DEGREE PROJECT IN MATHEMATICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2019

Customer Churn Prediction for PC Games

Probability of churn predicted for big-spenders using supervised machine learning

VALGERDUR TRYGGVADOTTIR

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ARCHITECTURE AND THE BUILT ENVIRONMENT

Customer Churn Prediction for PC Games

Probability of churn predicted for big-spenders using supervised machine learning

VALGERDUR TRYGGVADOTTIR

Degree Projects in Optimization and Systems Theory (30 ECTS credits)
Degree Programme in Applied and Computational Mathematics (120 credits)
KTH Royal Institute of Technology year 2019
Supervisor at Echo State: Josef Falk
Supervisor at KTH: Xiaoming Hu
Examiner at KTH: Xiaoming Hu

TRITA-SCI-GRU 2019:254
MAT-E 2019:68

Royal Institute of Technology
School of Engineering Sciences
KTH SCI
SE-100 44 Stockholm, Sweden
URL: www.kth.se/sci

Abstract

Paradox Interactive is a Swedish video game developer and publisher which has players all around the world. Paradox's largest platform in terms of amount of players and revenue is the PC. The goal of this thesis was to make a churn prediction model to predict the probability of players churning in order to know which players to focus on in retention campaigns. Since the purpose of churn prediction is to minimize loss due to customers churning the focus was on big-spenders (whales) in Paradox PC games.

In order to define which players are big-spenders the spending for players over a 12 month rolling period (from 2016-01-01 until 2018-12-31) was investigated. The players spending more than the 95th-percentile of the total spending for each period were defined as whales. Defining when a whale has churned, i.e. stopped being a big-spender in Paradox PC games, was done by looking at how many days had passed since the players bought something. A whale has churned if he has not bought anything for the past 28 days.

When data had been collected about the whales the data set was prepared for a number of different supervised machine learning methods. Logistic Regression, L1 Regularized Logistic Regression, Decision Tree and Random Forest were the methods tested. Random Forest performed best in terms of AUC, with $AUC = 0.7162$. The conclusion is that it seems to be possible to predict the probability of churning for Paradox whales. It might be possible to improve the model further by investigating more data and fine tuning the definition of churn.

Keywords: Customer churn prediction, whales, data analysis, machine learning, binary classification.

Sammanfattning

Paradox Interactive är en svensk videospelutvecklare och utgivare som har spelare över hela världen. Paradox största plattform när det gäller antal spelare och intäkter är PC:n. Målet med detta exjobb var att göra en churn-predikterings modell för att förutsäga sannolikheten för att spelare har "churnat" för att veta vilka spelare fokusen ska vara på i retentionskampanjer. Eftersom syftet med churn-prediktering är att minimera förlust på grund av kunderna som "churnar", var fokusen på spelare som spenderar mest pengar (valar) i Paradox PC-spel.

För att definiera vilka spelare som är valar undersöktes hur mycket spelarna spenderar under en 12 månaders rullande period (från 2016-01-01 till 2018-12-31). Spelarna som spenderade mer än 95:e percentilen av den totala spenderingen för varje period definierades som valar. För att definiera när en val har "churnat", det vill säga slutat vara en kund som spenderar mycket pengar i Paradox PC-spel, tittade man på hur många dagar som gått sedan spelarna köpte någonting. En val har "churnat" om han inte har köpt något under de senaste 28 dagarna.

När data hade varit samlad om valarna var datan förberedd för ett antal olika maskininlärningsmetoder. Logistic Regression, L1 Regularized Logistic Regression, Decision Tree och Random Forest var de metoder som testades. Random Forest var den metoden som gav bäst resultat med avseende på AUC, med $AUC = 0,7162$. Slutsatsen är att det verkar vara möjligt att förutsäga sannolikheten att Paradox valar "churnar". Det kan vara möjligt att förbättra modellen ytterligare genom att undersöka mer data och finjustera definitionen av churn.

Nyckelord: Kund churn prediktering, valar, dataanalys, maskinlärning, binär klassificering.

Preface

This master thesis was completed in the Department of Mathematics which is a part of the School of Engineerings Sciences at KTH Royal Institute of Technology. The thesis was done in collaboration with the consulting company Echo State and the video game developer and publisher Paradox Interactive.

Acknowledgement

I would like to thank Christoffer Fluch for contacting me on LinkedIn and together with Josef Falk giving me the opportunity to do this project. I want to thank Josef and Markus Ebbesson for being my supervisors at Echo State and giving me inspiration and helping me with their technical knowledge. I am also greatful for the warm welcome by all the others working at Echo State.

At Paradox Interactive I would like to thank Mathias Von Plato for the guidance and feedback throughout the project, Anabel Silva Rojas for the help with collecting and understanding the data, Alexander Hofverberg for helping me with his knowledge in Machine Learning and RStudio and Alessandro Festante for his help from the beginning of the project. In addition I thank all the others from the data and analytics teams at Paradox for creating a friendly and good work atmosphere.

I would like to thank to Xiaoming Hu, my supervisor and examinor at KTH, for his generous support. I owe all the teachers I have had throughout my education my sincerest gratitude for helping me getting the knowledge I have today.

Finally, I want to thank my family, boyfriend, Björgvin R. Hjálmarsson, roomate, Laufey Benediktsdóttir and friends for always being supportive and believing in me.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ANN	Artificial Neural Network
AUC	Area Under the ROC Curve
CRM	Customer Relationship Management
CV	Cross-validation
DLC	Downloadable Content
DT	Decision Tree
FN	False Negative
FP	False Positive
FPR	False Positive Rate
MSE	Mean Squared Error
OOB	Out-of-bag
PCA	Principal Component Analysis
RF	Random Forest
RFM	Recency, Frequency, Monetary
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
TN	True Negative
TNR	True Negative Rate
TP	True Positive

Chapter 1

Introduction

Nowadays, data is collected and created continuously and hence a huge amount of data is available. In the gaming industry telemetry data of player behavior is collected and this makes it possible to monitor how each player plays a game. In order to develop a profitable game it is important to analyze this data since there are many games on the market competing for players [1]. This can be done by using data mining to investigate CRM (*Customer Relationship Management*) objectives. The aim of CRM is to get profitable customers, keep them happy so they don't leave, and if they do leave then devise ways to get them back. These objectives may by summarized as (a) acquisition, (b) retention, (c) churn minimization and (d) win-back [2]. It has been shown that retaining existing customers is more cost-efficient than acquiring new users [3]. Churn prediction is therefore important in order to know which customers to target in retention campaigns.

The Swedish video game developer and publisher, Paradox Interactive, has a lot of available data about their users which they want to use to predict churn. The term *churn* means that a player has left the game permanently, i.e. the player is not a customer anymore. *Churn prediction* is used to find the probability of a player churning [4]. This makes it possible to extend the player's lifetime in a game, i.e. prevent the user from leaving.

1.1 Thesis objectives

The purpose of churn prediction is to minimize loss due to customers churning and consequently the focus should be on players who increase profits for the company when they are retained. This means it is more cost-efficient to target paying players rather than all users when predicting churn [5].

So-called *whales* are big-spenders and are the group of players which spend the most money on games. Therefore, "whale watching" is important for Paradox Interactive since the company doesn't want to lose their most valuable customers [1]. The purpose of this thesis is to answer the following research questions:

- Question 1:
  What is a good way to define which players are Paradox whales, i.e. players that are big spenders in Paradox PC games?

- Question 2:
  What is a good way to define if a Paradox whale has churned or not?

- Question 3:
  Which supervised machine learning model gives the best performance in predicting the probability of a whale churning?

The goal of this work is that Paradox Interactive can use the model to identify big spenders and see which ones of them are most likely to churn. This would help the company focus its customer retention efforts on the customers that are from a business perspective most important to retain.

1.2 Thesis disposition

The thesis will be structured as follows: Chapter 2 has a short description of Paradox Interactive's background. Chapter 3 follows with a literature review where related works are discussed. In addition, definitions of whales, churn and the data and algorithms commonly used for churn prediction are presented. In Chapter 4 the theory behind the model is described in detail. Chapter 5 goes through the method used for the model. The results are shown in Chapter 6. And finally Chapter 7 discusses the results and presents the main conclusions.

1.3 Programming Languages

The data needed for this thesis is kept in a cloud-built data warehouse. *SQL* queries were used to collect data from the data warehouse. *RStudio* was used for data analysis and cleaning as well as for programming the churn prediction model.

Chapter 2

Background

Paradox Interactive is a Swedish company which publishes video strategy games and has players all around the world [6]. The company was established in 2004 but its history goes back to 1998. It all started in a company called Target games which was a board game company based in Sweden. Later this company became Paradox Entertainment which finally became Paradox Interactive [7]. The company focuses mainly on the PC and console platform but have also released games on mobile [8].

2.1 Organization

The organization of Paradox Interactive is split into three fields: studios, publishing and White Wolf [9].

- Paradox has five internal development studios which are located in Stockholm and Umeå in Sweden, Delft in the Netherlands and Seattle and Berkeley in the USA. In addition there is a mobile development team in Malmö. [8].
- The publishing section publishes titles that are developed internally, by Paradox Studios, as well as titles developed by independent studios [7].
- In 2015 Paradox Interactive bought White Wolf Publishing which has developed games for 25 years. White Wolf also develops books, card games and TV series and focuses on connecting the product releases to each other [7, 10].

2.2 Revenue

The largest part of Paradox‘s revenues come from selling games. Games are sold through digital distributors such as *Steam*, *App Store* and *Google Play* as well as through Paradox's website (www.paradoxplaza.com). When a game is sold through a digital distributor the amount paid by the user is divided between Paradox and the distributor but when sold through the company's website Paradox receives the full amount [11].

The revenue from games can be divided into four categories [11]:

- *One-time payment*: The payment when players buy the game for the first time, i.e. *base game.*
- *Add-ons*: Additional content, such as upgrades, new worlds, equipment or music, for games that are already released.
- *Expansion packs*: Updated versions or extensions of an existing game.
- *Licenses*: Third parties can get the right to develop new products for certain brands.

Add-ons and expansion packs are also called *downloadable content* (DLC).

The revenue for 2018 was 1,127.7 million SEK and is mostly attributed to the games *Cities: Skylines*, *Crusader Kings II*, *Europa Universalis IV*, *Hearts of Iron* and *Stellaris.* This is a 39% increase from the year 2017. Currently, the biggest markets for Paradox games are in the USA, UK, China, Germany, France, Russia and Scandinavia. [8].

2.3 Games

Paradox's games portfolio has a broad range of brands with more than 100 titles, which fall into three categories: strategy, management and role-playing games [8, 12]. This makes Paradox special compared to other gaming companies since they can spread their risks among different projects. Most of the revenue from 2018 came from established games, i.e. base games and DLCs for games that were released in the years before. Therefore, the risks for revenue and profit fluctuations over time and reliance on new game releases are reduced [8].

All of the games published and developed by Paradox Interactive have the following things in common. The games are re-playable and hence have *sandbox* environments which makes each game session unique. They are intellectually challenging but still accessible as well as encouraging curiosity. In addition there should always be more to discover about the games' subjects behind the scenes. Finally, the gameplay is complemented by visuals, not the other way around [13].

The most important brands include *Age of Wonders*, *Cities: Skylines*, *Crusader Kings*, *Europa Universalis*, *Stellaris*, *Hearts of Iron*, *Magicka*, *Prison Architect* and the *World of Darkness* catalogue of brands [8].

2.4 Players

Most of Paradox's players come from western Europe or the USA and every month over three million players play a Paradox game. Paradox's largest platform in terms of amount of players and revenue is the PC [8].

Paradox interacts with their players through different channels each day. The company has five YouTube channels and over two million followers on social media. In addition Paradox has developed their own forum where players can express their opinions and discuss about the games. Every month there are more than 375,000 active users on the forum. The players therefore have an important role when it comes to developing products [8].

Players that play Paradox games don't need an account to play. However, a Paradox account is needed in order to post on the forum. There are many benefits in having an account, such as getting special deals and occasionally getting a game or DLC for free [14]. In the year 2017 there were over seven million players with a Paradox account but in 2018 this number increased to over nine million players. These players spend more time playing games and more money on expansions and new games than those who don't have an account on average [8].

