Predicting customer churn in the financial industry

Fredrik Fagerholm 37164
Master's Degree in Computer Science
Supervisors: Mats Neovius, Marina Walden
The Faculty of Science and Engineering
Åbo Akademi University
28 March 2022

Abstract

The focus of this thesis is to investigate how customer churn can be modelled at a company in the financial industry, by exploring the computational means of predicting customer churn. The goal is to build a working model for predicting the churn of borrowers and at the same time explore the main drivers of churn. The study is carried out in the form of a single-case study, and the purpose of this thesis is to present the theory and methods used, as well as document the process and findings.

The problem is defined as a classification task, and a random forest model is evaluated through cross-validation. Since the problem is framed in this way, the definition of what constitutes a churn event has a major impact on the applicability of the results. Therefore, great attention is given to this issue. Appropriate metrics are used to assess the performance of the model. The choice of evaluation metrics for the classifiers' performance is especially important due to the class imbalance in the data. The metrics *AUC* and *Cumulative gain* are chosen since these are insensitive to class imbalance, compared to metrics such as *Accuracy score*.

The model appears to capture the churn phenomenon quite well for the prepared data and is able to accurately predict which customers will churn during cross-validation. The model has an AUC score of 0.74.

Keywords: Churn, machine learning, classification, random forest.

Contents

Chapter 1

Introduction

Customers belong to the companies most valuable assets. Therefore, companies need to understand their customer base. The company should know how many customers are leaving the company and for which reasons. One approach towards solving these problems is to analyse customer retention, i.e. modelling customer churn. This thesis is about predicting customer churn using machine learning methods. This approach has shown to be useful in industries such as telecommunications, banking and insurance [19]. Companies in these industries can use such methods because of the ease of detecting when a customer has left the company, e.g. ending the contract with an internet service provider gives a signal at the exact moment of churn. This allows the company to compare active and churned customers and use the indicator for churn as the output in predictive models. These methods have also become increasingly popular with companies providing subscription-based services [25, 11, 8, 4, 9, 14].

This thesis focuses on a particular kind of churn, namely churn of customers with loans. The main interest is in learning which customers will transfer their loans in the future. To be able to apply machine learning methods, the problem is framed as a supervised learning task, more specifically a binary prediction problem. Since the interest lies in predicting future churn, the input and output that comprise the training data for the model are collected from different points in time. The output is obtained from data that are in the *future* relative

to the input data. A fixed time window of six months into the future is chosen since this gives enough time to apply retention measures to the customers that are predicted to churn.

The input data consist of information about savings, payments and loans as well as demographic data. The output is not directly available in the data but derived from one of the features in the data. The input features and definition of the output variable are discussed in sections 4.1 and 4.2.

A random forest model [1] is used to predict loan churners, i.e. customers who consciously decide to transfer their loan to another bank. A fixed time window of six months into the future is set, meaning that the binary output variable indicating churn is six months in the future from the date that the input data were gathered. The output is not directly available in the data, but a definition must be created based on the other available features that capture the churn behaviour adequately.

A company needs to understand which customers are going to leave and what the drivers of this behaviour are, because it is much more expensive to acquire new customers compared to retaining existing customers [27]. It has been suggested that it costs as much as 12 times more to gain a new customer compared to retaining an existing one [26]. This may, however, not always be the case, since there may be customers for whom the cost of retention exceeds the gain of keeping the customer, see section 2.1.2.

Customer relationship management (CRM) is an approach to understanding customers and building customer relationships [16]. It is an extensively applied strategy in industries such as telecommunications, banking, insurance and retail. One major goal of CRM is customer retention, that is, to prevent customers from leaving and, instead, staying with the current service provider. Inevitably, some customers will leave, predicting this churn helps make targeted retention strategies more accurate. This limits losses and improves marketing decisions [8].

A common challenge in customer churn prediction is that the data are

imbalanced, meaning that there are few customers who leave [28] since churn events are quite rare. The significant imbalance between the number of observations of the negative and the positive classes in the data may cause problems for machine learning models and their evaluation. The implications of imbalanced data are discussed in section 3.3.

It may be beneficial to use interpretable models that give insight into the problem. For example, logistic regression, decision trees or generalised additive models. However, as logistic regression only captures linear relationships between the inputs and outputs [12] it is not possible to capture more complex nonlinear phenomena. Decision trees, on the other hand, are susceptible to noise in the data [28] and have a tendency to overfit. For generalised additive models, it can be challenging to tune hyperparameters, especially when there is a high number of features.

The first goal of the thesis is to create a process that quantifies the churn that is taking place. The second goal is to create a model that can predict which customers are going to leave in the future as well as give an idea of what the drivers of churn may be.

Chapter 2

Customer churn

2.1 Definition of customer churn

Customer churn is a term used in CRM. It refers to the phenomenon of customers leaving, that is, the customers completely interrupting their relationship with the company [2]. It is a key business metric in several industries, because the cost of customer acquisition is much greater than the cost of customer retention [16, 2, 27]. Terms such as customer attrition and turnover are also used for this phenomenon.

There is a distinction between voluntary and involuntary churn [10]. Voluntary churn is when a customer on volition decides to leave or switch to another company. Involuntary churn is when the customer has to leave due to reasons outside of their control, e.g. moving to another country or going bankrupt. It is desirable to only capture voluntary churn, which the company is able to affect.

Churn can also be narrowed down to specific products or services, identifying a customer as a churner if the customer discontinues the use of that specific product. For example, a bank may be interested in the churn of bank accounts [19]. In this thesis, the focus is specifically on the churn of customers with some substantial volume of loans. The behaviour of customers moving their loans to

another bank is approximated by measuring changes in loan volumes.

2.1.1 Identifying churners

In many cases, it is not possible to identify customers who have left the company directly from the data. One must define a criterion for identifying whether a customer is to be labelled as a churner or not. This is not a trivial task. The definition should capture the phenomenon one is interested in studying. There will inevitably be some customers who are falsely labelled as churners without that being the case. Our definition of churn is based on the assumption that a customer who has had a large volume of loans, and has drastically decreased the loan volume in a short period of time, has most likely refinanced their loan. Three parameters must be specified in this definition:

1. A threshold for what constitutes a large loan
2. A threshold for what constitutes a large drop in the loan volume
3. A time frame during which the loan volume has changed

Both the distribution of, and changes in, the loan volumes are studied to find good values for these parameters. How the time frame is chosen affects not only which customers are labelled as churners, but also how far into the future eventual predictions can be made. With the problem at hand, the time frame is chosen to be six months, as this provides a reasonably long time to act if a customer is predicted to be a churner. Churn has been defined in this manner, also partly because of what data are available.

One problem with this definition is that it may label customers who have paid off a large proportion of their loan within the time frame as churners. To counteract this the threshold for what constitutes a large loan should be set sufficiently high, so that it becomes unlikely for customers to pay off such a large volume within the time frame.

2.1.2 Important groups of customers

Some groups of customers are more desirable to retain. If the goal of the churn analysis is to increase profits, it should focus on the profitable customers [2], and give the most reliable predictions for this subgroup. For example, "Loyal" customers is such a group, and they are profitable both in the short and long run. One approach is to give weight to the observations according to their importance in the classification model, while training the model gives a higher penalty for predicting the class wrong for an important observation.

2.2 Churn rate

Churn rate refers to the number or proportion of individuals who leave the company during a given period [19]. That is, the number of individuals who left divided by the size of the whole customer base, during the period. Churn rate can be calculated for any time span, however, it is usually done on a monthly or yearly basis due to the binding periods of contracts. Even if the service is not contract-based, the churn rate can be unreliable for very short periods because of fluctuations in the number of customers leaving.

2.3 Benefits of predicting churn

The benefits of understanding why and which customers are going to churn are that it makes it possible to

1. Apply directed measures to retain customers identified to be potential churners
2. Stop potential churners directly by improving their experience (through customer service and marketing)

This is desirable for the company, since acquiring new customers is much more costly than keeping valuable customers from leaving. Suppose a churn rate of 7% a year in a company with 1 million customers where each customer

contributes 50 euro on average per year. A discount rate of 6% is assumed.

In the first year, there are 1 million customers, the next year 930 thousand customers and then 864.9 customers and so on. The contribution for the first year is 50 million euro and for the next years 46.5 million euro and 43.2 million euro. With discounting these become $\frac{46.5}{(1+0.06)^1} \approx 43.9$ million euro and $\frac{43.2}{(1+0.06)^2} \approx 38.5$ million euro.

The total contribution over 25 years is 392.2 million euro. Assume that through analysing the reasons for customer churn, the churn rate can be decreased by 1%. This will increase the total contribution to 419.7 million euro, an increase of 27.5 million euro over the period of 25 years [10].

Year	**Customers**	**CF (MEur)**	**Discount**	**DCF (MEur)**	**Cumulative (MEur)**
0	1 000 000	50	1	50	50
1	930 000	46.5	0.943	43.9	93.9
2	864 900	43.2	0.89	38.5	132.4
3	804 357	40.2	0.84	33.8	166.1
4	748 052	37.4	0.792	29.6	195.7
5	695 688	34.8	0.747	26	221.7
6	646 990	32.3	0.705	22.8	244.5
7	601 701	30.1	0.665	20	264.6
8	559 582	28	0.627	17.6	282.1
9	520 411	26	0.592	15.4	297.5
10	483 982	24.2	0.558	13.5	311
11	450 104	22.5	0.527	11.9	322.9
12	418 596	20.9	0.497	10.4	333.3
13	389 295	19.5	0.469	9.1	342.4
14	362 044	18.1	0.442	8	350.4
15	336 701	16.8	0.417	7	357.4
16	313 132	15.7	0.394	6.2	363.6
17	291 213	14.6	0.371	5.4	369
18	270 828	13.5	0.35	4.7	373.8
19	251 870	12.6	0.331	4.2	377.9
20	234 239	11.7	0.312	3.7	381.6
21	217 842	10.9	0.294	3.2	384.8
22	202 593	10.1	0.278	2.8	387.6
23	188 412	9.4	0.262	2.5	390
24	175 223	8.8	0.247	2.2	392.2

Table 2.1: Economic cost of churn.

Churn prediction may improve the retention activities by helping to

understand which customers should be targeted. Suppose that a company with 1 million customers has a churn rate of 5% each year, i.e. 50 000 customers leave. Assume that the company runs a retention campaign with 25% effectiveness, that is, a fourth of the customers (12 500) reached by the campaign are retained. If the campaign is poorly targeted it may only reach 20% (10 000) of the churners of which 25% (2 500) are retained, the effective retention rate is $\frac{2500}{50000} = 5\%$. A large part of the effort is wasted due to poor targeting.

Churn prediction can alleviate the targeting problem by finding a greater part of the churners. With a predictive model that can identify 50% (25 000) of the churners, the retention rate can be as high as $\frac{25000 * 0.25}{50000} = 12.5\%$, an improvement of 250%. An effective retention rate of 100% is not realistic, since some of the churn may be involuntary.

2.4 Features

In general, four types of variables appear in the customer churn literature: customer behaviour, customer perceptions, customer demographics and macro environment variables [2, 10]. Customer behaviour features usually measure something related to the transaction between the customer and the company, for example, transaction recency and frequency, active products and services, and monetary value. Demographic features may include age, gender, education level, salary, postal code etc. Customer perceptions include features that quantify the view that the customer has of the company, such as results from customer satisfaction surveys. Features related to the macro environment are external factors such as a ”prosperity index”, e.g. GNP per capita [10]; company changes, for example changes in strategy and moving to digital services.

Chapter 3

Theory of customer churn analysis

3.1 Models

With the background information presented in the previous chapter, a model that is suitable for the task of customer churn prediction can be chosen. Since the input data consist of numerical and categorical features, and output is a binary categorical variable, a supervised binary classifier is a reasonable choice. This chapter describes the background for such models and how their performance can be evaluated.

3.2 Supervised learning

Within the field of machine learning, supervised learning consists of tasks where an algorithm should learn an association between a set of inputs and outputs. The inputs for a task may vary in nature; it can be continuous, categorical or ordinal data. Most algorithms in their basic form expect the data to be transformed into numerical form to be able to process it. The type of output determines the kind of learning task. Tasks may broadly be divided into two groups; if the outputs consist of continuous numerical data, it is a *regression*

task, whereas, if the outputs are discrete categories, it is called a *classification* task. Unsupervised learning differs from supervised learning in that the target outputs are unknown and some patterns should be learned from the input data alone.

The central concept of supervised learning is for an algorithm to learn a relationship between input and output data, a model of the relationship between certain inputs and their associated outputs. The model is *trained* on a set of data, and it learns a function between the inputs and outputs, so when given new data it produces outputs that are close to those that are expected.

