DEGREE PROJECT IN MATHEMATICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Predicting Customer Churn Rate in the iGaming Industry using Supervised Machine Learning

LOVISA GRÖNROS

IDA JANÉR

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Predicting Customer Churn Rate in the iGaming Industry using Supervised Machine Learning

LOVISA GRÖNROS
IDA JANÉR

Degree Projects in Financial Mathematics (30 ECTS credits)
Degree Programme in Industrial Engineering and Management
KTH Royal Institute of Technology year 2018
Supervisor at Mr Green: Urban Edlund
Supervisor at KTH: Henrik Hult
Examiner at KTH: Henrik Hult

TRITA-SCI-GRU 2018:172
MAT-E 2018:29

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

Prognostisering av kundbortfall inom iGaming-industrin med användning av övervakad maskininlärning

Sammanfattning

Mr Green är en av de ledande onlinespelsleverantörerna på den europeiska marknaden. Deras mission är att erbjuda underhållning och en överlägsen användarupplevelse till sina kunder. För att bättre kunna förstå sina kunder och deras livscykel är kundbortfall ett ytterst viktigt koncept. Det är också ett viktigt mått för att kunna utvärdera resultaten av marknadsföring. Denna rapport analyserar möjligheten att, med 24 timmars data över kundbeteende, kunna avgöra vilka kunder som kommer att lämna siten. Detta görs genom att undersöka olika modeller inom övervakad maskininlärning för att avgöra vilken som bäst fångar kundernas beteende. Modellerna som undersöks är logistisk regression, random forest och en linjär diskriminantanalys, samt två olika sammansättningsmodeller som använder sig av stacking och voting. Resultatet av denna studie är att en sammansättningsmodell som väger modellerna logistisk regression, random forest och en linjär diskriminantanalys ger den högsta förklaringsgraden på 75.94 %.

Abstract

Mr Green is one of the leading online game providers in the European market. Their mission is to offer entertainment and a superior user experience to their customers. To be able to better understand each individual customer and the entire customer life cycle the concept of churn rate is essential, which is also an important input value when calculating the return on marketing efforts. This thesis analyzes the feasibility to use 24 hours of initial data on player characteristics and behaviour to predict the probability of each customer churning or not. This is done by examining various supervised machine learning models to determine which model best captures the customer behaviour. The evaluated models are logistic regression, random forest and linear discriminant analysis, as well as two ensemble methods using stacking and voting classifiers. The main finding is that the best accuracy is obtained using a voting ensemble method with the three base models logistic regression, random forest and linear discriminant analysis weighted as $w = (0.005, 0.80, 0.015)$. With this model the attained accuracy is 75.94 %.

Acknowledgements

We would like to thank Professor Henrik Hult at the school of Mathematics at the Royal Institute of Technology for his guidance throughout this thesis project. We would also like to express our most sincere gratitude to Urban Edlund and Jiri Pallas at Mr Green for their deep knowledge and input within the area of data analysis and business insights, as well as Mattias Wedar for this thesis opportunity. Lastly, we would like to thank all our colleagues at Mr Green, especially Ulrika Hedbäck and Linn Wedin, for creating a fantastic work atmosphere and stimulating hard work and the strive to be a cut above the rest.

Table of Contents

List of Tables

List of Figures

1 Introduction

With an increasing interest in data and the ability to analyze it, many companies today hold a vast amount of data which, by the use of machine learning, enables new business opportunities. There are many applications within the area of machine learning and the overall objective is often to use existing data and learn from it to adapt to future behaviour (Kotsiantis, 2007). The iGaming company Mr Green has a complete digital product offering and thus possesses a large amount of data which could be used to better understand the player and enhance the user experience. An important area of implementation is marketing, where the goal is to become more relevant in the marketing strategies by understanding which customers to target and at what time.

1.1 Problematization and Research Questions

Mr Green has noticed that the first couple of days after a player has validated his or her account are crucial when it comes to creating a loyal customer, as a substantial amount of all customers churn during these days. Thus, being able to determine which customers are likely to churn as fast as possible is essential. This insight can facilitate personalized marketing, with the objective of increasing customer satisfaction and aiming for a more efficient use of marketing resources.

Therefore, this thesis focuses on investigating how player churn rate could be predicted using supervised machine learning. This is done by answering the following research questions:

- To what extent is it possible to predict the probability of each customer churning, using data from the first 24 hours after the player's first deposit?
- Which supervised machine learning model explains the player's churn behaviour the best?

1.2 Purpose

To be able to compete in a highly competitive industry it is important to understand the customers and their demands. Considering the wide range of games that Mr Green offers, in terms of different volatility and winning characterizations, this becomes even more essential. Machine learning enables Mr Green to get to know its customers, and to make data driven decisions, which becomes crucial in creating a more personalized gaming experience. A part of this movement towards a more data driven business strategy is to attract and retain customers at the site. This study's purpose is therefore to use a mathematical approach to develop and evaluate a model which can be used by Mr Green to enhance its marketing efficiency and decrease the customer churn rate.

1.3 Scope

The scope of this paper includes creating and training a machine learning model to predict which customers will churn. More specific, this thesis covers various supervised machine learning algorithms where the goal is to solve a binary classification problem. The implemented algorithms are logistic regression, random forest and linear discriminant analysis. Additionally, two different ensemble methods analyzing different combinations of the above mentioned base models are implemented. Furthermore, areas that might be of interest to the analysis, but lie outside of the scope of this paper, are presented as future work in Section 7.3.

1.4 Terminology

The term iGaming is a rather modernistic definition of the online gaming industry with a refreshed and more responsible touch. In this thesis, churning is defined as a customer not returning to the website within 30 days after her or his first deposit. A customer who does not churn is labeled as a 1 or positive, and a customer who churns is labeled as a 0 or negative. The different terms will be used interchangeably throughout this paper.

1.5 Outline

In section two we discuss the background to the problem by giving a brief introduction to Mr Green and the iGaming industry, as well as explaining how our thesis work will be a part of the business and impact the way Mr Green wishes to move forward. In section three we will present the data that was used as well as the data handling and data preparation. In section four we give the reader the essential mathematical theory and definitions to be able to follow the subsequent sections. Section five will cover a brief explanation of the methods used and the results are then presented in section six. Finally, section seven concludes the paper by discussing the problem, results and implementations.

2 Background

2.1 Mr Green

Mr Green has been a leading actor in the European online gaming industry since it was founded in 2007. The mission is to offer first class entertainment and a superior user experience. Mr Green views its product as a type of entertainment where the players play for the sake of having fun (*Mr Green & Co, 2017 Annual Report* 2018). Therefore, Mr Green focuses on creating a safe and trustful environment where the players are in control of their own gaming behaviour. Due to the strong competition within the iGaming industry it is of great importance to combine a strong brand and unique product offering with social responsibility to secure long term financial growth.

Mr Green possesses a broad spectrum of products. This includes casino games, sportsbook, number games and the world's first virtual casino called Live Beyond Live, as well as their Green Gaming tool, to help players keep track of their own gaming behaviour (ibid.). In 2017, Mr Green acquired Dansk Underholdning and Evoke Gaming to enter new geographical markets as well as increasing its product offering with brands like Redbet, Vinnarum Casino, Bertil and Mamma Mia Bingo. Mr Green is present in twelve different markets: Sweden, Denmark, Finland, Norway, Ireland, the Netherlands, Switzerland, United Kingdom, Germany, Italy, Malta and Austria, as well as an international web domain with players from around the world.

2.2 The iGaming Industry

In 2017, the European iGaming industry had sales of 970 billion Swedish crowns (ibid.). The industry is fast growing, and includes numerous of sub fields, ranging from the traditional casino-like features and sports betting to e-sports betting. There is however an important feature to the industry, namely that it is still in some sense an oligopoly in Sweden. AB Svenska Spel, including Casino Cosmopol and slot machines, ATG and Riksgäldens Premieobligationer are currently the only actors that have been authorized by the Swedish government to run betting and gaming services in Sweden (*Om tillstånd* 2017). The government, and the authority Lotteri

inspektionen, issue licenses to actors who wish to run any type of gaming or betting service, both physically and online. The laws Lotterilagen (1994:1000), Kasinolagen (1999:355) and Automatspelslagen (1982:636) together stipulate to whom a license should be given and how, along with regulations those actors must comply with. A license to run online poker, for instance, is issued by the Swedish government directly (*Tillstånd och spelformer* 2017). Despite this, due to the European Union article of free movement of services (Article 56) and the freedom of establishment (Article 49), companies can establish themselves within the EU and offer their services to the European population. Consequently, other companies than the ones mentioned above can offer their services in Sweden, often referred to as offshore companies. However, in the fall of 2015, the Swedish government decided to conduct an investigation on the possibilities of opening the Swedish online gaming industry to new actors, allowing them to apply for a license in Sweden (*Omreglering av Spelmarknaden* 2017). In March 2017, the investigation was presented, proposing to introduce a new system for licenses open for applications by actors in the industry, which will be in place late 2018 or early 2019. Along with many other iGaming companies, Mr Green is preparing to apply for a Swedish license.

The iGaming industry is characterized by heavy competition among the many actors in the market, since customers tend to play interchangeably between different casinos and betting sites. Consequently, the concept of churn and customer retention becomes vital. An important area of application is marketing, where efforts could be focused on customers predicted to churn to increase the return on marketing resources (Coussement and De Bock, 2013). A way to approach this issue is to look at past customer behaviour, as it has been suggested that it reflects future behaviour, that is a customer churning or not (Jolley et al., 2006).

