Profit Driven Decision Trees for Churn Prediction

Sebastiaan Höppner[a], Eugen Stripling[b], Bart Baesens[b,c,*], Seppe vanden Broucke[b], Tim Verdonck[a]

[a] *KU Leuven, Department of Mathematics, Celestijnenlaan 200B, 3001 Leuven, Belgium*
[b] *KU Leuven, Faculty of Economics and Business, Naamsestraat 69, 3000 Leuven, Belgium*
[c] *University of Southampton, School of Management, Highfield Southampton, SO17 1BJ, United Kingdom*

Abstract

Customer retention campaigns increasingly rely on predictive models to detect potential churners in a vast customer base. From the perspective of machine learning, the task of predicting customer churn can be presented as a binary classification problem. Using data on historic behavior, classification algorithms are built with the purpose of accurately predicting the probability of a customer defecting. The predictive churn models are then commonly selected based on accuracy related performance measures such as the area under the ROC curve (AUC). However, these models are often not well aligned with the core business requirement of profit maximization, in the sense that, the models fail to take into account not only misclassification costs, but also the benefits originating from a correct classification. Therefore, the aim is to construct churn prediction models that are profitable and preferably interpretable too. The recently developed expected maximum profit measure for customer churn (EMPC) has been proposed in order to select the most profitable churn model. We present a new classifier that integrates the EMPC metric directly into the model construction. Our technique, called ProfTree, uses an evolutionary algorithm for learning profit driven decision trees. In a benchmark study with real-life datasets from various telecommunication service providers, we show that ProfTree achieves significant profit improvements compared to classic accuracy driven tree-based methods.

Keywords: Artificial intelligence, customer churn prediction, classification, evolutionary algorithm, profit-based model evaluation

*Corresponding author: Bart Baesens

Email addresses: sebastiaan.hoppner@kuleuven.be (Sebastiaan Höppner), eugen.stripling@kuleuven.be (Eugen Stripling), bart.baesens@kuleuven.be (Bart Baesens), seppe.vandenbroucke@kuleuven.be (Seppe vanden Broucke), tim.verdonck@kuleuven.be (Tim Verdonck)

1. Introduction

Companies operating in saturated markets are continuously challenged to retain their customers. Besides spending resources on attracting new customers, they try to prevent existing customers from defecting. To that end, customer retention campaigns aim to identify which customers intend to switch to a competitor and present them with an incentive offer to remain with the company. However, detecting potential churners out of typically millions of customers is a difficult task. For that reason, companies increasingly rely on predictive churn models to remain competitive. Churn models aim to predict a customer's churning propensity by using behavioral and historical information. Yet, these models often focus on achieving maximum prediction accuracy rather than aiming their attention at the most important business requirement: profit maximization. For a retention campaign to be successful, it is not only crucial to correctly identify would-be churners, but also to detect those who are the most profitable to the business and thus worth retaining. The ideal churn prediction model is therefore capable of effectively identifying churners and simultaneously taking profit concerns of the business into account.

Traditionally, the performance of a churn model is evaluated using accuracy related measures, which do not take profit maximization into concern. For instance, for binary classification problems, a popular choice for selecting the winning model is the area under the ROC curve (AUC), because of its simplicity and intuitive interpretation. However, it has been shown by Hand (2009) that the AUC implicitly assumes that misclassification errors carry the same costs which also alter across classifiers. Reasonably, the cost of misclassifying a non-churner as a churner is different from the cost incurred when an actual churner is classified as a non-churner. Furthermore, misclassification costs are ultimately a property of the classification problem, and should be independent of the applied classifier. Besides taking costs into concern, it is also generally recommended to incorporate the benefits of making a correct classification into the performance metric (Elkan, 2001).

For predictive churn models, Verbraken et al. (2013) proposed a profit-based performance metric, called the *expected maximum profit measure for customer churn* (EMPC), which allows identifying the most profitable churn model. Performance is measured based on the average classification profit in which the specified costs and benefits are the ones associated with a retention campaign. Additionally, the proposed cost benefit framework provides the *expected profit maximizing fraction for customer churn* ($\overline{\eta}_{empc}$) that determines the optimal fraction of the customer base to target in the retention campaign. Verbraken et al. (2013) showed that there are great discrepancies between the EMPC and AUC when assessing a model's performance, and that the use of AUC as a model selection criterion

leads to suboptimal profits. For customer retention campaigns, it is therefore recommended to select the winning model based on the EMPC in order to achieve maximum profit.

Although the EMPC enables a profit-based model evaluation, profit concerns are not directly integrated into the model construction. Therefore, we propose a new profit maximizing classifier, called ProfTree, that optimizes the EMPC directly in the construction process of a classification tree. So far, profit measures have been used for post-construction tree evaluation. However, to the best of our knowledge, profit driven measures have not been used within the tree construction itself. In our approach, a classification tree method is utilized to estimate churn scores, that is, the probability of a customer leaving the company. These scores are required for computing the profit measure. We opt to use decision trees because they are typically easy to interpret and fast to compute. This is especially useful in a churn prediction setting to understand why customers defect and work out corresponding churn prevention strategies. The split rules of the decision tree are optimized according to the EMPC metric using an evolutionary algorithm. The choice for the usage of a global optimization method like an evolutionary algorithm is justified because classical forward-search recursive partitioning methods, such as CART (Breiman et al., 1984) and C4.5 (Quilan, 1993), will only yield locally optimal solutions.

This paper is organized as follows. Section 2 discusses the essential building blocks that are relevant to ProfTree as well as related work. Section 3 outlines the detailed explanations of our research contributions. Section 4 illustrates the usage of ProfTree on a real-life customer churn dataset and compares our method with other well-known classifiers. Section 5 describes the experimental setup and the results of an extensive benchmarking study. The empirical evaluation and comparison with traditional measures using a unique collection of churn data sets illustrates the difference between profit versus impurity-based tree construction and the impact thereof. Finally, we give some concluding remarks and potential directions for future research in Section 6.

2. Preliminaries

2.1. Customer Churn Prediction Formulated as a Classification Problem

Predicting customer churn can be formulated as a binary classification problem. The aim is to assign instances (i.e. customers) to one of the two classes $Y = \{\text{no churn, churn}\}$ based on their observed features $\boldsymbol{x} \in \boldsymbol{X}$. A popular method for dealing with binary classification problems is a decision tree because it is easy-to-use and offers high interpretability. Classification trees can also cope with complex data structures like nonlinearities and can naturally handle categorical features.

A classification tree aims at modeling the binary response variable Y by a vector of p predictor variables $\boldsymbol{X} = (X_1, \ldots, X_p)$. Throughout the text, we encode "churn" as 1 and "no churn" as 0, so {no churn, churn} becomes $\{0, 1\}$. Tree-based methods first partition the feature space $\boldsymbol{X}$ into a set of M rectangular regions R_m $(m = 1, \ldots, M)$ based on split rules, and then fit a (typically simple) model within each region $\{Y|\boldsymbol{X} \in R_m\}$, e.g. a constant like the mode. In this section, we will closely follow the notation used by Grubinger et al. (2014) to describe the partitioning algorithm, the parameter space and the optimization problem. As done by Grubinger et al. (2014), we only consider tree models with two-way splits and with some maximum number of terminal nodes M_{max}. Throughout the text, we denote a classification tree with M terminal nodes by

$$\theta = (v_1, s_1, \ldots, v_{M-1}, s_{M-1}) \,. \tag{1}$$

Note that if a tree model contains M terminal nodes, it consequently has $M-1$ internal splits. These internal nodes $r \in \{1, \ldots, M-1\}$ consist of a splitting variable $v_r \in \{1, \ldots, p\}$ and the associated split rule (or point) s_r. For ordered and numeric variables X_{v_r}, the split rule s_r involves a cutoff and there are $u-1$ possible splits if X_{v_r} takes u distinct values. For a categorical variable with k levels, the split rule contains a (non-empty) subset of $\{1, \ldots, k\}$ and there are $2^{k-1}-1$ possible splits. Based on the split rules, observations are sent to either the first or second subset. The product of these combinations forms all potential elements θ from Θ_M, the space of conceivable trees with M terminal nodes. The overall parameter space is then $\Theta = \bigcup_{M=1}^{M_{max}} \Theta_M$.

Let $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ denote the N observed predictor-response pairs in the dataset, where $y_i \in \{0, 1\}$ describes the response and $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^t$ represents the p associated predictor variables of instance i. For each customer, a probability estimate or score, $s \in [0, 1]$, can be calculated based on the observed features $\boldsymbol{x}_i$ of the customer. In the churn context, the probability of an instance being a churner is determined by a score function $s(\boldsymbol{X}, \theta)$ which is based on all explanatory variables $\boldsymbol{X}$ and the chosen tree structure θ from (1). The instances from class 0 (no churn) are assumed to have a lower score than the instances from class 1 (churn). We define the score of instance i as

$$s(\boldsymbol{x}_i, \theta) = \sum_{m=1}^{|\theta|} \hat{p}_m I(\boldsymbol{x}_i \in R_m) \tag{2}$$

where $|\theta|$ is the number of terminal nodes (i.e. regions R_m) and

$$\hat{p}_m = \frac{1}{N_m} \sum_{i:\boldsymbol{x}_i \in R_m} I(y_i = 1) \tag{3}$$

is the proportion of class 1 observations in node m which represents a region R_m with N_m observations. It is obvious from (2) and (3) that the churn scores lie between zero and one. Note that a

higher score indicates a higher likelihood of churning. These scores are often converted to predicted classes $\hat{y} \in \{0, 1\}$ by comparing them with a classification threshold $t \in [0, 1]$. All instances with a score s smaller than t are classified as non-churners, i.e. $s(\boldsymbol{x}_i, \theta) \leq t \Rightarrow \hat{y}_i = 0$, whereas instances with s larger than t are classified as churners, i.e. $s(\boldsymbol{x}_i, \theta) > t \Rightarrow \hat{y}_i = 1$.

