Aalto University

School of Science

Degree Programme in Mathematics and Operations Research

Ville Pohjalainen

Predicting service contract churn with decision tree models

Master's Thesis

Espoo, December 9, 2016

Supervisors:	Professor Ahti Salo
Advisor:	Olli Hänninen M.Sc. (Tech.)

Aalto University
School of Science
Degree Programme in Mathematics and Operations Research

ABSTRACT OF MASTER'S THESIS

Author:	Ville Pohjalainen		
Title: Predicting service contract churn with decision tree models			
Date:	December 9, 2016	**Pages:**	vii + 53
Major:	Systems and Operations Research	**Code:**	SCI3055
Supervisors:	Professor Ahti Salo		
Advisor:	Olli Hänninen M.Sc. (Tech.)		

Customer attrition is a central problem in sectors, whose revenue depend on customer relationships and it is more costly to acquire new customers than it is to retain current ones. Thus, targeted approaches are useful to reduce customer churn, given that the churning customers are correctly identified early enough. The objective of this thesis is to model the attrition of service contracts, which can be described as customers and to predict their risk of being cancelled.

Furthermore, causal reasons for predicting a contract for being risky and the predictive power of models are the key focus area. Understanding the reasons behind unsatisfactory customers is also important and can be used in the development of future business practices. However, this is only considered to a limited degree due to the proprietary and sensitive nature of the used data.

The modeling is done by applying Classification and regression trees, Random forests, Extremely randomized trees and XGBoost algorithms. It is found that XGBoost algorithm produces the best model in terms of accuracy, while we also gain an aggregate picture of the model's structure and related reasons for loosing service contracts. Rest of the models conform to these results and thus the models are capable of separating between risky and non-risky contracts.

Keywords: customer churn, decision trees, CRM, random forests, XG-Boost, machine learning

Language: English

Aalto-yliopisto
Perustieteiden korkeakoulu
Matematiikan ja operaatiotutkimuksen maisteriohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Ville Pohjalainen		
Työn nimi: Palvelusopimusten poistuman ennustaminen päätöspuumalleilla			
Päiväys:	9. joulukuuta 2016	**Sivumäärä:**	vii + 53
Pääaine:	Systeemi- ja operaatiotutkimus	**Koodi:**	SCI3055
Valvojat:	Professori Ahti Salo		
Ohjaaja:	Diplomi-insinööri Olli Hänninen		

Asiakaspoistuma on keskeinen ongelma aloilla, joiden liikevaihto määräytyy asiakassuhteista ja on kannattavampaa säilyttää jo olemassa olevat asiakassuhteet kuin hankkia uusia. Tämän takia kohdennetut kampanjat asiakaspoistuman vähentämiseksi ovat tärkeitä, jotta ne asiakkaat, jotka ovat aikeissa poistua voidaan tunnistaa tarpeeksi ajoissa. Työn tarkoituksena on mallintaa palvelusopimusten poistumaa ja ennustaa niiden peruuntumisen riskiä. Palvelusopimukset voidaan rinnastaa asiakassuhteisiin.

Työssä korostetaan mallien ennustekykyä ja niiden antamia syy-seuraussuhteita sopimusten peruuntumisille. Mallien antamaa tietoa voidaan tulevaisuudessa hyödyntää yritystoiminnan kehittämisessä. Mallien tulkittavuudessa ei kuitenkaan pureuduta syvälle yksityiskohtiin, koska käytetty data on luottamuksellista yksityisdataa yrityksen asiakastietojärjestelmistä.

Mallinnus tehdään käyttämällä Classification and regression tree, Random forests, Extremely randomized trees ja XGBoost algoritmeja. Jokainen malli kykenee erottelemaan peruuntumiselle riskialttiit sopimukset, minkä lisäksi niiden rakenteesta ja poistumaan vaikuttavista tekijöistä saadaan muodostettua kokonaiskuva. Käytetyistä päätöspuualgoritmeista XGBoost tuotti parhaan mallin, kun hyvyyttä tarkastellaan ennustetarkkuudella.

Asiasanat:	asiakaspoistuma, päätöspuut, CRM, random forest, XGBoost, koneoppiminen
Kieli:	Englanti

Acknowledgements

I am truly thankful to both, my supervisor Ahti Salo and instructor Olli Hänninen. Without their guidance and patience I do not believe this thesis would have seen the daylight. Also a big thank you for my good friend Vesa with helping to proofread this thesis and giving valuable feedback.

Espoo, December 9, 2016

Ville Pohjalainen

Abbreviations and Acronyms

CART	Classification and Regression Tree
CLV	Customer Lifetime Value
CRISP-DM	Cross Industry Standard Process for Data Mining
CRM	Customer Relations Management
CV	Cross-validation
ERT	Extremely Randomized Trees
FN	False Negative
FP	False Positive
FPR	False Positive Rate
LOOCV	Leave-one-out cross-validation
Lasso	Least absolute shrinkage and selection operator
LIME	Local Interpretable Model-Agnostic Explanations
MDG	Mean Decrease Gini
OOB	Out-of-bag error
RF	Random Forests
SKF	Stratified k-fold cross-validation
TN	True Negative
TP	True Positive
TPR	True Positive Rate
XGB	XGBoost

Contents

Chapter 1

Introduction

1.1 Background

Customer service related businesses rely on the volume of customers in order to generate revenue and income. To maximise revenues, it has been suggested that it may be more profitable to emphasize efforts in retaining the current customer base at the expense of hunting for new customer relationships [2] from the cost perspective. Thus, to prevent customers from churning, good customer relations management (CRM) practices must be in place and predictive modelling can be used as one of the tools, helping decision making on identifying the most likely customers to churn.

Churn modelling has been done across many fields including, but not limited to the financial [23, 33], online gambling [10] and telecom sectors [1, 19]. The modelling approaches vary greatly, and many algorithms have been used for the analysis, such as neural networks, decision trees, support vector machines, logistic regression and various others. Out of these algorithms, decision tree based algorithms like random forests have often performed the best [29] along other ensemble methods [10]. Relatively complete and comprehensive listings of applied approaches on various customer churn management studies are given in [17, 29, 32].

In addition to helping reduce customer retention, the ability to assign churn probabilities is helpful to customer lifetime value (CLV) analysis [17]. Unlike in churn prediction, the purpose of CLV is to directly maximize the value of a customer to a company during its life cycle.

From a broader point of view, a project format known as Cross Industry Standard Process for Data Mining (CRISP-DM) [30] is often applied to data mining projects. Kurgan and Musilek [22] refer to a poll in which 42% of the correspondents confirmed using CRIPS-DM in related projects. This also applies to churn prediction, because it is of paramount importance to understand what implications the implementation of targeting models has, what kind of data can be used, and how reliable this data is.

1.2 Objectives and structure

As mentioned in Section 1.1, by decreasing customer churn rate companies can gain increases in revenue more easily than by simply trying to attract new customers. Within the underlying framework, this thesis focuses on the development and analysis of suitable models for the prediction of service contract churn risks of a global industrial company. Albeit the setup differs slightly from the usual customer churn prediction, same predictive techniques commonly associated with the field of machine learning, can ne applied here. Although being an important and an interesting factor, CLV modelling is omitted in this study.

Further considerations are given to the evaluation scheme of the models and how they relate to business management. Ideally the model would predict the churn of a contract as early as possible, but in reality we have to set limits on the appropriate forecasting window. Business management translates to the amount of actions that can executed to retain contracts and to identify the optimal subset of risky contracts for retention efforts.

This thesis begins by explaining the underlying case in Section 2 and describing the data used. The prediction problem and data used, will be described in a sufficiently detailed manner, to give the reader a clear overall picture. Some properties of the data will not be covered in detail, because the data is confidential and sensitive CRM data.

In section 3 employed methods and algorithms are discussed in detail. This includes all the algorithms used for modeling, while also covering some possible ways to interpret the models. Section 4 details the practicalities of setting up the actual modeling and analysis.

Sections 5 and 6 present the most relevant results and splits analysis into two distinguishably separate cases. First, models are purely evaluated in the selected validation scheme, whereafter how the model is assessed in CRM through the lens of providing actionable insights. The latter is narrowed down to a very broad view, because the data is proprietary and thus prevents the presentation of in depth analysis. However, in the final section this considered to a degree and potential directions for future work are given.

Overall, this thesis aims to achieve the goals:

1: Identifying the most likely service contracts to churn and ranking them accordingly.

2: Providing high level explanations for the churn risks.

Chapter 2

Case formulation

2.1 Churn prediction

The fundamental goal is to build a predictive models and analyze their performance and respective properties in predicting service contract cancellations. Essentially, a predictive model tries to identify contracts which have a high probability of being canceled, in order to focus retention efforts on the appropriate subset of contracts early enough to reduce churn. Ideally, this is achieved by cyclically scoring the whole portfolio of contracts with the chosen model, while the scores or probabilities given by the model reflect the actual risk behind the contracts.

The modeling is done by developing decision tree models, which have presented good performance on similar tasks, as noted in Chapter 1 and have some favorable properties. A more in depth description about the models and model evaluation are given in sections 3 and 4. In the end, identifying contracts for retention efforts is not always as straightforward as selecting a certain portion of the riskiest contracts, because it is possible that some higher risk contracts cannot be salvaged. Hence, another important aspect is to lay rudimentary ground work on how to identify the optimal set of contracts for retention efforts.

Because prediction is conducted for service contracts, which can be owned by a single customer or a customer may hold a portfolio of contracts, there is a complex system of interlinked variables. For instance, given a customer with a portfolio of contracts, the model should distinguish between risks associated with a single contract versus overall customer dissatisfaction, which could lead to the cancellation of multiple contracts. The problem is alleviated by introducing variables on both at the contract and customer level. Some model paradigms also produce readily interpretable models, but not all, which is why data engineering and generating good understandable features is crucial. Even on a high level of only looking at the overall feature importance of a model may lead into new data insights, thus helping key personnel to apply targeted campaigns or other actions on the chosen contract owners.

An important aspect is to separate between customers with varying attributes and contracts with different scopes. A portion of the customer groups are already filtered out from the modeling process, so that the model can be aimed at specific groups, in which the model has most potential value. This is not to say it can not be used to predict the behavior of those not included in the modeling stage, however, the predictive accuracy may suffer. No explicit choices are made regarding the selected groups, because all of the targeted ones are predefined by business executives in advance.

Because there are multiple customer groups and the customer decides whether or not to continue a contract, further considerations regarding optimal subset of contracts to be targeted by retention efforts must be made. Analyzing contract cancellations rather than predicting customer churn, gets past the issue of having to clearly define what constitutes customer churn, because a lost contract can now be flagged straightforwardly as a churned one due to contractual terms.

2.2 Data overview and preprocessing

Data used in this thesis is real customer relations and contract level data, retrieved from several different sources, typically referred to as CRM systems.

Customer level data includes information on events such as customer complaints and satisfaction, whereas contract level detail focuses on the contract scope and on operational procedures during the contract's period of validity. Data is collected regularly, disregarding a few specific data sources, which are only available on an annual level. Time horizon of the whole dataset spans over five and a half years, though only a bit over four years worth of data is used. The latter is due to data upkeep reasons which renders the other tail of the data unreliable.