2.3 Contribution

Mr Green works intensively with monitoring its customers, and the so called player life cycle. This enables Mr Green to target the customers at the right time, in terms of marketing efforts. It is thus the hope that this thesis work at Mr Green will provide useful insight in customer behavior and the corresponding churn rate.

3 Data

The data on which the analysis was performed was made anonymous by Mr Green by assigning fictional customer ID-numbers to the customers, to ensure customer data privacy. Any sensitive information was also removed. As it was not possible for Mr Green to give out access to its data warehouse, specific data sets used in the analysis were extracted by our supervisors.

Two of the most essential components in data analysis is to understand the data, or having some sort of domain knowledge, along with securing the quality of the data. Building an analysis, and training a model on incorrect data would lead to errors in classifying new data, which could potentially lead to severe consequences. As for domain knowledge, it is often useful as an initial way of performing manual feature selection or even as an input to the model after manually clustering features. Moreover, this knowledge might also bring additional colour when analyzing the results. Hence, a substantial amount of time was spent understanding the product, the company and all the available features.

3.1 Mr Green's database

To perform this analysis, data on both a customer and transactional level was used. As the analysis was performed on a customer level, i.e one customer representing one observation, the transactional data had to be summarized to fit the customer level data. On a customer level, data is stored in terms of demographics, such as gender, age and country, and so called event information. The event information includes the first game played, the most played channel out of web, mobile web, iOS or Android and the date and time when they made their first deposit et cetera. On a transactional level, the data is stored in terms of logical transactions, consisting of multiple events, as opposed to real transactions which by definition is the equivalent of a single event or transaction. An example of this is a customer making a bet in a slot machine game, which together make up the real transaction. The corresponding logical transaction would then consist of the events of the customer making a bet, the transaction being processed, the outcome of the bet and the settlement of the bet. For this particular analysis, it is the real transactions that were of interest.

3.2 Features

To a large extent, transactional data is related to monetary transactions. This includes features such as the total number and the average size of deposits, number of bets and withdrawals et cetera. A natural consequence from storing transactional data is correlation in the data set. For instance, a player who has made a lot of bets will by definition have made a substantial amount of bets in at least one of the channels web, mobile web, iOS or Android, as all bets go through one of these categories. It is also likely that this player spent a rather high amount on bets in total, given that her or his average bet size is not materially small. In order to avoid bringing this correlation into the model, many of the variables were constructed to reflect an average over all of the customer's bets. For instance, the number of total bets was included as a feature as it is, whereas attributes such as the total money spent on or won from bets, the number of losses or gains, the bets made on different channels or different game categories were recorded as averages to the number of bets made. Likewise, the bets made during the day were divided in shares between four 6-hour long intervals, starting at midnight.

In addition, to summarize all transactional data into customer-level data, there is also a time dimension in the transactional level data, that is not so easily translated to a customer level. For instance, instead of simply looking at the total number of games played, one might be interested in how often a player switches between games, or what happens after their balance has reached zero, as opposed to looking at the total number of times that this happens. A way to achieve this is to look at online sessions, and summarize key statistics from this time period. After consulting with people at Mr Green, it was decided to define an online session as a session of continuous activity, with no break longer than 20 minutes. This also allows for analysis of the activity during the day, in terms of how many online sessions each customer had, and how the outcome, in terms of betting result, varied with the sessions.

As the data set contained observations (customers) from different countries, it is reasonable to expect that the bet size would differ between the countries. To account for this, all monetary variables were normalized using a coefficient reflecting the disposable income per country, as to make those variables directly comparable.

3.3 Data Processing

The data processing, in terms of modifying relevant features from the data set, was done using the software R, using packages such as `plyr`, `dplyr`, `tidyr` and `zoo`. Using R, data objects are stored as so called dataframes, which have the structure of a relational database. The output from the data processing in R was a data set on customer-level, including all relevant features. This of course includes a denormalization of existing databases, when joining different sub-tables together.

4 Theory

4.1 Supervised Machine Learning

Machine learning comprises a set of techniques to better understand large amounts of complex data (James et al., 2013). The aim is to find patterns and relations in the data to provide useful insights to a problem.

Machine learning algorithms are normally divided into three different categories: supervised machine learning, unsupervised machine learning and reinforcement learning. Supervised learning can then be divided further into regression and classification problems. This thesis will focus on the latter, that is a supervised classification problem. We start by defining this concept. In supervised machine learning a model is trained on a set of observations with the correct responses already provided, to eventually be able to predict these responses for previously unseen data (Marsland, 2012). The training often consists of numerically minimizing some cost function. In other words, supervised machine learning seeks to model the relation between the predictor variables, used interchangeably with the names features or attributes, and the outcome, or response variable. Mathematically, this is represented by the relation below

$$Y = f(X) \tag{1}$$

where Y is the response variable that we want to predict, X represents the features impacting the outcome and $f()$ is the function mapping the features to the response variable (James et al., 2013). In reality however, this relation might be difficult to model, and there is almost always an error term present, although very small in a good model. In classification problems, the response variable Y has a qualitative value, rather than a quantitative value, as is the case in regression problems. More precisely, each observation (x_i, y_i) belongs to a specific class k, which we try to find in supervised machine learning. To decide which class a certain observation belongs to, there exists different techniques, or models, estimating so called decision lines, to separate the different classes. In most cases it is the data and the problem itself that decide which model works best for those specific circumstances. In supervised machine learning, these models can be divided into two categories, parametric and

non-parametric models (James et al., 2013).

Parametric models assume that the relation $Y = f(X)$, and thus the decision line, have a specific form, for instance linear, quadratic, log-linear or radial. The problem of estimating the relation is thus reduced to estimating a set of parameters.

Non-parametric models on the other hand do not make any assumptions about the relation between the predictors and the response variable. As such, they do not reduce the problem of estimating the relation $Y = f(X)$ to estimating a set of parameters, as is the case with parametric models. Consequently, more observations are required to train a non-parametric model than a parametric model (ibid.).

As previously stated, a model should be chosen based on the data and the problem definition. There are however some advantages and disadvantages to both parametric and non-parametric models. As parametric models assume a specific relation between the predictors and the response, they tend to better fit the data if this relation actually corresponds to the relation assumed. However, if the data cannot be modeled according to a specific relation, a parametric model will perform worse, as it forces a relation that does not actually exist. Parametric models are therefore less flexible, as they model the reality according to some template. When the relation in the data is difficult to estimate, non-parametric models often perform better, as they are more flexible, and can thus be adjusted to better fit the actual relation in the data (ibid.). Generally, the optimal model is neither too flexible nor too strict, as both are prone to errors. This phenomenon, known as the bias-variance trade off, will be discussed further in Section 4.2.1.

4.2 Model Evaluation

4.2.1 The Bias-Variance Trade off

When training a model on a data set with the goal of being able to make predictions and classify unseen data, we want to be as accurate as possible, and minimize the error. A measure of the prediction error is the mean squared error, or the MSE. Although it is mainly used in regression problems, the principle can be generalized to classification problems as well. As a fact, the MSE can be broken down into three

components, as shown below

$$E\Big(y_0 - \hat{f}(x_0)\Big)^2 = Var(\hat{f}(x_0)) + [Bias(\hat{f}(x_0))]^2 + Var(\epsilon) \tag{2}$$

That is, the error consists of the variance of the prediction $\hat{f}(x_0)$, the squared bias of $\hat{f}(x_0)$ and the variance of the error term ϵ. The variance of the prediction is by how much the function $\hat{f}$ would change if it were estimated on a different training dataset, and the bias is the error that occurs from approximating a complex problem with a model that is too simple (James et al., 2013). The two go hand in hand, in the sense that if one of them is minimized, the other will grow. If a model is made very complex, and fit very well to the training data to reduce the bias, then that model will incorporate a lot of the randomness in that specific data set. This means that it will perform poorly when classifying unseen data, as that data might not have the same individual randomness as that of the training data. This problem is known as overfitting and will increase the variance in the model. On the other hand, if a model is made very simple, and not fit as well to the data, in order to reduce the variance, it is likely that this model too will perform worse in classifying new data, as a consequence of simplifying a more complex problem. In general, models that are more flexible will have a high variance and low bias, and models that are less flexible will have a high bias and low variance (ibid.).