2.2. General Fitness Function of Evolutionary Decision Trees

At the heart of most classification methods lies a fitness function that is optimized by the respective algorithm. In order to prevent that the estimated model is overfitted on the training sample, the complexity of the tree is often included in the algorithm's fitness function. As done by Breiman et al. (1984) and Grubinger et al. (2014), we measure the complexity of a tree by a function of the number of terminal nodes, $|\theta|$, without further considering the depth or the shape of the tree. The aim of the algorithm is then to find the classification tree which optimizes a given fitness function that describes some tradeoff between prediction performance and complexity (Grubinger et al., 2014):

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \operatorname{loss}\left(Y, s\left(\boldsymbol{X}, \theta\right)\right) + \operatorname{comp}(\theta) \tag{4}$$

where $\operatorname{loss}(\cdot, \cdot)$ represents a suitable loss function for Y. Popular loss functions for classification are the misclassification rate, deviance (i.e. cross-entropy) or Gini index. The function $\operatorname{comp}(\cdot)$ is monotonically non-decreasing in the number of terminal nodes $|\theta| = M$ of the tree θ. As a result, more complex models are penalized in the tree selection process as they are less favorable. Note that finding $\hat{\theta}$ requires a search over all spaces Θ_M for $M \in \{1, \ldots, M_{max}\}$. The classification algorithm EvTree, implemented by Grubinger et al. (2014), measures the quality of a classification tree as a function of its misclassification rate (MC) and the complexity of a tree by its number of terminal nodes M, weighted by $\log N$ and a user-specified parameter α:

$$\begin{aligned} \operatorname{loss}\left(Y, f\left(\boldsymbol{X}, \theta\right)\right) &= 2 \cdot \sum_{i=1}^{N} I(y_i \neq f(x_{i,\theta})) = 2N \cdot \operatorname{MC}\left(Y, f\left(\boldsymbol{X}, \theta\right)\right) \\ \operatorname{comp}(\theta) &= \alpha \cdot M \cdot \log N \end{aligned} \tag{5}$$

where $f\left(\boldsymbol{X}, \theta\right)$ denotes the prediction function based on all explanatory variables $\boldsymbol{X}$ and the chosen tree structure θ.

Even for medium sized problems with a relatively small number of observations and features, it is clear that the complete parameter space Θ in (4) can become very large. In fact, Hyafil and Rivest (1976) showed that building optimal binary decision trees, such that the expected number of splits required to classify an unknown sample is minimized, is NP-complete. Instead of searching all

possible combinations in Θ simultaneously and measuring their fitness, classic recursive partitioning algorithms, like CART (Breiman et al., 1984) and C4.5 (Quilan, 1993), only consider one split at a time. This means that at each internal node $r \in \{1, \ldots, M-1\}$, the split variable v_r and the corresponding split point s_r are selected to locally minimize the loss function. Moreover, CART and C4.5 employ an exhaustive search for the r-th split, jointly over (v_r, s_r). So-called unbiased recursive partitioning techniques modify this search by first selecting the variable v_r using statistical significance tests and subsequently selecting the optimal split s_r for that particular variable. This approach is used in conditional inference trees (see (Hothorn et al., 2006), for references to other algorithms) and avoids selecting variables with many potential splits more often than those with fewer potential splits. Nevertheless, all of these forward-search recursive partitioning methods only search each tuple (v_r, s_r) once without taking the subsequent split rules into account. Although it has been shown that this approach is an efficient heuristic, it typically leads to a local optimal solution. An alternative way to search over the parameter space of trees is to use global optimization methods like an evolutionary algorithm as is done by Grubinger et al. (2014).

2.3. Profit-based Classification Performance Evaluation

The quality of a churn model is traditionally assessed as a binary classification model, using a performance measure. Most performance metrics are extracted from a confusion matrix as shown in Table 1. Depending on a given classification threshold $t \in [0, 1]$, the confusion matrix tabulates the numbers of correct and incorrect classifications based on the churn scores s produced by the predictive model. If $s \leq t$, the model assigns a "no churn" label to the instance, and if $s > t$, the model assigns a "churn" label. The prior class probabilities of instances belonging to class 0 or 1 are represented by π_0 and π_1, respectively. Furthermore, $f_0(s)$ and $f_1(s)$ are the probability density functions of the classification scores, whereas $F_0(s)$ and $F_1(s)$ are the cumulative distribution

Predicted class	**Actual class**	
	Class 0	Class 1
Class 0	$\pi_0 F_0(t) N$	$\pi_1 F_1(t) N$
	$[b_0 = c(0 \mid 0)]$	$[c_1 = c(0 \mid 1)]$
Class 1	$\pi_0(1 - F_0(t))N$	$\pi_1(1 - F_1(t))N$
	$[c_0 = c(1 \mid 0)]$	$[b_1 = c(1 \mid 1)]$

Table 1: Confusion matrix with associated benefits (b_k) and costs (c_k), $k \in \{0, 1\}$, for a correct and incorrect classification, respectively.

functions of the scores for class 0 and 1, respectively. Examples of some well accepted measures for binary classification problems are *Recall*$(t) = F_0(t)$, *Precision*$(t) = \pi_0 F_0(t) / (\pi_0 F_0(t) + \pi_1 F_1(t))$, F_1-*measure*$(t) = 2\pi_0 F_0(t) / (\pi_0 + \pi_0 F_0(t) + \pi_1 F_1(t))$, $MER = \min_{\forall t}\{\pi_0 (1 - F_0(t)) + \pi_1 F_1(t)\}$ and $AUC = \int_{-\infty}^{+\infty} F_0(s) f_1(s) ds$.

Most of these performance measures are a function of the classification threshold. However, the area under the ROC curve (AUC) and the minimum error rate (MER) do not require to specify the classification threshold which makes them one of the most employed measures to objectively evaluate the classification performance. The AUC of a classifier can be interpreted as being the probability that a randomly chosen churner is predicted a higher score than a randomly chosen non-churner. Therefore, a higher AUC indicates superior classification performance.

The outcome of a classification task is typically used as input to the retention campaign which leads to costs for misclassifications and benefits for correct classifications. The cost or benefit related to labeling an instance from class k as a class l instance is denoted with $c(l|k)$, $k, l \in \{0, 1\}$. As indicated in Table 1, correct classifications are rewarded with a benefit $b_k = c(l = k|k)$ while misclassifications are penalized and associated with costs $c_k = c(l \neq k|k)$. Profit is then computed by offsetting the costs and benefits against each other. Note that *Recall*, *Precision*, F_1-*measure*, *MER* and AUC do not explicitly take classification costs or benefits into account. Therefore, they are only valid if the gains of correct classifications and the severities of misclassifications are equal. Churn prediction problems are typically dealing with high class imbalance where the minority class (i.e. churners) is of primary interest, thus making the assumption of equal benefits and costs unrealistic. Indeed, misclassifying an actual churner as a non-churner, also known as a false negative, results in the loss of a customer, while misclassifying a non-churner as a churner, also known as a false postive, leads to additional costs to the company (e.g. cost of contacting the customer and the cost of an incentive offer) since those misclassified customers do not intend to leave. As a result, applying accuracy-based performance measures for the evaluation and the selection of a classifier in churn management is not recommended.

Verbraken et al. (2013) proposed the cost benefit analysis framework for customer churn, which incorporates both the costs associated with a retention campaign and the benefits of retained customers. The framework defines the average classification profit function as:

$$P_C(t; \gamma, CLV, \delta, \phi) = CLV\,(\gamma(1 - \delta) - \phi)\,\pi_0 F_0(t) - CLV(\delta + \phi)\pi_1 F_1(t), \tag{6}$$

where t is the classification threshold and γ is the probability that a targeted would-be churner accepts a special offer and remains a customer. CLV represents the constant customer lifetime value

per retained customer (€200). The two dimensionless parameters $\delta = d/CLV$ and $\phi = f/CLV$ are derived from d, the constant cost of the retention offer (€10), and f, the constant cost of contacting a customer (€1). It is assumed that all costs involved are strictly positive and $CLV > d$. The values between brackets are the recommended default values for customer retention campaigns in the telecommunication sector as suggested by Verbraken et al. (2013).

The cost benefit framework comprises a deterministic and a probabilistic profit-based performance measure. The former is the *maximum profit measure for customer churn* (MPC):

$$MPC = \max_{\forall t} \{P_C(t; \gamma, CLV, \delta, \phi)\} \tag{7}$$

The MPC is deterministic in the sense that all parameters related to costs and benefits (γ, CLV, δ and ϕ) are assumed to be known. In a retention campaign, most of these parameters can be approximated with sufficient precision, except the probability γ that a customer will accept the offer. This can be solved by assigning a probability density function to γ, denoted as $h(\gamma)$, which yields the *expected maximum profit measure for customer churn* (EMPC):

$$EMPC = \int_{\gamma} P_C(t_{opt}(\gamma); \gamma, CLV, \delta, \phi) h(\gamma) \mathrm{d}\gamma, \tag{8}$$

where t_{opt} is the optimal classification threshold that maximizes the profit for given γ:

$$t_{opt} = \operatorname*{argmax}_{\forall t} \{P_C(t; \gamma, CLV, \delta, \phi)\} \tag{9}$$

Verbraken et al. (2013) decided to specify a beta distribution for $h(\gamma)$ with the restrictions that its parameters are $\alpha' > 1$ and $\beta' > 1$. Furthermore, the authors propose to set α' and β' equal to 6 and 14, respectively. The EMPC follows a probabilistic approach that considers a range of γ values, representing the uncertainty in that parameter. In this paper, we will focus on the EMPC because this scenario is more likely to be encountered in practice. Yet, the most profitable classifier can be identitief unambiguously by both profit measures.

Additionally, the proposed cost benefit framework provides the *(expected) profit maximizing fraction for customer churn*, $\overline{\eta}$ (Verbraken et al., 2013), which gives practitioners an estimate of the optimal fraction of the customer base which needs to be targeted in the retention campaign. Following the deterministic approach, it becomes:

$$\overline{\eta}_{mpc} = \pi_0 F_0(t_{opt}) + \pi_1 F_1(t_{opt}) \tag{10}$$

whereas the profit maximizing fraction derived from the EMPC approach is defined as:

$$\overline{\eta}_{empc} = \int_{\gamma} \left[\pi_0 F_0(t_{opt}(\gamma)) + \pi_1 F_1(t_{opt}(\gamma))\right] h(\gamma) \mathrm{d}\gamma \tag{11}$$

Making an arbitrary choice of taking, for example, the top 10% of predicted would-be churners, will likely result in suboptimal profits. Instead, all customers are ranked according to their predicted churn score and then $\overline{\eta}_{empc}$ indicates which top fraction of the predicted would-be churners should be targeted in the campaign. The result is a customer list with the top would-be churners. Therefore, the $\overline{\eta}$ estimates are appealing to practitioners since they help to determine how many customers should be targeted in the retention campaign for attaining maximal profit.