There are various algorithms for finding associations between input and output data. They rely on different ways to model the relationship between inputs and outputs, some allowing for non-linear relationships and some setting restrictions on the space of functions that may represent the relationship. In most cases, algorithms rely on a *loss function* that measures how well the model can predict the outputs in the training data. The parameters of the model are then adjusted to minimise the value of the loss function, also called an *objective function*, and this way obtain the trained model. Models can also have *hyper-parameters* that affect the loss function or how the optimisation takes place, and these can be tuned by the user.

Previously studied customer churn prediction models include Neural Networks, Support Vector Machines (SVM), Logistic Regression, Naive Bayes, Decision Trees [27] and Random Forest [2]. A boosted version of SVM with a polynomial kernel works best for the data used in [27], but which type of model works best for predicting customer churn is highly dependent on the data.

For this study, a random forest model was chosen after some preliminary testing. It gave a satisfactory performance while being easy to use. It also has some other desirable properties that are described further in the section 3.8.1.

3.3 Imbalanced data

The customers who leave usually comprise a small part of the data set, which can pose problems for the classifier since the group of churners is underrepresented. This is especially problematic since the churners are the class of interest. Many common classification algorithms do not work well for imbalanced data [6, 3].

The imbalance can be remedied through resampling; undersampling the majority class or oversampling the minority class [3, 13]; or by weighting the observations and putting a higher penalty on the misclassified observations when using ensemble-, or boosting techniques [3, 6, 5].

Potentially, it may be more costly to misclassify churners as non-churners, because this means no preventive measures are taken for these customers and they are therefore more likely to leave. Conversely, classifying non-churners as churners may not be as costly, since these customers only receive extra attention. Still, it should be taken into account that the budget for marketing and customer service probably is limited. The attempts are misdirected in this case, leading to unnecessary costs.

To avoid misclassifying churners as non-churners, "cost-sensitive learning" [17] can be applied. During training, a higher penalty is given for misclassification of observations of the positive class (churners) than for the negative class (non-churners). The default approach is to distribute the penalties evenly.

3.4 How to handle missing values

Missing values in the data is a common problem [10] that most models and learning algorithms cannot handle by themselves. The missing values have to be dealt with in some way before training the model. Leaving out observations or features that have missing values can be an option if it does not significantly reduce the size of the data set. However, this should be avoided, if possible, since

all the values in the discarded observation or feature are removed along with the missing value.

Another common solution is to impute the missing values by using information from the existing ones. One can either impute with an estimate from a model trained on the available data [10] or impute with the sample mean of the existing values. Alternatively, one can replace missing values with a predefined placeholder and add a new binary feature, taking the value one at the rows of the missing values. This method can pose problems if there are many features with missing values. Adding new binary features for each feature with missing values potentially doubles the total number of features without the indicator features not contributing much additional information.

When deciding how to deal with missing values it is important to take into account why the values are missing, if that information is known. The ways in which data can be missing is commonly divided into three types [24, 18].

The first type of missingness is *missing completely at random* (MCAR), where the reason for the data being missing is unrelated to the rest of the data. For example, with sensor data, if data are missing due to random sensor errors.

The second type is *missing at random* (MAR), meaning that the missing values of one variable can be explained by observed values of another variable. An example would be survey data where a certain demographic is less likely to answer certain questions.

The third type is *missing not at random* (MNAR), which means that the reason for values missing is neither MCAR nor MAR. An example would be a temperature sensor for which the number of missing values increases with the temperature.

The validity of a chosen method for dealing with missing values depends on which of the three types the missing values are; for example, if the data are MCAR, deleting observations with missing values produces unbiased estimates of means, variances and regression weights. Under the MAR assumption, imputing

the values using linear regression with the existing data as explanatory variables gives unbiased estimates of the regression weights of the imputation model if the explanatory variables are complete. However, it introduces bias for correlations and variance. The effect of different methods depending on the *missingness* type is a vast area and is out of the scope for this thesis, thus, it will not be explored further here.

3.5 Model validation

Model validation is the process of using the model to generate predictions on input data it was not trained on and then comparing the predictions to expected outputs. This procedure requires what is called a *validation* or *test* set, a data set that has not been used for training the model but which contains the same type of inputs and outputs as the training data. When training and evaluating machine learning models the available data are usually divided into these separate sets before starting to train the model, ensuring that the model does not come into contact with the validation data before the evaluation stage. The outputs in the validation set can be compared to the predicted outputs using various evaluation *metrics*. Some metrics for classification tasks will be described in the following sections.

3.5.1 Classification metrics

In binary classification tasks where the data are imbalanced, i.e. there are significantly more observations belonging to one class than to the other, it is important to use appropriate metrics for evaluating models to obtain a realistic estimate of the models' true performance [3]. The accuracy score does not give a good picture of a classifyer's performance when the data are highly imbalanced, especially when correct predictions of the minority class are of higher interest than those of the majority class. For example, in a data set where 99% of the observations belong to one of the classes, a model with a predictive accuracy of 99% may seem good. However, this level of accuracy can be achieved by a classifier

that simply predicts the majority class, making accuracy a poor performance metric when dealing with unbalanced data.

3.5.2 Binary classifiers

In a binary classification task, the test data consist of negative and positive examples. There are N negative examples and P positive examples. The model tries to predict the true classes (without having access to them), but some of the predictions may be wrong. There are four types of classifications:

1. True positives (TP): the true class is positive and the predicted class is positive.
2. False positives (FP): the true class is negative but the predicted class is positive, also called Type I error.
3. True negatives (TN): the true class is negative and the predicted class is negative.
4. False negative (FN): the true class is positive but the predicted class is negative, also called Type II error.

The number of each type of classification can be visualised using a *confusion matrix* [5]. A confusion matrix shows TP in the top left corner, FN in the top right corner, FP in the bottom left corner and TN in the bottom right corner. It holds that $TP + FN = P$ (the total number of positives), and $TN + FP = N$ (the total number of negatives). The classifier assigns $TP + FP$ to the positive class and $TN + FN$ to the negative class. Figure 3.1 shows a typical layout of a confusion matrix.

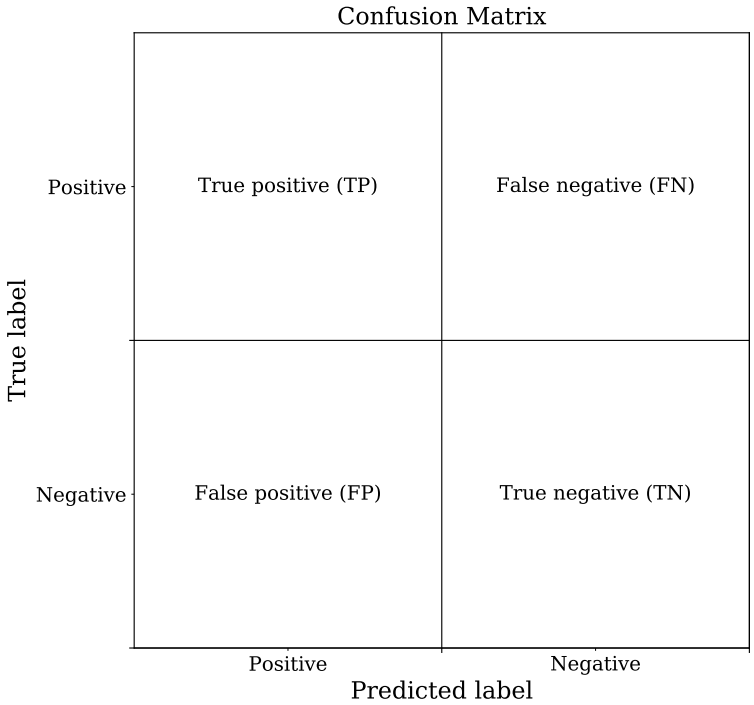


Figure 3.1: Layout of a confusion matrix.

Sensitivity, or true positive rate, measures the probability of predicting the positive class given that the observation belongs to the positive class. $Specificity$, or true negative rate, measures the probability of predicting the negative class given that the observation belongs to the negative class [12]. These metrics are defined by

$$Sensitivity = \frac{TP}{P} = \frac{TP}{TP + FN}$$

and

$$Specificity = \frac{TN}{N} = \frac{TN}{TN + FP}$$

Precision can be seen as a measure of relevance; it measures the proportions of times the true outcome is positive given that the predicted class is positive. *Accuracy* is the proportion of predictions that were correct, either positive or negative. $FP\ rate$ (false positive rate) is the proportion of times the predicted class is positive, but the true class is negative.

They are defined by

$$Precision = \frac{TP}{TP + FP}$$

$$Accuracy = \frac{TP + TN}{P + N}$$

and

$$FP\ rate = 1 - Specificity = \frac{FP}{TN + FP}$$

.

Regarding churn prediction, the churned customers would be defined as belonging to the positive class, and non-churners to the negative class. The metrics would be interpreted in the following way: if a customer has churned, and the classification model has labelled them as churned, it would be a True Positive; if the model labels the same customer as not churned, it would be a False Negative. Analogously, if the customer is not a churner and the prediction is "non-churn" it would be a True Negative, and for the prediction "churn" it would be a False Positive.

The derived metrics tell us something about the quality of the predictions and, therefore, also about the quality of the model. *Accuracy* tells us the proportion of the predictions that are correct compared to the total number of cases. That is, if the churn status of every customer in the customer base is predicted, then *Accuracy* measures how many of those predictions are correct as a percentage. *Sensitivity* measures the proportion of churners correctly identified as churners. The $FP\ rate$ measures the proportion of non-churners that are wrongly classified as churners. Ideally, for a classifier, the *Sensitivity* should be high and the $FP\ rate$ low.

Probabilistic classifiers and calibration

Probabilistic classifiers are models that, given an input, predicts a probability distribution over the set of classes. Compared to a binary classifier, whose outputs are either 0 or 1, a probabilistic classifier predicts a probability for the

observations belonging to each of the two classes. The same applies to a classifier predicting any number of classes.

The predicted probability distribution can be denoted using conditional probabilities. For a classifier with K classes, the predicted probability for class k would be written as $P(Y = k \mid X)$ where $k = 1, \ldots, K$, which reads as "The probability of class k given the input data X".

Many machine learning models are probabilistic classifiers in the sense that they can generate probability distributions over the classes. Such models include Naive Bayes, logistic regression, multilayer perceptrons, decision trees, and certain boosting and ensemble models. However, some of these models generate outputs that cannot be directly interpreted as class probabilities. From the models listed above, this applies to all models except logistic regression and multilayer perceptrons. This is a direct consequence of how the models work, what kind of objective function they try to optimise and how the optimisation is performed. That is, some probabilistic classifiers do not give a probability that corresponds to the true chance of an observation being a member of the class. Such classifiers are called uncalibrated [29]. That a classifier is calibrated means that if the classifier predicts a probability of 0.9 for a certain set of observations, approximately 90% of these observations belong to the positive class. For a calibrated classifier, a predicted probability can be directly interpreted as a confidence level. Some models produce well-calibrated probabilities by default. For example, logistic regression gives calibrated probabilities if the parametric assumptions are met [15], due to the model directly optimising the log-loss, while maximum margin methods such as random forest tend to push the mass of predicted probabilities away from 0 and 1 [20], making these models uncalibrated. Uncalibrated classifiers can be calibrated by the means of Platt Scaling or Isotonic Regression [20].

Any probabilistic classifier can be turned into a binary classifier by setting a threshold and classifying observations with a probability higher than the threshold to the positive class and the remaining to the negative class. For each threshold, the classes assigned to the observations might differ, since observations belonging to the same class may receive different predicted probabilities.

Therefore, given a probabilistic classifier, it can be turned into different binary classifiers by adjusting the threshold. Usually, the threshold is set to 0.5 by default, because it is assumed to be equally likely that an observation comes from either class. If the class distribution is believed to be something other than equal based on prior information, the threshold can be set accordingly.

When the threshold is adjusted, there is a trade-off between different types of misclassifications. For example, if the threshold is set to 1, all of the observations will be assigned to the negative class, since they need a predicted score higher than 1 to be assigned to the positive class. This results in the $Specificity$ being very high, while the $Sensitivity$ is very low. The metrics that rely on a specific threshold can be defined as functions of said threshold, for example, $TP(p)$ or $FN(p)$, where p is the threshold. This idea can be extended to the derived metrics $Sensitivity$, $Specificity$ etc.

All the metrics described above can be used for evaluating the performance of probabilistic classifiers, for any fixed threshold. If the classifier is to be evaluated independently of any threshold, other metrics should be used. Such metrics are described in the following sections.