Overfitting may also occur as a consequence of small datasets. A remedy to this is to use cross-validation, which is a type of resampling technique. Cross validation means splitting the training data into two or more subsets, and then training the model on all but one subset at a time, using the left out subset as a validation set, to estimate the test error. As cross-validation fits the model to multiple datasets and then averages the error, it can also be used as a validation technique, which can be compared to the actual test error, to ensure robustness of the model. A common approach is to use k-fold cross validation, which involves splitting the data into k different subsets. The model is then trained on $k - 1$ subsets, leaving one subset out as validation set (ibid.). This step is repeated for all k subsets, and test error is then computed as the average of the classification errors for all k left out samples, called the holdout in Figure 1 (Marsland, 2012). In k-fold cross validation, the test

error is computed as

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i \tag{3}$$

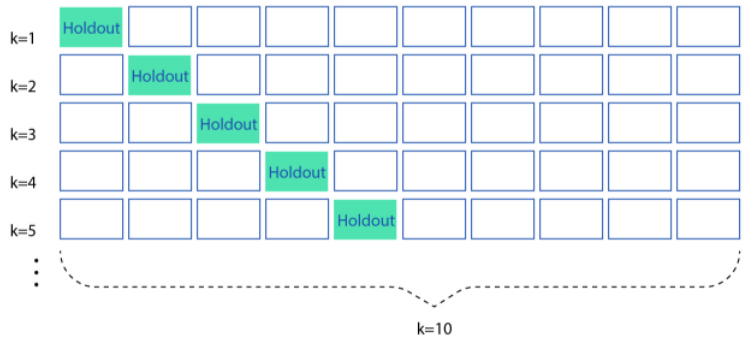

Figure 1: The process of cross validation

4.2.2 The Confusion Matrix

In addition to training a model to be as accurate as possible, which represents the absolute error rate, one might also be interested in analyzing the misclassifications further. Classifying an observation as negative when it is actually positive is known as a false negative (FN), and likewise, classifying an observation as positive which is actually negative is referred to as false positives (FP). Observations that are correctly classified are simply referred to as true positives (TP) or true negatives (TN) (Marsland, 2012). Depending on the problem, the two types of misclassifications may have more or less severe consequences. For instance, when classifying diseases among patients, one wants to keep the false negative rate as low as possible, to avoid sick patients being diagnosed as healthy. In the problem examined in this paper however, we want to keep the false positive rate as low as possible, as the aim is to identify players who will churn.

Due to the discussion above, a key feature in selecting the best model is to find one, which not only produces a high test accuracy, but also manages to keep the rate of some specific false prediction low, whether it be false positives or false negatives. A common approach in evaluating different models based on this aspect is the usage of the confusion matrix. The confusion matrix provides a convenient illustration of how many of the observations are classified correctly. If we let k be the

number of classes in a classification problem then the confusion matrix is a k-by-k square matrix. The predicted classes are shown along the horizontal axis and the vertical axis show the actual target values (Marsland, 2012). The diagonal elements show the number of correct predictions. See Table 1.

		Prediction outcome		
		0	1	total
Actual value	0	True Negative	False Positive	N
	1	False Negative	True Positive	P
	total	N′	P′	

Table 1: The confusion matrix for a binary response variable

When assigning an observation to a certain class based on the computed class probability, the default threshold is $p = 0.5$ in the case of a binary response variable. That is, if an observation has a probability greater than 0.5 of belonging to a certain class, say class 1, then that observation will be classified as a 1. Mathematically, if $P(Y = 1) > p$ where $p = 0.5$ we will classify Y as being from class 1. By adjusting this threshold, one can adjust the number of false positives and false negatives. By setting the threshold p larger than 0.5, the model will be stricter in terms of assigning observations to class 1, and thus decreasing the number of false positives. It is important to note however, that whereas we decrease the number of FPs, we will increase the number of FNs, as more observations will be labeled as negatives.

4.2.3 Feature Selection

Feature selection, or dimensionality reduction is the process of reducing the number of dimensions in the model, where each predictor represents one dimension. Depending on the data, this can be an important procedure. As previously discussed in Section 4.2.1, a model that is too complex runs the risk of being overfitted to the data, thus increasing the variance. Having too many predictor variables increases

the complexity of the model, and accordingly, the variance might also increase. Furthermore, having many predictors often decreases the interpretability of the model, making it less useful in practice (James et al., 2013). Therefore, it is a common approach to select a subset of the best features.

There exists a number of techniques to reduce the dimensions. Some techniques, such as principal component analysis (PCA) involve projecting the p predictors onto an D-dimensional subspace, where $D < p$. This is done by finding D linear combinations of the p predictors, and thus creating a new set of predictors. Other techniques, so called regularization or shrinkage techniques train the model using all predictors, but then shrinks all coefficients, making some close to or equal to zero. In other words, the least important predictors, that is those whose coefficients are small, will be removed from the model. Finally, there are techniques to select the best subset of features, repeatedly fitting the model to different subsets of the predictors and evaluating the test accuracy to find the best subset (ibid.). The latter approach will be used in this thesis project.

Ideally, to find the best subset, one would have to fit the model to all combinations of predictors. In practice however, this becomes impossible, as there are approximately 2^p different combinations of predictors, for a model containing p predictors. An alternative to this technique is to use a greedy approach, such as forwards or backwards stepwise selection. Forward stepwise selection starts with a model without any predictors, and then adds one predictor at a time to the model. At each step, all remaining predictors are considered and the one that results in the best fit of the model is added. This means that for each model containing i predictors $M(i)$, the model obtained will have the best fit, using $i = 1, ..., p$ predictors. The p models are then evaluated against each other, using the cross-validated prediction error, to find the best model. An important drawback of forward stepwise selection is that it does not evaluate all possible models. For instance, the variable that constitutes the best 1-variable model will be in all of the subsequent models, which will affect the impact on the goodness of fit that the next variables added will have. The variable that is in the best 1-variable model might not actually be in the best k-variable model, but since that variable has already been included in the model, it will stay in the k-variable model. Backwards stepwise selection works the same way, only the process is backwards, starting with a model containing all predictors, and then

removing one at a time (James et al., 2013).

4.3 Logistic Regression

4.3.1 Single Predictor

Logistic regression is derived from linear regression, as an alternative approach in order to predict qualitative output values (ibid.). Logistic regression captures the probability that the response variable, denoted Y, belongs to a particular class. It is a parametric approach, which assumes a linear relation between the predictors and the log-odds of the response variable. Logistic regression builds on linear regression, although the model is modified to create probabilities as outputs, that is outputs that lie in the interval $[0, 1]$. Thus, in the case of a single predictor, the logistic function is used to model probabilities

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \tag{4}$$

The logistic regression produces an S-shaped curve, ensuring that the output lies in the required interval. Re-writing the model as below, we obtain the odds, which is the ratio of probabilities between classes (left-hand side of the equation below).

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0+\beta_1 X} \tag{5}$$

Taking the logarithm of both sides, the log-odds is obtained, which has a linear relation to the predictors.

$$log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X \tag{6}$$

Due to the linear relation in equation 6, a one-unit change in X will change the log-odds by β_1. The size of the change in the actual probability, $p(X)$ however, will depend on the size of X, since the relation between $p(X)$ and X is not linear (equation 4). Additionally, if β_1 has a positive value, then an increase in X will lead to an increase in $p(X)$, and if β_1 is negative an increase in X will lead to a decrease in $p(X)$.

The parameters β_0 and β_1 are estimated using maximum likelihood. The betas $\hat{\beta}$ are estimated so that they maximize the maximum likelihood function below

$$l(\beta) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_i'=0} (1 - p(x_{i'})) \tag{7}$$

where $p(x_i)$ is the probability of $Y = 1$ for the ith observation and $p(x_{i'})$ is the probability of $Y = 0$, for the same observation. Once the betas have been estimated, the model can be used to make predictions on test data (James et al., 2013).

4.3.2 Multiple Predictors

In the case of more than one predictor, the probability $p(X)$ is defined as

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}} \tag{8}$$

where $X = (X_1, ..., X_p)$ is a vector consisting of p predictors. The log-odds ratio then becomes

$$log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p \tag{9}$$

Just like in the single-predictor case, maximum likelihood is used to fit the model to the data and estimate the betas. It is important to note however, that in the multiple-predictor case, the model risks being exposed to correlation between the predictors, which might impact the predictions. It is therefore important to perform a correlation analysis of the predictors, and remove those that are too heavily correlated (ibid.).

4.4 Linear Discriminant Analysis

4.4.1 Univariate Gaussian for One Predictor

Linear discriminant analysis, LDA, assumes a linear relation between the predictors and the response variable, and thus it produces a linear decision line, or decision boundary. Specifically, LDA seeks to model the distribution of the predictors, X, for each possible class, and then uses Bayes theorem to estimate the conditional probability of an observation belonging to each class, given its predictors (James et al., 2013). This conditional probability is defined as

$$P(Y = k|X = x) \tag{10}$$

The observation is then assigned to the class for which the conditional probability above is the largest. Bayes theorem uses the prior probability of an observation belonging to the k-th class, deonted π_k, along with the estimated density function of X, denoted $f_k(X) \equiv P(X = x|Y = k)$, for an observation from the k-th class, to estimate the posterior conditional probability stated above, that an observation belongs to the k-th class (ibid.). This relation is shown below

$$P(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} \tag{11}$$

In order to estimate $f_k(X)$ we need to make some assumption about its form. A conventional assumption is that it follows a normal or Gaussion distribution. In the case of a single predictor, $p = 1$, the normal density is defined as

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} exp\bigg(- \frac{1}{2\sigma_k^2}(x - \mu_k)^2 \bigg) \tag{12}$$

where μ_k and σ_k^2 are the mean and variance of the k-th class. Often σ_k^2 is assumed to be equal for all k, and is thus denoted σ^2. By plugging equation 12 into equation 11, we get

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2\sigma^2}(x - \mu_k)^2)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2\sigma^2}(x - \mu_l)^2)} \tag{13}$$

where $p_k(x) = P(Y = k|X = x)$ (James et al., 2013). Taking the log of the above equation, and rearranging the terms we obtain

$$\delta_k(x) = x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k) \tag{14}$$

LDA then approximates the parameters π_k, μ_k and σ^2 and plugs these into Bayes classifier, and then assigns the observation to the class for which $\delta_k(x)$ is the largest. The estimation of the prior probability π_k is simply computed as the fraction of the (training) observations that belong to the k-th class, for any random sample of observations from a population. The estimations of the parameters follow below

$$\hat{\pi}_k = \frac{n_k}{n} \tag{15}$$

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i \tag{16}$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2 \tag{17}$$

4.4.2 Multivariate Gaussian for Multiple Predictors

In the case of more than one predictor variable, $p > 1$, that is $X = (X_1, X_2, ..., X_p)$, we assume thus that each observation is drawn from a multivariate Gaussian distribution, that is $X \sim N(\mu, \Sigma)$, where $E(X) = \mu$ is the mean of X and $Cov(X) = \Sigma$ is the covariance matrix of X. μ_k is class-specific whereas Σ is common for all K classes (ibid.). The multivariate Gaussian distribution is then defied as

$$f(x) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} exp\bigg(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\bigg) \tag{18}$$

We then plug the density function for the k-th class into equation 11 and simplify, to get the below equation,

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + log\pi_k \tag{19}$$

where Bayes theorem assigns each observation to the class for which $\delta_k(x)$ is the largest. The estimation of the parameters μ_k and Σ are done in a similar way as in the univariate case, and π_k is computed in the same way. These estimations are then plugged into the above equation, to estimate Bayes theorem (James et al., 2013).