Based on the customer list of top would-be churners, Stripling et al. (2018) introduced three additional performance measures that are related to the notion of precision, recall, and the F_1 measure, but they are independent of the classification threshold t. The $\overline{\eta}$-precision or hit rate for customer churn, $\overline{\eta}_p$, is the proportion of correct identifications of would-be churners in the $\overline{\eta}_{empc}$-based customer list. The $\overline{\eta}$-recall for customer churn, $\overline{\eta}_r$, is the proportion of churners that is included in the $\overline{\eta}_{empc}$-based customer list. Precision and recall are often combined into the F_1 measure, which represents the harmonic mean between the two measures. The F_1 measure reaches its best value at 1 and worst at 0. The $\overline{\eta}$-based F_1 measure for customer churn, $\overline{\eta}_{F_1}$, is defined as $\overline{\eta}_{F_1} = 2 \cdot \overline{\eta}_p \overline{\eta}_r / \left(\overline{\eta}_p + \overline{\eta}_r\right)$. For all three measures, higher values indicate higher effectiveness of the customer list and thus better performance of the classifier. However, since these measures only calculate accuracy, the final model should be selected based on the EMPC as we aim for maximum profit. Nevertheless, it is interesting to compare hit rates ($\overline{\eta}_p$) among classifiers in order to assess their effectiveness of correctly identifying churners.

2.4. Evolutionary algorithms

An evolutionary algorithm (EA) is a metaheuristic algorithm inspired by the biological process of evolution to solve complex optimization problems (Eiben and Smith, 2015). Many different variants of EA have been proposed with the most prominent techniques being genetic algorithms (Holland, 1992), evolutionary programming (Fogel et al., 1966) and genetic programming (Koza, 1992). Despite the technical differences, the same fundamental idea is behind all evolutionary methods.

Within some defined environment, a population of individuals undergoes an evolution process that is guided by some fitness function, a measure for judging the quality of an individual. The goal is to find the fittest individual within that environment. In this evolution process, the population members compete against each other for survival, where each individual represents a candidate solution to the problem at hand. In the spirit of *survival of the fittest*, the evolutionary system is typically designed such that individuals with a higher fitness have a higher chance to survive. The fittest individual returned by the EA represents ultimately the final solution to the optimization problem.

At the start, one usually creates the population, the set of candidate solutions, in a random fashion and then repeatedly applies genetic operators to evolve the population toward the optimal solution. The most distinguished genetic operators are *selection*, *crossover*, and *mutation*. As the name suggests, the selection operator chooses (either deterministically or stochastically) individuals based on their fitness values that should undergo a genetic operation. Crossover is the recombination operator that is applied to two or more selected candidates (the so-called parents), which as a result produces one or more new candidates (the children). The inner values (called genes) of the parents are thereby exchanged in a predefined manner. The mutation operator is another way of creating a new candidate from one selected candidate, usually by randomly changing the genes. By repeatedly applying the genetic operators and updating the population, the EA converges toward the optimal solution from generation to generation, assuming it does not stop prematurely.

The immense flexibility of designing the building blocks of an EA system enables users to solve highly complex optimization problems. Assuming the building blocks are properly defined and no premature termination has occurred, EAs are capable of handling complex data structures such as classification and regression trees and return globally optimal solutions (Grubinger et al., 2014). It should therefore come as no surprised that thanks to this property EAs are often the preferred choice and have been applied to a large variety of optimization and search problems (Freitas, 2003).

2.5. Related work

The extensive literature review given by Verbeke et al. (2011) shows that predictive classification techniques for customer churn are increasingly researched. Various machine learning methods have been set up for the task of predicting customer churn, including support vector machines (Chen et al., 2012) and ensemble methodes (Van Wezel and Potharst, 2007). In saturated markets such as the telephone service industry, the attraction of new customers is eminently challenging and costs much more than preventing existing customers from churning. As a consequence, multiple data mining techniques have been researched and applied within the telco industry (Verbeke et al., 2012).

The idea of directly integrating the EMPC as a profit measure into the model construction is also considered by Stripling et al. (2015, 2018). Here, a classifier, called ProfLogit, is proposed whose internal model structure resembles a lasso-regularized logistic model. The model parameters of ProfLogit are described by a vector $\boldsymbol{\theta} = (\beta_0, \boldsymbol{\beta}) \in \mathbb{R}^{p+1}$, consisting of the intercept β_0 and the regression coefficients $\boldsymbol{\beta}$ for each of the p predictor variables. A genetic algorithm (GA) is then used to solve a real-parameter optimization problem and find the optimal parameter vector $\boldsymbol{\theta}$ that maximizes the EMPC directly in the training step. While a logistic regression model is fully characterized by its

parameter vector $\boldsymbol{\theta}$, whose size is fixed and known, a decision tree is made up of a more complex data structure. A tree model consists of nodes and splits as described in (1), but the size and structure of the model are not predetermined, except that the number of terminal nodes can not exceed the number of cases in the training set. The goal of the evolutionary algorithm (EA) in ProfTree is to find the optimal split rules that correspond to a maximum on the EMPC landscape. ProfTree is the first classifier to use the EMPC measure directly in the tree construction process. Some papers have used profit measures for ex-post tree evaluation. However, to the best of our knowledge, no papers have used profit driven measures for tree construction. We believe this to be of great relevance to both academics and practitioners focusing on profit, rather than traditional decision tree impurity criteria (e.g. Gini index, cross-entropy). An advance of using ProfTree is that it allows modelling nonlinear patterns which offers more flexibility than ProfLogit which in turn results in greater gains.

Besides the cost benefit framework of the EMPC, there are other frameworks that are used for customer churn predictive modeling. The framework of the EMPC, as proposed by Verbraken et al. (2013), allocates costs and benefits to each particular class. An alternative approach is to measure classification costs at the level of individual customers. This means that distinct costs are set for each respective customer which leads to a more detailed description of how costly a particular customer is. The classification performance of a classifier can then be measured by the total cost as the sum of all individual costs. This example-dependent cost-sensitive framework for churn is proposed by Bahnsen et al. (2015b). Their framework is purely cost-based since it does not account for any benefits coming from retained customers. For instance, unlike in the cost benefit framework of the EMPC, the CLV is considered as a cost for effectively churned customers rather than a benefit of retained customers. In Bahnsen et al. (2015a), the example-dependent cost-sensitive framework is introduced to decision trees. Their algorithm is based on a recursive partitioning approach which builds a decision tree by optimizing a cost based impurity measure that incorporates the different example-dependent costs as well as cost based pruning criteria.

In Glady et al. (2009), the concept of a churner is defined as a function of the customer lifetime value. More specifically, a churner is defined as a customer whose CLV is decreasing where CLV is defined as the discounted value of future marginal earnings related to the customer's activity. Additionally, a new loss function is introduced to assess the misclassification cost of a customer that is incurred by the decrease in CLV.

3. Profit Maximizing Classification Trees

In this section, we introduce our new ProfTree classification technique, which optimizes the splitting variables and associated split rules according to the EMPC, aiming to produce the most profitable classifier. An evolutionary algorithm is used to find the optimal tree model that corresponds to a maximum on the EMPC landscape. In what follows we describe the building blocks of ProfTree.

3.1. Fitness Function of ProfTree

The fitness function represents the requirements to which the population of trees should adapt. In general, these requirements are formulated by (4). ProfTree's fitness function is defined by substituting the loss function in (4) with the EMPC measure (8) and measuring the complexity of a tree as its number of terminal nodes, weighted by a user-specified parameter λ. This ultimately yields a profit-sensitive classification model:

$$\hat{\theta}^{empc} = \underset{\theta \in \Theta}{\operatorname{argmax}}\ EMPC(\theta) - \lambda \cdot |\theta| \tag{12}$$

where $\hat{\theta}^{empc}$ is the globally optimal classification tree according to the EMPC measure with minimized complexity. Due to the complex optimization function, ProfTree uses an EMPC-modified version of the EA, described in Grubinger et al. (2014), for learning globally optimal classification trees. Note, however, that alternative tree-based EAs exist (see, e.g., Soak et al. (2006); Raidl and Julstrom (2003); Palmer and Kershenbaum (1994, 1995); Jankowski and Jackowski (2014); Lin and Gen (2006); Carrano et al. (2007)). ProfTree is implemented in R and it exploits the functionalitities available in the *evtree* package (Grubinger et al., 2014).

3.2. The ProfTree algorithm

The pseudocode for the ProfTree algorithm is provided in Algorithm 1. The algorithm starts by initializing a population of randomly created trees. Each initial tree of the population is generated by adding a split rule to the root node in which both the splitting variable and corresponding split point are selected randomly. These trees are considered to be the first generation.

Once the population is generated, each individual tree is evaluated by the fitness function $EMPC(\theta) - \lambda \cdot |\theta|$ for a given value of the parameter λ. In order to improve the fitness of the trees, each of them are altered via so-called variation operators. These operators cause the trees to grow according to (12) in order to maximize their fitness. After evaluating each new solution and comparing it with the previous version, the algorithm will select the fittest trees as the survivors for the next generation. Once a new generation has been created from these survivors, the iteration

Algorithm 1: ProfTree: Profit Driven Decision Tree for Churn Prediction

Inputs: churn data set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ with p predictors;
λ, regularization parameter;
$|\mathcal{P}|$, number of trees in the population (default: 100);
minimum number of observations in each terminal node (default: 7);
minimum number of observations in each internal node (default: 20);
maximum tree depth (default: 9);
G, maximum number of iterations (default: 10,000);
probabilities for the five variation operators (default: 0.2 for each operator);
EMPC parameters: CLV (€200), d (€10), f (€1), α' (6) and β' (14);

Initialization of the evolutionary algorithm (EA)

Initialize: $g \leftarrow 0$ (generation index)
Create initial θ population, $\mathcal{P}_g$, of size $|\mathcal{P}|$. Each tree of the population is initialized with a valid, randomly generated, split rule in the root node.

