attributes operator. This operator converts a continuous variable of the rank attribute into four different classes using the following expression "*if(rank¿=7.5 && rank¡=10, "Excellent", if(rank¿=5 && rank¡7.5, "Average", if(rank¿=2.5 && rank¡5, "Poor", "Terrible")))*".

10. **Setting the ID Attribute Role**. This task changes the role of the movieid to the id attribute. By doing so, it will clarify the identification of the examples of concerned Example Set according to a movie id. The Set Role operator undertakes this task.

11. **Setting the Label Attribute Role**. This task changes the role of the rating attribute to a target attribute. The label role is a special role, serving as a goal attribute for learning operators. Labels identify the examples in any way and they must be predicted for new examples that are not yet characterised in such a manner. We use the Set Role operator to accomplish this task.

12. **Selection Attributes**. This task selects which attributes of an Example Set to keep and to be removed. Often, a need arises for selecting attributes before applying learning operators. We employ the Select Attributes operator to select some relevant attributes for learning processes.

5.2 Performance Measurement

Predictive tasks turn future uncertainties into usable probabilities Taylor (2011). Therefore, it is imperative to select appropriate methods for testing the quality of a predictive model. The matrix used to evaluate the performance of a classification model (classifier) is well-known as a confusion matrix Sokolova and Lapalme (2009). Table 5.2 shows an example of a Confusion Matrix for the case of the binary classification.

Table 5.2: A Confusion Matrix

		Predicted Label	
		Positive	**Negative**
Actual Label	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

The measurement of standard metrics (Accuracy, Precision, Recall) for classification are often based on the confusion matrix Sokolova and Lapalme (2009). In practice, accuracy is the most often used method to evaluate the classification performance Kotsiantis (2007). However, other researchers, such as Davis and Goadrich (2006), argue that simply using accuracy results can be misleading, suggesting using Precision-Recall. A different argument came from Williams et al. (2006). They compute three standard metrics: Accuracy, Precision, and Recall. They call this 'classification accuracy'. Sehgal et al. (2012) agree with Williams et al. (2006), they argued the best performing classification system is that having the high Accuracy, High Precision and High Recall value.

Accuracy is defined as the ability the classifier has in correctly predicting the class label, and its value is counted as the percentage of correctly classified class labels over the total number of class labels. While precision is the percentage of correct positive class label predictions, recall is the percentage of positive class labels that are predicted as positive. Table 5.3 presents the measures for multi-class classification Sokolova and Lapalme (2009). Additionally, F-Measure is a popular standard measure in the Information Retrieval communities, utilised to point out relations between the number of positive class labels and those given by a classifier. According to Moore et al. (2009), F-Measure can be used to detemnine the best performing classification model based on two reasons. Firstly, F-Measure is used to balance the importance of precision and recall. Secondly, F-measure balances the performance of the classes when there is substantial class imbalance in a dataset.

Table 5.3: Multi-Class Classification Measurement Sokolova and Lapalme (2009)

Measure	Formula	Evaluation Focus
Accuracy	$\frac{\sum_{i=1}^{l}=\frac{TP_i+TN_i}{TP_i+FN_i+FP_i+TN_i}}{l}$	The average per-class effectiveness of a classifier
Precision	$\frac{\sum_{i=1}^{l}=TP_i}{\sum_{i=1}^{l}=(TP_i+FN_i)}$	An average per-class agreement of the data class labels with those of a classifiers
Recall	$\frac{\sum_{i=1}^{l}=\frac{TP_i}{TP_i+FN_i}}{l}$	An average per-class effectiveness of a classifier to identify class labels
F-Measure	$2 * \frac{Precision*Recall}{Precision+Recall}$	Relations between data's positive labels and those given by a classifier based on a per-class average

Amongst popular methods to test and evaluate classification models such as holdout, k-fold cross validation and bootstrap, Kohavi (1995) recommended the k-fold cross-validation for the model selection. As the goal is to select an optimal classification model, Arlot and Celisse (2010) reported that the optimal value for k is between 5 and 10. Furthermore, the setting of the k value at less than 10 is computationally feasible Hastie et al. (2009). In this thesis, we utilise the k-fold cross validation to test and evaluate how well the classifiers perform with unseen data. We set k = 10 to compute accuracy, precision, recall and F-Measure. In the 10-fold cross validation, the data set is divided into 10 subsets and repeated in 10 iterations. Over each iteration, one subset is leveraged as the testing data while the nine subsets are utilised as training data. Across all 10 trials, performance statistics are calculated.

5.3 Experimental Results

In this chapter, we conduct experiments to address research question number one by following a method as described in Sub Section 3.3.1. We build classification models by applying five different classifiers (ANN, DT, kNN, RI & SVM) to five different datasets (IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes).

5.3.1 Artificial Neural Network (ANN)

The ANN, typically called Neural Network, leverages a back-propagation algorithm to train a feed-forward neural network in the learning process of predictive models. The ANN can be useful with modelling data sets that are non-linear and highly correlated Lek et al. (1996) Cook et al. (2000). Furthermore, the ANN is quite useful to manage outliers, so that it will not degrade the performance of the model. The following workflow indicates how to build classification models with ANN classifier.

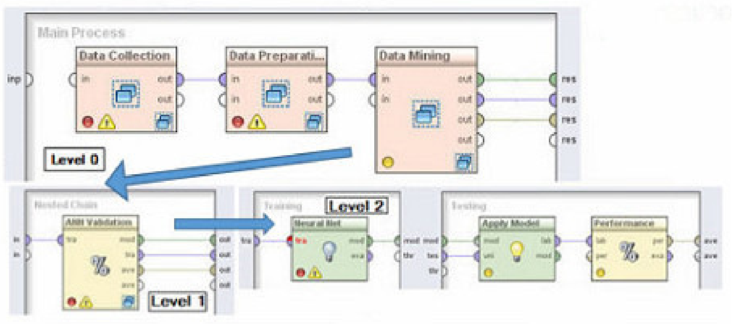


FIGURE 5.9: Building Classification Model with Artifical Neural Network (ANN)

In the RapidMiner, the following parameters are available to change and customise the ANN model:

1. **Hidden Layers:** This parameter is useful to define the structure of the neural network by describing the name and size of all hidden layers. The default value of the hidden layer size is -1. Then, the layer size is calculated by (*number of attribute* + *number of classes*)/2 + 1.

2. **Training Cycles:** This parameter specifies how many times a training cycle of neural network is repeated. The default value is 500. In a neural network, it is necessary to repeat the cycle many times, because in the specified training record, the previous weights are quite different.

3. **Learning Rate:** This parameter determines how much we change the weights at each step. The value range is between 0 and 1. A value closer to 0 means the new weight would be based more on the previous weight, and less on error correction. A value closer to 1 would be mainly based on error correction. The default value is 0.3.

4. **Momentum:** This parameter determines a value for adding a fraction of the previous weight the current weight. Also, it prevents local maxima and seeks to obtain globally optimised results. The default value is 0.2.

5. **Decay:** This parameter determines whether the learning rate should be decreased during learning or not. The default value is false. During training, it desirable that the error becomes closer to zero in the later portion of the record sequence. The appearance of any outliers in the last few records are able to decrease the performance of the model, and this parameter maintains the value of the learning rate closer to zero for the last training record.

6. **Shuffle:** The default value of this parameter is true. If the value is true, then, the data training can be sorted and shuffled to randomise the sequence. The benefit of the sequence in the model is that it becomes possible to cluster all groups of records consisting of nonlinear characteristics in the last segment of the training set.

7. **Normalize:** The ANN operator uses Nodes using a sigmoid transfer functions to activate the normalisation function. It expects inputs to lie in the range of -1 to 1. Any real value of the input should be normalised in an ANN model. The default value of this parameter is true.

8. **Error Epsilon:** The objective of the ANN model should be to minimise the error, but avoid making it zero. To avoid the degradation of performance, we can stop the model building process when the error is less than a threshold called the error epsilon.

Table C.1, C.3, C.5, C.7, and C.9 in Appendix C demonstrates confusion matrixs of ANN classifier when applied to IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Those confusion matrixs were able to visualize missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of ANN classifier when applied to those five datasets as shown in Table 5.4. The table present accuracy, precision, recall, and f-measure value for each dataset.

The result presented in Table C.1 indicates that the model bulit by ANN solely based upon a single data source (IMDb) yielded 86.58%, 81.85%, 74.86% and 78.20% accuracy, precision, recall and F-measure, respectively. On the other hand, the model based on IMDb and external data sources (IMDb++) achieved a model accuracy of 88.32%, precision of 83.57%, recall of 78.26% and F-measure of 80.83%. Therefore, it can be noticed that inclusion of external data sources in the classification model (IMDb++) enriched the classification accuracy by 1.74% compared to IMDB model. Moreover, the other three performance measurement (precision, recall and FMeasure) values are also higher in model IMDb++ compared to model IMDb. Collectively, this suggests that IMDb++ model is able to perform better in predicting movie ratings than IMDb model. This also suggests that information in external data sources has the ability to improve the performance for classifying movie popularities using ANN learner.

Meanwhile, when we used ANN classifier to build classification models based on three different datasets (FilmAffinity, MovieMeter, and RottenTomatoes), there is no model superior to any other. FilmAffinity and MovieMeter models attained (86,81% accuracy, 82.09% precision, 75.32% recall and 78.56% F-measure) and (88,01% accuracy, 80.35% precision, 71.32% recall and 75.77% F-measure), respectively, whilst, RottenTomatoes model achieved 89,77% accuracy, 74.04% precision, 69.11% recall and 71.49% F-measure. Among those three models, FilmAffinity has the best values in precision, recall, and F-Measure, while RottenTomatoes models achieved the best classification accuracy. In addition, MovieMeter model outperforms RottenTomatoes model in the values of precision, recall, and F-Measure.

By following Sehgal et al. (2012) (high accuracy, high precision and high recall) and Moore et al. (2009) (high F-Measure) recommendations, among the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity is the most promising model as it has the highest precision, recall and F-Measure values. Overall, the best performing classification model is IMDb++. Compared with IMDb and the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), IMDb++ has provided the highest precision, recall, and F-Measure values. It was only in the accuracy value that RottenTomates model achieved higher percentage than IMDb++.

5.3.2 Decision Tree

Decision Tree is one of the classifiers that provides the advantage of data representation. Compared with other classifiers, the decision tree data is easily interpreted. The

TABLE 5.4: The Movie Ratings Prediction Model Using ANN

Datasets	#Attr	10-Fold Cross Validation				Diff
		Accuracy	Precision	Recall	F-Measure	
IMDb	16	86.58	81.85	74.86	78.20	
IMDb++	23	88.32	83.57	78.26	80.83	1.74
FilmAffinity	23	86.81	82.09	75.32	78.56	0.23
MovieMeter	23	88.01	80.35	71.32	75.77	1.43
RottenTomatoes	23	89.77	74.04	69.11	71.49	3.19

following workflow 5.10 indicates how to build classification models with Decision Tree classifier.

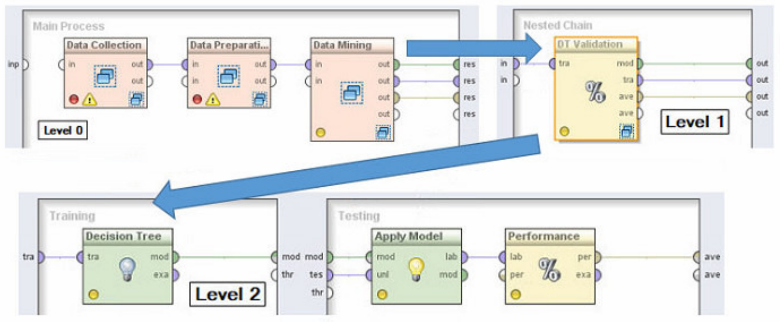


FIGURE 5.10: Building Classification Model with Decision Tree (DT)

To obtain the desired model, Decision Tree has the following parameters to be adjusted and customised:

1. **Criterion:** This parameter determines the criterion to which attributes will be selected for splitting. Four different values are suited to fill the criterion parameter: information gain, gain ration, gini index and accuracy. The default value is the gain ratio.

2. **Minimal size for split:** This parameter determines a limit to split nodes when the number of observations in the node is greater than or equal to the minimal size for split parameter. The default value is 4.

3. **Minimal leaf size:** This parameter determines a limit to perform tree generation when every leaf node subset has at least minimal leaf size number of instance.

4. **Minimal gain:** This parameter is useful to determine the minimal value of gain to split a node. The higher value of this parameters results in fewer splits and a smaller tree. The default value is 0.1

5. **Maximal depth:** This parameter specifies the maximum size of the decision tree. It restricts the tree generation until the tree depth equals the maximal depth. The default value is 20.

6. **Confidence:** This parameter sets the confidence level of the calculation of pessimistic error pruning. Its default value is 0.25.

Confusion matrixs to visualize the performance of DT learner when applied to IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets can be shown in Table E.1, E.3, E.5, E.7, and E.9, respectively in Appendix E. Furthermore, the confusion matrixs can be used to measure the performance of DT classifier (accuracy, precision, recall, and f-measure) for each dataset as indicated in Table 5.5.

The result presented in Table E.1 indicates that the model bulit by DT solely based upon a single data source (IMDb) yielded 84.38%, 81.68%, 72.83% and 77.00% accuracy, precision, recall and F-measure, respectively. On the other hand, the model based on IMDb and external data sources (IMDb++) achieved a model accuracy of 85.02%, precision of 83.05%, recall of 71.34% and F-measure of 76.75%. IMDb model achieved higher recall and F-Measure values compared to IMDB++, while IMDb++ attained higher accuracy and precision values compared to IMDb. In the comparison of confusion matrixs between IMDb (E.1) and IMDb++ (E.3) models, the inclusion of external data sources in IMDb++ model improved the system to predict Average class better than IMDb model. However, information in external data sources decreased the power of IMDb++ model to predict Excellent class compared with IMDb model, while both IMDB and IMDb++ have the same ability to predict Poor and Terrible classes. As a result, the additiion of external data sources in IMDb++ caused DT algorithm having difficulties to balance the importance of precision and recall, therefore, its performance is less better than IMDb model.

Meanwhile, when we used DT classifier to build classification models based on three different datasets (FilmAffinity, MovieMeter, and RottenTomatoes), there is no model superior to any other. FilmAffinity and MovieMeter models attained (82,01% accuracy, 88.21% precision, 54.32% recall and 67.24% F-measure) and (73.10% accuracy, 18.28% precision, 25.00% recall and 21.12% F-measure), respectively, whilst, RottenTomatoes model achieved 87.53% accuracy, 72.92% precision, 58.05% recall and 64.64% F-measure. Among those three models, FilmAffinity has the best values in precision, recall, and F-Measure, while RottenTomatoes models achieved the best classification accuracy. There was an extreme case, in which, the F-Measure value of MovieMeter model was only 21.12%. As shown in the Confusion Matrix E.7, DT learner was able to predict the Average class correctly 100%, however, it predicted the other classes correctly 0%.

By following Sehgal et al. (2012) (high accuracy, high precision and high recall) and Moore et al. (2009) (high F-Measure) recommendations, among the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity is the most promising model as it has the highest precision, and F-Measure values. Overall, the best performing classification model is IMDb. Compared with IMDb++ and the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), IMDb has provided the highest recall,

Table 5.5: The Movie Ratings Prediction Model Using Decision Tree

Datasets	**#Attr**	**10-Fold Cross Validation**				**Diff**
		Accuracy	**Precision**	**Recall**	**F-Measure**	
IMDb	16	84.38	81.68	72.83	77.00	
IMDb++	23	85.02	83.05	71.34	76.75	0.64
FilmAffinity	23	82.01	88.21	54.32	67.24	-2.37
MovieMeter	23	73.10	18.28	25.00	21.12	-11.28
RottenTomatoes	23	87.53	72.92	58.05	64.64	3.15

and F-Measure values. The highest F-Measure value is considered as the most powefull model to balance the performance of the classes when there is substansial class imbalance in a dataset.

Figure 5.11 shows a graphic of the tree that was built on the training data Rotten-Tomatoes. Some of the advantages of using decision trees includes the fact that they are easy to interpret and explain to users with minimal data mining knowledge, and implicitly perform feature selection. According to Figure 5.11, the avg_directors_rank is located on the root node of the decision tree. Subsequently, the avg_producers_rank is a node that is located directly below from the avg_directors_rank node. It is used by the decision tree to split two attributes: avg_cinematgrs_rank and avg_writers_rank. Based on this tree, it can be inferred that avg_directors and _rank avg_producers_rank are the two most influential attributes for the movie popularity. In terms of the second advantage of using decision tree, out of 23 attributes, the tree only involves 13 attributes (avg_directors_rank, avg_producers_rank, avg_cinematgrs_rank, opening_theaters, budget, avg_editors_rank, avg_composers_rank, sum_actors_rank, avg_prodcompanies_rank, sum_distributors_rank, avg_proddes_rank, sum_actress_rank, & avg_writers_rank) in the classification model. Therefore, it can be mentioned that the decision tree implicitly performs feature selection in the modelling process, only involving the most important attributes in the classification process.

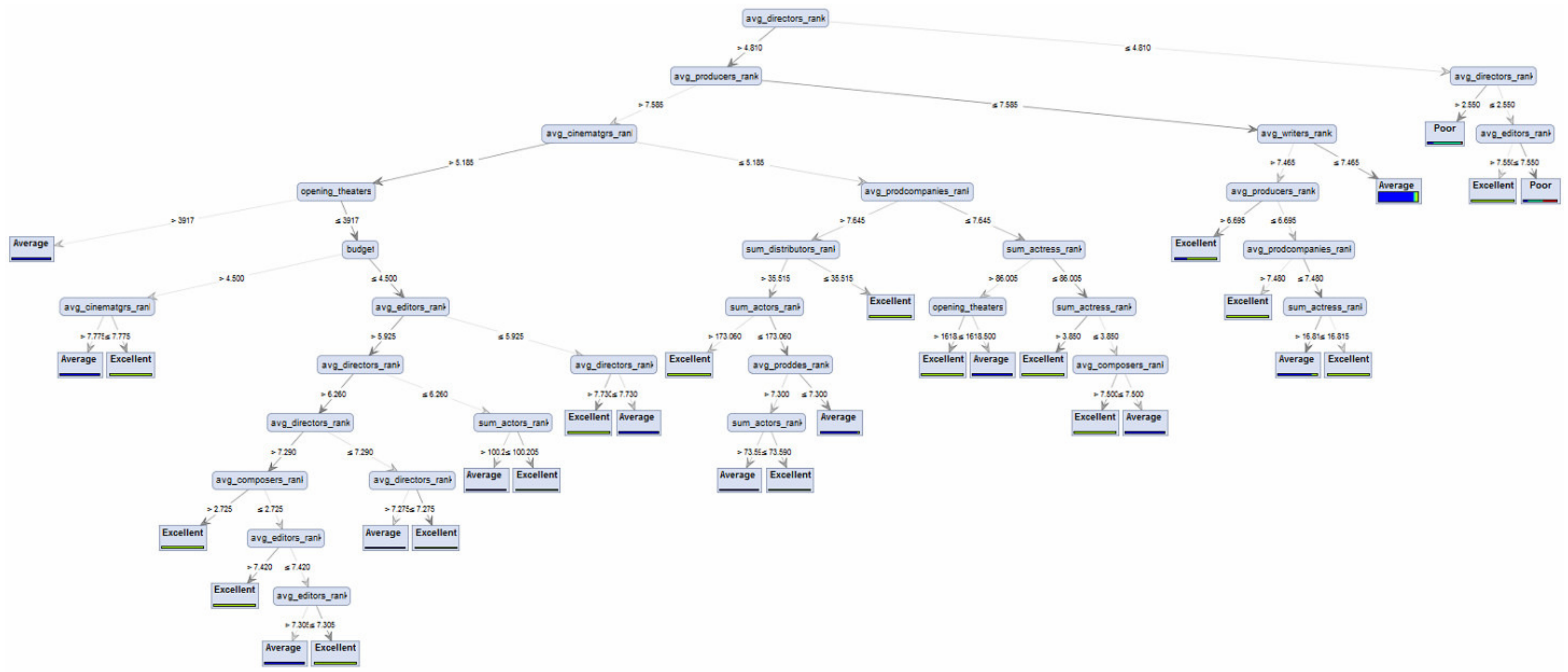


FIGURE 5.11: The Decision Tree model output of the RottenTomatoes data set

5.3.3 k-NN (k-Nearest Neighbour)

k-NN is a non-parametric learning algorithm because it does not build and generalise models from a dataset, and is able to work with categorical and numeric attributes. However, as the core of k-NN is finding the closest training record for a new unlabelled record using the distance computation, k-NN performs better with numeric attributes. All the numeric attributes should be in the same range for a fair comparison between them. Thus, one must apply a pre-processing technique: normalisation. For the normalisation methods, we select range transformation because we want to normalise all numerical values in the specified range between 0.0 and 1.0. This normalisation process is added in the Data Preparation workflow as shown in Figure 5.12.

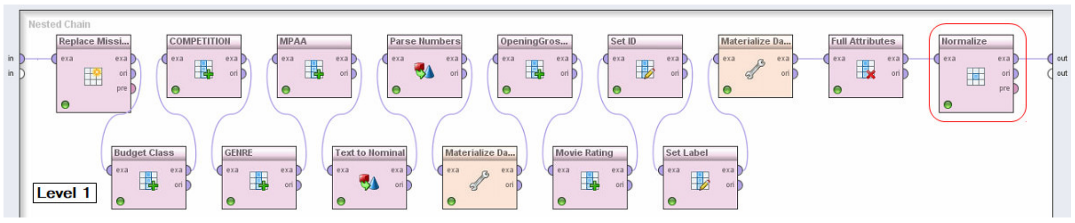


FIGURE 5.12: The Data Preparation with Normalization Process Workflow

To customize the k-NN, we can configure the following parameters:

1. **k:** The k parameter determines the number of training instances that are closest to the unseen instance. Mostly the k value is an odd integer. The default value is 1.

2. **Weighted Vote** This parameter sets a decision to consider the distance value in the predicting process.

3. **Measure Types:** This parameter selects the type of measure to be used for finding the nearest neighbours. The four options are available: mixed measures, nominal measures, numerical measures and Bregman divergences. The selection of this parameter steers the options for the net parameter (Measure). The default value is mixed measures.

4. **Measure:** This parameter selects the actual measure, such as Eucledian distance, Manhattan distance, and so on.

In our experiments, we set all the parameters with the default values. The workflow for building classification models using the k-NN algorithm can be shown in Figure 5.13. The data mining process starts with performing a 10-Fold Cross Validation, represented by Level 1 in Figure 5.13. The cross validation process is intended to estimate the statistical performance of k-NN algorithm to work on unseen data sets. To perform this task, we use the X-Validation operator in RapidMiner. We will perform a 10-fold cross

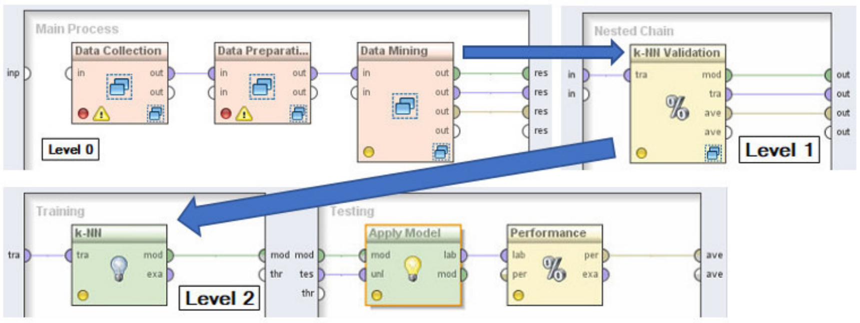


FIGURE 5.13: Data Mining Process for k-NN Classifier Workflow

validation, so that the number of validations parameter will be set at 10. This operator is a nested operator, meaning that has sub-process inside of it. As shown in Level 2 Figure 5.13, the X-Validation operator has two sub-processes: a training sub-process and a testing sub-process. The training one is filled with a learner operator which is the k-NN operator, whilst the testing one is comprised of two operators: apply model and performance, respectively. The apply model operator takes the test data and applies the k-NN model to predict the class type of the test records. The performance operator compares the predicted class with the labelled class for all of the test data.

Confusion matrixs to visualize the performance of k-NN algorithm when applied to IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets can be found in Table G.1, G.3, G.5, G.7, and G.9, respectively in Appendix G. Furthermore, the confusion matrixs can be used to measure the performance of k-NN classifier (accuracy, precision, recall, and f-measure) for each dataset as depicted in Table 5.6.

The result presented in Table 5.6 demonstrates that the model bulit by k-NN solely based upon a single data source (IMDb) yielded 89.19%, 85.36%, 81.66% and 83.47% accuracy, precision, recall and F-measure, respectively. On the other hand, the model based on IMDb and external data sources (IMDb++) achieved a model accuracy of 91.19%, precision of 88.44%, recall of 84.32% and F-measure of 86.33%. Therefore, it can be noticed that inclusion of external data sources in the classification model (IMDb++) enriched the classification accuracy by 2.00% compared to IMDB model. Moreover, the other three performance measurement (precision, recall and FMeasure) values are also higher in model IMDb++ compared to model IMDb. Collectively, this suggests that IMDb++ model is able to perform better in predicting movie ratings than IMDb model. This also suggests that information in external data sources has the ability to improve the performance for classifying movie popularities using k-NN learner.

Meanwhile, when we used k-NN classifier to build classification models based on three different datasets (FilmAffinity, MovieMeter, and RottenTomatoes), there is no model superior to any other. FilmAffinity and MovieMeter models attained (89.69% accuracy, 86.51% precision, 82.55% recall and 84.49% F-measure) and (90.64% accuracy, 86.42% precision, 80.13% recall and 83.16% F-measure), respectively, whilst, RottenTomatoes

Table 5.6: The Movie Ratings Prediction Model Using kNN

Datasets	#Attr	10-Fold Cross Validation				Diff
		Accuracy	Precision	Recall	F-Measure	
IMDb	16	89.19	85.36	81.66	83.47	
IMDb++	23	91.19	88.44	84.32	86.33	2.00
FilmAffinity	23	89.69	86.51	82.55	84.49	0.50
MovieMeter	23	90.64	86.42	80.13	83.16	1.45
RottenTomatoes	23	91.73	84.87	82.81	83.83	2.54

model achieved 91.73% accuracy, 84.87% precision, 82.81% recall and 83.83% F-measure. Among those three models, FilmAffinity has the best values in precision and F-Measure, while RottenTomatoes models achieved the best classification accuracy and recall. In addition, MovieMeter model outperforms RottenTomatoes model in the value of precision and its accurary value is higher than FilmAffinity.

By following Sehgal et al. (2012) (high accuracy, high precision and high recall) and Moore et al. (2009) (high F-Measure) recommendations, among the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity is the most promising model as it has the highest precision and F-Measure values. Overall, the best performing classification model is IMDb++. Compared with IMDb and the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), IMDb++ has provided the highest precision, recall, and F-Measure values. It was only in the accuracy value that RottenTomates model achieved higher percentage than IMDb++.

5.3.4 Rule Induction

Rule Induction is a classifier which concludes if-then rules from a data set. Those rules illustrate an inherent relationship between input and output variables in the data set. Rule Induction is not only used to classify unknown data, but also describes patterns in the data, easily understood by general users through the simple form of if-then rules. There are two methods to extract rules from the data set, the being direct method. This derives a rule set by leveraging the relationship between the attribute and class label in the dataset. The second method is indirect method, which obtains a rule set by converting a previously built decision tree model.

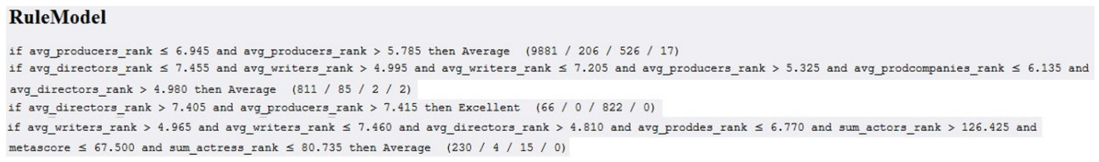
Rule Induction is capable not only for a predictive model but also a descriptive model. The description is in the simple form of if-then rules as shown in Figure 5.14, which can be easily understood by non-expert users. In the example of Figure 5.14, a rule set consists of three rules. Each individual rule r_i is called a *disjunct* or classification rule. A rule set can be represented as

$$R = \{r_1 \bigcap r_2 \bigcap r_3 \bigcap ...r_k\}$$

where k is the number of classification rule in a rule set. Individual classification rules can be described as

$$r_i = (antecedent\ or\ condition)\ \ then\ \ (consequent)$$

For instance, rule r_3: if avg_directors_rank $>$ 7.405 and avg_producers_rank $>$ 7.415 then Excellent. The *antecedent* or *condition* from the above example rule are avg_directors_rank $>$ 7.405 and avg_producers_rank $>$ 7.415. Whilst the consequent is Excellent. In Figure 5.14, there are parentheses next to each classification rule, demonstrating that the class distribution of the rule is extracted from the training dataset. For instance, the parentheses next to r_3 displays (66 / 0 / 822 /0). This means that 66 represents the Average class, while 822 is for the Excellent class, and the Poor and Terrible class is 0.


RuleModel

```
if avg_producers_rank ≤ 6.945 and avg_producers_rank > 5.785 then Average  (9881 / 206 / 526 / 17)
if avg_directors_rank ≤ 7.455 and avg_writers_rank > 4.995 and avg_writers_rank ≤ 7.205 and avg_producers_rank > 5.325 and avg_prodcompanies_rank ≤ 6.135 and
avg_directors_rank > 4.980 then Average  (811 / 85 / 2 / 2)
if avg_directors_rank > 7.405 and avg_producers_rank > 7.415 then Excellent  (66 / 0 / 822 / 0)
if avg_writers_rank > 4.965 and avg_writers_rank ≤ 7.460 and avg_directors_rank > 4.810 and avg_proddes_rank ≤ 6.770 and sum_actors_rank > 126.425 and
metascore ≤ 67.500 and sum_actress_rank ≤ 80.735 then Average  (230 / 4 / 15 / 0)
```

FIGURE 5.14: a Rule Set Output of the RottenTomatoes Data Set

Figure 5.15 shows the workflow for Rule Induction process. At the level 1, we use the X-Validation operator to implement a 10-Fold Cross Validation. There are four operators available in RapidMiner to generate rules from a data set: Rule Induction, Single Rule Induction (Attribute) and Tree To Rule. In this research, we select the direct method to generate rules. Therefore, we applied the Rule Induction operator to Level 2 of the workflow 5.15. This operator accepts an example set at the input for training data and supplies the rule set as the model output. There are four parameters available for users who want to customise the Rule Induction operator for desired modeling behaviour.

1. **Criterion:** The criterion parameter is useful to determine attribute selection and numerical splitting. This parameter provides two options: information gain and accuracy. The default value is information gain.

2. **Sample ratio:** This parameter determines the proportion of training data for growing, whilst the rest is for pruning. The default value is 0.9.

3. **Pureness:** This parameter determines the pureness level. The default value is 0.9.

4. **Minimal prune benefit:** This parameter specifies the minimal percentage of benefit so that the training data can be pruned. The default value is 0.25.

The model output from the RI operator is connected to the Apply Model operator, in order to apply the developed rule set against the testing data set. The labelled dataset generated from the Apply Model operator is connected to the Performance operator for classification. This is intended to create the performance vector of Rule Induction classifier.

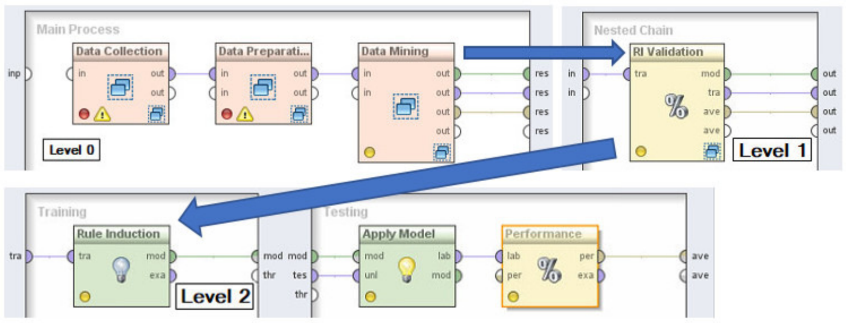


FIGURE 5.15: Data Mining Process for Rule Induction Classifier Workflow

Confusion matrixs to visualize the performance of RI classifier when applied to IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets can be found in Table H.1, H.3, H.5, H.7, and H.9, respectively in Appendix H. Furthermore, the confusion matrixs can be used to measure the performance of RI classifier (accuracy, precision, recall, and f-measure) for each dataset as depicted in Table 5.7.

The result presented in Table 5.7 indicates that the model bulit by RI solely based upon a single data source (IMDb) obtained 89.62%, 88.88%, 76.15% and 82.02% accuracy, precision, recall and F-measure, respectively. On the other hand, the model based on IMDb and external data sources (IMDb++) achieved a model accuracy of 89.79%, precision of 89.83%, recall of 76.71% and F-measure of 82.75%. Therefore, it can be noticed that inclusion of external data sources in the classification model (IMDb++) rose the classification accuracy by 0.17% compared to IMDB model. Moreover, all performance measurements (accuracy, precision, recall and F-Measure) values are also higher in model IMDb++ compared to model IMDb. Collectively, this suggests that IMDb++ model is able to perform better in predicting movie ratings than IMDb model. This also suggests that information in external data sources has the ability to improve the performance for classifying movie popularities using RI learner.

Meanwhile, when we used RI classifier to build classification models based on three different datasets (FilmAffinity, MovieMeter, and RottenTomatoes), there is no model superior to any other. FilmAffinity and MovieMeter models attained (90.56% accuracy, 89.70% precision, 77.69% recall and 83.26% F-measure) and (91.05% accuracy, 88.30% precision, 65.09% recall and 74.94% F-measure), respectively, whilst, RottenTomatoes model achieved 91.42% accuracy, 84.64% precision, 71.99% recall and 77.81% F-measure. Among those three models, FilmAffinity has the best values in precision and F-Measure, while RottenTomatoes models achieved the best classification accuracy and recall. In addition, MovieMeter model outperforms RottenTomatoes model in the value of precision and its accurary value is higher than FilmAffinity.

TABLE 5.7: The Movie Ratings Prediction Model Using Rule Induction

Datasets	#Attr	10-Fold Cross Validation				Diff
		Accuracy	Precision	Recall	F-Measure	
IMDb	16	89.62	88.88	76.15	82.02	
IMDb++	23	89.79	89.83	76.71	82.75	0.17
FilmAffinity	23	90.56	89.70	77.69	83.26	0.94
MovieMeter	23	91.05	88.30	65.09	74.94	1.43
RottenTomatoes	23	91.42	84.64	71.99	77.81	1.80

By following Sehgal et al. (2012) (high accuracy, high precision and high recall) and Moore et al. (2009) (high F-Measure) recommendations, among the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity is the most promising model as it has the highest precision, recall and F-Measure values. Overall, the best performing classification model is FilmAffinity. Compared with IMDb, IMDb++ and the two other datasets (MovieMeter, and RottenTomatoes), FilmAffinity has provided the highest recall, and F-Measure values. It was only in the accuracy value that MovieMeter and RottenTomates models achieved higher percentage than IMDb++, while its precision value is lower than IMDb++.

5.3.5 Support Vector Machines (SVM)

SVM, in simple terms, is a classification method which works on linearly separable data by maximising the margin between the separating hyperplanes and the data. To maximise the margin, SVM solves a constrained optimisation problem. From a more complex view, SVM is invoked to work with linearly non-separable data, dealing with this by transforming the data to a higher dimensional space using kernel functions. A kernel is used to transform nonlinear spaces into linear ones.

SVM can be implemented on numeric and categorical features, though it is more effective in working with numeric ones. Normalisation of all numeric features assist the optimisation part of training. Therefore, we modify our existing data preparation workflow by adding a normalisation process on it as shown in Figure 5.12. To perform a normalisation task, we select the range transformation method and specify the values for min and max parameters with 0.0 and 1.0, respectively.

Figure 5.16 shows a workflow with implemented RapidMiner to build classification models using SVM classifier. Level 1 of the workflow represents an implementation of 10-Fold Cross Validation using the X-Validation operator to give probability estimates as confidences. In level 2, or the nested layer from the X-Validation control, we add an SVM operator and Apply Model and Performance for Classification operators in the training panel.

There are four kernels types supported in RapidMiner: linear, polynomial, rbf and sigmoid. Our initial experiment using those kernels showed that the polynomial kernel gives

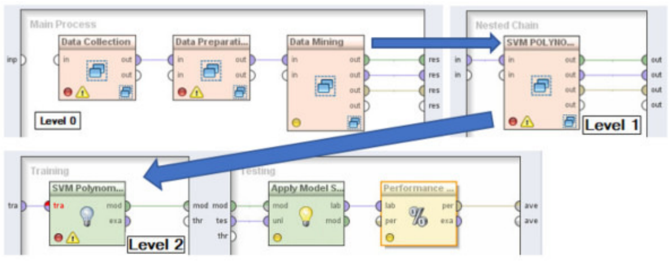


FIGURE 5.16: Data Mining Process for SVM Classifier Workflow

more satisfactory results in terms of accuracy. Therefore, we selected the polynomial kernel for this experiment. To work with the polynomial kernel, there are three parameters available in the SVM operator to customise for the desired modelling behaviour.

1. **Degree:** This parameter is used to specify the degree for a polynomial kernel function. The default value is 3.
2. **Gamma:** The value of gamma may strongly affect the performance and accuracy of SVM model. Using cross-validation may help in finding the optimal value of gamma. The default value is 0.0.
3. **C:** The value of C specifies the penalty parameter of the error classification. The default value is 0.0.

In this experiment, we leave all those parameters with default values.

Confusion matrixs to visualize the performance of SVM classifier with polynomial kernel when applied to IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets can be found in Table J.2, J.4, J.6, J.8, and J.10, respectively in Appendix J. Furthermore, the confusion matrixs can be used to measure the performance of SVM polynomial classifier (accuracy, precision, recall, and f-measure) for each dataset as depicted in Table 5.8.

The result presented in Table 5.8 indicates that the model bulit by SVM polynomial solely based upon a single data source (IMDb) obtained 84.23%, 81.35%, 65.41% and 72.52% accuracy, precision, recall and F-measure, respectively. On the other hand, the model based on IMDb and external data sources (IMDb++) achieved a model accuracy of 85.26%, precision of 81.50%, recall of 68.81% and F-measure of 74.62%. Therefore, it can be noticed that inclusion of external data sources in the classification model (IMDb++) rose the classification accuracy by 1.03% compared to IMDB model. Moreover, all others performance measurement (accuracy, precision, recall and F-Measure) values are also higher in model IMDb++ compared to model IMDb. Collectively, this suggests that IMDb++ model is able to perform better in predicting movie ratings than

TABLE 5.8: The Movie Ratings Prediction Model Using SVM Polynomial

Datasets	#Attr	10-Fold Cross Validation				Diff
		Accuracy	Precision	Recall	F-Measure	
IMDb	16	84.23	81.35	65.41	72.52	
IMDb++	23	85.26	81.50	68.81	74.62	1.03
FilmAffinity	23	84.17	82.47	64.94	72.66	-0.06
MovieMeter	23	84.48	65.45	40.96	50.38	0.25
RottenTomatoes	23	86.82	61.52	50.13	55.25	2.59

IMDb model. This also suggests that information in external data sources has the ability to improve the performance for classifying movie popularities using SVM polynomial learner.

Meanwhile, when we used SVM polynomial classifier to build classification models based on three different datasets (FilmAffinity, MovieMeter, and RottenTomatoes), there is no model superior to any other. FilmAffinity and MovieMeter models attained (84.17% accuracy, 82.47% precision, 64.94% recall and 72.66% F-measure) and (84.48% accuracy, 65.45% precision, 40.96% recall and 50.38% F-measure), respectively, whilst, RottenTomatoes model achieved 86.82% accuracy, 61.52% precision, 50.13% recall and 55.25% F-measure. Among those three models, FilmAffinity leads the best values in precision, recall, and F-Measure, while RottenTomatoes models achieved the best classification accuracy. In addition, MovieMeter model outperforms RottenTomatoes model in the value of precision and its accurary value is higher than FilmAffinity.

By following Sehgal et al. (2012) (high accuracy, high precision and high recall) and Moore et al. (2009) (high F-Measure) recommendations, among the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity is the best performing model as it has the highest precision, recall, and F-Measure values. Overall, the best performing classification model is IMDb++. Compared with IMDb and the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), IMDb++ has provided the highest precision, recall, and F-Measure values. It was only in the accuracy value that MovieMeter and RottenTomates models achieved higher percentage than IMDb++.

5.4 Parameter Optimization to ANN, DT, RI and SVM Parameters

As aforementioned in Chapter 5 on Decision Tree and SVM, cases can be noted in which the F-Measure results dropped to less than 56%. The extreme case was found when DT classifier applied to the MovieMeter dataset yielded the F-Measure value of 21.12%. Additionally, the SVM learner resulted in the poor value of F-Measure by 55.25% when applied to the RottenTomatoes dataset. In these examples, we chose to simply use default parameter values to build the DT and SVM models; the classification performance is usually an indicator as to whether we chose the right parameter combination for our classifiers. Keerthi and Lin (2003) describe that inappropriate parameter settings lead to poor learning results. In this research, we employed a grid search to discover and choose the best combination of ANN, DT, RI and SVM parameters.

Grid search systematically enumerates through a manually defined subset of the hyper-parameters space of a classification algorithm. The hyper-parameter values are set before the commencement of the classification process, and are defined using lower bound, upper bound and number of steps. To achieve a set of optimal hyper-parameter for a classifier, the grid search requires a performance metric to serve as its guide, such as k-fold cross validation. In this research, we set the k value with 10.

To perform hyper-parameter optimisation using Grid Search in RapidMiner, we designed a workflow, shown in Figure 5.17 and used the *Optimize Parameter (Grid)* operator. This operator selects hyper-parameters for a specific learning algorithm, and configures them. Moreover, the operator is nested, which has a sub-process in it. The sub-process will be executed for all combinations of selected values of the hyper-parameters. The *X-Validation* operator is applied to the sub-process of the *Optimise Parameter (Grid)* operator. The *X-Validation* operator is used to measure the statistical performance of the learning algorithm with all selected hyper-parameters, and is a nested operator with two inner sub-processes: training and testing. In training, we will insert one of the following learners (ANN, DT, RI and SVM). Meanwhile, in the testing sub-process, we add two operators: *Apply Model* and *Performance*, respectively.

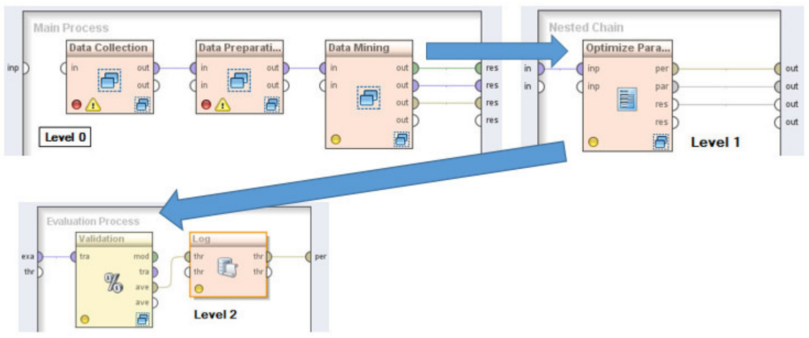


FIGURE 5.17: The Workflow of Parameter Optimization using Grid Search

5.4.1 Parameter Optimisation to ANN Parameters

We selected three hyper-parameters of ANN (training cycles, learning rates, and momentum) needed to be tuned. They have been selected to increace performances of ANN when applied to unseen data. The selected parameters are continuous, and we configured them with the default values – table 5.9 details the configuration of ANN hyper-parameters. The steps field specifies the number of combinations generated from the specified range. The scale field determines the pattern of combination values. In

TABLE 5.9: Hyper-parameters Range for ANN

Parameters	**Min**	**Max**	**Steps**	**Scale**	**Combinations**	**Number of Combinations**
Training Cycles	1.0	500.0	10	Linear	1, 51, 101, 151, 201, 251, 300, 350, 400, 450, 500	11
Learning Rate	0.0	1.0	10	Linear	0, 0.100, 0.200, 0.300, 0.400, 0.500, 0.600, 0.700, 0.800, 0.900, 1	11
Momentum	0.0	1.0	10	Linear	0, 0.100, 0.200, 0.300, 0.400, 0.500, 0.600, 0.700, 0.800, 0.900, 1	11
Total Number of Combinations						1331

the grid search, the number of parameters and the number of steps resulted in many combinations – 1331 (11 x 11 x 11) iterations were performed to get a set of optimal hyper-parameters for a ANN classifier.

To tune ANN parameters using a Grid Search algorithm, we needed to setup an inner process of the *X-Validation* operator in Workflow 5.17. The inner process of the X-Validation operator contains training and testing sections, as shown in Figure 5.18. The training section sets a classification algorithm for classifying a variable set, and a *Neural Net* operator was added to this section. Testing examines the performance of the learned model in the cross-validation manner, consisting of two operators: *Apply Model* and *Performance*. The *Apply Model* operator applies the ANN classification model to the testing dataset, and delivers the resultant labelled dataset as the output. This is used by the *Performance* operator to evaluate the model and to generate a performance vector containing information about various performance criteria.

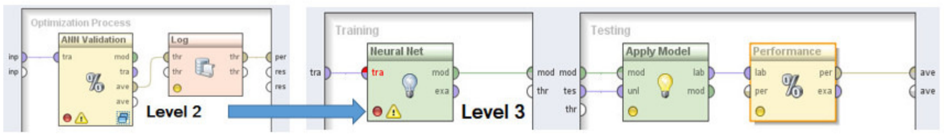


FIGURE 5.18: The Workflow for Optimizing ANN Parameters using Grid Search Algorithm

Table 5.10 indicates the results of ANN parameters optimisation using Grid Search when applied to full-attributes datasets (IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes) . There are no various values for the Learning Rate hyper-parameter. Its best parameter value is 0.1. In contrast, the values of training cycles and momentum parameters vary.

Confusion Matrixs C.2, C.4, C.6, C.8, and C.10, respectively in Appendix C visualizes the performances of Grid Search to optimize ANN parameters to full-attributes of the datasets. Furthermore, the confusion matrixs can be used to measure results of ANN parameter optimisation using Grid Search by using following performance measurements (accuracy, precision, recall, and f-measure) for each dataset as depicted in Table 5.10.

The result presented in Table 5.10 indicates that the use of Grid Search to optimize ANN parameters to build a classification model solely based upon a single data source (IMDb) obtained 86.60%, 82.06%, 75.40% and 78.59% accuracy, precision, recall and F-measure, respectively. On the other hand, the model built by ANN with optimized parameters based on IMDb and external data sources (IMDb++) achieved a model accuracy of 88.57%, precision of 84.86%, recall of 78.31% and F-measure of 81.45%. Therefore, it can be noticed that inclusion of external data sources in the classification model (IMDb++) rose the classification accuracy and F-Measure values by 1.97% and 2.86%, respectively, compared to IMDB model. Moreover, all others performance measurement (precision and recal) values are also higher in model IMDb++ compared to model IMDb. Collectively, this suggests that IMDb++ model is able to perform better in predicting movie ratings than IMDb model. This also suggests that information in external data sources has the ability to improve the performance for classifying movie popularities by leveraging the use of Grid Search to optimize ANN.

Meanwhile, when we used ANN with optimized parameters to build classification models based on three different datasets (FilmAffinity, MovieMeter, and RottenTomatoes), there is no model superior to any others. FilmAffinity and MovieMeter models attained (86.70% accuracy, 82.71% precision, 73.65% recall and 77.92% F-measure) and (88.10% accuracy, 80.40% precision, 69.46% recall and 74.53% F-measure), respectively, whilst, RottenTomatoes model achieved 90.14% accuracy, 75.55% precision, 67.76% recall and 71.44% F-measure. Among those three models, FilmAffinity leads the best values in precision, recall, and F-Measure, while RottenTomatoes models achieved the best classification accuracy.

By following Sehgal et al. (2012) (high accuracy, high precision and high recall) and Moore et al. (2009) (high F-Measure) recommendations, among the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity is the best performing model as it has the highest precision, recall, and F-Measure values. Overall, the best performing classification model is IMDb++. Compared with IMDb and the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), IMDb++ has provided the highest precision, recall, and F-Measure values.

5.4.2 Parameter Optimisation to Decision Tree Parameters

The Decision Tree learning algorithm may have the following hyper-parameters to optimise: Criterion, Minimal size for split, Minimal leaf size, Minimal gain, Maximal depth, and Confidence. We selected four hyper-parameters needed to be tuned for good performance on unseen data: criterion, minimal size for split, minimal leaf size, and maximal depth. The Criterion parameter has four possible combinations: accuracy, gain_ratio, gini_index, and information_gain. We only selected three combinations, eliminating 'accuracy'. The selected parameters are continuous, and we configured them with the

TABLE 5.10: The results of ANN Parameter Optimization using Grid Search (the Best Params field ordered by Training Cycles, Learning Rate and Momentum)

Datasets	**# Attr**	**10-Fold Cross Validation**				**Best Params**
		Accuracy	**Precision**	**Recall**	**F-Measure**	
IMDb	16	86.60	82.06	75.40	78.59	450, 0.1, & 0.6
IMDb++	23	88.57	84.86	78.31	81.45	500, 0.1, & 0.3
FilmAffinity	23	87.23	83.18	75.09	78.93	500, 0.1, & 0.0
MovieMeter	23	88.50	79.94	70.05	74.67	500, 0.1, & 0.3
RottenTomatoes	23	90.14	75.55	67.76	71.44	500, 0.1, & 0.5

TABLE 5.11: Hyper-parameters Range for Decision Tree

Parameters	**Min**	**Max**	**Steps**	**Scale**	**Combinations**	**Number of Combinations**
Criterion (C)					gain_ration, information_gain, gini_index	3
Minimal size for split (MSS)	1.0	100.0	10	Linear	1, 11, 21, 31, 41, 51, 60, 70, 80, 90, 100	11
Minimal leaf size (MLS)	1.0	100.0	10	Linear	1, 11, 21, 31, 41, 51, 60, 70, 80, 90, 100	11
Maximal depth (MD)	-1.0	100.0	10	Linear	-1, 9, 19, 29, 39, 50, 60, 70, 80, 90, 100	11
Total Number of Combinations						3993

default values – table 5.11 details the configuration of decision tree hyper-parameters. The steps field specifies the number of combinations generated from the specified range. The scale field determines the pattern of combination values. In the grid search, the number of parameters and the number of steps resulted in many combinations – 3993 (3 x 11 x 11 x 11) iterations were performed to get a set of optimal hyper-parameters for a Decision Tree classifier.

To tune DT parameters using a Grid Search algorithm, we needed to setup an inner process of the *X-Validation* operator in Workflow 5.17. The inner process of the X-Validation operator contains training and testing sections, as shown in Figure 5.19. The training section sets a classification algorithm for classifying a variable set, and a *Decision Tree* operator was added to this section. Testing examines the performance of the learned model in the cross-validation manner, consisting of two operators: *Apply Model* and *Performance*. The *Apply Model* operator applies the DT classification model to the testing dataset, and delivers the resultant labelled dataset as the output. This is used by the *Performance* operator to evaluate the model and to generate a performance vector containing information about various performance criteria.

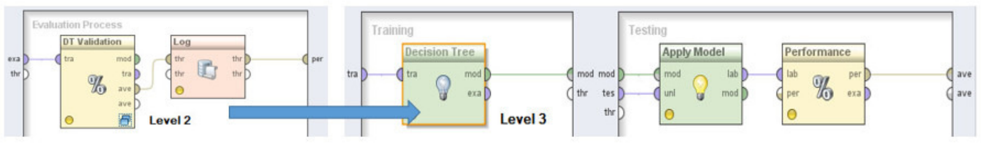


FIGURE 5.19: The Workflow for Optimizing DT Parameters using Grid Search Algorithm

Table 5.12 indicates the results of Decision Tree parameter optimisation using Grid Search when applied to full-attributes datasets (IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes) . There are no various values for the following hyper-parameters: criterion, minimal size for split, and minimal leaf size. Their best parameter values are gini_index, 1, and 1, respectively. In contrast, the values of maximal depth parameter vary.

Confusion Matrixs E.2, E.4, E.6, E.8, and E.10, respectively in Appendix E visualizes the performances of Grid Search to optimize Decision Tree parameters to full-attributes of the datasets. Furthermore, the confusion matrixs can be used to measure results of Decision Tree parameter optimisation using Grid Search by using following performance measurements (accuracy, precision, recall, and f-measure) for each dataset as depicted in Table 5.12.

The result presented in Table 5.12 indicates that the use of Grid Search to optimize Decision Tree parameters to build a classification model solely based upon a single data source (IMDb) obtained 98.16%, 96.70%, 96.58% and 96.64% accuracy, precision, recall and F-measure, respectively. On the other hand, the model built by Decision Tree optimized parameters based on IMDb and external data sources (IMDb++) achieved a model accuracy of 98.24%, precision of 96.79%, recall of 96.72% and F-measure of 96.75%. Therefore, it can be noticed that inclusion of external data sources in the classification model (IMDb++) rose 0.08% accuracy and 0.11% F-Measure values of IMDB model. Moreover, all others performance measurement (precision and recall) values are also higher in model IMDb++ compared to model IMDb. Collectively, this suggests that IMDb++ model is able to perform better in predicting movie ratings than IMDb model. This also suggests that information in external data sources has the ability to improve the performance for classifying movie popularities by leveraging the use of Grid Search to optimize Decision Tree.

Meanwhile, when we used Decision Tree with optimized parameters to build classification models based on three different datasets (FilmAffinity, MovieMeter, and RottenTomatoes), there is no model superior to any other. FilmAffinity and MovieMeter models attained (98.02% accuracy, 96.78% precision, 96.15% recall and 96.47% F-measure) and (97.88% accuracy, 94.98% precision, 93.65% recall and 94.31% F-measure), respectively, whilst, RottenTomatoes model achieved 98.04% accuracy, 95.81% precision, 95.19% recall and 95.50% F-measure. Among those three models, FilmAffinity leads the best

Table 5.12: The results of Decision Tree Parameter Optimization using Grid Search (the Best Params field ordered by C, MSS, MLS and MD)

Datasets	# Attr	10-Fold Cross Validation				Best Params
		Accuracy	Precision	Recall	F-Measure	
IMDb	16	98.16	96.70	96.58	96.64	gini_index, 1, 1, & 29
IMDb++	23	98.24	96.79	96.72	96.75	gini_index, 1, 1, & 29
FilmAffinity	23	98.02	96.78	96.15	96.47	gini_index, 1, 1, & 60
MovieMeter	23	97.88	94.98	93.65	94.31	gini_index, 1, 1, & 39
RottenTomatoes	23	98.04	95.81	95.19	95.50	gini_index, 1, 1, & 29

values in precision, recall, and F-Measure, while RottenTomatoes models achieved the best classification accuracy.

By following Sehgal et al. (2012) (high accuracy, high precision and high recall) and Moore et al. (2009) (high F-Measure) recommendations, among the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity is the best performing model as it has the highest precision, recall, and F-Measure values. Overall, the best performing classification model is IMDb++. Compared with IMDb and the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), IMDb++ has provided the highest accuracy, precision, recall, and F-Measure values.

5.4.3 Parameter Optimisation to Rule Induction Parameters

We optimized the following hyper-parameters: Sample Ratio, Pureness, and Minimal Prune Benefit to improve the performance of RI learner to predict unseen data. The selected parameters are continuous, and we configured them with the default values – table 5.13 details the configuration of Rule Induction hyper-parameters. The steps field specifies the number of combinations generated from the specified range. The scale field determines the pattern of combination values. In the grid search, the number of parameters and the number of steps resulted in many combinations – 1331 (11 x 11 x 11) iterations were performed to get a set of optimal hyper-parameters for a Rule Induction classifier.

To tune RI parameters using a Grid Search algorithm, we needed to setup an inner process of the *X-Validation* operator in Workflow 5.17. The inner process of the X-Validation operator contains training and testing sections, as shown in Figure 5.20. The training section sets a classification algorithm for classifying a variable set, and a *Rule Induction* operator was added to this section. Testing examines the performance of the learned model in the cross-validation manner, consisting of two operators: *Apply Model* and *Performance*. The *Apply Model* operator applies the RI classification model to the

TABLE 5.13: Hyper-parameters Range for Rule Induction

Parameters	Min	Max	Steps	Scale	Combinations	Number of Combinations
Sample ratio	0.0	1.0	10	Linear	0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1	11
Pureness	0.0	1.0	10	Linear	0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1	11
Minimal prune benefit	0.0	1.0	10	Linear	0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1	11
Total Number of Combinations						1331

testing dataset, and delivers the resultant labelled dataset as the output. This is used by the *Performance* operator to evaluate the model and to generate a performance vector containing information about various performance criteria.

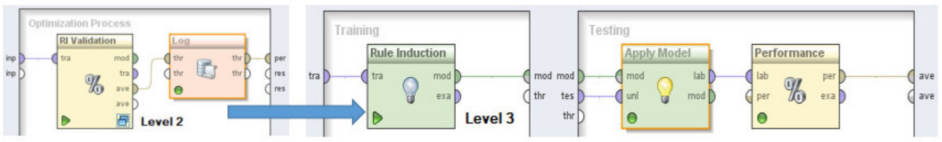


FIGURE 5.20: The Workflow for Optimizing RI Parameters using Grid Search Algorithm

Table 5.14 indicates the results of Result Induction parameter optimisation using Grid Search when applied to full-attributes datasets (IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes) . There are no various values for the following hyper-parameters: Sample Ratio and Pureness. Their best parameter values are 1.0 and 1.0, respectively. In contrast, the values of Minimal Prune Benefit parameter vary.

Confusion Matrixs H.2, H.4, H.6, H.8, and H.10, respectively in Appendix H visualizes the performances of Grid Search to optimize RI parameters to full-attributes of the datasets. Furthermore, the confusion matrixs can be used to measure results of RI parameter optimisation using Grid Search by using following performance measurements (accuracy, precision, recall, and f-measure) for each dataset as depicted in Table 5.14.

The result presented in Table 5.14 indicates that the use of Grid Search to optimize Rule Induction parameters to build a classification model solely based upon a single data source (IMDb) obtained 98.01%, 96.19%, 96.36% and 96.28% accuracy, precision, recall and F-measure, respectively. On the other hand, the model built by RI optimized parameters based on IMDb and external data sources (IMDb++) achieved a model accuracy of 98.15%, precision of 96.78%, recall of 96.61% and F-measure of 96.70%. Therefore, it can be noticed that inclusion of external data sources in the classification model (IMDb++) rose 0.14% accuracy and 0.42% F-Measure values of IMDB model. Moreover, all others performance measurement (precision and recall) values are also higher in model IMDb++ compared to model IMDb. Collectively, this suggests that

Table 5.14: The results of RI Parameter Optimization using Grid Search (the Best Params field ordered by Sample Ratio, Pureness and Minimal Prune Benefit)

Datasets	**# Attr**	**10-Fold Cross Validation**				**Best Params**
		Accuracy	**Precision**	**Recall**	**F-Measure**	
IMDb	16	98.01	96.19	96.36	96.28	1.0, 1.0, & 1.0
IMDb++	23	98.15	96.78	96.61	96.70	1.0, 1.0, & 0.0
FilmAffinity	23	98.27	96.90	96.20	96.55	1.0, 1.0, & 0.3
MovieMeter	23	97.91	95.16	93.52	94.33	1.0, 1.0, & 0.1
RottenTomatoes	23	98.02	94.05	94.30	94.18	1.0, 1.0, & 0.5

IMDb++ model is able to perform better in predicting movie ratings than IMDb model. This also suggests that information in external data sources has the ability to improve the performance for classifying movie popularities by leveraging the use of Grid Search to optimize Rule Induction.

Meanwhile, when we used RI with optimized parameters to build classification models based on three different datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity model is superior to the others. MovieMeter and RottenTomatoes models attained (97.91% accuracy, 95.16% precision, 93.52% recall and 94.33% F-measure) and (98.02% accuracy, 94.05% precision, 94.30% recall and 94.18% F-measure), respectively, whilst, RottenTomatoes model achieved 98.27% accuracy, 96.90% precision, 96.20% recall and 96.55% F-measure. Among those three models, FilmAffinity leads the best values in all performance measures.

By following Moore et al. (2009) (high F-Measure) suggestion, among the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity is the best performing model as it has the highest accuracy, precision, recall, and F-Measure values. Overall, the best performing classification model is IMDb++. Compared with IMDb and the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), IMDb++ has provided the highest F-Measure value even though FilmAffinity has contributed the highest accuracy and precision values.