Because most data is uploaded manually to different databases and/or flat file systems, it is susceptible to human error and extra care must be taken to prune possible erroneous data entries. Thus, the first task is to clean the data, which involves consulting a multitude of different experts who are aware of the possible caveats present in the data, and can provide assistance in the case of abnormalities or help understand them. To ensure the integrity of the data sets, the following procedures are carried out in applicable situations:

1: Removing corrupted and/or duplicate data (data cleansing)

2: Aggregation (data cleansing and variable creation)

3: Outliers and extreme values filtering (variable creation)

4: Impute missing values (variable creation)

5: Model output check (validation stage)

The first rule is applied in the initial phase of data cleansing, because there are a number of aggregate rows, unnecessary fields or duplicate values. Also, anything that cannot be verified by an expert or is deemed untrustworthy, is removed or handled according to a predefined heuristic in this phase. A portion of the contracts were split up into separate instances although all carrying the same contract identifier, thus requiring a level of aggregation on their behalf. Nevertheless, most of the necessary aggregation is only carried out while creating the predictors.

For the third step, some data entries clearly exhibited misusage of reporting systems or other bugs with them, which required either totally omitting those entries or imputing them according to mean of the variables across all the contracts. Note that not all of the values for the variables, which in a traditional sense would be considered either outliers or extreme values were filtered, because they were deemed representative and informative of the actual distribution. Imputation of missing values follows given guidelines for all of the variables, i.e., they are set at a constant value, which reflects the best estimate for the field of interest.

Although much effort is put into data cleansing, one must still remain alert to data discrepancies. Because of this, the fifth and last step is required to ensure that the model output is reliable. In short, it is checked that the models' predictions are sensible and there are no outliers or mishandled data due to the ETL (*extract, transform and load*). Extra care is taken to ensure there are no data leaks, where the algorithms could learn from the future events. A data leak is a situation in which a contract is lost and some of its historical values are then retrospectively changed even for the period it was active.

In the process of variable derivation, a number of techniques are employed: calculating rolling means or sums from the previous 12 months, aggregating data and applying suitable transformations, comparing various proportions within the contract scope and creation of flag variables based on set rules. In order to make the features comparable across a range of small to big customers, they must be normalized. For the contract level predictors, only the contract scope is considered and used to scale everything according to it's size. Customer based attributes are scaled both on contract level and customer level. The scaled variable V_i^{scaled} is of the form:

$$V_i^{scaled} = \frac{V_i}{f(C_i S_i)}, \tag{2.1}$$

Table 2.1: All the features used in modelling, categorized by the information they contain and variable type, being either continuous or binary.

Operational		**Customer**		**Costs**	
Variable	**Type**	**Variable**	**Type**	**Variable**	**Type**
f1	Continuous	f7	Continuous	f8	Continuous
f4	Continuous	f11	Binary	f10	Continuous
f5	Continuous	f12	Continuous	f28	Continuous
f13	Continuous	f25	Continuous		
f23	Continuous	f30	Continuous		
f26	Continuous				

Portfolio		**Contractual**	
Variable	**Type**	**Variable**	**Type**
f2	Continuous	f15	Binary
f3	Continuous	f16	Binary
f6	Continuous	f17	Binary
f9	Continuous	f18	Binary
f14	Continuous	f19	Binary
f24	Continuous	f20	Binary
f27	Continuous	f21	Binary
f29	Continuous	f22	Binary

where V_i stands for the unscaled variable at month i, for example a rolling sum of customer complaints, C_i and S_i stand for customer size (how many contracts he owns) and contract size, respectively. The function $f(\cdot)$ takes care of the actual scaling, and most often maps the aforementioned variables only to correspond to the aggregate contract size of a customer. Evidently the scaled variable on the customer level converges with the contract level variable, if the customer holds only a single contract. All the generated features will be used in the modelling, where contributions and effects are analyzed separately from the actual model performances.

Table 2.1 gives as detailed an account as possible on the features used, with their types and information contained listed. One-hot encoding was used for the binary variables, due to the limitations of the applied algorithms to correctly identify categorical variables.

Overall, there are 30 different variables, of which 17 belong to contract level and 13 to customer level predictors; these can be further broken down into 5 distinct categories based on the information provided by each of the features[1]. Essentially the customer level predictors are formed by the *portfolio* and *customer* based ones, whereas the others are considered as contract level predictors. Out of all the features, 21 are continuous and 9 are binary or flag variables, denoting some categorical feature associated with the contract or customer. The amount of categorical variables used to describe the contracts had to be limited to avoid the curse of dimensionality, because each one of them has to be one-hot encoded, thus increasing the number of dimensions exponentially relative to the amount of categories. In total, for the time period covered by the data set used for modeling, there are 28,600 unique contracts and 9,500 unique customers, which equates to 570,000 rows of data considering the time series associated with each contract. A quick summary is given in Table 2.2.

Table 2.2: Data and variable summary.

Variables	Total	30
	Contract level	13
	Customer level	17
Unique ids	Contracts	28,600
	Customers	9,500

Table 2.3 gives a general overview of how the data frame is setup, where the terms "timestamp" and "date" are interchangeable, and are both measured in months. The data frame also conveys the idea of how multiple contracts may be owned by a single customer over varying time periods. The target value is always a binary value, indicating whether or not the contract was canceled at the given month. It takes a value of 1 for a cancellations and 0 for non-cancellation. The target date is always offset by a given amount of

[1]Contract level predictor means that the feature is directly linked to the contract, contrary to a customer level variable, where the customer key acts as a link between a contract and the customer level feature.

time units - should a cancellation occur - preventing the models from using future data in predicting churn. In this context, a month refers to a single row in the data frame, and each time series is cut after the contract has been canceled. However, offsetting cannot be done to an arbitrary extent, as it reduces the amount of available data entries. Chapter 4 gives a detailed account on how this is implemented and taken into account in the modeling stage.

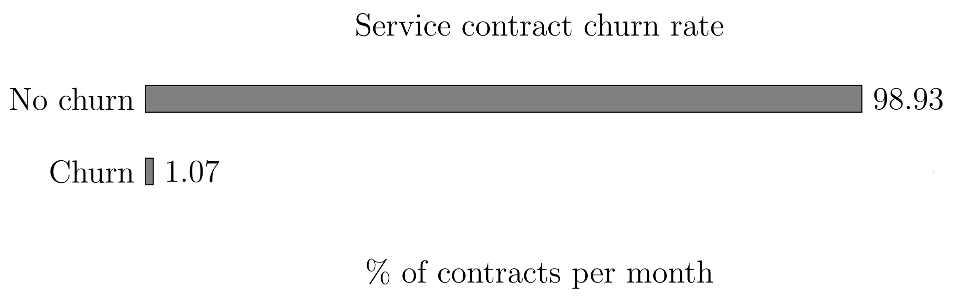

Figure 2.1: Distribution of cancelled and continued service contracts over the whole data set.

Defining the data frame in this way does give rise to a problem known as rare event prediction, because the monthly churn rates are largely offset by the amount of non-churning contracts per month. It directly affects the modelling paradigm and sets the ground for how the models should be evaluated. Further details are presented in the later chapters. Just to give a reference point of the actual distributions between churned and non-churned service contracts, Figure 2.1 shows the distribution of cancelled contracts versus retained ones per month.

Table 2.3: Data frame visualization. ID 1 refers to the contract and ID 2 to the customer.

Data frame						
ID 1	**ID 2**	**Date**	**Variable** X	**Variable** Y	...	**Target**
1	A	i	X_i	Y_i	...	0
		$i+1$	X_{i+1}	Y_{i+1}	...	0
		$i+2$	X_{i+2}	Y_{i+2}	...	0
		$i+3$	X_{i+3}	Y_{i+3}	...	0
2	A	$i+2$	X_{i+2}	Y_{i+2}	...	0
		$i+3$	X_{i+3}	Y_{i+3}	...	1
3	B	i	X_i	Y_i	...	0
		$i+1$	X_{i+1}	Y_{i+1}	...	0
		$i+2$	X_{i+2}	Y_{i+2}	...	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
ID N	ID M	$i-1$	X_{i-1}	Y_{i-1}	...	0
		i	X_i	Y_i	...	0
		$i+1$	X_{i+1}	Y_{i+1}	...	0
		$i+2$	X_{i+2}	Y_{i+2}	...	0
		$i+3$	X_{i+3}	Y_{i+3}	...	0

Chapter 3

Methodological approaches

3.1 Decision tree learning

Decision trees are generally used under a supervised setting (supervised learning) and can be trained both, either as regression trees or classification trees. Because they fall into the category of supervised learning algorithms, they require known output values or labels to be trained on, depending on which of the aforementioned learning situations they are exposed to.

Although there are a many methods for constructing decision trees, they all share a few core principles, namely, their structure and the nature of how they are trained. Training is based on recursively splitting the data into smaller and smaller subsets, which can be depicted as child nodes emerging from their respective parent nodes. Each split is defined by a set measure or rule and the tree is grown until a certain criterion is met or it cannot be grown any further. The criterion for stopping tree growth are in place to avoid it from overfitting to the training data, and could be as simple as setting a maximum tree depth, at a certain pre-defined level.

A common practice is to use binary splits in the tree growing phase, which is the case with all the algorithms used in this thesis, namely Classification and regression trees (CART), Random forests (RF), Extremely randomized trees (ERT) and XGBoost (XGB). For non-binary splits, other methods such as Chi-Square Automatic Interaction Detector [20] (CHAID) can be applied. Figures (3.1) and (3.2) give an example of how a decision tree might split the underlying space into separate rectangular areas. The fact that the decision trees are limited to splitting the space into rectangular areas can be a hindrance, especially in the case when the true decision boundary does not follow such a shape. This is partly remedied by other techniques, such as Random Rotational Ensembles (RRE) [4], which acts similarly to e.g., Random forests, except that it performs a random rotation on the data set prior to growing any tree in the ensemble. For a more extensive discussion on RREs, please refer to [4].

Pros	**Cons**
-Computationally efficient [26]	-No guarantees of global optimum
-Interpretability [18]	-Additive structure hard to capture [18]
-Non-parametric method	-Loss of interpretability with tree ensembles
-Can capture higher order interactions amongst variables [18]	

Table 3.1: Decision tree learning advantages and disadvantages.

Some of the main benefits and drawbacks of decision tree algorithms are listed in Table 3.1. A notable advantage of decision trees over regular discriminators such as logistic regression, are their ability to split the hyperspace into multiple brackets, instead of simply cutting it in half, portrayed in Figure 3.2.

The subsequent sections describe the splitting process in more detail in the following notational form. A single observation is described by the pair $(\mathbf{x_i}, y_i)$, with a total of $i = 1, 2, ..., N$ samples. A feature vector is represented by $\mathbf{x_i} = [x_{i1}\, x_{i2} \ldots x_{iM}]$, which consists of M different predictor variables. All features are either real or binary valued, assuming there are no missing data elements, that is, $x_{ij} \in \mathbb{R}$ and $x_{ik} \in \{0, 1\}\, \forall k \neq i$, for each $i = 1, 2, ..., N$, $j = 1, 2, ..., M$. The response variable takes on either continuous or categorical

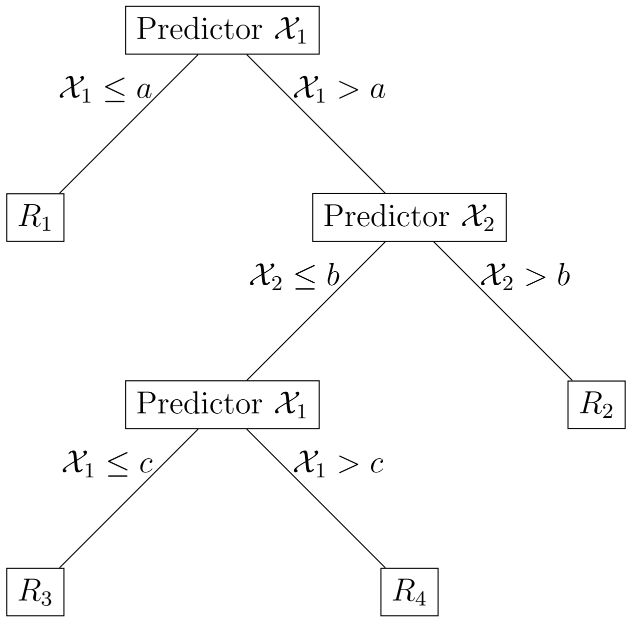

Figure 3.1: Example of a decision tree, where two variables (X_1 and X_2) recursively split the underlying space into four separate regions R_i.

