UNIVERSITY OF SOUTHAMPTON

Machine learning-assisted railway simulation modelling

by

Joanna Cameron Knight

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

in the
Faculty of Engineering and Physical Sciences
School of Engineering

August 2023

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES
SCHOOL OF ENGINEERING

Doctor of Philosophy

by Joanna Cameron Knight

Computer simulation models have the potential to aid decision-making in many industries. They are particularly beneficial when applied to situations that would be too costly or impractical to experiment with in reality. In the railway domain, such applications may include making infrastructure changes or updating a timetable.

A simulation is a representation of a system or object, not an exact copy. However, to be a meaningful planning aid, the output of a simulation needs to be sufficiently realistic. A review of existing railway simulation software identifies two features where established modelling techniques limit the realism of the output: train movements and signalling decisions.

This thesis develops a stochastic simulation that models train movements and signalling decisions using supervised machine learning. Train movements are modelled using quantile regression. The models are applied to replicate the distribution of travel durations without requiring any underlying assumptions about the properties of the distribution. Probabilistic classification is employed to predict signallers' actions, achieving accuracies of over 89%.

Experiments were conducted by running the stochastic simulation for a week's worth of activity on a section of the British network for 600 iterations. The outputs were compared to the true activity of the selected week. The true total travel duration of all the trains fell within the simulated minimum and maximum values achieved across all iterations. Moreover, the median total travel duration across all iterations settled to within 1% of the true value.

The results demonstrate that machine learning-assisted simulation modelling can be applied to make realistic performance assessments in the railway domain. There is the possibility to use such a model to compare alternative train regulation policies or to investigate the performance of a new timetable.

Contents

List of Figures

List of Tables

Acronyms

ANN Artificial neural network.

ARS Automatic route setting.

BNN Bayesian neural network.

CIF Common interface file.

CNN Convolutional neural network.

CORPUS Codes for Operations, Retail and Planning - a Unified Solution.

ECS Empty coaching stock.

EQD expected quantile division.

FOC Freight operating company.

GPS Global positioning system.

GRU Gated recurrent unit.

HPC High-performance computing.

ICP Interval covering percentage.

JSON Javascript object notation.

KS Komogorov-Smirnov.

LSTM Long short term memory.

LTS Least-trimmed squares.

MAE Mean absolute error (see Equation 3.3).

MAPE Mean absolute percentage error (see Equation 3.6).

MDN Mixture-density network.

MQE mean quantile error (see Equation 6.1).

MSE Mean squared error (see Equation 3.4).

OOP Object-oriented programming.

ORR The Office of Rail and Road.

OTMR On-train monitoring and recording.

RMSE Root mean squared error (see Equation 3.5).

RNN Recurrent neural network.

ROSCO Rolling stock company.

SMART Signal Monitoring and Reporting of Trains.

SRT Sectional running time.

STANOX Station number.

STP Short-term plan or planning.

SVM Support vector machine.

TD Train describer.

TIPLOC Timing point location.

TOC Train operating company.

TOPS Total Operations Processing System.

TPRs Timetable planning rules.

TRUST Train Running Under System TOPS.

UK United Kingdom.

UML Unified Modelling Language.

VSTP Very short-term plan or planning.

WTT Working timetable.

Glossary

R^2 the coefficient of determination, see Equation 3.7.

k-nearest neighbours a machine learning approach for classification or regression that makes predictions based on the k closest data records, see Section 3.3.2.

accuracy an evaluation metric for a classification model: the fraction of all records that have been correctly classified, see Equation 3.8.

artificial neural network a machine learning approach for classification or regression consisting of a network of neurons, see Section 3.3.5.

aspect The colour of a railway signal.

asynchronous (of a railway simulation model) when trains are run through a simulation in priority order, cf. synchronous.

axle counter A device that use sensors to count the axles of a train into and out of a block section.

Bayesian neural network a neural network model that can make probabilistic predictions.

benchmark model a model whose results can be used to compare with other models, other models would be expected to outperform the benchmark.

berth a field containing the headcode of a train in a block section of track.

binary classification (of a model) a classification model that has only two possible target values.

Brier loss a loss value for probabilistic predictions; calculates the mean square difference between the predicted probability and the true value; see Equation 8.1.

C-Class data data feed providing the movements of trains between berths.

capacity (on a railway network) the maximum number of train services that can travel over the infrastructure.

class (of object-oriented design) a logical unit of code containing attributes (data) and methods (behaviours).

classification (of a model) a supervised machine learning model that predicts discrete target values.

conflict detection the first step in a rescheduling process, identifying potential conflicting train services.

conflict resolution the second step in a rescheduling process, taking action to avoid a conflict.

controller (on a railway network) a person responsible for monitoring a larger network area than a signaller and managing wide-scale disruptions.

convolutional neural network a neural network model that accepts multi-dimensional input data and was originally designed to classify images.

cost function (of a machine learning algorithm) the sum of loss functions for all records in a data set.

Covid-19 an infectious disease caused by the SARS-CoV-2 virus.

cross-fold validation an approach for validating a machine learning model whereby the test data are set aside, and the remaining data are randomly partitioned into k sets. The training process then takes place over k iterations, with one of the k sets held out as the validation data in each iteration.

decision tree a machine learning approach for classification or regression, where the data are recursively paritioned to make predictions, see Section 3.3.3.

deep learning machine learning using an artificial neural network with many hidden layers.

delay (on a railway network) a positive time difference between the scheduled and actual times at a location.

Delay Attribution Board an industry body that oversees and arbitrates the process of assigning delays on the railway network to an underlying cause.

deterministic not random; (of a simulation model) without any random inputs - the same output will always be produced for given input parameters, cf. stochastic.

direct delay (on a railway network) a delay caused by a disturbance, cf. reactionary delay.

disturbance (on a railway network) an event that interferes with the running of the network.

dwell duration The length of time a train is stationary at a platform.

ensemble (of a machine learning model) a model that makes predictions using multiple individual models.

epoch an iteration of the building an artificial neural network using all the training data.

event-driven (of a simulation model) where the next time a change in status will occur for each object in the simulation is calculated, the simulation the steps forward in time to the next event, cf. time-driven.

expected quantile division (specific to this thesis) for a given value of τ and n records in the dataset, expected quantile division is defined as 'True' if there are fewer than $\tau * n$ records smaller than their predictions and fewer than $(1–\tau * n)$ records greater than their predictions. The expected quantile division is 'False' otherwise..

feature engineering the process of manipulating feature vectors through transformations.

feature vector an input vector to a machine learning model, will be paired with a target value.

fixed block A signalling system where the track is divided into blocks, cf. moving block.

FRISO (Flexible Rail Infrastructure Simulation of Operations) a microscopic railway simulation model used on the Dutch network.

gated recurrent unit a type of neuron used in a recurrent neural network.

gradient descent an optimisation algorithm that seeks to minimise a cost function.

gradient tree boosting a machine learning approach for classification or regression that uses an ensemble of decision trees, see Section 3.3.3.

Great British Railways the public body due to replace Network Rail in 2023.

headcode an identifier for a train service consisting of four alpha-numeric characters.

headway (on a railway network) a duration between two consecutive trains passing the same timing point location.

HERMES (Holistic Environment for Rail Modelling and Experimental Simulation) a microscopic synchronous railway simulation model, owned by Graffica.

heterogeneous (of a railway network) when there are a variety of the types of services on the network, cf. homogeneous.

heteroscedasticity when the variability of a variable is unequal across a second variable.

homogeneous (of a railway network) when the services follow the same stopping patterns and run at the same speeds, cf. heterogeneous.

hyperparameter (of a machine learning model) parameters set before the model is built.

infrastructure manager (of a railway network) an organisation responsible for the ownership and management of a railway network.

inheritance (of object-oriented design) a hierarchical structure with 'parent' and 'child' classes whereby the child class inherits the attributes and methods of the parent.

interlocking a feature of signalling systems that prevents conflicting movements.

interval covering percentage the percentage of records falling in a given prediction interval.

isotonic regression a method for calibrating machine learning models.

iteration one run of a stochastic simulation model.

least-trimmed squares (of machine learning) a method uses a subset of inlying observations to fit the machine learning model rather than the whole dataset.

linear programming an algorithm for optimisation where the objective function and constraints are linear.

linear regression a machine learning approach for regression problems that constructs a linear relationship between the feature vectors, see Section 3.3.1.

logistic regression a machine learning approach for linear classification, see Section 3.3.1.

long short term memory cell a type of neuron used in a recurrent neural network.

loss function (of a machine learning algorithm) a function that measures the difference between a prediction and its target value.

machine learning-assisted simulation modelling a modelling approach that combines machine learning with a simulation model, where the simulation model is the dominant model.

macroscopic (of a railway simulation model) when the infrastructure is modelled with little detail, cf. microscopic, mesoscopic.

margin (on a railway network) a duration that elapses between any two consecutive trains where one train crosses the path of the other.

mean quantile error (specific to this thesis) a metric which has been derived from the quantile loss function, see Equation 6.1.

mesoscopic (of a railway simulation model) when the infrastructure is modelled at a level of detail in between microscopic and macroscopic.

microscopic (of a railway simulation model) when the infrastructure is modelled in great detail, cf. macroscopic, mesoscopic.

mixture-density network a neural network model that output parameters of multiple distributions, which are then used to construct a mixture of distributions to predict the conditional probability distribution of the target value.

movement authority the method of communication for a train to proceed.

moving block A signalling system where the blocks move with the trains, cf. fixed block.

Network Rail the railway infrastructure manager in Great Britain.

neuron a cell or building-block of an artificial neural network.

node-edge graph a graph made up of nodes, connected by edges.

non-parametric (of a machine learning model) where the mapping function created by the model does not take a pre-determined form.

object (of object-oriented design) an instance of a class created at run-time.

objective function (of an optimisation problem) a function that it is desired to maximize or minimize.

open data data that are freely available with few conditions on their use.

OpenTrack a microscopic synchronous railway simulation model widely used in Europe, owned by OpenTrack Railway Technology Ltd.

operational (of a time frame) a time frame of less than a day ahead.

parametric (of a machine learning model) where the model takes a known form.

polymorphism (of object-oriented design) the ability of a child class to override the methods of a parent class.

precision an evaluation metric for a classification model: the fraction of records that have been correctly classified as a specific target value, see Equation 3.9.

pseudo-random number a number that appears random, but is generated by an algorithm.

public timetable a schedule of passenger services available to the public, may contain slight differences to the working timetable.

Python a high-level programming language distributed under a permissive free software license.

quantile a division of an ordered dataset that determines how many values are above or below.

quantile level a quantile value of an ordered dataset, denoted τ, whereby the probability that any value in the dataset is smaller than the quantile level is τ ($0 < \tau < 1$).

quantile regression an approach to regression that predicts conditional quantiles for the target value, see Section 3.3.6.

RailSys a microscopic synchronous railway simulation model widely used in Europe, owned by RMCon.

random forest a machine learning approach for classification or regression that uses an ensemble of decision trees, see Section 3.3.3.

random seed a number that is used to initialise an algorithm that generates pseudo-random numbers.

reactionary delay (on a railway network) a delay caused by another delayed train on the network, cf. direct delay.

recall an evaluation metric for a classification model: the fraction of records that belonged to a specific target value that were correctly classified, see Equation 3.10.

recurrent neural network a neural network model that retains an internal state (or memory), used for sequence prediction problems.

regression (of a model) a supervised machine learning model that predicts continuous target values.

restricted movement (a term specific to this thesis) when a train enters a berth and the status of the signal to exit the berth is at its most restrictive aspect.

robustness (of a railway timetable) the extent to which a timetable run to schedule and absorb small disturbances.

RTC (Rail Traffic Controller) a microscopic railway simulation model mainly used in North America, owned by Berkeley Simulation Software.

S-Class data data feed providing the state of signalling equipment.

signaller (on a railway network) a person responsible for managing traffic in a section of the network.

SIMONE (SImulation of MOdel NEtwork) a macroscopic railway simulation model used on the Dutch network.

simulation model a digital model that mimics the operations of a system.

step (relating to train movements) a directed movement between two berths.

stochastic random; (of a simulation model) with random input variables - the same input values may lead to different outputs, unless the first random seed is set as input, cf. deterministic.

strategic (of a time frame) a time frame of two or more years ahead.

supervised machine learning a branch of machine learning that aims to find a function f that maps each input value x_i to its output y_i, for a given a set of input and output pairs, $(x_1, y_1), ..., (x_N, y_N)$.

support vector machine a machine learning approach for classification or regression, see Section 3.3.4.

support vector regression a support vector machine predicting a continuous variable.

switch and crossing moveable track sections that allow movement between different rail lines.

synchronous (of a railway simulation model) when trains are run through a simulation in simultaneously, cf. asynchronous.

tactical (of a time frame) a time frame of one day up to two years ahead.

target value the desired output value of a machine learning model for a given feature vector.

test data (of machine learning) data that are used to provide a final evaluation of a machine learning model.

time-driven (of a simulation model) where all objects in a simulation model are updated at short, fixed, time intervals, cf. event-driven.

timing point location a location on a railway network that is used to create timetables and schedules - often at stations or junctions.

track circuit An electrical circuit used to detect the presence of a train in a block section.

track-based data data from sensors installed within the infrastructure.

traffic management the actions of signallers and controllers to manage railway traffic.

TRAIL (Transport, Reliability, Availability, and Integrated Logistics) a macroscopic synchornous railway simulation model used by Network Rail.

train detection the ability to know where a train is on the network.

train regulation the process of determining the priority of trains at junctions.

train-based data data from sensors fitted to on a train.

training data (of machine learning) data that are used to build a machine learning model.

unrestricted movement (a term specific to this thesis) when a train enters a berth and the status of the signal to exit the berth is not at its most restrictive aspect.

validation (of a simulation model) activities that investigate whether the simulation is fit for its intended purpose, cf. verification.

validation data (of machine learning) data that are used to evaluate a machine learning model during the build process.

verification (of a simulation model) activities that investigate whether the simulation has been implemented as designed, cf. validation.

VISION (Visualisation and Interactive Simulation of Rail Networks) a microscopic synchronous railway simulation model developed for use on the British network.

working timetable a schedule used by the rail industry for planning purposes, cf. public timetable.

Acknowledgements

This research was funded by Network Rail and EPSRC through an Industrial CASE research studentship. I am grateful for the helpful discussions with Ben Ford and Lloyd Barson from Network Rail.

I would like to thank Professor Andy Keane and Dr Ondrej Hovorka for their supervision, guidance and advice throughout this work. Also, Professor John Preston and Professor Ronghui Lui, who examined this work and provided a list of corrections, and Dr John Armstrong for his feedback following my progress reviews. Any mistakes or typing errors remaining in the thesis are my own.

I am indebted to the community of professionals and enthusiasts who have contributed to the public information on Network Rail's open data feeds, without which this work would have taken much longer. In particular, Peter Hicks of OpenTrainTimes Ltd has provided archives of open data and has put a generous amount of his knowledge regarding the data feeds into the public domain.

Much of the work presented in this thesis was carried out using the IRIDIS High-Performance Computing Facility at the University of Southampton.

Chapter 1

Introduction

1.1 Motivation

Simulation models can be valuable planning tools in many industries as they mimic systems and behaviours that might be too costly or impractical to experiment with in reality. The railway industry is no exception, and computer simulations have been used in British railway operations since the 1970s to aid infrastructure and timetable planning (McGuire and Linder, 1994).

A simulation is a representation of a system or object, not an exact copy. However, to be a meaningful planning aid, the output of a simulation needs to be sufficiently realistic. A review of existing railway simulation software in this thesis will identify two features where established modelling techniques limit the realism of the models' output.

The first feature is the representation of train movements. In current simulation models, it is not straightforward to model the stochastic nature of train movements, such as the variation that might arise from different driving behaviours. The result is that trains travel more consistently in most simulation models than in reality. This can affect assessments of the performance of a railway network.

The second feature is the signalling decisions taken within simulations, as existing models do not implement the full complexities of signalling decisions. Signallers manage railway traffic and make decisions that depend on the area they control and the traffic state. Although signallers follow general guidelines for managing traffic conflicts, not every scenario can be planned in advance, and decisions do not always generalise between junctions or areas. There may be inconsistencies in decision-making with two signallers taking different actions when presented with the same scenario. The representation of such decisions in a railway simulation affects how delays propagate in the simulation, which, again, can lead to misleading performance assessments.

This thesis develops a methodology for modelling train movements and signalling decisions utilising supervised machine learning within a simulation. Supervised machine learning is a modelling paradigm that aims to 'learn' rules from historical data. It is a sub-field of artificial intelligence, of which there are an increasing number of applications in the literature related to the railway domain (Bešinović et al., 2022).

Many machine learning algorithms can generalise complex, non-linear relationships in data to predict future outcomes. Such methods often require large amounts of data from which to learn and are candidates for predicting train movements and signalling decisions owing to the quantities of data available from the railway network. Hybrid machine learning and simulation models have been reported outside the railway domain but appear to be a novel approach for analysing railway traffic performance.

The remainder of this chapter provides background material relevant to this thesis. The naming conventions for terminology introduced in this and subsequent chapters are not universal. For example, outside of Great Britain, a signaller may be called a dispatcher. This thesis uses conventions most appropriate to the British railway industry.

Section 1.2 outlines the aim of the research and the objectives. Section 1.3 describes how the British railway industry is organised. Further background information is then provided in Section 1.4, discussing general features of railway operations with specific references to the British network. Simulation models are mainly used to support planning activities, and Section 1.5 describes three planning horizons in the railway industry: strategic, tactical, and operational. Machine learning and simulation models require large amounts of data, and Section 1.6 introduces two broad groups of operational railway data, namely track- and train-based data.

Chapter 2 builds on the background information to focus on railway simulation modelling. The chapter discusses current modelling practices and provides a literature review of applications. Simulation models' general strengths and weaknesses are discussed, and the implementation of train movements and signalling decisions are examined in more detail. Chapter 3 then introduces supervised machine learning and provides a literature review to illustrate the potential for such models to predict train movements and signalling decisions. The chapter also reviews machine learning-assisted simulation literature and concludes that this method could be applied in the railway domain.

Chapter 4 describes the modelling framework this research has developed, consisting of three parts: constructing datasets, building machine learning models, and developing a simulation model. The results from the three areas of work are covered in the subsequent chapters. Chapter 5 discusses the creation of the datasets and presents a small section of the network used to illustrate the work.

Chapter 6 presents the machine learning models for train movements. The machine learning models are applied in Chapter 7 within a machine learning-assisted simulation

model. Chapter 8 presents the machine learning models for representing signalling decisions, and Chapter 9 applies them within the simulation model. An early version of the work in these chapters was presented at the World Congress on Railway Research (Knight et al., 2022). Chapter 10 concludes the thesis by discussing the merits of the research and the opportunities for further work.

Further information on the data sources used in this thesis can be found in Appendix A, and additional details on the construction of the datasets in Appendix B. A dataset in support of this thesis including the raw data, processed data, results and machine learning models is available (Knight, 2023).

1.2 Aims and objectives

The aim of this research is to evaluate the application of machine learning-assisted simulation modelling for railway networks. Incorporating machine learning into simulation modelling is not an established technique in the railway industry; however, there is potential for this approach to be considered owing to the large amounts of data available in the railway domain.

In order to achieve this aim, the objectives of the research will be to develop three types of models:

- Machine learning models to predict the travel duration of trains between discrete points on the network
- Machine learning models to predict the outcome of traffic management decisions
- A simulation model that can utilise the machine learning models to determine some dynamics within the simulation

Through these models, this research will challenge established methods of modelling train movements and traffic management within railway simulations. The relative strengths and weaknesses of machine learning-assisted simulation and the potential applications for the approach will be considered.

1.3 The British railway industry

In Great Britain[1], like many European countries, the ownership and management of the railway infrastructure is provided by an infrastructure manager. The infrastructure manager in Great Britain is Network Rail[2], an arms-length public sector body. As

[1] Great Britain includes England, Scotland and Wales but excludes Northern Ireland where different organisations are in place.

[2] https://www.networkrail.co.uk/

Network Rail has sponsored this research, the perspective of an infrastructure manager will dominate throughout this thesis.

Network Rail's responsibilities include, but are not limited to:

- Maintaining and upgrading the railway infrastructure
- Operating and managing the performance of the network
- Creating timetables
- Managing railway traffic

At the time of writing, Network Rail is due to be replaced by a new public body called Great British Railways (Department for Transport, 2021b). The new organisation will absorb the responsibilities of Network Rail and will also take on additional tasks, such as the procurement of passenger services.

The operation of passenger train services on the network is mainly carried out by private train operating companies (TOCs) under franchise. However, the United Kingdom (UK) government has created companies which run rail franchises under public ownership known as 'operators of last resort'[3]; this may occur for several reasons, such as a franchise not delivering acceptable service levels. The UK government also provided financial support to private TOCs in response to the Covid-19 pandemic (National Audit Office, 2021). Train services will continue to be predominantly operated by private companies after the introduction of Great British Railways, although the system for contracting the companies will change.

Private rolling stock companies (ROSCOs) own the fleet of passenger trains and lease them out to the TOCs. Freight operating companies (FOCs) provide freight train services; these are also privately owned. The Office of Rail and Road (ORR) acts as a regulator for Network Rail and, to a lesser extent, the TOCs.

1.4 Features of railway operations

1.4.1 Infrastructure

The layout of rails restricts trains' movements with limited opportunities for overtaking or re-routing. Switches and crossings are moveable track sections that allow movement between different rail lines. Although they increase the number of routes across

[3] https://www.gov.uk/government/organisations/dft-olr-holdings-limited/about

a network, switches and crossings are also more vulnerable to wear and require regular maintenance. The failure of an infrastructure component can result in trains being cancelled or delayed.

Trains often travel at speeds that prevent them from stopping within the driver's sighting distance. A railway signalling system safely controls the movement of trains around the network. Two key features of a railway signalling system are:

- Train detection - the ability to know where a train is on the network
- Movement authority - the permission for a train to proceed

Most areas of the British railway network use fixed block signalling, where the track is divided into blocks and trains are detected using either track circuits or axle counters. Track circuits create an electrical circuit connecting the two rails in a block through a relay. When a train moves into a block, it causes the electrical current to bypass the relay, informing the signalling system that there is a train in that block. Axle counters are devices that use sensors to count the axles of a train into and out of a block; it will likewise determine if the block is occupied or clear. Neither detection method can identify a train's precise location.

Track-side colour-light signals are the most common method of communicating movement authority to drivers. The colours that display to the driver are known as aspects. A red aspect is the most restrictive; a train should not pass a red signal unless a signaller has granted explicit authority, which only happens in exceptional circumstances. A green aspect is the least restrictive and may be passed at the maximum permitted speed. Signals with only red and green aspects are known as two-aspect signals.

There are two-, three- and four-aspect signals across the network (see Figure 1.1). Three- and four-aspect signals include a yellow aspect which indicates caution, and the driver should be prepared to stop at the subsequent signal. A double yellow aspect indicates that the next signal is showing yellow.

Track-side signals are placed some distance ahead of the track they protect to allow a safe braking distance should a driver pass a red signal. Some older-style semaphore signals exist for parts of the network. There are also more modern systems for high-speed trains whereby the driver has the movement authority displayed in the cab.

An alternative signalling system is known as moving block signalling. Moving block signalling requires the train to have an onboard computer which transmits the train's location via radio transmissions. A central computer system then defines the blocks – the blocks 'move' with the trains – and transmits movement authority to drivers in the cab. The only part of the British railway network to implement moving block signalling

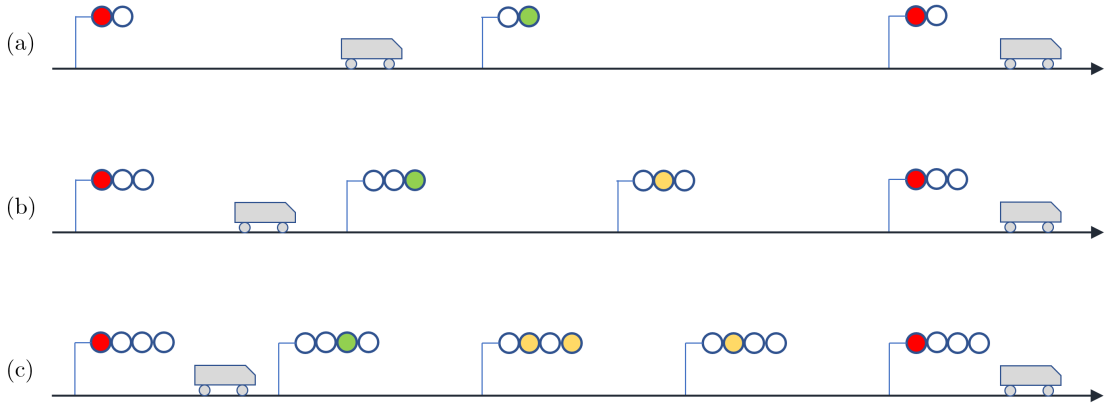

FIGURE 1.1: Representations of (a) two-aspect, (b) three-aspect, and (b) four-aspect signals

is the Docklands Light Railway[4], an automated metro system in London. All signalling systems referred to going forward will be fixed block unless otherwise stated.

A signaller is responsible for managing traffic in a section of the network. Signallers can set the routes that trains take and operate the signals in the area under their control. They do this by operating manual levers or using computer technology. There are many scenarios where signal states will change automatically; for example, when a train enters a block, all signals providing movement authority into the block will change to their most restrictive aspect.

Some areas of the network also have automatic route setting (ARS), whereby a computer sets trains' routes. ARS does not remove the need for signallers; a human signaller is still required to monitor the area and intervene where necessary, as the computer will not have full situational awareness.

A feature of signalling systems known as interlocking prevents signallers and ARS from taking unsafe actions, such as setting two conflicting routes simultaneously. Interlocking systems are designed to be 'fail-safe', meaning any fault with the signalling equipment will change the surrounding signals to red.

1.4.2 Demand

In the 20 years before the Covid-19 pandemic, the number of passenger rail journeys in Great Britain increased by 90% (Office of Rail and Road, 2020). However, the pandemic changed working patterns and behaviours, significantly decreasing the need for passenger rail services. At the time of writing, the number of passenger journeys is increasing but has not yet returned to pre-pandemic levels (Office of Rail and Road, 2022b). Passenger trains typically follow recurring daily or weekly schedules, with exceptions for major

[4] https://tfl.gov.uk/modes/dlr/

sporting events or public holidays where regular services may be enhanced or reduced as appropriate.

The amount of freight carried on the rail network was also affected by the pandemic, but it has since recovered to pre-pandemic levels (Office of Rail and Road, 2022a). Historically, coal was the main commodity transported by rail, but this has dramatically declined in recent years. Other commodities, such as construction aggregates, have become more common. The demand for rail freight is anticipated to increase further over the coming years, owing to factors such as a shortage of drivers for heavy goods vehicles and companies planning to reduce their carbon emissions (BBC News, 2021). Indeed, both passenger and freight rail services are expected to play a role if the UK Government's net-zero carbon emissions target is to be achieved by 2050 (Department for Transport, 2021a).

The majority of the British rail network is said to be heterogeneous owing to the mix of passenger and freight traffic and the variety of passenger trains' stopping schedules. A network is homogeneous if services follow the same stopping pattern and run at the same speeds. An example of a near-homogeneous network would be the London Underground.

1.4.3 Performance

A disturbance on the railway is an event that interferes with the running of the network. Disturbances could be:

- Technical - for example, signal failure
- Human - for example, variation in time for passengers to board a train
- Environmental - for example, a tree that has fallen on the rail tracks

A delay is a positive time difference between the scheduled and actual arrival or departure times. Disturbances are the source of delays, but a disturbance does not *always* lead to a delay (Cui et al., 2016). When a disturbance leads to a delay, it is called a direct delay. A direct delay may propagate throughout the network, causing reactionary delays to other trains (Delay Attribution Board, 2019).

Intervention may be required when trains are delayed and deviate from their schedule. Signallers manage traffic during small-scale disruptions, whereas a controller monitors larger network areas and coordinates the response to wide-scale disruptions (Ryan et al., 2009). Interventions may include changing the platform that a train arrives at or regulating a train at a junction. Train regulation is a term used to describe the process of determining the priority of trains at junctions on the network (Rail Safety and Standards

Board, 2021a). More generally, the actions of signallers and controllers may be referred to as traffic management.

It is common practice for delays below a given time threshold to have no reason attributed to them. Delays above a threshold are assigned to an underlying cause, although this is not always straightforward. In Great Britain, an industry body called the Delay Attribution Board oversees and arbitrates the process (Delay Attribution Board, 2019).

Industry performance measures are high-level and relatively naïve as they need to be straightforward to collate and understand. In Great Britain, an 'on time' measure indicates punctuality across all services at station stops (Network Rail, 2021). Network Rail measures punctuality as the percentage of trains that arrive early, 'on time' (within one minute of the scheduled arrival time), and within three, five, ten and fifteen minutes.

1.5 Planning in the railway environment

1.5.1 Strategic planning

There are three planning horizons, which are differentiated by the time frame in which plans are implemented. The strategic horizon generally concerns long-term planning two or more years ahead (Watson, 2008). For an infrastructure manager, this would typically involve decisions around infrastructure, such as investing in new railway lines or upgrading signalling systems.

Capacity is a crucial concept in strategic planning. Four different types of railway capacity are defined as follows (Krueger, 1999), (Abril et al., 2008):

- Theoretical capacity - the maximum number of train services that could travel over the infrastructure if they all ran to the same stopping pattern and there were no delays.
- Practical capacity – a more realistic upper limit for capacity, given a representative mix of train services that run on the network. It is reportedly around 60% to 75% of theoretical capacity.
- Used capacity - the actual capacity used by a timetable.
- Available capacity - the unused capacity, the difference between the practical and used capacity.

A signalling system impacts the capacity of the railway network (Goverde et al., 2013). For example, the length of blocks in fixed block signalling systems affects the capacity as only one train can occupy a block.

1.5.2 Tactical planning

Tactical planning can include a timescale from one day up to two years. In this time frame, the infrastructure is typically considered fixed, and the most notable activity for an infrastructure manager is the development of timetables (Watson, 2008). In order to create a timetable, (TIPLOCs) exist on the network. TIPLOCs represent features on the network, such as stations and junctions and are classed as mandatory or non-mandatory.

The working timetable (WTT) contains a schedule for all planned services. Arrival and departure times are given for TIPLOCs where a service is scheduled to stop. A passing time is provided for all mandatory TIPLOCs the train will traverse on its journey. In contrast, non-mandatory TIPLOCs will only be listed in a schedule if the train stops at the location.

For passenger services, another public timetable exists, containing only the arrival and departure times for train services that stop at stations. Additionally, the arrival and departure times in the public timetable may differ slightly from those in the WTT. Arrival times in the WTT may be slightly earlier than the public timetable, and departure times slightly later.

Network Rail publishes significant revisions to the timetables in December each year, with a typically smaller set of adjustments published in May. Creating a timetable is a job that takes months for even a medium-sized rail network (Carey and Crawford, 2007). Network Rail's consultation process for each bi-annual revision begins 64 weeks before implementation (Network Rail, 2019a).

Any change to a timetable must adhere to the timetable planning rules (TPRs). Examples of information included in the TPRs are (Network Rail, 2019b):

- Sectional running times (SRTs) - the duration it takes different types of train to run between two points of the network. The planned running times in the timetable must be greater than or equal to the equivalent sectional running times.
- Headway timings - the minimum duration that must elapse between two consecutive trains passing the same TIPLOC.
- Margin timings - the minimum duration which must elapse between any two consecutive trains where one train must cross the path of the other. These are generally required at junctions and on the approach to large stations.

Optimising a railway timetable is not straightforward as there is no single objective function to consider. The number of stakeholders involved in the railway and their demands create multiple, often conflicting objectives. Planners must also schedule infrastructure

maintenance and include the effects of this in the published timetable; speed restrictions or replacement bus services may be required during maintenance work.

Changes to scheduled trains can occur outside the bi-annual revision process in the form of short-term plans (STPs) and very short-term plans (VSTPs). These may be necessary to alter train schedules for events such as extreme weather or emergency maintenance work. STPs can occur any time after the timetable has been published up to 48 hours before a train is scheduled, and VSTPs concern timetable changes for trains due to run within 48 hours.

The development of a new timetable may also lead to train regulation policies being reviewed and updated in the tactical timeframe. However, there is currently no national process for the management of such policies (Rail Safety and Standards Board, 2021b).

1.5.3 Operational planning

Operational planning is concerned with timescales of less than a day and covers real-time traffic control across the network. If all trains adhere to their timetable, traffic management should be straightforward. However, if a train is running late, then intervention may be needed to reduce the impact of the delay.

During real-time railway operations, signallers monitor and control the traffic. Should delays occur, signallers may intervene by re-timing, re-ordering, re-routing or cancelling services to mitigate the impact (Quaglietta et al., 2013). There are two steps to a rescheduling process: the first is to identify the possible conflicts between services, and the second is a problem-solving phase concerning the strategy to resolve the conflict. This process may be summarised as conflict detection and conflict resolution (D'Ariano et al., 2014).

Naturally, there is less time for analysis in this time frame than in the previous horizons, and a decision-maker may need to rely on their training and experience. Network Rail develops and maintains train regulation policies that aim to support decision-making. However, such policies need to be clear enough for front-line staff to apply them correctly under pressure and do not necessarily cover all possible scenarios that may occur. Signallers do not always see the full consequences of their actions as they work in localised areas of the network, and trains typically pass through multiple areas (Kecman and Goverde, 2014). They may have to deal with delayed trains entering their section of the network and dispatch trains to adjacent areas without complete knowledge.

1.6 Railway data

1.6.1 Track-based data

Track-based data originate from sensors installed within the infrastructure. Train describer (TD) systems are the most common method of track-based monitoring on the British network. They record the location of trains at discrete points on the infrastructure. The British railway network divides into TD areas, and track circuits pass information to the TD system when trains pass into a block. There are many different TD systems across the British network, and two systems may not provide the same data.

TD data do not provide the exact location of trains, which is a limitation. They can only specify that a train exists between two measuring positions (Longo et al., 2012). Depending on the specific TD system, other shortcomings may include:

- Difficulty coupling signal changes to train movement data (Kecman and Goverde, 2014)
- The rounding of occupation and release times (typically to the nearest second) (Bešinović et al., 2013)
- The location of recording points not coinciding with TIPLOCs (Barson et al., 2016)

The lack of coincidence with TIPLOCs means that offset timings are applied to the recorded values to estimate the time a train passes or stops at locations used for performance monitoring. As a result, accurate arrival and departure times at stations cannot be calculated using TD data (Cule et al., 2011; Kecman and Goverde, 2015; Barson et al., 2016; Sørensen et al., 2017; Cerreto et al., 2018).

1.6.2 Train-based data

Trains on the British network are fitted with on-train monitoring and recording (OTMR) equipment (Barson et al., 2016). These data are not available in real-time; they are typically downloaded during maintenance. OTMR data can provide detailed information on the train state, such as speed, acceleration or traction demand. However, in Britain, these data are owned by the TOCs, not Network Rail, which adds a barrier to analysing and comparing such data from different companies.

Global positioning system (GPS) trackers may also be fitted to trains, although this is not mandatory for all trains on the British network. Both OTMR and GPS data are continuous and can be used to reconstruct exact behaviour in time and space (Longo

et al., 2012). Precise arrival and departure times can be calculated using train-based data sources.

1.7 Conclusion

This first chapter has provided background material on British railway operations. The information has not been exhaustive; instead, the emphasis has been on terminology and characteristics relevant to this thesis. The next chapter focuses on railway simulation modelling and builds on the concepts introduced here. In particular, a literature review will show how simulation models can support activities relevant to the three planning horizons.

Chapter 2

Railway Simulation Modelling

2.1 Introduction

The first chapter provided background on the organisation of the British railway industry and introduced terminology relevant to this work. The chapter discussed the main activities in the three planning horizons (strategic, tactical and operational) and presented two types of railway data (track- and train-based data). As this thesis investigates improvements to railway simulation modelling, this chapter concentrates on current practices.

A model is a representation of a system, object, or process, referred to here as an original. An original may be from any academic discipline, and it need not exist. For example, a new railway line could be modelled before construction. Models can be divided into two broad categories: physical and conceptual models. Physical models are typically small-scale representations of originals, whereas conceptual models are theoretical and use computational, mathematical or diagrammatic techniques. A model is designed to be used in place of an original for some purpose. A common purpose of models is prediction; other examples include explanation, education or training (Epstein, 2008).

A simulation model mimics the behaviour of an original and can be used to explore the dynamics of a complex system over time (Cui et al., 2016; Durán, 2020). This chapter discusses the current approaches to models that simulate railway operations. Such models are conceptual, and Section 2.2 discusses key characteristics specific to railway simulation models. The inputs and outputs of railway simulations are covered, as well as the dynamics of the models as they are executed.

All models are reductive with respect to their original (Kühne, 2005): they are not exact copies, and some degree of abstraction is necessary. An acceptable level of reduction will, in part, depend on the model's purpose, and a model should be evaluated to ensure

it is fit for its purpose. Section 2.3 considers general methods for evaluating simulation models which are not specific to the railway domain.

Section 2.4 introduces some established railway simulation software and briefly describes each model. This is followed by a literature review of applications of simulation models in Section 2.5. Three main application areas are considered: infrastructure, timetables and operations. This chapter concludes with a discussion in Section 2.6 of the strengths and weaknesses of the current state of railway simulation modelling. The discussion narrows to consider two specific weaknesses that this thesis will address.

2.2 Features of railway simulations

2.2.1 Modelling approach

Simulation models require input data to set up a scenario; in the railway domain, these data might include a model of infrastructure and train schedules. The model will then simulate the dynamics of a railway network for some duration (determined by the input data) and provide outputs summarising the activities in the simulation.

The representation of time in a simulation may be event- or time-driven. Event-driven models run by calculating the next event time for all objects in the simulation, such as the time a train will pass a signal or the time a signal will fail (Medeossi and de Fabris, 2018). The simulation then steps forward to the earliest next event time and updates the object(s) whose state has changed. The process continues until the simulation has ended. In time-driven simulation models, the status of objects in the simulation are updated at fixed short time intervals. Some simulations use a hybrid approach whereby characteristics such as train movements are calculated using time-driven algorithms, and other features such as route setting or signal failure are event-driven (Quaglietta et al., 2013).

A simulation model without any random variation is deterministic; such models will produce the same output for given input parameters. A simulation model that includes random variables as input values is stochastic. The random variables represent the probability of events, such as the probability of a fault with the train doors at the next station. As the simulation is executed, it will sample values from the probability distributions as required. Values are sampled using pseudo-random numbers: these are numbers that appear random but have been generated by an algorithm. Such algorithms are initialised with a number known as a random seed.

A single run of a stochastic simulation is called an iteration. Iterations with the same input but initialised with different random seeds are not guaranteed to produce the same output. A single iteration does not provide statistically meaningful results by

itself. Applying a stochastic simulation to a problem requires multiple iterations; the collective output of all the iterations can then be analysed to find likely behaviours and trends. The input data will determine the number of iterations a simulation will execute. Running multiple iterations adds to the computational expense of the model but is necessary when using such a model for evaluating network performance (Botte and D'Acierno, 2018).

2.2.2 Inputs

Infrastructure

In a railway simulation, it is typical to model the network's infrastructure as a node-edge graph. A microscopic model is one where the nodes represent a granular level of detail, such as representing the track down to the nearest metre, modelling all signals and each switch and crossing unit.

Macroscopic simulations model the network at a higher level. In a macroscopic simulation, a station may be defined by a single node (see Figure 2.1). Macroscopic models require fewer input data than microscopic models. Generally, they run faster using less computational effort (Botte and D'Acierno, 2018), making them better suited to modelling large areas of a network than microscopic models. The lack of detail represented in macroscopic models does, however, lead to less specific results and limits their potential applications. An intermediate approach is mesoscopic, although the delineation between mesoscopic and macroscopic is not well defined.

Rolling stock

A model will require different rolling stock characteristics; for example, an electric passenger train will behave differently from a diesel freight train. Characteristics may include features such as the length, power system or position at the simulation start time. The exact characteristics will be defined by the model, with some models allowing greater detail than others. The characteristics influence how trains move around the simulation, affecting their run durations (discussed in Section 2.2.3).

Schedules

Schedules, or timetables, for the rolling stock are usually required to determine the expected movements of the trains around the simulated area. The format of the schedule data will be defined during the development of the model, and some software may be capable of uploading several different data formats.

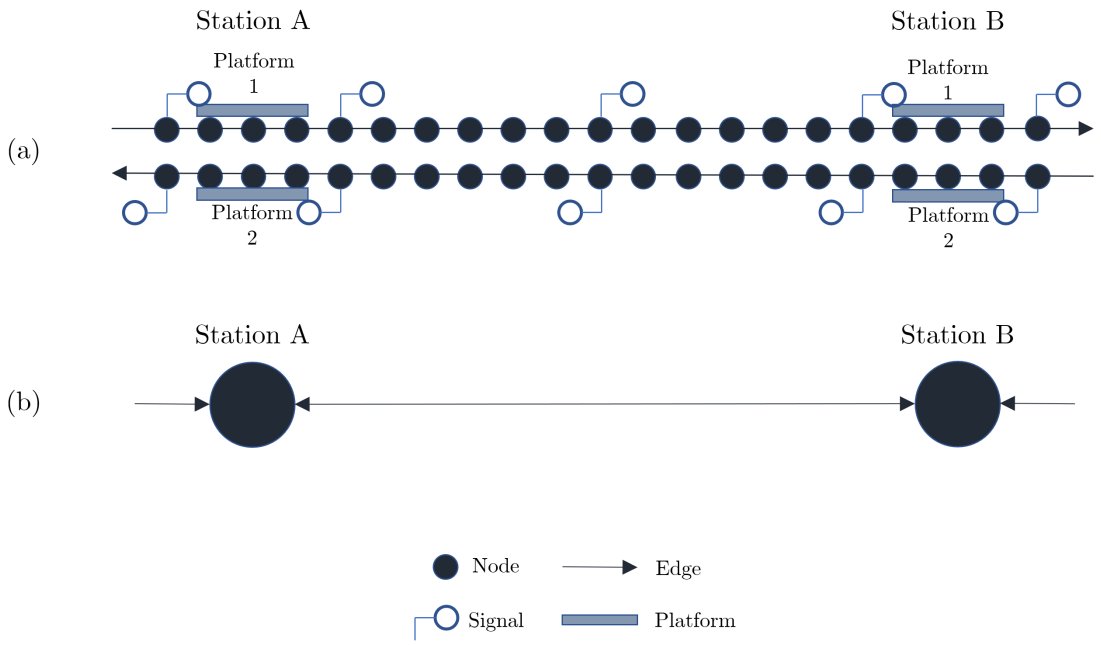

FIGURE 2.1: Example of (a) a microscopic network model, and (b) a macroscopic network model

Some simulation models may provide configuration to allow for flexibility in the departure time, which would allow a train to depart early or late with respect to its schedule. Although a passenger train would not depart earlier than its publicly scheduled time in Great Britain, this can be a characteristic of other types of operation, particularly in the freight industry.

Staff and crew schedules also affect the running of trains, these may be explicitly modelled in a simulation, or a model may assume that crew are always available for the running of a train.

Passenger demand

Not all operational railway simulations will consider passenger demand. Those that do would require input to determine passenger numbers at stations or on routes.

Configuration

Models will also require some input to configure a simulation before it runs. Configuration could include features such as the simulated start date and time and the length of simulated time. The amount of configuration will vary between software; a model with little configuration will limit its flexibility. However, allowing significant configuration in a model will increase its complexity, making it more difficult and time-consuming to set up and validate.

2.2.3 Simulation dynamics

Run durations

Medeossi and de Fabris (2018) state that all established microscopic simulation models use Newton's motion equations to determine the run durations of trains. Train simulators typically solve the motion equations in small time steps to determine a constant acceleration during each step, and then speed and position can be calculated (Douglas et al., 2017). Variability in run durations is usually represented by percentage values, called performance parameters, that are applied to each phase of motion to increase the calculated times (Watson and Medeossi, 2014).

Dwell durations

The dwell duration of a train at a station is the duration that the train is stationary at a platform and consists of various processes such as opening doors, passengers alighting, passengers boarding, and doors closing (Buchmueller et al., 2008). A train may also have to wait at a station after all passengers have boarded if it has arrived early. Not all simulations will model each stage of dwell duration separately; a time duration may represent the whole process of dwelling. The 'train waiting' process may not be carried out every time a train stops at a station; it will only happen if a train arrives early and is forced to wait until its scheduled departure time. Large stations may have additional procedures such as coupling or decoupling of trains. The variability in dwell durations is sometimes represented in simulations by a departure delay function (Longo and Medeossi, 2012).

Traffic management

Railway simulations must respect capacity constraints; for example, two trains cannot simultaneously occupy a platform or track section unless they are in the process of joining or decoupling. Simulations also need to take train regulation decisions to prioritise traffic across junctions. One method of prioritising demand is for the simulation to be asynchronous. In asynchronous simulations, the highest priority trains first run through the simulation; the simulation is performed again with trains of a second-order priority which have to give way to the highest priority trains if conflicts occur. The process is repeated until all trains are included.

Synchronous models represent all the traffic moving around the network simultaneously - this is more realistic but presents challenges about how traffic management decisions are modelled. Such decisions include determining the priority of trains at a junction or selecting the route a train should take. Despite this challenge, most railway simulation

models are synchronous. Most synchronous simulations will require some input data to configure the traffic management decisions.

Failures and disruptions

Some simulation models can represent issues such as a signal failure or a passenger requiring assistance at a station. These will require input from the user to set the failure rate. The sophistication of the traffic management logic in a simulation model will determine the scale of disruptions that can be resolved.

2.2.4 Outputs

The output of a railway simulator may include a record of all the events that occurred during the execution of the model, and some software can also provide a visualisation of the activities. Such detailed outputs are of interest when evaluating a model (see Section 2.3) or engaging stakeholders.

Summary statistics are often more helpful as output, such as the number of trains that arrived late at their destination or the total number of minutes delays. Stochastic simulators can provide this type of output for each iteration or calculate the mean value for each statistic over all iterations. If a simulation runs sufficient iterations, then the mean values should converge.

2.3 Evaluation

2.3.1 Purpose of evaluation

Owing to the reductive nature of simulation models, they need some form of evaluation to ensure that they are fit for their intended purpose. Computer simulation models are assessed using two activities known as verification and validation. The activities carried out during verification and validation are not prescriptive and will require the judgement of modellers and subject matter experts.

A simulation model may never be considered 100% verified or validated (Carson, 2002). Therefore, it is best practice to document all verification and validation activities, which should be periodically revisited. As well as providing evidence that the model is fit for purpose, a transparent evaluation process may inspire confidence in decision-makers who rely on the model's output.

2.3.2 Verification

Verification is concerned with ensuring that the program has been implemented as designed (Sargent, 2013). General software engineering techniques, such as debugging and code walk-throughs, are standard approaches to verification. Other verification activities could include:

- Tracking objects through the simulation, for example, following a specific train through its journey.
- Carrying out parameter analysis to verify the intuitive relationships between input and output data, such as longer dwelling durations leading to longer overall travel durations.
- Inputting unrealistic values into the simulation, such as a very high number of trains entering the network, which can uncover issues with the implementation.

2.3.3 Validation

Validation is concerned with assessing if the model is sufficient for its purpose (Sargent, 2013). Validation activities may be more subjective than those carried out during verification. For example, subject matter experts may review the model to check if the model can represent the most important events or if the output appears intuitive based on experience. Of course, two experts may disagree in their assessments.

More objectively, it may be possible to run the model using historical data. In this case, the model's output can be compared with actual events. However, some judgement is still required to determine if the output is 'close enough' to the actual results.

2.4 Existing software

Simulations of railway operations can be created using generic simulation software, such as Simul8[1], Arena[2] or AnyLogic[3]. However, several software packages are also designed specifically for railways, which are used more often in the literature than generic software.

Network Rail use at least three railway-specific simulation tools: RailSys, TRAIL and VISION (Bradford et al., 2016). This section introduces these and other established commercial simulation software and briefly describes each model. The list is not exhaustive but focuses on those models that appear most frequently in the literature.

[1] https://www.simul8.com/
[2] https://www.rockwellautomation.com/en-us/products/software/arena-simulation.html
[3] https://www.anylogic.com/

RailSys

RailSys is a microscopic synchronous simulation model owned by RMCon[4] and is widely used throughout Europe (Radtke and Bendfeldt, 2001). It models the network to an accuracy of one metre and can be used to assess new infrastructure, for timetable construction or to analyse timetables and bottlenecks. RailSys provides exports of infrastructure data, and many other models are designed to be able to take inputs in the form of RailSys infrastructure data.

TRAIL

There is no report of TRAIL (Transport, Reliability, Availability, and Integrated Logistics) being used for studies other than in Great Britain. It was initially developed for the oil and gas industry (Rail Safety and Standards Board, 2014). TRAIL models the track in block sections and is categorised by Watson and Medeossi (2014) as a macroscopic simulation model; the higher level of detail makes it more suitable for modelling larger areas of the network than RailSys. Each block section in TRAIL is characterised by failure probabilities and repair characteristics (Watson and Medeossi, 2014), making it suitable for assessing system performance (Bransby and Barter, 2010).

VISION

VISION (Visualisation and Interactive Simulation of Rail Networks) is a microscopic synchronous simulation model developed for use on the British network. It models individual signals and track circuits (Watson, 2008) and is primarily used for assessing signalling performance (McGuire and Linder, 1994; Rail Safety and Standards Board, 2014). Reports in the literature suggest that it is only suitable for modelling small sections of the network, is computationally slow and has limited outputs (Medeossi, 2010).

OpenTrack

OpenTrack (Nash and Huerlimann, 2004) is a railway simulation software initially developed as a research project at The Swiss Federal Institute of Technology and is now maintained and licensed by OpenTrack Railway Technology Ltd[5]. It is a microscopic synchronous simulation model that, along with RailSys, is one of Europe's most widely used and well-established models (Watson and Medeossi, 2014). Reports in the literature describe the software as highly configurable, allowing a very flexible configuration

[4] https://www.rmcon-int.de/railsys-en/
[5] http://www.opentrack.ch/opentrack/opentrack_e/opentrack_e.html

of infrastructure and the ability to represent many different signalling systems (de Fabris et al., 2018; Tischer et al., 2020). The software can also animate a simulation and provides a wide range of textual and graphical outputs.

HERMES

Graffica[6] developed the HERMES (Holistic Environment for Rail Modelling and Experimental Simulation) software following their experience of simulating air traffic. It is a synchronous microscopic simulation and has been designed to be compatible with many infrastructure and timetable data sources. It can present both schematic and geographic animations of simulations.

SIMONE

The model SIMONE (SImulation of MOdel NEtwork) is an event-driven macroscopic simulation designed for use on the Dutch railway network. Its high-level representation of infrastructure makes it capable of simulating the whole of the Dutch network (Middelkoop and Bouwman, 2001). It is built using Incontrol Enterprise Dynamics[7] – a generic simulation software package

FRISO

FRISO (Flexible Rail Infrastructure Simulation of Operations) is a simulation model developed with the Dutch infrastructure manager ProRail (Middelkoop and Loeve, 2006). It represents infrastructure at the microscopic level. It has been designed to investigate traffic management and can connect to an external traffic management system. Like SIMONE, it is created using Incontrol Enterprise Dynamics.

RTC

RTC (Rail Traffic Controller) is a commercial microscopic simulation model developed by Berkeley Simulation Software[8]. It is primarily used to simulate freight operations in North America, where the characteristics of the railway network are different to typical European operations.

[6] https://www.graffica.co.uk/graffica-rail/software-solutions/hermes-simulator/
[7] https://www.incontrolsim.com/
[8] https://www.berkeleysimulation.com/rtc.php

2.5 Applications

2.5.1 Infrastructure

Simulation models can be used to inform business cases for designing new or upgrading existing infrastructure, which is typically relevant to the strategic planning horizon. Holgate and Lawrence (1997) used VISION to compare signalling systems. The authors simulated three-, four- and six-aspect fixed block signalling systems and a moving block system. They simulated two different track layouts: the first was a single 12km line, and the second was about 70km of double tracks. The results recorded the delay experienced by each train in the simulation, demonstrating the relative performance of the different signalling systems.

They found that as the number of aspects in a fixed block system increased, the number of delayed trains decreased, as did the maximum delay that any individual train incurred. A six-aspect fixed block system produced a comparable maximum individual delay to the moving block signalling; however, fewer trains experienced delays under the moving block implementation. The authors acknowledged that many other factors, such as technology or cost, influence the decision to implement a moving block signalling system. However, they stated that simulations are the only practical means of assessing performance before a change.

Quaglietta (2014) presented a framework for designing a fixed block signalling system layout using an optimisation algorithm combined with a microscopic simulation model developed for academic research. An objective function was defined as the minimisation of the total cost of a layout. The cost comprised the installation cost, maintenance cost and the number of lost train runs. One problem constraint was that blocks around all stations must be the same length, L. However, the blocks between stations could vary in length. There was also a minimum length between signals that must not be violated and some mandatory locations for signals.

The method was iterative, with the optimisation algorithm providing the length L at each iteration. The value of L is used to build a signal layout which is then simulated. The simulation performed many iterations to estimate the cost of the signalling layout. The optimisation algorithm then calculated the next value of L, and the process began again until the value of L converged.

Sogin et al. (2013) used RTC to simulate the transition from a single- to double-track layout in North America. They simulated different track configurations and reported on each configuration's effect on capacity and operating conditions. While it was known before the study that introducing a double-track layout would increase the route's capacity, the work compared strategies for introducing the double-track layout on an incremental basis to get the maximum benefit over time.

Haramina et al. (2018) demonstrated the benefits of upgrading a suburban single-track railway line to a double-track in Croatia. As well as adding the extra track, they also included an extra station on the line. The authors used OpenTrack to simulate a cyclical timetable on the proposed new layout. They simulated three scenarios: the first was the traffic without disturbances, the second with a temporary speed restriction on part of the line, and the third with a failure on a crossing near one of the stations. The simulation model demonstrated how a timetable could operate on the layout and the impact of some specific disturbances on the traffic flow. The authors stated that the single-track was also modelled, but the results were not presented, so the reader cannot quantify the improvements that the double-track might provide.

Jensen et al. (2017) presented a framework and model for assessing the capacity of railway networks at a strategic level. A simulation model was developed specifically for the study and is not named. The authors stated that timetable dependant simulation models are unsuitable for strategic capacity assessments, as a firm timetable will not have been agreed with stakeholders until a later stage in the planning process. Even though the framework presented by the authors does not rely on a pre-defined timetable, it does require some knowledge of routes and stopping patterns that trains may be expected to take. The framework created sequences of possible trains, and the schedules of the trains were compressed so that only the minimum headway was applied. The sequences of trains were then sampled randomly and run through a stochastic simulation.

The framework, with the simulation model, was applied to a case study in Denmark that looks at upgrading two junctions by adding a fly-over, thus eliminating some conflict locations on the infrastructure. The output quantified the improvement of the proposed infrastructure upgrades; the work showed that capacity consumption could be significantly lowered with the proposed upgrades. The results revealed that one of the junctions under consideration had a more significant benefit than the other, so the study determined the priority of the upgrades.

The use of simulation modelling to assist the design of an upgraded signalling system on existing infrastructure was the focus of research by Vignali et al. (2020). OpenTrack was used to simulate the layout of the new signalling system, such as the locations of the marker boards and axle counters. They illustrated their methodology through a case study for a section of the Danish railway network. The work demonstrated a key strength of simulation modelling, whereby the model can evaluate multiple designs before implementation.

Often investing in new or upgrading existing infrastructure is not feasible, and alternative strategies for utilising the infrastructure may be considered. Khadem Sameni et al. (2011) used the RTC model to assess the efficiency of various combinations of freight traffic in North America. Freight traffic dominates long-distance North American railways, which generally operate on-demand, i.e. without timetables. The authors created

a new metric that estimates the profit of freight train runs. They used the simulation model's output to calculate the metric for a given input of trains.

The simulation model and metric were applied to a case study of a single-track line in North America. The authors modelled two types of freight trains representing trains that can carry different types of commodities. The various scenarios that were modelled altered the ratio between the two train types (i.e. the heterogeneity of the traffic was changed) and the amount of traffic. The results achieve the best utilisation of capacity for the given metric when either one of the train types consists of 75% of the traffic. As the ratio between the train types became more even, the total delay and costs significantly increased, reducing the profit.

2.5.2 Timetabling

Simulation modelling applied to activities concerning the creation, optimisation, and analysis of timetables often occurs in the tactical planning timeframe. Many data are required to create a timetable, and some data may be derived or verified by simulation models. For example, VISION is used by Network Rail to calculate headways, junction margins and platform re-occupation times (Hyland et al., 2016). Network Rail's TPRs also states that simulation tools determine the sectional running times (SRTs) required for timetable creation (Network Rail, 2019b).

Gröger (2004) presented an asynchronous railway simulation that can create a timetable by simulating trains in order of priority. However, compiling a timetable has a very high algorithmic complexity, and a simulation model does not appear to be an ideal tool for the task. Alternatively, a deterministic simulation model could be employed to verify that a timetable is feasible by simulating trains without delays to ensure no conflicts exist (Watson and Medeossi, 2014). This could be useful on complex networks where significant timetable changes are to be deployed.