Evaluate: $\forall \theta \in \mathcal{P}_g$, evaluate the fitness of tree θ as $EMPC(\theta) - \lambda \cdot |\theta|$

Main loop of the EA

while $\{g \leq G$ and termination conditions (Section 3.4) are not satisfied$\}$ **do**

Select: each tree θ is selected once to be modified by one of the variation operators;

Modify: one of the following variation operators is randomly selected for each θ:
split, prune, major split rule mutation, minor split rule mutation, and crossover

Evaluate: $\forall \theta \in \mathcal{P}_g$, evaluate the fitness of the new tree θ as $EMPC(\theta) - \lambda \cdot |\theta|$

Update: $\mathcal{P}_g$ to $\mathcal{P}_{g+1}$ where each parent tree competes with its offspring for a place in the population (Grubinger et al., 2014). The tree with lower fitness is rejected.
$g \leftarrow g + 1$

end while

return $\hat{\theta}$, the decision tree with the highest fitness in $\mathcal{P}_g$

starts again by evaluating the quality of each new tree in the population. The overall fitness of the population will increase by each iteration until a termination condition is met. The details of the termination conditions are given below. The result of the ProfTree algorithm is the tree with the highest fitness according to (12).

3.3. Variation operators

In each iteration, each tree in the population is modified by one of the variation operators which consist of four types of mutation operators and one crossover operator. The details of these operators

are described by Grubinger et al. (2014). The *split* operator adds a randomly generated split rule to a randomly selected terminal node. The *prune* operator prunes a random internal node, which has two terminal nodes as successors, into a terminal node. The *major split rule mutation* operator selects a random internal node and changes the entire split rule by altering both the corresponding split variable and the split point. The *minor split rule mutation* operator is similar to the previous one, except that it does not change the split variable; only the split point is slightly altered. Finally, the *crossover* operator randomly choses two "parent" trees and exchanges randomly selected subtrees between them. Due to this operator, some trees are selected more than once in each iteration as it needs a second parent.

3.4. Termination criteria

The ProfTree algorithm executes a minimum of 1,000 iterations after which it terminates when the quality of the best 5% of trees has not improved for 100 iterations. If this convergence criterium is not met, the algorithm will terminate after a user-specified number of iterations, which is set at 10,000 by default. The algorithm returns the tree with the highest performance according to (12).

3.5. Control parameters

The implementation of ProfTree contains several parameters that control the evolutionary search. The solution can be constrained to a minimum number of observations in each internal node, a minimum number of observations in each terminal node, and a maximum tree depth. The parameter λ controls the overall complexity (i.e. number of terminal nodes) of the trees in each generation. If λ is increased, the number of terminal nodes will decrease and vice versa. The evolutionary search can further be controlled by specifying the number of trees in the population (which is set at 100 by default) as well as the probabilities for the variation operators. In each modification step, one of these variation operators is chosen at random for each tree. The probabilities by which the operators are selected can be specified by the user. By default, each of the five variation operators are given a 20% probability of being selected.

3.6. Tuning the Regularization Parameter

The parameter λ governs the tradeoff between (expected) profit and tree size. In order to estimate the optimal regularization parameter value, λ_{opt}, we generate a grid of λ values as follows:

$$\Lambda = \{\lambda \mid \lambda_{min} \leq \lambda \leq \lambda_{max}\}. \tag{13}$$

Based on many preliminary studies on datasets with widely different dimensions, we found that the optimal value λ_{opt} typically resides in the interval $[0, 1]$. Therefore, we specify $\lambda_{max} = 1$. Any value below λ_{max} relaxes the penalization on the size of the tree. However, going down to $\lambda = 0$ will remove the penalization which typically leads to the problem of overfitting the classification tree. For this reason, we specify $\lambda_{min} = 0.01$. For the case study in Section 4 and the benchmark comparison in Section 5, the grid consists of the following $|\Lambda| = 21$ values:

$$\Lambda = \{0.01, 0.05, 0.10, 0.15, \ldots, 0.95, 1.00\} \tag{14}$$

The estimated optimal value λ_{opt} corresponds to the value on the grid with the highest EMPC performance (8) on a hold-out sample. Note that $|\Lambda| = 21$ is an arbitrary choice, and one should choose $|\Lambda|$ large enough to create a dense grid. In order to confidently determine λ_{opt}, a reliable performance estimate has to be obtained for each λ value of the grid. The underlying evolutionary algorithm of ProfTree is intrinsicly stochastic in nature. This requires that the analysis has to be repeated several times to obtain average performance estimates.

Therefore, we start by splitting the original dataset into a training set and test set, which are stratified according to the churn indicator. The test set will be used exclusively to evaluate the performance of the final model such that we obtain so-called out-of-sample estimates. Next, we perform five replications of twofold cross-validation (5×2 cv) (Dietterich, 1998; Demšar, 2006) on the training set in which each fold is stratified according to the churn indicator. In each replication, ProfTree is trained with a given λ on each fold (i.e. one half of the training set), and its EMPC performance is measured on the other fold (i.e. other half of the training set which functions as a kind of validation set or hold-out sample). The EMPC measure is hereby used with its default values as specified in Section 2.3. Next, the average of the 10 estimates becomes the associated performance estimate for the given λ. This procedure is performed for all $\lambda \in \Lambda$, and λ_{opt} corresponds to the λ value with the highest average EMPC performance. This aproach for tuning the regularization parameter is used for both the case study in Section 4 and the experiments in Section 5.

To determine the final classification performance, ProfTree is trained 50 times with λ_{opt} on the entire training set, and its final performance is evaluated based on the test set, which has previously not been used for either training or finding λ_{opt}. According to this procedure, the tuning of the regularization parameter and the final estimation of ProfTree's classification performance requires in total the construction of $|\Lambda| \times (5 \times 2) + 50 = 260$ models. Note that the approach explained in this section also applies to EvTree (Grubinger et al., 2014) since the method also relies on an evolutionary algorithm to build the tree which in turn requires a regularization parameter (5). This

Variable	Description
`churn`	Did the customer leave the telecom operator?
`region`	Region where the customer lives.
`prod_num`	Number that identifies the customer's product.
`active_months`	Time since the customer joined the operator (in months).
`contract_period`	Length of the contract period.
`revenue_avg`	Average revenue.
`nonpay_period`	How long did the customer not pay the bills?
`overdue_amt`	Amount that the customer is overdue.
`count_disconnect`	Number of times the service was disconnected.
`count_complaint`	Number of filed complaints.
`autopay`	Did the customer use the automatic payment option?

Table 2: Variables of the South Korean telecom churn data.

will be explained in more detail in Section 5. Classification tree methods like CART and C4.5, on the other hand, do not depend on a regularization parameter and are heuristic in nature.

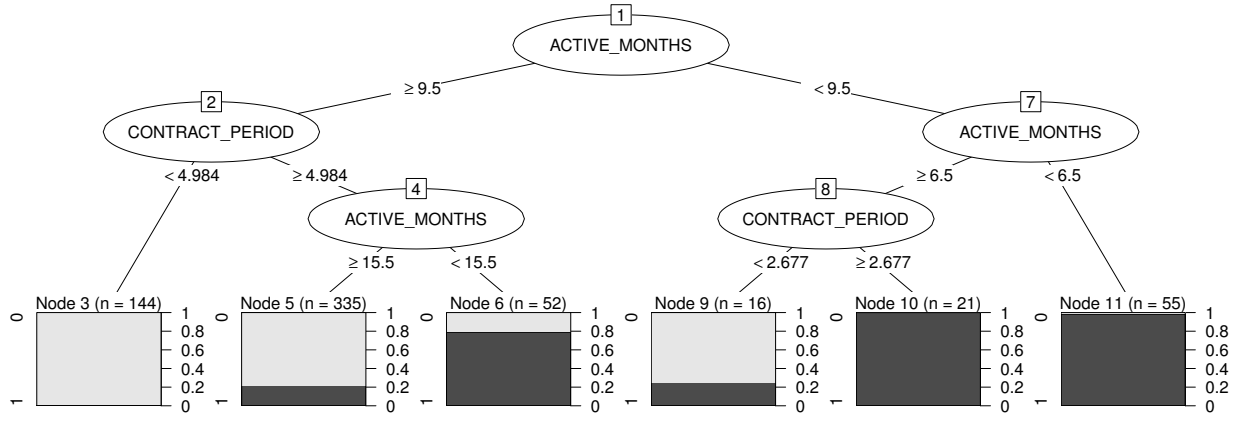
4. Case study

In this section, we will illustrate the benefit of using the ProfTree classifier on a customer churn dataset from a South Korean telecom operator. The dataset contains a sample of 889 customers and 10 explanatory variables (see Table 2); 277 of these customers (i.e. 31.16%) were recorded as churners. The goal is to build a model that predicts would-be churners while taking profitability into account. Moreover, we want the model to be easily interpretable such that the results can be communicated to the marketing department. For the purpose of this case study, we will therefore build a tree model with a maximal depth of three levels of splits. Besides ProfTree, we use other classification tree methods like EvTree, CART and CTree.

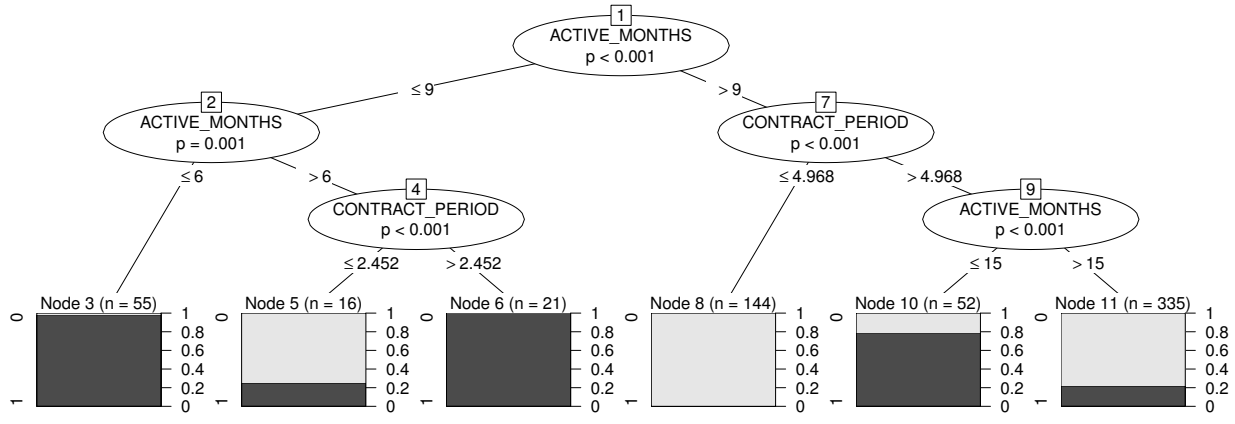
EvTree (Grubinger et al., 2014) uses an evolutionary algorithm for learning globally optimal classification trees. CART (Breiman et al., 1984) is a commonly used classification tree algorithm which uses a greedy heuristic approach, where split rules are selected in a forward stepwise search for recursively partitioning the data into groups.