3.5.3 Receiver operator characteristic curve

The receiver operator characteristic (ROC) curve is a common way to visually inspect the performance of a probabilistic binary classifier[5]. The false positive rate ($FP\ rate$) and true positive rate ($TP\ rate$) of the classifications are calculated at different probability thresholds [12, 3]. The different rates are then plotted against each other with the $FN\ rate$ on the x-axis and the $TP\ rate$ on the y-axis. Each point on the ROC curve shows the $FN\ rate$ and $TP\ rate$ for a specific threshold. The ROC curve is independent of the class balance and may, therefore, be used even when the data set is highly imbalanced.

The area under the ROC curve (AUC) can be used as a measure of performance. The value ranges from 0 to 1, where 1 is a perfect classification. An AUC of 0.5 means that the model performs as badly as a random classifier. To

calculate AUC, take each negative observation and count the number of positive examples with a higher assigned score than the score of the current negative observation. Thereafter, sum them up and divide everything by $P \cdot N$.

It is the same procedure as the one used for estimating the probability that a random positive example has a higher assigned score than a random negative example. An advantage of using AUC as a metric for performance is that the metric is independent from the classification threshold [5]

The ROC curve can be used to find the optimal threshold, in the sense of balancing the $FN\ rate$ and $TP\ rate$, i.e., the threshold that gives the highest $TP\ rate$ while keeping the $FN\ rate$ low. Which FN and $TP\ rates$ are considered acceptable is often problem-specific. Usually, the threshold is chosen with respect to the cost of misclassification.

In practice, *false negative* errors are usually more expensive than *false positive* errors. For example, in a system that checks the quality of machine parts, it is more expensive to miss a faulty part (*false negative*) than it is to mistake a correct part for a faulty one (*false positive*). In this example, it is assumed that faulty parts should be assigned to the positive class. If the cost of misclassification is unknown, one solution is to set the costs using the estimated class priors [22].

One way to find a threshold is to optimise the balance between the true positive and false negative rates. If the Youden's J statistic defined by $J = Sensitivity(p) + Specificity(p) - 1$ is maximised, one obtains the threshold that gives the highest sensitivity, while keeping the specificity high

$$p_{optimal} = \underset{p}{\operatorname{argmax}}\, Sensitivity(p) + Specificity(p)$$

.

3.5.4 Cumulative gains curve

The cumulative gains curve, also called the lift curve, gives a graphical representation of what percentage of customers one has to target in order to reach

a certain percentage of all churners. It is obtained by plotting the *Support* on the x-axis against the *Sensitivity* on the y-axis, for different probability thresholds. The *Support* is defined as $(TP + FP)/(N + P)$, that is, the observations that were predicted to be positive, divided by the total number of observations.

For a given churn probability threshold, the cumulative gains curve plots the fraction of all customers above the threshold against the fraction of all churners above the threshold [28].

3.6 Cross-validation

For a model to be useful in practice it has to perform well on data it has not seen before; thus, it is of interest to estimate how well it would perform on new data. This is quantified by the out-of-sample performance. The out-of-sample performance is the performance, as measured by any of the metrics described above, on new data that were not used for training the model. It measures how well the model can generalise outside of its training data and is hence indicative of how well the model is going to perform in a real setting.

However, it is not possible to obtain the true out-of-sample performance, but it can be estimated by using the available data. The performance on the training data (*training performance*) is usually a poor estimate of the out-of-sample performance [12]. The training performance usually increases with the complexity of the model, since a more complex model will more easily overfit to the training data and decrease the training error to zero. Therefore, other methods have to be used to obtain a reliable estimate of the true out-of-sample performance.

The performance estimate can be used both for selecting the best of several different models, as well as estimating how well the final model generalises. For traditional statistical models, e.g. logistic regression, it is common to measure the relative performance between models analytically, using Mallows's C_p statistic, *Akaike information criterion* or *Bayesian information criterion*. The final model can be chosen by comparing these metrics. This is less common for

machine learning models, since these methods require the maximum likelihood estimates to be computed, which is too complicated for many machine learning models.

Cross-validation can be used to estimate the models performance on unseen data; it is achieved by dividing the available data into separate data sets for the training and testing of the model. One common way to divide the data is to use 75% of the data for training and 25% for testing. The *training data* are used for training the model, after which predictions are made using the inputs of the test data. The performance is calculated based on the predicted values and the test outputs. There are no general rules about how to choose the sizes of the training and test sets. Usually, the training set is chosen to be larger, since this allows the model to better learn the relationships in the data. Yet, there is a trade-off, since a smaller test set results in less reliable estimates of the out-of-sample performance.

For the cross-validation estimates to be reliable, it is important that the training and test sets are *independent*. This condition can be violated in many, sometimes subtle, ways. For example, in the case of customer churn prediction, the observations in the data may be correlated with each other if there are several observations of the same customer. If the data are split into a training and test set, in a way such that there are observations of the same customers in both the training and test sets, it will be easier for the model to make predictions for these observations, and the performance estimates will be biased. The bias comes from the fact that the cross-validation procedure, in this case, does not correspond to how the model would be used in reality. In reality, there will be new customers joining the company for which there are no previous observations, and the performance estimate does not give information about how well the model will predict churn for these customers. This can be avoided by not including observations of the same customers in both the training and test sets.

3.6.1 K-fold cross-validation

K-fold cross-validation works similarly to the train-test split procedure described above, with the exception that the procedure is repeated k times. The data set is partitioned into k different subsets of the same size, then $k - 1$ of the subsets are used as the training set and the k:th subset is used as the test set. This is repeated for each of the subsets. The choice of k heavily affects the computational burden of running the cross-validation, as most models have to be retrained in each fold [12]. A common choice for k is 5 or 10. If k is chosen to be the same as the number of observations in the data set, one obtains *leave-one-out cross-validation* [12]. There is also a trade-off between the bias and variance of the performance estimates depending on the value of k. A lower value for k makes the estimates more pessimistic, since fewer data can be used for training, but it keeps the variance low. Higher values of k, with leave-one-out being the most extreme, gives an almost unbiased estimate of the performance, but the variance may be very high due to the training sets being so similar to each other.

Figure 3.2 illustrates how k-fold cross-validation would work with $k = 4$ and a data set with 20 observations. In each iteration, one-fourth of the observations are used as the test set and the rest are used for training the model. The model is trained on the training set and predictions are made on the test set. Performance metrics are computed either separately for each fold and then averaged, or computed on the whole set of predictions after all folds have been completed.

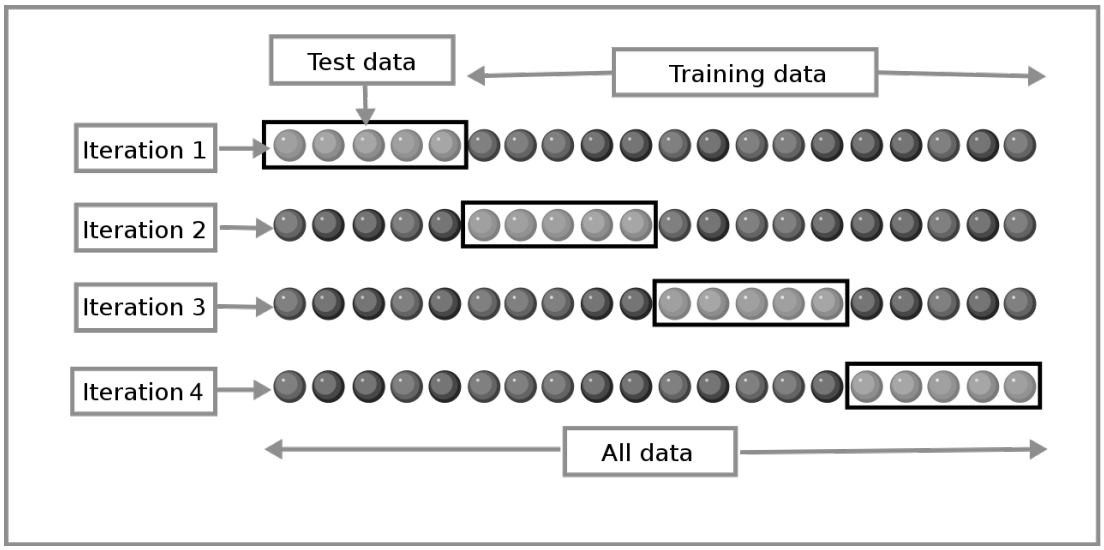


Figure 3.2: K-fold cross-validation with $k = 4$ [7].

3.7 Decision trees

Tree-based methods work by dividing the feature space into subspaces and fitting a simple model in each subspace [12]. The splitting of the feature space can be conceptualised as branching in a tree, hence the name. Each split is typically made as a decision for a single feature, assuming that the feature is continuous, one branch goes to the left for values smaller than a threshold and another branch goes to the right for values greater or equal to the threshold.

The splits are made to minimise the prediction error on the training data. The splitting continues until certain groups of observations end up in the leaf nodes of the tree. The criteria for stopping are determined by which algorithm is used for the tree construction as well as the hyper-parameters of the algorithm. The simple splits make for an easily interpretable model, as it is easy to investigate at which thresholds the splits have been made in the trained model.

Decision trees work for both regression and classification tasks. In the case of regression, the mean of the output values of the training observations in a certain node is the predicted output value. For classification, the predicted class is chosen as the most prevalent class in the node. For example in a regression model, suppose that the output values of a collection of observations in a node are $2.4, 5.5, 1.7$ and 4.3. If a new observation ends up in this node, the predicted

output for this observation would be the mean of the 4 observations in the node: 3.475. In the case of binary classification where the observations have classes $0, 1, 1$ and 1 the predicted output class would be 1 since it is the most prevalent class in the node. In the case of a tie, the predicted class could be chosen as any of the classes occurring in the node, as long as it is done consistently.

There are various algorithms for constructing decision trees from data. The *ID3* (Iterative Dichotomiser 3) algorithm was developed by Quinlan [23] in 1986. *ID3* learns a classification model from categorical data. It was succeeded by the *C4.5* algorithm which can use continuous input data by splitting the continuous feature into discrete ranges. The learned trees are then converted to sets of if-then rules, and the improvement in prediction accuracy resulting from each rule is then evaluated to determine the order in which the rules should be applied. Furthermore, the rule set is pruned to lower the chance of overfitting. Further improvements were made in the *C5.0* algorithm, being faster and more memory efficient. The *CART* (Classification and Regression Trees) algorithm is similar to *C4.5* but also supports continuous outputs, i.e. regression.

Decision trees can have high variance, meaning that a small change in the training data can affect the model drastically. This is mainly due to the hierarchical nature of the model [12]. A change in a split close to the root can affect the splits made deeper in the tree, significantly changing the topology of the tree.

3.7.1 CART algorithm for classification

The CART algorithm can construct decision trees for both regression and classification. Below is a description of the algorithm used for training a tree for a classification task.

For a classification task with K classes and training data with N observations in a d dimensional input space, there are training vectors $x_i \in R^d, i = 1, \ldots, N$ and labels $y_i \in \{0, 1, \ldots K-1\}$. The algorithm should partition the input space in a way such that each node is as pure as possible, i.e. the nodes

contain mostly one kind of label. To achieve this, a measure of impurity H is minimised. For node m, representing a region R_m of the feature space containing N_m observations, let

$$p_{mk} = \frac{1}{N_m} \sum_{y_i \in R_m} I(y_i = k)$$

be the proportion observations belonging to class k in node m.

In order to partition the feature space, a measure for the quality of partitions is required. One way is to measure the impurity $H(R_m)$ of the regions R_m in a partition. Commonly used measures of impurity in classification are *Gini* and *entropy*.

Gini impurity is defined as

$$\sum_k p_{mk}(1 - p_{mk})$$

The Gini impurity will be lower if one of the classes is more heavily represented in the node and higher if the classes are close to equally represented. It is a measure of how often a randomly chosen element from the region R_m would be incorrectly classified if it were randomly labelled according to the distribution of labels in the subset.

Entropy is defined as

$$-\sum_k (p_{mk} \log p_{mk})$$

Entropy measures the average level of information that is needed to describe the distribution of labels of the observations in the region. The entropy is low if one class is more prevalent in the region. If all classes are close to equally represented, the entropy, and therefore also the impurity, is high. To find the globally optimal partition is computationally infeasible in general, hence the splits are computed in a greedy fashion.

Let the data at node m be represented by R. Consider a feature j and a splitting threshold t, the split $\theta = (j, t)$ divides the feature space into two regions

$R_{left}(\theta) = \{(x_i, y_i) \mid x_{i,j} \leq t\}$ and $R_{right}(\theta) = \{(x_i, y_i) \mid x_{i,j} > t\}$

The impurity G for the split is computed using the formula

$$G(R, \theta) = \frac{n_1}{N_m} H(R_{left}(\theta)) + \frac{n_2}{N_m} H(R_{right}(\theta))$$

where n_1 and n_2 are the number of observations in region R_{left} and R_{right} respectively, and N_m is the total number of observations at node m. $G(R, \theta)$ can be seen as a weighted average of the subregion impurities. To find the split that minimises the impurity, all possible thresholds are evaluated for all features in R.