Kroon et al. (2008) presented a framework similar to that of Quaglietta (2014), whose work was discussed in Section 2.5.1. Kroon et al. (2008) developed an iterative optimisation algorithm that ran a simulation model at each iteration. The initial input to the model is a feasible cyclic timetable for a given network and probabilities regarding delay scenarios. The model aims to improve the timetable by minimising the average delays of the trains.

The simulation model utilised by Kroon et al. (2008) appears to have been developed specifically for the research and is not named; however, it is synchronous and stochastic. The optimisation in each iteration was solved using linear programming, and the output of the optimisation step was a new timetable, which the simulation model then ran. The process repeated until the optimisation output converged. The method was applied to a case study from a section of the Dutch network. The simulation model showed that the

average delay of the original timetable is higher and the punctuality at stations lower than the optimised timetable.

Mattsson (2007) used RailSys to study the behaviour of a section of the Swedish network following an exceptional infrastructure fault - a fire in an interlocking system which occurred in the year 2000. Following the fault, traffic on the line was reduced as repairs were carried out. The study showed how the average delays would have been affected if fewer or more trains had been allowed to run. For validation, they also simulated the number of trains that did run. The results demonstrated the positive relationship between capacity utilisation and average delay and could be used to inform future scheduling decisions.

Marinov and Viegas (2011) used Simul8 (a generic simulation package) to develop a stochastic mesoscopic model of freight operations in a rail network. Freight services operate differently than passenger services. A passenger service will depart at its scheduled time even if no passengers are on board. However, if a freight train has not loaded with its stock, it may be delayed or even cancelled. The authors presented a case study on the Portuguese railway network, which investigated the performance at freight yards. They ran two scenarios through their simulation model. The first scenario comprised freight trains making only minor deviations from their schedules, and the second scenario allowed large deviations sampled from an exponential distribution. Comparing the results of the two scenarios shows that when freight trains deviated significantly from their schedule, they spent longer on average queuing in freight yards.

Vromans et al. (2006) combined simulation modelling with analytical methods to investigate the relationship between the heterogeneity of cyclical timetables and delays. They present two measures of heterogeneity; the first is the sum of shortest headway reciprocals (SSHR):

$$\text{SSHR} = \sum_{i=1}^{n} \frac{1}{h_i^-} \tag{2.1}$$

where h_i^- is the smallest scheduled headway between trains i and $i+1$ on the track section, and there are n timetables. As the timetable is cyclical, timetable n is assumed to be followed by the first timetable. A lower SSHR means there is more homogeneity than a high value. This measure penalises headways at departure as heavily as headways at arrival. However, headways at arrival seem to be more critical than at departure, which motivates the second measure presented in the paper, the sum of arrival headway reciprocals (SAHR):

$$\text{SAHR} = \sum_{i=1}^{n} \frac{1}{h_i^A} \tag{2.2}$$

where h_i^A is the headway at arrival between trains i and $i+1$ on the track section, and there are n timetables. Neither are absolute measures, but they enable comparison between timetables. The authors demonstrated the validity of the metrics using a case study involving the simulation of two theoretical timetables. Both timetables have the same number of trains, but one has a realistic mix of trains (and is heterogeneous), and the other timetable was altered to be more homogeneous. The SSHR and SAHR both decrease for the homogeneous timetable.

This research simulated the timetables using the software SIMONE and ran 16 experiments with varying levels of direct delays for both timetables. The probability of a delay occurring and the size of the mean delay time varied between experiments. The results demonstrated a significant decrease of 65% in the average delay in the homogeneous timetable across all experiments. They also demonstrated a significant decrease in reactionary delays in the homogeneous case. The simulation's output also showed that a few large direct delays impacted overall punctuality more than many small direct delays. The work demonstrated that the metrics could be used to make an analytical assessment of the timetable, with lower values more likely to result in fewer delays than higher values.

Lindfeldt (2011) also used the SSHR and SAHR metrics to measure the heterogeneity of timetables in an investigation of reactionary delays. The work used RailSys to model a variety of scenarios that have different timetables and heterogeneity levels. They simulated the scenarios with direct delays applied to entry times (to the simulation), running durations, and dwell durations. The results showed that secondary delays were longer for timetables with higher SSHR and SAHR.

A desirable quality of a timetable that frequents the literature is that of robustness, although there is no agreed definition of the term. Andersson et al. (2013, p. 95) define a robust timetable as 'one in which trains are able to keep their original train slots despite small primary[9] delays and without causing unrecoverable delays to other trains'. Cacchiani and Toth (2018, p. 93) state that robustness aims to determine 'timetables that "perform well" under disturbances, avoiding delay propagation as much as possible'.

One approach that can increase the robustness of a timetable is to insert time supplements into the schedule. Increasing the scheduled travel duration of a train may allow minor delays to recover without propagating between services. However, such strategies need careful implementation as they carry additional consequences. Time supplements increase journey times and reduce the network's available capacity (Lusby et al., 2018), potentially impacting passenger satisfaction (Transport Focus, 2020). There is, therefore, a trade-off between optimality and robustness, known as the 'price of robustness' (Salido et al., 2008; Cacchiani and Toth, 2018).

[9] A primary delay is alternative terminology for a direct delay.

Takeuchi et al. (2007) created a passenger-focused robustness index. The index they developed is a measure of passenger disutility representing the discomfort index for each passenger on a train. The index is naïve in that it assumes longer waiting times for passengers making a connection is always worse; having a shortened connection time (due to a delayed incoming train) can be equally disruptive and stressful. They employed a stochastic simulation to calculate the index for the current timetable for a case study area on the Japanese railway network. The simulation model is unnamed and capable of representing passenger behaviour as well as train dynamics. The authors then used the simulation model to demonstrate how an altered timetable can lead to a reduction (i.e. an improvement) in the index. These results could feed back into the timetable creation process to develop a timetable that might lead to higher passenger satisfaction.

Hyland (2012) presented a study to understand the impact of a new conceptual timetable on a section of the British network. Two variations of a new timetable were investigated; one variation had two fewer trains per hour than the other, but both had more trains per hour than the existing timetable. Two simulation models were used in the study. RailSys was used to validate the new timetables and provided the train schedules as an export. TRAIL was employed to model the performance of the existing and new timetables; the RailSys timetable exports were used as input by TRAIL.

The primary metric for the study was the percentage of services that arrive at their destination (or model boundary) within ten minutes of their scheduled arrival time. The output from TRAIL showed a close correlation with actual performance when the existing timetable was simulated. Around 85% of services arrived within ten minutes of their scheduled time for the existing timetable. This reduced to around 67% for both versions of the new timetable. The model also predicted a higher percentage of services that would be cancelled using the new timetables compared with the existing timetable. The results revealed that it is not possible to maintain the existing levels of performance with the new timetables without any further measures, such as improving reliability.

2.5.3 Operations

The papers discussed in this section consider modelling operational rules and traffic management decisions. Some studies may inform the planning activities in the operational horizon; however, others may fall within the tactical timeframe, given that they do not consider real-time planning.

Liu et al. (2013) used a mesoscopic simulation model to challenge established rules of railway operation on the British network. The study investigated the impact of fundamental rule changes as trains approach a signal showing a yellow aspect. The current practice is conservative, with the train expected to decelerate at the sight of the signal and to be prepared to stop. Three alternative scenarios are constructed, which

change this rule. Two of the scenarios delay the point at which the train decelerates. The last scenario is extreme as the train does not decelerate at all on the assumption that the conflicting route will be cleared in time.

The authors stated that existing railway simulation models cannot study such rule changes. Therefore, a time-driven, synchronous mesoscopic simulation model of road traffic was adapted to the railway domain. The scenarios were applied to a model of a railway junction where a double-track route diverges into two, leading to one track crossing the path of another.

In addition to modelling different rules, the headways between the trains were varied to determine the earliest time the second train to pass the junction could be scheduled for each rule. As the simulation is deterministic, the speed, acceleration and deceleration do not vary between runs, although the location of the deceleration varies between the operating rules. The simulation demonstrated that the less conservative rules enable shorter headways between the trains, increasing capacity at the junction. The authors acknowledge that further work is required before such rule changes are applied due to safety concerns.

Hofman et al. (2006) employed simulation modelling to investigate three strategies for recovering following a delay:

- Early turn-around
- Insertion of on-time trains (limited to one train per direction and only at the main stations)
- Cancelling of entire train services

They investigated the strategies using a model built with the generic simulation software ARENA. Experiments included running 12 timetable scenarios without recovery and then with the three strategies. The measure they used to compare the effectiveness of the recovery methods is regularity which they calculate as:

$$\text{Regularity} = 100 \times \left(1 - \frac{\text{Late departures}}{\text{Departures in Total}}\right) \tag{2.3}$$

A high regularity value is desirable. The 'turn-around' and 'cancellation' strategies provided similar levels of regularity for timetables with a low number of services. The 'cancellation' strategy provided the largest regularity values for timetables with a high number of train services. The strategy of inserting trains appeared to be the least effective method.

Middelkoop et al. (2013) used FRISO to compare traffic management strategies around a station on the Dutch network that is a capacity bottleneck. They simulated three

scenarios that represent different methods for resolving conflicts. The first was the current practices of train controllers in the area, which was that scheduled order is maintained when trains travel on the same destination track and, otherwise, a first-come, first-served priority is established. The second scenario was a first-come, first-served priority for all conflicts, regardless of the destination track. The third scenario was a resolution determined by an external traffic management system (TMS), which could connect to FRISO. The TMS could predict conflicts and optimise traffic in real-time using heuristics. Resolutions the TMS could employ include altering the priority of trains at junctions and using alternative routes. It could also change the advisory speed of a train as it approaches conflict locations.

The authors simulated six hours for 25 iterations for each scenario; they did not discuss why they only considered 25 iterations – it appears to be a low amount to generate statistically meaningful results. The results showed that the average delay per train is lowest for the TMS scenario, and the punctuality is highest compared with the other scenarios. The authors stated that future work involves realising the benefits of the TMS optimiser in real-time operations.

Quaglietta et al. (2016) considered the real-time automation of traffic management on a railway network. The approach compared two conflict resolution algorithms, ROMA (D'Ariano and Pranzo, 2009) and RECIFE (Pellegrini et al., 2016), which can resolve conflicts in real-time. The algorithms are used to manage traffic in a simulation model. The authors used the simulator HERMES and experimented with three case studies on sections of European networks: the East Coast Main Line in the UK, the Utrecht–Eindhoven–Tilburg–Nijmegen corridor in the Netherlands and the Iron Ore line at the border between Sweden and Norway.

The results of the ROMA and RECIFE algorithms were compared to the scenario where conflicts are resolved by running trains in their scheduled order through junctions. The total departure and maximum delays were significantly lower when traffic management algorithms were applied, compared to prioritising the trains according to their scheduled order. However, the ROMA and RECIFE algorithms exhibited different behaviour when resolving conflicts. The results do not compare the actions that signallers would have taken (assuming they would employ more sophisticated decisions than maintaining the scheduled order of trains), so it is unknown if the conflict resolution algorithms could outperform experienced signallers.

Gaming simulations are characterised by human interaction during the modelling process and, owing to this, are reported to have high stakeholder engagement across various subject domains (Roungas et al., 2018). A railway gaming suite has been developed in conjunction with the Dutch infrastructure manager ProRail, which uses the simulation model FRISO (Meijer, 2012). An interface was developed to allow human signallers to control the traffic in the simulation model.

Middelkoop et al. (2012) apply the gaming suite to a case study on the Dutch network. The modelled scenario concerns the network around a station with planned infrastructure upgrades. The available track around the station will change several times during construction, leading to less capacity and amended timetables. An interface was developed to allow human signallers to control the traffic in the simulation model. The case study evaluated operational rules for controlling traffic during the infrastructure upgrade. However, the paper focused on the digital infrastructure required to run a gaming simulation and did not provide detailed case study results.

2.6 Discussion

2.6.1 Strengths and weakness

An evident strength of railway simulation modelling is that such models can be applied to problems and scenarios that would be costly or impossible to experiment with in real life. The literature review gave many examples, such as upgrading infrastructure or analysing a new timetable. To this end, simulation is particularly well suited for supporting decision-making in the strategic and tactical planning horizons.

Simulation models also have the potential to be understood by many different stakeholders in the railway industry without requiring them to have an understanding of mathematical modelling. Simulation software that provide visual representations of the underlying model can help increase stakeholders' confidence. However, modellers also have a role to play by communicating clearly to stakeholders in order for them to understand the limitations of a model. Bransby and Barter (2010) evaluated a project undertaken to analyse the performance of a new timetable using RailSys and TRAIL. They found that the outputs of the models were, to an extent, misunderstood by decision-makers, reporting that some believed the models would directly forecast the performance levels. This misunderstanding led, in part, to the introduction of a timetable that performed worse than expected.

Analytical models may provide an alternative approach to simulations for some problems. Some examples of analytical methods include an analytical model of train delay propagation at stations and junctions (Yuan and Hansen, 2007) and the use of max-plus algebra to assess the capacity of junctions or stations (Bešinović and Goverde, 2018). Like simulation models, analytical models will also include assumptions and simplifications and require evaluation. Decision makers may potentially find analytic models more abstract and challenging to understand without some mathematical knowledge. Bešinović (2020) states that analytical measures of robustness cannot capture the operational dynamics of a railway system; however, Yuan (2007) argues that simulation

generally offers less insight into structural relations between input and output than analytical models. Mattsson (2007) observes that analytical methods do not require as much input data as simulation models and can often be applied without an exact timetable, making them useful for strategic decisions where timetables may be unknown.

Microscopic simulations require large amounts of input data to create a detailed representation of the networks and the trains. This requirement for data can be a significant overhead in any project using a microscopic simulator, as compiling the input data can be very time-consuming. Also, the more input data that are used, the more potential sources of error in a model and extra time is required for verification and validation. There will also be an impact on the computational time spent running the simulation model compared with a macroscopic model. While macroscopic models are better suited to running large sections of a network than microscopic models, their lack of detail limits the results that can be obtained.

Cui et al. (2018) observe that commercial simulation tools are not always transparent regarding the assumptions that the models make, and there is a lack of published literature on the workings of commercial railway simulation software. They state that existing software does not sufficiently document the internal workflows, such as how the logic of the signalling system or traffic management algorithms.

Medeossi and de Fabris (2018) review existing simulation models and recognise that most models are microscopic and synchronous. They discuss weaknesses common to such models and identify two primary root causes: the stochastic nature of train movements and traffic management decisions are not fully represented. These two issues are discussed in the following subsections.

2.6.2 Stochastic train movements

While deterministic run and dwell durations are used to construct railway timetables, realised train dynamics are stochastic, and trains do not run precisely to their scheduled times. Driver behaviour, traffic conditions, vehicle performance or weather conditions can all cause variation.

All established simulation models use motion equations to model train movements (Medeossi and de Fabris, 2018). Variation in train run durations can be represented by sampling from a probability distribution of performance parameters for each phase of the motion equation. Estimating suitable parameters is complex (Medeossi and de Fabris, 2018); however, studies have shown how suitable distributions can be determined for each phase of motion using train-based (Medeossi et al., 2011) and track-based (Bešinović et al., 2013) data. In addition, existing commercial software only allows *one* performance parameter for the entire motion (Medeossi and de Fabris, 2018), and a single performance

factor across all phases of motion cannot fully represent the stochastic nature of train run durations (de Fabris et al., 2008).

Even if the method could be applied in simulation models, sampling from the probability distributions for each phase of motion is random and not conditional on the situation of the train. For example, Bešinović et al. (2013) acknowledge that delayed and on-time trains may have different performance parameters for many phases of motion: a delayed train may be more likely to cruise closer to the maximum permitted speed. Similarly, there may be relationships between the phases of motion: a driver cruising significantly below the maximum permitted speed may also be cautious under braking.

Regarding dwell durations, passenger trains that stop at a station exhibit different characteristics depending on whether they are running ahead or behind their schedule. Early-running trains dwell until the departure time, and there are also departure imprecisions whereby trains do not leave at the exact scheduled time (Watson and Medeossi, 2014). Conversely, late-running trains will typically dwell for the minimum amount of time necessary. Therefore, early-running trains can be modelled by looking at the variation in departure time compared to the scheduled departure time. Late-running trains can be modelled by variations in minimal dwell durations. However, most commercial simulation models cannot model the two phenomena (Medeossi and de Fabris, 2018), and the only option is to select a single distribution to represent all variability for stopping times at stations.

Moreover, passenger train dwell durations are affected by the number of passengers boarding and alighting at a station. In turn, the number of passengers can be influenced by the station, the time of day (peak or off-peak travel), and the time a train previously stopped at the station. Most simulations implicitly model passengers and cannot fully represent all the factors influencing passenger numbers and dwell duration. For example, a single distribution may be selected for dwell duration variability for a group of similar stations at peak times, and another distribution may be selected for off-peak times (Watson and Medeossi, 2014). However, identifying 'peak' times, similar groups of stations and suitable distributions all add to the time it takes to collate the input data.

The impact of the existing modelling methods for run and dwell durations means that the full variation of both is not realised. As discussed, both run and dwell durations are *conditional* on environmental, physical and human factors often not explicitly modelled in railway simulations. The existing methods for modelling train movements result in trains moving more consistently around a simulation than they would in reality.

Infrastructure managers, such as Network Rail, could benefit from a railway simulation that could model realistic variations in the run and dwell times of trains. Such a simulation would have the potential to give a more realistic understanding of the performance of a new timetable. It could also be used to examine the likelihood of conflicts

occurring at different junctions with a new timetable, providing the opportunity to feed amendments into the timetabling process.

2.6.3 Traffic management logic

Traffic management algorithms are crucial to synchronous rail traffic simulations but are challenging to develop (Sipilä, 2015). A realistic representation of traffic control is required if a simulation model is to be used to assess infrastructure or a timetable; simplistic representations of signalling decisions can lead to deficiencies in a simulation's output.

There are many types of actions that signallers take daily to manage traffic. These generally aim to avoid conflicts and could be determining the priority of trains at a junction or re-routing trains through busy stations. More significant interventions may be required for severe disruptions. Such decisions can involve changing a train's schedule, such as deciding a late-running train should not stop at scheduled stations, cancelling a service before it reaches its destination, or even creating an unplanned train service.

No known examples of microscopic simulations have sufficiently sophisticated traffic management algorithms to model severe disruptions (Medeossi and de Fabris, 2018). Bešinović (2020) states that simulation models tend not to apply realistic traffic management decisions and are therefore not suitable for modelling more than a limited number of disruptions. Moreover, studies using simulation models often limit the direct delay distributions to avoid deadlocks (Sipilä, 2015). Deadlocks in a simulation occur when there is a conflict that the program cannot resolve (Pachl, 2011; Mazzanti et al., 2014), for example, if two trains are running toward each other down a bidirectional section of track.

However, synchronous simulators must have some algorithm to make train regulation decisions that set priorities at conflict locations on the network; otherwise, interlocking rules would be violated. An example of a strategy for setting the priority of trains at a junction is 'first come, first served', whereby the first train to arrive at a junction has priority (McGuire and Linder, 1994). Such a strategy is simple to implement but not very sophisticated and does not mimic the complexity of the behaviour of experienced signallers.

In reality, signallers follow more complex recovery strategies; they may follow train regulation policies, but these may vary by area or region. Some tools, such as OpenTrack, allow the setting of logic at the signal level; however, it could be a considerable task to determine and validate logic for every signal for a large section of the network. As discussed in Section 2.5.3, Middelkoop et al. (2013) and Quaglietta et al. (2016) both demonstrate how simulation models could be used to assess the performance of real-time conflict resolution algorithms. However, these external decision support systems seek to

optimise decision-making rather than modelling existing practices. Middelkoop et al. (2013) also modelled current signalling practices at a Dutch station, but these may not generalise to another station or junction, and there was no discussion in the paper about how strictly the signallers followed the rules.

There does not appear to be any discussion in the literature that considers the consistency in signalling decisions, let alone how to represent such variation in a simulation model. There is no guarantee that two signallers would take the same action when presented with an identical scenario, as the experience of a signaller will play a role in the decision-making.

Network Rail, or any infrastructure manager, could benefit from a simulation that could model actual signaller behaviour, including any inconsistencies that may be inherent in such behaviour. Network Rail set train regulation policy, and such a simulation would allow them to compare alternative policies with current signaller behaviour. It may also be helpful to assess expected behaviour against current behaviour to see if signallers are able to make improvements on set policies given their experience. Such a simulation could also be beneficial for analysing the performance of new timetables if they could realistically model the actions signallers take rather than how train regulation policies state they should behave.

2.7 Conclusion

This chapter has provided background on the simulation of railway operations and has discussed the range of features such models can have. As models are only ever representations of reality, they require some degree of abstraction and methods for evaluating simulations were presented. A selection of established simulation models was introduced, followed by examples of how these models have been applied in the literature.

The literature review demonstrated that one strength of railway simulations is their ability to model scenarios that are not feasible to experiment with in reality. Examples of such scenarios typically occur in the strategic and tactical planning horizons concerning infrastructure or timetable changes. They can also capture the dynamics of operations that may not be realised using analytical methods, and decision-makers can understand the approach as a simulation mimics reality.

Some general weaknesses of simulation models were discussed in this chapter, such as the time they can take to set up or the potential for misinterpretation of the output. Two specific shortcomings concerning the implementation of railway simulations were discussed in more detail: representing the dynamics of train movements and traffic management logic. The established method for modelling train movements using motion equations fails to represent the stochastic nature of train movements. Setting up a

simulation model or building an algorithm to represent realistic traffic management decisions is challenging. Existing methods for implementing traffic management logic also do not consider any uncertainty in the decision-making.

Limitations in the representation of train movements and traffic management decisions have a negative impact on a simulation model's ability to mimic reality. Improvements in these areas would be beneficial to an infrastructure manager as they would have the potential to improve the assessments of a new timetable or new train regulation policies. This thesis will focus on these two weaknesses, and the next chapter will look at supervised machine learning as an alternative method for modelling these features.

Chapter 3

Supervised Machine Learning

3.1 Introduction

The previous chapter introduced operational railway simulation modelling and identified two limitations of existing software concerning train movements and traffic management. Both phenomena should, to some extent, be predictable if enough information is provided. For example, timetables and operational rules (such as speed restrictions) could be used to make predictions of travel durations between stations. Information such as how late a train is running could also inform a prediction – late-running trains may travel closer to the maximum permitted speed than early-running trains. Similarly, signallers will follow train regulation policies to resolve many conflicts: for example, passenger trains may often be granted priority over freight trains.

Supervised machine learning is a modelling approach that makes predictions by identifying patterns from past data. This chapter will present supervised machine learning as an alternative approach for predicting train movements and traffic management decisions. It will also explore the use of supervised machine learning models to drive events within simulations. Other types of machine learning, such as unsupervised and reinforcement learning, are outside the scope of this thesis and are not discussed in this chapter. For the remainder of this thesis, the reader may assume that any mention of 'machine learning' is about supervised machine learning unless otherwise specified.

Section 3.2 provides background information on supervised learning, covering terminology and concepts necessary for this thesis. Information on different supervised machine learning algorithms follows in Section 3.3. An exhaustive list of algorithms is not provided; only methods that appear in the reviewed literature and this thesis are presented.

Section 3.4 presents a literature review covering machine learning applications for predicting travel durations, dwell durations and traffic management decisions in the railway and other transport domains. Having established that supervised machine learning is

an approach that can model train movements and signalling decisions, Section 3.5 introduces machine learning-assisted simulation modelling. This is a hybrid approach where machine learning models are deployed into a simulation model and used to influence the dynamics of the simulation. Section 3.6 discusses the potential of the methods covered in this chapter and the gaps in the literature.

3.2 Background

3.2.1 Supervised learning

Machine learning is a broad sub-field of artificial intelligence, of which supervised machine learning is but one approach. Given a set of input and output pairs, $(x_1, y_1), ..., (x_N, y_N)$, a supervised machine learning model aims to find a function f that maps each input value x_i to its output y_i (Hastie et al., 2009b). In this work, the inputs will be called feature vectors, and the outputs will be target values. A pair of input and output values (x_i, y_i) is a record, and the output of $f(x_i)$ is a prediction, denoted as $\hat{y}_i$.

Supervised machine learning is a contrasting approach to simulation modelling. Simulation models require rules to be programmed or input before running the model. In comparison, a supervised machine learning model provides rules as output, which can then be applied to another data set to make predictions. Figure 3.1 visualises the different approaches between the two modelling paradigms.

3.2.2 Types of prediction

The output of a model can be discrete or continuous. Models that predict discrete target values are called classification models, whereas regression models predict continuous values. For example, a model that predicts whether a train will arrive late at its final destination is a classification problem. Alternatively, the problem could be re-formulated as a regression problem if the model predicts the number of minutes a train will arrive late.

3.2.3 Datasets

Before any modelling begins, the data are typically divided into three sub-sets known as training, validation and test data sets (Chollet, 2017). The training data are used to build the model (the process is explained in Section 3.2.6). The model is then evaluated on both the training and validation data. The model has not yet 'seen' the validation data, and the evaluation score will indicate how well the model can generalise to new data. The process is often repeated many times to find a suitable model.

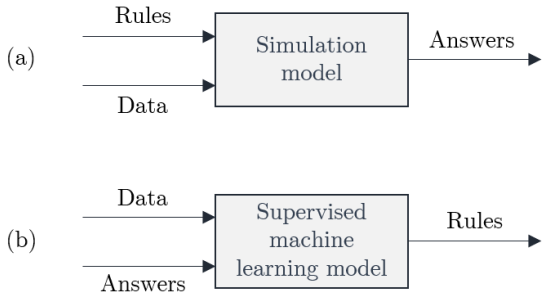

FIGURE 3.1: A simplified visualisation of the main differences between (a) a simulation model, and (b) a supervised machine learning model, adapted from Chollet (2017)

The validation data cannot provide a *final* assessment of a model, as the evaluation of the validation data will inform the model's configuration. It is said that the information about the validation data 'leaks' into the model. Therefore, validation data are only used for model selection. Once a model has been selected and built, the model can then be applied to the test data. This will determine how well the model performs on unseen data.

3.2.4 Data pre-processing

A set of pairs of feature vectors and target values will usually require pre-processing before being used to train a model (Kuhn and Johnson, 2013a). Pre-processing activities may include:

- Handling missing values
- Scaling features (some algorithms require that the features are on a similar scale or come from a normal distribution)
- Encoding categorical features
- Removing redundant features
- Transformation of features
- Creation of new features

The term feature engineering may be used to describe the process of manipulating feature vectors through transformations.

3.2.5 Types of model

Machine learning models are parametric when the mapping function takes a known form before fitting the data. An example of a parametric model would be a linear function of

the form:

$$f(x) = \theta_0 + \theta_1 x$$

where θ_0 and θ_1 are to be determined. Parametric models are sometimes classed as statistical rather than machine learning methods (Jiang et al., 2020).

Non-parametric models still have parameters to learn, but the number of parameters is not fixed in advance as such methods do not pre-determine the form of the mapping function. Non-parametric models are less restrictive than parametric models and can produce more complex, non-linear relationships in the data.

All but the simplest models have hyperparameters that allow configuration of the model; they are so named to distinguish them from the parameters the model will learn. Complex algorithms may have many hyperparameters, which need to be tuned during the modelling process to find optimal values.

Ensemble methods make predictions using multiple individual models. An example of a hyperparameter for an ensemble is the maximum number of models from which it can be constructed.

3.2.6 Fitting the model

Algorithms rely on optimisation to identify the configuration of parameters to fit a function to the training data. The optimisation process utilises a loss function, $\mathcal{L}$, to measure the difference between a prediction and its target value. There are many forms the loss function can take (Wang et al., 2020), with regression and classification problems requiring different loss functions. An example of a loss function that is suitable for a regression problem is the squared error:

$$\mathcal{L}(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2 \tag{3.1}$$

A loss function that may be employed for binary classification (where y_i can only take values of 0 or 1) is the logistic loss:

$$\mathcal{L}(y_i, \hat{y}_i) = -(y_i \log \hat{y}_i) + (1 - y_i) \log (1 - \hat{y}_i) \tag{3.2}$$

The loss function only measures the accuracy of the model for one record. A cost function is the sum of the loss function for all records in a dataset. An optimisation algorithm will seek to minimise the cost function using the training data.

A standard optimisation algorithm is gradient descent, and the reader is referred to any introductory textbook on machine learning, such as Chollet (2017), for a description of

the process. In practice, gradient descent is not guaranteed to find the optimal solution if the cost function exhibits local minima. There have been many optimisation algorithms that are improvements on gradient descent, some of which will be referenced in Section 3.3.5.

3.2.7 Evaluation

A measure of success is necessary to evaluate the quality of a machine learning model. The metric will assess the quality of the mapping from the feature vectors to the target values. As with the loss function, appropriate evaluation measures will differ according to whether the model is classification or regression. Examples of metrics for regression models include the mean absolute error (MAE), the mean squared error (MSE), the root mean squared error (RMSE), the mean absolute percentage error (MAPE) and the coefficient of determination (R^2):

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.3}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.4}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.5}$$

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{|y_i|} \tag{3.6}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.7}$$

where y_i is the true value of the i^{th} record, $\hat{y}_i$ is the predicted value, and $\bar{y}$ is the mean value of all the y_i values. A lower value is 'better' in the case of MAE, MSE, RMSE and MAPE, and a higher value for the coefficient of determination.

Some examples of metrics for binary classification models include accuracy, precision and recall:

$$\text{Accuracy} = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \tag{3.8}$$

$$\text{Precision} = \frac{T_p}{T_p + F_p} \tag{3.9}$$

$$\text{Recall} = \frac{T_p}{T_p + F_n} \tag{3.10}$$

where the two possible target values are referred to as positive and negative, T_p is the number of positive records correctly classified (true-positive), T_n is the number of negative records correctly classified (true-negative), F_p is the number of positive records incorrectly classified (false-positive), and F_n is the number of negative records incorrectly classified (false-negative).

If the amount of available data is relatively small, then an alternative evaluation method is to apply cross-fold validation (Bergmeir and Benítez, 2012). In cross-fold validation, the test data is still set aside, and the remaining data are randomly partitioned into k sets. The training process then takes place over k iterations, with one of the k sets held out as the validation data in each iteration. Typically, an average of the evaluation metric over all validation sets is used for model selection.

An evaluation metric calculated for a model does not provide enough information to assess a model. The quality of a model can only be determined relative to other models. For this reason, it is best practice to have a benchmark model created by some means other than machine learning to provide a value for the metric that any machine learning model must exceed if it is considered 'good'.

3.2.8 Uncertainty

Supervised machine learning models are not guaranteed to make accurate predictions; there will be sources of error in all but the simplest of problems. The uncertainty of predictions can sometimes be reduced by including additional knowledge or training data. However, there may be data records where the uncertainty is irreducible, and no additional knowledge or data exists to improve the prediction, and it is often advantageous to quantify the uncertainty of predictions.

For classification problems, uncertainty can be represented by predicting a probability for every possible target value. A classification model is said to be well-calibrated if the probability predictions correspond to true probabilities, and there are methods available to improve the calibration of a classifier. An exploration of uncertainty in regression problems often involves predicting a conditional probability distribution instead of making a point prediction.

3.3 Approaches

3.3.1 Linear methods

Linear regression

Linear regression is an example of a parametric supervised learning model where the prediction takes the form:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

Where $\hat{y}$ is the predicted value, n is the number of features, x_i is the ith feature, and θ_j is the jth model parameter to be learned. Linear regression models can be solved exactly using linear algebra if the cost function is the squared error (see equation 3.1), although this becomes more computationally complex as the number of features increases.

Logistic regression

Despite its name, logistic regression is most commonly used for binary classification problems where there are only two possible target values (Kuhn and Johnson, 2013a). If there are n features, x_i is the ith feature, and θ_j is the jth model parameter to be learned, then the prediction takes the form:

$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5 \\ 1 & \text{if } \hat{p} \geq 0.5 \end{cases}$$

where:

$$\hat{p} = \sigma(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n)$$

The sigmoid function provides an output that has a value between zero and one. For any parameter t it is defined as:

$$\sigma(t) = \frac{1}{1 + e^{-t}} \tag{3.11}$$

3.3.2 K-nearest neighbours

K-nearest neighbours is a non-parametric model that can be applied to regression and classification problems (Altman, 1992). It makes predictions based on the k closest

samples from the training data, where k is a hyperparameter to be determined during the model selection process. Another hyperparameter to be set is the definition of 'distance' between points. One standard definition of distance is the Minkowski distance:

$$\left(\sum_{j=1}^{P} |x_{aj} - x_{bj}|^q \right)^{1/q}$$

Where $x_a = (x_{a1}, \ldots, x_{aP})$ and $x_b = (x_{b1}, \ldots, x_{bP})$ are two records consisting of P features, and the value of q is a hyperparameter to be determined.

In the case of classification, the result is the most common class among the k-nearest neighbours. For regression, the output value is the mean of the values among the k-nearest neighbours. If the training dataset is large, then k-nearest neighbours can be computationally intensive as the distance between each record is calculated.

3.3.3 Ensembles of decision trees

Decision trees

A decision tree is an example of a non-parametric model, where the data are recursively partitioned to make predictions (Hastie et al., 2009a); they can be used for both classification and regression predictions. Decision trees have the advantage of being simple to understand and visualise. One means of visualisation is of nodes representing the data's partitions, as shown in Figure 3.2.

The first node is the root node, and the final nodes in the tree are called leaf nodes. Each leaf node contains the training data that end up in that node. For classification, the prediction is made based on the most frequent target value in the node, and for regression problems, the mean of the target values is calculated.

There are different algorithms for building decision trees; one common algorithm is the classification and regression tree (CART) algorithm. The algorithm recursively splits the data in each node by a single feature, k, and a threshold for the feature, t_k. It determines the feature and threshold based on the values that produce the lowest value for the chosen cost function.

Examples of hyperparameters of a decision tree are:

- The maximum depth of the tree
- The minimum number of records in a node before splitting
- The minimum number of records in a leaf

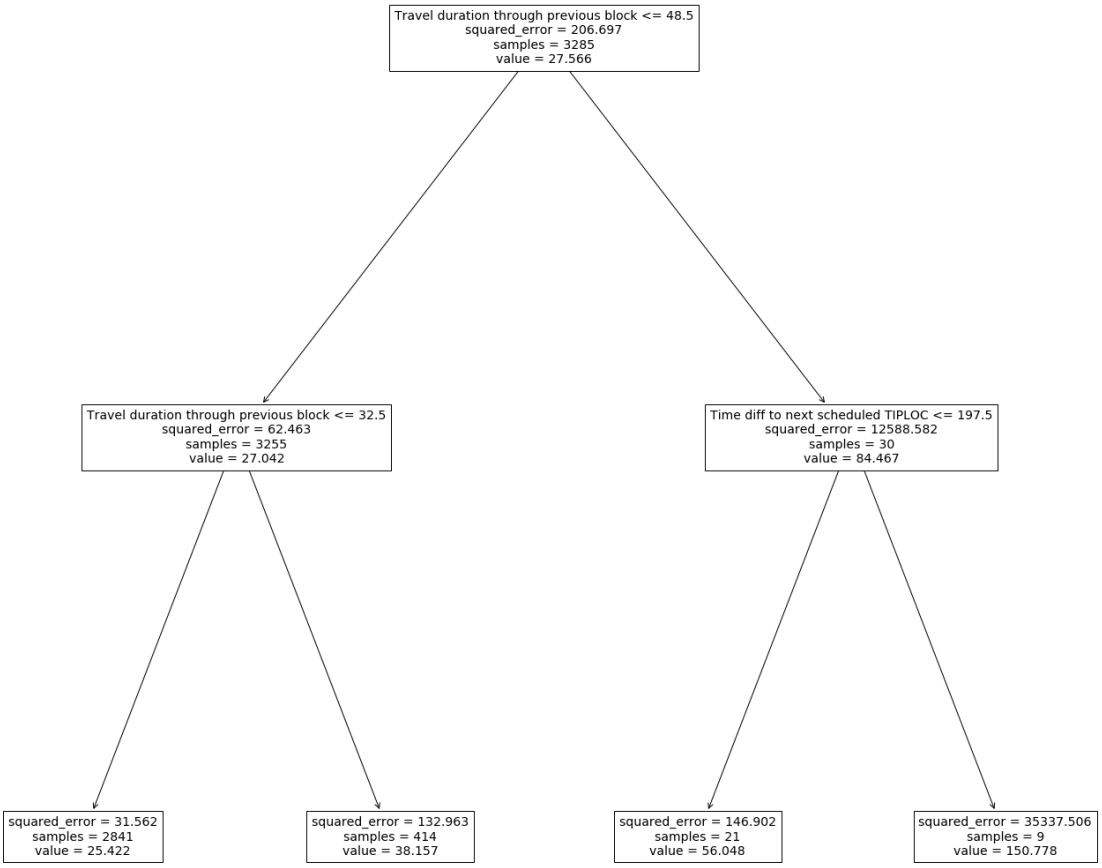

FIGURE 3.2: An example of a decision tree used to make predictions about travel duration through a block of the railway based on two features: the travel duration through the previous block and the time difference to the next scheduled TIPLOC

A single decision tree rarely makes sufficiently good predictions, and ensembles of trees are more frequently applied in the literature.

Random forests

Random forests are an ensemble of decision trees that can be applied to classification and regression problems. These algorithms implement a method known as bootstrap aggregating or bagging, where many decision trees are built independently with random samples of the input data (Breiman, 1996). The result then comes by taking an average of the prediction over all the trees.

Gradient tree boosting

Gradient tree boosting is another ensemble of decision trees and is also suitable for classification and regression. A gradient tree boosting algorithm trains a sequence of decision trees, with each tree fitting the residual errors of the previous model (Friedman, 2002). A prediction is made by summing the predictions of all the trees. Whereas the

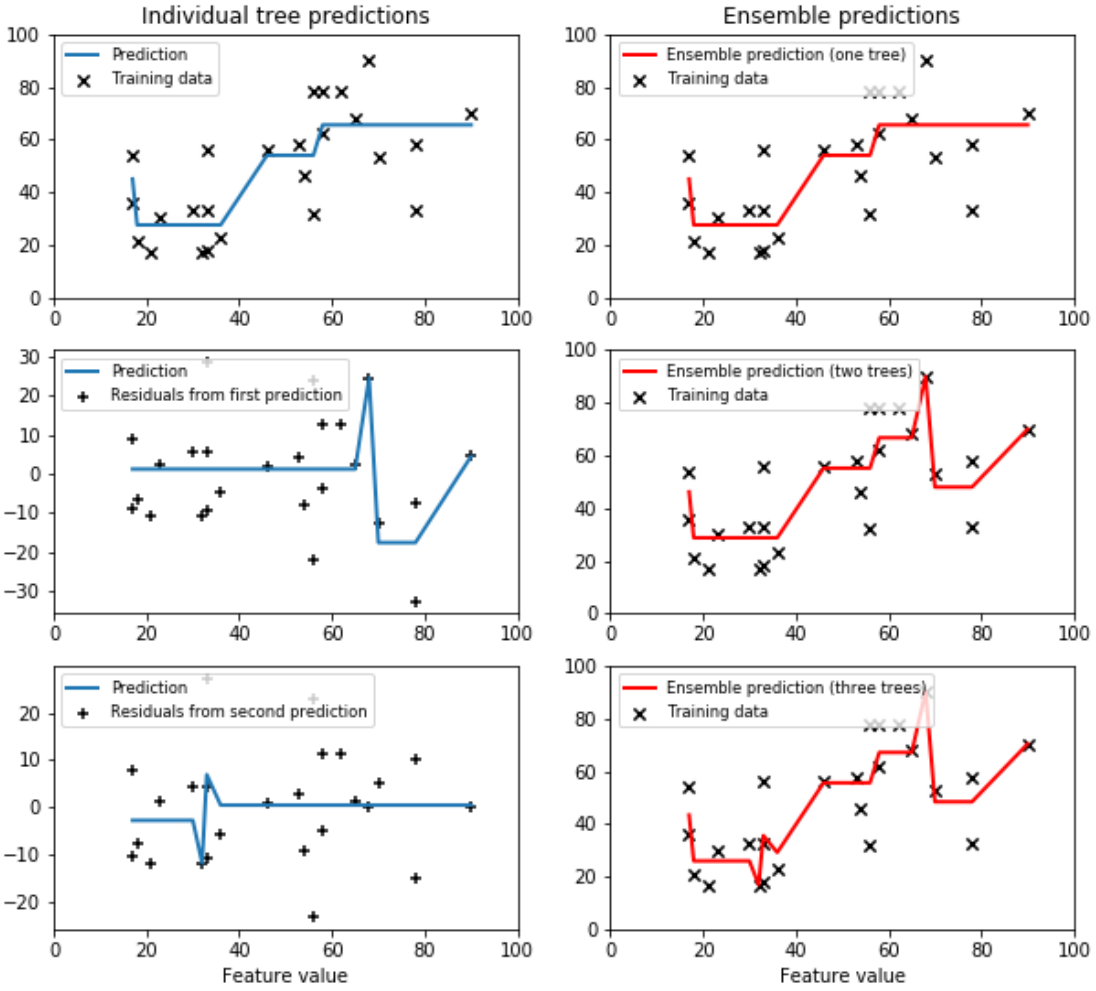

FIGURE 3.3: An example of three iterations of a gradient boosting algorithm. The charts on the left of the figure show the individual tree predictions, and the charts on the right show the ensemble predictions with a learning rate of one.

individual models in a bagging ensemble are independent of each other, the models in a boosting ensemble are trained sequentially, with the output of one model influencing the next.

Figure 3.3 shows an example of gradient boosting with three decision trees; the individual trees' and ensemble predictions are shown at each step. In reality, hundreds of trees may be used in a boosting ensemble.

Both random forests and gradient boosting models will inherit the hyperparameters of a decision tree; they will also require additional hyperparameters, such as the maximum number of decision trees to be used. Another hyperparameter for gradient boosting is the learning rate; this sets the contribution of each tree in the ensemble. A learning rate less than one will mean the model will require more trees to make a sufficiently good prediction but will often generalise to unseen data better than if the learning rate is equal to one.

3.3.4 Support vector machines

Support vector machines (SVMs) can be applied to regression and classification problems, although they are reported to be particularly powerful for classification with complex datasets (Géron, 2019). SVMs can fit linear and non-linear models. For linear binary classification, the aim is to fit a plane that acts as a decision boundary between the two classes. There is the added constraint that the plane should be as far away from the closest training instances of each class as possible – these instances are called support vectors. For a regression problem, a linear SVM fits a plane that contains as many instances as possible within a margin ϵ; the value of ϵ is a hyperparameter to be determined.

SVMs fit non-linear datasets using a mathematical technique known as a kernel trick. The details of how SVM algorithms fit data are out of the scope of this thesis, and the reader is referred to Hastie et al. (2009c) for an explanation.

3.3.5 Artificial neural networks

Artificial neural networks (ANNs) can be used for classification and regression problems. They are composed of layers of neurons: one input layer, one output layer and one or more 'hidden' layers (see Figure 3.4a). The phrase deep learning is often used when referring to an ANN with many hidden layers. Figure 3.4b shows a simple neuron, where the neurons' inputs are represented by x_1 to x_n. The activation of a neuron, j, is defined as:

$$a_j = \sum_{i=1}^{n} x_i w_{ji} + b_i \tag{3.12}$$

where w_{ji} are weights and b_i is a bias value. These are the parameters to be determined during the training process.

The output of the neuron is defined as:

$$y_j = \phi(a_j) \tag{3.13}$$

where ϕ is some activation function, which can take many forms. The sigmoid function (3.11) is one commonly used activation function.

The training process aims to find weights for the network that minimise the value of the loss function for the training data. ANNs are not usually trained on the whole training data set at once. Instead, they are trained on subsets of the training data, known as batches. The loss function is calculated for a batch of training data. Then the gradients

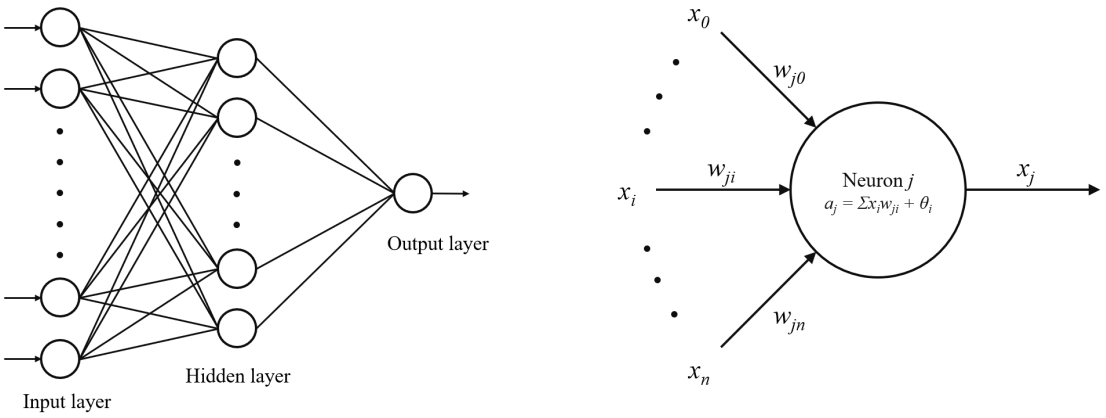

FIGURE 3.4: Representations of (a) a neural network with one hidden layer, and (b) a simple neuron used in a hidden layer

of the weights are calculated with respect to the loss function using differential calculus in a process known as backpropagation.

An optimisation method then determines how the gradients are used to update the weights. While gradient descent can be applied to neural networks, alternative optimisers have also been developed, including Adagrad (Duchi et al., 2011), RMSProp[1] (an unpublished algorithm) and Adam (Kingma and Ba, 2015). These optimisers have been designed to converge faster than gradient descent.

The weights will be updated for each successive batch until all the training data have been used; one iteration of all training data is known as an epoch. The number of epochs that the training process lasts for is usually determined when further iterations provide no further improvement on the value of the loss function.

Neural networks may have more complicated structures than that shown in Figure 3.4, and many different types of neurons can be used within a network. The remainder of this section briefly describes some common types of network that utilise different neurons.

Recurrent neural networks

Recurrent neural networks (RNNs) are a type of ANN whose neurons retain an internal state (or memory) (Chollet, 2017). RNNs are designed for sequence prediction problems, such as text prediction, but they may also be applied to time series modelling. However, there are many different variants of neurons that can be used in an RNN, such as gated recurrent units (GRU) (Cho et al., 2014) and long short term memory (LSTM) cells (Hochreiter and Schmidhuber, 1997).

[1] http://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf

Convolutional neural networks

Convolutional neural networks (CNNs) use neurons that accept multi-dimensional input data (Chollet, 2017). They were initially designed to classify images, and this is the area of application where they continue to have the most success. However, there are examples of CNNs used for other problems, such as time series modelling (Lim and Zohren, 2021).

Bayesian neural networks

Bayesian neural networks (BNN) make probabilistic predictions by fitting distributions to model weights and the output neuron (MacKay, 1992).

Mixture-density networks

Mixture-density networks (MDNs) are another neural network approach that predicts the conditional probability distribution of the target value (Bishop, 1994). MDNs output parameters of multiple distributions, which are then used to construct a mixture of distributions.

3.3.6 Quantile regression

Quantile regression (Koenker and Bassett, 1978) is an approach that predicts specific conditional quantiles of a target value; it achieves this by minimising the loss function for a quantile level τ:

$$L_\tau = \begin{cases} (y - \hat{y})\tau, & \text{if } y \geq \hat{y} \\ (\hat{y} - y)(1 - \tau), & \text{otherwise} \end{cases} \tag{3.14}$$

where $0 < \tau < 1$.

When $\tau = 0.5$, the quantile loss is equivalent to the absolute loss where the median value is predicted. If, for example, the 0.05 and 0.95 quantiles are estimated, then the 90% prediction interval can also be estimated. As the main idea behind quantile regression is a loss function, it can be applied using various approaches, including linear models, SVMs (Huang et al., 2014), decision tree ensembles (Meinshausen, 2006; He et al., 2017) and ANNs (Deng et al., 2020).

3.4 Applications

3.4.1 Travel durations

Kecman and Goverde (2015) developed machine learning models to predict run durations. The data were filtered to obtain conflict-free trains, i.e. those not held at red signals, and freight trains were excluded. The authors aimed to create models that did not require detailed descriptions of rolling stock, platform design, or passenger data. Features include 'block length', 'distance from the last scheduled stop' and 'distance to the next scheduled stop'. Time was represented as a predictor variable by a binary variable indicating whether it was in peak hours or not. The authors acknowledged the difficulty in distinguishing between peak and non-peak times, which are not clear-cut and may not be the same for all stations in the dataset.

Three models were developed for the global data set using linear regression, decision trees and random forests. The authors note the presence of outliers in their target value and therefore used the least-trimmed squares (LTS) method to estimate the parameters for the linear regression model. The LTS method uses a subset of inlying observations to fit the model rather than the whole dataset.

The random forest model provided the highest R^2 value of 0.78. Local models were also developed that estimated run durations for particular blocks. The local models were developed using linear regression only and had comparable performance to the best global model. However, a local model cannot be generalised to other blocks. The models were built using only three months of data, so it was not feasible to investigate any seasonal impact on run times.

Barbour et al. (2018) used machine learning to predict the arrival times of freight trains in the United States using two years of historical data. The authors used a simple statistical model as a benchmark: it uses the median run duration from a train's location to its destination to predict the arrival time. They then built various machine learning models, including linear regression, support vector regression, random forests and artificial neural networks. The MAE was used as an evaluation metric. The machine learning models all outperformed the benchmark, and the random forest model produced the most significant improvement.

Wen et al. (2020) developed a model using a network of LSTM cells to predict the arrival delay at Dutch railway stations. The problem was structured as a time series with previous delays at a station being used to predict the subsequent delay. The authors experimented with various numbers of previous train records but found little benefit to including more than one previous train. Therefore, the data were constructed into a time series of two trains; the delay of the second train was the value to be predicted.

The LSTM model was compared to two other models: one random forest and one ANN without recurrent units. The LSTM model provided the lowest MAE and MSE values.

Huang et al. (2020) designed an artificial neural network that utilises both LSTM cells and basic neurons to predict the arrival time of a train at its next station on the Chinese railway. The authors used five preceding trains in each time series. The model structure is calibrated by varying the number of LSTM layers to investigate the impact on time and MAE. They compared the output to several other models, including a random forest, support vector regression and some other formulations of ANN. The model with the LSTM cells provided the lowest MAE values.

Other transport domains have also exploited machine learning to predict travel durations. Petersen et al. (2019) made short-term predictions of up to one hour in the future for bus arrival times in Copenhagen. The approach taken was to divide the problem into three sub-models. The first sub-model is a fully connected ANN with two hidden layers. It predicts travel duration using only temporal features, for example, the day of the month, day of the week and time of day. The authors noted that a large amount of variability in travel duration could be explained by seasonal factors (such as the time of day), so this model represents the seasonal influences. The second sub-model used a network of convolutional LSTM cells to model travel durations across journey segments. The third sub-model forecasted the dwell durations at bus stops. An exponential smoothing method was applied for this model due to the lack of real-time demand data. The output of the three models was then fed into a fully connected neural network to provide an ensemble prediction.

The evaluation metric used by Petersen et al. (2019) is the RMSE, and they compared their model with two benchmarks: a historical average model and a linear regression model. They demonstrated that the RMSE increases the further out in time that the predictions are made for all three modelling approaches. The model presented by the authors provides a smaller RMSE than the two benchmarks. For example, for a forecast period of 15 minutes in the future, the ensemble model had an RMSE of 4.38 minutes, the linear model was 4.77 minutes, and the historical average was 6.05 minutes.

O'Sullivan et al. (2016) used quantile regression to construct prediction intervals of bus arrival times. They used the quantile regression to demonstrate that the predictions exhibit heteroscedasticity: they are less certain the further away the bus is from its destination. They also found quantile regression valuable as they could choose asymmetric quantiles. They observed that the errors were biased towards underestimating the arrival time, so they used the 97.5% and 7.5% quantiles. Using quantiles of 95% and 5% lead to a larger absolute interval.

An ensemble of Bayesian neural networks was presented by van Hinsbergen et al. (2009) to predict travel durations on a Dutch motorway. They used the output distributions

from the models to construct a 95% prediction interval for each prediction. They demonstrated that the intervals were larger at peak times of day and that over 97% of the actual travel durations fell within the 95% prediction intervals.

Parslov et al. (2021) developed machine learning models to be used as a decision support tool for bus drivers. They discussed how bus routes often include locations where passengers may transfer between services. It is often left to a driver's judgement whether or not to wait for a connecting service before departing. If they depart before the connecting service arrives, passengers who want to transfer will be delayed. However, if they wait a long time for the incoming service, non-transferring passengers will be delayed. The authors stated that it is essential that bus drivers have predictions of arrival times of connecting services to make a decision and that understanding the uncertainty around any prediction can lead to improved decisions.

The paper compared quantile regression using a artificial neural network to a Bayesian neural network for predicting bus travel durations with prediction intervals. Their dataset comprised 17 weeks of data for two bus routes split into 29 and 39 segments, or 'links', respectively. The authors considered building models to predict travel duration through single and multiple links. They used the first 13 weeks as training data, the next two weeks as validation data (for tuning hyperparameters) and the remaining two weeks were held back as test data to provide a final evaluation of the models. They reported results on the test data only.

The experiments made travel duration predictions at 15, 30 and 45 minutes into the future using five prediction intervals: 20%, 60%, 80%, 90% and 95%. The authors identified several metrics to compare the two different models. One metric was the interval covering percentage (ICP), which measures the percentage of records falling in a given prediction interval – if the prediction interval is, for example, 90%, then approximately 90% of records should be within the interval bounds. Another metric is the RMSE, which was used to evaluate the point predictions.

The quantile regression model produced lower values of RMSE for point predictions than the Bayesian model, although the value of the RMSE did increase as the time horizon moved from 15 to 30 and 45 minutes. The quantile regression model produced ICP values very close to the prediction interval values for the 80%, 90% and 95% prediction intervals. However, the closeness dropped off as the time interval increased, with a slight underestimation of the interval. The quantile regression model also outperformed the Bayesian neural network with respect to the ICP for the 60% prediction interval. However, the results were more inconsistent than the larger intervals, and the Bayesian neural network outperformed the quantile regression model at the 20% prediction interval.

The study concluded that the quantile regression model performed better overall but acknowledged that there were specific situations where the Bayesian model was stronger

and that both approaches have the potential to represent uncertainty in travel duration predictions. The authors demonstrated how the results could be integrated into a decision support tool that could help bus drivers decide whether to wait for an incoming connecting service or not. They quantified a reduction in delays if the models representing uncertainty were used instead of point predictions.

3.4.2 Dwell durations

As well as modelling run durations, Kecman and Goverde (2015) built models to estimate the dwell durations of passenger trains. They created models using linear regression with least-trimmed squares (LTS), a decision tree and a random forest. They demonstrated that the random forest model has a lower MSE and higher R^2 than a single regression tree, suggesting it is more robust to outliers. It also outperforms the linear regression model. Their results showed a strong correlation between arrival delays and dwell duration for large stations but a less significant correlation for small stations where only local trains were scheduled to stop.

Li et al. (2016) recognised that the number of passengers boarding and alighting at a railway station is influential in determining dwell durations. However, data on passenger numbers can be challenging to collect, so the authors aimed to find substitute variables to reflect passenger demand. They developed two models: one linear regression and the other a k-nearest neighbours model. In common with other studies, the authors split the data into 'peak' and 'off-peak' times.

A case study was presented using three months' worth of data, totalling 17,306 records for a single short-stop station. The data were divided into two sets, the first is used for training the model parameters, and the second is used to validate the models. However, it appears the whole data set was used to investigate which features to include in the models, and no test data set was held out to evaluate the models on unseen data. Linear regression models had the best performance for peak hour data with a MAPE between 11.5% and 14.2%. The k-nearest neighbours model performed best for non-peak hours with a MAPE of 19.95%. Possible reasons for lower MAPE values for peak-hour predictions were not explored. The authors concluded that dwell duration estimation is possible without explicit passenger data, although they acknowledged that the study was limited by only considering one station.

Jiang et al. (2018) identified that the number of passengers is not by itself a sufficient predictor of dwell duration. They developed a prediction model for dwell duration using data from an automatic fare collection system which provided the number of passengers inbound and outbound from a station. They also used data from an automatic train supervision system to provide each train's precise dwell duration, headways and arrival

times. They used one month's worth of data for a single station on Shanghai's metro network. They listed factors that might affect the dwell duration:

- Inbound and outbound passenger flow
- Direction of travel
- The congestion on the train
- Deviation from schedule
- The actual headway (this can affect the train and platform capacity utilisation)

The authors built a support vector regression model to predict dwell duration. The data were portioned randomly into two sets: the first set comprised 75% of the records and was used for training, and the remaining data were held out for testing. After experimenting with different model configurations, they achieved a model with an MSE of 2.87 seconds and an R^2 value of 0.66 on the test data. These results were also close to those achieved when making predictions on the training data, indicating that the model has not been overfitted. The results were favourable compared to a linear regression model, which produced an MSE of 4.00 and an R^2 value of 0.55. The authors concluded that advances in passenger detection technologies, such as Bluetooth or video surveillance, might provide more detailed information on passenger flow and distribution on a platform in future.