Chapter 3

Literature Review

In this chapter there will first be an overview of previous works related to this thesis. Churn predictions have been studied in industries such as mobile telecommunications, insurance and banking as well as in the gaming industry [15]. The performance of a model predicting churn depends on both the quality of the data and which learning algorithm is chosen. Hence, the related studies about churn prediction focus on these terms [16]. In addition some studies focus more on how the target group and churn is defined. Therefore a further investigation of common definitions of whales and churn will be done next. Finally the data and methods that are commonly used for churn prediction will be stated.

3.1 Related Works

The expected profit from preventing churn for online games was considered in [5]. According to this paper the focus should not only be on maximizing accuracy when making a churn prediction model but also on maximizing the profits expected from preventing churn. The results showed that it is more cost-efficient to focus on loyal customers than all the players in a churn prediction. Additionally, the study shows that social relations influences churn and that the weekly playtime of churners typically starts to decrease around ten weeks before they churn.

In [17], churn was predicted for high-value players in free-to-play casual social games. The purpose of this article was twofold, to predict churn for high-value players and to investigate the business impact of the model. For churn prediction *Artificial Neural Network* (ANN) performed best in terms of AUC (*area under the ROC curve*). *A/B testing* was used to measure the business impact where in-game currency was given for free. The results showed however, that this did not have a noticeable impact on churn rates.

A similar approach was used in [4]. The focus was on early churn prediction since most of the players in social online games leave the game in early days. In addition churn prevention was investigated. Only data from the players' first day in the game was used which made the prediction challenging. The classification algorithm that performed best in terms of AUC and F1 score for churn prediction was *Gradient Boosting*. The churn prevention was done by sending personalized push notifications with information about the game. There was no cost associated with this step in contrast to what was done in [17]. The result showed that by using this method churn can be reduced up to 28%.

In [18] *Survival Analysis* was used to predict churn for whales. Usually binary classification is used to tackle this problem but the drawback is that these methods can not predict the exact time when players churn. However, with Survival Analysis the output is the probability of churn as a function of time and the focus is on when the churn event happens. The results show that by using survival ensembles the accuracy of churn prediction improves.

The data available in online games is often only data about the user behaviour. Login records are therefore something that publishers can always rely on having from all games. RFM (*recency, frequency, monetary*) analysis which is a simple time series' feature representation can be used to make predictions only from login data, but it has its drawbacks. In this article a *frequency analysis* approach for feature representation from login records is used to predict churn. The results showed that it is possible to increase profits from retention campaigns 20% more than if RFM was used [19].

In many industries churners are a small proportion of the customers and hence churn is a rare event. This causes the data set to have an imbalanced class distribution which is a problem for classification methods used to predict churn. In [16] state-of-the-art techniques to handle class imbalance in churn predictions were compared.

3.2 Churn Definitions

In markets such as telecommunication there is a contractual relationship with the customer which makes the definition of churn a well-defined event since the customers cancel the service when they want to leave permanently [19]. In these industries late customer churn is usually what is investigated [4].

In the gaming industry the churn event is more difficult to define. Most of the literature about churn prediction in gaming is about free-to-play games, both mobile and online games. In these games players don't need to pay anything to start playing but they have the possibility to make in-app purchases. In free-to-play games there is not a contractual relationship with the customer like in other industries as mentioned before. This makes it easy for the players to leave the game [17]. New players in these games usually leave the game in early days. Hence, early churn prediction is important for free-to-play games[4].

In the gaming industry a player is usually defined as a churner according to a certain inactivity time. Deciding how long this inactivity period can be before a player will be defined as a churner is a difficult task. If the period is too long then some players that have churned will be mis-classified as not churned. This will increase false negatives (FN) and will have the effect that players that could have been retained will be lost and this will decrease profits. If the period is too short players that have not churned will be mis-classified as churners, that is false positives (FP) will increase which results in increased costs. [5].

The definition of churn varies a lot between different fields and also depends on the goal and data available each time. According to [17] and [4] players in free-to-play games have churned if they have not played the game for 14 days. For free-to-play mobile social games a player has churned if he has not connected to the game for 10 days in a row [18]. For loyal customers, in online games, a user will be defined as a churner if he does not buy an update within a certain time. It is assumed that he is not interested in the game anymore and has left it [5]. To define churn in [17] the distribution of inactivity days, that is days between logins, was investigated for high-value players.

3.3 Whale Definitions

In [18] the target group was *whales*, i.e. top spenders. There are three reasons for why they focus on this group when predicting churn for mobile social games. The first reason is that they don't behave like the other players. According to this article, whales play almost every day and are therefore usually the most active players. Hence, by looking at their inactivity time it is possible to define these players as churned if they have been inactive for a certain period of time. The second reason is that there is more data available about their activity since they are active so often. Therefore, whales are more likely to stay in the game when something has been done in order to retain them. The last reason is that 50% of the in-app purchases revenues come from their spending although whales are only 10% of paying customers.

According to [17] whales, or high-value-players, in free-to-play casual social games are defined as the players that have spent more than the 90-th percentile of total revenue the last 90 days. This means that these players are in the top 10% of paying players according to their spending over the last 90 days.

In [5], for free-to-play casual social online games, loyalty grades were assigned to players according to how much they spend and play a game each week in order to define which players were long-term loyal customers. For a thirty week period the grade change for each player was monitored and the users that maintained a high grade were defined as loyal customers.

3.4 Algorithms used for Churn Modeling

According to published studies, churn prediction is usually modeled as a binary classification problem. Unfortunately, there seems to be no algorithm that outperforms the others for churn predictions in general [16].

According to [15] the five classification methods that are used most in the telecommunication industry to predict churn are *Logistic Regression*, *Decision Trees* (DT), *Naïve Bayes*, *Support Vector Machines* (SVM) and multi-layer *Artificial Neural Networks* (ANN). In recent years ensemble methods have become popular among researchers since they improve model performance compared to single classification models. The four main types of ensemble methods are *Bagging*, *Boosting*, *Random Forest* (RF) and *Hybrid ensemble* [16].

This is in compliance with methods that are mentioned in papers investigating churn predictions for the gaming industry. In [17] a single hidden layer ANN, Logistic Regression, DT and SVM were compared on two data sets. ANN performed best in terms of AUC. The AUC for one data set was 0.815 while for the other it was 0.930. In [4] Logistic Regression, DT, RF, Naïve Bayes and Gradient Boosting were compared. Gradient Boosting gave the best performance with AUC of 0.83. RF and Logistic Regression also performed quite well but Decision Tree had the worst performance.

3.5 Data used for Churn Modeling

The data which is usually used for churn prediction in the gaming industry can be divided into three categories. The first category is data about activity in the game, for example logins per day. The second category is data related to revenue, that is data about when and what each player is buying. The third category is data about the player himself, for example where he is from, how old he is and for how long he has played the game [17].

Chapter 4

Theory

The field of *Machine Learning* has a wide range of methods to understand data and learn from it. These methods can be divided into two main categories: *supervised* and *unsupervised* machine learning. In supervised machine learning a model is trained on a set of *features* or *predictors* (input) with a known *response* (output). The goal is to be able to use the model to predict the response from previously unseen data. However, in unsupervised learning the output is unknown but the methods can be used to learn the structure and relationships from the data [20, 21].

4.1 Supervised Machine Learning

Supervised learning can be split into *regression* and *classification.* Usually regression is used for *quantitative* (continuous) response variables whereas classification is used for *qualitative* (categorical) response variables. When a qualitative response is predicted each observation is assigned to a category or class, i.e. the observation is *classified.* However, classification methods often first predict the probability of belonging to each class and in that sense behave like methods used for regression [21]. The focus will be on classification in the following sections.

Notation

The input data is the matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, with n *observations* and p features:

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,p} \\ x_{2,1} & x_{2,2} & \dots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,p} \end{bmatrix}, \tag{4.1}$$

where $x_{i,j}$ is the value of the ith observation and jth feature, $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, p$. Observations are written as $\mathbf{x}_i^T$ and features as $\mathbf{X}_j$:

$$\mathbf{x}_i^T = \begin{bmatrix} x_{i,1} & x_{i,2} & \dots & x_{i,p} \end{bmatrix}, \qquad i = 1, 2, \dots, n \tag{4.2}$$

$$\mathbf{X}_j = \begin{bmatrix} x_{1,j} & x_{2,j} & \dots & x_{n,j} \end{bmatrix}^T, \qquad j = 1, 2, \dots, p \tag{4.3}$$

Equation 4.1 can therefore be rewritten as:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} = \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \dots & \mathbf{X}_p \end{bmatrix}, \tag{4.4}$$

The response is defined as $\mathbf{Y} \in \mathbb{R}^{n \times 1}$:

$$\mathbf{Y} = \begin{bmatrix} y_1 & y_2 & \dots & y_n \end{bmatrix}^T \tag{4.5}$$

The relationship between the input $\mathbf{X}$ and the response $\mathbf{Y}$ is:

$$\mathbf{Y} = f(\mathbf{X}) + \epsilon, \tag{4.6}$$

where ϵ is the *irreducible error*, which is random and independent of $\mathbf{X}$ with $E[\epsilon] = 0$ [20, 21]. This error will be explained in more detail in Section 4.1.

Supervised machine learning is used to estimate the function f. The estimation for f is represented by $\hat{f}$ and is used to predict the output $\mathbf{Y}$ from the input data $\mathbf{X}$ [21]:

$$\hat{\mathbf{Y}} = \hat{f}(\mathbf{X}) \tag{4.7}$$

Training and Test Set

The goal is to have a model that predicts the output as correctly as possible. In order to know if the algorithm is performing well the predicted responses, $\hat{\mathbf{Y}}$, can be compared to the known responses, $\mathbf{Y}$. This process is referred to as *training* or *learning* in supervised machine learning. If the model is trained on the whole data set the errors can only be checked for that data which makes it impossible to know if the model will perform well on unseen data, which is the ultimate goal. Hence, the original data set has to be split into a training set and test set [22].

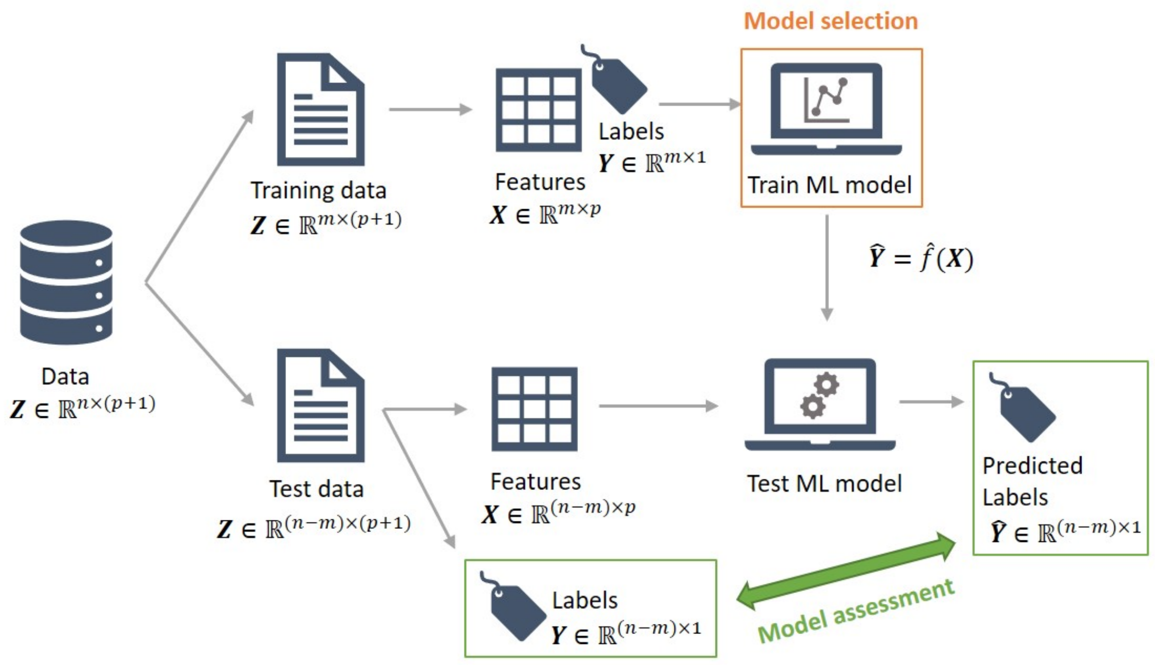

Model selection
Data
$Z \in \mathbb{R}^{n\times(p+1)}$
Training data
$Z \in \mathbb{R}^{m\times(p+1)}$
Features
$X \in \mathbb{R}^{m\times p}$
Labels
$Y \in \mathbb{R}^{m\times 1}$
Train ML model
$\hat{Y} = \hat{f}(X)$
Test data
$Z \in \mathbb{R}^{(n-m)\times(p+1)}$
Features
$X \in \mathbb{R}^{(n-m)\times p}$
Test ML model
Predicted Labels
$\hat{Y} \in \mathbb{R}^{(n-m)\times 1}$
Labels
$Y \in \mathbb{R}^{(n-m)\times 1}$
Model assessment

Figure 4.1: Supervised machine learning.