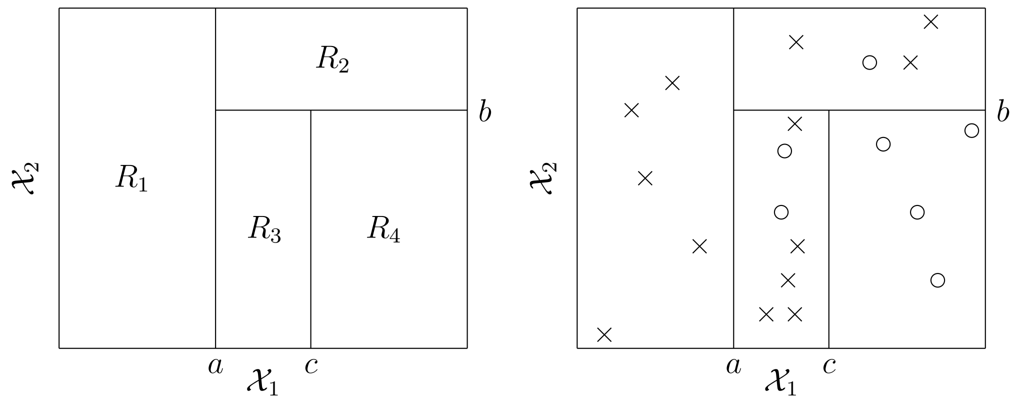

Figure 3.2: The image on the left depicts variable space of $\mathcal{X}_1$ and $\mathcal{X}_2$ split according to the decision tree in figure 3.1. On the right, "x" and "o" mark the different observations, e.g., churning and non-churning customers, falling into separate regions of the input space, as dictated by the same decision tree.

values, depending on whether one deals with a regression or a classification model - ordinal targets versus categorical ones. Finally, per tree, the data is split into H different regions $R_1, R_2, ..., R_H$.

3.1.1 Classification and regression trees

Originally introduced in 1984 by Breiman et al.[8], CART is a decision tree learning algorithm, capable of performing both, regression and classification. It also serves either directly as the backbone of some the other introduced algorithms [9] or have elements of it heavily incorporated in them [7].

On a general level, a CART is grown, irrespective whether or not it is used for regression or classification, by a greedy algorithm. Thus, the tree finding locally optimal splits at each node, as long as the tree is grown. However, there are no guarantees of a globally optimal solution, and thus the model might get stuck in a local optimum. Due to computational resources, finding the globally optimal solution is often infeasible, with a large amount of variables and data.

To describe how the algorithm functions, let t denote the current parent node. Now arriving at a node t, and splitting it by the optimal variable j and split point x^* separates the underlying space into two half-planes, as indicated by equation (3.1)

$$R_h^{(t)}(j, x^*) = \{\mathcal{X}|\mathcal{X}_j \leq x^*\} \text{ and } R_{h+1}^{(t)}(j, x^*) = \{\mathcal{X}|\mathcal{X}_j > x^*\}. \tag{3.1}$$

Note that $\mathcal{X} \subseteq \mathbf{X}$ is assumed to be all the data arriving at the selected node, and should it be the first split, then the equality holds. Furthermore, depending on whether a classification or a regression tree is grown, a corresponding measure is then used for finding the locally optimal splits. Considering a classification tree, a common choice for a measure is to use the Gini impurity, defined as

$$Q_h = \sum_{k=1}^{K} p_{hk}(1 - p_{hk}), \tag{3.2}$$

where $p_{hk} = \frac{1}{N_h} \sum_{\mathbf{x_i} \in R_h} \mathbb{1}(y_i = k)$ is an indicator function for proportion of the given class k in the group h. For binary target variables this can be further reduced into the form

$$p_{hk} = 1 - \sum_{k=1}^{K} p_{hk}^2 = 2p_h(1 - p_h).$$

At each node, observations are classified as belonging to the majority class $k(h) = \underset{k}{\operatorname{argmax}}\, p_{hk}$. In order to find the actual optimal variable j and split point x^*, classification tasks seek to maximize the decrease in impurity when splitting the parent node, according to chosen impurity measure $I(\cdot)$

$$\Delta I = I(t) - p_L I(t_L) - p_R I(t_R), \tag{3.3}$$

where t is a reference to the parent node, t_L and t_R to the respective child nodes. The terms p_L and p_R reflect the proportions of data arriving to either of the child nodes from the parent node, namely $p = N_{child}/N_{parent}$.

It is easy to expand the given formulation to cover regression trees simply by switching the used impurity measure or use other measures for classification, such as information gain. For the derivation of the regression algorithm and further information on other possible impurity measures, a detailed account is given in [18].

Although CART is a non-parametric method, often implementations of the algorithm contain some parameters governing the tree growing phase, such as those affecting stopping criterion, sample weights and pruning.

3.1.2 Random forests and Extremely randomized trees

Random Forests and Extremely Randomized Trees - henceforth referred to as either RF or ERT - are fundamentally very similar techniques with slight differences, former being developed by Leo Breiman [7] and the second by Pierre Geurts et al. [15]. In principle, both are ensemble methods which combine the predictions from many decision trees and average over their predictions in a set fashion. Formally, let T be a collection of decision trees grown either by using the RF or ERT algorithm and $T_i \subseteq T$ a single decision tree grown on the ith iteration of said algorithm. Then, given an instance $\mathbf{x}$ and number of models N, the corresponding prediction is

$$\hat{y} = \frac{1}{N} \sum_{i=1}^{N} T_i(\mathbf{x}). \tag{3.4}$$

For classification trees, another option would be to let the single trees cast their votes on a sample and then assign it the class with the majority of the votes. However, averaging over the predicted class probabilities has the advantages of reducing the variance in the expected generalization error and retains the possibility of interpreting the prediction as a probability [24].

Although averaging models has been found to generally increase predictive accuracy, it is worth noting if many predictions of the models are correlated, the effect of model averaging diminishes. This can been seen from equation (3.5), which is the variance of the expected generalization error of a point $\mathbf{x}$ with $\mathcal{M}$ different models, derived from the bias and variance decomposition

$$Var(\mathbf{x}) = \rho(\mathbf{x})\sigma^2(\mathbf{x}) + \frac{1-\rho(\mathbf{x})}{\mathcal{M}}\sigma^2(\mathbf{x}). \tag{3.5}$$

Here $\rho(\mathbf{x})$ and $\sigma^2(\mathbf{x})$ refer to the Pearson correlation coefficient and total variance of all the single models, respectively. The full derivation and discussion of the bias variance decomposition is available in [24], which covers RFs in detail.

Growing a RF or an ERT follows largely the principles of learning a CART model. Even though both algorithms comprise many trees, they also introduce slight modifications in the process of learning a single tree in the ensemble. The biggest differences are when forming splits and they can be summarized as follows:

> **RF**: For each split, choose a random subset v of random variables without replacement from the set $\mathcal{V}$ containing all the variables and next choose the optimal split according to the selected impurity measure. A common choice for the amount of variables to choose, is to limit the cardinality of the random subset $|v| \leq \sqrt{|\mathcal{V}|}$. RF generally uses bootstrapping in the process of growing multiple trees.
>
> **ERT**: Choose possible splitting variables similarly as with RF, however, instead of finding the optimal split point for each of the chosen variables, it is chosen uniformly at random from an interval, defined by the minimum and maximum of the variable. Afterwards each of the splits are scored and one with the best score is chosen as the splitting variable. ERT does not make use of bootstrapping.

The bootstrapping which RF makes use of can be described as follows: given a data set $\mathcal{X}$ used for training an algorithm, a random subset $\mathcal{D}_i \subseteq \mathcal{X}$, called the bootstrap sample, is drawn uniformly and by replacement at each iteration of the algorithm where a new tree is grown and added into the ensemble. Learning an ensemble model this way is referred to either as bootsrap aggregating, or simply bagging [5]. By using bootstrapping, one can estimate the performance of the classifier by analyzing out-of-bag error (OOB), which is a measure for prediction error. It is formed by evaluating the predictive performance of all the models - herein the grown decision trees of the ensemble - by letting them score all those observations they were not trained on [6].

Another common factor between RF and ERT is that neither of the said algorithms use pruning, thus each tree is grown until stopping criteria are met and are then left intact, fully grown. However, this may lead to overfitting of single trees, which is partially remedied in the RF algorithm by applying bootstrapping, which helps reduce it by subjecting each new tree in the ensemble to slightly different data set.

3.1.3 Gradient boosted trees

Gradient boosted trees were first implemented in [13] and [14], with some modifications, namely the stochastic variation of the algorithm. In principle, the algorithm works by sequentially constructing decision trees and fitting them against so called "pseudo"-residuals, which in turn can be interpreted as gradients - hence the name. CART is mainly used for constructing the base learners, with the added possibility of bootstrapping in order to induce a form of regularization. The recently introduced XGBoost, a modification of the gradient boosting algorithm, which in addition to exploiting parallelized computing, also applies a separate form of regularization to the procedure. Although the gradient descent algorithm in itself has to proceed sequentially, growing a single tree can be parallelized - a key factor in the XGBoost algorithm [9] - and the regularizing term may be simply added to the loss function under optimization, punishing for model complexity. Regularization is essential, as it reduces overfitting.

Formally as defined in [9] and following established notational conventions, with the exception that $y_i \in \mathbb{R}$, a tree ensemble gives an output of

$$\hat{y}_i = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \tag{3.6}$$

where f_k belongs to the space of CART trees, $\mathcal{F}$. In order to calculate the gradients, one has to define an objective function, which in XGBoost is of the form

$$\mathcal{L}_t(y_i, \hat{y}_i) = \sum_{i=1} l(y_i, \hat{y}_{i,t}) + \sum_{k} \Omega(f_k) \tag{3.7}$$

$$\Omega(f) = \gamma L + \frac{1}{2}\lambda ||w||^2, \tag{3.8}$$

where L is the number of leaves, w the score or leaf weight and both, γ and λ are regularization constants with t being a reference to the number of trees fitted. Because gradient boosting proceeds sequentially fitting more trees against the gradients, $\hat{y}_{i,t}$ may at the i-th iteration be approximated by another function f_t

$$\mathcal{L}_t(y_i, \hat{y}_i) = \sum_{i=1}^{N} l\Big(y_i, \hat{y}_{i,(t-1)} + f_t(\mathbf{x}_i)\Big) + \Omega(f_t). \tag{3.9}$$

Here $l(\cdot)$ denotes the loss function which can be adjusted according to the learning objectives. For instance, a common choice in classification is to set it as the logarithmic loss discussed in Section 3.3.2. The rest of the derivation for XGB progresses by taking a second order approximation of the loss function l, and in turn deriving the optimal weights w and the value of the objective function, given a tree structure.