The CTree algorithm (Hothorn et al., 2006) builds a conditional inference tree. This is a non-parametric class of decision trees which are embedded into a theory of conditional inference procedures (Strasser and Weber, 1999). The CTree algorithm is a recursive partitioning algorithm that,

like CART and C4.5 (Quilan, 1993), searches for the best possible decision tree by considering one split at a time. At each internal node, the split variable and the corresponding split point are selected to locally minimize the loss function. While CART and C4.5 employ an exhaustive search to find the next best split, CTree is an *unbiased* recursive partitioning technique which modifies this search by first selecting the split variable using statistical significance tests and subsequently selecting the optimal split point for that particular variable. This approach avoids selecting variables with many potential splits more often than those with fewer potential splits. In summary, the CTree algorithm works as follows: (1) Test the global null hypothesis of independence between any of the input variables and the response. Stop if this hypothesis cannot be rejected. Otherwise select the input variable with strongest association to the response. (2) Implement a binary split on the selected input variable. (3) Recursively repeat steps 1) and 2).



(a)



(b)

Figure 1: Trees for customer churn prediction constructed by CART (a) and CTree (b).

We randomly partition the dataset into a 70% training set (623 cases) and 30% test set (266 cases), stratified according to the churn indicator to obtain similar churn distributions in the training and test set as observed in the original dataset. All trees are constrained to have a minimum of 10 observations per terminal node, 20 observations per internal node, and a maximum tree depth of 3. Furthermore, the conditional inference tree is constructed with a significance level of 1% rather than the default 5% level since it seems more appropriate for 623 observations. Another well-known recursive partitioning method is C4.5 (Quilan, 1993). However, the results for C4.5 on this case study are not reported because the tree depth cannot be restricted by the method's implementation.

First, we grow the forward-search trees by applying CART and CTree on the training set (Figure 1). Although the dataset contains 10 explanatory variables, CART and CTree only use the variables `active_months` and `contract_period` to predict the churning propensity of the customers. Interestingly, CART and CTree come more or less to the same solution. Despite using slightly different split points for the respective split variables, the six end nodes of both trees have the same churn propensities. For example, according to CART (resp. CTree), a person who is a customer at the telecom provider for less than 6 (resp. 6.5) months, has a churn propensity of 98% as can be seen in Node 11 (resp. Node 3). Similarly, leaf Node 3 of CART corresponds with leaf Node 8 of CTree.

EvTree's evolutionary algorithm contains a user-specified parameter α which plays the same role as the parameter λ within ProfTree as it regulates the complexity of the trees that are grown. In order to find the optimal value for α in EvTree, resp. λ in ProfTree, we apply a search over the grid $\Lambda = \{0.01, 0.05, 0.10, 0.15, \ldots, 0.95, 1.00\}$ in combination with 5×2-fold cross-validation on the training set as described in Section 3.6. The EMPC criterion is hereby used with its default values (as specified in Section 2.3) to select the optimal value for α, resp. λ. EvTree reaches the highest average EMPC of 12.49 at $\alpha_{opt} = 0.20$ (Figure 2a) while ProfTree reaches its highest average EMPC of 12.66 at $\lambda_{opt} = 0.10$ (Figure 2b).

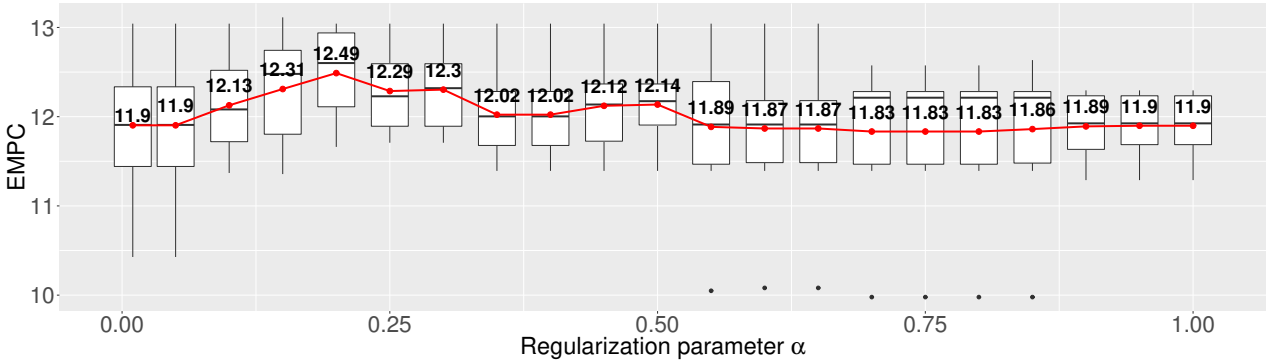
Figure 3 shows the average time needed to train ProfTree on half the training set (i.e. one fold of the 5×2 cross-validation) for the different λ values of the grid. For $\lambda = 0.10$, it requires approximately 33 seconds to fit ProfTree on a set with 312 observations and 10 predictor variables. As λ becomes larger, the penalization on the size of the trees increases. This leads to smaller trees which in return decreases the time to fit ProfTree.

Next, we apply both EvTree and ProfTree once on the entire training set with the respective optimal value of their regularization parameter (Figure 4). EvTree uses the same two variables (`active_months` and `contract_period`) as CART and CTree to predict churners, but each method

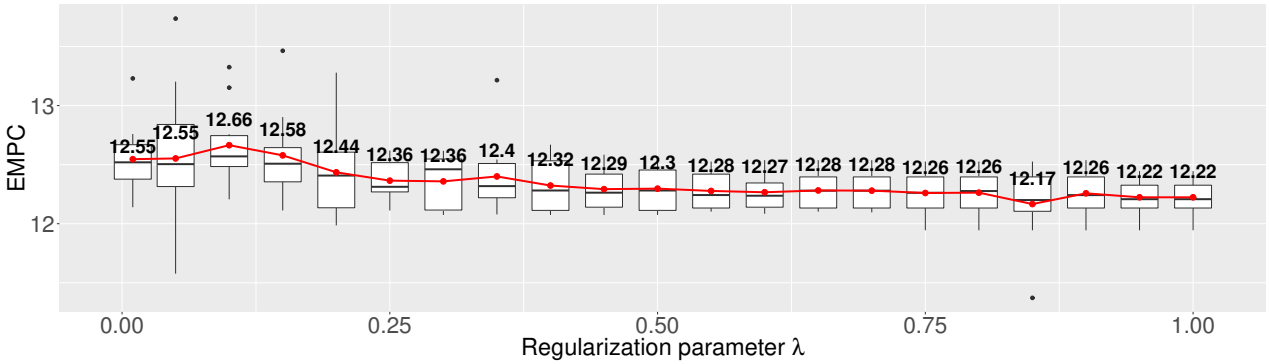
uses different cutoff values. On the other hand, ProfTree incorporates two additional variables (`revenue_avg` and `region`) in its classification tree. Furthermore, EvTree's first split variable is `contract_period` while this variable only occurs in the third level of ProfTree's solution and is only applied on customers that are active between 11 and 20 months. When a customer stays with the telecom operator for 20 months or more, ProfTree employs variables like average revenue and region to determine the customer's likelihood to churn since losing a long-time customer may cause a bigger loss or profit than a recently joined customer.

All trees can easily be interpreted and communicated to the management professionals of the retention campaign. However, we still need to compare their performance, especially their (expected) profitability. The previously trained classification trees are therefore evaluated on the test set. Table 3 contains the results from the different classifiers for various performance measures.

Note that CART and CTree have the same performance on the test set because they predict the same churn propensity for similar customers. ProfTree has the highest EMPC and MPC value



(a)



(b)

Figure 2: Average EMPC performance (red dot) for various values of EvTree's α (a) and ProfTree's λ (b).