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, G(R, \theta)$$

If the feature is continuous, there may be numerous possible thresholds that have to be evaluated. To reduce the computational burden, a continuous feature may be binned so that only thresholds at the boundary points of the bins have to be evaluated.

The optimal splits are calculated recursively for subsets $R_{left}(\theta^*)$ and $R_{right}(\theta^*)$ until one of the criteria for stopping is met. Stopping criteria can be a maximum allowable depth of the tree, a maximum number of leaf nodes, a minimum number of observations in a leaf node or a minimum required decrease in impurity.

The optimal split is found by brute force search, that is, the impurity is computed for every possible value, for every possible feature, and the split that gives the lowest impurity is chosen. It should be noted that this tree-building procedure is greedy in the sense that at every stage the impurity is only minimised for the current split, without regards to the global optimum. Therefore, it is not guaranteed that the tree that is built is the one with the lowest impurity. Still, the quality of the produced tree is often good enough when the training data set is sufficiently large, and the suboptimal solution may even help avoid overfitting to the data.

The tree depth is a model hyper-parameter that may be tuned to adapt the model to the data. A very large tree may overfit the training data, while a small tree may not capture all the relevant structures in the data.

To perform prediction for new observations the tree is traversed according to the splitting criteria in each node, until a leaf node is reached. The predicted class of an observation in the leaf node m is the majority class in the node

$$k(m) = \underset{k}{\operatorname{argmax}}\, p_{mk}$$

. That is, the class k that has the highest proportion p_{mk} in node m is chosen.

3.7.2 Example

The artificial "two moons" data set seen in figure 3.3 is an example of a binary classification problem with a non-linear decision boundary. The light dots belong to one class and the dark dots to the other class. There are two features: X_0 and X_1.

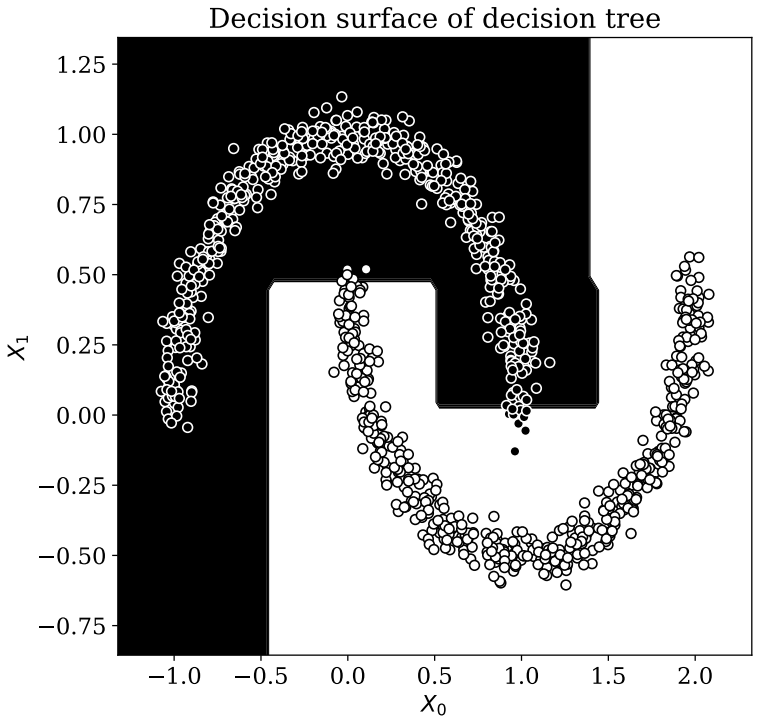


Figure 3.3: Decision surface of the learned model

If a decision tree is fitted to the data with the restrictions of 7 as the maximum number of leaf nodes and a maximum depth of 4, then the tree structure shown in figure 3.4 is learned. The tree has been constructed using the CART algorithm with Gini as the measure of impurity.

The coloured regions in figure 3.3 are the decision surfaces, i.e. the regions in feature space where an observation would be predicted to belong to a certain class. The tree has learned a non-linear decision boundary, which fits the data quite well.

The rectangular regions of the decision surface are the results of the splits in the tree. The first split divides the feature space into two regions: $R_{left}((1, 0.46)) = \{(x_i, y_i) \mid x_{i,1} \leq 0.46\}$ and $R_{right}((1, 0.46)) = \{(x_i, y_i) \mid x_{i,1} > 0.46\}$. The first region contains the observations for which feature X_1 is less than or equal to 0.46 (the left subtree), and the second region contains the observations for which feature X_1 is greater than 0.46 (the right subtree). This partition of the feature space is the horisontal dividing line at $X_1 = 0.46$ seen in figure 3.3. Similarly, the other splits in the tree correspond to the other lines that divide the light and dark regions. Each subtree divides the regions further.

In figure 3.4, the nodes in the tree are coloured by the class that would be predicted at that node. The array in each node shows the number of observations of each class, and the prediction is chosen as the majority class in a node. For example, a new observation $[X_0, X_1] = [1.50, 0.5]$ would end up in the rightmost node in the tree, since the first condition $0.5 \leq 0.46$ is false and the second condition $1.5 \leq 1.424$ is also false. The predicted class would be 1 (light) since it has the majority with 8 to 0 in that node.

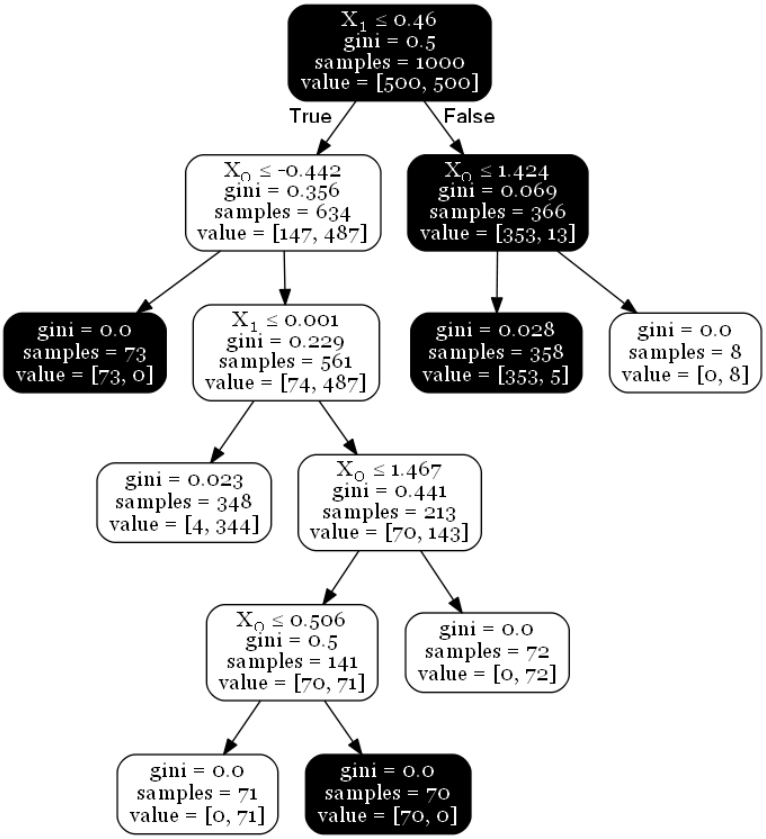


Figure 3.4: Decision tree learned from the data

3.8 Ensemble methods

In ensemble methods, the goal is to combine several models into a single model, in order to improve its robustness and ability to generalise, compared to any of the single models alone. One form of ensemble method is *Bagging* (Bootstrap aggregating). Bagging is a method where several models are trained on bootstrap samples of the same training data, and the predictions of the individual models are averaged to produce the final prediction. Bootstrap sampling means that the training data set is sampled with replacement to produce training sets for each of the models in the ensemble. By averaging, variance is reduced, which helps avoid overfitting the training data [18]. The essential idea is to average many noisy but approximately unbiased models and, thus, reduce the variance.

The models used in Bagging should be able to capture complex relations in the data, hence, trees are an ideal candidate since they are highly flexible. Furthermore, they are approximately unbiased if grown sufficiently deep. Each

tree generated in bagging is identically distributed and, therefore, the expectation of an average of the trees is the same as the expectation of any individual tree. This means that the bias of the bagging model is the same as that of the individual trees, and the only way to improve the model is through variance reduction [12].

The variance of the average of N identically distributed random variables with variance σ^2 and positive pairwise correlation ρ is

$$\rho\sigma^2 + \frac{1-\rho}{N}\sigma^2$$

. As N grows, the second term diminishes. This means that the variance of the model can be decreased by increasing the number of trees in the ensemble. However, the variance is limited by the term $\rho\sigma^2$ and, thus, by the correlation between the trees.

3.8.1 Random forest

Random forest is an ensemble method, closely related to bagging, based on decision trees as the base learner. It was introduced by Breiman in 2001 [1]. The model consists of an ensemble of several trees. Each tree in the ensemble is built using a bootstrap sample from the training data. Furthermore, during tree construction, only a random subset of the features is considered at each node split. These modifications introduce randomisation into the learning process, and the random feature selection makes the trees less correlated [12].

The idea is that random forest can improve upon the bagging method by making the trees less correlated, without increasing the variance too much. The random feature selection works by selecting k of the d features to split on, $k \leq d$, at each splitting. By default, the decision tree algorithms chose one feature to split on from all the features. The value k is typically chosen to be $\lfloor\sqrt{d}\rfloor$. In practice, this parameter should be tuned for best performance.

The predictions of each tree in the ensemble are combined to produce the final prediction. This is achieved by taking either the most prevalent class or

averaging in case of probabilistic predictions.

Due to the random selection of features at each split, the bias of each tree is slightly increased. The general trend is that the bias increases as the number of selected variables at each split decreases [12]. However, due to the averaging during prediction, the variance of the model decreases, compensating for the increase in bias.

With random forest, the training of individual trees is performed independently and can thus be performed in parallel. Therefore, this allows for the training time to be decreased by taking advantage of parallel computing architectures.

3.8.2 Model explainability

Model explanation is the concept of trying to understand why a machine learning model gives certain predictions for certain inputs. This is especially relevant for black-box models, meaning models that are too complicated for humans to understand directly. The difficulty understanding may arise from the model including parameters that do not have an intuitive interpretation, or simply that it has a large number of parameters.

Random forest models usually consist of a large number of trees, each with their own set of parameters specifying the splits. Even if individual trees are usually considered to be interpretable models, in a random forest the trees have not been trained on identical input data since the data is randomised, making them incomparable to each other. Also, predictions of individual trees are combined to produce the final prediction of the model, making it more difficult to understand the connection between input and output. Thus, random forest can be seen as a black-box model.

3.8.3 Feature importance

In a random forest model, one way to assess which features the models find important is through calculating *feature importances*. Feature importance provides estimates of the predictive power of the features. The importance value for one feature tells us how much that feature contributes to the prediction, compared to the other features.

The importance of a feature can be calculated as the normalised total reduction of the impurity caused by splitting on that feature. In a random forest, the importance values are calculated in the individual trees, and then averaged over the whole forest. Since the impurity measures how well the observations belonging to different classes have been separated in the split, this gives a direct way to measure how well that feature can separate the classes.

3.8.4 Partial dependency

The impact a feature has on the score can be investigated by plotting the partial dependence of the feature. One-way partial dependence plots show the interaction between the target feature and the output. The partial dependence of a feature is computed by averaging the logarithm of the predicted scores for every combination of the target value and the rest of the features.

To estimate the partial dependence for a feature:

- The partial dependence of a specific value of the feature is computed by holding that value constant and varying all other features over their possible values and predicting the score for every combination.
- Then the logarithm of the scores is computed, and the mean is subtracted from the scores. The partial dependence is the mean of these scores.
- This is repeated for every possible value of the feature.

Chapter 4

Data set

4.1 Data set

The data set used in this study consists of customer data from a bank. The data are updated daily, and at the end of each month, one snapshot of the data are saved. The data consist of information about approximately 300 000 customers, including demographic data, savings amounts, payment and loan amounts.

The data set used for analysing and modelling customer churn consists of data from the period 31 January 2015 to 31 December 2017. The data are filtered according to five criteria. The customers included in the study have a total loan volume of 50 000 euro or higher. This threshold was given by the client company and is used to avoid including customers who do not have a substantial amount of loans. Only *private customers* are included in the analysis. The term *private customer* is used by the company to denote customers who are not corporate and institutional customers. Only customers who are Finnish citizens are included in the analysis. Customers who have displayed payment problems in the past are excluded. These conditions together limit the number of customers to about 40 000 in each monthly data set.