3.2 K-fold cross-validation

The basis of cross-validation lies in partitioning the data set into multiple training and testing blocks, as illustrated by the Figure (3.3). The algorithm is fitted sequentially on each training set and then tested against a hold out set overall ten times, which is referred to as 10-fold cross validation. Other cross-validation methods do exist, such as leave-one-out cross-validation (LOOCV) [21], which equals K-fold cross-validation, if the number of folds equals the size of the data set. One use case for cross-validation arises should data be scarce, then it may prove useful to evaluate models under a cross-validation setup, instead of splitting the data into single separate training and testing sets. Another application is the tuning of model parameters, in which case the results from all the cross-validation rounds are averaged, and the mean is used as an estimate for the algorithm's predictive capabilities with the given parameters.

A drawback in K-fold cross-validation is that its variance is hard to estimate, because the test errors are dependent as they are all sampled from the same data set. Although being an unbiased estimator for the expected prediction error, it becomes biased when trying to estimate it's variance. Thus, it is difficult to form confidence intervals for the K-fold cross-validation schemes, and may hinder the model selection procedure [3].

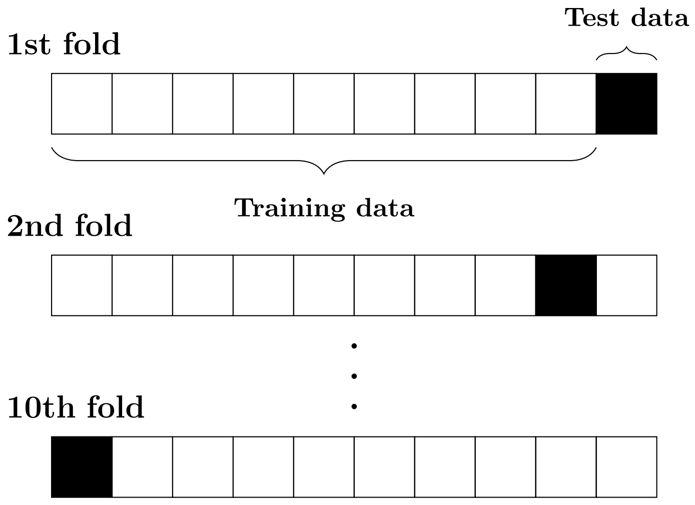

Figure 3.3: Example of a 10-fold cross-validation scheme, where the white boxes indicate the training data used for modeling in each fold, and the black box corresponds to the test set.

The stratified k-fold cross-validation (SKF) is an extension to k-fold cross-validation so that class balances are kept similar across all the folds [21]. Kohavi [21] suggested that using a stratified cross validation scheme may yield less bias and variance in estimating a model parameter.

3.3 Metrics for model performance

Model performance is measured across a variety of metrics, each summarizing different aspects of model functionality and goodness of fit. In total, three measures and their applicability are covered.

3.3.1 F-score

F-score or equivalently F-measure can be used to assess binary classification results. The more common version is defined as the harmonic mean of precision and recall. Both are derived from the true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), from which the F_{score} follows:

$$PPV = \frac{TP}{TP + FP} \tag{3.10}$$

$$TPR = \frac{TP}{TP + FN} \tag{3.11}$$

$$FPR = \frac{FP}{FP + TN} \tag{3.12}$$

$$FOR = \frac{FN}{TP + FN} \tag{3.13}$$

$$F_{score} = 2\frac{PPV \cdot TPR}{PPV + TPR}. \tag{3.14}$$

Here PPV stand for positive predictive value or precision, TPR for true positive rate or recall, FPR for false positive rate or fall-out and FOR is an abbreviation for false omission rate.

		Predicted value	
		Churn	No churn
True value	Churn	TP	FN
	No churn	FP	TN

Figure 3.4: Confusion matrix for a binary classification task.

3.3.2 Logarithmic loss

The logarithmic loss may be used in classification problems and is primarily measure of the classification accuracy, which aims also to express how well the model performs, given its confidence on each of its predictions. Thus, given the true classes y_i, with N samples, K classes and the respective class probabilities p_{ik}, logarithmic loss is defined as

$$\mathcal{L}_{logloss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\log(p_{ik}). \tag{3.15}$$

3.3.3 Receiver operating characteristic and lift curve

Receiver operating characteristic (ROC) captures a model's sensitivity to the ratio of true positives versus false negatives. It is a suitable and often used measure for binary classification problems - an extension for multi-class case does exist [12] - because it readily describes the TPR as a function of FPR. Naturally, it is preferable to have a high ratio of TPR versus FPR, which would show up as a steep rising curve on the ROC chart. For a visual benchmark, one may compare the actual ROC curve from a model to a line spanning between the coordinates $(0, 0)$ to $(1, 1)$ in the ROC space. The line can be interpreted as random guessing or a constant prediction, representing an equal proportion of true and false positives as predicted by the model.

Another measure relating to ROC is the area under ROC curve (AUC), which is a scalar metric calculated as defined by its name. This is a more common way of comparing different classification models with each other, as they are now summarized by a single statistic. However, AUC does come with some caveats, such as being insensitive to the prediction probabilities and they also summarize model performance over regions that are of no practical interest or are operationally infeasible [11].

Lift curve assesses the models performance conditional on the underlying distribution of classes and which class is being predicted. The curve itself is a ratio of all the positive instances and all the cumulated data up to the point, scaled by the proportion of the positive instances. Furthermore, it is assumed that data is arranged in a descending order according to the probability of each sample belonging to the reference class. Thus, lift is a measure of how well the model finds associations between the data and the predicted class, with higher lift values indicating better model performance. A base rate of 1 refers to the point where the model cannot distinguish between the two classes and the situation resembles random guessing.

Given a model M, which assigns each observation $\mathbf{x} \in X$ a probability $p_M(\mathbf{x})$ of belonging to the positive class $y = 1$, a p^* cumulative lift can be defined as follows

$$\text{Lift}(X, p_M|p^*) = \frac{\big|\{\mathbf{x} \in X|\, y(\mathbf{x}) = 1,\, p_M(\mathbf{x}) \geq p^*\}\big|}{\big|\{\mathbf{x} \in X|\, p_M(\mathbf{x}) \geq p^*\}\big|\big|\{\mathbf{x} \in X|\, y(\mathbf{x}) = 1\}\big|}, \tag{3.16}$$

where $y(\mathbf{x})$ represents the true class of observation $\mathbf{x}$.

An alternative and an equal interpretation of the lift curve is to convert it to the response rate of the model. The only difference is that the base rate is now exchanged with the class distribution of the class under consideration. It is also entirely possible for the model to perform worse than the base rate, however, this does not imply that under those scenarios it is better to choose the other class as the predicted one. An example of of both, ROC and lift curves are illustrated in Figure 3.5.

3.4 Model interpretability

Model interpretability may be approached in a number of ways. Two distinct paradigms are considered, the first, which aims at distinguishing explaining factors on an aggregate level, and secondly, trying to explain directly the amount weight each feature contributes to the made prediction.

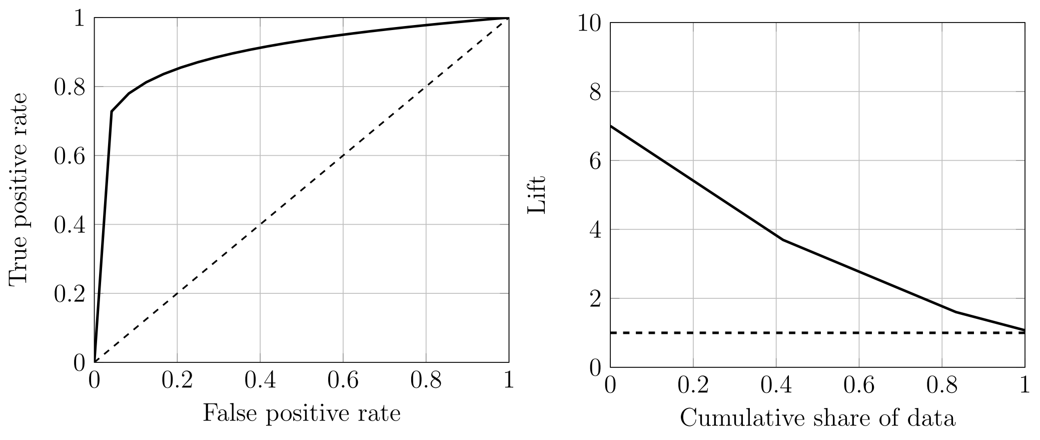

Figure 3.5: The left side image corresponds to a ROC chart and the one on the right to a lift curve. Note that the title of the x-axis is interchangeable on both images, on the premise that the samples are ranked in a descending fashion based upon their probability of belonging to the same class.

3.4.1 Eliciting feature importances from trees

One way of understanding feature importances is to measure the Mean Decrease of Gini (MDG), with the single tree variant of it introduced in [8]. The single tree case can be defined as

$$Imp(\mathcal{X}_j) = \sum_{t \in \phi} \Delta I(\hat{x}^*_{j,t}, t), \tag{3.17}$$

where $\hat{x}^*_{j,t}$ is the optimal split on $\mathcal{X}_j$ at the node t, which belongs to the set of nodes, denoted by ϕ. Ideally, $\hat{x}^*_{j,t}$ is as close as possible to the original splitting point x^*_t in terms of decreasing the select impurity measure. The impurity $\Delta I(\cdot)$ is the same as in equation (3.3). If the variable is the actual splitting variable at node t, it is referred to as a primary splitter, or conversely, a surrogate splitter in the case it is not. However, some algorithms only consider the primary splits, when calculating the feature importances, such as the decision tree implementations from [27].

For multiple trees, MDG is a weighted sum of the decrease in the impurity for each variable and all the nodes it splits. Weighting is done by accounting for the probability for an observation of reaching a given node, which in turn is approximated by the proportion of samples split by the node. As expressed in both [24] and [25]:

$$Imp(\mathcal{X}_j) = \frac{1}{N_T} \sum_T \sum_{t \in \phi_T} p(t) \Delta I(\hat{x}^*_{t,j}, t), \tag{3.18}$$

where $p(t) = N_t / N_{total}$ is the proportion of samples reaching the node t, belonging to the set of splits ϕ_T in tree T. Only the primary splits are considered for equation (3.18). Thus, if variable j is not the splitter at t, $\mathcal{X}_j$ receives a score of 0 from that split.

It is possible to use feature importance measures such as MDG in the initial model building phase to select an appropriate set of futures. However, this is problematic in the presence of highly correlated variables, since the interpretation of these importance measures becomes obscure [16].

3.4.2 Local interpretable model-agnostic explanations

As the tile of this section suggests, LIMEs goal is to explain the predictions of any model, regardless of the algorithm or method used for modeling [28]. This is done by exploring the model and constructing linear approximations near the observation of interest.

Adopting the notation of the original author, let $\mathbf{x} \in \mathbb{R}^M$ be the original representation of an instance to be explained. Furthermore, let $\mathbf{x}' \in \{0,1\}^{M'}$ be a binary vector indicating the interpretable version of the instance and $g \in \mathcal{G}$ the model explaining it, belonging to the class $\mathcal{G}$ of all possible interpretable models, which has the same domain of $\{0,1\}^{M'}$ as does $\mathbf{x}$. It only remains to define a few more functions for the algorithm. Firstly, the classification function $f(\mathbf{x})$, which acts over $\mathbb{R}^M \to \mathbb{R}$, indicating the probability of an instance belonging to a certain class and $\Pi_{\mathbf{x}}(\mathbf{z})$ - a proximity measure between a sample instance $\mathbf{z}$ and an observation $\mathbf{x}$. The target function to be minimized, which according to [28] ensures both local fidelity and model interpretability, is defined as $\mathcal{L}(f, g, \Pi_{\mathbf{x}})$ and obtained from

$$\xi(\mathbf{x}) = \underset{g \in \mathcal{G}}{\arg\min}\, \mathcal{L}(f, g, \Pi_{\mathbf{x}}) + \Omega(g), \tag{3.19}$$

where the term $\Omega(g)$ is a measure of model complexity, thus acting as regularizing component favoring simpler models. Ideally this leads to models, which are readily interpretable by humans and consequently produce pertinent insights into the data.