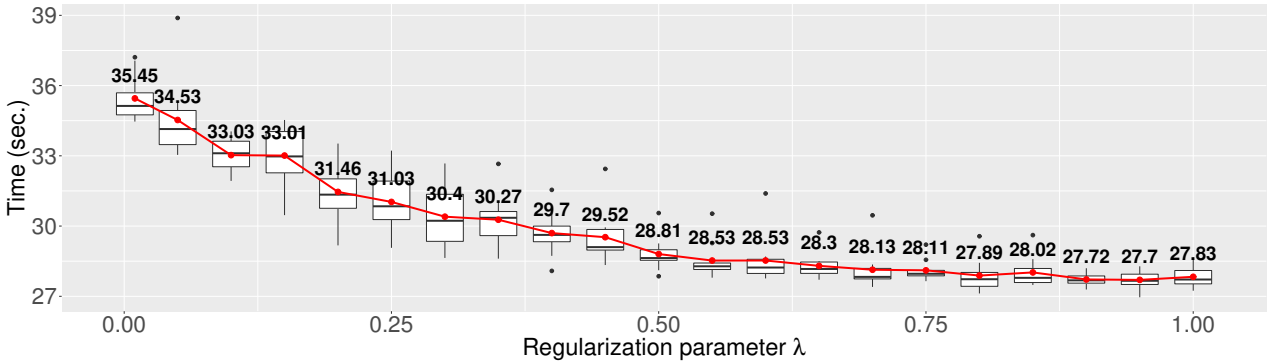
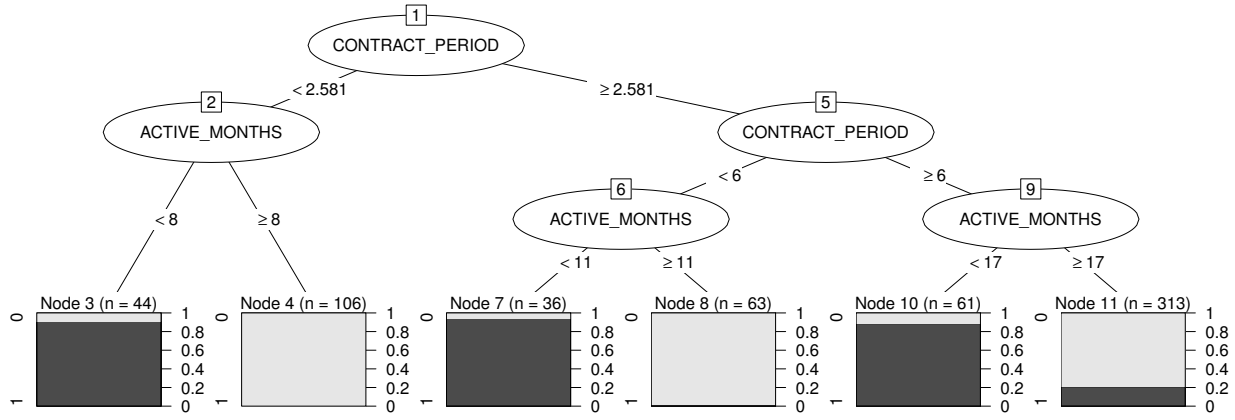
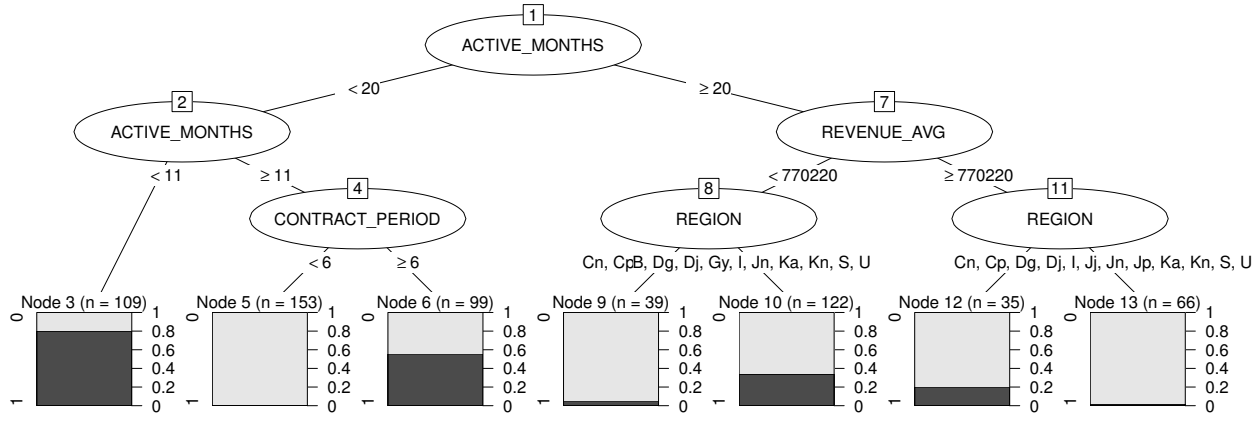


Figure 3: Average computation time (red dot) of ProfTree on half of the training set of the South Korean churn dataset for different values of λ.



(a)



(b)

Figure 4: Trees for customer churn prediction constructed by EvTree (a) and ProfTree (b).

	EMPC	MPC	$\overline{\eta}_p$	$\overline{\eta}_r$	$\overline{\eta}_{F_1}$	AUC	MER
ProfTree	**12.868**	**12.793**	**0.448**	0.940	**0.607**	0.824	0.222
EvTree	12.637	12.553	0.431	**0.976**	0.598	**0.831**	**0.173**
CART	12.126	12.053	0.413	0.976	0.581	0.797	0.195
CTree	12.126	12.053	0.413	0.976	0.581	0.797	0.195

Table 3: Performances of various tree classifiers on the South Korean telecom churn dataset.

among all the tree classifiers. Compared to EvTree, it yields (expected) profit gains of 0.231 EUR per customer. A telecommunication service provider has typically several thousands or even millions of customers. Hence, the potential profit gain could be enormous. EvTree has built the tree with the highest AUC and the lowest MER. However, as this case study shows, this does not imply that EvTree's solution is the most profitable. Furthermore, ProfTree exhibits the highest $\overline{\eta}$-precision (44.8%), meaning it most effectively identifies churners correctly. All tree classifiers have an $\overline{\eta}$-recall that is close to 1, meaning they are capable of detecting almost all churners, while ProfTree has the highest $\overline{\eta}$-based F_1 measure (60.7%). In summary, this case study illustrates how ProfTree can be employed to build a classification tree that balances profitability and complexity by searching a larger space of potential trees.

It is worth mentioning that in this setup, several runs of the ProfTree algorithm with the same parameters typically lead to the same tree. However, this may not always be the case due to the stochastic nature of the search algorithm and the potential vast search space. As a result, trees with different structures but similar fitness function values may be found by subsequent runs of ProfTree. We alleviated this problem in the case study by restricting the maximal tree depth to 3,

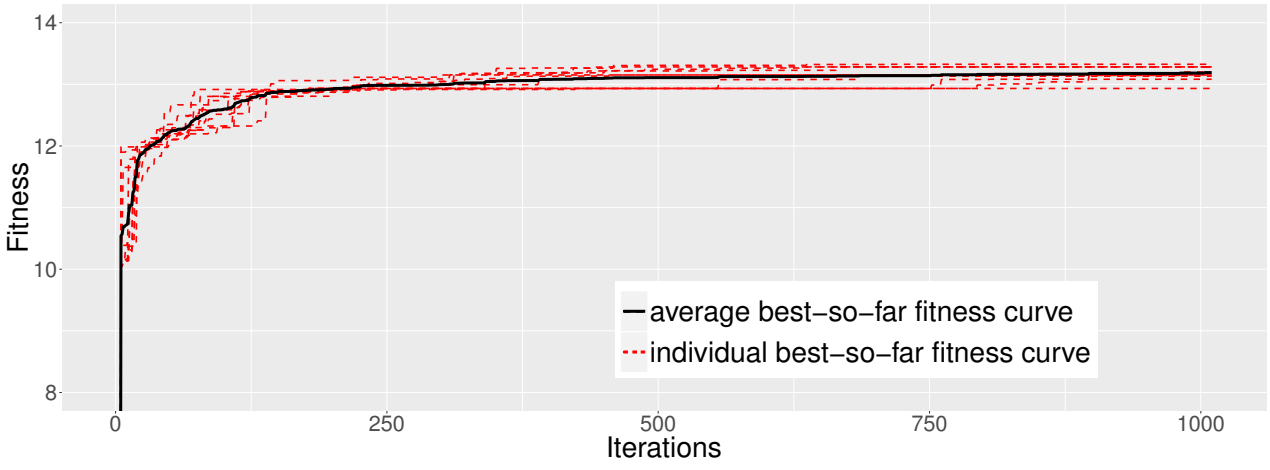


Figure 5: Fitness curve for each of 10 runs of ProfTree on the training set of the Korean churn dataset.

yielding a unique solution. Figure 5 shows the convergence pattern of 10 runs of ProfTree on the training set (70%) of the South Korean churn dataset. At each iteration, the fitness of the fittest tree in the population is shown where fitness is measured by (12). Each of the 10 fitness curves reaches the maximum solution quite fast, since the performance stabilizes already after 250 iterations. Furthermore, each run of ProfTree leads to a slightly different solution, but their corresponding fitness functions are very similar.

5. Empirical evaluation

In this section, we compare ProfTree with EvTree, CART, C4.5 and CTree in a more rigorous context. Both the CART and C4.5 algorithm have the option to include a final, global pruning step to simplify the tree by snipping off the least important splits. We consider both CART and C4.5 with and without this pruning step. We apply the classification algorithms to several real-life churn datasets from various telecommunication service providers. These datasets are either publicly available or have been provided to our research group from various telco operators, located around the world (i.e., North and South America, East Asia, and Europe). The datasets are described in Table 4. The performance of each classifier is evaluated by using EMPC, MPC, AUC, MER, $\overline{\eta}_p$, $\overline{\eta}_r$ and $\overline{\eta}_{F_1}$. Hereby we apply EMPC's default values as specified in Section 2.3. We begin by describing the experimental setup, followed by presenting the results of the benchmark study, and a discussion.

5.1. Experimental setup of the benchmark study

Each dataset is randomly partitioned into a 70% training and 30% test set, stratified according to the churn indicator to obtain similar churn distributions in the training and test set as observed in the original dataset. Only the UCI training and test sets from the UCI machine learning repository (Bache and Lichman, 2013) are already available from the source. When preprocessing the data, the observations (i.e. customers) with missing values are removed before the analysis. No other data preparation steps are applied because the tree-based methods do not require standardization of the continuous predictor variables or transformation of the categorical predictors. Furthermore, we do not have to remove strongly correlated features because multicollinearity among the predictors is not an issue for classification tree methods. The benchmark datasets are described in Table 4. The sample sizes of the training sets range from 623 instances (*Korean1*) to 45,501 instances (*Duke2*). The number of attributes varies between 9 (*Duke1* and *Belg3*) and 48 (*Duke2*). The types of attributes vary among datasets, and include datasets which have both categorical and numerical variables.

To find the optimal regularization parameters λ and α within ProfTree and EvTree, respectively, we apply the procedure discussed in Section 3.6: five replications of 2-fold cross-validation on the training set (Dietterich, 1998; Demšar, 2006). For each dataset, we fit each classifier on the training set and evaluate it on the independent test set to obtain out-of-sample classification performance estimates. Since both ProfTree and EvTree are stochastic in nature, they are trained 50 times on the entire training set with their respective optimal penalization parameters λ_{opt} and α_{opt}, and each time their final performance is evaluated on the test set. As suggested by Grubinger et al. (2014), ProfTree, EvTree, CART and CTree models are constrained to a minimum number of 20 observations per internal node, 7 observations per terminal node, and a maximum tree depth of 9. Apart from that, the default settings of the algorithms are used. C4.5 is only constrained to a minimum number of 7 samples per terminal node, since other restrictions are available in this implementation. Both CART and C4.5 models are considered with and without the pruning step. Note that we apply EMPC's default values as specified in Section 2.3.