The threshold of 50 000 euro for the total loan volume has been chosen

in an attempt to leave out customers without a home loan from the analysis. The decision is based on the observation that the number of customers with a certain loan volume size "stabilises" for volumes higher than 50 000 euro before gradually decreasing.

Figure 4.1 shows the number of customers who had a loan volume within a certain range, each range increasing with 10 000 euro increments. For example, there were about 2 600 customers with a total loan volume between 50 and 60 thousand euro at the end of October 2016. The distribution of loan volumes is quite similar for every monthly snapshot, within the time span considered. In this figure, customers with a total loan volume below 50 000 euro are also included.

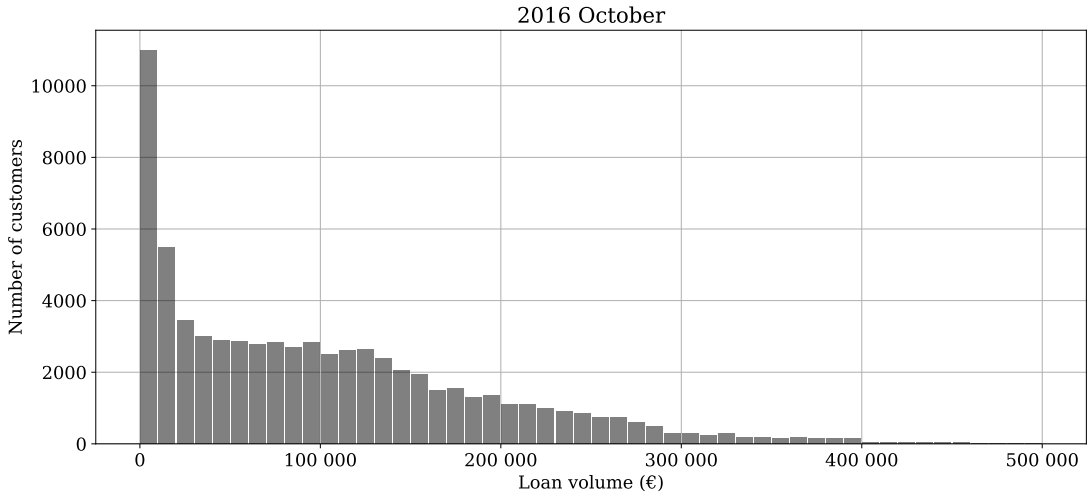


Figure 4.1: Distribution of loan volumes

If the number of customers in each range is subtracted successively from the total number of customers, the complementary cumulative distribution is obtained. In figure 4.2, the complementary cumulative distribution of loan volumes is shown. Each bar shows the number of customers who have a loan volume greater than or equal to the value on the x-axis. The vertical dashed line indicates the 50 000 euro threshold. The number of customers with a volume greater than or equal to the threshold, as well as, the proportion they constitute of the customer base, is shown in the legend. In this figure, customers with a total loan volume of less than 50 000 euro are also included. In October 2016, about 41 000 customers had a total loan volume of 50 000 euro or more. This number constitutes about 61% of the customers considered.

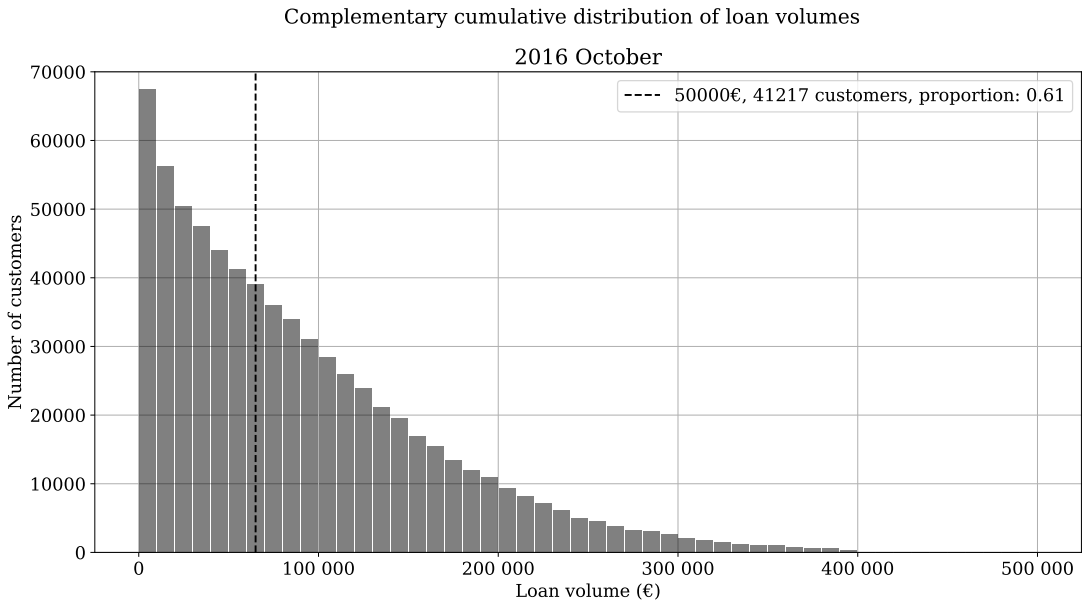


Figure 4.2: Complementary cumulative distribution of loan volumes

4.2 Definition of the churn variable

In this thesis, churn is defined in terms of changes in loan volumes. The definition is assumed to label those customers who transfer a large part of their loans as churners.

Definition A customer is said to have churned if their total loan volume has dropped by 80% or more, six months into the future. This percentage threshold is chosen, since it seems to capture the churn phenomenon.

The churn phenomenon is assumed to be stable over time, meaning that the indicators of churn are the same in the future as they are now. For example, if the age of a customer is found to be a strong predictor of churn, it should also be a strong predictor in the future. This is a modelling assumption that affects how the data are prepared for the model, and how the results should be interpreted.

The complementary cumulative distribution of changes in loan volumes for October 2016 can be seen in figure 4.3. Here, only customers with a total loan volume of 50 000 euro or more are included.

The x-axis shows how much the loan volume has decreased over six

months in percentages, and the y-axis shows how many customers have had a decrease of this percentage or greater. For example, about 1 000 customers have shown a decrease of 80% or more in their total loan volume and would, according to the definition, be labelled as churners. Note that the scale of the y-axis is logarithmic.

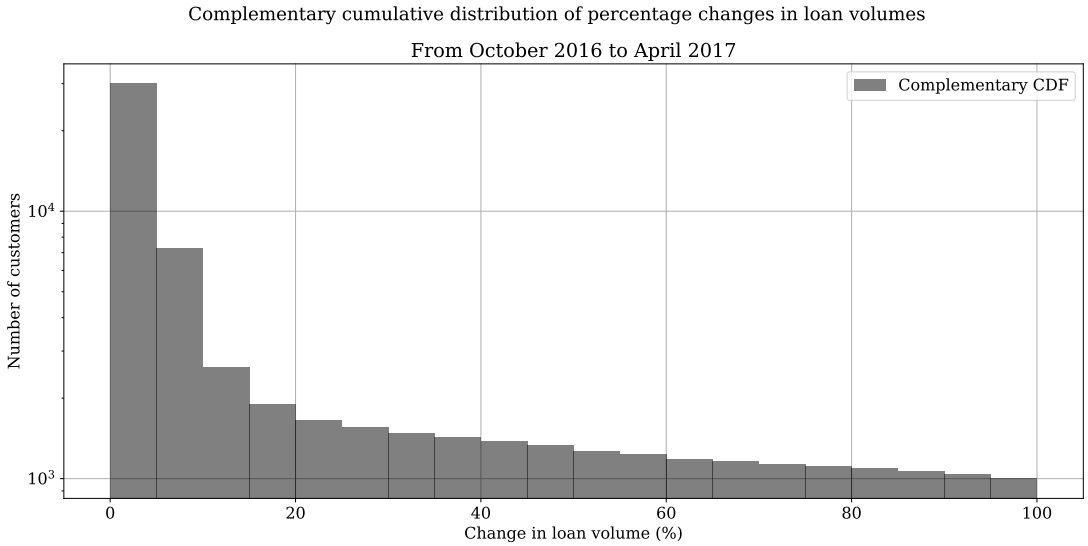


Figure 4.3: Percentage change in loan volume

The churn rate defined in this manner is quite stable over time, with about 1 000 customers labelled as churners per month. However, this number may not be representative of the true number of churners due to how the churners are counted. A customer who has churned, according to the definition, can still be a customer in the following months and again be labelled as a churner.

For example, if a customer has a loan volume of 120 000 euro on 31 January 2015 and transfers the loan in June 2015, it means that the loan volume on 30 June 2015 will be 0 euro. Then it means that the customer will be labelled as a churner 31 January 2015, since the change in volume is over 80%. Even if the customer only pays off a small amount of the loan in February, the customer will still be labelled as a churner 28 February 2015, since the decrease in loan volume will still be over 80%.

Some churners are accounted for twice, meaning that the actual churn rate is lower than the one shown in the previous figure. In the predictive model,

this issue may be handled by sampling the data, for example by only using the most recent observation of each customer and, thus, avoiding an ambiguous churn status. If double counting is avoided by keeping track of the customers that have already been labelled as churners in previous months, the calculated number of churners is significantly lower, about 200 per month.

Figure 4.4 shows the ratio of customers labelled as churners, without double counting, as a ratio of the total number of customers (dark coloured line), as well as the total number of customers (light coloured line) for each month. All customers have a total loan volume greater or equal to 50 000 euro before the moment of churn. Notably, the churn rate is quite low, 0.5% on average for the period in question.

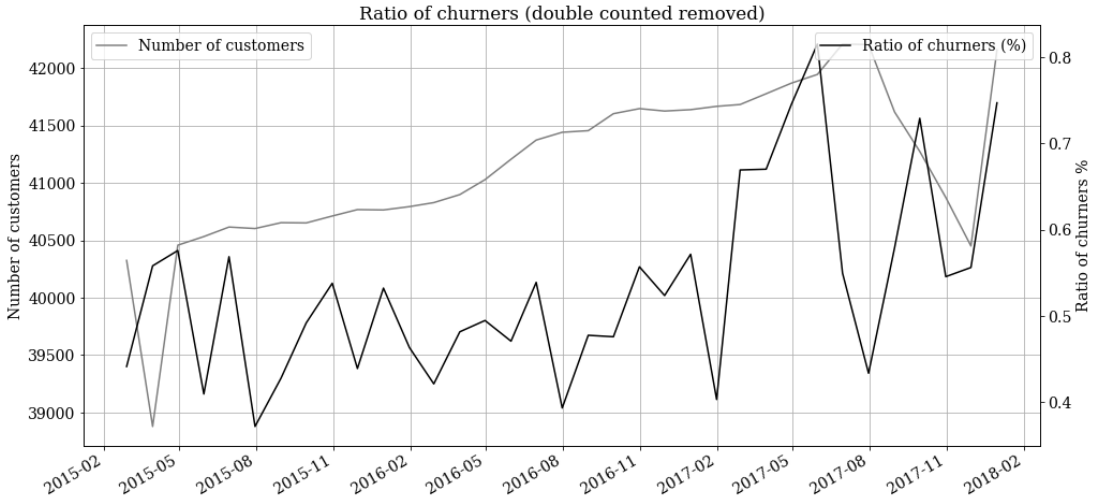


Figure 4.4: Ratio of churners to the number of customers included.

4.3 Implications of the definition

A major issue with the definition is that it does not explicitly capture the event of interest, namely when a loan has been transferred. The definition tries to capture churn by measuring a proxy, a large change in the loan volume. This results in customers possibly being labelled as churners even though they are not, for example, if a customer has paid off over 80% of the loan within six months. We try to avoid this kind of mislabelling by choosing a reasonably high starting volume and a required change in volume.

4.4 Features

Table 4.1 shows the features used in this study. These features have been chosen since it is believed that they can be relevant for predicting churn, and data for them have been stored as monthly snapshots for a long time. The left column in the table shows the name of the feature, and the right column contains a short description of each feature. Most of the descriptions are obvious, but a few require some elaboration. *Premium* and *Privileged* are customer segments defined by the client company, the definition is based on which products the customer subscribes to. For example, to be eligible for the Premium banking product, the customer must have funds, savings or products worth 50 000 euro at the company.

The feature named *Sector* is a categorical variable indicating the economic sector the customer works in. It includes seven categories: "Non-profit institutions serving households", "Employers and own-account workers", "Employees", "Recipients of property income", "Recipients of pensions", "Recipients of other transfers" and "Non-financial corporations, excl. housing corporations, national private".