The matrix **Z** corresponds to the original data set which consists of the input data **X** and the corresponding output **Y**:

$$\mathbf{Z} = \left[\begin{array}{c|c} \mathbf{X} & \mathbf{Y} \end{array}\right] \tag{4.8}$$

The observations, $\mathbf{z}_i = (\mathbf{x}_i, y_i)$, are split randomly into the training and test sets according to some proportion, where the larger proportion is the training set. The proportion depends on the amount of data available [22]. This means that the original data set has n observations, the training set has m observations ($m < n$) and the test set has $n-m$ observations (see Figure 4.1, drawn in Microsoft PowerPoint). The training set will be used to train the model how to estimate f and the unseen test set will be used to evaluate the performance of the model [21, 22].

Methods to Estimate f

There are two different approaches used by supervised machine learning methods to estimate the function f: *parametric* and *non-parametric* methods. Parametric methods make assumption about the shape of f. This simplifies the problem of estimating the function greatly since only a set of parameters need to be estimated. These methods are therefore easy to understand. A drawback to this approach is that the correct shape of f usually does not match the assumptions that were made. Parametric methods produce a small range of shapes in order to estimate f and are therefore *restrictive* methods [21].

In contrast, non-parametric methods do not make any assumptions about the shape of f and hence the danger of the estimation being very different from f is avoided. Non-parametric methods are *flexible* since they produce a wide range of shapes for estimating f. A drawback is that a huge amount of observations are needed for a good estimate of f [21].

There is a *trade-off* between interpretability and flexibility which means that when flexibility increases the interpretability decreases [21].

Training and Test error

The accuracy of the estimated function $\hat{f}$ is determined from the *training error rate*, i.e. the proportion of incorrectly classified labels:

$$\text{Training error rate} = \frac{1}{m}\sum_i I(y_i \neq \hat{y}_i), \quad i \in \{\text{training data}\}, \tag{4.9}$$

where m is the number of observations in the training set, $\hat{y}_i$ is the class label predicted by $\hat{f}$ for the ith observation and I is an *indicator function*:

$$I(y_i \neq \hat{y}_i) = \begin{cases} 1, & \text{if } y_i \neq \hat{y}_i \quad \text{(misclassified)} \\ 0, & \text{if } y_i = \hat{y}_i \quad \text{(correctly classified)} \end{cases} \tag{4.10}$$

The test error is given by the following equation:

$$\text{Test error rate} = \frac{1}{n-m}\sum_i I(y_i \neq \hat{y}_i), \quad i \in \{\text{test data}\}, \tag{4.11}$$

where $n - m$ is the number of observations in the test data set.

Generally, test error rate is of more interest than the training error rate. The goal is that the model performs well on previously unseen data and therefore the test error should be minimized. A classifier with a small test error is a good classifier [21].

The Bias-Variance Trade-off

The *bias-variance trade-off* term explains the connection between model complexity and training and test error. The term will be explained from the regression's point of view. In contrast to classification, where the test error is determined from the misclassification rate (Equation 4.11), the test error for regression is computed by the *mean squared error* (MSE):

$$MSE = \frac{1}{n-m}\sum_i (y_i - \hat{f}(\mathbf{x}_i))^2, \quad i \in \{\text{test data}\} \tag{4.12}$$

The expected test MSE can be split into two terms: *reducible* and irreducible errors (see Equation 4.6). The expected test MSE is given by:

$$E[(y_i - \hat{f}(\mathbf{x}_i))^2] = \underbrace{\mathrm{Var}(\hat{f}(\mathbf{x}_i)) + [\mathrm{Bias}(\hat{f}(\mathbf{x}_i))]^2}_{\text{reducible}} + \underbrace{Var(\epsilon)}_{\text{irreducible}}, \quad i \in \{\text{test data}\} \quad (4.13)$$

The irreducible error is an error that can not be reduced since there can be variables that would be helpful for predicting $\mathbf{Y}$ which are not measured and can therefore not be used. However, the reducible error can be minimized by estimating f more accurately [21].

The reducible error can further be split into two terms: *variance* and the squared *bias*. Variance corresponds to how much $\hat{f}$ would change if the estimation would be performed for a different training data set. Bias on the other hand represents how well the estimate, $\hat{f}$, matches f. There is a trade-off between variance and bias, so if variance increases bias decreases and the other way around. The goal is to have both bias and variance as low as possible [21].

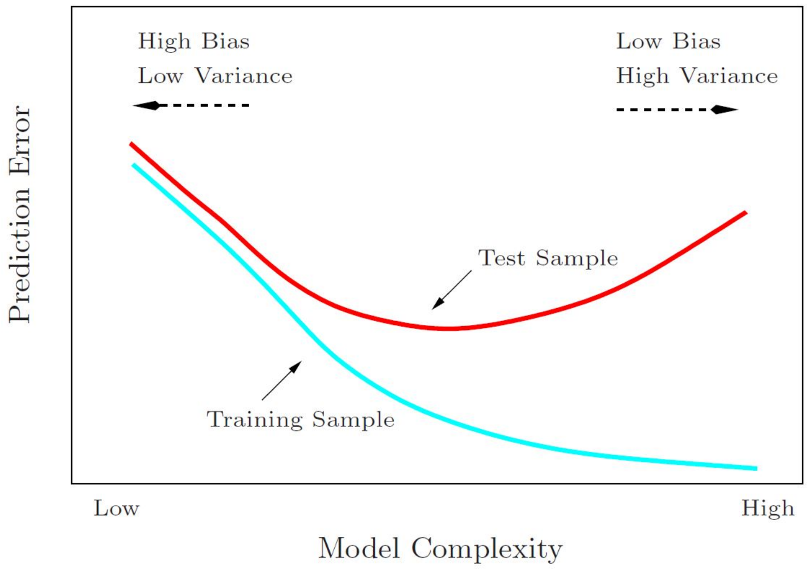


Figure 4.2: Training and test error as a function of model complexity.

In Figure 4.2 [20] the connection between bias and variance, model complexity and training and test error is presented. When model complexity is low the estimate, $\hat{f}$ fits the training data really poorly. This means that both the training and test errors will be high. This results in high bias and low variance. In contrast, when the model complexity is high $\hat{f}$ fits the training data very well. This means that the model has been trained too well, i.e. has memorized the training data and therefore also the *noise*. This phenomenon is called *overfitting*. This leads to poor performance in predicting the response from previously unseen data, causing a large test error and low bias [21, 22].

Model complexity and the risk of overfitting can be reduced by using *cross-validation*, *feature selection* and *dimension reduction*. The process where a suitable flexibility of a model is selected, is called *model selection*, whereas *model assessment* is the process where the performance of the model is evaluated (see Figure 4.1) [21].

4.2 Model Selection

Cross-Validation

In order to know when the learning process of the algorithm should be stopped before it overfits, one more set of data is needed: the *validation set*. This set will be used to validate the learning [22]. This process is called cross-validation (CV) and is used to estimate the test error and select the flexibility level of a model [21].

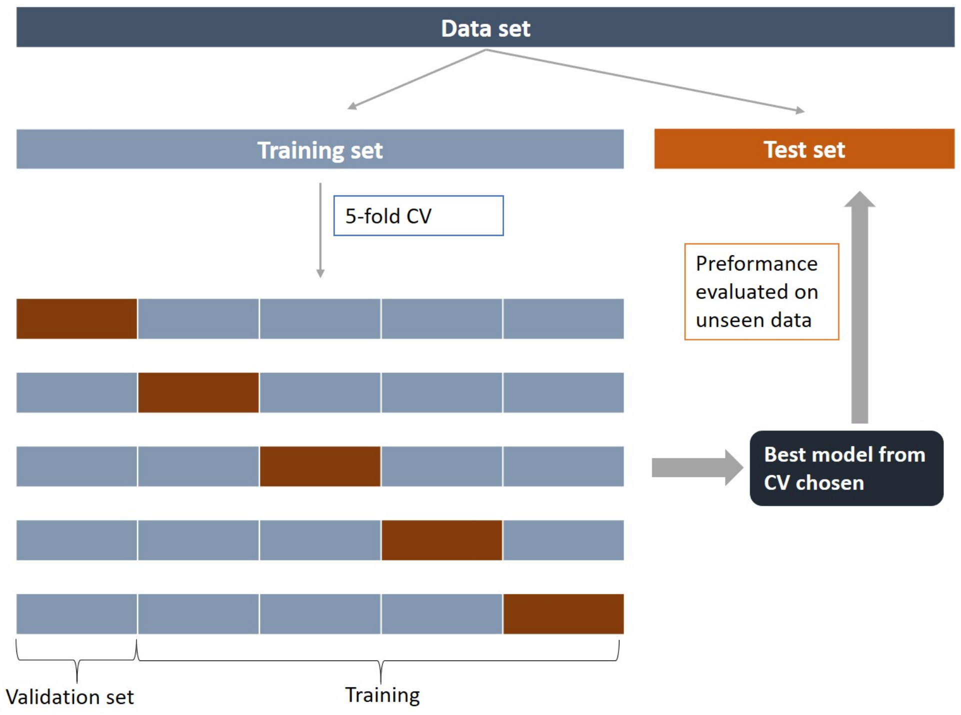

Data set
Training set
Test set
5-fold CV
Preformance evaluated on unseen data
Best model from CV chosen
Validation set
Training

Figure 4.3: 5-fold cross-validation.

Different methods for cross-validation exist but the focus will be on *k-fold CV*. The training set is split randomly into k-folds that are approximately of the same size. A common choice is $k = 5$ or $k = 10$. One of the folds is used as a validation set and the $k - 1$ folds are used for training (see Figure 4.3, drawn in Microsoft PowerPoint). The test error is computed on the validation set. This process is performed k times, where each time a different fold is used for validation . The test error is estimated as the average of the test errors from the k validation sets and is called the *cross-validation error* [21, 22]:

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k} I(y_i \neq \hat{y}_i). \tag{4.14}$$

In order to decide the appropriate flexibility of a model *hyperparameter tuning* has to be done. This is commonly done with a process called *grid search.* A grid of the tuning parameter that should be optimized is defined and cross-validation is performed for each value in the grid. The value that gives the lowest cross-validation error is chosen and the model is re-fit according the selected value of the tuning parameter [20, 21].

Feature Selection

Feature selection is used to select the features that are important and exclude irrelevant variables in order to make a model less complex and hence reducing the variance [21]. *Subset* and *shrinkage methods* for feature selection will be explained shortly in the following.

In subset methods a subset of the p available features are selected and used to fit the model. One such method is *Forward Subset Selection.* It starts with no features and then adds features to the subset one-at-a-time. The feature that is added each time is the one that improves the fit the most. This continues until all of the p features are in the subset [21].

In shrinkage methods on the other hand all of the p features are used to fit a model while shrinking the estimated parameters towards zero. This reduces variance and in some cases if the parameters are estimated to be exactly zero these methods perform variable selection. This is the case in *L1 Regularized Logistic Regression* which will be explained further in Section 4.4 [21].

Dimension Reduction

Feature selection methods use a subset of the original features or shrink their estimated parameters towards zero in order to reduce the complexity of a model. Dimension reduction methods on the other hand, transform the p features by computing $M < p$ different linear combinations of them which are then used to fit a model [21]. A drawback is that these methods reduce interpretability.

