NOVA IMS Information Management School

MGI

Mestrado em Gestão de Informação
Master Program in Information Management

Credit scoring with advanced analytics

Applying machine learning methods for credit risk assessment at the Frankfurter Sparkasse

Jonas Reichenbach

Project Work presented as partial requirement for obtaining the Master's degree in Information Management.

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

CREDIT SCORING WITH ADVANCED ANALYTICS

Applying machine learning methods for credit risk assessment at the Frankfurter Sparkasse

in cooperation with

ec4u expert consulting ag

by

Jonas Reichenbach

m2016097

Project Work presented as partial requirement for obtaining the Master's degree in Information Management, with a specialization in Information Systems and Technologies Management.

Advisor: Roberto Henriques, PhD

Advisor ec4u: Alexander Beck, PhD

Date of delivery: 29.03.2018

Abstract

The need for controlling and managing credit risk obliges financial institutions to constantly reconsider their credit scoring methods. In the recent years, machine learning has shown improvement over the common traditional methods for the application of credit scoring. Even small improvements in prediction quality are of great interest for the financial institutions.

In this thesis classification methods are applied to the credit data of the Frankfurter Sparkasse to score their credits. Since recent research has shown that ensemble methods deliver outstanding prediction quality for credit scoring, the focus of the model investigation and application is set on such methods. Additionally, the typical imbalanced class distribution of credit scoring datasets makes us consider sampling techniques, which compensate the imbalances for the training dataset. We evaluate and compare different types of models and techniques according to defined metrics. Besides delivering a high prediction quality, the model's outcome should be interpretable as default probabilities. Hence, calibration techniques are considered to improve the interpretation of the model's scores. We find ensemble methods to deliver better results than the best single model. Specifically, the method of the Random Forest delivers the best performance on the given data set. When compared to the traditional credit scoring methods of the Frankfurter Sparkasse, the Random Forest shows significant improvement when predicting a borrower's default within a 12-month period. The Logistic Regression is used as a benchmark to validate the performance of the model.

Keywords

Credit Scoring; Classification Techniques; Ensemble Methods; Basel Accords

Table of Contents

Table of Figures

Table of Tables

Table of Algorithms

Abbreviations

ADASYN	Adaptive-Synthetic-Sampling
AUC	Area Under the Curve
Ada	Adaptive Boosting
BCBS	Basel Committee of Banking Supervision
CV	Cross-Validation
EAD	Exposure at Default
EL	Expected Loss
ET	Extra Trees
FPR	False-Positives Rate
FraSpa	Frankfurter Sparkasse
Grad	Gradient Boosting
IQR	Interquartile Range
IRB	Internal Ratings Based
IRB-A	Internal Ratings Based Advanced
ISO	Isotonic Regression
LGD	Loss given Default
LR	Logistic Regression
MSE	Mean Square Error
PD	Probability of Default
PR	Precision-Recall
RF	Random Forest
RM	Rolling-Average Mean
ROC	Receiver-Operator-Characteristics
ROS	Random Over-sampling
RUS	Random Under-sampling
SMOTE	Synthetic Minority Over-sampling Technique
STD	Standard deviation
SVC	Support Vector Machine Classifier
TTEP	Time-to-empty-pockets
TPR	True-Positives Rate
Tree	Decision Tree

1. Introduction

The digitalization enables humanity to create data in a speed that was not imaginable some years ago. Due to the accelerating nature of this process, 90 percent of the data that is stored nowadays was generated within the last decade (Raschka, Olson 2015). Due to the vast amount and complexity of the data, it is not directly interpretable by humans. Therefore, methods are needed to deliver understandable insights from the data. Turning large collections of data into knowledge is summarized in the term data mining. A subfield of data mining, machine learning, consists of the computational analysis of data and the generation of predictions without human interaction (Han et al. 2012). Machine learning has developed a large popularity due to the amount of available data and the increase in available computational performance. An important advantage of machine learning over traditional methods is the automatic generation of rules from the structure and patterns of given data. These rules are applied to unseen data to derive predictions (Han et al. 2012). The advantages of machine learning are used in the thesis to predict credit defaults, based on real-world customer data. In July 2017, total of €1.182 trillion were lent to employees and other individuals in Germany. This presents an increase of 3.5% to the same period in the previous year (Deutsche Bundesbank 2018). Therefore, banks need to know about the likeliness of a credits future default to cut the potential negative impacts, or to avoid handing out the credit at all. For financial institution the benefit of such a predictive model is estimated to be a decrease of 6%-25% of total losses due to credit default under conservative assumptions (Khandani et al. 2010).

1.1. Context

The granting of credits is considered a key activity of the banking sector. The credit business generates profits for the financial institutions as well as contributes to the community by allowing investments with insufficient liquidity. Although, it can quickly become a significant source of risk for the lender (Ala'raj, Abbod 2016). In 2008 the financial markets found themselves confronted with catastrophic losses triggered by the thread of credit defaults in the mortgage markets. Even high rated portfolios lost large percentages of their value. Results of this financial crisis spread globally to virtually every economy and market (Longstaff 2010). Having knowledge of the credit risk could not be more important to financial institutions in their own interest.

The riskiness of a credit is evaluated through credit scoring. Credit scoring is a task which uses “formal statistical methods […] for classifying applicants for credit into 'good' and 'bad' risk classes” (Hand, Henley 1997). According to LOUZADA ET AL. (2016) credit scoring “is a numerical expression based on a level analysis of customer credit worthiness”. Credit scoring can be determined by the default probability. However, some institutions combine other information in the credit score as well. Elementary, the lender discriminates between borrowers who will repay the loan and those who will potentially not pay the loan back entirely. In traditional credit scoring, a set of rules is manually derived based on historical data to determine the default probability of borrowers. However, creating such rules manually implies previous knowledge of the data and its patters, which is not necessary with the application of machine learning (Abellán, Castellano 2017). The information the credit score is based on is mainly the borrower’s information possessed by the financial institution. This usually involves various characteristics about the borrower, including their financial history with the institution and

sociodemographic information such as income and age. Furthermore, external data sources can be considered as well (Crook et al. 2007).

The Basel Accords, published by the Basel Committee of Banking Supervision (BCBS), define requirements financial institutions must fulfil. Specifically, the Basel Accord published in June 2004, also known as Basel II, determines the minimum capital requirements of an institution's credits, based on the internal evaluation of the credit risk (BCBS 2006). This is a revolutionary approach compared to static previous evaluation techniques since financial institutions could rely on their own evaluation of a borrower's risk to determine the credit risk for the first time. Hence, financial institutions are also legally pressured to enhance the performance of credit scoring methods.

Basel II provides three different methods to determine the capital requirements: The Standardised Approach, the Internal Ratings Based (IRB) Approach and the Securitisation Framework. Institutions which comply with certain conditions and were approved to use the IRB approach are permitted to estimate the key credit risk drivers internally, resulting in a capital requirement for a given exposure. This allows financial institutions to have full control about their capital requirements (BCBS 2006).

The risk components of the IRB are:

- Probability of default (PD)
- Loss given default (LGD)
- Exposure at default (EAD)
- Effective maturity

Financial institutions forecast the average level of credit losses, the Expected Loss (EL), as a cost of doing credit business. To determine the EL of a credit, the risk components of the IRB are used, defining the EL as $EL = PD * EAD * LGD$ (BCBS 2005).

When further investigating the IRB, two approaches of the IRB are defined in the Basel Accords. The foundation approach and the advanced approach (IRB-A). Firstly, the foundation approach lets the bank estimate the PD, while all other elements are estimated by a legally assigned supervisor. Secondly, the advanced approach enables the bank to have full control over the estimate and calculation of all IRB elements (BCBS 2005). However, to make use of a IRB-A, several requirements need to be fulfilled (BCBS 2006). The required capital depends on the risk and the size of the given exposure. Basel Accords encourage financial institutions to provide sophisticated and accurate estimations on the key risk drivers. Using the IRB-A method often results in lower capital requirements for the institution.

Over time the Basel Committee updated the recommendations on banking law, resulting in multiple Basel Accords numbered according to their appearance. Currently three versions of the Basel regulations are available. The documents released during a certain period are categorized as a regulatory wave with a specific name. Documents published from 1999 – 2008 are named Basel II (Penikas 2015). However, time revealed some weak spots in the documents (Blum 2008) and were followed up by the financial crisis, which proofed the necessity of regulatory adjustments. The Basel III document aims at strengthening the risk management of the banks, as well as increasing the transparency and disclosures. Additionally, it consists of more accurate calculations for of default risk, especially when the scoring is conducted by an external rating agency (BCBS 2011).

In conclusion, obtaining knowledge of the credit risk is not only crucial for banks due to internal decisions, but is also driven by the legal need to compensate potential credit losses with capital (Abellán, Castellano 2017).

1.2. Motivation

The first approach to determine a credit score was conducted by Altman (1968). A lot of development of various models has been conducted since, however the Logistic Regression, a relatively simple model, is still the industry standard (Lessmann et al. 2015). Recent research has shown improvements of modern machine learning methods over the traditional statistical approaches in credit scoring tasks (Abellán, Castellano 2017).

Many different machine learning algorithms have proven to be suitable for solving credit scoring problems. Such as neural networks (West 2000; Angelini et al. 2008; Zhao et al. 2015; Luo, Wu 2017), support vector machines (Schebesch, Stecking 2005; Chen et al. 2009; Harris 2015), decision trees (Khandani et al. 2010), hybrid models (Tian-Shyug L., I-Fei C. 2005) and different ensemble methods (Ala'raj, Abbod 2016; Hung, Chen 2009; Marqués et al. 2012; Nanni, Lumini 2009; Xiao et al. 2016; Xia et al. 2017). On the task of credit scoring, ensemble methods showed an improved performance over single model approaches in various comparisons (Lessmann et al. 2015; Baesens et al. 2003; Louzada et al. 2016). Consequently, a lot of recent research focuses on ensembles methods. Most of the ensemble methods make use of relatively simple combination methods, such as the bagging schema (Breiman 1996) using various base models. Very recent research has also experimented with ensembles of heterogenous base models (Lessmann et al. 2015; Ala'raj, Abbod 2016). Due to the throughout positive feedback of ensemble methods, we focus on these machine learning methods in this thesis.

The advantages of intelligent systems, such as machine learning, were stated by GOONATILAKE, TRELEAVEN (1995), who mention five appealing aspects:

- **Learning:** Identification of patterns in a data set, where the performance increases with more data.
- **Adaption:** Adaption to changes in the data set.
- **Flexibility:** Working with incomplete data.
- **Transparency:** Showing the path of the internal processes. Hence, we can visualize the decisions and derive conclusions from the system.
- **Discovery:** Exploring internal relations of data.