There were some missing values in the data, but all were in the continuous variables and the missingness was not random. It was concluded in discussions with the client that the missing values in these columns are a different encoding for the values being zero and, therefore, all the missing values could be replaced with zeros.

Name	Description
Private banking	Private banking customer
Premium	Customer belongs to certain customer segment according to the company definition
Privileged customer	Customer belongs to certain customer segment according to the company definition
Customer category	Customer type
Sector	Sector (economy)
Cross sales index	Number of active product categories
Cross sales product categories	Active product categories
Age	Age of the customer
Civil status	Civil status (categories)
Language	Customer service language
Postal code	Postal code of residence
Revenue 12M	Revenue of customer (running mean of last 12 months)
Predicted revenue	Predicted revenue of customer (calculated based on active products and assets)
Payment volume	Amount of money used for payments by the customer
Savings volume 1	Amount of money in assets that bring profit to the company: e.g. deposits/funds
Savings volume 2	Amount of money in assets that do not bring profit to the company: e.g. stocks
Loan volume	The combined amount of money in loans
Payment volume (shared)	Payment volume for shared accounts
Savings volume (shared)	Savings volume for shared accounts
Loan volume (shared)	Loan volume for shared accounts

Table 4.1: Features used in the model

Chapter 5

Empirical study

5.1 Empirical study

A model for predicting loan churn is built and evaluated using customer data from 2015-01-31 to 2017-12-31. The model learns to score customers according to their likelihood to churn, giving a higher score to customers who are more likely to churn.

5.1.1 Predictive model

A random forest model is used to predict customer churn. In this thesis, the random forest implementation from scikit-learn 0.19.1 [21] is used. The hyperparameters used for the model are described in table 5.1. The hyperparameters control different aspects of how the model is trained, but all of them can be explained in terms of the theory described in section 3.7.

The parameter *number of trees* is the number of decision trees used in the ensemble model, in this case, it is chosen to be 100, which is quite moderate. A moderate number of trees makes training and evaluating the model faster, while having the benefits of the ensemble. *Split criterion* is the impurity measure H used to decide the quality of the splits. *Maximum number of features* is the number of randomly selected features that are considered while performing each

split of the feature space. The parameter value is kept at the recommended value, i.e. the square root of the number of features.

The following parameters are stopping criteria. *Maximum depth* determines how deep the trees in the ensemble are allowed to grow, while the depth is defined as the largest number of splits from the root node to a leaf node. *Minimum number of samples for split* is a stopping criterion for a branch of a tree. If the number of observations in one node is under this limit, the node is not split. *Minimum number of samples in leaves* determines if a split can be done, by considering the number of observations in the resulting child nodes. If the resulting child nodes have fewer observations than this value, the split is not performed. *Maximum number of leaf nodes* determines if splitting should stop for a tree by checking the number of leaf nodes; should the number of leaves exceed this limit, splitting stops. *Minimum impurity decrease* determines if a split should occur by checking the decrease in impurity. If the resulting decrease is lower than this threshold, the split is not performed. The stopping criteria used here has been chosen as to not restrict the building of trees in other ways than the training data would allow for.

5.1.2 Data preparation

The data set contains the information described in table 4.1 as inputs, as well as a binary output variable. The output variable takes the values 1 if the customer is a churner in that month or otherwise 0. There are 36 months in this period and about 40 000 customers per month, which results in a little over 1 480 000 observations in total. Still, a majority of these observations are of the same customers, since there is neither a large number of customers joining nor leaving during this period.

The data are prepared for training and evaluating the classifier model by using the most recent observation of each customer. Since most of the customers in all the periods are the same, the size of the data set is reduced significantly.

Hyperparameter	**Value**	**Description**
Number of trees	100	The number of trees in the forest.
Split criterion	Cross-Entropy	The measure of quality for splits.
Maximum number of features	$\sqrt{41}$	The number of features considered when looking for the best split.
Maximum depth	unlimited	The maximum depth of trees.
Minimum number of samples for split	2	The minimum number of observations required to split an internal node.
Minimum number of samples in leaves	1	The minimum number of observations required at a leaf node.
Maximum number of leaf nodes	Unlimited	Restriction on the number of leaf nodes in each tree.
Minimum impurity decrease	0	A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

Table 5.1: Random forest hyperparameters

In this sampled data set, there are 52 681 customers, 7 313 of whom are churners (14%). Since every observation in the sampled data set comes from distinct customers, it is assumed that every observation is independent.

Then 10-fold cross-validation is performed using the sampled data. This means that in each fold 90% of the data is used for training and 10% is left for testing the model. In each fold, the training and test data are different, and the cross-validation is stratified, meaning that the ratio of churners is the same in every fold.

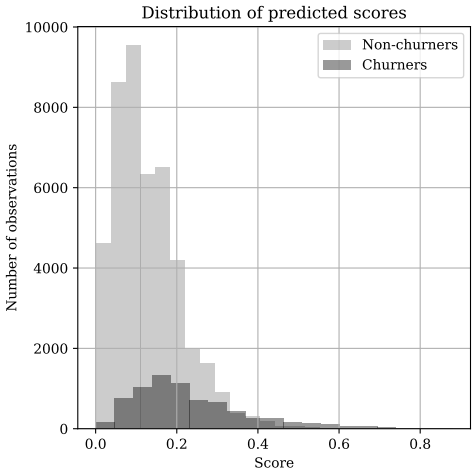
The results are predictions for every observation in the data set, where the observation the prediction is made for was not used to train the model. This means that the predictions are not biased, which would be the case if they were made on the same observations as were used for training the model. Thus, they can be used to calculate unbiased estimates of the model's out-of-sample

performance.

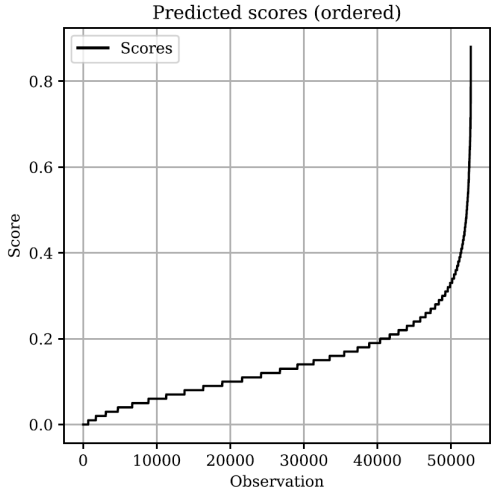
5.2 Results

The distribution of the predicted scores is shown in figure 5.1a. Due to the imbalance in the data set, the predicted scores are concentrated around 0.14. The classifier is not able to perfectly separate the classes, meaning that the predicted scores for some churners are lower than the scores for some non-churners. The histogram suggests that the classifier gives low scores to most customers.

The scores ordered by their magnitude 5.1b were plotted for the observations in the test set. The plot follows a sigmoid shape, which is typical for the scores produced by a random forest model. The sharp increase in the curve shows that there are fewer customers who receive a high predicted score. This reflects the fact that the proportion of churners to non-churners in the data is low.



(a) Distribution of predicted scores.



(b) Scores in sorted order.

Figure 5.1: Predicted scores.

5.2.1 Calibration

The scores should not be interpreted as probabilities directly. This is because the model is uncalibrated. In figure 5.2, the predicted scores and the average number of positive observations are plotted against each other. The

scores and true classes have been sorted by the magnitude of the predicted scores, grouped into 10 bins and averaged. The curve shows how well the scores of the classifiers correspond to the true empirical probabilities in the data. A classifier is perfectly calibrated if the calibration curve follows the diagonal line. For example, if a calibrated classifier predicts a probability of 0.80 for a group of observations, 80% of them belong to the positive class.

According to this plot, the model is quite well calibrated, but it gives slightly too high scores for non-churners and too low scores for churners. This behaviour is expected of the random forest algorithm [20]. Attempts at calibration were made by applying Platt Scaling and Isotonic Regression, but it did not improve the results. Even if the scores cannot be interpreted as probabilities directly, they can still be used to order the customers according to their likelihood to churn.

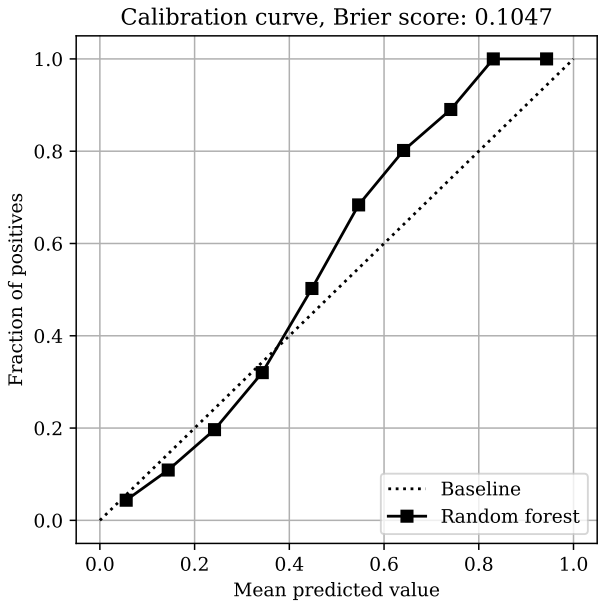


Figure 5.2: Calibration curve for model predictions.

5.2.2 Model performance

To turn the predicted scores into a predicted class, a threshold has to be chosen. Depending on where this threshold is placed, the number of false positive and false negative predictions vary. The trade-off can be visualised by

plotting the true positive rate (TPR) against the false positive rate (FPR). The resulting plot is called the receiver operator characteristic (ROC) curve. The true positive rate estimates the models probability of detection, i.e. the probability that the model will identify a true churner. The false positive rate estimates the probability of a false alarm, i.e. the probability that the model will label a non-churner as a churner.

The area under the curve (AUC) can be used for summarising the performance of the classifier. An AUC of 1 is the best case, where the classifier can distinguish between the positive and negative class perfectly, whereas 0.5 is the worst, meaning that the predictions are no better than randomly assigned scores. The AUC can be interpreted as the probability that the classifier will rank a randomly chosen churner higher than a randomly chosen non-churner.

The ROC curve for the predictions is shown in figure 5.3. Since the evaluation is done through a 10-fold CV, the models in each fold differ slightly due to them being trained on different parts of the data set. To obtain an overall estimate of the performance of the classifier, the ROC curves of each fold are averaged using the arithmetic mean. In the figure, an interval with the width of one standard deviation is drawn around the mean ROC curve. The area under the curve is shown in the legend, also with a one standard deviation interval.

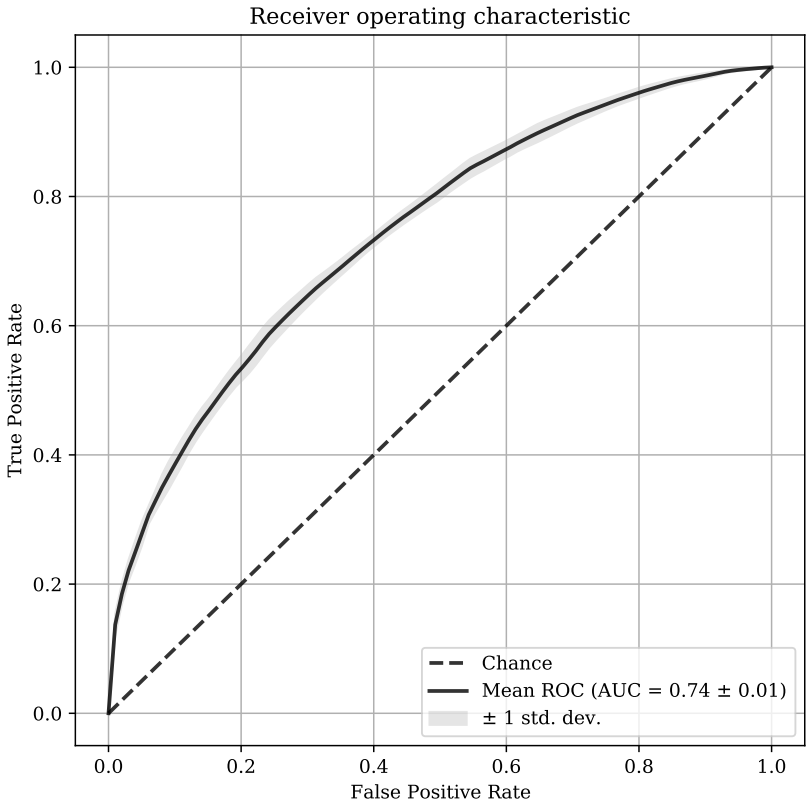


Figure 5.3: ROC curve for model predictions

Figure 5.4 shows the cumulative gains curve for the predictions. The cumulative gains curve shows the percentage of the overall number of churners that would be identified by considering a certain percentage of the total number of customers. For example, the height of the curve at 20% of the total number of customers is about 50%. This means that if the classifier is used to score a data set and sort it by the scores, we expect 50% of the churners to be in the group with the top 20% highest scores. Thus, if the data set has 40 000 customers of which 200 are churners, and 8 000 customers (20%) with the highest scores are selected, we expect to find 100 (50%) of the churners.