5.2. Results of the experiment

Figure 6 shows the boxplot of the 50 EMPC values of ProfTree for each churn dataset. We compare the average EMPC of ProfTree (⧫) with the one from the respective best competitive classifier (▲). In order to avoid overlapping labels and obscuring the view of the estimates, we only show the result from the respective best competitive technique. When averaging the ranks over the datasets, ProfTree has the overall best performance in terms of EMPC and MPC (Figure 7). Our new method occupies the first place in six out of nine datasets in terms of EMPC, resulting in

ID	**Instances**		**Churn rate [%]**		**Attributes**		
	Training set	Test set	Training set	Test set	Binary	Nominal	Metric
Belg1	40,678	17,433	13.96	13.96	1	10	13
Belg2	39,404	16,887	11.18	11.17	1	10	13
Belg3	2,589	1,109	13.29	13.26	1	-	8
Chile	4,940	2,116	29.15	29.11	2	-	35
Duke1	8,750	3,749	39.31	39.32	-	1	8
Duke2	45,501	19,501	49.56	49.56	1	-	47
Korean1	9,522	4,079	22.59	22.58	1	3	10
Korean2	623	266	31.14	31.20	1	2	7
UCI	3,333	1,667	14.49	13.44	2	2	15

Table 4: Description of the churn datasets: number of instances, churn rate, number of types of attributes.

an average rank of 1.56 on a scale from 1 to 7. ProfTree delivers in 6 out of 9 datasets the most profitable churn model, and in 4 out of the 6 best cases the outperformance is significant. For the other datasest, ProfTree falls closely behind the best competitive method. The profit gains range from 0.01 to 0.18€ per customer. In the case of the *Belg2* dataset, ProfTree's EMPC estimates range from 0.89 to 1.08€ per customer, and its average performance equals $\overline{\text{EMPC}} = 0.95 \pm 0.04$€. This is on average 10.5% better than the respective best competitive model (CTree). In the worst case (*Chile*), ProfTree's EMPC estimates range from 9.07 to 9.46€, and its average performance equals $\overline{\text{EMPC}} = 9.29 \pm 0.10$€. This is on average 1.4% worse than the respective best competitive classifier (CTree). When measuring the MPC performance, ProfTree has an average rank of 1.44, being the best churn model in 7 out of 9 datasets (Figure 7).

Furthermore, ProfTree exhibits the third highest $\overline{\eta}$-precision (average rank: 3.44). It also has the

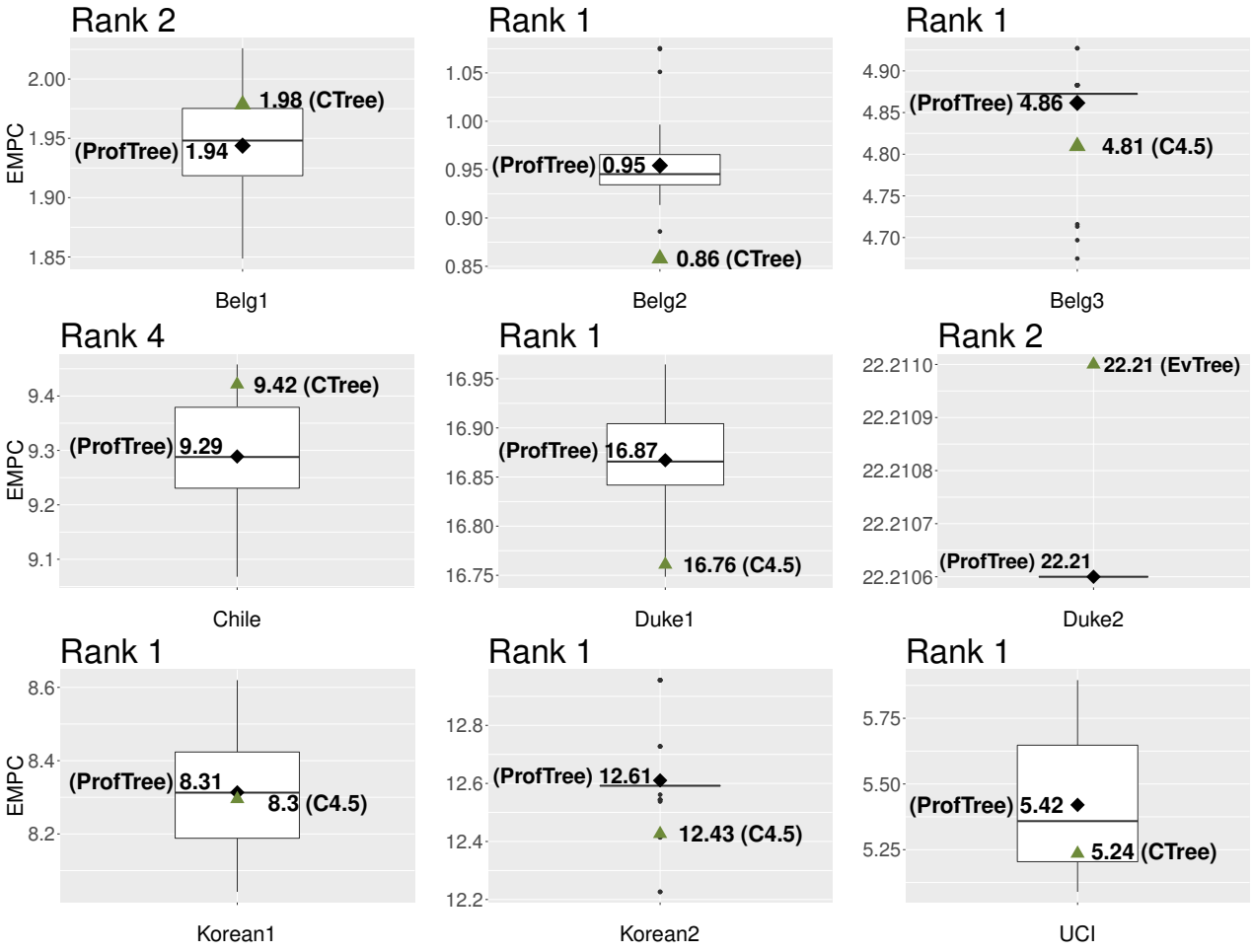


Figure 6: The boxplots are based on 50 EMPC estimates of ProfTree measured on the test set. ⧫: ProfTree; ▲: best competitive classifier respective to the dataset.

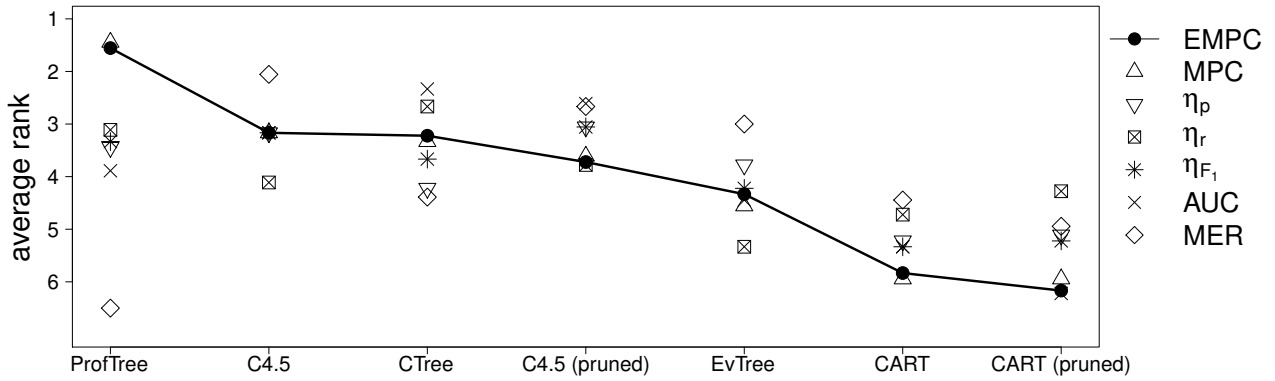


Figure 7: Average rank of the classifier over the different datasets for various performance metrics.

second highest average $\overline{\eta}$-recall (average rank: 3.11), thus it is well capable of detecing the would-be churners. When considering the $\overline{\eta}$-based F_1 measure, ProfTree is the third best performing classifier with an average rank of 3.33, being the best churn model in three out of nine datasets. While ProfTree has the best performances in terms of EMPC and MPC, ProfTree has the overall poorest MER performance, having in six out of nine datasets a rank of 7 (average rank: 6.5). In contrast, the MER performance for all other tree classifiers, except CTree, is better than the profit-based EMPC measure. This illustrates that the best accuracy-based model is not necessarily the most profitable. Because ProfTree maximizes EMPC, rather than minimizing the misclassification error, significant discrepancies between these two measures were expected. Similarly, in the case of AUC performance, ProfTree is ranked quite low with an average rank of 3.89.

Computational aspects

Table 5 shows the average time it took to train ProfTree and EvTree on each of the datasets. The required sources were measured on an Intel core i5 with 2.7 GHz and 8 GB RAM. The smallest of the datasets, *Korean2*, needed less than 1 minute of training time for both ProfTree and EvTree. The longest time it took to fit ProfTee was 75 minutes on the dataset *Belg1*, while it took more than 4 hours to train EvTree on the largest dataset *Duke2*. Although ProfTree has a similar complexity as the approach by Grubinger et al. (2014), we acknowledge that the training times of both evolutionary algorithms (EA) are rather slow compared to other classification algorithms.

The recorded times to fit the EA depend heavily on the size of the trees that are built by the EA. The size of the trees in each generation is regularized by the parameter λ in ProfTree, resp. α in EvTree. A smaller value for the regularization parameter results in a softer penalty on the size of

the trees, which means that trees can grow larger. This increases the parameter space Θ in (4) which makes it harder to find the tree with the highest fitness. Figure 3 illustrates how the regularization parameter λ has an impact on the computation time.