5.4.4 Parameter Optimisation to SVM Parameters

A SVM learner equipped with a Polynomial kernel has three hyper-parameters that need to be optimised for satisfactory performance on unknown data: C, degree, and γ. For the degree parameter, we specified the lower and upper bound of the range: 1 and 5, respectively. We aimed to check 5 values from the specified range, thus, we determined the value of step field as 5. Subsequently, we selected the linear pattern for the scale field, therefore, the combinations selected are 1, 2, 3, 4, and 5. There are no exact range

TABLE 5.15: Hyper-parameters Range for SVM with Polynomial Kernel

Parameters	**Min**	**Max**	**Steps**	**Scale**	**Combinations**	**NoC**
Degree	1.0	5.0	5	linear	1, 2, 3, 4, 5	6
C	0.001	10000.0	10	logarithmic (legacy)	0.001, 1.513, 5.311, 14.850, 38.813, 99.006, 250.205, 630.002, 1584.021, 3980.431, 10000	11
γ	0.001	10000.0	10	logarithmic (legacy)	0.001, 1.513, 5.311, 14.850, 38.813, 99.006, 250.205, 630.002, 1584.021, 3980.431, 10000	11
Total Number of Combinations (NoC)						726

values for the C and γ parameters. To aid the grid search in finding a set of optimal hyper-parameters, we specified the wider range for both C and γ parameters to be between 0.001 and 10000. Moreover, we decided to specify 10 steps to check the values between the specified range, and selected the logarithmic (legacy) pattern for the scale field. To obtain a set of optimal hyper-parameters for a SVM learner with Polynomial kernel, we performed 726 iterations. Table 5.15 details the SVM hyper-parameters used in our experiment.

For finding the right settings for a SVM classifier with Polynomial kernel using a Grid Search algorithm, an inner process of the *X-Validation* operator was set up in Workflow 5.17. The inner process of the X-Validation operator contains training and testing sections shown in Figure 5.21. Training sets a classification algorithm for classifying a variable set, and we inserted an *SVM* operator here. The testing section examines the performance of the learned model in the a cross-validation manner. For the current problem, the process consists of two operators: *Apply Model* and *Performance*. The *Apply Model* operator applies the SVM learning model to the testing dataset and delivers the resultant labelled dataset as the output, which is used by the *Performance* operator to evaluate the model and generate a performance vector containing information about various performance criteria.

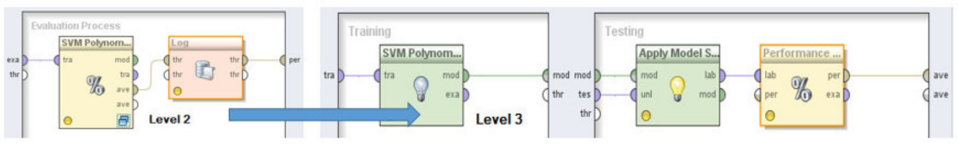


FIGURE 5.21: The Workflow for Optimizing SVM Parameters using Grid Search Algorithm

Our experiment results, collected in table 5.16, indicate that when the Grid Search is applied to full-attributes datasets, the best values of SVM hyper-parameters (C, Degree, and γ) are varied. For the degree parameter, 4 and 5 are found as the best values, whilst for the C and γ parameters, clearly correlations amongst those parameters cannot be found. In some datasets the γ values are higher than the V values. Other datasets show vice versa.

Confusion Matrixs J.2, J.4, J.6, J.8, and J.10, respectively in Appendix J visualizes the performances of Grid Search to optimize the parameters of SVM with Polynomial kernel to full-attributes of the datasets. Furthermore, the confusion matrixs can be used to measure results of SVM Polynomial parameter optimisation using Grid Search by using following performance measurements (accuracy, precision, recall, and f-measure) for each dataset as indicated in Table 5.16.

The result presented in Table 5.16 demonstrates that the use of Grid Search to optimize SVM Polynomial parameters to build a classification model solely based upon a single data source (IMDb) obtained 91.27%, 88.01%, 85.26% and 86.61% accuracy, precision, recall and F-measure, respectively. On the other hand, the model built by SVM Polynomial optimized parameters based on IMDb and external data sources (IMDb++) achieved a model accuracy of 93.93%, precision of 90.56%, recall of 89.22% and F-measure of 89.88%. Therefore, it can be noted that inclusion of external data sources in the classification model (IMDb++) rose accuracy, precision, recall, and F-Measure values of IMDb model by 2.66%, 2.55%, 3.96%, and 3.27%, respectively. Collectively, this suggests that IMDb++ model is able to perform better in predicting movie ratings than IMDb model. This also suggests that information in external data sources has the ability to improve the performance for classifying movie popularities by leveraging the use of Grid Search to optimize SVM with Polynomial kernel.

Meanwhile, when we used SVM Polynomial with optimized parameters to build classification models based on three different datasets (FilmAffinity, MovieMeter, and RottenTomatoes), there is no model superior to any other. FilmAffinity and MovieMeter models attained (92.53% accuracy, 89.47% precision, 88.27% recall and 88.87% F-measure) and (92.53% accuracy, 87.93% precision, 86.67% recall and 87.29% F-measure), respectively, whilst, RottenTomatoes model achieved 94.25% accuracy, 87.85% precision, 85.08% recall and 86.44% F-measure. Among those three models, FilmAffinity leads the best values in precision, recall, and F-Measure, while RottenTomatoes models achieved the best classification accuracy. In addition, MovieMeter achieved higher values in precision, recall, and F-Measure, respectively.

By following Sehgal et al. (2012) (high accuracy, high precision and high recall) and Moore et al. (2009) (high F-Measure) recommendations, among the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity is the best performing model as it has the highest precision, recall, and F-Measure values. Overall, the best performing classification model is IMDb++. Compared with IMDb and the three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes), IMDb++ has provided the highest precision, recall, and F-Measure values. The use of Grid Search to optimize SVM with Polynomial kernel resulted in models in predicting movie ratings ranked from best to worst as follows: IMDb++, FilmAffinity, MovieMeter, IMDb, and RottenTomatoes, respectively.

TABLE 5.16: The results of SVM Polynomial Parameter Optimization using Grid Search

Datasets	# Attr	10-Fold Cross Validation Accuracy	Precision	Recall	F-Measure	Best Params
IMDb	16	91.27	88.01	85.26	86.61	C = 0.001 D = 4 γ = 14.85
IMDb++	23	93.93	90.56	89.22	89.88	C = 0.001 D = 4 γ = 14.85
FilmAffinity	23	92.53	89.47	88.27	88.87	C = 0.001 D = 5 γ = 5.311
MovieMeter	23	92.53	87.93	86.67	87.29	C = 0.001 D = 5 γ = 5.311
RottenTomatoes	23	94.25	87.85	85.08	86.44	C = 0.001 D = 4 γ = 14.85

5.5 The Results Comparison of Classification Models

We summarised the results of the experiment using five single classifiers with default parameters (ANN, DT, k-NN, RI and SVM Polynomial) and four different classifiers (ANN, DT, RI and SVM Polynomial) with optimized parameters in Table 5.18. We applied those classifiers in classifying movie ratings solely based upon a single data source (IMDb), IMDB++ (based on IMDb and external data sources), three other datasets (FilmAffinity, MovieMeter, and RottenTomatoes). We excluded k-NN algorithm to be optimized as because it only has a single parameter k. Our initial experiment showed that the value of k before and after optimization is the same which is 1. Firstly, we applied those classifiers with default parameters to build classification models based on five different datasets (IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes). After that, we implemented parameter optimization with grid search to get a set of optimal hyper-parameters of those classifiers. Then, we made a comparison between model built before and after optimization process for each classifier based on accuracy, precision, recall and F-Measure. To get the best performing model, we compared those models with results of the other classifiers.

The result presented in Table 5.17 reveals that the inclusion of external data sources in building classification models as indicated by IMDb++ models are able to improve results of IMDb models. It was found that only an IMDb++ model built by Decision Tree failed to improve performances of IMDb model. When classified using Decision Tree with default parameters, IMDb achieved better performances in recall and F-Measure values (72.83% and 77.00%, respectivel) compared to IMDb++ which yielded 71.34% recall and 76.75% f-measure.

Table 5.17: Results Comparison of Classifiers with Default and Optimized Parameters Applied to IMDb and IMDb++ Datasets

Classifiers	**Datasets**	**Accuracy**		**Precision**		**Recall**		**F-Measure**	
Default Parameters									
ANN	IMDb++	86.58	1.74	81.85	1.72	74.86	3.40	78.20	2.63
	IMDb++	88.32		83.57		78.26		80.83	
DT	IMDb	84.38	0.64	81.68	1.37	72.83	1.49	77.00	0.25
	IMDb++	85.02		83.05		71.34		76.75	
k-NN	IMDb	89.19	2.00	85.36	3.08	81.66	2.66	83.47	2.86
	IMDb++	91.19		88.44		84.32		86.33	
RI	IMDb	89.62	0.17	88.88	0.95	76.15	0.56	82.02	0.73
	IMDb++	89.79		89.83		76.71		82.75	
SVM	IMDb	84.23	1.03	81.35	0.15	65.41	3.40	72.52	2.10
	IMDb++	85.26		81.50		68.81		74.62	
Optimized Parameters									
ANN	IMDb	86.60	1.97	82.06	2.80	75.40	2.91	78.59	2.86
	IMDb++	88.57		84.86		78.31		81.45	
DT	IMDb	98.16	0.08	96.70	0.09	96.58	0.14	96.64	0.11
	IMDb++	98.24		96.79		96.72		96.75	
RI	IMDb	98.01	0.14	96.19	0.59	96.36	0.25	96.28	0.42
	IMDb++	98.15		96.78		96.61		96.70	
SVM	IMDb	91.27	2.66	88.01	2.55	85.26	3.96	86.61	3.27
	IMDb++	93.93		90.56		89.22		89.88	

As presented in Table 5.18, the use of Grid Search to optimize ANN learner has successfully improved performances of ANN with default parameters in all measures when applied to IMDb (a model solely based on a single data source) and IMDb++ (a model based on IMDb and external data sources). The improvements of IMDb model reached 0.02% accuracy (from 86.58% to 86.60%), 0.48% precision (81.85% to 82.06%), 0.54% recall (74.86% to 75.40%), and 0.39% F-Measure (78.20% to 78.59%). To IMDb++ model, ANN with optimized parameters successfully improved accuracy, precision, recall and F-Measure values of ANN with default parameters by 0.25% (from 88.32% to 88.57%), 1.29% (83.57% to 84.86%), 0.05% (78.26% to 78.31%), and 0.62% (80.83% to 81.45%), respectively. However, ANN optimization processes failed to improve performances of ANN with default parameters when applied to the other datasets (FilmAffinity, MovieMeter, and RottenTomatoes). Recall value has declined 0.23% (75.32% to 75.09%) when ANN with optimzed parameters applied to FilmAffinity dataset. The second drop occured at precision, recall and F-Measure values by 0.41% (80.35% to 79.94%), 1.27% (71.32% to 70.05%), and 1.10% (75.77% to 74.67%), respectively when

ANN with optimized parameters applied to MovieMeter dataset. Lastly, recall and F-Measure values have deteriorated by 1.35% (69.11% to 67.76%) and 0.05% (71.49% to 71.44%), respectively when the optimized ANN applied to RottenTomatoes dataset.

The result presented in Table 5.18 shows that DT optimized with Grid Search has improved all models generated by DT with default parameters in all performance measures. In a model solely based on a single data source (IMDb), improvements reached 13.76% accuracy (from 84.38% to 98.14%), 15.11% precision (81.68% to 96.79%), 23.75% recall (72.83% to 96.58%), and 19.68% F-Measure (77.00% to 96.68%). Improvements of accuracy of 13.21% (85.02 to 98.23%), precision of 13.76% (83.05% to 96.81%) , recall of 25.31% (71.34% to 96.65%) , and 19.98% F-Measure (76.75% to 96.73%) were achieved for IMDb++ (a model based on IMDb and external data sources). FilmAffinity model attained improvements 16.01% (82.01% to 98.02%), 8.57% (88.21% to 96.78%), 41.83% (54.32% to 96.15%), and 29.23% (67.24% to 96.47%) accuracy, precision, recall, and F-Measure, respectively. MovieMeter has enrichment accuracy of 24.78%, precision of 76.70%, recall of 68.65%, and F-Measure of 73.19%, while RottenTomatoes has enhancement 10.51%, 22.89%, 37.14%, and 30.86% for accuracy, precision, recall, and F-Measure values, respectively. Inline with Sehgal et al. (2012) and Moore et al. (2009) suggestions, as IMDb++ model produced by DT with optimized parameters has higher accuracy, precision, recall, and F-Measure values, it can be claimed that IMDb++ as the best performance model in classifying movie ratings. The degradation of 0.41% precision (80.35% to 79.94%), 1.27% recall (71.32% to 70.05%), and 1.10% F-Measure (75.77% to 74.67%) were found when the optimized ANN applied to MovieMeter dataset.

As indicated in Table 5.18, the use of Grid Search to get a set of optimal RI hyper-parameters have successfully improved all performance measurements of RI with default parameters when applied to all datasets (IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes). In a model solely based on a single data source (IMDb), the advancement reached 8.39% accuracy (from 89.62% to 98.01%), 7.31% precision (88.88% to 96.19%), 20.21% recall (76.15% to 96.36%), and 14.26% F-Measure (82.02% to 96.28%). The improvements of IMDb++ (a model based on IMDb and external data sources) achieved 8.36% (89.79 to 98.15%) accuracy, 6.95% (89.83% to 96.78%) precision, 19.90% (76.71% to 96.61%) recall, and 13.95% F-Measure (82.75% to 96.70%). FilmAffinity model attained improvements 7.71% (90.56% to 98.27%), 7.20% (89.70% to 96.90%), 18.51% (77.69% to 96.20%), and 13.29% (83.26% to 96.55%) accuracy, precision, recall, and F-Measure, respectively. MovieMeter has enrichment accuracy of 6.86% (91.05% to 97.91%), precision of 6.86% (88.30% to 95.16%), recall of 28.43% (65.09% to 93.52%), and F-Measure of 19.39% (74.94% to 94.33%), while RottenTomatoes has enhancement 6.60% (91.42% to 98.02%), 9.41% (84.64% to 94.05%), 22.31% (71.99% to 94.30%), and 16.37% (77.81% to 94.18%) for accuracy, precision, recall, and F-Measure values, respectively.

The result presented in Table 5.18 shows that SVM with Polynomial kernel optimized with Grid Search has improved all models generated by SVM Polynomial with default parameters in all performance measures. In a model solely based on a single data source (IMDb), the improvement reached 6.91% accuracy (from 84.23% to 91.14%), 6.64% precision (81.35% to 87.99%), 19.76% recall (65.41% to 85.17%), and 14.04% F-Measure (72.52% to 86.56%). Improvements of accuracy of 8.67% (85.26 to 93.93%), precision of 9.06% (81.50% to 90.56%) , recall of 20.41% (68.81% to 89.22%) , and 15.26% F-Measure (74.62% to 89.88%) were achieved for IMDb++ (a model based on IMDb and external data sources). FilmAffinity model attained improvements 8.36% (84.17% to 92.53%), 7.00% (82.47% to 89.47%), 23.33% (64.94% to 88.27%), and 16.21% (72.66% to 88.87%) accuracy, precision, recall, and F-Measure, respectively. MovieMeter has enrichment accuracy of 8.05%, precision of 22.48%, recall of 45.71%, and F-Measure of 36.91%, while RottenTomatoes has enhancement 7.43%, 26.33%, 34.95%, and 31.19% for accuracy, precision, recall, and F-Measure values, respectively. Accoridng to Sehgal et al. (2012) and Moore et al. (2009) recommendations, as IMDb++ model produced by SVM Polynomial with optimized parameters has higher precision, recall, and F-Measure values, it can be claimed that IMDb++ as the best performance model in classifying movie ratings.

It is perceived that the use of Grid Search to optimize DT, RI and SVM Polynomial classifiers with default parameters resulted in remarkable improvements. For each classifier, accuracy improvement has been achieved for all models (IMDb, IMDb++, FIlmAffininty, MovieMeter and RottenTomatoes). In addition, all the models yielded higher precision, recall, and F-Measure values compared to those generated by classifiers with default parameters. However, the performance of ANN classifier with optimized parameters did not attain to get significant results compared to ANN learner with default parameters when applied to IMDb and IMDb++ datasets. Indeed, when applied to FilmAffinity dataset, recall value of ANN with default parameters is higher than the one with optimized parameters. When applied to MovieMeter dataset, ANN with optimized parameters underperformed in presicion, recall, and F-Measure values than ANN with optimized parameters. When applied to RottenTomatoes dataset, ANN with optimized parameters suffered the declining values of recall and F-Measure compared to ANN with default parameters. Considerations to decide the best performing model is based on higher accuracy, precision, and recall values Sehgal et al. (2012) and higher F-Measure value Moore et al. (2009). Bearing in mind that there is no model provides higher accuracy, precision, recall, and F-Measure values. Hence, we followed Moore et al. (2009) suggestion to use F-Measure value to decide the best performing model. Therefore, we can say that IMDb++ model built by Decision Tree with optimized parameters is the best model among all models generated by the other classifiers (ANN, DT, k-NN, RI, and SVM Polynomial) as it has provided the highest F-Measure value (96.75%). The second best is IMDb++ model generated by RI classifier with optimized parameters. Its F-Measure value is slightly smaller by 0.05% than IMDb++ built by DT optimized parameters.

The results of the current study support Keerthi and Lin (2003) and Lin et al. (2008), where the authors mentioned that improper hyper-parameter settings may lead to deprived classification performances. Also, the results support Bergstra et al. (2011), where they stated that better configuration of existing classification techniques yields advances to classification performances rather than novel approaches to perform learning. The use of Grid Search has provided the highest improvement of F-Measure value by 73.19% (DT with optimized parameters applied to IMDb++ dataset), while the lowest improvement F-Measure value 13.83% has been provided by RI learner with optimized parameters applied to IMDb++ dataset. The higher F-Measure value means the more abilility of classifier to handle unbalanced dataset. MovieMeter dataset consisted 408, 11702, 3653, and 245 instances for classes excellent, average, poor, and terrible, respectively. The confusion matrix E.7 in Appendix E presents the result of DT algorithm with default parameters when applied to MovieMeter dataset. It can be seen that DT classifer was able to correctly classify the average class 100%, however, it totally was not able to correctly classify the rest of the classes. Therefore, this resulted in enough high accuracy by 73.10% but very low F-Measure by 21.12% as indicated in Table 5.16. The set of optimal DT hyper-parameters generated by Grid Search has succesfully improved F-Measure value to 96.73%. As indicated in Confusion Matrix E.8 in Appendix E, there were advances in the performance of DT classifier, so that, it was able to correctly classify excellent, average, poor, and terrible classes by 217, 11153, 3529, and 370 instances, respectively.

The current study provides two evidences. Firstly, the best approach to build a classification model for classifying movie ratings is to employ Decision Tree (DT) algorithm with optimized parameters. To get a set of optimal hyper-parameters, we utilized an optimization algorithm, Grid Search. It has been proven to get an optimal solution for a classification problem, then, a classifier with only default parameters is not enough. It is imperative to make use of optimization algorithm to get the best combination of parameters of classification algorithm. Secondly, the inclusion of external data sources is able to improve the performance of classifiers in classifying movie ratings. DT learner with optimized parameters attained the best classification model when applied to IMDb++ dataset that is based on a single data source (IMDb) and external data sources.

Table 5.18: Classifiers with Default Parameters (DP) and Optimized Parameters (OP) results comparison

Datasets	**Accuracy**	**Precision**	**Recall**	**F-Measure**
ANN (DP VS OP)				

Continued on next page

Table 5.18 – continued from previous page

Datasets	Accuracy		Precision		Recall		F-Measure	
IMDb	86.58	0.02	81.85	0.48	74.86	0.54	78.20	0.39
	86.60		82.06		75.40		78.59	
IMDb++	88.32	0.25	83.57	1.29	78.26	0.05	80.83	0.62
	88.57		84.86		78.31		81.45	
FilmAffinity	86.81	0.42	82.09	1.09	75.32	0.23	78.56	0.37
	87.23		83.18		75.09		78.93	
MovieMeter	88.01	0.49	80.35	0.41	71.32	1.27	75.77	1.10
	88.50		79.94		70.05		74.67	
RottenTomatoes	89.77	0.37	74.04	1.51	69.11	1.35	71.49	0.05
	90.14		75.55		67.76		71.44	
k-NN (DP)								
IMDb	89.19		85.36		81.66		83.47	
IMDb++	91.19		88.44		84.32		86.33	
FilmAffinity	89.69		86.51		82.55		84.49	
MovieMeter	90.64		86.42		80.13		83.16	
RottenTomatoes	91.73		84,87		82.81		83.83	
Decision Tree (DP VS OP)								
IMDb	84.38	13.78	81.68	15.02	72.83	23.75	77.00	19.64
	98.16		96.70		96.58		96.64	
IMDb++	85.02	13.22	83.05	13.74	71.34	25.38	76.75	20.00
	98.24		96.79		96.72		96.75	
FilmAffinity	82.01	16.01	88.21	8.57	54.32	41.83	67.24	29.23
	98.02		96.78		96.15		96.47	
MovieMeter	73.10	24.78	18.28	76.70	25.00	68.65	21.12	73.19
	97.88		94.98		93.65		94.31	
RottenTomatoes	87.53	10.51	72.92	22.89	58.05	37.14	64.64	30.86
	98.04		95.81		95.19		95.50	
Rule Induction (DP VS OP)								
IMDb	89.62	8.39	88.88	7.31	76.15	20.21	82.02	14.26
	98.01		96.19		96.36		96.28	
IMDb++	89.79	8.36	89.83	6.95	76.71	19.90	82.75	13.95
	98.15		96.78		96.61		96.70	
FilmAffinity	90.56	7.71	89.70	7.20	77.69	18.51	83.26	13.29
	98.27		96.90		96.20		96.55	
MovieMeter	91.05	6.86	88.30	6.86	65.09	28.43	74.94	19.39
	97.91		95.16		93.52		94.33	
RottenTomatoes	91.42	6.60	84.64	9.41	71.99	22.31	77.81	16.37
	98.02		94.05		94.30		94.18	
SVM Polynomial (DP VS OP)								
IMDb	84.23	7.04	81.35	6.66	65.41	19.85	72.52	14.09
	91.27		88.01		85.26		86.61	
IMDb++	85.26	8.67	81.50	9.06	68.81	20.41	74.62	15.26
	93.93		90.56		89.22		89.88	

Continued on next page

Table 5.18 – continued from previous page

Datasets	Accuracy		Precision		Recall		F-Measure	
FilmAffinity	84.17	8.36	82.47	7.00	64.94	23.33	72.66	16.21
	92.53		89.47		88.27		88.87	
MovieMeter	84.48	8.05	65.45	22.48	40.96	45.71	50.38	36.91
	92.53		87.93		86.67		87.29	
RottenTomatoes	86.82	7.43	61.52	26.33	50.13	34.95	55.25	31.19
	94.25		87.85		85.08		86.44	

5.6 Summary

This chapter presents the development of classification models for classifiying movie popularities. We used a single data source (IMDb) to build IMDb model. Beside that, we created additional variabels from external data sources and merged them with a single data source (IMDb) to build IMDb++ model. Also, we used three other data sources (FilmAffinity, MovieMeter, and RottenTomatoes) to build FilmAffinity, MovieMeter, and RottenTomatoes models, respectively. This chapter presents experimentation 1 to address research question 2 regarding the use of classifiers with default and optimized parameters when applied to various datasets to predict movie ratings. This experimentation applied five different classifiers (ANN, DT, k-NN, RI, and SVM Polynomial) with default parameters into five datasets (IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes). Subsequently, we applied Grid Search algorithm to ANN, DT, RI, and SVM Polynomial to get the best setting of their parameters. After that, we applied those four classifiers with optimized parameters into the datasets (IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes). We used linked data technologies to integrate data from disparate data sources and queried data using SPARQL queries in building classification models. We used four performance measurements (accuracy, prediction, recall, and F-Measure) to analyse the impact of optimization processes to the models. The result of the experimentation indicates that the classifiers with optimized parametets are superior to those with default parameters. Also, the result highly supports that Decision Tree with optimized parameters is the best approach when applied to IMDb++ dataset which is based on a single data source (IMDb) and external data sources.

Therefore, the experimentation result highlights the importance of optimization process to boost up the performance of classifiers. Also, the result suggest that linked data technologies can be used to explore relevant data sources and to create additional variables to support the process of knowledge discovery. As a result, it would be achieved the improvement of classifier performances in predicting movie rating.

Chapter 6

Feature Selection for Movie Classification Models

In this chapter, we focus on employing the feature selection technique. We applied Genetic Algorithm (GA) and Particle Swarm Optimisation (PSO) algorithms to reduce a dataset to their essential characteristics or features. We analysed its effect as to whether the reduced datasets would achieve a similar or even better classification performance to using all available attributes. Also, we identified the most impactful attributes in contributing to movie popularity.

6.1 Feature Selection System Architecture

Feature selection serves a threefold purpose Guyon and Elisseeff (2003) Liu and Yu (2005): it improves the performance of the data mining algorithm, it provides a faster and more cost-effective data mining processes, and it provides a better understanding of the outcome of the modelling for the analyst. The motivations for incorporating feature selection in the classification process is to remove input (independent) attributes that may strongly correlate with one another, and to retain input attributes that may strongly correlate with the output (dependent) attribute. In contrast, the inclusion of irrelevant, redundant and noisy attributes in the classification model building process phase can result in poor classification accuracy and higher computation costs Hall and Holmes (2003). In this chapter, we perform some experiments to filter out unimportant attributes by leveraging two feature selection models: GA and PSO. In doing so, we aim to narrow down and identify the most important attributes in the movie classification models, and produce better classification accuracy based on the few most essential attributes. The system architecture of feature selection is shown in Figure 6.1.

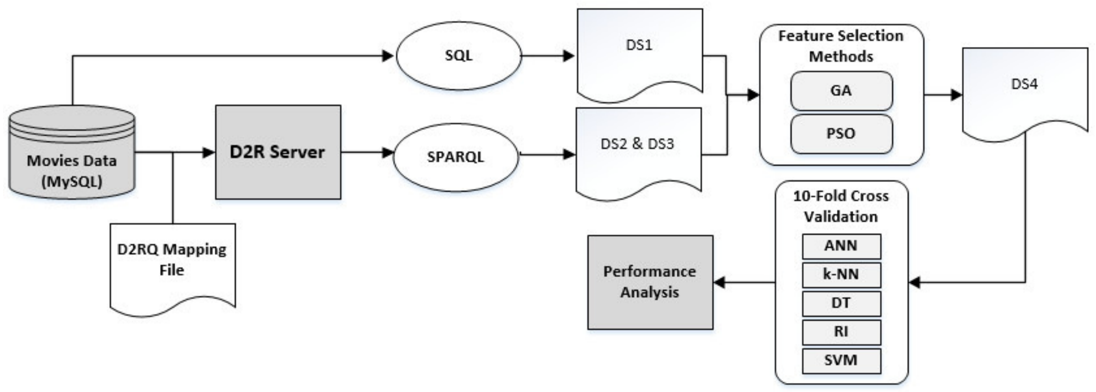


FIGURE 6.1: The Architecture of Feature Selection System

6.2 Feature Selection Algorithms

The following section explains how GA and PSO are used to select the most valuable features. As described in Chapter 2, filter and wrapper technique are the two types of feature selection methods. Applying wrapper techniques will most often achieve better results than the filter techniques, because the former are trained and will adjust to the specific machine learning algorithm Hall and Holmes (2003) Chen et al. (2006). Additionally, they have the ability to take into account feature dependencies Saeys et al. (2007). In this research, we employed GA and PSO as the search algorithms to find the best features or subsets. Based on the general framework of wrapper feature selection methods by Li et al. (2016), the logic of the wrapper-type feature selection function is depicted in Figure 6.2. Generally, the logic has three process levels. The root level contains a task to choose a feature selection algorithm. The next level is dedicated to set up the learning process, mainly used to evaluate the performance of classification algorithm. The bottom level performs the training and testing of the specific classification algorithm.

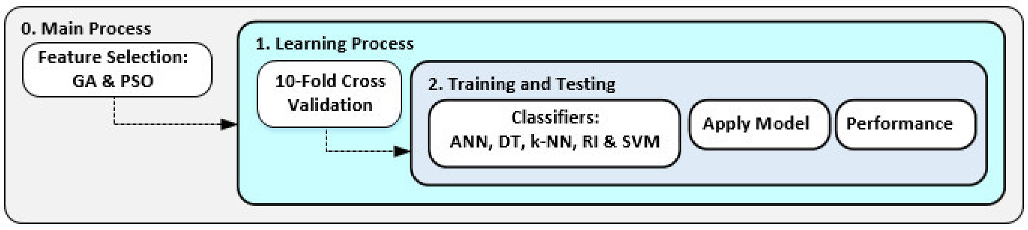


FIGURE 6.2: The Logic of Wrapper-Type Feature Selection Method

To evaluate the performance of feature selection algorithms, we use a measurement called Fraction of features (FF). FF is the ratio of the number of features used by the classifier to the total number of features in the dataset. The greater value of FF means the lesser number of attributes being reduced by the feature selection algorithm.

6.2.1 Genetic Algorithm (GA) Search

We use GA as the first searching algorithm to obtain the most relevant attributes from a dataset. A GA is a search heuristic that imitates the process of natural evolution, such as inheritance, mutation, selection, and crossover to solve search problems. The mutation is used to switch features on and off. The crossover serves as interchanging used features, while the selection means selecting individuals who will recombine for the next generation.

A genetic algorithm (GA) works as follows:

1. Initialisation.
   This generates an initial population, P, consisting of ρ individuals. Each individual in the population represents a classification model. The number of genes is the total number of attributes in the dataset. The genes are binary values, and represent the presence (1) or absence (0) of particular features in the model. A GA is a stochastic optimization method, so that, the genes of the individuals are usually initialised randomly.

2. Fitness Assignment and Evaluation.
   This comes after the population has been generated and initialised. There are two tasks in this step. Firstly, the fitness to each individual is assigned within the population. Secondly, the classification model is trained with the training data, and then selection error is evaluated with the selection data. The aim of the second task is to evaluate the fitness of each individual, where individuals with low fitness indicates a high selection error. Individuals with greater fitness will have a greater chance of being selected for recombination.

3. Selection.
   It is a step to select the individuals that will recombine for the next generation. The individuals selected are the most survived and fitted to the environment So that, the fitness level is the main factor for individuals to be selected. Given P the population size, the number of selected individuals is P/2. An elitism selection method is used to directly select an individual who has the fittest level. The rest selected individuals can be selected using various selection methods, such as: uniform, cut, roulette wheel, stochastic universal sampling, Boltzman, rank, tournament, non dominated sorting, etc.

4. Crossover.
   This generates a new population, as it picks the selected individuals gathered from the selection step and combines their features to obtain offspring for the new population. This is repeated until the new population has the same size as the original one. One point, uniform and shuffle can be chosen to perform crossover processes.

5. Mutation.
   This randomly changes the value of the offspring attributes. The resultant offspring may have high similarity to the parents, and cause a new generation with low diversity. Therefore, it is required to modify the value of some offspring features at random. After the mutation process, a new generation may have more diversity.

6.2.2 Particle Swarm Optimization (PSO) Search

One problem of feature selection can be in finding the most efficient global search technique. Evolutionary Computation (EC) is a popular technique to perform global searching. PSO is a relatively novel EC technique Kennedy and Eberhart (1995) Shi and Eberhart (1998) which works based on swarm intelligence. Compared with other EC algorithms such as genetic algorithms (GAs) and genetic programming (GP), PSO is computationally less expensive and can converge more quickly Xue et al. (2013). Therefore, PSO has been used as an effective technique in many fields, including feature selection Unler and Murat (2010) Chen et al. (2012).

A Particle Swarm Optimisation (PSO) works as follows:

1. Initialisation.
   This creates and initialises the particles in the search space population. The PSO algorithm works on a swarm having a set of particles, in which, each particle moves around the search space. Each particle in the swarm represents a potential solution (fitness) and has a position vector, a velocity vector, the best previous position of the particle ($Pbest_i$), and the index of the best particle (Gbest). The position and velocity vectors of i^{th} particle in the D dimensional search space are represented as $x_i = (x_i1,\ x_i2,\ ...,\ x_iD)$ and $v_i = (v_i1,\ v_i2,\ ...,\ v_iD)$, respectively. The range value of v_i vector is limited between $-v_{max}$ and v_{max}. This intends to reduce the potency of particles to leave the search space.

2. Update position and velocity.
   In PSO, optimal solutions can be obtained by updating the position and the velocity of each particle using the two following equations:

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \tag{6.1}$$

$$v_{id}^{t+1} = w * v_{id}^{t} + v_{id}^{t+1} + c_1 * r_1 * (p_{id} - x_{id}^{t}) + c_2 * r_2 * (p_{gd} - x_{id}^{t}) \tag{6.2}$$

   where t represents the t^{th} iteration in the evolutionary process. $d \in D$ represents the d^{th} dimension in the search space. w is inertia weight, used to control the impact of the previous velocities on the current velocity. c_1 and c_2 are acceleration constants. r_1 and r_2 are random values uniformly distributed in [0,1]. p_{id} and

p_{gd} denote the elements of $Pbest$ and $Gbest$ in the d^{th} dimension. The velocity is limited by a predefined maximum velocity, v_{max}, and $v_{id}^{t+1} \in [-v_{max}, v_{max}]$.

The PSO algorithm has been described in Section 2.1.2 but there is a slight modification when it applies to feature selection problems. The flow chart of a PSO search is explained in Figure 6.3 (Jwo and Chang (2009) Seal et al. (2014)).

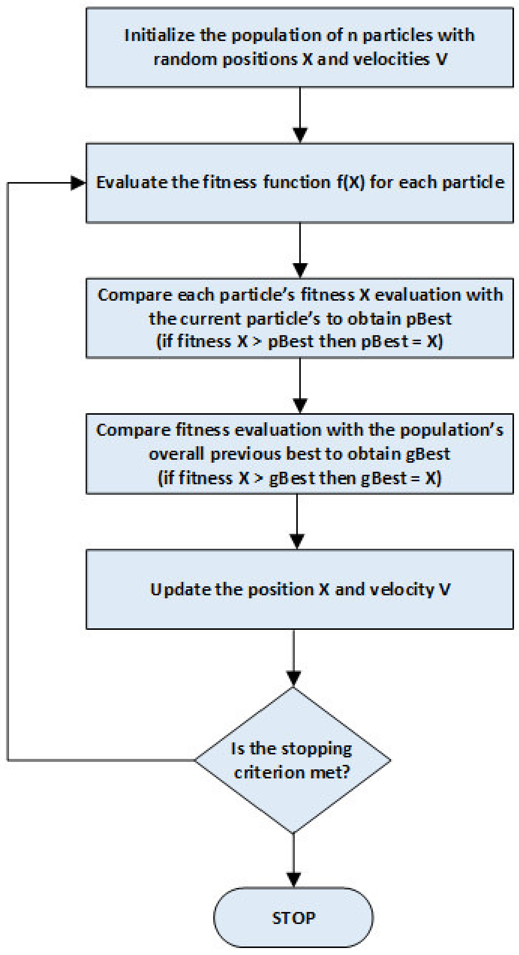

Initialize the population of n particles with random positions X and velocities V
Evaluate the fitness function f(X) for each particle
Compare each particle's fitness X evaluation with the current particle's to obtain pBest (if fitness X > pBest then pBest = X)
Compare fitness evaluation with the population's overall previous best to obtain gBest (if fitness X > gBest then gBest = X)
Update the position X and velocity V
Is the stopping criterion met?
STOP

FIGURE 6.3: The Flow Chart of the PSO algorithm for Feature Selection

6.3 Experimental Results

The goal of this experiment is to build a classification model that contains as few attributes as possible, without compromising the predictive ability of the model. To achieve this, we employed a wrapper-type feature selection method, in order to iteratively select features to add or remove from an original dataset, based on whether the newly added or removed variable improves the accuracy. We leveraged the two feature selection algorithms: Genetic Algorithm (GA) and Particle Swarm Optimisation (PSO) to reduce the number of attributes, while at the same time maintaining or improving the accuracy. For the classification algorithms, we use the five following classifiers: Artificial Neural Network (ANN), Decision Tree (DT), k-Nearest Neighbour (k-NN), Rule Induction (RI), and Support Vector Machine (SVM).

As indicated in Figure 6.2, the logic to implement feature selection techniques is not "linear", but a nested logic. We implemented the logic of Figure 6.2 to build classification models using RapidMiner. To implement the feature selection algorithm and to evaluate the learning process, we designed a workflow as shown in Figure 6.4. We used the *Optimise Selection* operator to implement a GA or PSO method, and used the 10-Fold Cross Validation to measure the statistical performance of a learning algorithm. In RapidMiner, 10-Fold Cross Validation is implemented using the *X-Validation* operator.

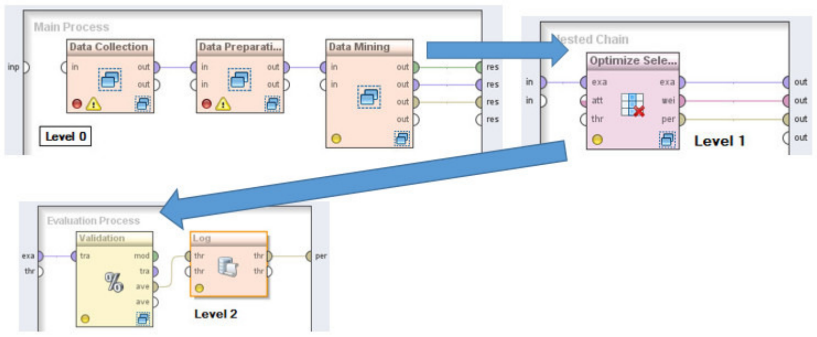


FIGURE 6.4: The Workflow to setup a feature selection method

To obtain the desired model, Genetic Algorithm (GA) has the following parameters to be adjusted and customised:

1. **min number of attributes:** This parameter specifies the minimum number of attributes used for the combinations to be generated and tested. The default value is 1.

2. **population size:** This parameter determines the population size, which means the number of individuals per generation. The default value is 5.

3. **maximum number of generations:** This parameter determines a limit for the number of generations, and the algorithm should be terminated when it reaches this limit. The default value is 30.

4. **initialize probability:** This parameter is useful to determine an attribute to be switched on. The default value is 0.5

5. **mutation probability:** This parameter specifies the probability for an attribute to be changed. The default value is -1. If the value is -1 the probability will be *1* n where n is the total number of attributes.

6. **crossover probability:** This parameter sets the probability for an individual to be selected for crossover. The default value is 0.5.

The following parameters are available for users who wish to customise PSO for desired modelling behaviour:

1. **population size:** This parameter determines the number of individuals per generation. The default value is 5.
2. **maximum number of generations:** This parameter determines a limit for the number of generations and the algorithm should be terminated when it reaches this. The default value is 30.
3. **inertia weight:** This parameter determines the initial weight for the old weighting. The default value is 1.0.
4. **local best weight:** This parameter specifies the weight or the individual's best position during run. The default value is 1.0.
5. **global best weight:** This parameter determines the weight for the population's best position during run. The default value is 1.0.

We ran the GA and PSO algorithms for feature selection using default parameters. In the subsections that follow, we describe learning processes for each classification algorithm involved in the feature selection tasks.

6.3.1 Artificial Neural Network (ANN)

According Figure 6.2, the last process of the logic performing wrapper feature selection is to tuck the training and testing process inside another sub-process, the learning process. In this sub section, we trained the feature selection algorithm using ANN learning algorithm with default and optimized parameters, and inserted the training and testing process inside the *Validation* operator in Workflow 6.4. The training and testing process consists of three operators. The *Neural Net* operator is inside the training section, while the *Apply Model* operator and the *Performance* operator are in the testing section. Figure 6.5 shows the workflow to configure the ANN classifier to train the feature selection algorithm.

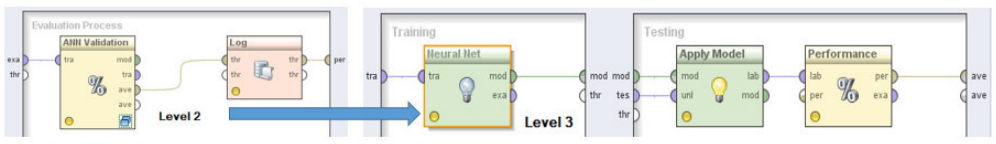


FIGURE 6.5: The Workflow for Learning Process using ANN

Table 6.1 shows the results of the wrapper-type feature selection using the ANN classification algorithm with default parameters. The figure indicates that feature selection

TABLE 6.1: The Results of Wrapper-Type Feature Selection With ANN Classifier with default parameters

Datasets	Selected Features	Mtd	#
IMDb (16)	costume designer, genre, director, producer, competition, editor, actor, cinematographer, mpaa, writer, actress, production company, budget, distributor, composer	PSO	15
IMDb++ (23)	writer, actor, cinematographer, opening gross, mpaa, director, metascore, opening theaters, golden globe won, actress, budget, golden globe nominee, oscar won, producer, competition, product designer, composer, production company, costume designer, genre, editor, distributor	PSO	22
FA (23)	actor, actress, budget, cinematographer, composer, costume designer, director, distributor, editor, golden globe won, metascore, mpaa, opening gross, opening theaters, oscar nominee, production company, product designer, producer, writer	GA	19
MM (23)	actress, budget, cinematographer, competition, composer, costume designer, director, distributor, editor, golden globe nominee, golden globe won, metascore, mpaa, oscar nominee, oscar won, production company, product designer, producer, writer	GA	19
RT (23)	mpaa, metascore, opening theaters, actress, production company, oscar won, producer, budget, oscar nominee, writer, editor, costume designer, director, distributor, cinematographer, opening gross, genre, golden globe nominee, golden globe won, competition, product designer, composer	PSO	22

using GA can reduce the unimportant attributes more effectively than PSO. For example, in the FilmAffinity and MovieMeter dataset, GA reduced 4 attributes from 23 to 19. On the other hand, PSO was only able to remove one irrelevant attribute from 16 to 15 on the IMDb dataset, while the IMDb++ and RottenTomatoes dataset were reduced from 23 to 22. In the original IMDb dataset, the Production Company considered is not an essential feature to improve the performance of ANN classifier in predicting the movie popularity. Therefore, it has been removed by the PSO algorithm from the reduced IMDb dataset as shown in Table 6.1. For the FilmAffinity dataset, two original attributes were removed (competition & genre), and two additional attributes eliminated (golden globe nominee & oscar won).

We trained the five reduced datasets, as displayed in Table 6.1, using five different classifiers with default parameters. Table C.11, C.13, C.15, C.17, and C.19 in Appendix C demonstrates confusion matrixs of ANN classifier with default parameters when applied to reduced attibutes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Those confusion matrixs were able to visualize missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of ANN classifier with default parameters when applied to those five reduced datasets as shown in Table 6.2. The table present accuracy, precision, recall, and f-measure value for each dataset.

Table 6.2: The Movie Ratings Prediction Model Using ANN default parameters with Feature Selection

Datasets	**#Attr**	**10-Fold Cross Validation**				**Diff**
		Accuracy	**Precision**	**Recall**	**F-Measure**	
IMDb	15	86.59	82.05	74.91	78.32	
IMDb++	22	88.52	84.64	78.47	81.44	3.12
FilmAffinity	19	86.85	82.10	74.12	77.91	0.41
MovieMeter	19	88.04	79.28	71.49	75.18	3.14
RottenTomatoes	22	89.81	75.75	67.77	71.54	6.78

The results are shown in Table 6.2. The results of augmenting the original dataset (IMDb) with additional attributes from external data sources demonstrates that even in using the reduced datasets by selecting several key features, IMDb++ dataset can still improve on the level of accurate classification of the IMDb dataset. IMDb++ model provided the improvements of accuracy, precision, recall, and F-Measures values of IMDb model by 1.93% (86.59% to 88.52%), 2.59% (82.05% to 84.64%), 3.56% (74.91% to 78.47%), and 3.12% (78.32% to 81.44%), respectively. Meanwhile, when ANN classifier with default parameters applied to the other reduced-datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity model attained 86.85% accuracy, 82.10% precision, 74.12% recall and 77.91% F-measure. MovieMeter model yielded 88.04% accuracy, 79.28% precision, 71.49% recall and 75.18% F-measure, while RottenTomatoes model achieved 89.81% accuracy, 75.75% precision, 67.77% recall and 71.54% F-measure.

Table 6.3 shows the results of the wrapper-type feature selection using the ANN classification algorithm with optimized parameters. We applied a set of optimal hyper-parameters for ANN classifier as attained in Table 5.10 Chapter 5. The figure indicates that feature selection using GA outperformed PSO in reducing unimportant attributes in all datasets. Subsequently, we trained the reduced-datasets appeared in Table 6.3 using ANN classifier with optimized parameters. Table C.12, C.14, C.16, C.18, and C.20 in Appendix C demonstrates confusion matrixs of ANN classifier with optimized parameters when applied to reduced-attibutes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Those confusion matrixs were able to visualize missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of ANN classifier with optimized parameters when applied to those five reduced datasets as shown in Table 6.4. The table presents accuracy, precision, recall, and f-measure value for each dataset.

The result presented in Table 6.4 indicated that results of augmenting the original dataset (IMDb) with additional attributes from external data sources demonstrates that even in using the reduced-datasets by selecting several key features, IMDb++ dataset can still improve on the level of accurate classification of the IMDb dataset. IMDb++ model provided the improvements of accuracy, precision, recall, and F-Measures values of IMDb model by 1.64% (86.46% to 88.10%), 1.66% (83.21% to 84.87%), 3.33% (73.33% to 76.66%), and 2.60% (77.96% to 80.56%), respectively. Meanwhile, when

Table 6.3: The Results of Wrapper-Type Feature Selection With ANN Classifier with optimized parameters

Datasets	Selected Features	Mtd	#
IMDb (16)	actor, actress, cinematographer, composer, director, distributor, editor, production company, product designer, producer, writer, budget, mpaa	GA	13
IMDb++ (23)	actor, actress, budget, cinematographer, composer, costume designer, director, distributor, editor, genre, golden globe nominee, metascore, oscar nominee, oscar won, production company, producer, writer	GA	17
FA (23)	actor, actress, budget, cinematographer, costume designer, director, distributor, editor, golden globe won, metascore, opening gross, opening theaters, oscar nominee, oscar won, production company, product designer, producer, writer	GA	18
MM (23)	actress, budget, cinematographer, competition, composer, costume designer, director, distributor, editor, golden globe nominee, metascore, mpaa, opening gross, opening theaters, oscar nominee, oscar won, product designer, producer, writer	GA	19
RT (23)	actress, cinematographer, competition, composer, costume designer, director, distributor, editor, golden globe nominee, metascore, oscar nominee, oscar won, production company, product designer, producer, writer	GA	16

Table 6.4: The Movie Ratings Prediction Model Using ANN optimized parameters with Feature Selection

Datasets	#Attr	10-Fold Cross Validation				Diff
		Accuracy	Precision	Recall	F-Measure	
IMDb	13	86.46	83.21	73.33	77.96	
IMDb++	17	88.10	84.87	76.66	80.56	2.60
FilmAffinity	18	86.63	82.59	73.72	77.90	0.06
MovieMeter	19	88.37	81.05	68.17	74.05	3.91
RottenTomatoes	16	89.06	74.59	60.29	66.68	11.28

ANN classifier with optimized parameters applied to the other reduced-datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity model attained 86.63% accuracy, 82.59% precision, 73.72% recall and 77.90% F-measure. MovieMeter model yielded 88.37% accuracy, 81.05% precision, 68.17% recall and 74.05% F-measure, while RottenTomatoes model achieved 89.06% accuracy, 74.59% precision, 60.29% recall and 66.68% F-measure.

6.3.2 Decision Tree (DT)

According to the logic of performing wrapper feature selection as shown in Figure 6.2, the inner process of feature selection is the learning process. To apply a wrapper-type feature selection, such as GA and PSO, using a DT learning algorithm, we need to setup an inner process of the X-Validation operator in Workflow 6.4. The inner process of the X-Validation operator contains a model-building environment and a model evaluation environment as shown in Figure 6.6. The former environment sets a learning algorithm for classifying an example set, and we decided to add a *Decision Tree* operator in this environment. The model evaluation environment examines the performance of the learned model in a cross validation manner. For the current problem, the process consists of two operators: *Apply Model* and *Performance*. The *Apply Model* operator applies the Decision Tree classification model on the testing dataset and delivers the resultant labeled dataset as the output. This is then used by the *Performance* operator to evaluate the model and generate a performance vector containing information about various performance criteria.

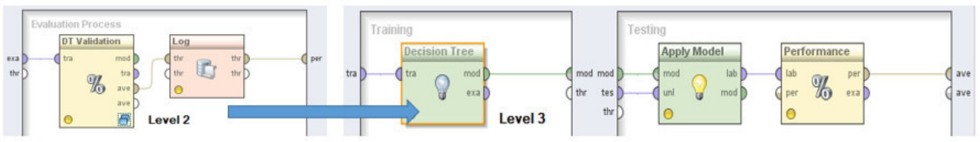


FIGURE 6.6: The Workflow for Learning Process using Decision Tree

Table 6.5 indicates the results of wrapper-type feature selection using the Decision Tree learning algorithm with default parameters. It is clear that GA achieves the highest level of feature selection by removing the largest number of attributes of both FilmAffinity and MovieMeter datasets by 12 (from 23 attributes to 11 attributes). PSO was only able to remove one irrelevant attribute in the IMDb++ and RottenTomatoes datasets. In the FilmAffinity dataset, GA removed six additional attributes from external data sources and only retained the *oscar won* attribute. In the MovieMeter dataset, GA preserved two attributes (*oscar nominee* & *golden globe won*) gathered from external data sources. On the other hand, in building a classification model, PSO only left out one external attribute. The *oscar nominee* attribute is not used in IMDb++ dataset, while the *golden globe won* is not involved in the RottenTomatoes dataset.

We trained the five reduced-datasets, as displayed in Table 6.5, using five different classifiers with default parameters. Table E.11, E.13, E.15, E.17, and E.19 in Appendix E demonstrates confusion matrixs of DT classifier with default parameters when applied to reduced-attibutes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Those confusion matrixs were able to visualize missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of DT classifier with default parameters when applied to those five reduced-datasets as shown in Table 6.6. The table present accuracy, precision, recall, and f-measure value for each dataset.

The results presented in Table 6.6 shows that IMDb++ model provided improvements of accuracy and precision values of IMDb model by 0.63% (84.40% to 85.03%) and 1.35% (81.71% to 83.06%), respectively. However, IMDb++ model failed to improve recall and F-Measure values of IMDb model. It was found declining values of recall and F-Measure by 1.57% (72.93% to 71.36%) and 0.31% (77.07% to 76.76%), respectively.

TABLE 6.5: The Results of Wrapper-Type Feature Selection with Decision Tree Classifier with Default Parameters

Datasets	**Selected Features**	**Mtd**	**#**
IMDb (16)	actor, actress, director, editor, production company, producer, competition, genre, writer, budget, cinematographer, mpaa, composer, costume designer	PSO	14
IMDb++ (23)	writer, actor, cinematographer, opening gross, mpaa, director, metascore, opening theaters, golden globe won, actress, budget, golden globe nominee, oscar won, producer, competition, product designer, composer, production company, costume designer, genre, editor, distributor	PSO	22
FA (23)	actor, composer, director, editor, production company, writer, oscar won, budget, competition, genre, opening gross	GA	11
MM (23)	actress, composer, costume designer, distributor, editor, production company, product designer, oscar nominee, golden globe won, competition, genre	GA	11
RT (23)	opening theaters, cinematographer, production company, writer, oscar nominee, mpaa, director, golden globe nominee, distributor, budget, producer, actress, metascore, competition, editor, costume designer, oscar won, product designer, genre, opening gross, actor, composer	PSO	22

TABLE 6.6: The Movie Ratings Prediction Model Using Decision Tree Default Parameters with Feature Selection

Datasets	**#Attr**	**10-Fold Cross Validation**				**Diff**
		Accuracy	**Precision**	**Recall**	**F-Measure**	
IMDb	14	84.40	81.71	72.93	77.07	
IMDb++	22	85.03	83.06	71.36	76.76	0.31
FilmAffinity	11	82.18	88.80	54.81	67.78	9.29
MovieMeter	11	73.10	18.28	25.00	21.12	55.95
RottenTomatoes	22	87.53	72.92	58.05	64.64	12.43

Therefore, it can be mentioned that the inclusion of external data sources in IMDb++ model was not success to improve the classification performance of IMDb model (the model soley based upon single data source). Meanwhile, when DT classifier with default parameters applied to the other reduced-datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity model attained 82.18% accuracy, 88.80% precision, 54.81% recall and 67.78% F-measure. MovieMeter model yielded 73.10% accuracy, 18.28% precision, 25.00% recall and 21.12% F-measure, while RottenTomatoes model achieved 87.53% accuracy, 72.92% precision, 58.05% recall and 64.64% F-measure.

Table 6.7 shows the results of the wrapper-type feature selection using the DT classification algorithm with optimized parameters. We applied a set of optimal hyper-parameters for DT classifier as attained in Table 5.12 Chapter 5. The figure indicates that feature selection using GA outperformed PSO in reducing unimportant attributes in all datasets. Subsequently, we trained the reduced-datasets appeared in Table 6.7 using DT classifier with optimized parameters. Table E.12, E.14, E.16, E.18, and E.20 in Appendix E

Table 6.7: The Results of Wrapper-Type Feature Selection With DT Classifier with optimized parameters

Datasets	Selected Features	Mtd	#
IMDb (16)	budget, cinematographer, composer, costume designer, director, distributor, production company, product designer, producer, writer	GA	10
IMDb++ (23)	budget, composer, director, distributor, editor, mpaa, opening gross, opening theaters, oscar nominee, production company, product designer, producer, writer	GA	13
FA (23)	budget, cinematographer, competition, composer, director, distributor, editor, mpaa, opening gross, oscar nominee, production company, product designer, producer, writer	GA	14
MM (23)	actor, budget, cinematographer, competition, composer, director, editor, golden globe nominee, mpaa, opening gross, product designer, producer, writer	GA	13
RT (23)	budget, competition, composer, costume designer, director, distributor, golden globe won, metascore, opening theaters, oscar won, product designer, producer, writer	GA	13

demonstrates confusion matrixs of DT classifier with optimized parameters when applied to reduced-attibutes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Those confusion matrixs were able to visualize missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of DT classifier with optimized parameters when applied to those five reduced datasets as shown in Table 6.8. The table presents accuracy, precision, recall, and f-measure value for each dataset.

The result presented in Table 6.8 indicated that results of augmenting the original dataset (IMDb) with additional attributes from external data sources demonstrates that even in using the reduced-datasets by selecting several key features, IMDb++ dataset can still improve on the level of accurate classification of the IMDb dataset. IMDb++ model provided the improvements of accuracy, precision, recall, and F-Measures values of IMDb model by 0.06% (98.47% to 98.53%), 0.07% (97.73% to 97.80%), 0.02% (97.40% to 97.42%), and 0.05% (97.56% to 97.61%), respectively. Meanwhile, when DT classifier with optimized parameters applied to the other reduced-datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity model attained 98.31% accuracy, 96.91% precision, 96.71% recall and 96.81% F-measure. MovieMeter model yielded 98.24% accuracy, 95.60% precision, 94.53% recall and 95.06% F-measure, while RottenTomatoes model achieved 98.44% accuracy, 93.81% precision, 96.23% recall and 95.01% F-measure.

TABLE 6.8: The Movie Ratings Prediction Model Using DT optimized parameters with Feature Selection

Datasets	**#Attr**	**10-Fold Cross Validation**				**Diff**
		Accuracy	**Precision**	**Recall**	**F-Measure**	
IMDb	10	98.47	97.73	97.40	97.56	
IMDb++	13	98.53	97.80	97.42	97.61	0.05
FilmAffinity	14	98.31	96.91	96.71	96.81	0.75
MovieMeter	13	98.24	95.60	94.53	95.06	2.50
RottenTomatoes	13	98.44	93.81	96.23	95.01	2.55

6.3.3 k-NN (k-Nearest Neighbour)

According to the logic diagram of performing feature selection in Figure 6.2, the main process contains an inner learning process. The learning process is used to tuck a model-building environment and a model-evaluation environment. To implement a wrapper-type feature selection method such as GA and PSO using k-NN classification algorithm, it is required to setup an inner process of the *X-Validation* operator in Workflow 6.4. The inner process of the *X-Validation* operator contains a model-building environment for the training process and a model-evaluation environment for the testing one as shown in Figure 6.7. The training section sets a learning algorithm for classifying an example set. We selected a *k-NN* operator to add to the training section. The testing section examines the performance of the learned model in the cross validation manner. For the current problem, the process consists of two operators: *Apply Model* and *Performance*. The *Apply Model* operator applies the k-NN classification model on the testing dataset and delivers the resultant labeled dataset as the output. This is used by the *Performance* operator to evaluate the model and to generate a performance vector which contains information about various performance criteria.

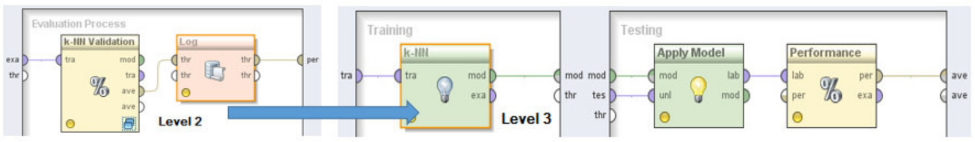


FIGURE 6.7: The Workflow for Learning Process using k-NN

Table 6.9 indicates that the k-NN classifier fits better when implemented with PSO than GA in the wrapper-type variable selection method. PSO achieved the best results in four datasets (IMDb, IMDb++, MovieMeter & RottenTomatoes), while GA obtained its best result only in the Filmaffinity dataset. PSO trained with k-NN has the smallest percentage of FF by 37.5% when implemented in the IMDb dataset, meaning that PSO selected the least number of attributes in IMDb dataset. In the RottenTomatoes dataset, PSO eliminated the largest number of unimportant attributes by 12 (from 23 attributes to 11 attributes). From the 23 features of FilmAffinity dataset, GA only selected the 12 most important features. Out of 11 features of the reduced RottenTomatoes dataset, PSO considered the three following parameters (*oscar nominee*, *golden globe nominee*,

Table 6.9: The Results of Wrapper-Type Feature Selection With k-Nearest Neighbour Classifier Default Parameters

Datasets	Selected Features	Mtd	#
IMDb(16)	actress, production company, producer, writer, competition, costume designer	PSO	6
IMDb++(23)	opening theaters, cinematographer, director, product designer, producer, writer, oscar won, oscar nominee, golden globe nominee, budget, actor, opening gross, mpaa, metascore, golden globe won, actress, competition, composer, production company, costume designer, genre, editor, distributor, actor	PSO	13
FA(23)	opening theaters, cinematographer, director, distributor, editor, producer, writer, oscar won, oscar nominee, golden globe won, mpaa, opening gross	GA	12
MM(23)	opening theaters, cinematographer, director, production company, product designer, producer, oscar won, oscar nominee, golden globe nominee, budget, mpaa, opening gross, writer, actor, actress, composer, costume designer, distributor	PSO	18
RT(23)	cinematographer, director, production company, product designer, producer, writer, oscar nominee, golden globe nominee, opening gross, composer, costume designer	PSO	11

opening gross) gathered from external data sources as parts of the most important attributes for improving the classification performance. In the reduced FilmAffinity dataset, GA takes into account the following four additional attributes (*oscar won*, *oscar nominee*, *golden globe won*, & *opening gross*) and the final four original attributes as the most prominent features.

Table 6.10 shows classification performances of k-NN with default parameters when implemented in different reduced-attributes datasets, as displayed in Table 6.9. Table G.2, G.4, G.6, G.8, and G.10 in Appendix G demonstrates confusion matrixs of kNN classifier with default parameters when applied to reduced attibutes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Those confusion matrixs were able to visualize missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of ANN classifier with default parameters when applied to those five reduced datasets as shown in Table 6.10. The table present accuracy, precision, recall, and f-measure value for each dataset.

The result presented in Table 6.10 shows that by augmenting the original dataset (IMDb) with additional attributes from external data sources (IMDb++) demonstrates that even in using the reduced datasets by selecting several key features, IMDb++ dataset can still improve on the level of accurate classification of the IMDb dataset. IMDb++ model provided the improvements of accuracy, precision, recall, and F-Measures values of IMDb model by 0.34% (97.84% to 98.18%), 1.13% (96.11% to 97.24%), 0.28% (95.91% to 96.19%), and 0.70% (96.01% to 96.71%), respectively. Meanwhile, when kNN classifier

TABLE 6.10: The Movie Ratings Prediction Model Using kNN Default Parameters with Feature Selection

Datasets	#Attr	10-Fold Cross Validation				Diff
		Accuracy	Precision	Recall	F-Measure	
IMDb	6	97.84	96.11	95.91	96.01	
IMDb++	13	98.18	97.24	96.19	96.71	0.70
FilmAffinity	12	98.04	97.09	95.79	96.44	0.43
MovieMeter	18	97.84	96.11	92.97	94.51	1.50
RottenTomatoes	11	98.19	92.04	94.29	93.15	2.86

with default parameters applied to the other reduced-datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity model attained 98.04% accuracy, 97.09% precision, 95.79% recall and 96.44% F-measure. MovieMeter model yielded 97.84% accuracy, 96.11% precision, 92.97% recall and 94.51% F-measure, while RottenTomatoes model achieved 98.19% accuracy, 92.04% precision, 94.29% recall and 93.15% F-measure.