These aspects make machine learning suitable for the credit scoring task, as well as for many other disciplines. In this thesis we make use of the mentioned advantages to deliver a transparent scoring method, which is flexible towards changes in the data. Moreover, we use machine learning to determine indicators for a future credit default on the given data set.

1.3. Goal

The goal of this master thesis is to investigate suitable machine learning models for credit scoring. For this purpose, we investigate machine learning methods from a theoretical perspective and then switch

to the practical application. For the practical part, we have access to the credit data of the Frankfurter Sparkasse (FraSpa) and their internal credit scores. FraSpa is currently using traditional methods to evaluate the credit risk of their borrowers. Since machine learning methods have proven to deliver superior performance for the task of credit scoring, FraSpa would like to investigate the potential of such methods applied to their own data. For a final quantitative result of this thesis, the machine learning scores we obtain are compared to the internal scores of FraSpa according to defined metrics.

1.4. Methodology

For creating the predictive model we consider SIDDIQI (2012), who introduces the process of credit scoring with the following elements:

1. Building a statistical model from historical data
2. Application of the model to predict the borrower's credit risk scores
3. Measure the accuracy of the model

Following the proposed steps, a classical credit scoring model is created. Interestingly, these elements are almost equal to the machine learning process which we consider applying machine learning models. The information, if a credit will default in a defined period, is derived from the internal ratings of FraSpa. Here, a borrower is considered defaulted if certain criteria are fulfilled. The machine learning models use the borrower's information to predict a default in a defined future period. The models output is a continuous probabilistic estimation in [0, 1]. Hence, the model produces a confidence for its certainty of the default. We consider this outcome as score. However, notice that the score does not necessarily represent the default probability of the borrower.

The prediction accuracy of the machine learning models is measured with defined metrics. Here, we use the score of the models and compare it to the occurrence of a future default. Finally, we compare the model and technique which was found to be the most successful with the internal scores of the FraSpa.

1.5. Structure of the thesis

In Chapter 2 we introduce the theoretical concepts of the relevant statistics and machine learning methods. Starting with the definition of input and output in Chapter 2.1, followed by the introduction of machine learning process and model fitting issues, which are shown in Chapter 2.2 and 2.3.

Furthermore, we show the concepts of selected models in Chapter 2.4, including the Logistic Regression and decision trees. The success of ensembles in recent research on credit scoring tasks makes us focus on such methods. Thus, we investigate three distinct types of ensemble methods in Chapter 2.5. In addition, we study approaches to balance data sets with an imbalanced class distribution and calibration techniques in the Chapters 2.6 and 2.7. In Chapter 2.8 we show methods used for data processing and model selection. Following that, we consider the suitable measures to determine the classification performance of our models in Chapter 2.9.

Chapter 3 shows the data we have obtained from FraSpa as well as the application of machine learning models. Here, we illustrate the structure of the data set and the manipulation we apply to it. Since the data is derived from multiple data sources, the combination and load of the tables is explained as well.

Moreover, the special machine learning techniques we introduced in the second Chapter are applied to a 3-month target. We use the information obtained from the evaluation of the methods to select which ones we want to display the results of in the following chapter.

The results are described in Chapter 4, which is split in two sections, since we used two different target variables. At first, we compare models and different approaches to improve the classification performance in Chapter 4.1. Secondly, resulting from the previous model comparison, we have the necessary knowledge to select the best performing approach to use for the comparison with the internal scores, which is shown in Chapter 4.3.

Chapter 5 concludes the findings of the previous chapters and summarizes the results.

2. Concepts

Machine learning is the field of algorithms and computers learning from data. Of especial importance for this thesis is the automatic discovery of complex patterns. Since this behaviour is usually ascribed to human beings, the algorithms appear intelligent (Han et al. 2012). Practically, this means to take "…a record of the actions of our subjects, learn from this record, and then create a model of these activities that will inform our understanding of this context going forward…" (Conway, White 2012). Hence, the machine learning model learns the patterns, creates rules based on these patterns and applies them to unseen data to create a prediction.

Within machine learning three different subfields reside:

- **Supervised learning** is the learning from labelled data. It is further split into the tasks classification, which uses categorical labels, and regression, which has continuous labels.
- **Unsupervised learning** is learning from unlabelled data. Here, classical tasks are the clustering of data according to certain attributes.
- **Reinforced learning** uses an active participation of the user, e.g. an expert, to label a presented example, which the program uses to improve its performance (Han et al. 2012).

Credit scoring considers the two classes of 'good' and 'bad' credits, which means categorical labels are needed. Hence, it is of the type supervised learning. We will focus on machine learning methods that suit classification tasks in the further course of this thesis.

2.1. Definition of input and output

The supervised learning approach uses labelled data to inspect the relationship between the input X and the output Y. The P input columns of data set are assigned as $X^1, \dots, X^P$ where each input X^j represents one feature of the data set. The P input columns can be interpreted as dimensions of the data. We refer to these input columns as features. The features are combined in the feature matrix X, where $X = (X^1, \dots, X^P)$. Each data sample consists one value for all P features. Supervised machine learning algorithms construct their decision rules based on the relation of X and Y (D' Angelo 2016). Consequently, the ideal relation of input and output can be viewed as $Y = f(X) + \varepsilon$, where $f()$ is provided by the model and ε is some random noise in the data. During training phase, also known as the model fitting, the model learns from the relation between input and output, which gets projected onto the mentioned function $f()$ (Bishop 2006). The data used during this process is conveniently called the training data. Once the model is trained, the learned patterns can be applied to unseen data. For the evaluation of a model's prediction quality, we predict the target of data samples with known targets. The data used for this evaluation is called the test data (Hastie et al. 2009).

For the task of credit scoring, as it is defined throughout this thesis, the true target $\boldsymbol{y}$ is defined as

$$\boldsymbol{y} = \begin{cases} 1 \; if \; credit \; defaults \; within \; period \\ 0 \; otherwise \end{cases},$$

where the period considered needs to be defined upfront. The outcome of the model is a probabilistic estimate $\boldsymbol{p}$ which is in $\boldsymbol{p} \epsilon [0, 1]$, which we refer to as *score*. $\boldsymbol{p}$ can be translated into the prediction $\hat{\boldsymbol{y}}$. Specifically, we define $\hat{\boldsymbol{y}}$ as

$$\hat{\boldsymbol{y}} = \begin{cases} 1 \; if \; \boldsymbol{p} \geq c \\ 0 \; otherwise \end{cases},$$

where c is a variable threshold $\boldsymbol{p} \epsilon [0, 1]$. The goal of the machine learning algorithm is to create a $\hat{\boldsymbol{y}}$ which minimizes the expected loss $L(\hat{\boldsymbol{y}} - \boldsymbol{y})$, with the actual target values $\boldsymbol{y}$ and the model's prediction $\hat{\boldsymbol{y}}$. Beware that the model's scores do not necessarily present the default probability, since it is merely a way of the model to show its confidence on the outcome. We investigate the theoretical issue of transforming the score to a default probability in Chapter 2.7 and apply them in Chapter 3.3.8.

2.2. The machine learning process

The roadmap of obtaining a predictive machine learning model consists of four sequential sub processes: the preprocessing of the data set, the application of the learning algorithm to the data set, its evaluation and optimization with defined metrics and finally the implementation of the model in a productive environment.

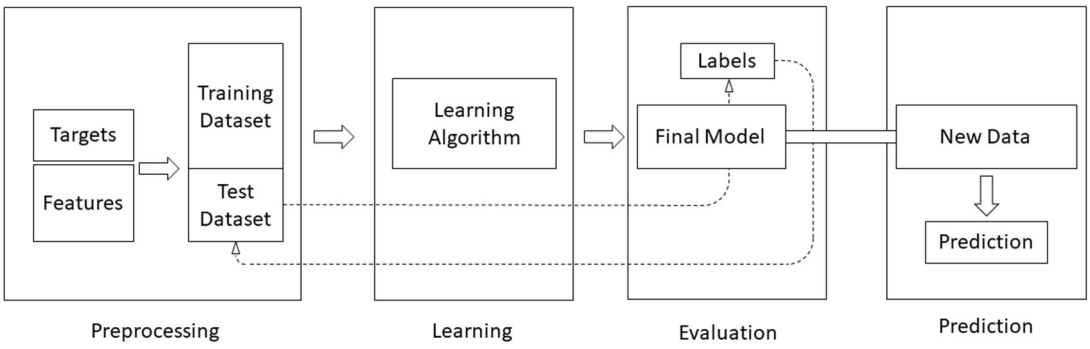

Figure 2.1: Machine learning roadmap. Inspired by Raschka, Olson (2015, p. 11). We show a high-level perspective for this plot.

The high-level process of fitting one model and its evaluation is shown in Figure 2.1. We briefly explain the different tasks. We have seen that data for supervised machine learning tasks consist of the input features and the corresponding output targets. We need to create a solid data set as basis for the model's predictions in the preprocessing phase, where we alter the data to suit the requirements of the machine learning models. Moreover, we obtain additional information from the raw data through feature extraction.

We split the data samples of the obtained data set into two mutually exclusive groups. The resulting data sets are the train set and the test set. In the learning phase, the model is fit to the train data and the corresponding training targets, hence it learns the relation between input and output here. In the evaluation phase, the predicted targets are compared to the true targets according to defined metrics. This allows to determine the prediction quality of the model. Be aware that it is crucial to maintain the independence of the test set. For this purpose, no evaluation and model testing must be done using the test set. Otherwise, decisions would be made on basis of the performance on the test set. This would result in biased results and not comply with a scenario, where the model needs to make predictions on unseen data.

The processes we introduce in this thesis are more complex than the shown high-level machine learning roadmap. However, all methods we introduce can be broken down into elementary parts, which show the similarity to the shown high-level process.

2.3. Bias and Variance

Hastie et al. 2009 Chapter 2.9 describes how model fitting issues arise in regards to over- and underfitting and is considered as a source for this Chapter. To further investigate these issues, we need to inspect the three elements of the expected error of a model. Geman et al. (1992) identified the bias and variance decomposition, which is a powerful tool for analyzing supervised learning scenarios. Originally, this theory came up when neural networks were analyzed. However, it applies to all machine learning models. The conventional formulation of the decomposition calculates the expected error of a model as the sum of three quantities:

- **Bias** measures the error of the learning algorithm's average guess and the target
- **Variance** measures the volatility of the learning algorithm over similar data sets
- **Noise** is the lower bound on the expected error of any learning algoritm

Hence, we can define the expected error as

$$Err(X) = bias(X) + var(X) + noise(X).$$

Firstly, the bias determines the how well the average guess of the model fits the target. The bias can also be interpreted as the shape of the learning algorithm. Secondly, the variance measures how much the same learning algorithm differs over similar data sets. A high variance means that the models differ much from each other, resulting in a bad generalization of the model and unstable predictions. Lastly, the noise is not depending on the model, since it arises from noise in the data. Therefore, we not further investigate this element. The ideal model delivers stable and accurate predictions, hence the model's variance and bias are low.