4.3 Model Assessment

When the model has been trained to perform as well as possible the next step is to test how it performs on unseen data.

Confusion Matrix

There are four possible outcomes for a classification model. These outcomes can be represented in a *confusion matrix* [23]. The matrix can be seen in Figure 4.1.

		Actual Values	
		Positive	Negative
Predicted Values	Positive	TP	FP
	Negative	FN	TN

Table 4.1: The confusion matrix.

- **True Positive (TP)**: The number of positive instances classified correctly as positive.
- **False Positive (FP)**: The number of negative instances classified wrongly as positive.
- **True Negative (TN)**: The number of negative instances classified correctly as negative.
- **False Negative (FN)**: The number of positive instances classified wrongly as negative.

Evaluation Metrics

The evaluation metrics used for evaluating the performance can be defined from *TP*, *FP*, *TN* and *FN*. Common evaluation metrics will be described below. They all take values between 0 and 1.

- **Accuracy**: The proportion of correct predictions among all predictions [15].

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.15}$$

- **Specificity**: Number of instances correctly predicted as negative divided by the number of actual negative instances. This is also called the *true negative rate (TNR)* [21].

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{4.16}$$

  Note that

$$1 - \text{Specificity} = \frac{FP}{TN + FP}, \tag{4.17}$$

  is the *false positive rate (FPR)*.

- **Recall** or **Sensitivity**: Number of instances correctly predicted as positive divided by the number of actual positive instances. This is also called the *true positive rate (TPR)* [21].

$$\text{Recall} = \text{Sensitivity} = \frac{TP}{TP + FN} \tag{4.18}$$

- **Precision**: The proportion of correctly predicted positive instances among all that were classified as positive [15].

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.19}$$

- **F-measure**: Takes both precision and recall into account, i.e. is a harmonic mean of them [15].

$$\text{F-measure} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4.20}$$

Accuracy is generally not the most informative metric for the results. The pairs specificity and sensitivity (Equations 4.16 and 4.18) and recall and precision (Equation 4.18 and 4.19) are better for measuring the performance of a classifier [22].

There is a trade-off between precision and recall, so when one increases the other decreases. It is therefore better to use the *F-measure* to see how the classifier is performing [22].

A drawback with accuracy, precision and F-measure is that they are affected if the class distribution changes. This is because these metrics are computed from values from both columns in the confusion matrix. However, specificity and sensitivity will not be affected by a change in the class distribution [23]. Hence, the focus will be on these metrics and the *ROC curve* and the *AUC*.

The ROC Curve and AUC

Since the goal is to find *probability of churn* the response will be predicted as a quantitative output, i.e. probabilities $\in [0, 1]$. A *threshold* value can then be decided in order to assign the predicted probabilities the corresponding class labels. If the probability is higher than this threshold the observation will be assigned to the positive class, otherwise it will be assigned to the negative class [20, 23].

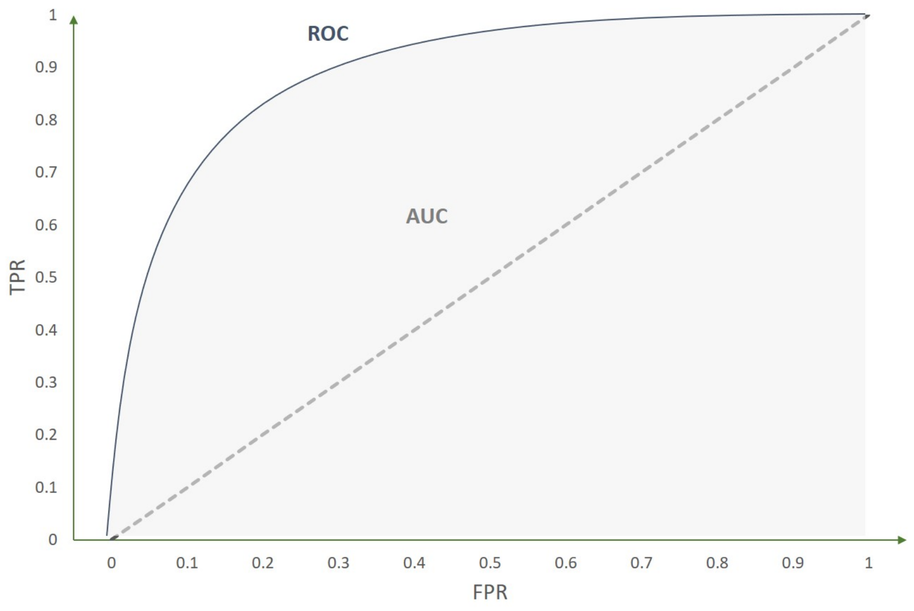


Figure 4.4: The ROC curve and AUC.

The ROC (*receiver operating characteristic*) curve has FPR on the x-axis and TPR on the y-axis and is used to compare classifiers for all threshold values [17]. Each point on the curve represents the performance of a probabilistic classifier for different threshold values [19]. By changing the threshold the TPR (sensitivity) and FPR (1-specificity) can be adjusted [21].

Figure 4.4 (drawn in Microsoft PowerPoint) shows an example of what the ROC curve looks like. The diagonal line $y = x$ shows what the ROC curve looks like if the classes were guessed randomly [23]. In order to compare classifiers by looking at the ROC curve the AUC (*area under the ROC curve*) can be computed. The AUC can take values between 0 and 1. The AUC for the diagonal line is 0.5 and hence in order for a classifier to be better than random guessing the AUC has to be greater than that. The goal is to maximize AUC in order to get a good classifier [17, 19].

4.4 Learning Algorithms

Churn prediction is commonly modeled as a binary classification problem and hence the focus will be on supervised machine learning algorithms that suit that approach. Firstly, *Logistic Regression*, which is a parametric method, will be explained. Secondly, the tree-based methods *Decision Tree* and *Random Forest*, which are non-parametric methods, will be explained.

Logistic Regression

Logistic Regression predicts the probability of the output $\mathbf{Y}$ belonging to a certain class rather than directly predicting the response of $\mathbf{Y}$. The response is binary:

$$\mathbf{Y} = \begin{cases} 0, & \text{if an observation belongs to the negative class} \\ 1, & \text{if an observation belongs to the positive class} \end{cases} \tag{4.21}$$

The method models the relationship between $p(\mathbf{X}) = \mathbb{P}(\mathbf{Y} = 1|\mathbf{X})$ and $\mathbf{X}$ [21].

Single Predictor

First the focus will be on predicting a binary response from one feature. In *Linear Regression* the relationship between $p(\mathbf{X})$ and $\mathbf{X}$ is assumed to be linear:

$$p(\mathbf{X}) = \beta_0 + \beta_1\mathbf{X}, \tag{4.22}$$

where β_0 (intersect) and β_1 (slope) are unknown parameters. When a binary classification is performed by fitting a straight line to a binary output some of the predicted probabilities can be $p(\mathbf{X}) < 0$ and $p(\mathbf{X}) > 1$. This is a problem since probabilities should be between 0 and 1 [21].

In order to get outputs within the interval $[0, 1]$, Logistic Regression is used. The relationship between $p(\mathbf{X})$ and $\mathbf{X}$ is modeled according to the *logistic function*:

$$p(\mathbf{X}) = \frac{e^{\beta_0+\beta_1\mathbf{X}}}{1 + e^{\beta_0+\beta_1\mathbf{X}}} \tag{4.23}$$

The function has an S-shaped curve which makes sure that the probabilities are between 0 and 1 (see Figure 4.5 [21]).

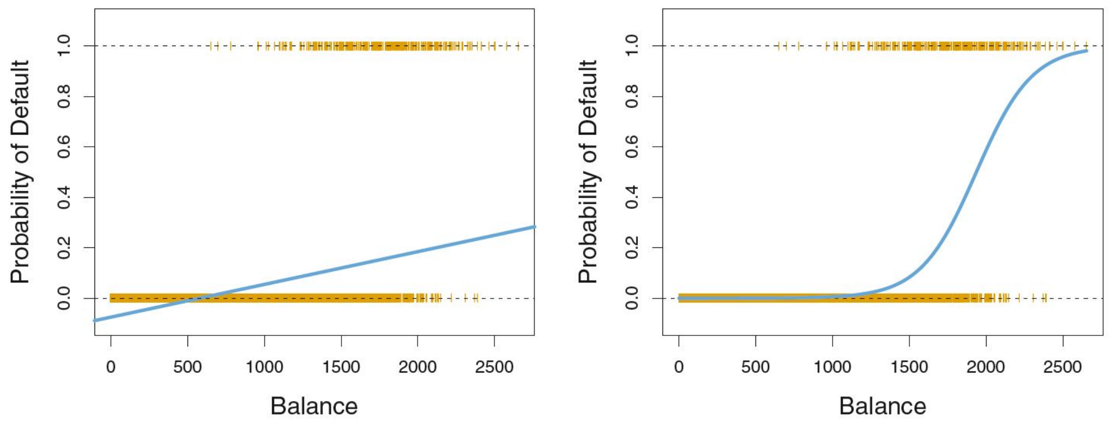


Figure 4.5: An example of how Linear Regression (left plot) and Logistic Regression (right plot) can look like. The x-axis is the single predictor and the y-axis is the probability of belonging to the positive class.

The logistic function can be rewritten to the form:

$$\frac{p(\mathbf{X})}{1 - p(\mathbf{X})} = e^{\beta_0+\beta_1\mathbf{X}}. \tag{4.24}$$

The quantity on the left-hand side is called the *odds* and can take values in the interval $[0, \infty]$. The logarithm is taken of both sides in Equation 4.24:

$$\log(\frac{p(\mathbf{X})}{1 - p(\mathbf{X})}) = \beta_0 + \beta_1\mathbf{X}. \tag{4.25}$$

Log-odds or *logit* is what the quantity on the left-hand side is called. The right hand side is the same as in Equation 4.22, i.e. the logit is linear in $\mathbf{X}$. From this it can be seen that Logistic Regression assumes there is a linear relationship between the log-odds of $p(\mathbf{X})$ and the feature $\mathbf{X}$, i.e. the method estimates linear decision boundaries [20, 21].

The unknown parameters β_0 and β_1 are estimated from the training set by the *maximum likelihood* function

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(\mathbf{x}_i) \prod_{i':y_{i'}=0} (1 - p(\mathbf{x}_{i'})) \tag{4.26}$$

The estimated parameters, $\hat{\beta_0}$ and $\hat{\beta_1}$, should maximize this function. In other words, the goal of the maximum likelihood method is to estimate β_0 and β_1 so that the predicted probability, $\hat{p}(\mathbf{x}_i)$, is close to zero for an observation that should belong to the negative class and close to one for an observation that should belong to the positive class [21].

Multiple Predictors

Now the functions used for Logistic Regression above will be extended for multiple features. Equation 4.24 becomes:

$$\log(\frac{p(\mathbf{X})}{1 - p(\mathbf{X})}) = \beta_0 + \beta_1\mathbf{X}_1 + \beta_2\mathbf{X}_2 + \ldots + \beta_p\mathbf{X}_p. \tag{4.27}$$

The logistic function can therefore be written as:

$$p(\mathbf{X}) = \frac{e^{\beta_0+\beta_1\mathbf{X}_1+\beta_2\mathbf{X}_2+\ldots+\beta_p\mathbf{X}_p}}{1 + e^{\beta_0+\beta_1\mathbf{X}_1+\beta_2\mathbf{X}_2+\ldots+\beta_p\mathbf{X}_p}} \tag{4.28}$$

Similarly to when there is a single predictor the parameters, $\beta_0, \beta_1, \beta_1, \ldots, \beta_p$, are estimated with the maximum likelihood function [21].

The logistic regression algorithm is summarized below.

Logistic Regression [21]:

Learning

- The parameters $\beta_0, \beta_1, \beta_1, \ldots, \beta_p$ are estimated from the training set by the maximum likelihood function (Equation 4.26).

Prediction

- $\hat{\beta_0}, \hat{\beta_1}, \hat{\beta_2} \ldots \hat{\beta_p}$ are plugged into the logistic function (Equation 4.28) to predict $p(\mathbf{X})$ for previously unseen data.

L1 Regularized Logistic Regression

A common problem in a two-class Logistic Regression is that the data is perfectly separable, i.e. a feature or features perfectly predict the output. This causes the estimated parameters from the maximum likelihood to be undefined [20]. *L1 Regularized Logistic Regression* or *Lasso* can be used to overcome this problem.