4.5 Decision Trees and Random Forests

Tree-Based models, also called decision trees, are methods that involve arranging and segmenting the predictor space into a number of separable regions. Decision trees are non-parametric models, which means that they do not make any assumption about the form of the relation between the predictors and response variable. As a consequence, trees tend to suffer from high variance, and they are easily overfitted to the training data, making them perform worse than other supervised learning models. There are however some techniques, that involve some form of resampling, to improve the prediction accuracy of the trees. If these resampling techniques are used the tree-based models are comparable to other models. Decision trees can be used in both regression and classification problems, and this section will focus on the latter (ibid.).

4.5.1 Decision Trees

Simple decision trees make classification by stratifying the observations into different regions, depending on their value for some features. Starting from the top of the tree (which is actually the trunk), observations are split into two sub spaces, or nodes, according to some variable threshold. The nodes are then split into new sub spaces according to some other variable threshold, and so on, until some stopping criterion has been reached. The higher the impact a predictor variable has on the response variable, the earlier the tree will split the observations based on that variable. The nodes are connected via branches and the end nodes are referred to as terminal nodes or leaves, whereas the inner nodes are referred to as internal nodes. To make a prediction for a new observation, we simply pass it through the tree and check all splitting criterion, starting at the top, and eventually end up in an end node. The observation is then assigned to the class that represents the majority in that end node (ibid.).

The goal with the splitting of the observations is to create sub spaces, or end nodes, such that the classification error rate is minimized. The classification error rate is defined as the fraction of training observations in an end node that do not belong to the most commonly occurring class in that node, defined as

$$E = 1 - \max_k(\hat{p}_{mk}) \tag{20}$$

$\hat{p}_{mk}$ is the proportion of training observations in the m-th region that are from the k-th class. In practice however, two other measures are often used, one of which is the Gini index, defined as

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{21}$$

The Gini Index measures the total variance across the K classes. The cross-entropy measure is an alternative to the Gini index, as defined below

$$D = -\sum_{k=1}^{K} \hat{p}_{mk}\log\hat{p}_{mk} \tag{22}$$

Both the Gini-index and the cross-entropy measure are minimized when $\hat{p}_{mk}$ is close to 0 or 1, resulting in measures of so called node purity. Node purity is important, as it makes classifications more certain.

Thus, the tree is split at each node, starting at the top when all observations belong to the same node, evaluating which variable split will result in the greatest decrease in the classification error rate, and then makes that split. That is, at a given node, we select the predictor X_j and a threshold s, such that the split of the observations into the sub spaces $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ yields to the largest reduction in the classification error rate. This method is known as recursive binary splitting, which is a greedy approach. It is greedy since at each node, it only considers the next split, and which predictor X_j and threshold s will lead to the greatest reduction in the classification error rate, without taking future splits into account. Although evaluating all possible combination of orders of the variables would have ensured finding the best tree, this approach becomes impossible with a large number of features (James et al., 2013).

A problem with the above approach is that the trees are easily overfitted, and thus suffer from high variance. Fitting smaller trees, i.e stopping the splitting earlier, often instead leads to a higher bias. A solution to this is to grow a large tree, and then to prune it back into a smaller subtree. As with the process of growing the tree, considering every possible subtree is impossible, and we therefore use cost complexity pruning, and select a small number of subtrees to consider. The subtrees are chosen as a function of a tuning parameter α, which controls the bias-variance tradeoff of the subtrees. The best subtree is chosen based on an estimation of the test error, from cross validation. That tree is then chosen as the final classifier (James et al., 2013).

4.5.2 Random Forests

As previously stated, a major disadvantage of decision trees is that they often suffer from high variance, and are thus not able to perform classifications with high accuracy. A remedy for this problem is random forests. Random forests use a form of bootsrapping, to reduce the overall variance, by fitting a decsion tree to each of the bootstrapped datasets and then averaging them. It is important to note that this method relies on the trees not being too correlated, because if they are, then averaging will not lower the total variance (ibid.). This could happen if for instance there is one or multiple predictors with a high impact on the response variable. Then each time those variables are picked from bootstrapping, they will naturally be split high up in the tree, making those trees similar, or correlated. To tackle this, random forests only pick a fraction of the predictors when bootstrapping, in order to produce different datasets, which will result in less correlated trees. The number of predictors that are used in the bootstrapping m is usually $m \approx \sqrt{p}$, where p is the number of predictors (ibid.).

4.6 Ensemble Methods

The idea behind ensemble methods is to use multiple learning algorithms to achieve a higher predictive performance than one could obtain with a single learning algorithm. By applying several different learning algorithms with slightly different results it is possible to create a model that captures the individual strengths of

each algorithm. This kind of learning is called ensemble learning and the challenge consists of determining which learners to use and how to ensure that multiple learners learn different things, otherwise it would not make sense to try to enhance the performance by combining them (James et al., 2013). Some of the most common ensemble methods are presented below

- Voting — This ensemble method is one of the easiest to understand and implement. The technique is based on creating multiple classification models, called base models. Each base model could be created on the same data set with different algorithms, or the same algorithm is used but on different splits of the training data. The models' predictions are then compared, and the ensemble classification is computed as the most common class among the base model predictions (ibid.).
- Stacking - When using stacking the idea is to first create several separate base models and thereafter let the results from these base models be the input values/features to a new meta learning algorithm (*A Kaggler's Guide to Model Stacking in Practice* 2018)
- Boosting - The most common boosting method used today is called adaptive boosting, or simply AdaBoost. This algorithm feeds weights to each data point based on how successful it has been classified in previous runs. The AdaBoost algorithm is conceptually very simple and it will not be applied in this thesis, hence the reader is referred to chapter 13 in An Introduction to Statistical Learning by James et al., 2013 for further reading.
- Bagging - Bagging is a variance reducing algorithm and most commonly used with decision tress. The name Bagging stands for bootstrap aggregating and got its name from the idea of creating a classifier by bootstraping samples from the original data set, i.e sampling with replacement. The benefit of doing this is to receive different learners with slightly different performance, and then averaging the models to a single model. As an example, this method is used in random forests to create one of two forms of randomness in the random forest algorithm (ibid.).

4.6.1 Voting

Voting is used for classification problems and as explained above, the idea is to create several learning algorithms called based models and compare these to find the most likely prediction. For a binary classification problem, let $\hat{y}_i^m$ be the predicted outcome of the i-th observation with base model m and the probability of observation i belonging to class 0 and 1 are denoted $p_{i,0}^m$ and $p_{i,1}^m$, respectively.

One of the ways to combine the base models' predicitons is by using **Majority Voting**. In this approach every model makes a prediction for each observation and the final ensemble prediction is the one that receives the majority of the votes. For example, if $\hat{y}_i^1 = 0$, $\hat{y}_i^2 = 0$ and $\hat{y}_i^3 = 1$ then the majority voting would predict the i-th observation to be a 0. However, if none of the predictions receive more than half of the votes, the ensemble method cannot make a stable prediction for that observation. Therefore it makes sense to use an odd number of base classifiers. If we assume that each individual classifier has a success rate of p, the probability of the ensemble getting the correct answer is a binomial distribution of the form

$$\sum_{m=M/2+1}^{M} \binom{M}{m} p^m (1-p)^{M-m} \tag{23}$$

where M is the number of base models. Due to this, there is a lot of prediction power behind the voting classifier. Even if the single base model only classifies about half of the observations correct, with several classifiers together, the probability of correctly classifying an observation will land closer to 1 since the sum in the equation above approaches 1 for large values of M (James et al., 2013).

Another approach to a voting ensemble algorithm is to take the weighted average probabilities, also called **soft voting**. In contrast to majority voting, soft voting returns the label which has the highest average probability, see Table 2.