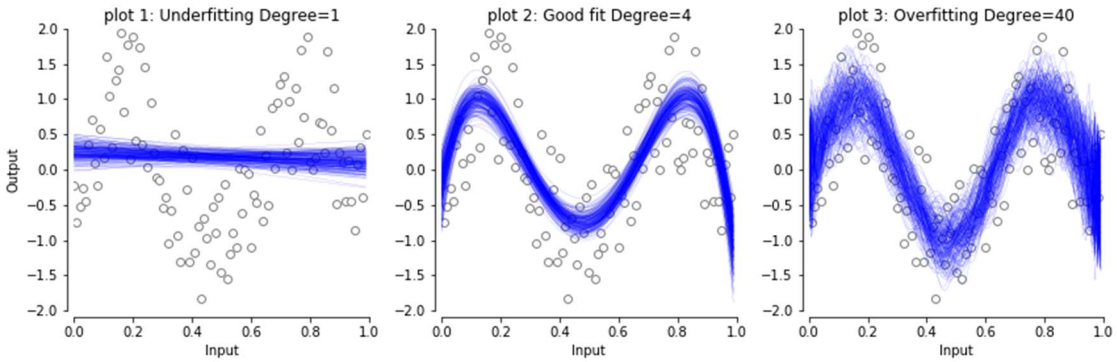

Figure 2.2: Examples of model fitting. Inspired by Louppe (2014, p. 58). We fit 200 polynomials for each level of degree to the artificial, sinus shaped data.

To illustrate the over- and underfitting issues, we consider a sinus shaped data set and fit polynomials of different degree to the data. The results are shown in Figure 2.2, where we consider polynomials of three different degrees which are fit to artificial, sinus shaped data.

In plot 1, we have fitted polynomials of the degree 1, which are essentially linear regressions. As we can observe, the function does not fit the shape of the data due to the low complexity of the polynomials. The sinus function cannot be recognized from the polynomials' shapes, which means that the bias of the polynomials is high, since the average guess's error is high. However, the polynomials do not differ a lot from each other, indicated by the similar shape and location. Hence, the variance is low. The high bias and low variance of the model indicate an underfitting scenario. A solution may be

to increase the degree of freedom of the model, which would result in a more flexible model and better fit to the given data.

Plot 3 shows the opposite scenario to the first plot. Here, we use polynomials of the degree of 40 and repeat the procedure. The resulting individual polynomials differ a lot from each other, which indicates a high variance. However, the sinus shape can be observed by the shape of the polynomials, resulting in a rather low bias. We deduce that the individual polynomials fit the data points too closely, which makes the polynomials represent the data set's noise as well. An example for noise in data are outliers, which do not fit in the overall patterns of the data. A model with high variance and low bias indicates an overfitting scenario. To counteract we can add restrictions to the mode, reduce its flexibility or increase the amount of data. However, the latter option is often expensive or just not feasible.

The polynomials shown in plot 2 have a degree of 4 and fit the given data well. We can clearly see the sinus shape of the underlying data samples; therefore, bias is low. Also, the variance is much lower than in the overfitting scenario since the polynomials are more alike and closer together. The combination of low variance and high bias is what we want to achieve in the machine learning scenarios. Hence, we consider the polynomials in plot 2 a good fitting function to the given data (Louppe 2014).

Due to the negative correlation of bias and variance, their relation is called the bias-variance-trade-off. A rather low bias makes the variance increase, and the other way around. This makes us consider their relationship as a trade-off. To obtain a low expected error, we need to find a good balance between bias and variance during the model selection.

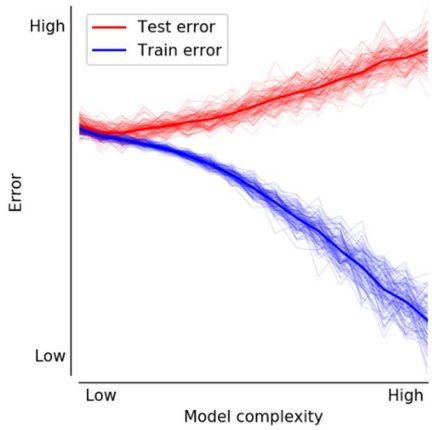

Figure 2.3: Error on train and test set over model complexity. Inspired by Hastie et al. (2009, p. 38). We consider 100 models for each level of complexity.

Plotting the errors of the train and test set over the model complexity illustrates the bias variance trade-off separated for training and testing. Figure 2.3 shows such a plot using 100 models for each level of complexity. A lower complexity reduces the variance and increases the bias, a more complex model triggers opposite effects. Notice how the models with high complexity achieve an exceptionally low error score on the train set. This is due to the very close fit of the model to the training data, which enables small errors. The slight changes of the test make the rather complex models having large errors on the test set.

Models with a rather low complexity have a larger train error, since the models are less flexible and do not fit the training data that closely. However, the ability of less complex models to generalize is better,

hence the test error is lower compared to complex models. There is a perfect level of complexity. This is where a lower complexity would increase train and test error and higher complexity only reduces the train error. This is the desired level of complexity and illustrates the break-even point. We conclude, improving the train score of the model must always be cross checked with the scores on the test set (Zhou 2012).

2.4. Models

We have shown in the previous chapters how the credit scoring task belongs to the category of supervised learning. More specifically, credit scoring considers two possible target characteristics, which are that the borrower defaults or not. Hence, it is a binary classification problem. This limits our choice of suitable learning algorithms to classification models. With these algorithms we could model data with J classes $J \epsilon \mathbb{N}$. If we would model a continuous target, we had to consider regression methods.

For the investigation of classification models, we choose the Logistic Regression to begin with. It is a relatively simple model, but widely used in the application of credit scoring (Lessmann et al. 2015). Furthermore, we want to go into more detail with different ensemble models, due to their throughout positive results in credit scoring applications, as we figured in Chapter 1.2.

2.4.1. Logistic Regression

The Logistic Regression model has been developed due to the need to model linear functions in X. We to determine the probability of a data sample $\boldsymbol{x}$ belonging to one of the J classes. The advantage of the Logistic Regression is the use of the sigmoid function, which ensures the probabilities stay in [0, 1] and sum up to one (Hastie et al. 2009). Therefore it suits the prediction of a categorical variable, such as the target variable in credit scoring (Hand, Henley 1997). Even though the Logistic Regression is a relatively simple model, research has shown great results for credit scoring, when compared to a more complex neural network for the average case (West 2000). Therefore, we further investigate the elements of the Logistic Regression. The sigmoid function is defined as

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

and is applied to the linear combination of X. Here z is the output of the addition of the P features and their weights. Hence, we can calculate z for one data sample $\boldsymbol{x}$ as $z = \boldsymbol{w}^T \boldsymbol{x} = w_0 + w_1 x_1 + \cdots + w_P x_P$ (Raschka, Olson 2015). Analysing the individual parts of the model, the confusing name of the method becomes clearer, since the Logistic Regression is simply a linear regression into a logistic function (Richert, Coelho 2013).

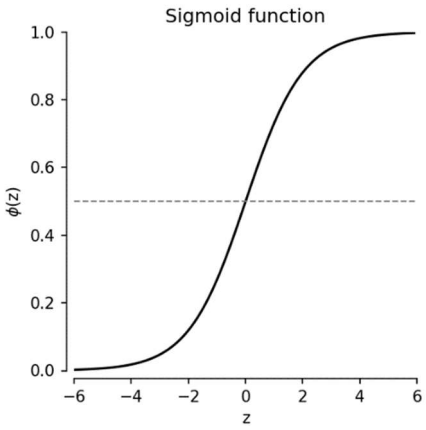

Figure 2.4: The sigmoid function. We consider $\phi(z)$ for values of z in [-6, 6].

The function plotted in Figure 2.4 shows the characteristic shape of the sigmoid function. Clearly, we see that all values of $\phi(z)$ are in $\phi(z)\epsilon[0,1]$. This emphasizes the use of the function for classification tasks.

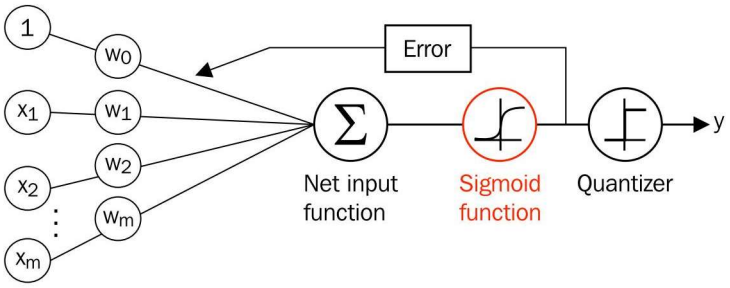

Figure 2.5: Logistic Regression process (Raschka, Olson 2015, p. 58)

The entire Logistic Regression process is visualized in Figure 2.5. The fitting of a Logistic Regression consists of multiple iterations, where the error of the sigmoid function's outcome $\phi(z)$ and the true target modifies the weights, with the goal to minimize the resulting error. After a defined number of iterations, the model is fit to the data and ready to provide predictions. For the prediction, the weights are applied to the values of a data sample $\boldsymbol{x}$, where we want to predict the target. Again, we derive $z = \boldsymbol{w}^T\boldsymbol{x}$ and apply the sigmoid function, resulting in a probabilistic estimate of the data sample. The quantizer is transforming the outcome of the sigmoid's function in a prediction, where the score $\boldsymbol{p}$ is the sigmoid's outcome, using a defined threshold c. See Chapter 2.1 for the transformation of $\boldsymbol{p}$ in $\widehat{\boldsymbol{y}}$.

According to BISHOP (2006) the logistic likehood function for two class $\boldsymbol{y}\epsilon\{0,1\}$ and n data samples is defined as

$$L_{Logistic} = \frac{1}{N}\sum_{i=1}^{n}(y_i \log(p_i) + (1-y_i)\log(1-p_i)).$$

Since the optimization of the loss is implemented as a minimizing problem, the cost function uses the negativized logistic loss function. Additionally, a regulation term has been added, which limits the weights and influences the flexibility of the model. This allows to alter the model's flexibility and change the bias-variance decomposition.

2.4.2. Decision Tree

This chapter is based on ZHOU (2012) Chapter 1.2.2 and HASTIE ET AL. (2009) Chapter 9.2. Here, we want to investigate the decision tree model. Since credit scoring is a classification task, we merely consider decision trees for classification. However, altering the decision tree slightly makes it also suitable for regression tasks.