Technically, the minimization problem set by equation (3.19), is solved by sampling uniformly around $\mathbf{x}$ and each of the samples are weighted according to $\Pi_{\mathbf{x}}(\mathbf{z}) = exp(-D(\mathbf{x}, \mathbf{z})^2/\sigma^2)$. Thus samples closer to $\mathbf{x}$ receive more weight than those that are further away. Now, let $g \in \mathcal{G}$ be defined as $g(\mathbf{z}') = \mathbf{w}_g^T \mathbf{z}'$, where $\mathbf{z}' \in \{0, 1\}^{d'}$ and $\mathbf{w}_g$ the weights. Finally, let the loss function be of the form

$$\mathcal{L}(f, g, \Pi_{\mathbf{x}}) = \sum_{\mathbf{z}, \mathbf{z}' \in \mathcal{Z}} \Pi_{\mathbf{x}}(\mathbf{z}) \big(f(\mathbf{z}) - g(\mathbf{z}')\big)^2. \tag{3.20}$$

Defining these terms allows LIME to proceed and optimize the target function of (3.19). First a number of samples are generated near the point of interest and then K important features are chosen by applying the least absolute shrinkage and selection operator method (Lasso) [31]. Once these steps are taken, the weights $\mathbf{w}_g$ are fitted through ordinary least squares.

Chapter 4

Implementation

4.1 Model training

A number of methodologies exist for training a model and subsequently assessing its properties of generalization, that is, what to expect from it's performance on yet unseen data. Some often applied practices include simply splitting the data into separate training, testing and validation sets, whereas other validation designs, such as K-fold cross-validation, rely on splitting the data into multiple instances for training and testing in an iterative fashion.

Here, the chosen procedure follows the principles of splitting the data initially into training and testing sets. Furthermore, the algorithms are initially be trained by applying Stratified K-fold cross-validation scheme using the training data only, and once overall satisfactory results are obtained regarding any some of the hyperparameters, the whole training data will be used to train a complete model. Finally, the hold out set (testing set) will be used for validation. Although all of the algorithms leave ample of space for potential parameter optimization, only the tree depth will be explored in depth. Each algorithm is set to contain a minimum of 20 records per leaf for a split to be considered and the classes are weighed according to their current distribution in the given training phase data set. RF and ERT both contain 500 trees, whereas XGB is set to do 400 iterations, which directly translates into the amount of trees constructed.

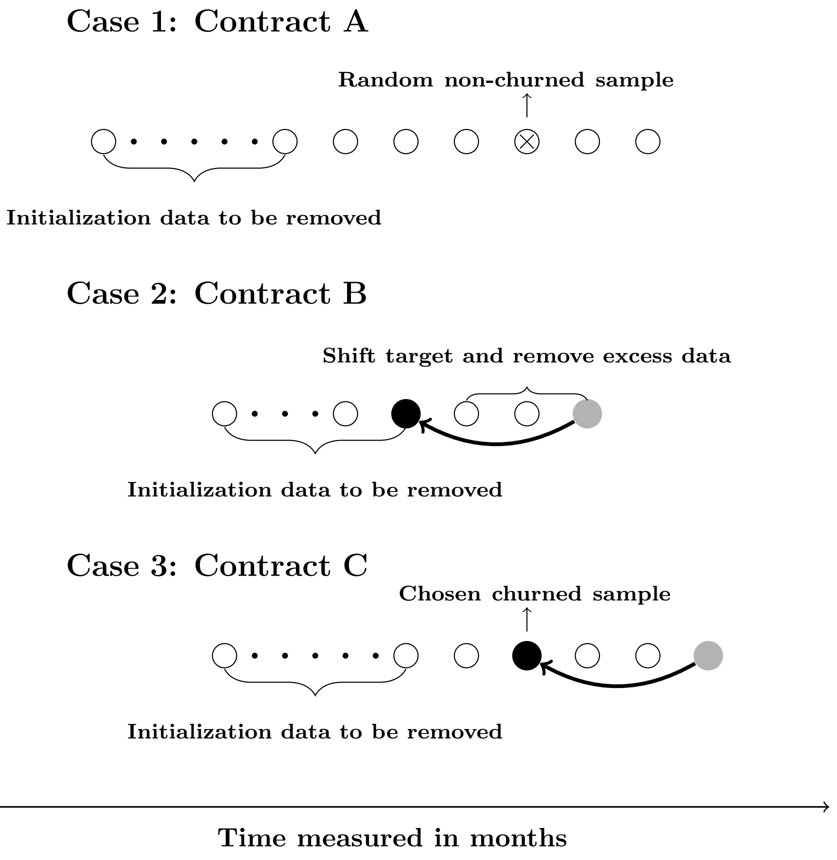
Case 1: Contract A
Random non-churned sample
Initialization data to be removed
Case 2: Contract B
Shift target and remove excess data
Initialization data to be removed
Case 3: Contract C
Chosen churned sample
Initialization data to be removed
Time measured in months

Figure 4.1: Three cases illustrating all the possible ways of sampling data for modeling. In the first case for *Contract A*, one sample is added to the used data set, whilst the data for *Contract B* ends up being discarded totally - original churn date marked with gray and this will always be either shifted or discarded. *Contract C* illustrates how a churned target variable is shifted backwards in time and then used as a churned sample. Note that churned samples are explicitly chosen, and randomly assigned only when partitioning the full data set into training and testing sets.

Figure 4.1 shows how the data is transformed into a balanced data set of churned and non-churned contracts for the training set. It should be noted that the first three months of each contract are used to initialize many of the variables, are therefore screened from the whole data set, including the test set[1]. Should the data history span a shorter time period, the contract is simply dropped from the data set. To balance the training data set in terms of non-churned and churned contracts, each non-churned contract's time series is trimmed by randomly sampling a single data point from it and adding it to the new, balanced training data set. For the churned contracts, the target label is shifted three months backwards in time and only these events are added to the data set. Shifting the target label indicating a churn event also prevents potential data leaks.

The split into training and testing sets is conducted randomly by using the customer as an identifier and is carried out so that 70% of the data is preserved for training and 30% testing. The latter set contains the full time series for each of the service contracts, unlike the training set. The reason for using a customer to form the splits is to avoid situations, in which a customer could have multiple contracts, and thus potentially leak information into the test set.

All the programming was done with Python 3.5 with the help of the libraries presented in [9, 27, 28].

4.2 Model evaluation

4.2.1 Model accuracy

Unlike in the training phase, where much emphasis is on single metrics measuring overall model performance, the test set focuses more on the model lift and AUC (or the ROC curve). Both of these describe how well the models rank the service contracts from the riskiest to those that are the least likely to be cancelled, with the former metric being agnostic to whether or not the model is actually predicting churn or not. Unlike in the training phase, no truncated time series will be used, instead, each month for each contract will be considered as separate instances when evaluating the model. However, a

[1]Although not explicitly depicted in Figure 4.1, it is also possible for a contract to be totally left out, if it has less than a 3 month history.

three month grace period is given to each of the churn predictions. Thus, if a model predicts a contract will churn and it indeed does so within the following grace period, this is considered as a correct prediction, a true positive in this case. For the test set, the initial target variable is shifted by one, two and three month backwards in time from the date it is marked as lost. This is shown in Figure 4.2 which illustrates how the grace period is formed and added to the test set.

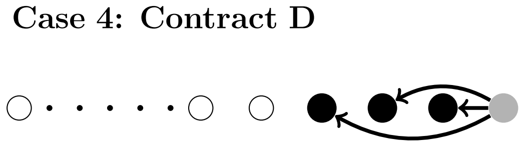

Figure 4.2: *Contract D* shows how the actual churn date is shifted - marked as gray and removed from the test set - to encompass the previous three months, if possible.

Judging by Figure 4.2, it is possible that a model predicts for a contract to churn on time stamp i (denote this with $\hat{y}_i = 1$), but not on the following month ($\hat{y}_{i+1} = 0$), when indeed it does so: $y_i = y_{i+1} = y_{i+2} = 1$). In this particular case, $\hat{y}_i = 1$ would be a true positive and $\hat{y}_{i+1} = 0$ a false positive. This example can be easily extended to cover other variations of the situation and is similar to a sliding window methodology, when it comes to prediction evaluation.

Overall, the predictive models aim to capture the churn event ideally three months ahead, which serves as a motivation for using the described training and validation setups, respectively. It should be noted, however, that inevitably an influencing factor is the limited historical data on detailed contractual terms, such as exact cancellation procedures, which would allow the fine tuning of the model evaluation scheme or creation of additional variables to capture these aspects. Now the contracts are scored continuously and evaluated as described, assuming each one of them may be cancelled at any point in time.

4.2.2 Importance of features

A number of different metrics were chosen to assess the model performance. This is due to the fact that each of the measures captures unique characteristics of model performance and, judging it from a multiple different angles, gives a better overall understanding of its nature. This relates also to the importance of trying to grasp what the model is actually predicting and why. The purpose is not just to train a model which seemingly performs well under conventional metrics, but also to validate the sensibility of the results and potentially to gain new business understanding. Thus actually being able to explain individual predictions may prove very valuable.

Model structure, as defined by the features, are purely examined in the framework in section 3. In practice, this means assessing the feature importances, derived from the models, and comparing their relevances amongst the different models. Any individual features will not be discussed, although it would be possible in the context of LIME, because it is not possible to go into detail about what each of the features are precisely about. Thus it makes in depth analysis on the feature level obsolete, however, LIME is used to produce aggregate level descriptions of variable influences, based on the categories presented in the Table 2.1.

Chapter 5

Computational results and model evaluation

This section presents the main numerical results of this thesis. We first present the overall results from the training and testing phases on how the models fared against the chosen metrics. We then analyze and discuss them in the remainder of the chapter.

5.1 Resulting model performance

Table 5.1 shows the accuracy of all algorithms, derived by taking the average accuracies over all stratified k-folds during the training scheme, with varying tree depths. The displayed metrics are F-score, AUC and logarithmic loss. Out of the chosen algorithms, ERT seems the only one that still benefits from added maximum tree depth after the range of 5-8 maximum splits with regard to all metrics. It is also evident that the logarithmic loss of CART increases as a function of the tree depth. The other algorithms seem to have a decreasing logarithmic loss as function of tree depth. However, the other two metrics are rather stable once the maximum number of allowed splits exceeds 6. Hence, for the full training the tree depth were set at 6 for CART, 7 for both RF and XGB, and finally the ERT algorithm was trained by using

Table 5.1: Results for tenfold stratified training with varying maximum tree depth, where each value represents the average of the folds.

	CART			**RF**		
Depth	**F-Score**	**Log-loss**	**AUC**	**F-Score**	**Log-loss**	**AUC**
3	0.380	0.656	0.624	0.390	0.644	0.671
4	0.356	0.642	0.648	0.389	0.624	0.692
5	0.344	0.653	0.635	0.410	0.601	0.706
6	0.408	0.680	0.676	0.425	0.579	0.724
7	0.370	0.753	0.663	0.427	0.558	0.731
8	0.371	0.826	0.674	0.416	0.540	0.738

	ERT			**XGB**		
Depth	**F-Score**	**Log-loss**	**AUC**	**F-Score**	**Log-loss**	**AUC**
3	0.384	0.663	0.638	0.434	0.544	0.722
4	0.393	0.652	0.657	0.432	0.521	0.729
5	0.391	0.640	0.674	0.432	0.508	0.728
6	0.406	0.627	0.690	0.436	0.49	0.737
7	0.414	0.614	0.705	0.431	0.478	0.738
8	0.429	0.601	0.716	0.429	0.474	0.734

a maximal depth of 8 trees.