Base model	class 0	class 1
classifier 1	$w_1 p^1_{i,0}$	$w_1 p^1_{i,1}$
classifier 2	$w_2 p^2_{i,0}$	$w_2 p^2_{i,1}$
classifier 3	$w_3 p^3_{i,0}$	$w_3 p^3_{i,1}$
weighted average	$\frac{\sum_{k=1}^{T=3} w_k p^k_{i,0}}{3}$	$\frac{\sum_{k=1}^{T=3} w_k p^k_{i,1}}{3}$

Table 2: Probability overview for a soft voting classifier

4.6.2 Stacking

When using stacking as an ensemble method the idea is to first train multiple machine learning algorithms, $L_1, L_2, ..., L_M$ on a data set $S_1, ..., S_M$, which consists of feature vectors $x_{i,j}$ and their labels y_i. The output from the base classifiers denoted $C_1, C_2, ..., C_M$ where $C_i = L_i(S_i)$ are then used to generate a new data set as the input to a meta classifier. The meta classifier could be trained on either the predicted class labels or the class probabilities from the base classifiers. A convenient attribute to stacking is that the base classifiers can be fit to different subsets of the feature space in the training data set (Zenko and Dzeroski, 2004).

4.7 Confidence Intervals

In order to generalize the results obtained from this analysis, confidence intervals could be created for some of the computed statistics, such that the results hold for any given data set. In terms of the model accuracy, a confidence interval was obtained as per below (Blom et al., 2005). Note that the test accuracy refers to the share of correct classifications. If n is the number of observations, and r is the number of correct classifications, then r is a random variable with an approximate binomial distribution with

$$r \sim Bin(p, n) \approx N\Big(np, np(1-p)\Big) \tag{24}$$

where we approximate $\hat{p}$ as

$$\hat{p} = \frac{r}{n} \approx N\left(p, \frac{p(1-p)}{n}\right) \tag{25}$$

A confidence interval for the test accuracy statistic is then defined as

$$\hat{p} \pm 1.96\sigma = \frac{r}{n} \pm 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \tag{26}$$

We can see that as $n \to \infty$ we have that $\sigma \to 0$. We can thus apply the results, i.e. the test accuracy, for new datasets with the above confidence interval.

5 Methodology

5.1 Methodology Setting

Given the structure of the data set, with historical data with correct labels being available, the problem was recognized as a supervised machine learning problem. Since the final model is to be implemented in an actual business environment, the method and solution were carefully agreed upon with stakeholders at Mr Green, to ensure feasibility.

In terms of data processing and an initial feature selection, the latter was done in close collaboration with people at Mr Green, who all posses a wide knowledge within the area of customer behaviour. As described in Section 3, the data processing consisted of cleaning up the data set, creating new features on customer level as opposed to transactional level as well as managing categorical features using one-hot encoding.

For the implementation the programming language Python was used, together with packages such as `NumPy` and `Pandas` which are used for scientific computing and handling and `Scikit Learn` which is used for data analysis and machine learning. An illustration of the process of supervised machine learning is shown in Figure 2. The version control program Github was used to facilitate the process of creating and sharing code between the two authors of this thesis.

Several new features were created from the extracted data set by grouping and creating clusters of the variables. An example of this is the two age groups labeled `age_group1` and `age_group2`, illustrated in Table 3 for a selection of the grouping.

age	age_group1	age_group2
18	<20	18-20
24	20-30	20-25
59	50-60	55-60

Table 3: Examples of the two new age groups created

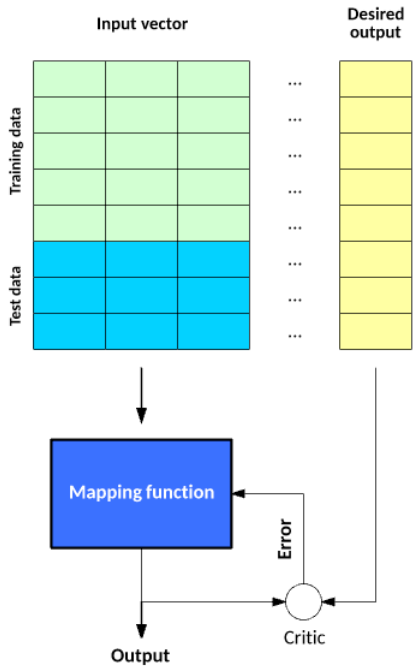

Figure 2: The process of supervised machine learning

5.2 Base Models

The idea of training different models is that the models would capture different attributes of the data set. As previously mentioned, the model performance will depend on the structure of the data set, and so a logistic regression, which assumes a linear relation between the predictors and the log odds, a random forest which assumes no specific relation at all, along with an LDA model, which assumes a linear relation between the predictors and the response variable, were thought to capture different aspects of the data. To strengthen this hypothesis, an initial comparison between different models was made, training models on the full data set with all the available features, and then comparing the test accuracy. The models that were tested were a K-nearest neighbours (KNN) model, with different values of K, quadratic discriminant analysis (QDA), naive Bayes, and a support vector machine (SVM), with different kernels. These models produced a lower test accuracy, and thus they were not chosen. In addition to this, the SVM took a long time to train, making it impractical to use.

The three models were then optimized, by performing both feature selection and

hyperparameter tuning. For the logistic regression and the LDA, recursive feature elimination (RFE) was used, which is the Python implementation of step wise feature selection. For the random forest, a univariate method was used, evaluating model accuracy for different thresholds of the gini value. In addition to this, a principal component analysis (PCA) was considered, to reduce dimensionality, before training the model. However, this was not implemented, due to the loss of interpretability which may occur, when mapping the features space to a lower dimension (James et al., 2013). The tuning of the hyper parameters was then performed on the best feature subset model, for all three base models. It turned out however, that for most of the hyperparameters, there was practically no difference in model performance. The three models, optimized individually, were then used as inputs for the meta models.

5.3 Ensemble Models

When analyzing and comparing the results from the base models, two different ensemble models were used with different approaches, to attempt to improve the accuracy of the model. By merging the separate base models' outcome the risk of misclassifications is decreased.

With the voting classifier approach the three base models were first trained and then the final classification was determined using weighted soft voting. The base models were trained on the same data set, with a large amount of features without any specific feature selection. To be able to find the weight vector $w = (w_1, w_2, w_3)$ which resulted in the highest model accuracy the classifier was trained 60 times with different weight distributions.

When creating the stacking classifier the base models were trained on different data sets, chosen to optimize the individual models' test accuracy. Thereafter, the probability of each observation either belonging to class 0 or 1 for all three models was stored in a dataframe with $3 * 2$ columns. This dataframe was then used as the input data when training the stacking classifier, with a logistic regression model as the aggregating classifier.

6 Results

The original data set consisted of 101 features, out of which 80 were continuous and 21 were categorical. After one-hot encoding of the categorical variables the full data set consisted of 234 features. The continuous variables were scaled when used in the logistic regression and linear discriminant analysis, by removing the mean and scaling to unit variance. The reason for this was to avoid the absolute size of the feature values affecting the sizes of the $\hat{\beta}$ values. In this chapter the results from each model is presented separately in the subsequent subsections.

Some of the techniques used were common for either two or more models. In the feature selection process, RFE is used for both the logistic regression and the linear discriminant analysis. In terms of accuracy evaluation, an assessment of how the models perform using different probability thresholds is done, as discussed in Section 4.2.1. This assessment is the same for all three base models, and the two meta models. Tables of threshold levels, the share of positive and negative misclassifications along with confidence intervals for these shares will be presented in the respective subsection. The confidence interval is computed to make the results applicable to new data sets, in terms of the classification error, given an estimated probability of an observation belonging to a certain class. The classification error is computed as the share of false positives or negatives out of all observations being classified as positive or negatives respectively. This analysis is performed on the best model for each base model, that is after performing feature selection and tuning any hyperparameters. The best final test accuracy for each model is presented in Table 4.

Model name	Accuracy for best model
Logistic regression	0.7336
Random forest	0.7540
Linear discriminant analysis	0.7269
Stacking classifier	0.7523
Voting classifier	0.7595

Table 4: Test accuracy for the three base models and two ensemble models

6.1 Logistic Regression

6.1.1 Feature Selection

Logistic regression is sensitive to interdependence between the features. That is the model should have little or no multicollinearity. Therefore, the first step was to decrease the number of features to reduce the dependencies between features. To analyze how the features correlate to each other a correlation matrix was created, see Figure 3. The features that are highly correlated to other features, represented by yellow and dark blue in the correlation matrix, express the same underlying behaviour as another feature and were therefore removed from the data set.

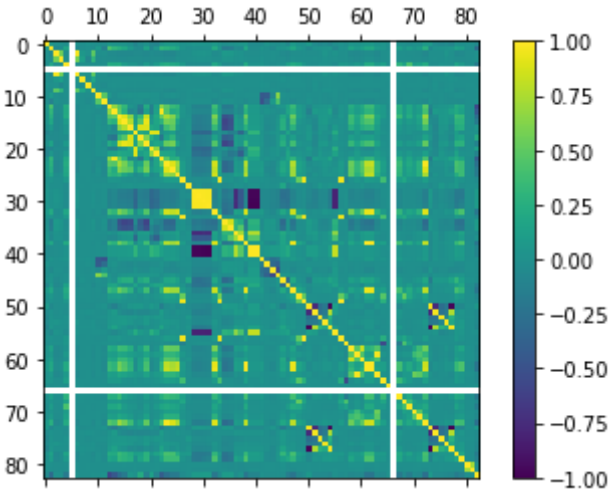

Figure 3: Correlation matrix for the features in the data set

Through the correlation analysis the data set was reduced to 51 features that are relatively uncorrelated to each other (106 with dummies). The model was then created with recursive feature elimination (RFE). For each specified number of features the model was trained and the model scores were compared to determine what the optimal numbers of feature was. The result of a small selection of the RFE analysis with a five-step interval is presented in Table 5.