The *log likelihood* for Logistic Regression is given by:

$$\begin{aligned}\ell(\boldsymbol{\beta}) &= \sum_{i=1}^{N}\{y_i \log(p(\mathbf{x}_i)) + (1-y_i)\log(1-p(\mathbf{x}_i))\} \\ &= \sum_{i=1}^{N}\{y_i(\beta_0 + \boldsymbol{\beta}^T\mathbf{x_i}) - \log(1+e^{\beta_0+\boldsymbol{\beta}^T\mathbf{x}_i})\} \qquad (4.29)\end{aligned}$$

The penalized version with the L1 penalty is used in the shrinkage method Lasso and is written as:

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{N}\{y_i\boldsymbol{\beta}^T\mathbf{x_i} - \log(1+e^{\boldsymbol{\beta}^T\mathbf{x}_i})\} - \lambda \underbrace{\sum_{j=i}^{p}|\beta_j|}_{\text{L1 penalty}}, \qquad (4.30)$$

where $\beta_0, \beta_1, \beta_1, \ldots, \beta_p$, are estimated by maximizing this function and hence the λ has to be chosen carefully [20].

As stated in Section 4.2 estimated parameters are shrunken towards zero when using shrinkage methods. When Lasso is used the L1 penalty forces some of the estimated parameters to be exactly zero. This happens when λ is large enough. Because of this effect, Lasso performs feature selection.

Decision Tree

Decision Trees (DT) can be used both for regression and classification and are easy to interpret and represent graphically.

A tree is grown using a *recursive binary splitting* where the feature space is split into J simpler regions, $R_1, R_2, \ldots, R_J$. This tree is then used to predict the output of previously unseen data [21, 22]. The DT algorithm is explained in more detail below.

Decision Tree [21]:

Learning: Growing a DT

- Start with all the features from the training set. This is called the *root node* of the tree and is at the top.
- The feature space is split on the *most informative* feature. This creates two *child nodes* according to the splitting. These nodes are connected to the root node by *branches*.
- This process is repeated recursively for each node, where the corresponding feature space is split by the most informative feature.
- The process continues until a stopping criterion is met. These nodes are called *leafs* and indicate the regions $R_1, R_2, \ldots, R_J$.

Prediction

The region where the new observation belongs is found from the grown DT. Two approaches are commonly used to predict the class of the new observation:

1. The majority vote of the class labels of the training observations in the region is used to determine the class.
2. The proportions of each class in the region, i.e. the probability of belonging to a class, is computed.

The second approach is usually of more interest.

The most informative feature for classification is found by computing the *gini index*:

$$\text{Gini index} = \sum_{k=1}^{K} \hat{p}_{l,k}(1 - \hat{p}_{l,k}), \tag{4.31}$$

where $\hat{p}_{l,k}$ stands for the proportion of training observations in the region l that have class k. The gini index measures the total variance among the K classes [21].

Cross-entropy is also commonly used to find the feature to split:

$$\text{Cross-entropy} = -\sum_{k=1}^{K} \hat{p}_{l,k} log(\hat{p}_{l,k}) \tag{4.32}$$

From Equation 4.31 it can be seen that the gini index is low when the majority of observations are from the same class. This is also the case with cross-entropy (Equation 4.32). These two measures are therefore used to measure *node purity* or the quality of a split [21]. The most informative feature which is used for the splitting is the one that has the lowest gini index or cross-entropy.

A drawback with DTs is that there is a large risk they will overfit since they use all the features that are given to them [22]. This leads to high variance. *Pruning* can be used to reduce variance. Pruning methods are split into two categories: *pre-pruning* and *post-pruning*. In pre-pruning the tree building stops before it is complete. The tree is built until the test error no longer decrease more than some threshold by making a split. In post-pruning the complete tree is first built and then the tree is pruned back in order to reduce the test error for unseen observations [21, 24].

Using a single DT does not result in a good prediction accuracy compared to other classification algorithms. However, by combining trees and averaging the results reduces the variance and hence improves the predictive performance. Various methods exist for doing this and one of them is Random Forest [21].

Random Forest

Random Forest (RF) is an ensemble method. For classification, a number of decision trees are built on bootstraps of the training set where majority voting of class labels or average of class proportions among trees is used to predict the outcome.

Bootstrap replicate will first be defined and then the RF algorithm will be stated.

Bootstrap replicate [21]:

- A bootstrap data set, $\mathbf{Z}^{*1}$, is created by selecting randomly n observations from the original data set, $\mathbf{Z}$, which has n observations.
- The same observation can appear more than once in the same bootstrap set since the sampling is done with replacement.
- This is repeated B times so B different bootstrap sets are created: $\mathbf{Z}^{*1}, \mathbf{Z}^{*2}, \ldots, \mathbf{Z}^{*B}$.

Random Forest:

Learning: Growing a RF

For $b \in \{1, 2, \dots, B\}$:

- A bootstrap sample $\mathbf{Z}^{*b}$ is drawn from the training set
- A large decision tree T_b is grown on the bootstrap sample. The following steps are preformed and repeated recursively for each node until a stopping criterion is met:
  - Randomly choose q features from the p possible features. A common choice for classification is $q \approx \sqrt{p}$.
  - The most informative feature is chosen among the q features.
  - This feature is used to perform a binary split into two child nodes.

The output is the random forest $\{T_b\}_1^B$ [20, 21].

Prediction

To predict the class of a new observation, $\mathbf{x}$, there are two approaches similarly to the once mentioned for DT:

1. Majority vote

$$\hat{C}_{RF}^{B}(\mathbf{x}) = \text{majority vote}\{\hat{C}_b(\mathbf{x})\}_1^B, \tag{4.33}$$

   where $\hat{C}_b(\mathbf{x})$ is the predicted class from the bth tree from the RF. In other words, the RF gets a class vote from each individual tree and then takes majority vote among these classes to classify [20, 21].

2. For the terminal node that the new observation belongs to in each tree the class proportions are computed. These class probabilities are then averaged over all the trees [25]

The trees that are grown on each bootstrap are grown deep and do not need to be pruned. This leads to high variance and low bias for each individual tree. By averaging these B trees the variance will be reduced. Increasing the value of B (number of trees grown) will not lead to overfitting [21].

The reason for only choosing q features among the p available is because this *decorrelates* the trees. When all features are used, the process is called Bagging. If there is one very strong feature and the others are moderately strong the collection of bagged trees would all be very similar to each other since most of the trees would use the strong feature as the first split. This causes the predictions from these trees to be highly correlated. Taking the average of correlated trees does not reduce the variance as much as averaging uncorrelated trees, which is the case in RF ($q < p$) [21].

The test error of a RF can be estimated without using CV. As described above, when creating each bootstrap sample n observations are chosen randomly from the original data set and sampling is done with replacement. This means that some observations will appear many times in the same bootstrap and some will not appear at all. The observations that are not used to grow the corresponding tree, T_b, are called *out-of-bag* (OOB). This means that it is possible to predict the output for the ith observation from trees that were grown on bootstraps where the ith observation did not appear. This is done for all the observations and an overall OOB classification error is computed which is an estimate of the test error [21].

A single tree is easier to interpret than a RF. However, *variable importance* can be accessed for RF. For classification trees the decrease of the gini index (Equation 4.31) due to splitting over a certain feature can be added up and averaged over all B trees. High values correspond to important features [21].

Chapter 5

Method

The *Cross-Industry Standard Process for Data Mining* (CRISP-DM Model) described in [26] was followed in this thesis. This process can be seen in Figure 5.1 [27] and what was done in each stage will be described in detail in the following sections. The query language *SQL* was used to collect data from the database where the data is stored and *RStudio* was used to prepare the data and for creating the model.

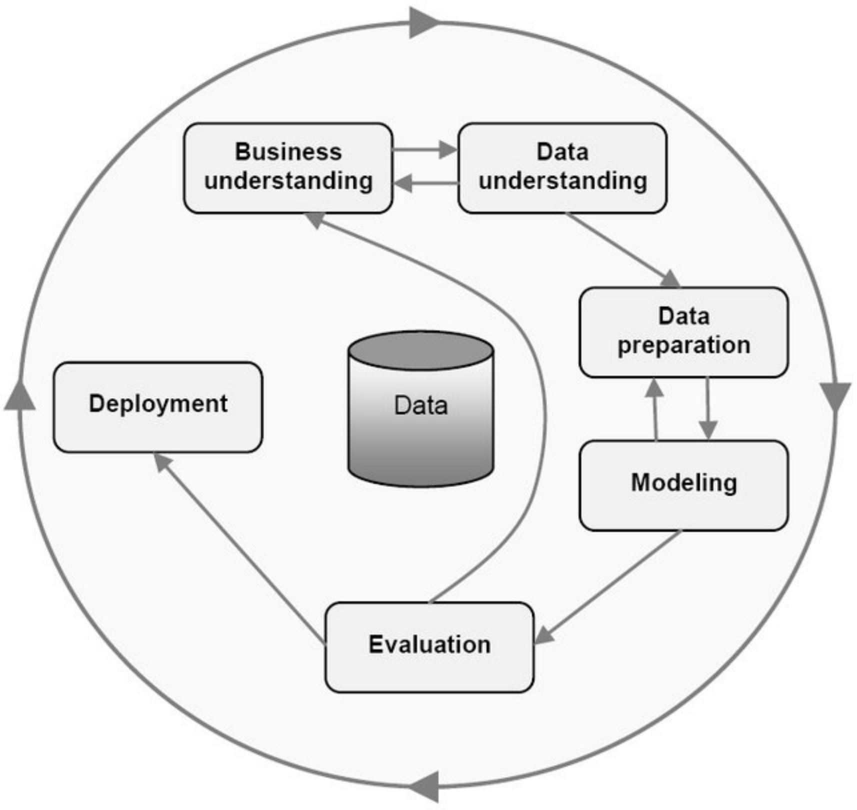


Figure 5.1: CRISP-DM Model.

5.1 Business Understanding

In this first step of the process the thesis objectives were discussed with Paradox. As stated in Section 1.1 the goal of this thesis is to identify whales and predict the probability of them churning. By doing this, Paradox can target the group of whales that are likely to churn in order to retain them and hence increase revenues. In order to reach this goal the definition of Paradox whales and when they churn had to be decided.

Definitions

From Sections 3.2 and 3.3 it is clear that the definitions of whales and when they churn are not universally agreed upon. Rather, these tend to be defined in relation to the particular project and circumstances at hand. This is also the approach used in the current work.

The following constraints were considered:

- **Paradox PC games** (exclude console and mobile). These games are *premium games* which means that they are not free-to-play.
- **Players with a Paradox account**.
- Player has to be a **Paradox customer** before the period in question starts. This makes sure new customers that start within the period are excluded. The term Paradox customer means that the player has to either have a Paradox account or have purchased something connected to a PC game. The reason for this definition is that a player can make a purchase before creating an account and it is important to include those players as well.

Whales

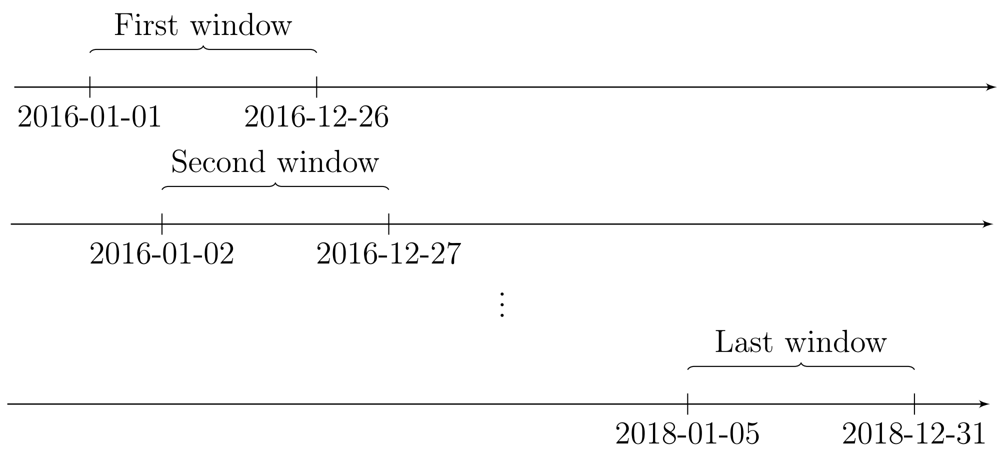
After discussing with the company it was decided to look at how much players spend over 12 months since there are fluctuations in spending depending on seasons and new releases. It was also decided to look at spending only on DLCs (add-ons and expansions).