3.4.3 Traffic management decisions

Dündar and Şahin (2013) developed an artificial neural network (ANN) to mimic the decisions of signallers. The model used the details of two trains approaching a conflict location as input. The model predicted which train would be allowed to pass the conflict location first and which would be delayed. The study did not use a large amount of data: only 173 records were used to train the model, and the authors stated that 158 were used for validation *and* testing. This implies that all the data were used for evaluation while training the model.

The authors reported that the ANN could correctly predict which train would have priority for 327 records, an accuracy of over 98%. This suggests that the model has been able to 'learn' the actions of signallers; however, there is no evidence that the model would generalise to unseen data and no comparison to a non-machine learning benchmark. They used the ANN to compare signalling decisions with a traffic management strategy they developed using a genetic algorithm. They found that the resolutions performed by the genetic algorithm could produce smaller total delays than the ANN. The authors concluded that improvements could be made in the existing signalling strategy.

Luo et al. (2022) identified actions that signallers or controllers take to resolve conflicts or recover delays on a railway network as changing:

- planned dwell durations
- assigned tracks or platforms
- the operating speed of a train
- the departure order of trains

The authors built a machine learning model that predicts decisions for two particular actions: changing planned dwell durations and changing the operating speed of trains. These actions are usually taken to avoid route conflicts with other trains. The authors proposed the method as being useful as a traffic management support tool.

The model had two stages. The first stage was a classification prediction that identifies whether the action is to increase, decrease or maintain (according to schedule) the dwell or run duration. The second stage is a regression model predicting the change in duration if the action deviates from the schedule. The classification and regression models were built using a deep forest, an approach sharing characteristics of random forests and deep neural networks (Zhou and Feng, 2017). They built the models using over a year and a half of data from two Chinese high-speed railway lines. They appear not to have held back any data for testing. The data were not evenly proportioned into the three classes for the classification problem, so the authors included a pre-processing step that generated additional synthetic data for underrepresented classes.

The authors compared the output of the classification model to other machine learning models, including an SVM, an ANN and a random forest model. The deep forest model produced a higher accuracy for both railway lines than the other machine learning models, although it often did not significantly outperform the random forest model's results. Both lines achieved higher accuracy for dwell duration than run duration. When considering other metrics, such as recall or precision, the random forest model could outperform the deep forest – although the scores between the two approaches were close.

Alternative machine learning models were also built for the regression problem, and the deep forest model produced the lowest MAE compared with the other approaches. However, other metrics (RMSE and R^2) sometimes favoured the random forest approach. As no non-machine learning benchmark model was provided, it is difficult to justify how 'good' these results are.

Box (2014) presented a road traffic simulation model with junctions and signals. A computer game interface was developed for the simulation, allowing a human player to interact with the simulation and control the signals.

The study collected data from a human player controlling the signals in the model with varying levels of simulated traffic. A new record gets added to the training data each time the player presses a key to change the state of a junction. The number of records collected was not explicitly stated; however, it did state that the player played six 30-minute simulated games. The data were used to train a small neural network with two layers to predict the state of each junction given a snapshot of the traffic as input. The accuracy of the supervised model was not reported, but it was used within a simulation model – which will be discussed in Section 3.5.3.

Pham et al. (2020) used machine learning to predict the actions of air traffic controllers. They considered only aircraft being controlled en-route (rather than at airports), where the airspace is divided into sectors. The tasks are divided between two controllers in each sector: a planning controller and an executive controller. The planning controller is responsible for processing flight plan information and organising traffic flow into the sector. Once the aircraft enters the sector, then the executive controller manages its path. The paper aimed to predict the actions of the planning controller to investigate whether some of the tasks could be automated.

The authors used the random forest and gradient tree boosting algorithms to build classification and regression models. Three classification models were built using each algorithm that respectively predicts whether the controller makes a change in the speed, course or altitude of the aircraft. Similarly, three regression models were built using each algorithm predicting the action values, for example, the ground speed rate.

The study used six months' worth of data for one airspace sector managed by the Singapore Area Control Centre. The authors used cross-fold validation (with ten folds) but did not hold back any test data. Therefore, the paper presented results from evaluations of the model's data during training. Accuracy results from the two classification approaches and three actions vary from 78.5% to 99.2%. For the regression models, the metric R^2 was used, and this varied from 0.667 to 0.870. The results showed that neither random forests nor gradient tree boosting trees consistently outperformed the other. Similarly to other studies that have been reviewed, there was no comparison to a non-machine learning benchmark, which means that it is difficult to judge the efficacy of applying machine learning to this problem. The authors acknowledged that each sector of airspace has varying operating characteristics, and therefore it is unlikely that the models would generalise to a different sector.

3.5 Machine learning-assisted simulation modelling

3.5.1 Hybrid approaches

Simulation and machine learning are two different modelling paradigms; however, there have been examples in the literature of hybrid methods that harness both approaches (Greasley and Edwards, 2021). In a review of hybrid models, von Rueden et al. (2020) divides the literature into two rough groups, depending on whether the dominant approach is simulation or machine learning.

Where simulation is the dominant approach, the authors refer to this as machine learning-assisted simulation modelling. This approach typically deploys machine learning models within a simulation to determine some simulation parameters. Conversely, if machine learning is the dominant approach, then this is referred to as simulation-assisted machine learning. An example of simulation-assisted machine learning is the use of a simulation model to generate synthetic data that the machine learning model can use as input.

The remainder of this section will review literature on machine learning-assisted simulation models only. Within this scope, only supervised machine learning is considered, as this has been the focus of this chapter thus far.

3.5.2 Railway domain

A literature review of artificial intelligence in railway research did not identify any examples of machine learning-assisted simulation (Tang et al., 2022); however, a more recent journal article presented an example of machine learning-assisted simulation of freight yards (Minbashi et al., 2023).

Minbashi et al. (2023) built a random forest model to predict deviations in the departure time of freight trains from a yard, which outperformed a statistical benchmark model. The machine learning model was then used to determine the departure times of trains from the freight yard within a macro-simulation. The authors do not explicitly state whether the simulation is deterministic or stochastic; as the paper contains no discussion about iterations or variations in travel durations, it is assumed to be deterministic.

The authors presented a case study on a section of track between two freight yards on the Swedish railway network over four months. The hybrid approach was evaluated by calculating the arrival times of freight trains at their destination. The mean absolute error (MAE) between the simulation and actual (known) arrival times was 35 minutes.

To evaluate the quality of the result from the hybrid model, the authors also compared the expected arrival times (from the timetable) to the known arrival times, which produced an MAE of 42 minutes. Further, predictions for arrival times made using

the machine learning model only (without the simulation model) were compared to the known arrival times, producing an MAE of 39 minutes. The authors acknowledge that the machine learning-assisted simulation model does not significantly reduce the MAE compared to these other approaches. They discuss that this may be due to the different modelling approaches not being directly comparable.

Minbashi et al. (2023) inspected the deviation in arrival times produced by the hybrid model and compare it to the known arrival deviation for the case study data. The actual arrival deviations are skewed to the right, and the authors observe that the machine learning-assisted simulation model underestimates the number of larger deviations in the tail of the distribution.

3.5.3 Other domains

Box (2014) developed a supervised learning model to mimic the actions of a human controlling signals at road traffic junctions (discussed in Section 3.4.3). This model was then used to control road traffic within a simulation. The study investigated whether established traffic control algorithms could outperform an algorithm learned from human behaviours. The author simulated traffic flow and varied the traffic control algorithms used in each scenario. The supervised learning model for traffic control outperformed the established control methods regarding the average delay and variance over the delay. The author concluded that a supervised traffic control model learned from human behaviour may be a viable alternative to established automated traffic control models.

Examples of such machine learning-assisted simulation models also exist in the healthcare domain. Olave-Rojas and Nickel (2021) used machine learning models to determine key simulation parameters for a model of emergency medical services. For example, the authors developed a supervised machine learning model of the travel speed of ambulances. The learned model was then used in the simulations to predict the travel durations of ambulances to emergencies and hospitals.

Abuhay et al. (2021) presented a framework for integrating supervised machine learning models that predict hospital admission rates, patients' length of stay, and cost of treatments with a simulation that models patient flow within a hospital. Although the paper demonstrated how the two modelling approaches could be combined, there were no results to determine the method's effectiveness.

Elbattah and Molloy (2018) developed an approach for analysing specific questions surrounding hip fractures in elderly patients, such as how to predict a patient's length of stay or the patient's discharge destination. They achieved this by using an unsupervised machine learning technique to cluster patients before simulating patients through their care journey. A supervised machine learning model was then applied to the simulation

output to predict the discharge destination. The model aimed to help plan for future resource requirements.

Gartner and Padman (2020) reported that acceptable waiting times significantly affect patient satisfaction in healthcare settings. They used a simulation model to understand patient flow through a hospital's emergency department; the model demonstrated how altering staff patterns can impact patients' waiting times. They also trained a classification model to predict whether patients under-estimate, correctly estimate or over-estimate their waiting times in a hospital emergency department. This model could then enhance the simulation output by predicting the effect of staffing changes on the patients' perceived waiting time and hence patient satisfaction.

Outside of the healthcare domain, Reed et al. (2021) presented a machine learning-assisted simulation model representing vehicle operations in a caving mine. The vehicles are used to load, transport and then unload mined material. Compared with the other papers that have been reviewed, the unique aspect of this work is that a mixture-density network (MDN) was used to determine the probability distributions of random variables in the simulation model. This means that the machine learning models can model stochastic behaviour compared to the other papers that use machine learning regression models to make point predictions. MDNs were used to determine various parameters in a discrete event simulation model, including vehicle travel durations.

The simulation model was used to evaluate three different layouts of the production area of a mine where the vehicle operations occurred. The results were able to identify the impact of waste rock on the production rate – something that previous simulation models had not identified – and the authors state that the results show that the model was able to represent complex stochastic behaviour.

3.6 Discussion

The literature review in Section 3.4 showed that supervised machine learning has been applied to make run time, dwell duration and traffic management predictions in the railway and other transport domains. Supervised machine learning models make predictions based on a selection of features, which makes them an attractive alternative to established methods for modelling run and dwell durations. The previous chapter discussed how existing methods struggle to model conditional run and dwell times. By contrast, the literature review in this chapter demonstrated estimations for run and dwell times using conditional features such as the punctuality of a train (Wen et al., 2020) and whether the train is running in 'peak' or 'off-peak' times (Li et al., 2016).

The use of supervised machine learning to make traffic management predictions appears to be less well-established, which Luo et al. (2022) also note. This may be because

relatively few use cases require a model to mimic existing traffic management decisions. There are, by contrast, many examples that consider the *optimisation* of conflict resolution problems (Meng and Zhou, 2011; Pellegrini et al., 2012; D'Ariano et al., 2014; Samà et al., 2017). However, the work of Box (2014) (discussed in Section 3.4.3) suggested that a supervised model that learned from a human traffic controller could be used as a proxy for an optimal strategy as humans typically could outperform traffic management algorithms.

The review identified several weaknesses that were common to many of the studies. For example, few studies compared the performance of machine learning models to a non-machine learning benchmark model. Machine learning is not a straightforward technique and requires significant time to pre-process the data and build models. Therefore, it is beneficial to know how well they perform compared to alternative basic predictive methods to gauge the advantage of applying machine learning. In the reviewed literature, only Barbour et al. (2018) and Petersen et al. (2019) compared machine learning models to a non-machine learning method. Some authors may consider linear regression a statistical model rather than machine learning, as several studies used linear regression as a comparison. Regardless of whether linear regression is perceived as a machine learning technique, it is not a meaningful benchmark if the underlying relationship between the feature vectors and the target values is non-linear.

Another weakness of many reviewed studies is a failure to report results on test data that the machine learning models had not seen during the building process. Instead, most papers reported results on data used to train the models. While this *indicates* how well a model can make predictions, the training or validation data results should not be used to make a final assessment of the model. Such an approach will not identify if the model has been overfitted to the training data and gives no information about how it generalises to unseen data.

Quantifying the uncertainty of supervised machine learning predictions appeared to be absent in research from the railway domain. However, several studies reviewed outside the railway domain demonstrated the application of Bayesian neural networks and quantile regression to go beyond making point predictions for travel durations (van Hinsbergen et al., 2009; O'Sullivan et al., 2016; Parslov et al., 2021). There was no discussion of uncertainty in the papers reviewed that were making dwell durations or traffic management predictions.

The combination of machine learning and simulation techniques is an area of research that appears to be gathering more attention, and Section 3.5 introduced machine learning-assisted simulation modelling. However, the reviewed literature demonstrated that the approach can be applied to transportation modelling, such as travel durations. Moreover, Reed et al. (2021) showed that an uncertainty-aware machine learning model could be used to represent the stochastic movements of vehicles in a simulation.

3.7 Conclusion

This chapter has provided background information about supervised machine learning and some detail about different algorithms and approaches to predictive modelling. A literature review has revealed that supervised machine learning is an approach that can be used to make predictions about train dynamics. It has demonstrated the potential to make predictions about traffic control decisions, although this topic is less established in the existing literature. The review also identified an opportunity to apply uncertainty-aware methods to predictive models in the railway sector.

Moreover, this chapter has introduced machine learning-assisted simulation modelling, of which there are no known studies in the railway domain. Given the potential of machine learning to make predictions about train movements and traffic management decisions, such models could be deployed to control some dynamics within a simulation model. Suppose these models could also go beyond point predictions to predict probabilities or quantile estimates. In that case, this approach could model variation in train movements and a more realistic representation of signaller decisions within a simulation model. The following chapter will lead on from these conclusions to present the modelling framework developed for this thesis.

Chapter 4

Modelling framework

4.1 Introduction

The previous chapter introduced supervised machine learning and described common algorithms used to build predictive models. The chapter reviewed literature on supervised machine learning to predict travel durations, dwell durations and traffic management decisions in many transport domains. Machine learning-assisted simulation modelling was introduced by reviewing research outside the railway domain. The conclusion was that supervised machine learning models have the potential to predict train movements and signalling decisions and that these could be deployed to control the dynamics within a railway simulation model. Moreover, uncertainty-aware machine learning techniques could be exploited in stochastic simulations.

This chapter outlines the modelling framework developed for this thesis; it lays the groundwork for the detail in future chapters. The approach for this work is to build machine learning models using historical railway data that can be deployed within a simulation model. The machine learning models will be built before running the simulation model. They will be called within the model to make predictions regarding the movement of trains and traffic management decisions, as shown in Figure 4.1. There are many factors in traffic management, and this work will focus on just one: conflict resolution.

There are three distinct aspects of the work. The first is the creation of datasets to represent infrastructure, train movements and conflicts, which will be described in Section 4.2. The second aspect of the modelling process is described in section 4.3: the creation of uncertainty-aware machine learning models to predict train movements and traffic management decisions. The third step is outlined in Section 4.4, which is the development of a simulation model that can utilise the machine learning models.

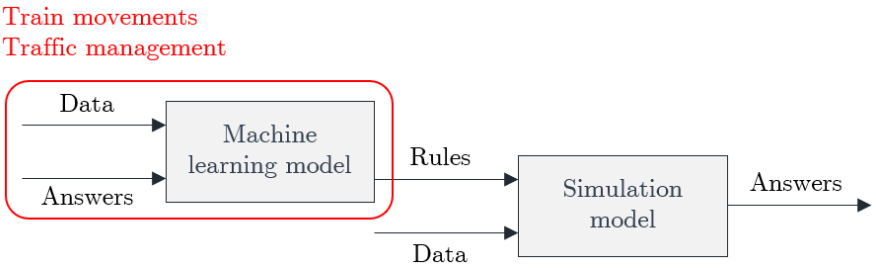

FIGURE 4.1: A simplified visualisation of the modelling approach used in this thesis, cf. Figure 3.1

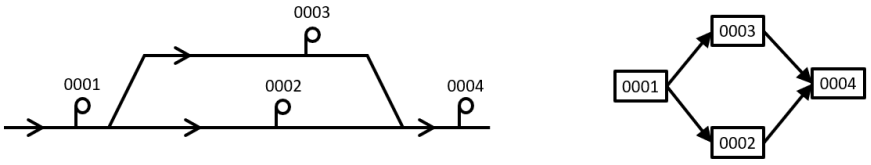

FIGURE 4.2: Illustration of how a track is represented as a graph of berths, adapted from Teshima et al. (2014)

4.2 Data sets

4.2.1 Infrastructure dataset

The purpose of the infrastructure dataset is to represent the railway network which will be achieved by modelling a network of berths. A berth usually represents a block section of track on the approach to a main signal. In this thesis, there is a one-to-one relationship between the signals and berths, and only one train can occupy a berth at any time[1]. A movement between berths is called a step, and the dataset will also need to contain valid steps between the berths. The network is therefore represented as a directed node-edge graph representing the possible steps, as shown in Figure 4.2. For example, in Figure 4.2, a valid step is between berth 0001 to berth 0003; however, it is not permissible to step from berth 0003 to berth 0001.

The dataset also needs to identify and record conflicting steps. Such steps are mutually exclusive and cannot be traversed simultaneously. For example, in Figure 4.2, it is not permissible for a train to move from step 0002 into berth 0004 at the same time as a train stepping from berth 0003 into berth 0004. In this example, it is clear that these steps are mutually exclusive as they both lead to the same berth, and only one train can occupy a berth at any time. Not all conflicting steps are so easy to identify, which will be discussed further in Appendix Section B.2.3.

[1]This does not apply across the whole network. Some berths may not be associated with a main signal, for example some long platforms may be formed of multiple berths. Also, there will be locations where trains join and divide, leading to two physical trains in a berth for some short period.

4.2.2 Train movements

A dataset of train movements will be required to build machine learning models that predict travel durations. This dataset is created using historical data, and each record represents the time a train stepped into a berth. Data about a train's timetable is also provided for each record.

4.2.3 Train conflicts

A dataset of conflicts between trains is required to build machine learning models that predict the resolution of conflicts. Pairs of conflicting steps will have been identified in the infrastructure dataset. Each record in the dataset comprises data about two trains approaching conflicting steps, such as each train's current berth and timetable. The train that was given priority at the conflict location will be recorded.

4.3 Machine learning models

4.3.1 Aim of models

This work has developed two different types of models. The first predicts the travel duration of trains between discrete points on the network, and the second predicts traffic management decisions for conflict resolution. The train movement and conflict datasets, described in the previous section, will be used for the modelling, respectively. Both model types follow the same evaluation and modelling strategies (described in the following sections) but predict different target values.

The travel duration predictions are regression models. They predict the time it takes for trains to travel through a berth on the network. The models will be developed to represent uncertainty in the predictions by modelling the distribution of the target value.

The conflict resolution decision predictions focus on predicting conflict resolutions between pairs of trains, which will be formulated as a classification problem. A value of 0 or 1 will indicate which train has priority. While this is only one aspect of traffic management, it has been necessary to narrow the focus given the timescales of the work. Other traffic management policies, such as route selection or cancellation of trains, could also be modelled as an extension to this work if this approach is successful. The models will be calibrated to improve the quality of probabilistic predictions.

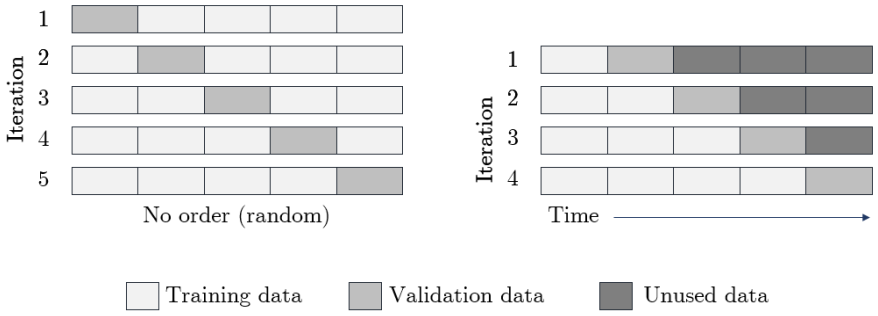

FIGURE 4.3: An illustration of the difference between cross-fold and prequential block evaluation

4.3.2 Evaluation strategy

In order to assess the machine learning models, a suitable evaluation metric will be selected for the classification and regression models, respectively. Benchmark models will be created, and the machine learning models are expected to at least outperform any benchmark model with respect to the selected metric.

The training, validation and test data will not be selected randomly. There is the possibility that the data may evolve over time. For example, operating conditions in 2018 may differ from those in 2021 which could impact travel durations or signalling decisions. In such cases, if the test data are selected randomly, they will not be independent, which may undermine results (Bergmeir and Benítez, 2012). Instead, the data are ordered by date and time and then segmented into fourteen batches, each containing 90 consecutive days' worth of data. The fourteenth batch of data is held out as the test data.

During the model-building, a form of prequential block evaluation (Cerqueira et al., 2019) will be used whereby the data remain ordered. In this process, the training data grows with each iteration of model-building and the model is evaluated sequentially on a validation batch, see Figure 4.3.

4.3.3 Model building

Building the machine learning models is divided into four stages described in this section, and the process applies to both types of model.

Preparation

The first stage involves the selection of an evaluation metric, benchmark models, and supervised machine learning algorithms. The data are prepared for machine learning, which involves selecting and transforming feature vectors.

Exploration

In the exploration phase, a wide variety of hyperparameters for the selected machine learning models are considered. Several steps are taken to reduce the duration of this stage of the work as it is potentially time-consuming. Firstly, the models are only trained on the first seven batches of data, with the eighth batch used as validation data. Secondly, only point predictions are made in this phase for the regression models (rather than predicting a full distribution), and classification models will not undergo a calibration phase. Thirdly, a random search is performed across all possible combinations of hyperparameters, meaning not all of them will be selected. These decisions ensure that a wide variety of different configurations can be explored within a reasonable timeframe.

The benchmark models will also be built using the same training data and evaluated on the same validation data. The evaluation of the machine learning models can then be compared to the benchmarks and each other.

Selection

This stage considers a narrower focus than the previous phase. The machine learning configurations that provide the best evaluation score are chosen from the exploration phase. The models' hyperparameters are tuned by allowing slight variations in the selected values. These models are then built and evaluated using a larger dataset – batches one to thirteen (inclusive) are utilised. The models will first be built using batches one to eight and evaluated on batch nine, then built using batches one to nine and evaluated on batch ten, and so on, up to evaluation on batch thirteen – as per Figure 4.4. A single evaluation of each model is obtained by taking the mean of the evaluation metric over the five iterations. This approach to evaluation will demonstrate whether the models can make consistent predictions over time. The benchmark models will also be built and evaluated using the same approach.

As with the exploration phase, regression models will only be built to make point predictions, and classification models will not undergo a calibration phase.

Final evaluation

A single configuration of each model will be selected from the previous stage based on the evaluation metric. This model will be built using batches one to thirteen. Classification models will undergo a calibration phase will be developed to model the distribution of the target value. The models will be evaluated on batch fourteen, thus far unseen during the modelling process. The results can be compared to the evaluation of the benchmark models, which will also use batch fourteen as test data.

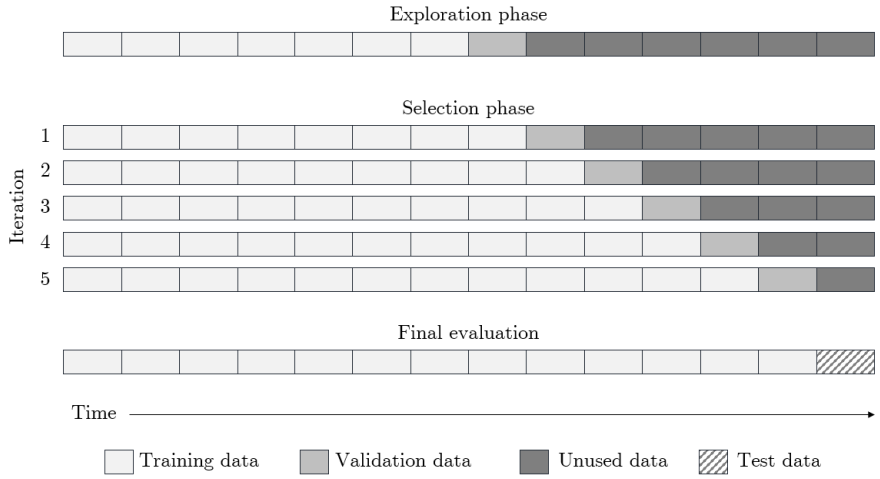

FIGURE 4.4: An illustration of the evaluation strategy used in this thesis

4.3.4 Tools

The Python programming language[2] was selected to develop the machine learning models. Python is a high-level programming language distributed under a permissive free software license, meaning there are few restrictions on its use. It is a popular language for machine learning and data science, with many packages available to make data-intensive processes simple to implement. This work has used Python version 3.6.13.

Two Python open-source software libraries are employed to develop the supervised machine learning models: Scikit-learn[3] version 0.24.2 and Tensorflow[4] version 2.6.2.

As supervised machine learning can be computationally expensive, the research utilised the University of Southampton's IRIDIS[5] high-performance computing (HPC) facility. This enabled much of the programming work to be carried out in parallel, compared to using a personal computer.

4.4 Simulation model

4.4.1 Design principles

A new simulation model has been developed, rather than using any existing software. This decision was taken because the simulation will need to integrate with machine learning solutions, which will be easier to achieve using custom-built software rather than extending existing software. The simulation model itself is not the main contribution of

[2] https://www.python.org/
[3] https://scikit-learn.org/stable/
[4] https://www.tensorflow.org/
[5] https://www.southampton.ac.uk/isolutions/staff/iridis.page

the thesis; to this end, there has been no attempt to replicate the detail or functionality of some models, such as RailSys or OpenTrack. Instead, the focus is on the potential for a machine learning-assisted simulation modelling approach.

4.4.2 Simulation paradigm

Time-driven processes often occur in railway simulations that use motion equations to control the movement of trains. This work will not use motion equations, so a time-driven simulation is unnecessary. Instead, the model is implemented using an event-driven approach. As described in Section 2.2, this means that the simulation 'steps' forward in time to the next earliest event time and then updates the objects in the simulation if their state changes at that time. This approach reduces the time it takes to run each iteration.

4.4.3 Programming paradigm

The simulation model will implement an object-oriented programming (OOP) approach. OOP contrasts with procedural programming, whereby code is executed in a sequence (Sage, 2019). This thesis does not provide background on object-oriented programming, but the following paragraphs briefly discuss some features of OOP which benefit this work. The reader is referred to introductory texts on the subject, such as Sage (2019) or Kong et al. (2021), should they require further detail.

The cornerstone of object-oriented design is the class. A class is a logical unit of code containing attributes (data) and methods (behaviours) (Kong et al., 2021). During run-time, the program creates instances of the classes known as objects. This helps reduce the amount of written code compared with procedural programming.

Another advantage of OOP is the concept of inheritance (Kong et al., 2021). Classes may be designed in a hierarchical structure with 'parent' and 'child' classes. The child classes will inherit the attributes and methods of the parent, thus again reducing the amount of code. Child classes can extend their parent(s) by introducing new attributes and methods; moreover, it is also possible to override the methods of a parent class which is an example of polymorphism. Inheritance and polymorphism will be valuable for experimenting with different logical structures for traffic management and train movements.

4.4.4 Tools

Python was selected as the programming language for building the simulation model: the same language used to build the machine learning models. This decision makes the machine learning and simulation models easier to integrate than if different languages were used. Python is also capable of implementing object-oriented designs. One disadvantage of using Python for developing simulation models is that it is slower to execute than alternative languages, such as C++ or Java (Dagkakis and Heavey, 2016). This would be a consideration if the work were extended or deployed in a business environment.

As with the machine learning models, the simulation is also developed to run on the University of Southampton's HPC facility.

4.5 Conclusion

This chapter explained the modelling framework developed for this research by outlining three areas of work. The first area concerns the foundation for the modelling work: the development of datasets. A brief description of the datasets was provided. Chapter 5 will introduce the data sources for this work, discuss the creation of the datasets in more detail, and illustrate them through data analysis.

The second aspect of work is the development of supervised machine learning models. This chapter explained that two different model types are constructed: regression models predicting train travel durations and classification models predicting conflict resolution decisions. The chapter outlined the strategy for evaluating and building the models and documented the programming tools used to develop the models. Chapters 6 and 8 present supervised machine learning results for the two model types, respectively.

The final area of research is the development of a simulation model that can integrate with machine learning models to drive the dynamics of the simulation. Chapter 7 introduces the simulation model in more detail and provides results for modelling train movements using the supervised machine learning models. Chapter 9 will present results using conflict resolution machine learning within the simulation.

Chapter 5

Data

5.1 Introduction

The previous chapter introduced the modelling framework developed and applied in this research. The framework divides into three sections, and this chapter focuses on the first part, which concerns the creation of datasets. The datasets are crucial to the machine learning and simulation modelling work, and this chapter describes the data sources and dataset construction. Key characteristics of the data will be illustrated in tabular and graphical forms.

A small area of the British railway network has been selected to demonstrate the processes described in this and subsequent chapters, and Section 5.2 introduces the features of the selected area. Section 5.3 presents the data sources used in the construction of the datasets; it will begin with a discussion on open data sources and then describe each of the data feeds and datasets used as raw data in this work. Further information on the data sources is provided in Appendix A.

Data from these sources were collected from 2018-04-06 to 2021-09-16 (inclusive) – a total of 1260 days – and used to create the three datasets introduced in the previous chapter regarding infrastructure, train movements and train conflicts. Sections 5.4, 5.5 and 5.6 discuss these datasets, respectively. Further details on some aspects of the dataset construction are provided in Appendix B. Each section describes the construction of the dataset and provides visualisations of the data.

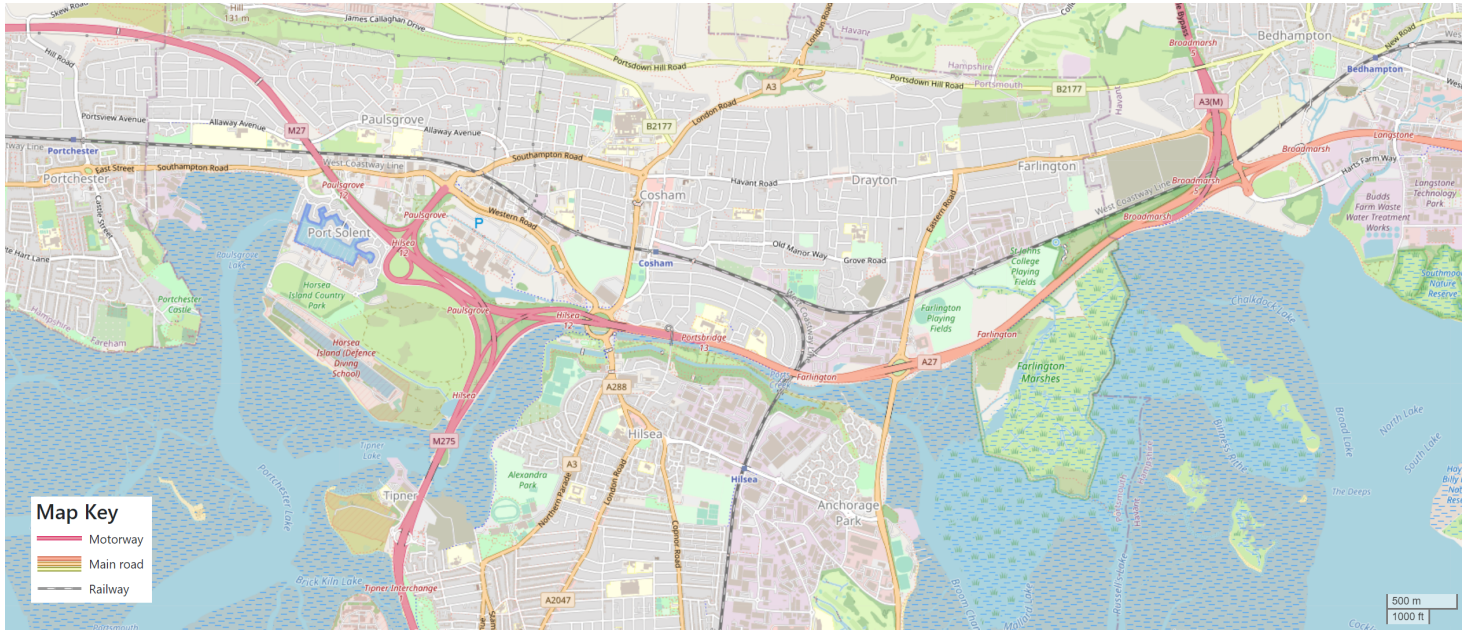

FIGURE 5.1: Map of demonstration area, credit: © OpenStreetMap contributors

5.2 Demonstration area

The area of the British railway network selected to demonstrate this work is part of the Havant train describer (TD) on the south coast of England, shown in Figure 5.1. The area comprises of double-track and contains four stations: Cosham, Bedhampton, Hilsea and Portchester. Cosham is a 'main' station where most passenger trains are scheduled to stop; the other 'local' stations are serviced less frequently. There are three- and four-aspect signals in the area and two level crossings, one outside Cosham station (to the east) and the other outside of Bedhampton (to the west).

The area was selected for several reasons. Firstly, it has some simplifying features, such as no locations where trains frequently originate, terminate, join or separate. Secondly, the area contains three junctions, providing conflict locations to demonstrate the conflict resolution models. Lastly, freight and passenger services pass through the area, allowing a comparison of the effectiveness of the modelling methods on different train types.

5.3 Data sources

This research has only utilised open data sources; such data are free to obtain and have few conditions for their use. This decision removed a dependence on Network Rail (the sponsor of this work) to provide data. It also ensures that the work is easier to replicate or extend in future than if commercially sensitive data were used. The primary data source has been Network Rail's open data feeds (Network Rail, 2020), with additional data from Network Rail's delay attribution data files[1]. Restricting this research to the use of open data sources does lead to some limitations. These will be highlighted in discussions throughout the remainder of this thesis.

The data feeds and datasets used in this research are listed below, and some justification for their selection is provided. The descriptions of each dataset will be limited to the features of the data that are relevant to this work.

CORPUS reference data

The CORPUS (Codes for Operations, Retail and Planning - a Unified Solution) computer system provides location reference data. The data are stored in JSON format, and the open data file is updated monthly. The data can translate the different codes representing locations on the network. In particular, the translation required for this

[1]These are not provided in a data feed, instead they can be downloaded from their webpage on open data: https://www.networkrail.co.uk/who-we-are/transparency-and-ethics/transparency/open-data-feeds/

research is between STANOX codes (station numbers used in some other data sets) and TIPLOC codes.

SMART reference data

The SMART (Signal Monitoring and Reporting of Trains) computer system aggregates data from train describers. This feed is a file of reference data from the SMART system stored in JSON format, and the open data file is updated monthly. The data provides STANOX codes for locations on the network and the associated movements between blocks that relate to arrivals or departures from that location. The data also provide offset values used to estimate arrival and departure times.

TD C-Class data

The train describer (TD) C-Class data feed is available for most areas of the British railway and provides low-level detail about the position of trains through a network of berths. A berth usually represents a block section of track on the approach to a main signal, and all the examples in this thesis will involve a one-to-one relationship between signals and berths. A sequence of four alpha-numeric characters identifies each berth.

Trains are identified by headcodes (also consisting four alpha-numeric characters), and the dataset consists of records each time the front of a train 'steps' (moves) into a berth. The timestamps of the movements are given to the nearest second. Headcodes can be interposed manually by a signaller if there is a obstruction or maintenance work; such action ensures that a berth is considered occupied and therefore it will not be possible for the signalling system to provide movement authority into the berth.

The data are transmitted in JSON format in real-time. This data feed was selected to represent train movements as it is the most detailed open-source track-based data available on the British network. While track-based data have some disadvantages (see Section 1.6), no train-based open data are available.

The C-Class data do not provide information on the state of signals or infrastructure. Complimentary S-Class data data exist to provide some information about infrastructure, such as signal or route status. However, the S-Class data has some limitations: it is not available for all TD areas with C-Class data, requires significant work to decode for each area and does not contain the same data for all areas where it does exist. The data on signal status from the S-Class feed is also limited as it only indicates whether a signal is currently at its most restrictive aspect or not; it does not provide details on other aspects. The S-Class data has not been included in this work.

Timetable data

There are two types of files: a 'full' and an 'update' extract. The full extract contains all timetables, and the update extract includes any changes applied to the full data. The update file will have been created as part of the short-term planning (STP) process. Files for the full timetable are provided weekly, and update files are provided daily.

There are two formats for timetable data: JSON and CIF (common interface file). The CIF format is the industry standard and is a plain text file with each record a fixed length of 80 characters and is the format used in this work.

Timetable data are also released as part of the very short-term planning (VSTP) process. These data are not used as a data source because archived data for the VSTP timetable feed was not readily available. The timetables in the VSTP feed are of low volume, so although the decision not to include these data will mean that some schedules are not included or updated, the impact will be small.

TRUST train movement data

The train movement data are in JSON format and provide records from the TRUST (Train Running Under System TOPS) computer system. TRUST takes the schedule of each train and records the actual departure, arrival and passing times at locations on the schedule. These data are less granular than the TD C-Class data and have, therefore, not been used to represent train movements; however, two fields in these data have been used to match the timetable data to the TD C-Class data.

Delay attribution data

The delay attribution data are stored in Microsoft Excel files, and updates appear monthly. The data contain the causes of delays for train services where the overall delay was over a specified threshold – currently set at three minutes.

5.4 Infrastructure dataset

5.4.1 Construction

This section describes how features of the infrastructure have been identified and modelled using the CORPUS, SMART and TD C-Class data. Some features that have been excluded are also discussed, as these may impact results in later chapters. This work is not the first application of Network Rail's open data to create a model of the network,

and the websites of Railcam UK Ltd[2] and OpenTrainTimes Ltd[3] helped validate this work.

Berths and steps

In order to construct a network model from the TD C-Class data, a reasonably large data sample is required – the data used in this work covered the date range from 2018-04-06 to 2021-09-16 (inclusive), which is over three years. Each record of the C-Class data represents a train movement and consists of the headcode of a train, the timestamp of the movement and the berths the train is moving from and to. This information can identify the berths in the area of interest and the valid steps (movements) between berths.

The demonstration area consists of 40 berths. Most berths have only one step in and out; however, some berths have two steps in or out around the junction areas. Erroneous steps occasionally appear in the data feed. These do not appear frequently, and a threshold can be applied to filter out invalid steps.

Stations and TIPLOCs

The SMART and CORPUS reference data are used to identify the location of passenger stations and other TIPLOCs on the network. A TIPLOC is not unique to a berth; for example, a passenger station will have TIPLOC in each berth that contains a platform. Table 5.1 provides the locations of the TIPLOCs in the demonstration area. The berth identifiers have been prefixed with 'HT' to make it clear they are in the Havant TD area.

Conflict locations

The C-Class data were mined to identify conflicting steps. These are pairs of mutually exclusive steps, and trains cannot travel across them simultaneously. The conflicts were identified by comparing all pairs of steps in the data and recording the minimum time difference between a train traversing each step. Two steps are not mutually exclusive if their minimum time difference is zero.

This approach was tested beyond the demonstration area and was found to be successful at identifying pairs of steps that were not mutually exclusive. However, some pairs of steps were erroneously labelled as conflicting. Typically, this occurred for steps that were not very common, and there was not as much data available. This led to an additional check that both steps could be associated with the same TIPLOC.

[2] https://railcam.uk/
[3] https://www.opentraintimes.com/

The pairs of conflict steps identified in the demonstration area are listed in Table 5.2, along with their associated TIPLOCs. Two types of conflict can be identified. In this work, conflict pairs with IDs 1-3 will be termed 'crossing' conflicts, as the pairs do not contain the same berths. Conversely, conflict locations with IDs 4-6 are 'joining' conflicts, as both steps in the pair move into the same berth.

Signals

In the demonstration area, one main signal is associated with each berth, providing the movement authority out of the berth. These are the only type of signal to be modelled. Signal statuses can only be inferred by the location of trains on the network at any given time. For example, if a train is in berth A, all berths that step into berth A will have their signals at the most restrictive aspect. This is sufficient for the current work but could be improved in future work by incorporating the TD S-Class data into the model.

Level crossings

The infrastructure model does not contain the locations of level crossings, as these are not provided in the data sources.

Berth	TIPLOC	Platform
HT_0029	Bedhampton	1
HT_0032	Bedhampton	2
HT_0316	Cosham	1
HT_0321	Cosham	2
HT_0318	Cosham Junction	-
HT_0325	Cosham Junction	-
HT_0400	Cosham Junction	-
HT_0039	Farlington Junction	-
HT_0040	Farlington Junction	-
HT_0401	Farlington Junction	-
HT_0046	Hilsea	1
HT_0043	Hilsea	2
HT_0308	Portchester	1
HT_0313	Portchester	2
HT_0041	Portcreek Junction	-
HT_0042	Portcreek Junction	-
HT_0320	Portcreek Junction	-

TABLE 5.1: Location of TIPLOCs in the demonstration area

Conflict pair ID	Conflict A		Conflict B		TIPLOC
	From berth	To berth	From berth	To berth	
1	HT_0037	HT_0401	HT_0042	HT_0040	Farlington Junction
2	HT_0323	HT_0325	HT_0401	HT_0318	Cosham Junction
3	HT_0044	HT_0042	HT_0325	HT_0041	Portcreek Junction
4	HT_0042	HT_0040	HT_0400	HT_0040	Farlington Junction
5	HT_0320	HT_0318	HT_0401	HT_0318	Cosham Junction
6	HT_0325	HT_0041	HT_0039	HT_0041	Portcreek Junction

TABLE 5.2: Location of conflicts in the demonstration area

5.4.2 Visualisation

A visualisation of the demonstration area is shown in Figure 5.2. Visualising the network area also helps to validate the conflict pairs listed in Table 5.2.

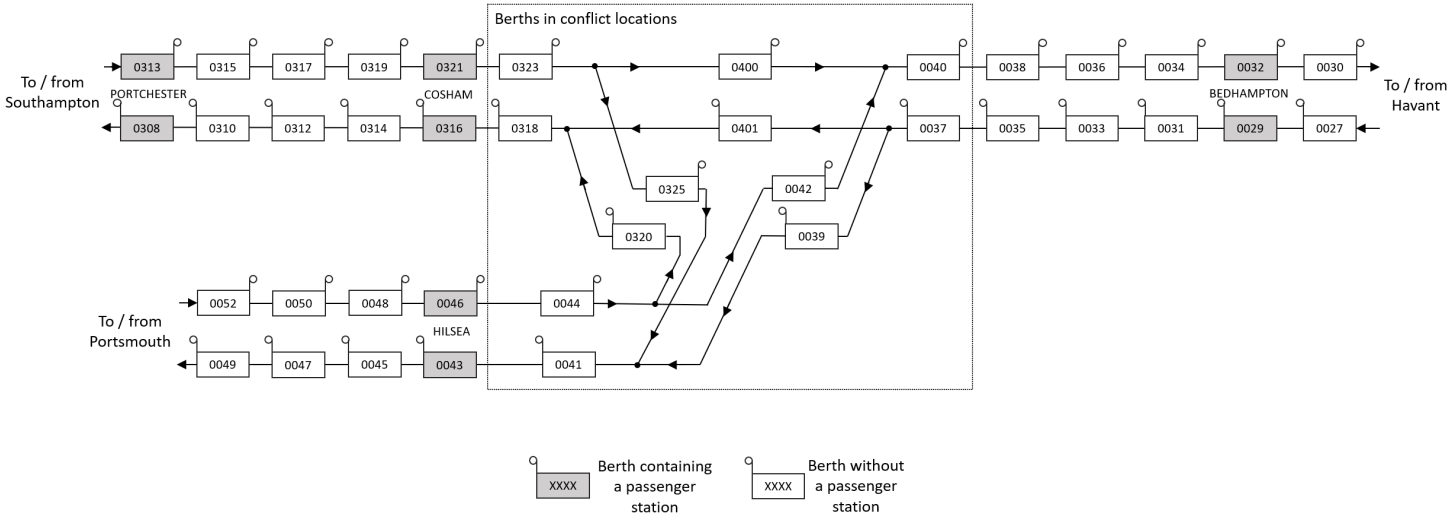

FIGURE 5.2: Model of demonstration area

5.5 Train movements dataset

5.5.1 Construction

The TD C-Class data contain records for train movement, line blockages and maintenance work. The dataset of train movements was created by matching TD C-Class data to long- and short-term timetables. Both data sources contain a headcode for each movement and timetable, respectively. However, the headcode assigned to a train in a timetable does not always appear for the associated records in the C-Class data; this happens mainly for freight trains where deliberate obfuscation of the headcodes in the live C-Class feed can occur for security reasons. The TRUST train movement data matches the two data sources: it contains the same headcode that appears in the C-Class data and another identifier used in the timetable. Neither value is unique, and the date and time values are also required to match the records.

The TRUST train movement data were also required to match the TD C-Class data with the delay attribution data. The delay attribution data were used to identify movements in the TD C-Class data where a delay occurred. The timetable data provides a category for each train, and these were used to identify three main types of train: passenger trains, freight trains, and empty coaching stock (ECS).

Table 5.3 shows a sample from the dataset, reconstructing a train's journey through the demonstration area. Each record represents the time the front of a train moves into a berth, and the travel duration is the total time it takes for the front of the train to travel into the next berth. The travel duration will become the value the machine learning models will predict. The berths are not of equal length; therefore, the travel durations through different berths are not strictly comparable. However, the data show that the train spent significantly longer in a berth where it had to wait for another train to clear a conflicting route (HT_0401). Longer durations also occur at stations where the train dwells, with a delay due to a disorder affecting the duration through HT_0316.

Table 5.3 does not show the complete list of features in the dataset, and a full list of features used in the machine learning models is provided in the following chapter.

Headcode	Berth	Time of day (seconds)	Travel duration (seconds)	Includes dwell time	Train waits for conflict	Number of conflicts	Direct delay
1J70	HT_0027	52042	20	FALSE	FALSE	0	No delay
1J70	HT_0029	52062	31	FALSE	FALSE	0	No delay
1J70	HT_0031	52093	20	FALSE	FALSE	0	No delay
1J70	HT_0033	52113	28	FALSE	FALSE	0	No delay
1J70	HT_0035	52141	38	FALSE	FALSE	0	No delay
1J70	HT_0037	52179	36	FALSE	FALSE	0	No delay
1J70	HT_0401	52215	568	FALSE	FALSE	1	No delay
1J70	HT_0318	52783	69	FALSE	FALSE	0	No delay
1J70	HT_0316	52852	370	TRUE	FALSE	0	Disorder/drunks or trespass
1J70	HT_0314	53222	73	FALSE	FALSE	0	No delay
1J70	HT_0312	53295	44	FALSE	FALSE	0	No delay
1J70	HT_0310	53339	64	FALSE	FALSE	0	No delay
1J70	HT_0308	53403	134	FALSE	FALSE	0	No delay

Table 5.3: Sample of train movement data

Berth	Number of routes in	Number of routes out	Main station	Local station
HT_0029	1	1	No	Yes
HT_0033	1	1	No	No
HT_0037	1	2	No	No
HT_0318	2	1	No	No
HT_0316	1	1	Yes	No

TABLE 5.4: Demonstration berths used in visualisations of the train movement dataset

5.5.2 Visualisation

This section illustrates some features of the train movement dataset that influence the travel duration of trains through berths. The features presented are characteristics that will be relevant in later chapters. Examples include the berth the train travels through, the power type, and the route the train is taking. Five berths from the demonstration area (listed in Table 5.4) are used to visualise the data. The first seven batches of data were used to construct all the charts.

Figure 5.3 shows histograms of the travel duration of electric passenger trains through the five berths. Some data were removed before creating the histograms to represent only unhindered movements. Removed records include those associated with a delay and those where the berth's signal has been inferred to display a yellow or red aspect.

All the histograms are skewed to the right. The skewness is unsurprising as there is a lower limit on travel duration determined by the physical capabilities of a train and operating conditions, such as speed limits. Conversely, there is no upper limit on travel duration: a train could break down and not move for hours. As signal statuses are inferred from the data, not all records where the signal is at a restrictive aspect will have been identified. Therefore, there may be high travel durations due to the train being held at a red signal.

The distribution for berth HT_0033 has a typical shape for berths that do not contain a station and have only one route in and out. Both HT_0029 and HT_0316 contain passenger stations; however, their histograms in Figure 5.3 show noticeably different distributions of the travel duration. The difference is due to HT_0029 containing a 'local' station and HT_0316 a 'main' station. Both show two peaks in the distribution of travel durations, representing trains that pass through the berth without stopping and those that dwell at the station. The fact that most passenger trains stop at the station in HT_0316 explains the much larger peak at longer travel durations for this berth.

Run durations through a berth that include dwelling time are typically longer and subject to more significant variation than those without a scheduled station stop. Figure 5.4 demonstrates this with boxplots of travel duration through HT_0029 and HT_0316;

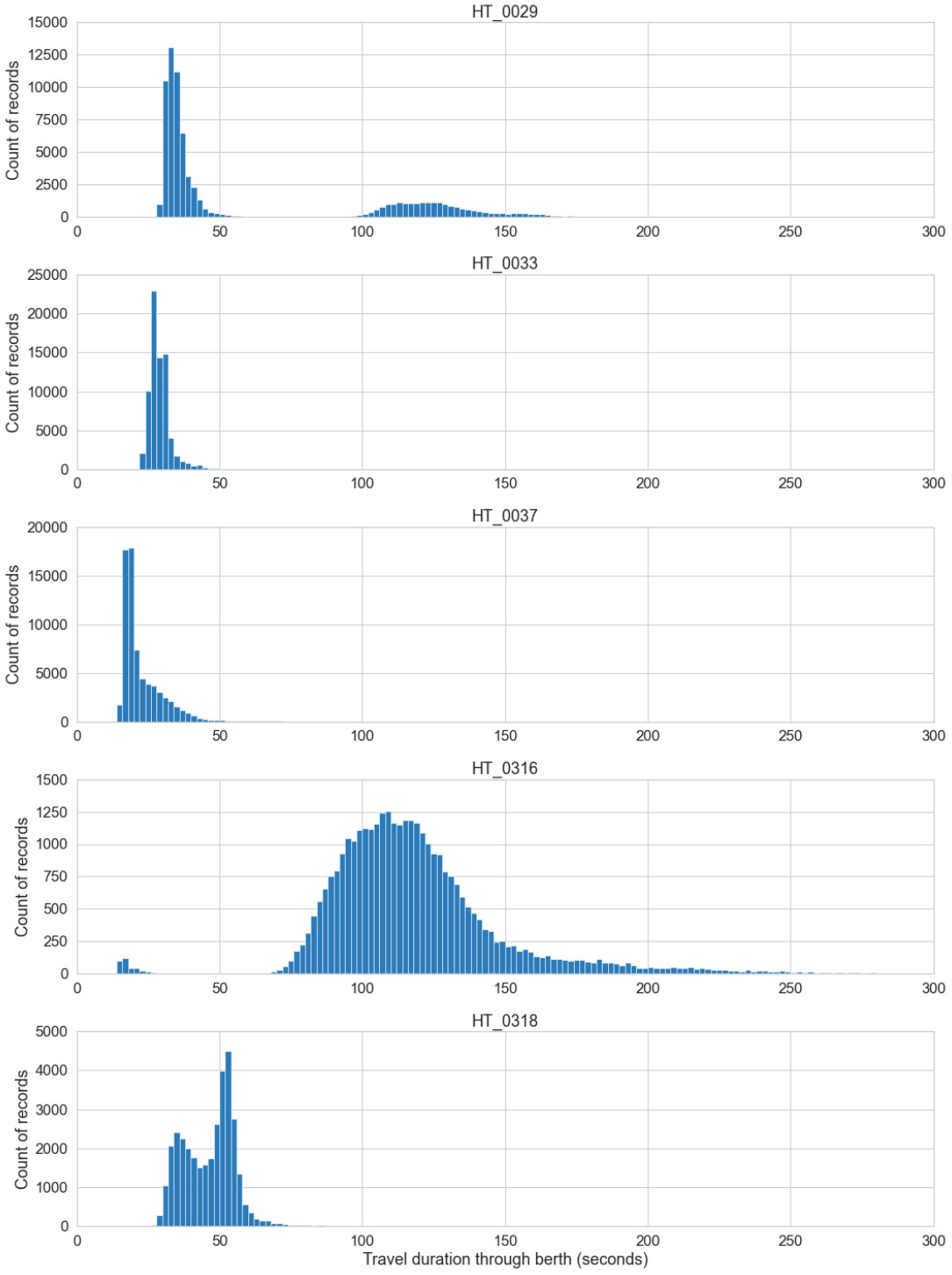

FIGURE 5.3: Histograms showing travel duration by berth for unhindered electric-powered passenger trains

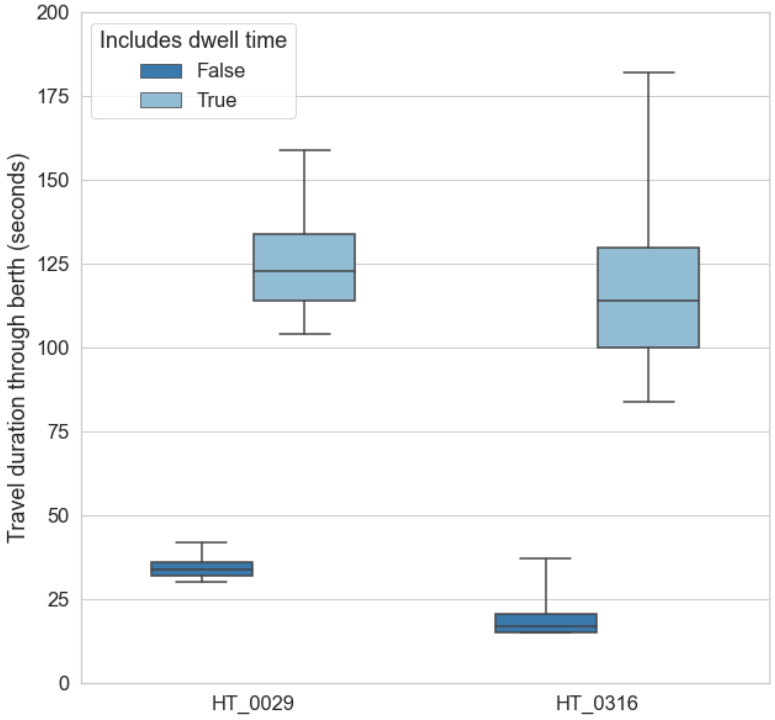

FIGURE 5.4: Boxplots of unhindered travel durations through HT_0029 and HT_0316 for electric passenger trains

the data are divided by whether the trains dwell at the station. The whiskers on the boxplots represent the 5^{th} and 95^{th} percentiles[4].

The histograms for HT_0037 and HT_0318 in Figure 5.3 also show different distributions from HT_0033. There are two possible routes out of berth HT_0037 and two into berth HT_0318. These characteristics affect the travel duration through a berth, and Figures 5.5 and 5.6 show boxplots for travel time through berths HT_0037 and HT_0318 divided by the next and previous berths, respectively.

Many other features affect travel duration that are not illustrated here. Examples include temporal features (such as time of the day or day of the week), the train type (passenger, freight, ECS), the power type of the train (electric or diesel) and the inferred signal aspect.

[4]The whiskers on boxplots are often set at 1.5 multiplied by the interquartile range; however, this is not necessarily suitable when the data are not normally distributed.

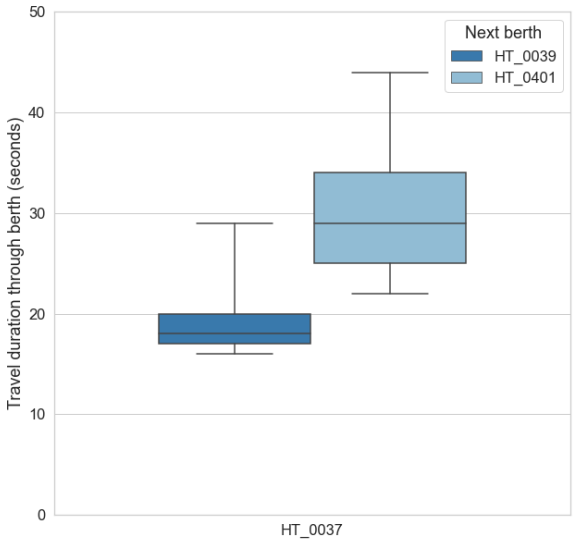

FIGURE 5.5: Boxplots of unhindered travel durations through HT_0037 by the next berth for electric passenger trains

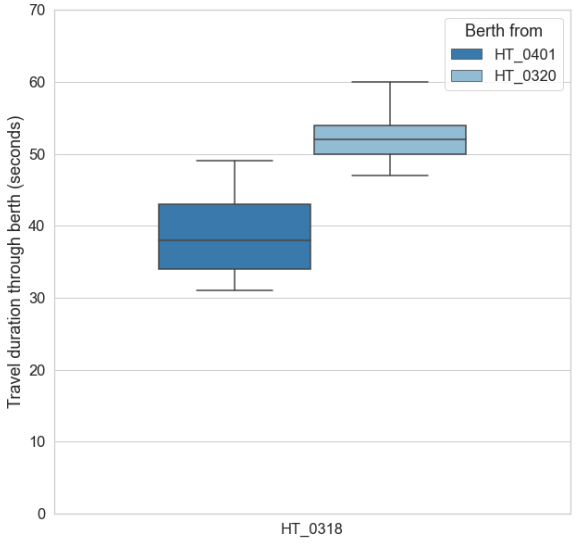

FIGURE 5.6: Boxplots of unhindered travel durations through HT_0318 by the previous berth for electric passenger trains

5.6 Train conflicts dataset

5.6.1 Construction

The train conflicts dataset was constructed by matching the TD C-Class data to the schedule data, using the same method as the train movements. The data were filtered on records that were within seven steps of one of the steps in the conflict locations. They are then mined to find pairs of trains approaching the distinct steps in a conflict location.