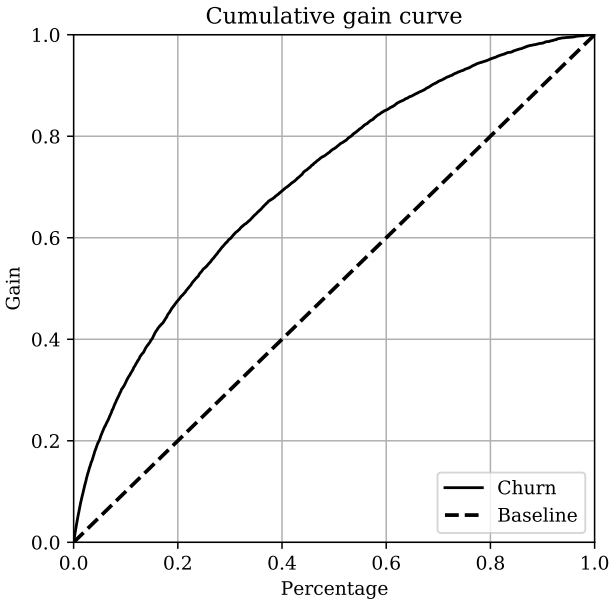


Figure 5.4: Cumulative gains curve for model predictions

The threshold that gives the best balance between TPR and FPR is found by maximising Youden's J statistic. The statistic can be seen as the height of the ROC curve above the diagonal line. The J statistic for different score thresholds is shown in figure 5.5. We see that the maximum value is obtained when the threshold is set to about 0.17.

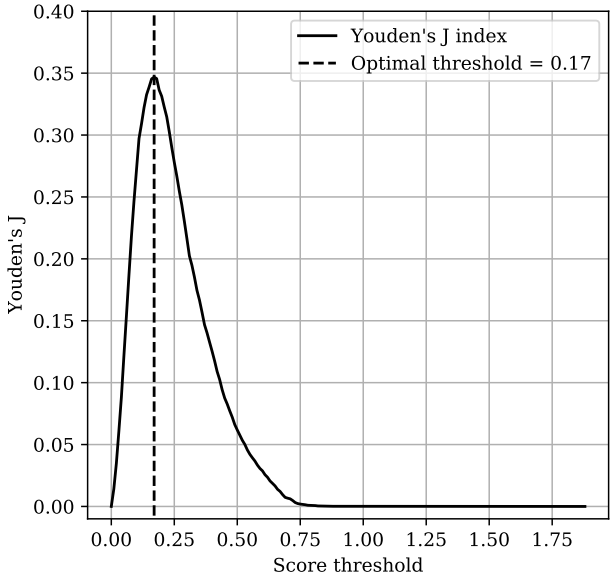
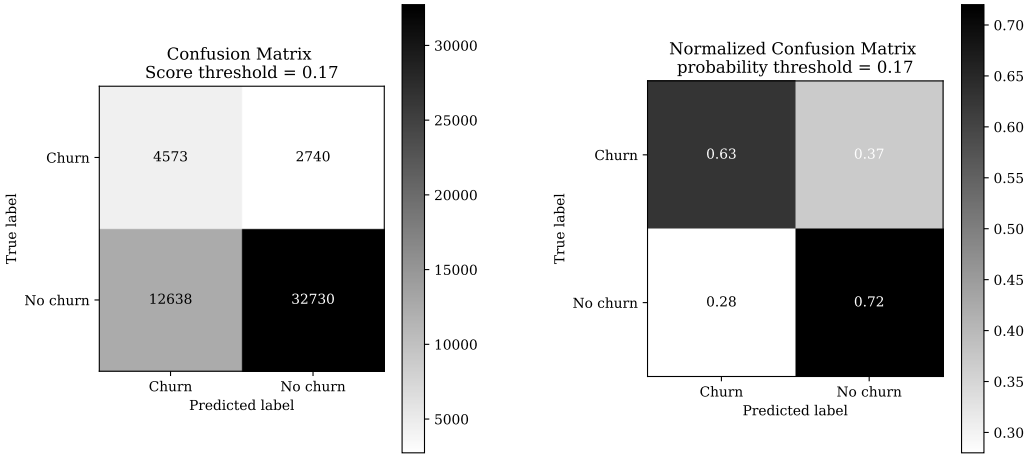


Figure 5.5: Youden's J statistic, and the score threshold that gives the highest value.

The optimal threshold is used to produce the predictions in the following confusion matrices in figure 5.6. With this threshold, the model can classify 67% of the churners correctly, while classifying 72% of the non-churners correctly.



(a) Confusion matrix. (b) Normalised Confusion matrix.

Figure 5.6: Confusion matrices.

5.3 Model evaluation

5.3.1 Feature importances

The top 15 of the features ordered by importance are shown in figure 5.7. The features that the model finds most important seem to be related to the revenue, loan volumes, and location.

5.3.2 Partial dependency

The partial dependence, described in section 3.8.4, shows how the values of the features affect the predicted score, according to the model. The partial dependence for a few selected features is shown in figure 5.8. There seems to be a non-linear dependence between these features and the predicted churn score. For example, in the feature *Loan volume* there is a decrease in the partial dependence between 50 000 and 170 000 euro, after which the score increases until 400 000 euro.

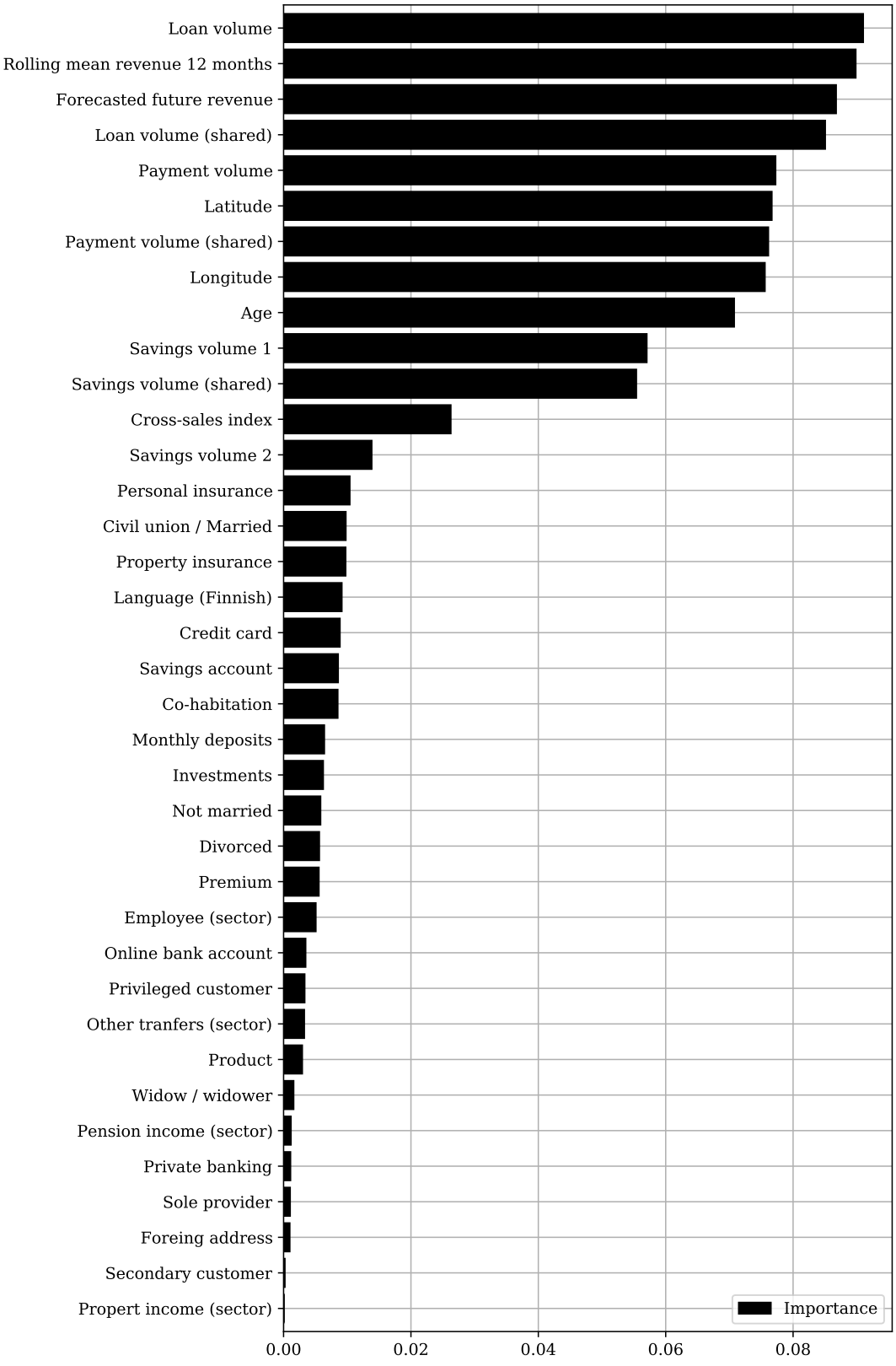


Figure 5.7: Feature importances.

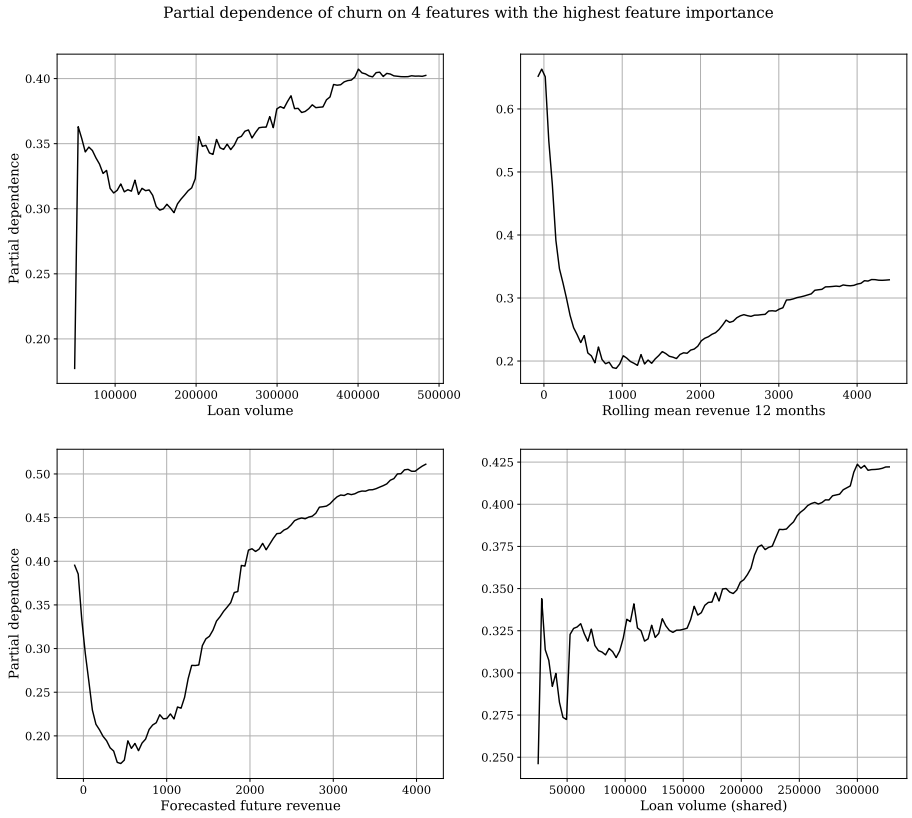


Figure 5.8: Partial dependence for four features.

Chapter 6

Conclusions

The purpose of this work has been to review customer churn prediction and its applications, theoretically and through empirical study. This was accomplished through a study of the literature and a single case study. The goal of the study was to establish if it is viable to use a machine learning model for modelling and predicting customer churn at the client company.

The first part of the thesis gave a general overview of customer churn, how it can be defined, the economic benefits of churn prediction, and variables that can be used for predicting churn. Challenges common for customer churn-related data, such as class imbalance and missing values, were also discussed. Then the modelling aspect of churn analysis was described, which machine learning models are typically used, as well as their advantages and disadvantages. Furthermore, the procedures for evaluating models and their explainability were described.