6.3.4 Rule Induction

Figure 6.2 shows the nested logic in performing a wrapper-type variable selection method. The logic nested contains three different levels of process, which are: the main process, the learning process and the training and testing process. Figure 6.4 shows the workflow when implementing the logic of the main and the learning processes. To apply a wrapper-type feature selection such as GA and PSO using a Rule Induction classifier with default and optimized parameters, we need to setup an inner process of the X-Validation operator in Workflow 6.4. The inner process of the X-Validation operator contains a training section and a testing section as shown in Figure 6.8. The training section sets a learning algorithm for classifying an example set. We added a *Rule Induction* operator in the training section. The testing section examines the performance of the learned model in the cross validation manner. For the current problem, the process consists of two operators: *Apply Model* and *Performance*. The *Apply Model* operator applies the Rule Induction classification model on the testing dataset and delivers the resultant labelled dataset as the output. This is then used by the *Performance* operator to evaluate the model and to generate a performance vector containing information about various performance criteria.

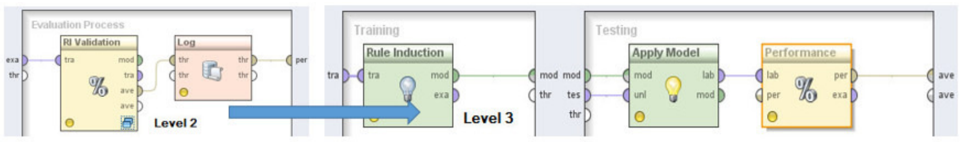


FIGURE 6.8: The Workflow for Learning Process using Rule Induction

Table 6.11 illustrates the results of wrapper-type feature selection in five different variable sets (IMDb, IMDb++, FilmAffinity, MovieMeter, & RottenTomatoes) using a Rule Induction classification algorithm with default parameters. PSO dominates in achieving

Table 6.11: The Result of Wrapper-Type Feature Selection using Rule Induction Classifier with Default Parameters

Datasets	Selected Features	Mtd	#
IMDb (16)	composer, cinematographer, genre, producer, mpaa, budget, director, production company, distributor, actor, product designer, competition, costume designer	PSO	13
IMDb++ (23)	budget, competition, director, metascore, mpaa, golden globe nominee, production company, product designer, golden globe won, oscar won	GA	10
FA (23)	costume designer, director, production company, writer, metascore, golden globe nominee, mpaa, opening gross, actor, producer, golden globe won, oscar nominee, competition, cinematographer	PSO	14
MM (23)	opening theaters, cinematographer, costume designer, distributor, editor, writer, golden globe nominee, opening gross, genre, production company, product designer, budget, golden globe won, actress, composer, actor, director, competition, oscar nominee, producer, metascore	PSO	21
RT (23)	cinematographer, composer, costume designer, director, editor, production company, product designer, producer, writer, metascore, oscar nominee	PSO	11

the best results of feature selection process than GA, found in four datasets, while GA is only superior in the IMDB++ dataset. Nevertheless, GA achieved the largest number of attribute reduction by 13 in the IMDb++ dataset, while PSO reached the best one by 12 in the RottenTomatoes dataset. In the RottenTomatoes dataset, PSO was only able to eliminate two irrelevant attributes. Interestingly, GA produced the reduced IMDb++ dataset with the proportion of the original attributes and the additional external attributes by 6:4. Those external attributes are *metascore*, *golden globe nominee*, *golden globe won*, and *oscar won*. On the other hand, the proportion yielded by PSO in the reduced RottenTomatoes dataset is 9:2, in which the additional external attributes are *metascore* and *oscar nominee*.

We trained the five reduced datasets, as displayed in Table 6.11, using five different classifiers with default parameters. Table H.11, H.13, H.15, H.17, and H.19 in Appendix H demonstrates confusion matrixs of RI classifier with default parameters when applied to reduced attibutes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Those confusion matrixs were able to visualize missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of RI classifier with default parameters when applied to those five reduced datasets as shown in Table 6.12. The table present accuracy, precision, recall, and f-measure value for each dataset.

The result presented in Table 6.12 shows that by augmenting the original dataset (IMDb) with additional attributes from external data sources (IMDb++ dataset) demonstrates that even in using the reduced datasets by selecting several key features, IMDb++ dataset can still improve on the level of accurate classification of the IMDb dataset.

TABLE 6.12: The Movie Ratings Prediction Model Using Rule Induction Default Parameters with Feature Selection

Datasets	#Attr	10-Fold Cross Validation				Diff
		Accuracy	Precision	Recall	F-Measure	
IMDb	13	90.14	89.21	77.85	83.14	
IMDb++	10	90.59	90.24	79.98	84.80	1.66
FilmAffinity	14	90.92	89.62	79.30	84.15	1.01
MovieMeter	21	91.00	88.34	65.29	75.09	8.05
RottenTomatoes	11	91.57	85.86	71.62	78.10	5.04

IMDb++ model provided the improvements of accuracy, precision, recall, and F-Measures values of IMDb model by 0.45% (90.14% to 90.59%), 1.03% (89.21% to 90.24%), 2.13% (77.85% to 79.98%), and 1.66% (83.14% to 84.80%), respectively. Meanwhile, when RI classifier with default parameters applied to the other reduced-datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity model attained 90.92% accuracy, 89.62% precision, 79.30% recall and 84.15% F-measure. MovieMeter model yielded 91.00% accuracy, 88.34% precision, 65.29% recall and 75.09% F-measure, while RottenTomatoes model achieved 91.57% accuracy, 85.86% precision, 71.62% recall and 78.10% F-measure.

Table 6.13 shows result of the wrapper-type feature selection using the RI classification algorithm with optimized parameters. We applied a set of optimal hyper-parameters for RI classifier as attained in Table 5.14 Chapter 5. The figure indicates that feature selection using GA outperformed PSO in reducing unimportant attributes in all datasets. Subsequently, we trained the reduced-datasets appeared in Table 6.13 using RI classifier with optimized parameters. Table H.12, H.14, H.16, H.18, and H.20 in Appendix H demonstrates confusion matrixs of RI classifier with optimized parameters when applied to reduced-attibutes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Those confusion matrixs were able to visualize missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of RI classifier with optimized parameters when applied to those five reduced datasets as shown in Table 6.14. The table presents accuracy, precision, recall, and f-measure value for each dataset.

The result presented in Table 6.14 indicates by augmenting the original dataset (IMDb) with additional attributes from external data sources demonstrates that even in using the reduced-datasets by selecting several key features, IMDb++ dataset can still improve on the level of accurate classification of the IMDb dataset. IMDb++ model provided the improvements of accuracy, precision, recall, and F-Measures values of IMDb model by 0.33% (97.93% to 98.26%), 0.25% (96.33% to 96.58%), 0.30% (96.63% to 96.93%), and 0.27% (96.48% to 96.75%), respectively. Meanwhile, when RI classifier with optimized parameters applied to the other reduced-datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity model attained 97.79% accuracy, 96.37% precision, 95.24%

Table 6.13: The Result of Wrapper-Type Feature Selection using Rule Induction Classifier with Optimized Parameters

Datasets	Selected Features	Mtd	#
IMDb (16)	actress, director, distributor, producer, writer	GA	5
IMDb++ (23)	actress, budget, composer, director, editor, metascore, opening gross, oscar nominee, production company, product designer, producer	GA	11
FA (23)	actor, composer, costume designer, director, genre, golden globe nominee, metascore, mpaa, oscar nominee, oscar won, production company, product designer, producer, writer	GA	14
MM (23)	budget, cinematographer, competition, composer, director, distributor, opening gross, product designer, producer, writer	GA	10
RT (23)	actor, budget, cinematographer, competition, composer, director, opening gross, production company, product designer, producer, writer	GA	11

Table 6.14: The Movie Ratings Prediction Model Using Rule Induction Optimized Parameters with Feature Selection

Datasets	#Attr	10-Fold Cross Validation				Diff
		Accuracy	Precision	Recall	F-Measure	
IMDb	5	97.93	96.33	96.63	96.48	
IMDb++	12	98.26	96.58	96.93	96.75	0.27
FilmAffinity	14	97.79	96.37	95.24	95.81	0.67
MovieMeter	10	98.13	96.38	94.79	95.58	0.90
RottenTomatoes	11	98.30	95.28	95.71	95.49	0.99

recall and 95.81% F-measure. MovieMeter model yielded 98.13% accuracy, 96.38% precision, 94.79% recall and 95.58% F-measure, while RottenTomatoes model achieved 98.30% accuracy, 95.28% precision, 95.71% recall and 95.49% F-measure.

6.3.5 Support Vector Machine (SVM)

Figure 6.2 shows a scheme in performing the wrapper-type variable selection method. As it can be seen, the inner process of the learning process is a training environment to build models and a model evaluation environment to test the model. To implement a wrapper-type feature selection such as GA and PSO using a SVM classifier, we need to setup an inner process of the X-Validation operator in Workflow 6.4. The inner process of the X-Validation operator contains a training section and a testing section as shown in Figure 6.9. The training section sets a classification algorithm for classifying a variable set, and we added a *SVM* operator to this section. The testing section examines the performance of the learned model in a cross validation manner. For the current problem, the process consists of two operators: *Apply Model* and *Performance*. The *Apply Model* operator applies the SVM classification model to the testing dataset and delivers the resultant labelled dataset as the output, which is used by the *Performance* operator to evaluate the model and to generate a performance vector containing information about various performance criteria.

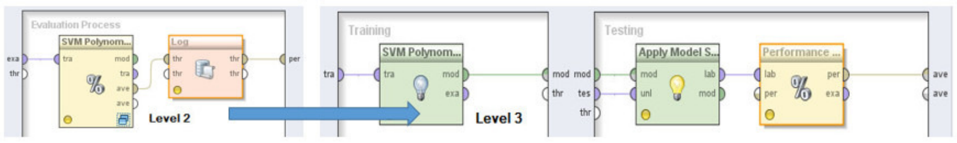

Evaluation Process
SVM Polynom...
Log
Level 2
Training
SVM Polynom...
Level 3
Testing
Apply Model S...
Performance ...
exa thr tra mod tra ave ave thr thr thr thr per
tra tra mod exa mod thr mod tes thr
mod unl lab mod lab per per exa ave ave

FIGURE 6.9: The Workflow for Learning Process using SVM

TABLE 6.15: The Results of Wrapper-Type Feature Selection with SVM Polynomial Kernel Classifier

Datasets	Selected Features	Mtd	#
IMDb (16)	actor, actress, composer, director, distributor, editor, producer, writer, budget, competition, mpaa	GA	11
IMDb++ (23)	opening theaters, actor, actress, cinematographer, composer, director, distributor, editor, producer, writer, metascore, oscar nominee, budget, competition, mpaa, opening gross	GA	16
FA (23)	opening theaters, actor, composer, director, distributor, editor, product designer, producer, writer, oscar won, golden globe nominee, budget, competition, mpaa	GA	14
MM (23)	composer, director, distributor, product designer, producer, writer, metascore, oscar won, oscar nominee, golden globe won, budget, mpaa, opening gross	GA	13
RT (23)	actor, director, distributor, editor, production company, producer, writer, metascore, oscar nominee, golden globe won, budget, competition, mpaa, opening gross	GA	14

Table 6.15 indicates the results of wrapper-type feature selection using SVM classification algorithm with default parameters. SVM optimization using GA is superior than PSO to remove independent attributes that may be strongly uncorrelated to the predicted or dependent attribute. GA achieved the lowest of FF (Fraction of Features) by 56.52% in MovieMeter dataset, while the highest of FF at 68.75% was obtained by GA in the IMDb dataset. Therefore, GA reduced the number of attributes more so in the MovieMeter dataset than in the IMDb dataset. The average FF of GA is 63.16%. Amongst the four non-IMDb datasets, IMDb++, MovieMeter and RottenTomatoes preserves four of seven additional attributes from external sources, while FilmAffinity only includes three attributes. For instance, RottenTomatoes consists of the following external attributes: *metascore*, *oscar nominee*, *golden globe won*, and *opening gross*, while FilmAffinity is comprised of *opening theaters*, *oscar won* and *golden globe nominee*. Among the seven additional external attributes, *metascore* is always present in IMDb++, MovieMeter, and RottenTomatoes datasets and is only absent in the FilmAffinity dataset.

We trained the five reduced datasets, as displayed in Table 6.15, using five different classifiers with default parameters. Table J.11, J.13, J.15, J.17, and J.19 in Appendix J demonstrates confusion matrixs of SVM Polynomial classifier with default parameters when applied to reduced attibutes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Those confusion matrixs were able to visualize

Table 6.16: The Movie Ratings Prediction Model Using SVM Polynomial Default Parameters with Feature Selection

Datasets	#Attr	10-Fold Cross Validation				Diff
		Accuracy	Precision	Recall	F-Measure	
IMDb	11	84.80	81.31	65.64	73.42	
IMDb++	16	85.89	82.94	69.99	75.92	2.50
FilmAffinity	14	84.40	83.61	63.61	72.25	1.17
MovieMeter	13	84.86	65.74	40.75	50.32	23.10
RottenTomatoes	14	87.34	61.75	51.52	56.17	17.25

missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of SVM Polynomial classifier with default parameters when applied to those five reduced datasets as shown in Table 6.16. The table present accuracy, precision, recall, and f-measure value for each dataset.

The result presented in Table 6.16 shows that by augmenting the original dataset (IMDb) with additional attributes from external data sources (IMDb++) demonstrates that even in using the reduced datasets by selecting several key features, IMDb++ dataset can still improve on the level of accurate classification of the IMDb dataset. IMDb++ model provided the improvements of accuracy, precision, recall, and F-Measures values of IMDb model by 1.09% (84.80% to 85.89%), 1.63% (81.31% to 82.94%), 4.35% (65.64% to 69.99%), and 2.50% (73.42% to 75.92%), respectively. Meanwhile, when SVM Polynomial classifier with default parameters applied to the other reduced-datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity model attained 84.40% accuracy, 83.61% precision, 63.61% recall and 72.25% F-measure. MovieMeter model yielded 84.86% accuracy, 65.74% precision, 40.75% recall and 50.32% F-measure, while RottenTomatoes model achieved 87.34% accuracy, 61.75% precision, 51.52% recall and 56.17% F-measure.

Table 6.17 shows result of the wrapper-type feature selection using SVM Polynomial classification algorithm with optimized parameters. We applied a set of optimal hyper-parameters for SVM classifier as attained in Table 5.16 Chapter 5. The figure indicates that feature selection using GA outperformed PSO in reducing unimportant attributes in all datasets. Subsequently, we trained the reduced-datasets appeared in Table 6.17 using SVM Polynomial classifier with optimized parameters. Table J.12, J.14, J.16, J.18, and J.20 in Appendix J demonstrates confusion matrixs of SVM Polynomial classifier with optimized parameters when applied to reduced-attibutes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Those confusion matrixs were able to visualize missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of SVM Polynomial classifier with optimized parameters when applied to those five reduced datasets as shown in Table 6.18. The table presents accuracy, precision, recall, and f-measure value for each dataset.

TABLE 6.17: The Result of Wrapper-Type Feature Selection using SVM Polynomial Classifier with Optimized Parameters

Datasets	Selected Features	Mtd	#
IMDb (16)	actor, actress, budget, cinematographer, competition, composer, costume designer, director, distributor, editor, mpaa, production company, product designer, producer, writer	GA	15
IMDb++ (23)	actress, budget, cinematographer, competition, composer, costume designer, director, distributor, editor, metascore, mpaa, opening gross, opening theaters, oscar nominee, production company, product designer, producer, writer	GA	18
FA (23)	actress, budget, cinematographer, competition, composer, costume designer, director, distributor, editor, metascore, mpaa, opening gross, opening theaters, oscar nominee, oscar won, production company, product designer, producer, writer	GA	19
MM (23)	actress, budget, cinematographer, competition, composer, costume designer, director, distributor, editor, golde globe won, metascore, mpaa, opening gross, opening theaters, oscar nominee, production company, product designer, producer, writer	GA	19
RT (23)	actors, actress, budget, cinematographer, competition, composer, costume designer, director, distributor, editor, golde globe nominee, metascore, mpaa, opening gross, opening theaters, oscar nominee, production company, product designer, producer, writer	GA	20

TABLE 6.18: The Movie Ratings Prediction Model Using SVM Polynomial Optimized Parameters with Feature Selection

Datasets	#Attr	10-Fold Cross Validation				Diff
		Accuracy	Precision	Recall	F-Measure	
IMDb	15	92.44	90.91	87.97	89.41	
IMDb++	18	94.97	93.31	92.11	92.71	3.33
FilmAffinity	19	94.08	92.61	91.68	92.14	2.73
MovieMeter	19	93.50	90.76	89.69	90.22	0.81
RottenTomatoes	20	95.45	93.74	92.91	93.32	3.91

The result presented in Table 6.18 indicates by augmenting the original dataset (IMDb) with additional attributes from external data sources demonstrates that even in using the reduced-datasets by selecting several key features, IMDb++ dataset can still improve on the level of accurate classification of the IMDb dataset. IMDb++ model provided the improvements of accuracy, precision, recall, and F-Measures values of IMDb model by 2.53% (92.44% to 94.97%), 2.40% (90.91% to 93.31%), 4.14% (87.97% to 92.11%), and 3.33% (89.41% to 92.71%), respectively. Meanwhile, when SVM Polynomial classifier with optimized parameters applied to the other reduced-datasets (FilmAffinity, MovieMeter, and RottenTomatoes), FilmAffinity model attained 94.08% accuracy, 92.61% precision, 91.68% recall and 92.14% F-measure. MovieMeter model yielded 93.50% accuracy, 90.76% precision, 89.69% recall and 90.22% F-measure, while RottenTomatoes model achieved 95.45% accuracy, 93.74% precision, 92.91% recall and 93.32% F-measure.

6.3.6 Results Comparison of Classification Models with Feature Selection

The experiment results are summarised in Table 6.19 to serve as a clear comparison of the performance of wrapper-type feature selection using GA and PSO tuned to classifiers either with default or optimized parameters. Out of 25 classification models built using classifiers with default parameters, GA was used to select the most important features of 11 models, while the further 14 models used PSO to filter their most important attributes. PSO produced the highest as well as the lowest of FF values of 95.65% and 37.50%, respectively. PSO fitted to kNN classifier with default parameters reduced the number of attributes of the IMDb dataset from 16 to 6 (FF = 37.5%). This was the greatest number of attribute reduction amongst classifiers with default parameters. PSO tailored to to ANN and DT classifiers resulted in the smallest number of attribute reduction in IMDb++ and RottenTomatoes datasets. It was only able to reduce one attribute in each dataset. Meanwhile, GA was superior than PSO when tuned to classifers with optimized parameters. Indeed, the use of GA with RI optimized parameters attained the lowest of FF value by 31.25%. It succesfully reduced the number of attributes of IMDb dataset from 16 to 5. Overall, the use of wrapper-type feature selection tuned to classifiers with optimized parameters yielded smaller average FF compared to classifiers with default parameters. GA tuned to ANN classifier with optimized parameters declined FF average of the GA and PSO tailored to ANN classifier with default parameters by 12,93% (90.05% to 77.12%). GA tuned to DT learner with optimized parameters reduced FF average of the GA and PSO fitted to the one with default parameters by 16.30% (74.89% to 58.59%). GA tuned to RI learner with optimized parameters dropped FF average of the GA and PSO tuned to the one with default parameters by 19.25% (65.34% to 46.09%). This suggests that wrapper-type feature selection tuned to classifiers with optimized parameters has ability to reduce more unimportant attributes than the ones with default parameters. Also, it can be noted that GA is more fit to use with classifiers with optimized parameters than those with default parameters to perform wrapper-type feature selection.

Table 6.20 presents classification results of classifiers with default and optimized parameters when applied to reduced-attributes of IMDb and IMDb++ datasets. While IMDb is the model solely based upon a single data source, namely IMDb, IMDb++ is based on IMDb and external data sources. The reduced-attributes of both datasets were generated by applying wrapper-type feature selection suited to classifiers with default and optimized parameters. When applied to classifiers with default parameters, mostly, reduced-attributes IMDb++ models attained better accuracy, precision, recall, and F-Measure values than reduced-attributes IMDb modesl. However, a reduced-attributes IMDb++ model built by Decision Tree classifier with default parameters suffered declining values of recall and F-Measure by 1.57% and 0.31%, respectively. IMDb model with reduced-attributes built by Decision Tree learner achieved 72.93% and 77.07% recall and

TABLE 6.19: The Comparison of Feature Selection Results using GA and PSO

Full-Attributes Datasets: IMDb (16), IMDb++ (23), FilmAffinity (23), MovieMeter (23) & RottenTomatoes (23)											
Datasets		**ANN**		**DT**		**k-NN**		**RI**		**SVM**	
		# Attr	**FF (%)**	**# Attr**	**FF (%)**	**# Attr**	**FF (%)**	**# Attr**	**FF (%)**	**# Attr**	**FF (%)**
DP	**GA**										
	IMDb									11	68.75
	IMDb++							10	45.45	16	69.57
	FA	19	82.61	11	47.83	12	52.17			14	60.87
	MM	19	82.61	11	47.83					13	56.52
	RT									14	60.87
	PSO										
	IMDb	15	93.75	14	87.50	6	37.50	13	81.25		
	IMDb++	22	95.65	22	95.65	13	56.52				
	FA							14	60.87		
	MM					18	78.26	21	91.30		
	RT	22	95.65	22	95.65	11	47.83	11	47.83		
Average FF			**90.05**		**74.89**		**54.46**		**65.34**		**63.32**
OP	**GA**										
	IMDb	13	81.25	10	62.50			5	31.25	15	93.75
	IMDb++	17	73.91	13	56.52			11	47.43	18	78.26
	FA	18	78.26	14	60.87			14	60.87	19	82.61
	MM	19	82.61	13	56.52			10	43.48	19	82.61
	RT	16	69.57	13	56.52			11	47.43	20	86.96
Average FF			**77.12**		**58.59**				**46.09**		**84.84**

F-Measure, respectively, while IMDb++ with reduced-attributes yielded a model recall of 71.36% and F-Measure of 76.76%. Meanwhile, when applied to classifiers (ANN, DT, RI, and SVM) with optimized parameters, IMDb++ models have higher values of accuracy, precision, recall, and F-Measure compared to IMDb models. The result presented in Table 5.17 (Chapter 5) and Table 6.20 (Chapter 6) indicate two important facts. Firstly. either using full-attributes of default-attributes, classifiers with optimized parameters generates IMDb++ models which are better in all performance measures (accuracy, precision, recall, and F-Measures) than IMDb models. Secondly, by using decision tree with default parameters, IMDb++ model only failed to improve performances (recall and F-Measure) of IMDb model when applied to either full or reduced attributes. Thirdly, the best F-Measure value was achieved by IMDb++ model built using Decisio Tree with optimized parameters. Full-attributes IMDb++ obtained 96.75% of F-Measure value, while Reduced-attributes IMDb++ yielded a model F-Measure of 97.61%.

Table 6.20: Comparisons of Wrapper-Type Feature Selection Suited to Classifiers with Default and Optimized Parameters When Applied to IMDb and IMDb++ Datasets

Classifiers	**Datasets**	**Accuracy**	**Precision**	**Recall**	**F-Measure**
Default Parameters					

Continued on next page

Table 6.20 – continued from previous page

Classifiers	**Datasets**	**Accuracy**		**Precision**		**Recall**		**F-Measure**	
ANN	IMDb	86.59	1.93	82.05	2.59	74.91	3.56	78.32	3.12
	IMDb++	88.52		84.64		78.47		81.44	
DT	IMDb	84.40	0.63	81.71	1.35	72.93	1.57	77.07	0.31
	IMDb++	85.03		83.06		71.36		76.76	
k-NN	IMDb	97.84	0.34	96.11	1.13	95.91	0.28	96.01	0.70
	IMDb++	98.18		97.24		96.19		96.71	
RI	IMDb	90.14	0.45	89.21	1.03	77.85	2.13	83.14	1.64
	IMDb++	90.59		90.24		79.98		84.80	
SVM	IMDb	84.80	1.09	81.31	1.63	65.64	4.35	73.42	2.50
	IMDb++	85.89		82.94		69.99		75.92	
Optimized Parameters									
ANN	IMDb	86.46	1.64	83.21	1.66	73.33	3.33	77.96	2.60
	IMDb++	88.10		84.87		76.66		80.56	
DT	IMDb	98.47	0.06	97.73	0.07	97.40	0.02	97.56	0.05
	IMDb++	98.53		97.80		97.42		97.61	
RI	IMDb	97.93	0.33	96.33	0.25	96.63	0.60	96.48	0.27
	IMDb++	98.26		96.58		96.93		96.75	
SVM	IMDb	92.44	2.53	90.91	2.40	87.97	4.14	89.41	3.30
	IMDb++	94.97		93.31		92.11		92.71	

Table 6.22 summarizes classification results of four classifiers (ANN, Decision Tree, Rule Induction, and SVM Polynomial) either using default or optimized parameters when applied to reduced-attributes datasets generated from wrapper-type feature selection processes. The figures show that wrapper-type feature selection suited to Decision Tree, Rule Induction, and SVM Polynomial with optimized parameters have succesfully improves performances the ones with default parameters. However, the use of ANN with optimized parameters yielded less better results compared to ANN with default parameters when applied to all datasets with reduced-attributes. When applied to reduced-attributes IMDb model (a model solely based on a single data source), ANN classifier with optimized parameters suffered declining 0.13% (86.59% to 86.46%), 1.58% (74.91% to 73.33%), and 0.36% (78.32% to 77.96%) accuracy, recall, and F-Measure values, respectively, while precision value improved 1.16% (82.05% to 83.21%). To reduced-attributes IMDb++ (a model based on IMDb and external data sources), ANN with optimized parameters experienced decreasing accuracy of 0.42% (88.52% to 88.10%), recall of 1.81% (78.47% to 76.66%), and F-Measure of 0.88% (81.44% to 80.56%), while there was an increase of precision value by 0.23% (84.64% to 84.87%). ANN with optimization processes also failed to improve performances of ANN with default parameters when applied to the other reduced-attributes datasets (FilmAffinity, MovieMeter, and RottenTomatoes). While accuracy, recall, and F-Measure values declined 0.22% (86.85%

to 86.63%), 0.40% (74.12% to 73.72%), and 0.01% (77.91% to 77.90%), respectively, precision value increased 0.49% (82.10% to 82.59%), when ANN with optimzed parameters applied to reduced-attributes FilmAffinity dataset. To reduced-attributes Moviemeter, ANN with optimized parameters dropped at recall and F-Measure values by 3.32% (71.49% to 68.17%), and 1.13% (75.18% to 74.05%), respectively, while accuracy and precision increased 0.33% (88.04% to 88.37%) and 1.77% (79.28% to 81.05%). Lastly, ANN with optimized parameters experienced declining in all performance measures (accuracy, precision, recall and F-Measure) by 0.75% (89.81% to 89.06%), 1.16% (75.75% to 74.59%), 7.48% (67.77% to 60.29%), and 4.86% (71.54% to 66.68%), respectively, when applied to reduced-attributes RottenTomatoes dataset.

The result presented in Table 6.22 shows that DT classifier with optimized parameters has succesfully improved the one with default parameters in all performance measures when applied to all reduced-attributes datasets. In a model solely based on a single data source (IMDb), improvements reached 14.07% accuracy (from 84.40% to 98.47%), 16.02% precision (81.71% to 97.73%), 24.47% recall (72.93% to 97.40%), and 20.49% F-Measure (77.07% to 97.56%). DT learner with optimized parameters enriched accuracy, precision, recall, and F-Measure values by 13.50% (85.03 to 98.53%), 14.74% (83.06% to 97.80%), 26.06% (71.36% to 97.42%) , and 20.85% F-Measure (76.76% to 97.61%), respectively, when applied to IMDb++ (a model based on IMDb and external data sources). FilmAffinity model attained improvements 16.13% (82.18% to 98.31%), 8.11% (88.80% to 96.91%), 41.90% (54.81% to 96.71%), and 29.03% (67.78% to 96.81%) accuracy, precision, recall, and F-Measure, respectively. MovieMeter has enrichment accuracy of 25.14% (73.10% to 98.24%), precision of 77.32% (18.28% to 95.60%), recall of 69.53% (25.00% to 94.53%), and F-Measure of 73.94% (21.12% to 95.06%), while RottenTomatoes has enhancement 10.91% (87.53% to 98.44%), 20.89% (72.92% to 93.81%), 38.18% (58.05% to 96.23%), and 30.37% (64.64% to 95.01%) for accuracy, precision, recall, and F-Measure values, respectively.

As indicated in Table 6.22, the use of RI with optimized parameters have successfully improved all performance measurements of RI with default parameters when applied to all reduced-attributes datasets (IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes). The advancement of IMDb model reached 7.79% accuracy (from 90.14% to 97.93%), 7.12% precision (89.21% to 96.33%), 18.78% recall (77.85% to 96.63%), and 13.34% F-Measure (83.14% to 96.48%). The improvements of IMDb++ (a model based on IMDb and external data sources) achieved 7.67% (90.59 to 98.26%) accuracy, 6.34% (90.24% to 96.58%) precision, 16.95% (79.98% to 96.93%) recall, and 11.95% F-Measure (84.80% to 96.75%). FilmAffinity model attained improvements 6.87% (90.92% to 97.79%), 6.75% (89.62% to 96.37%), 15.94% (79.30% to 95.24%), and 12.60% (84.15% to 96.75%) accuracy, precision, recall, and F-Measure, respectively. MovieMeter has enrichment accuracy of 7.13% (91.00% to 98.13%), precision of 8.04% (88.34% to 96.38%), recall of 29.50% (65.29% to 94.79%), and F-Measure of 20.49% (75.09% to 95.58%), while RottenTomatoes has enhancement 6.73% (86.82% to 98.30%), 9.42% (61.52% to 95.28%), 24.09% (50.13% to 95.71%), and 17.39% (55.25% to 95.49%) for accuracy, precision, recall, and F-Measure values, respectively.

The result presented in Table 6.22 shows that SVM Polynomial with optimized parameters has improved all models generated by SVM Polynomial with default parameters in all performance measures when applied to reduced-attributes datasets (IMDB, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes). In a model solely based on a single data source (IMDb), the improvement reached 6.91% accuracy (from 84.23% to 91.14%), 6.64% precision (81.35% to 87.99%), 19.76% recall (65.41% to 85.17%), and 14.04% F-Measure (72.52% to 86.56%). Improvements of accuracy of 8.67% (85.26 to 93.93%), precision of 9.06% (81.50% to 90.56%) , recall of 20.41% (68.81% to 89.22%) , and 15.26% F-Measure (74.62% to 89.88%) were achieved for IMDb++ (a model based on IMDb and external data sources). FilmAffinity model attained improvements 9.68% (84.40% to 94.08%), 9.00% (83.61% to 92.61%), 28.07% (63.61% to 91.68%), and 19.89% (72.25% to 92.14%) accuracy, precision, recall, and F-Measure, respectively. MovieMeter has enrichment accuracy of 8.64% (84.86% to 93.50%), precision of 25.02% (65.74% to 90.76%), recall of 48.94% (40.75% to 89.69%), and F-Measure of 39.90% (50.32% to 90.22%), while RottenTomatoes has enhancement 8.11% (87.34% to 95.45%), 31.99% (61.75% to 93.74%), 41.39% (51.52% to 92.91%), and 37.16% (56.17% to 93.33%) for accuracy, precision, recall, and F-Measure values, respectively.

Table 6.23 summarizes classification results of classifiers either using default or optimized parameters when applied to full and reduced atttributes datasets. Classification processes either using default or optimized parameters applied to full-attributes datasets were found in Chapter 5, while Chapter 6 presents the ones either using default or optimized parameters when applied to reduced-attributes datasets. The figures indicate that wrapper-type feature selection suited to ANN classifier with default parameters have better classification performance (F-Measure value) when applied to datasets with reduced-attributes than full-attributes. The improvements were found in 3 datasets: IMDb, IMDb++, and RottenTomatoes by 0.12% (78.20% to 78.32%), 0.61% (80.83% to 81.44%), and 0.05% (71.49% to 71.54%), respectively. However, ANN with default parameters when applied to reduced-attributes datasets does not always perform better than using full-attributes datasets. The declining of F-Measure values were found in 2 datasets: FilmAffinity (dropped 0.65%, from 78.56% to 77.91%) and MovieMeter (75.77% to 75.18%). Indeed, there were none improvements of F-Measure values when ANN with optimized parameters applied to all reduced-attributes datasets. Overall, ANN achieved the best performance when using optimized parameters and applied to full-attributes IMDb dataset. This resulted in 81.45% of F-Measure.

It can be seen in Table 6.23 that either using default or optimized parameters, Decision Tree performed better in reduced-attributes datasets than full-attributes datasets. However, it was found two cases in using DT to classify reduced-attributes datasets. DT with default parameters was only able to maintain accuracy, precision, recall, and F-Measure values of the full-attributes MovieMeter and RottenTomatoes datasets, while DT with optimized parameters failed to improve precision and F-Measure values by 2% (95.81% to 93.81%) and 0.49% (95.50% to 95.01%), respectively when applied to RottenTomatoes dataset with less attributes. In using Decision Tree to build movie classification models, the highest F-Measure value was achieved by appling a set of optimal parameters to reduced-attributes IMDb++ dataset. A F-Measure of 97.61% was achieved.

Table 6.23 shows that the application of k-NN classifier with a default parameter to reduced-attributes datasets outperformed performance of k-NN when applied to full-attributes datasets. By using reduced-attributes IMDb dataset, k-NN achieved an increase in F-Measure of full-attributes IMDb model by 9.68% (86.33% to 96.01%). In reduced-attributes IMDb++ dataset, k-NN improved F-Measure of full-attributes IMDb++ model by 10.38% (86.33% to 96.71%). In reduced-attributes datasets (FilmAffinity, MovieMeter, and RottenTomatoes), k-NN improved F-Measure values of the full-attributes FilmAffinity, MovieMeter, and RottenTomatoes models by 11.95% (84.49% to 96.44%), 11.35% (83.16% to 94.51%), and 9.32% (83.83% to 93.15%), respectively. k-NN classifier achieved the best performance when applied to IMDb++ reduced dataset. This obtained F-Measure 96.71%.

The results presented in Table 6.23 shows that the use of RI classifier with default parameters to all reduced-attributes datasets have better classification performance than using full-attributes datasets. The application of RI with optimized parameters to reduced-attributes datasets (IMDb, IMDb++, MovieMeter, and RottenTomatoes) were able to improve classification performance of those with full-attributes. It was only in reduced-attributes FilmAffinity dataset, RI with optimized parameters failed to improve classification results using full-attributes FilmAffinity dataset. The use of reduced-attributes FilmAffinity dataset dropped accuracy, precision, recall, and F-Measure values of RI wtih optimized parameters applied to full-attributes by 0.48% (98.27% to 97.79%), 0.53% (96.90% to 96.37%), 0.96% (96.20% to 95.24%), and 0.74% (96.55% to 95.81%), respectively. Overall, the best F-Measure achieved using RI classifier was to use optimized parameters applied to reduced-attributes IMDb++ dataset. This attained F-Measure 96.75%.

According to figures presented in Table 6.22 and 6.23, it can be summarized two interesting facts. Firstly, the use of wrapper-type feature selection either using GA or PSO is more likely to improve classification performance, however, it does not always to improve performance of classifiers. Secondly, the use of classifiers (DT, RI, and SVM) with optimized parameters are more likely to result in more promising results when applied to reduced-attributes datasets. All the models yielded higher accuracy, precision, recall, and F-Measure values compared to results of those with default parameters. Based on results indicated in Table 6.22 and 6.23, it can also be used to consider the best performing model in classifying movie popularities. It can be perceived that Decision Tree with optimized parameters applied to the reduced-attributes IMDb++ dataset attained the highest accuracy, precision, recall, and F-Measure values by 98.53%, 97.80%, 97.42%, and 97.61%, respectively. Two previous researchers, Sehgal et al. (2012) and Moore et al. (2009), have their own arguments related with the best performing model. Sehgal et al. (2012) argue that the best performing model is the model that can provide higher accuracy, precision, and recall values. Moore et al. (2009) consider higher F-Measure value means better model performance. The reduced-attributes IMDb++ model are inline with Sehgal et al. (2012) and Moore et al. (2009) recommendations, as as it has provided the highest accuracy, recall, and F-Measure values. Therefore, it can be said that IMDb++ as the best movie classification model among all models presented in Table 6.23.

Table 6.21 shows selected attributes as well their sources for the reduced-attributes IMDb++ model. This model consists 13 attributes that generated by applying wrapper-type feature selection using Genetic Algorithm suited to DT classifier with optimized parameters. IMDb List Files constitute downloadable files from IMDb website that are possible to convert into a single database. Following convert those files into a database, the database is used as a primary data source. Opening Gross and Opening Theaters are additional attributes that come from BoxOfficeMojo, while Oscar Nominee as an additional attribute is extracted from IMDb website using web scraping technique. The inclusion of external data such as: opening gross, opening theaters, and oscar nominee in the classification model enriched accuracy, precision, recall, and F-Measure values by 0.29%, 1.01%, 0.70%, and 0.86%, respectively compared to the reduced-attributes IMDb model (model solely based upon a single data source). Moreover, the inclusion of additional attributes may lead to new insights.

Table 6.21: Selected Attributes of The Reduced-Attributes IMDb++ Model Generated By Wrapper-Type Feature Selection Using GA Suited to DT with Optimized Parameters

No	**Attribute Name**	**Data Sources**	**No**	**Attribute Name**	**Data Sources**
1	Budget	IMDb List Files	8	Opening Theaters	BoxOfficeMojo
2	Composer	IMDb List Files	9	Oscar Nominee	IMDb Website
3	Director	IMDb List Files	10	Production Company	IMDb List Files
4	Distributor	IMDb List Files	11	Production Designer	IMDb List Files
5	Editor	IMDb List Files	12	Producer	IMDb List Files
6	Mpaa	IMDb List Files	13	Writer	IMDb List Files
7	Opening Gross	BoxOfficeMojo			

The current study provides two evidences. Firstly, the best approach to build a classfiication model for classifying movie ratings is to employ wrapper-type feature selection using Genetic Algorithm suited to Decision Tree with optimized parameters. We set the four following DT parameters Criterion, Minimal Size for Split, Minimal Leaf Size, and Maximal Depth with following values gini index, 1, 1, and 29 as indicated in Table 5.12 in Chapter5. Those parameters were obtained by applying Grid Search algorithm for parameter optimization. It can be highlighted to get optimal classification results, hence, it is required to apply advance techniques in data mining. Those techniques are parameter optimization using Grid Search, wrapper-type feature selection using Genetic Algorithm, and Decision Tree classifeir with optimized parameters. Secondly, the inclusion of external data sources is able to improve the performance of classifiers in classifying movie ratings. DT learner with optimized parameters attained the best classification model when applied to the reduced-attributes IMDb++ dataset that is based on a single data source (IMDb) and external data sources.

6.4 Summary

This chapter presents the development of classification models for classifying movie popularities based on reduced-attributes datasets. This chapter presents experimentation 2 to address research question 3 regarding the use of wrapper-type feature selection to generate an improved movie popularities classification model. We investigated two wrapper-type feature selection algorithms, Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), respectively. For each algorithm, we suited it into five different classifiers (ANN, DT, k-NN, RI and SVM Polynomial) either using default and optimized parameters and applied it to five datasets (IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes). Results of wrapper-type feature selection processes are reduced-attributes datasets that expected to boost up classification performance. We used four performance measurements (accuracy, prediction, recall, and F-Measure) to analyse the impact of feature selection processes to the classification models. The result of the experimentation shows that wrapper-type feature selection using GA performed better than using PSO. Also, the result highly supports that wrapper-type feature selection suited to Decision Tree with optimized parameters is the best approach when applied to IMDb++ dataset (a dataset based on single data source and external data sources).

Therefore, the experimentation result highlights the importance of multiple approaches to boost up the performance of classifiers. Those approaches may be parameter optimization using Grid Search, wrapper-type feature selection using Genetic Algorithm, and classifiers with optimized parameters. Also, the result suggest that linked data technologies can be used to explore relevant data sources and to create additional variables to support the process of knowledge discovery. As a result, it would be achieved the improvement of classifier performances in classifying movie popularities.

Table 6.22: Summary of Classification Results of Classifiers with Default Parameters (DP) and Optimized Parameters (OP) When Applied to Reduced-Attributes Datasets

Datasets		Accuracy		Precision		Recall		F-Measure	
ANN Applied to Reduced-Attributes Datasets									
IMDb	DP	86.59	0.13	82.05	1.16	74.91	1.58	78.32	0.36
	OP	86.46		83.21		73.33		77.96	
IMDb++	DP	88.52	0.42	84.64	0.23	78.47	1.81	81.44	0.88
	OP	88.10		84.87		76.66		80.56	
Film Affinity	DP	86.85	0.22	82.10	0.49	74.12	0.40	77.91	0.01
	OP	86.63		82.59		73.72		77.90	
Movie Meter	DP	88.04	0.33	79.28	1.77	71.49	3.32	75.18	1.13
	OP	88.37		81.05		68.17		74.05	
Rotten Tomatoes	DP	89.81	0.75	75.75	1.16	67.77	7.48	71.54	4.86
	OP	89.06		74.59		60.29		66.68	
RI Applied to Reduced-Attributes Datasets									
IMDb	DP	90.14	7.79	89.21	7.12	77.85	18.78	83.14	13.34
	OP	97.93		96.33		96.63		96.48	
IMDb++	DP	90.59	7.67	90.24	6.34	79.98	16.95	84.80	11.95
	OP	98.26		96.58		96.93		96.75	
Film Affinity	DP	90.92	6.87	89.62	6.75	79.30	15.94	84.15	12.60
	OP	97.79		96.37		95.24		96.75	
Movie Meter	DP	91.00	7.13	88.34	8.04	65.29	29.50	75.09	20.49
	OP	98.13		96.38		94.79		95.58	
Rotten Tomatoes	DP	86.82	6.73	61.52	9.42	50.13	24.09	55.25	17.39
	OP	98.30		95.28		95.71		95.49	

Datasets		Accuracy		Precision		Recall		F-Measure	
DT Applied to Reduced-Attributes Datasets									
IMDb	DP	84.40	14.07	81.71	16.02	72.93	24.47	77.07	20.49
	OP	98.47		97.73		97.40		97.56	
IMDb++	DP	85.03	13.50	83.06	14.74	71.36	26.06	76.76	20.85
	OP	98.53		97.80		97.42		97.61	
Film Affinity	DP	82.18	16.13	88.80	8.11	54.81	41.90	67.78	29.03
	OP	98.31		96.91		96.71		96.81	
Movie Meter	DP	73.10	25.14	18.28	77.32	25.09	69.53	21.12	73.94
	OP	98.24		95.60		94.53		95.06	
Rotten Tomatoes	DP	87.53	10.91	72.92	20.89	58.05	38.18	64.64	30.37
	OP	98.44		93.81		96.23		95.01	
SVM Applied to Reduced-Attributes Datasets									
IMDb	DP	84.80	7.64	83.31	7.60	65.64	22.33	73.42	15.99
	OP	92.44		90.91		87.97		89.41	
IMDb++	DP	85.89	9.08	82.94	10.37	69.99	22.12	75.92	16.79
	OP	94.97		93.31		92.11		92.71	
Film Affinity	DP	84.40	9.68	83.61	9.00	63.61	28.07	72.25	19.89
	OP	94.08		92.61		91.68		92.14	
Movie Meter	DP	84.86	8.64	65.74	25.02	40.75	48.94	50.32	39.90
	OP	93.50		90.76		89.69		90.22	
Rotten Tomatoes	DP	87.34	8.11	61.75	31.99	51.52	41.39	56.17	37.16
	OP	95.45		93.74		92.91		93.33	

Table 6.23: Summary of Classification Results of Classifiers with Default and Optimized Parameters When Applied to Full-Attributes (FA) and Reduced-Attributes (RA) Datasets

Datasets		Accuracy		Precision		Recall		F-Measure	
ANN with Default Parameters									
IMDb	FA	86.58	0.01	81.85	0.20	74.86	0.05	78.20	0.12
	RA	86.59		82.05		74.91		78.32	
IMDb++	FA	88.32	0.20	83.57	1.07	78.26	0.21	80.83	0.61
	RA	88.52		84.64		78.47		81.44	
Film Affinity	FA	86.81	0.04	82.09	0.01	75.32	1.20	78.56	0.65
	RA	86.85		82.10		74.12		77.91	
Movie Meter	FA	88.01	0.03	80.35	1.07	71.32	0.17	75.77	0.59
	RA	88.04		79.28		71.49		75.18	
Rotten Tomatoes	FA	89.77	0.04	74.04	1.71	69.11	1.34	71.49	0.05
	RA	89.81		75.75		67.77		71.54	
DT with Default Parameters									
IMDb	FA	84.38	0.02	81.68	0.03	72.83	0.10	77.00	0.07
	RA	84.40		81.71		72.93		77.07	
IMDb++	FA	85.02	0.01	83.05	0.01	71.34	0.02	76.75	0.01
	RA	85.03		83.06		71.36		76.76	
Film Affinity	FA	82.01	0.17	88.21	0.59	54.32	17.04	67.24	9.52
	RA	82.18		88.80		71.36		76.76	
Movie Meter	FA	73.10	0.00	18.28	0.00	25.00	0.00	21.12	0.00
	RA	73.10		18.28		25.00		21.12	
Rotten Tomatoes	FA	87.53	0.00	72.92	0.00	58.05	0.00	64.64	0.00
	RA	87.53		72.92		58.05		64.64	
k-NN with a Default Parameter									
	FA	89.19		85.36		81.66		83.47	

Datasets		Accuracy		Precision		Recall		F-Measure	
ANN with Optimized Parameters									
IMDb	FA	86.60	0.14	82.06	1.15	75.40	2.07	78.59	0.63
	RA	86.46		83.21		73.33		77.96	
IMDb++	FA	88.57	0.47	84.86	0.01	78.31	1.65	81.45	0.89
	RA	88.10		84.87		76.66		80.56	
Film Affinity	FA	87.23	0.60	83.18	0.59	75.09	1.37	78.93	1.03
	RA	86.63		82.59		73.72		77.90	
Movie Meter	FA	88.04	0.33	79.28	1.77	71.49	3.32	75.18	1.13
	RA	88.37		81.05		68.17		74.05	
Rotten Tomatoes	FA	90.14	1.08	75.55	0.96	67.76	7.47	71.44	4.76
	RA	89.06		74.59		60.29		66.68	
DT with Optimized Parameters									
IMDb	FA	98.16	0.31	96.70	0.03	96.58	0.82	96.64	0.92
	RA	98.47		97.73		97.40		97.56	
IMDb++	FA	98.24	0.29	96.79	1.01	96.72	0.70	96.75	0.86
	RA	98.53		97.80		97.42		97.61	
Film Affinity	FA	98.02	0.29	96.78	0.13	96.15	0.56	96.47	0.34
	RA	98.31		96.91		96.71		96.81	
Movie Meter	FA	97.88	0.36	94.98	0.62	93.65	0.88	94.31	0.75
	RA	98.24		95.60		94.53		95.06	
Rotten Tomatoes	FA	98.04	0.40	95.81	2.00	95.19	1.04	95.50	0.49
	RA	98.44		93.81		96.23		95.01	

Continued on next page

Table 6.23 – continued from previous page

Datasets		Accuracy		Precision		Recall		F-Measure		Datasets		Accuracy		Precision		Recall		F-Measure	
IMDb	RA	97.84	8.65	96.11	10.75	95.91	14.25	96.01	12.54										
IMDb++	FA	91.19	6.99	88.44	8.80	84.32	11.87	86.33	10.38										
	RA	98.18		97.24		96.19		96.71											
Film Affinity	FA	89.69	8.35	86.51	10.58	82.55	13.24	84.49	11.95										
	RA	98.04		97.09		95.79		96.44											
Movie Meter	FA	90.64	7.20	86.42	9.69	80.13	12.84	83.16	11.35										
	RA	97.84		96.11		92.97		94.51											
Rotten Tomatoes	FA	91.73	6.46	84.87	7.17	82.81	11.48	83.83	9.32										
	RA	98.19		92.04		94.29		93.15											
RI with Default Parameters										**RI with Optimized Parameters**									
IMDb	FA	89.62	0.52	88.88	0.33	76.15	1.70	82.02	1.12	IMDb	FA	98.01	0.08	96.19	0.14	96.36	0.27	96.28	0.20
	RA	90.14		89.21		77.85		83.14			RA	97.93		96.33		96.63		96.48	
IMDb++	FA	89.79	0.80	89.83	0.41	76.71	3.27	82.75	2.05	IMDb++	FA	98.15	0.11	96.78	0.20	96.61	0.32	96.70	0.05
	RA	90.59		90.24		79.98		84.80			RA	98.26		96.58		96.93		96.75	
Film Affinity	FA	90.56	0.36	89.70	0.08	77.69	1.61	83.26	0.89	Film Affinity	FA	98.27	0.48	96.90	0.53	96.20	0.96	96.55	0.74
	RA	90.92		89.62		79.30		84.15			RA	97.79		96.37		95.24		95.81	
Movie Meter	FA	91.05	0.05	88.30	0.04	65.09	0.20	74.94	0.15	Movie Meter	FA	97.91	0.22	95.16	1.22	93.52	1.27	94.33	1.25
	RA	91.00		88.34		65.29		75.09			RA	98.13		96.38		94.79		95.58	
Rotten Tomatoes	FA	91.42	0.15	84.64	1.22	71.99	0.37	77.81	0.29	Rotten Tomatoes	FA	98.02	0.28	94.05	1.23	94.30	1.41	94.18	1.31
	RA	91.57		85.86		71.62		78.10			RA	98.30		95.28		95.71		95.49	
SVM with Default Parameters										**SVM with Optimized Parameters**									
IMDb	FA	84.23	0.57	81.35	0.04	65.41	0.23	72.52	0.90	IMDb	FA	91.27	1.17	88.01	2.90	85.26	2.71	86.61	2.80
	RA	84.80		81.31		65.64		73.42			RA	92.44		90.91		87.97		89.41	
IMDb++	FA	85.26	0.63	81.50	1.44	68.81	1.18	74.62	1.30	IMDb++	FA	93.93	1.04	90.56	2.75	89.22	2.89	89.88	2.83
	RA	85.89		82.94		69.99		75.92			RA	94.97		93.31		92.11		92.71	
Film Affinity	FA	84.17	0.23	82.47	1.14	64.94	1.33	72.66	0.41	Film Affinity	FA	92.53	1.55	89.47	3.14	88.27	3.41	88.87	3.27
	RA	84.40		83.61		63.61		72.25			RA	94.08		92.61		91.68		92.14	
	FA	84.48		65.45		40.96		50.38			FA	92.53		87.93		86.67		87.29	

Continued on next page

Table 6.23 – continued from previous page

Datasets		Accuracy		Precision		Recall		F-Measure		Datasets		Accuracy		Precision		Recall		F-Measure	
Movie Meter	RA	84.86	0.38	65.74	0.29	40.75	0.21	50.32	0.06	Movie Meter	RA	93.50	0.97	90.76	2.83	89.69	3.02	90.22	2.93
Rotten Tomatoes	FA	86.82	0.52	61.52	0.23	50.13	1.39	55.25	0.92	Rotten Tomatoes	FA	94.25	1.20	87.85	5.89	85.08	7.83	86.44	6.88
	RA	87.34		61.75		51.52		56.17			RA	95.45		93.74		92.91		93.32	

Chapter 7

Ensemble Classifiers for Movie Classification Models

Ensemble is a process where multiple different models are built to predict a labelled class, by either using different learning algorithms, the same learning algorithm but with different parameter choices, different learning algorithms on different attribute sets or different learning algorithms with different training data. The Ensemble model then aggregates the prediction of each base model, and select the best one to predict the unseen data. We applied ensemble learners to the following datasets: IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes either using full or reduced attributes. We applied ensemble learners (bagging and boosting) to Artificial Neural Network (ANN), Decision Tree (DT), k-NN, Rule Induction (RI), and SVM Polynomial classifiers.

In order to address the third research question described in Chapter 1 Section 1.4, we details experiments as follows:

- Firstly, we applied bagging algorithm to the ANN, DT, k-NN, RI, and SVM Polynomial classifiers with default parameters to full and reduced attributes IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets.
- Secondly, we applied bagging algorithm to the ANN, DT, RI, and SVM Polynomial classifiers with optimized parameters to full and reduced attributes IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets.
- Thirdly, we applied boosting algorithm to the ANN, DT, k-NN, RI, and SVM Polynomial classifiers with default parameters to full and reduced attributes IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets.
- Lastly, we applied boosting algorithm to the ANN, DT, RI, and SVM Polynomial classifiers with optimized parameters to full and reduced attributes IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets.

The objective of these experiments were to investigate which approach could best find an optimal solution for classifying the movie datasets as iterated in our study case. We aimed to explore as to whether boosting or bagging could achieve a better performance than the base classifiers when applied to all datasets. The experiment diagram is outlined in Figure 7.1.

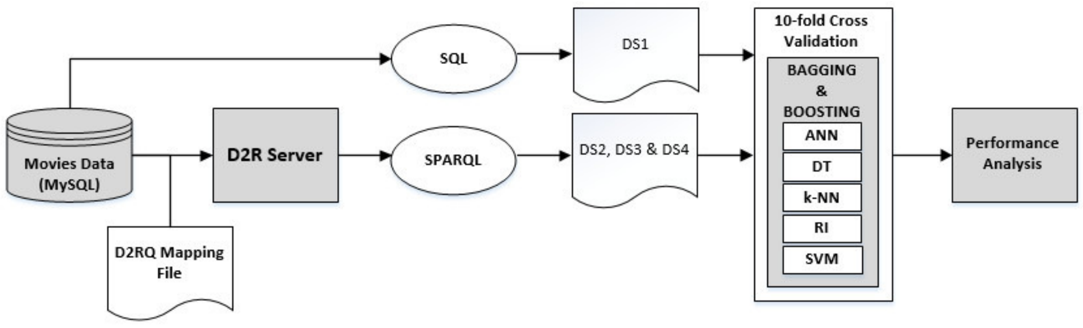


FIGURE 7.1: The Experiment Diagram of Bagging and Boosting

7.1 Bagging Ensemble Classifier

Bagging stands for Bootstrap Aggregating. This is a method to divide the original training set into different training sets using bootstrapping, and for each training set, a model is built using the base learner. A bootstrap training set is constructed by sub-sampling the training set with a replacement, where the size of a training set matches that of the original training set. Each training set contains approximately 63% unique training records compared to the original training sets. Finally, bagging assesses all the learners by majority voting and the most-voted class is predicted. Bagging improves the stability of unstable models, where slight changes in the training data affect their performance, by combining multiple hypotheses of the same data, so that the new aggregate hypothesis reduces these training data variations.

Figure 7.2 depicts the logic of the bagging ensemble method. Generally, this has two levels of process. The root level contains a task to set up the validation process of the learning process. The validation process contains two following processes: training and testing, respectively. The training process determines the setting which builds ensemble models using the bagging method. The bagging process has a sub-process, which specifies a learning algorithm as the base classifier. The testing process is designed to examine the resultant ensemble model as the output of the training process.

As shown in Figure 7.2, the logic of bagging ensemble method is not a ”linear” but a nested logic. To implement the logic of Figure 7.2 in RapidMiner, we design a workflow as shown in Figure 7.3. We use 10-Fold Cross Validation to measure the statistical performance of a bagging ensemble. In RapidMiner, 10-Fold Cross Validation is implemented using the *X-Validation* operator, a nested operator with two inner sub-processes:

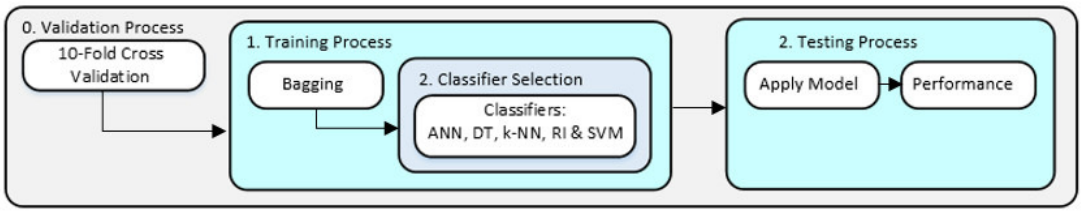


FIGURE 7.2: The Logic of Bagging Ensemble Method

training and testing. In the training process, we tuck the *Bagging* operator, which only has an inner sub-process with one model. We will insert one of the five following learners (ANN, DT, k-NN, RI & SVM) into *Bagging* inner sub-process. The bagging operator has two parameters:

- Sample ratio: sets the number of examples to be used for training, the default value is 0.9
- Iteration: sets the maximum number of iterations, the default value is 10

The iterations parameter of the *Bagging* operator is set to 10, therefore, there will be 10 iterations of its sub-process. In the testing sub-process, we add two operators: *Apply Model* and *Performance*, respectively. The *Apply Model* operator executes the meta models as the output of the *Bagging* operator on the test dataset. The output of the *Apply Model* operator is the labelled test dataset and will be used by the *Performance* operator to evaluate the performance of the bagging ensemble model.

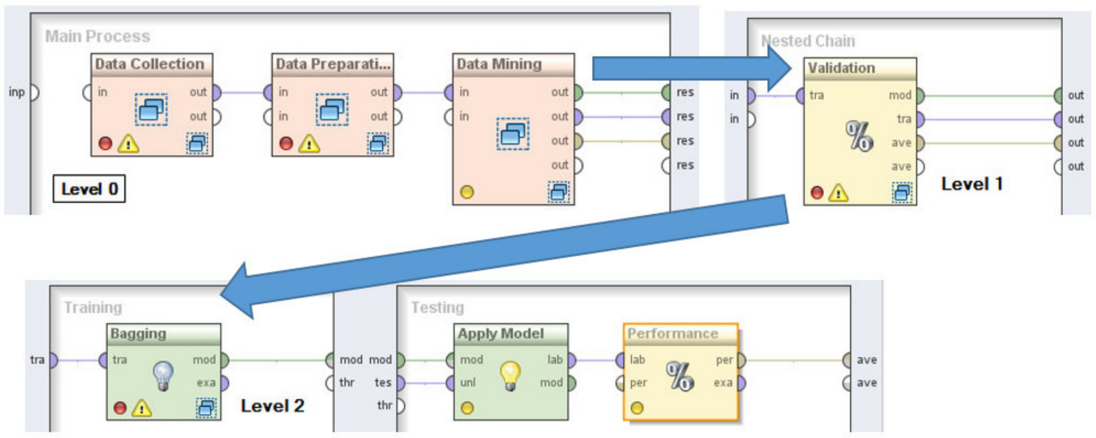


FIGURE 7.3: The Workflow to Setup the Bagging Ensemble Method

7.1.1 Artificial Neural Network (ANN)

To implement ANN as the base learner to build a bagging ensemble method, we design a workflow as shown in Figure 7.4. We tuck the *Neural Net* operator in the inner sub-process of the *Bagging* operator. The Bagging meta model serves as one model with multiple ANN base models inside. The ANN base model results are aggregated using simple voting. Table C.21, C.25, C.29, C.33, and C.37 in Appendix C demonstrates confusion matrixs of Bagging-ANN with default parameters when applied to full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Confusion matrixs of Bagging-ANN with default parameters when applied to reduced-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets are presented in Table C.23, C.27, C.31, C.35, and C.39, respectively in Appendix C.

The use of Bagging-ANN with optimized parameters when applied to the full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets results in the following confusion matrix tables: C.22, C.26, C.30, C.34, and C.38, respectively, as shown in Appendix C. Applied to those with reduced-attributes, Bagging-ANN with optimized parameters yields confusion matrixs C.24, C.28, C.32, C.36, and C.40, respectively, as shown in Appendix C. Those confusion matrixs were able to visualize missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of bagging to ANN classifier either default or optimized parameters when applied to those datasets either using full or reduced attributes as shown in Table 7.1. The table present accuracy, precision, recall, and f-measure value for each dataset.