Number of features	Test accuracy
65	0.7289
70	0.7305
75	0.7309
80	0.7304
85	0.7306
88	**0.7336**
90	0.7311
95	0.7309
100	0.7310
105	0.7310

Table 5: Test accuracy related to number of features using logistic regression when applying RFE

For logistic regression the best model was attained with 88 features with a test accuracy of 0.7336. This result was validated using 10-fold cross validation, where the mean of the evaluated 10 fold's accuracy was 0.7310. Hence, the model generalizes well to the data set. The top ten features with the greatest effect on the outcome are shown in Table 6.

Rank	Feature	β coefficient
1	hours_between_start_end	1.4255
2	bon_win_ge_bet_share	-1.0751
3	bon_small_win_share	-1.0295
4	game_minutes	0.5161
5	first_deposit_vertical_game_sportsbook	0.2882
6	cluster_7_rtp_vol_jack_hitfreq_rank_7	0.2851
7	most_frequent_category_sportsbook	-0.2685
8	game_sessions	0.2667
9	category_sb_share	-0.2197
10	operator_1	-0.2064

Table 6: Top ten features for the best model with logistic regression

6.1.2 Probability Threshold

As explained in chapter 1, the long term purpose of this thesis is to retain customers at the website by predicting which customers are likely to churn. Consequently, it is important for the model to classify the positive observations correctly, thus minimizing the amount of false positives is more important than minimizing the false negatives. A false negative classification will in theory result in Mr Green targeting that customer with promotions, while in fact, that might not have been necessary, as that customer probably would have returned to the site either way, resulting in unnecessary marketing costs. On the other hand, a false positive classification could in theory lead to Mr Green missing out on targeting that customer, who is then likely to churn. Due to this reasoning, an analysis of the effect of changing the probability threshold is essential. We define the threshold as the value p for which we let an observation be classified as 1 or 0 if $P(Y = 1) > p$, respectively $P(Y = 0) \geq 1-p$.

As is seen the Tables 7 and 8, setting a large value on p for $P(Y = 1)$ results in a very small percentage of misclassifications on the label 1 with the trade off being that the percentage of misclassification on the label 0 is large. The choice of the value p depends on the trade off between the marketing cost for false negatives misclassifications versus the importance of retaining customers.

$P(Y = 1) > p$	Share FP	Conf. int.	Positive classifications	FP
0.9	0.0297	0.0045	5377	160
0.8	0.0593	0.0056	6788	403
0.7	0.0926	0.0063	8138	754
0.6	0.1368	0.0067	9854	1349
0.5	0.2030	0.0070	12640	2567
0.4	0.2769	0.0068	16221	4492
0.3	0.3245	0.0067	18733	6080
0.2	0.3346	0.0067	19204	6427
0.1	0.3369	0.0066	19284	6497

Table 7: Share of false positives for each probability threshold for the logistic regression classifier

$P(Y = 0) \geq 1 - p$	Share FN	Conf. int.	Negative classifications	FN
0.1	0.5443	0.0083	13910	7571
0.2	0.5123	0.0088	12499	6403
0.3	0.4847	0.0093	11149	5404
0.4	0.4540	0.0100	9433	4283
0.5	0.4085	0.0118	6647	2715
0.6	0.3454	0.0168	3066	1059
0.7	0.2437	0.0358	554	135
0.8	0.1325	0.0729	83	11
0.9	0.3333	0.5334	3	1

Table 8: Share of false negatives for each probability threshold for the logistic regression classifier

6.1.3 Hyperparameters

To optimize the logistic regression model even further a hyperparameter analysis was performed. With the logistic regression classifier from scikit-learn the model

can be modified by changing the solver and the penalize term of the algorithm. For small data sets the default solver `liblinear` is usually a good choice, whereas for large data sets the `sag` and `saga` are preferred. However, the fast convergence of `sag` and `saga` is only guaranteed when the features in the data set are scaled. The results of this analysis is presented in Table 9. As one might tell from the results Table 9, when scaling the data set changing the hyperparameter does not affect the accuracy of the model. However, when using unscaled data the accuracy differs. The penalty L1 together with the solver `liblinear` and the L2 penalty with the `newton cg` solver result in the highest accuracy, where L1 and `liblinear` are chosen due to their substantially shorter running time.

Penalty	Solver	Scaled dataset	Unscaled dataset
L1	`liblinear`	0.7270	0.7270
L1	`saga`	0.7266	0.6824
L2	`lbfgs`	0.7265	0.7135
L2	`newton-cg`	0.7266	0.7271
L2	`sage`	no convergence	0.6825

Table 9: Hyper parameter analysis for the logistic regression classifier

6.2 Random Forest

6.2.1 Feature selection

As previously explained, the method used for feature selection for the random forest model was a gini threshold model, only including features whose gini values are above some threshold. The different thresholds were based on the mean gini value of all features, which was computed as `rand_forest_full_variables_gini_mean` 0.00813. The test accuracy was then evaluated for each model after applying the threshold, and the model with the highest test accuracy was chosen as the best model. Table 10 below shows the gini threshold, along with the number of selected features as well as the test accuracy.

Gini threshold	Number of features	Test accuracy
$0.0 \cdot mean$	123	0.7532
$0.05 \cdot mean$	104	0.7533
$0.1 \cdot mean$	96	0.7531
$0.2 \cdot mean$	**83**	**0.7540**
$0.3 \cdot mean$	71	0.7535
$0.4 \cdot mean$	55	0.7526
$0.5 \cdot mean$	50	0.7529
$0.75 \cdot mean$	37	0.7494
$1.0 \cdot mean$	32	0.7481
$1.25 \cdot mean$	31	0.7413
$1.5 \cdot mean$	27	0.7395
$2.0 \cdot mean$	22	0.7385
$2.5 \cdot mean$	17	0.7357
$3.0 \cdot mean$	14	0.7375

Table 10: Number of features and corresponding test accuracy for each gini threshold level

As is seen in the table, the test accuracy is fairly similar for all thresholds. The highest test accuracy, as highlighted in bold, of 0.7540 is obtained when the model consists of 83 features, which is the case when the gini threshold is set to $0.2 \cdot mean$. To validate these results, a k-fold cross validation, with $k = 10$ was performed using the best model, resulting in a mean cross-fold accuracy of 0.7544, which is very similar to the test accuracy. A table of the top ten features, based on their gini value is shown in Table 11 .

Rank	Feature	Gini value
1	hours_between_start_end	0.0844
2	rm_deposit_euro	0.0431
3	rm_result_euro	0.0428
4	age	0.0393
5	rm_bet_euro	0.03670
6	ol_sessions	0.0363
7	end_ol_sessions_with_0_euro_share	0.0340
8	game_minutes	0.0339
9	max_session_length_minutes	0.0290
10	rm_win_euro	0.0280

Table 11: Top ten most important features and their respective gini value

6.2.2 Probability threshold

The two tables below show the share of misclassifications, positives in the first table and negative in the second table, where p indicates the probability threshold presented in the first column.

$P(Y = 1) > p$	Share FP	Conf. int.	Positive classifications	FP
0.9	0.0148	0.0033	5148	76
0.8	0.0391	0.0046	6808	266
0.7	0.0814	0.0057	8699	708
0.6	0.1391	0.0065	10996	1529
0.5	0.2016	0.0068	13513	2724
0.4	0.2583	0.0068	15953	4120
0.3	0.2982	0.0067	17749	5293
0.2	0.3216	0.0067	18760	6034
0.1	0.3310	0.0067	19120	6329

Table 12: Share of false positives for each probability threshold with the random forest classifier

Table 12 above shows the positive misclassification rate for each probability threshold, and a corresponding confidence interval. As expected, the number of false positives increase as we lower the threshold, or more simply, become less strict in terms of classifying an observation as a 1, or positive. The default threshold of 0.5 results in a misclassification rate of approximately 20%.

$P(Y = 0) \geq 1 - p$	Share FN	Conf. int.	Negative classifications	FN
0.1	0.5473	0.0082	14139	7738
0.2	0.5023	0.0088	12479	6268
0.3	0.4551	0.0095	10588	4819
0.4	0.4032	0.0106	8291	3343
0.5	0.3500	0.0123	5774	2021
0.6	0.2930	0.0154	3334	977
0.7	0.2302	0.0210	1538	354
0.8	0.1594	0.0313	527	84
0.9	0.1138	0.0482	167	19

Table 13: Share of false negatives for each probability threshold with the random forest classifier

Table 13 above shows the negative misclassification rate for each threshold. Comparing the equivalent threshold between the two tables, the negative misclassification rate is consistently slightly higher than the positive one, indicating that the model is better at classifying positive observations.

6.2.3 Hyperparameters

Hyperparameters were tuned one at a time, evaluating the resulting test accuracy, to find the optimal value of the parameters. The Python implementation of some of the hyperparamters for which tuning was tested are shown in the table below.

Tuning paramteter	Implementation	Python default	Optimal value
`n_estimators`	Number of trees in the forest	10	300
`min_samples_leaf`	Minimum number of samples in a node	2	2
`max_leaf_nodes`	Maximum number of leaf nodes	None	None
`min_impurity_decrease`	Minimum impurity decrease required to split a node	0	0

Table 14: Hyperparameter tuning for random forest

As is seen in the table, the only hyperparameter for which the optimal value was different than the default one was the number of trees in the forest. The other parameters did not improve the test accuracy significantly when changed, and so the default value was used. Note that the parameter deciding the number of features to be considered for each split is by default $m = \sqrt{p}$, which as explained in Section 4.5.2, is the recommended size, and thus it was not tuned. Figure 4 below illustrates the test accuracy associated with the different number of trees in the forest.