Decision tree algorithms use a divide-and-conquer method to split the train set D recursively into two smaller subsets D^L and D^R, curtailing the number instances of different classes in the subset. Each split of the data is done through a data test, represented by a node t in the decision tree. Such a data test consists of a feature and a value, which determine the way the data is split up. At the point where no more data tests are conducted, a leaf is created, representing a specific class. Each decision tree has a root node, where the growth of the tree begins. From there on, branches of nodes lead the data splits to their leaves.

During the prediction of data samples, the data tests applied to data samples. Hence, the data sample follows a specific path in the decision tree. Once the data sample has reached a leaf, the data sample is assigned to the majority class of data samples in the respective leaf.

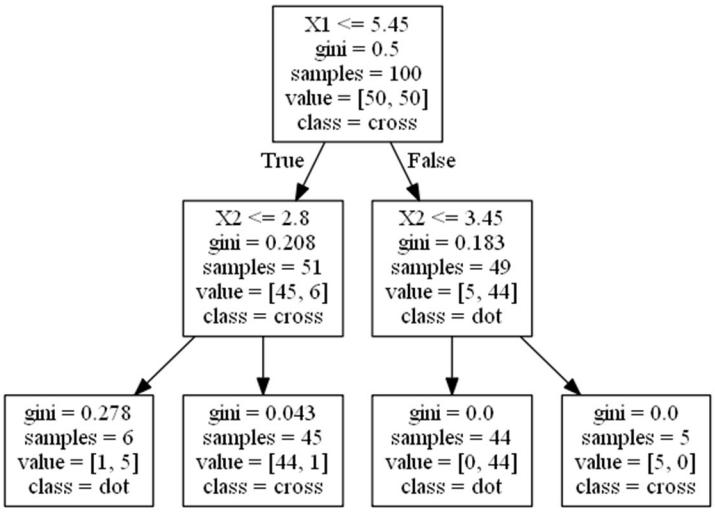

Figure 2.6: Example of a decision tree.

For illustration a decision tree is shown in Figure 2.6. A great advantage of single decision trees is the simple interpretation of the decision path. Especially if measures need to be derived from the tree, this comes in handy even without previous experience with such models. The corresponding decision boundaries of this decision tree are plotted in Figure 2.7. Data samples being in the darker area are predicted as class 'dot', whereas samples in the brighter area are predicted as class 'cross'. Be aware that this is merely an illustrative example with a low complexity. Decision trees are capable of modelling data with arbitrary complexity. More complex models are deeper, meaning more splits are performed sequentially. This leads to the tree becoming larger and $|D^t|$ becoming smaller compared to $|D|$. On the opposite, trees with less sequential nodes are called shallow trees.

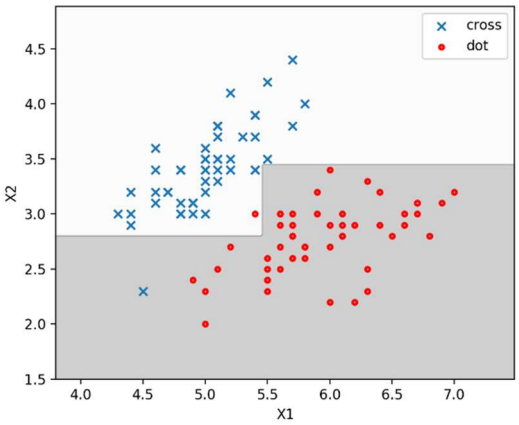

Figure 2.7: Decision boundaries of example decision tree.

As shown before, a learning algorithm must be fit to the data before predictions can be obtained. The training of a decision tree is called the grow of the tree, since the development of the splits occurs here. The tree is grown while a stopping criterion is not matched.

Algorithm 1: GrowTree CART Algorithm according to (D' Angelo 2016, p. 42)

Input: Training data D

Output: DecisionTree

Process:

```
1.  while Stop() is not true do
2.      (D^L, D^R) = Split(D)
3.      GrowTree(D^L)
4.      GrowTree(D^R)
5.  end
```

The high-level pseudo code of the growth of a decision tree using the CART algorithm is shown in Algorithm 1. According to LOUPPE (2014), as well as the pseudo code in Algorithm 1, the growth of a tree consists of two elements: splitting and stopping. We will further investigate these two elements in the following chapters.

2.4.2.1. Growing a tree: Splitting

For growing the tree, it must be decided by some measure how the node should split up the data. According to BREIMAN ET AL. (1984) this is called the impurity measure, evaluating how good the split of the node is. A purer node delivers better results for prediction, since it consists of a more imbalanced number of samples belonging to different classes. A split consists of the selection of a certain feature and a value. The tree is supposed to choose the split that delivers the purest nodes, according to the defined measure. Therefore, the approach of growing a tree is to iteratively split nodes into more pure nodes, until some stopping criteria is met.

Over the years, algorithms with different measures of the purity of D were developed. The ID3 algorithm introduced by QUINLAN (1986) uses the *entropy* to determine the pureness of a train set D. We define $P_j = P(j|D)$ as the probability of choosing a sample of class j in D (Reh 2017). For the train set D and J classes, the entropy Ent is defined as

$$Ent(D) = -\sum_{j=1}^{J} P_j \log P_j.$$

Another approach is the CART algorithm (Breiman et al. 1984) which uses the *Gini index* for determining the pureness of D. The Gini index indicates how often a randomly chosen object is assigned incorrectly (D' Angelo 2016). Using the defined variables, the Gini index is defined as

$$Gini(D) = 1 - \sum_{j=1}^{J} P_j^2.$$

To use these impurity criteria to determine the best split, the information gain is introduced. When splitting D into subsets $D_1, \dots, D_k$, the information gain G of D and one of the subsets $D_1, \dots, D_k$ can be determined with

$$G(D; D_1, \dots, D_k) = I(D) - \sum_{i=1}^{k} \frac{|D_k|}{|D|} I(D_k),$$

where the impurity I can be determined by the Gini or the entropy, hence $I \epsilon \{Ent, Gini\}$ (Zhou 2012). The feature-value combination with the highest information gain is selected for the split. For the credit scoring task, we have a binary target $\boldsymbol{y} \epsilon \{0, 1\}$, hence $J = 2$. Over different values of $P(y = 1|D)$ the introduced impurity indexes are shown in Figure 2.8.

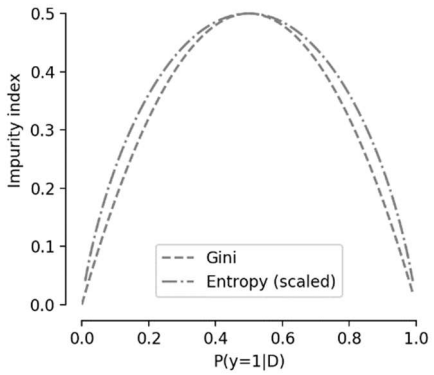

Figure 2.8: Impurity indexes comparison of Gini and Entropy. We consider J=2.

Clearly, we can see the maximum impurity being at $P(y = 1|D) = 0.5$ for both indexes. The minima are at $P(y = 1|D) \epsilon \{0, 1\}$. This is reasonable, since the highest chance of a misclassification exists, if classes are perfectly mixed in D. Hence, the probability to choose the incorrect sample would be 0.5. Since both indexes have almost the same curve, it does not make a large difference which is chosen for a decision tree. One should rather focus on other parameters of the decision tree.

An important parameter for the growth of a decision tree is the maximum number of random features considered when selecting the best feature-value combination. To analyse how the parameters are implemented in the libraries used for the implementation, we refer to elements of code like `this`. The implementation of the decision tree we use during application is the `DecisionTreeClassifier` from the module `sklearn.tree`. In this module, the number of maximum features to consider is implemented as the `max_features` attribute.

2.4.2.2. Growing a tree: Stopping

This chapter is based on LOUPPE (2014) Chapter 3.5. When growing the tree, one must determine measures for when to stop the growth. The procedure of growing a tree already implies two stopping criteria:

- If all values of the targets in the node t are homogenous. For a classification problem this means all samples in t belong to the same class.
- If the input variables are all locally constant in the data of t.

Growing a tree fully, without any additional stopping criteria, would result in a complex model, which fits the train set well. However, it generalizes poorly on the different data. The reason is that the model may have overfit the given data, as shown in Chapter 2.3. We use early stopping criteria to prevent the tree from becoming too complex. Another reason to make use of the early stopping criteria is the need to reduce the computational costs of the algorithm, where the size of the tree is reduced. Together with the two previously mentioned criteria defined by the algorithm itself, the growth is stopped if any of the criteria is met. The additional early stopping criteria and their implementation in the `DecisionTreeClassifier` are:

- Set t as terminal node if it contains less than N_{min} samples (`min_samples_split`).
- Set t as terminal node if its depth d_t is greater or equal to a defined maximum d_{max} (`max_depth`).
- Set t as a terminal node if the total decrease of impurity is smaller than a fixed threshold (`min_impurity_decrease`).
- Set t as terminal node if there is no split so that the child nodes both count at least N_{leaf} samples (`min_samples_leaf`).

Another strategy for reducing the chance of overfitting is the so-called *post-pruning*. Here, the tree is fully grown and afterwards *pruned* to reduce the size of the tree. This is done by sequentially removing nodes from the tree and evaluating the predictions on an additional validation data set. Once further pruning does not decrease the validation error, the optimal tree is found. In *post-pruning*, the tree is *prune* after it is fully grown. Using an early-stopping parameter is also called *pre-pruning*.

2.5. Ensemble methods

Ensemble methods a set of approaches, which uses multiple models and combines their outcome in some way. These models are called the base models, since they form the ground for the ensemble's prediction. The combination of the base models' predictions creates results with a higher accuracy than a single model could have achieved. We divide the ensemble methods into two subsets, the homogenous and the heterogenous ensembles. Homogenous ensembles use one type of learning algorithm, whereas heterogenous ensembles use different types of learning algorithms (Zhou 2012).

The consensus of the multiple base models is generated by some voting mechanism (Han et al. 2012). Figure 2.9 shows the general concept of homogenous ensemble learning. For $m = 1, \dots, M$ base models a distribution D_m is taken from the training data D. Each base model h_m of the learning algorithm $\mathcal{L}$ is trained on its own distribution D_m. Since the M distributions of D are diverse, the resulting models are diverse as well. During the prediction, each of the base models is using the same

data samples as input and creates its individual output. The M predictions of the individual base models are then combined as the predictions of the ensemble model.