Table 5.5 shows a sample of the dataset constructed for the first pair of conflict locations listed in Table 5.2. (HT_0037 to HT_0401 and HT_0042 to HT_0040). For each record, the train due to travel from HT_0037 to HT_0401 is listed as train A. The machine learning models will predict whether or not the first train gets priority at the junction.

The data in Table 5.5 show that trains do not always travel in their scheduled order – records 1 and 4 provide examples of trains travelling out of order. Records 3 and 4 show freight and ECS trains having priority over passenger trains, where the passenger trains are further away from the conflict location. Conversely, record 2 shows an ECS train waiting at the conflict location to prioritise a passenger train scheduled to pass the conflict location first.

The same pair of trains can appear many times in the dataset as one record is created for each combination of berths on the trains' journeys to the junction. Table 5.5 only shows a sample of the features available in the dataset; a complete list of features used in the machine learning models is provided in Chapter 8.

5.6.2 Visualisation

The first and fourth conflict locations from Table 5.2 have been selected to illustrate some characteristics of conflict resolution. Both relate to the same TIPLOC, but the first conflict location is a pair of 'crossing' conflicts, and the fourth is a pair of 'joining' conflicts.

Figure 5.7 displays heatmaps for each conflict location. The x and y axes represent the number of steps each train needs to take before reaching the conflict location. The values in the heatmap are the percentage of records that the train travelling across the conflict on the y-axis has priority.

For the first conflict location, there is a strong correlation between the distance of the trains to the conflict location and which train has priority. If a train is further away from the conflict location than the other train, it is less likely to be granted priority.

	Record ID	**1**	**2**	**3**	**4**
	Train with priority	B	A	A	A
	Train scheduled first	A	A	A	B
	Date	20180412	20180412	20180412	20180412
Train A	**Headcode**	1J86	1N97	788B	5Y05
	Number of steps to conflict	4	5	2	3
	Train type	Passenger	Passenger	Freight	ECS
	Current berth	HT_0031	HT_0029	HT_0035	HT_0033
	Category	Express	Express	Unknown	Departmental
	Destination	Southampton	Southampton	Merehead Quarry	Southampton up goods loop
Train B	**Headcode**	1P64	5P14	2S64	1P42
	Number of steps to conflict	2	1	3	6
	Train type	Passenger	ECS	Passenger	Passenger
	Current berth	HT_0044	HT_0042	HT_0046	HT_0052
	Category	Express	ECS	Ordinary	Express
	Destination	London Waterloo	Petersfield	Littlehampton	London Waterloo

TABLE 5.5: Sample of data from the train conflict dataset for conflict location pair HT_0037 to HT_0401 and HT_0042 to HT_0040. For all records, Train A will travel across HT_0037 to HT_0401, and Train B will travel across HT_0042 to HT_0040

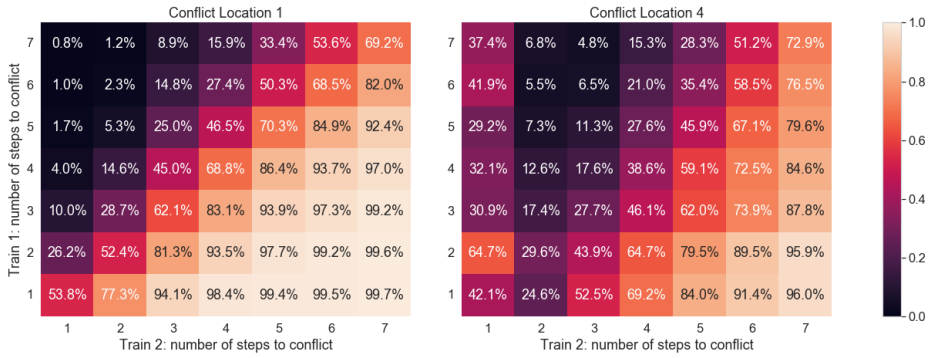

FIGURE 5.7: Heatmaps showing the percentage of records where train A has priority based on distance to conflict location, for conflict locations 1 and 4. Train A is the train travelling across HT_0037 to HT_0401 for conflict location 1 and HT_0042 to HT_0040 for conflict location 4.

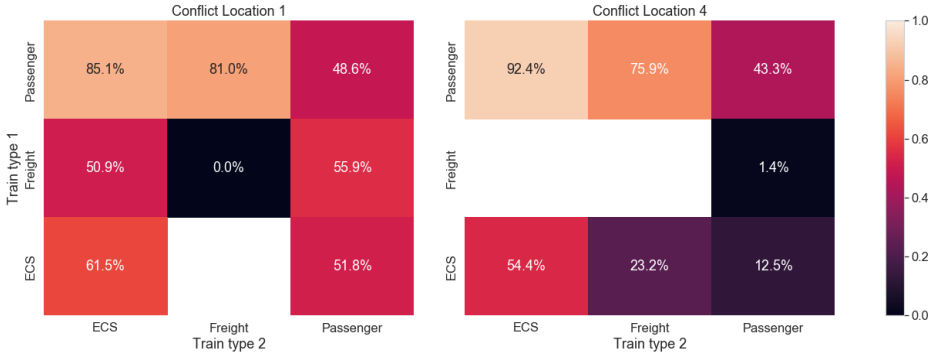

FIGURE 5.8: Heatmaps showing percentage of records where train A has priority based on train type, for conflict locations 1 and 4. Train A is the train travelling across HT_0037 to HT_0401 for conflict location 1 and HT_0042 to HT_0040 for conflict location 4.

The fourth conflict location does not show the same pattern, which may be because the conflict is a 'joining' conflict. At the distances under consideration, express passenger trains are likely to get priority at these junctions over slower services, as the second train to pass the conflict location would end up following the first.

Figure 5.8 shows two similar heatmaps for the conflicts, but this time with train type on the x and y axes. These show that passenger trains are more likely to get priority over ECS and freight trains for the fourth conflict location than the first. It is likely that this is also due to the fourth conflict location being a 'joining' pair of conflicts, and the order in which the trains travel impacts their onward journeys.

5.7 Conclusion

This chapter has described how the open data sources from Chapter 4 were used to construct three datasets that are key to this research. The datasets concerned infrastructure, train movements and conflicts. The next chapter will use train movement data to build machine learning models to predict the travel time of trains through berths in the demonstration area. The analysis shown in this chapter will inform decisions around the preparation of the data prior to the machine learning model being built.

Chapter 6

Predicting train movements

6.1 Introduction

The previous chapter explained the construction of the datasets used in this research. This chapter demonstrates the use of the train movement dataset to develop supervised machine learning models that predict the travel durations between discrete points on the network. The demonstration area introduced in the last chapter is also used as a case study to illustrate the work. The five berths used to visualise features of train movements in the previous chapter will also be used in this chapter to present examples of the results.

Section 6.2 reaffirms the aim of the models in light of the analysis shown in the previous chapter. The process for constructing the machine learning models was introduced in Chapter 4 (see section 4.3.3), and this chapter follows that process. The preparation for the modelling work is discussed in Section 6.3. It includes the segmentation of the dataset, the selection of the benchmark models and machine learning algorithms, the selection and transformation of data features, and the choice of evaluation metric.

Section 6.4 covers the exploration phase of modelling; in this phase, a wide variety of models and hyperparameters are applied to create predictive models. The most successful models from the exploration phase are carried forward to the selection phase, the results of which are presented in 6.5. From this, individual models are selected and built for a selection of quantiles with a final evaluation in Section 6.6. Section 6.7 discusses the contributions of the research and further work that could follow on from the results.

6.2 Requirements of the models

The aim of this research is to evaluate the use of machine learning-assisted simulation modelling (see Section 1.2). Therefore these models have to be developed so that they can be integrated into the simulation model. To that end, the travel duration predictions must be made between discrete points on the network that will be modelled in the simulation – these are will be berths.

From the analysis in the previous chapter, two different movement types have been identified for consideration in this work. The first will be termed a restricted movement, whereby when a train enters a berth, the status of the signal to exit the berth is inferred as being at its most restrictive aspect. In this case, the train is at least prepared to stop before exiting the berth and, in many cases, will stop. All other records are classed as unrestricted movements, whereby the signal status to exit a berth is not inferred to be at its most restrictive.

For restricted movements, the aim is to predict the duration between the obstruction or conflict being cleared and the train moving to the next berth. The selection of this target variable is not perfect; for example, there will be examples where movement authority has been provided before the train has reached the signal and, therefore, never comes to a stop. However, it is necessary to represent restricted movements within a simulation model, so this work will, in part, investigate the suitability of this target variable.

For unrestricted movements, the aim is to predict the travel duration through a berth. As this work uses track-based data, obtaining accurate data on dwell durations is not feasible. Therefore, dwell durations will not be explicitly predicted. For berths that include passenger stations, the predicted travel duration will include the dwell duration for those trains due to stop at the station.

After the machine learning models have been trained, it will be necessary to save the hyperparameters and structure of the models so that they can be loaded into the simulation model.

6.3 Preparation

6.3.1 Linking data records

Each record in the dataset is linked to its five previous movements giving each record a short history. For train movements near the boundary of the demonstration area, the history will include movements outside the demonstration area. Records that do not contain a history of five movements are discarded from the dataset. Following this

process, each record in the dataset consists of six movements: one 'main' movement (of which the travel duration is to be predicted) and five 'historical' movements.

6.3.2 Segmentation of dataset

The data are segmented by the berth of the main movement, the movement type of the main movement, and the train type. The train type will not change between the main and historical movements (the transition between ECS and passenger services, for example, is not modelled). However, the movement type of historical movements can differ from the main movement type. For example, five unrestricted movements could have preceded a restricted main movement. As discussed in the previous chapter, the signal status is only inferred from the data, so the method for identifying restricted movements is unlikely to find all the records that should belong to this group.

Table 6.1 lists the total number of records for each berth, train type and movement type of the main movement showing variation in the number of records in each segment. Supervised machine learning models are built for each data segment with more than 200 records. Machine learning methods require reasonable amounts of training data to be effective, and the threshold of 200 records is somewhat arbitrary. Analysis of the results will consider whether it is a sufficient number of records.

Further to segmenting the data by berth, train and movement type, each record is allocated to a batch number between 1 and 14 based on the date of the main movement. Table 6.2 shows the start date of each batch and, as an example, shows the number of records for unrestricted passenger train movements for berth HT_0029. Even though each batch contains 90 days, there are not the same number of records in each batch due to train schedule fluctuations and data quality. In particular, Batch 9 has significantly fewer records than previous batches, owing to the restrictions that were in place during the Covid-19 pandemic in the UK – only essential travel was allowed, and fewer train services were scheduled.

6.3.3 Benchmark models

Benchmark models are built for each combination of berth, movement, and train type. Benchmark models predict the target variables using more straightforward methods than machine learning. For a machine learning model to be deemed successful, the *least* it has to do is outperform its associated benchmarks.

The benchmarks created in this work are similar to that of Barbour et al. (2018), where the median travel duration represented a benchmark. As the benchmarks are computationally inexpensive, several different approaches are considered, with different benchmarks for the two movement types.

Berth	Unrestricted			Restricted		
	Passenger	**ECS**	**Freight**	**Passenger**	**ECS**	**Freight**
HT_0027	**139,406**	**4,558**	**726**	**588**	87	5
HT_0029	**139,928**	**4,650**	**732**	50	0	0
HT_0030	**138,145**	**7,326**	**731**	**453**	136	3
HT_0031	**140,092**	**4,689**	**735**	25	0	0
HT_0032	**138,424**	**7,429**	**721**	147	26	0
HT_0033	**140,107**	**4,694**	**770**	59	3	0
HT_0034	**137,720**	**7,217**	**719**	**823**	**235**	1
HT_0035	**141,249**	**5,143**	**769**	136	7	2
HT_0036	**137,786**	**7,423**	**727**	168	27	0
HT_0037	**138,948**	**4,964**	**724**	**2,459**	167	41
HT_0038	**137,794**	**7,447**	**729**	17	2	1
HT_0039	**95,534**	**1,950**	14	**8,927**	132	8
HT_0040	**136,448**	**7,187**	**728**	**1,284**	**250**	2
HT_0041	**148,360**	**3,483**	27	**15,134**	**447**	10
HT_0042	**94,375**	**2,840**	17	**6,763**	**489**	0
HT_0043	**161,500**	**3,904**	35	**1,951**	25	0
HT_0044	**156,754**	**5,810**	21	**2,446**	172	0
HT_0045	**163,280**	**3,916**	33	128	5	0
HT_0046	**158,726**	**5,928**	22	**426**	21	0
HT_0047	**163,213**	**3,920**	31	**246**	4	1
HT_0048	**153,459**	**4,545**	15	**1,192**	**422**	0
HT_0049	**162,554**	**3,886**	30	**968**	37	1
HT_0050	**151,786**	**4,933**	3	**256**	34	0
HT_0052	**151,502**	**4,896**	3	153	12	0
HT_0308	**94,444**	**5,439**	**756**	**259**	101	6
HT_0310	**93,562**	**4,671**	**690**	**1,181**	**857**	56
HT_0312	**94,710**	**5,496**	**738**	41	41	2
HT_0313	**94,512**	**5,430**	**709**	**208**	52	0
HT_0314	**94,568**	**5,234**	**729**	153	**306**	11
HT_0315	**95,144**	**5,547**	**716**	**343**	59	0
HT_0316	**88,283**	**5,117**	**719**	**6,477**	**430**	24
HT_0317	**95,672**	**5,957**	**737**	81	13	0
HT_0318	**89,254**	**5,110**	**669**	**5,616**	**539**	74
HT_0319	**95,409**	**5,931**	**738**	**366**	49	2
HT_0320	**53,239**	**2,388**	4	**4,739**	**259**	0
HT_0321	**95,418**	**5,966**	**738**	**259**	15	0
HT_0323	**93,332**	**5,805**	**732**	**2,342**	151	3
HT_0325	**48,656**	**1,038**	12	**10,443**	**814**	7
HT_0400	**35,978**	**2,643**	**561**	**607**	**1,459**	154
HT_0401	**35,122**	**2,779**	**633**	**1,777**	**266**	107
Total	4,724,393	197,289	18,943	79,691	8,151	521
Number of berths with over 200 records	40	40	26	28	13	0

TABLE 6.1: The total number of records for each berth, train and movement type of the main movement

Batch number	Start date	Count of records
1	2018-04-06	10,966
2	2018-07-05	11,266
3	2018-10-03	10,393
4	2019-01-01	11,268
5	2019-04-01	10,866
6	2019-06-30	11,451
7	2019-09-28	9,864
8	2019-12-27	10,647
9	2020-03-26	5,472
10	2020-06-24	9,959
11	2020-09-22	9,227
12	2020-12-21	8,224
13	2021-03-21	10,285
14	2021-06-19	10,040

TABLE 6.2: Start dates of the fourteen batches of data with the count of main records of unrestricted passenger movements for berth HT_0029

Unrestricted movements

Two benchmark models have been constructed for unrestricted movements for each combination of berth and train type. The first candidate divides the training data for each combination into trains due to stop in the berth (i.e. their travel duration includes dwell time) and those that do not. The median travel duration is then calculated for stopping and non-stopping trains for each combination; these values are used to make predictions.

The second model divides the training data by the unique values in more fields, namely:

- Whether the train is stopping in the berth
- The berth the train has stepped from (i.e. which route the train is taking into the berth)
- The next berth the train will step into (i.e. which route the train is taking out of the berth)
- The next TIPLOC the train is scheduled to stop at
- The previous scheduled TIPLOC that the train stopped at
- The power type of the train
- The TOC code of the train

This model takes the median travel time for each combination of the unique values of the fields existing in the training data. These are then used to make predictions. In the event that a combination of values appears in the validation data but does not appear in the training data, then the prediction from the first benchmark is used instead.

Restricted movements

Three benchmark models have been created for restricted movements. The first two benchmarks are the same for the unrestricted movements but with a different target variable. The third benchmark divides the data further using two additional features. The features are:

- Whether the next berth is occupied at the time the train enters the berth
- Whether the train has to wait for a conflicting route to be cleared before moving out of the berth

6.3.4 Machine learning models

As with the benchmark models, the machine learning models are built for each combination of berth, movement, and train type – a similar approach to the 'local' models of Kecman and Goverde (2015). Three broad 'families' of machine learning models were selected as they had been applied successfully to similar problems in the literature: linear models, tree-based ensembles and ANN. The linear and ensemble models are built using the Python library Scikit-learn, and all ANN models use Tensorflow. Like the benchmarks, the predictions are rounded to the nearest integer before evaluation.

Ensemble methods

Random forests and gradient tree boosting are two candidates for representing tree-based ensemble methods. Initial experimentation showed that they had similar predictive capabilities. Of the two approaches, gradient tree boosting was selected to represent ensemble methods as this can be applied using a quantile regression loss function using Scikit-learn. Although the random forest algorithm can be extended to quantile regression (Meinshausen, 2006), no implementations would be immediately compatible with the simulation model. For example, an implementation called Scikit-garden[1] appeared to only be compatible with Scikit-learn up to version 0.21.3, an earlier version than used in the simulation model.

[1] https://scikit-garden.github.io/

A limited number of hyperparameters were selected to vary between models. These included the number of decision trees in each ensemble, the maximum depth of each tree, the maximum number of features that each tree uses to determine the best split, the minimum number of samples required in the node of a tree before performing the split and the learning rate.

Artificial neural networks

ANNs are an alternative approach to tree-based ensembles. Travel durations may be considered part of a time series, with travel durations through previous berths influencing the travel duration through the next berth. Recurrent neurons were designed to apply to sequential problems, and examples were reviewed in the literature where networks with recurrent neurons were applied to similar problems, such as Wen et al. (2020), Huang et al. (2020) and Petersen et al. (2019).

Three different cells were selected to try in ANNs: basic neurons, simple recurrent units and long-short term memory (LSTM) cells. Various architectures of network were selected, where variables such as the number of hidden layers, the number of neurons in each layer, the activation function and the learning rate were all treated as hyperparameters that required tuning. The optimiser Adam (Kingma and Ba, 2015) was used for all ANNs.

6.3.5 Uncertainty

As stated in Section 4.3.1, the models will be developed to represent uncertainty in the predictions. Quantile regression has been selected to predict quantiles of the target value, thus going beyond making point predictions. This method has been chosen over other uncertainty-aware machine learning approaches (such as Bayesian neural networks) because it can be applied to many different algorithms, such as tree-based ensembles and artificial neural networks. Quantile regression also has the advantage of not requiring any assumptions about the distribution of the target variable before training a model. The work of Parslov et al. (2021) (reviewed in Section 3.4.1) also showed a favourable performance of quantile regression compared to Bayesian neural networks when predicting bus travel times.

There are, however, some disadvantages to using quantile regression. For example, with the linear and ensemble methods, a new model has to be trained for each quantile required. There is also no guarantee that the quantile predictions are ordered – known as the 'quantile crossing problem' (Gressmann et al., 2018). Lastly, quantile regression will only *indicate* the distribution of the target variable rather than predicting a full probability distribution or density function.

Quantile regression is carried out in the exploration and selection phases using only the quantile $\tau = 0.5$. In the final evaluation phase, models are built using the following values of τ:

$$0.05, 0.15, 0.25, 0.35, 0.45, 0.5, 0.55, 0.65, 0.75, 0.85, 0.95$$

The same hyperparameters selected for the final model where $\tau = 0.5$ have been used to build models for the other values of τ – although there is no guarantee that these will be close to optimal for other values of τ.

The quantile crossing problem has been addressed by building the models in a specific order, with the model for $\tau = 0.5$ built first. The model for $\tau = 0.45$, $h_{0.45}$, is then built, and the prediction is set to:

$$\hat{y}_{0.45} := min(\hat{y}_{0.5}, h_{0.45}(x))$$

Where $\hat{y}_{0.5}$ is the prediction from $\tau = 0.5$ and $h_{0.45}$ is the model for $\tau = 0.45$. The process is recursive, so for some value of $\tau < 0.45$, the prediction becomes:

$$\hat{y}_{\tau} := min(\hat{y}_{\tau+0.1}, h_{\tau}(x))$$

For values of τ greater than 0.5, the maximum of the two predictions is taken. So the prediction for $\tau = 0.55$ is set to:

$$\hat{y}_{0.55} := max(\hat{y}_{0.5}, h_{0.55}(x))$$

The other values of $\tau > 0.55$ are then:

$$\hat{y}_{\tau} := max(\hat{y}_{\tau-0.1}, h_{\tau}(x))$$

This process ensures that the prediction for a τ_i will always be less than that for τ_j where $\tau_i < \tau_j$.

6.3.6 Feature selection

The data analysis in the previous chapter informed the selection of features for machine learning. Table 6.3 lists the features that were included in the models. The data type of each feature is listed (whether it is categorical, numeric or Boolean) before any transformations have taken place. The train type and movement type are not included in the

list of features, as the models will be built for each combination of train and movement type. Therefore, for each model, these values will be the same for all records.

Some available features in the open data have not been selected. For example, the origin and destination of each train are excluded from the list of features. Initial analysis showed that these did not significantly influence travel durations through berths. Also, as they are categorical variables, their inclusion would lead to a large number of features after transforming the data, which would slow the model-building process.

In contrast to similar work, such as that of Kecman and Goverde (2015), no feature representing 'peak' hours has been created. The notion of 'peak' travel hours is not precisely defined, and creating a binary variable to represent peak hours instead of using time of day may lead to a loss of information. However, a 'pandemic' indicator was introduced to indicate the travel advice issued during the Covid-19 pandemic.

The data for some features, such as TOC code and category, are not provided in the open data sources for freight trains.

Feature name	**Data type**	**Description**
Movement timestamp	Numeric	The time at which the train stepped into the current berth
Local time	Numeric	The time of day in seconds
Weekday	Boolean	True if the day of the week is Mon – Fri; false otherwise
Public holiday	Boolean	True if the day is a public holiday; false otherwise
Permanent	Boolean	True if the service is in the WTT; false if it is part of the STP
Lockdown status	Categorical	A value representing the severity of lockdowns and travel advice during the Covid-19 pandemic
Power type	Categorical	The power type of the train
TOC	Categorical	A code representing the train operating company
Category	Categorical	A code representing the category of the train in the timetable, e.g. express passenger, ordinary passenger
Step	Categorical	The berths that the train is moving from and to concatenated, e.g. HT_0037_HT_0401

Table 6.3 continued from previous page

Feature name	Data type	Description
Travel duration through current berth	Numeric	The target value for unrestricted movements – the travel duration through the berth the train has stepped into
Next berth	Categorical	The next berth that the train will step into
Next berth occupied	Boolean	True if the next berth is occupied; false otherwise
Next + 1 berth occupied	Boolean	True if the next but one berth is occupied; false otherwise
Alternate route occupied	Boolean	True if the current berth has an alternative route out of it (not taken by the current train) where the berth is occupied; false otherwise
Alternate route + 1 occupied	Boolean	True if the next berth has an alternative route out of it (not taken by the current train) where the berth is occupied; false otherwise
Includes dwell duration	Boolean	True if the travel duration through the current berth will include a dwell duration; false otherwise
Wait for conflict	Boolean	True if another train passes across a conflict step while the train is in the current berth; false otherwise
Number of conflicts	Numeric	The number of conflicting trains that passed while the train was in the current berth
Direct delay	Boolean	True if there is a direct delay associated with the record; false otherwise
Next TIPLOC	Categorical	The name of the next TIPLOC in the train's schedule that the train will pass or stop at
Previous stopping TIPLOC	Categorical	The name of the previous TIPLOC the train stopped at
Next stopping TIPLOC	Categorical	The name of the next TIPLOC the train will stop at (may be different from 'Next TIPLOC', which includes passing TIPLOCS)
Scheduled activity	Categorical	Taken from the timetable, a code representing the activity of the train at the next TIPLOC

Table 6.3 continued from previous page

Feature name	Data type	Description
Time difference to previous stopping train	Numeric	For trains that are dwelling in the current berth, the time difference from the train previously dwelling in the berth
Next TIPLOC time difference	Numeric	The time difference between the time the train stepped into the current berth and the scheduled time the train is due at the next TIPLOC
Waiting duration	Numeric	For restricted movements only – the duration in the berth before the conflict location was cleared
Is priority	Boolean	For ECS only, True if the headcode begins with '3'; false otherwise
Travel duration through the previous berth	Numeric	The travel duration through the berth the train has moved from

Table 6.3: The features used for building machine learning models to predict travel duration

6.3.7 Data transformation

Categorical values cannot be used as input to the machine learning algorithms and must be transformed into numeric values. This transformation was achieved using one-hot encoding (Chollet, 2017), whereby a column is created for each unique value of a categorical variable. The transformed column values are then set to '1' or '0', depending on the feature value of the record, as demonstrated in Table 6.4.

	Before transformation	
Record ID	**Power type**	**Category**
1	Electric	Express
2	Electric	Ordinary
3	Diesel	Express
4	Diesel	Ordinary

	After transformation			
Record ID	**Electric**	**Diesel**	**Express**	**Ordinary**
1	1	0	1	0
2	1	0	0	1
3	0	1	1	0
4	0	1	0	1

TABLE 6.4: An example of how one-hot encoding transforms categorical variables

The machine learning algorithms that have been selected require numerical features to be approximately the same scale; otherwise, the optimisation algorithms used will be less efficient. This could be achieved by simply scaling numerical features between zero and one. However, some additional transformations were carried out to some features before the scaling.

The cosine and sine functions were applied to the time of day and movement timestamp features to create two new features to represent each. Applying these two functions to the time of the day to create two new features means that a time of 23:59 is represented as being 'close' to the time of 00:01. Similarly, the movement timestamp is transformed so that the timestamp representing midnight on 1st January 2018 is the same as 1st January 2019.

Other numeric features, such as the travel duration through the previous berth or the time difference to the next scheduled TIPLOC, contain extreme values and are skewed to the right. If the only transform to be applied to these data is to scale them between zero and one, then most data points are extremely close to zero. Instead, a non-linear quantile transform (Krzysztofowicz, 1997) is first applied to the data to transform them into a more normal distribution. The scaling can then occur so that all the numeric data are between zero and one.

The target value and Boolean features do not require transformation. The training data are used for required calculations, and then the transforms are applied to both the training and validation (or test) data. The training data will vary in each modelling phase, so the transformations are re-calculated in each phase.

6.3.8 Data manipulation

As explained in Section 6.3.1, each 'main' record in the dataset has been linked to the five previous records of the train service. This work will consider three different approaches to including the previous records of a train when making predictions, which requires the data into different structures.

Each record in the dataset is associated with a train t that has stepped into berth b_n. The record can be represented as a vector composed of 'static' and 'dynamic' variables: s_t and d_{tb_n}, respectively, see Figure 6.1(a). Static variables do not change throughout a train's journey, such as the power type. Dynamic variables can change with each record of the train's movement, such as the time difference to the next TIPLOC.

The linear and gradient tree boosting machine learning algorithms can only accept one-dimensional input; however, ANNs can be designed for multi-dimensional input. A process has been developed that can create the data in each of the three formats shown

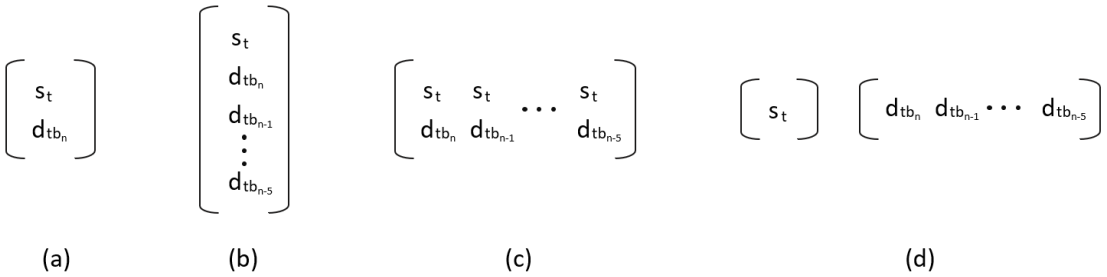

FIGURE 6.1: Representations of input data: (a) a single record, (b) a record concatenated with five previous dynamic records, (c) a two-dimensional representation of six concatenated records, and (d) a static record with six dynamic records

in Figures 6.1(b), (c) and (d). The format of the input data in Figure 6.1(b) is one-dimensional. It contains the static data from the main record and the dynamic data from the main and historical records. Figure 6.1(c) shows a single, two-dimensional input vector with the disadvantage of repeating the static data for all six timesteps. Conversely, Figure 6.1(d) presents two input vectors: a one-dimensional input vector of static data and a two-dimensional input vector of dynamic data.

6.3.9 Evaluation metrics

The mean absolute error (MAE) is used to evaluate the models when $\tau = 0.5$. The MAE is intuitive to understand and is more robust to outliers than alternative metrics, such as the RMSE. The coefficient of determination was not considered a candidate metric for this modelling work as it depends on the variation in the target value (Kuhn and Johnson, 2013b). The travel durations through some of the berths have relatively high variation, particularly in the case of berths containing a passenger station, and it can be possible to create simple models that result in relatively high R^2 values.

The MAE can be used to compare the results for models representing travel through the same berth. However, as the berths are not a standard length and have differing characteristics, such as some that contain a passenger station, the metric cannot provide a comparison between berths.

For all other values of τ, two metrics have been derived to evaluate the models. The first metric will be called the expected quantile division (EQD) and is a Boolean value. If a model has been built using quantile regression with, for example, $\tau = 0.25$, then approximately 25% of the target values should be less than their predicted values and 75% above. For the models built in this work, the predictions are rounded to integers, so many predictions may equal the actual value. Therefore, for a given value of τ and n records in the dataset, EQD is defined as 'True' if there are fewer than $\tau \times n$ records smaller than their predictions and fewer than $(1–\tau \times n)$ records greater than their predictions. The EQD is 'False' otherwise.

	Unrestricted			**Restricted**	
	Passenger	ECS	Freight	Passenger	ECS
Benchmark 1	1	5	17	8	3
Benchmark 2	39	35	9	13	5
Benchmark 3	-	-	-	7	5

TABLE 6.5: The number of benchmark models that provided the lowest MAE for each movement and train type

The second metric will be called the mean quantile error (MQE) and has been derived from the quantile loss function for a given τ:

$$\text{MQE} = \frac{1}{n}\sum_{i=1}^{n}(\tau \cdot max(y_i - \hat{y}_i, 0) + (1-\tau) \cdot max(\hat{y}_i - y_i, 0)) \tag{6.1}$$

Where there are n records in the test data, y_i is the target value for the i^{th} record, and $\hat{y}_i$ is the predicted value. This metric can be used to compare the benchmark and machine learning models where a lower value is preferable.

6.4 Exploration phase results

As explained in Chapter 4, the first seven batches were used as training data in the exploration phase and the eighth batch as validation data. The aim is to identify whether machine learning models can outperform the benchmark models and the types of machine learning algorithms that are most effective.

Table 6.5 shows the number of berths for which each benchmark model produced the lowest MAE on the validation dataset. None of the benchmarks consistently outperformed the other across the different train and movement types. For each berth, movement and train type, the MAE of the machine learning models will be compared to the lowest MAE produced for that combination by the benchmarks.

The linear and gradient boosting models took one-dimensional input of the form shown in Figure 6.1(b). ANNs were constructed using different input forms, creating five broad groups of ANN. ANNs with basic neurons also took one-dimensional input. ANNs with recurrent neurons and LSTM cells were constructed using two-dimensional input (as per Figure 6.1(c)) and two inputs of one dimension and two dimensions (as per Figure 6.1(d)).

For the models with one-dimensional input, not all the historical dynamic data were used and instead, the travel time through the previous berth was the only feature included. All the dynamic features for the main record were still used in the input. When all the historical dynamic features were included, this led to a large number of input features.

	Unrestricted			Restricted		
ML Model	**Passenger**	**ECS**	**Freight**	**Passenger**	**ECS**	**Total**
Linear regression	0	0	0	0	0	0
Gradient boosting	15	30	22	20	3	90
ANN with basic neurons	1	0	1	0	1	3
ANN with RNN cells and single 2D input	3	1	0	0	3	7
ANN with RNN cells and two inputs (1 x 1D and 1 x 2D)	15	7	2	3	5	32
ANN with LSTM cells and single 2D input	0	0	0	3	0	3
ANN with LSTM cells and two inputs (1 x 1D and 1 x 2D)	6	2	1	2	1	12

TABLE 6.6: The number of machine learning models that provided the lowest MAE for each movement and train type in the exploration phase

Early results showed that the gradient boosting algorithms produced worse results than when only the previous travel times were included.

Hyperparameters were selected using a random search from pre-determined lists. A mean number of 99 machine learning models were built for each berth, movement and train type combination. For each combination, it was possible to build a machine learning model that produced a lower MAE on the validation data than the benchmark models. There were some berth, movement and train type combinations where machine learning models could find more significant improvements on the best benchmark model than others. For example, the smallest improvement was a reduction of 2% of the benchmark MAE, and the most significant improvement was a reduction of 65%.

There were also combinations of hyperparameters for machine learning models that produced higher MAEs than the benchmarks. This result demonstrates the importance of the benchmarks in aiding the evaluation of machine learning models.

The types of machine learning models that produced the lowest MAE for each combination of berth, movement and train type are recorded in Table 6.6. The linear regression model did not outperform the other machine learning models for any berth and train

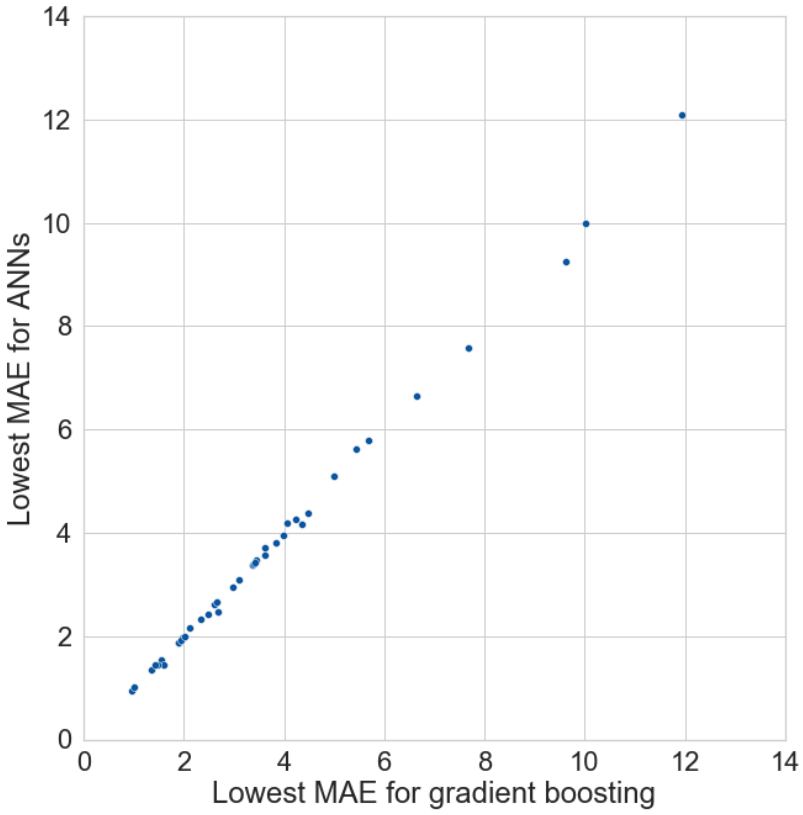

FIGURE 6.2: The lowest MAE for a gradient boosting and an ANN in the exploration phase for models representing unrestricted passenger movements

type combination. No single type of model consistently outperformed the other. Indeed, in most cases, the lowest MAE scored by a gradient boosting model and ANN for each combination of berth, movement and train type were reasonably close. Figure 6.2 shows a scatter plot of the lowest MAE achieved by a gradient boosting against the lowest MAE for an ANN for unrestricted passenger train movements (there is one point for each berth). The scatter plot demonstrates that both model types could achieve similar MAEs with appropriate hyperparameters.

Moreover, the ANN models proved to be slower to train than the gradient boosting models. The ANN models would take hours to train, whereas the gradient boosting models could be measured in minutes.

6.5 Selection phase results

This modelling phase aims to select a single machine learning model for each berth movement and train type combination. The exploration phase found that ANNs could not make predictions with significantly lower MAEs than gradient tree boosting models

(see Figure 6.2) and also took longer to train. Therefore, only gradient boosting models were carried forward to this phase. For each combination of berth, movement and train type, the four gradient boosting models that produced the lowest MAEs were built in this phase with a few variations around some hyperparameters to tune the models further.

As described in Chapter 4, the benchmark and machine learning models were evaluated by taking the mean of the MAEs across the sequential validation batches. For two combinations of berth, movement and train type, the machine learning models could not outperform the benchmarks. Both were restricted movements and had small amounts of training data compared to other combinations. This result may imply that the minimum number of training samples required for machine learning models (currently set at 200) should be increased. However, there were also many combinations with small quantities of training data which were successful in outperforming the benchmark.

The sequential validation across batches nine to thirteen can be used to investigate how well the models generalise over time. Figure 6.3 shows the MAE by batch for berth HT_0318 for unrestricted movements of all three train types. The MAE is consistent across all the validation batches for passenger train movements. This result shows that the predictability of travel durations through this berth does not appear to be affected by timetable changes throughout the period or external factors such as the Covid-19 pandemic. The predictions for ECS and freight trains are worse than for passenger trains: the MAEs are higher than for passenger trains, and there is no consistency in the scores on the validation batches.

The poorer performance of the models for freight and ECS could be because there are fewer training records available. There could also be additional features that are not available in the open data that would improve the prediction. For freight trains, this could be the type of freight being transported or the length of the train.

Figure 6.4 shows the scores across the validation batches for restricted movements in HT_0318. The values of the MAE for passenger trains are less consistent across the batches than the unrestricted movements but are still lower than the scores for ECS.

The trends shown in Figures 6.3 and 6.4 do not occur across all the berths. Figure 6.5 shows the MAEs by batch for unrestricted movements for berth HT_0316. This plot shows higher MAEs for passenger trains than ECS and freight with less consistency between the batches. The differences compared to HT_0318 will be because there is a main station in HT_0316 and most passenger trains dwell there. There is more variation in dwell durations than in run durations for passenger trains, and in general, those berths with passenger stations produced higher MAE scores than those without.

Batch nine has the highest MAE for passenger trains, which coincides with the start of the travelling restrictions in the UK owing to the Covid-19 pandemic. This result

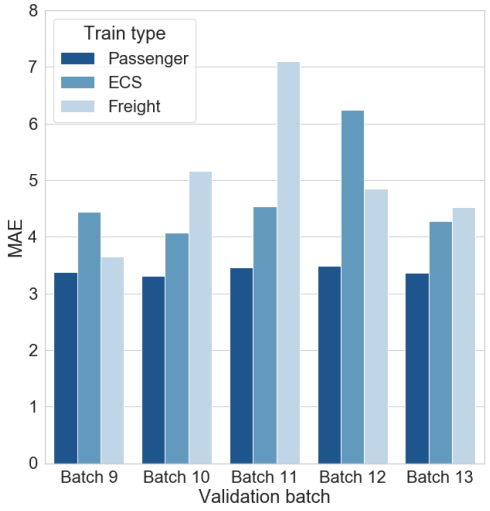

FIGURE 6.3: The MAE by batch and train type for unrestricted movements through HT_0318

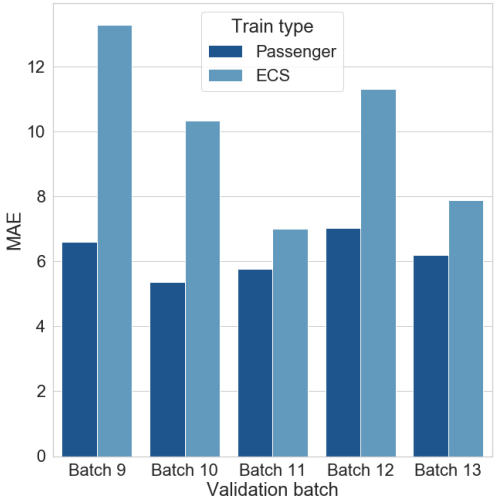

FIGURE 6.4: The MAE by batch and train type for restricted movements through HT_0318

suggests that dwell durations may have become less predictable during this time, which is unsurprising. Dwell durations are affected by the number of passengers boarding and alighting at a station, and the models' input does not explicitly include passenger numbers. Without restrictions on travel, features such as the time of day and day of the week can implicitly model the number of passengers.

Figure 6.6 plots the MAE by batch for restricted movements through HT_0316. The models for passenger trains produce higher MAEs than for ECS apart from batch 12, where the ECS scores an extremely high value. The validation data of ECS for batch 12 consists only of nine records, and the high MAE is caused by one record with an unusually high target value being predicted poorly. The higher evaluation score for

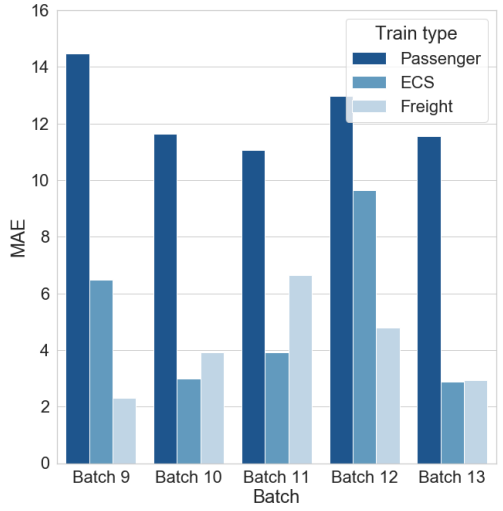

FIGURE 6.5: The MAE by batch and train type for unrestricted movements through HT_0316

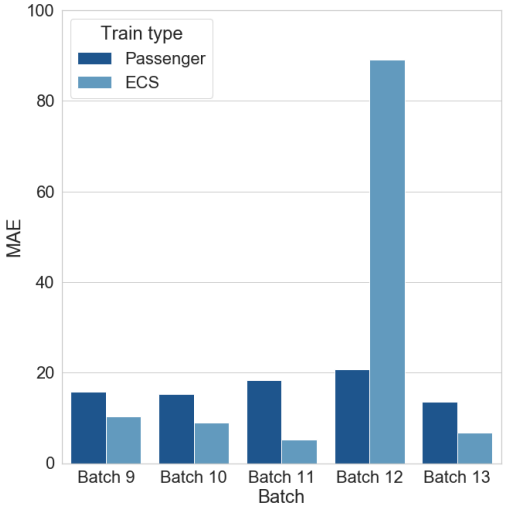

FIGURE 6.6: The MAE by batch and train type for restricted movements through HT_0316

passenger trains, in general, may be because trains usually dwell in HT_0316. A train may not have finished its dwelling activities when the conflict was cleared, leading to more variation in the target variable.

6.6 Final evaluation

6.6.1 Overview

Only the machine learning model with the hyperparameters that produced the lowest MAE in the selection phase is built in this phase for each combination of berth, movement and train type. Batches one to thirteen act as training data and the fourteenth batch are the test data, providing the final evaluation of the model – it is the only batch not used during the process so far. Eleven models are built for each combination of berth, movement and train type, one of the following quantile levels of τ:

$$0.05, 0.15, 0.25, 0.35, 0.45, 0.5, 0.55, 0.65, 0.75, 0.85, 0.95$$

The benchmark models are also created for all values of τ.

6.6.2 Results for $\tau = 0.5$

The quality of the predictions using the models with $\tau = 0.5$ is considered first. For unrestricted movements of passenger trains and ECS, the models for all berths outperformed the benchmarks. There was one berth (HT_0400) where the unrestricted movements for freight trains did not outperform the best benchmark. For those models that outperformed their benchmarks, the percentage improvement varied from 4% to 67%. The MAEs ranged from 0.79 to 12.61. Table 6.7 shows the results for the five example berths used to illustrate the data in the previous chapter. Although the MAE values are not strictly comparable between berths (as the berths have different lengths), it is noted that for passenger trains, HT_0029 and HT_0316 have higher MAEs than the other berths. Berth HT_0033 provides a low MAE for all train types; this is a straightforward berth with only one route in and out and no passenger stations.

For the restricted movements, there were six berths for passenger trains that did not outperform the benchmarks and four for ECS. Those models that did not outperform the benchmarks had fewer than 1000 records across all fourteen batches, although plenty of models outperformed their benchmarks with a similar number of records. The MAEs for restricted movements ranged from 5.34 to 160.76 for those machine learning models that did outperform their benchmarks. The MAE of 160.76 was an outlier and was achieved for passenger trains in berth HT_0050, where the total number of records was only 256. The next highest MAE was 27.79.

Figure 6.7 shows a scatter plot of MAE against the number of training records for all the segments of data that were modelled. The same data are also plotted having taken the logarithm base 10 of both values. The logarithmic plot shows the relationship between

the number of records: segments with a lower number of records tend to have higher MAE values than those with a large number of records. The results discussed thus far suggest that the minimum number of records for machine learning should be increased to 1000 records.

Table 6.8 shows the results for restricted movements for three of the five sample berths. Machine learning models were not built for restricted movements through HT_0029 and HT_0033 for either train type and HT_0037 for ECS.

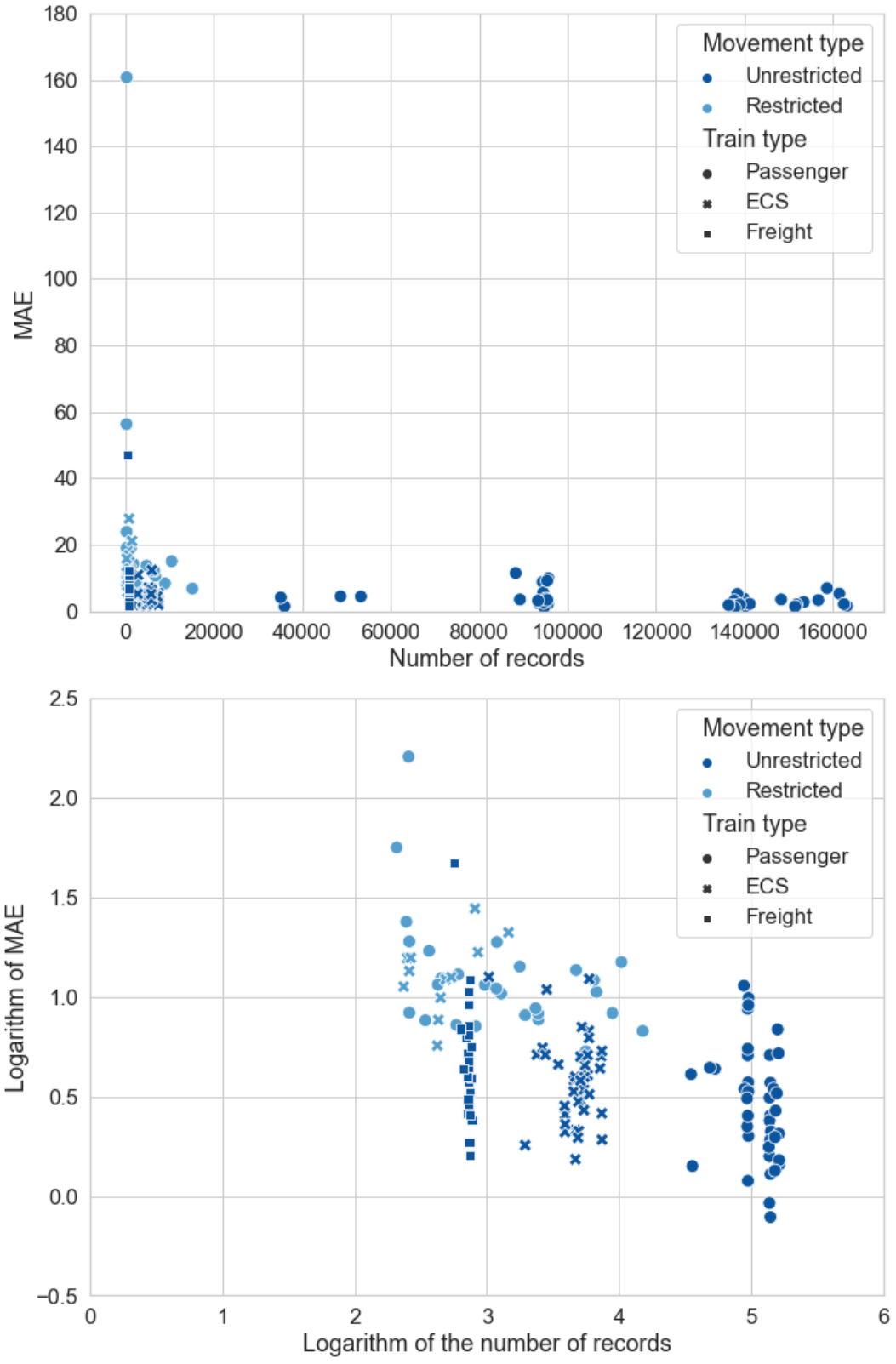

FIGURE 6.7: The MAE plotted against the number of training records for each segment where $\tau = 0.5$

		HT_0029	HT_0033	HT_0037	HT_0318	HT_0316
Passenger trains	**Number of records in test data**	10,040	10,058	9,993	6,639	6,573
	Benchmark MAE	5.73	2.03	3.69	4.05	18.52
	Machine learning MAE	3.71	1.29	1.91	3.45	11.40
	% improvement on benchmark	35.14%	36.31%	48.23%	14.69%	38.48%
ECS	**Number of records in test data**	415	418	417	520	523
	Benchmark MAE	6.98	3.97	4.47	6.62	5.69
	Machine learning MAE	3.34	2.14	2.12	5.02	3.80
	% improvement on benchmark	52.16%	46.02%	53.60%	24.16%	33.18%
Freight trains	**Number of records in test data**	61	65	57	63	59
	Benchmark MAE	5.10	3.37	12.26	7.53	5.98
	Machine learning MAE	4.10	2.40	7.16	4.37	3.06
	% improvement on benchmark	19.61%	28.77%	41.63%	41.89%	48.81%

TABLE 6.7: Final evaluation of five example berths for unrestricted movements

		HT_0037	HT_0318	HT_0316
Passenger trains	**Number of records in test data**	199	428	492
	Benchmark MAE	7.67	5.94	16.12
	Machine learning MAE	7.70	5.34	12.15
	% improvement on benchmark	-0.33%	10.22%	24.64%
ECS	**Number of records in test data**	-	15	19
	Benchmark MAE	-	13.32	11.27
	Machine learning MAE	-	12.58	7.67
	% improvement on benchmark	-	5.53%	31.95%

TABLE 6.8: Final evaluation of three example berths for restricted movements

	Value of τ									
Train type	**0.05**	**0.15**	**0.25**	**0.35**	**0.45**	**0.55**	**0.65**	**0.75**	**0.85**	**0.95**
Passenger	True	True	True	True	True	True	True	True	True	False
ECS	True	True	True	True	True	True	False	False	False	False
Freight	False	True	True	True	True	True	True	True	False	False

TABLE 6.9: The values of EQD by train type and τ for unrestricted movements through HT_0318

6.6.3 Results for other τ values

The values of τ that are not equal to 0.5 are not evaluated using the MAE but by the percentage of true values that are strictly less than or greater than their predicted values. This section also uses berths HT_0318 and HT_0316 to illustrate the results. The values of the EQD metric for HT_0318 are shown in Table 6.9. Figure 6.8 visualises the metrics by showing the percentage of records where the true target values lie below, equal, and above their predicted values for all values of τ for unrestricted movements through HT_0318. For each model, the percentage of target values above and below their predicted values are within the expected range if the value of τ sits within the percentage of records equal to the predicted value.

For the models of unrestricted passenger trains through HT_0318, the EQD is false only where $\tau = 0.95$. Figure 6.8 shows that approximately 6% of the target values exceed their predicted value instead of the expected maximum of 5%. This percentage is still reasonably close, and the binary nature of the metric does not capture any such nuance. There are more values of τ where the metric is false for ECS and freight; however, Figure 6.8 shows that the margins are close for many of those models. There are only 59 records in the test data for unrestricted freight movements, which may not be a large enough sample size for calculating the EQD.

The MQE was calculated for all the models and compared to the same metric for the benchmark. For all unrestricted movements of all three train types through berth HT_0318, the machine learning models for all values of τ were less than their equivalent benchmarks. This result shows that the machine learning models for HT_0318 can outperform their benchmark counterparts with respect to the quantile loss function.

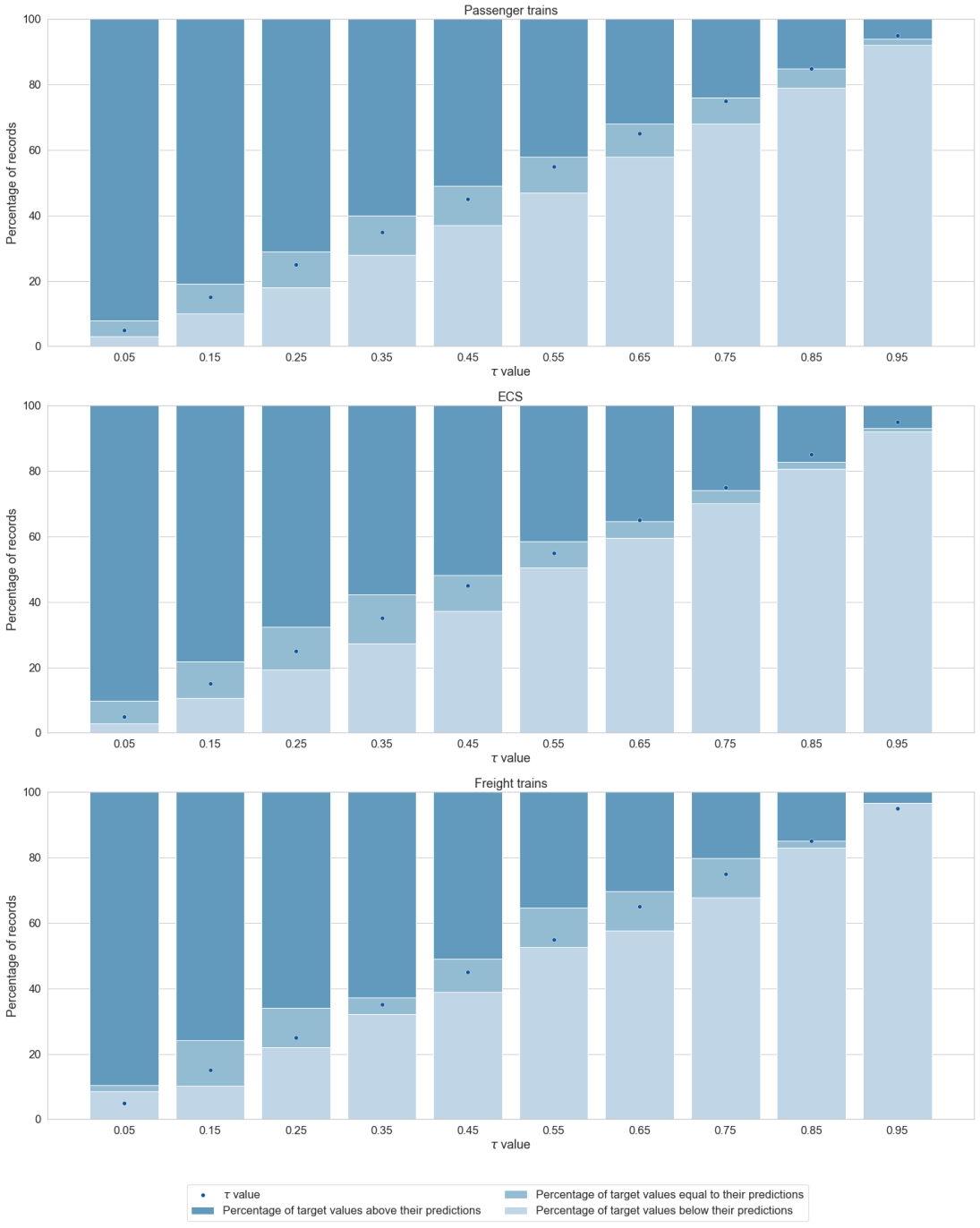

FIGURE 6.8: A visualisation of EQD by train type and τ for unrestricted movements through HT_0318

	Value of τ									
Train type	**0.05**	**0.15**	**0.25**	**0.35**	**0.45**	**0.55**	**0.65**	**0.75**	**0.85**	**0.95**
Passenger	False	False	False	False	True	False	True	True	True	False
ECS	False	False	False	False	False	False	False	False	False	False

TABLE 6.10: The values of EQD by train type and τ for restricted movements through HT_0318

	Value of τ									
Train type	**0.05**	**0.15**	**0.25**	**0.35**	**0.45**	**0.55**	**0.65**	**0.75**	**0.85**	**0.95**
Passenger	True	True	True	False	False	True	False	False	False	False
ECS	True	True	True	True	True	True	True	True	True	True
Freight	True	True	True	True	True	True	True	True	True	False

TABLE 6.11: The values of EQD by train type and τ for unrestricted movements through HT_0316

Table 6.10 shows the EQD for restricted movements through HT_0318 by quantile. However, there are only 19 records in the test data for restricted ECS movements, and the EQD is not a suitable metric with such a small sample. The MQE for the machine learning models for four values of τ were less than the benchmark for machine learning models.