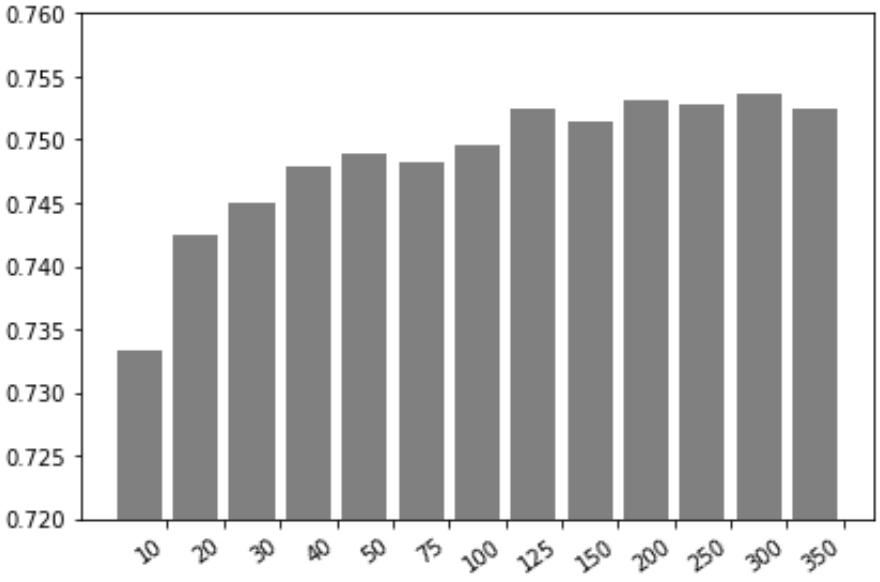

Figure 4: Test accuracy for different number of trees

6.3 Linear Discriminant Analysis

6.3.1 Feature Selection

As in the logistic regression model, recursive feature elimination was used in the feature selection. The model was then constructed repeatedly and the worst performing feature based on the β coefficient was removed each iteration. This process was repeated until all the features had been ranked according to their contribution. Since the model needs to be re-trained each iteration this becomes rather time consuming. A table of the number of chosen features for the RFE, with the corresponding test accuracy is shown below.

Number of features	Test accuracy
65	0.7168
70	0.7215
75	0.7208
80	0.7257
85	0.7244
90	0.7249
95	0.7268
100	0.7266
105	**0.7269**

Table 15: Test accuracy related to number of features using LDA

As is seen in the table, the test accuracy varies across the different number of features included in the model. The best test accuracy, as highlighted in bold, of 0.7269, was obtained with all 105 features. Using 10-fold cross validation one can conclude that the model generalizes well since the mean of the 10-fold evaluations is 0.7240. The top ten features are presented below in table 16.

Rank	Feature	β coefficient
1	rm_bet_count_18_23	13.3096
2	rm_bet_count_12_17	13.0515
3	rm_bet_count_06_11	10.4382
4	rm_bet_count_00_05	7.8879
5	bon_win_ge_bet_share	-2.3873
6	bon_small_win_share	-2.3873
7	rm_bet_count_share_of_last_day	-1.2633
8	cluster_7_rtp_vol_jack_hitfreq_rank_99	1.2633
9	most_freq_category_no_category	1.2633
10	hours_between_start_end	0.9879

Table 16: Top ten most important features and their respective β value with LDA

6.3.2 Probability Thresholds

After having identified the best LDA model, it was assessed based on the distribution of misclassifications, in terms of false positives and false negatives, for different probability thresholds. The probability thresholds and resulting misclassifications are illustrated in Table 17 and 18.

$P(Y = 1) > p$	Share FP	Conf. int.	Positive classifications	FP
0.9	0.0169	0.0030	4131	70
0.8	0.0425	0.0050	6208	264
0.7	0.0843	0.0061	7924	668
0.6	0.1435	0.0068	10119	1453
0.5	0.2108	0.0070	12967	2734
0.4	0.2865	0.0068	16600	4757
0.3	0.3267	0.0067	18772	6133
0.2	0.3358	0.0067	19207	6450
0.1	0.3379	0.0067	19286	6518

Table 17: Share of false positives for each probability threshold with LDA classifier

Table 17 shows the share of positive misclassifications, out of all observations that were classified as positive. The default threshold of 0.5 generates a positive misclassification rate of approximately 21%.

$P(Y = 0) \geq 1 - p$	Share FN	Conf. int.	Negative classifications	FN
0.1	0.5744	0.0079	15156	8707
0.2	0.5217	0.0086	13079	6824
0.3	0.4850	0.0092	11363	5512
0.4	0.4474	0.0101	9168	4102
0.5	0.4011	0.0120	6320	2535
0.6	0.3442	0.0180	2687	925
0.7	0.2504	0.0374	515	129
0.8	0.1375	0.0755	80	11
0.9	0.0000	0.0000	1	0

Table 18: Share of false negatives for each probability threshold with LDA classifier

Table 18 shows the share of negative misclassifications for each probability threshold. It is important to note, that as we become more strict in terms of classifying an observation as negative, less observations will pass that threshold. The uncertainty as we move to more extreme thresholds is also reflected in the increasing confidence interval. In the last row of the table, when an observation is only classified as negative if its probability of being negative is greater than 0.9, there was only one observation classified as negative, which was correctly classified, resulting in a misclassification rate of 0 %.

6.3.3 Hyperparameters

As LDA has no hyperparameters, the best model was simply based on the feature selection.

6.4 Meta model: Stacking Classifier

The inputs to the stacking classifier are the outputs of the three base models, that is the computed class probabilities from the logistic regression, the random forest and the linear discriminant analysis, for a total of six features. Due to this nature, no further feature selection was performed. As for the base models, each was optimized individually, in terms of the number of features and any potential hyperparameters. The obtained test accuracy was 0.7523, which is slightly higher than that of the logistic regression and the linear discriminant analysis but slightly lower than that of the random forest.

6.4.1 Probability Thresholds

An assessment of the misclassification distribution was performed for the stacking model as well. The two tables below show the share of positive and negative misclassifications for each probability threshold.

$P(Y = 1) > p$	Share FP	Conf. int.	Positive classifications	FP
0.9	0.3365	0.0089	10765	3622
0.8	0.3368	0.0086	11728	3950
0.7	0.3370	0.0083	12400	4179
0.6	0.3363	0.0081	12955	4357
0.5	0.3382	0.0080	13453	4550
0.4	0.3388	0.0068	13965	4732
0.3	0.3396	0.0077	14504	4925
0.2	0.3399	0.0075	15174	5157
0.1	0.3383	0.0073	16062	5434

Table 19: Share of false positives for each probability threshold for the stacking classifier

Table 19 above shows the share of positive misclassifications. Unlike previous models, this share stays relatively constant across the different thresholds. However the

number of positive classifications increase as we lower the threshold, i.e. become less strict to classify an observation as positive, as expected.

$P(Y = 0) \geq 1 - p$	Share FN	Conf. int.	Negative classifications	FN
0.1	0.6639	0.0100	8522	5685
0.2	0.6645	0.0106	7559	5023
0.3	0.6650	0.0111	6887	4580
0.4	0.6638	0.0116	6332	4203
0.5	0.6682	0.0121	5834	3898
0.6	0.6704	0.0127	5322	3568
0.7	0.6736	0.0133	4783	3222
0.8	0.6769	0.0143	4113	2784
0.9	0.6738	0.0162	3225	2173

Table 20: Share of false negatives for each probability threshold for the stacking classifier

Table 20 above shows the share of negative misclassifications for the different thresholds. Just as the share of positive misclassifications, the share of negative misclassifications do not change by much across the probability thresholds. They are however significantly higher than the positive ones, and one can conclude that the stacking model performs worse in terms of classifying negative observations, compared to the positives.

6.5 Meta model: Voting Classifier

When creating the voting classifier the three base models were first trained separately with the complete data set without any initial feature selection. The reason for this was that the voting classifier function in scikit-learn uses the same data set for all base models. The probability of each outcome, for each observation, for all three models was then stored. The voting classifier then calculated the average weighted probability and determined the most likely class out of $\{0, 1\}$. To identify the voting classifier with the highest accuracy the process was repeated for different weight distributions. A selection of the iterations and results is presented in Table

21, where w_1 is the weight for logistic regression, w_2 for random forest and w_3 for LDA. The best result is obtained when the weight for the random forest classifier, w_2, is large and the ratio of the remaining weight is greater for the LDA classifier than the logistic regression.

w_1	w_2	w_3	Accuracy
0.0500	0.95	0.0000	0.7552
0.0125	0.95	0.0375	0.7578
0.0000	0.95	0.0500	0.7583
0.0500	0.90	0.0500	0.7561
0.0250	0.90	0.0750	0.7576
0.0000	0.90	0.1000	0.7590
0.1500	0.85	0.0000	0.7540
0.1125	0.85	0.0375	0.7556
0.0750	0.85	0.0750	0.7567
0.0375	0.85	0.1125	0.7579
0.0000	0.85	0.1500	0.7590
0.2000	0.80	0.0000	0.7505
0.1500	0.80	0.0500	0.7540
0.0100	0.80	0.1000	0.7562
0.0500	**0.80**	**0.1500**	**0.7595**
0.0000	0.80	0.2000	0.7594
0.0625	0.75	0.1875	0.7590
0.0000	0.75	0.2500	0.7589
0.0000	0.70	0.3000	0.7588
0.0000	0.65	0.3500	0.7593

Table 21: Test accuracy for different weights between the three base models where w_1 is the weight for logistic regression, w_2 for random forest and w_3 for LDA

6.5.1 Probability Thresholds

Much like previous models, a higher probability threshold results in a lower rate of false positive misclassifications, which is preferable.