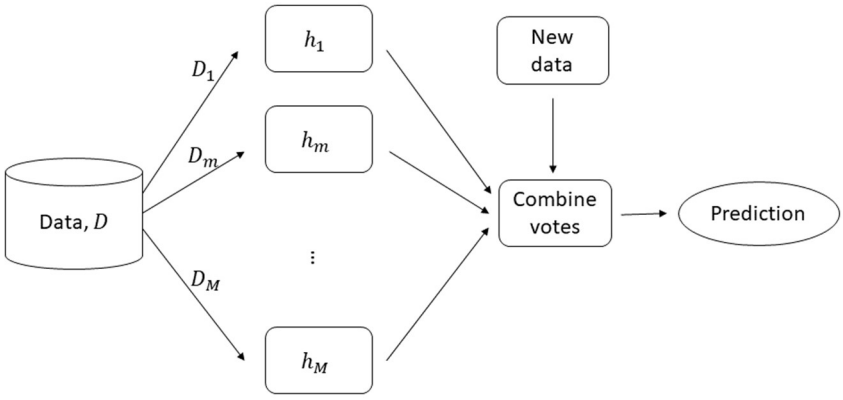

Figure 2.9: General architecture of ensemble methods. Inspired by HAN ET AL. (2012, p. 378).

It was also shown that ensembles work especially well if the base models have errors in different parts of the instance space. Many algorithms use feature selection to create diverse base models. In contrast to normal feature selection methods, where the goal is to find the most relevant subsets of D, ensemble methods try to find subsets creating disagreement among the base models (Tsymbal et al. 2005). As a result, not all ensemble methods necessarily need accurate base models. The integration of low accuracy base models has shown positive outcomes. A low accuracy base model is called a *weak learner*. Generally, all kinds of learning algorithms can be used as base model of an ensemble, as long as it suits the requirements of the overall prediction task. ABELLÁN, CASTELLANO (2017) conducted a comprehensive comparison of different combinations of base predictors and ensemble methods using credit data. They figured that the use of an imprecise base model improves the result in comparison to a more complex and precise model.

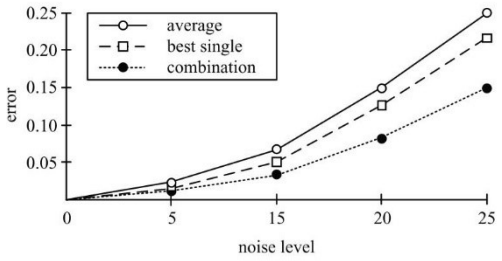

Figure 2.10: Advantage of the ensemble methods over single models against the noise level. (Zhou 2012, p. 17).

The plot shown in Figure 2.10 shows how ensembles have lower errors than the best single model. This effect rises with an increase of the noise level in the data. One of the drawbacks of ensemble methods is the larger computational effort. The computational costs grow linearly with the number of base models, since M base models must be considered during fitting and predicting. Additionally, the base models' predictions need to be combined. However, ZHOU (2012) argues that the computational costs are not much higher for an ensemble compared to a single model. As reason he points out, that single learners need to be evaluated many more times than ensembles for model selection. From an economic perspective the improvement of the prediction quality does not always legitimate the increase in computational costs of a complex model (Harris 2015). Therefore, one must know the

requirements of the modelling task upfront and be aware of the task's focus to find a solution within the given resources.

The voting algorithms of ensemble methods can generally be divided into two subtypes: those that change the distribution of the train set according to the previous base model's performance (*boosting*) and those that do not (*bagging*) (Zhou 2012). In the following chapter the methods are explained in more detail. Furthermore, we investigate the heterogenous base ensemble method *stacking*.

2.5.1. Boosting

Boosting is an ensemble method based on the work of KEARNS (1988). Together with ZHOU (2012) Chapter 2 it forms the source for this chapter. The idea of boosting is as follows. Assume to have multiple weak models, which can give good results in some specific areas of the data set, but are inaccurate on the other areas. Also, the models' predictions are diverse. The first base model h_1 is obtained by applying the learning algorithm $\mathfrak{L}$ on the data set D. As expected, the classifier performs well merely in a specific area of the data set. We can now derive a new distribution D' from D. D' depends on the first base model h_1 and focuses the data samples of bad performance quality. For the second base model h_2, we apply the learning algorithm $\mathfrak{L}$ to the new distribution D'. Hence, h_2 focuses on data samples where h_1 has performed poorly. This procedure is continued for all M base models, where each base model focuses on the errors of the previous base model. By combining the results of the base models, the boosting ensemble ideally performs well over the entire data set D. Therefore, a nearly perfect performance can be achieved in an ideal environment.

Algorithm 2: General Boosting Algorithm according to (Zhou 2012, p. 24)

Input: Train set $D = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_1)\}$;
Base learning algorithm $\mathfrak{L}$;
Number of base models M;
True targets $\boldsymbol{y}$;

Output: Boosting predictor H;

Process:

1. $D_1 = D$; #initialize the first distribution
2. **for** $m = 1, \dots, M$:
3. $h_m = \mathfrak{L}(D_m)$; #create base model from distribution
4. $\varepsilon_m = P_{\boldsymbol{x} \sim D_m}(h_m(\boldsymbol{x}) \neq \boldsymbol{y})$; #calculate error
5. $D_{m+1} = adjust\_distribution(D_m, \varepsilon_m)$;
6. **end**
7. $H(\boldsymbol{x}) = combine\_outputs(\{h_1(\boldsymbol{x}), \dots, h_M(\boldsymbol{x})\})$

We show the general boosting process in Algorithm 2 on a high level. Since we are rather interested in the idea of a boosting algorithm, the elements $adjust\_distribution$ and $combine\_outputs$ functions are not shown in detail. These two elements depend on the chosen type of boosting algorithm.

A popular ensemble algorithm applying boosting is the `AdaBoost` algorithm (Freund, Schapire 1996). We use this algorithm in the application part of this thesis. Here the $adjust\_distribution$ and $combine\_outputs$ functions are defined by the algorithm. The distribution of a base model is derived

by calculating the exponential loss of the previous base model. Hence, large errors have a high influence on the new distribution. The results of the base models are combined by the additive weight combination

$$H(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m h_m(\boldsymbol{x})$$

with

$$\alpha_m = \frac{1}{2}\ln\left(\frac{1-\varepsilon_m}{\varepsilon_m}\right).$$

Here α_m is a weight assigned to the base model h_m and determined in a way that the overall exponential loss is minimized. We show the weight function determining α_m of the `AdaBoost` algorithm for a more intuitive understanding of the weight calculation.

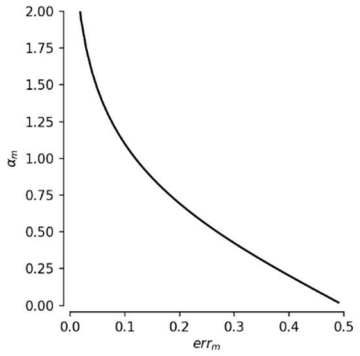

Figure 2.11: `AdaBoost` *weight calculation for the combination of base models. We consider values of* ε_m *in* $\varepsilon_m \epsilon [0, 0.5]$.

We show the relation of err_m and the resulting weight α_m for a base model in Figure 2.11. It can be observed, how base models with larger errors are have less impact on the overall prediction of the ensemble. Hence, the AdaBoost algorithm focuses on accurate models during the combination of base model predictions. One should also notice, that a base model with err_m >0.5 is ignored during the computation within the AdaBoost algorithm. Hence, we merely plot α_m for $\varepsilon_m \epsilon [0, 0.5]$.

It has been observed that boosting takes its benefit mostly from reducing the bias and reducing the variance as well (Bauer, Kohavi 1999; Zhou 2012). QUINLAN (1996) applied boosting with decision trees as base models and figured that the ability of the ensemble to generalize does not decrease with the number of base models.

However, the nature of the `AdaBoost` algorithm comes with downsides as well. Incorrectly predicted instances gain a large impact in the following distribution, due to the exponential influence of incorrectly data samples. Furthermore, this forces the algorithm to concentrate on such areas and makes it sensitive to noise in the data. Later versions of boosting algorithms solved this issue by depressing the maximum weight of a data sample.

2.5.2. Bagging and the Random Forest

Bagging has been discovered by BREIMAN (1996) and stands for ***b**ootstrap **agg**regation.* Bootstrap means each random distribution D_m^{boot} for the M base models is sampled from D with replacement.

Hence, D does not change while sampling and all D_m^{boot} are based on the same data. The aggregation of the bagging algorithm is done by a majority vote of the base models.

We consider ZHOU (2012) Chapter 3 as a source for this Chapter. In contrast to the previously shown boosting, bagging does not rely on the performance of the previous base model when determining a base model's distribution. Again, we consider M base models for the ensemble. The taken distribution D_m^{boot} has the same number of features as the original data, but only contains a random subset of data samples of D (Abellán, Castellano 2017). Each base model h_m is trained with its own distribution D_m^{boot}, resulting in diverse base models. The bagging schema is an ensembling method, which performs well when classifiers are accurate and unstable.

Algorithm 3: Bagging Algorithm according to (Zhou 2012, p. 49)

Input: Train set $D = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x_n}, y_1)\}$;
Base learning algorithm $\mathfrak{L}$;
Number of base models M;
True targets $\boldsymbol{y}$;

Output: Bagging predictor H;

Process:

1. **for** $m = 1, \dots, M$:
2. D_m^{boot} of D; #get a bootstrap sample from D
3. $h_m = \mathfrak{L}(D_m^{boot})$; #create base model from distribution
4. **end**
5. $H(\boldsymbol{x}) = argmax \sum_{m=1}^{M} 1_{\{h_m(\boldsymbol{x})=\boldsymbol{y}\}}$

Note that the fifth line of the bagging pseudo code only applies to classification problems. Regression problems cannot make use of the voting mechanism in that manner. Bagging for regression tasks uses averaging as a way to combine the predictions.

As for boosting, research has shown that the most accurate predictions using bagging can be reached when the base models are diverse (Tsymbal et al. 2005; Dietterich 2000). This results in a better ability of the ensemble to generalize, due to a decrease of the variance of the ensemble while keeping the bias on the same level or having a slightly increasing bias. LOUPPE (2014) comes to similar findings and recommends to use rather strong randomization techniques to derive the bootstrap distributions. Hence, the resulting base models are more diverse, and the variance of the ensemble is reduced.

Following up, the *Random Forest* is investigated, which is a specific model using bagging. The Random Forest was first introduced by BREIMAN (2001), who describes the method as a combination of his previous work, bagging and decision trees. Hence, the name Random Forest. Using decision trees as base models turns out to be a good choice, since they provide a high accuracy as well as diversity, both are needed for using the potential of a strong randomized ensemble, like bagging.

Generally, the accuracy of a Random Forest is comparable to a `AdaBoost` algorithm. However it is more robust towards outliers (Han et al. 2012).