In order to state the definition of whales for this thesis two other definitions will be stated first.

Definition 5.1: Percentile
The value which q percent of the observations from a data set are less than or equal to is called the *q-th percentile* of the data set, $q \in (0 : 100)$ [28].

Definition 5.2: Rolling Twelve Month Window
Within the company it is assumed that a month is 30 days and therefore a *twelve month window* is defined to be 360 days. *Rolling window* means that for each day that passes the window should move by one day.

It was decided to look at spending in each window from 2016-01-01 until 2018-12-31:

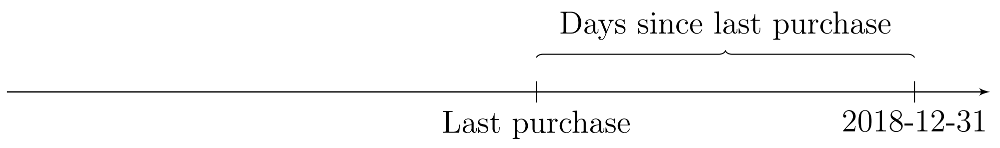


From these two definitions above whales can now be defined.

Definition 5.3: Whales
For each time period in the *rolling twelve month window* the *95-th percentile* of the players' spending on DLCs is calculated. The players that spend more than this value are defined as *whales*. If a player fulfills this condition in at least one window he will always be defined as a whale.

Churn

Since the target group of this thesis is big spenders the focus was on their purchase behaviour rather than playing behaviour. It was decided to investigate how many days had passed since the whales' last purchase (including all products for PC games):



The distribution of the days since last purchase for the whales was plotted in order to decide how many days were allowed to pass from their last purchase before they are defined as churners. In addition the average number of days between purchases was investigated. This was discussed with Paradox and decided to choose 28 days as the limit.

Definition 5.4: Churn of Whales
A whale has *churned* if he has not bought anything for more than 28 days, i.e. 4 weeks. *Churn* here means that the player is not a whale anymore, i.e. he has stopped being a big spender for Paradox PC games.

5.2 Data Understanding

Getting an understanding of the data was the most time consuming part of this project. In order to be able to define whales and when they churn some data analysis had to be done first. Therefore, going back and forth from this second step of the process and the first step was needed (see Figure 5.1). When the Buisness Understanding was complete the whales were filtered out according to Definition 5.3 and labeled as *churners* (1) and *non-churners* (0) according to Definition 5.4. The next step was then to collect information about these players.

The data that was collected can be divided into three categories as mentioned in Section 3.5. An example of what kind of features were collected:

- **Activity in games**: Number of days played, session time and inactivity time.

- **Purchase behaviour**: Number of purchases, which products were purchased, amount spent and time between purchases.
- **Player information**: Anonymized game platform ID and country.

Since the data was labeled according to if 28 days had passed from the last purchase or not the feature "date of last purchase" had to be removed from the data set. Similarly, it was decided to collect data until 28 days before 2018-12-31 in order not to have information from the "future".

5.3 Data Preparation

After all the data had been collected the data preparation was performed in RStudio. First some of the information collected had to be transformed such that each row (observation) was connected to a whale ID and each column (feature) contained information about the corresponding whale. When all of the data had this format it was joined into a *data frame* (which is a table) combined from an ID column, the input **X** (Equation 4.1) and the output **Y** (Equation 4.5), the class labels:

$$\left[\begin{array}{c|cccc|c} ID_1 & x_{1,1} & x_{1,2} & \dots & x_{1,p} & y_1 \\ ID_2 & x_{2,1} & x_{2,2} & \dots & x_{2,p} & y_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ ID_n & x_{n,1} & x_{n,2} & \dots & x_{n,p} & y_n \end{array}\right] = \left[\begin{array}{c|c} \mathbf{ID} & \mathbf{Z} \end{array}\right] \tag{5.1}$$

The following data preparation was done:

- All the whales have not been Paradox customers for the same amount of time. Therefore the amount spent by each whale had to be divided by the number of days since they bought for the first time, i.e. amount spent per day.
- Similarly, the session time for each whale had to be divided by the number of days since they played for the first time, i.e. session time per day
- Some rows had too many missing values and were therefore removed from the data set.
- The format of date columns was changed into five new categorical columns: year (2013-2018), month (1-12), week (1-4), day (1-7) and isWeekend (0/1).

The data was next split randomly into a training set, $\frac{3}{4}$ of the observations, and a test set, $\frac{1}{4}$ of the observations. The ID column was removed from the training set since it should not go into the learning algorithm.

The libraries used in RStudio for the data preparation are shown in Table 5.1.

	Library
Data frames	`dplyr`, `tibble`, `data.table`, `reshape2`
Dates	`lubridate`
Categorical features	`forcats`
Strings	`stringr`
Read from files	`readr`

Table 5.1: Libraries used for data preparation in RStudio.

5.4 Modeling

The training set was used to train the models. The machine learning methods that were tested were Logistic Regression, Decision Tree and Random Forest.

It is quite common to try Logistic Regression before anything else since it is simple and easy to understand. Logistic Regression is a parametric method which assumes there is a linear relationship between the features and the log-odds of the output. DT and RF are on the other hand non-parametric and do not assume any relationship beforehand. Using a single DT does usually not give good performance compared to other classifiers but was used to compare with RF which is a popular and powerful prediction method.

The pre-processing of the data that was done for all methods except DT was imputation of missing values. Missing values for continuous features were imputed by the median of the corresponding feature. Two categorical features had missing values and it was decided to impute them with "Unknown", i.e. make a new category for the missing values.

Logistic Regression was first trained with all the available features. The model did not handle the whole feature set and the results could not be used since the data was perfectly separable. More pre-processing was done in order to reduce the amount of features. Highly correlated features and features with *near-zero variance* were removed. Near-zero variance features are features that only have one unique value or features that have both few unique values compared to the number of observations and have a large ratio between the frequency of the most common value and the second most common value [29]. Forward subset selection was then performed. However, this also resulted in the perfect separation problem. The reason for Logistic Regression not working could be that there are too many categorical features in the feature set. Therefore Logistic Regression was also tried only on numerical features but that lead to the same problem.

An alternative to Logistic Regression is to use L1 Regularized Logistic Regression, where some of the parameters are shrunken to zero, which worked. Next Decision Tree and Random Forest were tested. All of the three models performed best when all the original features were kept.

In the following the training process for the three final models will be explained in detail.

- **L1 Regularized Logistic Regression**
  - Pre-processing:
    Imputation of missing values
  - 5-fold CV:
    Performed with the `cv.glmnet` function. An automatic grid search was performed where the optimal value of the tuning parameter λ was determined.
  - Model trained on the whole training set:
    Performed with the `glmnet` function with the optimal value of λ
- **Decision Tree**
  - 10-fold CV:
    Performed with the `train` function (`caret` library) with method = "rpart". An automatic grid search was performed where 50 values of the tuning parameter *cp* (*complexity parameter*) where tried.
  - Model trained on the whole training set:
    Performed with the function `rpart` with the optimal value of *cp*.

- **Random Forest**
  - Pre-processing:
    Imputation of missing values
  - 5-fold CV:
    Performed with the `train` function (`caret` library) with method = "parRF". A grid search was performed for the tuning parameter `mtry` (number of randomly selected features at each split) where four values close to $q = \sqrt{p}$ where tried.
  - Model trained on the whole training set:
    Performed with the function `randomForest` with the optimal value of `mtry`.

No feature selection methods were needed since `glmnet`, `rpart`, `parRF` and `randomForest` perform feature selection automatically.

A summary of the libraries used in RStudio for modeling and evaluation can be seen in Table 5.2.

	Library
Training and predicting	`caret`
L1 Regularized LR	`glmnet`
Decision Tree	`rpart`
Random Forest	`randomForest`
ROC curve	`pROC`

Table 5.2: Libraries used for modeling and evaluation in RStudio.

5.5 Evaluation

To evaluate the performance of the models the test data set was used. The test set was pre-processed according to the training set. In order to make predictions from the trained models the `predict` function (`caret` library) was used. The evaluation metric AUC was used to compare classifiers. Different probability threshold values were compared for the other evaluation metrics listed in Section 4.3.

Chapter 6

Results

The final data set has 75 features and 106944 observations. 27 features are continuous and the remaining 48 are categorical. The majority class is non-churners. The data was split randomly into a training set, $\frac{3}{4}$ of the observations, and a test set, $\frac{1}{4}$ of the observations.

The results from the methods L1 Regularized Logistic Regression, Decision Tree and Random Forest will be explained in the following sections. The AUC evaluation metric was used to compare the classifiers and the best results obtained from each machine learning method can be seen in Table 6.1.

	AUC
L1 Regularized LR	0.6855
Decision Tree	0.6655
Random Forest	0.7162

Table 6.1: Final results from the models.

6.1 L1 Regularized Logistic Regression

Pre-Processing

First missing values were imputed. Continuous features were imputed by the median of the corresponding feature and categorical features with "Unknown". The library `glmnet` was used to perform the L1 Regularized Logistic Regression (Lasso). The `glmnet` function only works for numerical features and therefore *one-hot-encoding*, where categorical features are encoded to numerical, is done automatically when using the function. This increased the features from 75 to 168.

Correlation analysis is not needed since Lasso picks one of the correlated features and discards the others, i.e. shrinks their parameters to zero. Scaling is not needed since the `glmnet` function also performs scaling automatically but returns the parameters on the original scale [30].

Model Selection

Hyperparameter Tuning

The `glmnet` method has two tuning parameters, λ and α. The value of α was held constant at 1, to use the Lasso shrinkage method, while hyperparameter tuning was performed with 5-fold CV to find the optimal value of λ.

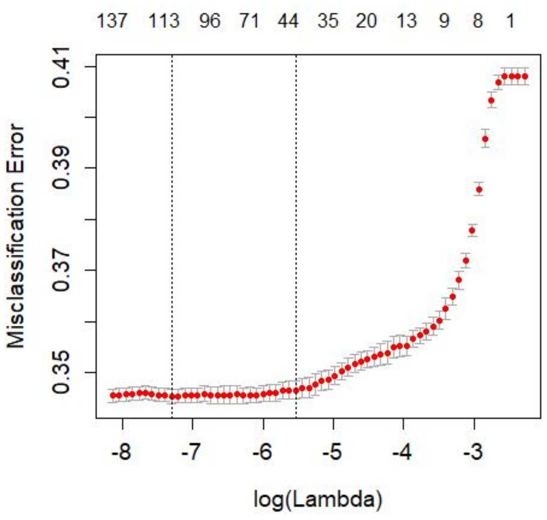


Figure 6.1: The $\log(\lambda)$ plotted as a function of the cross-validation error. The error bars on the plot correspond to the upper and lower standard deviations. The top axis correspond to the number of non-zero features.

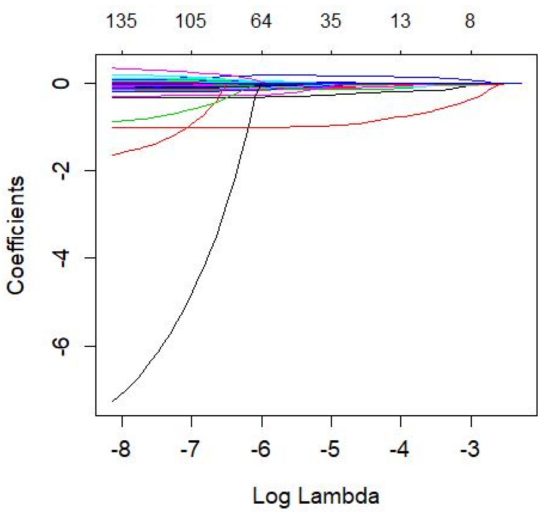


Figure 6.2: The $\log(\lambda)$ plotted as a function of the parameter values. The top axis correspond to the number of non-zero features.