Table 6.11 shows the EQD by train type and τ for unrestricted movements through HT_0316, with Figure 6.9 illustrating the metric. There are six false EQD values for passenger trains through HT_0316 compared to one for HT_0318. As with the results in the previous phase of work, this is also likely to be caused by the more significant variation in dwelling durations. The visualisation of the EQD metric in Figure 6.9 shows that the data divisions are not too far from the expected values. The MQE for all τ values, train types, and movement types were lower for the machine learning models than their corresponding benchmarks.

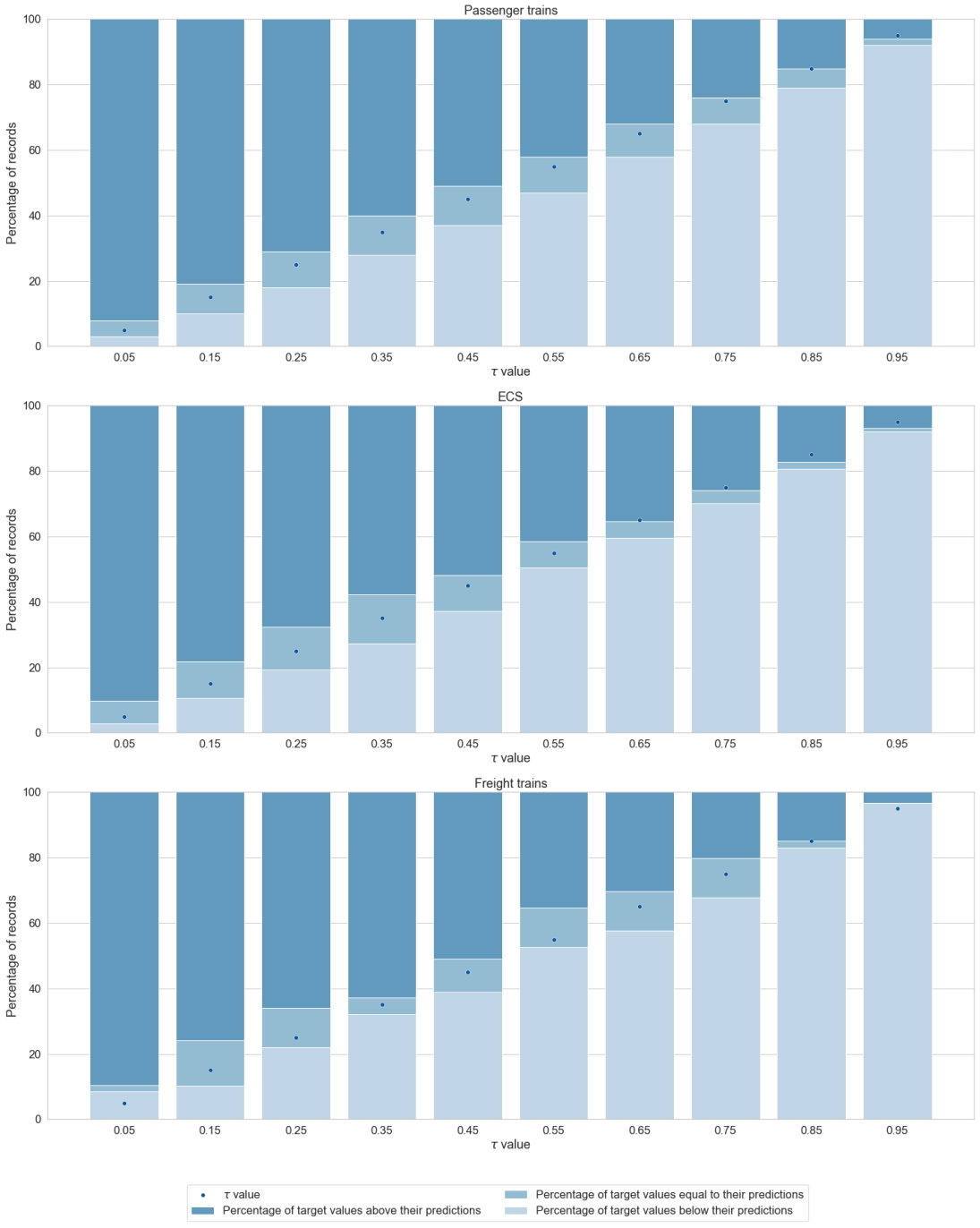

FIGURE 6.9: A visualisation of EQD by train type and τ for unrestricted movements through HT_0316

6.6.4 Point predictions versus random quantile predictions

Figure 6.10 shows histograms of predicted travel duration through berth HT_0318 for all unrestricted movements of passenger trains for batch 14. Five machine learning models were used to generate the histogram data: $\tau = 0.05, 0.25, 0.5, 0.75$ and 0.95. All five histograms in Figure 6.10 exhibit a bimodal distribution but are shifted along the x-axis with the histogram for $\tau = 0.05$ furthest to the left and $\tau = 0.95$ furthest to the right.

A histogram of the actual travel durations for batch 14 is shown in Figure 6.11, and none of the distributions shown in Figure 6.10 appears to be a particularly close replica of the true distribution. By contrast, Figure 6.12 shows a histogram where a prediction was made using a random value of τ for each record. This was achieved by generating a pseudo-random number for each record and assigning it to a value of τ based on its value. For example, if the pseudo-random number was less than 0.1, then the record was assigned $\tau = 0.05$; otherwise, if the pseudo-random number was less than 0.2, then the record was assigned $\tau = 0.15$, and so on. The value of $\tau = 0.5$ was not used for this generation of these predictions. Visually, the histogram in Figure 6.12 appears significantly closer to the distribution of actual values shown in Figure 6.11.

Figure 6.13 shows the cumulative distribution of the actual travel durations using the test data set. The cumulative distributions for $\tau = 0.05$, 0.5 and 0.95 are also shown along with the distribution for the predictions made with random τ values. Plotting the cumulative distributions is another way of visualising the similarity of the distributions. Figure 6.13 shows that the distribution created using the random τ values follows the actual distribution much closer than the distributions of individual τ values.

The Komogorov-Smirnov (KS) statistic is the greatest difference between two cumulative distributions. Figure 6.14 shows a bar plot of the KS statistic for all values of τ compared to the actual distribution for unrestricted movements through HT_0318 using the test data. The KS statistic for the distribution created by selecting random τ values is also displayed. When the random τ values are applied, the KS statistic is significantly smaller than for any individual τ value.

The MAEs for the models are also shown in Figure 6.14. This shows that the MAE was not the lowest when the random τ values were applied, which is unsurprising as the τ values were selected randomly for each record. The KS statistic and MAE together demonstrate that selecting a random value of τ for each record can produce superior predictions at the population level but is not guaranteed to do so for individual records.

The KS statistic and MAE shown in Figure 6.14 are likely to vary if a different set of random numbers are generated. There may be a selection of random numbers that could produce predictions that result in a lower MAE than for all the individual τ values. These results will be significant when the machine learning models determine the travel durations in the simulation model.

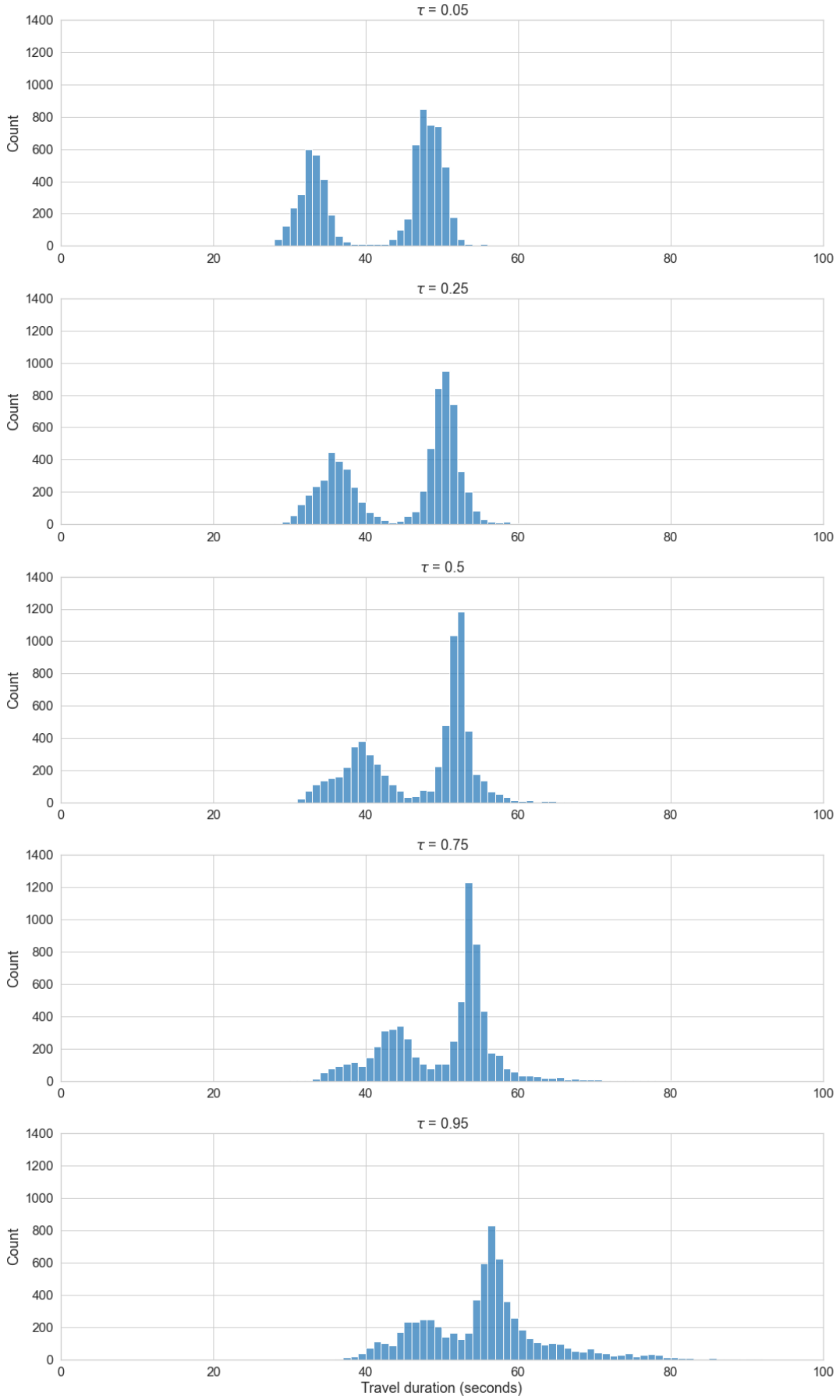

FIGURE 6.10: Histograms showing the predictions of unrestricted travel durations through HT_0318 using the models for $\tau = 0.05, 0.25, 0.5, 0.75$ and 0.95

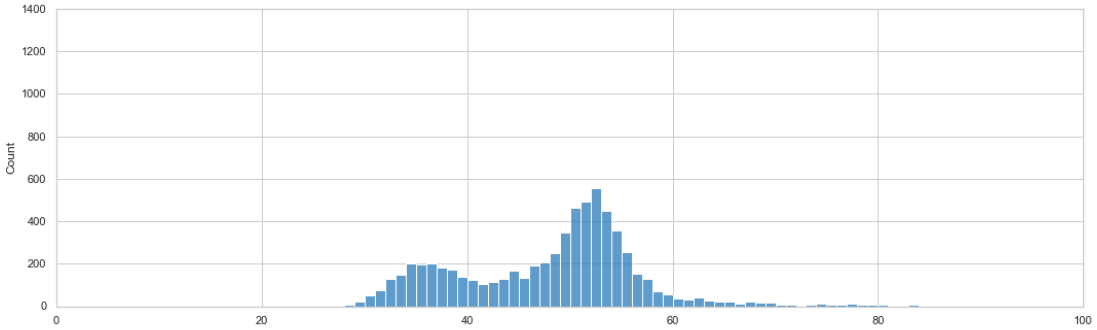

FIGURE 6.11: Histogram of the true values of the target variable of unrestricted travel durations through HT_0318

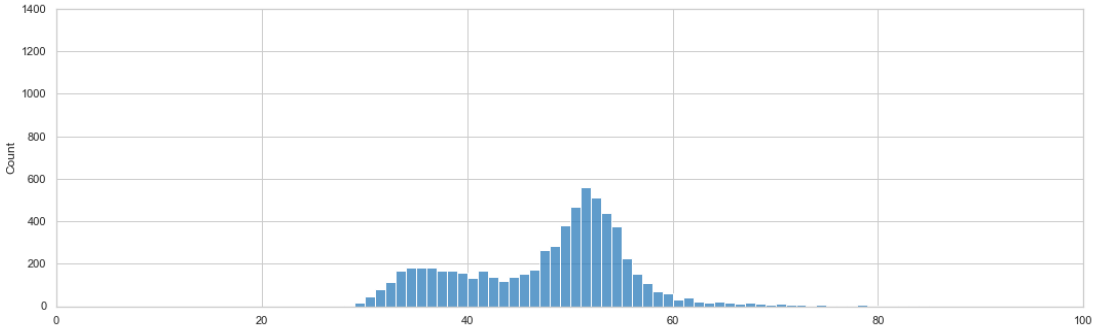

FIGURE 6.12: Histogram of the predicted values of the target variable of unrestricted travel durations through HT_0318 using a random value of τ for each record

This result is not unique to modelling unrestricted movements through berth HT_0318. Using random values of τ to make predictions for movements through other berths is capable of recreating the distribution of travel durations. However, the effect may be less noticeable when the MAE for the $\tau = 0.5$ model is very low or when there are a small number of samples in the test data.

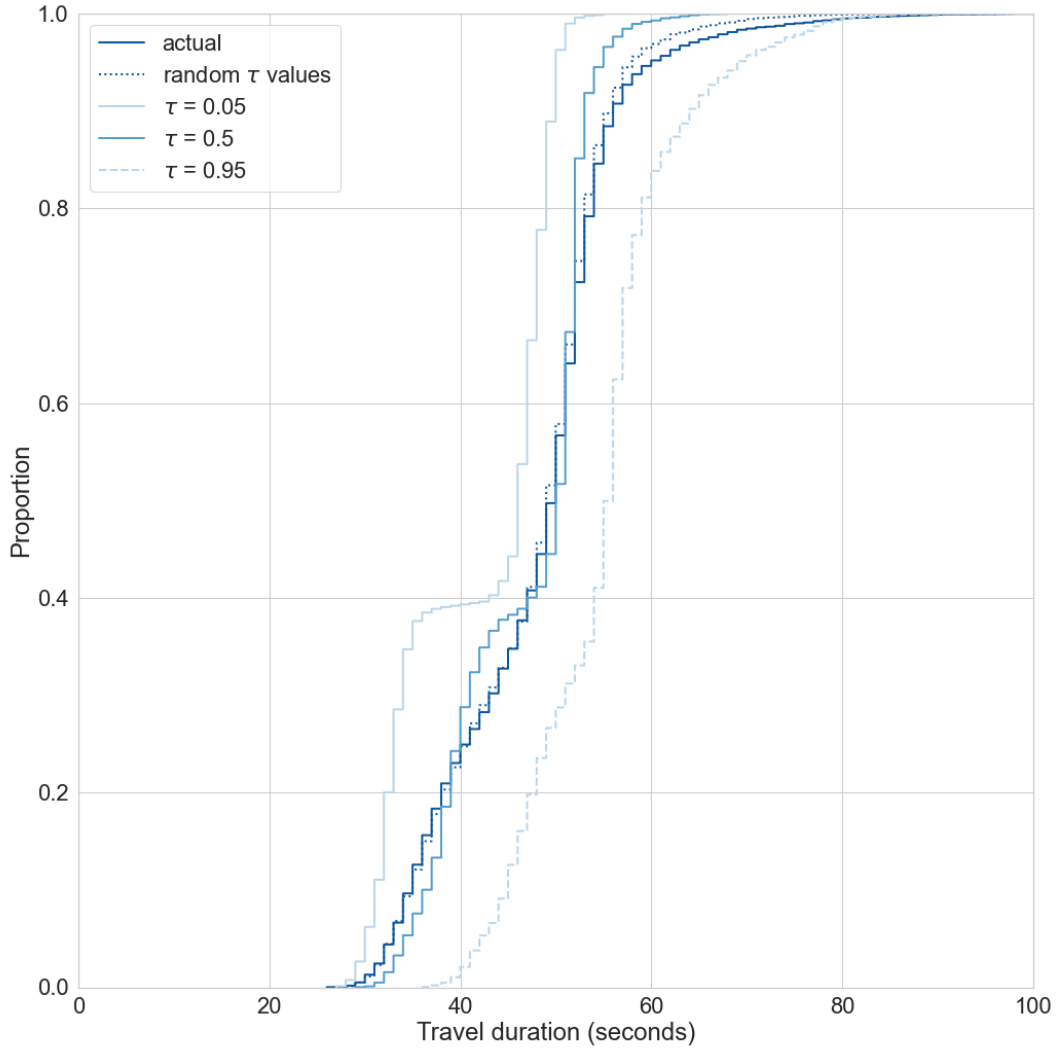

FIGURE 6.13: Cumulative distribution functions of the actual unrestricted travel durations through HT_0318 and the travel durations generated using $\tau = 0.05$, 0.5 and 0.95, and random τ values

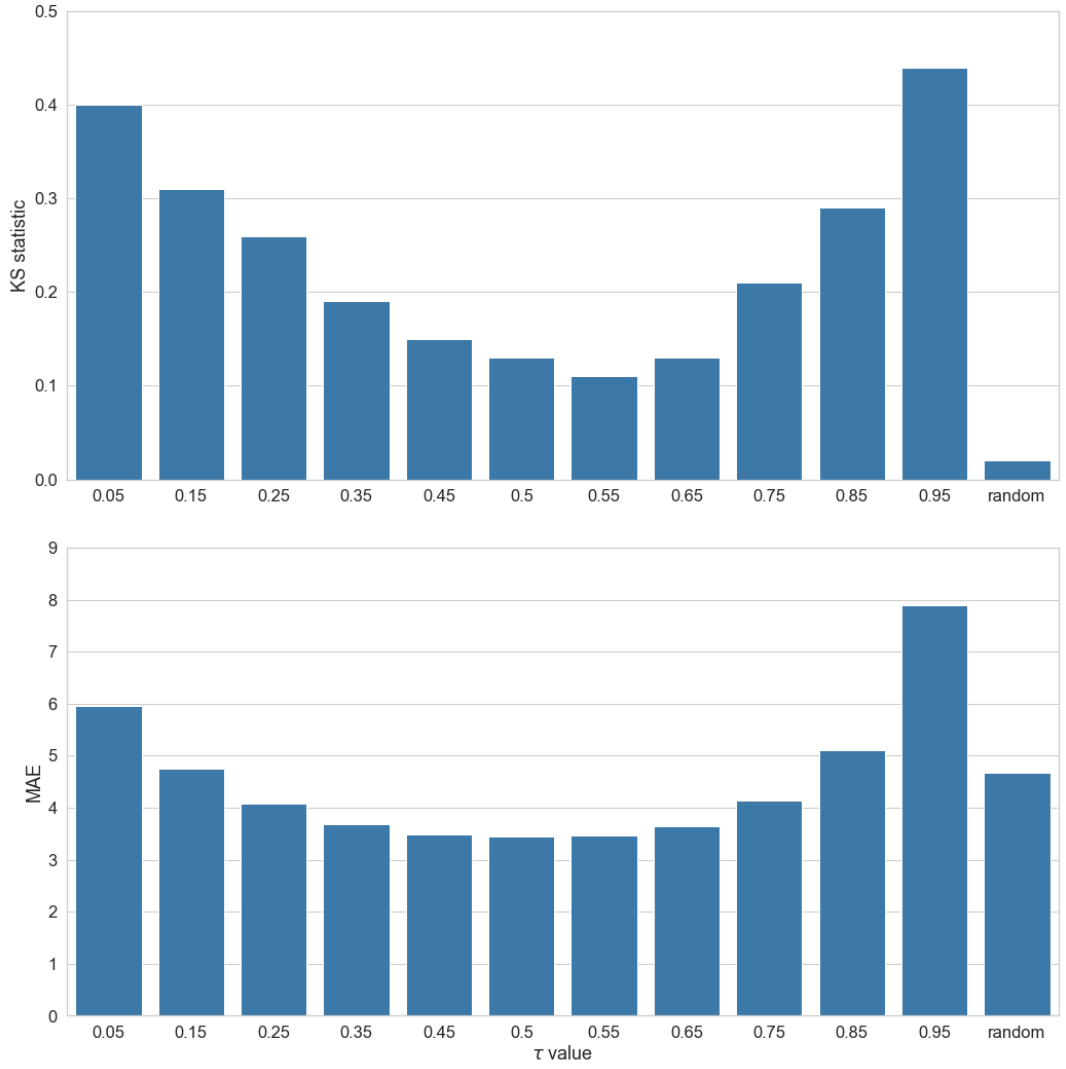

FIGURE 6.14: Bar plots showing the Komogorov-Smirnov statistic and the MAE for each value of τ and the random τ values for unrestricted movements through HT_0318

6.7 Discussion

6.7.1 Contributions to the literature

The work presented in this chapter has investigated the use of machine learning to make predictions about train travel durations using track-based data. Kecman and Goverde (2015) also used track-based data to model travel durations of unrestricted movements through berths using machine learning techniques. However, this research has built on that of Kecman and Goverde (2015) in several ways, which will be considered in turn.

Benchmark models

Kecman and Goverde (2015) did not compare their machine learning models to non-machine learning benchmarks. This work has demonstrated that not all machine learning models will outperform a simple statistical benchmark model, highlighting the importance of developing a benchmark to assess the quality of machine learning models.

Quantity of data

Kecman and Goverde (2015) used three months' worth of track-based data in their work. This work has utilised over three years of data. The results in Sections 6.4 and 6.5, where machine learning models did not outperform their benchmarks, occurred for segments of the dataset where there were smaller quantities of data available, such as for freight trains or restricted movements.

The amount of data used in this research also allowed the quality of the predictions over time to be examined. The results in Section 6.5 showed that external conditions, such as changes during the Covid-19 pandemic, could impact the quality of predictions for some scenarios.

Inclusion of freight and ECS

This work included modelling the travel durations of freight trains and ECS, which were excluded from the work of Kecman and Goverde (2015). Freight trains and ECS appear to be subject to more variation in travel durations than passenger trains, and, as a result, the MAE for these train types was often higher than passenger trains for berths without stations. However, the values were not orders of magnitude higher and might be improved with more data.

Modelling restricted movements

Making predictions for restricted movements has not been identified in the literature, and Kecman and Goverde (2015) only included 'conflict-free' records in their dataset. There are fewer records for restricted movements than unrestricted movements, meaning such movements were not modelled for many berths. The berths and train type combinations with greater quantities of data outperformed the benchmarks, indicating that the movements are predictable given sufficient information. Many options for improving these results are discussed in the following section.

Comparing ensembles and ANNs

Kecman and Goverde (2015) developed machine learning models using linear regression, decision trees and random forests. Other work, such as that of Barbour et al. (2018), Wen et al. (2020) and Huang et al. (2020), compared random forests with ANNs to predict arrival times and delays. Barbour et al. (2018) found that random forests outperformed ANNs, whereas Wen et al. (2020) and Huang et al. (2020) developed ANNs that outperformed the random forests.

Approaches using neural networks are popular in the literature, with neurons such as recurrent and LSTM proving popular with sequential and time-series predictions. The results presented in Section 6.4 demonstrated that ANNs could sometimes outperform gradient tree boosting models, but the margin of improvement was relatively small. Gradient tree boosting models could produce comparable results with less computational effort.

Quantile regression

This chapter's main contribution is applying quantile regression to predict train travel durations. No examples of quantile regression in the railway domain were found in the literature. This work demonstrated that quantile regression can predict the distribution of the target variable. The ability to predict a distribution is desirable when the full range of the target variable is required for modelling, an application of which will be demonstrated in the following chapter.

There are also potential applications of quantile regression to the railway domain outside the scope of this thesis. For example, O'Sullivan et al. (2016) and Parslov et al. (2021) used the approach to estimate prediction intervals for bus arrival times and travel durations, which could be applied to train travel durations. Prediction intervals could be constructed for the travel duration of a train to some TIPLOC on its route. These prediction intervals could be integrated into a decision-support tool for traffic management.

6.7.2 Further work

The following chapter will extend this work by demonstrating the application of the machine learning models in a simulation. Aside from applying the models in a simulation, there are many areas of study that could follow from the results in this chapter, just as this research has built on the work of Kecman and Goverde (2015). Some possible areas of further work are considered in turn.

Inclusion of S-Class data

The train describer S-Class data was not included in the dataset for this work. As a result, the status of signals had to be inferred from the location of trains. The inclusion of S-Class data would identify whether a signal is at its most restrictive aspect or not. The benefits of this would be twofold. Firstly, the data segmentation into unrestricted and restricted movements would improve. Currently, there are likely to be records labelled as unrestricted movements where the train has not been granted movement authority out of a berth. However, it is impossible to identify all red aspects with the current data.

Secondly, for restricted movements, the S-Class data would provide the time at which movement authority was granted. The target variable could become the time between the movement authority being granted and the train stepping into the next berth. Currently, the target variable is the time from the conflict being cleared to the train moving into the next berth, but the movement authority will not be granted at the exact time the conflict is cleared.

Inclusion of other data sources

While this work has only employed open data sources, other data may improve the models' predictive power. Examples include freight train features excluded from the open data for security reasons, such as the type of freight being transported. Other features, such as the number of coaches on a train or the length of the berths, may also improve the results.

Use of train-based data

The work could also be approached using train-based data. Such data would allow dwelling durations to be predicted separately.

Metrics for quantile regression

Some improvements could be made to the metrics developed in this work to assess the quantile regression models. For example, the EQD did not assess how 'close' the model is to dividing the data by the expected quantities.

Hyperparameter tuning for quantile regression

In this work, the hyperparameters selected for $\tau = 0.5$ were also used to construct models using other values of τ. There is no guarantee that the selected hyperparameters will be appropriate for other τ values. Further work would be beneficial to investigate whether the models could be improved by tuning the hyperparameters.

ANNs for quantile regression

One disadvantage of quantile regression using gradient tree boosting is that a separate model is required for each value of τ. If ANNs were used instead, it is feasible to construct a single network with multiple outputs – one for each value of τ. A prototype of such a network was created as part of this work. However, ANNs were computationally more expensive than ensemble methods and building one ANN with eleven outputs may not be faster than building eleven gradient-boosting models.

6.8 Conclusion

This chapter has presented the results of machine learning models developed to predict the travel duration of trains in the demonstration area. The results showed that benchmark models were necessary to assess the performance of machine learning models. Machine learning models could fail to outperform their benchmarks if there was a small amount of training data or inappropriate hyperparameters were selected.

The work presented in the chapter built on work reviewed in the literature by modelling ECS and freight trains as well as passenger trains and by considering restricted train movements. The results showed that the quality of the predictions varied by berth, movement and train type, with characteristics such as the location of passenger stations affecting the quality of the predictions. A greater quantity of data was used compared to other studies, allowing the predictive quality of the models to be examined over time.

Quantile regression was applied to predict various quantiles of the target variable, ranging from 0.05 to 0.95. The work demonstrated that selecting a random quantile to

predict each record in the test data made it possible to produce a distribution of the target variable that was very close to the true distribution.

The next chapter will provide more detail on the simulation model that has been developed and will deploy these machine learning models into the simulation to determine the movement of trains around the simulation. In particular, the next chapter will sample train travel durations from the range of quantile regression models in a stochastic simulation. These experiments will demonstrate the benefit of modelling the distribution of the target variable.

Chapter 7

Simulating train movements

7.1 Introduction

The previous chapter presented the results of machine learning models built to predict train travel durations through berths. The work established how the distribution of the travel durations could be predicted by sampling from quantile regression models. The ability to model the distribution of travel durations is of importance in this chapter, which presents a machine learning-assisted simulation. This chapter will demonstrate how the machine learning models developed in the previous chapter can be exploited within a simulation to model stochastic train movements.

The experiments presented in this chapter simulate a week's worth of activity in the demonstration area. A week was selected from the date range of the test data set from the previous chapter, and that week's schedules were used as input to the simulation. As the actual movements of the trains during the week are known, the simulation's output can be compared to reality. The traffic management decisions are simplified to replicate the actions taken in the historical data; this allows a more direct comparison between the travel durations in the historical data and the simulation model's output.

The simulation model was introduced in Chapter 4, where some high-level principles of the model were discussed. This chapter provides more detail about the design of the simulation model in Section 7.2. Section 7.3 describes some of the verification activities that have been carried out to ensure the fundamental features of the model work as intended. Section 7.4 documents how the simulation model has been configured for the experiments.

The simulation was run in deterministic and stochastic modes, and Section 7.5 presents both results; these activities also constitute the validation activities of the simulation. Section 7.6 summarises the contributions of the work presented in this chapter and discusses some opportunities for further research.

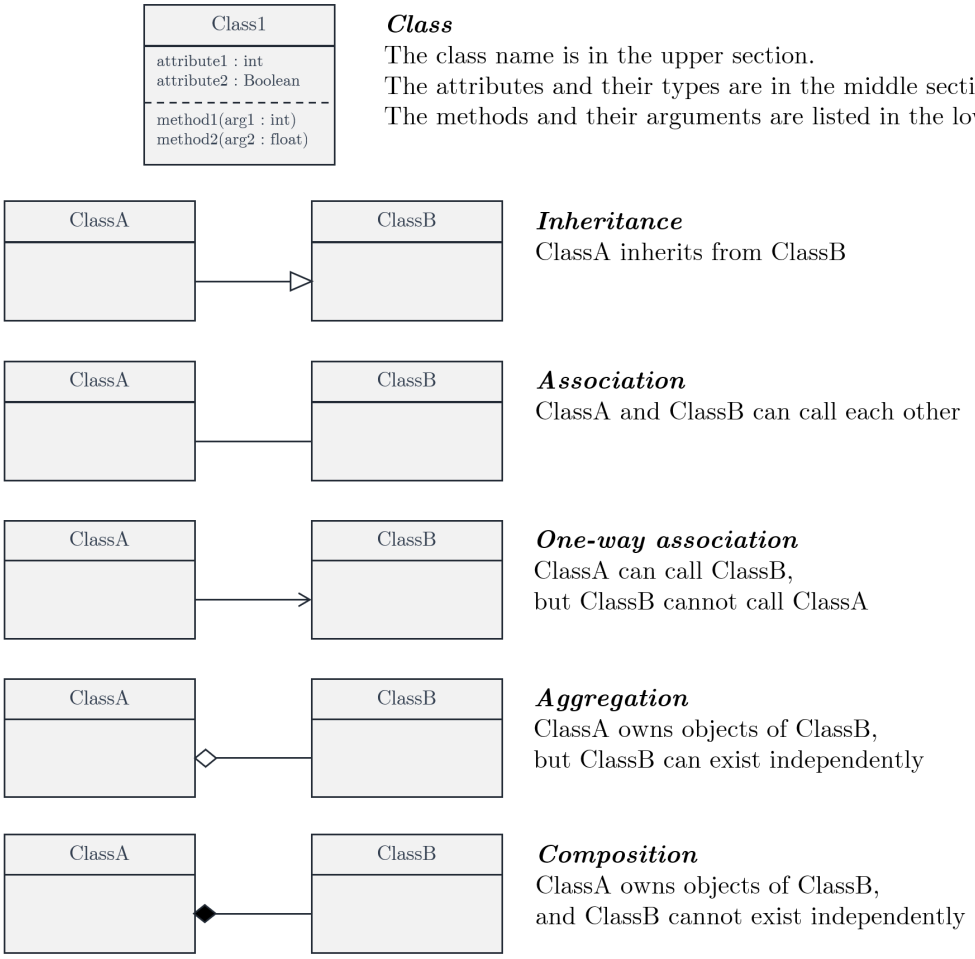

FIGURE 7.1: A key to the UML figures used in this chapter

7.2 Simulation model design

7.2.1 Overview

This section describes the high-level design of the simulation model. The design is visualised in a Unified Modelling Language (UML) conceptual class design[1]. Figure 7.1 provides the key to the notation used in the UML, and the reader is referred to introductory texts, such as Fowler (2004), should further information on UML be required.

The simulation results, presented later in this chapter, are communicated without the need for detailed knowledge of the development of the simulation. Consequently, the UML diagrams are not comprehensive and show only the main features of each class. For example, complete lists of attributes and methods are not provided. Similarly, the accompanying descriptions of the logic do not cover all eventualities.

[1] https://www.uml.org/

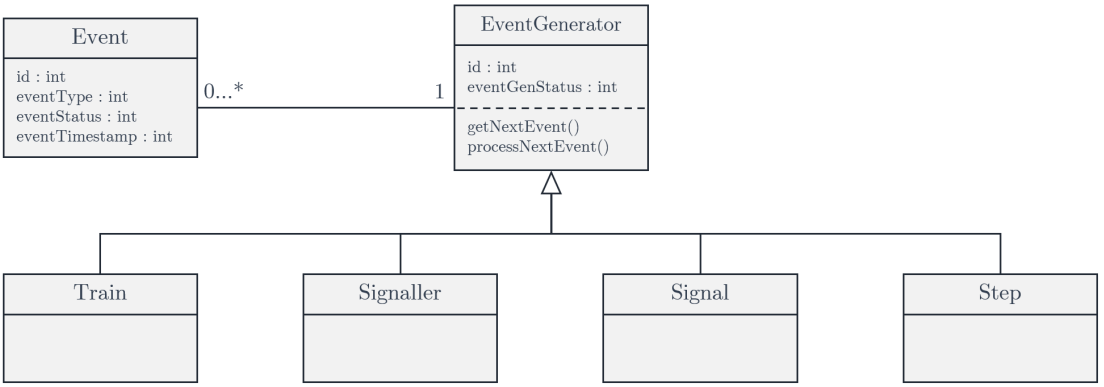

FIGURE 7.2: A UML representation of the classes involved with the generation and processing of events

As the simulation is event-driven, Section 7.2.2 begins by documenting the event-related classes. The remaining detail of the simulation is divided into four categories covering the infrastructure, rolling stock, travel duration and traffic management.

7.2.2 Events

The `Event` class stores details of events which will take place in the simulation. Two attributes of the `Event` class are the `eventType` and `eventTimestamp`. The `eventType` dictates the kind of event that will occur, such as 'train moves into next berth' or 'signaller sets route'. The `eventTimestamp` specifies the time the event will take place.

The `EventGenerator` class stores a list of zero or many `Event` objects and has methods to get and process the next event, where the 'next' event is the one with the earliest timestamp. Figure 7.2 shows the relationship between the two event classes and the four classes which inherit from `EventGenerator`: `Train`, `Signaller`, `Signal`, and `Step` – these will all be described in more detail in subsequent sections. Each of the classes that inherit from `EventGenerator` will overwrite the method which processes the next event with appropriate logic.

After the input has been loaded, the simulation is initiated with the starting timestamp (given as input) and runs a recursive loop. Each iteration of the loop will get all the `EventGenerator` objects in the simulation and the timestamp of their next event (if one exists). The event with the earliest timestamp will be processed, and the simulated time will be updated to the event timestamp. The process repeats until the end timestamp has been reached or there are no more events to process.

7.2.3 Infrastructure

Figure 7.3 displays the class model of infrastructure in the simulation. The `Network` to be modelled will be composed of one or more `Areas`. Each `Area` is, in turn, composed of `Berths` and is associated with precisely one `Signaller`. A step is a directed movement between two berths, and each `Berth` object is associated with one or more `Steps`, zero or one `Signal` and zero or more `TIPLOCs`. In the demonstration area, all berths have exactly one signal. The `Area` class stores the rules about which steps are mutually exclusive.

The `Berth` class has a Boolean attribute called `isBoundary`. Boundary berths are an artificial concept in the model; whereas ordinary berths can be occupied by at most one train, a boundary berth can be occupied by many trains. Boundary berths will hold trains that have yet to enter or have already left the simulated area.

The `Signal` class has an attribute called `aspect`; a value of 0 represents a red aspect, and a value of 1 all other aspects. This implementation is sufficient for the machine learning models to determine the travel durations, as they infer the signal aspect based on the location of trains. While it would be feasible to model three- or four-aspect signals, such an implementation would not change the predictions made by the machine learning models. The `Signal` class also has methods to change its aspect.

The `Step` class contains an attribute called `isConflict`, which is `True` or `False`, depending on whether the step is part of at least one conflict. Its value does not change throughout the simulation. Another Boolean attribute that does not change its value `isLocked`; this attribute is set to `True` if the step is the only step leading in and out of two berths. If a step has the value `isLocked` set to `True`, it is permanently set, and the signaller does not have to set and release it. The time it takes to set and release a step are also recorded as attributes.

The `Signaller` class is responsible for setting steps for a train through the method `createSetRouteEvents`. The details of how this method identifies and resolves conflicts will be covered in a later chapter; for the work in this chapter, the signaller resolves conflicts using the same priorities as the historical data.

Steps that are not locked are released after a train passes over them, and constraints exist such that only one step into and out of a berth may be set at any time. Signals will change to their red aspect any time there is no route set out of a berth or if there is a train occupying the next berth.

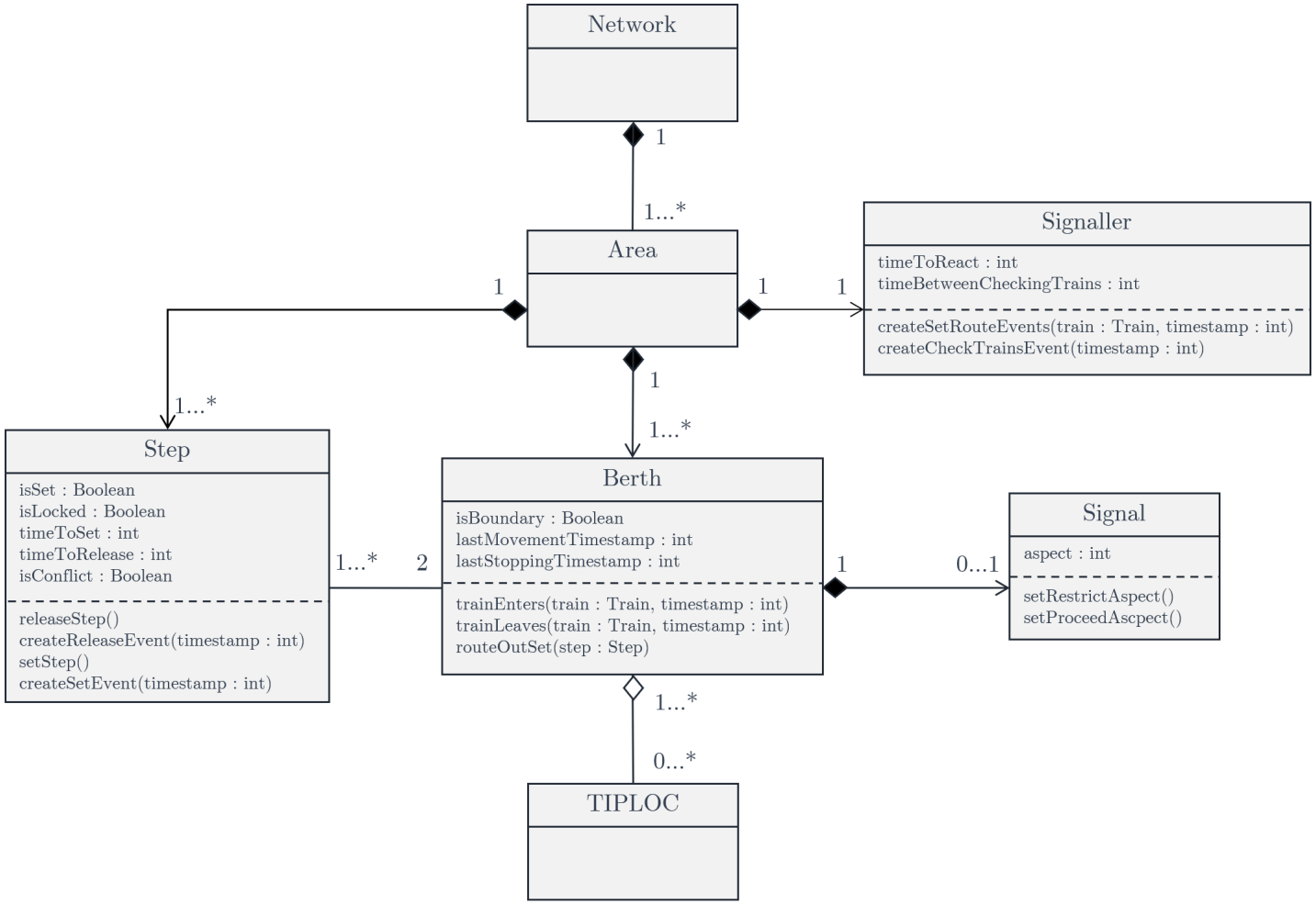

FIGURE 7.3: A UML representation of the classes concerning the railway infrastructure

7.2.4 Rolling stock

Figure 7.4 displays the model of the trains and schedules in the simulation. The `Train` class is associated with two or three `Berths` as it will record the previous, current and successive berths on its journey. At the beginning of its journey, a train will only hold the current and successive berths. Conversely, a train at the end of its journey will only hold the previous and current berths. Trains in the simulated area will hold all three types of berth.

The `Berth` class only stores the trains currently occupying it, which may be none. Berths may be occupied by multiple trains if services join or separate (which will not occur in the scenarios modelled in this chapter) or if the berth is labelled as a boundary.

The `Train` class consists of attributes that do not change between schedules, such as the power type or train operating company. Each train is composed of one or more schedules. Each `Schedule` is associated with precisely one train - this means that schedules that repeat daily or weekly are currently created multiple times in the model. The `Schedule` class contains attributes such as the train type and whether it is an express service. Note that the train type is not necessarily consistent across all schedules: if a train has two schedules, the first may be an ECS and the second a passenger train.

Two classes inherit from the main `Schedule` class. The first is `FreeRouteSchedule` which is made up of `ScheduleItems`. The schedule items are ordered, each containing a TIPLOC and the scheduled time the train is due to arrive, depart or pass. They store very similar data as Network Rail's timetables. The second type of schedule class is `FixedRouteSchedule`, made up of `RouteItem` objects. The route items are ordered, and there is one item for each berth the train will travel through. When using the `FixedRouteSchedule` class, the train will follow the route laid out by the `RouteItems` objects. In contrast, if the `FreeRouteSchedule` class is used, each train's route will be determined during the simulation. Only the `FixedRouteSchedule` is employed in this thesis.

The `RouteItem` class contains attributes such as the berths the train moves between, the next scheduled TIPLOC and the scheduled timestamp. The actual travel duration and movement timestamp are also recorded as attributes but will only be used if it is necessary to mimic the historical movements of the trains.

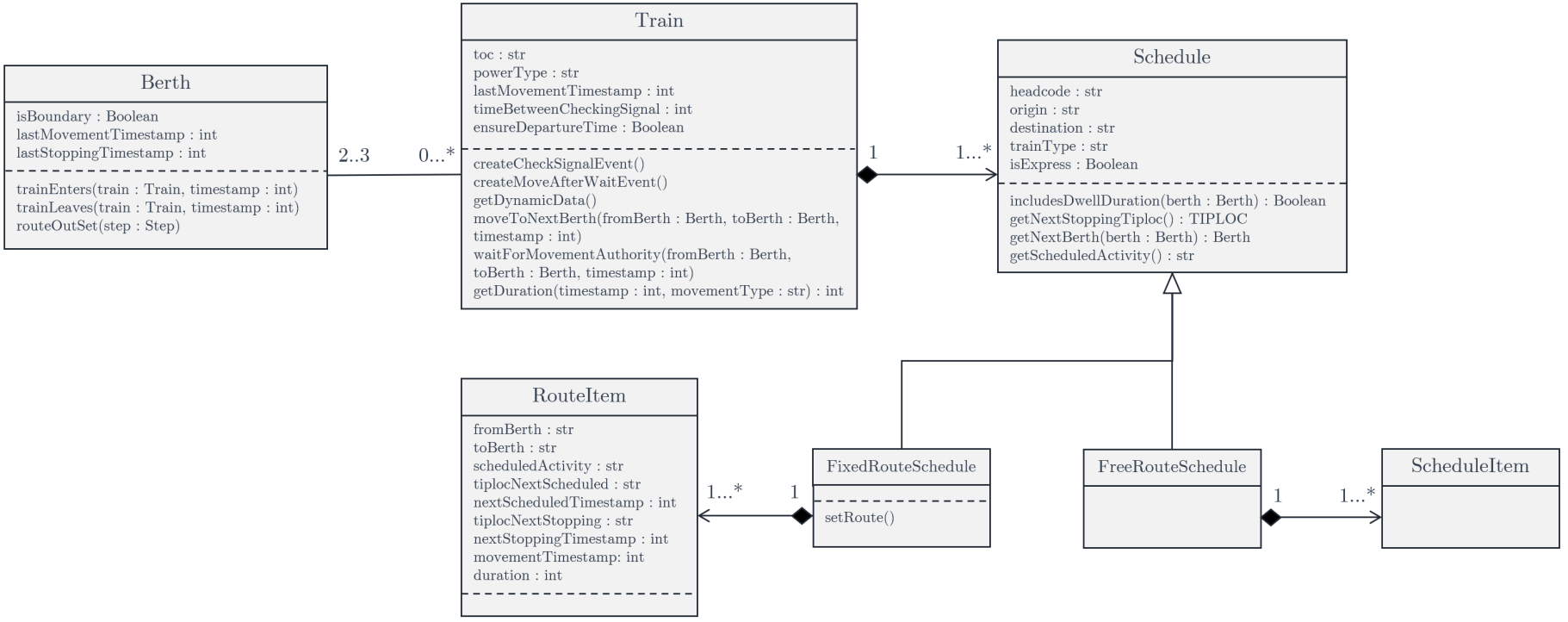

FIGURE 7.4: A UML representation of the classes concerning rolling stock and schedules

7.2.5 Travel durations

The `ModelDuration` class determines the unrestricted and restricted travel durations. The `Network` class is associated with exactly two instances of `ModelDuration`, one for each movement type, see Figure 7.5. Holding the models for travel duration at the network level means they only have to be loaded once, reducing the amount of memory required compared to if they were loaded for each Train object.

Each time a `Train` object moves into a `Berth`, it calls its `getDuration` method, which returns the predicted travel duration from the 'unrestricted' `ModelDuration` object via the `Network`. Similarly, if a train is ready to move out of a berth after being held at a red signal, a predicted duration is returned from the 'restricted' `ModelDuration` object.

The child classes of `ModelDuration` override the `initialiseModels` and `predictDuration` methods with their own logic. Two child classes of `ModelDuration` are used in this chapter. The first `ModelDurationExact` uses known historic travel durations for each movement. This feature is used when calibrating the model's parameters and can only be run in deterministic mode.

The second child class to be used is `ModelDurationML`. If the simulation is deterministic, then the instances of `ModelDurationML` load the machine learning models for every berth and train type combination for $\tau = 0.5$. If the simulation is stochastic, then the models for all other values of τ are loaded. When a duration is calculated in stochastic simulations, a pseudo-random number is generated to determine which τ value shall be used. As well as loading the machine learning models, the `ModelDurationML` class loads other constructs necessary to transform data before making a prediction – all the essential items were saved during the final phase of the machine learning development.

The `ModelDurationML` class also loads all the benchmark models. There are some combinations of berth, movement and train type for which a machine learning model was not built, such as restricted movements for freight trains. For such situations, a benchmark model is used instead. If a benchmark model does not exist due to a lack of data, then a default model is loaded, which is the first benchmark model for the train types across all berths. Although the default models may not be very sophisticated, they are required infrequently. There is nothing to stop a machine learning model from predicting a negative value. Although this is an unlikely scenario, if a negative value is predicted, the model will revert to using the benchmark model instead.

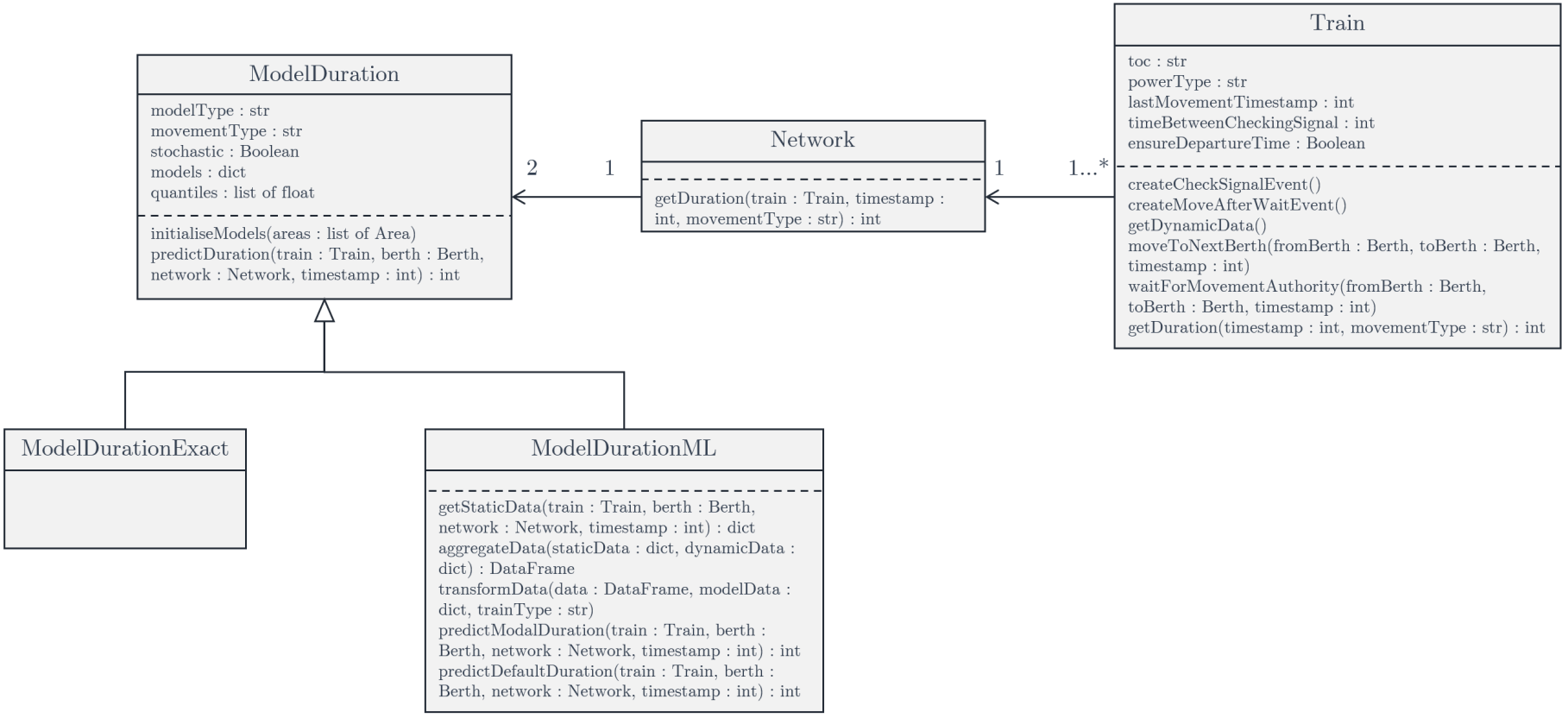

Figure 7.5: A UML representation of the classes concerning the prediction of travel durations

Entry times to the simulation are currently modelled deterministically; they are the same for each iteration of the simulation and are provided as part of the input for each train's timetables. Direct delays are also modelled deterministically and, therefore, occur for the same trains in the same berths for each iteration. The locations of direct delays are also provided as part of the route items loaded for each train's timetable. Extending the simulation to allow for stochastic entry times and direct delays are discussed later in the chapter as possible opportunities for further work.

7.2.6 Traffic management

The `Signaller` class checks the network for potential conflicts and takes action to resolve the conflicts by setting routes. For the experiments in this chapter, the simulation will run using a week's worth of timetable data that has already occurred, and the signaller will always give priority to trains according to the historical data. Detail on how this is achieved and information about the classes and available traffic management models are presented in Chapter 9.

7.3 Verification

The simulation model can output all the events in an iteration, which allows trains to be traced throughout the simulation. The output can verify fundamental aspects of the simulation, such as the number of trains that entered the simulated area equals the number that exited plus the number in the simulated area at the end.

The ability to determine travel durations from input data using the `ModelDurationExact` class was exploited to verify the implementation of the conflict resolution. Input scenarios were constructed which would guarantee conflicts between trains. These scenarios and the detailed output confirmed that multiple trains would not occupy a non-boundary berth and that two trains would not pass over conflicting steps simultaneously.

To predict travel durations using the machine learning models, the simulation model has to aggregate and transform the data for a train using precisely the same process as the training data were manipulated before the machine learning models were built. The simulation was configured to output all the transformed data for an iteration. The output matched transformations of the same data from the machine learning processes. These tests verified that the data are being transformed correctly in the simulation.

The simulation was run over multiple stochastic iterations, setting the initial random seed the same for each iteration. The output of all iterations was the same, as expected. Secondly, when each iteration was initialised with a different random seed, the results were different but varied within reasonable bounds.

7.4 Simulation set-up

7.4.1 Infrastructure inputs

The simulation is set up with four areas of input – shown in Figure 7.6. The demonstration area is labelled `HT`, and then there are three boundary areas labelled `B1`, `B2` and `B3`. The berths in these areas are set to be boundary berths and can, therefore, hold multiple trains before and after they leave the main simulated area. The network data is loaded in text files of berths, steps and conflicts.

7.4.2 Train and timetable inputs

A week's worth of timetable data was selected for the experiments. The data were taken from the dataset created for this work of train movements (described in Section 5.5), so they only contained timetables that could be matched to a movement record. The selected week was from 3 a.m. on 2021-08-09 to 3 a.m. on 2021-08-16. The date range was within the test data set (batch 14) used for machine learning. It was chosen as there appeared to be good-quality data in the selected range, with a high proportion of timetable data matching the TD train movement data. The start time of 3 a.m. was selected as there were no trains in the area then, meaning the simulated area could start empty.

Six trains were missing a few steps through the demonstration area, and these were added manually. Two trains missing more significant amounts of movement data were excluded from the input.

It is also necessary to load data for each train for the five steps before entering the demonstration area, as the machine learning models use the travel durations through the five previous berths. There were some services for which these data were not available. Those services were still modelled, but the predictions were made using the benchmark models until the trains had moved five steps into the demonstration area.

Of the trains in the input data, 14 were freight trains, 131 started as ECS, and the remainder (2,442) were passenger trains. Six trains changed schedules in the area during the simulated week from ECS to a passenger schedule. Overall, 2,587 schedules were included in the input data.

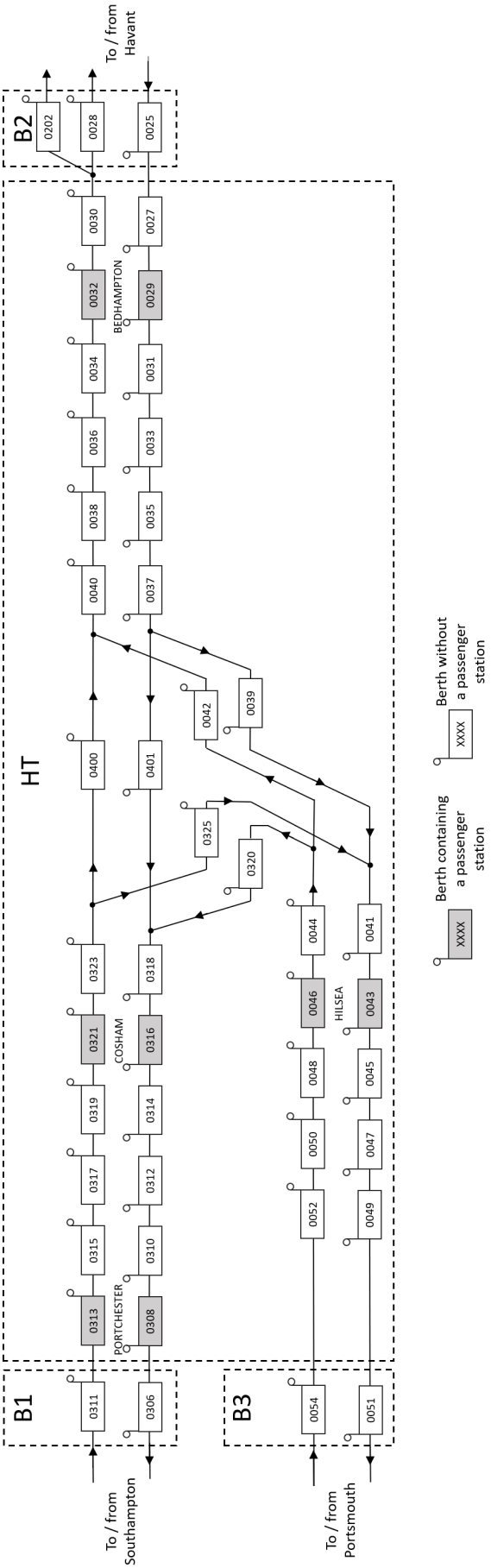

FIGURE 7.6: A representation of the demonstration area in the simulation model

7.4.3 Parameter calibration

The simulation has many parameters that configure the logic of algorithms. Three parameters are used in this section to illustrate how they were calibrated:

- Parameter 1: The time interval it takes for a route to be set after a signaller has selected it[2].
- Parameter 2: How frequently a signaller checks the area to see if any trains are being held at a red signal.
- Parameter 3: Whether a train can leave a berth before its scheduled departure time (a value of `True` or `False`)

The process for determining appropriate values for the parameters used the week of historical timetable data. The process required the simulation to run in deterministic mode using the `ModelDurationExact` class. The first stage required the simulation to mimic the travel durations of the historical data exactly, and the example parameters were given the following values:

- Parameter 1: 0 seconds (routes are set instantaneously)
- Parameter 2: 1 second (the signaller checks the area at the smallest possible interval)
- Parameter 3: `True` (trains will leave a berth after they have reached the duration in the input data, regardless of scheduled departure times)

The total travel duration of all trains through the simulation area was 1,470,631 seconds (408.51 hours) which was confirmed to be the same as the total travel durations in the historical data.