2.5.3. Stacking

Another way of combining the results from different models is stacking. It was first introduced by WOLPERT (1992) who defines the aim of stacking as achieving the best possible capability to generalize. In contrast to the previously mentioned boosting and bagging, which are homogenous ensemble methods, stacking is usually used as a heterogenous ensemble method and combines different types of base learning algorithms.

This approach consists of models on two sequential levels. On the first level multiple base models are trained on the original data, as seen in the previous approaches. However, stacked ensembles can use different types of base learning algorithms. On the second level, an additional model, the meta-model, is trained on a new dataset. The meta-model uses the output of the base models and the targets from the original data as input. Like any other model, the meta-model learns from the relation of the data and the targets. Thus, the stacking approach uses learning as a mean to combine the base models' outcomes (Zhou 2012). This may show in an improved prediction quality compared to more simple combination techniques. In contrast the boosting and bagging, stacking does not alter the base model's distribution. All base models are trained on the full train set D.

Algorithm 4: Stacking Algorithm according to (Zhou 2012, p. 84)

Input: Train set $D = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_1)\}$;
Base models $\mathfrak{L}_1, \dots, \mathfrak{L}_M$;
Meta-model $\mathfrak{L}$;
Number of base models M;

Output: Stacking predictor $H(\boldsymbol{x})$;

Process:

1. **for** $m = 1, \dots, M$:
2. $h_m = \mathfrak{L}_m(D)$; #train each base model on the train set
3. **end**
4. $D' = \emptyset$; #create meta-model data set
5. **for** $i = 1, \dots, n$: #for each data sample
6. **for** $m = 1, \dots, M$:
7. $z_{im} = h_m(\boldsymbol{x}_i)$; #obtain output from base model
8. **end**
9. $D' = D' \cup \left((z_{i1}, \dots, z_{iT}), y_i\right)$; #create new data set with the #outcomes of the base models #and the targets of the #original data
10. **end**
11. $h' = \mathfrak{L}(D')$; #fit the meta-model to D'
12. $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), \dots, h_M(\boldsymbol{x}))$

We show the general stacking procedure in Algorithm 4. The the algorithm can be split into three parts. In line 1-3, each base model is fit on the data set D. In line 4-10 the output of the base models on the original train set D is obtained, which are combined in a new data set D'. In line 11-12 the meta-model is fit to the new data set D', consisting of the output of the base models.

The output of the base models can be the predictions $\hat{\boldsymbol{y}}$. However, TING, WITTEN (1999) have shown that the use of the probabilistic estimates $\boldsymbol{p}$ can lead to better prediction quality of the stacked model, since the confidence of the base model's prediction is included in $\boldsymbol{p}$.

2.6. Sampling techniques

We consider HE, GARCIA (2009) as a major source on sampling techniques, which are used to improve the prediction quality on data sets with an unequal class distribution. Therefore, sampling techniques merely apply to classification tasks. The term imbalanced data is usually used for data with significant or extreme cases of class imbalances. These are often tasks where the minority class is of higher importance than the majority class. Imbalanced classes appear often in biology or in medicine. An example is the classification of patients into two groups, those that have cancer and those that do not. Here, classifying a non-cancer patient as cancerous would have less severe consequences, than classifying a cancer patient as non-cancerous. The cancer patient may not be treated correctly.

A similar imbalanced class distribution applies to the credit default task, which we are conducting in this thesis. Over the given period most borrowers do not default. This is great for the financial institution but makes the machine learning task rather difficult. To successfully model such an imbalanced data set, the predictive quality of the minority class needs to be especially high. However, the majority class must not be neglected during model selection.

We use decision trees to illustrate the difficulties which arise when working with imbalanced class distributions:

1. The sequential splitting of the dataset results in smaller numbers of the minority class in the leaves and fewer leaves describing the minority class. Hence, the confidences for the minority class are small.
2. Patterns of the minority class that are based on the conjunction of different feature conjunctions may not be learned by the decision tree, especially in a high dimensional problem.

Typically, the use of sampling methods for imbalanced data modify one of the classes in a way that the resulting data set is balanced. It has been shown that the use of sampling techniques to obtain a balanced dataset improves the overall prediction quality (Weiss, Provost 2001; Laurikkala 2001). Although, there are cases where the original data set provided better results. The use of such sampling methods makes it possible to use models, which are not especially tuned to perform well on imbalanced data sets. Note that, during the model selection, we can apply sampling techniques merely to the train set.

There are two types of sampling techniques to balance out an imbalanced data set D:

- *Over-sampling* adds artificial data samples to the existing minority class
- *Under-sampling* reduces the number of data samples of the majority class

In the simplest application, both these versions can be applied randomly. Hence, random data samples of the minority class would be duplicated (over-sampling) or random data samples of the majority class are removed from the data set (under-sampling). These random methods are called **R**andom **O**ver-**S**ampling (ROS) and **R**andom **U**nder-**S**ampling (RUS). However, ROS is likely to lead to overfitting and

ROS removes important information from the data set. Therefore, we introduce more advanced methods of over- and under-sampling.

In terms of over-sampling, the **S**ynthetic **M**inority **O**ver-sampling **Te**chnique (SMOTE) has shown successful results in application (Chawla et al. 2002). SMOTE generates artificial instances on basis of similarities minority class data samples. An artificial instance is calculated like

$$\boldsymbol{x}_{new} = \boldsymbol{x} + (\hat{\boldsymbol{x}} - \boldsymbol{x}) * \delta,$$

where $\boldsymbol{x}$ is an existing data sample of the minority class, $\hat{\boldsymbol{x}}$ is a randomly selected k nearest neighbour of $\boldsymbol{x}$ and δ is a random number $\delta \epsilon [0, 1]$. The resulting artificial data sample $\boldsymbol{x}_{new}$ is located randomly on the straight line between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$.

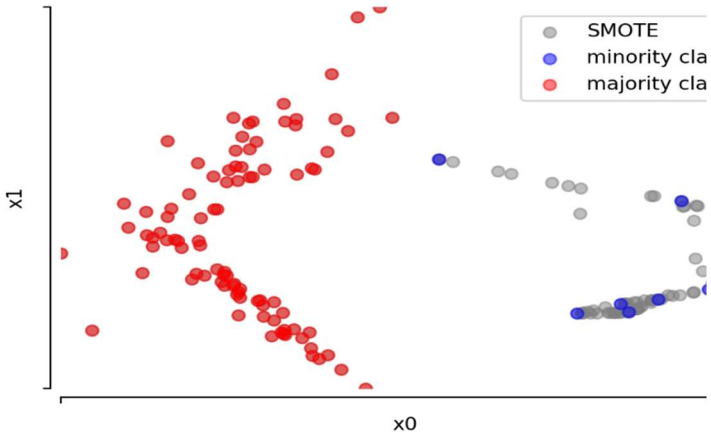

Figure 2.12: SMOTE over-sampling of the minority class. We consider plotting two-dimensional data samples. The artificial data samples created by SMOTE are shown in grey.

Figure 2.12 visualizes the artificially generated data samples by SMOTE in a two-dimensional feature space. The over-sampled data samples are located between the existing samples of the minority class. More dense areas in the minority class also lead to denser over-sampled data samples. Overall, the method seems to capture the attributes of the minority class relatively well and no outliers are created by SMOTE. However, for an increasing number of samples and dimensions, the computational costs for SMOTE are rather high. An issue is that the artificial data samples might be too unspecific to reflect the attributes of the minority class.

An advanced version is the Borderline-SMOTE introduced by HAN ET AL. (2005). The method aims to create new data close to the border between the classes. Particularly, Borderline-SMOTE considers the k-nearest neighbours of a random minority sample, like the regular SMOTE. Additionally, Borderline SMOTE considers the number of majority class instances within the k-nearest neighbours of $\boldsymbol{x}$. If there are exclusively data samples of the minority class, an artificial instance would not be beneficial, but rather add to the noise. Only if the number of majority class instances is within a previously defined range, $\boldsymbol{x}_{new}$ is created. Hence, it is supposed to strengthen the decision boundaries of the classifier in an area where misclassification is likely to happen.

Another over-sampling technique is **Ada**ptive **Syn**thetic Sampling (ADASYN) by HE ET AL. (2008). The method aims to use a density distribution to find the number of artificial samples that need to be created for each data sample of the minority. During the process, each minority class sample gets a weight assigned, which is adaptively changed to adapt to the distribution.

The mentioned techniques are often used in combination with data cleaning techniques to reduce the overlapping instances, which were created by the over-sampling technique. BATISTA ET AL. (2004) recommends using SMOTE with the data cleaning technique Tomek links or the Neighbourhood Cleaning rule.

Under-sampling techniques create a balanced data set as well. However, they use the inverse of the oversampling approach and reduce the number of majority class samples in D. A sophisticated method for undersampling is using unsupervised machine learning methods to merge majority class data samples. An example is the use of k-nearest neighbour classifier, which replaces the k nearest majority class data samples with a data sample in between the k samples. The method was proposed by ZHANG, MANI (2003) with multiple implementations, one of them is the NearMiss implementation.

2.7. Calibration

The score $\boldsymbol{p}$ allows to deduce the confidence of the classification model for a given data sample. Hence, a ranking of the confidences can be created. For some applications, this relative separation is sufficient information. In other applications, the score must reflect the probability of a data sample to belong to a class. For this purpose, we consider calibration techniques which are applied to scores, to improve the interpretability of the score as a probability. We consider ZADROZNY, ELKAN (2001) as source for this chapter.

For the prediction of credit defaults, it is crucial to not only know which credit is more likely to default than some other, but also to be aware of the probability that a credit defaults. Therefore, we desire a directly interpretable score. We define the estimated frequency

$$ef = P(y = 1|\boldsymbol{p} = v),$$

where v is a score $v \epsilon \boldsymbol{p}$. Here, we obtain ef for all scores $\boldsymbol{p}$ with $\boldsymbol{p} = v$. To visualize the relationship between the score and the default probability, we compute the estimated frequency ef against the binned scores in a calibration diagram. The use of bins allows the computation of ef for more samples, hence the resulting ef is more stable. The computed ef for a given bin presents the probability of an arbitrary data sample x_i with score p_i to belong to the class 1.

We will now present two calibration techniques, which can be applied to any model using the models' scores $\boldsymbol{p}$ and the true target $\boldsymbol{y}$. To apply calibration techniques, the scores of the model need to be sorted according to $\boldsymbol{y}$. Also, we need to consider a cross-validation to obtain calibrated scores for all data samples. The cross-validation technique is shown in Chapter 2.8.