Thus, by judging the metrics alone, XGB model outperforms the rest of the algorithms in almost all of the instances across all metrics. Only when the maximum depth is set at 8, RF performs better if evaluated by AUC, and even then the difference is marginal - approximately 0.004 units. Note that AUC has a maximum value of 1.0 and receiving an AUC of 0.5 indicates the model is doing no better than “random guessing”.

After running all the algorithms with the full training set and finalized parameters, the final results are obtained for each model. The ROC curve and lift chart in Figure 5.1 displays visually how each of the models compare to each other over the test set. XGB model clearly outperforms once again all others, which is also evident by reading from Table 5.2, summarizing each algorithms predictive capabilities. All metrics are now worse off they were during the training phase. XGB suffers the smallest degradation in terms of the models predictive power.

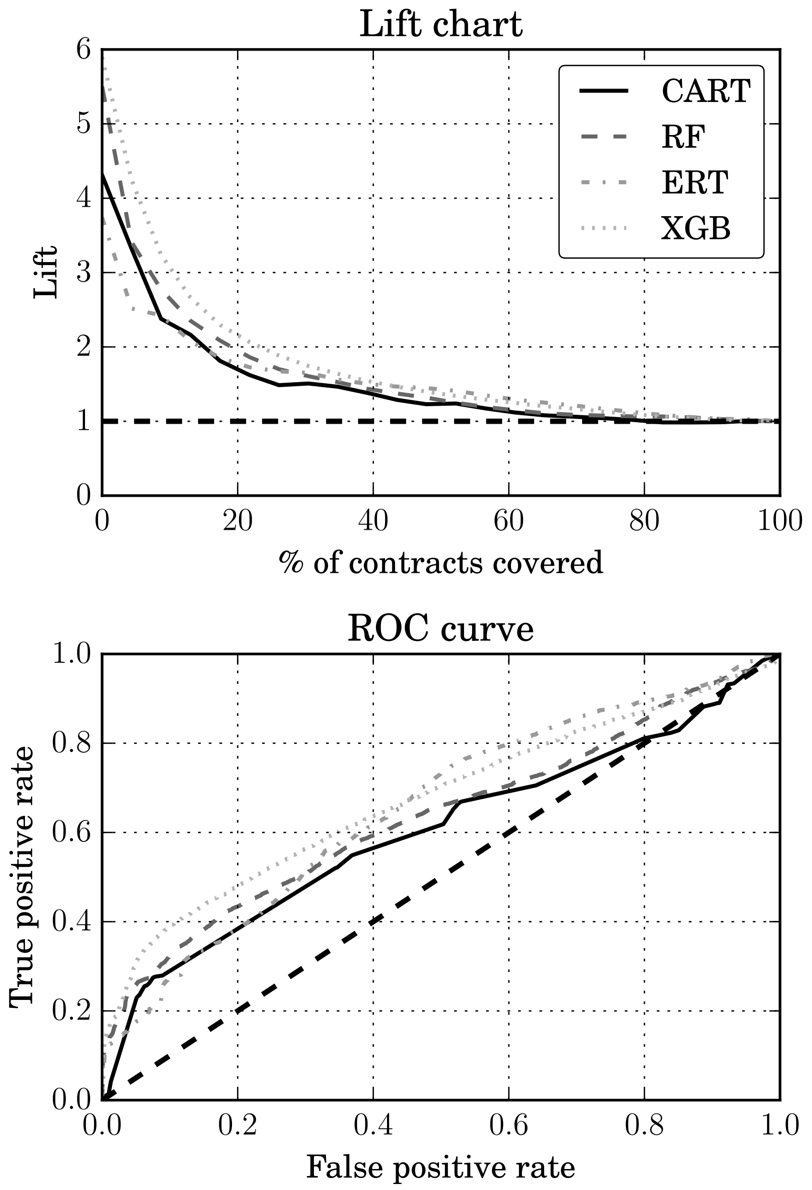

Figure 5.1: Performance on the test set for all algorithms with tuned parameters.

Table 5.2: Aggregate results for the testing set, as evaluated by all the used metrics.

	F-score	**Logarithmic loss**	**AUC**
CART	0.075	0.700	0.602
RF	0.107	0.609	0.641
ERT	0.098	0.644	0.652
XGB	0.133	0.518	0.678

From the Lift and ROC charts in Figure 5.1, it can be seen that the models are capable of ranking the data points in terms of how risky or likely they are to churn over the following three month period. For XGB, the lift is still above 2 after covering 20% of the total contract base, meaning that for every non-churning contract in this category, there are two contracts that will be cancelled. This has direct implications for how to allocate resources to enhance retention rate. The ROC curve tells the same story, albeit from a different angle.

Finally, Figure 5.2 shows the confusion matrices for each of the models and prediction specific statistics calculated for them. CART achieves the best TPR of 61.8%, but it does so at the cost of predicting a contract to churn much more often, which in turn is reflected in it's FPR, also the highest. With respect to TPR, ERT and XGB receive similar scores, but XGB scores are better with the rest of the statistics, and has the best PPV of 7.7%. These indicate how each model would be able to make use of resources aimed at reducing contract churn.

5.2 Prominent variables

Feature importances across all the models are shown in Figures 5.3-5.4, which equal the MDG all except for the XGB model. For XGB the feature importances are calculated by taking the number of times a feature has been used to split across all the trees as an estimate for feature importance. All feature importances are scaled to fit the range from zero to one.

CART

True class	Predicted class: Churn	Predicted class: No churn	
Churn	3715	2297	**TPR** 61.8%
No churn	89221	88472	**FPR** 50.2%
	PPV 4.0%	**FOR** 2.5%	

RF

True class	Predicted class: Churn	Predicted class: No churn	
Churn	2761	3251	**TPR** 45.9%
No churn	42770	134923	**FPR** 24.1%
	PPV 4.0%	**FOR** 2.3%	

ERT

True class	Predicted class: Churn	Predicted class: No churn	
Churn	3060	2952	**TPR** 50.9%
No churn	53416	124277	**FPR** 30.1%
	PPV 5.4%	**FOR** 2.3%	

XGB

True class	Predicted class: Churn	Predicted class: No churn	
Churn	3060	2952	**TPR** 50.9%
No churn	36823	140870	**FPR** 20.7%
	PPV 7.7%	**FOR** 2.1%	

Figure 5.2: Confusion matrices for each model.

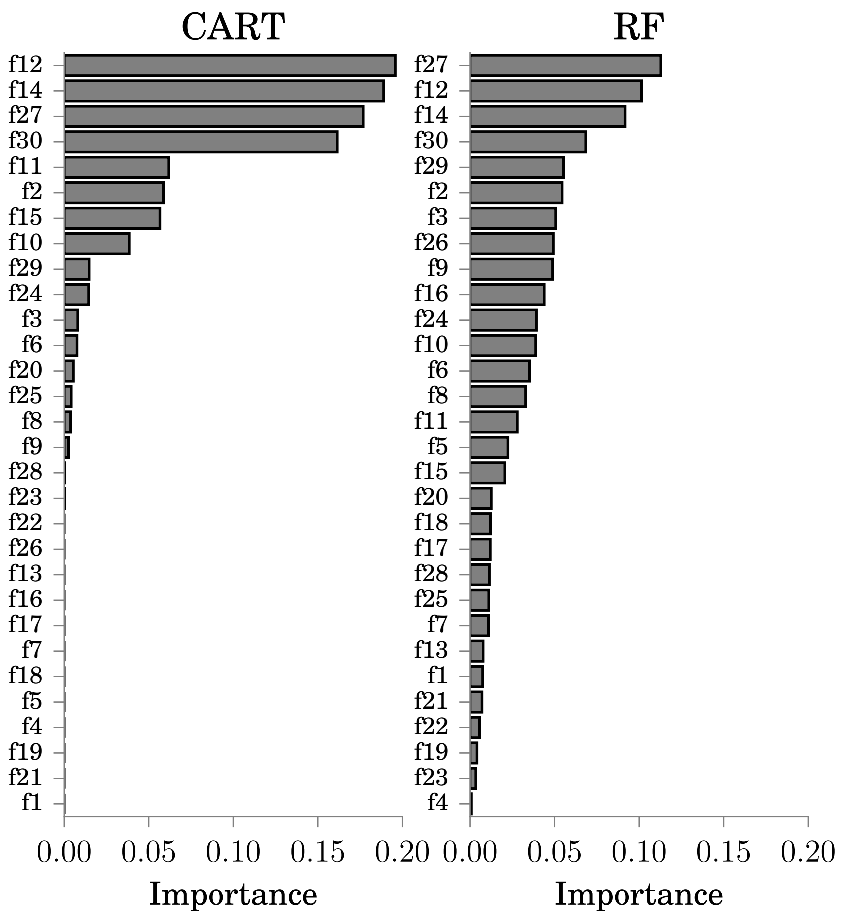

Figure 5.3: Feature importances as indicated by the trained CART and RF models, respectively.

CART model produces a tree which heavily weights the top four variables over others, with many of the features not receiving any weight at all. Out of the 10 most highly ranked features, three are customer related, five are linked to the customer portfolio level and only one feature deals either with the associated costs or contractual options. In comparison, RF has a more evenly distributed set of feature importances, although the types of variables are almost the same with two customer attributes, six portfolio features, one contractual and one operational variable.

ERT and XGB show more variability in the types of features, with both

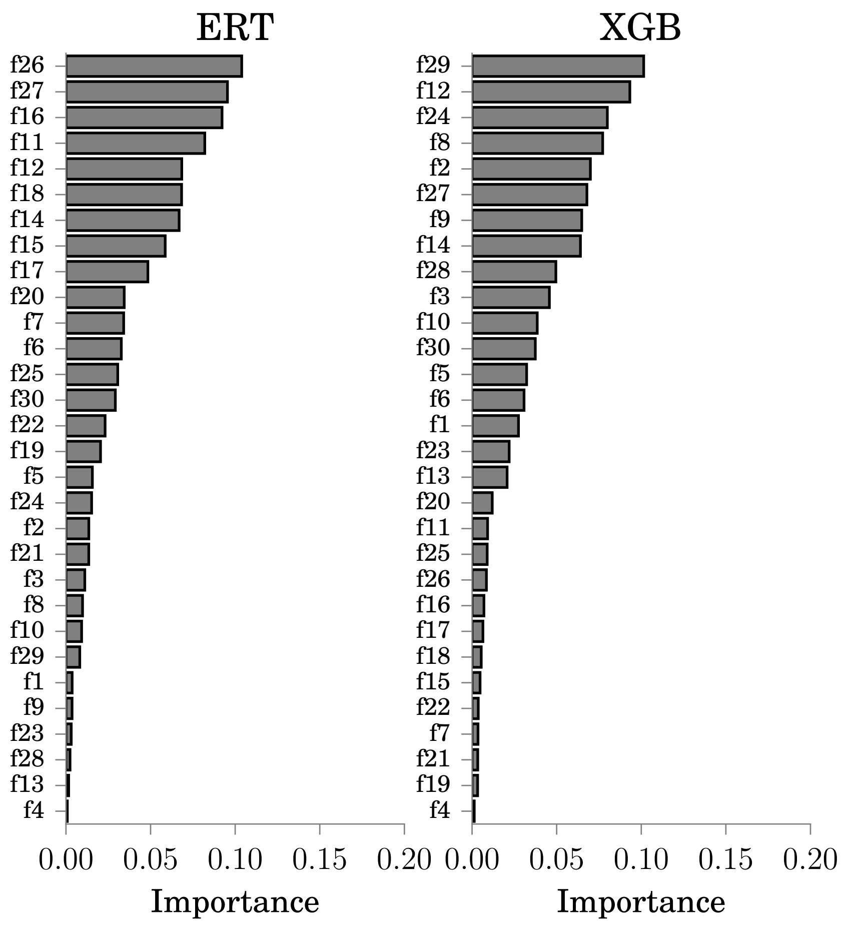

Figure 5.4: Feature importances as indicated by the trained ERT and XGB models, respectively.

having similar distribution of feature importances as RF. Interestingly, ERT has the highest number of contractual variables marked as high importance features, contrary to rest of the models, which have at most only one contractual term among the 10 best features. Additionally, ERT identifies one operational characteristic, two customer and portfolio related drivers behind contract risk. XGB is focused very much on the portfolio level indicators with seven features from this category, while also presenting some influence from cost and customer related factors, but none from the operational side.