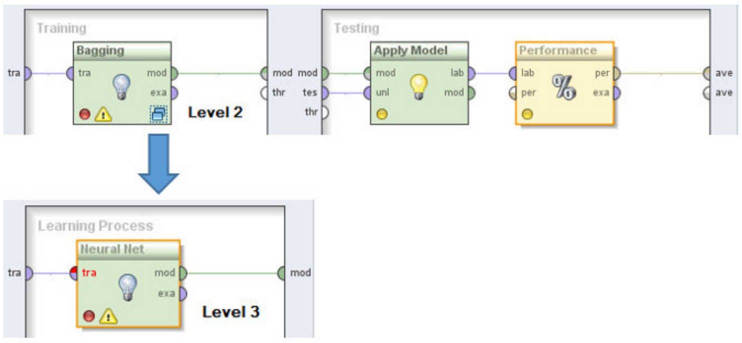


FIGURE 7.4: The Workflow for Bagging Ensemble Method with ANN

The result presented in Table 7.1 indicates that the best result of bagging-ANN is achieved by applying bagging-ANN with optimized parameters to the full-attributes IMDb++ dataset. This attained accuracy, precision, recall, and F-Measure values by 90.02%, 88.09%, 80.08%, and 83.90%, respectively. This model improved the classification performance of the full-attributes IMDb++ model generated by ANN with optimized parameters as a single classifier. The improvements were accuracy of 1.45% (88.57% to 90.02%), precision of 3.23% (84.86% to 88.09%), recall of 1.77% (78.31% to 80.08%), and F-Measure of 2.45% (81.45% to 83.90%).

Table result presented in Table 7.1 also suggest that the use of bagging-ANN either using default or optimized parameters results in better classification models when applied to datasets with full-attributes than reduced-attributes. Out of 10 predictive models generated by bagging-ANN, it was only 1 model that bagging-ANN with default parameters performed better when applied to the reduced-attributes than full-attributes of IMDb dataset. The reduced-attributes IMDb model improved recall and F-Measure values of the full-attributes IMDb model by 0.82% (75.06% to 75.88%) and 0.25% (80.11% to 80.36%), respectively.

7.1.2 Decision Tree (DT)

To implement DT as the base learner to build a bagging ensemble method, we designed a workflow, shown in Figure 7.5. We applied the *Decision Tree* operator in the sub-process of the *Bagging* operator. The Bagging meta model serves as one model with multiple DT base models inside, and the results from the DT base models are aggregated using simple voting. Table E.21, E.25, E.29, E.33, and E.37 in Appendix E demonstrates confusion matrixs of Bagging-DT with default parameters when applied to full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Confusion matrixs of Bagging-DT with default parameters when applied to reduced-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets are presented in Table E.23, E.27, E.31, E.35, and E.39, respectively in Appendix E.

The use of Bagging-DT with optimized parameters when applied to the full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets results in the following confusion matrix tables: E.22, E.26, E.30, E.34, and E.38, respectively, as shown in Appendix E. Applied to those with reduced-attributes, Bagging-DT with optimized parameters yields confusion matrixs E.24, E.28, E.32, E.36, and E.40, respectively, as shown in Appendix E. Those confusion matrixs were able to visualize misclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of bagging-DT classifier either using default or optimized parameters when applied to those datasets with full and reduced datasets as shown in Table 7.2. The table present accuracy, precision, recall, and f-measure value for each dataset.

The result presented in Table 7.2 demonstrates that the best result of bagging-DT is achieved by applying bagging-DT with optimized parameters to the reduced-attributes IMDb dataset. This yielded accuracy, precision, recall, and F-Measure values by 98.49%, 97.75%, 96.82%, and 97.28%, respectively. This model failed to improve the classification performance of the reduced-attributes IMDb model generated by DT with optimized parameters as a single classifier. Recall and F-Measure values declined by 0.58% (97.40% to 96.82%) and 0.28% (97.56% to 97.28%), respectively, while accuracy and precision slightly increased by 0.02% (98.47% to 98.49%) and 0.02% (97.73% to 97.75%), respectively.

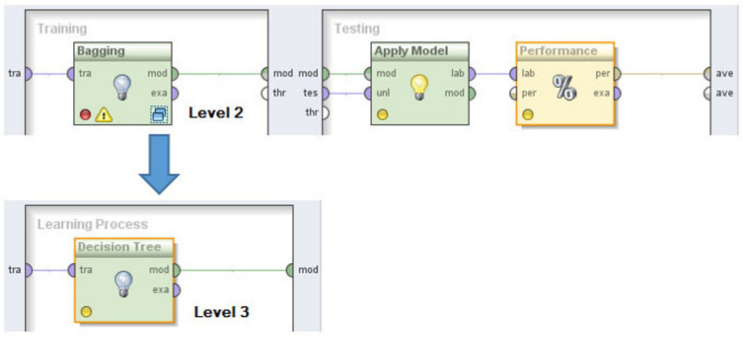


FIGURE 7.5: The Workflow for Bagging Ensemble Method with Decision Tree

Similar to the classification resutls of bagging-ANN, the result presented in Table 7.2 also suggests that the use of bagging-DT either using default or optimized parameters results in better classification models when applied to datasets with full-attributes than reduced-attributes. Out of 10 predictive models generated by bagging-DT, there were three cases that bagging-DT performed better when applied to the datasets with reduced-attributes than full-attributes. Firstly, the reduced-attributes IMDb model generated by bagging-DT with default parameters improved precision and F-Measure values of the full-attributes IMDb model by 0.04% (81.89% to 81.93%) and 0.01% (73.41% to 73.42%), respectively. Secondly, the reduced-attributes IMDb model generated by bagging-DT with optimized parameters improved accuracy, precision, recall and F-Measure values of the full-attributes IMDb model by 0.09% (98.40% to 98.49%), 0.39% (97.36% to 97.75%), 0.14% (96.68% to 96.82%) and 0.26% (97.02% to 97.28%), respectively. Lastly, the reduced-attributes IMDb++ model generated by bagging-DT with optimized parameters improved accuracy, precision, and F-Measure values of the full-attributes IMDb++ model by 0.05% (98.44% to 98.49%), 0.50% (97.31% to 97.81%), and 0.20% (97.07% to 97.27%), respectively.

7.1.3 k-Nearest Neighbour (kNN)

To implement k-NN as the base learner to build a bagging ensemble method, we designed a workflow as shown in Figure 7.6, and set the *k-NN* operator in the inner sub-process of the *Bagging* operator. The Bagging meta model serves as one model with multiple k-NN base models inside, the results from which are aggregated using simple voting. Table G.11, G.13, G.15, G.17, and G.19 in Appendix G demonstrates confusion matrixs of Bagging-kNN with default parameters when applied to full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Confusion matrixs of Bagging-kNN with default parameters when applied to reduced-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets are presented in Table G.12, G.14, G.16, G.18, and G.20, respectively in Appendix G. Table 7.3 shows the results of the bagging ensemble method with k-NN as the base learner.

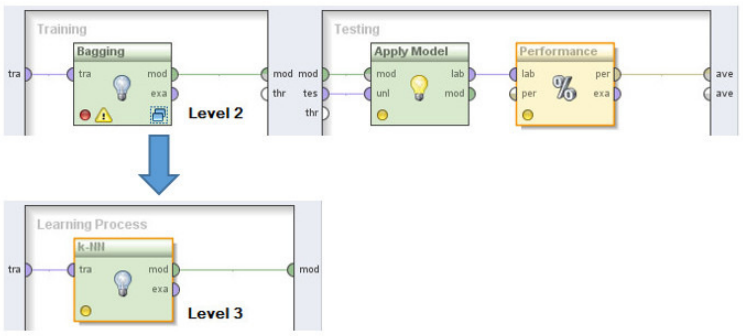


FIGURE 7.6: The Workflow for Bagging Ensemble Method with k-NN

The result presented in Table 7.3 exhibits that the best result of bagging-kNN is achieved by applying bagging-kNN with default parameters to the reduced-attributes IMDb++ dataset. This produced accuracy, precision, recall, and F-Measure values by 98.15%, 96.21%, 97.14%, and 96.67%, respectively. This model failed to improve the classification performance of the reduced-attributes IMDb++ model generated by kNN with default parameters as a single classifier. Accuracy, precision and F-Measure values declined by 0.03% (98.18% to 98.15%), 1.03% (97.24% to 96.21%) and 0.04% (96.71% to 96.67%), respectively, while recall slightly increased by 0.95% (96.19% to 97.14%).

In contrast to the classification resutls of bagging-ANN and bagging-DT, the result presented in Table 7.3 suggests that the use of bagging-kNN using default parameters results in better classification models when applied to datasets with reduced-attributes than full-attributes. Bagging-kNN to reduced-attributes datasets improved significantly F-Measure values of bagging-kNN to full-attributes datasets, from 9.33% to 12.79%.

7.1.4 Rule Induction (RI)

To implement RI as the base learner to build a bagging ensemble method, we designed a workflow as shown in Figure 7.7. We applied the *Rule Induction* operator to the inner sub-process of the *Bagging* operator. The Bagging meta model serves as one model with multiple RI base models inside. The RI base model results are aggregated using simple voting. Table H.21, H.25, H.29, H.33, and H.37 in Appendix H demonstrates confusion matrixs of Bagging-RI with default parameters when applied to full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Confusion matrixs of Bagging-RI with default parameters when applied to reduced-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets are presented in Table H.23, H.27, H.31, H.35, and H.39, respectively in Appendix H.

The use of Bagging-RI with optimized parameters when applied to the full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets results in the following confusion matrix tables: H.22, H.26, H.30, H.34, and H.38, respectively, as shown in Appendix H. Applied to those with reduced-attributes, Bagging-RI with optimized parameters yields confusion matrixs H.24, H.28, H.32, H.36, and H.40, respectively, as shown in Appendix H. Those confusion matrixs were able to visualize missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of bagging-RI classifier with default and optimized parameters when applied to those five reduced datasets either using full or reduced attributes as shown in Table 7.4. The table present accuracy, precision, recall, and f-measure value for each dataset.

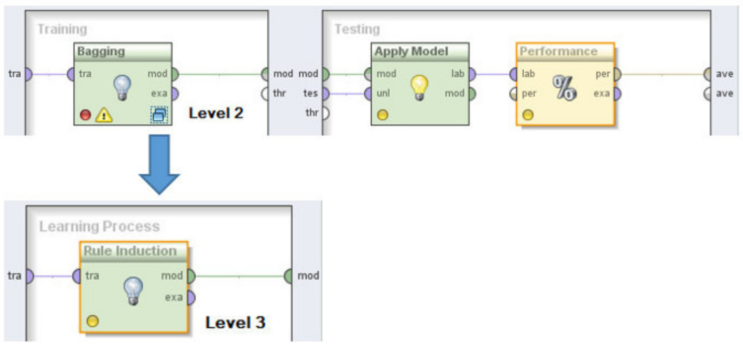


FIGURE 7.7: The Workflow for Bagging Ensemble Method with Rule Induction

The result presented in Table 7.4 demonstrates that the best result of bagging-RI is obtained by applying bagging-RI with optimized parameters to the full-attributes IMDb++ dataset. This attained accuracy, precision, recall, and F-Measure values by 98.71%, 98.24%, 96.86%, and 97.54%, respectively. This model successfully improved the classification performance of the full-attributes IMDb++ model generated by RI with optimized parameters as a single classifier. Accuracy, Recall and F-Measure values increased by 0.56% (98.15% to 98.71%), 1.46% (96.78% to 98.24%) and 0.84% (96.70% to 97.54%), respectively, while recall value declined by 0.25% (96.86% to 96.61%).

Similar to the classification resutls of bagging-ANN and bagging-DT, the result presented in Table 7.4 suggests that the use of bagging-RI either using default or optimized parameters results in better classification models when applied to datasets with full-attributes than reduced-attributes. Out of 10 predictive models generated by bagging-RI, there were four cases that bagging-RI performed better when applied to the datasets with reduced-attributes than full-attributes. Firstly, the reduced-attributes IMDb model generated by bagging-RI with default parameters improved accuracy, recall, and F-Measure values of the full-attributes IMDb model by 0.33% (91.17% to 91.50%), 0.32% (78.95% to 79.27%) and 0.12% (85.37% to 85.49%), respectively. Secondly, the reduced-attributes IMDb++ model generated by bagging-RI with default parameters increased accuracy, recall and F-Measure values of the full-attributes IMDb++ model by 0.73% (91.17% to 91.90%), 2.83% (78.83% to 81.66%), and 1.46% (85.33% to 86.79%), respectively. Thirdly, the reduced-attributes FilmAffinity model generated by bagging-RI with optimized parameters improved precision, recall, and F-Measure values of the full-attributes FilmAffinity model by 0.01% (98.24% to 98.25%), 0.18% (96.01% to 96.19%), and 0.10%

(97.11% to 97.21%), respectively. Lastly, the reduced-attributes RottenTomatoes model generated by bagging-RI with optimized parameters improved accuracy, precision, recall, and F-Measure values of the full-attributes RottenTomatoes model by 0.01% (98.71% to 98.72%), 1.04% (96.20% to 97.24%), 0.73% (95.22% to 95.95%), and 0.88% (95.71% to 96.59%), respectively.

7.1.5 Support Vector Machine (SVM)

To implement SVM as the base learner to build a bagging ensemble method, we designed a workflow as shown in Figure 7.8. We inserted the *SVM* operator in the inner sub-process of the *Bagging* operator. The Bagging meta model serves as one model with multiple SVM base models inside. The SVM base model results are aggregated using simple voting. Table J.21, J.25, J.29, J.33, and J.37 in Appendix J demonstrates confusion matrixs of Bagging-SVM Polynomial with default parameters when applied to full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Confusion matrixs of Bagging-SVM Polynomial with default parameters when applied to reduced-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets are presented in Table J.23, J.27, J.31, J.35, and J.39, respectively in Appendix J.

The use of Bagging-DT with optimized parameters when applied to the full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets results in the following confusion matrix tables: J.22, J.26, J.30, J.34, and J.38, respectively, as shown in Appendix J. Applied to those with reduced-attributes, Bagging-SVM Polynomial with optimized parameters yields confusion matrixs J.24, J.28, J.32, J.36, and J.40, respectively, as shown in Appendix J. Those confusion matrixs were able to visualize missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of bagging-SVM Polynomial either using default or optimized parameters when applied to those five full and reduced datasets as shown in Table 7.5. The table present accuracy, precision, recall, and f-measure value for each dataset.

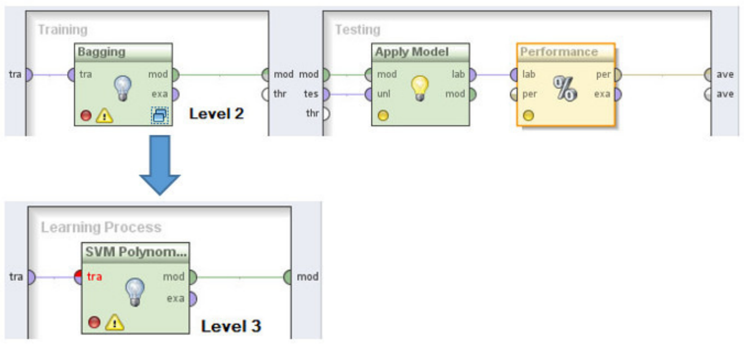


FIGURE 7.8: The Workflow for Bagging Ensemble Method with SVM

The result presented in Table 7.5 indicates that the best result of bagging-SVM Polynomial is achieved by applying bagging-SVM Polynomial with optimized parameters to the reduced-attributes IMDb++ dataset. This gained accuracy, precision, recall, and F-Measure values by 94.60%, 92.47%, 91.77%, and 92.12%, respectively. This model failed to improve the classification performance of the reduced-attributes IMDb++ model generated by SVM Polynomial with optimized parameters as a single classifier. Accuracy, precision, recall and F-Measure values declined by 0.37% (94.97% to 94.60%), 0.84% (93.31% to 92.47%), 0.34% (92.11% to 91.77%) and 0.59% (92.71% to 92.12%).

Similar to the classification results of bagging-kNN, table result presented in Table 7.5 suggests that the use of bagging-SVM Polynomial either using default or optimized parameters results in better classification models when applied to datasets with reduced-attributes than full-attributes. Out of 10 predictive models generated by bagging-SVM Polynomial, there was only 1 model that bagging-SVM Polynomial with default parameters performed better when applied to the datasets with full-attributes than reduced-attributes.The reduced-attributes FilmAffinity model failed to improve recall and F-Measure values of the full-attributes FilmAffinity model by 1.28% (64.70% to 63.42%) and 0.53% (72.66% to 72.13%), respectively.

Table 7.1: Results of Bagging-ANN Applied to Full-Attributes (FA) and Reduced-Attributes (RA) Datasets

Datasets		Accuracy		Precision		Recall		F-Measure		Datasets		Accuracy		Precision		Recall		F-Measure	
Bagging-ANN with Default Parameters										**Bagging-ANN with Optimized Parameters**									
IMDb	FA	87.92	0.08	85.90	0.49	75.06	0.82	80.11	0.25	IMDb	FA	87.89	0.53	85.73	0.50	75.42	0.85	80.25	0.71
	RA	87.84		85.41		75.88		80.36			RA	87.36		85.23		74.57		79.54	
IMDb++	FA	90.03	0.31	88.58	0.47	79.59	0.69	83.85	0.60	IMDb++	FA	90.02	1.13	88.09	1.50	80.08	2.67	83.90	2.16
	RA	89.72		88.11		78.90		83.25			RA	88.89		86.59		77.41		81.74	
Film Affinity	FA	88.37	0.24	86.11	0.50	76.48	0.58	81.01	0.55	Film Affinity	FA	88.46	0.78	85.84	1.79	76.50	1.42	80.90	1.59
	RA	88.13		85.61		75.90		80.46			RA	87.68		84.05		75.08		79.31	
Movie Meter	FA	89.44	0.31	84.00	0.59	72.20	0.00	77.65	0.25	Movie Meter	FA	89.41	0.33	84.35	1.10	71.47	3.02	77.37	2.24
	RA	89.13		83.41		72.20		77.40			RA	89.08		83.25		68.45		75.13	
Rotten Tomatoes	FA	91.43	0.34	80.71	2.80	70.19	1.34	75.09	2.33	Rotten Tomatoes	FA	91.56	1.32	80.40	2.81	68.77	5.04	74.13	4.15
	RA	91.09		77.91		68.26		72.76			RA	90.24		77.59		63.73		69.98	

Table 7.2: Results of Bagging-DT Full-Attributes (FA) and Reduced-Attributes (RA) Datasets

Datasets		Accuracy		Precision		Recall		F-Measure		Datasets		Accuracy		Precision		Recall		F-Measure	
Bagging-DT with Default Parameters										**Bagging-DT with Optimized Parameters**									
IMDb	FA	83.80	0.01	81.89	0.04	66.52	0.01	73.41	0.01	IMDb	FA	98.40	0.09	97.36	0.39	96.68	0.14	97.02	0.26
	RA	83.79		81.93		66.51		73.42			RA	98.49		97.75		96.82		97.28	
IMDb++	FA	84.86	0.23	86.37	1.33	66.88	0.02	75.39	0.50	IMDb++	FA	98.44	0.05	97.31	0.50	96.84	0.10	97.07	0.20
	RA	84.63		85.04		66.90		74.89			RA	98.49		97.81		96.74		97.27	
Film Affinity	FA	82.16	0.06	88.61	0.23	55.06	0.31	67.92	0.17	Film Affinity	FA	98.41	0.14	97.75	0.40	95.80	0.10	96.76	0.14
	RA	82.22		88.84		54.75		67.75			RA	98.27		97.35		95.90		96.62	
Movie Meter	FA	73.10	0.00	18.28	0.00	25.00	0.00	21.12	0.00	Movie Meter	FA	98.31	0.13	96.93	0.27	93.85	0.18	95.37	0.22
	RA	73.10		18.28		25.00		21.12			RA	98.18		96.66		93.67		95.15	

Continued on next page

Table 7.2 – continued from previous page

Datasets		Accuracy		Precision		Recall		F-Measure		Datasets		Accuracy		Precision		Recall		F-Measure	
Rotten Tomatoes	FA	87.67	0.00	78.53	0.00	62.86	0.01	69.83	0.00	Rotten Tomatoes	FA	98.55	0.03	96.69	0.70	95.68	0.43	96.18	0.56
	RA	87.67		78.53		62.87		69.83			RA	98.58		95.99		95.25		95.62	

Table 7.3: Results of Bagging-kNN Full-Attributes (FA) and Reduced-Attributes (RA) Datasets

Datasets		Accuracy		Precision		Recall		F-Measure		Datasets		Accuracy		Precision		Recall		F-Measure	
k-NN with a Default Parameter																			
IMDb	FA	89.13	8.70	81.01	14.89	85.53	10.57	83.21	12.79										
	RA	97.83		95.90		96.10		96.00											
IMDb++	FA	91.19	6.96	84.32	11.89	88.44	8.70	86.33	10.34										
	RA	98.15		96.21		97.14		96.67											
Film Affinity	FA	89.67	8.36	82.47	13.29	86.49	10.59	84.44	11.98										
	RA	98.03		95.76		97.08		96.42											
Movie Meter	FA	90.64	7.20	80.13	12.84	86.42	9.69	83.16	11.35										
	RA	97.84		92.97		96.11		94.51											
Rotten Tomatoes	FA	91.73	6.46	82.79	11.50	84.88	7.16	83.82	9.33										
	RA	98.19		94.29		92.04		93.15											

Table 7.4: Results of Bagging-RI Full-Attributes (FA) and Reduced-Attributes (RA) Datasets

Datasets		Accuracy		Precision		Recall		F-Measure		Datasets		Accuracy		Precision		Recall		F-Measure	
RI with Default Parameters										**RI with Optimized Parameters**									
IMDb	FA	91.17	0.33	92.93	0.16	78.95	0.32	85.37	0.12	IMDb	FA	98.71	0.12	98.02	0.17	97.01	0.03	97.51	0.06
	RA	91.50		92.77		79.27		85.49			RA	98.59		97.85		97.04		97.45	

Continued on next page

Table 7.4 – continued from previous page

Datasets		Accuracy		Precision		Recall		F-Measure		Datasets		Accuracy		Precision		Recall		F-Measure	
IMDb++	FA	91.17	0.73	93.01	0.40	78.83	2.83	85.33	1.46	IMDb++	FA	98.71	0.15	98.24	0.08	96.86	0.02	97.54	0.04
	RA	91.90		92.61		81.66		86.79			RA	98.56		98.16		96.84		97.50	
Film	FA	92.23	1.31	93.15	3.53	79.90	0.60	86.02	1.87	Film	FA	98.69	0.03	98.24	0.01	96.01	0.18	97.11	0.10
Affinity	RA	90.92		89.62		79.30		84.15		Affinity	RA	98.66		98.25		96.19		97.21	
Movie	FA	92.40	0.05	92.81	0.58	67.60	0.15	78.22	0.10	Movie	FA	98.50	0.03	97.88	0.15	94.51	0.10	96.17	0.13
Meter	RA	92.45		92.23		67.75		78.12		Meter	RA	98.53		97.73		94.41		96.04	
Rotten	FA	92.42	0.00	90.94	0.86	74.47	2.45	81.89	1.84	Rotten	FA	98.71	0.01	96.20	1.04	95.22	0.73	95.71	0.88
Tomatoes	RA	92.42		90.08		72.02		80.05		Tomatoes	RA	98.72		97.24		95.95		96.59	

Table 7.5: Results of Bagging-SVM Polynomial Full-Attributes (FA) and Reduced-Attributes (RA) Datasets

Datasets		Accuracy		Precision		Recall		F-Measure		Datasets		Accuracy		Precision		Recall		F-Measure	
SVM with Default Parameters										**SVM with Optimized Parameters**									
IMDb	FA	84.10	0.78	80.84	2.22	65.10	1.01	72.12	1.51	IMDb	FA	90.89	1.37	87.17	3.19	85.35	2.56	86.25	2.87
	RA	84.88		83.06		66.11		73.63			RA	92.26		90.36		87.91		89.12	
IMDb++	FA	85.07	0.70	81.14	1.65	68.59	1.21	74.34	1.40	IMDb++	FA	93.28	1.32	89.26	3.21	89.23	2.54	89.25	2.87
	RA	85.77		82.79		69.80		75.74			RA	94.60		92.47		91.77		92.12	
Film	FA	84.16	0.27	82.84	0.77	64.70	1.28	72.66	0.53	Film	FA	92.01	1.89	88.73	3.37	88.18	3.51	88.45	3.44
Affinity	RA	84.43		83.61		63.42		72.13		Affinity	RA	93.90		92.10		91.69		91.89	
Movie	FA	84.41	0.39	65.37	0.20	40.82	0.02	50.26	0.07	Movie	FA	91.80	1.52	86.28	3.69	85.75	3.81	86.02	3.74
Meter	RA	84.80		65.57		40.80		50.33		Meter	RA	93.32		89.97		89.56		89.76	
Rotten	FA	86.64	0.53	61.51	0.06	49.60	1.58	54.91	0.99	Rotten	FA	93.99	1.66	85.39	5.13	83.62	5.42	84.50	5.27
Tomatoes	RA	87.17		61.57		51.18		55.90		Tomatoes	RA	95.65		90.52		89.04		89.77	

7.2 Boosting Ensemble Classifier

As with bagging, boosting is another form of ensemble learning which combines many weak learners into a single, strong learner. It consists of sequential learning of the classifiers where the first one is built and tested with the whole dataset, and based on the output, the next model is built. Boosting focuses on the points that are misclassified, or difficult to learn. Incorrectly classified instances are given a larger weight, so that they will have a higher propensity of selection for the next iteration. In contrast, correctly classified instances will have lower weights.

Introduced by Freund and Schapire (1995), AdaBoost is one of the most popular boosting ensemble approaches. It stands for "Adaptive Boosting", and its ensemble model has m base classifiers and n training records. AdaBoost runs for a number of iterations, following these steps:

1. The set of training records is treated as a weighted distribution D_i on the i^{th} iteration.
2. Each training record is given an uniform weight $w_i = 1/\text{n}$.
3. The base classifier is trained using the distribution D_i.
4. The weight of the classifier (α_i) can be calculated as $\alpha_i = \frac{1}{2}\ln\left(\frac{1-\epsilon_i}{\epsilon_i}\right)$, where ϵ_i is the error rate for the base classifier.
5. Next, the weights of all training records are updated by:
   - If the class of the i^{th} record is predicted correctly then $D_{i+1}(j) = \frac{D_i(j)}{Z_i}e^{-\alpha_i}$
   - Otherwise $D_{i+1}(j) = \frac{D_i(j)}{Z_i}e^{\alpha_i}$.

   To make D_{i+1} is a distribution, then, it is required to choose Z_i as a normalization factor.
6. The final classifier is the weighted sum of the base classifiers at each iteration i with the weight given to each of the classifiers being α_i.

Figure 7.9 depicts the logic of boosting ensemble method. Generally, the logic has two levels of process. The root level contains a task to setup the validation process of the learning process. The validation process contains two following processes: training and testing, respectively. The training process determines the setting to build ensemble models using boosting method. The boosting process has a sub-process which specifies a learning algorithm as the base classifier. The testing process is designed to examine the resultant ensemble model as the output of the training process.

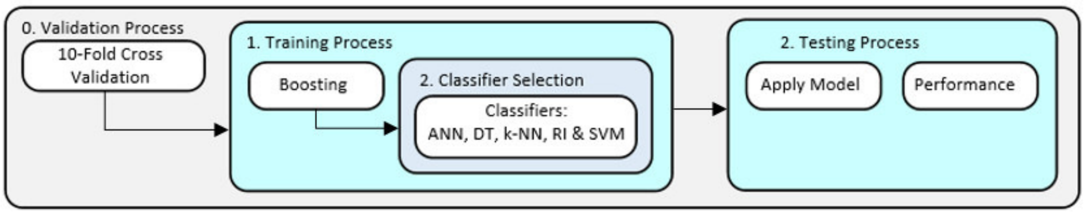


FIGURE 7.9: The Logic of Boosting Ensemble Method

As shown in Figure 7.9, the logic of boosting ensemble method is not ”linear”, but a nested logic. To implement the logic of Figure 7.9 in RapidMiner, we designed a workflow as shown in Figure 7.10. We also used a 10-Fold Cross Validation to measure the statistical performance of a boosting ensemble, implemented using the *X-Validation* operator, a nested operator with two inner sub-processes: training and testing. In the training process, we inserted the *AdaBoost* operator which has only an inner sub-process. *Boosting* has only one model in the inner sub-process, and we inserted one of the five following learners (ANN, DT, k-NN, RI & SVM) into this. The AdaBoost operator has only one parameter: ”iteration”, used to set the maximum number of iterations, with the default value at 10.

The iterations parameter of the *AdaBoost* operator is set to 10, therefore, there will be 10 iterations of its sub-process. Meanwhile, in the testing sub-process, we add two operators: *Apply Model* and *Performance*, respectively. The *Apply Model* operator executes the meta models as the output of the *AdaBoost* operator on the test dataset. The output of the *Apply Model* operator is the labelled test dataset, used by the *Performance* operator to evaluate the performance of the boosting ensemble model.

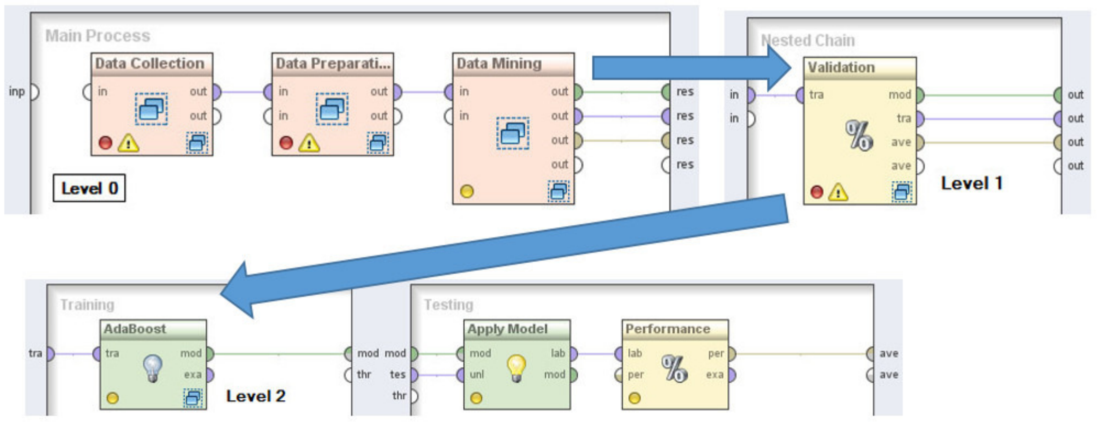


FIGURE 7.10: The Workflow to Setup the Boosting Ensemble Method

7.2.1 Artificial Neural Network (ANN)

To implement ANN as the base learner to build a boosting ensemble method, we use *AdaBoost* operator in conjunction with the *Neural Net* operator as the learning algorithm as shown in the workflow in Figure 7.11. The *AdaBoost* operator is applied in the training sub-process of the *X-Validation* control. The *Neural Net* operator is applied in the sub-process of the *AdaBoost* operator. The *AdaBoost* operator produced an ANN new model in each iteration and there are different weights for each ANN model. The AdaBoost generated a combination of ANN base models that outperformed the original model. Table C.41, C.45, D.1, D.5, and D.9 in Appendix C and D demonstrates confusion matrixs of Boosting-ANN with default parameters when applied to full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Confusion matrixs of Boosting-ANN with default parameters when applied to reduced-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets are presented in Table C.43, C.47, D.3, D.7, and D.11, respectively in Appendix C and D.

The use of Bagging-ANN with optimized parameters when applied to the full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets results in the following confusion matrix tables: C.42, C.46, D.2, D.6, and D.10, respectively, as shown in Appendix C and D. Applied to those with reduced-attributes, Bagging-ANN with optimized parameters yields confusion matrixs C.44, C.48, D.4, D.8, and D.12, respectively, as shown in Appendix C and D. Those confusion matrixs were able to visualize missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of boosting-ANN either using default or optimized parameters when applied to those five reduced datasets with full and reduced datasets as shown in Table 7.6. The table present accuracy, precision, recall, and f-measure value for each dataset.

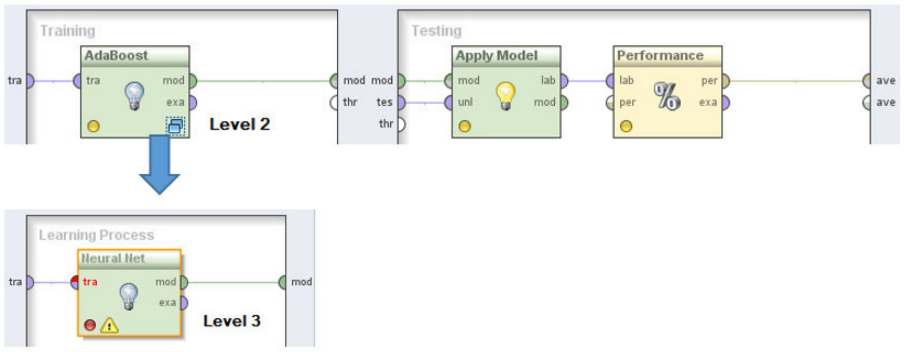


FIGURE 7.11: The Workflow for Boosting Ensemble Method with ANN

The result presented in Table 7.6 indicates that the best result of boosting-ANN is achieved by applying boosting-ANN with optimized parameters to the full-attributes IMDb++ dataset. This attained accuracy, precision, recall, and F-Measure values by 90.95%, 88.72%, 84.27%, and 86.44%, respectively. This model improved the classification performance of the full-attributes IMDb++ model generated by ANN with optimized parameters as a single classifier. The improvements were accuracy of 2.38% (88.57% to 90.95%), precision of 3.86% (84.86% to 88.72%), recall of 5.96% (78.31%

to 84.27%), and F-Measure of 4.99% (81.45% to 86.44%). Moreover, the full-attributes IMDb++ model improved accuracy, precision, recall, and F-Measure values of full-attributes IMDb++ model generated by bagging-ANN with optimized parameters by 0.93% (90.02% to 90.95%), 0.63% (88.09% to 88.72%), 4.19%% (80.08% to 84.27%), and 2.54% (83.90% to 86.44%), respectively.

Table result presented in Table 7.6 suggests that the use of boosting-ANN either using default or optimized parameters results in better classification models when applied to datasets with full-attributes than reduced-attributes. Out of 10 predictive models generated by boosting-ANN, it was only 1 model that boosting-ANN with default parameters performed better when applied to the reduced-attributes than full-attributes of MovieMeter dataset. The reduced-attributes MovieMeter model improved accuracy, precision and F-Measure values of the full-attributes MovieMeter model by 0.01% (89.48% to 89.49%), 0.48% (85.21% to 86.69%) and 0.60% (80.31% to 80.91%), respectively.

7.2.2 Decision Tree (DT)

To implement DT as the base learner to build a boosting ensemble method, we designed a workflow as shown in Figure 7.12. We used the *AdaBoost* operator in conjunction with the *Decision Tree* operator as the learning algorithm to achieve the goal. The *AdaBoost* operator was applied to the training sub-process of the *X-Validation* control, whilst the *Decision Tree* operator was applied in the sub-process of the *AdaBoost* operator. The *AdaBoost* operator produced an DT new model in each iteration, with different weights for each DT model. AdaBoost generated a combination of the DT base models that outperformed the original DT model. Table E.41, E.45, F.1, F.5, and F.9 in Appendix E and F demonstrates confusion matrixs of Boosting-ANN with default parameters when applied to full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Confusion matrixs of Boosting-ANN with default parameters when applied to reduced-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets are presented in Table E.43, E.47, F.3, F.7, and F.11, respectively in Appendix E and F.

The use of Bagging-DT with optimized parameters when applied to the full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets results in the following confusion matrix tables: E.42, E.46, F.2, F.6, and F.10, respectively, as shown in Appendix E and F. Applied to those with reduced-attributes, Boosting-DT with optimized parameters yields confusion matrixs E.44, E.48, F.4, F.8, and F.12, respectively, as shown in Appendix E and F. Those confusion matrixs were able to visualize missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of boosting-DT either using default or optimized parameters when applied to those five full and reduced datasets as shown in

Table 7.7. The table present accuracy, precision, recall, and f-measure value for each dataset.

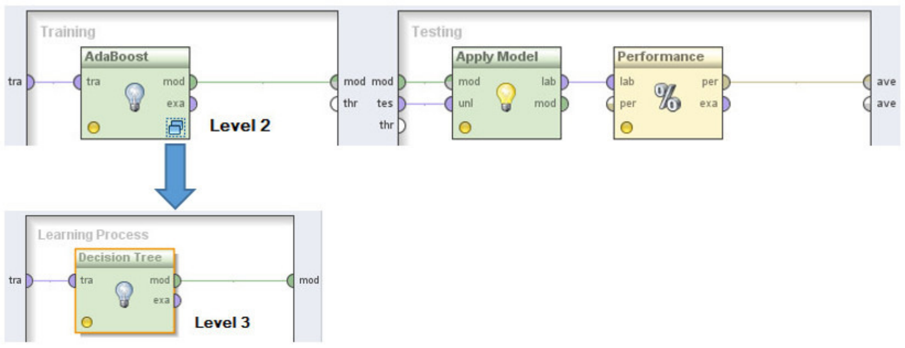


FIGURE 7.12: The Workflow for Boosting Ensemble Method with Decision Tree

The result presented in Table 7.7 demonstrates that the best result of boosting-DT is obtained by applying boosting-DT with optimized parameters to the reduced-attributes IMDb dataset. This attained accuracy, precision, recall, and F-Measure values by 98.31%, 97.12%, 96.81%, and 96.96%, respectively. This model improved the classification performance of the full-attributes IMDb model generated by DT with optimized parameters as a single classifier. The improvements were accuracy of 0.15% (98.16% to 98.31%), precision of 0.42% (96.70% to 97.12%), recall of 0.23% (96.58% to 96.81%), and F-Measure of 0.32% (96.64% to 96.96%). However, the reduced-attributes IMDb model declined accuracy, precision, recall, and F-Measure values of reduced-attributes IMDb model generated by bagging-DT with optimized parameters by 0.18% (98.49% to 98.31%), 0.63% (97.75% to 97.12%), 0.01%% (96.82% to 96.81%), and 0.32% (97.28% to 96.96%), respectively.

Table result presented in Table 7.7 suggests that the use of boosting-DT either using default or optimized parameters results in better classification models when applied to datasets with reduced-attributes than full-attributes. Out of 10 predictive models generated by boosting-DT, there were three models that boosting-DT did not perform better or just maintained when applied to the datasets with full-attributes than reduced-attributes. Firstly, the full-attributes FilmAffinity model generated by boosting-DT with optimized parameters decreased precision, recall and F-Measure values of the reduced-attributes FilmAffinity model by 0.20% (96.78% to 96.58%), 0.24% (96.15% to 95.91%) and 0.23% (96.47% to 96.24%), respectively. Secondly, the full-attributes MovieMeter model generated by boosting-DT with default parameters maintained accuracy, precision, recall and F-Measure values of the reduced-attributes MovieMeter model by 0.00% (73.10% to 73.10%), 0.00% (18.28% to 18.28%), 0.00% (25% to 25%) and 0.00% (21.12% to 21.12%), respectively. Lastly, the full-attributes RottenTomatoes model generated by boosting-DT with default parameters maintained accuracy, precision, recall, and F-Measure values of the reduced-attributes RottenTomatoes model by 0.00% (87.53% to

87.53%), 0.00% (72.92% to 72.92%), 0.00% (58.05% to 58.05%) and 0.00% (64.64% to 64.64%), respectively.

7.2.3 k-Nearest Neighbour (kNN)

To implement k-NN as the base learner to build a boosting ensemble method, we used *AdaBoost* operator in conjunction with the *k-NN* operator as the learning algorithm as shown in the workflow in Figure 7.13. The *AdaBoost* operator was applied to the training sub-process of the *X-Validation* control. The *k-NN* operator was applied to the sub-process of the *AdaBoost* operator, which produced a k-NN new model in each iteration, with different weights for each k-NN model. AdaBoost generated a combination of the k-NN base models that performed better than the original k-NN model. Table G.21, G.23, G.25, G.27, and G.29 in Appendix G demonstrates confusion matrixs of Boosting-kNN with default parameters when applied to full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Confusion matrixs of Boosting-kNN with default parameters when applied to reduced-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets are presented in Table G.22, G.24, G.26, G.28, and G.30, respectively in Appendix G. Table 7.8 shows the results of boosting ensemble method with k-NN as the base learner.

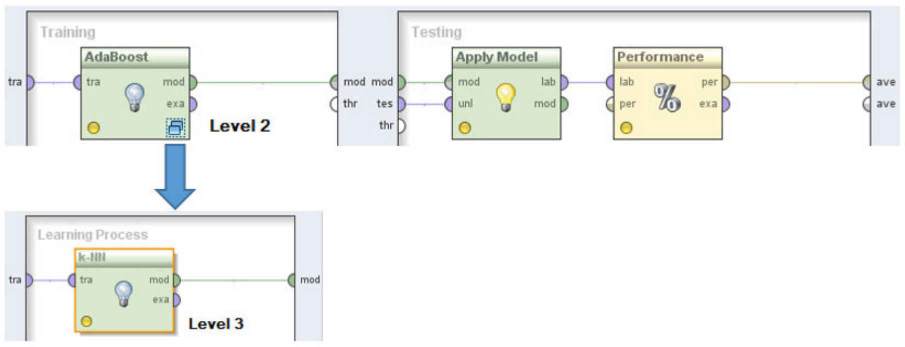


FIGURE 7.13: The Workflow for Boosting Ensemble Method with k-NN

The result presented in Table 7.8 indicates that the best result of boosting-kNN is gained by applying boosting-kNN with default parameters to the reduced-attributes IMDb++ dataset. This yielded accuracy, precision, recall, and F-Measure values by 98.16%, 96.21%, 97.15%, and 96.68%, respectively. This model failed to improve the classification performance of the reduced-attributes IMDb++ model generated by kNN with default parameters as a single classifier. The accuracy, precision and F-Measure values dropped to 0.02% (98.18% to 98.16%), 1.03% (97.24% to 96.21%), and 0.03% (96.71% to 96.68%), respectively. Moreover, this model slightly improved the accuracy, recall, and F-Measure values of the reduced-attributes IMDb++ model generated by bagging-kNN with default parameters by 0.01% (98.15% to 98.16%), 0.01%% (97.14% to 97.15%), and 0.01% (96.67% to 96.68%), respectively.

In contrast to the classification resutls of boosting-ANN, the result presented in Table 7.8 suggests that the use of boosting-kNN using default parameters results in better classification models when applied to datasets with reduced-attributes than full-attributes. Boosting-kNN to reduced-attributes datasets improved significantly F-Measure values of boosting-kNN to full-attributes datasets, from 6.46% to 14.73%.

7.2.4 Rule Induction (RI)

To implement RI as the base learner in building a boosting ensemble method, we use *AdaBoost* operator in conjunction with the *Rule Induction* operator as the learning algorithm, shown in the workflow in Figure 7.14. The *AdaBoost* operator was applied during the training sub-process of the *X-Validation* control, while the *Rule Induction* operator was applied in the sub-process of the *AdaBoost* operator. The *AdaBoost* operator produced a RI new model in each iteration, with different weights for each RI model, generating a combination of RI base models that outperformed the original model. Table H.41, H.45, I.1, I.5, and I.9 in Appendix H and I demonstrates confusion matrixs of Boosting-RI with default parameters when applied to full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Confusion matrixs of Boosting-RI with default parameters when applied to reduced-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets are presented in Table H.43, H.47, I.3, I.7, and I.11, respectively in Appendix H and I.

The use of Boosting-RI with optimized parameters when applied to the full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets results in the following confusion matrix tables: H.42, H.46, I.2, I.6, and I.10, respectively, as shown in Appendix H and I. Applied to those with reduced-attributes, Boosting-RI with optimized parameters yields confusion matrixs H.44, H.48, I.4, I.8, and I.12, respectively, as shown in Appendix H and I. Those confusion matrixs were able to visualize missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of boosting-RI either using default or optimized parameters when applied to those five full and reduced datasets as shown in Table 7.9. The table present accuracy, precision, recall, and F-Measure value for each dataset.

The result presented in Table 7.9 demonstrates that the best result of boosting-RI is obtained by applying boosting-RI with optimized parameters to the reduced-attributes IMDb++ dataset. This gained accuracy, precision, recall, and F-Measure values by 98.16%, 96.79%, 96.71%, and 96.75%, respectively. This model maintained the classification performance of the reduced-attributes IMDb++ model generated by RI with optimized parameters as a single classifier. Both the models have the same F-Measure values by 96.75%. The reduced-attributes IMDb++ model generated by the RI classifier

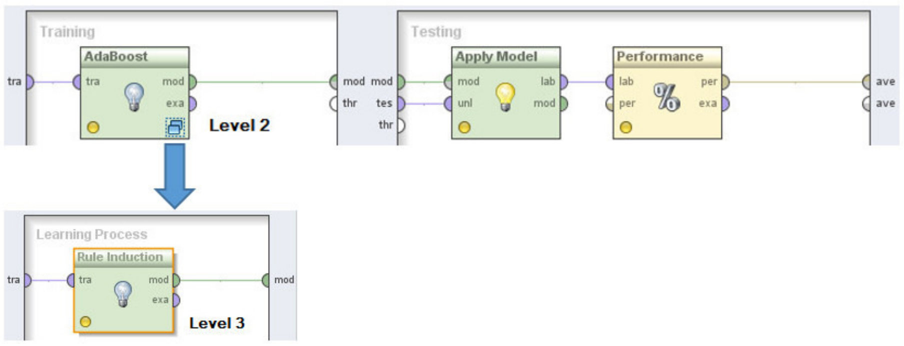


FIGURE 7.14: The Workflow for Boosting Ensemble Method with Rule Induction

improved the accuracy and the recall of the reduced-attributes IMDb++ model by generated by the boosting-RI by 0.10% (98.16% to 98.26%) and 0.22% (96.71% to 96.93%), respectively, while the precision value declined by 0.21% (96.79% to 96.58%). However, the classification performance of the reduced-attributes IMDb++ model generated by the boosting-RI was not able to improve the reduced-attributes IMDb++ model generated by bagging-RI with optimized parameters. The accuracy, precision, recall, and F-Measure values declined by 0.40% (98.56% to 98.16%), 1.37% (98.16% to 96.79%), 0.13%% (96.84% to 96.71%), and 0.75% (97.50% to 96.75%), respectively.

Table result presented in Table 7.9 suggests that the use of boosting-RI either using default or optimized parameters is more likely to result in better classification models when applied to datasets with full-attributes than reduced-attributes. Out of 10 predictive models generated by boosting-RI, there were four models that the boosting-RI to reduced-attributes datasets performed better than the boosting-RI to full-attributes datasets. Firstly, the reduced-attributes IMDb model generated by boosting-RI with optimized parameters improved the accuracy, recall and F-Measure values of the full-attributes IMDb model by 0.07% (97.86% to 97.93%), 0.45% (96.18% to 96.63%) and 0.10% (96.38% to 96.48%), respectively. Secondly, the reduced-attributes IMDb++ model generated by boosting-RI with optimized parameters increased accuracy, precision, recall and F-Measure values by 0.01% (98.15% to 98.16%), 0.01% (96.78% to 96.79%), 0.10% (96.61% to 96.71%) and 0.05% (96.70% to 96.75%), respectively. Thirdly, the reduced-attributes MovieMeter model generated by boosting-RI with optimized parameters elevated all performance measures (accuracy, precision, recall and F-Measure) by 0.22% (97.91% to 98.13%), 1.22% (95.16% to 96.38%), 1.27% (93.52% to 94.79%) and 1.25% (94.33% to 95.58%), respectively. Lastly, the reduced-attributes RottenTomatoes model generated by boosting-RI with optimized parameters enhanced accuracy, precision, recall, and F-Measure values by 0.28% (98.02% to 98.30%), 1.23% (94.05% to 95.28%), 1.41% (94.30% to 95.71%) and 1.31% (94.18% to 95.49%), respectively.

7.2.5 Support Vector Machine (SVM)

To implement SVM as the base learner to build a boosting ensemble method, we used *AdaBoost* operator in conjunction with the *SVM* operator as the learning algorithm as shown in the Figure 7.15 workflow. The *AdaBoost* operator was applied to the training sub-process of the *X-Validation* control, and the *SVM* operator was applied during the sub-process of the *AdaBoost* operator. This produced a SVM new model in each iteration with different weights for each model. AdaBoost generated a combination of SVM base models that outperformed the original model. Table J.41, J.45, K.1, K.5, and K.9 in Appendix J and K demonstrates confusion matrixs of Boosting-SVM Polynomial with default parameters when applied to full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets, respectively. Confusion matrixs of Boosting-SVM Polynomial with default parameters when applied to reduced-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets are presented in Table J.43, J.47, K.3, K.7, and K.11, respectively in Appendix J and K.

The use of Boosting-SVM Polynomial with optimized parameters when applied to the full-attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets results in the following confusion matrix tables: J.42, J.46, K.2, K.6, and K.10, respectively, as shown in Appendix J and K. Applied to those with reduced-attributes, Boosting-SVM Polynomial with optimized parameters yields confusion matrixs J.44, J.48, K.4, K.8, and K.12, respectively, as shown in Appendix J and K. Those confusion matrixs were able to visualize missclassified data in each class in each dataset. Furthermore, the confusion matrixs contributes to measure the performance of SVM Polynomial classifier either default or optimized parameters when applied to those five full and reduced datasets as shown in Table 7.10. The table present accuracy, precision, recall, and f-measure value for each dataset.

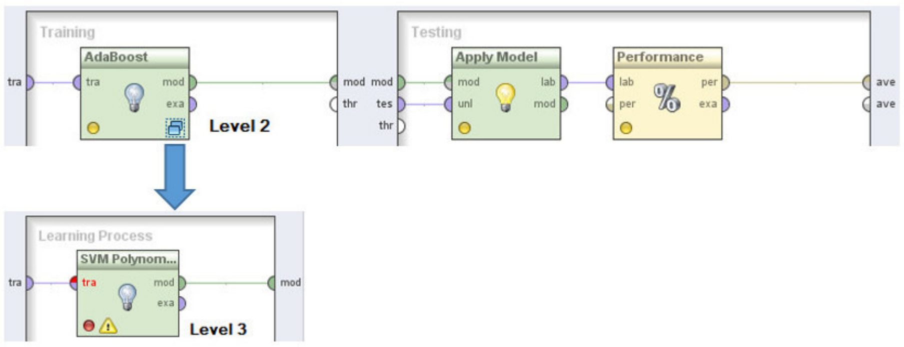


FIGURE 7.15: The Workflow for Boosting Ensemble Method with SVM

The result presented in Table 7.10 exhibits that the best result of boosting-SVM Polynomial is yielded by applying boosting-SVM Polynomial with optimized parameters to the reduced-attributes IMDb++ dataset. This attained accuracy, precision, recall, and

F-Measure values by 94.97%, 93.31%, 92.11%, and 92.71%, respectively. This model maintained the classification performance of the reduced-attributes IMDb++ model generated by SVM Polynomial with optimized parameters as a single classifier. Both the models have the same accuracy, precision, recall, and F-Measure values by 94.97%, 93.31%, 92.11%, and 92.71%, respectively. However, the classification performance of the reduced-attributes IMDb++ model generated by the boosting-SVM Polynomial was able to improve the reduced-attributes IMDb++ model generated by the bagging-SVM Polynomial with optimized parameters. The accuracy, precision, recall, and F-Measure values increased by 0.37% (94.60% to 94.97%), 0.84% (92.47% to 93.31%), 0.34%% (91.77% to 92.11%), and 0.59% (92.12% to 92.71%), respectively.

Table result presented in Table 7.10 suggests that the use of boosting-SVM Polynomial either using default or optimized parameters is more likely to result in better classification models when applied to datasets with reduced-attributes than full-attributes. Out of 10 predictive models generated by boosting-SVM Polynomial, there were two models that the boosting-SVM Polynomial to full-attributes datasets performed better than the boosting-SVM Polynomial to reduced-attributes datasets. Firstly, the full-attributes FilmAffinity model generated by boosting-SVM Polynomial with default parameters improved the recall and F-Measure values of the reduced-attributes FilmAffinity model by 1.33% (63.61% to 64.94%), and 0.41% (72.25% to 72.66%), respectively. Lastly, the full-attributes MovieMeter model generated by boosting-SVM Polynomial with default parameters enhanced recall, and F-Measure values by 0.21% (40.75% to 40.96%) and 0.06% (50.32% to 50.38%), respectively.

Table 7.6: Results of Boosting-ANN Applied to Full-Attributes (FA) and Reduced-Attributes (RA) Datasets

Datasets		Accuracy		Precision		Recall		F-Measure		Datasets		Accuracy		Precision		Recall		F-Measure	
Boosting-ANN with Default Parameters										**Boosting-ANN with Optimized Parameters**									
IMDb	FA	87.92	0.68	84.21	0.48	78.60	1.29	81.31	0.92	IMDb	FA	87.80	0.68	83.94	0.34	78.47	1.74	81.11	1.09
	RA	87.24		83.73		77.31		80.39			RA	87.12		83.60		76.73		80.02	
IMDb++	FA	90.82	0.55	88.28	0.31	83.11	1.20	85.62	0.50	IMDb++	FA	90.95	2.61	88.72	4.43	84.27	5.93	86.44	5.24
	RA	90.27		88.59		81.91		85.12			RA	88.34		84.29		78.34		81.20	
Film Affinity	FA	88.99	0.06	86.86	1.24	80.90	0.17	83.77	0.67	Film Affinity	FA	89.19	1.80	86.59	3.68	81.04	4.19	83.73	3.96
	RA	88.93		85.62		80.73		83.10			RA	87.39		82.91		76.85		79.77	
Movie Meter	FA	89.48	0.01	85.21	0.48	75.95	0.09	80.31	0.60	Movie Meter	FA	89.51	0.05	84.63	0.10	75.91	1.43	80.03	0.75
	RA	89.49		86.69		75.86		80.91			RA	89.46		84.73		74.48		79.28	
Rotten Tomatoes	FA	91.84	0.32	83.52	1.07	78.90	0.05	81.15	0.54	Rotten Tomatoes	FA	91.62	1.98	82.68	2.64	81.25	6.95	81.96	4.90
	RA	91.52		82.45		78.85		80.61			RA	89.64		80.04		74.30		77.06	

Table 7.7: Results of Boosting-DT Applied to Full-Attributes (FA) and Reduced-Attributes (RA) Datasets

Datasets		Accuracy		Precision		Recall		F-Measure		Datasets		Accuracy		Precision		Recall		F-Measure	
Boosting-DT with Default Parameters										**Boosting-DT with Optimized Parameters**									
IMDb	FA	84.38	0.02	81.68	0.03	72.83	0.10	77.00	0.07	IMDb	FA	98.09	0.22	96.59	0.53	96.41	0.40	96.50	0.46
	RA	84.40		81.71		72.93		77.07			RA	98.31		97.12		96.81		96.96	
IMDb++	FA	85.02	0.01	83.05	0.01	71.34	0.02	76.75	0.01	IMDb++	FA	98.04	0.22	96.28	0.53	96.27	0.32	96.27	0.43
	RA	85.03		83.06		71.36		76.76			RA	98.26		96.81		96.59		96.70	
Film Affinity	FA	82.01	0.17	88.21	0.59	54.32	0.49	67.24	0.54	Film Affinity	FA	98.02	0.03	96.78	0.20	96.15	0.24	96.47	0.23
	RA	82.18		88.80		54.81		67.78			RA	98.05		96.58		95.91		96.24	
Movie Meter	FA	73.10	0.00	18.28	0.00	25.00	0.00	21.12	0.00	Movie Meter	FA	97.88	0.01	94.98	0.26	93.65	0.00	94.31	0.13
	RA	73.10		18.28		25.00		21.12			RA	97.87		95.24		93.65		94.44	

Continued on next page

Table 7.7 – continued from previous page

Datasets		Accuracy		Precision		Recall		F-Measure		Datasets		Accuracy		Precision		Recall		F-Measure	
Rotten Tomatoes	FA	87.53	0.00	72.92	0.00	58.05	0.00	64.64	0.00	Rotten Tomatoes	FA	97.98	0.43	95.59	0.34	95.27	0.41	95.43	0.04
	RA	87.53		72.92		58.05		64.64			RA	98.41		95.25		95.68		95.47	

Table 7.8: Results of Boosting-kNN Applied to Full-Attributes (FA) and Reduced-Attributes (RA) Datasets

Datasets		Accuracy		Precision		Recall		F-Measure		Datasets	Accuracy	Precision	Recall	F-Measure
Boosting k-NN with a Default Parameter														
IMDb	FA	89.21	8.63	81.18	14.73	85.48	10.63	83.28	12.73					
	RA	97.84		95.91		96.11		96.01						
IMDb++	FA	91.19	6.97	84.32	11.89	88.44	8.71	86.33	10.35					
	RA	98.16		96.21		97.15		96.68						
Film Affinity	FA	89.69	8.35	82.55	13.24	86.51	10.58	84.49	11.95					
	RA	98.04		95.79		97.09		96.44						
Movie Meter	FA	90.64	7.20	80.11	12.86	86.41	9.70	83.14	11.37					
	RA	97.84		92.97		96.11		94.51						
Rotten Tomatoes	FA	91.73	6.46	82.81	11.48	84.87	7.17	83.83	9.32					
	RA	98.19		94.29		92.04		93.15						

Table 7.9: Results of Boosting-RI Applied to Full-Attributes (FA) and Reduced-Attributes (RA) Datasets

Datasets		Accuracy		Precision		Recall		F-Measure		Datasets		Accuracy		Precision		Recall		F-Measure	
Boosting-RI with Default Parameters										**Boosting-RI with Optimized Parameters**									
IMDb	FA	97.01	0.63	97.46	0.35	92.55	1.26	94.94	0.83	IMDb	FA	97.86	0.07	96.58	0.25	96.18	0.45	96.38	0.10
	RA	96.38		97.11		91.29		94.11			RA	97.93		96.33		96.63		96.48	

Continued on next page

Table 7.9 – continued from previous page

Datasets		Accuracy		Precision		Recall		F-Measure		Datasets		Accuracy		Precision		Recall		F-Measure	
IMDb++	FA	96.89	1.91	97.41	0.60	92.91	4.69	95.11	2.79	IMDb++	FA	98.15	0.01	96.78	0.01	96.61	0.10	96.70	0.05
	RA	94.98		96.81		88.22		92.32			RA	98.16		96.79		96.71		96.75	
Film Affinity	FA	97.51	0.69	98.04	0.45	92.17	1.80	95.02	1.18	Film Affinity	FA	98.09	0.30	96.85	0.48	96.20	0.96	96.52	0.71
	RA	96.82		97.59		90.37		93.84			RA	97.79		96.37		95.24		95.81	
Movie Meter	FA	95.98	0.97	97.02	0.17	82.67	3.98	89.27	2.44	Movie Meter	FA	97.91	0.22	95.16	1.22	93.52	1.27	94.33	1.25
	RA	95.01		96.85		78.69		86.83			RA	98.13		96.38		94.79		95.58	
Rotten Tomatoes	FA	96.10	0.84	96.23	1.42	83.12	1.80	89.19	1.64	Rotten Tomatoes	FA	98.02	0.28	94.05	1.23	94.30	1.41	94.18	1.31
	RA	95.26		94.81		81.32		87.55			RA	98.30		95.28		95.71		95.49	

Table 7.10: Results of Boosting-SVM Polynomial Applied to Full-Attributes (FA) and Reduced-Attributes (RA) Datasets

Datasets		Accuracy		Precision		Recall		F-Measure		Datasets		Accuracy		Precision		Recall		F-Measure	
IMDb	FA	84.33	0.47	81.63	1.68	65.61	0.03	72.75	0.67	IMDb	FA	90.97	1.47	87.53	3.38	84.80	3.17	86.14	3.27
	RA	84.80		83.31		65.64		73.42			RA	92.44		90.91		87.97		89.41	
	FA	85.26		81.50		68.81		74.62			FA	93.93		90.56		89.22		89.88	
Boosting-SVM Polynomial with Default Parameters										**Boosting-SVM Polynomial with Optimized Parameters**									
IMDb++			0.63		1.44		1.18		1.30	IMDb++			1.04		2.75		2.89		2.83
	RA	85.89		82.94		69.99		75.92			RA	94.97		93.31		92.11		92.71	
Film Affinity	FA	84.17	0.23	82.47	1.14	64.94	1.33	72.66	0.41	Film Affinity	FA	92.53	1.55	89.47	3.14	88.27	3.41	88.87	3.27
	RA	84.40		83.61		63.61		72.25			RA	94.08		92.61		91.68		92.14	
Movie Meter	FA	84.48	0.38	65.45	0.29	40.96	0.21	50.38	0.06	Movie Meter	FA	92.53	0.97	87.93	2.83	86.67	3.02	87.29	2.93
	RA	84.86		65.74		40.75		50.32			RA	93.50		90.76		89.69		90.22	
Rotten Tomatoes	FA	86.82	0.52	61.52	0.23	50.13	1.39	55.25	0.92	Rotten Tomatoes	FA	94.25	1.31	87.85	3.04	85.08	4.86	86.44	3.46
	RA	87.34		61.75		51.52		56.17			RA	95.56		90.89		89.94		89.90	

7.3 Summary

Table 7.11 summarises the classification results using bagging and boosting to k-NN with default parameters and the other classifiers (ANN, DT, RI, and SVM Polynomial) either using default or optimized parameters when applied to full and reduced attributes of IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes datasets. The use of ensemble methods with the ANN as the base classifier resulted in a more robust classification algorithm when the boosting-ANN with optimized parameters applied to the full-attributes IMDb++ dataset. This IMDb++ model achieved the best of F-Measure value by 86.44%. By using DT as the base classifier, the use of bagging ensemble method to the DT classifier with optimized parameters obtained the highest of F-Measure value by 97.28% when applied to the reduced-attributes IMDb dataset. By using k-NN as the base classifier, the best F-Measure of 96.68% was achieved by using boosting-kNN with a default parameter when applied to the reduced-attributes IMDb++ dataset. By using RI as the base classifier, the use of bagging-RI with optimized parameters applied to the full-attributes IMDb++ dataset was succesfully to achieve the more robust classification model by obtaining the highest F-Measure of 97.54%. Lastly, by using SVM Polynomial as the base classifier, a more robust classification model by F-Measure of 92.71% was obtained by employing the boosting-SVM Polynomial with optimized parameters applied to the reduced-attributes IMDb++ dataset. It can be highlighted that the use of two ensemble methods to the five different classifiers, the bagging-RI with optimized parameters attained a more robust classification model which was the full-attributes IMDb++ model.