		Belg1	Belg2	Belg3	Chile	Duke1	Duke2	Korean1	Korean2	UCI
Time	ProfTree	75.72	55.73	2.99	6.31	12.54	57.36	16.00	0.92	4.73
	EvTree	28.16	15.83	3.38	3.94	63.02	373.69	25.36	0.54	6.82

Table 5: Average time (in minutes) to fit ProfTree and EvTree on the respective training set, based on 50 estimates.

5.3. Discussion

ProfTree achieves, on average, significant profit gains compared to the competitive techniques. The benchmark study shows that ProfTree is the overall most profitable classifier in terms of EMPC and MPC. The benchmark studies further illustrate that model selection purely based on accuracy related performance measures, such as AUC and MER, likely results in considerable less profitable models. This is clearly demonstrated by the fact that ProfTree, which maximizes profit in its construction step, is the overall most profitable classifier but simultaneously has the worst MER values (see Figure 7). Interestingly, ProfTree exhibits the third highest $\overline{\eta}$-precision despite the fact that the $\overline{\eta}$-based performance measures ($\overline{\eta}_p$, $\overline{\eta}_r$ and $\overline{\eta}_{F_1}$) are related to the notion of accuracy. This is a desirable property of ProfTree because, although a company's primary objective is to maximize profits, another important requirement is that the marketers have as many true would-be churners on their target list as possible. In other words, ProfTree is an effective classifier to correctly identify churners, which allows companies to not only focus their marketing resources on the customers that intend to churn but also to focus on those who are the most profitable to the company. In addition to high precision, ProfTee exhibits the second highest $\overline{\eta}$-recall, meaning it can detect a large proportion of would-be churners. Furthermore, all $\overline{\eta}$-based performance metrics are optimized for maximum profit, which means that churn management campaigns are able to focus their efforts primarily on customers that are profitable to the company.

6. Conclusions and future work

In this paper, we presented a new churn classification method called ProfTree[1] that uses an evolutionary algorithm to directly optimize the EMPC (Verbraken et al., 2013) in the model construction

[1]R code implementation of ProfTree will be made available at github.com/SebastiaanHoppner/ProfTree

step of a decision tree. As a result, ProfTree aims to actively construct the most profitable model for a customer retention campaign. We exploit an evolutionary algorithm for learning profit driven decision trees according to the regularized EMPC fitness function (12). One major benefit of using decision trees as the underlying classifier is that the model can be easily interpreted which helps to understand why customers defect.

In our benchmark study, ProfTree is the overall most profitable model compared to 6 other tree-based methods. The study consisted of applying the classifiers to 9 real-life churn datasets, and evaluated their out-of-sample classification performance using accuracy, cost, and profit related performance measures. We conclude that model selection based on accuracy, like MER or AUC, leads to less profitable results. In almost all cases, ProfTree outperforms its competitors, leading to significantly higher profits; whereas, in the worst case, its profit losses are relatively small compared to the respective best competitive model.

Furthermore, the benchmark study shows that ProfTree has a high hit rate and recall rate, which makes it an effective model to both correctly identify churners as well as to detect the largest proportion of would-be churners. In this paper, we have shown that our proposed method aligns best with the core business requirement of profit maximization. Moreover, ProfTree produces customer lists with high $\overline{\eta}$-precision which can be used in the retention campaign for targeting the most profitable potential churners.

Concerning future research, we intend to combine the ProfTree algorithm with random forests (Breiman, 2001). This method, called ProfForest, aims at further improving the profit maximizing property by building a large collection of profit induced trees, and then aggregating them. An extensive empirical evaluation and comparison between the profit-based methods ProfTree, ProfForest and the approach of Stripling et al. (2018) can then be conducted. Although ProfTree has a similar complexity as the approach by Grubinger et al. (2014), we acknowledge that the training times of the evolutionary algorithm (EA) are rather slow compared to other classification algorithms. As future work, we intend to replace the EA with efficient heuristics like the ones used in CART Breiman et al. (1984) and C4.5 Quilan (1993) in order to construct fast profit-based trees. Besides using the ProfTree algorithm for churn prediction, it also has potential in other analytical tasks where profit measures can be used (Elkan, 2001), such as credit scoring (Verbraken et al., 2014) and fraud detection. The main adaptation of ProfTree to these tasks would consist of replacing the expected maximum profit measure for customer churn (EMPC) by the appropriate metric.

Acknowledgements

This work was supported by the BNP Paribas Fortis Chair in Fraud Analytics and Internal Funds KU Leuven under Grants C16/15/068 and C24/15/001.

References

Bache, K., Lichman, M., 2013. Uci machine learning repository (http://archive.ics.uci.edu/ml), university of california, school of information and computer science. Irvine, CA.

Bahnsen, A. C., Aouada, D., Ottersten, B., 2015a. Example-dependent cost-sensitive decision trees. Expert Systems with Applications 42 (19), 6609–6619.

Bahnsen, A. C., Aouada, D., Ottersten, B., 2015b. A novel cost-sensitive framework for customer churn predictive modeling. Decision Analytics 2 (1), 1–15.

Breiman, L., 2001. Random forests. Machine learning 45 (1), 5–32.

Breiman, L., Friedman, J., Stone, C. J., Olshen, R. A., 1984. Classification and regression trees. CRC press.

Carrano, E. G., Fonseca, C. M., Takahashi, R. H., Pimenta, L. C., Neto, O. M., 2007. A preliminary comparison of tree encoding schemes for evolutionary algorithms. In: Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on. IEEE, pp. 1969–1974.

Chen, Z.-Y., Fan, Z.-P., Sun, M., 2012. A hierarchical multiple kernel support vector machine for customer churn prediction using longitudinal behavioral data. European Journal of operational research 223 (2), 461–472.

Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. Journal of Machine learning research 7 (Jan), 1–30.

Dietterich, T. G., 1998. Approximate statistical tests for comparing supervised classification learning algorithms. Neural computation 10 (7), 1895–1923.

Eiben, A. E., Smith, J. E., 2015. Introduction to evolutionary computing, 2nd Edition. Springer.

Elkan, C., 2001. The foundations of cost-sensitive learning. In: International joint conference on artificial intelligence. Vol. 17. Lawrence Erlbaum Associates Ltd, pp. 973–978.

Fogel, L. J., Owens, A. J., Walsh, M. J., 1966. Artifical Intelligence through simulated evolution. John Wiley & Sons, New York.

Freitas, A. A., 2003. A survey of evolutionary algorithms for data mining and knowledge discovery. In: Advances in Evolutionary Computing. Springer, pp. 819–845.

Glady, N., Baesens, B., Croux, C., 2009. Modeling churn using customer lifetime value. European Journal of Operational Research 197 (1), 402–411.

Grubinger, T., Zeileis, A., Pfeiffer, K.-P., 2014. evtree: Evolutionary learning of globally optimal classification and regression trees in R. Journal of Statistical Software 61 (1), 1–29.

Hand, D. J., 2009. Measuring classifier performance: a coherent alternative to the area under the roc curve. Machine learning 77 (1), 103–123.

Holland, J. H., 1992. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. MIT press.

Hothorn, T., Hornik, K., Zeileis, A., 2006. Unbiased recursive partitioning: A conditional inference framework. Journal of Computational and Graphical statistics 15 (3), 651–674.

Hyafil, L., Rivest, R. L., 1976. Constructing optimal binary decision trees is np-complete. Information processing letters 5 (1), 15–17.

Jankowski, D., Jackowski, K., 2014. Evolutionary algorithm for decision tree induction. In: IFIP International Conference on Computer Information Systems and Industrial Management. Springer, pp. 23–32.

Koza, J. R., 1992. Genetic Programming: On the programming of computers by means of natural selection. MIT press, Cambridge.

Lin, L., Gen, M., 2006. Node-based genetic algorithm for communication spanning tree problem. IEICE transactions on communications 89 (4), 1091–1098.

Palmer, C. C., Kershenbaum, A., 1994. Representing trees in genetic algorithms. In: Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on. IEEE, pp. 379–384.

Palmer, C. C., Kershenbaum, A., 1995. An approach to a problem in network design using genetic algorithms. Networks 26 (3), 151–163.

Quilan, J. R., 1993. C4.5: Programs for machine learning. Morgan Kaufmann Publichers, San Mateo.

Raidl, G. R., Julstrom, B. A., 2003. Edge sets: an effective evolutionary coding of spanning trees. IEEE Transactions on evolutionary computation 7 (3), 225–239.

Soak, S.-M., Corne, D. W., Ahn, B.-H., 2006. The edge-window-decoder representation for tree-based problems. IEEE Transactions on Evolutionary Computation 10 (2), 124–144.

Strasser, H., Weber, C., 1999. On the asymptotic theory of permutation statistics. Mathematical Methods of Statistics 8, 220–250.

Stripling, E., vanden Broucke, S., Antonio, K., Baesens, B., Snoeck, M., 2015. Profit maximizing logistic regression modeling for customer churn prediction. IEEE International Conference on Data Science and Advanced Analytics (DSAA' 2015). Paris (France), 19-21 October 2015 (pp. 1-10). Paris, France: IEEE.

Stripling, E., vanden Broucke, S., Antonio, K., Baesens, B., Snoeck, M., 2018. Profit maximizing logistic model for customer churn prediction using genetic algorithms. Swarm and Evolutionary Computation 40, 116–130.

Van Wezel, M., Potharst, R., 2007. Improved customer choice predictions using ensemble methods. European Journal of Operational Research 181 (1), 436–452.

Verbeke, W., Dejaeger, K., Martens, D., Hur, J., Baesens, B., 2012. New insights into churn prediction in the telecommunication sector: A profit driven data mining approach. European Journal of Operational Research 218 (1), 211–229.

Verbeke, W., Martens, D., Mues, C., Baesens, B., 2011. Building comprehensible customer churn prediction models with advanced rule induction techniques. Expert Systems with Applications 38 (3), 2354–2364.

Verbraken, T., Bravo, C., Weber, R., Baesens, B., 2014. Development and application of consumer credit scoring models using profit-based classification measures. European Journal of Operational Research 238 (2), 505–513.

Verbraken, T., Verbeke, W., Baesens, B., 2013. A novel profit maximizing metric for measuring classification performance of customer churn prediction models. IEEE Transactions on Knowledge and Data Engineering 25 (5), 961–973.