Although the simulation can mimic the historical data using the initial parameter values, some parameters, such as Parameters 1 and 3, are not set to realistic values. Other parameters, such as Parameter 2, can significantly impact the time it takes the simulation to run – a low value means more events are generated, which increases the run time compared to a high value.

The subsequent stages of the process aim to find values for the parameters which do not affect the output of the simulation. While the `ModelDurationExact` class will determine the travel durations from the input data, the parameter values can also impact the travel

[2]Currently this is the same parameter for all routes; an improvement would be to allow this to be varied by route

durations. For example, if Parameter 1 is set to a high value, it may result in a train waiting while a route is set, extending its total run duration.

The process identified that the values of Parameters 1 and 2 could be set to 20 seconds without affecting the total travel duration of the output. This also decreased the time it took to execute the simulation.

However, when Parameter 3 was set to `False`, the total travel time of all trains increased to 1,472,541 seconds, roughly an extra 32 minutes, compared to the actual total duration. This parameter only affects those train services with scheduled stops in the demonstration area. When the parameter is set to `False`, the earliest a train can leave a berth after stopping is the scheduled departure time.

As the total travel duration increased when Parameter 3 is set to `False`, this implies that there are trains in the data which left berths ahead of the scheduled departure time. An analysis of the input data found 176 instances of trains departing a berth ahead of their scheduled time. All but three of the trains departed the berth less than a minute ahead of their scheduled departure time. Many cases occurred in berths HT_0316 (Cosham) and HT_0032 (Bedhampton), where the boundary of the berth appears close to the station. These instances probably reflect trains departing on time; the time source the drivers use when departing a station may not be precisely synchronised with the time source used by the train describer system.

However, three trains departed a station around three minutes ahead of their scheduled time. It was not possible to find an explanation for these, and the trains will continue to be modelled in the simulation. The simulation will be run with Parameter 3 set to `False` so that trains cannot leave the berth ahead of their scheduled departure time.

7.5 Simulation results

7.5.1 Deterministic simulation

The simulation was run in deterministic mode using the machine learning models where $\tau = 0.5$ to determine the travel durations through the berths. The total travel duration of all trains through the simulation was 1,403,056 seconds or 389.74 hours, which is around 19 hours shorter than the actual total travel duration. The difference is 4.59% (to two decimal places) of the true value. This result is unsurprising given that the models are predicting the median and that the distribution of travel durations is known to be skewed to the right. The machine learning models are less likely to predict the higher values of travel durations.

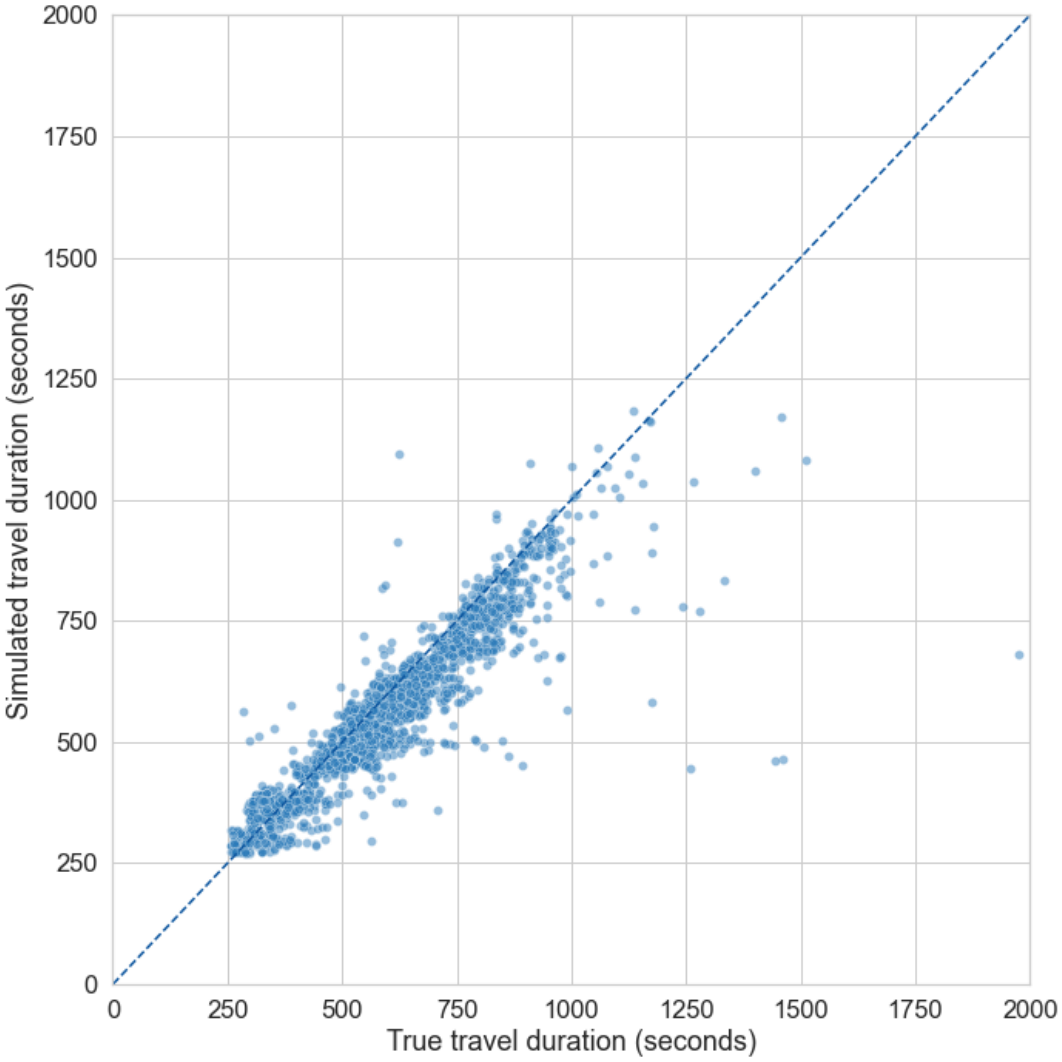

FIGURE 7.7: A scatter plot showing the true travel durations on the x-axis and the simulated travel durations on the y-axis

Figure 7.7 shows a scatter plot of the true travel durations of trains on the x-axis and the predicted travel duration on the y-axis. The dashed line represents the equilibrium between the prediction and the actual values. Most of the points in Figure 7.7 are clustered near the equilibrium line. However, there are more points below the line than above, showing that more travel durations are underestimated rather than overestimated.

7.5.2 Stochastic simulation

The simulation was then run in stochastic mode, with each travel duration sampled from the models using τ values from 0.05 to 0.95. Each time a travel duration is required, a pseudo-random number is generated, which determines a value of τ to use. The model for $\tau = 0.5$ was not used in this simulation.

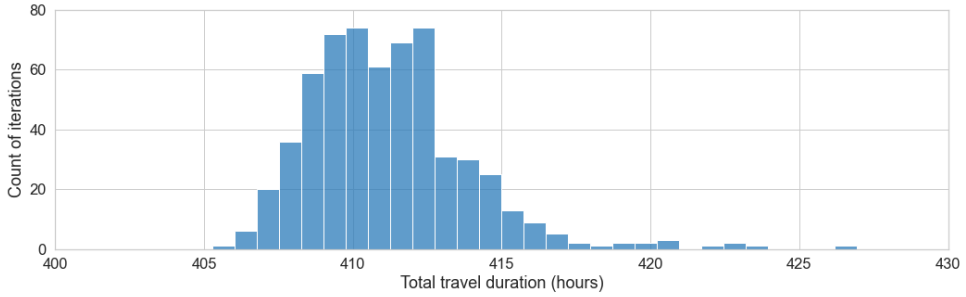

FIGURE 7.8: Histogram showing the total travel duration across all iterations of the stochastic simulation

Multiple iterations of the simulation are necessary, as one iteration of a stochastic simulation will not provide meaningful results. The rolling median total travel duration was used to determine a minimum number of iterations. For each iteration, the absolute difference between the rolling median value of the current and previous iterations was calculated. The minimum number of iterations was deemed to be reached when the absolute difference settled to less than 0.01% of the latest median value. This occurred after 162 iterations and remained below 0.01% for the remaining iterations. A total of 600 iterations were run overall. A histogram of the total travel durations achieved over all the iterations of the stochastic simulation is presented in Figure 7.8, showing that they are also skewed to the right.

Figure 7.9 shows the value of the rolling median over the 600 iterations of the stochastic simulation, which settled to 1,479,002 seconds or 410.83 hours. Additionally, the figure plots the minimum and maximum values achieved over all the iterations, the true total travel duration from the historical data and the value obtained from the deterministic simulation. The plot shows that the actual total travel duration falls between the minimum and maximum values achieved by the simulation.

The median total travel duration the stochastic simulation achieves is roughly two and a half hours more than the true value. The actual total travel duration is just one realisation of how the events of the simulated week could have unfolded; therefore, there is no expectation that the median total travel duration of the stochastic simulation should converge to the actual value. However, the difference between the median and true values is 0.57% of the true value, which demonstrates that the stochastic simulation can model more realistic travel durations than the deterministic simulation achieved.

Indeed, Figure 7.9 shows that the total travel duration achieved with the deterministic simulation is outside the bounds of the stochastic simulation. This demonstrates the importance of modelling the skewed nature of travel durations. Without the ability to model the longer travel durations that occur, the total travel duration will be underestimated.

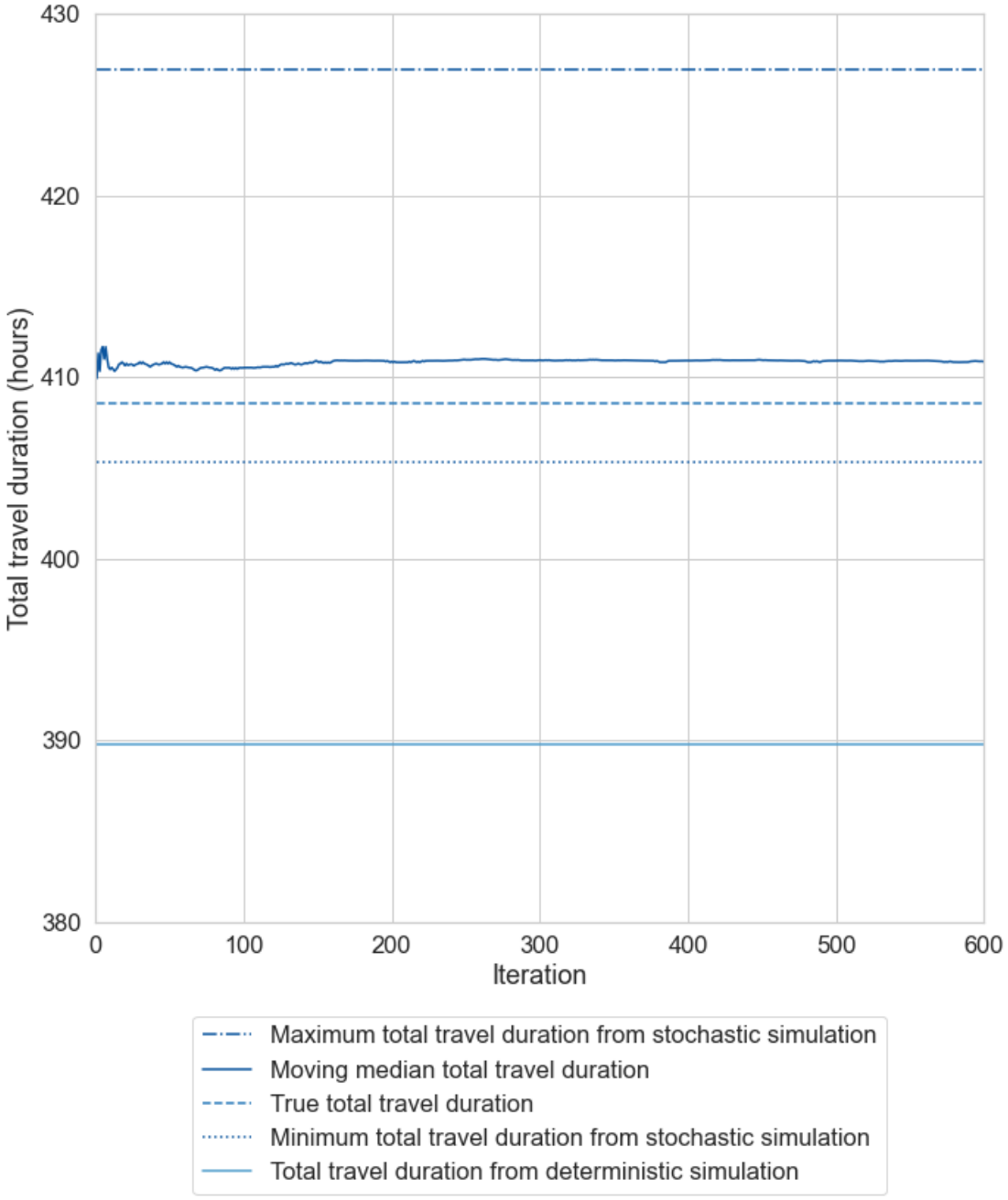

FIGURE 7.9: Plot of the median total travel duration by iteration

7.6 Discussion

7.6.1 Contributions to the literature

Reviews of established railway simulations identified that they could not fully represent the stochastic nature of trains' run and dwell durations (Watson and Medeossi, 2014; Medeossi and de Fabris, 2018). The simulation presented in this chapter introduces a novel approach to modelling the travel durations of trains using machine learning models. No prior examples of machine learning-assisted simulation modelling have been identified in the railway domain nor of the combination of quantile regression and simulation modelling in any area.

The previous chapter demonstrated that the quantile regression models could be sampled to create a realistic distribution of travel durations, and this chapter utilised those models within the simulation. The stochastic simulation produced variation in the total travel duration with each iteration. The true total travel duration of the modelled week was within the minimum and maximum bounds of the simulated values. The rolling median total travel duration across the iterations also converged to within 0.57% of the true value. Conversely, modelling the travel durations deterministically underestimated the total duration by 19 hours.

This approach could be particularly beneficial for modelling new or updated timetables as it could model realistic variations in train movements so that the performance of a timetable might be assessed. The stochastic results in this chapter showed that the distribution of total travel durations was found to skew to the right, allowing an infrastructure manager to understand a timetable's full range of performance.

7.6.2 Limitations of the approach

Despite the favourable results presented in this chapter, the approach has some limitations. It is likely prohibitive to adapt any existing simulation to use machine learning to model the movement of trains as they all determine train movements using motion equations. If this approach is applied in industry, a new simulation would need to be developed, or the one presented in this work would need to be extended.

The simulation developed in this research benefits from and may not be feasible without some high-performance computing (HPC) facility. The machine learning models are currently loaded at the start of the simulation. However, as the modelled area increases, so does the memory requirement to store the machine learning models. An alternative approach using less memory would be to load the machine learning models as and when required, although this approach is more computationally time-consuming. Making predictions with machine learning models within a simulation also adds to the

computational effort of such a model. The relevant data have to be aggregated and transformed for every prediction made.

Machine learning-assisted simulation modelling will not be appropriate for all use cases. For example, it is unlikely to be a suitable approach if new infrastructure is to be considered. The machine learning models are built using historical data and, in their current form, will not generalise to unseen situations, such as a new track layout. Incorporating features of the berths, such as the length or gradient, into the machine learning models may overcome this issue. Other scenarios that may not be appropriate include introducing new classes of trains, such as high-speed trains.

7.6.3 Further work

Chapter 9 will extend the work presented here by modelling stochastic traffic management decisions within the simulation. Further developments could be made to the functionality of the simulation model that were not feasible in the timescales of this work. For example, the entry times of trains to the simulation are modelled deterministically. Quantile regression could also be employed to provide a stochastic model of the entry times of the trains. Similarly, direct delays are modelled deterministically, and historical data could be analysed to determine the frequency and location of direct delays.

Improvements could also be made to the efficiency of the simulation. For example, the simulation currently checks the next event of all `EventGenerator` objects; this could be updated to consider only 'active' objects. When a train leaves the simulation, it could be marked as 'inactive', preventing it from being checked. Additionally, instead of having a signaller object check its area at set time intervals for trains being held at red signals, a train held at a red signal could alert the signaller, thus reducing the number of events required.

7.7 Conclusion

This chapter has introduced the first known example of machine learning-assisted simulation modelling in the railway domain and possibly the first application of quantile regression in stochastic models in any domain. The chapter presented the main features of the simulation with an emphasis on how the travel durations are applied.

A week's worth of activity was simulated in deterministic and stochastic modes. The deterministic simulation was achieved using the $\tau = 0.5$ models to determine train travel durations through berths. The total travel duration of all trains in the deterministic simulation was around 19 hours lower than the true total duration. The stochastic

simulation used the quantile regression models for other values of τ and sampled from them randomly to achieve variation in the run durations. The median total travel duration of the stochastic simulation settled over the iterations to within two hours of the actual total duration.

The stochastic simulation also demonstrated the range of total travel durations that might be realised. The true total travel duration fell within the bounds of the minimum and maximum values from the stochastic simulation. Modelling travel durations stochastically would be beneficial when simulating changes such as a new timetable or traffic management rules. The stochastic travel durations would demonstrate the full range of the impact of such changes compared to a deterministic model.

Modelling traffic management decisions will become the focus of the following two chapters. Chapter 8 will build machine learning models to predict traffic management decisions, and Chapter 9 will apply those machine learning models within the simulation model.

Chapter 8

Predicting signalling decisions

8.1 Introduction

The previous chapter introduced the machine learning-assisted simulation model and demonstrated its application by modelling stochastic travel durations. This chapter changes the focus to predicting traffic management decisions using machine learning. The six pairs of conflict locations in the demonstration area will be used to illustrate the work.

There are many aspects to traffic management on the railways, and this chapter only considers conflict resolution. Section 8.2 will discuss conflict resolution and the aim of the models. Section 8.3 discusses how the data have been prepared, the creation of benchmark models and the selection of machine learning algorithms. The section also introduces a method for calibrating the probabilistic output of the machine learning models, called isotonic regression.

Section 8.4 presents the results of the exploration modelling phase, comparing the machine learning results to the performance of the benchmarks. The selection phase results will be presented in Section 8.5, examining whether the evaluation metric evolves over time. The final evaluation is presented in Section 8.6, where the quality of the models is assessed on the test data. Section 8.7 concludes with a discussion of the outcomes of this work.

8.2 Requirements of the models

As with the machine learning models built in Chapter 6, these models must be compatible with the simulation model developed as part of this research. The research aims to evaluate machine learning-assisted simulation, so although the models developed in this chapter will be analysed and evaluated in their own right, the main aim is to apply them within the simulation model.

Given a pair of trains approaching conflicting locations, the models aim to predict which train will be prioritised. The conflicts are labelled 'A' and 'B' for each conflict pair. For each record in the dataset, the trains are also labelled 'A' and 'B', with train 'A' always travelling towards conflict 'A'.

An example of the problem formulation can be found in Figure 8.1. The figure shows train 'A' due to cross a conflicting step between HT_0323 and HT_0325 and train 'B' due to travel between HT_0401 and HT_0318. The problem is structured as a probabilistic classification problem with the output being between zero and one. A value greater than 0.5 will mean that in the deterministic case, the model predicts that the train travelling in the direction of train 'A' has priority; otherwise, train 'B' travels first over the conflict location. The probabilistic output will be used in the stochastic simulation model. For example, a predicted value of 0.75 will be interpreted as 75% of the time, train 'A' has priority. Note that these models aim to *predict* the actions of signallers; they are **not** seeking to find the optimal decision to take.

In the case of Figure 8.1, both trains 'A' and 'B' are three steps away from their conflicting steps. The dataset constructed for this work, described in Section 5.6, consists of pairs of trains between one and seven steps away from the conflict. However, the data used in this modelling work will only consider trains up to five steps away from the conflict location. In the following chapter, the simulation model will also only resolve conflicts up to five steps away from conflict locations. The limitation to five steps is due to the size of the demonstration area.

These models do not consider other features of traffic management, such as route selection. Also, the models only predict the resolution of conflicts between pairs of trains, conflict resolution between three or more trains is not considered.

As with the machine learning models from Chapter 6, it will also be necessary to be able to save the structure and hyperparameters of these models so that they can be loaded by the simulation model at a later date.

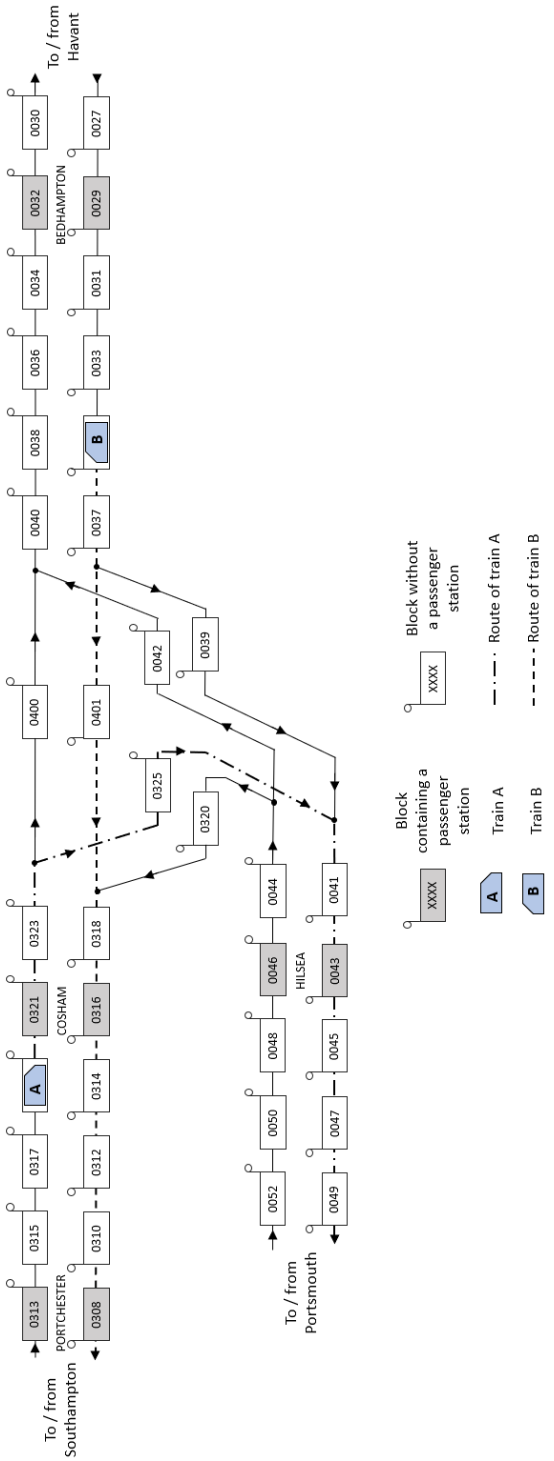

FIGURE 8.1: An illustration of a conflict between two trains: 'A' and 'B'

8.3 Preparation

8.3.1 Segmentation of dataset

The data are segmented by each pair of conflict locations, and separate predictive models will be built for each segment. Unlike the train movement data, the conflict data are not segmented further by train type, as conflicts exist between different types of trains. Each record in the dataset is allocated to a batch number depending on the date it occurred.

Table 8.1 lists the total number of records for each conflict pair; only those conflicts where both trains are within five steps of the location are included in the count. The table shows that conflicts occur more frequently at some junctions than others. Similar to the travel movement data, the number of records is not spread evenly over the fourteen batches. For example, batch 9 contains significantly fewer records than other batches as the date range for that batch coincides with the initial travel restrictions in the UK at the start of the Covid-19 pandemic.

Table 8.1 also lists the percentage of records where train 'A' is given priority in the data; this shows that the two classes to be predicted are not balanced. For example, train 'A' is prioritised in only 13.6% of records across all the batches for conflict pair numbered five. This inequality means that for this conflict pair, a model which constantly predicts that train 'B' is given priority would have an accuracy of 86.4%.

Conflict pair ID	Conflict A		Conflict B		Total number of records	% of records where train 'A' has priority
	From berth	To berth	From berth	To berth		
1	HT_0037	HT_0401	HT_0042	HT_0040	115,559	45.9%
2	HT_0323	HT_0325	HT_0401	HT_0318	84,390	25.4%
3	HT_0044	HT_0042	HT_0325	HT_0041	258,159	67.1%
4	HT_0042	HT_0040	HT_0400	HT_0040	109,670	37.8%
5	HT_0320	HT_0318	HT_0401	HT_0318	78,897	13.6%
6	HT_0325	HT_0041	HT_0039	HT_0041	166,776	30.3%

TABLE 8.1: The number of records for each conflict pair in the dataset

8.3.2 Modal benchmark models

Two different types of benchmark models are constructed in this work. This section describes the first type of model, referred to as 'modal' benchmark models. They are similar in concept to the benchmark models used in Chapter 6 for the travel duration models. They require the data to be grouped by different features, and then the modal (most common) value for the train that gets priority is taken as the prediction for each data group.

The following list describes how the data were grouped:

- **Modal 1:** The records are not grouped.
- **Modal 2:** The records are grouped by each train's current berth.
- **Modal 3:** The records are grouped by each train's current berth, destination, and type (passenger, freight or ECS).
- **Modal 4:** The records are grouped by each train's current berth, destination, type (passenger, freight or ECS) and where 'A' or 'B' is scheduled to go first.

These models are easy to interpret and can be extended to probabilistic output by considering the percentage of records where train 'A' has priority.

8.3.3 Programmatic benchmark models

The second type of benchmark model to be developed are 'programmatic' models. These models are constructed of conditional statements: logic which follows an 'if—then—else' structure. Like the modal models, these are simple to interpret but cannot be extended to produce probabilistic output. The following models were developed:

- **Programmatic 1:** The train scheduled to travel across the conflict first will have priority.
- **Programmatic 2:** Passenger trains have priority over ECS and freight. Otherwise, trains will be prioritised according to their scheduled order.
- **Programmatic 3:** A passenger train will have priority over a freight train or ECS, and express passenger trains will have priority over non-express trains if it is at least one signal closer to the conflict. Otherwise, the trains will be prioritised according to their scheduled order.

- **Programmatic 4:** A train that is stopping between its current location and the conflict location will give priority to a train that is not stopping only if the non-stopping train is closer to the conflict location. Otherwise, the trains will be prioritised according to their scheduled order.
- **Programmatic 5:** A train that is stopping between its current location and the conflict location will give priority to a train that is not stopping. A passenger train will have priority over a freight train or ECS, and express passenger trains will have priority over non-express trains, but only if it is at least one signal closer to the conflict. Otherwise, the trains will be prioritised according to their scheduled order.

8.3.4 Machine learning models

Machine learning models are built using three classification approaches: gradient tree boosting, random forests, and artificial neural networks (ANNs). For the gradient boosting and random forest algorithms, hyperparameters that were varied included the number of decision trees in each ensemble, the maximum depth of each tree, the maximum number of features that each tree uses to determine the best split, the minimum number of samples required in the node of a tree before performing the split and the learning rate. For the ANNs, only basic neurons were used. The hyperparameters that varied between the ANN models were the number of neurons in each hidden layer, the number of hidden layers and the learning rate.

Table 8.1 showed that the two classes to be predicted are not balanced, i.e. there are not 50% of each class in the dataset. Another hyperparameter was created for all algorithms to determine whether or not to apply weights to each training data record, which are inversely proportional to the frequency of each class. If the weights are applied, they impact each record's importance; for example, in tree-based methods, the weights will affect how the decision trees are split.

8.3.5 Calibration

The selected machine learning algorithms can produce probabilistic output, i.e. a value between zero and one. However, the output is not guaranteed to be well-calibrated; that is, the probabilistic predictions do not necessarily correspond to the true probabilities. Therefore, an additional calibration step will be carried out after the models have been built in the final evaluation phase of the work.

Two established methods of calibration were considered: Platt scaling (Platt, 2000) and isotonic regression (Zadrozny and Elkan, 2002). Initial experiments showed little difference between the results of both methods. Further, a review of both methods

concluded that Platt scaling was more effective when the dataset is small, but that isotonic regression is more powerful on larger datasets (Niculescu-Mizil and Caruana, 2005). As there are large amounts of data available for this work, isotonic regression was selected to demonstrate the potential for using calibration.

Isotonic regression adjusts the probabilistic output of a machine learning classification model, in doing so it creates a new model. If the original model that has been created is f, then isotonic regression aims to find a calibrated model, $\hat{f}$, which minimise the following:

$$\sum_{i=1}^{n} \left(y_i - \hat{f}(x_i) \right)^2$$

Where $\hat{f}(x_i) >= \hat{f}(x_j)$ whenever $f(x_i) >= f(x_j)$, x_i is the i_{th} record of the input data and y_i is the true value. In this case $f(x_i), \hat{f}(x_i) \in [0, 1]$ are the probabilistic outputs of the model. The resulting model $\hat{f}$ is a step-wise non-decreasing function, which will be demonstrated in Section 8.6.

8.3.6 Feature selection

The features used to build the machine learning models are listed in Table 8.2. The names of the two conflict locations are not included in the list of features as the data are segmented by these, and, for each pair, the conflict location 'A' is always listed first. Even though the number of steps to the conflict location is a numeric value, it has been treated as a categorical variable which will be an integer between one and five inclusive.

There are potentially many values for the origin, destination, and stopping TIPLOC features. As these are categorical variables, they will be transformed using one-hot encoding leading to a new column for each distinct value. A large number of data features are not always beneficial for machine learning, particularly if some represent infrequent categorical values. Therefore, only the ten most frequent values are encoded. If a record has, for example, an origin value that is not one of the ten most frequent, it will be assigned to an 'unknown' category.

Feature name	Data type	Description
Train A has priority	Boolean	True if train A has priority at the conflict; false otherwise. This is the target variable.
Train A is scheduled first	Boolean	True if train A is scheduled to pass the conflict location ahead of train B; false otherwise
Public holiday	Boolean	True if the day is a public holiday; false otherwise.

Table 8.2 continued from previous page

Feature name	Data type	Description
Weekend	Boolean	True if the day is a Saturday or Sunday; false otherwise.
Hour of the day	Numeric	The hour of the day in which train A moved into its berth
Month of the year	Numeric	The month of the year in which train A moved into its berth
Lockdown status	Categorical	A value representing the severity of lockdowns and travel advice during the Covid-19 pandemic
Timestamp A	Numeric	The timestamp train A moved into its current berth
Timestamp B	Numeric	The timestamp train B moved into its current berth
Berth A	Categorical	The berth train A is currently in
Berth B	Categorical	The berth train B is currently in
Train A stops before conflict	Boolean	True if train A is scheduled to stop between its current location and the conflict location; false otherwise
Train B stops before conflict	Boolean	True if train B is scheduled to stop between its current location and the conflict location; false otherwise
Stopping TIPLOC after conflict A	Categorical	The next TIPLOC that train A is scheduled to stop at after passing the conflict A
Stopping TIPLOC after conflict B	Categorical	The next TIPLOC that train B is scheduled to stop at after passing conflict B
Travel duration of train A through previous berth	Numeric	The travel duration of the train A through the previous berth it travelled through
Travel duration of train B through previous berth	Numeric	The travel duration of train B through the previous berth it travelled through
Train type A	Categorical	The type of train A: passenger, ECS or freight
Train type B	Categorical	The type of train B: passenger, ECS or freight
Number of steps train A	Categorical	The number of steps train A has before reaching the conflict location
Number of steps train B	Categorical	The number of steps train B has before reaching the conflict location
Origin A	Categorical	The origin TIPLOC of train A
Origin B	Categorical	The origin TIPLOC of train A

Table 8.2 continued from previous page

Feature name	Data type	Description
Destination A	Categorical	The destination TIPLOC of train A
Destination B	Categorical	The destination TIPLOC of train B
TOC A	Categorical	The code representing the TOC of train A
TOC B	Categorical	The code representing the TOC of train B
Electric A	Boolean	True if train A is electric powered; false otherwise
Electric B	Boolean	True if train B is electric powered; false otherwise
Express A	Boolean	True if train A is an express passenger train; false otherwise
Express B	Boolean	True if train B is an express passenger train; false otherwise
Permanent A	Boolean	True if train A service is in the WTT; false if it is part of the STP
Permanent B	Boolean	True if train B service is in the WTT; false if it is part of the STP
Scheduled time difference to conflict A	Numeric	The time difference between train A stepping into the current berth and the time it is due to pass the conflict location
Scheduled time difference to conflict B	Numeric	The time difference between train B stepping into the current berth and the time it is due to pass the conflict location

Table 8.2: The features used for building machine learning models to predict the resolution of conflicts

8.3.7 Data transformation

The transformations are carried out in a very similar manner as for the travel duration models. The categorical variables are transformed using one-hot encoding, and the Boolean variables do not require transformation. The hour of the day, the month of the year and timestamp features are all transformed using sine and cosine transformations before being scaled to between zero and one. The travel duration through the previous berth, and the scheduled time difference to the conflict were first transformed using a quantile transformer and then scaled to between zero and one.

8.3.8 Evaluation metrics

The accuracy has been selected to evaluate the non-probabilistic results. This metric is simple to understand and provides the percentage of records assigned to the correct class. Only non-probabilistic results are reported for the evaluation and selection phases.

For the probabilistic classification results, the Brier loss can be used to evaluate the quality of the predictions. The loss produces a value between zero and one and calculates the mean square difference between the predicted probability and the true value:

$$\text{Brier loss} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{p_i})^2 \tag{8.1}$$

Where there are n records, $\hat{p_i}$ is the predicted probability value, and y_i is the true value for the i^{th} record.

8.4 Exploration phase results

For each pair of conflict locations, the machine learning model with the highest accuracy outperformed both types of benchmark models. However, as with the travel duration models, it was possible to select hyperparameters for the machine learning models so that they produced lower accuracy than the benchmarks.

The fourth modal benchmark model consistently produced the highest accuracy of all the modal benchmarks across all the conflict pairs. However, the programmatic model that achieved the highest accuracy varied across the conflict pairs. The gradient tree boosting and random forest models produced the highest accuracies of the machine learning models for all pairs of conflict locations. Similar to the results of the travel duration predictions, the ANNs were capable of producing accuracies that were close to that of the ensemble methods but proved slower to build.

8.5 Selection phase results

For each pair of conflict locations, the two sets of hyperparameters and models that produced the highest accuracies were carried forward to this phase. This decision meant that gradient tree boosting and random forest models were the only machine learning models represented in the selection phase. This modelling phase aimed to identify a single model and set of hyperparameters to build the final model for each conflict pair. As such, the selected machine learning models were built several times with small variations in the hyperparameters.

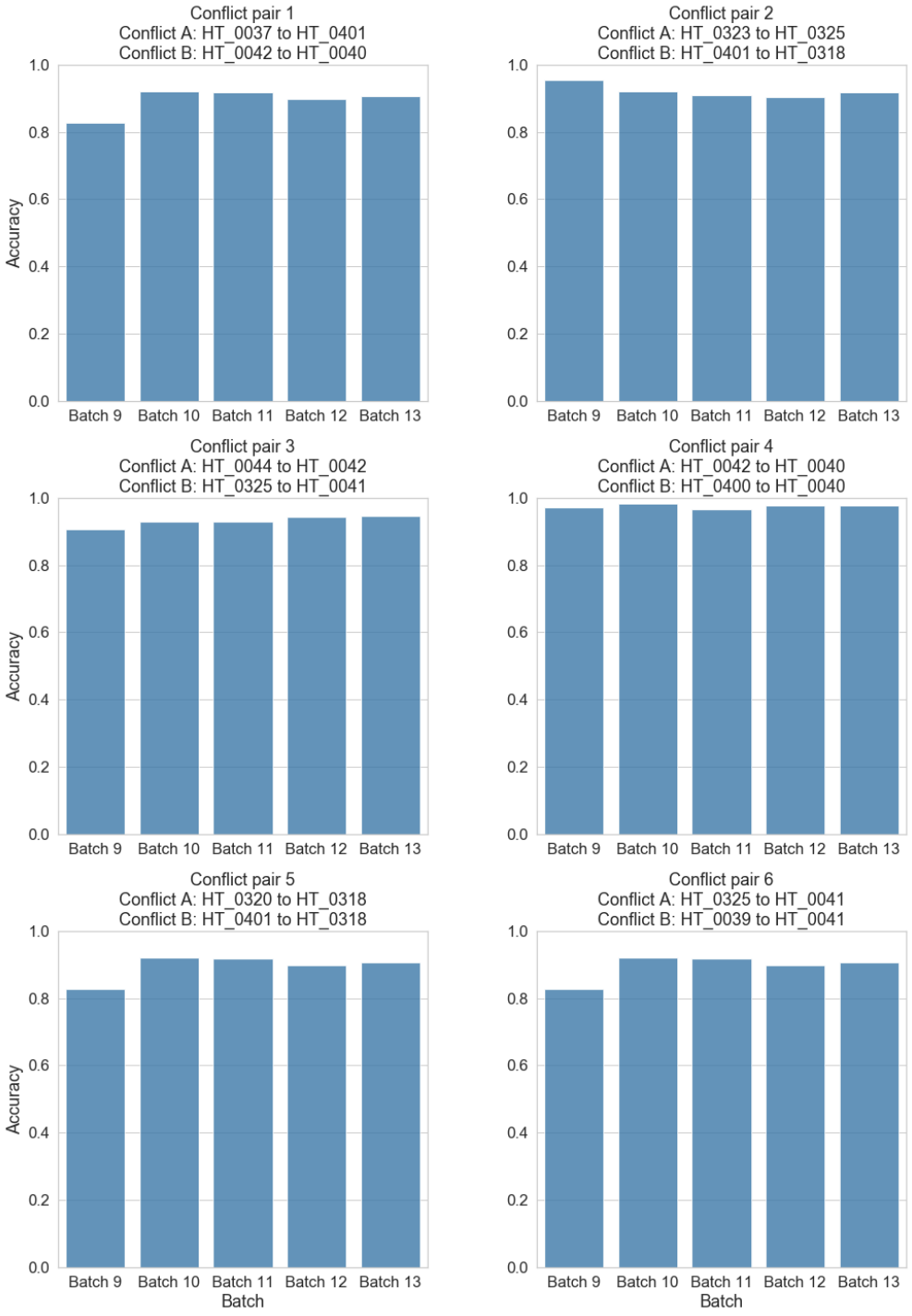

FIGURE 8.2: Bar charts showing the validation accuracy by batch number for each round of evaluation in the selection phase

The benchmark and machine learning models were evaluated using prequential block evaluation with the mean accuracy across all validation batches used to compare the models. The machine learning models were, again, able to outperform the benchmarks. For three pairs of conflict locations, a gradient tree boosting model produced the highest mean accuracy over the validation batches; for the other three conflict pairs, a random forest model was superior.

Figure 8.2 shows bar plots of the accuracy of the machine learning models that produced the highest mean accuracy for each conflict pair. Conflict pairs 1, 5 and 6 all exhibit a significantly lower accuracy for batch 9 than the other batches. Batch 9 coincides with the start of the Covid-19 pandemic, and some operational priorities will have changed during this time. These results show that operational changes can impact the accuracy of the machine learning models. However, the accuracies for batches 10 to 13 are reasonably consistent for all the conflict pairs suggesting that any timetable changes during these periods did not significantly impact the accuracy of the models.

8.6 Final evaluation

In the final stage of building the machine learning models, batches 1 to 13 were used as the training data, and batch 14 acted as the test data. The model and hyperparameters which produced the highest mean accuracy across the validation batches in the model selection phase were built as the final model. The training data were shuffled, and then 80% of the data were used to train an initial machine learning model, f, for each conflict pair. The remaining 20% of the data were used to calibrate the model using isotonic regression, creating an updated model, $\hat{f}$, for each conflict pair (see the details of the calibration process in Section 8.3.5).

A calibration curve can visualise if a classification model is well-calibrated. Calibration curves are constructed by dividing the interval between zero and one into bins of fixed width and plotting the mean predicted probability against the true frequency of each bin. Figure 8.3 shows calibration curves using a bin width of 0.1 for the model predictions for the first conflict pair. The dashed line shows where the curve would sit if it were a perfect fit.

Figure 8.3 shows that the classification model is well-calibrated for the training dataset: the calibration curve closely follows the line of equilibrium. The calibration is not as successful for the test dataset, with the calibration curve further away from the equilibrium. However, Figure 8.4 shows a bar chart of the number of records in each bin and the majority of the records in the test dataset have predictions which fall in the first and last bins: $[0, 0.1)$ and $[0.9, 1]$. Figure 8.3 showed that the predictions for these bins are well-calibrated. Therefore, even though the machine learning model is not as well-calibrated on the test data, the impact is restricted to a small proportion of the

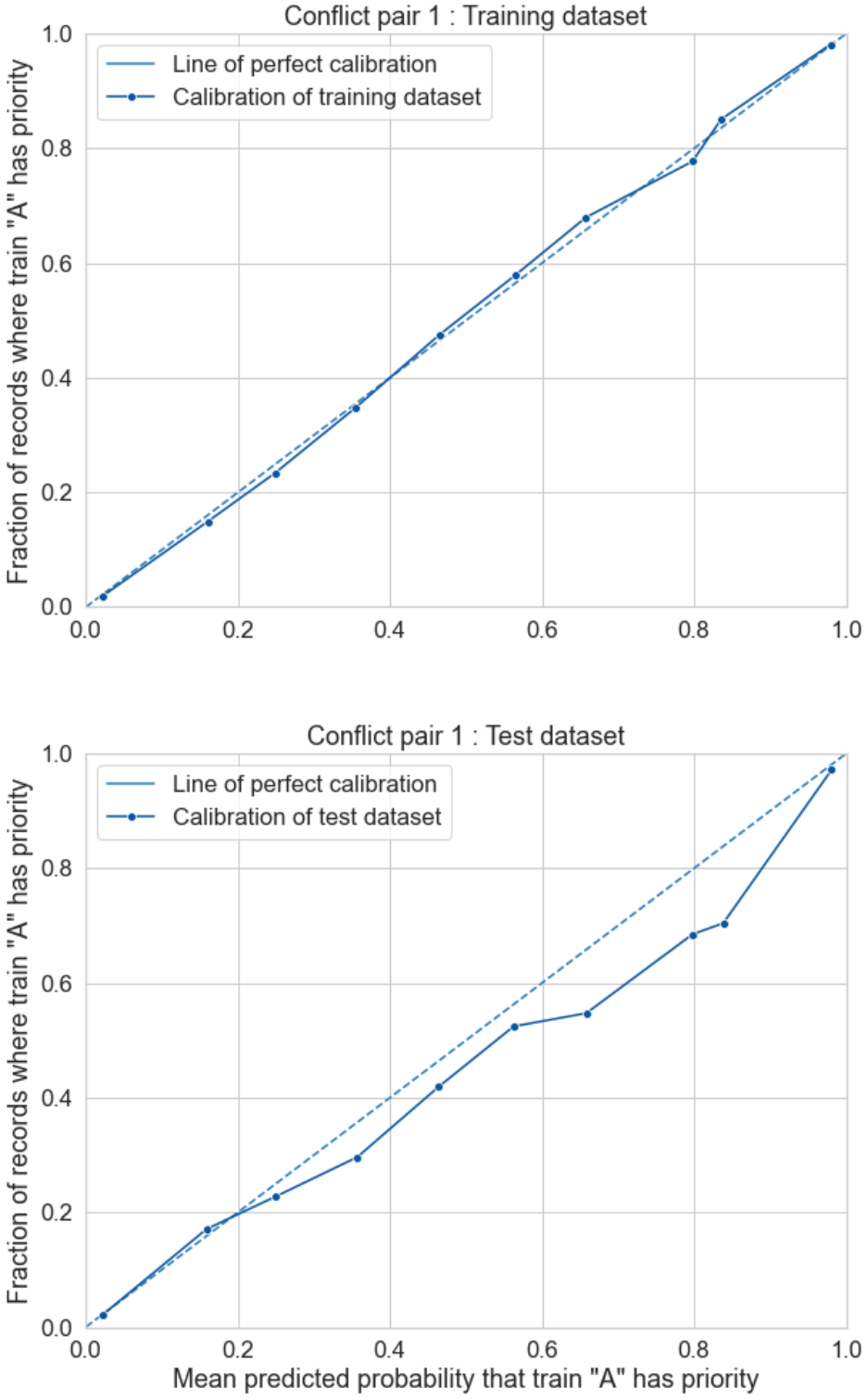

FIGURE 8.3: Calibration curves for the training and test data of the first conflict pair

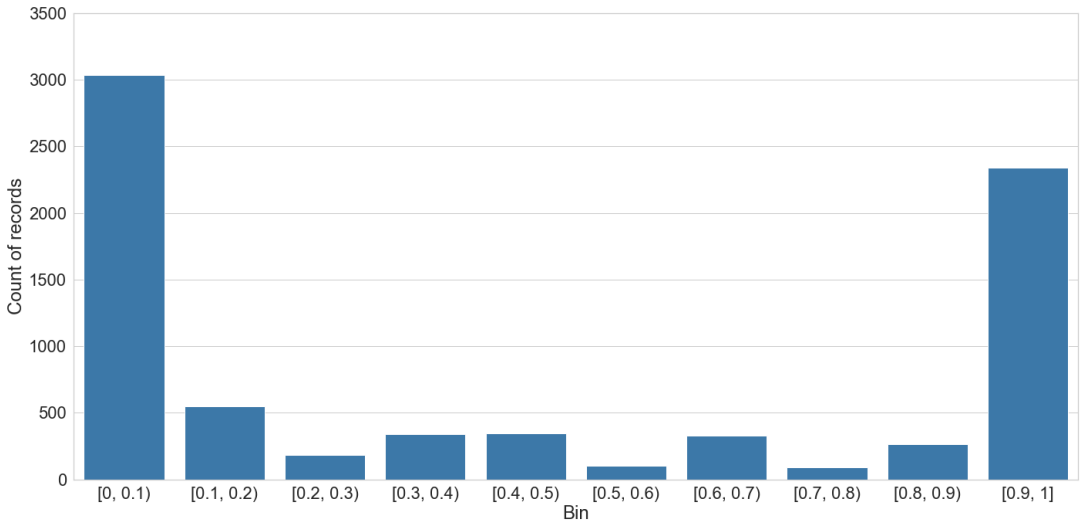

FIGURE 8.4: A bar chart showing the number of records in the test dataset for the first conflict pair where the prediction falls into discrete bins

data. This is reflected in the value of the Brier loss, which for the training dataset is 0.068, and for the test data set is 0.076. The scores for both datasets are close, although the value for the training dataset is lower, indicating that the model is better calibrated than the test data.

Table 8.3 shows the Brier loss for the fourth modal benchmark model (this had the highest accuracy of all the modal benchmarks) and the machine learning models. For all pairs of conflicts, the machine learning models produce lower Brier loss values than the modal model for each dataset. For conflict pairs with IDs 1, 3, 5 and 6, the Brier loss score on the test data is close to that of the training data. Conflict pairs 2 and 4 have more significant differences between the scores on the training and test datasets.

The accuracy of the machine learning models on the test dataset was higher than the modal and programmatic benchmarks for all conflict pairs. Table 8.4 records the highest accuracy for each type of model and conflict pair on the test dataset. Although the accuracies of the machine learning classifiers are higher than the other model types, they do not significantly outperform the benchmarks. However, given the size of the datasets, an increase in accuracy of just 2% results in hundreds of additional records in the test dataset being classified correctly. The machine learning models also have the advantage of predicting probabilities, which can be applied in a stochastic simulation.

The results also show that the three 'joining' conflict pairs (with IDs 4, 5 and 6) have higher accuracies than the crossing conflicts. The two trains in a conflict will travel onwards on the same track at 'joining' locations. Therefore, trains that are slow or stop frequently are unlikely to be prioritised over faster services. This observation is a likely reason why the 'joining' conflicts are more predictable than the 'crossing' conflicts. The

Brier loss scores for the 'joining' conflicts are also lower than for the 'crossing' conflicts (see Table 8.3), indicating that the models for the 'joining' conflicts are better calibrated.

The models presented here do not have full situational awareness of each conflict; they only consider two trains approaching a conflict location. As such, it is unsurprising that the models cannot make predictions with 100% accuracy. Factors external to the features of the two trains may also inform the decisions, and there may be some variation based on human factors. However, there may be many scenarios where the ability to predict decisions with at least 89% accuracy is sufficient, given that the models can also quantify the uncertainty of each prediction. The simulation model presented in the following chapter will present one application of the models where the accuracy is sufficient.

Conflict Pair ID	Conflict A		Conflict B		Brier Loss for Modal 4 model		Brier Loss for machine learning model	
	From Berth	To Berth	From Berth	To Berth	Train data	Test data	Train data	Test data
1	HT_0037	HT_0401	HT_0042	HT_0040	0.083	0.092	0.068	0.076
2	HT_0323	HT_0325	HT_0401	HT_0318	0.085	0.089	0.016	0.074
3	HT_0044	HT_0042	HT_0325	HT_0041	0.069	0.066	0.046	0.049
4	HT_0042	HT_0040	HT_0400	HT_0040	0.021	0.040	0.003	0.028
5	HT_0320	HT_0318	HT_0401	HT_0318	0.028	0.041	0.026	0.025
6	HT_0325	HT_0041	HT_0039	HT_0041	0.077	0.074	0.040	0.039

TABLE 8.3: The Brier loss scores for the final machine learning models on the training and test data, compared to the scores of the fourth modal benchmark

Conflict Pair ID	Modal				Programmatic					Machine Learning	
	1	2	3	4	1	2	3	4	5	RF	GB
1	57.77%	83.59%	85.57%	**87.00%**	82.81%	82.81%	**86.22%**	83.20%	86.12%	-	**89.13%**
2	71.94%	85.86%	86.63%	**87.04%**	76.61%	76.61%	80.85%	83.92%	**84.34%**	**89.41%**	-
3	66.72%	81.38%	86.02%	**91.57%**	87.38%	87.38%	83.16%	**89.78%**	83.63%	**93.64%**	-
4	57.36%	76.76%	93.56%	**94.08%**	**95.71%**	**95.71%**	94.58%	87.77%	86.11%	**96.47%**	-
5	89.75%	89.75%	94.45%	**94.80%**	93.97%	93.97%	**95.00%**	94.04%	94.79%	-	**96.58%**
6	75.24%	72.04%	81.81%	**89.92%**	87.46%	87.46%	79.34%	**89.29%**	79.45%	**94.78%**	-

TABLE 8.4: The accuracies of the final modal, programmatic, machine learning models on the test dataset to two decimal places; the highest accuracies for each conflict pair and category of model are highlighted in bold and the highest overall accuracy for each conflict pair is underlined. RF = Random Forest, GB = Gradient Tree Boosting

8.7 Discussion

8.7.1 Contributions to the literature

This chapter has presented machine learning classification models that predict the resolution of pairs of trains approaching conflict locations. As such, the models attempt to mimic the actions taken by signallers. The work is similar to that of Dündar and Şahin (2013), where an ANN was developed to predict the priority given to pairs of trains. However, the modelling work presented in this chapter is the first known example of such predictions using British track-occupation data. It has also extended the work of Dündar and Şahin (2013) in various ways, which will be discussed in turn.

Benchmark models

While the ANN developed by Dündar and Şahin (2013) produced a high accuracy, they did not compare the result to a simpler benchmark model. The work in this chapter developed two types of benchmark models: modal and programmatic. The results showed that machine learning models could outperform the benchmarks, although not by significantly large amounts. There may be some use cases where a simple benchmark model would be sufficiently accurate.

Quantity of data

The work of Dündar and Şahin (2013) used 331 data records taken from a ten-day period, of which only 173 were used to train the neural network. By contrast, this work has used over 800,000 records over three years and six conflict pairs. This amount of data has also allowed for the accuracy of the models to be analysed over time. The operational changes due to the Covid-19 pandemic were shown to affect the accuracy of the models for some conflict locations.

Comparing ensembles and ANNs

The research presented here compared different machine learning approaches. Tree-base ensembles achieved higher accuracies than the ANNs while using less computational time.

Conflict location types

This work distinguishes between ‘crossing’ and ‘joining’ conflict locations. The results found that machine learning classifiers could predict the resolution of conflicts with

higher accuracy for 'joining' locations, suggesting they are more predictable than 'crossing' locations.

Probabilistic classification

This work has gone beyond predicting the models' accuracy to consider the calibration of probabilistic predictions. The results have shown that the machine learning models can produce probabilistic predictions with a lower Brier loss than the modal benchmarks[1].

8.7.2 Further work

The following chapter will continue this work by applying the predictive models within the machine learning-assisted simulation to model traffic management decisions. Other than applying the models within a simulation, there are additional opportunities to carry on this work, which will be considered in turn.

Additional features

As discussed in Section 8.6, the models do not have full situational awareness and only consider the two trains involved in the conflict. Further work could investigate whether additional features representing the location of other trains could be incorporated into the model to improve the accuracy of the predictions. For example, the resolution of a conflict may be influenced by a third train closely following one of the others.

One problem with including features of additional trains is how to encode these data. There are many different locations that additional trains could occupy, and many situations will not have any additional trains at all. If additional trains were included in the input, then there is the potential to end up with a large, sparse set of features. A sparse set of input features can lead to models that take a long time to build, and the additional features may not improve the accuracy.

Expansion of the problem space

The models built in this chapter considered trains up to five steps from a conflict location. A natural extension of the work would be to extend the problem space to investigate whether predictions could be made about conflict resolution when the trains are further away from a conflict location.

[1]Note that probabilistic versions of the programmatic models were not could not be constructed in their current form so could not be compared using the Brier loss.

Generalisation of the problem

The models that have been developed are specific to the particular junctions and conflict locations in the demonstration area. The work identified that 'joining' and 'crossing' conflict locations have distinct characteristics regarding the decisions that signallers make, and further work could investigate whether more general models could be developed. It may be that a single model could be created for, say, all 'joining' conflicts in a given train describer area.

Analysis of signallers

The models could also be used to analyse signallers' decision-making, which could inform regulation policies or training. The signaller that made each decision is unknown in the open data, but if that information were available, the predictability between signallers could be examined. An interesting question to investigate would be whether experienced signallers were more or less predictable than their junior counterparts.

Other traffic management decisions

There is the opportunity to investigate whether machine learning could also be used to predict other traffic management actions, such as platform selection, route selection or service cancellation. For example, at large stations, trains sometimes end up dwelling at a different platform to the one listed in their schedule due to disruption to other services. Medeossi and de Fabris (2018) identified that existing simulators struggle to model such changes in platforms; machine learning models may present an opportunity to make predictions about which platform would be selected for each train.

Calibration process

The calibration of the machine learning models could be considered in an earlier phase of the modelling process, such as during the selection phase. This would allow alternative methods for calibration to be investigated.

8.8 Conclusion

This chapter has presented machine learning classification models which predict the priority of two trains approaching a conflict location. The machine learning models were able to outperform statistical and programmatic benchmark models. The results show that tree-based ensemble methods were the most accurate predictors, and despite

not having full situational awareness, they were able to make predictions with at least 89% accuracy. The calibration of the machine learning models was also examined, and their probabilistic predictions also outperformed benchmark models.

The next chapter will demonstrate how these models can be applied within the machine learning-assisted simulation model introduced previously. The models developed in this chapter will be embedded within the simulation to determine signalling actions.

Chapter 9

Simulating signalling decisions

9.1 Introduction

The previous chapter demonstrated how supervised machine learning could be applied to predict the resolution of conflicts on the railway for six conflict locations. Resolving conflicts was framed as a classification problem with the models predicting which of two trains has priority at a junction. Tree-based ensemble methods were built and produced accuracies of over 89% on an unseen set of test data. None of the models could predict the resolution with 100% accuracy, and the work also investigated the probabilistic output of the models.

This chapter applies the models built in the previous chapter within the machine learning-assisted simulation model. Chapter 7 presented some details of the simulation model, and Section 9.2 provides further information about how traffic management decisions are implemented. Sections 9.3 and 9.4 briefly discuss verification and the set-up of the simulation, respectively. Section 9.5 presents the results of the deterministic and stochastic simulations using machine learning models for conflict resolution. Section 9.6 discusses the approach, highlighting the contributions of this research, limitations of the approach and opportunities for further work.

9.2 Traffic management in the simulation

9.2.1 General logic

Chapter 7 provided details of the main areas of the simulation model developed for this thesis. This section continues the presentation of the simulation model by describing the traffic management functionality. Figure 9.1 displays a UML diagram of the main classes involved in traffic management.