Table 7.12 compares the classification results of single classifiers and ensemble classifiers (bagging and boosting). Interestingly, the use of ensemble methods (bagging and boosting) with the ANN either using default or optimized parameters as the base models is able to improve the classification performance of the ANN as a single classifier to all models either with full or reduced attributes. The improvements of bagging-ANN vary from 1.08% to 3.6%, while the boosting-ANN results in improvements from 0.64% to 10.52%. In the implementation of ensemble methods using DT classifier as the base model, the bagging-DT with default parameters increased the F-Measure values of the DT classifier with default parameters in the full-attributes models (FilmAffinity and RottenTomatoes) and the reduced-attributes models (RottenTomatoes) by 0.68% (67.24% to 67.92%), 5.19% (64.64% to 69.83%), and 5.19% (64.64% to 69.83%), respectively. The bagging-DT with optimized parameters outperformed the F-Measure values of the DT single classifers with optimized parameters in all full-attributes models (IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes), while the bagging-DT with optimized parameters were only able to improve the F-Measure values of the DT single classifiers with optimized parameters in the reduced-attributes MovieMeter and RottenTomatoes models by 0.09% (95.06% to 95.15%), and 0.61% (95.01% to 95.62%), respectively. The boosting-DT with default parameters failed to improve the F-Measure values of all the models either with full or reduced attributes. While the boosting-DT with optimized parameters was not able to increase the F-Measure values when applied to all full-attributes models, the boosting-DT with optimized parameters was only able to enhance the F-Measure of the reduced-attributes RottenTomatoes model by 0.46% (95.01% to 95.47%).

The result presented in table 7.12 also shows the use of the bagging-RI either using default or optimized parameters can improve the performance of almost all classification models generated by the RI single classifiers either with default or optimized parameters. There was only in the reduced-attributes FilmAffinity model that the bagging-RI with default parameters can just maintain the F-Measure value of the RI single classifier with default parameters with 84.15%. The use of boosting-RI with default parameters was fully able to improve the classification performance of the RI single classifier in all models either with full or reduced attributes. The improvements varied from 9.45% to 14.33%. However, the use of boosting-RI with optimized parametes was more likely incapable to increase the performance of the RI single classifiers. There were only two models that the boosting-RI performed better that the RI single classifiers. Firstly, the boosting-RI improved the F-Measure value of the full-attributes IMDb model by 0.1% (96.28% to 96.38%). Secondly, the boosting-RI increased the reduced-attributes MovieMeter model by 0.27% (95.58% to 95.85%). The use of SVM Polynomial as the base model either using default or optimized parameters for applying a bagging method were not able to improve the classification results of the SVM Polynomial single classifiers. The improvements were negative when they applied to datasets with full and reduced attributes. The positive improvements were found only in the reduced-attributes IMDb and MovieMeter datasets. When applied to those datasets, the bagging-SVM Polynomial with default parameters increased the F-Measure values by 0.21% (73.42% to 73.63%) and 0.01% (50.32% to 50.33%), respectively. The boosting-SVM Polynomial with default parameters was only the boosting method that was success to improve the F-Measure value of the SVM Polynomial single classifer when applied to the full-attributes IMDb dataset. The improvement reached 0.23% (72.52% to 72.75%). The rest of the boosting-SVM Polynomial either using default or optimized parameters failed to increase the F-Measure values of the SVM Polynomial single classifiers when applied to the rest of datasets either with full or reduced attributes.

Table 7.13 summarises all of experiment results conducted in this research. In order to build a more robust classification model, the ANN classifier achieved its best performance when it was used in the setting of optimized parameters as the base model of the boosting method. This obtained the F-Measure value of 86.44% when applied to the full-attributes IMDb++ dataset. The use of the DT classifier to build a more robust classification model obtained the highest of F-Measure by 97.61%. This result was gained applying the DT classifier with optimized parameters to reduced-attributes IMDb++ dataset. By using the k-NN classifier, a more robust classification model was obtained by employing the k-NN with default parameters to reduced-attributes IMDb++ dataset. This resulted in F-Measure of 96.71%. Experimentations to get a more robust classification model, the RI classifier reached its best working when it was employeed in the setting of optimized parameters as the base model of the bagging method. This attained the F-Measure value of 97.54% when applied to the full-attributes IMDb++ dataset. To build a more robust classification model, the SVM Polynomial learner with optimized parameters was better optimized when it was involved in the wrapper-type feature selection. This yielded the F-Measure value of 93.32% when applied to the reduced-attributes RottenTomatoes dataset. These results suggest that a more robust classification model is achieved by 97.61%. The robust model is gained by applying the wrapper-type feature selection suited to the DT classifier with optimized parameters to train the reduced-attributes IMDb++ dataset. As the IMDb++ dataset is built by including additional attributes from external data sources into an existing main dataset. It can be noticed that the inclusion of external data sources may lead to improve the classification performance.

Table 7.11: Summary of Classification Results of Ensemble Methods (Bagging and Boosting)

Datasets	BAGGING				BOOSTING			
	Default Parameters (DP)		Optimized Parameters (OP)		Default Parameters (DP)		Optimized Parameters (OP)	
	Full Attributes (FA)	Reduced Attributes (RA)	FA	RA	FA	RA	FA	RA
Artificial Neural Network (ANN)								
IMDb	80.11	80.36	80.25	79.54	81.31	80.39	81.11	80.02
IMDb++	83.85	83.25	83.90	81.74	85.62	85.12	86.44	81.20
Film Affinity	81.01	80.46	80.90	79.31	83.77	83.10	83.73	79.77
Movie Meter	77.65	77.40	77.37	75.13	80.31	80.91	80.03	79.28
Rotten Tomatoes	75.09	72.76	74.13	69.98	81.15	80.61	81.96	77.06
k-NN								
IMDb	83.21	96.00			83.28	96.01		
IMDb++	86.33	96.67			86.33	96.68		
Film Affinity	84.44	67.75			84.49	96.44		
Movie Meter	83.16	21.12			83.14	94.51		
Rotten Tomatoes	83.82	69.83			83.83	93.15		
SVM Polynomial								
IMDb	72.12	73.63	86.25	89.12	72.75	73.42	86.14	89.41
IMDb++	74.34	75.74	89.25	92.12	74.62	75.92	89.88	92.71
Film Affinity	72.66	72.13	88.45	91.89	72.66	72.25	88.87	92.14

Datasets	BAGGING				BOOSTING			
	DP		OP		DP		OP	
	FA	RA	FA	RA	FA	RA	FA	RA
Decision Tree (DT)								
IMDb	73.41	73.42	97.02	97.28	77.00	77.07	96.50	96.96
IMDb++	75.39	74.89	97.07	97.27	76.75	76.76	96.27	96.70
Film Affinity	67.92	67.75	96.76	96.62	67.24	67.78	96.47	96.24
Movie Meter	21.12	21.12	95.37	95.15	21.12	21.12	94.31	94.44
Rotten Tomatoes	69.83	69.83	96.18	96.62	64.64	64.64	95.43	95.47
Rule Induction (RI)								
IMDb	85.37	85.49	97.51	97.45	94.94	94.11	96.38	96.48
IMDb++	85.33	86.79	97.54	97.50	95.11	92.32	96.70	96.75
Film Affinity	86.02	84.15	97.11	97.21	95.02	93.84	96.52	95.81
Movie Meter	78.22	78.12	96.17	96.04	89.27	86.83	94.33	95.85
Rotten Tomatoes	81.89	80.05	95.71	96.59	89.19	87.55	94.18	95.49

Continued on next page

Table 7.11 – continued from previous page

Datasets	BAGGING				BOOSTING				Datasets	BAGGING				BOOSTING			
	Default Parameters (DP)		Optimized Parameters (OP)		Default Parameters (DP)		Optimized Parameters (OP)			DP		OP		DP		OP	
	Full Attributes (FA)	Reduced Attributes (RA)	FA	RA	FA	RA	FA	RA		FA	RA	FA	RA	FA	RA	FA	RA
Movie Meter	50.26	50.33	86.02	89.76	50.38	50.32	87.29	90.22									
Rotten Tomatoes	54.91	55.90	84.50	89.77	55.25	56.17	86.44	89.90									

Table 7.12: Comparison of Classification Results Between Single Classifiers (SC) and Ensemble Classifiers (Bagging (BG) and Boosting (BS)

Datasets	Default Parameters										Optimized Parameters									
	Full-Attributes					Reduced-Attributes					Full-Attributes					Reduced-Attributes				
	SC	BG	+/-	BS	+/-	SC	BG	+/-	BS	+/-	SC	BG	+/-	BS	+/-	SC	BG	+/-	BS	+/-
Artificial Neural Network (ANN)																				
IMDb	78.20	80.11	1.91	81.31	3.11	78.32	80.36	2.04	80.39	2.07	78.59	80.25	1.66	81.11	2.52	77.96	79.54	1.58	80.02	2.06
IMDb++	80.83	83.85	3.02	85.62	4.79	81.44	83.25	1.81	85.12	3.68	81.45	83.90	2.45	86.44	4.99	80.56	81.74	1.18	81.20	0.64
Film Affinity	78.56	81.01	2.45	83.77	5.21	77.91	80.46	2.55	83.10	5.19	78.93	80.90	1.97	83.73	4.80	77.90	79.31	1.41	79.77	1.87
Movie Meter	75.77	77.65	1.88	80.31	4.54	75.18	77.40	2.22	80.91	5.73	74.67	77.37	2.70	80.03	5.36	74.05	75.13	1.08	79.28	5.23
Rotten Tomatoes	71.49	75.09	3.60	81.15	9.66	71.54	72.62	1.22	80.61	9.07	71.44	74.13	2.69	81.96	10.52	66.68	69.98	3.30	77.06	10.38
Decision Tree (DT)																				
IMDb	77.00	73.41	3.59	77.00	0.00	77.07	73.42	3.65	77.07	0.00	96.64	97.02	0.38	96.50	0.14	97.56	97.28	0.28	96.96	0.60
IMDb++	76.75	75.39	1.36	76.75	0.00	76.76	74.89	1.87	76.76	0.00	96.75	97.07	0.32	96.27	0.48	97.61	97.27	0.34	96.70	0.91
Film Affinity	67.24	67.92	0.68	67.24	0.00	76.76	67.75	9.01	67.78	8.98	96.47	96.76	0.29	96.47	0.00	96.81	96.62	0.19	96.24	0.57

Continued on next page

Table 7.12 – continued from previous page

Datasets	Default Parameters										Optimized Parameters									
	Full-Attributes					Reduced-Attributes					Full-Attributes					Reduced-Attributes				
	SC	**BG**	+/-	**BS**	+/-	**SC**	**BG**	+/-	**BS**	+/-	**SC**	**BG**	+/-	**BS**	+/-	**SC**	**BG**	+/-	**BS**	+/-
Movie Meter	21.12	21.12	0.00	21.12	0.00	21.12	21.12	0.00	21.12	0.00	94.31	95.37	1.06	94.31	0.00	95.06	95.15	0.09	94.44	0.62
Rotten Tomatoes	64.64	69.83	5.19	64.64	0.00	64.64	69.83	5.19	64.64	0.00	95.90	96.18	0.28	95.43	0.47	95.01	95.62	0.61	95.47	0.46
k-NN with a Default Parameter																				
IMDb	83.47	83.21	0.26	83.28	0.19	96.01	96.00	0.01	96.01	0.00										
IMDb++	86.33	86.33	0.00	86.33	0.00	96.71	96.67	0.04	96.68	0.03										
Film Affinity	84.49	84.44	0.05	84.49	0.00	96.44	96.42	0.02	96.44	0.00										
Movie Meter	83.16	83.16	0.00	83.14	0.02	94.51	94.51	0.00	94.51	0.00										
Rotten Tomatoes	83.83	83.82	0.01	83.83	0.00	93.15	93.15	0.00	93.15	0.00										
Rule Induction (RI)																				
IMDb	82.02	85.37	3.35	94.94	12.92	83.14	85.49	2.35	94.11	10.97	96.28	97.51	1.23	96.38	0.10	96.48	97.45	0.97	96.48	0.00
IMDb++	82.75	85.33	2.58	95.11	12.36	84.80	86.79	1.99	92.32	7.52	96.70	97.54	0.84	96.70	0.00	96.75	97.50	0.75	96.75	0.00
Film Affinity	83.26	86.02	2.76	95.02	11.76	84.15	84.15	0.00	93.84	9.69	96.55	97.11	0.56	96.52	0.03	95.81	97.21	1.40	95.81	0.00
Movie Meter	74.94	78.22	3.28	89.27	14.33	75.09	78.12	3.03	86.83	11.74	94.33	96.17	1.84	94.33	0.00	95.58	96.04	0.46	95.85	0.27
Rotten Tomatoes	77.81	81.89	4.08	89.19	11.38	78.10	80.05	1.95	87.55	9.45	94.18	95.71	1.53	94.18	0.00	95.49	96.59	1.10	95.49	0.00
SVM Polynomial																				
IMDb	72.52	72.12	0.40	72.75	0.23	73.42	73.63	0.21	73.42	0.00	86.61	86.25	0.36	86.14	0.47	89.41	89.12	0.29	89.41	0.00
IMDb++	74.62	74.34	0.28	74.62	0.00	75.92	75.74	0.18	75.92	0.00	89.88	89.25	0.63	89.88	0.00	92.71	92.12	0.59	92.71	0.00
Film Affinity	72.66	72.66	0.00	72.66	0.00	72.25	72.13	0.12	72.25	0.00	88.87	88.45	0.42	88.87	0.00	92.14	91.89	0.25	92.14	0.00
Movie Meter	50.38	50.26	0.12	50.38	0.00	50.32	50.33	0.01	50.32	0.00	87.29	86.02	1.27	87.29	0.00	90.22	89.76	0.46	90.22	0.00

Continued on next page

Table 7.12 – continued from previous page

Datasets	Default Parameters										Optimized Parameters									
	Full-Attributes					Reduced-Attributes					Full-Attributes					Reduced-Attributes				
	SC	**BG**	+/-	**BS**	+/-	**SC**	**BG**	+/-	**BS**	+/-	**SC**	**BG**	+/-	**BS**	+/-	**SC**	**BG**	+/-	**BS**	+/-
Rotten Tomatoes	55.25	54.91	0.34	55.25	0.00	56.17	55.90	0.27	56.17	0.00	86.44	84.50	1.94	86.44	0.00	93.32	89.77	3.55	89.90	3.42

Table 7.13: Results of Classifiers with Default Parameters (DP) and Optimized Parameters (OP) Applied to Full-Attributes (FA) and Reduced-Attributes (RA) Datasets in F-Measure Values

Datasets	Parameter Optimiza tion		Feature Selection		BAGGING				BOOSTING			
	DP	OP	DP	OP	DP		OP		DP		OP	
	FA		RA		FA	RA	FA	RA	FA	RA	FA	RA
Artificial Neural Network (ANN)												
IMDb	78.20	78.59	78.32	77.96	80.11	80.36	80.25	79.54	81.31	80.39	81.11	80.02
IMDb++	80.83	81.45	81.44	80.56	83.85	83.25	83.90	81.74	85.62	85.12	86.44	81.20
Film Affinity	78.56	78.93	77.91	77.90	81.01	80.46	80.90	79.31	83.77	83.10	83.73	79.77
Movie Meter	75.77	74.67	75.18	74.05	77.65	77.40	77.37	75.13	80.31	80.91	80.03	79.28
Rotten Tomatoes	71.49	71.44	71.54	66.68	75.09	72.76	74.13	69.98	81.15	80.61	81.96	77.06
Decision Tree (DT)												
IMDb	77	96.64	77.07	97.56	73.41	73.42	97.02	97.28	77.00	77.07	96.50	96.96
IMDb++	76.75	96.75	76.76	97.61	75.39	74.89	97.07	97.27	76.75	76.76	96.27	96.70
Film Affinity	67.24	96.47	76.76	96.81	67.92	67.75	96.76	96.62	67.24	67.78	96.47	96.24
Movie Meter	21.12	94.31	21.12	95.06	21.12	21.12	95.37	95.15	21.12	21.12	94.31	94.44
Rotten Tomatoes	64.64	95.90	64.64	95.01	69.83	69.83	96.18	95.62	64.64	64.64	95.43	95.47
k-NN												
IMDb	83.47		96.01		83.21	96.00			83.28	96.01		
IMDb++	86.33		96.71		86.33	96.67			86.33	96.68		
Film Affinity	84.49		96.44		84.44	96.42			84.49	96.44		
Movie Meter	83.16		94.51		83.16	94.51			83.14	94.51		
Rotten Tomatoes	83.83		93.15		83.82	93.15			83.83	93.15		
Rule Induction (RI)												
IMDb	82.02	96.28	83.14	96.48	85.37	85.49	97.51	97.45	94.94	94.11	96.38	96.48
IMDb++	82.75	96.70	84.80	96.75	85.33	86.79	97.54	97.50	95.11	92.32	96.70	96.75
Film Affinity	83.26	96.55	84.15	95.81	86.02	84.15	97.11	97.21	95.02	93.84	96.52	95.81
Movie Meter	74.94	94.33	75.09	95.58	78.22	78.12	96.17	96.04	89.27	86.83	94.33	95.85
Rotten Tomatoes	77.81	94.18	78.10	95.49	81.89	80.05	95.71	96.59	89.19	87.55	94.18	95.49
SVM Polynomial												
IMDb	72.52	86.61	73.42	89.41	72.12	73.63	86.25	89.12	72.75	73.42	86.14	89.41
IMDb++	74.62	89.88	75.92	92.71	74.34	75.74	89.25	92.12	74.62	75.92	89.88	92.71
Film Affinity	72.66	88.87	72.25	92.14	72.66	72.13	88.45	91.89	72.66	72.25	88.87	92.14
Movie Meter	50.38	87.29	50.32	90.22	50.26	50.33	86.02	89.76	50.38	50.32	87.29	90.22
Rotten Tomatoes	55.25	86.44	56.17	93.32	54.91	55.90	84.50	89.77	55.25	56.17	86.44	89.90

Chapter 8

Discussion

This chapter incorporates twofold sections. First, it addresses both findings and evidences for the four research questions. Secondly, it asserts the limitations of the study.

8.1 Research Question 1

Classification is one of the knowledge discovery techniques performed in a supervised learning technique. Knowledge can be found in the primary data by using appropriate algorithms and tools. Moreover, knowledge can also be found in external data which needs to be linked to the primary data. As for the main data, i used the IMDb plain text data files and converted them into a MySQL database. By leveraging the data, we built multiple-class classifications for predicting movie ratings. We involved 16 input attributes (actor, actress, budget, cinematographer, competition, composer, costume designer, director, editor, genre, mpaa, production company, production designer, producer and writer). I then determined user ratings as an output attribute and classified this into four categories (Excellent, Average, Poor, and Terrible). We called the generated dataset, the IMDb dataset.

Subsequently, we created additional attributes from external data to enhance the value of the classification process in the main data. Following the local data being defined, i linked the data to chosen Linked Data (LD) dataset (DBpedia[1]) using a movie title considered as a linking factor. Once the IMDb local data ware linked to DBpedia, we could then explore and discover the existing links in DBpedia, pointing to the related entities in other datasets by performing SPARQL queries to the DBpedia public endpoint. We automatically extracted a Wikidata page id by using DBpedia *owl:sameAS* links to Wikidata[2].

[1]http://dbpedia.org/

[2]https://wikidata.org/

Wikidata is similar to DBpedia, comprising of a free and open knowledge base. If DBpedia can be accessed using a SPARQL query, Wikidata can be accessed programmatically by using a Wikidata API. I queried Wikidata using the Wikidata page id to collect links to BoxOfficeMojo[3], FilmAffinity[4], IMDb[5], Metacritic[6], MovieMeter[7], and RottenTomatoes[8]. These collected links are highly valuable to navigate external data sources to obtain detailed information.

Movie data (e.g., Golden Globe nominee, Golden Globe won, Academy Awards (Oscar) nominee, and Academy Awards (Oscar) won) were collected from the IMDb website. We extracted the movie data from the specific link of IMDb by applying a web scraping technique. The same approach was applied to retrieve data about opening gross and opening theaters from the BoxOfficeMojo website, and metascore from the Metacritic website. We scraped the web pages according to the specific links of BoxOfficeMojo and Metacritic, respectively, and successfully obtained new seven attributes from the three following external data sources: BoxOfficeMojo, IMDb and Metacritic. I then merged those attributes with the previous sixteen attributes from the main data source to generate the new dataset, IMDb++, with 23 input attributes in total and a single output attribute.

The final attribute to gain from external data sources is user rating. This was found in the FilmAffinity, MovieMeter, and RottenTomatoes websites, and used as an output attribute in the three new datasets: FilmAffinity, MovieMeter, and RottenTomatoes. All the input attributes of the three datasets match the input attributes of the IMDb++ dataset. To generate the FilmAffinity dataset, user rating value must be taken from the FilmAffinity website. I scraped a specific page of the website based on the link gathered from Wikidata to extract the user rating value. I also queried the user rating value in the MovieMeter website, using the MovieMeter API. We then passed the MovieMeter movie id to the API, and received the return value in JSON format, which was required to be parsed so as to obtain the useful format of user rating to generate the MovieMeter dataset. To get a user rating from the RottenTomatoes website, i leveraged the OMDb[9] API. I passed the IMDb movie id as the parameter to the OMDb API to receive user rating data, which served as an output attribute to generate the RottenTomatoes dataset.

To evaluate whether the linked data can augment the results of knowledge discovery in the IMDb dataset (the dataset generated from the main data source) or not, i compared the use of it against the IMDb++ dataset on various supervised learning techniques.

[3]http://www.boxofficemojo.com/
[4]https://www.filmaffinity.com/
[5]http://www.imdb.com/
[6]http://www.metacritic.com/
[7]https://www.moviemeter.nl/
[8]https://www.rottentomatoes.com/
[9]http://www.omdbapi.com/

8.2 Research Question 2

The second research question deals with the identification of the best approach to get the best performance model in classifying movie popularity whether using a classifier with default or optimized parameters. The results of the first experimentation (experiments performed in Chapter 5) provide evidence that IMDb++ model which has the best F-Measure value by 96.75% is the best performing model. The use of Grid Search algorithm to optimize Decision Tree classifier results in a set of optimal hyper-parameter for Decision Tree. The setting of optimal hyper-parameters is as follows: Criterion (C) of gini_index, Mimimal Size for Split (MSS) of 1, Minimal Leaf Size (MLS) of 1, and Maximal Depth (MD) of 29. The Decision Tree configured with those parameters is used to train IMDb++ dataset in order to generate IMDb++ model. As the IMDb++ dataset is built from the main data source (IMDb) and external data sources, therefore, it can be noted that the inclusion of external data in the classification model may improve classification performance of a model solely based on a single data source. The Decision Tree with optimized parameters when applied to the IMDb (a dataset solely based on a single data source) attained accuracy, precision, recall, and F-Measure values by 98.16%, 96.70%, 96.58%, and 96.64%, respectively. The Decision Tree with optimized parameters when applied to the IMDb++ dataset yielded accuracy of 98.24%, precision of 96.79%, recall of 96.72%, and F-Measure of 96.75%. The IMDb++ model enriched performance of the IMDb model in all measures (accuracy, precision, recall, and F-Measure) by 0.08%, 0.09%, 0.14%, and 0.11%, respectively.

The results of the first experimentation also indicate that the classification models using multiple data sources perform better compared to the models solely based upon a single data source. When generated using the other classifiers (Artificial Neural Network (ANN), k-NN, Rule Induciton (RI), and SVM Polynomial), IMDb++ models performed better than the IMDb models in classifying movie popularity. Being generated by ANN classifier with optimized parameters, the IMDb++ model improved accuracy, precision, recall, and F-Measure values of the IMDb model by 1.97% (86.60% to 88.57%), 2.8% (82.06% to 84.86%), 2.91% (75.40% to 78.31%), and 2.86% (78.59% to 81.45%), respectively. As built by k-NN classifier with a default parameter (k), the IMDb++ model enriched accuracy of 2% (89.19% to 91.19%), precision of 3.08% (85.36% to 88.44%), recall of 2.66% (81.66% to 84.32%) and F-Measure of 2.86% (83.47% to 86.33%) compared to the IMDb model. When trained by RI classifier with optimized parameters, the IMDb++ model increased classification performance of the IMDb model in all measures (accuracy, precision, recall, and F-Measure) by 0.14% (98.01% to 98.15%), 0.59% (96.19% to 96.78%), 0.25% (96.36% to 96.61%), and 0.42% (96.28% to 96.70%), respectively. Lastly, produced by SVM Polynomial with optimized parameters, IMDb++ outperformed IMDb model in all performance measures with 2.66% accuracy (91.27% to 93.93%), 2.55% precision (88.01% to 90.56%), 3.96% recall (85.26% to 89.22%), and 2.27% accuracy (86.61% to 89.88%).

In terms of the performances of the Decision Tree with default and optimized parameters, the use of the Decision Tree with optimized parameters improves the classification results remarkably compared to the Decision Tree with default parameters. The IMDb model built by the Decision Tree with optimized parameters enhanced accuracy, precision, recall, and F-Measure values by 13.78% (84.38% to 98.16%), 15.02% (81.68% to 96.70%), 23.75% (72.83% to 96.58%), and 19.64% (77% to 96.64%), respectively compared to the IMDb model generated by the Decision Tree with default parameters. The use of parameter optimization to the Decision Tree boosted performances of the IMDb++ model compared to to IMDb++ built by the Decision Tree without parameter optimization. There was significant improvements of accuracy, precision, recall, and F-Measure values by 13.22% (85.02% to 98.24%), 13.74% (83.05% to 96.79%), 25.38% (71.34% to 96.72%), and 20% (76.75% to 96.75%), respectively. In addition, the results of the other three classifiers (ANN, RI, and SVM Polynomial) also indicate that those classifiers perform better when using optimized parameters rather than the default parameters when applied to both IMDb and IMDb++ datasets. It was found that the ANN with optimized parameters slightly increased the performances of the ANN with default parameters when applied to the IMDb and IMDb++ datasets. It can be noticed that a F-Measure value of the IMDb model just increased 0.39% (78.20% to 78.59%), while F-Measure value of the IMDb++ model increased 0.62% (80.83% to 81.45%). Contrarily, the use of RI and SVM Polynomial classifiers with optimized parameters have improved significantly the performance of those classifiers with default parameters when applied to the IMDb and IMDb++ model. RI with optimized parameters improved F-Measure values of the IMDb and IMDb++ by 14.26% (82.02% to 96.28%) and 13.95% (82.75% tp 96.70%), respectively, while SVM Polynomial with optimized parameters increased F-Measure values the IMDb of 14.09% (72.52% to 86.61%) and IMDb++ of 15.26% (74.62% to 89.88%). These findings support previous studies conducted in applying Grid Search algorithm to optimize SVM parameters (Larochelle et al. (2007) and Decision Tree parameters (Camilleri and Neri (2014). These findings may also complement the existing studies in parameter optimization, espececially, the use Grid Search algorithm to optimize parameters of ANN and RI classifiers.

The classification models based on a main data source and additional attributes from external data sources (IMDb++ model) have achieved a good level of performance classification when compared to the previous studies to classify movie popularity. In a previous study by Asad et al. (2012), they used downloadable list files from IMDb website to classify movie popularities and employed the following two classifiers: Decision Tree (C45) and Decision Rules (PART). The study obtained model accuracies of 77.36% and 77.7234%, respectively. In another study, Latif and Afzal (2016) gained data from IMDb and Wikipedia and they built movie popularity classification models using the following classifiers: Simple Logistic, Logistic regression, Decision tree (J48), Naïve Bayes, PART, and Neural Network. They attained 84.34% and 84.15%, 82.42%, 79.66%, 79.52%, and 79.07%, respectively, of model accuracies. Comparatively, I achieved the model accuracy of the classification model based on a single data source (IMDb model) by 98.16%, and 98.24% of the model accuracy by including external data sources (IMDb++ model). This study denotes favorable results compared to the previous researches.

8.3 Research Question 3

The third research question (What is the best model solely based on a single data source or with additional variables from external data sources?) was aimed to unfold the wrapper-type feature selection algorithm, that can be used to that can be used to produce the best performing model in classifying movie popularity. The results of second experimentation (experiments performed in Chapter 6) indicate that IMDb++ which has the best F-Measure value by 97.61% is the best performing model. This result improved F-Measure of the best performing model described in 8.2 by 0.86% (96.75% to 97.61%). The application of wrapper-type feature selection algorithm, Genetic Algorithm (GA) suited to Decision Tree classifier with optimized parameters has selected the most significant attributes for the model development. Out of 23 full-attributes IMDb++ dataset, GA elliminated 7 atttributes and only selected the following 13 attributes: budget, composer, director, distributor, editor, mpaa, opening gross, opening theaters, oscar nominee, production company, product designer, producer, and writer. Out of 13 selected-attributes, there were three following attributes: opening gross, opening theaters, and oscar nominee which were from external data sources. Meanwhile, GA suited to DT with optimized parameters has selected ten most important attributes (budget, cinematographer, composer, costume designer, director, distributor, production company, product designer, producer, and writer) of IMDb dataset. When DT with optimized parameters applied to IMDb and IMDb++ datasets with reduced-attributes, the IMDb++ model slightly improved the performances of IMDb model in all measures (accuracy, precision, recall, and F-Measure) by 0.06% (98.47% to 98.53%), 0.07% (97.73% to 97.80%), 0.02% (97.40% to 97.42%), and 0.05% (97.56% to 97.61%), respectively. Consistenly, the IMDb++ models outperformed the IMDb models either in the setting of full or reduced attributes. Therefore, it is noteworthy that the inclusion of attributes from external data sources may improve classification performance of the model solely based on a single data source (IMDb).

The results of the second experimentation also supporthe proposition that the classification models using multiple data sources perform better compared to the models solely based upon a single data source. When generated using the other classifiers (ANN, k-NN, RI, and SVM Polynomial), the IMDb++ models performed better than IMDb models in classifying movie popularity. Generated by ANN classifier with optimized parameters, IMDb++ model improved accuracy, precision, recall, and F-Measure values of IMDb model by 1.64% (86.46% to 88.10%), 1.66% (83.21% to 84.87%), 3.33% (73.33% to 76.66%), and 2.60% (77.96% to 80.56%), respectively. Built by k-NN classifier with a default parameter (k), the IMDb++ model increased accuracy of 0.34% (97.84% to 98.18%), precision of 1.13% (96.11% to 97.24%), recall of 0.28% (95.91% to 96.19%) and F-Measure of 0.70% (96.01% to 96.71%) compared to IMDb model. Tested by RI classifier with optimized parameters, IMDb++ model increased the performance classification of IMDb model in all measures (accuracy, precision, recall, and F-Measure) by 0.33% (97.93% to 98.26%), 0.25% (96.33% to 96.58%), 0.60% (96.63% to 96.93%), and 0.27% (96.48% to 96.75%), respectively. Lastly, produced by SVM Polynomial with optimized parameters, IMDb++ outperformed the IMDb model in all performance measures which are 2.53% accuracy (92.44% to 94.97%), 2.40% precision (90.91% to 93.31%), 4.14% recall (87.97% to 92.11%), and 3.30% accuracy (89.41% to 92.71%).

Table 6.22 in Chapter 6 provides evidences that the use of wrapper-type feature selection suited to DT, RI, and SVM Polynomial classifiers with optimized parameters resulted in better classification results compared to those classifiers with default parameters. In contrast, the application of wrapper-type feature selection suited to ANN classifier with default parameters performed better compared to ANN with optimized parameters. This results suggest that to get optimal results in applying wrapper-type feature selection, it is required to prepare two pre-processing steps. Firstly, it is recommended to select a proper parameter optimization algorithm or to refer to common values used in existing literatures. This step is likely more urgent to apply when we employ DT, RI, and SVM classifiers in wrapper-type feature selection. Secondly, it is needed to set up a set of optimal parameters to those classifiers or simply to use default parameters for ANN classifier.

In terms of performances of classifiers applied to full and reduced attributes of IMDb and IMDb++ datasets, the following classifiers (DT, RI, and SVM Polynomial) with optimized parameters performed better when applied to reduced-attributes than full-attributes. When built by DT with optimized parameters, the IMDb model with fewer attributes improved accuracy, precision, recall, and F-Measure values of IMDb model with full attributes by 0.31% (98.16% to 98.47%), 0.03% (96.70% to 96.73%), 0.82% (96.58% to 97.40%), and 0.92% (96.64% to 97.56%), respectively, while the IMDb++ model with reduced attributes increased full-attributes of IMDb++ model in accuracy of 0.29% (98.24% to 98.53%), precision of 1.01% (96.79% to 97.80%), recall of 0.70% (96.72% to 97.42%), and F-Measure of 0.86% (96.75% to 97.61%). In RI classifier with optimized parameters, the reduced-attributes IMDb model raised precision, recall, and F-Measure values of full-attributes of IMDb model by 0.14% (96.19% to 96.33%), 0.27% (96.36% to 96.63%), and 0.2% (96.28% to 96.48%), respectively. The IMDb++ model with a lesser number of attributes outperformed full-attributes of IMDb++ model in performance measures (accuracy, recall, and F-Measure) by 0.11% (98.15% to 98.26%), 0.32% (96.61% to 96.93%), and 0.05% (96.70% to 96.75%), respectively. When generated by SVM Polynomial with optimized parameters, reduced-attributes of IMDb model improved accuracy, precision, recall, and F-Measure values of IMDb model with full attributes by 1.17% (91.27% to 92.44%), 2.90% (88.01% to 90.91%), 2.71% (85.26% to 87.97%), and 2.80% (86.61% to 89.41%), respectively, while IMDb++ model with fewer attributes increased full-attributes IMDb++ model in accuracy of 1.04% (93.93% to 94.97%), precision of 2.75% (90.56% to 93.31%), recall of 2.89% (89.22% to 92.11%), and F-Measure of 2.83% (89.88% to 92.71%). These findings support the previous work conducted by Kim et al. (2015) in applying Genetic Algorithm to get a set of optimal attributes to forecast movie box office. The findings may also provide an insight that in some circumstances the inclusion of DT, RI, and SVM Polynomial classifiers with default parameters in wrapper-type feature selection is not sufficient. Hence, it is more likely to use Genetic Algorithm suited to DT, RI, and SVM Polynomial with optimized parameters to get optimal results in applying wrapper-type feature selection.

8.4 Research Question 4

The fourth research question (What is the best model solely based on a single data source or with multiple sources?) is intended to identify the use of ensemble (bagging and boosting) able to improve performance classification of previous approaches in order to get the best performing model in classifying movie popularity. The results of third experimentation (experiments performed in Chapter 7) provide evidence that IMDb++ model is the best performing model as it has the best F-Measure value by 97.54%. The IMDb++ model has been produced by Bagging-RI with optimized parameter applied to IMDb++ dataset (based on an IMDb data source and external data sources) with full-attributes. It has details of performance measure as follows: accuracy of 98.71%, precision of 98.24%, and recall of 96.86%. This result was slightly better compared to IMDb model generated by Bagging-RI with optimized parameters using full-attributes IMDb dataset. This model has F-Measure of 97.51%, while the other performance measures were accuracy, precision, and recall by 98.71%, 98.02%, and 97.01%, respectively. The IMDb++ model has better performance compared to the best performance model (IMDb++) generated in the first experimentation, from 96.75% to 97.54%. However, the IMDb++ model has a worse performance than the best performing model produced in the second experimentation (IMDb++ model which has F-Measure by 97.61%). These evidences bring up a conclusion that out of all experiments have been conducted in this thesis, the IMDb++ model (the best performing model generated in the second experimentation) with F-Measure of 97.61% is the best performing model for classifying movie popularity.

The use of bagging and boosting to k-NN classifier with default parameters results in better performance of IMDb++ models than IMDb models in the setting of full-attributes and reduced-attributes. Consistently, the application of bagging and boosting to ANN and SVM Polynomial classifiers either using default or optimized parameters provided better classification results of IMDb++ models than IMDb models in the setting of full and reduced attributes. Contrarily, in DT and RI classifiers, the use of bagging attained partial results that IMDb++ models are better than IMDb models. In the DT classifier, Bagging-DT with optimized parameters achieved better F-Measure value of the IMDb model with reduced attributes (97.28%) than IMDb++ model (97.27%). The IMDb models outperformed IMDb++ models when boosting-DT with default and optimized parameters were applied to each reduced and full attributes IMDb and IMDb++ datasets. In the RI classifier, Bagging-RI with default parameters applied to the IMDb model with full-attributes obtained F-Measure of 85.37% which was better than F-Measure of the full-attributes IMDb++ model (85.33%). Boosting-RI with default parameters applied to the reduced-attributes IMDb model gained F-Measure of 94.11% which was better than F-Measure of the reduced-attributes IMDb++ model (92.32%).

In terms of the comparation of classification performance between single classifiers and ensemble classifiers (presented in Table 7.12 in Chapter 7) , the use of bagging and boosting to ANN classifier have constanly improved the classification performance of ANN as single classifier, while k-NN as single classifer surpassed the classification performance of bagging k-NN and boosting k-NN. In ANN classifier, the improvement of F-Measure varies from 0.64% to 10.38%. The highest improvement was 10.38% where the boosting ANN with optimized parameters increased F-Measure value of the reduced-attributes of RottenTomatoes model from 66.68% to 77.06%. The smallest improvement by 0.64% occured at boosting-ANN with optimized parameters when applied to the reduced-attributes of IMDb++ dataset, from 80.56% to 81.20%. Out of 10 classification models built by bagging-DT with default parameters, it was found that there were 3 positive improvements, 2 neutral improvements, and 4 negative improvements. RottenTomatoes models either using full or reduced attributes contributed to each improvement of 5.19%, from 64.64% to 69.83%. In the bagging-DT with optimized parameters, there ware 7 positive improvements and 3 negative improvements. The highest improvement was achieved by reduced-attributes RottenTomatoes model of 0.61% (95.01% to 95.62%). There ware 9 neutral improvements and 1 negative improvement in the use of boosting-DT with optimized parameters, while boosting-DT with optimized parameters attained 1 positive improvement, 2 neutral improvements, and 7 negative improvements.

Similar to ensemble methods with DT classifier, the use of ensemble methods (bagging and boosting) to Rule Induction (RI) and SVM Polynomial provided inconsistent improvment in the performance of single classifiers. The Ensemble methods to RI with default parameters yielded 19 positive improvements and 1 neutral improvement. The best improvement was achieved by boosting-RI of 14.33% (74.94% to 89.27%) when applied to the full-attributes MovieMeter dataset, while the lowest improvement was obtained by bagging-RI of 1.95% (78.10% to 80.05%) when applied to the reduce-attributes RottenTomatoes dataset. The application of ensemble method to RI with optimized parameters resulted in 12 positive improvements, 7 neutral improvements, and 1 negative improvement. The improvements varied from 0.27% to 1.84%. The highest improvement, from 94.33% to 96.17%, was achieved by bagging-RI with optimized parameters to the full-attributes MovieMeter, while the smallest improvement was obtained by boosting-RI with optimized parameters to the reduced-attributes MovieMeter, from 95.58% to 95.85%. Lastly, the use of bagging and boosting to SVM Polynomial attained 3 positive improvements, 18 neutral improvement, and 19 negative improvements. Boosting-SVM Polynomial with default parameters contributed to the highest improvement of 0.23% (72.52% to 72.75%) when applied to the full-attributes IMDb dataset, while the lowest improvement of 0.01% (50.32% to 50.33%) was contributed by boosting-SVM Polynomial with default parameters to the reduced-attributes MovieMeter dataset.

8.5 Limitations of the Current Research Study

This section highlights several limitations found in this research. These are utterly spelled out in the bullet-points below.

- For the purpose of this research, some movie-related data were gathered by employing web srapers, which may conflict with the terms of service of data sources. We address this issue by obeying the robots.txt convention in extracting data from web pages using web scrapers. In the future research studies, it is recommended to use data sources which provide data accessing tools to reduce the use of web scrapers.

- All of the three experimentations in this research contained four unbalanced datasets (IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes). Each of dataset consists of 16008 instances and was classified into four class labels for classifying movie popularity, such as: excellent, average, poor, and terrible. Both of the IMDb and IMDb++ dataset contained 1596 instances in the excellent class, 11681 instances in the average class, 2382 instances in the poor class, and 349 instances in the terrible class. The FilmAffinity dataset contained 967 instances in the excellent class, 11179 instances in the average class, 3484 instances in the poor class, and 378 instances in the terrible class. The MovieMeter dataset contained 408 instances in the excellent class, 11702 instances in the average class, 3653 instances in the poor class, and 245 instances in the terrible class. The RottenTomatoes dataset contained 2257 instances in the excellent class, 12356 instances in the average class, 1310 instances in the poor class, and 85 instances in the terrible class. It was difficult to have a balanced dataset as we had to rely on users of movie-related web sites who willingly participate to vote their favourite movies. Though, the current research study has overcome this issue by applying three approaches (use linked data to get additional attributes from external data sources, employing advance data mining techniques, and considering to use accuracy, precision, recall, and F-Measure values to validate the classification performance), the future research studies are recommended to use balance datasets to develop more robust classification models.

- FilmAffinity, MovieMeter, and RottenTomatoes websites do not provide information of movie ratings based on user votes as complete as IMDb website. Therefore, it is difficult to make a favorable comparison among classification models based on IMDb, FilmAffinity, MovieMeter, and RottenTomatoes. In the future studies, it is recommended to get new resources that provide information of movie ratings as complete as IMDb.

- The formats of the main data source (IMDb list files) used in this research study are in a number of inconsistently structured text files. Therefore, it is required to transform those files into a format suitable for data mining. To adress this issue, I used a third-party tool, namely JMDb. This tool imports IMDb list files into a MYSQl database. However, JMDB does not support updating data. It is required to reimport and recreate a new database for updating the data. In the future research, it is recommended to provide tools or techniques to transform IMDb list files into a database as well as features to update the data without creating a database.

- There is no information of links to IMDb website in the IMDb list files. We coped this issue by investigating several string matching techniques to match movies title in the IMDb list file and IMDb website. When those links created, it is possible to extend connection with an external data source, such as DBpedia, as DBpedia is built based on linked data technology. Therefore, it is possible to exploit the potentiality of linked data technology to get movies-related resources. In the future research of this study, it is recommended to wrap a number of movies-title string matching techniques into a tool that allows to perform string matching systematically and automatically.

- ANN Parameter Optimization using Grid Search is incapable of improving classification performance significantly. In the future research, it is recommended to investigate some methods to improve ANN parameter optimization.

- The performance of Grid Search to optimize Decision Tree, Rule Induction, and SVM Polynomial is very powerful and able to improve the classification remarkably. However, the execution speed is very slow. In the future research, it is recommended to investigate some methods to speed up the performance of Grid Search.

- Wrapper-type feature selection suited to Rule Induction and SVM Polynomial with optimized parameters can perform significantly to improve classification results. However, the execution time takes very long. Therefore, it is recommended to investigate some methods to speed up the execution time of feature selection processes.

Chapter 9

Conclusions and Future Works

This chapter underpins some points in four sections. Section 9.1 reviews the purpose of this PhD research. Section 9.2 outlines the major findings of the research and section 9.3 makes recommendations for future research based on the findings of the current research. Eventually, section 9.4 concludes this thesis.

9.1 Purpose of this PhD Research

The purposes of this PhD research are to examine the capability of linked data technologies in supporting data mining processes, (2) to investigate the types of the best supervised learning techniques in classifying movie popularity, and (3) to develop movie popularity classification models using a single data source and multiple data sources by employing single classifiers (Artificial Neural Network (ANN), Decision Tree (DT), k-NN, Rule Induction (RI), and SVM Polynomial), parameter optimization using Grid Search algorithm, wrapper-type feature selection using Genetic Algorithm and Particle Swarm Optimization suited to the classifiers with default and optimized parameters, and ensemble methods (bagging and boosting to classifiers with default and optimized parameters applied to full and reduced datasets).

9.2 Major Findings

Based upon the results of the experimentations, we can define the major findings of this research study as follows:

- The creation of additional attributes from external data sources can be done by leveraging linked data technologies and performing links discovery. SPARQL queries were used to extract Wikidata links from DBpedia data. Wikidata API can be used to elicit the corresponding Wikidata page. The Wikidata page may provide links to the external data sources, such as: IMDb, BoxOfficeMojo, FilmAffinity, Metacritic, MovieMeter, and RottenTomatoes. Based on IMDb links, I extracted Golden Globe Nominee, Golden Globe Won, Oscar Nominee, and Oscar Won attributes. I extracted Opening Gross and Opening Theaters attributes from a specific page in BoxOfficeMojo based on the corresponding link. Lastly, I extracted Metascore attribute from Metacritic website according link displayed in the Wikidata page. As a result, seven additional attributes were obtained from external sources entailing Golden Globe Nominee, Golded Globe Won, Oscar Nominee, Oscar Won, Opening Gross, Opening Theaters, and Metascore..

- To enhance the value of data mining processes, I link IMDb database (transformed from IMDb list files) and IMDb website by matching movie titles. The matching process contains 24 different steps and successfuly matches 88048 movie titles. As the link was created, I can mix 16 original attributes from IMDb database and seven additional attributes from external data sources (IMDb website = 4 attributes; BoxOfficeMojo = 2 attributes; Metacritic = 1 attribute) to build classification models in classifying movie popularity.

- From the result of the first experimentation, I found that movie popularity classification model generated by Decision Tree with optimized parameters applied to the full-attributes IMDb++ dataset is the best performing model. This model achieves the highest F-Measure value of 96.75%, while the other three measurements (accuracy, precision, and recall) obtain 98.24%, 96.79%, and 96.72%, respectively. The use of Grid Search is to optimize Decision Tree results in the best configuration of Decision Tree parameters as follows: criterion set with gini_index, minimal size for split of 1, minimal leaf size of 1, and maximal depth of 29. This finding suggests that the use of parameter optimization using Grid Search algorithm is able to improve remarkably the classification performance of Decision Tree, Rule Induction and SVM Polynomial classifiers with default parameters, while parameter optimization does not improve significantly the classifiaction performance of Artificial Neural Network with default parameters.

- From the result of the second experimentation, we found that the use of wrapper-type feature selection using Genetic Algorithm (GA) suited to Decision Tree with optimized parameters attained the best peforming model when applied to IMDb++ dataset. This model achieves the best F-Measure of 97.61%, while the other measurements are accuracy of 98.53%, precision of 97.80%, and recall of 97.42%. We also figured out that this approach succesfully selected the most significant attributes for the model development. Out of 23 full-attributes IMDb++ dataset, GA elliminated 7 atttributes and only selected the following 13 attributes: budget, composer, director, distributor, editor, mpaa, opening gross, opening theaters, oscar nominee, production company, product designer, producer, and writer. Out of

13 selected-attributes, three following attributes: opening gross, opening theaters, and oscar nominee are from external data sources. Therefore, it can be said that the use of wrapper-type selection using Genetic Algorithm suited to Decision Tree, Rule Induction and SVM Polynomial is likely more able to improve classification performance significantly than those classifiers applied to full-attributes datasets. In addition, the use of wrapper-type selection using Genetic Algorithm suited to Decision Tree, Rule Induction and SVM Polynomial with optimized parameters perform better than using default parameters.

- From the result of the third experimentation, we discovered that among two ensemble methods: bagging and boosting, bagging performed better than boosting. The use of bagging Rule Induction with optimized parameters applied to the full-attributes IMDb dataset resulted in the highest F-Measure value by 97.54%, while the other measurements achieved accuracy of 98.51%, precision of 98.24%, and recall of 96.86%. It can be said that the use of ensemble methods (bagging and boosting) to Decision Tree, Rule Induction and SVM Polynomial with optimized parameters is more likely to perform better than the use of ensemble methods to those classifiers with default parameters.
- Consistently, the results of first, second and third experimentations indicate that the most robust classification model for each experimentation was the one built based on the IMDb++ dataset. It is built by including additional attributes from external data sources into a main-single data source (IMDb). Therefore, conclusively, combining a main data source and external data sources can better support in classifying movie popularity. The advantages are not only to improve the classification performance but also to lead to new insights.

9.3 Future Work

Based on the literature reviewed and the three experimentation presented and discussed in this thesis, the following recommendations are made for the future research:

- The current research used some data from external data sources for either integration or creation additional attributes using web scrapers. Future research should avoid using web scrapers and focus to use external open data sources or official data-accessing tools.
- The current research created additional attributes based on data from websites. Future research can consider adding data from social media channels.
- The use of Grid Search to optimize ANN parameters did not result in satisfactory improvements. Future research should investigate some methods to improve ANN parameter optimization.

- The current research focused only to use linked data technologies to get movie-related data sources in order to create additional attributes from external data sources. Future research can consider investigating the capabilities of linked data in supporting data mining. algorithms.
- The processing time of Grid Search to optimize Decision Tree, Rule Induction, and SVM Polynomial is extremely slow. Further research can consider using Evolutionary Algorithm or Apache Hadoop to speed up the time processing without compromising the predictive ability of the model.
- The processing time of wrapper-type feature selection using Genetic Algorithm suited to Rule Induction and SVM Polynomial with optimized parameters are extremely slow and it may lead to very long execution time. Further research should investigate the use of Apache Hadoop or other methods to speed up the execution time.

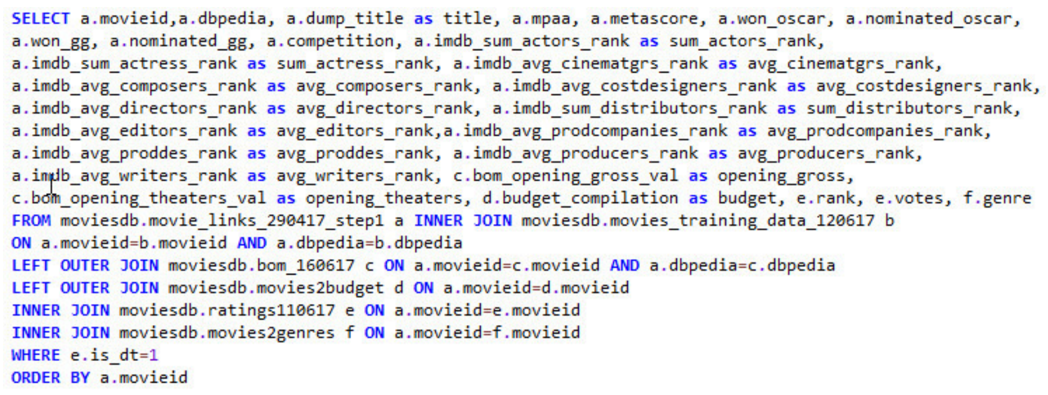
9.4 Conclusions

This thesis sought to examine the capability of linked data technologies in supporting data mining processes to develop more robust movie popularity classification models. Linked data techonologies provide supporting flexibilities to extend a main data source by linking it to external movie-related data sources in order to create additional attributes from those sources. Therefore, this will bring up a new dataset. Moreover, the aim has been to develop movie popularity classification models based solely on a main data source and multiple data sources (combining a main data souce and additional attributes from external data sources). This thesis also attempted to examine the capability of supervised learning techniques in producing more robust models in classifying movie popularity. The results of the experimentations confirm that classification model using multiple data sources perform better in classifying movie ratings than using a single data source. Meanwhile, the best approaches to produce more robust movie popularity classification models are by combining Decision Tree classifier, Grid Search, and wrapper-type feature selection using Genetic Algorithm. Grid Search is utilized to optimize Decision Tree parameters. Subsequently, applying wrapper-type feature selection using Genetic Algotihm suited to Decision Tree with optimized parameters is aimed to select the most significant attributes for developing movie popularity classification models. Lastly, the robust classification model can be achieved by employing Decision Tree with optimized parameters to train the reduced-attribute dataset.

Appendix A

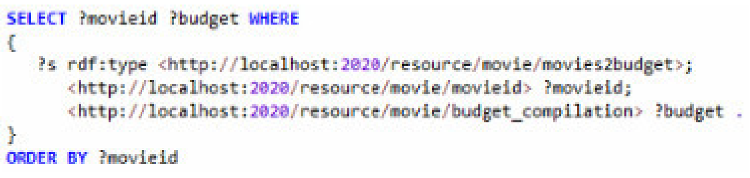
SQL and SPARQL Queries

This appendix lists all SQL and SPARQL queries which are used in this research.