Firstly, we investigate the isotonic regression (ISO), which was proposed by ZADROZNY, ELKAN (2002). The ISO uses pair-adjacent violators to find the best fitting stepwise-isotonic function, which minimizes the error of predicted and true targets. The score of a data sample p_i needs to be larger than the score of the previous data sample p_{i-1}. If this is not true, the two data samples are called a pair-adjacent-violators and the scores of the samples are replaced by their mean. Applying this process over all the data samples ensures that the scores are sorted in an ascending manner (Horn 2016).

Secondly, we can use the sigmoid function and apply it to the scores $\boldsymbol{p}$ (SIG). The resulting scores are supposed to reflect the probability better, since the relation of ef and score is more linear.

Interestingly, we can use the MSE as a measure for calibration quality. This application of the MSE was introduced by BRIER (1950). For comprehensiveness we define the *Brier* score as

$$Brier\ score = \frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2.$$

Note that a low Brier score implies a good calibration quality of the prediction, since the metric is defined as a loss.

2.8. Other machine learning techniques

This chapter deals with the explanation of further techniques used during the application of the data. First, a technique for the preprocessing of the data is described. The relative sizes of the different features are interpreted as different importances by some of the models, like the Logistic Regression. It is necessary to bring all features on the same scale before applying the model. A specific form of feature scaling is the normalization, where all values are projected on a scale $[0, 1]$, making all features having the same initial importance to the machine learning algorithm. However, the relative values within a feature stay the same. For each feature $\boldsymbol{x_j}$ in the data set we obtain the minimum and maximum value. Then, we apply each data sample x_{ij} in $\boldsymbol{x_j}$ to the normalization. According to XIA ET AL. (2017) the feature normalization is defined as:

$$x_{ij}^{normalized} = \frac{x_{ij} - \min(\boldsymbol{x_j})}{\max(\boldsymbol{x_j}) - \min(\boldsymbol{x_j})}.$$

Second, the dummy transformation, another processing step of the data, is investigated. Here, categorical features can be transformed into binary features. Using the original categorical data as input is not possible since the input must be numeric. Using values to encode the categories would imply a relation of the categories. Hence, the dummy transformation of categorical features must be considered. We refer to the dummy transformation as described by D' ANGELO (2016). Assuming $\boldsymbol{x_j}$ to be a categorical feature with the categories $\{c_1, \dots, c_p\}$, a dummy is given according to

$$dummy_i^{\boldsymbol{x_j}} = \begin{cases} 1\ if\ \boldsymbol{x_j} = c_i \\ 0\ if\ \boldsymbol{x_j} \neq c_i \end{cases}$$

which creates a new column in the dataset for each unique category in $\boldsymbol{x_j}$. For features with many unique categories this leads to an explosion of new features. The resulting dataset becomes less intuitively readable. However, from a computational point of view this transformation is efficient (Raschka, Olson 2015). The new features are named according to the represented category.

Third, for the model selection a technique is considered to create predictions for all data samples in the train set. This task is fulfilled by using the cross-validation (CV). We refer the definition of the CV by HAN ET AL. (2012). In an k-fold CV, the data set is randomly split into k exclusive *folds*. We iterate over all k folds, where the corresponding fold is used as test set. All other folds are for the training of the model. After fitting the model in all k iterations and creating outcomes for all k folds, we have obtained outcomes for all data samples in the data set. This makes the k-fold CV suitable to evaluate a model on the entire train set, without losing the independence of the test set.

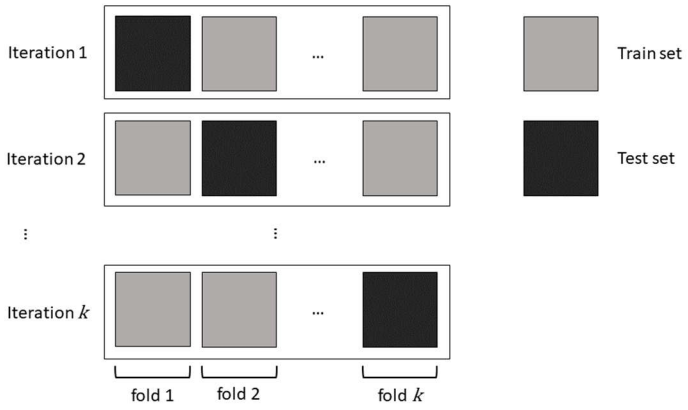

Figure 2.13: Example process of a cross-validation. Inspired by REH (2017, p. 32).

A special form of the CV is the stratified CV, where the folds are picked in a way, that all k folds contain approximately the same class distribution as the original data set. Especially for imbalanced data sets this is important. Hence, the stratified CV is also great for the task of credit scoring. From here on, we always consider a stratified version of the k-fold CV, when referring to a CV.

2.9. Metrics

After fitting the model to the data and obtaining outcomes, the quality of the prediction needs to be evaluated. Having a quantitative measure is especially important when comparing different types of models and to show improvements during the model selection. Choosing the wrong metrics for the present problem will result in uninformative results, which may even lead to wrong conclusions. The task of credit scoring was defined as a classification task, since $\boldsymbol{y}\epsilon\{0,1\}$. Therefore, only metrics for classification tasks are investigated. First, the number of correct and incorrect predictions is count:

$Positives\ (P)$ the number of correct predictions

$Negatives\ (N)$ the number of incorrect predictions

However, these metrics do not give information of which class the predicted data samples are. Note that for the following metrics, the definition of the positive and the negative class depends on the definition of the user. There are four possible outcomes, when comparing the true target $\boldsymbol{y}$ and the model's prediction $\widehat{\boldsymbol{y}}$ of a given data sample (Han et al. 2012):

$True-positives\ (TP)$ the number of correct positive predictions

$True-negatives\ (TN)$ the number of correct negative predictions

$False-positives\ (FP)$ the number of incorrect positive predictions

$False-negatives\ (FN)$ the number of incorrect negative predictions

We compare $\boldsymbol{y}$ and $\widehat{\boldsymbol{y}}$ for each data sample and simply count the number of the occurrences per metric. Obviously, the perfect classification model would only have correctly predicted data samples, therefore FP and FN would equal 0. We show the four introduced metrics in a *confusion matrix* (Raschka, Olson 2015) for a quick but comprehensive overview of the prediction quality.

		Predicted class	
		0	1
Actual class	0	TN	FP
	1	FN	TP

Table 2.1: Example of a binary confusion matrix. We consider $\boldsymbol{y}\epsilon\{0,1\}$, hence $J = 2$.

For a binary problem with two classes $J = 2$ and $\boldsymbol{y}\epsilon\{0,1\}$, we show an example of a confusion matrix and the assignment of the metrics in Table 2.1. The predicted target is shown in the columns and the true target is shown in the rows.

We can use the introduced metrics TP, TN, FP, FN for further calculations of prediction quality rates, instead to absolute numbers. Hence, they are independent from the number of data samples. Note that most of the presented metrics are known by more than one name. We refer to them as defined in HAN ET AL. (2012). The most intuitive is the percentage of correct and incorrect predictions on the data set:

$accuracy = \frac{TP+TN}{P+N}$ percentage of correct predictions

$error\ rate = \frac{FP+FN}{P+N}$ percentage of incorrect predictions

However, when the target variables are imbalanced, reaching an impressing accuracy and low error rate is not hard and not a good indicator for the classification performance. The incorrectly classified minority class would have a diminishing small contribution to the overall accuracy. Imagine a model, which always predicts the majority class. The accuracy of such a model would be high, though the performance on the minority class would not be acceptable, since no minority samples would be predicted at all. Therefore, other metrics are used as well, suiting the evaluation of imbalanced data sets better. Again, we refer to HAN ET AL. (2012) for the definition of the metrics:

$sensitivity = \frac{TP}{P}$ percentage of correctly predicted positives

$specificity = \frac{TN}{N}$ percentage of correctly predicted negatives

$precision = \frac{TP}{TP+FP}$ measure of exactness on the positive class

$recall = \frac{TP}{TP+FN}$ measure of completeness on the positive class

$F_1 = \frac{2*precision*recall}{precision+recall}$ harmonic mean of precision and recall

For the imbalanced class in this thesis we consider the precision, recall and the F1 score to evaluate the models' prediction qualities. Since the F1 is the harmonic mean of precision and recall, it is only high if both measures are high as well. If one measure is high, but the other is low, the resulting F1 would be rather low as well. This is due to the product of both measures being in the numerator and the addition in the denominator. Since we desire a model which has a good balance of precision and recall, the F1 score suits our requirements well.

2.9.1. Receiver-Operating-Characteristics curve

Another metric to evaluate the classification performance of a model is the **R**eceiver-**O**perating-**C**haracteristics (ROC) curve. It is especially useful for the comparison of different models. We describe the curve according to HAN ET AL. (2012). For this curve, the **T**rue **P**ositive **R**ate (TPR) is PLOT against the **F**alse **P**ositive **R**ate (FPR), where $TPR = sensitivity = \frac{TP}{P}$ and $FPR = 1 - specificity = \frac{FP}{P}$. For a binary target variable, the ROC curve visualizes the rate which the model can recognize positive targets correctly versus the rate at which it mistakenly identifies negative targets as positives.

Creating such a plot requires a model which can return not only the predicted class $\hat{\boldsymbol{y}}$, but the corresponding probability estimation $\boldsymbol{p}$, which we refer to as *score* as defined in Chapter 2.1. The score allows the predictions to be sorted according to the model's confidence of an instance belonging to a class. A variable threshold c defines all tuples as positive where $\boldsymbol{p} \geq c$, all other tuples are considered negative. For each unique probability in the tuples, the threshold is adjusted and the TPR and FPR is calculated for the given threshold. Therefore, we can also write TPR and FPR as functions $TPR(c)$ and $FPR(c)$ depending on the threshold c. Generally, the higher the TPR and the lower the FPR of a given point on the ROC curve, the better the performance (Baesens et al. 2003). For visualization, the TPR is plot against the FPR for all unique scores in a descending order, resulting in the in the typical shape of the ROC curve.

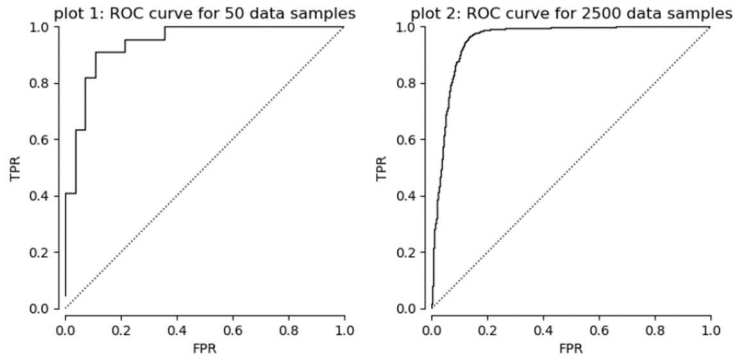

Figure 2.14: ROC curves for different number of instances. We show a ROC plot for 50 and for 2500 data samples. The dotted line presents the ROC curve of random guessing.