From Figure 6.1 the $\log(\lambda)$ for the values tried by `cv.glmnet` can be seen as a function of the cross-validation error (Equation 4.14). According to [30] the two vertical lines correspond to λ_{min}, which gives the minimum value of the cross-validation error, and λ_{1se}, which results in the most regularized model where the misclassification error is within one standard deviation of the minimum error. In Figure 6.2 the $\log(\lambda)$ is plotted as a function of the parameter values. Moving to the right more parameters are shrunken to zero, hence reducing the number of non-zero parameters.

The optimal value was $\lambda_{min} = 0.0006726743$ and this caused 55 parameters to be shrunk to zero, i.e. the features were reduced from 168 to 113. The final model was then trained on the whole training set with $\alpha = 1$ and $\lambda = \lambda_{min}$.

Model Assessment

The performance was evaluated on the test set and the results for a few different thresholds can be seen in Table 6.2. The AUC was 0.6855 and the ROC curve can be seen in Figure 6.3.

	Threshold				
	0.3	**0.4**	**0.5**	**0.6**	**0.7**
Accuracy	0.5544	0.6357	0.6569	0.6451	0.6218
Sensitivity	0.8463	0.6251	0.3966	0.2202	0.1005
Specificity	0.3534	0.6431	0.8363	0.9378	0.9809
Precision	0.4742	0.5468	0.6253	0.7092	0.7840
F1	0.6078	0.5833	0.4853	0.3361	0.1782
TP	9230	6817	4325	2402	1096
FN	1676	4089	6581	8504	9810
TN	5595	10180	13238	14845	15528
FP	10235	5650	2592	985	302

Table 6.2: Evaluation metrics according to chosen thresholds for L1 Regularized Logistic Regression.

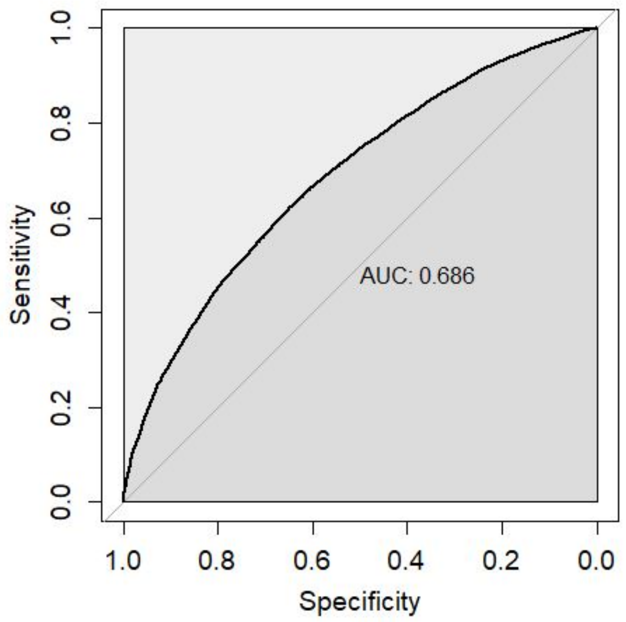


Figure 6.3: ROC curve and AUC for L1 Regularized Logistic Regression.

6.2 Decision Tree

Pre-Processing

No pre-processing had to be done. The `rpart` method uses an inbuilt default action for handling missing values. This action only removes rows if the response variable or all the features are missing. This means that `rpart` has the ability to partially retain missing observations [31].

Model Selection

Hyperparameter Tuning

Decision Trees are built by choosing each time the most informative feature to split the feature space. Therefore, the first feature chosen is the most important feature. Decision Trees use all features by default but usually pruning is performed. In the `rpart` library which is used the parameter *cp* (*complexity parameter*) can be tuned to control the post-pruning of the tree. This means that all features are used and the tree is fully grown and then pruned back according to the value of *cp* which controls the number of splits made [31].

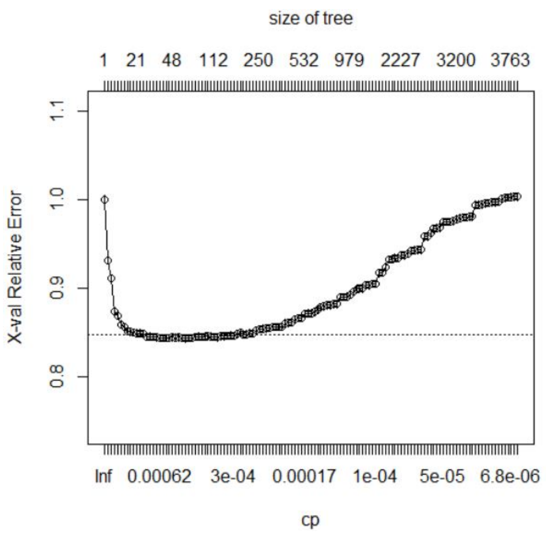


Figure 6.4: The *cp* as a function of cross-validation error.

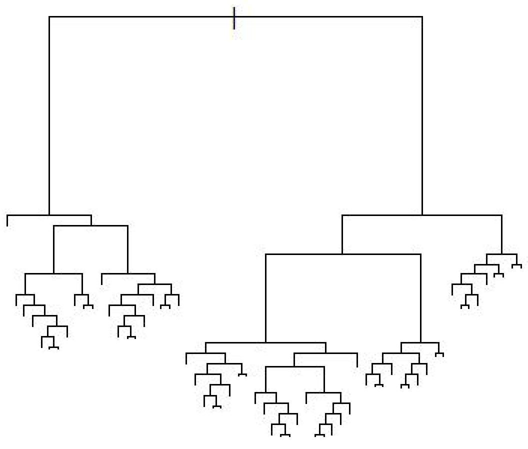


Figure 6.5: The Decision Tree.

The value of *cp* was tuned by using 10-fold CV where 50 different values for *cp* were tried. From Figure 6.4 it can be seen how the cross-validation error (Equation 4.14) changes as *cp* decreases and the number of splits increases. The optimal value was $cp = 0.000488878$ which allowed the tree to split 60 times. The tree was then built according to this cp value on the whole training set (see Figure 6.5).

Model Assessment

The performance was evaluated on the test set and the results for a few different thresholds can be seen in Table 6.3. The AUC was 0.6655 and the ROC curve can be seen in Figure 6.6.

	Threshold				
	0.3	**0.4**	**0.5**	**0.6**	**0.7**
Accuracy	0.5910	0.6557	0.6612	0.6581	0.6165
Sensitivity	0.7129	0.5062	0.4257	0.3107	0.8674
Specificity	0.5071	0.7587	0.8234	0.8973	0.9815
Precision	0.4991	0.5911	0.6242	0.6759	0.7635
F1	0.5871	0.5454	0.5062	0.4258	0.1558
TP	7775	5521	4643	3389	946
FN	3131	5385	6263	7517	9960
TN	8027	12011	13035	14205	15537
FP	7803	3819	2795	1625	293

Table 6.3: Evaluation metrics according to chosen thresholds for Decision Tree.

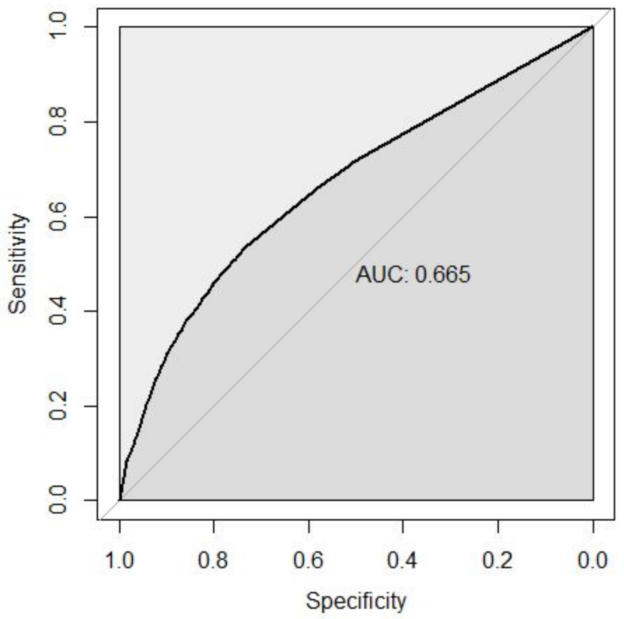


Figure 6.6: ROC curve and AUC for Decision Tree.

6.3 Random Forest

Pre-processing

Missing values had to be imputed. Continuous features were imputed by the median of the corresponding feature and categorical features with "Unknown".

Model Selection

Hyperparameter Tuning

Random Forest chooses $q \approx \sqrt{p}$ features randomly from the available features each time and chooses the most informative one of them to split the feature space. The value of q is a tuning parameter for `parRF` (`caret` library) which is called `mtry`, i.e. how many features should be chosen at random each time.

Since there are 75 features $q \approx \sqrt{75} = 8.660254$. Therefore it was decided to use the grid `mtry` $= c(8, 10, 12, 15)$. The grid search was performed by using 5-fold CV and the optimal value was `mtry` $= 15$ (see Table 6.4).

mtry	**Accuracy**
8	0.6700080
10	0.6702947
12	0.6703445
15	0.6703445

Table 6.4: Values of mtry used in grid search.

The model was then trained with this value of `mtry` on the whole training set with the `randomForest` function. The model was trained with `ntree` $= 1500$ but from Figure 6.7 it can be seen that a smaller amount of trees, for example 1000, could have been chosen.

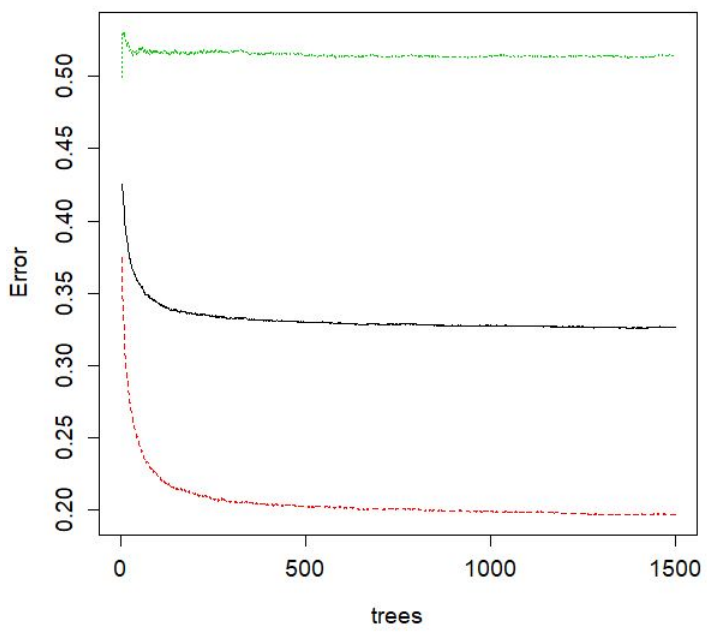


Figure 6.7: Class errors and OOB error as a function of number of trees grown. Green: Class error for the positive class, i.e. FP/TP. Red: Class error for the negative class, i.e. FN/TN. Black: OOB error.

Model Assessment

The performance was evaluated on the test set and the results for a few different thresholds can be seen in Table 6.5. The AUC was 0.7162 and the ROC curve can be seen in Figure 6.8.

	Threshold				
	0.3	**0.4**	**0.5**	**0.6**	**0.7**
Accuracy	0.5426	0.6360	0.6771	0.6618	0.6294
Sensitivity	0.9001	0.7335	0.4879	0.2735	0.1172
Specificity	0.2963	0.5689	0.8075	0.9293	0.9822
Precision	0.4684	0.5396	0.6358	0.7272	0.8198
F1	0.6162	0.6218	0.5521	0.3975	0.2051
TP	9816	8000	5321	2983	1278
FN	1090	2906	5585	7923	9628
TN	4694	9005	12782	14711	15549
FP	11140	6825	3048	1119	281

Table 6.5: Evaluation metrics according to chosen thresholds for Random Forest.