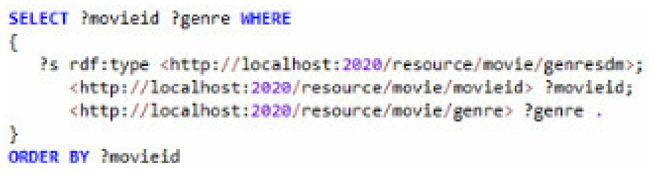
```
SELECT a.movieid,a.dbpedia, a.dump_title as title, a.mpaa, a.metascore, a.won_oscar, a.nominated_oscar,
a.won_gg, a.nominated_gg, a.competition, a.imdb_sum_actors_rank as sum_actors_rank,
a.imdb_sum_actress_rank as sum_actress_rank, a.imdb_avg_cinematgrs_rank as avg_cinematgrs_rank,
a.imdb_avg_composers_rank as avg_composers_rank, a.imdb_avg_costdesigners_rank as avg_costdesigners_rank,
a.imdb_avg_directors_rank as avg_directors_rank, a.imdb_sum_distributors_rank as sum_distributors_rank,
a.imdb_avg_editors_rank as avg_editors_rank,a.imdb_avg_prodcompanies_rank as avg_prodcompanies_rank,
a.imdb_avg_proddes_rank as avg_proddes_rank, a.imdb_avg_producers_rank as avg_producers_rank,
a.imdb_avg_writers_rank as avg_writers_rank, c.bom_opening_gross_val as opening_gross,
c.bom_opening_theaters_val as opening_theaters, d.budget_compilation as budget, e.rank, e.votes, f.genre
FROM moviesdb.movie_links_290417_step1 a INNER JOIN moviesdb.movies_training_data_120617 b
ON a.movieid=b.movieid AND a.dbpedia=b.dbpedia
LEFT OUTER JOIN moviesdb.bom_160617 c ON a.movieid=c.movieid AND a.dbpedia=c.dbpedia
LEFT OUTER JOIN moviesdb.movies2budget d ON a.movieid=d.movieid
INNER JOIN moviesdb.ratings110617 e ON a.movieid=e.movieid
INNER JOIN moviesdb.movies2genres f ON a.movieid=f.movieid
WHERE e.is_dt=1
ORDER BY a.movieid
```

Figure A.1: SQL Query to retrieve IMDB movies data



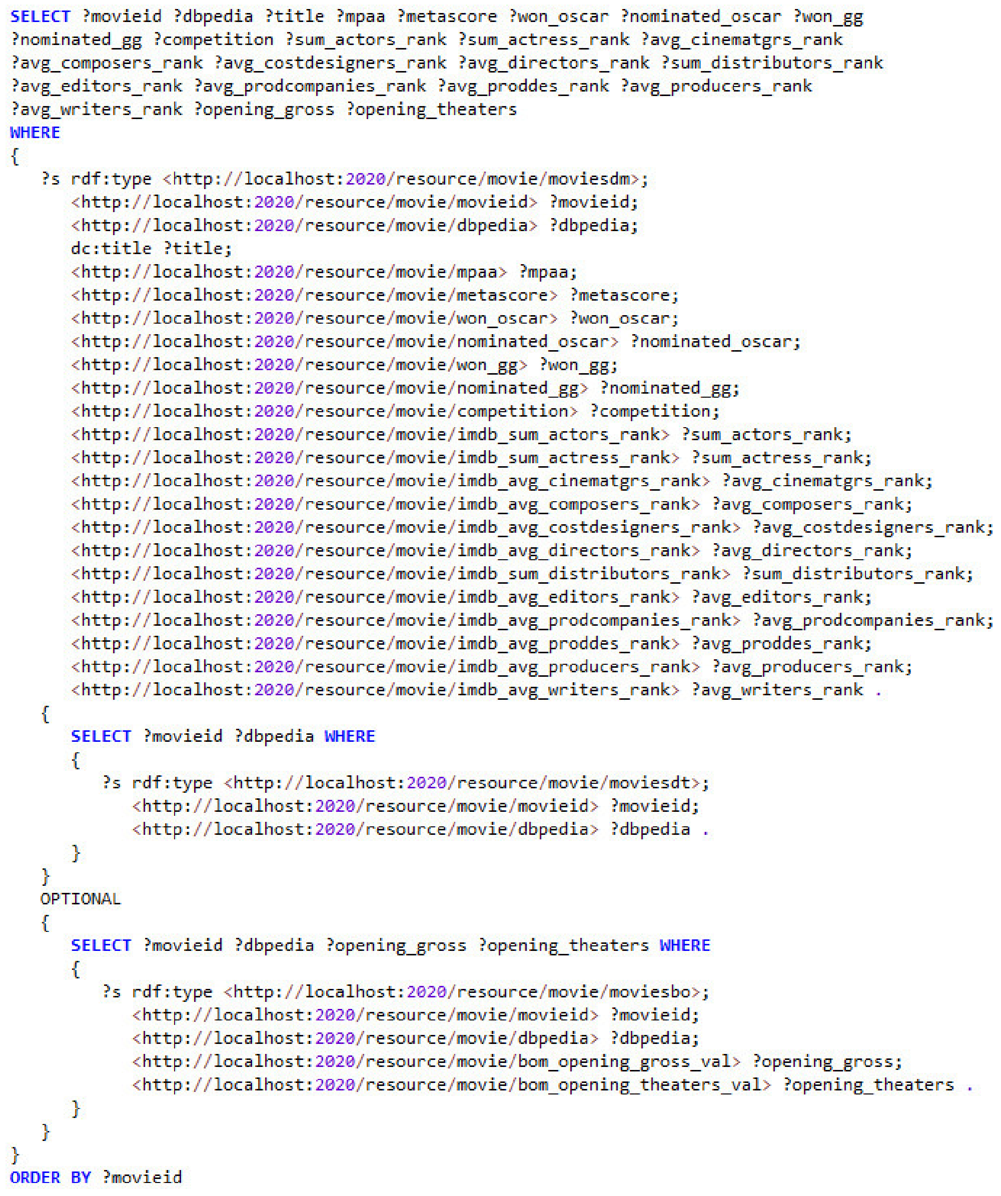
```
SELECT ?movieid ?budget WHERE
{
   ?s rdf:type <http://localhost:2020/resource/movie/movies2budget>;
      <http://localhost:2020/resource/movie/movieid> ?movieid;
      <http://localhost:2020/resource/movie/budget_compilation> ?budget .
}
ORDER BY ?movieid
```

Figure A.2: SPARQL Query to retrieve movies budget data



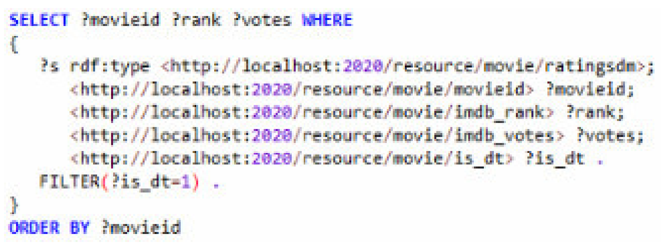
```
SELECT ?movieid ?genre WHERE
{
   ?s rdf:type <http://localhost:2020/resource/movie/genresdm>;
      <http://localhost:2020/resource/movie/movieid> ?movieid;
      <http://localhost:2020/resource/movie/genre> ?genre .
}
ORDER BY ?movieid
```

Figure A.3: SPARQL Query to retrieve movies genre data



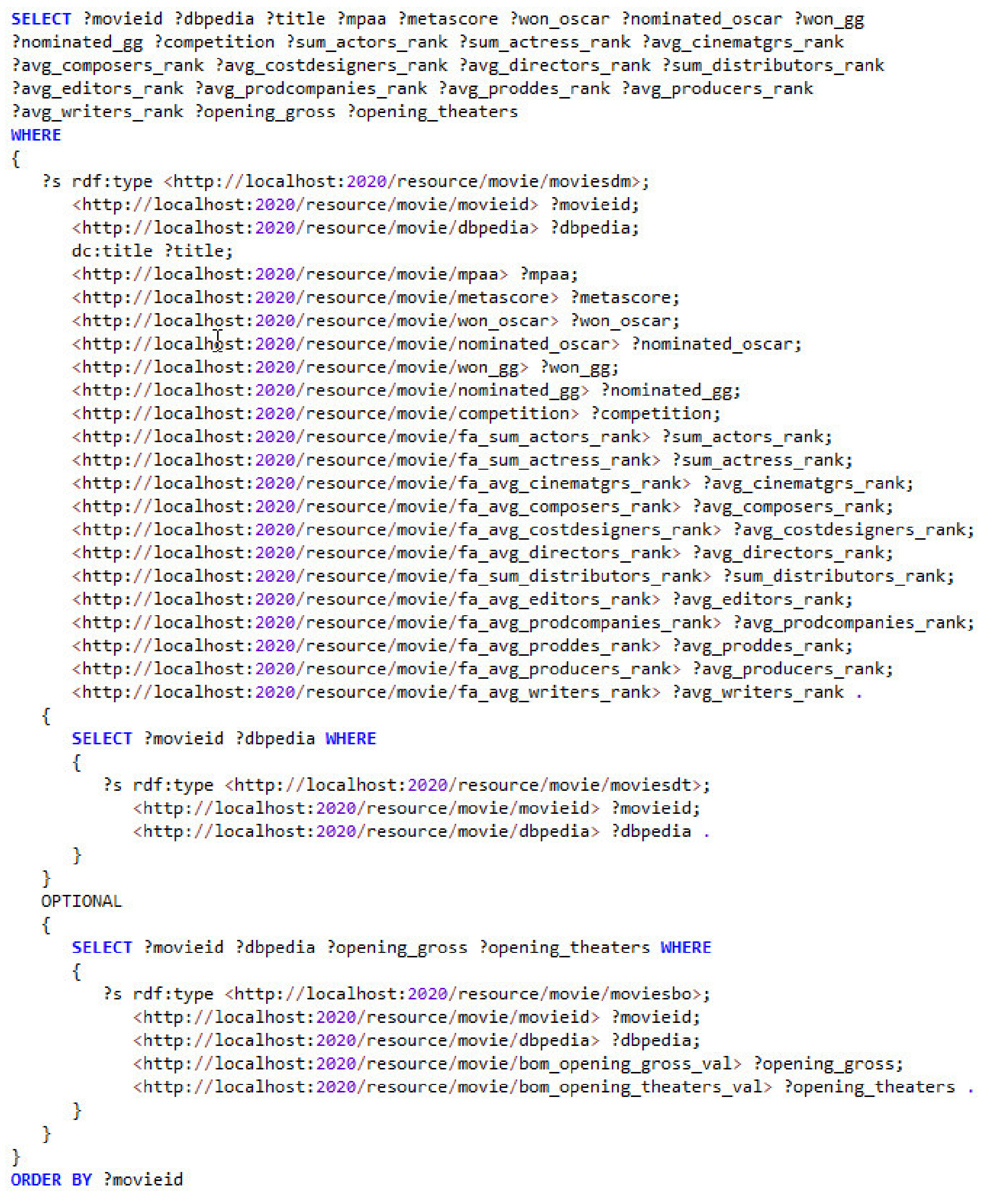
```
SELECT ?movieid ?dbpedia ?title ?mpaa ?metascore ?won_oscar ?nominated_oscar ?won_gg
?nominated_gg ?competition ?sum_actors_rank ?sum_actress_rank ?avg_cinematgrs_rank
?avg_composers_rank ?avg_costdesigners_rank ?avg_directors_rank ?sum_distributors_rank
?avg_editors_rank ?avg_prodcompanies_rank ?avg_proddes_rank ?avg_producers_rank
?avg_writers_rank ?opening_gross ?opening_theaters
WHERE
{
   ?s rdf:type <http://localhost:2020/resource/movie/moviesdm>;
      <http://localhost:2020/resource/movie/movieid> ?movieid;
      <http://localhost:2020/resource/movie/dbpedia> ?dbpedia;
      dc:title ?title;
      <http://localhost:2020/resource/movie/mpaa> ?mpaa;
      <http://localhost:2020/resource/movie/metascore> ?metascore;
      <http://localhost:2020/resource/movie/won_oscar> ?won_oscar;
      <http://localhost:2020/resource/movie/nominated_oscar> ?nominated_oscar;
      <http://localhost:2020/resource/movie/won_gg> ?won_gg;
      <http://localhost:2020/resource/movie/nominated_gg> ?nominated_gg;
      <http://localhost:2020/resource/movie/competition> ?competition;
      <http://localhost:2020/resource/movie/imdb_sum_actors_rank> ?sum_actors_rank;
      <http://localhost:2020/resource/movie/imdb_sum_actress_rank> ?sum_actress_rank;
      <http://localhost:2020/resource/movie/imdb_avg_cinematgrs_rank> ?avg_cinematgrs_rank;
      <http://localhost:2020/resource/movie/imdb_avg_composers_rank> ?avg_composers_rank;
      <http://localhost:2020/resource/movie/imdb_avg_costdesigners_rank> ?avg_costdesigners_rank;
      <http://localhost:2020/resource/movie/imdb_avg_directors_rank> ?avg_directors_rank;
      <http://localhost:2020/resource/movie/imdb_sum_distributors_rank> ?sum_distributors_rank;
      <http://localhost:2020/resource/movie/imdb_avg_editors_rank> ?avg_editors_rank;
      <http://localhost:2020/resource/movie/imdb_avg_prodcompanies_rank> ?avg_prodcompanies_rank;
      <http://localhost:2020/resource/movie/imdb_avg_proddes_rank> ?avg_proddes_rank;
      <http://localhost:2020/resource/movie/imdb_avg_producers_rank> ?avg_producers_rank;
      <http://localhost:2020/resource/movie/imdb_avg_writers_rank> ?avg_writers_rank .
   {
      SELECT ?movieid ?dbpedia WHERE
      {
         ?s rdf:type <http://localhost:2020/resource/movie/moviesdt>;
            <http://localhost:2020/resource/movie/movieid> ?movieid;
            <http://localhost:2020/resource/movie/dbpedia> ?dbpedia .
      }
   }
   OPTIONAL
   {
      SELECT ?movieid ?dbpedia ?opening_gross ?opening_theaters WHERE
      {
         ?s rdf:type <http://localhost:2020/resource/movie/moviesbo>;
            <http://localhost:2020/resource/movie/movieid> ?movieid;
            <http://localhost:2020/resource/movie/dbpedia> ?dbpedia;
            <http://localhost:2020/resource/movie/bom_opening_gross_val> ?opening_gross;
            <http://localhost:2020/resource/movie/bom_opening_theaters_val> ?opening_theaters .
      }
   }
}
ORDER BY ?movieid
```

FIGURE A.4: SPARQL Query to retrieve IMDB++ movies data



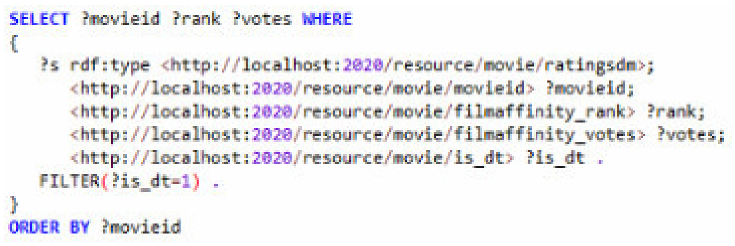
```
SELECT ?movieid ?rank ?votes WHERE
{
   ?s rdf:type <http://localhost:2020/resource/movie/ratingsdm>;
      <http://localhost:2020/resource/movie/movieid> ?movieid;
      <http://localhost:2020/resource/movie/imdb_rank> ?rank;
      <http://localhost:2020/resource/movie/imdb_votes> ?votes;
      <http://localhost:2020/resource/movie/is_dt> ?is_dt .
   FILTER(?is_dt=1) .
}
ORDER BY ?movieid
```

FIGURE A.5: SPARQL Query to retrieve IMDB++ movies rank data



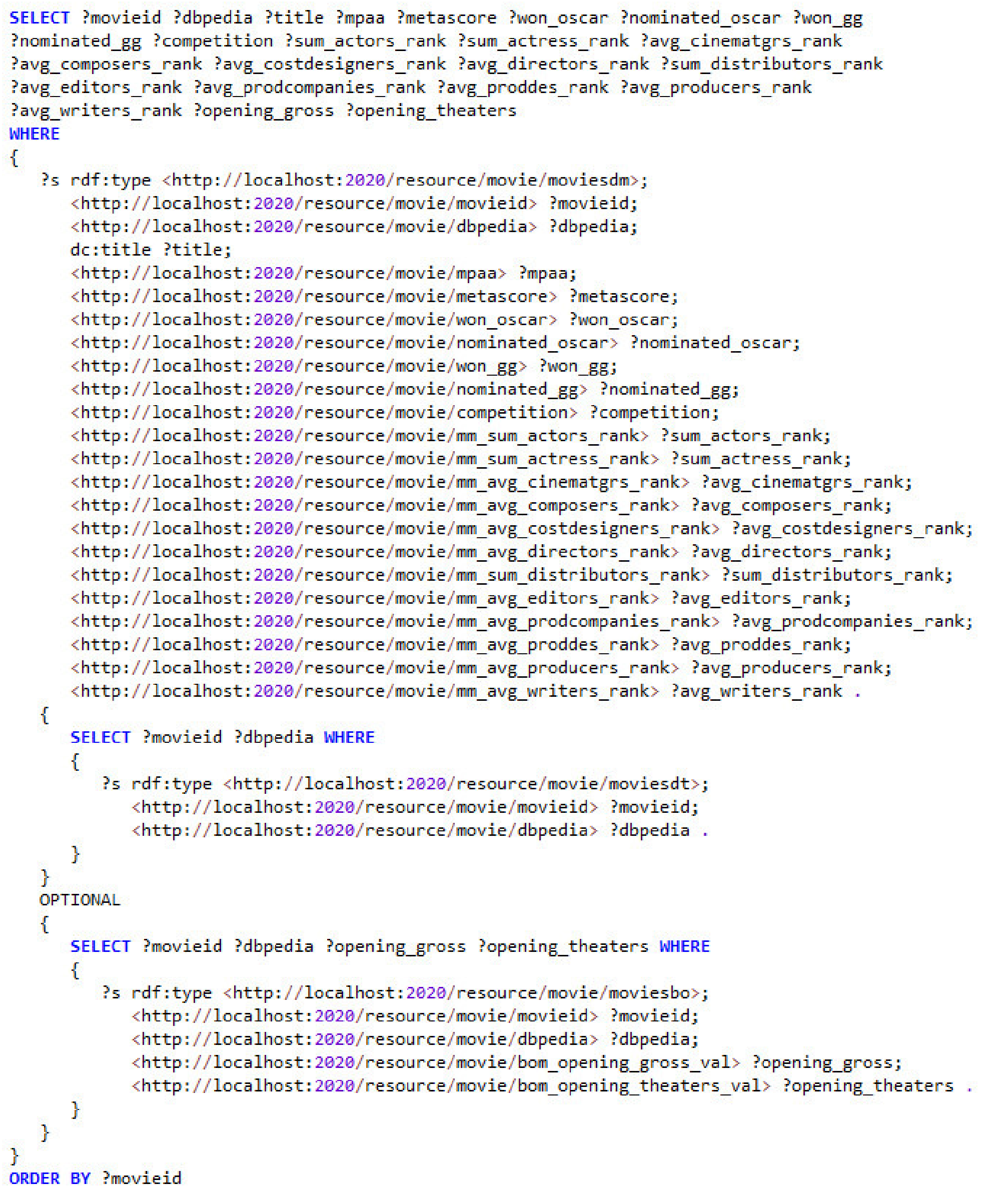
```
SELECT ?movieid ?dbpedia ?title ?mpaa ?metascore ?won_oscar ?nominated_oscar ?won_gg
?nominated_gg ?competition ?sum_actors_rank ?sum_actress_rank ?avg_cinematgrs_rank
?avg_composers_rank ?avg_costdesigners_rank ?avg_directors_rank ?sum_distributors_rank
?avg_editors_rank ?avg_prodcompanies_rank ?avg_proddes_rank ?avg_producers_rank
?avg_writers_rank ?opening_gross ?opening_theaters
WHERE
{
   ?s rdf:type <http://localhost:2020/resource/movie/moviesdm>;
      <http://localhost:2020/resource/movie/movieid> ?movieid;
      <http://localhost:2020/resource/movie/dbpedia> ?dbpedia;
      dc:title ?title;
      <http://localhost:2020/resource/movie/mpaa> ?mpaa;
      <http://localhost:2020/resource/movie/metascore> ?metascore;
      <http://localhost:2020/resource/movie/won_oscar> ?won_oscar;
      <http://localhost:2020/resource/movie/nominated_oscar> ?nominated_oscar;
      <http://localhost:2020/resource/movie/won_gg> ?won_gg;
      <http://localhost:2020/resource/movie/nominated_gg> ?nominated_gg;
      <http://localhost:2020/resource/movie/competition> ?competition;
      <http://localhost:2020/resource/movie/fa_sum_actors_rank> ?sum_actors_rank;
      <http://localhost:2020/resource/movie/fa_sum_actress_rank> ?sum_actress_rank;
      <http://localhost:2020/resource/movie/fa_avg_cinematgrs_rank> ?avg_cinematgrs_rank;
      <http://localhost:2020/resource/movie/fa_avg_composers_rank> ?avg_composers_rank;
      <http://localhost:2020/resource/movie/fa_avg_costdesigners_rank> ?avg_costdesigners_rank;
      <http://localhost:2020/resource/movie/fa_avg_directors_rank> ?avg_directors_rank;
      <http://localhost:2020/resource/movie/fa_sum_distributors_rank> ?sum_distributors_rank;
      <http://localhost:2020/resource/movie/fa_avg_editors_rank> ?avg_editors_rank;
      <http://localhost:2020/resource/movie/fa_avg_prodcompanies_rank> ?avg_prodcompanies_rank;
      <http://localhost:2020/resource/movie/fa_avg_proddes_rank> ?avg_proddes_rank;
      <http://localhost:2020/resource/movie/fa_avg_producers_rank> ?avg_producers_rank;
      <http://localhost:2020/resource/movie/fa_avg_writers_rank> ?avg_writers_rank .
   {
      SELECT ?movieid ?dbpedia WHERE
      {
         ?s rdf:type <http://localhost:2020/resource/movie/moviesdt>;
            <http://localhost:2020/resource/movie/movieid> ?movieid;
            <http://localhost:2020/resource/movie/dbpedia> ?dbpedia .
      }
   }
   OPTIONAL
   {
      SELECT ?movieid ?dbpedia ?opening_gross ?opening_theaters WHERE
      {
         ?s rdf:type <http://localhost:2020/resource/movie/moviesbo>;
            <http://localhost:2020/resource/movie/movieid> ?movieid;
            <http://localhost:2020/resource/movie/dbpedia> ?dbpedia;
            <http://localhost:2020/resource/movie/bom_opening_gross_val> ?opening_gross;
            <http://localhost:2020/resource/movie/bom_opening_theaters_val> ?opening_theaters .
      }
   }
}
ORDER BY ?movieid
```

FIGURE A.6: SPARQL Query to retrieve FilmAffinity movies data



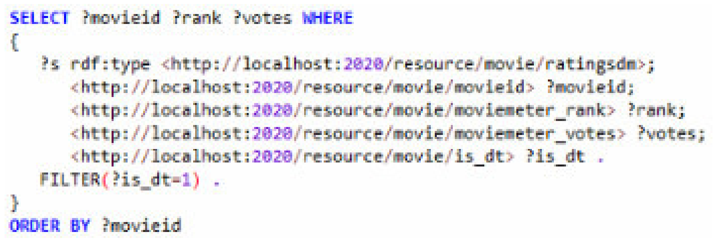
```
SELECT ?movieid ?rank ?votes WHERE
{
   ?s rdf:type <http://localhost:2020/resource/movie/ratingsdm>;
      <http://localhost:2020/resource/movie/movieid> ?movieid;
      <http://localhost:2020/resource/movie/filmaffinity_rank> ?rank;
      <http://localhost:2020/resource/movie/filmaffinity_votes> ?votes;
      <http://localhost:2020/resource/movie/is_dt> ?is_dt .
   FILTER(?is_dt=1) .
}
ORDER BY ?movieid
```

FIGURE A.7: SPARQL Query to retrieve FilmAffinity movies rank data



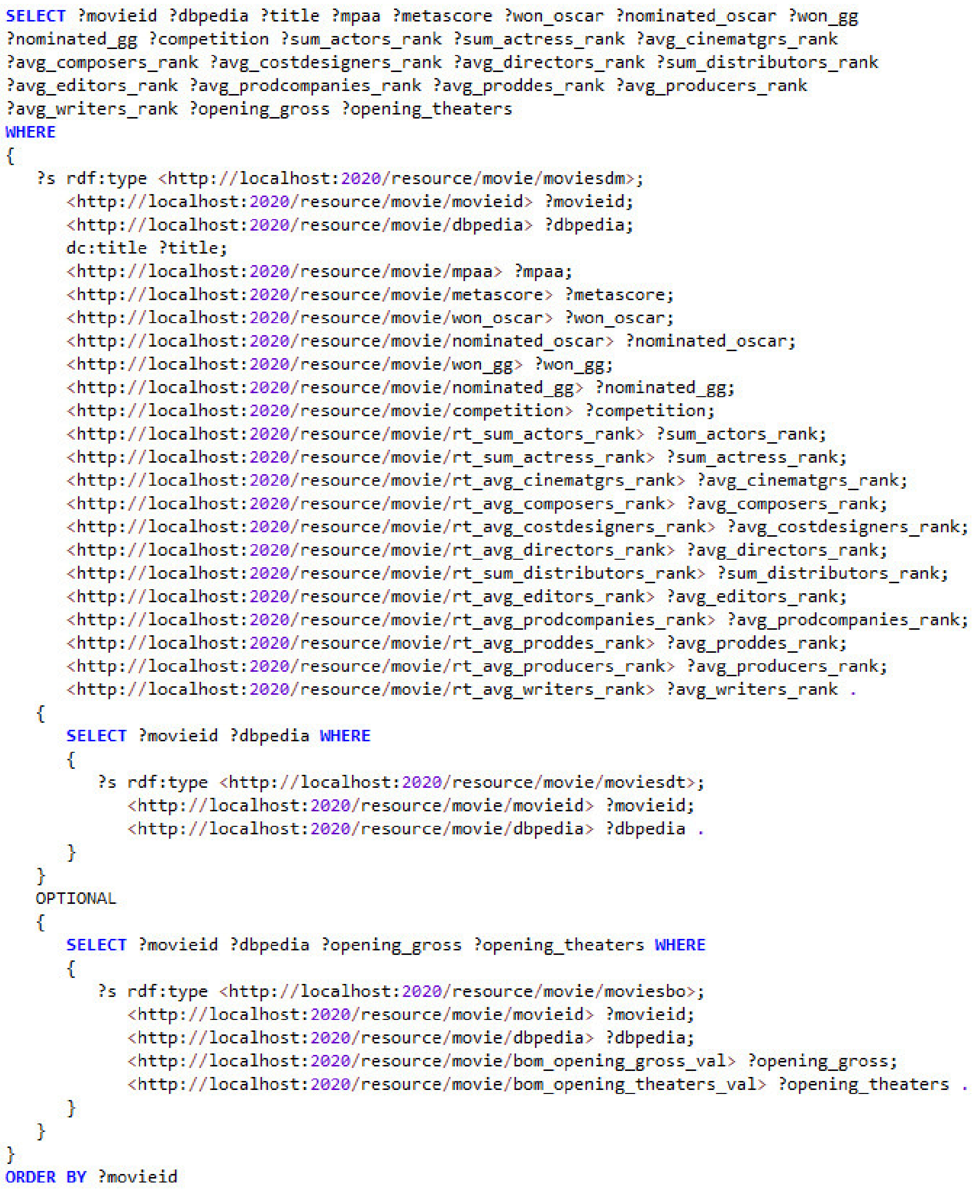
```
SELECT ?movieid ?dbpedia ?title ?mpaa ?metascore ?won_oscar ?nominated_oscar ?won_gg
?nominated_gg ?competition ?sum_actors_rank ?sum_actress_rank ?avg_cinematgrs_rank
?avg_composers_rank ?avg_costdesigners_rank ?avg_directors_rank ?sum_distributors_rank
?avg_editors_rank ?avg_prodcompanies_rank ?avg_proddes_rank ?avg_producers_rank
?avg_writers_rank ?opening_gross ?opening_theaters
WHERE
{
   ?s rdf:type <http://localhost:2020/resource/movie/moviesdm>;
      <http://localhost:2020/resource/movie/movieid> ?movieid;
      <http://localhost:2020/resource/movie/dbpedia> ?dbpedia;
      dc:title ?title;
      <http://localhost:2020/resource/movie/mpaa> ?mpaa;
      <http://localhost:2020/resource/movie/metascore> ?metascore;
      <http://localhost:2020/resource/movie/won_oscar> ?won_oscar;
      <http://localhost:2020/resource/movie/nominated_oscar> ?nominated_oscar;
      <http://localhost:2020/resource/movie/won_gg> ?won_gg;
      <http://localhost:2020/resource/movie/nominated_gg> ?nominated_gg;
      <http://localhost:2020/resource/movie/competition> ?competition;
      <http://localhost:2020/resource/movie/mm_sum_actors_rank> ?sum_actors_rank;
      <http://localhost:2020/resource/movie/mm_sum_actress_rank> ?sum_actress_rank;
      <http://localhost:2020/resource/movie/mm_avg_cinematgrs_rank> ?avg_cinematgrs_rank;
      <http://localhost:2020/resource/movie/mm_avg_composers_rank> ?avg_composers_rank;
      <http://localhost:2020/resource/movie/mm_avg_costdesigners_rank> ?avg_costdesigners_rank;
      <http://localhost:2020/resource/movie/mm_avg_directors_rank> ?avg_directors_rank;
      <http://localhost:2020/resource/movie/mm_sum_distributors_rank> ?sum_distributors_rank;
      <http://localhost:2020/resource/movie/mm_avg_editors_rank> ?avg_editors_rank;
      <http://localhost:2020/resource/movie/mm_avg_prodcompanies_rank> ?avg_prodcompanies_rank;
      <http://localhost:2020/resource/movie/mm_avg_proddes_rank> ?avg_proddes_rank;
      <http://localhost:2020/resource/movie/mm_avg_producers_rank> ?avg_producers_rank;
      <http://localhost:2020/resource/movie/mm_avg_writers_rank> ?avg_writers_rank .
   {
      SELECT ?movieid ?dbpedia WHERE
      {
         ?s rdf:type <http://localhost:2020/resource/movie/moviesdt>;
            <http://localhost:2020/resource/movie/movieid> ?movieid;
            <http://localhost:2020/resource/movie/dbpedia> ?dbpedia .
      }
   }
   OPTIONAL
   {
      SELECT ?movieid ?dbpedia ?opening_gross ?opening_theaters WHERE
      {
         ?s rdf:type <http://localhost:2020/resource/movie/moviesbo>;
            <http://localhost:2020/resource/movie/movieid> ?movieid;
            <http://localhost:2020/resource/movie/dbpedia> ?dbpedia;
            <http://localhost:2020/resource/movie/bom_opening_gross_val> ?opening_gross;
            <http://localhost:2020/resource/movie/bom_opening_theaters_val> ?opening_theaters .
      }
   }
}
ORDER BY ?movieid
```

FIGURE A.8: SPARQL Query to retrieve MovieMeter movies data



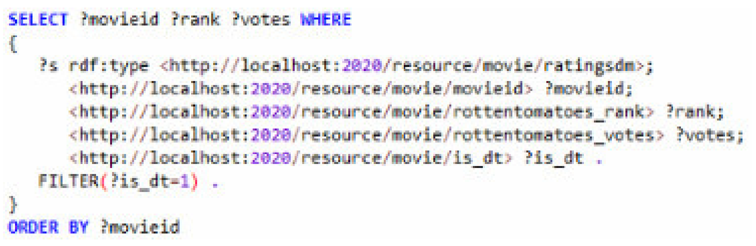
```
SELECT ?movieid ?rank ?votes WHERE
{
   ?s rdf:type <http://localhost:2020/resource/movie/ratingsdm>;
      <http://localhost:2020/resource/movie/movieid> ?movieid;
      <http://localhost:2020/resource/movie/moviemeter_rank> ?rank;
      <http://localhost:2020/resource/movie/moviemeter_votes> ?votes;
      <http://localhost:2020/resource/movie/is_dt> ?is_dt .
   FILTER(?is_dt=1) .
}
ORDER BY ?movieid
```

FIGURE A.9: SPARQL Query to retrieve MovieMeter movies rank data



```
SELECT ?movieid ?dbpedia ?title ?mpaa ?metascore ?won_oscar ?nominated_oscar ?won_gg
?nominated_gg ?competition ?sum_actors_rank ?sum_actress_rank ?avg_cinematgrs_rank
?avg_composers_rank ?avg_costdesigners_rank ?avg_directors_rank ?sum_distributors_rank
?avg_editors_rank ?avg_prodcompanies_rank ?avg_proddes_rank ?avg_producers_rank
?avg_writers_rank ?opening_gross ?opening_theaters
WHERE
{
   ?s rdf:type <http://localhost:2020/resource/movie/moviesdm>;
      <http://localhost:2020/resource/movie/movieid> ?movieid;
      <http://localhost:2020/resource/movie/dbpedia> ?dbpedia;
      dc:title ?title;
      <http://localhost:2020/resource/movie/mpaa> ?mpaa;
      <http://localhost:2020/resource/movie/metascore> ?metascore;
      <http://localhost:2020/resource/movie/won_oscar> ?won_oscar;
      <http://localhost:2020/resource/movie/nominated_oscar> ?nominated_oscar;
      <http://localhost:2020/resource/movie/won_gg> ?won_gg;
      <http://localhost:2020/resource/movie/nominated_gg> ?nominated_gg;
      <http://localhost:2020/resource/movie/competition> ?competition;
      <http://localhost:2020/resource/movie/rt_sum_actors_rank> ?sum_actors_rank;
      <http://localhost:2020/resource/movie/rt_sum_actress_rank> ?sum_actress_rank;
      <http://localhost:2020/resource/movie/rt_avg_cinematgrs_rank> ?avg_cinematgrs_rank;
      <http://localhost:2020/resource/movie/rt_avg_composers_rank> ?avg_composers_rank;
      <http://localhost:2020/resource/movie/rt_avg_costdesigners_rank> ?avg_costdesigners_rank;
      <http://localhost:2020/resource/movie/rt_avg_directors_rank> ?avg_directors_rank;
      <http://localhost:2020/resource/movie/rt_sum_distributors_rank> ?sum_distributors_rank;
      <http://localhost:2020/resource/movie/rt_avg_editors_rank> ?avg_editors_rank;
      <http://localhost:2020/resource/movie/rt_avg_prodcompanies_rank> ?avg_prodcompanies_rank;
      <http://localhost:2020/resource/movie/rt_avg_proddes_rank> ?avg_proddes_rank;
      <http://localhost:2020/resource/movie/rt_avg_producers_rank> ?avg_producers_rank;
      <http://localhost:2020/resource/movie/rt_avg_writers_rank> ?avg_writers_rank .
   {
      SELECT ?movieid ?dbpedia WHERE
      {
         ?s rdf:type <http://localhost:2020/resource/movie/moviesdt>;
            <http://localhost:2020/resource/movie/movieid> ?movieid;
            <http://localhost:2020/resource/movie/dbpedia> ?dbpedia .
      }
   }
   OPTIONAL
   {
      SELECT ?movieid ?dbpedia ?opening_gross ?opening_theaters WHERE
      {
         ?s rdf:type <http://localhost:2020/resource/movie/moviesbo>;
            <http://localhost:2020/resource/movie/movieid> ?movieid;
            <http://localhost:2020/resource/movie/dbpedia> ?dbpedia;
            <http://localhost:2020/resource/movie/bom_opening_gross_val> ?opening_gross;
            <http://localhost:2020/resource/movie/bom_opening_theaters_val> ?opening_theaters .
      }
   }
}
ORDER BY ?movieid
```

Figure A.10: SPARQL Query to retrieve RottenTomatoes movies data



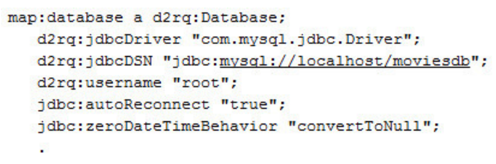
```
SELECT ?movieid ?rank ?votes WHERE
{
   ?s rdf:type <http://localhost:2020/resource/movie/ratingsdm>;
      <http://localhost:2020/resource/movie/movieid> ?movieid;
      <http://localhost:2020/resource/movie/rottentomatoes_rank> ?rank;
      <http://localhost:2020/resource/movie/rottentomatoes_votes> ?votes;
      <http://localhost:2020/resource/movie/is_dt> ?is_dt .
   FILTER(?is_dt=1) .
}
ORDER BY ?movieid
```

Figure A.11: SPARQL Query to retrieve RottenTomatoes movies rank data

Appendix B

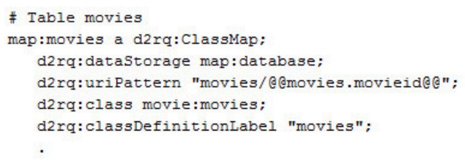
D2RQ Mapping

This appendix contains some statements to mapping a MySQL database to RDF which is used in this research.



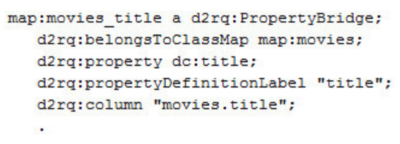
```
map:database a d2rq:Database;
   d2rq:jdbcDriver "com.mysql.jdbc.Driver";
   d2rq:jdbcDSN "jdbc:mysql://localhost/moviesdb";
   d2rq:username "root";
   jdbc:autoReconnect "true";
   jdbc:zeroDateTimeBehavior "convertToNull";
   .
```

Figure B.1: Defining a Database Connection using D2RQ



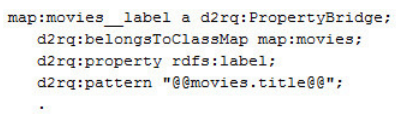
```
# Table movies
map:movies a d2rq:ClassMap;
   d2rq:dataStorage map:database;
   d2rq:uriPattern "movies/@@movies.movieid@@";
   d2rq:class movie:movies;
   d2rq:classDefinitionLabel "movies";
   .
```

Figure B.2: A Mapping of the movies table using a ClassMap Object



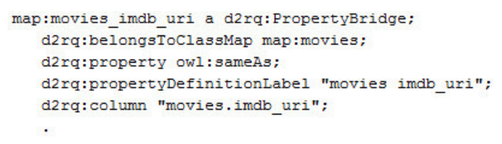
```
map:movies_title a d2rq:PropertyBridge;
   d2rq:belongsToClassMap map:movies;
   d2rq:property dc:title;
   d2rq:propertyDefinitionLabel "title";
   d2rq:column "movies.title";
   .
```

Figure B.3: A Mapping from the movies.title field to the dc:title



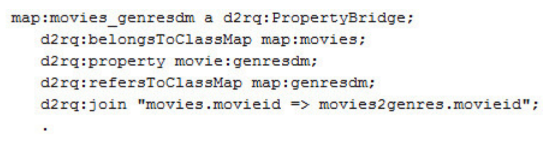
```
map:movies__label a d2rq:PropertyBridge;
   d2rq:belongsToClassMap map:movies;
   d2rq:property rdfs:label;
   d2rq:pattern "@@movies.title@@";
   .
```

Figure B.4: A Mapping from the movies.title field to the rdfs:label



```
map:movies_imdb_uri a d2rq:PropertyBridge;
   d2rq:belongsToClassMap map:movies;
   d2rq:property owl:sameAs;
   d2rq:propertyDefinitionLabel "movies imdb_uri";
   d2rq:column "movies.imdb_uri";
   .
```

Figure B.5: A Mapping from the movies.imdb_uri field to the owl:sameAs



```
map:movies_genresdm a d2rq:PropertyBridge;
   d2rq:belongsToClassMap map:movies;
   d2rq:property movie:genresdm;
   d2rq:refersToClassMap map:genresdm;
   d2rq:join "movies.movieid => movies2genres.movieid";
   .
```

Figure B.6: A Mapping between two table movies and movies2genres with the linking factor movieid

Appendix C

Confusion Matrixs of ANN

This appendix contains all confusion matrixs gathered of experimentation using ANN learner.

Table C.1: The Confusion Matrix of ANN with default parameters applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1025	328	4	0	75.53
	Average	566	10971	731	12	89.34
	Poor	4	379	1605	79	77.65
	Terrible	1	3	42	258	84.87
Class Recall (%)		64.22	93.92	67.38	73.93	

Table C.2: The Confusion Matrix of ANN with optimized parameters applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1011	373	0	0	73.05
	Average	580	10961	717	14	89.32
	Poor	4	346	1625	69	79.50
	Terrible	1	1	40	266	86.36
Class Recall (%)		63.35	93.84	68.22	76.22	

Table C.3: The Confusion Matrix of ANN with default parameters applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1155	241	3	3	82.38
	Average	439	11084	696	11	90.63
	Poor	0	350	1630	65	79.71
	Terrible	2	6	53	270	81.57
Class Recall (%)		72.37	94.89	68.43	77.36	

TABLE C.4: The Confusion Matrix of ANN with optimized parameters applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1150	254	6	0	81.56
	Average	441	11067	641	12	91.00
	Poor	4	358	1700	75	79.55
	Terrible	1	2	35	262	87.33
Class Recall (%)		72.06	94.74	71.37	75.07	

TABLE C.5: The Confusion Matrix of ANN with default parameters applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	643	139	5	0	81.70
	Average	322	10569	989	7	88.91
	Poor	2	464	2416	103	80.94
	Terrible	0	7	74	268	76.79
Class Recall (%)		66.49	94.54	69.35	70.90	

TABLE C.6: The Confusion Matrix of ANN with optimized parameters applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	613	131	6	0	81.73
	Average	351	10589	918	8	89.24
	Poor	2	459	2495	103	81.56
	Terrible	1	0	65	267	80.18
Class Recall (%)		63.39	94.72	71.61	70.63	

TABLE C.7: The Confusion Matrix of ANN with default parameters applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	225	36	5	0	84.59
	Average	181	11107	972	7	90.54
	Poor	2	555	2599	81	80.29
	Terrible	0	4	77	157	65.97
Class Recall (%)		55.15	94.92	71.15	64.08	

TABLE C.8: The Confusion Matrix of ANN with optimized parameters applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	213	35	7	1	83.20
	Average	193	11171	934	7	90.78
	Poor	2	489	2635	89	81.96
	Terrible	0	7	77	148	63.79
Class Recall (%)		52.21	95.46	72.13	60.41	

TABLE C.9: The Confusion Matrix of ANN with default parameters applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1737	361	6	0	82.56
	Average	511	11741	419	8	92.60
	Poor	8	242	860	44	74.52
	Terrible	1	12	25	33	46.48
Class Recall (%)		76.96	95.02	65.65	38.82	

TABLE C.10: The Confusion Matrix of ANN with optimized parameters applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1715	317	7	0	84.11
	Average	535	11847	441	12	92.30
	Poor	7	184	837	43	78.15
	Terrible	0	8	25	30	47.62
Class Recall (%)		75.99	95.88	63.89	35.29	

TABLE C.11: The Confusion Matrix of ANN with default parameters applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1034	361	3	0	73.96
	Average	559	10981	754	11	89.24
	Poor	2	335	1587	79	79.23
	Terrible	1	4	38	259	85.76
Class Recall (%)		64.79	94.01	66.62	74.21	

TABLE C.12: The Confusion Matrix of ANN with optimized parameters applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	898	257	2	0	77.61
	Average	695	11112	776	14	88.21
	Poor	2	311	1565	69	80.38
	Terrible	1	1	39	266	86.64
Class Recall (%)		56.27	95.13	65.70	76.22	

TABLE C.13: The Confusion Matrix of ANN with default parameters applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1192	276	2	0	81.09
	Average	398	11019	646	16	91.22
	Poor	5	382	1703	77	78.59
	Terrible	1	4	31	256	87.67
Class Recall (%)		74.69	94.33	71.49	73.35	

Table C.14: The Confusion Matrix of ANN with optimized parameters applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1107	231	2	0	82.61
	Average	487	11091	699	11	90.26
	Poor	2	354	1650	83	78.99
	Terrible	0	5	31	255	87.63
Class Recall (%)		69.36	94.95	69.27	73.07	

Table C.15: The Confusion Matrix of ANN with default parameters applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	613	140	3	1	80.98
	Average	353	10446	823	9	89.81
	Poor	0	589	2598	122	78.51
	Terrible	1	4	60	246	79.10
Class Recall (%)		63.39	93.44	74.57	65.08	

Table C.16: The Confusion Matrix of ANN with optimized parameters applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	566	144	3	1	79.27
	Average	400	10586	974	10	88.44
	Poor	0	447	2446	97	81.81
	Terrible	1	2	61	270	80.84
Class Recall (%)		58.53	94.70	70.21	71.43	

Table C.17: The Confusion Matrix of ANN with default parameters applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	220	41	2	1	83.33
	Average	184	11062	914	7	90.92
	Poor	4	591	2652	78	79.76
	Terrible	0	8	85	159	63.09
Class Recall (%)		53.92	94.53	72.60	64.90	

Table C.18: The Confusion Matrix of ANN with optimized parameters applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	196	33	5	2	83.05
	Average	212	11182	963	10	90.42
	Poor	0	480	2628	93	82.10
	Terrible	0	7	57	140	68.63
Class Recall (%)		48.04	95.56	71.94	57.14	

Table C.19: The Confusion Matrix of ANN with default parameters applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1676	300	4	0	84.65
	Average	573	11822	438	12	92.04
	Poor	7	226	847	42	75.49
	Terrible	1	8	21	31	50.82
Class Recall (%)		74.26	95.68	64.66	36.47	

Table C.20: The Confusion Matrix of ANN with optimized parameters applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1605	335	2	1	82.60
	Average	648	11838	497	14	91.08
	Poor	3	180	803	59	76.84
	Terrible	1	3	8	11	47.83
Class Recall (%)		71.11	95.81	61.30	12.94	

Table C.21: The Confusion Matrix of Bagging ANN with Default Parameters Applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1014	225	2	0	81.71
	Average	581	11152	701	13	89.60
	Poor	0	302	1658	86	81.04
	Terrible	1	2	21	250	91.24
Class Recall (%)		63.53	95.47	69.61	71.63	

Table C.22: The Confusion Matrix of Bagging ANN with Optimized Parameters Applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	998	238	1	0	80.68
	Average	594	11159	704	10	89.51
	Poor	3	283	1653	80	81.87
	Terrible	1	1	24	259	90.88
Class Recall (%)		62.53	95.53	69.40	74.21	

Table C.23: The Confusion Matrix of Bagging ANN with Default Parameters Applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1026	257	2	0	79.84
	Average	568	11090	668	9	89.91
	Poor	1	334	1690	84	80.13
	Terrible	1	0	22	256	91.76
Class Recall (%)		64.29	94.94	70.95	73.35	

TABLE C.24: The Confusion Matrix of Bagging ANN with Optimized Parameters Applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	975	248	5	0	79.40
	Average	619	11126	727	18	89.08
	Poor	1	307	1627	74	80.99
	Terrible	1	0	23	257	91.46
Class Recall (%)		61.09	95.25	68.30	73.64	

TABLE C.25: The Confusion Matrix of Bagging ANN with Default Parameters Applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1201	169	1	0	87.60
	Average	393	11238	649	13	91.42
	Poor	1	274	1711	74	83.06
	Terrible	1	0	21	262	92.25
Class Recall (%)		75.25	96.21	71.83	75.07	

TABLE C.26: The Confusion Matrix of Bagging ANN with Optimized Parameters Applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1199	198	3	0	85.64
	Average	395	11213	624	15	91.56
	Poor	1	269	1732	67	83.71
	Terrible	1	1	23	267	91.44
Class Recall (%)		75.13	95.99	72.71	76.50	

TABLE C.27: The Confusion Matrix of Bagging ANN with Default Parameters Applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1163	172	1	0	87.05
	Average	431	11236	654	14	91.09
	Poor	2	273	1702	73	83.02
	Terrible	0	0	25	262	91.29
Class Recall (%)		72.87	96.19	71.45	75.07	

TABLE C.28: The Confusion Matrix of Bagging ANN with Optimized Parameters Applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1146	221	3	0	83.65
	Average	448	11155	679	10	90.75
	Poor	1	304	1677	88	81.01
	Terrible	1	1	23	251	90.94
Class Recall (%)		71.80	95.50	70.40	71.92	

Table C.29: The Confusion Matrix of Bagging ANN with Default Parameters Applied to the full-attributes FilmAffinity dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	665	74	4	0	89.50
	Average	301	10722	924	9	89.68
	Poor	1	383	2497	106	83.60
	Terrible	0	0	59	263	81.68
Class Recall (%)		68.77	95.91	71.67	69.58	

Table C.30: The Confusion Matrix of Bagging ANN with Optimized Parameters Applied to the full-attributes FilmAffinity dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	632	86	5	0	87.41
	Average	333	10672	836	8	90.07
	Poor	1	421	2588	102	83.16
	Terrible	1	0	55	268	82.72
Class Recall (%)		65.36	95.46	74.28	70.90	

Table C.31: The Confusion Matrix of Bagging ANN with Default Parameters Applied to the reduced-attributes FilmAffinity dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	625	83	3	0	87.90
	Average	341	10703	909	8	89.48
	Poor	1	393	2511	101	83.53
	Terrible	0	0	61	269	81.52
Class Recall (%)		64.63	95.74	72.07	71.16	

Table C.32: The Confusion Matrix of Bagging ANN with Optimized Parameters Applied to the reduced-attributes FilmAffinity dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	578	117	5	0	82.57
	Average	388	10634	866	10	89.38
	Poor	0	428	2551	95	82.99
	Terrible	1	0	62	273	81.25
Class Recall (%)		59.77	95.12	73.22	72.22	

Table C.33: The Confusion Matrix of Bagging ANN with Default Parameters Applied to the full-attributes MovieMeter dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	220	17	2	0	92.05
	Average	188	11238	885	7	91.23
	Poor	0	440	2701	79	83.88
	Terrible	0	7	65	159	68.83
Class Recall (%)		53.92	96.03	73.94	64.90	

Table C.34: The Confusion Matrix of Bagging ANN with Optimized Parameters Applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	214	12	1	0	94.27
	Average	194	11236	879	5	91.25
	Poor	0	447	2708	85	83.58
	Terrible	0	7	65	155	68.28
Class Recall (%)		52.45	96.02	74.13	63.27	

Table C.35: The Confusion Matrix of Bagging ANN with Default Parameters Applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	213	19	0	0	91.81
	Average	195	11164	856	6	91.35
	Poor	0	512	2728	76	82.27
	Terrible	0	7	69	163	68.20
Class Recall (%)		52.21	95.40	74.68	66.53	

Table C.36: The Confusion Matrix of Bagging ANN with Optimized Parameters Applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	193	18	0	0	91.47
	Average	215	11192	859	12	91.15
	Poor	0	483	2738	96	82.54
	Terrible	0	9	56	137	67.82
Class Recall (%)		47.30	95.64	74.95	55.92	

Table C.37: The Confusion Matrix of Bagging ANN with Default Parameters applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1746	218	4	0	88.72
	Average	509	11970	406	9	92.83
	Poor	2	159	887	43	81.30
	Terrible	0	9	13	33	60.00
Class Recall (%)		77.36	97.88	67.71	38.82	

Table C.38: The Confusion Matrix of Bagging ANN with Optimized Parameters applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1744	207	6	0	89.12
	Average	508	11981	388	8	92.98
	Poor	5	160	905	50	80.80
	Terrible	0	8	11	27	58.70
Class Recall (%)		77.27	96.97	69.08	31.76	

Table C.39: The Confusion Matrix of Bagging ANN with Default Parameters Applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1747	245	3	0	87.57
	Average	506	11942	425	10	92.70
	Poor	4	159	865	47	80.47
	Terrible	0	10	17	28	50.91
Class Recall (%)		77.40	96.65	66.03	32.94	

Table C.40: The Confusion Matrix of Bagging ANN with Optimized Parameters Applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1676	284	0	0	85.51
	Average	580	11911	460	10	91.90
	Poor	1	154	842	58	79.81
	Terrible	0	7	8	17	53.12
Class Recall (%)		74.26	96.40	64.27	20.00	

Table C.41: The Confusion Matrix of Boosting ANN with Default Parameters applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1068	298	1	0	78.13
	Average	525	10997	619	9	90.51
	Poor	2	385	1728	58	79.52
	Terrible	1	1	34	282	88.68
Class Recall (%)		66.92	94.14	72.54	80.80	

Table C.42: The Confusion Matrix of Boosting ANN with Optimized Parameters applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1066	337	1	0	75.93
	Average	525	10994	639	7	90.37
	Poor	4	345	1712	59	80.75
	Terrible	1	5	30	283	88.71
Class Recall (%)		66.79	94.12	71.87	81.09	

Table C.43: The Confusion Matrix of Boosting ANN with Default Parameters Applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1015	330	4	0	75.24
	Average	578	10953	632	9	89.99
	Poor	1	398	1720	62	78.86
	Terrible	2	0	26	278	90.85
Class Recall (%)		63.60	93.77	72.21	79.66	

Table C.44: The Confusion Matrix of Boosting ANN with Optimized Parameters Applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1020	344	2	0	74.67
	Average	574	10963	663	8	89.80
	Poor	2	373	1690	68	79.23
	Terrible	0	1	27	273	90.70
Class Recall (%)		63.91	93.85	70.95	78.22	

Table C.45: The Confusion Matrix of Boosting ANN with Default Parameters Applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1269	203	0	0	86.21
	Average	326	11179	547	6	92.71
	Poor	0	299	1807	59	83.46
	Terrible	1	0	28	284	90.73
Class Recall (%)		79.51	95.70	75.86	81.38	

Table C.46: The Confusion Matrix of Boosting ANN with Optimized Parameters Applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1280	183	3	0	87.31
	Average	315	11154	525	7	92.94
	Poor	0	343	1831	47	82.44
	Terrible	1	1	23	295	92.19
Class Recall (%)		80.20	95.49	76.87	84.53	

Table C.47: The Confusion Matrix of Boosting ANN with Default Parameters Applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1218	180	0	0	87.12
	Average	376	11159	572	9	92.10
	Poor	2	339	1793	59	81.76
	Terrible	0	3	17	281	93.36
Class Recall (%)		76.32	95.53	75.27	80.52	

Table C.48: The Confusion Matrix of Boosting ANN with Optimized Parameters Applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1157	266	0	0	81.31
	Average	436	11036	657	7	90.94
	Poor	2	378	1685	78	78.63
	Terrible	1	1	40	264	86.27
Class Recall (%)		72.49	94.48	70.74	75.64	

Appendix D

Confusion Matrixs of ANN – continuation

This appendix contains all confusion matrixs gathered of experimentation using Artiicial Neural Network (ANN) classifier.

Table D.1: The Confusion Matrix of Boosting ANN with Default Parameters Applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	739	101	3	0	87.66
	Average	226	10572	793	7	91.15
	Poor	2	506	2644	81	81.78
	Terrible	0	0	44	290	86.83
Class Recall (%)		76.42	94.57	75.89	76.72	

Table D.2: The Confusion Matrix of Boosting-ANN with Optimized Parameters Applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	746	105	0	0	87.66
	Average	218	10570	761	3	91.50
	Poor	2	503	2676	89	81.83
	Terrible	1	1	47	286	85.37
Class Recall (%)		77.15	94.55	76.81	75.66	

TABLE D.3: The Confusion Matrix of Boosting ANN with Default Parameters Applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	706	117	3	0	85.47
	Average	259	10552	745	4	91.28
	Poor	1	508	2681	77	82.06
	Terrible	1	2	55	297	83.66
Class Recall (%)		73.01	94.39	76.95	78.57	

TABLE D.4: The Confusion Matrix of Boosting-ANN with Optimized Parameters Applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	649	140	2	0	82.05
	Average	317	10464	810	5	90.24
	Poor	1	575	2604	101	79.37
	Terrible	0	0	68	272	80.00
Class Recall (%)		67.11	93.60	74.74	71.96	

TABLE D.5: The Confusion Matrix of Boosting ANN with Default Parameters Applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	292	23	3	0	91.82
	Average	116	11121	845	7	91.99
	Poor	0	555	2760	87	81.13
	Terrible	0	3	45	151	75.88
Class Recall (%)		71.57	95.03	75.55	61.63	

TABLE D.6: The Confusion Matrix of Boosting-ANN with Optimized Parameters Applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	287	30	0	0	90.54
	Average	119	11056	772	6	92.50
	Poor	2	613	2835	89	80.11
	Terrible	0	3	46	150	75.38
Class Recall (%)		70.34	94.48	77.61	61.22	

TABLE D.7: The Confusion Matrix of Boosting ANN with Default Parameters Applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	285	18	0	0	94.06
	Average	123	11142	873	5	91.76
	Poor	0	539	2743	85	81.47
	Terrible	0	3	37	155	79.49
Class Recall (%)		69.85	95.21	75.09	63.27	

TABLE D.8: The Confusion Matrix of Boosting ANN with Optimized Parameters Applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	272	26	4	0	90.07
	Average	135	11136	839	4	91.93
	Poor	1	537	2765	93	81.42
	Terrible	0	3	45	148	75.51
Class Recall (%)		66.67	95.16	75.69	60.41	

TABLE D.9: The Confusion Matrix of Boosting ANN with Default Parameters Applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1870	274	4	0	87.06
	Average	385	11877	396	0	93.83
	Poor	2	198	896	27	79.79
	Terrible	0	7	14	58	73.42
Class Recall (%)		82.85	96.12	68.40	68.24	

TABLE D.10: The Confusion Matrix of Boosting-ANN with Optimized Parameters Applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1838	284	5	0	86.41
	Average	418	11844	369	4	93.74
	Poor	1	219	918	15	79.62
	Terrible	0	9	18	66	70.97
Class Recall (%)		81.44	95.86	70.08	77.65	

TABLE D.11: The Confusion Matrix of Boosting ANN with Default Parameters Applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1829	258	1	0	87.60
	Average	428	11842	370	2	93.67
	Poor	0	251	921	25	76.94
	Terrible	0	5	18	58	71.60
Class Recall (%)		81.04	95.84	70.31	68.24	

TABLE D.12: The Confusion Matrix of Boosting ANN with Optimized Parameters Applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1669	345	1	0	82.83
	Average	585	11784	447	4	91.92
	Poor	2	222	843	27	77.06
	Terrible	1	5	19	54	68.35
Class Recall (%)		73.95	95.37	64.35	63.53	

Appendix E

Confusion Matrixs of Decision Tree

This appendix contains all confusion matrixs gathered of experimentation using Decision Tree (DT) classifier.

Table E.1: The Confusion Matrix of DT with default parameters applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1022	695	21	0	58.80
	Average	573	10779	905	16	87.83
	Poor	0	206	1448	74	83.80
	Terrible	1	1	8	259	96.28
Class Recall (%)		64.03	92.28	60.79	74.21	

Table E.2: The Confusion Matrix of DT with optimized parameters applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1536	56	2	0	96.36
	Average	57	11558	78	1	98.83
	Poor	2	66	2287	16	96.48
	Terrible	1	1	15	332	95.13
Class Recall (%)		96.24	98.95 5	96.01	95.13	

Table E.3: The Confusion Matrix of DT applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	896	467	16	0	64.97
	Average	699	11007	910	16	87.14
	Poor	0	206	1448	74	83.80
	Terrible	1	1	8	259	96.28
Class Recall (%)		56.14	94.23	60.79	74.21	

Table E.4: The Confusion Matrix of DT with optimized parameter applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1542	56	2	0	96.38
	Average	50	11556	71	1	98.96
	Poor	3	65	2298	17	96.43
	Terrible	1	4	11	331	95.39
Class Recall (%)		96.62	98.93	96.47	94.84	

Table E.5: The Confusion Matrix of DT applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	139	8	0	0	94.56
	Average	827	10868	1545	36	81.86
	Poor	0	303	1931	152	80.93
	Terrible	1	0	8	190	95.48
Class Recall (%)		14.37	97.22	55.42	50.26	

Table E.6: The Confusion Matrix of DT with optimized parameters applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	919	47	2	0	94.94
	Average	44	11029	84	0	98.85
	Poor	2	103	3389	24	96.33
	Terrible	2	0	9	354	96.99
Class Recall (%)		95.04	98.66	97.27	93.65	

Table E.7: The Confusion Matrix of DT applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	0	0	0	0	0
	Average	408	11702	3653	245	73.10
	Poor	0	0	0	0	0
	Terrible	0	0	0	0	0
Class Recall (%)		0	100	0	0	

Table E.8: The Confusion Matrix of DT with optimized parameters applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	217	4	15	0	91.95
	Average	2	11553	107	38	98.74
	Poor	26	120	3529	0	96.03
	Terrible	0	25	2	370	93.20
Class Recall (%)		88.57	98.73	96.61	90.69	

Table E.9: The Confusion Matrix of DT applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1404	300	0	0	82.39
	Average	852	11886	591	16	89.07
	Poor	0	163	705	52	76.63
	Terrible	1	7	14	17	43.59
Class Recall (%)		62.21	96.20	53.82	20.00	

Table E.10: The Confusion Matrix of DT with optimized parameters applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	2165	90	1	0	95.97
	Average	90	12218	70	1	98.70
	Poor	2	47	1234	6	95.73
	Terrible	0	1	5	78	92.86
Class Recall (%)		95.92	98.88	94.20	91.76	

Table E.11: The Confusion Matrix of DT with default parameters applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1019	692	21	0	58.83
	Average	576	10782	905	16	87.81
	Poor	0	206	1448	72	83.89
	Terrible	1	1	8	261	96.31
Class Recall (%)		63.85	92.30	60.79	74.79	

Table E.12: The Confusion Matrix of DT with optimized parameters applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1544	61	1	0	96.14
	Average	52	11572	66	1	98.98
	Poor	0	48	2309	10	97.55
	Terrible	0	0	6	338	98.26
Class Recall (%)		96.74	99.07	96.94	96.85	

Table E.13: The Confusion Matrix of DT with default parameters applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	897	467	16	0	65.00
	Average	698	11007	910	16	87.14
	Poor	0	206	1448	74	83.80
	Terrible	1	1	8	259	96.28
Class Recall (%)		56.20	94.23	60.79	74.21	

TABLE E.14: The Confusion Matrix of DT with optimized parameters applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1536	48	0	0	96.97
	Average	60	11584	61	1	98.96
	Poor	0	48	2314	9	97.60
	Terrible	0	1	7	339	97.69
Class Recall (%)		96.24	99.17	97.15	97.13	

TABLE E.15: The Confusion Matrix of DT with default parameters applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	127	6	0	0	95.49
	Average	839	10896	1544	35	81.84
	Poor	0	277	1932	142	82.18
	Terrible	1	0	8	201	95.71
Class Recall (%)		13.13	97.47	55.45	53.17	

TABLE E.16: The Confusion Matrix of DT with optimized parameters applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	927	42	0	0	95.67
	Average	38	11062	79	1	98.94
	Poor	2	73	3391	19	97.30
	Terrible	0	2	14	358	95.72
Class Recall (%)		95.86	98.95	97.33	94.71	

TABLE E.17: The Confusion Matrix of DT with default parameters applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	0	0	0	0	00.00
	Average	245	11702	3653	408	73.10
	Poor	0	0	0	0	00.00
	Terrible	0	0	0	0	00.00
Class Recall (%)		00.00	100.00	00.00	00.00	

TABLE E.18: The Confusion Matrix of DT with optimized parameters applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	382	22	3	0	93.86
	Average	26	11588	96	5	98.92
	Poor	0	90	3539	23	96.91
	Terrible	0	2	15	217	92.74
Class Recall (%)		93.63	99.03	96.88	88.57	

TABLE E.19: The Confusion Matrix of DT with default parameters applied to the reduced-attributes RottenTomatoes dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1404	300	0	0	82.39
	Average	852	11886	591	16	89.07
	Poor	0	163	705	52	76.63
	Terrible	1	7	14	17	43.59
Class Recall (%)		62.21	96.20	53.82	20.00	

TABLE E.20: The Confusion Matrix of DT with optimized parameters applied to the reduced-attributes RottenTomatoes dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	2186	67	2	0	96.94
	Average	69	12251	52	1	99.01
	Poor	2	35	1242	4	96.80
	Terrible	0	3	14	80	82.47
Class Recall (%)		96.85	99.15	94.81	94.12	

TABLE E.21: The Confusion Matrix of Bagging DT with Default Parameters Applied to the full-attributes IMDb dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	516	291	12	0	63.00
	Average	1079	11185	915	16	84.77
	Poor	0	204	1444	63	84.40
	Terrible	1	1	11	270	95.41
Class Recall (%)		32.33	95.75	60.62	77.36	

TABLE E.22: The Confusion Matrix of Bagging DT with Optimized Parameters Applied to the full-attributes IMDb dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1540	40	3	0	97.28
	Average	53	11590	77	0	98.89
	Poor	2	50	2291	18	97.04
	Terrible	1	1	11	331	96.22
Class Recall (%)		96.49	99.22	96.18	94.84	

TABLE E.23: The Confusion Matrix of Bagging DT with Default Parameters Applied to the reduced-attributes IMDb dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	515	293	12	0	62.80
	Average	1080	11183	915	16	84.76
	Poor	0	204	1445	63	84.40
	Terrible	1	1	10	270	95.74
Class Recall (%)		32.27	95.74	60.66	77.36	

Table E.24: The Confusion Matrix of Bagging DT with Optimized Parameters Applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1542	39	0	0	97.53
	Average	53	11587	66	0	98.98
	Poor	1	55	2308	19	96.85
	Terrible	0	0	8	330	97.63
Class Recall (%)		96.62	99.20	96.89	94.56	

Table E.25: The Confusion Matrix of Bagging DT with Default Parameters Applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	511	117	5	0	80.73
	Average	1084	11359	922	16	84.89
	Poor	0	204	1444	62	84.44
	Terrible	1	1	11	271	95.42
Class Recall (%)		32.02	97.24	60.62	77.65	

Table E.26: The Confusion Matrix of Bagging DT with Optimized Parameters Applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1546	47	2	0	96.93
	Average	47	11584	71	0	98.99
	Poor	2	49	2298	18	97.08
	Terrible	1	1	11	331	96.22
Class Recall (%)		96.87	99.17	96.47	94.84	

Table E.27: The Confusion Matrix of Bagging DT with Default Parameters applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	518	161	8	0	75.40
	Average	1077	11315	919	16	84.90
	Poor	0	204	1444	62	84.44
	Terrible	1	1	11	271	95.42
Class Recall (%)		32.46	96.87	60.62	77.65	

Table E.28: The Confusion Matrix of Bagging DT with Optimized Parameters applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1539	41	0	0	97.41
	Average	57	11593	71	1	98.90
	Poor	0	46	2304	18	84.44
	Terrible	0	1	7	330	97.63
Class Recall (%)		96.43	99.25	96.73	94.56	

TABLE E.29: The Confusion Matrix of Bagging DT with Default Parameters Applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	144	9	0	0	94.12
	Average	822	10868	1537	36	81.94
	Poor	0	302	1942	144	81.32
	Terrible	1	0	5	198	97.06
Class Recall (%)		14.89	97.22	55.74	52.38	

TABLE E.30: The Confusion Matrix of Bagging DT with Optimized Parameters Applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	913	28	2	0	96.82
	Average	51	11087	69	0	98.93
	Poor	2	64	3407	31	97.23
	Terrible	1	0	6	347	98.02
Class Recall (%)		94.42	99.18	97.79	91.80	

TABLE E.31: The Confusion Matrix of Bagging DT with Default Parameters Applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	130	7	0	0	94.89
	Average	836	10891	1536	34	81.91
	Poor	0	281	1942	146	81.98
	Terrible	1	0	6	198	96.59
Class Recall (%)		13.44	97.42	55.74	52.38	

TABLE E.32: The Confusion Matrix of Bagging DT with Optimized Parameters Applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	914	30	2	0	96.62
	Average	51	11071	76	1	98.86
	Poor	1	78	3396	27	96.97
	Terrible	1	0	10	350	96.95
Class Recall (%)		94.52	99.03	97.47	92.59	

TABLE E.33: The Confusion Matrix of Bagging DT with Default Parameters Applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	0	0	0	0	00.00
	Average	408	11702	3653	245	73.10
	Poor	0	0	0	0	00.00
	Terrible	0	0	0	0	00.00
Class Recall (%)		00.00	100.00	00.00	00.00	

TABLE E.34: The Confusion Matrix of Bagging DT with Optimized Parameters Applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	369	10	2	0	96.85
	Average	39	11598	87	3	98.90
	Poor	0	93	3554	25	96.79
	Terrible	0	1	10	217	95.18
Class Recall (%)		90.44	99.11	97.29	88.57	

TABLE E.35: The Confusion Matrix of Bagging DT with Default Parameters Applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	0	0	0	0	00.00
	Average	408	11702	3653	245	73.10
	Poor	0	0	0	0	00.00
	Terrible	0	0	0	0	00.00
Class Recall (%)		00.00	100.00	00.00	00.00	

TABLE E.36: The Confusion Matrix of Bagging DT with Optimized Parameters Applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	371	13	0	0	96.61
	Average	37	11587	99	3	98.81
	Poor	0	101	3543	27	96.51
	Terrible	0	1	11	215	94.71
Class Recall (%)		90.93	99.02	96.99	87.76	

TABLE E.37: The Confusion Matrix of Bagging DT with Default Parameters Applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1408	304	0	0	82.24
	Average	849	11885	589	15	89.11
	Poor	0	162	708	37	78.06
	Terrible	0	5	13	33	64.71
Class Recall (%)		62.38	96.19	54.05	38.82	

TABLE E.38: The Confusion Matrix of Bagging DT with Optimized Parameters Applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	2176	54	1	0	97.53
	Average	81	12275	57	1	98.88
	Poor	0	26	1247	6	97.50
	Terrible	0	1	5	78	92.86
Class Recall (%)		96.41	99.34	95.19	91.76	

TABLE E.39: The Confusion Matrix of Bagging DT with Default Parameters Applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1409	304	0	0	82.25
	Average	848	11885	589	15	89.11
	Poor	0	162	708	37	78.06
	Terrible	0	5	13	33	64.71
Class Recall (%)		62.43	96.19	54.05	38.82	

TABLE E.40: The Confusion Matrix of Bagging DT with Optimized Parameters Applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	2186	55	1	0	97.50
	Average	70	12269	52	2	99.00
	Poor	1	31	1250	7	96.97
	Terrible	0	1	7	76	90.48
Class Recall (%)		96.85	99.30	95.42	89.41	

TABLE E.41: The Confusion Matrix of Boosting DT with Default Parameters Applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1022	695	21	0	58.80
	Average	573	10779	905	16	87.83
	Poor	0	206	1448	74	83.80
	Terrible	1	1	8	259	96.28
Class Recall (%)		64.04	92.28	60.79	74.21	

TABLE E.42: The Confusion Matrix of Boosting DT with Optimized Parameters Applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1534	56	2	0	96.36
	Average	59	11550	76	1	98.84
	Poor	2	74	2289	18	96.06
	Terrible	1	1	15	330	95.10
Class Recall (%)		96.12	98.88	96.10	94.56	

TABLE E.43: The Confusion Matrix of Boosting DT with Default Parameters Applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1019	692	21	0	58.83
	Average	576	10782	905	16	87.81
	Poor	0	206	1448	72	83.89
	Terrible	1	1	8	261	96.31
Class Recall (%)		63.85	92.30	60.79	74.79	

TABLE E.44: The Confusion Matrix of Boosting DT with Optimized Parameters Applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1544	58	4	0	96.14
	Average	51	11558	64	0	99.01
	Poor	1	64	2304	18	96.52
	Terrible	0	1	10	331	96.78
Class Recall (%)		96.74	98.95	96.73	94.84	

TABLE E.45: The Confusion Matrix of Boosting DT with Default Parameters Applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	896	467	16	0	64.97
	Average	699	11007	910	16	87.14
	Poor	0	206	1448	74	83.80
	Terrible	1	1	8	259	96.28
Class Recall (%)		56.14	94.23	60.79	74.21	

TABLE E.46: The Confusion Matrix of Boosting DT with Optimized Parameters Applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1531	69	2	0	95.57
	Average	61	11554	83	2	98.75
	Poor	3	56	2279	17	96.77
	Terrible	1	2	18	330	94.02
Class Recall (%)		95.93	98.91	95.68	94.56	

TABLE E.47: The Confusion Matrix of Boosting DT with Default Parameters applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	897	467	16	0	65.00
	Average	698	11007	910	16	87.14
	Poor	0	206	1448	74	83.80
	Terrible	1	1	8	259	96.28
Class Recall (%)		56.20	94.23	60.79	74.21	

TABLE E.48: The Confusion Matrix of Boosting DT with Optimized Parameters applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1543	64	1	0	95.96
	Average	52	11553	66	0	98.99
	Poor	0	59	2306	21	96.65
	Terrible	1	5	9	328	95.63
Class Recall (%)		96.68	98.90	96.81	93.98	

Appendix F

Confusion Matrixs of Decision Tree – continuation

This appendix contains all confusion matrixs gathered of experimentation using Decision Tree (DT) classifier.