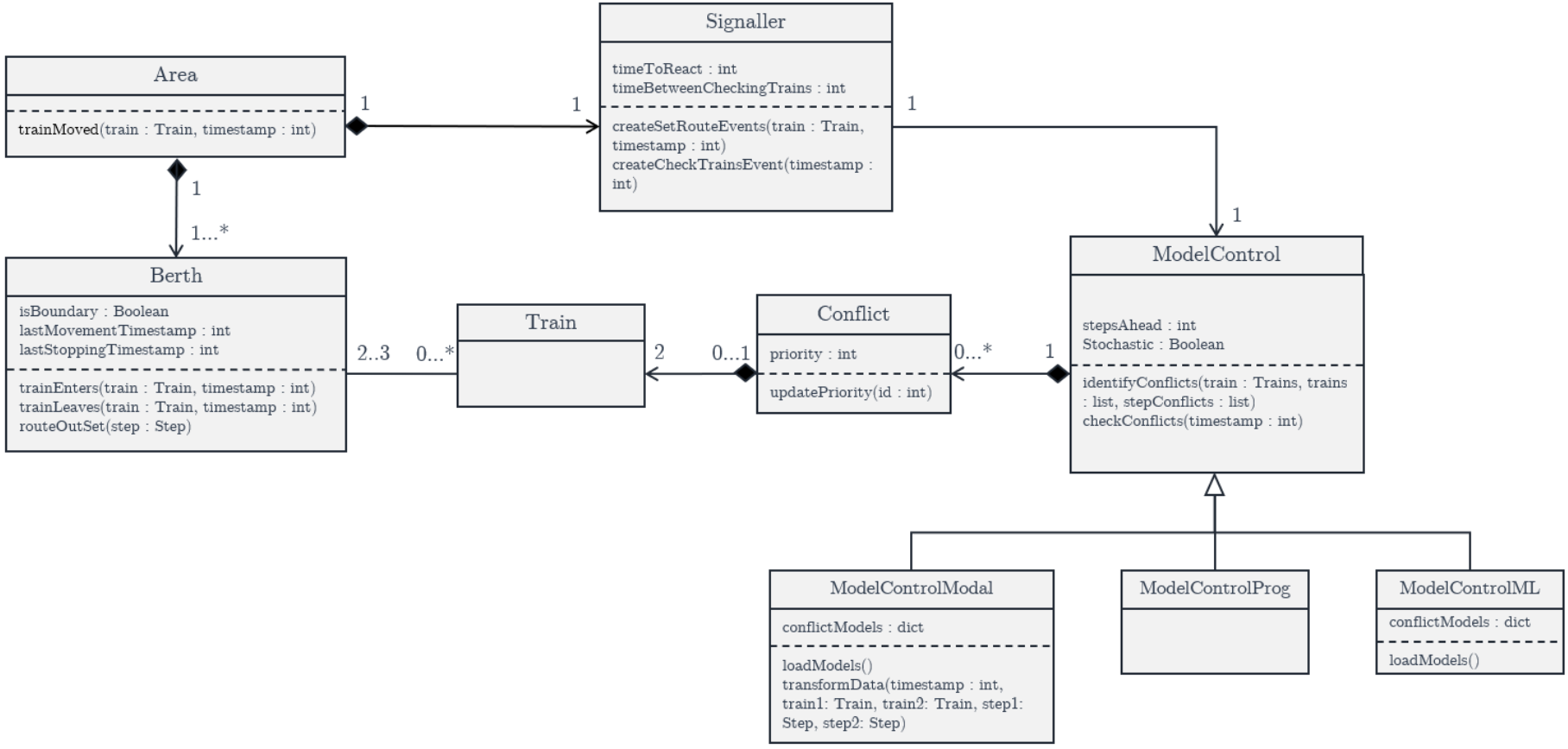

FIGURE 9.1: A UML representation of the classes concerning traffic management

Each `Signaller` is associated with precisely one `ModelControl` object. The `ModelControl` class defines the functionality for resolving conflicts within the simulation. The various child classes which inherit from `ModelControl` override the functions with their own logic. One crucial attribute of the `ModelControl` class is `stepsAhead`, which records the number of steps the model can 'look ahead' to resolve conflicts for each train. This attribute is calibrated in the input data, and for the experiments in this chapter is set to five. A higher number increases the computational time to run the simulation, whereas a lower number risks missing the opportunity to analyse some conflicts.

The `ModelControl` is composed of `Conflicts`. Each `Conflict` is, in turn, composed of two `Trains`, and it records which of the trains will be given priority. When a Train moves into a `Berth`, the berth object calls the `trainMoved` method of its `Area`. The `Area` object, in turn, calls the `createSetRouteEvents` method of its `Signaller`, sending the `Train` object as an argument.

The `createSetRouteEvents` method of the `Signaller` class calls the `identifyConflicts` method of its `ModelControl` object, sending a list of `Train objects` in the `Area` and a list of pairs of conflict steps in the area. The `identifyConflicts` method is not overwritten by any child classes of `ModelControl`. The method looks ahead a given number of steps on a train's journey and creates a new `Conflict` object for any newly identified conflicts. The number of steps the method looks ahead is set as a parameter in the input file and is set to five for the work in this chapter.

Having ensured that the `ModelControl` object has an up-to-date list of `Conflicts`, the `checkConflicts` method of `ModelControl` is then called with the current timestamp passed as an argument. The `checkConflicts` method checks that the priorities of each `Confict` are still valid and sets the priority for any that have not yet been set. The logic in the `checkConflicts` method is overwritten in each child class, and the logic specific to each child is described in the following subsections.

The remainder of the `createSetRouteEvents` then calculates how many steps can be set for the given `Train` object up to the number of steps the `ModelControl` object can look ahead. The number of steps that can be set will be determined by any `Conflict` objects, their priority and the location of other trains. The steps are not set instantly; instead, events are created to process the setting of steps after a fixed delay.

Logic has been developed to prevent some scenarios where deadlocks may occur. However, the simulation is not guaranteed to avoid all deadlocks.

9.2.2 Modal control model

The class `ModelControlModal` implements the fourth modal benchmark model. This model is the only modal benchmark to be implemented, as it consistently outperformed

the others. The model was saved in a text format and could be loaded when the simulation was initialised.

9.2.3 Programmatic control model

The class `ModelControlProg` inherits from `ModelControl`. It requires an attribute to be passed in the input data to determine which logic will be applied when checking conflicts. The options include the logic representing the five programmatic benchmark models. In addition, there are options for the following logic:

- Exact: this will model the traffic management decisions as per the historical data
- Varied programmatic: this will model vary the programmatic logic by pair of conflicts; the logic to use for each pair will be determined in the input data

The logic for modelling the exact traffic management decisions from the historical data uses the timestamps of the actual movements (stored in the `Schedule` objects) to determine the order trains passed over the junction.

All the programmatic logic is 'hard-coded' into the simulation; it is not loaded from external files.

9.2.4 Machine learning control model

The class `ModelControlML` deploys machine learning models to make decisions about the resolution of conflicts. The machine learning models built in the final phase of the modelling process in Chapter 8 were saved, along with the necessary objects for transforming the data. These are loaded at the beginning of the simulation and used to make predictions throughout the execution of the model.

It is possible to configure the simulation model to use the probabilistic output of the machine learning models to make stochastic predictions. In this case, a pseudo-random number, p, is generated for each prediction. If p is less than the predicted probability, then train 'A' has priority; otherwise, train 'B' has priority.

9.3 Verification

In addition to the verification activities discussed in Chapter 7, further work was required to verify the implementation of the traffic management logic. Activities undertaken include code walk-throughs and test scenarios run through the simulation to confirm

that the logic was implemented correctly. The test scenarios were run using exact travel durations from input data; this allowed scenarios to be constructed where the traffic management model would prioritise, say, train 'A', but train 'B' would be the train to arrive first at the conflict location. The test would verify that train 'B' is held at the conflict location to allow train 'A' to pass through first.

9.4 Simulation set-up

The experiments in this chapter use the same infrastructure inputs as in Chapter 7 (shown in Figure 7.6). The same timetable data were also used for this chapter's experiments, which ran from 3 a.m. on 2021-08-09 to 3 a.m. on 2021-08-16.

The simulation was calibrated with the same parameters as in Chapter 7. However, unlike the previous work, the experiments in this chapter will vary the traffic management models rather than the actions being the same as in the actual data. Some of the experiments model the train travel durations using the actual travel durations from the historical data, and others use the machine learning models used in Chapter 7.

9.5 Simulation results

9.5.1 Deterministic simulation

The traffic management models were initially assessed by running the simulation in deterministic mode. In this scenario, the travel durations are determined by the historical data, so trains are set to take the same time to travel through berths as they did in reality. With this configuration, a traffic management model that assigns priorities with 100% accuracy will result in the same total travel duration as the historical data.

However, suppose a traffic management model does not set priorities with 100% accuracy. For example, if train 'A' is prioritised through a junction at the expense of train 'B', which is a decision that does not match the action in the historical data. In that case, train 'A' will not run any faster through the simulation as the travel durations are taken from this historical data. As train 'B' is deprioritised, it will be held at a red signal and spend longer in the berth preceding the junction than in the historical data. Once train 'B' is granted movement authority, it will carry on its journey, taking the travel durations from the historical data. However, its overall travel duration will be longer than the historical data. Therefore, any traffic management model that does not replicate the historical actions with 100% accuracy will result in a greater total travel duration of trains than in the historical data.

From the historical data, the true total travel duration of all trains through the simulation area is 1,470,631 seconds (408.51 hours), which was noted in Chapter 7. Table 9.1 shows the total travel duration of the trains in the simulation when it was run in deterministic mode, applying different traffic management models. As well as the machine learning models being used to determine the resolution of conflicts, the fourth modal model and the programmatic models were also applied in separate simulation runs.

Table 9.1 shows that the total travel duration for each scenario is higher than the actual total duration. This result is expected as none of the traffic management models could classify the signalling actions with 100% accuracy. The scenario that employed the machine learning models had the lowest total travel duration and was within 1% of the total duration. This result demonstrates that the machine learning models can mimic the signalling actions more accurately than the other models, which agrees with the results in the last chapter.

However, some scenarios involving programmatic models produce total durations that do not follow the results of the previous chapter. For example, the varied programmatic model (which varies the programmatic by conflict location according to which logic produced the highest accuracy) has a higher total travel duration than the first and fourth programmatic models. This observation shows that selecting the models with the highest accuracy for each conflict location does not necessarily result in the most realistic model within the simulation.

9.5.2 Stochastic simulation

Chapter 7 demonstrated that simulating travel durations stochastically by sampling from quantile regression models can lead to realistic variation in train travel durations. Over 600 iterations, the median total travel duration settled to within 0.57% of the actual total travel durations. That result was achieved by simulating the same traffic management actions as the historical data. This section presents results for modelling the stochastic train movements with conflict resolutions also determined by the machine learning models.

Table 9.2 shows the results of three scenarios. All three scenarios modelled the train travel durations stochastically using the machine learning models, and, in each case, the simulation ran for 600 iterations. The first scenario is from Chapter 7, with the traffic management decisions taken from the historical data.

The second scenario uses machine learning to model the traffic management decisions but models them deterministically. Note that this scenario does not mean that exactly the same traffic management decisions are being applied in each iteration (as is the case with the first scenario); instead, it means that the prediction of each machine learning model is rounded to zero or one without using a random number to decide the value.

Traffic management model	**Total travel duration (seconds)**	**% increase on the actual duration**
Programmatic 1: Priority is given according to the scheduled order of trains	1,530,951	4.10%
Programmatic 2: Passenger trains have priority over ECS and freight	1,542,944	4.92%
Programmatic 3: Passenger trains have priority over ECS and freight; express passenger trains have priority over non-express if closer to conflict	1,537,782	4.57%
Programmatic 4: A train gives priority to a non-stopping train if the non-stopping train is closer	1,529,963	4.03%
Programmatic 5: A stopping train gives priority to a non-stopping train if the non-stopping train is closer; passenger trains have priority over ECS and freight; express passenger trains have priority over non-express if closer to conflict	1,536,914	4.51%
Programmatic varied: the logic is varied by conflict location according to the logic that produced the highest accuracy for each location	1,534,817	4.36%
Modal 4	1,497,246	1.81%
Machine learning	1,481,046	0.71%

TABLE 9.1: Total travel duration of trains travelling through the simulation for each traffic management model

The models will still make predictions according to the situation they are presented with, so the travel durations of the trains may affect the decisions that are taken. The third scenario also uses the machine learning traffic management models but applies them stochastically.

Table 9.2 presents the minimum, maximum and median total travel duration for all three scenarios over the 600 iterations. For the three scenarios, the actual duration falls within the minimum and maximum bounds, and the median total travel duration settles to within 1% of the actual value after 600 iterations. This result confirms that the machine learning models of traffic management can be used alongside the stochastic travel durations to achieve realistic signalling decisions.

The results in Table 9.2 show that stochastic traffic management actions have little impact on the output compared to deterministic actions. This observation contrasts the results in Chapter 7, which presented superior results for stochastic travel durations compared to the deterministic scenario. There are several possible reasons for stochastic travel durations having a more significant impact than stochastic traffic management. The first concerns the number of trains each affects. The travel durations are modelled for every combination of berth and train. However, over the course of a week, many trains will pass over the conflict locations without needing a signaller to resolve a conflict with another train. The exact number of trains requiring a conflict resolution decision will vary between iterations; however, if the historical data were simulated exactly, then around 58% of the trains would be involved in at least one conflict resolution.

The second reason the implementation of stochastic traffic management decisions may not have a significant impact is the distribution of probabilities in the output. Figure 8.4 showed the distribution of the probabilistic output of the machine learning model for the first conflict pair on the test data. Most of the outputs were less than 0.1 or greater than or equal to 0.9, which was also true of the other conflict pairs. Therefore, most predictions are made with a high degree of certainty and will lead to less variation in a stochastic simulation.

Scenario	**Total travel duration (seconds)**			**% difference between the median and actual total travel duration**
	Minimum	**Maximum**	**Median**	
1: Conflict resolution from historical data	1,458,983	1,536,951	1,479,002	0.57%
2: Conflict resolution from machine learning (deterministic)	1,448,368	1,523,528	1,465,576	-0.34%
3: Conflict resolution from machine learning (stochastic)	1,432,377	1,525,856	1,482,772	0.83%

TABLE 9.2: The minimum, maximum and median total travel duration of trains over 600 iterations of stochastic scenarios; all scenarios used machine learning to model the travel durations.

9.6 Discussion

9.6.1 Contributions to the literature

The work presented in this chapter is the first known application of supervised machine learning to make traffic management decisions in a simulation. The results of the deterministic simulation that used historical data to model train travel times showed that the machine learning models could replicate the actions to resolve conflicts more closely than the alternative benchmark models.

This work has also demonstrated how conflicts can be resolved stochastically by utilising the probabilistic output of the machine learning models. There was no major difference in the results between modelling the conflict resolution actions deterministically and stochastically. However, there may be scenarios where it is advantageous. For example, a stochastic model may be beneficial for complex junctions, for conflict locations where a high accuracy could not be achieved, or if a scenario was modelled with a deliberately high number of conflicting trains.

One of the advantages of the approach is that it models the actions that signallers *actually* take. This method could be used to compare the signallers' actions with those specified by train regulation policies. It could show how closely the signallers follow recommended actions and whether they adapt to make more or less optimal decisions than the expected actions.

The ability to mimic signallers' actions may also be beneficial when modelling the performance of a new or updated timetable. The machine learning models will predict the actions signallers will take when presented with a new timetable, which could help provide realistic performance assessments. It could also provide an opportunity to compare their actions to other strategies. If a more optimal strategy was developed, then this could be communicated to signallers prior to the introduction of the timetable.

Another advantage of this method of modelling traffic management actions is that building the machine learning models for each conflict location could largely be automated. As shown in the previous chapter, the logic applied at each conflict location will likely be unique, with the 'joining' and 'crossing' conflicts exhibiting different characteristics. If a user has to create logic for each conflict location in a simulation, setting up a model could take significant time. In contrast, there is little intervention required from a user when creating the machine learning models once the process is established.

9.6.2 Limitations of the approach

This approach has similar limitations as those discussed when modelling train travel durations in Chapter 7; in summary:

- It may be difficult to adapt existing simulation models to use machine learning for traffic management
- The work benefits from the use of some high-performance computing facility
- The approach is unlikely to be suitable if new infrastructure is to be modelled

It may be possible to overcome the final point if more generalised machine learning models could be built, as suggested in Section 8.7.2.

In addition, the results in Chapter 8 showed that the accuracy of the machine learning models for traffic management might reduce if there are significant changes to operating policies, such as during the start of the Covid-19 pandemic. However, the ability to model the resolution of conflicts stochastically in a simulation by using the probabilistic output may compensate for a slight decline in accuracy.

9.6.3 Further work

As discussed in Section 7.6.3, this work would benefit from extending the existing simulation model to model direct delays and entry times stochastically. Additionally, the simulation could be run with traffic management models that are able to resolve conflicts further than five steps ahead. The current limitation on five steps means that some conflicts may not get identified due to one train being further away than five steps. In Chapter 5, the analysis in Figure 5.7 showed that a significant proportion of trains would get held in berth HT_0400 to wait for a train six or seven steps away. Extending the number of steps to look ahead to seven (or higher) would ensure these would be identified.

If additional features such as direct delays and entry times are also modelled stochastically, then the simulation could be applied to provide an analysis of the performance of a new or updated timetable. The results could be used to identify 'critical' services at high risk of being delayed by looking at the distribution of delays or travel duration of individual train services over all iterations.

This approach to simulation modelling could be used to analyse signalling decisions. The simulation model could consider the optimality of decisions by comparing signallers' current traffic management actions (represented by the machine learning models), train regulation policies, and alternative strategies. Many studies have looked at the optimisation of traffic management actions. The simulation model presented in this chapter could provide a framework to model the status quo regarding traffic management and alternative strategies. The ability of the simulation approach presented in this chapter to model the variation in travel durations using machine learning would be beneficial when considering the impact of different strategies.

9.7 Conclusion

This chapter has applied the machine learning models built to predict the resolution of conflicts within a simulation model of the demonstration area. The machine learning models could resolve conflicts and achieve realistic results when comparing the total travel duration from the model to the actual value. When the trains travelled throughout the simulation with their historical travel durations and the machine learning models were applied deterministically, the total travel duration of the model was within 0.71% of the true value.

The chapter has also demonstrated how the resolution of conflicts can be modelled stochastically by using the probabilistic output of the machine learning models. The results showed that there might be less benefit to modelling conflict resolutions stochastically than the train travel durations. However, it may be more useful for other locations on the network where the accuracy of the machine learning models might not be so high.

The next chapter will summarise the results of this thesis and make recommendations for future work.

Chapter 10

Conclusion

10.1 Introduction

This thesis began by identifying simulation modelling as a valuable approach for aiding many planning decisions in the railway domain. It is particularly well suited to problems that would be impractical or too costly to experiment with on the actual railway network, such as analysing the performance of a new timetable. Train movements and signalling actions were recognised as two features crucial to simulating a railway, but they had some limitations in their implementation in existing simulation software. Machine learning-assisted simulation modelling was identified as a potential approach to overcoming some limitations, and this thesis aimed to develop and evaluate a machine learning-assisted simulation approach to railway modelling.

The objectives of this work (outlined in Section 1.2) were to develop machine learning models to predict travel durations and traffic management decisions and a machine learning-assisted railway simulation model. The previous four chapters presented the results relating to these three objectives.

Section 10.2 summarises the main results and most significant contributions. Section 10.3 makes recommendations for future work, and Section 10.4 discusses the policy implications and potential applications of the work

10.2 Main results

10.2.1 Machine learning for predicting travel durations

Established simulation models use motion equations to model travel durations. A weakness of this approach reported in the literature is that it cannot fully represent the stochastic nature of train movements. This thesis identified supervised machine learning as a potential approach to predict train travel durations. The hypothesis was that the variation in travel durations could be predicted as the machine learning models would make predictions conditional on various features such as the train type, the punctuality of the service and the previous travel durations.

Supervised machine learning models were developed to predict the travel duration of trains through 40 berths on the British network using Network Rail's open data feeds. The data were segmented by two movement types (unrestricted and restricted) and three types of train (passenger, freight and ECS). Machine learning models were built for each segment with more than 200 data records. Gradient tree boosting was selected as an algorithm for the models as it could achieve similar results to artificial neural networks (ANNs) with less computational effort. Quantile regression was used to create a different model for each of the following values for the quantile τ:

$$0.05, 0.15, 0.25, 0.35, 0.45, 0.5, 0.55, 0.65, 0.75, 0.85, 0.95$$

The models built using the absolute loss function ($\tau = 0.5$) were generally not sufficient to replicate the full range of variation in travel durations. However, the results did vary by berth, with the lowest MAE achieved of 0.79 on the test data for unrestricted movements of passenger trains travelling through berth HT_0031. The results for the $\tau = 0.5$ models may be improved with additional features or, in some cases, larger samples of training data.

Without access to additional data, the work showed that the distribution of travel durations could be modelled by selecting a τ value at random to predict each record. Selecting a random τ value does not guarantee an improvement in the individual predictions for the records; however, it does lead to an improved model at the population level representing the distribution of travel durations. This work has demonstrated an alternative method for modelling travel duration using quantile regression that can reflect the inherent variation in the data. One key strength of the approach is that no assumptions about the underlying distribution of the travel durations are required. The result is relevant when travel durations need to be modelled stochastically, such as in the simulation model developed in this thesis.

10.2.2 Machine learning for predicting traffic management decisions

The representation of traffic management decisions in simulation models was also identified as a limitation of existing software. Established simulations allow only simple rules to be provided or are very time-consuming to set up more complex rules. Supervised machine learning was identified as a possible approach to resolve conflicts as the models can be trained to make conditional predictions, given a range of input features.

Machine learning models were built that predicted the outcome of a conflict between two trains at six conflict locations. Tree-based ensembles outperformed ANNs, and the models achieved an accuracy of at least 89%. The models were able to outperform statistical and programmatic benchmark models. Moreover, the models went through a calibration stage, and the probabilistic output was evaluated using the Brier loss. The models were not as well-calibrated on the test data as on the training data set. However, the Brier loss for the machine learning models was lower than that for the modal benchmark models indicating that the machine learning models were better calibrated than the benchmarks. The work demonstrated that machine learning could be used to predict traffic management decisions despite the models not having full situational awareness.

10.2.3 Machine learning-assisted simulation

This work developed a simulation that could utilise machine learning models to determine the travel durations of trains and the actions of signallers. Experiments were carried out that simulated a week's worth of activity on a section of the British railway network. The application of machine learning models varied in each experiment, and the simulation was run in deterministic and stochastic modes. The total duration of all the trains travelling through the simulation was compared with the true total travel duration for the week.

When both the travel durations and signalling decisions were modelled stochastically in the simulation, the actual total travel duration fell within the minimum and maximum total durations achieved over 600 iterations. Moreover, the median total travel duration across all iterations of the simulation was within 1% of the actual value. These results demonstrate that machine learning-assisted simulation modelling is a viable approach capable of modelling realistic performance. As the machine learning models are built using historical data, they represent the actual behaviour of signallers and train movements rather than theoretical or idealised behaviours.

The experiments also showed that modelling travel durations stochastically had a more significant impact on the output than modelling the signalling actions stochastically. Deterministic travel durations would significantly underestimate the total travel duration of all the trains on the network. However, the ability to model signalling decisions

stochastically should not be dismissed. There may also be other locations on the network where the ability to model signalling decisions stochastically would have a more significant impact.

10.3 Recommendations for future work

10.3.1 Machine learning for modelling travel durations

If the work were to continue within Network Rail, then it may be beneficial to integrate some additional data sources that are not publicly available. The first is additional data on freight trains, such as the type of freight they are transporting. The second is the train describer S-Class data. While the S-Class data are available in Network Rail's open data feeds, few sources of open information are available to assist with decoding the data. In the case of the Havant TD area, some of the available information was out of date and incorrect. Including S-Class data would improve the identification of restricted movements. It would also allow the prediction for restricted movements to be changed to the time difference between movement authority being granted and the train stepping into the next berth.

There is also an open data feed on temporary speed restrictions. Although this is open data, there were no open archives of the data that were found to be available, so it was not included in the work. If Network Rail held archives of the data concerning speed restrictions, this could be used to improve the models. Records related to movements under temporary speed restrictions could either be removed or the details of the restrictions could be added to the data set.

Some of the machine learning models that were built appeared to suffer from having a small quantity of training data. There are two approaches to this issue. The first is to include larger quantities of historical data (if these were archived by Network Rail). The second would be to include features of berths such as berth length and gradient and to build more general models. For example, it might be possible to build a single model for freight trains rather than building one for each berth.

Quantile regression proved to be a powerful approach to modelling the distribution of travel durations, and there are several options for further investigation in this area. For example, there is no reason why an evenly distributed number of quantiles should be generated. As the distributions are skewed to the right, it may not be necessary to predict all the quantiles less than 0.5 and perhaps predict more that are greater than 0.5.

The current approach to quantile regression suffers from not investigating quantiles other than 0.5 earlier in the development cycle. There is no reason why suitable hyperparameters for $\tau = 0.5$ will also be appropriate for other τ values. However, introducing other quantile levels earlier in the development cycle could be time-consuming. It would be interesting to investigate quantile regression further using ANNs, whereby a single model could be built with multiple outputs – one for each quantile. Initial investigations showed that such an approach would be feasible.

10.3.2 Machine learning for modelling traffic management decisions

If the work were to continue with Network Rail, then it would be worth investigating whether there are any data to identify which signallers took each decision. Such data could be handled in pseudo-anonymised form with a unique identifier used for each signaller along with the length of service – rather than using personal data. These data could lead to some interesting analysis about the predictability of different signallers and investigate whether their experience impacts their decision-making.

Further work would be recommended to investigate whether more general models could be built – perhaps one for 'crossing' conflicts and another for 'joining' conflicts. Generalised models would lead to fewer being required and more training data available for each model. There are likely to be limitations regarding the generality of the models, which would need to be explored. For example, whether a single train describer area is the extent of generality.

Another area of interest would be developing a method to represent a wider network area for input into the models. The location of other trains in the vicinity of a conflict location may affect the decision that is taken, and including them in the input to a model may improve accuracy. Some initial experimentation in this work found that developing a suitable representation was not trivial. For example, it is necessary to identify which locations on the network will influence the decision if they are occupied. These locations may not be limited to the berths directly leading to the conflict being modelled. A two- or three-dimensional array of data is a plausible representation of the data, which would lead to ANNs being used to make predictions. Given the high accuracy of the existing models, it is likely that there are relatively few examples where additional train services will influence the decision. Therefore, the records where there are additional trains may need to be weighted in some way for the impact of the additional train to be realised in the model.

Despite the complexity of modelling a wider network area as input data, it would be beneficial as it could lead to other traffic management decisions being predicted. For example, if platform changes or route selection are to be modelled, then these would

require a greater network area as input. Such models could also be included within a simulation model to expand the potential of traffic management decision-making.

This thesis has focused solely on the use of supervised machine learning. Reinforcement learning is another paradigm of machine learning which has been applied to optimisation problems concerning traffic management and rescheduling (Šemrov et al., 2016; Zhu et al., 2017; Ning et al., 2019; Zhu et al., 2020; Kubosawa et al., 2022). Indeed, both Zhu et al. (2017) and Kubosawa et al. (2022) use a simulation model to demonstrate the effectiveness of the strategies developed using reinforcement learning. Whether reinforcement learning could also be applied to model signallers' existing (rather than optimal) strategies is an open question and could follow from this work.

10.3.3 Machine learning-assisted simulation

The simulation model would benefit from further development, allowing entry times into the simulation area and direct delays to be modelled stochastically. Entry times have the potential to be modelled using quantile regression using the train movements dataset, although some additional features may be required. For example, there is likely to be a relationship between the delay with which trains enter an area of the network and the delay with which trains leave, as delays can propagate throughout the network. It may be that some additional features concerning the overall delay of trains in the previous few hours would be beneficial to the modelling.

The simulation would also benefit from additional metrics being developed and recorded throughout the execution of the model. For example, recording the delay of trains as they pass TIPLOCs or logging secondary delays, such as when a fast train is following a slower train.

A graphical user interface could be developed for the model. An interface could make setting up the model easier for a user. The interface could show the network layout and be used to animate an iteration of the stochastic simulation. Such visualisation may help to increase confidence in the output of the model.

10.4 Implications for Network Rail

10.4.1 Planning horizons

Chapter 2 identified simulation modelling as mainly being applied to problems in the strategic and tactical planning horizons. A typical application of simulation modelling in the strategic planning horizon is investigating the impact of new or upgraded infrastructure. The work presented in this thesis is unlikely to be suitable for such applications

in its current form. This is because the machine learning models are built to make predictions using data from existing infrastructure – this was discussed in Sections 7.6.2 and 9.6.2.

Therefore, the work is more suited to the tactical planning horizon where the infrastructure is typically considered fixed. The machine learning-assisted simulation approach would be particularly suitable for supporting decision-making around timetable changes and train regulation policies, which will be discussed in the following two sub-sections.

10.4.2 Timetable analysis

The validation of the machine learning-assisted simulation analysed in Sections 7.5.2 and 9.5.2 demonstrated that the machine learning models could produce realistic variation in run times and mimic signallers' behaviour within the simulation. With the addition of entry times and direct delays being modelled stochastically, the approach to simulation could be used to analyse the performance of timetables. A particular strength of this approach to analysing timetable performance is the ability of the quantile regression machine learning models to sample train travel durations from the tail of the skewed distribution.

10.4.3 Train regulation

Network Rail could also apply the simulation to compare existing signallers' behaviour to train regulation policies. As shown in Chapter 9, the simulation model can run using programmatic train regulation logic. Network Rail could construct a programmatic rule base for train regulation using existing train regulation policy (train regulation policies are not 'open' data, so they were not applied within this thesis). Modelling a timetable scenario with the existing train regulation policies and again using the machine learning models to mimic the existing signallers' actions would provide insight for Network Rail. It would be similar to the approach used by Box (2014), which compared traffic control policies in a road traffic simulation. The results would show Network Rail whether signallers are broadly following existing policies or are under- or out-performing the policies.

The simulation model could also be used by Network Rail to analyse changes to train regulation policy. Again, this would be achieved by modelling a timetable scenario once with the machine learning models to represent existing signaller behaviour and then again with a programmatic signalling strategy to represent the updated train regulation policy. The existing behaviour could then be compared to the proposed changes in the metrics generated by the simulation. Simulation modelling also allows for analysis over a range of different traffic scenarios, such as modelling a scenario where a significant

number of trains arrive in the area with extreme delays. This allows for alternative train regulation strategies to be compared under various conditions.

One further application around the area of train regulation would be in the scenario where a new or updated timetable is due to be introduced. As already discussed, the machine learning-assisted simulation can be used to analyse the performance of the new timetable under existing signalling practices. However, analysis of the result may be used to identify whether there is any information or training that would be beneficial to communicate to signallers before introducing the new timetable.

10.5 Closing

Network Rail, like other infrastructure managers, generates and collects vast quantities of data. This research has sought to demonstrate the potential of the data and to exploit the data to improve aspects of simulation modelling. This thesis has presented a novel approach to railway simulation modelling that uses machine learning models to control the movements of trains and take signalling decisions within the simulation. Variation in train movements was achieved using quantile regression, and signalling decisions were modelled using probabilistic classification. The work demonstrated that the approach to simulation was capable of modelling realistic railway performance, which had the benefit of using historical data to determine the dynamics of the simulation rather than theoretical or idealised behaviours. This chapter has laid out several options for further work, both concerning the further development of the machine learning and simulation models.

Appendix A

Data sources

A.1 Introduction

This appendix provides a more detailed overview of the data sources used in this thesis. Brief descriptions of the data sources were provided in Section 5.3. Some information in Section 5.3 will be repeated so that this appendix can be read without referring to the main body. However, the descriptions of the data sources are not comprehensive and only the information relevant to this thesis has been included[1]. This appendix may interest researchers working in the railway industry or those wanting to understand the construction of the datasets, which are discussed further in Appendix B.

A.2 CORPUS reference data

The CORPUS (Codes for Operations, Retail and Planning - a Unified Solution) computer system provides location reference data. The data are stored in JSON format, and the open data file is updated monthly. The data can translate the different codes representing locations on the network. In particular, the translation required for this research is between STANOX codes (station numbers used in some other data sets) and TIPLOC codes.

Table A.1 describes the fields in the CORPUS data, and Table A.2 shows the relevant fields for the six TIPLOCs in the demonstration area of this thesis.

[1]For additional information, the reader is directed to a website about the open data feeds maintained by people who work with the data: https://wiki.openraildata.com/

Field	Description
STANOX	A station number five digits long
UIC	A five-character numeric code for train services running to/from continental Europe
3ALPHA	A three-letter location code
TIPLOC	A code to represent a TIPLOC
NLC	A six-character national location code (NLC)
NLCDESC	Description of the NLC

Table A.1: The fields in the CORPUS dataset.

STANOX	3ALPHA	TIPLOC	NLCDESC
86248	PTC	PCHESTR	PORTCHESTER
86301	CSA	COSHAM	COSHAM
86302		COSHAMJ	COSHAM JUNCTION
86332	HLS	HILSEA	HILSEA
86333		PCRKJN	PORTCREEK JUNCTION
86334		FRLNGTJ	FARLINGTON JUNCTION
86339	BDH	BDHMPTN	BEDHAMPTON

Table A.2: The CORPUS data for the demonstration area

A.3 SMART reference data

The SMART (Signal Monitoring and Reporting of Trains) computer system aggregates data from train describers. This feed is a file of reference data from the SMART system stored in JSON format, and the open data file is updated monthly. The data provides STANOX codes for locations on the network and the associated movements between blocks that relate to arrivals or departures from that location. The data also provide offset values used to estimate arrival and departure times. The fields in the SMART dataset are shown in Table A.3, and Table A.4 displays the data for the locations in the demonstration area of this thesis. The TD field is omitted from Table A.4 as all the locations are from the same TD area.

Field	Description
TD	A two-character code representing the train describer area
STANME	Abbreviation for the location
STANOX	The station number for the location (also applies to junctions and other infrastructure)
PLATFORM	The platform number – if applicable
FROMBERTH	The berth that the movement is from
TOBERTH	The berth that the movement is to
EVENT	Either A (arrive up), B (depart up), C (arrive down) or D (depart down).
BERTHOFFSET	The difference in seconds between the time that the movement occurs and the event is assumed to occur for the purposes of performance monitoring

Table A.3: The fields in the SMART dataset

STANME	STANOX	PLATFORM	FROMBERTH	TOBERTH	EVENT	BERTHOFFSET
PTCHESTER	86248	1	0310	0308	A	33
PTCHESTER	86248	1	0308	0306	B	-71
PTCHESTER	86248	2	0311	0313	C	39
PTCHESTER	86248	2	0313	0315	D	-70
COSHAM	86301	1	0318	0316	A	39
COSHAM	86301	1	0316	0314	B	-9
COSHAM	86301	2	0319	0321	C	32
COSHAM	86301	2	0321	0323	D	-18
COSHAM JN	86302		0320	0318	B	14
COSHAM JN	86302		0401	0318	B	8
COSHAM JN	86302		0323	0400	D	2
COSHAM JN	86302		0323	0325	D	4
HILSEA	86332	1	0048	0046	A	52
HILSEA	86332	1	0046	0044	B	-31
HILSEA	86332	2	0041	0043	C	28
HILSEA	86332	2	0043	0045	D	-32
PORTCREEK	86333		0044	0320	B	9
PORTCREEK	86333		0044	0042	B	16
PORTCREEK	86333		0325	0041	D	20
PORTCREEK	86333		0039	0041	D	17
FLINGTONJ	86334		0042	0040	B	9
FLINGTONJ	86334		0520	0040	B	20
FLINGTONJ	86334		0400	0040	B	20
FLINGTONJ	86334		0037	0039	D	7
FLINGTONJ	86334		0037	0401	D	7
BEDHAMPTN	86339	1	0027	0029	A	27
BEDHAMPTN	86339	1	0029	0031	B	-51
BEDHAMPTN	86339	2	0034	0032	C	45
BEDHAMPTN	86339	2	0032	0030	D	-10

Table A.4: The SMART data for the demonstration area

```
{"SF_MSG":{"time":"1567901445000","area_id":"ES","address":"2D","msg_type":"SF","data":"06"}}
{"SG_MSG":{"time":"1567901445000","area_id":"WA","address":"97","msg_type":"SG","data":"00000000"}}
{"SF_MSG":{"time":"1567901445000","area_id":"NK","address":"47","msg_type":"SF","data":"FC"}}
{"CC_MSG":{"to":"L344","time":"1567901445000","area_id":"X2","msg_type":"CC","descr":"5D84"}}
{"SF_MSG":{"time":"1567901445000","area_id":"ES","address":"2D","msg_type":"SF","data":"04"}}
{"CC_MSG":{"to":"L344","time":"1567901445000","area_id":"X3","msg_type":"CC","descr":"5D84"}}
{"CT_MSG":{"time":"1567901445000","area_id":"G1","msg_type":"CT","report_time":"0110"}}
{"SG_MSG":{"time":"1567901445000","area_id":"WA","address":"9B","msg_type":"SG","data":"00000000"}}
{"CA_MSG":{"to":"0427","time":"1567901445000","area_id":"HT","msg_type":"CA","from":"0425","descr":"2P77"}}
{"CT_MSG":{"time":"1567901445000","area_id":"R3","msg_type":"CT","report_time":"0110"}}
{"SF_MSG":{"time":"1567901445000","area_id":"HT","address":"19","msg_type":"SF","data":"08"}}
{"SF_MSG":{"time":"1567901445000","area_id":"CA","address":"19","msg_type":"SF","data":"4E"}}
{"SG_MSG":{"time":"1567901445000","area_id":"WA","address":"9F","msg_type":"SG","data":"00000000"}}
{"CT_MSG":{"time":"1567901445000","area_id":"TN","msg_type":"CT","report_time":"0110"}}
{"SF_MSG":{"time":"1567901445000","area_id":"R2","address":"DB","msg_type":"SF","data":"08"}}
{"SF_MSG":{"time":"1567901445000","area_id":"D0","address":"69","msg_type":"SF","data":"02"}}
{"SF_MSG":{"time":"1567901445000","area_id":"X0","address":"65","msg_type":"SF","data":"05"}}
{"CT_MSG":{"time":"1567901445000","area_id":"SX","msg_type":"CT","report_time":"0110"}}
{"SF_MSG":{"time":"1567901445000","area_id":"ES","address":"24","msg_type":"SF","data":"06"}}
```


FIGURE A.1: Sample of TD data in JSON format

A.4 TD C-Class data

The train describer (TD) data feed is available for most areas of the British railway and the data are transmitted in JSON format in real-time. Figure A.1 shows a sample of raw data; all the records are for a single timestamp (1567901445000) but represent various TD areas and both S-Class data and C-Class data messages. Table A.5 lists the possible message types that can occur.

Only the C-Class data are used in this thesis, and they provide low-level detail about the position of trains through a network of berths. Trains are identified by headcodes, and the dataset consists of records each time the front of a train 'steps' into a berth. The timestamps of the movements are given to the nearest second. Signallers may also interpose or cancel headcodes from a berth which will be shown as message types CC and CB, respectively.

Message type	Description
CA	A C-Class message that represents a step from one berth to another of the object in the description (descr) field
CB	A C-Class message that remove the description in the 'from' berth
CC	A C-Class message that Interposes a description into the 'to' berth
CT	A C-Class 'heartbeat' message that is sent periodically from each train describer
SF	A S-Class message that updates the signalling data
SG	A S-Class refresh message that sends the current state of a signal element - sent in a batch for each train describer
SH	A S-Class message that represents the end of the SG refresh messages

TABLE A.5: The message types in the TD data feed

Field	Message type CA	CB	CC	CT	Description
time	Y	Y	Y	Y	Timestamp in milliseconds since 1st January 1970
area_id	Y	Y	Y	Y	Two-letter alphanumeric code represented the TD area that the message originates from
msg_type	Y	Y	Y	Y	Type of message: CA, CB, CC or CT
from	Y	Y			The berth the train is leaving
to	Y		Y		The berth the train is entering
descr	Y	Y	Y		Four-letter alphanumeric code representing the headcode of the train
report_time				Y	Unknown, but unused in this thesis

TABLE A.6: The fields in the C-Class data feed; not all fields are used in every message type with a 'Y' indicating that the field is used for a message type.

Table A.6 describes the fields populated in C-Class messages. An example of a message representing a C-Class step is shown in line nine of Figure A.1. The headcode of the train that the record relates to is held in the `descr` field; in the example, it is 2P77. The record states that train 2P77 stepped from berth 0427 to 0425 in train describer Havant (HT) with a timestamp of 1567901445000. The timestamps in the TD data are in milliseconds and is the time since 1st January 1970 (the UNIX epoch).

One issue specific to the open C-Class messages is that if the message relates to a freight train, then the headcode given in the open data feed is not the same as the one used in the signalling system. The reason for this obfuscation may be to do with some level of security surrounding freight trains and their cargo.

A.5 Timetable data

There are two types of files: a 'full' and an 'update' extract. The full extract contains all timetables, and the update extract includes any changes applied to the full data. The update file will have been created as part of the short-term planning (STP) process. Files for the full timetable are provided weekly, and update files are provided daily.

Timetable data are also released as part of the very short-term planning (VSTP) process. These data are not used as a data source because archived data for the VSTP timetable feed was not readily available. The timetables in the VSTP feed are of low volume, so although the decision not to include these data will mean that some schedules are not included or updated, the impact will be small.

There are two formats for timetable data: JSON and CIF (common interface file). The CIF format is the industry standard and is a plain text file with each record a fixed length of 80 characters and is the format used in this work. A sample of the CIF timetable

```
248677  BSNW909731712101805130000001 PXX1T119211124631204 EMU    100      B            P
248678  BX          SWY
248679  LOELGH    0826 08263         TB
248680  LIELGHSJN           0827H00000000
248681  LIHEDGEND 0831H0832      08320832         T
248682  LIBOTLEY  0835 0836      08350836         T
248683  LIFARENJN           0841H00000000
248684  LIFAREHAM 0843 0844H     084308443        T
248685  LIPCHESTR 0849 0849H     08490849         T
248686  LICOSHAM  0853H0854H     08540854         T
248687  LICOSHAMJ           0856 00000000                    1
248688  LIPCRKJN            0858 00000000
248689  LIHILSEA  0859H0900      09000900         T
248690  LIFRATTNE           0902H00000000   DML
248691  LIFRATTON 0903H0904H     090409043        T
248692  LIPSEA    0907H0909H     090809092        T
248693  LTPHBR    0913 09134     TF
248694  BSNW909741712101805130000001 PXX1T159215124631204 EMU    100      B            P
248695  BX          SWY
248696  LOELGH    0926 09263         TB
248697  LIELGHSJN           0927H00000000
248698  LIHEDGEND 0931H0932      09320932         T
248699  LIBOTLEY  0935 0936      09350936         T
248700  LIFARENJN           0941H00000000
248701  LIFAREHAM 0943 0944H     094309443        T
248702  LIPCHESTR 0949 0949H     09490949         T
248703  LICOSHAM  0953H0954H     09540954         T
248704  LICOSHAMJ           0956 00000000                    5H
248705  LIPCRKJN            1002H00000000
248706  LIHILSEA  1003H1004      10001000         T
248707  LIFRATTNE           1006H00000000   DML
248708  LIFRATTON 1007H1008H     100810083        T
248709  LIPSEA    1011 1012      101110122        T
248710  LTPHBR    1015 10154     TF
```

FIGURE A.2: Sample of timetable data in CIF format

data is shown in Figure A.2. Each line of the timetable data relates to a record type, and the record types used in this work are listed in Table A.7.

Tables A.8, A.9, A.10, A.11 and A.12 list each record type's fields. Each field is allocated a set number of characters in the record, listed in the 'size' column of each table. The descriptions in each table are not necessarily exhaustive: they only give relevant details to this thesis.

There may be several BS records for a single schedule; that is, there may be many BS records with the same Train UID (see Table A.8). This will occur when a timetable is updated, perhaps due to engineering works or special events leading to revisions. An updated version of a timetable is indicated by the third letter of the BS record being 'R' for 'revision'.

The records in Figure A.2 will be used as examples of interpreting the schedule record. The identifier for a schedule is located in the fourth to ninth characters (inclusive) of the BS record. The identifier for the first schedule in Figure A.2 is W90973, and the identifier for the second schedule is W90974. Characters ten to fifteen (inclusive) represent the start date of the schedule, and characters sixteen to twenty-one (inclusive) represent the schedule's end date. The first schedule in Figure A.2 runs from the 10th of December 2017 (171210) to the 13th of May 2018 (180513). The following seven characters are

Record Identity	Record Type	Description
BS	Basic schedule	Contains information about the schedule, such as the start and end dates, the number of days of the week the schedule runs and the power type of the train.
BX	Basic schedule extra details	Some additional information about the schedule, such as the TOC code.
LO	Location origin	Information about the TIPLOC of origin on the train's schedule, such as the departure time.
LI	Location intermediate	Information about each intermediate TIPLOC on the train's schedule, such as the arrival and departure times for TIPLOCs where the train is scheduled to stop, and passing times otherwise.
LT	Location terminate	Information about the terminating TIPLOC in the train's schedule, such as the arrival time.

TABLE A.7: The relevant record types in the schedule data.

either zero or one, depending on which days of the week the timetable is due to run. In Figure A.2, both trains are scheduled to run on Sunday only.

The timetable data also contain the headcode for a train, which is stored in the characters 33 to 36 (inclusive) of a BS record; the headcodes in Figure A.2 are 1T11 and 1T15. However, in the open data, the headcode for freight trains will be left blank.

Field	Size	Description
Record identity	2	Always BS for Basic Schedule records.
Transaction type	1	The type of schedule, either N (new), D (deleted) or R (revised).
Train UID	6	An identifier for the schedule: one letter and five numbers. There may be multiple BS records for one Train UID as a schedule may get deleted or revised.
Date runs from	6	The start date of the schedule in the format YYMMDD.
Date runs to	6	The end date of the schedule in the format YYMMDD.
Days run	7	A seven-bit field representing whether a train runs on each day of the week. If a bit is set to 1 then the train runs, 0 otherwise. The first bit represents Monday and the seventh is Sunday.
Bank holiday running	1	If it is set to X, then it does not run on Bank Holiday Mondays.
Train status	1	A one-digit code representing the status of the schedule, either F (freight train in the LTP), P (passenger train in LTP), 1 (passenger train in STP) or 2 (freight train in STP)
Train category	2	Two-digit code representing a category of train. Examples include XX (express passenger), OO (ordinary passenter), EE (empty coaching stock), DD (departmental).
Train identity	4	Headcode of the train used as the signalling ID. In the open data feed this will be blank for freight trains.
Course indicator	1	Not used in this work
Train service code	8	Not used in this work
Portion ID	1	Not used in this work
Power type	3	Up to three digits representing the power type of the train. Examples include D (diesel), E (electric), EMU (electric multiple unit)
Timing load	4	A code representing the traction type of the train
Speed	3	Not used in this work
Operating characteristics	6	Not used in this work
Seating class	1	Not used in this work
Sleepers	1	Not used in this work
Reservations	1	Not used in this work
Connection indicator	1	Not used in this work
Catering code	4	Not used in this work
Service branding	4	Not used in this work
Spare	1	Not used in this work
STP indicator	1	One of C (STP cancellation of permanent schedule), N (new STP schedule), O (STP revision of permenant schedule) or P (permenant schedule)

Table A.8: The fields in the Basic Schedule records, only those used in this research are described

Field	Size	Description
Record identity	2	Always BX for this record type
Traction class	4	Not used in this work
UIC code	5	Not used in this work
ATOC code	2	Two-character code representing the train operating company
Applicable timetable code	1	Not used in this work
Reserved field	8	Not used in this work
Reserved field	1	Not used in this work
Spare	57	Not used in this work

TABLE A.9: The fields in the Basic Schedule Extra Details records, only those used in this research are described

Field	Size	Description
Record identity	2	Always LO for this record type
Location	8	TIPLOC of origin
Scheduled departure	5	Scheduled departure time in the WTT in format hhmm, with an optional suffix of 'H' to represent a half-minute.
Public departure	4	The departure time in the public timetable
Platform	3	The scheduled platform of departure
Line	3	Not used in this work
Engineering allowance	2	Not used in this work
Pathing allowance	2	Not used in this work
Activity	12	A code representing any activity of the train at the TIPLOC
Performance allowance	2	Not used in this work
Reserved field	3	Not used in this work
Spare	57	Not used in this work

TABLE A.10: The fields in the Location Origin records, only those used in this research are described

Field	Size	Description
Record identity	2	Always LI for this record type
Location	8	TIPLOC of the intermediate location
Scheduled arrival	5	Scheduled arrival time in the WTT in format hhmm, with an optional suffix of 'H' to represent a half-minute.
Scheduled departure	5	Scheduled departure time in the WTT in format hhmm, with an optional suffix of 'H' to represent a half-minute
Scheduled pass	5	Scheduled passing time in the WTT in format hhmm, with an optional suffix of 'H' to represent a half-minute. An LI record must have either this field or the 'Scheduled arrival' and 'Scheduled departure' fields.
Public arrival	4	Scheduled arrival time in the public timetable in format hhmm. Blank if the train is scheduled to pass the TIPLOC.
Public departure	4	Scheduled departure time in the public timetable in format hhmm. Blank if the train is scheduled to pass the TIPLOC.
Platform	3	The platform number
Line	3	Not used in this work
Path	3	Not used in this work
Activity	12	A code representing any activity of the train at the TIPLOC
Engineering allowance	2	Not used in this work
Pathing allowance	2	Not used in this work
Performance allowance	2	Not used in this work
Reserved field	5	Not used in this work
Spare	15	Not used in this work

TABLE A.11: The fields in the Location Intermediate records, only those used in this research are described

Field	**Size**	**Description**
Record identity	2	Always LT for this record type
Location	8	TIPLOC of the intermediate location
Scheduled arrival	5	Scheduled arrival time in the WTT in format hhmm, with an optional suffix of 'H' to represent a half-minute.
Public arrival	4	Scheduled arrival time in the public timetable in format hhmm. Blank if the train is scheduled to pass the TIPLOC.
Platform	3	The platform number
Path	3	Not used in this work
Activity	12	Not used in this work
Reserved field	3	Not used in this work
Spare	40	Not used in this work

TABLE A.12: The fields in the Location Terminate records, only those used in this research are described

```
1 {"header":{"msg_type":"0001","source_dev_id":"","user_id":"","original_data_source":
  "TSIA","msg_queue_timestamp":"1523053318000","source_system_id":"TRUST"},
2 "body":{"schedule_source":"C","train_file_address":null,"schedule_end_date":"2018-05-19",
  "train_id":"21604VC107","tp_origin_timestamp":"2018-04-07","creation_timestamp":
  "1523056918000","tp_origin_stanox":"","origin_dep_timestamp":"1523064060000",
  "train_service_code":"57622670","toc_id":"00","d1266_record_number":"00000",
  "train_call_type":"AUTOMATIC","train_uid":"H34622","train_call_mode":"NORMAL",
  "schedule_type":"O","sched_origin_stanox":"21186","schedule_wtt_id":"604VC",
  "schedule_start_date":"2017-12-11"}},
3 {"header":{"msg_type":"0003","source_dev_id":"","user_id":"","original_data_source":
  "SMART","msg_queue_timestamp":"1523053319000","source_system_id":"TRUST"},
4 "body":{"event_type":"ARRIVAL","gbtt_timestamp":"1523055780000","original_loc_stanox":"",
  "planned_timestamp":"1523055780000","timetable_variation":"21","original_loc_timestamp":
  "","current_train_id":"","delay_monitoring_point":"true","next_report_run_time":"",
  "reporting_stanox":"42140","actual_timestamp":"1523057040000","correction_ind":"false",
  "event_source":"AUTOMATIC","train_file_address":null,"platform":" 3","division_code":"29"
  ,"train_terminated":"true","train_id":"701T492706","offroute_ind":"false",
  "variation_status":"LATE","train_service_code":"22209000","toc_id":"29","loc_stanox":
  "42140","auto_expected":"true","direction_ind":"DOWN","route":"0","planned_event_type":
  "DESTINATION","next_report_stanox":"","line_ind":""}}
```


FIGURE A.3: Sample of TRUST data in JSON format

A.6 TRUST train movement data

The train movement data are in JSON format and provide records from the TRUST (Train Running Under System TOPS) computer system. TRUST takes the schedule of each train and records the actual departure, arrival and passing times at TIPLOC locations on the schedule. Figure A.3 shows a sample of the TRUST data. There are eight different types of messages in the TRUST data, but this thesis only uses message types `0001` and `0003`, which are activation and movement messages, respectively. As these data are only used to match the TD C-Class data to the timetable data, only the essential elements of each record are briefly described.

Activation message

An activation message is created each time a train is due to start its schedule. The body of an activation message contains two fields crucial to this research: the `train_id` and the `train_uid`. The `train_id` field is ten characters long, and the third to sixth characters (inclusive) represent the train's headcode used within the open data feeds. This means that if the record relates to a freight train, the headcode in the TRUST data will match that given in the C-Class data, even though neither will match the headcode in the signalling system. However, the `train_id` is only guaranteed to be unique once in a calendar month.

The `train_uid` field contains six characters; the field is the same as the identifier for the schedule (discussed in Section A.5).

Movement message

The movement messages are logged each time a train arrives, passes or departs a TIPLOC. A `train_id` field appears in each movement message, which will match the associated activation message. The `loc_stanox` field contains the STANOX of the location where the movement message is recorded.

Appendix B

Dataset construction

B.1 Introduction

This appendix presents some further detail on how the datasets were constructed. The reader may need to refer to Appendix A, which provides some field names referenced in this appendix. This appendix does not detail every aspect of the data processing; instead, only the most critical or complicated features are described. Some simplifications have also been made in the descriptions of the processes.

B.2 Infrastructure dataset

B.2.1 Berths and steps

The process for creating the dataset for infrastructure data involved first filtering the TD data on the CA, CB and CC messages for the modelled TD area (see Section A.4 for descriptions of the TD messages and their fields). It is then straightforward to identify berths and valid steps in the area. The berths are all the values in the `from` and `to` fields in the C-Class data, and the steps are all the pairs of berths that appear in the `from` and `to` fields.

In the list of berths, some will appear that belong to neighbouring TD areas. This is due to steps over the boundary of a TD area. These berths are simple to identify as they will have at least one character as a prefix for the berth; however, if multiple TD areas are to be modelled, some manual verification of the boundary areas may be required.

There are also some examples of erroneous steps appearing in the C-Class data. The reason for this is unknown but assumed to be a latency issue with the data feed. Therefore, it is necessary to have a reasonably large amount of TD data to count the times

each berth and step appears. Steps with very low counts (for example, fewer than ten records) that appear over a year or more can be excluded from the dataset.

B.2.2 Locations of TIPLOCs

The CORPUS and SMART reference data can be joined on the STANOX code to identify the berth locations of the TIPLOCs. The CORPUS data provides the TIPLOC (see Table A.1), and the `FROMBERTH`, `TOBERTH`, `EVENT` and `OFFSET` fields from the SMART data (see Table A.3) can be used to identify the berth locations of the TIPLOCs.

B.2.3 Conflict locations

Identifying the conflict location pair requires identifying mutually exclusive steps with the following process:

1. Filter the raw data on CA messages for the TD area of interest
2. For each unique pair of steps, s_i and s_j, identify the minimum time difference between records relating to each step
3. Any pairs of steps that have a minimum time difference of zero can be discarded
4. The remaining pairs of steps will need to be reviewed to determine whether they all relate to conflict locations

The fourth step in the process is not automated and benefits from some expert knowledge of the TD area. For example, some pairs with a large time difference may be because some steps are not frequent – even when considering years' worth of data.

B.3 Train movements dataset

B.3.1 Matching C-Class data to `train_uid` field

A critical part of creating the train movements dataset is the ability to match C-Class data to a timetable. A train is identified in the C-Class data in the `descr` field, which provides a headcode for the train. However, if the train is a freight train, then this headcode in the open data feed does not reflect the headcode used by the signallers. Similarly, the headcode in the timetable data is blank for freight trains in the open data feeds. The headcode is also not guaranteed to be unique across all TD areas. Therefore, matching the data on headcode is not feasible for freight trains and would not necessarily be trivial for other trains.

Instead, this work matched the two datasets using intermediate data sources: the TRUST and SMART data. The first step is to match all possible C-Class records to a `train_uid` from the TRUST data. The `train_uid` is also used in the schedule data, so it can be used subsequently to match the C-Class data to timetable data (outlined in Section B.3.2). This process has multiple stages and is intended to be carried out on a single TD area:

1. **Filter raw data on date range:** The C-Class, TRUST, and schedule data were filtered on records collected within a given date range. No more than a month's worth of data can be used to complete this matching process, as the TRUST `train_id` values are only unique within a month. Further, at least one 'full' schedule file must be available in the date range; typically, a 'full' schedule file is available once a week. Therefore, the date range must be at least a week but less than a month. For this work, a date range of a week was used in most cases, apart from when there was not a 'full' schedule file available in a week.

2. **Discard irrelevant data:** Not all the data are relevant, and some can be discarded:

   (a) TRUST data: Only the activation and movement messages are required – the rest can be discarded.

   (b) C-Class and SMART data: Only the messages relating to the TD area of interest need to be kept.

   (c) Schedule data: Only those schedules that pass through the TD area need to be kept (identified by the TIPLOCs in the LO, LI and LT records – see Section A.5 for descriptions of these records).