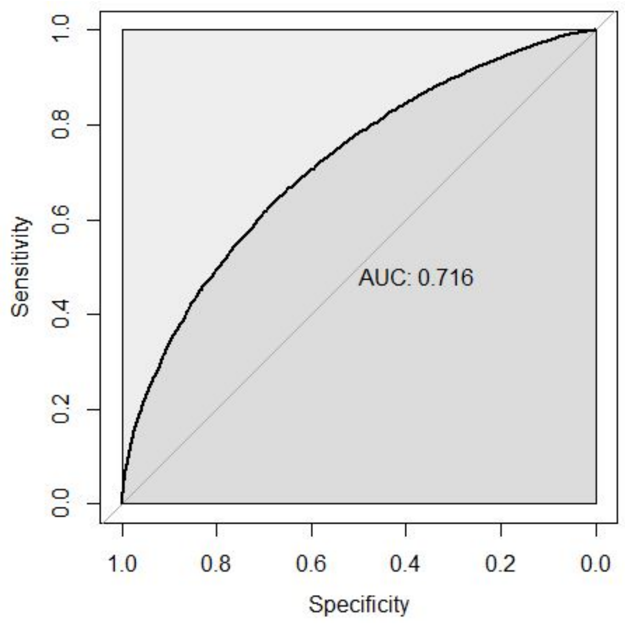


Figure 6.8: ROC curve and AUC for Random Forest.

Chapter 7

Discussion

7.1 Findings

The main findings of this project are the answers to the research questions stated in Section 1.1. In the following the questions will be answered.

Question 1:
What is a good way to define which players are Paradox whales, i.e. players that are big spenders in Paradox PC games?

In order to take seasonal fluctuations and new releases into account it was decided to look at spending on DLCs for players with a Paradox account over a 12 month rolling period. The players spending more than the 95th-percentile of the total spending by the players for each period were defined as whales.

When this definition was decided a histogram of cohorts vs. amount spent over a 12 month period was plotted. The amount spent by each player (fulfilling the constraints mentioned in Section 5.1) over this period was summed up and then the players where divided into 20 cohorts. The first cohort corresponds to the players spending less than the $5th$-percentile, the second cohort the once that spent more that the $5th$-percentile but less then the $10th$-percentile and so on until the last cohort, the whales, which corresponds to the players spending more than the $95th$-percentile. Each bar on the histogram corresponds to one cohort. The histogram exhibited exponential-like growth and hence the height difference between the last and the second last bar was much larger than between the second last bar and the bar before. This showed clearly that one group of players spend the most and therefore the conclusion is that this definition is good for Paradox whales.

Question 2:
What is a good way to define if a Paradox whale has churned or not?

"A Paradox whale has churned if he has not bought anything for 28 days" was the definition that was used in this thesis. The definition was decided according to the distributions of days since last purchase for the players where the end date was 2018-12-31 (since data was collected until then). This approach has its drawbacks since this distribution of days since last purchase would look different if the end date was for example 2018-07-31 instead of 2018-12-31. Defining the churn event is a difficult task since many factors have an impact on when players buy, for example seasons and releases of game products. A further investigation on the players purchase behaviours, releases and seasonal fluctuations would have to be taken into account in order to get a better churn definition.

Question 3:
Which supervised machine learning model gives the best performance in predicting the probability of a whale churning?

The model that performed best in terms of AUC was Random Forest with $AUC = 0.7162$. This method outperformed both L1 Regularized Logistic Regression and Decision Tree. It was expected that the performance of the DT would be worse than RF since RF is an ensemble method that combines many DT and averages them. The reason for L1 Regularized Logistic Regression performing worse than RF could be that the linear relationship Logistic Regression is assuming does not explain the data well.

The accuracy for Random Forest was highest when the probability threshold was 0.5 (Table 6.5), but as stated in Section 4.3 accuracy is not the most interesting evaluation metric. The main focus was on having a balance between sensitivity and specificity and trying to get both metrics as high as possible. The trade-off between FP and FN, i.e. marketing costs for retaining players and the loss when players churn, has to be considered when an appropriate probability threshold is chosen. The focus is on retaining players that are likely to churn, hence the focus should be more on minimizing the FN. For the thresholds shown in Table 6.5, 0.4 is probably the best choice since sensitivity is high, 0.7335, and specificity is higher than random, 0.5689.

7.2 Further studies

Investigating the definition of churn and collecting more data about the whales could possibly improve the performance of the model.

Defining churn is a difficult task when there is not a contractual relationship between the customers and the company since human behaviour can be random. The definition used in this thesis only looked at how many days had passed since the players last made a purchase. It is obvious that more things have to be taken into account to make the definition better. One important thing that should be looked into is whether a whale has already bought everything available from the inventory and that is the reason for why the player has not bought anything for 28 days. In addition seasonal fluctuations and peaks in spending because of releases should be taken into account.

More data about the whales could possibly improve the performance of the model. No information about player behaviour within each game was used in this work. Having information about weather the player is doing well or not and more game specific data probably play a big role in weather the players stop spending much on Paradox PC products or not.

Another suggestion is to look at one PC game at a time. The games can be really different and some games have more products that are possible to buy than others. It is assumed that this has a great impact on both the purchase and playing behaviour of the players. Therefore it might be better to make separate churn models for each game.

It would also be interesting to do a further investigation on the cohorts close to the whale cohort. These players spend a lot of money but not enough to be defined as whales. It might be possible to turn these players into whales.

Finally, more machine learning methods should be tried such as SVM, Naïve Bayes and Gradient Boosting as suggested by literature (Section 3.4). It might be the case that these methods manage to capture the patterns in the data better than the methods tried here.

7.3 Conclusion

The conclusion is that it seems to be possible to predict the probability of churning for Paradox whales. The model that gave the best performance was Random Forest with $AUC = 0.7162$. It is concluded that this model can be a good basis for a churn model for Paradox. The performance of the model is acceptable but can be further improved. By investigating the definition of churn, collecting more game specific data about the players and try other machine learning methods it might be possible to achieve better performance.

Bibliography

[1] M. Seif El-Nasr, A. Drachen, and A. Canossa, *Game Analytics Maximizing the Value of Player Data.* Springer, 2013. [Online]. Available: https://link-springer-com.focus.lib.kth.se/content/pdf/10.1007%2F978-1-4471-4769-5.pdf

[2] V. Kumar and J. A. Petersen, *Statistical Methods in Customer Relationship Management.* John Wiley & Sons, Ltd, 2012. [Online]. Available: www.wiley.com.

[3] P. Bertens and A. Guitart Andafricaand Andafrica Periáñez, "Games and Big Data: A Scalable Multi-Dimensional Churn Prediction Model," *IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 33–36, 2017. [Online]. Available: https://arxiv.org/pdf/1710.02262.pdf

[4] M. Miloševí, N. Živí, and I. Andjelkoví, "Early churn prediction with personalized targeting in mobile social games," *Expert Systems With Applications*, vol. 83, pp. 326–332, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2017.04.056

[5] E. Lee, B. Kim, S. Kang, B. Kang, Y. Jang, and H. K. Kim, "Profit Optimizing Churn Prediction for Long-term Loyal Customer in Online games," 2018. [Online]. Available: http://www.ieee.org/publications_standards/publications/rights/index.html

[6] "About us | Paradox Interactive." [Online]. Available: https://www.paradoxplaza.com/about-us-static-info-ca.html

[7] "Our business | Paradox Interactive - Global Games Publisher." [Online]. Available: https://www.paradoxinteractive.com/en/our-business/

[8] “Annual Report 2018,” Tech. Rep., 2019. [Online]. Available: www.paradoxinteractive.com

[9] “This is Paradox | Paradox Interactive - Global Games Publisher.” [Online]. Available: https://www.paradoxinteractive.com/en/this-is-paradox/

[10] “White Wolf | Paradox Interactive - Global Games Publisher.” [Online]. Available: https://www.paradoxinteractive.com/en/white-wolf/

[11] “Revenue model | Paradox Interactive - Global Games Publisher.” [Online]. Available: https://www.paradoxinteractive.com/en/revenue-model/

[12] “Paradox: "If a game can’t be played for 500 hours we probably shouldn’t be publishing it".” [Online]. Available: https://www.gamesindustry.biz/articles/2018-05-31-paradox-if-a-game-cant-be-played-for-500-hours-we-probably-shouldnt-be-publishing-

[13] “Game Pillars | Paradox Interactive - Global Games Publisher.” [Online]. Available: https://www.paradoxinteractive.com/en/game-pillars/

[14] “Sign up | Paradox Interactive Forums.” [Online]. Available: https://forum.paradoxplaza.com/forum/index.php?register&_ga=2.160588602.646327372.1556914089-1315729764.1556914089

[15] T. Vafeiadis, K. Diamantaras, G. Sarigiannidis, and K. Chatzisavvas, “A comparison of machine learning techniques for customer churn prediction,” 2015. [Online]. Available: http://dx.doi.org/10.1016/j.simpat.2015.03.003

[16] B. Zhu, B. Baesens, and S. K. L. M. Vanden Broucke, “An empirical comparison of techniques for the class imbalance problem in churn prediction,” *Information Sciences*, vol. 408, pp. 84–99, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2017.04.015

[17] J. Runge and P. Gao, *Chum Prediction for High-Value Players in Casual Social Games.* [Online]. Available: https://ieeexplore-ieee-org.focus.lib.kth.se/ielx7/6919811/6932856/06932875.pdf?tp=&arnumber=6932875&isnumber=6932856&ref=

[18] Periáñez, A. Saas, A. Guitart, and C. Magne, “Churn Prediction in Mobile Social Games: Towards a Complete Assessment Using Survival Ensembles,” 2016. [Online]. Available: http://www.yokozunadata.com/research/churn.pdf

[19] E. G. Castro and M. S. G. Tsuzuki, “Churn Prediction in Online Games Using Players’ Login Records: A Frequency Analysis Approach,”

IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES, vol. 7, no. 3, p. 255, 2015. [Online]. Available: http://www.ieee.org/publications_standards/publications/rights/index.html

[20] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2001. [Online]. Available: https://link-springer-com.focus.lib.kth.se/content/pdf/10.1007%2F978-0-387-21606-5.pdf

[21] G. James, D. Witten, T. Hastie, and R. Tibshirani, *Springer Texts in Statistics An Introduction to Statistical Learning.* Springer, 2013. [Online]. Available: http://www.springer.com/series/417

[22] S. Marsland, *Machine Learning: An Algorithmic Perspective*, 2nd ed. CRC Press LLC, 2014. [Online]. Available: https://ebookcentral-proquest-com.focus.lib.kth.se/lib/kth/reader.action?docID=1591570

[23] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006. [Online]. Available: www.elsevier.com/locate/patrec

[24] N. Patel, S. P. B. Patel, and A. Prof, "Study of Various Decision Tree Pruning Methods with their Empirical Comparison in WEKA Saurabh Upadhyay," Tech. Rep. 12, 2012. [Online]. Available: https://pdfs.semanticscholar.org/025b/8c109c38dc115024e97eb0ede5ea873fffdb.pdf

[25] W. O. Library, T. Dankowski, and A. Ziegler, "Calibrating random forests for probability estimation," 2016. [Online]. Available: https://onlinelibrary-wiley-com.focus.lib.kth.se/doi/pdf/10.1002/sim.6959

[26] C. Shearer, "The CRISP-DM Model: The New Blueprint for Data Mining," *Journal of Data Warehousing*, vol. 5, pp. 13–22, 2000. [Online]. Available: www.dw-institute.com/journal.htm

[27] G. Kraljević and S. Gotovac, "Modeling Data Mining Applications for Prediction of Prepaid Churn in Telecommunication Services," *Automatika*, vol. 51, no. 3, pp. 275–283, 4 2018.

[28] D. A. Wolfe and G. Schneider, *Intuitive Introductory Statistics.* Springer, 2017. [Online]. Available: http://www.springer.com/series/417

[29] "nearZeroVar function | R Documentation." [Online]. Available: https://www.rdocumentation.org/packages/caret/versions/6.0-84/topics/nearZeroVar

[30] T. Hastie and J. Qian, “Glmnet Vignette,” Stanford, Tech. Rep., 2014. [Online]. Available: http://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html#intro

[31] T. M. Therneau and E. J. Atkinson, “An Introduction to Recursive Partitioning Using the RPART Routines,” Tech. Rep., 2019. [Online]. Available: https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf

TRITA -SCI-GRU 2019:254