The second part focused on the empirical analysis. First, the data set at hand was explained, what features were available and how the churn variable was defined. Next, the methods discussed in the first part was applied to the data. Moreover, the data preparation, modelling, as well as the evaluation of the model were described.

The results show that economic variables related to income, spending

and saving are the most important followed by demographic variables. By studying partial dependencies of the features, it was observed that the churn score is highly dependent on the magnitude of certain features. By evaluating the model using cross-validation, it was found that the model has a reasonable performance on the data set, with an AUC score of 0.74. This gives support to the viability of using machine learning for predicting customer churn in this setting.

Att förutspå kundbortfall inom finansindustrin

Introduktion

Denna avhandling behandlar hur kundbortfall (eng. Customer churn) kan förutsägas vid ett företag inom bankindustrin. Med kundbortfall avses ett avslutat kund-förhållande. Mer specifikt handlar det om sådana kunder som flyttar sina lån från en bank till en annan. Fokus ligger på att undersöka hur dessa kunder kan identifieras genom användning av maskininlärning. Frågan närmas på detta sätt eftersom det kan beskrivas som binär klassificering inom ramverket av väglett lärande (eng. supervised learning). Därtill har företaget en stor mängd data om kunderna, som kan användas vid träningen av maskininlärningsmodeller.

Avhandlingen är strukturerad som en fallstudie, och kan delas in i tre delar. Den första delen tar upp begreppet kundbortfall i allmänhet: hur det kan definieras, varför fenomenet är av intresse för företag och vad nyttan är med att försöka förutsäga kundbortfall. Den andra delen tar upp grundläggande teori för maskininlärning: vanliga problem med data som ska användas för väglett lärande, hur prediktionsmodeller kan evalueras, och teorin för beslutsträd och ensemblemodeller behandlas eftersom denna typ av modeller används i den empiriska delen av avhandlingen. I den tredje delen beskrivs det praktiska arbetet som gjorts för ett företag, som del av ett konsulteringsuppdrag. I denna del redogörs för olika aspekter på de data som används som inlärningsmaterial för prediktionsmodellen, och hur kundbortfall definieras för detta ändamål. Sedan

presenteras den valda maskininlärningsmodellen, parametrar som användes vid träning av modellen och resultaten från evalueringen av modellen. Det avslutande stycket lägger fram de slutsatser som gjorts utifrån studien.

Kundbortfall

Termen kundbortfall används för att beskriva företeelsen då kunder säger upp sitt förhållande med ett företag för gott. Trots att det är enkelt att beskriva begreppet, kan det vara svårt för ett företag att veta om, och exakt när, en kund avslutar sitt förhållande. Inom branscher där förhållandet är kontraktsbaserat, såsom försäkrings- och telekommunikationsbranchen, kan det vara lättare att avgöra när förhållandet sagts upp. Däremot kan det vara svårare för ett företag inom detaljhandeln att säga exakt när bortfallet sker, eftersom det kan ske gradvis.

Ofta krävs det att man fastställer en definition för när en kund kan anses ha fallit bort, vilket gjordes i denna avhandling. En ändamålsenlig definition skapades genom att välja de kunder som ansågs ha en ansenlig mängd lån, 50 000 euro eller mera. Detta krävdes för att endast inkludera kunder med bostadslån. Sedan kontrollerades hur mycket mängden lån hade förändrats över sex månader. Ifall mängden sjunkit över 80% ansågs kunden ha förflyttat sitt lån. Förändringen på 80% valdes utifrån mönster som kunde hittas i data, och för att det ansågs vara en så stor förändring att lånet inte kunde ha betalats inom loppet av sex månader. Tidsgränsen sex månader valdes på företagets begäran, och ansågs vara en lämpligt lång tid för att kunna åtgärda det genom kundservice eller marknadsföring om kunden skulle förutses falla bort inom den tiden.

Teori för prediktionsmodeller

Den grundläggande idén med maskininlärning är att automatiskt lära sig mönster som finns i data. Det innefattar algoritmer som med hjälp av den information som finns i data kan lära sig att lösa uppgifter de inte har explicita instruktioner

för att lösa. Med väglett lärande avses sådana problem där man har exempel bestående av både in- och utdata. Algoritmens uppgift är att lära sig regler som kopplar indata till rätt utdata, och att generalisera från de givna exemplen.

I denna avhandling har maskininlärningsmodellen "slumpmässig skog" (eng. random forest) använts för att lära algoritmen kopplingen mellan kunddata och benägenheten att flytta sitt lån. Modellen faller under kategorin ensemblemodeller, och består av en samling beslutsträd som tränats med slumpmässigt valda delmängder av träningsdata. Den slumpmässiga skogen gör förutsägelser genom att ta medeltalet av förutsägelserna från de enskilda träden som ingår i modellen. Beslutsträd är en enkel modell som byggts upp av en algoritm som delar upp indata i olika regioner så att varje region innehåller så liknande exempel som möjligt. Slumpskogar ger ofta bättre resultat än ett enskilt beslutsträd, eftersom de slumpmässiga valen motverkar beslutsträdens benägenhet att överanpassa (eng. overfit) sig till data.

Utöver val av modell är det också avgörande att modellen evalueras på ett korrekt sätt. Då är det viktigt att man väljer ett sätt att mäta prestandan hos modellen som tar i beaktande egenskaper hos data man utför evalueringen med. I typen av data som behandlas här finns det ofta en stor obalans i träningsdata, eftersom det är mycket färre kunder som faller bort jämfört med de kunder som är kvar. Det är också viktigt att evalueringen utförs på ett sätt som inte introducerar bias. Exempelvis skulle det ge felaktiga resultat om modellen evalueras med samma data som den tränats med. Av den orsaken ska korsvalidering användas så att tränings- och testdata hålls åtskilda.

Data och empirisk studie

Data som används för denna studie är kunddata från en bank som samlades under perioden 31 januari 2015 till 31 december 2017. Data för träning av modellen väljs ut så att endast finländska privatkunder med över 50 000 euro lån beaktas. Kunder med betalningsproblem från tidigare tas inte med. En kund markeras som bortfallen om mängden lån som denne har sjunkit med över 80% inom 6 månader.

Ett problem med denna definition är att kunder som markerats som bortfallna under en månad, kommer att vara markerade som bortfallna även i följande månad. Till exempel om en kund har ett lån på 120 000 euro i januari och flyttar lånet i maj, så kommer kunden att markeras som bortfallen eftersom mängden lån sjunkit med över 80% mellan januari och juli. Men samma kund kommer också att markeras som bortfallen i februari eftersom mängden lån sjunkit med över 80% från februari till augusti. För att få giltiga träningsdata tas därför endast den senaste observationen av kunden med i mängden av träningsdata. Således är varje exempel i träningsdata från unika kunder och kan ses som oberoende.

Maskininlärningsmodellen evalueras enligt tiodelad korsvalidering. Det betyder att träningsdata delas upp i tio lika stora delar där balansen mellan klasserna, bortfallen eller icke-bortfallen, är jämn. I varje iteration av korsvalideringen skapas prediktioner för en av de tio delarna, efter att modellen tränats på de nio resterande delarna. Detta upprepas för alla tio delar så att modellen har skapat förutsägelser för alla data utan att ha tränats med samma data som förutsägelserna görs för. De förutsagda klasserna jämförs med de korrekta klasserna och från dessa räknas modellens AUC (eng. Area under Receiver Operating Characteristic Curve) poäng, som ger ett estimat av hur bra modellen är.

Avslutning

I denna avhandling har prediktionen av kundbortfall undersökts genom att studera den existerande litteraturen och att utföra en fallstudie. Målet var att bilda en uppfattning om möjligheterna för en bank att använda sig av maskininläring för att förutsäga när deras kunder kommer att flytta sina lån till en annan bank. Vad som avses med kundbortfall har behandlats samt hur det kan definieras, vad nyttan är med att förutsäga kundbortfall och vilka typer av variabler som kan användas för ändamålet. Vidare har grundläggande aspekter angående väglett lärande och evaluering av maskininlärningmodeller diskuterats. Modellen slumpträd har diskuterats mer ingående, eftersom den valdes för den empiriska studien. Därefter har bakgrundsdetaljerna, själva experimentet och resultaten presenterats. Med

ett AUC-poäng på 0,74 kan den slutliga modellen anses vara relativt bra. Detta ger stöd för att en maskininlärningsmodell skulle vara användbar för att förutsäga kundbortfall.

Bibliography

[1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] Wouter Buckinx and Dirk Van den Poel. Customer base analysis: partial defection of behaviourally loyal clients in a non-contractual FMCG retail setting. *European Journal of Operational Research*, 164(1):252–268, 2005.

[3] Jonathan Burez and Dirk Van den Poel. Handling class imbalance in customer churn prediction. *Expert Systems with Applications*, 36(3, Part 1):4626–4636, 2009.

[4] Jonathan Burez and Dirk Van den Poel. Crm at a pay-tv company: Using analytical models to reduce customer attrition by targeted marketing for subscription services. *Expert Syst. Appl.*, 32:277–288, 02 2007.

[5] Nitesh V. Chawla. Data Mining for Imbalanced Datasets: An Overview. In *Data Mining and Knowledge Discovery Handbook*, pages 853–867. Springer-Verlag, New York, 2005.

[6] Chao Chen, Andy Liaw, and Leo Breiman. Using random forest to learn imbalanced data. Technical Report 1999, University of California, Berkeley, 2004.

[7] Wikimedia Commons. File:k-fold cross validation en.jpg — wikimedia commons, the free media repository, 2018. [Online; accessed 27-August-2018].

[8] Kristof Coussement, Dries F Benoit, and Dirk Van den Poel. Improved marketing decision making in a customer churn prediction context using generalized additive models. *Expert Systems with Applications*, 37(3):2132–2143, 2010.

[9] Kristof Coussement and Dirk [Van den Poel]. Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques. *Expert Systems with Applications*, 34(1):313 – 327, 2008.

[10] Dirk Van den Poel and Bart Larivière. Customer attrition analysis for financial services using proportional hazard models. *European Journal of Operational Research*, 157(1):196–217, 2004.

[11] Bryan Gregory. Predicting customer churn: Extreme gradient boosting with temporal data, 2018.

[12] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[13] Benlan He, Yong Shi, Qian Wan, and Xi Zhao. Prediction of Customer Attrition of Commercial Banks based on SVM Model. *Procedia Computer Science*, 31:423–430, 2014.

[14] Bingquan Huang, Mohand Tahar Kechadi, and Brian Buckley. Customer Churn Prediction in Telecommunications. *Expert Syst. Appl.*, 39(1):1414–1425, jan 2012.

[15] Meelis Kull, Telmo M Silva Filho, and Peter Flach. Beyond sigmoids: How to obtain well-calibrated probabilities from binary classifiers with beta calibration. *Electronic Journal of Statistics*, 11:5052–5080, 2017.

[16] Viswanathan Kumar and John Andrew Petersen. *Statistical Methods in Customer Relationship Management*. John Wiley & Sons, Ltd, Chichester, UK, aug 2012.

[17] Charles X. Ling and Victor S. Sheng. Cost-sensitive learning. In *Encyclopedia of Machine Learning*, pages 231–235. Springer US, Boston, MA, 2004.

[18] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.], 2013.

[19] Teemu Mutanen, Sami Nousiainen, and Jussi Ahola. Customer Churn Prediction –a Case Study in Retail Banking. In *Proceedings of the 2010 Conference on Data Mining for Business Applications*, pages 77–83, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[20] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning - ICML '05*, pages 625–632, 2005.

[21] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. Scikit-learn: Machine Learning in {P}ython. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[22] Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson, and Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 159–166. IEEE, dec 2015.

[23] J Ross Quinlan. Induction of Decision Trees quinlan.pdf. *Machine Learning*, 1(1):81–106, 1986.

[24] DONALD B. RUBIN. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

[25] Philip Spanoudes and Thomson Nguyen. Deep learning in customer churn prediction: Unsupervised feature learning on abstract company independent feature vectors, 2017.

[26] Gholamreza Torkzadeh, Jerry Cha Jan Chang, and Gregory W. Hansen. Identifying issues in customer relationship management at Merck-Medco. *Decision Support Systems*, 42(2):1116–1130, nov 2006.

[27] Thanasis Vafeiadis, Konstantinos I. Diamantaras, George Sarigiannidis, and Konstantinos Ch. Chatzisavvas. A comparison of machine learning tech-

niques for customer churn prediction. *Simulation Modelling Practice and Theory*, 55:1–9, 2015.

[28] Yaya Xie, Xiu Li, E W T Ngai, and Weiyun Ying. Customer churn prediction using improved balanced random forests. *Expert Systems with Applications*, 36(3, Part 1):5445–5449, 2009.

[29] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. *Icml*, pages 1–8, 2001.