A final summary of the feature representation among the ten best fea-

Table 5.3: Number of times a feature from a set category was one ranked among the 10 best predictors according to the feature importance metric in use.

Model	**Operational**	**Customer**	**Costs**	**Portfolio**	**Contractual**
CART	0	3	1	5	1
RF	1	2	0	5	1
ERT	1	2	0	2	5
XGB	0	1	2	7	0
Σ	**2**	**8**	**3**	**19**	**7**

tures is in Table 5.3, which clearly indicates the importance of portfolio level variables. Second biggest factor stems from customer based attributes, which is almost equal to the number of contributions from contractual factors.

5.3 Predictive capability and model explanations

The clear discrepancies in the training and test set performances are clear enough warrant further exploration. Although explicitly chosen, the modeling setup creates a situation in which there are class imbalances of different magnitude among the training and testing sets, as shown in Figure 5.5, contributing to the problem. This is especially evident when one uses F-score to measure the models. A possible reason for this is the presence of many early predictions, where contracts receive high risk scores indicating that they are about to be cancelled, but these time periods do not fit the evaluation period where they would be correctly labelled as churned. Looking back at the confusion matrix in Table 5.2 indicates that the models catch many of the churning contracts to some extent, judging by the TPR, but these come with the cost of low PPV values.

This suggests that capturing the exact churning points is in practice hard. Many variables are rolling averages or sums, which could potentially aggravate the problem, because a sudden pike in a risk factor could be hidden

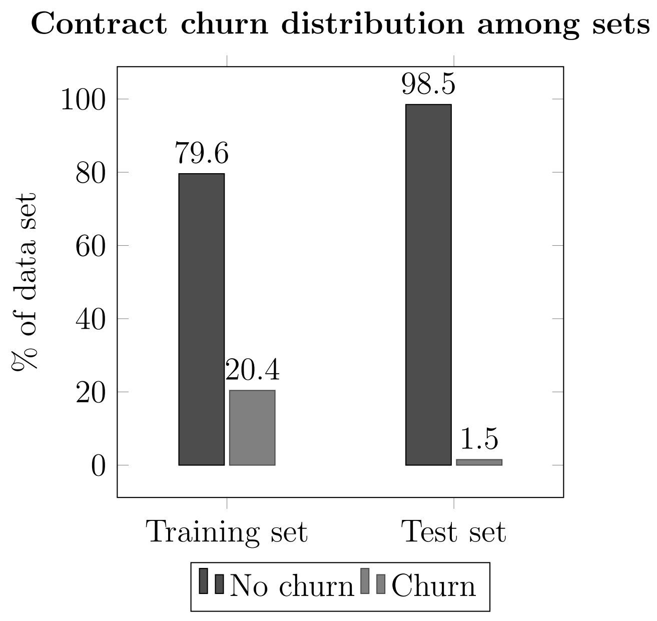

Figure 5.5: Relative amount of churn, separated by the data set in question.

underneath the rolling average window, or perhaps be left outside of it, in the case of summation.

Some rudimentary analysis of the signal strength, namely the churn probability, is given in Figure 5.6. In it is depicted the average churn probability as a function of time until the last available date in the time series for both, the churned contracts and those which have not been cancelled. Evidently, the churned contracts have systematically a higher mean, although it is shadowed by relatively high standard deviation as measured in the sample. The variability is much greater for the churned contracts overall, which helps understand the high number of FPs, contributing to a low PPV.

Although it is hard for the models to capture the exact churn date, the ROC and lift charts suggest they are capable of separating contracts in terms of how risky they are. The problem is, that it does not address whether or not there are multiple instances of the same contract within the same categories. Thus the top scoring categories are possibly inflated with a number of TPs for a limited set of contracts. But even so, the original assessment holds and this is subject to a separate discussion, as it is mainly to due with how the model is validated.The same effect of inflation is also present in the confusion matrix, as mentioned previously.

Looking at the number of FPs in the confusion matrix and just by selecting all the contracts that were cancelled and prematurely marked as churned, there are already 11,620 false positives in the XGB model in the testing set. On average, the signals were 14.9 months early with a standard deviation of 9.1 months - both relatively high compared to the actual prediction window. These could be related to some of the contractual details - eg. allowed cancellation periods - which were unavailable in the data set.

Similar effects can be observed with the rest of the algorithms. XGB was best of when it came to the mean and standard deviation of the early signals. The number of premature FPs were within 2% of each other for RF, ERT and XGB, constituting approximately 32% of all the FPs. Early false predictions for churn are also reflected in the logarithmic losses.

In this regard, CART does not perform so well, because it has 18,969 early false positives, which is roughly 21% of all churn predictions. The relatively weak performance of the the CART model in comparison to the rest is not surprising. Using only the greedy algorithm for building the tree does not guarantee globally optimal solutions and is subject to overfitting, which is partially suggested by the Figures 5.1 and 5.3. In the former, towards the end of the ROC and lift curves, CART drops below the constant line, denoting random guessing. The latter image shows how much of the data were already split by the first four dominant features, contrary to the rest of the algorithms, whose feature importance distributions were much more equal.

A hindrance for all of the algorithms are correlated features and Figure 5.7 depicts the correlations amongst all the continuous variables. In the figure, some of the variables on the diagonal are highly correlated. These are mainly the operational and portfolio level predictors. To a degree this was to be expected, because many of the portfolio level predictors are derivatives of the operational level variables scaled by the customer total portfolio size,

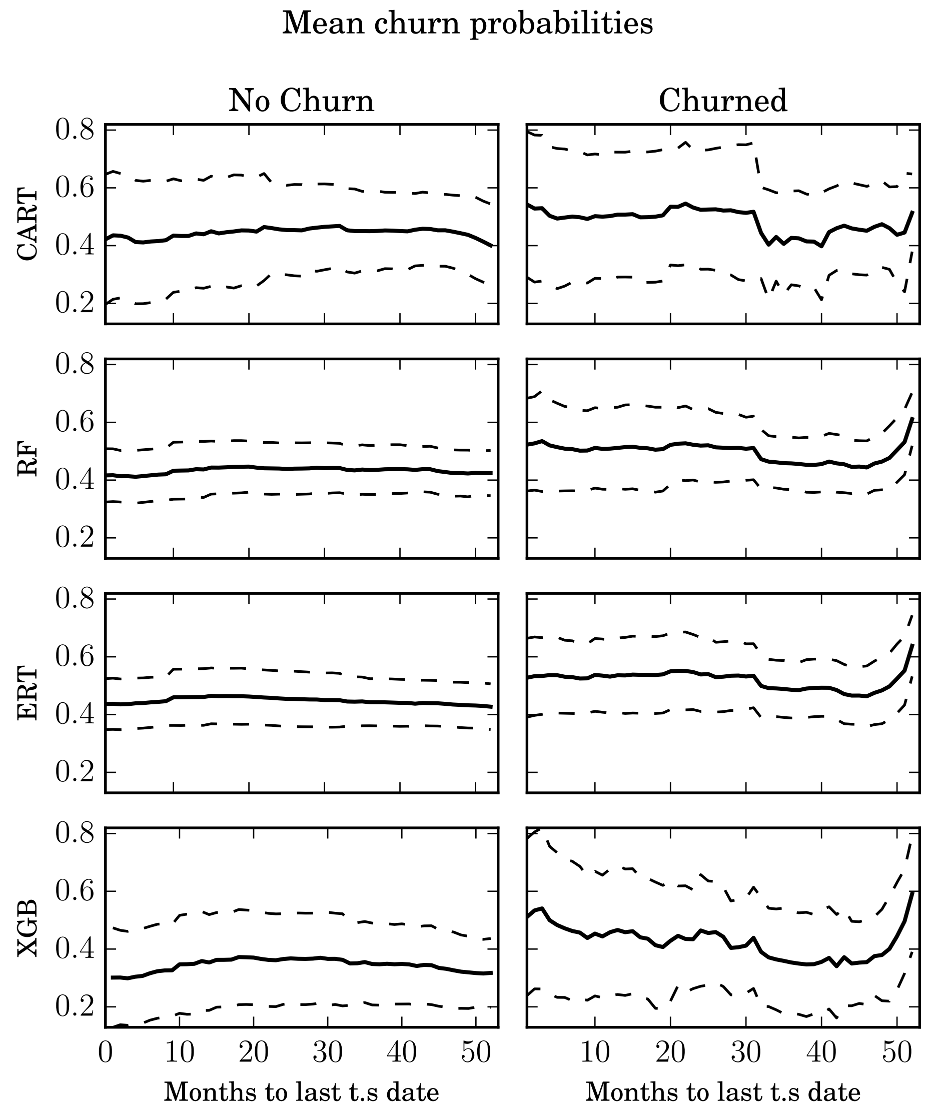

Figure 5.6: Average churn signal from all the contracts, grouped by its class - churned contracts vs not churned - and measured against the time until the last date in each contracts time series. Dashed lines are one standard deviation away from the mean.

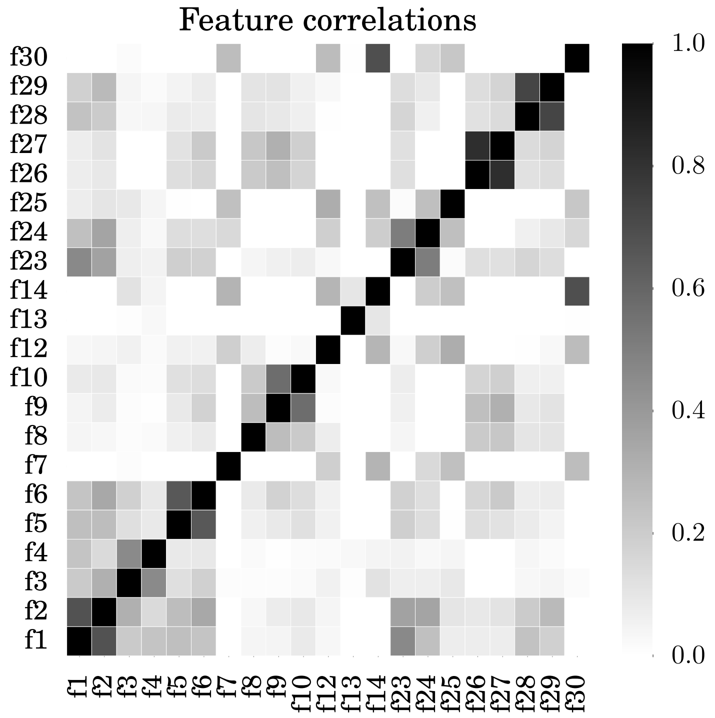

Figure 5.7: Correlation heat map for all the continuous features. All the negative correlations have been set to zero for readability. The biggest negative correlation was approximately -0.23. The feature correlations were calculated for the training set only.