Table F.1: The Confusion Matrix of Boosting DT with Default Parameters Applied to the full-attributes FilmAffinity dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	139	8	0	0	94.56
	Average	827	10868	1545	36	81.86
	Poor	0	303	1931	152	80.93
	Terrible	1	0	8	190	95.48
Class Recall (%)		14.37	97.22	55.42	50.26	

Table F.2: The Confusion Matrix of Boosting DT with Optimized Parameters Applied to the full-attributes FilmAffinity dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	919	47	2	0	94.94
	Average	44	11029	84	0	98.85
	Poor	2	103	3389	24	96.33
	Terrible	2	0	9	354	96.99
Class Recall (%)		95.04	98.66	97.27	93.65	

TABLE F.3: The Confusion Matrix of Boosting DT with Default Parameters Applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	127	6	0	0	95.49
	Average	839	10896	1544	35	81.84
	Poor	0	277	1932	142	82.18
	Terrible	1	0	8	201	95.71
Class Recall (%)		13.13	97.47	55.45	53.17	

TABLE F.4: The Confusion Matrix of Boosting DT with Optimized Parameters Applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	916	46	2	0	95.02
	Average	47	11039	79	2	98.85
	Poor	2	94	3390	25	96.55
	Terrible	2	0	13	351	95.90
Class Recall (%)		94.73	98.75	97.30	92.86	

TABLE F.5: The Confusion Matrix of Boosting DT with Default Parameters Applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	0	0	0	0	00.00
	Average	408	11702	3653	245	73.10
	Poor	0	0	0	0	00.00
	Terrible	0	0	0	0	00.00
Class Recall (%)		00.00	100.00	00.00	00.00	

TABLE F.6: The Confusion Matrix of Boosting DT with Optimized Parameters Applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	370	25	2	0	93.20
	Average	38	11553	107	2	98.74
	Poor	0	120	3529	26	96.03
	Terrible	0	4	15	217	91.95
Class Recall (%)		90.69	98.73	96.61	88.57	

TABLE F.7: The Confusion Matrix of Boosting DT with Default Parameters Applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	0	0	0	0	00.00
	Average	408	11702	3653	245	73.10
	Poor	0	0	0	0	00.00
	Terrible	0	0	0	0	00.00
Class Recall (%)		00.00	100.00	00.00	00.00	

Table F.8: The Confusion Matrix of Boosting DT with Optimized Parameters Applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	374	22	2	0	93.97
	Average	34	11555	113	7	98.68
	Poor	0	122	3523	23	96.05
	Terrible	0	3	15	215	92.27
Class Recall (%)		91.67	98.74	96.44	87.76	

Table F.9: The Confusion Matrix of Boosting DT with Default Parameters Applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1404	300	0	0	82.39
	Average	852	11886	591	16	89.07
	Poor	0	163	705	52	76.63
	Terrible	1	7	14	17	43.59
Class Recall (%)		62.21	96.20	53.82	20.00	

Table F.10: The Confusion Matrix of Boosting DT with Optimized Parameters Applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	2165	101	1	0	95.50
	Average	89	12201	64	1	98.75
	Poor	3	53	1240	6	95.24
	Terrible	0	1	5	78	92.86
Class Recall (%)		95.92	98.75	94.66	91.76	

Table F.11: The Confusion Matrix of Boosting DT with Default Parameters Applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1404	300	0	0	82.39
	Average	852	11886	591	16	89.07
	Poor	0	163	705	52	76.63
	Terrible	1	7	14	17	43.59
Class Recall (%)		62.21	96.20	53.82	20.00	

Table F.12: The Confusion Matrix of Boosting DT with Optimized Parameters Applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	2191	79	1	0	96.48
	Average	66	12227	44	1	99.10
	Poor	0	47	1259	7	95.89
	Terrible	0	3	6	77	89.53
Class Recall (%)		97.08	98.96	96.11	90.59	

Appendix G

Confusion Matrixs of k-NN

This appendix contains all confusion matrixs gathered of experimentation using k-Nearest Neighbors (k-NN) classifier.

TABLE G.1: The Confusion Matrix of k-NN applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1165	320	2	0	78.35
	Average	430	11034	563	8	91.68
	Poor	1	321	1785	47	82.87
	Terrible	0	6	32	294	88.55
Class Recall (%)		72.99	94.46	74.94	84.24	

TABLE G.2: The Confusion Matrix of k-NN applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1534	68	0	0	95.76
	Average	60	11537	102	4	98.58
	Poor	0	73	2264	18	96.13
	Terrible	2	3	16	327	93.97
Class Recall (%)		96.12	98.77	95.05	93.70	

TABLE G.3: The Confusion Matrix of k-NN applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1292	204	0	0	86.36
	Average	303	11189	529	6	93.03
	Poor	1	284	1824	50	84.48
	Terrible	0	4	29	293	89.88
Class Recall (%)		80.95	95.79	76.57	83.95	

TABLE G.4: The Confusion Matrix of k-NN applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1534	50	0	0	96.84
	Average	61	11580	97	1	98.65
	Poor	0	50	2275	20	97.01
	Terrible	1	1	10	328	96.47
Class Recall (%)		96.12	99.13	95.51	93.98	

TABLE G.5: The Confusion Matrix of k-NN applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	759	149	0	1	83.50
	Average	207	10585	726	5	91.86
	Poor	1	442	2715	73	84.03
	Terrible	0	3	43	299	86.67
Class Recall (%)		78.49	94.69	77.93	79.10	

TABLE G.6: The Confusion Matrix of k-NN applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	909	40	0	0	95.78
	Average	57	11071	113	1	98.48
	Poor	0	67	3361	23	97.39
	Terrible	1	1	10	354	96.72
Class Recall (%)		94.00	99.03	96.47	93.65	

TABLE G.7: The Confusion Matrix of k-NN applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	299	44	0	0	87.17
	Average	108	11171	754	3	92.81
	Poor	0	487	2860	62	83.89
	Terrible	1	0	39	180	81.82
Class Recall (%)		73.28	95.46	78.29	73.47	

TABLE G.8: The Confusion Matrix of k-NN applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	368	13	0	0	96.59
	Average	40	11603	162	5	98.25
	Poor	0	83	3477	26	96.96
	Terrible	0	3	14	214	92.64
Class Recall (%)		90.20	99.15	95.18	87.35	

TABLE G.9: The Confusion Matrix of k-NN applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1822	270	2	0	87.01
	Average	431	11861	365	3	93.69
	Poor	4	219	930	11	79.90
	Terrible	0	6	13	71	78.89
Class Recall (%)		80.73	95.99	70.99	83.53	

TABLE G.10: The Confusion Matrix of k-NN applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	2165	70	0	0	96.87
	Average	91	12248	62	2	98.75
	Poor	1	32	1230	8	96.77
	Terrible	0	6	18	75	75.76
Class Recall (%)		95.92	99.13	93.89	88.24	

TABLE G.11: The Confusion Matrix of Bagging kNN with Default Parameters Applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1143	312	2	0	78.45
	Average	452	11063	577	11	91.41
	Poor	1	301	1771	47	83.54
	Terrible	0	5	32	291	88.72
Class Recall (%)		71.62	94.71	74.35	83.38	

TABLE G.12: The Confusion Matrix of Bagging kNN with Default Parameters Applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1534	68	0	0	95.76
	Average	60	11536	102	4	98.58
	Poor	0	74	2264	18	96.10
	Terrible	2	3	16	327	93.97
Class Recall (%)		96.12	98.76	95.05	93.70	

TABLE G.13: The Confusion Matrix of Bagging kNN with Default Parameters Applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1292	204	0	0	86.36
	Average	303	11189	529	6	93.03
	Poor	1	284	1824	50	84.48
	Terrible	0	4	29	293	89.88
Class Recall (%)		80.95	95.79	76.57	83.95	

Table G.14: The Confusion Matrix of Bagging kNN with Default Parameters Applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1538	56	0	0	96.49
	Average	57	11573	99	0	98.67
	Poor	0	51	2273	21	96.93
	Terrible	1	1	10	328	96.47
Class Recall (%)		96.37	99.08	95.42	93.98	

Table G.15: The Confusion Matrix of Bagging kNN with Default Parameters Applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	756	149	0	1	83.44
	Average	210	10585	726	5	91.84
	Poor	1	442	2715	73	84.03
	Terrible	0	3	43	299	86.67
Class Recall (%)		78.18	94.69	77.93	79.10	

Table G.16: The Confusion Matrix of Bagging kNN with Default Parameters Applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	908	40	0	0	95.78
	Average	58	11070	113	1	98.47
	Poor	0	68	3361	23	97.36
	Terrible	1	1	10	354	96.72
Class Recall (%)		93.90	99.02	96.47	93.65	

Table G.17: The Confusion Matrix of Bagging kNN with Default Parameters Applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	299	44	0	0	87.17
	Average	108	11171	754	3	92.81
	Poor	0	487	2860	62	83.90
	Terrible	1	0	39	180	81.82
Class Recall (%)		73.28	95.46	78.29	73.47	

Table G.18: The Confusion Matrix of Bagging kNN with Default Parameters Applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	368	13	0	0	96.59
	Average	40	11603	162	5	98.25
	Poor	0	83	3477	26	96.96
	Terrible	0	3	14	214	92.64
Class Recall (%)		90.20	99.15	95.18	87.35	

Table G.19: The Confusion Matrix of Bagging kNN with Default Parameters Applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1822	270	2	0	87.01
	Average	431	11862	366	3	93.68
	Poor	4	218	929	11	79.95
	Terrible	0	6	13	71	78.89
Class Recall (%)		80.73	96.00	70.92	83.53	

Table G.20: The Confusion Matrix of Bagging kNN with Default Parameters Applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	2165	70	0	0	96.87
	Average	91	12248	62	2	98.75
	Poor	1	32	1230	8	96.77
	Terrible	0	6	18	75	75.76
Class Recall (%)		95.92	99.13	93.89	88.24	

Table G.21: The Confusion Matrix of Boosting kNN with Default Parameters Applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1162	309	2	0	78.89
	Average	433	11048	566	11	91.62
	Poor	1	319	1782	50	82.81
	Terrible	0	5	32	288	88.62
Class Recall (%)		72.81	94.58	74.81	82.52	

Table G.22: The Confusion Matrix of Boosting kNN with Default Parameters Applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1534	68	0	0	95.76
	Average	60	11537	102	4	98.58
	Poor	0	73	2264	18	96.14
	Terrible	2	3	16	327	93.97
Class Recall (%)		96.12	98.77	95.05	93.70	

Table G.23: The Confusion Matrix of Boosting kNN with Default Parameters Applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1292	204	0	0	86.36
	Average	303	11189	529	6	93.03
	Poor	1	284	1824	50	84.48
	Terrible	0	4	29	293	89.88
Class Recall (%)		80.95	95.79	76.57	83.95	

TABLE G.24: The Confusion Matrix of Boosting kNN with Default Parameters Applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1538	56	0	0	96.49
	Average	57	11574	99	0	98.67
	Poor	0	50	2273	21	96.97
	Terrible	1	1	10	328	96.47
Class Recall (%)		96.37	99.08	95.42	93.98	

TABLE G.25: The Confusion Matrix of Boosting kNN with Default Parameters Applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	759	149	0	1	83.50
	Average	207	10585	726	5	91.86
	Poor	1	442	2715	73	84.03
	Terrible	0	3	43	299	86.67
Class Recall (%)		78.49	94.69	77.93	79.10	

TABLE G.26: The Confusion Matrix of Boosting kNN with Default Parameters Applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	909	40	0	0	95.79
	Average	57	11071	113	1	98.48
	Poor	0	67	3361	23	97.39
	Terrible	1	1	10	354	96.72
Class Recall (%)		94.00	99.03	96.47	93.65	

TABLE G.27: The Confusion Matrix of Boosting kNN with Default Parameters Applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	299	44	0	0	87.17
	Average	108	11171	754	3	92.81
	Poor	0	487	2850	62	83.85
	Terrible	1	0	39	180	81.82
Class Recall (%)		73.28	95.46	78.23	73.47	

TABLE G.28: The Confusion Matrix of Boosting kNN with Default Parameters Applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	368	13	0	0	96.59
	Average	40	11603	162	5	98.25
	Poor	0	83	3477	26	96.96
	Terrible	0	3	14	214	92.64
Class Recall (%)		90.20	99.15	95.18	87.35	

Table G.29: The Confusion Matrix of Boosting kNN with Default Parameters Applied to the full-attributes RottenTomatoes dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1822	270	2	0	87.01
	Average	431	11861	365	3	93.69
	Poor	4	219	930	11	79.90
	Terrible	0	6	13	71	78.89
Class Recall (%)		80.73	95.99	70.99	83.53	

Table G.30: The Confusion Matrix of Boosting kNN with Default Parameters Applied to the reduced-attributes RottenTomatoes dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	2165	70	0	0	96.87
	Average	91	12248	62	2	98.75
	Poor	1	32	1230	8	96.77
	Terrible	0	6	18	75	75.76
Class Recall (%)		95.92	99.13	93.89	88.24	

Appendix H

Confusion Matrixs of Rule Induction

This appendix contains all confusion matrixs gathered of experimentation using Rule Induction (RI) learner.

Table H.1: The Confusion Matrix of RI with default parameters applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1137	153	2	0	88.00
	Average	455	11412	802	22	89.92
	Poor	3	112	1552	81	88.79
	Terrible	1	4	26	246	88.81
Class Recall (%)		71.24	97.70	65.16	70.49	

Table H.2: The Confusion Matrix of RI with optimized parameters applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1537	71	4	0	95.35
	Average	56	11555	93	1	98.72
	Poor	2	54	2265	16	96.92
	Terrible	1	1	20	332	93.79
Class Recall (%)		96.80	98.92	95.09	95.13	

Table H.3: The Confusion Matrix of RI with default parameters applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1150	145	1	0	88.73
	Average	445	11416	802	20	90.01
	Poor	0	120	1558	79	88.67
	Terrible	1	0	21	250	91.91
Class Recall (%)		72.06	97.73	65.41	71.63	

Table H.4: The Confusion Matrix of RI with optimized parameters applied to the full-attributes IMDb++ dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1543	53	2	0	96.56
	Average	49	11550	81	1	98.88
	Poor	2	76	2288	17	96.01
	Terrible	2	2	11	331	95.67
Class Recall (%)		96.68	98.88	96.05	94.84	

Table H.5: The Confusion Matrix of RI with default parameters applied to the full-attributes FilmAffinity dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	629	80	2	1	88.34
	Average	337	10826	680	13	91.31
	Poor	0	269	2781	103	88.20
	Terrible	1	4	21	261	90.94
Class Recall (%)		65.05	96.84	79.82	69.05	

Table H.6: The Confusion Matrix of RI with optimized parameters applied to the full-attributes FilmAffinity dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	923	33	3	1	96.15
	Average	41	11069	82	1	98.90
	Poor	1	74	3387	24	97.16
	Terrible	2	3	12	352	95.39
Class Recall (%)		95.45	99.02	97.22	93.12	

Table H.7: The Confusion Matrix of RI with default parameters applied to the full-attributes MovieMeter dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	126	13	1	1	89.36
	Average	281	11407	717	9	91.88
	Poor	1	278	2914	107	88.30
	Terrible	0	4	21	128	83.66
Class Recall (%)		30.88	97.48	79.77	52.24	

Table H.8: The Confusion Matrix of RI with optimized parameters applied to the full-attributes MovieMeter dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	371	17	0	0	95.62
	Average	36	11577	125	2	98.61
	Poor	1	103	3509	27	96.40
	Terrible	0	5	19	216	90.00
Class Recall (%)		90.93	98.93	96.06	88.16	

Table H.9: The Confusion Matrix of RI with default parameters applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1593	182	1	1	89.65
	Average	656	12063	360	20	92.09
	Poor	6	107	937	23	87.33
	Terrible	2	4	12	41	69.49
Class Recall (%)		70.58	97.63	71.53	48.24	

Table H.10: The Confusion Matrix of RI with optimized parameters applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	2176	94	1	0	95.82
	Average	79	12196	56	3	98.88
	Poor	1	64	1245	8	94.46
	Terrible	1	2	8	74	87.06
Class Recall (%)		96.41	98.71	95.04	87.06	

Table H.11: The Confusion Matrix of RI with default parameters applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1111	144	8	0	87.97
	Average	484	11355	641	28	90.78
	Poor	0	180	1710	67	87.38
	Terrible	1	2	23	254	90.71
Class Recall (%)		69.61	97.21	71.79	72.78	

Table H.12: The Confusion Matrix of RI with optimized parameters applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1530	67	3	0	95.63
	Average	64	11525	78	0	98.78
	Poor	1	87	2287	14	95.73
	Terrible	1	2	14	335	95.17
Class Recall (%)		95.86	98.66	96.01	95.99	

Table H.13: The Confusion Matrix of RI with default parameters applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1050	125	0	0	89.36
	Average	544	11363	561	9	91.07
	Poor	2	186	1805	57	88.05
	Terrible	0	7	16	283	92.48
Class Recall (%)		65.79	97.28	75.78	81.09	

Table H.14: The Confusion Matrix of RI with optimized parameters applied to the reduced-attributes IMDb++ dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1545	63	3	0	95.90
	Average	49	11544	73	0	98.95
	Poor	2	71	2289	13	96.38
	Terrible	0	3	17	336	94.38
Class Recall (%)		96.80	98.83	96.10	96.28	

Table H.15: The Confusion Matrix of RI with default parameters applied to the reduced-attributes FilmAffinity dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	650	79	3	0	88.80
	Average	315	10818	641	13	91.78
	Poor	0	279	2813	91	88.38
	Terrible	2	3	27	274	96.72
Class Recall (%)		67.22	96.77	80.74	72.49	

Table H.16: The Confusion Matrix of RI with optimized parameters applied to the reduced-attributes FilmAffinity dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	910	41	4	0	95.29
	Average	55	11030	97	0	98.64
	Poor	1	108	3368	32	95.98
	Terrible	1	0	15	346	95.58
Class Recall (%)		94.12	98.67	96.67	91.53	

Table H.17: The Confusion Matrix of RI with default parameters applied to the reduced-attributes MovieMeter dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	122	13	2	1	88.41
	Average	285	11410	729	7	91.79
	Poor	1	274	2903	104	88.45
	Terrible	0	5	19	133	84.71
Class Recall (%)		29.90	97.50	79.47	54.29	

Table H.18: The Confusion Matrix of RI with optimized parameters applied to the reduced-attributes MovieMeter dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	374	17	0	0	95.65
	Average	33	11575	108	4	98.76
	Poor	1	108	3534	16	96.58
	Terrible	0	2	11	225	94.54
Class Recall (%)		91.67	98.91	96.74	91.84	

Table H.19: The Confusion Matrix of RI with default parameters applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1615	184	3	0	89.62
	Average	638	12070	362	21	92.20
	Poor	2	100	935	25	88.04
	Terrible	2	2	10	39	73.58
Class Recall (%)		71.56	97.69	71.37	45.88	

Table H.20: The Confusion Matrix of RI with optimized parameters applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	2175	77	2	0	96.50
	Average	80	12229	48	4	98.93
	Poor	1	48	1254	3	96.02
	Terrible	1	2	6	78	89.66
Class Recall (%)		96.37	98.97	95.73	91.76	

Table H.21: The Confusion Matrix of Bagging RI with Default Parameters Applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1192	83	1	0	93.42
	Average	400	11530	759	18	90.74
	Poor	3	68	1611	70	91.95
	Terrible	1	0	11	261	95.60
Class Recall (%)		74.69	98.71	67.63	74.79	

Table H.22: The Confusion Matrix of Bagging RI with Optimized Parameters Applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1539	30	0	0	98.09
	Average	56	11619	62	0	98.99
	Poor	0	32	2311	17	97.92
	Terrible	1	0	9	332	97.08
Class Recall (%)		96.43	99.47	97.02	95.13	

Table H.23: The Confusion Matrix of Bagging RI with Default Parameters Applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1163	70	2	0	94.17
	Average	432	11512	651	26	91.21
	Poor	0	99	1715	66	91.22
	Terrible	1	0	14	257	94.49
Class Recall (%)		72.87	98.55	72.00	73.64	

Table H.24: The Confusion Matrix of Bagging RI with Optimized Parameters Applied to the reduced-attributes IMDb dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1537	30	2	0	97.96
	Average	58	11612	72	0	98.89
	Poor	0	39	2298	14	97.75
	Terrible	1	0	10	335	96.82
Class Recall (%)		96.30	99.41	96.47	95.99	

Table H.25: The Confusion Matrix of Bagging RI with Default Parameters Applied to the full-attributes IMDb++ dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1189	81	2	0	93.47
	Average	405	11537	760	18	90.70
	Poor	1	63	1609	71	92.26
	Terrible	1	0	11	260	95.59
Class Recall (%)		74.50	98.77	67.55	74.50	

Table H.26: The Confusion Matrix of Bagging RI with Optimized Parameters Applied to the full-attributes IMDb++ dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1539	26	2	0	98.21
	Average	55	11624	65	0	98.98
	Poor	1	31	2309	19	97.84
	Terrible	1	0	6	330	97.92
Class Recall (%)		96.43	99.51	96.94	94.56	

Table H.27: The Confusion Matrix of Bagging RI with Default Parameters Applied to the reduced-attributes IMDb++ dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1064	86	0	0	92.52
	Average	529	11471	481	5	91.87
	Poor	2	122	1888	56	91.30
	Terrible	1	2	13	288	94.74
Class Recall (%)		66.67	98.20	79.26	82.52	

Table H.28: The Confusion Matrix of Bagging RI with Optimized Parameters Applied to the reduced-attributes IMDb++ dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1537	32	0	0	97.96
	Average	58	11602	70	0	98.91
	Poor	0	47	2308	18	97.26
	Terrible	1	0	4	331	98.51
Class Recall (%)		96.30	99.32	96.89	94.84	

TABLE H.29: The Confusion Matrix of Bagging RI with Default Parameters Applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	648	32	2	0	95.01
	Average	317	10981	602	10	92.20
	Poor	2	166	2862	95	91.58
	Terrible	0	0	18	273	93.81
Class Recall (%)		67.01	98.23	82.15	72.22	

TABLE H.30: The Confusion Matrix of Bagging RI with Optimized Parameters Applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	919	16	2	0	98.08
	Average	47	11119	61	0	99.04
	Poor	0	44	3415	32	97.82
	Terrible	1	0	6	346	98.02
Class Recall (%)		95.04	99.46	98.02	91.53	

TABLE H.31: The Confusion Matrix of Bagging RI with Default Parameters Applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	650	79	3	0	88.80
	Average	315	10818	641	13	91.78
	Poor	0	279	2813	91	88.38
	Terrible	2	3	27	274	89.54
Class Recall (%)		67.22	96.77	80.74	72.49	

TABLE H.32: The Confusion Matrix of Bagging RI with Optimized Parameters Applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	915	19	0	0	97.97
	Average	51	11107	58	0	99.03
	Poor	0	53	3421	28	97.69
	Terrible	1	0	5	350	98.31
Class Recall (%)		94.62	99.36	98.19	92.59	

TABLE H.33: The Confusion Matrix of Bagging RI with Default Parameters Applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	137	6	0	0	95.80
	Average	269	11519	643	4	92.63
	Poor	2	177	2997	103	91.40
	Terrible	0	0	13	138	91.39
Class Recall (%)		33.58	98.44	82.04	56.33	

Table H.34: The Confusion Matrix of Bagging RI with Optimized Parameters Applied to the full-attributes MovieMeter dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	372	4	0	0	98.94
	Average	36	11617	87	0	98.95
	Poor	0	81	3558	24	97.13
	Terrible	0	0	8	221	96.51
Class Recall (%)		91.18	99.27	97.40	90.20	

Table H.35: The Confusion Matrix of Bagging RI with Default Parameters Applied to the reduced-attributes MovieMeter dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	135	6	1	0	95.07
	Average	273	11515	628	1	92.74
	Poor	0	180	3009	104	91.38
	Terrible	0	1	15	140	89.74
Class Recall (%)		33.09	98.40	82.37	57.14	

Table H.36: The Confusion Matrix of Bagging RI with Optimized Parameters Applied to the reduced-attributes MovieMeter dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	372	7	2	0	97.64
	Average	36	11625	90	0	98.93
	Poor	0	69	3555	25	97.42
	Terrible	0	1	6	220	96.92
Class Recall (%)		91.18	99.34	97.32	89.80	

Table H.37: The Confusion Matrix of Bagging RI with Default Parameters Applied to the full-attributes RottenTomatoes dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1632	130	2	1	92.46
	Average	624	12160	345	21	92.47
	Poor	1	65	957	17	92.02
	Terrible	0	1	6	46	86.79
Class Recall (%)		72.31	98.41	73.05	54.12	

Table H.38: The Confusion Matrix of Bagging RI with Optimized Parameters Applied to the full-attributes RottenTomatoes dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	2177	43	0	0	98.06
	Average	80	12297	51	4	98.91
	Poor	0	15	1251	5	98.43
	Terrible	0	1	8	76	89.41
Class Recall (%)		96.46	99.52	95.50	89.41	

TABLE H.39: The Confusion Matrix of Bagging RI with Default Parameters Applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1633	123	2	1	92.84
	Average	624	12158	336	23	92.52
	Poor	0	73	967	24	90.88
	Terrible	0	2	5	37	84.09
Class Recall (%)		72.35	98.40	73.82	43.53	

TABLE H.40: The Confusion Matrix of Bagging RI with Optimized Parameters Applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	2179	47	0	0	97.89
	Average	78	12288	48	3	98.96
	Poor	0	20	1258	4	98.13
	Terrible	0	1	4	78	93.98
Class Recall (%)		96.54	99.45	96.03	91.76	

TABLE H.41: The Confusion Matrix of Boosting-RI with Default Parameters applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1421	20	0	0	98.61
	Average	175	11643	224	10	96.61
	Poor	0	18	2146	20	98.26
	Terrible	0	0	12	319	96.37
Class Recall (%)		89.04	99.67	90.09	91.40	

TABLE H.42: The Confusion Matrix of Boosting-RI with Optimized Parameters applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1527	68	4	0	95.50
	Average	67	11541	101	1	98.56
	Poor	1	70	2266	16	96.30
	Terrible	1	2	11	332	95.95
Class Recall (%)		95.68	98.80	95.13	95.13	

TABLE H.43: The Confusion Matrix of Boosting-RI with Default Parameters Applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1386	27	0	0	98.09
	Average	209	11625	271	7	95.98
	Poor	0	29	2102	26	97.45
	Terrible	1	0	9	316	96.93
Class Recall (%)		86.84	99.52	88.25	90.54	

TABLE H.44: The Confusion Matrix of Boosting-RI with Optimized Parameters Applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1530	67	3	0	95.63
	Average	64	11525	78	0	98.78
	Poor	1	87	2287	14	95.73
	Terrible	1	2	14	335	95.17
Class Recall (%)		95.86	98.66	96.01	95.99	

TABLE H.45: The Confusion Matrix of Boosting-RI with Default Parameters applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1394	25	0	0	98.24
	Average	201	11640	225	1	96.46
	Poor	0	16	2146	18	98.44
	Terrible	1	0	11	330	96.49
Class Recall (%)		87.34	99.65	90.09	94.56	

TABLE H.46: The Confusion Matrix of Boosting-RI with Optimized Parameters applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1543	53	2	0	96.56
	Average	49	11550	81	1	98.88
	Poor	2	76	2288	17	96.01
	Terrible	2	2	11	331	95.66
Class Recall (%)		96.68	98.88	96.05	94.84	

TABLE H.47: The Confusion Matrix of Boosting-RI with Default Parameters Applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1274	30	0	0	97.70
	Average	322	11633	392	9	94.15
	Poor	0	18	1982	25	97.88
	Terrible	0	0	8	315	97.52
Class Recall (%)		79.82	99.59	83.21	90.26	

TABLE H.48: The Confusion Matrix of Boosting-RI with Optimized Parameters Applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1541	59	2	0	96.19
	Average	53	11560	88	0	98.79
	Poor	1	61	2278	15	96.73
	Terrible	1	1	14	334	95.43
Class Recall (%)		96.55	98.96	95.63	95.70	

Appendix I

Confusion Matrixs of Rule Induction - continuation

This appendix contains all confusion matrixs gathered of experimentation using Rule Induction (RI) learner.

TABLE I.1: The Confusion Matrix of Boosting-RI with Default Parameters applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	810	14	0	0	98.30
	Average	157	11137	157	0	97.26
	Poor	0	28	3322	38	98.05
	Terrible	0	0	5	340	98.55
Class Recall (%)		83.76	99.62	95.35	89.95	

TABLE I.2: The Confusion Matrix of Boosting-RI with Optimized Parameters applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	916	41	2	0	95.52
	Average	49	11050	89	2	98.75
	Poor	1	87	3381	20	96.90
	Terrible	1	1	12	356	96.22
Class Recall (%)		94.73	98.85	97.04	94.18	

TABLE I.3: The Confusion Matrix of Boosting-RI with Default Parameters Applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	749	14	0	0	98.17
	Average	217	11136	205	3	96.32
	Poor	0	29	3272	33	98.14
	Terrible	1	0	7	342	97.71
Class Recall (%)		77.46	99.62	93.92	90.48	

TABLE I.4: The Confusion Matrix of Boosting-RI with Optimized Parameters Applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	910	41	4	0	95.29
	Average	55	11030	97	0	98.64
	Poor	1	108	3368	32	95.98
	Terrible	1	0	15	346	95.58
Class Recall (%)		94.11	98.67	96.67	91.53	

TABLE I.5: The Confusion Matrix of Boosting-RI with Default Parameters applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	216	2	0	0	99.08
	Average	192	11662	373	0	95.38
	Poor	0	38	3270	28	98.02
	Terrible	0	0	10	217	95.59
Class Recall (%)		52.94	99.66	89.52	88.57	

TABLE I.6: The Confusion Matrix of Boosting-RI with Optimized Parameters applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	371	17	0	0	95.62
	Average	36	11577	125	2	98.61
	Poor	1	103	3509	27	96.40
	Terrible	0	5	19	216	90.00
Class Recall (%)		90.93	98.93	96.06	88.16	

TABLE I.7: The Confusion Matrix of Boosting-RI with Default Parameters Applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	188	0	0	0	100.00
	Average	219	11654	479	6	94.30
	Poor	1	48	3165	37	97.35
	Terrible	0	0	9	202	95.73
Class Recall (%)		46.08	99.59	86.64	82.45	

Table I.8: The Confusion Matrix of Boosting-RI with Optimized Parameters Applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	374	17	0	0	95.65
	Average	33	11575	108	4	98.76
	Poor	1	108	3534	16	96.58
	Terrible	0	2	11	225	94.54
Class Recall (%)		91.67	98.91	96.74	91.84	

Table I.9: The Confusion Matrix of Boosting-RI with Default Parameters applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1956	24	1	0	98.74
	Average	301	12320	254	11	95.61
	Poor	0	12	1051	18	97.22
	Terrible	0	0	4	56	93.33
Class Recall (%)		86.66	99.71	80.23	65.88	

Table I.10: The Confusion Matrix of Boosting-RI with Optimized Parameters applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	2176	94	1	0	95.82
	Average	79	12196	56	3	98.88
	Poor	1	64	1245	8	94.46
	Terrible	1	2	8	74	87.06
Class Recall (%)		96.41	98.71	95.04	87.06	

Table I.11: The Confusion Matrix of Boosting-RI with Default Parameters Applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1887	1	44	0	97.67
	Average	0	1015	19	17	96.57
	Poor	370	289	12292	13	94.82
	Terrible	0	5	1	55	90.16
Class Recall (%)		83.61	77.48	99.48	64.71	

Table I.12: The Confusion Matrix of Boosting-RI with Optimized Parameters Applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	2175	77	2	0	96.50
	Average	80	12229	48	4	98.93
	Poor	1	48	1254	3	96.02
	Terrible	1	2	6	78	89.66
Class Recall (%)		96.37	98.97	95.73	91.76	

Appendix J

Confusion Matrixs of SVM with the Polynomial Kernel

This appendix contains all confusion matrixs gathered of experimentation using Support Vector Machine (SVM) classifier with the Polynomial Kernell.

TABLE J.1: The Confusion Matrix of SVM Polynomial with default parameters applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	633	157	0	0	80.13
	Average	953	11190	915	9	85.64
	Poor	10	326	1429	109	76.25
	Terrible	0	8	38	231	83.39
Class Recall (%)		39.66	95.80	59.99	66.19	

TABLE J.2: The Confusion Matrix of SVM Polynomial with optimized parameters applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1238	197	0	0	86.27
	Average	347	11140	423	1	93.53
	Poor	10	338	1929	44	83.11
	Terrible	1	6	30	304	89.15
Class Recall (%)		77.57	95.37	80.98	87.11	

TABLE J.3: The Confusion Matrix of SVM Polynomial with default parameters applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	868	225	0	0	79.41
	Average	719	11108	897	7	87.25
	Poor	9	343	1446	115	75.59
	Terrible	0	5	39	227	83.76
Class Recall (%)		54.39	95.09	60.71	65.04	

TABLE J.4: The Confusion Matrix of SVM Polynomial with optimized parameters applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1439	105	1	0	93.14
	Average	150	11295	341	1	95.83
	Poor	6	277	2002	48	85.81
	Terrible	1	4	38	300	87.46
Class Recall (%)		90.16	96.70	84.05	85.96	

TABLE J.5: The Confusion Matrix of SVM Poly with default parameters applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	432	90	0	0	82.76
	Average	529	10592	1206	5	85.89
	Poor	6	497	2238	161	77.12
	Terrible	0	0	40	212	84.13
Class Recall (%)		44.67	94.75	64.24	56.08	

TABLE J.6: The Confusion Matrix of SVM Poly with optimized parameters applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	855	85	0	0	90.96
	Average	104	10678	477	1	94.83
	Poor	7	410	2962	59	86.15
	Terrible	1	6	45	318	85.95
Class Recall (%)		88.42	95.52	85.02	84.13	

TABLE J.7: The Confusion Matrix of SVM Polynomial with default parameters applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	18	0	0	0	100.00
	Average	381	11169	1316	10	86.74
	Poor	9	533	2337	235	75.05
	Terrible	0	0	0	0	00.00
Class Recall (%)		04.41	95.45	63.97	00.00	

TABLE J.8: The Confusion Matrix of SVM Polynomial with optimized parameters applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	196	3	40	0	82.01
	Average	0	11157	506	55	95.21
	Poor	49	501	3107	1	84.94
	Terrible	0	41	0	352	89.57
Class Recall (%)		80.00	95.34	85.05	86.27	

TABLE J.9: The Confusion Matrix of SVM Polynomial with default parameters applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1325	241	2	0	84.50
	Average	930	11986	721	7	87.85
	Poor	2	129	587	78	73.74
	Terrible	0	0	0	0	00.00
Class Recall (%)		58.71	97.01	44.81	00.00	

TABLE J.10: The Confusion Matrix of SVM Polynomial with optimized parameters applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	2016	164	0	0	92.48
	Average	237	11976	270	8	95.88
	Poor	2	211	1032	13	82.03
	Terrible	2	5	8	64	81.01
Class Recall (%)		89.32	96.92	78.78	75.29	

TABLE J.11: The Confusion Matrix of SVM Polynomial with default parameters applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	738	159	0	0	82.27
	Average	854	11174	906	16	86.29
	Poor	3	346	1454	125	75.41
	Terrible	1	2	22	208	90.71
Class Recall (%)		46.24	95.66	61.04	59.60	

TABLE J.12: The Confusion Matrix of SVM Polynomial with optimized parameters applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	738	159	0	0	82.27
	Average	854	11174	906	16	86.29
	Poor	3	346	1454	125	75.41
	Terrible	1	2	22	208	90.71
Class Recall (%)		46.24	95.66	61.04	59.60	

TABLE J.13: The Confusion Matrix of SVM Polynomial with default parameters applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	932	212	0	0	81.47
	Average	662	11133	890	12	87.68
	Poor	1	334	1458	110	76.62
	Terrible	1	2	34	227	85.98
Class Recall (%)		58.40	95.31	61.21	65.04	

Table J.14: The Confusion Matrix of SVM Polynomial with optimized parameters applied to the reduced-attributes IMDb++ dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	932	212	0	0	81.47
	Average	662	11133	890	12	87.68
	Poor	1	334	1458	110	76.62
	Terrible	1	2	34	227	85.98
Class Recall (%)		58.40	95.31	61.21	65.04	

Table J.15: The Confusion Matrix of SVM Polynomial with Default Parameters applied to the reduced-attributes FilmAffinity dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	442	85	0	0	83.87
	Average	522	10617	1191	6	86.07
	Poor	2	476	2268	188	77.30
	Terrible	1	1	25	184	87.20
Class Recall (%)		45.71	94.97	65.10	48.68ffff	

Table J.16: The Confusion Matrix of SVM Polynomial with Optimized Parameters applied to the reduced-attributes FilmAffinity dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	882	60	1	0	93.53
	Average	82	10820	443	0	95.37
	Poor	2	297	3010	29	90.17
	Terrible	1	2	30	349	91.36
Class Recall (%)		91.21	96.79	86.39	92.33	

Table J.17: The Confusion Matrix of SVM Polynomial with Default Parameters applied to the reduced-attributes MovieMeter dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	7	0	0	0	100.00
	Average	395	11174	1249	16	87.07
	Poor	6	528	2404	229	75.91
	Terrible	0	0	0	0	00.00
Class Recall (%)		62.07	96.82	47.18	00.00	

Table J.18: The Confusion Matrix of SVM Polynomial with Optimized Parameters applied to the reduced-attributes MovieMeter dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	372	23	0	0	94.18
	Average	36	11256	491	0	95.53
	Poor	0	421	3130	35	87.28
	Terrible	0	2	32	210	86.07
Class Recall (%)		91.18	96.19	85.68	85.71	

Table J.19: The Confusion Matrix of SVM Polynomial with default parameters applied to the reduced-attributes RottenTomatoes dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1401	271	2	0	83.69
	Average	845	11963	690	9	88.57
	Poor	11	122	618	76	74.73
	Terrible	0	0	0	0	00.00
Class Recall (%)		62.07	96.82	47.18	00.00	

Table J.20: The Confusion Matrix of SVM Polynomial with optimized parameters applied to the reduced-attributes RottenTomatoes dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1497	69	1	0	95.53
	Average	95	11400	308	0	96.59
	Poor	4	205	2055	21	89.93
	Terrible	0	7	18	328	92.92
Class Recall (%)		93.80	97.59	86.27	93.98	

Table J.21: The Confusion Matrix of Bagging SVM Polynomial with Default Parameters Applied to the full-attributes IMDb dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	608	146	1	0	80.53
	Average	977	11200	914	9	85.50
	Poor	11	327	1421	107	76.15
	Terrible	0	8	46	233	81.18
Class Recall (%)		38.10	95.88	59.66	66.76	

Table J.22: The Confusion Matrix of Bagging SVM Polynomial with Optimized Parameters Applied to the full-attributes IMDb dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1233	204	0	0	85.80
	Average	355	11077	415	1	93.49
	Poor	7	395	1931	40	81.37
	Terrible	1	5	36	308	88.00
Class Recall (%)		77.26	94.83	81.07	88.25	

Table J.23: The Confusion Matrix of Bagging SVM Polynomial with Default Parameters Applied to the reduced-attributes IMDb dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	731	152	0	0	82.79
	Average	860	11180	895	15	86.33
	Poor	4	345	1461	119	75.74
	Terrible	1	4	26	215	87.40
Class Recall (%)		45.80	95.71	61.33	61.60	

Table J.24: The Confusion Matrix of Bagging SVM Polynomial with Optimized Parameters Applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1279	168	0	0	88.39
	Average	312	11200	400	0	94.02
	Poor	4	309	1965	24	85.36
	Terrible	1	4	17	325	93.66
Class Recall (%)		80.14	95.88	82.49	93.12	

Table J.25: The Confusion Matrix of Bagging SVM Polynomial with Default Parameters Applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	849	224	0	0	79.12
	Average	738	11105	905	7	87.06
	Poor	9	348	1434	112	75.35
	Terrible	0	4	43	230	83.03
Class Recall (%)		53.20	95.07	60.20	65.90	

Table J.26: The Confusion Matrix of Bagging SVM Polynomial with Optimized Parameters Applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1412	118	1	0	92.23
	Average	176	11204	327	1	95.70
	Poor	7	355	2008	40	83.32
	Terrible	1	4	46	308	85.79
Class Recall (%)		88.47	95.92	84.30	88.25	

Table J.27: The Confusion Matrix of Bagging SVM Polynomial with Default Parameters Applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	909	209	0	0	81.31
	Average	684	11131	887	12	87.55
	Poor	2	338	1461	108	76.53
	Terrible	1	3	34	229	85.77
Class Recall (%)		56.95	95.29	61.33	65.62	

Table J.28: The Confusion Matrix of Bagging SVM Polynomial with Optimized Parameters Applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1433	101	1	0	93.36
	Average	158	11318	297	0	96.14
	Poor	4	255	2065	22	88.02
	Terrible	1	7	19	327	92.37
Class Recall (%)		89.79	96.89	86.69	93.70	

Table J.29: The Confusion Matrix of Bagging SVM Polynomial with Default Parameters Applied to the full-attributes FilmAffinity dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	420	84	0	0	83.33
	Average	540	10592	1198	4	85.88
	Poor	7	503	2249	162	76.99
	Terrible	0	0	37	212	85.14
Class Recall (%)		43.43	94.75	64.55	56.08	

Table J.30: The Confusion Matrix of Bagging SVM Polynomial with Optimized Parameters Applied to the full-attributes FilmAffinity dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	829	101	1	0	89.04
	Average	132	10583	447	1	94.80
	Poor	5	491	2990	50	84.56
	Terrible	1	4	46	327	86.51
Class Recall (%)		85.73	94.67	85.82	86.51	

Table J.31: The Confusion Matrix of Bagging SVM Polynomial with Default Parameters Applied to the reduced-attributes FilmAffinity dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	445	83	0	0	84.28
	Average	519	10614	1182	6	86.15
	Poor	2	481	2277	193	77.11
	Terrible	1	4	25	179	86.89
Class Recall (%)		46.02	94.95	65.36	47.35	

Table J.32: The Confusion Matrix of Bagging SVM Polynomial with Optimized Parameters Applied to the reduced-attributes FilmAffinity dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	873	60	1	0	93.47
	Average	91	10765	407	0	95.58
	Poor	2	352	3042	27	88.87
	Terrible	1	2	34	351	90.46
Class Recall (%)		90.28	96.30	87.31	92.86	

Table J.33: The Confusion Matrix of Bagging SVM Polynomial with Default Parameters Applied to the full-attributes MovieMeter dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	16	0	0	0	100.00
	Average	382	11157	1314	9	86.74
	Poor	10	545	2339	236	74.73
	Terrible	0	0	0	0	00.00
Class Recall (%)		03.92	95.34	64.03	00.00	

Table J.34: The Confusion Matrix of Bagging SVM Polynomial with Optimized Parameters Applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	347	54	0	1	86.32
	Average	59	11043	498	0	95.20
	Poor	2	602	3113	52	82.59
	Terrible	0	3	42	192	81.01
Class Recall (%)		85.05	94.37	85.22	78.37	

Table J.35: The Confusion Matrix of Bagging SVM Polynomial with Default Parameters Applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	8	0	0	0	100.00
	Average	394	11162	1248	15	87.07
	Poor	6	540	2405	230	75.61
	Terrible	0	0	0	0	00.00
Class Recall (%)		01.96	95.39	65.84	00.00	

Table J.36: The Confusion Matrix of Bagging SVM Polynomial with Optimized Parameters Applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	374	29	0	0	92.80
	Average	33	11203	465	0	95.74
	Poor	1	468	3154	38	86.15
	Terrible	0	2	34	207	85.19
Class Recall (%)		91.67	95.74	86.33	84.49	

Table J.37: The Confusion Matrix of Bagging SVM Polynomial with Default Parameters Applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1299	235	2	0	84.57
	Average	954	11998	735	7	87.62
	Poor	2	123	573	78	73.84
	Terrible	2	0	0	0	00.00
Class Recall (%)		57.55	97.10	43.74	00.00	

Table J.38: The Confusion Matrix of Bagging SVM Polynomial with Optimized Parameters Applied to the full-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	2002	191	0	0	91.29
	Average	249	11989	304	7	95.54
	Poor	0	169	993	16	84.30
	Terrible	6	7	13	62	70.45
Class Recall (%)		88.70	97.03	75.80	72.94	

Table J.39: The Confusion Matrix of Bagging SVM Polynomial with Default Parameters Applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1389	277	2	0	83.27
	Average	859	11957	700	9	88.41
	Poor	9	122	608	76	74.60
	Terrible	0	0	0	0	00.00
Class Recall (%)		61.54	96.77	46.41	00.00	

Table J.40: The Confusion Matrix of Bagging SVM Polynomial with Optimized Parameters Applied to the reduced-attributes RottenTomatoes dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	2060	119	0	0	94.54
	Average	188	12118	241	1	96.57
	Poor	4	117	1060	11	88.93
	Terrible	5	2	9	73	82.02
Class Recall (%)		91.27	98.07	80.92	85.88	

Table J.41: The Confusion Matrix of Boosting SVM Polynomial with Default Parameters applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	637	149	0	0	81.04
	Average	949	11196	910	10	85.69
	Poor	10	328	1434	107	76.32
	Terrible	0	8	38	232	83.45
Class Recall (%)		39.91	95.85	60.20	66.48	

Table J.42: The Confusion Matrix of Boosting SVM Polynomial with Optimized Parameters applied to the full-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1242	212	0	0	85.42
	Average	347	11115	445	1	93.34
	Poor	6	349	1905	47	82.57
	Terrible	1	5	32	301	88.79
Class Recall (%)		77.82	95.15	79.97	86.25	

Table J.43: The Confusion Matrix of Boosting SVM Polynomial with Default Parameters Applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	738	159	0	0	82.27
	Average	854	11174	906	16	86.29
	Poor	3	346	1454	125	75.41
	Terrible	1	2	22	208	89.27
Class Recall (%)		46.24	95.66	61.04	59.60	

Table J.44: The Confusion Matrix of Boosting SVM Polynomial with Optimized Parameters Applied to the reduced-attributes IMDb dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1292	174	0	0	88.13
	Average	297	11229	418	0	94.01
	Poor	6	274	1952	25	86.49
	Terrible	1	4	12	324	95.01
Class Recall (%)		80.95	96.13	81.95	92.84	

Table J.45: The Confusion Matrix of Boosting SVM Polynomial with Default Parameters applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	868	225	0	0	79.41
	Average	719	11108	897	7	87.25
	Poor	9	343	1446	115	75.59
	Terrible	0	5	39	227	83.76
Class Recall (%)		54.39	95.09	60.71	65.04	

Table J.46: The Confusion Matrix of Boosting SVM Polynomial with Optimized Parameters applied to the full-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1439	105	1	0	93.14
	Average	150	11295	341	1	95.83
	Poor	6	277	2002	48	85.81
	Terrible	1	4	38	300	87.46
Class Recall (%)		90.16	96.70	84.05	85.96	

Table J.47: The Confusion Matrix of Boosting SVM Polynomial with Default Parameters Applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	932	212	0	0	81.47
	Average	662	11133	890	12	87.68
	Poor	3	334	1458	110	76.62
	Terrible	1	2	34	227	85.98
Class Recall (%)		58.40	95.31	61.21	65.04	

Table J.48: The Confusion Matrix of Boosting SVM Polynomial with Optimized Parameters Applied to the reduced-attributes IMDb++ dataset

		Actual				Class
		Excellent	Average	Poor	Terrible	Precision (%)
Predicted	Excellent	1459	96	3	0	93.65
	Average	134	11361	307	0	96.26
	Poor	2	219	2057	23	89.40
	Terrible	1	5	15	326	93.95
Class Recall (%)		91.42	97.26	86.36	93.41	

Appendix K

Confusion Matrixs of SVM with the Polynomial Kerne - continuationl

This appendix contains all confusion matrixs gathered of experimentation using Support Vector Machine (SVM) classifier with the Polynomial Kernell.

TABLE K.1: The Confusion Matrix of Boosting SVM Polynomial with Default Parameters applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	432	90	0	0	82.76
	Average	529	10592	1206	5	85.89
	Poor	6	497	2238	161	77.12
	Terrible	0	0	40	212	84.13
Class Recall (%)		44.67	94.75	64.24	56.08	

TABLE K.2: The Confusion Matrix of Boosting SVM Polynomial with Optimized Parameters applied to the full-attributes FilmAffinity dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	855	85	0	0	90.96
	Average	104	10678	477	1	94.83
	Poor	7	410	2962	59	86.15
	Terrible	1	6	45	318	85.95
Class Recall (%)		88.42	95.52	85.02	84.13	

Table K.3: The Confusion Matrix of Boosting SVM Polynomial with Default Parameters applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	442	85	0	0	83.87
	Average	522	10617	1191	6	86.07
	Poor	2	476	2268	188	77.30
	Terrible	1	1	25	184	87.20
Class Recall (%)		45.71	94.97	65.10	48.68	

Table K.4: The Confusion Matrix of Boosting SVM Polynomial with Optimized Parameters applied to the reduced-attributes FilmAffinity dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	882	60	1	0	93.53
	Average	82	10820	443	0	95.37
	Poor	2	297	3010	29	90.17
	Terrible	1	2	30	349	91.36
Class Recall (%)		91.21	96.79	86.39	92.33	

Table K.5: The Confusion Matrix of Boosting SVM Polynomial with Default Parameters applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	18	0	0	0	100.00
	Average	381	11169	1316	10	86.74
	Poor	9	533	2337	235	75.05
	Terrible	0	0	0	0	0.00
Class Recall (%)		4.41	95.45	63.97	0.00	

Table K.6: The Confusion Matrix of Boosting SVM Polynomial with Optimized Parameters applied to the full-attributes MovieMeter dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	352	41	0	0	89.57
	Average	55	11157	506	10	95.21
	Poor	1	501	3107	49	84.94
	Terrible	0	3	40	196	82.01
Class Recall (%)		86.27	95.34	85.05	80.00	

Table K.7: The Confusion Matrix of Boosting SVM Polynomial with Default Parameters applied to the reduced-attributes MovieMeter dataset

		Actual				Class
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	7	0	0	0	100.00
	Average	395	11174	1249	16	87.07
	Poor	6	528	2404	229	75.91
	Terrible	0	0	0	0	0.00
Class Recall (%)		1.72	95.49	65.81	0.00	

Table K.8: The Confusion Matrix of Boosting SVM Polynomial with Optimized Parameters applied to the reduced-attributes MovieMeter dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	372	23	0	0	94.18
	Average	36	11256	491	0	95.53
	Poor	0	421	3130	35	87.28
	Terrible	0	2	32	210	86.07
Class Recall (%)		91.18	96.19	85.68	85.71	

Table K.9: The Confusion Matrix of Boosting SVM Polynomial with Default Parameters applied to the full-attributes RottenTomatoes dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1325	241	2	0	84.50
	Average	930	11986	721	7	87.85
	Poor	2	129	587	78	73.74
	Terrible	0	0	0	0	0
Class Recall (%)		58.71	97.01	44.81	0	

Table K.10: The Confusion Matrix of Boosting SVM Polynomial with Optimized Parameters applied to the full-attributes RottenTomatoes dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	2016	164	0	0	92.48
	Average	237	11976	270	8	95.88
	Poor	2	211	1032	13	82.03
	Terrible	2	5	8	64	81.01
Class Recall (%)		89.32	96.92	78.78	75.29	

Table K.11: The Confusion Matrix of Boosting SVM Polynomial with Default Parameters applied to the reduced-attributes RottenTomatoes dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	1401	271	2	0	83.69
	Average	845	11963	690	9	88.57
	Poor	11	122	618	76	74.73
	Terrible	0	0	0	0	0.00
Class Recall (%)		62.07	96.82	47.18	0.00	

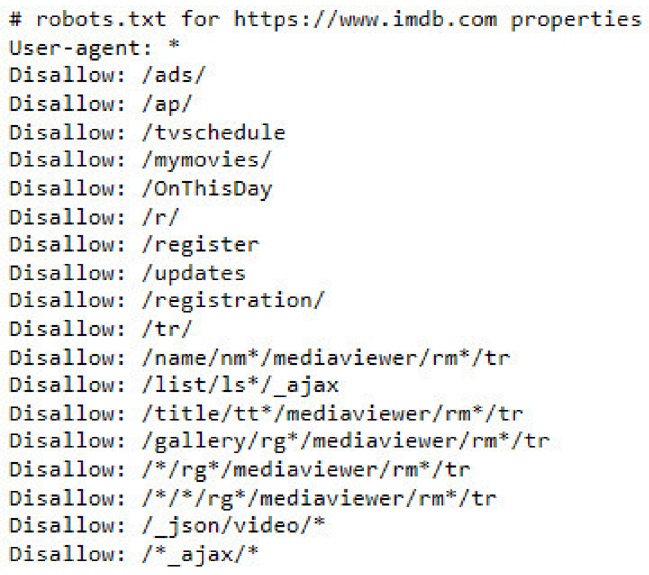
Table K.12: The Confusion Matrix of Boosting SVM Polynomial with Optimized Parameters applied to the reduced-attributes RottenTomatoes dataset

		Actual				**Class**
		Excellent	**Average**	**Poor**	**Terrible**	**Precision (%)**
Predicted	**Excellent**	2054	121	0	0	94.44
	Average	200	12096	225	2	96.59
	Poor	1	137	1076	11	87.84
	Terrible	2	2	9	72	84.71
Class Recall (%)		91.01	97.90	82.14	84.71	

Appendix L

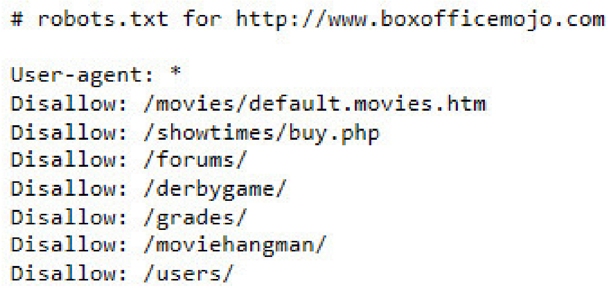
The Web Robot Files

This appendix contains some web robot files used in the IMDb, BoxOfficeMojo websites.