$P(Y = 1) > p$	Share FP	Conf. int.	Positive classifications	FP
0.9	0.0107	0.0034	3456	37
0.8	0.0374	0.0051	5484	205
0.7	0.0846	0.0064	7209	610
0.6	0.1430	0.0071	9213	1317
0.5	0.2049	0.0074	11449	2346
0.4	0.2720	0.0074	13767	3744
0.3	0.3120	0.0073	15229	4751
0.2	0.3321	0.0073	15876	5273
0.1	0.3385	0.0073	16064	5438

Table 22: Share of false positives for each probability thresholds for the voting classifier

$P(Y = 0) \geq 1 - p$	Share FN	Conf. int.	Negative classifications	FN
0.1	0.5705	0.0086	12635	7208
0.2	0.5041	0.0095	10607	5348
0.3	0.4535	0.0103	8882	4028
0.4	0.3971	0.0115	6878	2731
0.5	0.3283	0.0135	4642	1524
0.6	0.2590	0.0178	2324	604
0.7	0.1728	0.0252	862	149
0.8	0.1116	0.0420	215	24
0.9	0.0370	0.0712	27	1

Table 23: Share of false negatives for each probability thresholds for the voting classifier

7 Discussion

7.1 Findings

After performing the analysis presented in this report, the research questions can be answered.

- To what extent is it possible to predict the probability of each customer churning, using data from the first 24 hours after the player's first deposit?

The highest test accuracy of 75% is obtained using the voting classifier, which is one of the meta models. A natural consequence of the voting is that the meta model will perform at least as good as the best base model, since it will weigh better models greater. The difference however in test accuracy between the random forest, which was the best performing base model and the voting model is rather small. One could however still argue that it is preferable to use the voting classifier as a final model, since it incorporates the behavior of multiple models. It is also possible that as the accuracy of the base models improve, the accuracy of the voting model will improve by even more.

- Which supervised machine learning model explains the players' churn behaviour the best?

Out of the base models it is random forest that performs the best in terms of test accuracy, albeit only marginally compared to the logistic regression and the LDA. A potential reason for this could be that the non-parametric approach captures the characteristics of the data set the best, as opposed to the other two models, both assuming a type of linear relation. Another reason to why random forest performs the best could be the fact that it incorporates a type of ensemble technique when building multiple trees.

In addition to the answers to the research questions, an interesting finding is that the base models all choose slightly different features to be included in the models, especially for the top ten highest rated features. A potential reason for this could be that the data does not sufficiently explain the actual churn behaviour. Had the data set been able to capture the churn behaviour of the customers to a larger extent, one would expect the features selected for the three models to have a greater overlap.

However, it is also natural to expect that the models would choose different features, as they assume different relations in the data, that is between the predictors and the response variable.

Another interesting finding is that the test accuracy is approximately the same for all models, both the base models and the meta models. Since the models all capture different aspects of the data set, as previously mentioned, and they have all been optimized individually, one can assume that the test accuracy is related to the quality of the data set. The data set mainly consists of demographical or monetary features, for instance bet size or winnings. There are a few variables related to the player activity, such as online sessions, the channel used or the top category of games. After discussions with colleagues at Mr Green, we have found that the latter category usually play an important role in terms of customer behavior. Consequently, it is reasonable to assume that the model accuracy could have been improved, had that category of the variables been more extensive. An example would be more data on the website interface and how the player browses around on the site.

Moreover, human behavior incorporates a high amount of randomness. For instance, two players might have the same demographics and exhibit a similar gaming pattern, however one might choose to leave whereas the other one chooses to stay. No matter the data set, this factor will always be difficult to model. Nonetheless, a more extensive data set is more likely to result in an accurate model.

7.2 The Contribution of the Findings

7.2.1 The feasibility of the model

Since the overall goal with this thesis is to create a model that is easily implemented, the running time is an essential part of the evaluation. Collecting and creating the data set for 24 hours takes about 5 hours. Although this might appear long, it is important to keep in mind that this part of the execution is only performed when re-training the model. The model itself should then be run on a daily basis, to be able to instantly target customers predicted to churn. The running time for making predictions is approximately 10 minutes, making it manageable on a daily basis.

7.2.2 Marketing implementations

One of the objectives with the model is to optimize the marketing strategy at Mr Green. Before implementing the model it is impossible to predict which customers will churn. Consequently, Mr Green needs to target all their customers. By implementing the model, Mr Green would be able to predict which customers are likely to churn, and focus their marketing accordingly. To analyze this concept, we assume that Mr Green would only target customers who are predicted to churn. The false positives will thus not be targeted, and could in theory represent customers who could have been converted, had Mr Green had a chance to target them. Analogously, the number of all positive classifications could then be seen as the amount of customers that Mr Green will not have to target. Looking at marketing costs, one could then view the positive classifications times the average marketing cost per customer as costs saved compared to today, when marketing is directed to all customers. To further analyze savings, one should subtract the equivalent cost for the false negative classifications, assuming that these customers would have stayed even without the marketing effort. Lastly, one should assess the cost of losing the customers, which the model predicted to be false positives.

Of course, the false predictions will be impossible to know beforehand. Even after the 30 days, when the actual outcome will be known it will be impossible to tell if the customers churning or staying was impacted by the marketing effort, which was based on the initial predictions. The tables showing the number of positive and negative misclassifications for various probability thresholds then become useful. As the results are made generic with the use of confidence intervals, one could assess the average error rate of an observation, given its estimated probability. Studying the tables, it is clear that an observation with a high class probability will have a lower error rate on average. This could be a fruitful insight when looking at which customers to target.

7.3 Future Improvements

One hypothesis is that it is possible to increase the accuracy by collecting data for a longer time interval than 24 hours. The data on the customer behaviour then contains more information, making it easier to predict the true outcome $\{0, 1\}$.

Therefore, a proposed model improvement is to run the model with data for 1-8 days (24 h, 48 h, 72 h, 96 h, 120 h, 144 h, 168 h, 192 h) after the customer's first deposit. Since many people have strong weekly habits the authors of this thesis believe that 8 days is an important time interval to examine closer. For example, if a player starts playing on Thursday night it is likely that she or he will return next Thursday night. Accordingly, using 192 hours of data to train the model might capture a behaviour that shorter time intervals fail to capture.

Another future improvement would be to try to collect more data on each customer and their behaviour on the website to better capture the likelihood of a customer churning. This is motivated with the words from one of the supervisors Jiri Pallas, "The model only is as good as the data". For example, an interesting feature to analyze is if the customer at any time chooses to click on the help button, which could indicate that the customer is relatively new to online gaming. Another potential feature is information about the customers' interaction with customer service.

7.4 Conclusion

We conclude that it is indeed possible to predict the customer churn rate, at an accuracy of approximately 75%. The model that performed the best was the meta model using voting to combine the three base models logistic regression, random forest and LDA. We also concluded that in order to make an even more accurate classification, a more detailed data set would have been necessary.

8 References

A Kaggler's Guide to Model Stacking in Practice (2018). URL: http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice (visited on 05/10/2018).

Blom, G. et al. (2005). *Sannolikhetsteori och statistikteori med tillämpningar*. 5th ed. Studentlitteratur, pp. 166–171, 291–292.

Coussement, K. and De Bock, K. (2013). "Customer churn prediction in the online gambling industry: The beneficial effect of ensemble learning". In: *Journal of Business Research*, pp. 1629–1636.

James, G. et al. (2013). *An Introduction to Statistical Learning*. New York, USA: Springer Texts in Statistics, pp. 1, 6, 130–149, 175–184, 278–280, 303–330.

Jolley, B., Mizerski, R., and Olaru, D. (2006). "How habit and satisfaction affects player retention for online gambling". In: *Journal of Business Research*, pp. 770–777.

Kotsiantis, S. (2007). "Supervised Machine Learning: A Review of Classification Techniques". In: *Emerging Artificial Intelligence Applications in Computer Engineering*, pp. 3–22.

Marsland, S. (2012). *Machine Learning - An Algorithmic Perspective*. 2nd ed. New York, USA: Chapman & Hall, pp. 20, 22–26.

Mr Green & Co, 2017 Annual Report (2018). URL: http://www.mrg.se/en/financial-information/financial-reports (visited on 01/05/2018).

Om tillstånd (2017). URL: https://www.lotteriinspektionen.se/tillstand-och-spelformer/ (visited on 12/01/2017).

Omreglering av Spelmarknaden (2017). URL: https://www.lotteriinspektionen.se/fragor--svar/omreglering-av-spelmarknaden/ (visited on 12/01/2017).

Tillstånd och spelformer (2017). URL: https://www.lotteriinspektionen.se/tillstand-och-spelformer/ (visited on 12/01/2017).

Zenko, B. and Dzeroski, S. (2004). "Is Combining Classifiers with Stacking Better than Selecting the Best One?" In: *Machine Learning*, pp. 255–273.

TRITA -SCI-GRU 2018:172