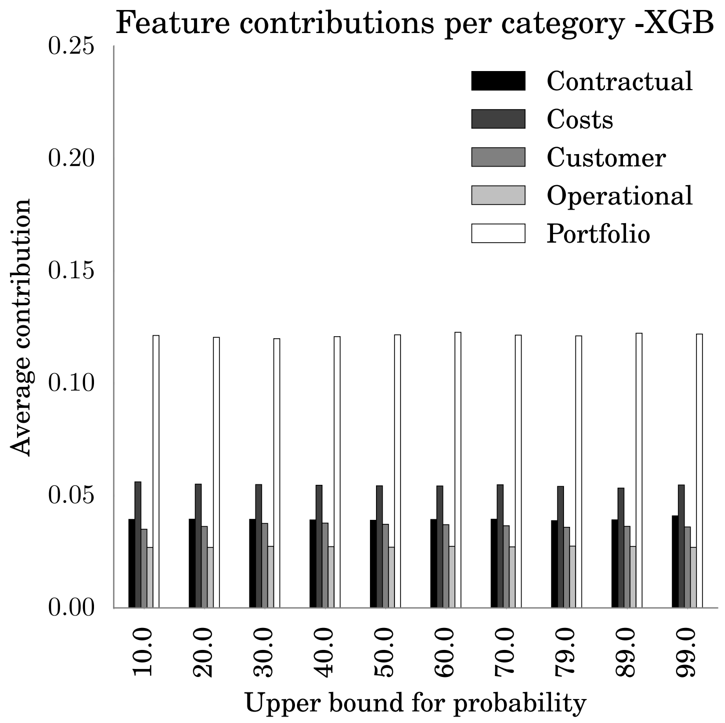

Figure 5.8: Feature weight contributions from the XGB model, grouped by the respective categories of each of the features. The training set was used in the image construction.

thus inducing obvious correlations through the smaller sized customers. This helps explain why the portfolio variables overshadow the much of the rest, evident also in the Figure 5.8.

Furthermore, Figure 5.8 shows how at least in the case of the XGB model, the relative feature contributions are very similar irrespective of the associated risk of loosing the contract. This suggest that the risks are driven on average by the same features across the groups. However, these observations do not rule out the possibility of single features exhibiting more complex or comprehensible relationships, such as positive correlations with increasing amount of customer complaints and it's associated weight, given by LIME.

Table 5.3 also shows how all except ERT model favor portfolio level variables. Part of the reason is the fact that ERT chooses the split points for each of the considered splitting variables also randomly, and then choosing the best variable based on an impurity measure. Hence binary variables will always produce the splits, whereas continuous variables are prone to having worse reduction in the impurity measure.

Chapter 6

Discussion and conclusions

There were two goals set for this thesis, separate the truly risky service contracts from the total population and secondly, identify the causal drivers behind the risks. All of the applied algorithms were able to perform the first task, providing some predictive value, when it came to ordering contracts by their churn probabilities. For the used validation scheme, XGBoost proved to be the most effective one, with RF and ERT exhibiting similar performance and CART being the worst. It was expected that the ensemble methods would outperform a single decision tree fitted by the CART algorithm which was the case. This is in line with existing literature and the theory behind the applied modeling techniques.

When it came to analyzing the results, it was interesting to note how much effect early false predictions can have and how early these are caught by the models. This lead to further questions about the proper conduct of model. Now, all the models are punished severely for false early predictions, even though many of the variables will not change much over time because of their design. Under the current validation scheme, even if the models are correctly predicting many months ahead that a service contract is likely to be cancelled, such predictions will be penalized, no matter what the outcome.

Resolution at which the models are scored and judged naturally affects the outcome. Such contractual details were not available which would allow the construction of a more elaborate validation scheme, in which where service contracts would only be scored when they may be cancelled. On the other hand, even if such information were available, not all contracts have a set period when they must be cancelled.

Another way around this would be to add features describing such conditions, eg., a flag variable denoting that the service contract is near its cancellation period or that it does not have one. This way the models could still be scored continuously without the need for a specific validation scheme, because the algorithms should now be able to learn from the added features, which contracts may be cancelled during any given period. The aforementioned is part of process known as feature engineering, which is key area to focus on. Instead of spending time tuning the model, finding the best algorithm and it's parameters, it is often more helpful to generate informative variables, which the algorithms can effectively learn and generate better models based on them.

Also, the staying power of the models was not considered. It is paramount to know when the predictive power of a model deteriorates enough to warrant re-modeling. Accounting for such factors could have been considered, but were chosen not to, because of the relatively short time period compared to the average contract length. The effect has been considered in literature, e.g., [29].

Trying to explain the factors behind the risks lead to the discovery of some correlated features from the portfolio domain that dominated many of the models. Thus, it is hard to actually determine, whether or not these are the actual drivers contributing to the risks. The results from LIME also support these findings, but leave still call for room for further explanation, as it is designed to provide insights into single predictions, even if the prediction is given by a black box model. Nevertheless, if the model is only trained on correlated features, which potentially mask other relevant ones, this will not be very effective.

The ability to explain individual predictions is of great interest and de-serves much more emphasis. This was not discussed thoroughly in any related literature with comparable data sets, although [32] emphasized the comprehensibility of the models, so that they are intuitive and match existing domain knowledge. Exploring algorithms such as LIME are prominent options to look into, because of their potentially high value from the business perspective. Not only would it add another layer of model validation, ensuring the models predictions are comprehensible, but improve the chances of receiving actionable insights. These are hard to draw from the implemented models directly, as they do not provide established methods to explain predictions case by case. The exception here is CART, which can be easily visualized and explained how it arrives at its predictions.

Compared to other related work [10], in terms of the churn prediction setting, this study had the added benefit of not having to explicitly define churn. Since the predictions were done for service contracts, most of the time it was clear when a contract has been cancelled, instead of defining periods of customer inactivity as them being churned.

Ultimately the biggest challenges were those of choosing a proper validation scheme and engineering quality features. Albeit being a difficult setting to operate under, each model was able to differentiate between risky service contracts and to provide insights into each of the models overall structure. Both of these aspects were left open for future line of development to further improve the models and, in the end, to help provide actionable predictions under an evaluation scheme that best corresponds to existing business practices.

Bibliography

[1] Ahn, J., Han, S., and Lee, Y. Customer churn analysis: Churn determinants and mediation effects of partial defection in the Korean mobile telecommunications service industry. *Telecommunications Policy 30*, 10-11 (2006), 552–568.

[2] Athanassopoulos, A. D. Customer satisfaction cues to support market segmentation and explain switching behavior. *Journal of Business Research 47*, 3 (2000), 191–207.

[3] Bengio, Y., and Grandvalet, Y. No unbiased estimator of the variance of k-fold cross-validation. *The Journal of Machine Learning Research 5* (2004), 1089–1105.

[4] Blaser, R., and Fryzlewicz, P. Random rotation ensembles. *The Journal of Machine Learning Research 17*, 1 (2016), 126–151.

[5] Breiman, L. Bagging predictors. *Machine Learning 24*, 2 (1996), 123–140.

[6] Breiman, L. Out-of-bag estimation. Technical report, Department of Statistics, University of California, Berkeley, CA, 1996.

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

[8] Breiman, L., Friedman, J., Olshen, R., and Stone, C. *Classification and Regression Trees.* Wadsworth and Brooks, Monterey, CA, 1984.

[9] Chen, T., and Guestrin, C. Xgboost: A scalable tree boosting system. *CoRR abs/1603.02754* (2016).

[10] Coussement, K., and De Bock, K. Customer churn prediction in the online gambling industry: The beneficial effect of ensemble learning. *Journal of Business Research 66*, 9 (2013), 1629–1636.

[11] Eiland, E. E., and Liebrock, L. M. Efficacious End User Measures Part 1: Relative Class Size and End User Problem Domains. *Advances in Artificial Intelligence 2013* (2013), 2:2–2:2.

[12] Ferri, C., Hernández-Orallo, J., and Salido, M. A. Volume under the ROC surface for multi-class problems. In *Machine Learning: ECML 2003, 14th European Conference on Machine Learning, Cavtat-Dubrovnik, Croatia, September 22-26, 2003, Proceedings* (2003), pp. 108–120.

[13] Friedman, J. H. Stochastic gradient boosting. *Computational Statistics and Data Analysis 38* (1999), 367–378.

[14] Friedman, J. H. Greedy function approximation: A gradient boosting machine. *Annals of Statistics 29* (2000), 1189–1232.

[15] Geurts, P., Ernst, D., and Wehenkel, L. Extremely randomized trees. *Machine Learning 63*, 1 (2006), 3–42.

[16] Gregorutti, B., Michel, B., and Saint-Pierre, P. Correlation and variable importance in random forests. *Statistics and Computing* (2016), 1–20.

[17] Gür Ali, O., and Aritürk, U. Dynamic churn prediction framework with more effective use of rare event data: The case of private banking. *Expert Systems with Applications 41*, 17 (2014), 7889–7903.

[18] Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. *The Elements of Statistical Learning : Data Mining, Inference, and Prediction.* Springer series in statistics. Springer, New York, 2009.

[19] Idris, A., Khan, A., and Lee, Y. S. Intelligent churn prediction in telecom: Employing mRMR feature selection and rotboost based ensemble classification. *Applied Intelligence 39*, 3 (2013), 659–672.

[20] Kass, G. V. An exploratory technique for investigating large quantities of categorical data. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 29*, 2 (1980), 119–127.

[21] Kohavi, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2* (San Francisco, CA, USA, 1995), IJCAI'95, Morgan Kaufmann Publishers Inc., pp. 1137–1143.

[22] Kurgan, L. A., and Musilek, P. A survey of knowledge discovery and data mining process models. *The Knowledge Engineering Review 21*, 1 (2006), 1–24.

[23] Larivière, B., and den Poel, D. V. Investigating the role of product features in preventing customer churn, by using survival analysis and choice modeling: The case of financial services. *Expert Systems with Applications 27*, 2 (2004), 277–285.

[24] Louppe, G. *Understanding Random Forests: From Theory to Practice.* PhD thesis, University of Liege, Belgium, 2014. arXiv:1407.7502.

[25] Louppe, G., Wehenkel, L., Sutera, A., and Geurts, P. Understanding variable importances in forests of randomized trees. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 431–439.

[26] Norouzi, M., Collins, M., Johnson, M. A., Fleet, D. J., and Kohli, P. Efficient non-greedy optimization of decision trees. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 1729–1737.

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

[28] Ribeiro, M. T., Singh, S., and Guestrin, C. "why should I trust you?": Explaining the predictions of any classifier. *CoRR abs/1602.04938* (2016).

[29] Risselada, H., Verhoef, P. C., and Bijmolt, T. H. Staying power of churn prediction models. *Journal of Interactive Marketing 24*, 3 (2010), 198–208.

[30] Shearer, C. The CRISP-DM Model: The new blueprint for data mining. *Journal of Data Warehousing 5*, 4 (2000), 13–22.

[31] Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B 58* (1994), 267–288.

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

[33] Xie, Y., Li, X., Ngai, E. W. T., and Ying, W. Customer churn prediction using improved balanced random forests. *Expert Systems with Applications 36*, 3 (2009), 5445–5449.