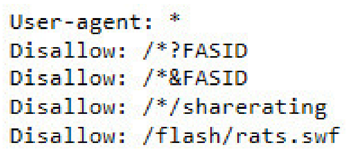
```
# robots.txt for https://www.imdb.com properties
User-agent: *
Disallow: /ads/
Disallow: /ap/
Disallow: /tvschedule
Disallow: /mymovies/
Disallow: /OnThisDay
Disallow: /r/
Disallow: /register
Disallow: /updates
Disallow: /registration/
Disallow: /tr/
Disallow: /name/nm*/mediaviewer/rm*/tr
Disallow: /list/ls*/_ajax
Disallow: /title/tt*/mediaviewer/rm*/tr
Disallow: /gallery/rg*/mediaviewer/rm*/tr
Disallow: /*/rg*/mediaviewer/rm*/tr
Disallow: /*/*/rg*/mediaviewer/rm*/tr
Disallow: /_json/video/*
Disallow: /*_ajax/*
```

Figure L.1: The Web Robots for the IMDb website



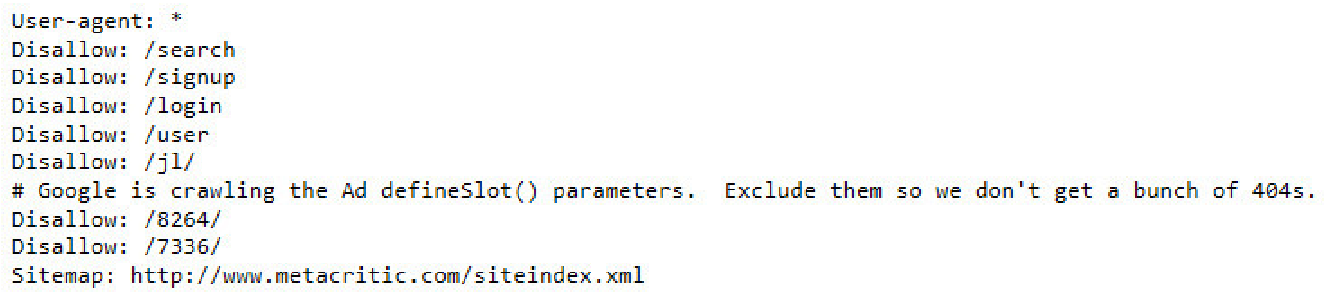
```
# robots.txt for http://www.boxofficemojo.com

User-agent: *
Disallow: /movies/default.movies.htm
Disallow: /showtimes/buy.php
Disallow: /forums/
Disallow: /derbygame/
Disallow: /grades/
Disallow: /moviehangman/
Disallow: /users/
```

Figure L.2: The Web Robots for the BoxOfficeMojo website



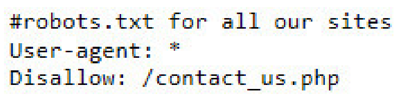
```
User-agent: *
Disallow: /*?FASID
Disallow: /*&FASID
Disallow: /*/sharerating
Disallow: /flash/rats.swf
```

FIGURE L.3: The Web Robots for the FilmAffinity website



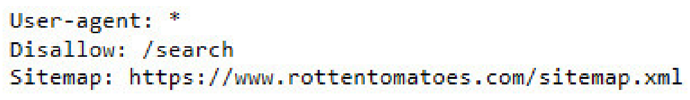
```
User-agent: *
Disallow: /search
Disallow: /signup
Disallow: /login
Disallow: /user
Disallow: /jl/
# Google is crawling the Ad defineSlot() parameters.  Exclude them so we don't get a bunch of 404s.
Disallow: /8264/
Disallow: /7336/
Sitemap: http://www.metacritic.com/siteindex.xml
```

FIGURE L.4: The Web Robots for the Metacritic website



```
#robots.txt for all our sites
User-agent: *
Disallow: /contact_us.php
```

FIGURE L.5: The Web Robots for the MovieMeter website



```
User-agent: *
Disallow: /search
Sitemap: https://www.rottentomatoes.com/sitemap.xml
```

FIGURE L.6: The Web Robots for the RottenTomatoes website

Appendix M

Summary of Input Attributes of the Datasets

This appendix details input attributes of the following datasets: IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes.

Table M.1: Summary of Input Attributes for The IMDb Dataset

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
1	Actor Rank (Sum)	1	Positive Number	IMDb List Files	Saraee et al. (2004); Asad et al. (2012)	Native
2	Actress Rank (Sum)	1	Positive Number	IMDb List Files	Saraee et al. (2004); Asad et al. (2012)	Native
3	Budget	9	1 - 9	IMDb List Files	Saraee et al. (2004); Asad et al. (2012); Latif and Afzal (2016)	Native
4	Cinematographer Rank (Avg)	1	Positive Number	IMDb List Files		Native
5	Competition	3	High, Medium, Low	IMDb List Files	Simonoff and Sparrow (2000); Sharda and Delen (2006)	Native
6	Composer Rank (Avg)	1	Positive Number	IMDb List Files		Native
7	Costume Designer Rank (Avg)	1	Positive Integer	IMDb List Files		Native
8	Director Rank (Avg)	1	Positive Number	IMDb List Files	Saraee et al. (2004); Asad et al. (2012)	Native
9	Distributor Rank (Sum)	1	Positive Number	IMDb List Files		Native
10	Editor Rank (Avg)	1	Positive Number	IMDb List Files		Native

Continued on next page

Table M.1 – continued from previous page

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
11	Genre	20	Action, adventure, thriller, biography, crime, drama, horror, comedy, fantasy, animation, mystery, music, war, documentary, romance, sci-fi, western, family, sport, short	IMDb List Files	Sharda and Delen (2006); Latif and Afzal (2016)	Native
12	MPAA	5	R, PG, PG-13, G, NR	IMDb List Files	Simonoff and Sparrow (2000); Sharda and Delen (2006); Latif and Afzal (2016)	Native
13	Production Company Rank (Avg)	1	Positive Number	IMDb List Files		Native
14	Production Designer Rank (Avg)	1	Positive Number	IMDb List Files		Native
15	Producer Rank (Avg)	1	Positive Number	IMDb List Files		Native
16	Writer Rank (Avg)	1	Positive Number	IMDb List Files		Native

Table M.2: Summary of Input Attributes for The IMDb++ Dataset

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
1	Actor Rank (Sum)	1	Positive Number	IMDb List Files	Saraee et al. (2004); Asad et al. (2012)	Native
2	Actress Rank (Sum)	1	Positive Number	IMDb List Files	Saraee et al. (2004); Asad et al. (2012)	Native

Continued on next page

Table M.2 – continued from previous page

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
3	Budget	9	1 - 9	IMDb List Files	Saraee et al. (2004); Asad et al. (2012); Latif and Afzal (2016)	Native
4	Cinematographer Rank (Avg)	1	Positive Number	IMDb List Files		Native
5	Competition	3	High, Medium, Low	IMDb List Files	Simonoff and Sparrow (2000); Sharda and Delen (2006)	Native
6	Composer Rank (Avg)	1	Positive Number	IMDb List Files		Native
7	Costume Designer Rank (Avg)	1	Positive Number	IMDb List Files		Native
8	Director Rank (Avg)	1	Positive Number	IMDb List Files	Saraee et al. (2004); Asad et al. (2012)	Native
9	Distributor Rank (Sum)	1	Positive Number	IMDb List Files		Native
10	Editor Rank (Avg)	1	Positive Number	IMDb List Files		Native
11	Genre	20	Action, adventure, thriller, biography, crime, drama, horror, comedy, fantasy, animation, mystery, music, war, documentary, romance, sci-fi, western, family, sport, short	IMDb List Files	Sharda and Delen (2006); Latif and Afzal (2016)	Native

Continued on next page

Table M.2 – continued from previous page

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
12	MPAA	5	R, PG, PG-13, G, NR	IMDb List Files	Simonoff and Sparrow (2000); Sharda and Delen (2006); Latif and Afzal (2016)	Native
13	Production Company Rank (Avg)	1	Positive Number	IMDb List Files		Native
14	Production Designer Rank (Avg)	1	Positive Number	IMDb List Files		Native
15	Producer Rank (Avg)	1	Positive Number	IMDb List Files		Native
16	Writer Rank (Avg)	1	Positive Number	IMDb List Files		Native
17	Golden Globe Nominee	1	Positive Integer	IMDb Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived
18	Golden Globe Win	1	Positive Integer	IMDb Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived
19	Metascore	1	Positive Integer	Metacritic Website	Schaible et al. (2015); Latif and Afzal (2016)	Derived
20	Opening Gross	9	1 - 9	BoxOfficeMojo Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived
21	Opening Theaters	1	Positive Integer	BoxOfficeMojo Website	Simonoff and Sparrow (2000); Sharda and Delen (2006); Latif and Afzal (2016)	Derived

Continued on next page

Table M.2 – continued from previous page

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
22	Oscar Nominee	1	Positive Integer	IMDb Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived
23	Oscar Win	1	Positive Integer	IMDb Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived

Table M.3: Summary of Input Attributes for The FilmAffinity Dataset

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
1	Actor Rank (Sum)	1	Positive Integer	IMDb List Files; FilmAffinity Website	Saraee et al. (2004); Asad et al. (2012)	Derived
2	Actress Rank (Sum)	1	Positive Integer	IMDb List Files; FilmAffinity Website	Saraee et al. (2004); Asad et al. (2012)	Derived
3	Budget	9	1 - 9	IMDb List Files	Saraee et al. (2004); Asad et al. (2012); Latif and Afzal (2016)	Derived
4	Cinematographer Rank (Avg)	1	Positive Integer	IMDb List Files; FilmAffinity Website		Derived
5	Competition	3	High, Medium, Low	IMDb List Files	Simonoff and Sparrow (2000); Sharda and Delen (2006)	Derived

Continued on next page

Table M.3 – continued from previous page

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
6	Composer Rank (Avg)	1	Positive Integer	IMDb List Files; FilmAffinity Website		Derived
7	Costume Designer Rank (Avg)	1	Positive Integer	IMDb List Files; FilmAffinity Website		Derived
8	Director Rank (Avg)	1	Positive Integer	IMDb List Files; FilmAffinity Website	Saraee et al. (2004); Asad et al. (2012)	Derived
9	Distributor Rank (Sum)	1	Positive Integer	IMDb List Files; FilmAffinity Website		Derived
10	Editor Rank (Avg)	1	Positive Integer	IMDb List Files; FilmAffinity Website		Derived
11	Genre	20	Action, adventure, thriller, biography, crime, drama, horror, comedy, fantasy, animation, mystery, music, war, documentary, romance, sci-fi, western, family, sport, short	IMDb List Files	Sharda and Delen (2006); Latif and Afzal (2016)	Derived

Continued on next page

Table M.3 – continued from previous page

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
12	MPAA	5	R, PG, PG-13, G, NR	IMDb List Files	Simonoff and Sparrow (2000); Sharda and Delen (2006); Latif and Afzal (2016)	Derived
13	Production Company Rank (Avg)	1	Positive Integer	IMDb List Files; FilmAffinity Website		Derived
14	Production Designer Rank (Avg)	1	Positive Integer	IMDb List Files; FilmAffinity Website		Derived
15	Producer Rank (Avg)	1	Positive Integer	IMDb List Files; FilmAffinity Website		Derived
16	Writer Rank (Avg)	1	Positive Integer	IMDb List Files; FilmAffinity Website		Derived
17	Golden Globe Nominee	1	Positive Integer	IMDb Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived
18	Golden Globe Win	1	Positive Integer	IMDb Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived
19	Metascore	1	Positive Integer	Metacritic Website	Schaible et al. (2015); Latif and Afzal (2016)	Derived

Continued on next page

Table M.3 – continued from previous page

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
20	Opening Gross	9	1 - 9	BoxOfficeMojo Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived
21	Opening Theaters	1	Positive Integer	BoxOfficeMojo Website	Simonoff and Sparrow (2000); Sharda and Delen (2006); Latif and Afzal (2016)	Derived
22	Oscar Nominee	1	Positive Integer	IMDb Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived
23	Oscar Win	1	Positive Integer	IMDb Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived

Table M.4: Summary of Input Attributes for The MovieMeter Dataset

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
1	Actor Rank (Sum)	1	Positive Integer	IMDb List Files; MovieMeter Website	Saraee et al. (2004); Asad et al. (2012)	Derived
2	Actress Rank (Sum)	1	Positive Integer	IMDb List Files; MovieMeter Website	Saraee et al. (2004); Asad et al. (2012)	Derived

Continued on next page

Table M.4 – continued from previous page

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
3	Budget	9	1 - 9	IMDb List Files	Saraee et al. (2004); Asad et al. (2012); Latif and Afzal (2016)	Derived
4	Cinematographer Rank (Avg)	1	Positive Integer	IMDb List Files; MovieMeter Website		Derived
5	Competition	3	High, Medium, Low	IMDb List Files	Simonoff and Sparrow (2000); Sharda and Delen (2006)	Derived
6	Composer Rank (Avg)	1	Positive Integer	IMDb List Files; MovieMeter Website		Derived
7	Costume Designer Rank (Avg)	1	Positive Integer	IMDb List Files; MovieMeter Website		Derived
8	Director Rank (Avg)	1	Positive Integer	IMDb List Files; MovieMeter Website	Saraee et al. (2004); Asad et al. (2012)	Derived
9	Distributor Rank (Sum)	1	Positive Integer	IMDb List Files; MovieMeter Website		Derived
10	Editor Rank (Avg)	1	Positive Integer	IMDb List Files; MovieMeter Website		Derived

Continued on next page

Table M.4 – continued from previous page

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
11	Genre	20	Action, adventure, thriller, biography, crime, drama, horror, comedy, fantasy, animation, mystery, music, war, documentary, romance, sci-fi, western, family, sport, short	IMDb List Files	Sharda and Delen (2006); Latif and Afzal (2016)	Derived
12	MPAA	5	R, PG, PG-13, G, NR	IMDb List Files	Simonoff and Sparrow (2000); Sharda and Delen (2006); Latif and Afzal (2016)	Derived
13	Production Company Rank (Avg)	1	Positive Integer	IMDb List Files; MovieMeter Website		Derived
14	Production Designer Rank (Avg)	1	Positive Integer	IMDb List Files; MovieMeter Website		Derived
15	Producer Rank (Avg)	1	Positive Integer	IMDb List Files; MovieMeter Website		Derived
16	Writer Rank (Avg)	1	Positive Integer	IMDb List Files; MovieMeter Website		Derived

Continued on next page

Table M.4 – continued from previous page

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
17	Golden Globe Nominee	1	Positive Integer	IMDb Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived
18	Golden Globe Win	1	Positive Integer	IMDb Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived
19	Metascore	1	Positive Integer	Metacritic Website	Schaible et al. (2015); Latif and Afzal (2016)	Derived
20	Opening Gross	9	1 - 9	BoxOfficeMojo Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived
21	Opening Theaters	1	Positive Integer	BoxOfficeMojo Website	Simonoff and Sparrow (2000); Sharda and Delen (2006); Latif and Afzal (2016)	Derived
22	Oscar Nominee	1	Positive Integer	IMDb Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived
23	Oscar Win	1	Positive Integer	IMDb Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived

Table M.5: Summary of Input Attributes for The RottenTomatoes Dataset

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
1	Actor Rank (Sum)	1	Positive Integer	IMDb List Files; RottenTomatoes Website	Saraee et al. (2004); Asad et al. (2012)	Derived
2	Actress Rank (Sum)	1	Positive Integer	IMDb List Files; RottenTomatoes Website	Saraee et al. (2004); Asad et al. (2012)	Derived
3	Budget	9	1 - 9	IMDb List Files	Saraee et al. (2004); Asad et al. (2012); Latif and Afzal (2016)	Derived
4	Cinematographer Rank (Avg)	1	Positive Integer	IMDb List Files; RottenTomatoes Website		Derived
5	Competition	3	High, Medium, Low	IMDb List Files	Simonoff and Sparrow (2000); Sharda and Delen (2006)	Derived
6	Composer Rank (Avg)	1	Positive Integer	IMDb List Files; RottenTomatoes Website		Derived
7	Costume Designer Rank (Avg)	1	Positive Integer	IMDb List Files; RottenTomatoes Website		Derived

Continued on next page

Table M.5 – continued from previous page

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
8	Director Rank (Avg)	1	Positive Integer	IMDb List Files; RottenTomatoes Website	Saraee et al. (2004); Asad et al. (2012)	Derived
9	Distributor Rank (Sum)	1	Positive Integer	IMDb List Files; RottenTomatoes Website		Derived
10	Editor Rank (Avg)	1	Positive Integer	IMDb List Files; RottenTomatoes Website		Derived
11	Genre	20	Action, adventure, thriller, biography, crime, drama, horror, comedy, fantasy, animation, mystery, music, war, documentary, romance, sci-fi, western, family, sport, short	IMDb List Files	Sharda and Delen (2006); Latif and Afzal (2016)	Derived
12	MPAA	5	R, PG, PG-13, G, NR	IMDb List Files	Simonoff and Sparrow (2000); Sharda and Delen (2006); Latif and Afzal (2016)	Derived
13	Production Company Rank (Avg)	1	Positive Integer	IMDb List Files; RottenTomatoes Website		Derived

Continued on next page

Table M.5 – continued from previous page

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
14	Production Designer Rank (Avg)	1	Positive Integer	IMDb List Files; RottenTomatoes Website		Derived
15	Producer Rank (Avg)	1	Positive Integer	IMDb List Files; RottenTomatoes Website		Derived
16	Writer Rank (Avg)	1	Positive Integer	IMDb List Files; RottenTomatoes Website		Derived
17	Golden Globe Nominee	1	Positive Integer	IMDb Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived
18	Golden Globe Win	1	Positive Integer	IMDb Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived
19	Metascore	1	Positive Integer	Metacritic Website	Schaible et al. (2015); Latif and Afzal (2016)	Derived
20	Opening Gross	9	1 - 9	BoxOfficeMojo Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived
21	Opening Theaters	1	Positive Integer	BoxOfficeMojo Website	Simonoff and Sparrow (2000); Sharda and Delen (2006); Latif and Afzal (2016)	Derived

Continued on next page

Table M.5 – continued from previous page

NO	Attribute Name	No of Values	Possible Values	Data Sources	References	Note
22	Oscar Nominee	1	Positive Integer	IMDb Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived
23	Oscar Win	1	Positive Integer	IMDb Website	Simonoff and Sparrow (2000); Latif and Afzal (2016)	Derived

Appendix N

A Full-Example of Instance of the Datasets

This appendix depicts a full-examples of instance of the following datasets: IMDb, IMDb++, FilmAffinity, MovieMeter, and RottenTomatoes.

Table N.1: Full Example of Hacksaw Ridge The Movie based on the IMDb Dataset

No	Attribute Name	Original Values	Discrete Values
1	MovieID	2941364	
2	DBPedia	http://dbpedia.org/resource/Hacksaw_Ridge	
3	IMDb	https://www.imdb.com/title/tt2119532/	
4	SumActorsRank	480.71	
5	SumActressRank	13.39	
6	Budget	40,000,000	5
7	AvgCinematographerRank	6.68	
8	Competition	Low Competition	1
9	AvgComposerRank	6.00	
10	AvgCostumeDesignerRank	6.17	
11	AvgDirectorRank	7.66	
12	SumDistributorRank	43.66	
13	AvgEditorRank	7.01	
14	Genre	Biography, Drama, History, War	5, 9, 12, 24
15	MPAA	NR	6
16	AvgProductionCompanyRank	6.81	
17	AvgProductionDesignerRank	6.36	
18	AvgProducerRank	6.58	
19	AvgWriterRank	7.31	
20	Rating	8.20	Excellent

Table N.2: Full Example of Hacksaw Ridge The Movie based on the IMDb++ Dataset

No	Attribute Name	Original Values	Discrete Values
1	MovieID	2941364	
2	DBPedia	http://dbpedia.org/resource/Hacksaw_Ridge	
3	IMDb	https://www.imdb.com/title/tt2119532/	
4	SumActorsRank	480.71	
5	SumActressRank	13.39	
6	Budget	40,000,000	5
7	AvgCinematographer Rank	6.68	
8	Competition	Low Competition	1
9	AvgComposerRank	6	
10	AvgCostumeDesigner Rank	6.17	
11	AvgDirectorRank	7.66	
12	SumDistributorRank	43.66	
13	AvgEditorRank	7.01	
14	Genre	Biography, Drama, History, War	5, 9, 12, 24
15	MPAA	NR	6
16	AvgProductionCompany Rank	6.81	
17	AvgProductionDesigner Rank	6.36	
18	AvgProducerRank	6.58	
19	AvgWriterRank	7.31	
20	GoldenGlobeNominee	3	
21	GoldenGlobeWon	0	
22	Metascore	71	
23	OpeningGross	15,190,758	3
24	OpeningTheaters	2886	
25	OscarNominee	4	
25	OscarWon	2	
26	Rating	8.20	Excellent

Table N.3: Full Example of Hacksaw Ridge The Movie based on the FilmAffinity Dataset

No	Attribute Name	Original Values	Discrete Values
1	MovieID	2941364	
2	DBPedia	http://dbpedia.org/resource/Hacksaw_Ridge	
3	FilmAffinity	https://www.filmaffinity.com/en/film550924.html	
4	SumActorsRank	436.67	
5	SumActressRank	12.72	
6	Budget	40,000,000	5

Continued on next page

Table N.3 – continued from previous page

No	Attribute Name	Original Values	Discrete Values
7	AvgCinematographer Rank	6.27	
8	Competition	Low Competition	1
9	AvgComposerRank	5.69	
10	AvgCostumeDesigner Rank	5.93	
11	AvgDirectorRank	7.10	
12	SumDistributorRank	42.00	
13	AvgEditorRank	6.67	
14	Genre	Biography, Drama, History, War	5, 9, 12, 24
15	MPAA	NR	6
16	AvgProductionCompany Rank	6.41	
17	AvgProductionDesigner Rank	6.03	
18	AvgProducerRank	6.20	
19	AvgWriterRank	6.73	
20	GoldenGlobeNominee	3	
21	GoldenGlobeWon	0	
22	Metascore	71	
23	OpeningGross	15,190,758	3
24	OpeningTheaters	2886	
25	OscarNominee	4	
25	OscarWon	2	
26	Rating	7.20	Average

Table N.4: Full Example of Hacksaw Ridge The Movie based on the MovieMeter Dataset

No	Attribute Name	Original Values	Discrete Values
1	MovieID	2941364	
2	DBPedia	http://dbpedia.org/resource/Hacksaw_Ridge	
3	MovieMeter	https://www.moviemeter.nl/film/235590	
4	SumActorsRank	445.29	
5	SumActressRank	12.67	
6	Budget	40,000,000	5
7	AvgCinematographer Rank	6.51	
8	Competition	Low Competition	1
9	AvgComposerRank	5.68	
10	AvgCostumeDesigner Rank	5.66	
11	AvgDirectorRank	7.06	
12	SumDistributorRank	41.70	
13	AvgEditorRank	6.66	

Continued on next page

Table N.4 – continued from previous page

No	Attribute Name	Original Values	Discrete Values
14	Genre	Biography, Drama, History, War	5, 9, 12, 24
15	MPAA	NR	6
16	AvgProductionCompany Rank	6.42	
17	AvgProductionDesigner Rank	6.02	
18	AvgProducerRank	6.13	
19	AvgWriterRank	6.77	
20	GoldenGlobeNominee	3	
21	GoldenGlobeWon	0	
22	Metascore	71	
23	OpeningGross	15,190,758	3
24	OpeningTheaters	2886	
25	OscarNominee	4	
25	OscarWon	2	
26	Rating	7.44	Average

Table N.5: Full Example of Hacksaw Ridge The Movie based on the RottenTomatoes Dataset

No	Attribute Name	Original Values	Discrete Values
1	MovieID	2941364	
2	DBPedia	http://dbpedia.org/resource/Hacksaw_Ridge	
3	RottenTomatoes	https://www.rottentomatoes.com/m/hacksaw_ridge	
4	SumActorsRank	499.68	
5	SumActressRank	13.74	
6	Budget	40,000,000	5
7	AvgCinematographer Rank	6.84	
8	Competition	Low Competition	1
9	AvgComposerRank	6.20	
10	AvgCostumeDesigner Rank	6.35	
11	AvgDirectorRank	7.54	
12	SumDistributorRank	45.44	
13	AvgEditorRank	6.98	
14	Genre	Biography, Drama, History, War	5, 9, 12, 24
15	MPAA	NR	6
16	AvgProductionCompany Rank	6.93	
17	AvgProductionDesigner Rank	6.50	
18	AvgProducerRank	6.73	
19	AvgWriterRank	7.53	
20	GoldenGlobeNominee	3	

Continued on next page

Table N.5 – continued from previous page

No	Attribute Name	Original Values	Discrete Values
21	GoldenGlobeWon	0	
22	Metascore	71	
23	OpeningGross	15,190,758	3
24	OpeningTheaters	2886	
25	OscarNominee	4	
25	OscarWon	2	
26	Rating	8.60	Excellent

Bibliography

Rif overview (second edition). Working group note, 2 2013.

Andrew A. Adams and Rachel. McCrindle. *Pandora's Box: Social and Professional Issues of the Information Age*. John Wiley & Sons, 2008.

Mehreen Ahmed, Maham Jahangir, Hammad Afzal, Awais Majeed, and Imran Siddiqi. Using crowd-source based features from social media and conventional features to predict the movies popularity. In *SmartCity*, pages 273–278. IEEE Computer Society, 2015. ISBN 978-1-5090-1893-2.

Suad Aldarra and Emir Muñoz. A linked data-based decision tree classifier to review movies. In Johanna Völker, Heiko Paulheim, Jens Lehmann, and Vojtech Svátek, editors, *KNOW@LOD*, volume 1365 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46:175–185, 1992.

Matthew N. Anyanwu and Sajjan G. Shiva. Comparative analysis of serial decision tree classification algorithms. *International Journal of Computer Science and Security, (IJCSS)*, 3(3), 2009.

Krushikanth R. Apala, Merin Jose, Supreme Motnam, Chien-Chung Chan, Kathy J. Liszka, and Federico de Gregorio. Prediction of movies box office performance using social media. In Jon G. Rokne and Christos Faloutsos, editors, *ASONAM*, pages 1209–1214. ACM, 2013. ISBN 978-1-4503-2240-9.

Sylvain Arlot and Alain Celisse. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4:40–79, 2010.

Khalid Ibnal Asad, Tanvir Ahmed, and Md. Saiedur Rahman. Movie popularity classification based on inherent movie attributes using c4.5,part and correlation coefficient. *CoRR*, abs/1209.6070, 2012.

Rodrigo Coelho Barros, Márcio Porto Basgalupp, André Carlos Ponce Leon Ferreira de Carvalho, and Alex Alves Freitas. A survey of evolutionary algorithms for decision-tree induction. *IEEE Trans. Systems, Man, and Cybernetics, Part C*, 42(3):291–312, 2012.

Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2):105–139, 1999.

Wouter Beek, Laurens Rietveld, Hamid R. Bazoobandi, Jan Wielemaker, and Stefan Schlobach. Lod laundromat: A uniform way of publishing other people's dirty data. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig A. Knoblock, Denny Vrandecic, Paul T. Groth, Natasha F. Noy, Krzysztof Janowicz, and Carole A. Goble, editors, *Semantic Web Conference (1)*, volume 8796 of *Lecture Notes in Computer Science*, pages 213–228. Springer, 2014. ISBN 978-3-319-11963-2.

David A. Bell and Hui Wang. A formalism for relevance and its application in feature subset selection. *Machine Learning*, 41(2):175–195, 2000.

Richard E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *NIPS*, pages 2546–2554, 2011.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

P. Berkhin. A survey of clustering data mining techniques. *Grouping Multidimensional Data*, pages 25–71, 2006.

Tim Berners-Lee. Linked data. World wide web design issues, July 2006.

Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

Michael J. A. Berry and Gordon S. Linoff. *Mastering Data Mining.* Wiley, 2000.

Michael W. Berry, editor. *Survey of Text Mining.* Springer, 2003.

Nitin Bhatia and Vandana. Survey of nearest neighbor techniques. *CoRR*, abs/1007.0085, 2010.

I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM Trans. Knowl. Discov. Data*, 1(1), March 2007.

Chris Bizer and Andy Seaborne. D2RQ treating non-RDF databases as virtual rdf graphs. In *ISWC*, 2004.

Christian Bizer and Richard Cyganiak. D2r server – publishing relational databases on the semantic web. Poster at the 5th International Semantic Web Conference (ISWC2006), 2006.

Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

Matt Bogard, Tuesdi Helbig, Gina Huff, and Chris James. A comparison of empirical models for predicting student retention. White paper, Office of Institutional Research, Western Kentucky University, 2010.

L. Breiman, J Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Pacific Grove, 1984.

Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

Michel Camilleri and Filippo Neri. Parameter optimization in decision tree learning by using simple genetic algorithms. *WSEAS TRANSACTIONS on SYSTEMS*, 13: 582–591, 2014.

Fernando Canet, Miguel Ángel Valero, and Lluís Codina. Quantitative approaches for evaluating the influence of films using the imdb database. pages 151–172, 2016.

Doina Caragea, Jun Zhang, Jie Bao, Jyotishman Pathak, and Vasant Honavar. Algorithms and software for collaborative discovery from autonomous, semantically heterogeneous, distributed information sources. In Achim G. Hoffmann, Hiroshi Motoda, and Tobias Scheffer, editors, *Discovery Science*, volume 3735 of *Lecture Notes in Computer Science*, page 14. Springer, 2005. ISBN 3-540-29230-6.

Soumen Chakrabarti. *Mining the Web: Analysis of Hypertext and Semi Structured Data*. Morgan Kaufmann, August 2002. ISBN 1558607544.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Puvadon Changkaew and Rachada Kongkachandra. Automatic movie rating using visual and linguistic information. In *First International Conference on Integrated Intelligent Computing*. IEEE, 2010. ISBN 978-0-7695-4152-5/10.

Dunren Che, Mejdl S. Safran, and Zhiyong Peng. From big data to big data mining: Challenges, issues, and opportunities. In Bonghee Hong, Xiaofeng Meng, Lei Chen, Werner Winiwarter, and Wei Song, editors, *DASFAA Workshops*, volume 7827 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2013. ISBN 978-3-642-40269-2.

Li-Fei Chen, Chao-Ton Su, and Kun-Huang Chen. An improved particle swarm optimization for feature selection. *Intell. Data Anal.*, 16(2):167–182, 2012.

You Chen, Yang Li, Xueqi Cheng, and Li Guo. Survey and taxonomy of feature selection algorithms in intrusion detection system. In Helger Lipmaa, Moti Yung, and Dongdai Lin, editors, *Inscrypt*, volume 4318 of *Lecture Notes in Computer Science*, pages 153–167. Springer, 2006. ISBN 3-540-49608-4.

Sung Jin Cho and Mark A. Hermsmeier. Genetic algorithm guided selection: Variable selection and subset selection. *Journal of Chemical Information and Computer Sciences*, 42(4):927–936, 2002.

Peter Clark and Tim Niblett. The cn2 induction algorithm. *Machine Learning*, 3: 261–283, 1989.

W. Cohen. Fast effective rule induction. In *Machine Learning: Proceedings of the 12th International Conference*, 1995.

D.F. Cook, C. T. Ragsdale, and R. L. Major. Combining a neural network with a genetic algorithm for process parameter optimization. *Engineering Applications of Articial Intelligence*, 13:391–396, 2000.

T.M Cover and P.E. Hart. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, IT-13:21–27, 1967.

Padraig Cunningham and Sarah Jane Delany. k-nearest neighbour classifiers. Technical report ucd-csi-2007-4, University College Dublin and Dublin Institute of Technology, 2007.

Sanmay Das. Filters, wrappers and a boosting-based hybrid for feature selection. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, *ICML*, pages 74–81. Morgan Kaufmann, 2001. ISBN 1-55860-778-1.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *ICML*, 2006.

Renato Cordeiro de Amorim. Computational methods of feature selection, huan liu, hiroshi motoda, crc press, boca raton, fl (2007), 440 pp, isbn 978-1-58488-878-9. *Inf. Process. Manage.*, 45(4):490–493, 2009.

Ramon López de Mántaras. A distance-based attribute selection measure for decision tree induction. *Machine Learning*, 6:81–92, 1991.

Dursun Delen. A comparative analysis of machine learning techniques for student retention management. *Decision Support Systems*, 49(4):498–506, 2010.

Dursun Delen and Ramesh SHARDA. Predicting the financial success of hollywood movies using an information fusion approach. *Industrial Engineering Journal*, 21(1): 30–37, 2010.

Dursun Delen, Ramesh Sharda, and Prajeeb Kumar. Movie forecast guru: A web-based dss for hollywood managers. *Decision Support Systems*, 43(4):1151–1170, 2007.

Robin Dhamankar, Yoonkyong Lee, AnHai Doan, Alon Y. Halevy, and Pedro Domingos. imap: Discovering complex mappings between database schemas. In Gerhard Weikum, Arnd Christian König, and Stefan Deßloch, editors, *SIGMOD Conference*, pages 383–394. ACM, 2004. ISBN 1-58113-859-8.

Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

Anastasia Dimou, Ruben Verborgh, Miel Vander Sande, Erik Mannens, and Rik Van de Walle. Machine-interpretable dataset and service descriptions for heterogeneous data access and retrieval. pages 145–152. ACM, 2015. ISBN 978-1-4503-3462-4.

Microsoft Docs. Data partitioning guidance. https://docs.microsoft.com/en-us/azure/architecture/best-practices/data-partitioning, July 2016. 2016-07-13.

Eulanda Miranda dos Santos, Robert Sabourin, and Patrick Maupin. Overfitting cautious selection of classifier ensembles with genetic algorithms. *Information Fusion*, 10 (2):150–162, 2009.

Yadin. Dudai. *The Neurobiology of Memory*. Oxford University Press, Oxford, 1989.

Ramón Díaz-Uriarte and Sara Alvarez de Andrés. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7:3, 2006.

R C Eberhart and J Kennedy. A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, 1995.

A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

J.F. Elder. The generalization paradox of ensembles. *Journal of Computational and Graphical Statistics*, 12(4):853–864, 2003.

Jeffrey Ericson and Jesse Grodman. A predictor for movie success. 2013. CS229, Stanford University.

Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandecic. Introducing wikidata to the linked data web. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig A. Knoblock, Denny Vrandecic, Paul T. Groth, Natasha F. Noy, Krzysztof Janowicz, and Carole A. Goble, editors, *Semantic Web Conference (1)*, volume 8796 of *Lecture Notes in Computer Science*, pages 50–65. Springer, 2014. ISBN 978-3-319-11963-2.

Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery: An overview. In *Advances in Knowledge Discovery and Data Mining*, pages 1–34. 1996.

Imola Fodor. A survey of dimension reduction techniques. Technical report, 2002.

Y. Freund and R. E. Schapire. A Decision Theoretic Generalization of On-Line Learning and an Application to Boosting. In Paul M. B. Vitányi, editor, *Second European Conference on Computational Learning Theory (EuroCOLT-95)*, pages 23–37, 1995.

Frauke Friedrichs and Christian Igel. Evolutionary tuning of multiple svm parameters. *Neurocomputing*, 64:107–117, 2004. Trends in Neurocomputing: 12th European Symposium on Artificial Neural Networks 2004.

Mohamed Medhat Gaber and Mohamed Bahy Bader-El-Den. Optimisation of ensemble classifiers using genetic algorithm. In Manuel Graña, Carlos Toro, Jorge Posada, Robert J. Howlett, and Lakhmi C. Jain, editors, *KES*, volume 243 of *Frontiers in Artificial Intelligence and Applications*, pages 39–48. IOS Press, 2012. ISBN 978-1-61499-104-5.

Fabien Gandon and Guus Schreiber. RDF 1.1 XML syntax. W3C recommendation, W3C, February 2014.

Pierre Geurts, Alexandre Irrthum, and Louis Wehenkel. Supervised learning with decision tree-based methods in computational and systems biology. *Molecular BioSystems*, 5:1593–1605, dec 2009.

M. Ghiassi, David Lio, and Brian Moon. Pre-production forecasting of movie revenues with a dynamic artificial neural network. *Expert Syst. Appl.*, 42(6):3176–3193, 2015.

M. Ghiassi and H. Saidane. A dynamic architecture for artificial neural networks. *Neurocomputing*, 63:397–413, 2005.

C. Lee Giles, Yang Sun, and Isaac G. Councill. Measuring the web crawler ethics. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *WWW*, pages 1101–1102. ACM, 2010. ISBN 978-1-60558-799-8.

David E. Goldberg. Genetic and evolutionary algorithms come of age. *Commun. ACM*, 37(3):113–119, 1994.

Peter Grabusts. The choice of metrics for clustering algorithms. volume 11 of *Proceedings of the 8th International Scientific and Practical Conference*, 2011.

Magdalena Graczyk, Tadeusz Lasota, Bogdan Trawinski, and Krzysztof Trawinski. Comparison of bagging, boosting and stacking ensembles applied to real estate appraisal. In Ngoc Thanh Nguyen, Manh Thanh Le, and Jerzy Swiatek, editors, *ACIIDS (2)*, volume 5991 of *Lecture Notes in Computer Science*, pages 340–350. Springer, 2010. ISBN 978-3-642-12100-5.

Thomas R. Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*, 43:907–928, 1993.

Jerzy W. Grzymala-Busse and Ming Hu. A comparison of several approaches to missing attribute values in data mining. In Wojciech Ziarko and Y. Y. Yao, editors, *Rough Sets and Current Trends in Computing*, volume 2005 of *Lecture Notes in Computer Science*, pages 378–385. Springer, 2000. ISBN 3-540-43074-1.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003. ISSN 1532-4435.

Mark A. Hall and Geoffrey Holmes. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Trans. Knowl. Data Eng.*, 15(6):1437–1447, 2003.

Wendy Hall and Thanassis Tiropanis. Web evolution and web science. *Computer Networks*, 56(18):3859–3865, 2012.

Han and Kamber. *Data Mining. Concepts and Techniques.* Morgan Kaufmann, San Francisco, LA, 2001.

Jianwei Han and Kevin Chen-Chuan Chang. Data mining for web intelligence. *IEEE Computer*, 35(11):64–70, 2002.

David J. Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining.* MIT Press, 2001. ISBN 9780262332521.

O. Hassanzadeh and M. P. Consens. Linked Movie Data Base. In *Proceedings of the WWW2009 workshop on Linked Data on the Web (LDOW2009)*, April 2009.

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: Data mining, inference, and prediction.* Springer, New York, 2001.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer, New York, 2009.

Jim Hendler and Tim Berners-Lee. From the semantic web to social machines: A research challenge for ai on the world wide web. *Artif. Intell.*, 174(2):156–161, 2010.

Lawrence B. Holder, Ingrid Russell, Zdravko Markov, Anthony G. Pipe, and Brian Carse. Current and future trends in feature selection and extraction for classification problems. *IJPRAI*, 19(2):133–142, 2005.

John H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

Martin Hric, Michal Chmulik, and Roman Jarina. Model parameters selection for svm classification using particle swarm optimization. In *Proceedings of 21st International Conference Radioelektronika 2011*. IEEE, 2011. ISBN 978-1-61284-324-7.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003.

Ping-Yu Hsu, Yuan-Hong Shen, and Xiang-An Xie. Predicting movies user ratings with imdb attributes. In Duoqian Miao, Witold Pedrycz, Dominik Slezak, Georg Peters, Qinghua Hu, and Ruizhi Wang, editors, *RSKT*, volume 8818 of *Lecture Notes in Computer Science*, pages 444–453. Springer, 2014. ISBN 978-3-319-11739-3.

Cheng-Lung Huang and Jian-Fan Dun. A distributed pso-svm hybrid system with feature selection and parameter optimization. *Appl. Soft Comput.*, 8(4):1381–1391, 2008.

Mohammed J. Islam, Q. M. Jonathan Wu, Majid Ahmadi, and Maher A. Sid-Ahmed. Investigating the performance of naive- bayes classifiers and k- nearest neighbor classifiers. *JCIT*, 5(2):133–137, 2010.

Roger M. Jarvis and Royston Goodacre. Genetic algorithm optimization for preprocessing and variable selection of spectroscopic data. *Bioinformatics*, 21(7):860–868, 2005.

Liangxiao Jiang, Zhihua Cai, Dianhong Wang, and Siwei Jiang. Survey of improving k-nearest-neighbor for classification. In J. Lei, editor, *FSKD (1)*, pages 679–683. IEEE Computer Society, 2007. ISBN 0-7695-2874-0.

Dah-Jing Jwo and Shun-Chieh Chang. Particle swarm optimization for gps navigation kalman filter adaptation. In *Aircraft Engineering and Aerospace Technology*, volume 81, pages 343–352. Emerald Insight, 2009.

S. Kabinsingha, S. Chindasorn, and C. Chantrapornchai. A movie rating approach and application based on data mining. volume 2, July 2012.

Murat Kantarcioglu and Chris Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Trans. Knowl. Data Eng.*, 16(9): 1026–1037, 2004.

S. Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computation*, 15(7):1667–1689, 2003.

James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

Taegu Kim, Jungsik Hong, and Pilsung Kang. Box office forecasting using machine learning algorithms based on sns data. *International Journal of Forecasting*, 31(31): 364–390, 2015.

Mei Kobayashi and Koichi Takeda. Information retrieval on the web. *ACM Comput. Surv.*, 32(2):144–173, 2000. ISSN 0360-0300.

Ron Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *IJCAI*, pages 1137–1145, 1995.

Ron Kohavi and George Harrison John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1–2):273–323, 1997. In Special Issue on Relevance.

Martijn Koster. Guidelines for robots writers. http://www.robotstxt.org/guidelines.html., 1993.

Martijn Koster. A standard for robot exclusion. http://www.robotstxt.org/orig.html., 1996.

Sotiris B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica (Slovenia)*, 31(3):249–268, 2007.

Vijay Kotu and Bala Deshpande. *Predictive Analytics and Data Mining: Concepts and Practice with RapidMiner.* Morgan Kaufmann, Amsterdam, 2015. ISBN 978-0-12-801460-8.

Neeraj Koul, Ngot Bui, and Vasant Honavar. Scalable, updatable predictive models for sequence data. In Taesung Park, Stephen Kwok-Wing Tsui, Luonan Chen, Michael K. Ng, Limsoon Wong, and Xiaohua Hu, editors, *BIBM*, pages 681–685. IEEE Computer Society, 2010. ISBN 978-1-4244-8306-8.

Neeraj Koul, Cornelia Caragea, Vasant Honavar, Vikas Bahirwani, and Doina Caragea. Learning classifiers from large databases using statistical queries. In *Web Intelligence*, pages 923–926. IEEE Computer Society, 2008. ISBN 978-0-7695-3496-1.

Zlatko J. Kovačić. Early prediction of student success: Mining students enrolment data. In *Proceedings of Informing Science & IT Education Conference (InSITE) 2010*, 2010.

K. Krishna and M. Narasimha Murty. Genetic k-means algorithm. 1999.

P. Langley. Selection of relevant features in machine learning, 1994.

Hugo Larochelle, Dumitru Erhan, Aaron C. Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In Zoubin Ghahramani, editor, *ICML*, volume 227 of *ACM International Conference Proceeding Series*, pages 473–480. ACM, 2007. ISBN 978-1-59593-793-3.

Michael T. Lash and Kang Zhao. Early predictions of movie success: The who, what, and when of profitability. *Journal of Management Information Systems*, 33(3):874–903, 2016.

Muhammad Hassan Latif and Hammad Afzal. Prediction of movies popularity using machine learning techniques. *IJCSNS International Journal of Computer Science and Network Security.*, 16(8), 2016.

Kun Chang Lee and Heeryon Cho. Performance of ensemble classifier for location prediction task: Emphasis on markov blanket perspective. *International Journal of u- and e- Service, Science and Technology*, 3(3), 9 2010.

Kyung Jae Lee and Woojin Chang. Bayesian belief network for box-office performance: A case study on korean movies. *Expert Systems with Applications*, 36(1):280 – 291, 2009. ISSN 0957-4174.

Sovan Lek, Stephane Aulagnier, Marc Delacoste, Philippe Baran, Ioannis Dimopoulos, Jacques Lauga, and Stéphane Aulagnier. Application of neural networks to modelling nonlinear relationships in ecology. *Ecological Modelling*, 90:39–52, 1996.

Victor R. Lesser, Bryan Horling, Frank Klassner, Anita Raja, Thomas Wagner, and Xiaoqin Zhang. Big: An agent for resource-bounded information gathering and decision making. *Artif. Intell.*, 118(1-2):197–244, 2000.

Frank Hung-Fat Leung, Hak-Keung Lam, Sai-Ho Ling, and Peter Kwong-Shun Tam. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Trans. Neural Networks*, 14(1):79–88, 2003.

Frank Leymann. Web services: Distributed applications without limits. In Gerhard Weikum, Harald Schöning, and Erhard Rahm, editors, *BTW*, volume 26 of *LNI*, pages 2–23. GI, 2003. ISBN 3-88579-355-5.

Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *CoRR*, abs/1601.07996, 2016.

Peng Li and Seiji Yamada. A movie recommender system based on inductive learning. In *Conference on Cybernetics and Intelligent Systems Singapore, 1-3 December, 2004*. IEEE, 2004.

Shih-Wei Lin, Kuo-Ching Ying, Shih-Chieh Chen, and Zne-Jung Lee. Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert Syst. Appl.*, 35(4):1817–1824, 2008.

Bin Liu, Shu-Gui Cao, and Wu He. Distributed data mining for e-business. *Information Technology and Management*, 12(2):67–79, 2011.

Chien-Liang Liu, Wen-Hoar Hsaio, Chia-Hoang Lee, Gen-Chi Lu, and Emery Jou. Movie rating and review summarization in mobile environment. *IEEE Trans. Systems, Man, and Cybernetics, Part C*, 42(3):397–407, 2012.

Huan Liu and Hiroshi Motada. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, 1998.

Huan Liu, Hiroshi Motoda, Rudy Setiono, and Zheng Zhao. Feature selection: An ever evolving frontier in data mining. In Huan Liu, Hiroshi Motoda, Rudy Setiono, and Zheng Zhao, editors, *FSDM*, volume 10 of *JMLR Proceedings*, pages 4–13. JMLR.org, 2010.

Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. Knowl. Data Eng.*, 17(4):491–502, 2005.

Rahul Malhotra, Narinder Singh, and Yaduvir Singh. Genetic algorithms: Concepts, design for optimization of process controllers. *Computer and Information Science*, 4 (2):39–54, 2011.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

Frank Manola and Eric Miller. RDF primer, W3C recommendation. Technical report, W3C, 2004.

Mladen Marovic, Marko Mihokovic, Mladen Miksa, Sinisa Pribil, and Alan Tus. Automatic movie ratings prediction using machine learning. In *MIPRO*, pages 1640–1645. IEEE, 2011. ISBN 978-1-4577-0996-8.

Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language overview. W3C recommendation, World Wide Web Consortium, February 2004.

David A. Medler. A brief history of connectionism. *Neural Computing Surveys*, 1:61–101, 1998.

Tahar Mehenni and Abdelouahab Moussaoui. Data mining from multiple heterogeneous relational databases using decision tree classification. *Pattern Recognition Letters*, 33 (13):1768–1775, 2012.

M Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1998.

Samuel Moore, Daniel D'Addario, James Kurinskas, and Gary M. Weiss. Are decision trees always greener on the open (source) side of the fence? In Robert Stahlbock, Sven F. Crone, and Stefan Lessmann, editors, *DMIN*, pages 185–188. CSREA Press, 2009. ISBN 1-60132-099-X.

Sreerama K. Murthy, Simon Kasif, Steven Salzberg, and Richard Beigel. Oc1: A randomized induction of oblique decision trees. In Richard Fikes and Wendy G. Lehnert, editors, *AAAI*, pages 322–327. AAAI Press / The MIT Press, 1993. ISBN 0-262-51071-5.

Filippo Neri and Michels Camilleri. An algorithmic approach to parameter selection in machine learning using meta-optimization techniques. In *Proceedings of the 5th International Conference on Circuits, Systems, Control, Signals (CSCS '14)*, 2014.

VR. Nithin, M. Pranav, Sarath. Babu PB, and A. Lijiya. Predicting movie success based on imdb data. volume 03, pages 365–368, June 2014.

Zoran Obradovic and Slobodan Vucetic. Challenges in Scientific Data Mining: Heterogeneous, Biased, and Large Samples. Technical report, Temple University, Center for Information Science and Technology, 2004.

Hari Omand and Alok Kumar Gupta. Feature selection and decision tree: A combinational approach for intrusion detection. In Biju Issac and Nauman Israr, editors, *Case Studies in Secure Computing Achievements and Trends*, chapter 2, page 27–48. CRC Press, 2014.

David W. Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *J. Artif. Intell. Res. (JAIR)*, 11:169–198, 1999.

Fernando E. B. Otero, Alex Alves Freitas, and Colin G. Johnson. Inducing decision trees with an ant colony optimization algorithm. *Appl. Soft Comput.*, 12(11):3615–3626, 2012.

James R. Otto, James H. Cook, and Q. B. Chung. Extensible markup language and knowledge management. *J. Knowledge Management*, 5(3):278–285, 2001.

Elias Pampalk, Arthur Flexer, and Gerhard Widmer. Improvements of audio-based music similarity and genre classificaton. In *ISMIR*, pages 628–633, 2005.

G Pant, P Srinivasan, and F Menczer. Crawling the Web. In M Levene and A Poulovassilis, editors, *Web Dynamics*. Springer, 2004.

Matteo Pardo and Giorgio Sberveglieri. Classification of electronic nose data with support vector machines. *Sensors and Actuators B: Chemical*, 107(2):730–737, 2005. ISSN 0925-4005.

Heiko Paulheim, Petar Ristoski, Evgeny Mitichkin, and Christian Bizer. Data mining with background knowledge from the web. 2014.

Erick C. Paz. A survey of parallel genetic algorithms. *Calculateurs Paralleles, Reseaux et Systems Repartis*, 10(2):141–171, 1998.

Verónika Peralta. Extraction and integration of movielens and imdb data. Technical report, Laboratoire PRiSM, Université de Versailles, 45, avenue des Etats-Unis 78035 Versailles Cedex FRANCE, 7 2007a.

Verónika Peralta. Matching of movielens and imdb movie titles. Technical report, Laboratoire PRiSM, Université de Versailles, 45, avenue des Etats-Unis 78035 Versailles Cedex FRANCE, 3 2007b.

Leif E. Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, 2006.

Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. W3C Recommendation, 2008.

Nahid Quader, Md. Osman Gani, and Dipankar Chaki. Performance evaluation of seven machine learning classification techniques for movie box office success prediction. In *2017 3rd International Conference on Electrical Information and Communication Technology (EICT), 7-9 December 2017, Khulna, Bangladesh*. IEEE, 2017a.

Nahid Quader, Md. Osman Gani, Dipankar Chaki, and Md. Haider Ali. A machine learning approach to predict movie box-office success. In *In 2017 20th International Conference of Computer and Information Technology (ICCIT) , Dhaka, Bangladesh*. IEEE, 2017b.

J. R. Quinlan. Learning with continuous classes. pages 343–348. World Scientific, 1992.

J.R. Quinlan. Induction of decision trees. *Journal of Machine Learning*, 1:81–106, 1986.

Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001. ISSN 1066-8888.

Yves Raimond, Christopher Sutton, and Mark Sandler. Automatic interlinking of music datasets on the semantic web, 2008.

Jaime Reinoso, Adrian Silvescu, Doina Caragea, Jyotishman Pathak, and Vasant Honavar. Information extraction and integration from heterogeneous, distributed, autonomous information sources : A federated ontology-driven query-centric approach. In *IRI*, pages 183–191. IEEE Systems, Man, and Cybernetics Society, 2003.

Travis Ginmu Rhee and Farhana H. Zulkernine. Predicting movie box office profitability: A neural network approach. In *ICMLA*, pages 665–670. IEEE, 2016. ISBN 978-1-5090-6167-9.

Petar Ristoski and Heiko Paulheim. Semantic web in data mining and knowledge discovery: A comprehensive survey. *Web Semantics: Science, Services and Agents on the World Wide Web*, pages –, 2016. ISSN 1570-8268.

Richard J. Roiger and Michael W. Geatz. *Data Mining: A Tutorial-Based Primer*. Addison Wesley, 2003.

Lior Rokach. Ensemble-based classifiers. *Artif. Intell. Rev.*, 33(1-2):1–39, 2010.

F. Rosenblatt. *Principles of Neurodynamics*. Spartan, New York, 1962.

André L. D. Rossi, Luis Debiaso, and André Carlos Ponce Leon Ferreira de Carvalho. Bio-inspired optimization techniques for svm parameter tuning. In Marley Maria Bernardes Rebuzzi Vellasco, Marcílio Carlos Pereira de Souto, and Jés Jesus Fiais Cerqueira, editors, *SBRN*, pages 57–62. IEEE Computer Society, 2008. ISBN 978-0-7695-3361-2.

Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.

Satya S. Sahoo, Wolfgang Halb, Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau Jr, Sören Auer, Juan Sequeda, and Ahmed Ezzat. A survey of current approaches for mapping of relational databases to rdf, 01 2009.

Rizauddin Saian and Ku Ruhana Ku-Mahamud. Hybrid ant colony optimization and simulated annealing for rule induction. In David Al-Dabass, Alessandra Orsoni, Athanasios A. Pantelous, Gregorio Romero, and Jesús Félez, editors, *EMS*, pages 70–75. IEEE, 2011. ISBN 978-1-4673-0060-5.

Mohamad Saraee, S White, and J Eccleston. A data mining approach to analysis and prediction of movie ratings. In A Zanasi, C.A Brebbia, and N.F.F Ebecken, editors, *Data Mining V. Data Mining, Text Mining and their Business Applications*, page 343–352. WIT Press, 2004.

T. Hitendra Sarma, P. Viswanath, D. Sai Koti Reddy, and S. Sri Raghava. An improvement to k-nearest neighbor classifier. *CoRR*, abs/1301.6324, 2013.

Johann Schaible, Zeljko Carevic, Oliver Hopt, and Benjamin Zapilko. Utilizing the open movie database api for predicting the review class of movies. In Johanna Völker, Heiko Paulheim, Jens Lehmann, and Vojtech Svátek, editors, *KNOW@LOD*, volume 1365 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

Michael A. Schuh, Rafal A. Angryk, and John W. Sheppard. Evolving kernel functions with particle swarms and genetic programming. In G. Michael Youngblood and Philip M. McCarthy, editors, *FLAIRS Conference*. AAAI Press, 2012. ISBN 978-1-57735-558-8.

Ayan Seal, Suranjan Ganguly, Debotosh Bhattacharjee, Mita Nasipuri, and Consuelo Gonzalo-Martín. Feature selection using particle swarm optimization for thermal face recognition. In Rituparna Chaki, Khalid Saeed, Sankhayan Choudhury, and Nabendu Chaki, editors, *ACSS (1)*, pages 25–35. Springer, 2014. ISBN 978-81-322-1985-9.

Laveena Sehgal, Neeraj Mohan, and Dr. Parvinder S. Sandhu. Quality prediction of function based software using decision tree approach. In *International Conference on Computer Engineering and Multimedia Technologies (ICCEMT'2012) September 8-9, 2012 Bangkok (Thailand)*, 2012.

Gabriel Serme, Anderson Santana de Oliveira, Julien Massiera, and Yves Roudier. Enabling message security for restful services. In Carole A. Goble, Peter P. Chen, and Jia Zhang, editors, *ICWS*, pages 114–121. IEEE Computer Society, 2012. ISBN 978-1-4673-2131-0.

Ramesh Sharda and Dursun Delen. Predicting box-office success of motion pictures with neural networks. *Expert Syst. Appl.*, 30(2):243–254, 2006.

Yuhui Shi and Russell C. Eberhart. A Modified Particle Swarm Optimizer. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 69–73, Washington, DC, USA, May 1998. IEEE Computer Society.

Yao Shijia, Zhu Liuzhang, and Zhu Ming. Performance evaluation of seven machine learning classification techniques for movie box office success prediction. In *2010 3rd International Conference on Computer Science and Information Technology , 9-11 July 2010, Chengdu, China*. IEEE, 2010.

Jeffrey S. Simonoff and Ilana R. Sparrow. Predicting movie grosses: Winners and losers, blockbusters and sleepers. *Chance.*, 13(3):15–24, 2000.

Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manage.*, 45(4):427–437, 2009.

J. M. Steppe and K. W. Bauer Jr. Feature saliency measures. *Computers & Mathematics with Applications*, 33(8):109–126, 1997.

Gerd Stumme, Andreas Hotho, and Bettina Berendt. Semantic web mining - state of the art and future directions. *Journal of Web Semantics*, 4(2):124–143, 2006. ISSN 1570-8268.

Abdulhamit Subasi. Classification of emg signals using pso optimized svm for diagnosis of neuromuscular disorders. *Comp. in Bio. and Med.*, 43(5):576–586, 2013.

Vojtech Svátek, Jindrich Mynarz, and Heiko Paulheim. The linked data mining challenge 2014: Results and experiences. In Johanna Völker, Heiko Paulheim, Jens Lehmann, Harald Sack, and Vojtech Svátek, editors, *KNOW@LOD*, volume 1243 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, 2013.

James Taylor. *Decision Management Systems: A Practical Guide to Using Business Rules and Predictive Analytics*. IBM Press, 2011.

M. Thelwall and D. Stuart. Web crawling ethics revisited: Cost, privacy, and denial of service. *Journal of the American Society for Information Science and Technology*, 57 (13):1771–1779, 2006.

O'Reilly Tim. What is web 2.0? design patterns and business models for the next generation of software. 2005.

Anthony S. J. Tjiong and Sildomar T. Monteiro. Feature selection with pso and kernel methods for hyperspectral classification. In *IEEE Congress on Evolutionary Computation*, pages 1762–1769. IEEE, 2011. ISBN 978-1-4244-7834-7.

Juan Trujillo and Alejandro Maté. Business intelligence 2.0: A general overview. In Marie-Aude Aufaure and Esteban Zimányi, editors, *eBISS*, volume 96 of *Lecture Notes in Business Information Processing*, pages 98–116. Springer, 2011. ISBN 978-3-642-27357-5.

Kimitaka Tsutsumi, Kazutaka Shimada, and Tsutomu Endo. Movie review classification based on a multiple classifier. In Hee-Rahk Chae, Jae-Woong Choe, Jong Sup Jun, Youngchul Jun, and Eun-Jung Yoo, editors, *PACLIC*. The Korean Society for Language and Information / ACL, 2007.

Efraim Turban, Ramesh Sharda, David King, and Dursun Delen. *Business Intelligence: A Managerial Approach, Second Edition*. Pearson Education, 2011.

Alper Unler and Alper Murat. A discrete particle swarm optimization method for feature selection in binary classification problems. *European Journal of Operational Research*, 206(3):528–539, 2010.

Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In *KDD*, pages 639–644. ACM, 2002. ISBN 1-58113-567-X.

Antal van den Bosch. Using induced rules as complex features in memory-based language learning. In Walter Daelemans and Rémi Zajac, editors, *CoNLL/LLL*, pages 73–78. ACL, 2000.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. reCAPTCHA: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

Denny Vrandecic. Wikidata: a new platform for collaborative data collection. In Alain Mille, Fabien L. Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab, editors, *WWW (Companion Volume)*, pages 1063–1064. ACM, 2012. ISBN 978-1-4503-1230-1.

Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, September 2014. ISSN 0001-0782.

Nigel Williams, Sebastian Zander, and Grenville J. Armitage. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. *Computer Communication Review*, 36(5):5–16, 2006.

I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey McLachlan, Angus Ng, Bing Liu, Philip Yu, Zhi-Hua Zhou, Michael Steinbach, David Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, January 2008. ISSN 0219-1377.

Bing Xue, Mengjie Zhang, and Will N. Browne. Particle swarm optimization for feature selection in classification: A multi-objective approach. *IEEE Trans. Cybernetics*, 43 (6):1656–1671, 2013.

Qiang Yang and Xindong Wu. 10 challenging problems in data mining research. *International Journal of Information Technology and Decision Making*, 5(4):597–604, 2006.

Xiaoxin Yin and Jiawei Han. Efficient classification from multiple heterogeneous databases. In Alípio Jorge, Luís Torgo, Pavel Brazdil, Rui Camacho, and João Gama, editors, *PKDD*, volume 3721 of *Lecture Notes in Computer Science*, pages 404–416. Springer, 2005. ISBN 3-540-29244-6.

Hwanjo Yu and Sungchul Kim. Svm tutorial - classification, regression and ranking. In Grzegorz Rozenberg, Thomas Bäck, and Joost N. Kok, editors, *Handbook of Natural Computing*, pages 479–506. Springer, 2012. ISBN 978-3-540-92909-3.

Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In Tom Fawcett and Nina Mishra, editors, *ICML*, pages 856–863. AAAI Press, 2003. ISBN 1-57735-189-4.

Li Zeng, Ling Li, Lian Duan, Kevin Lü, Zhongzhi Shi, Maoguang Wang, Wenjuan Wu, and Ping Luo. Distributed data mining: a survey. *Information Technology and Management*, 13(4):403–409, 2012.

Guoqiang Peter Zhang. Neural networks for classification: a survey. *IEEE Trans. Systems, Man, and Cybernetics, Part C*, 30(4):451–462, 2000.

Li Zhang, Jianhua Luo, and Suying Yang. Forecasting box office revenue of movies with bp neural network. *Expert Syst. Appl.*, 36(3):6580–6587, 2009.

Wenbin Zhang and Steven Skiena. Improving movie gross prediction through news analysis. In *Web Intelligence*, pages 301–304. IEEE Computer Society, 2009. ISBN 978-0-7695-3801-3.

Howard Zhou, Tucker Hermans, Asmita V. Karandikar, and James M. Rehg. Movie genre classification via scene categorization. In Alberto Del Bimbo, Shih-Fu Chang, and Arnold W. M. Smeulders, editors, *ACM Multimedia*, pages 747–750. ACM, 2010. ISBN 978-1-60558-933-6.