3. **Match TRUST movement messages to activation messages:** Use the `train_id` field in the TRUST movement messages to match them to the activation message with the same `train_id`. Attach the `train_uid` field from the activation to the movement message.

4. **Match C-Class movements to TRUST movements:** Not all C-Class messages will be matched in this step; they will be limited to those related to the berths where the TIPLOCs are located, as TRUST movement messages only appear for these berths. The headcodes in the C-Class data can be matched to the third to sixth characters (inclusive) of the TRUST `train_id`. The timestamp in the TRUST data should be the timestamp in the C-Class data plus the berth offset from the SMART data for the appropriate berth. A small margin of 30 seconds on either side of the timestamps was used in case they did not match exactly. Further, the STANOX of the berth that the C-Class record is stepping into should match the value of the `loc_stanox` field in the TRUST message.

5. **Add `train_uid` field onto matched C-Class movements:** Those C-Class movements that were matched in the previous step can now have the `train_uid` field that is stored on the movement message appended onto the C-Class record.
6. **Sort C-Class data:** Sort all the C-Class movement data by headcode, then timestamp ascending.
7. **Add `train_uid` field onto other C-Class movements:** See the flowchart in Figure B.1 for this process.
8. **Sort C-Class data:** Sort all the C-Class movement data by headcode, then timestamp *descending*.
9. **Add `train_uid` field onto other C-Class movements:** Repeat step 7, apart from the decision in the dashed diamond in Figure B.1 will match the `from` berth in the `current_record` to the `to` berth in the `next_record`. This step is necessary as the first record in the C-Class data for a train is not guaranteed to relate to a TIPLOC.

Not all C-Class records will be matched successfully to a `train_uid` value. For example, some records will relate to engineering work or blockages. Others may be associated with schedules from the very short-term plan (VSTP), which this research has not included.

B.3.2 Matching C-Class data to timetables

Having matched a C-Class record to its associated `train_uid` field, data contained in a timetable can then be added to the C-Class data. Matching the data from the BS records from a timetable (see Section A.7 for a description of the record types in the schedule data) is straightforward once the `train_uid` has been added to each C-Class data. The BS records can be matched on the `train_uid` field, and data from the BS records (such as the power type, status and category) can be added to each C-Class record.

Each BS record will be associated with precisely one LO record and one LT record; therefore, the origin and destination TIPLOCs can also be added to each C-Class record that has a `train_uid` associated with it. Note that the origin and destination TIPLOCs may be outside of the TD area of interest.

For each timetable in the dataset, the intermediate LI records in the schedule can then be matched to the C-Class records that represent the same TIPLOC. Each C-Class record with a match will add to its record whether the train is scheduled to stop or pass the TIPLOC, which can be identified by whether there are arrival and departure times for the TIPLOC (see Table A.10 for descriptions of the fields). The scheduled passing or arrival and departure times can also be recorded so it will be possible to identify the time difference to log whether the train is running early or late.

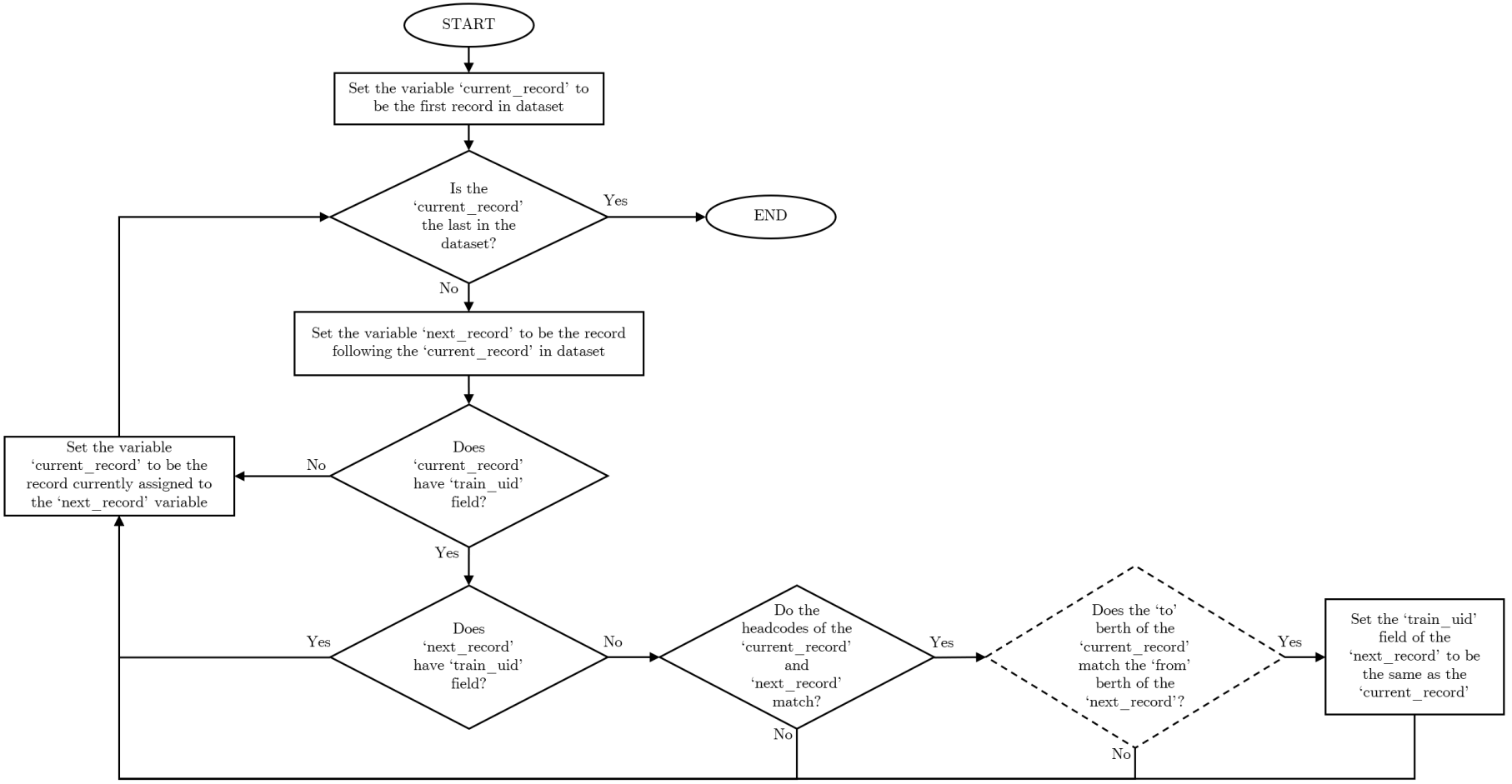

FIGURE B.1: Flowchart demonstrating how to match the `train_uid` value to C-Class records not associated with a TIPLOC. This process assumes that the C-Class dataset is sorted by headcode and timestamp ascending. The query in the dashed diamond will change if the data are sorted by headcode and timestamp *descending*

B.3.3 Train types

There is not a field in any of the datasets neatly classifies a train as ECS, freight or passenger. Instead, if the `status` field in the schedule data is either 'F' or '2', the record is assumed to be a freight train. If the `category` field in the schedule data is 'EE', 'EL', or 'ES', then the record is assumed to be ECS. If the `category` field in the schedule data is 'OO' or 'XX', then the record is assumed to be a passenger train. If this work was to be extended to other TD areas, additional `status` and `category` values may need to be considered.

B.3.4 Restricted movements

A restricted movement has been defined in this thesis as one where a train enters a berth, and the status of the signal to exit the berth is at its most restrictive aspect. It has been necessary to infer the aspect of a signal based on the locations of trains on the network. Given a berth 'XXXX', there are two scenarios where the signal associated with berth 'XXXX' is assumed to be at its most restrictive aspect. The first is if berth 'YYYY' is occupied and there is a step leading from berth 'XXXX' to berth 'YYYY'; that is, the next berth is occupied. The second is that if berth 'XXXX' is the first berth in a step involved in a conflict pair, and a train passes over the other step in the pair; that is, movement authority out of berth 'XXXX' cannot be given until at least the conflicting step has been cleared.

The aim is for each C-Class movement to identify if the signal of the berth the train is stepping into is at its most restrictive aspect. This identification is carried out on about a day's worth of data at a time to limit the computational requirement. There is typically a gap overnight when no trains are running, so this limitation does not lead to many restricted movements being missed.

In order to identify both types of restricted movement, each C-Class record must log the next berth the train will move into (after its current step) and the time at which this movement will occur. These fields will be called `berth_next` and `timestamp_next`, respectively. The `timestamp_next` field will only be held temporarily and will be dropped after the data has been processed. These fields can be identified following the process in flowchart in Figure B.2.

The two types of restricted movement can then be identified by following the flowcharts in Figures B.3 and B.4. These processes will create two new fields on all the C-Class records. The first is `next_berth_occupied` which will be set to `True` if the next berth the train is due to step into is occupied at the time of the movement. The second is `wait_for_conflict` which is set to `True` if the train is occupying its current berth while another train passes over a conflicting step.

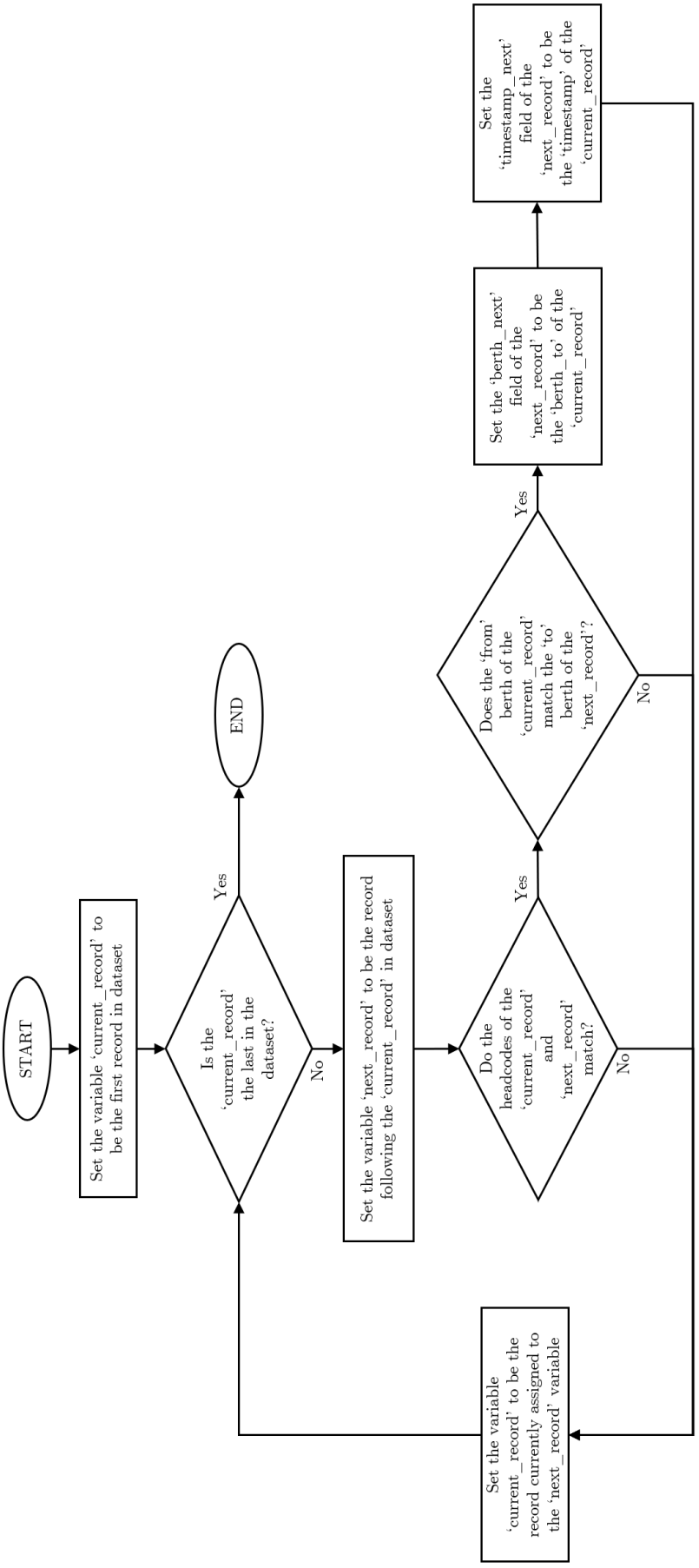

FIGURE B.2: Flowchart for recording the `berth_next` and `timestamp_next` fields in the C-Class data. This process assumes that the C-Class dataset is sorted by headcode and timestamp ascending.

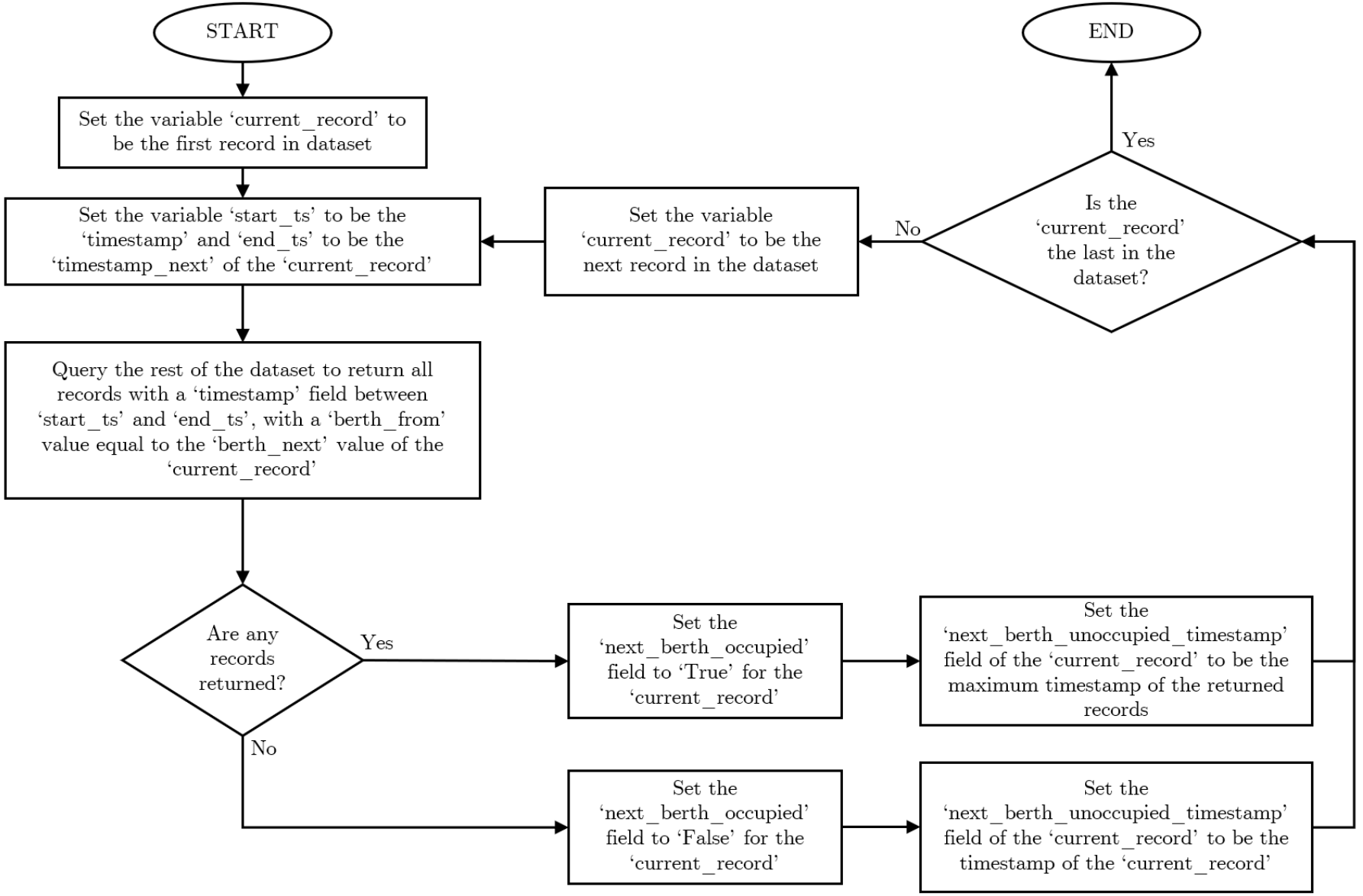

FIGURE B.3: Flowchart for identifying restricted movements where the next berth is occupied.

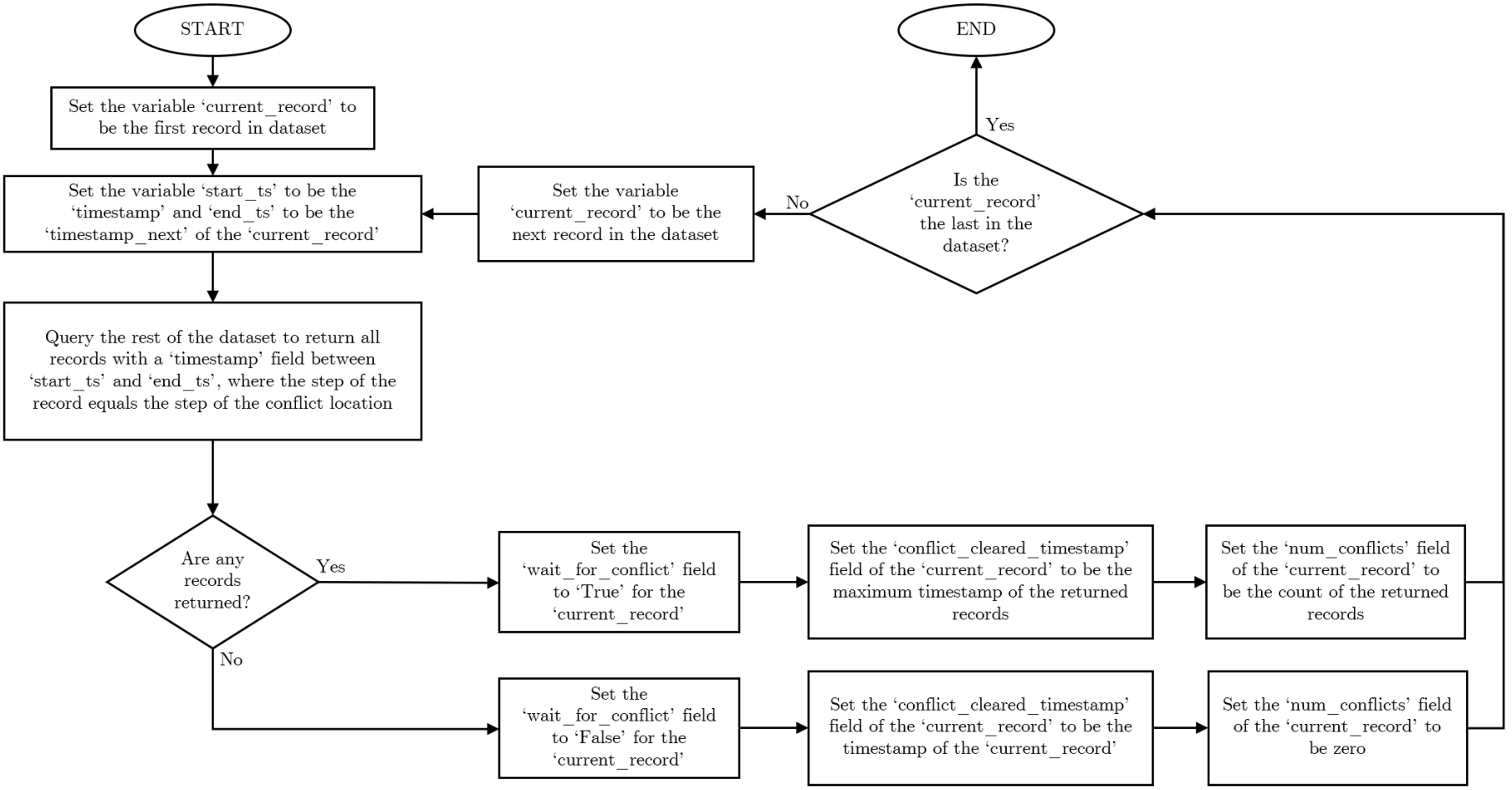

FIGURE B.4: Flowchart for identifying restricted movements where the train has had to wait for a conflict to be cleared

B.4 Train conflicts dataset

The train conflicts dataset is constructed by applying processes to the train movements dataset. The train conflicts dataset consists of pairs of trains approaching conflict locations within five steps, and the main requirement for constructing this dataset is to identify pairs of records.

For each conflict location, it is first necessary to identify those records that represent trains approaching the location. It is also necessary to create additional data fields, such as the number of steps to the conflict location and the timestamp the train stepped across the conflict location. The flowchart in Figure B.5 demonstrates this process for a single conflict location. The process in Figure B.5 is applied for each conflict location, and the records without the `steps_to_conflict` and `timestamp_conflict` fields can be discarded. Note that some records may be identified for multiple conflict locations depending on the route of the train.

Once potential records for each conflict location have been identified, the pairs of conflict locations can be considered. The flowchart in Figure B.6 shows how pairs of records are identified that represent two trains moving towards conflicting locations. It is necessary to set one conflict location as 'A' and the other as 'B'; the labelling must be consistent throughout the process. The process in Figure B.6 then needs to be repeated, with the initial records being selected from dataset 'B' and dataset 'A' being queried. The logical steps where datasets 'A' and 'B' are interchanged are highlighted with a dashed border. There will be duplicate records logged in the resulting 'conflicts' dataset, which can be dropped to leave a set of unique records.

The resulting dataset of conflicts are formed of pairs of trains, each approaching conflicting locations. The `train_A_has_priority` field assigned in Figure B.6 becomes the target value, and the `timestamp_conflict` field can be dropped from the dataset.

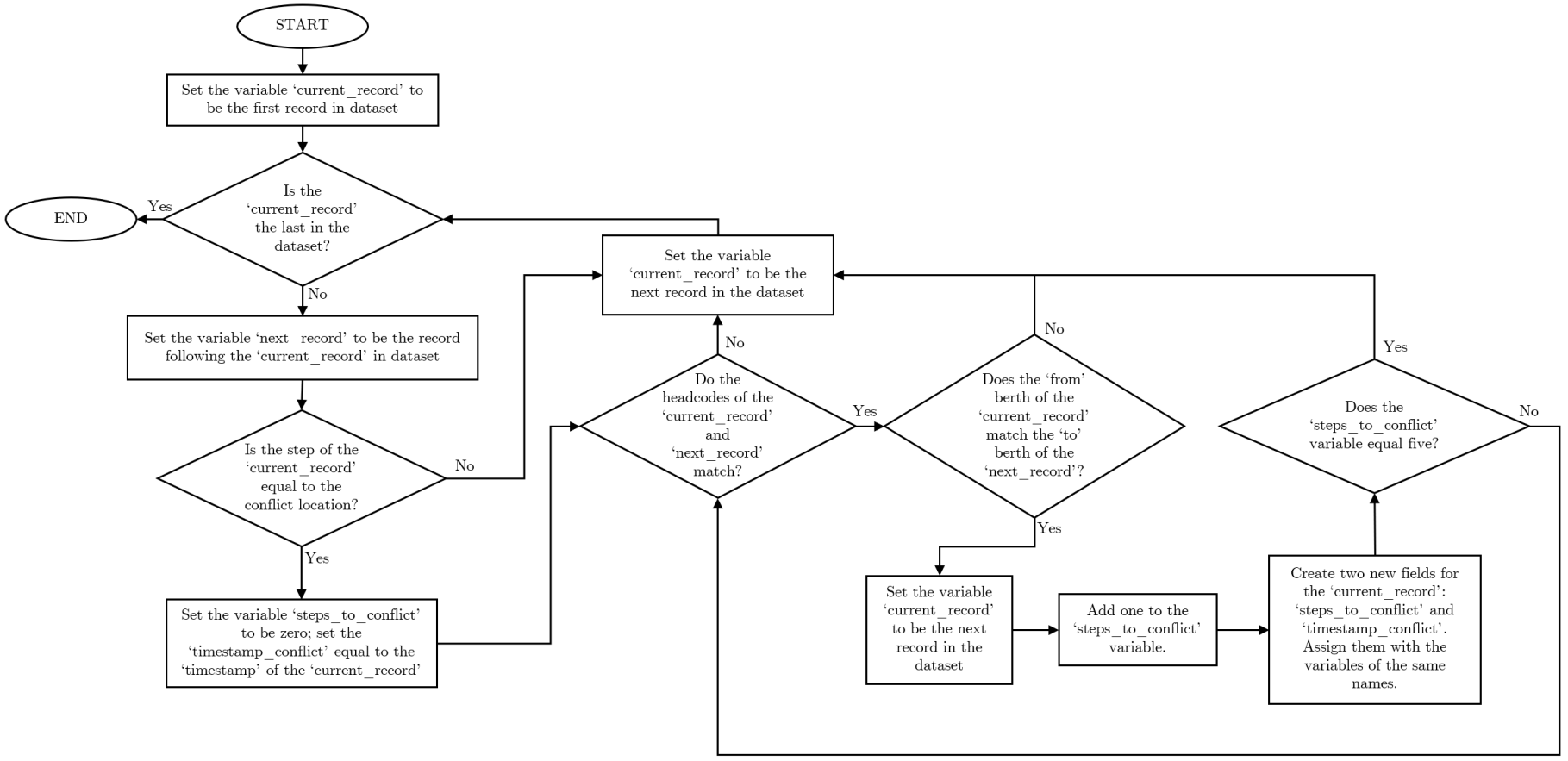

FIGURE B.5: Flowchart for recording the `steps_to_conflict` and `timestamp_conflict` fields on records of the train movements dataset. This process is applied for a single conflict location on a copy of the train movements dataset and assumes that the data are sorted by headcode and timestamp descending.

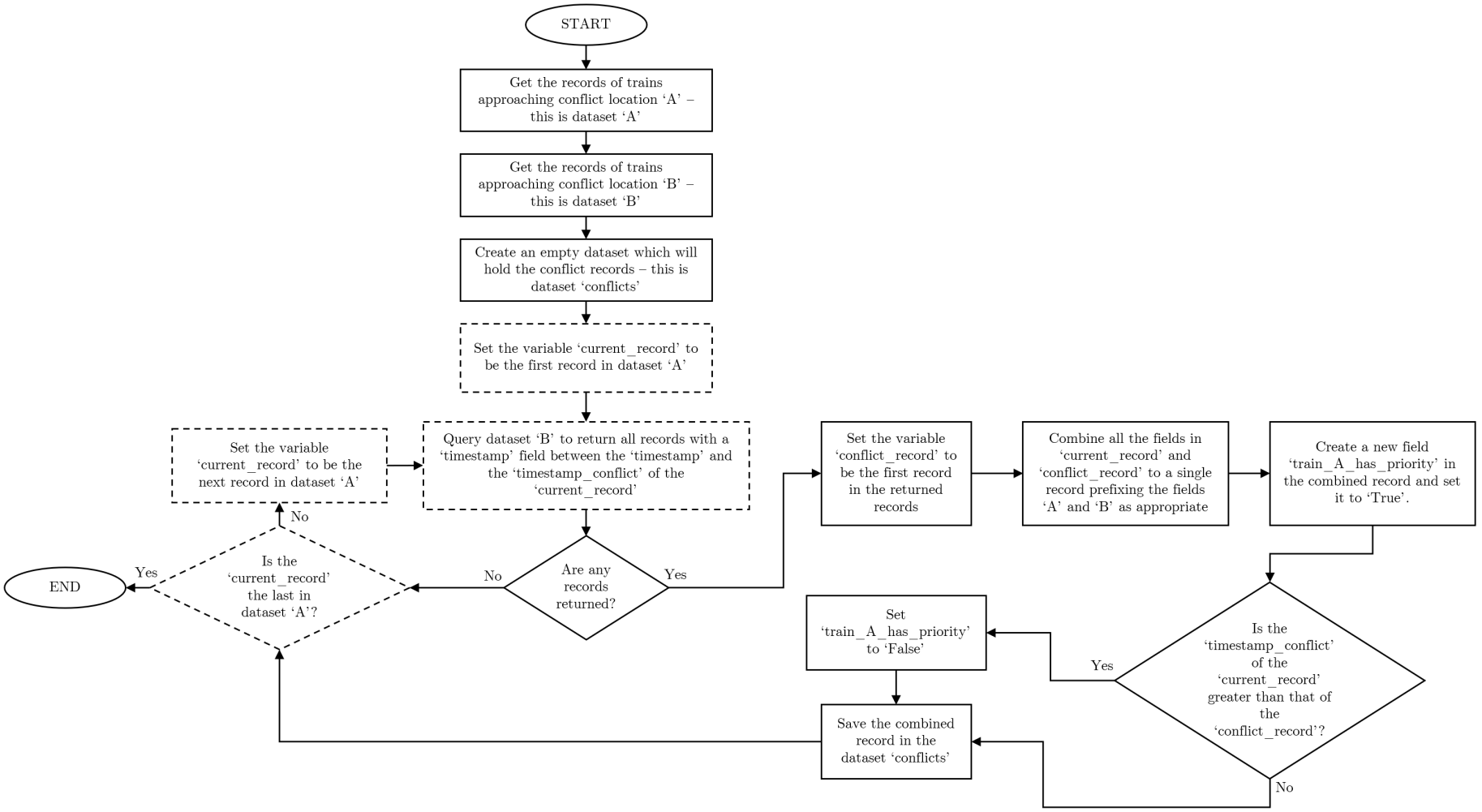

FIGURE B.6: Flowchart for identifying pairs of conflicting trains. This process is applied for a conflict location pair on the datasets generated in Figure B.5

Bibliography

M. Abril, F. Barber, L. Ingolotti, M. A. Salido, P. Tormos, and A. Lova. An assessment of railway capacity. *Transportation Research Part E: Logistics and Transportation Review*, 44(5):774–806, 2008.

T. M. Abuhay, A. L. Mamuye, S. Robinson, and S. V. Kovalchuk. Why machine learning integrated patient flow simulation? In M. Fakhimi, D. Robertson, and T. Boness (eds), *Proceedings of SW21 The OR Society Simulation Workshop*, pages 375–384. Operational Research Society, 2021.

N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

E. Andersson, A. Peterson, and J. T. Krasemann. Quantifying railway timetable robustness in critical points. *Journal of Rail Transport Planning & Management*, 3(3): 95–110, 2013.

W. Barbour, C. Samal, S. Kuppa, A. Dubey, and D. B. Work. On the data-driven prediction of arrival times for freight trains on U.S. railroads. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2289–2296, 2018.

L. Barson, W. Fung, and A. Toossi. Infrastructure/train borne measurements in support of railway system performance modelling. *INCOSE International Symposium*, 26(1): 1261–1275, 2016.

BBC News. Rail freight boosted by HGV driver shortage and climate change. Web Page, Visited on: 10/08/2022, 2021.

C. Bergmeir and J. M. Benítez. On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191:192–213, 2012.

N. Bešinović. Resilience in railway transport systems: a literature review and research agenda. *Transport Reviews*, 40(4):1–22, 2020.

N. Bešinović, L. De Donato, F. Flammini, R. M. P. Goverde, Z. Lin, R. Liu, S. Marrone, R. Nardone, T. Tang, and V. Vittorini. Artificial intelligence in railway transport: Taxonomy, regulations, and applications. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):14011–14024, 2022.

N. Bešinović and R. M. P. Goverde. Capacity assessment in railway networks. In R. Borndörfer, T. Klug, L. Lamorgese, C. Mannino, M. Reuther, and T. Schlechte (eds), *Handbook of Optimization in the Railway Industry. International Series in Operations Research & Management Science*, volume 268, pages 25–45. Springer, Cham., 2018.

N. Bešinović, E. Quaglietta, and R. M. P. Goverde. A simulation-based optimization approach for the calibration of dynamic train speed profiles. *Journal of Rail Transport Planning & Management*, 3(4):126–136, 2013.

C. M. Bishop. Mixture density networks. Technical report, Aston University, Birmingham, 1994.

M. Botte and L. D'Acierno. Dispatching and rescheduling tasks and their interactions with travel demand and the energy domain: Models and algorithms. *Urban Rail Transit*, 4(4):163–197, 2018.

S. Box. Supervised learning from human performance at the computationally hard problem of optimal traffic signal control on a network of junctions. *Royal Society Open Science*, 1(4):140211, 2014.

A. Bradford, B. Hyland, A. Toossi, and A. Velavs. Brighton mainline timetable optimisation study. Technical report, RSSB, London, 2016.

H. Bransby and W. Barter. West Coast main line lessons learnt report. Technical report, The Office of Rail and Road, London, 2010.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

S. Buchmueller, U. Weidmann, and A. Nash. Development of a dwell time calculation model for timetable planning. In J. Allan, E. Arias, C.A. Brebbia, C. Goodman, A.F. Rumsey, G. Sciutto, and N. Tomii (eds), *Computers in Railways XI*, volume 103, pages 525–534, 2008.

V. Cacchiani and P. Toth. Robust train timetabling. In R. Borndörfer, T. Klug, L. Lamorgese, C. Mannino, M. Reuther, and T. Schlechte (eds), *Handbook of Optimization in the Railway Industry*, pages 93–115. Springer International Publishing, Cham, Switzerland, 2018.

M. Carey and I. Crawford. Scheduling trains on a network of busy complex stations. *Transportation Research Part B: Methodological*, 41(2):159–178, 2007.

J. S. Carson. Model verification and validation. In *Proceedings of the Winter Simulation Conference*, volume 1, pages 52–58, 2002.

V. Cerqueira, L. Torgo, and I. Mozetic. Evaluating time series forecasting models: An empirical study on performance estimation methods. *Machine Learning*, 109(11): 1997–2028, 2019.

F. Cerreto, B. F. Nielsen, O. Nielsen, and S. Harrod. Application of data clustering to railway delay pattern recognition. *Journal of Advanced Transportation*, 2018:1–18, 2018.

K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, 2014.

F. Chollet. Deep Learning with Python. Manning Publications Co., USA, 1st edition, 2017.

Y. Cui, U. Martin, and J. Liang. PULSim: User-based adaptable simulation tool for railway planning and operations. *Journal of Advanced Transportation*, 2018:11, 2018.

Y. Cui, U. Martin, and W. Zhao. Calibration of disturbance parameters in railway operational simulation based on reinforcement learning. *Journal of Rail Transport Planning & Management*, 6(1):1–12, 2016.

B. Cule, B. Goethals, S. Tassenoy, and S. Verboven. Mining train delays. In J. Gama, E. Bradley, and J. Hollmén (eds), *Advances in Intelligent Data Analysis X*, pages 113–124. Springer Berlin Heidelberg, 2011.

G. Dagkakis and C. Heavey. A review of open source discrete event simulation software for operations research. *Journal of Simulation*, 10(3):193–206, 2016.

A. D'Ariano, F. Corman, and D. Pacciarelli. Rescheduling. In I. A. Hansen and J. Pachl (eds), *Railway Timetabling & Operations*, Chapter 3. Eurail Press, Hamburg, Germany, 2014.

A. D'Ariano and M. Pranzo. An advanced real-time train dispatching system for minimizing the propagation of delays in a dispatching area under severe disturbances. *Networks and Spatial Economics*, 9(1):63–84, 2009.

S. de Fabris, G. Longo, and G. Medeossi. Automated analysis of train event recorder data to improve micro-simulation models. In J. Allan, E. Arias, C.A. Brebbia, C. Goodman, A. F. Rumsey, G. Sciutto, and N. Tomii (eds), *Computers in Railways XI*, pages 575–583, 2008.

S. de Fabris, G. Medeossi, and G. Montanaro. TRENISSIMO: Improving the microscopic simulation of railway networks. In G. Passerini, J. M Mera, N. Tomii, and N.P. Tzieropoulos (eds), *Computers in Railways XVI*, pages 199–211, 2018.

Delay Attribution Board. Delay attribution principles and rules. Statement of good practice, Delay Attribution Board, London, UK, 2019.

Z. Deng, B. Wang, H. Guo, C. Chai, Y. Wang, and Z. Zhu. Unified quantile regression deep neural network with time-cognition for probabilistic residential load forecasting. *Complexity*, 2020:9147545, 2020.

Department for Transport. Decarbonising transport - a better, greener Britain. Guidance document, Department for Transport, London, 2021a.

Department for Transport. Great British Railways, the Williams-Shapps plan for rail. Policy paper, Department for Transport, London, 2021b.

H. Douglas, P. Weston, D. Kirkwood, S. Hillmansen, and C. Roberts. Method for validating the train motion equations used for passenger rail vehicle simulation. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 231(4):455–469, 2017.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

J. M. Durán. What is a simulation model? *Minds and Machines*, 30(3):301–323, 2020.

S. Dündar and İ. Şahin. Train re-scheduling with genetic algorithms and artificial neural networks for single-track railways. *Transportation Research Part C: Emerging Technologies*, 27:1–15, 2013.

M. Elbattah and O. Molloy. Analytics using machine learning-guided simulations with application to healthcare scenarios. In *Analytics and Knowledge Management*, Chapter 10. Auerbach Publications, New York, 1st edition, 2018.

J. M. Epstein. Why model? *Journal of Artificial Societies and Social Simulation*, 11 (4):12, 2008.

M. Fowler. UML distilled : A brief guide to the standard object modeling language. Addison-Wesley, Boston, 3rd edition, 2004.

J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

D. Gartner and R. Padman. Machine learning for healthcare behavioural OR: Addressing waiting time perceptions in emergency care. *Journal of the Operational Research Society*, 71(7):1087–1101, 2020.

R. M. P. Goverde, F. Corman, and A. D'Ariano. Railway line capacity consumption of different railway signalling systems under scheduled and disturbed conditions. *Journal of Rail Transport Planning & Management*, 3(3):78–94, 2013.

A. Greasley and J. S. Edwards. Enhancing discrete-event simulation with big data analytics: A review. *Journal of the Operational Research Society*, 72(2):247–267, 2021.

F. Gressmann, F. J. Király, B. Mateen, and H. Oberhauser. Probabilistic supervised learning. *Machine Learning*, 2018.

T. Gröger. Timetable simulation by sophisticated conflict resolution between train paths. In J. Allan, G. Sciutto, and S. Sone (eds), *Computers in Railways IX*, pages 573 – 582. WIT Press, 2004.

A. Géron. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Inc., Sebastopol, Canada, 2nd edition. ISBN 9781492032649, 2019.

H. Haramina, I. Talan, and B. Mihaljević. Improvement of suburban railway services by infrastructure and timetable modifications based on simulation modelling. *Transport Problems*, 13(3):15–27, 2018.

T. Hastie, R. Tibshirani, and J. Friedman. Additive models, trees, and related methods, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, pages 295–336. Springer, New York, NY, USA, 2009a.

T. Hastie, R. Tibshirani, and J. Friedman. Overview of supervised learning, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, pages 9–41. Springer, New York, NY, USA, 2009b.

T. Hastie, R. Tibshirani, and J. Friedman. Support vector machines and flexible discriminants, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, pages 417–458. Springer, New York, NY, USA, 2009c.

T. He, M. Heidemeyer, F. Ban, A. Cherkasov, and M. Ester. SimBoost: a read-across approach for predicting drug–target binding affinities using gradient boosting machines. *Journal of Cheminformatics*, 9(1):24, 2017.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 1997.

M. Hofman, L. Madsen, J. Groth, J. Clausen, and J. Larsen. Robustness and recovery in train scheduling - a simulation study from DSB S-tog a/s. In *6th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS'06)*, volume 5 of *OpenAccess Series in Informatics (OASIcs)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2006.

D. Holgate and P. Lawrence. The relative performance benefits of fixed and moving block. In *World Congress on Railway Research (WCRR)*, 1997.

P. Huang, C. Wen, L. Fu, J. Lessan, C. Jiang, Q. Peng, and X. Xu. Modeling train operation as sequences: A study of delay prediction with operation and weather data. *Transportation Research Part E: Logistics and Transportation Review*, 141, 2020.

X. Huang, L. Shi, and J. A. K. Suykens. Support vector machine classifier with pinball loss. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(5):984–997, 2014.

B. Hyland. Making the case for a whole system strategic approach to reliability improvement: Modelling reliability study. Technical report, RSSB, London, 2012.

B. Hyland, N. Best, A. Toossi, and A. Bradford. UK railway system performance – gaining insight through systematic analysis and modelling. *INCOSE International Symposium*, 26(1):1246–1260, 2016.

L. W. Jensen, A. Landex, O. A. Nielsen, L. G. Kroon, and M. Schmidt. Strategic assessment of capacity consumption in railway networks: Framework and model. *Transportation Research Part C: Emerging Technologies*, 74:126–149, 2017.

T. Jiang, J. L. Gradus, and A. J. Rosellini. Supervised machine learning: A brief primer. *Behavior Therapy*, 51(5):675–687, 2020.

Z. Jiang, J. Gu, Y. Han, W. Fan, and J. Chen. Modeling actual dwell time for rail transit using data analytics and support vector regression. *Journal of Transportation Engineering, Part A: Systems*, 144(11), 2018.

P. Kecman and R. M. P. Goverde. Train delay prediction. In I. A. Hansen and J. Pachl (eds), *Railway Timetabling & Operations*, Chapter 12, pages 237–252. Eurail Press, Hamburg, Germany, 2014.

P. Kecman and R. M. P. Goverde. Predictive modelling of running and dwell times in railway traffic. *Public Transport*, 7(3):295–319, 2015.

M. Khadem Sameni, M. Dingler, J. M. Preston, and C. P.L. Barkan. Profit-generating capacity for a freight railroad. In *Transportation Research Board 90th Annual Meeting*, Washington DC, USA, 2011.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations*, San Diego, 2015.

J Knight. Dataset in support of the southampton doctoral thesis 'machine learning-assisted railway simulation modelling'. Dataset, 2023.

J. Knight, A. Keane, and O. Hovorka. Modelling traffic management decisions using a hybrid machine learning and simulation approach. In *World Congress on Railway Research (WCRR)*, 2022.

R. Koenker and G. Bassett. Regression quantiles. *Econometrica*, 46(1):33–50, 1978.

Q. Kong, T. Siauw, and A. M. Bayen. Object-oriented programming. In *Python Programming and Numerical Methods*, Chapter 7, pages 121–134. Academic Press, 2021.

L. Kroon, G. Maróti, M. R. Helmrich, M. Vromans, and R. Dekker. Stochastic improvement of cyclic railway timetables. *Transportation Research Part B: Methodological*, 42(6):553–570, 2008.

H. Krueger. Parametric modeling in rail capacity planning. In *Association for Computing Machinery 1999 Winter Simulation Conference Proceedings. 'Simulation - A Bridge to the Future'*, volume 2, pages 1194–1200 vol.2, New York, USA, 1999.

R. Krzysztofowicz. Transformation and normalization of variates with specified distributions. *Journal of Hydrology*, 197(1):286–292, 1997.

S. Kubosawa, T. Onishi, M. Sakahara, and Y. Tsuruoka. Railway operation rescheduling system via dynamic simulation and reinforcement learning. Report, 2022.

M. Kuhn and K. Johnson. Data pre-processing. In *Applied Predictive Modeling*, Chapter 2, pages 27–59. Springer, New York, NY, USA, 2013a.

M. Kuhn and K. Johnson. Measuring performance in regression models. In *Applied Predictive Modeling*, Chapter 5, pages 95–100. Springer, New York, NY, USA, 2013b.

T. Kühne. What is a model? In *Language Engineering for Model-Driven Software Development*, Dagstuhl Seminar Proceedings. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2005.

D. Li, W. Daamen, and R. M. P. Goverde. Estimation of train dwell time at short stops based on track occupation event data: A study at a dutch railway station. *Journal of Advanced Transportation*, 50(5):877–896, 2016.

B. Lim and S. Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):1–14, 2021.

A Lindfeldt. Investigating the impact of timetable properties on delay propagation on a double-track line using extensive simulation. In *Railway Engineering*, London, UK, 2011.

R. Liu, A. Whiteing, and A. Koh. Challenging established rules for train control through a fault tolerance approach: applications at a classic railway junction. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 227(6):685–692, 2013.

G. Longo and G. Medeossi. Enhancing timetable planning with stochastic dwell time modelling. In *Computers in Railways XIII*, volume 127, pages 461–471, 2012.

G. Longo, G. Medeossi, and A. Nash. Estimating train motion using detailed sensor data. In *Transportation Research Board 91st Annual Meeting*, Washington DC, United States, 2012.

J. Luo, Q. Peng, C. Wen, W. Wen, and P. Huang. Data-driven decision support for rail traffic control: A predictive approach. *Expert Systems with Applications*, 207:118050, 2022.

R. M. Lusby, J. Larsen, and S. Bull. A survey on robustness in railway planning. *European Journal of Operational Research*, 266(1):1–15, 2018.

D. J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

M. Marinov and J. Viegas. A mesoscopic simulation modelling methodology for analyzing and evaluating freight train operations in a rail network. *Simulation Modelling Practice and Theory*, 19(1):516–539, 2011.

L-G. Mattsson. Railway capacity and train delay relationships. In Alan T. Murray and Tony H. Grubesic (eds), *Critical Infrastructure: Reliability and Vulnerability*, pages 129–150. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

F. Mazzanti, G. O. Spagnolo, S. Della Longa, and A. Ferrari. Deadlock avoidance in train scheduling: A model checking approach. In F. Lang and F. Flammini (eds), *Formal Methods for Industrial Critical Systems*, pages 109–123. Springer International Publishing, 2014.

M. McGuire and D. Linder. Train simulation on British Rail. *WIT Transactions on The Built Environment*, 6:437 – 444, 1994.

G. Medeossi. Capacity and reliability on railway networks, a simulative approach. PhD Thesis, Università degli studi di Trieste, 2010.

G. Medeossi and S. de Fabris. Simulation of rail operations. In R. Borndörfer, T. Klug, L. Lamorgese, C. Mannino, M. Reuther, and T. Schlechte (eds), *Handbook of Optimization in the Railway Industry*, Chapter 1, pages 1–24. Springer International Publishing, Cham, Switzerland, 2018.

G. Medeossi, G. Longo, and S. de Fabris. A method for using stochastic blocking times to improve timetable planning. *Journal of Rail Transport Planning & Management*, 1(1):1–13, 2011.

S. Meijer. Introducing gaming simulation in the Dutch railways. *Procedia - Social and Behavioral Sciences*, 48:41–51, 2012.

N. Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7 (35):983–999, 2006.

L. Meng and X. Zhou. Robust single-track train dispatching model under a dynamic and stochastic environment: A scenario-based rolling horizon solution approach. *Transportation Research Part B: Methodological*, 45(7):1080–1102, 2011.

A. Middelkoop and L. Loeve. Simulation of traffic management with FRISO. In *Computers in Railways X*, pages 501–509, 2006.

D. Middelkoop and M. Bouwman. Simone: Large scale train network simulations. In *Proceeding of the 2001 Winter Simulation Conference*, volume 2, pages 1042–1047, 2001.

D. Middelkoop, M. Mazzarello, and D. Vries. Optimizing train traffic: Demonstrating benefits in a case study. In *5th International Seminar on Railway Operations Modelling and Analysis - RailCopenhagen*, 2013.

D. Middelkoop, J. Steneker, S. Meijer, E. Sehic, and M. Mazzarello. Simulation backbone for gaming simulation in railways: A case study. In *Proceedings of the 2012 Winter Simulation Conference*, pages 1–13, 2012.

N. Minbashi, H. Sipilä, C-W. Palmqvist, M. Bohlin, and B. Kordnejad. Machine learning-assisted macro simulation for yard arrival prediction. *Journal of Rail Transport Planning & Management*, 25:100368, 2023. ISSN 2210-9706.

A. Nash and D. Huerlimann. Railroad simulation using Opentrack. In J. Allan, R.J. Hill, C.A. Brebbia, G. Sciutto, and S. Sone (eds), *Computers in Railways IX*, pages 45–54, 2004.

National Audit Office. A financial overview of the rail system in England. Report, 2021.

Network Rail. The network code. Report, Accessed: 2021-04-06, 2019a.

Network Rail. Timetable planning rules - Wessex - 2020 timetable - version 4.0. Report, Accessed: 2021-04-06, 2019b.

Network Rail. Open data feeds. Web page, Accessed: 2020-05-06, 2020.

Network Rail. Railway performance. Web page, Accessed: 2021-04-06, 2021.

A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, page 625–632, 2005.

L. Ning, Y. Li, M. Zhou, H. Song, and H. Dong. A deep reinforcement learning approach to high-speed train timetable rescheduling under disturbances. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3469–3474, 2019.

Office of Rail and Road. Passenger journeys - Table 12.5. Report, Accessed: 12/03/2020, 2020.

Office of Rail and Road. Freight rail usage and performance - January to March 2022. Report, Accessed: 2023-01-30, 2022a.

Office of Rail and Road. Passenger rail usage - January to March 2022. Report, Accessed: 2023-01-30, 2022b.

D. Olave-Rojas and S. Nickel. Modeling a pre-hospital emergency medical service using hybrid simulation and a machine learning approach. *Simulation Modelling Practice and Theory*, 109:102302, 2021.

A. O'Sullivan, F. C. Pereira, J. Zhao, and H. N. Koutsopoulos. Uncertainty in bus arrival time predictions: Treating heteroscedasticity with a metamodel approach. *IEEE Transactions on Intelligent Transportation Systems*, 17(11):3286–3296, 2016.

J. Pachl. Deadlock avoidance in railroad operations simulations. In *90th Annual Meeting of Transportation Research Board in Washington DC*, 2011.

A. Parslov, N. C. Petersen, and F. Rodrigues. Short-term bus travel time prediction for transfer synchronization with intelligent uncertainty handling. *Expert Systems with Applications*, page 120751, 2021.

P. Pellegrini, G. Marlière, and J. Rodriguez. Real time railway traffic management modeling track-circuits. In Daniel Delling and Leo Liberti (eds), *12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, pages 23–34, 2012.

P. Pellegrini, G. Marlière, and J. Rodriguez. A detailed analysis of the actual impact of real-time railway traffic management optimization. *Journal of Rail Transport Planning & Management*, 6(1):13–31, 2016.

N. C. Petersen, F. Rodrigues, and F. C. Pereira. Multi-output deep learning for bus arrival time predictions. *Transportation Research Procedia*, 41:138–145, 2019.

D-T. Pham, D. Alam, and V. Duong. An air traffic controller action extraction-prediction model using machine learning approach. *Complexity*, 2020:1659103, 2020.

J. Platt. Probabilities for SV machines. In A. J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans (eds), *Advances in Large Margin Classifiers*, Chapter 5, pages 61–74. MIT Press Ltd., 2000.

E. Quaglietta. A simulation-based approach for the optimal design of signalling block layout in railway networks. *Simulation Modelling Practice and Theory*, 46:4–24, 2014.

E. Quaglietta, F. Corman, and R. M. P. Goverde. Stability analysis of railway dispatching plans in a stochastic and dynamic environment. *Journal of Rail Transport Planning & Management*, 3(4):137–149, 2013.

E. Quaglietta, P. Pellegrini, R. M. P. Goverde, T. Albrecht, B. Jaekel, G. Marlière, J. Rodriguez, T. Dollevoet, B. Ambrogio, D. Carcasole, M. Giaroli, and G. Nicholson. The ON-TIME real-time railway traffic management framework: A proof-of-concept using a scalable standardised data communication architecture. *Transportation Research Part C: Emerging Technologies*, 63:23–50, 2016.

A. Radtke and J. Bendfeldt. Handling of railway operation problems with RailSys. In *Proceedings of the 5th World Congress on Rail Research*, Cologne, Germany, 2001.

Rail Safety and Standards Board. Brighton mainline reliability modelling: Systems analysis infrastructure reliability T1019. Report, 2014.

Rail Safety and Standards Board. Research in brief: Understanding what makes a good train regulation decision T1178. Report, 2021a.

Rail Safety and Standards Board. T1178 understanding what makes a good train regulation decision - final report. Report, 2021b.

S. Reed, M. Löfstrand, and J. Andrews. Modelling stochastic behaviour in simulation digital twins through neural nets. *Journal of Simulation*, 16(5):1–14, 2021.

B. Roungas, A. Verbraeck, and S. Meijer. The future of contextual knowledge in gaming simulations: A research agenda. In *Proceedings of the 2018 Winter Simulation Conference*, pages 2435–2446, Gothenburg, Sweden, 2018.

B. Ryan, J. R. Wilson, S. Sharples, and T. Clarke. Attitudes and opinions of railway signallers and related staff, using the rail ergonomics questionnaire (REQUEST). *Applied Ergonomics*, 40(2):230–238, 2009.

K. Sage. Concise guide to object-oriented programming: An accessible approach using Java. Springer International Publishing, Cham, Switzerland, 2019.

M. A. Salido, F. Barber, and L. Ingolotti. Robustness in railway transportation scheduling. In *2008 7th World Congress on Intelligent Control and Automation*, pages 2880–2885, Chongqing, China, 2008.

M. Samà, A. D'Ariano, F. Corman, and D. Pacciarelli. A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations. *Computers & Operations Research*, 78:480–499, 2017.

R. G. Sargent. Verification and validation of simulation models. *Journal of Simulation*, 7(1):12–24, 2013.

H. Sipilä. Simulation of rail traffic: methods for timetable construction,delay modeling and infrastructure evaluation. PhD Thesis, KTH Royal Institute of Technology, Stockholm, 2015.

S.L. Sogin, C.T. Dick, Y-C. Lai, and C.P.L. Barkan. Analyzing the incremental transition from single to double track railway lines. In *International Association of Railway Operations Research (IAROR) 5th International Seminar on Railway Operations Modelling and Analysis*, 2013.

A. Ø. Sørensen, A. D. Landmark, N. O. E. Olsson, and A. A. Seim. Method of analysis for delay propagation in a single-track network. *Journal of Rail Transport Planning & Management*, 7(1):77–97, 2017.

Y. Takeuchi, N. Tomii, and C. Hirai. Evaluation method of robustness for train schedules. *Quarterly Report of RTRI*, 48(4):197–201, 2007.

R. Tang, L. De Donato, N. Bešinović, F. Flammini, R. M. P. Goverde, Z. Lin, R. Liu, T. Tang, V. Vittorini, and Z. Wang. A literature review of artificial intelligence applications in railway systems. *Transportation Research Part C: Emerging Technologies*, 140:103679, 2022.

H. Teshima, S. Hori, A. Shimura, and N. Sato. Railway track layout modelling and its application to an automatic route setting system. In C.A Brebbia, N. Tomii, P. Tzieropoulos, and J.M. Mera (eds), *Computers in Railways XIV*, volume 135, pages 75–86, 2014.

E. Tischer, P. Nachtigall, and J. Široký. The use of simulation modelling for determining the capacity of railway lines in the czech conditions. *Open Engineering*, 10(1):224–231, 2020.

Transport Focus. National Rail passenger survey - Autumn 2019 - main report. Report, 2020.

C. P. Ij van Hinsbergen, J. W. C. van Lint, and H. J. van Zuylen. Bayesian committee of neural networks to predict travel times with confidence intervals. *Transportation Research Part C: Emerging Technologies*, 17(5):498–509, 2009.

V. Vignali, F. Cuppi, C. Lantieri, N. Dimola, T. Galasso, and L. Rapagnà. A methodology for the design of sections block length on ETCS L2 railway networks. *Journal of Rail Transport Planning & Management*, 13:100160, 2020.

L. von Rueden, S. Mayer, R. Sifa, C. Bauckhage, and J. Garcke. Combining machine learning and simulation to a hybrid modelling approach: Current and future directions. In M. R. Berthold, A. Feelders, and G. Krempl (eds), *Advances in Intelligent Data Analysis XVIII*, pages 548–560. Springer International Publishing, 2020.

M. J. C. M. Vromans, R. Dekker, and L. G. Kroon. Reliability and heterogeneity of railway services. *European Journal of Operational Research*, 172(2):647–665, 2006.

Q. Wang, Y. Ma, K. Zhao, and Y. Tian. A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, 2020.

R. Watson. Train planning in a fragmented railway: a British perspective. PhD Thesis, Loughborough University, 2008.

R. Watson and G. Medeossi. Simulation. In I. A. Hansen and J. Pachl (eds), *Railway timetabling & Operations*, Chapter 10, pages 191–215. Eurailpress, 2nd edition, 2014.

C. Wen, W. Mou, P. Huang, and Z. Li. A predictive model of train delays on a railway line. *Journal of Forecasting*, 39(3):470–488, 2020.

J. Yuan. Dealing with stochastic dependence in the modeling of train delays and delay propagation. In *International Conference on Transportation Engineering*, pages 3908–3914, 2007.

J. Yuan and I. A. Hansen. Optimizing capacity utilization of stations by estimating knock-on train delays. *Transportation Research Part B: Methodological*, 41(2):202–217, 2007.

B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 694–699, 2002.

Z-H. Zhou and J. Feng. Deep forest: Towards an alternative to deep neural networks. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 3553–3559, 2017.

L. Zhu, Y. He, F. R. Yu, B. Ning, T. Tang, and N. Zhao. Communication-based train control system performance optimization using deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 66(12):10705–10717, 2017.

Y. Zhu, H. Wang, and R. M. P. Goverde. Reinforcement learning in railway timetable rescheduling. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2020.

D. Šemrov, R. Marsetič, M. Žura, L. Todorovski, and A. Srdic. Reinforcement learning approach for train rescheduling on a single-track railway. *Transportation Research Part B: Methodological*, 86:250–267, 2016.