Figure 2.14 shows two ROC curves for different numbers of data samples. We notice a smoother plot of the ROC curve in plot 2. This is due the more unique $TPR(c)$ and $FPR(c)$ values which are plot on the diagram. A perfect classifier has a rectangular curve (Raschka, Olson 2015). A random guessing model would have a diagonal curve, since an increase in TPR has the same increase in FPR. We consider the **A**rea **U**nder the **C**urve (AUC) for quantitative comparisons of different curves. The AUC of the ROC curve is referred to as AUC-ROC. The perfect model has an AUC-ROC of 1, a random guessing model would score AUC-ROC of 0.5. Therefore, no classifier should have an AUC-ROC below 0.5 since this would indicate a prediction quality worse than random guessing. However, solely judging by the AUC-ROC does not deliver a comprehensive evaluation. A classification model could have a better AUC-ROC score than another classifier, but perform worse in a specific area (Fawcett 2003). In a binary classification setting, an intuitive interpretation of the AUC-ROC is that it provides an estimated probability that a randomly chosen data sample of the positive class is correctly classified higher than a randomly chosen instance from the negative class (Baesens et al. 2003).

2.9.2. Precision-Recall curve

A less frequently used measure to evaluate classification models is the **P**recision-**R**ecall (PR) curve. It shows the precision and the recall for a variable threshold c. Again, we can plot the values of the PR curve as functions c, resulting in the $precision(c)$ against $recall(c)$ for all unique scores of the model.

A major difference between the ROC curve and the PR curve is the nature of the baseline. While it is fixed with the ROC curve, meaning the classification models' performances over different data sets can be compared without adjustment of the curve, the PR curve baseline moves with the class distribution of the data set. Hence, comparing PR curves of different data sets does not deliver comparable results.

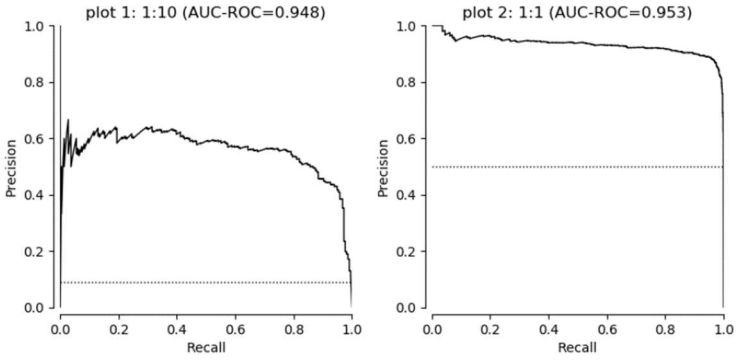

Figure 2.15: Comparison of two PR curves. We plot the PR curve for two different datasets with different class distributions. Their moving baseline is illustrated by the dotted horizontal line. We show the AUC-ROC of the same predictions in the header of each plot.

Figure 2.15 visualizes the behaviour of the moving baseline. We show the PR curves for two different datasets. Plot 1 shows a PR curve for a dataset with a class distribution of 1:10, plot 2 shows a PR curve for a data set with a balanced class distribution. The classification model and the number of total data samples remain unchanged. The PR curve shows significant differences between the performances on the different datasets. Clearly, we see that the prediction quality on the imbalanced data set in plot 1 is much worse than on the balanced data set in plot 2. However, the ROC-AUC alters only slightly and fails to capture the poor performance of the model on the imbalanced dataset. Only precision and recall reveal the performance difference on different class distributions (Saito, Rehmsmeier 2015). Since the credit scoring is a highly imbalanced problem to solve, we use the ROC curve in combination with the PR curve to evaluate classifier performance.

The ROC and the PR curve depend on the threshold, both are well suited for evaluating and comparing the performance of classification models without the requirement to select a threshold upfront. They help choosing the best threshold c for the classification task, according to the task's requirements. Nevertheless, since c itself is not shown in both plots, we cannot read the corresponding c of a given point in the curve. Hence, we also consider plotting the precision and recall against c for the comprehensive evaluation of the models.

3. Data and Application

To obtain a good understanding of the provided data set and the targets, the correlations need to be understood. For the credit scoring task this means for identify indicators for a default in the data. Being aware of the indicators will provide knowledge about the importance of given features, which can be used during the feature extraction. Therefore, the model has a more informative data set to obtain the predictions.

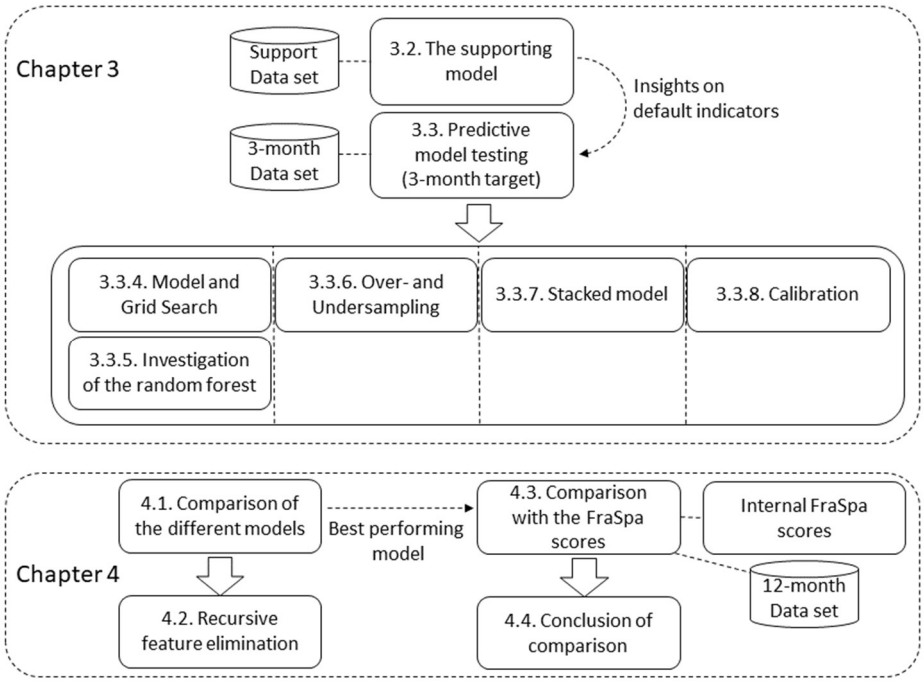

Figure 3.1: Structural overview of Chapter 3 and Chapter 4.

The overview of the elements of the application and data (Chapter 3) and the results (Chapter 4) is illustrated in Figure 3.1. We now present how we the original data set is altered and used for three different purposes.

First, we use the data to create a model, which is merely made for the investigation of the features as an indicator for a future credit default. We call this model the *supporting model* and present the findings in Chapter 3.2.

Second, we create a data set where the target is a 3-month future period. Hence, defaults within the next 3 months are considered in the target. For some specific features historical data is available and can be considered as a short timeseries. From the historical features the information is extracted through different aggregation functions. Since the supporting model gives insights, which features are good indicators for a future default, the focus for the aggregation methods can be set accordingly. On this data set the machine learning techniques are applied as shown in the theoretical part. The evaluation of the different techniques is shown in Chapter 3.3, where each subchapter contains one technique. Furthermore, the selected model-parameter combinations are compared to each other using the train and test set in Chapter 4.1. Additionally, the results of the recursive feature elimination are shown in Chapter 4.2.

Third, the technique with the best prediction quality is compared it to the scoring method of FraSpa. Therefore, a data set with a 12-month target needs to be considered. Unfortunately, the information from the historical features cannot be extracted due to limitations in the data. Hence, the 12-month data set is to contain less information than the 3-month data set. We compare the best model to the FraSpa scores in Chapter 4.3.

For the application, we use the programming language **python** and different libraries. Specifically, the library `numpy` is used to handle arrays, `pandas` to create data frames and apply functions over them and `matplotlib.pyplot` to visualize our results as plots. The machine learning models are used as implemented in `sklearn`, where essential modules for preprocessing and model selection are provided as well. Before further investigating the application of the models, which we have explored Chapter 2.4, the data source and the processing applied to it is shown in the following chapters.

3.1. Data

The data used for this thesis consists of credit customer data of FraSpa. All credits issued by FraSpa are considered, which are available in the data source. This includes consumer credits, mortgage credits and others. The borrowers are private persons as well as rather small companies. The source of the data is an internal database from the department of risk controlling. We consider three tables do build the data sets for the application:

- *person* consists of customer information based on the borrower's level
- *account* consist of customer information based on the level *account*
- *rating* consist of the internal ratings, as shown in Table 4.4, based on the level *person*

The tables *person* and *account* contain the information used for creating the features of the data set. Notice that the level *account* holds the information of the individual credits. It could also be called the *credit table*, but we refrain from doing so due to the consistency with the internal naming. *person* contains information of the borrower, *rating* contains the internal ratings of each borrower with a monthly frequency. We use the information from *rating* to create the default target. Since one person can have more than one account, a 1-to-n relationship exists here. This is important for the data processing, particularly when handling categorical features. Since the rating defined for each borrower, we need to aggregate all other information on the same level. Hence, the information from *account* must be aggregated on the borrower's level.

These three tables are used to create the data sets for all three purposes. Merely by changing the processing of the data, the resulting data set becomes suitable for the corresponding task.

3.2. Supporting model

With the supporting model indicators for a future default in the data are identified. This lets us focus on such indicators during the feature extraction of the 3-month data set. Nevertheless, the independence of the test set must not be lost, hence only the credits in the train set are considered in for the application. Another purpose of the supporting model is to compare the original data of the historical features to the trend over the time.

Six continuous features of the table *person* over all available historical periods are considered. Note that in for the present data one period is one month. Categorical features are not considered, due to their unlikeliness to change over different periods. The selected features for the supporting model are:

- *Feature 1*: Amount of overdraw
  The amount of money behind schedule. This field is only available if the credit is not payed back according to the schedule.
- *Feature 2*: Sum of credit volumes
  The sum of all credit volumes of the borrower.
- *Feature 3*: Balance of the account
  The current balance of the credit.
- *Feature 4*: Change in credit volume previous quartal
  The volume the credit has changed during the previous quartal.
- *Feature 5*: Change in credit volume previous month
  The volume the credit has changed in the previous month.
- *Feature 6*: Number of accounts overdrawn
  Total number of accounts overdrawn by the borrower.

Since the trend of the features was expected to be important to determine a default, a **R**olling-average **M**ean (RM) was applied to each of the features with a window of three periods. The result is a smoothened-out version of the timeseries. The process of creating the data set for the supporting model is:

1. Sort data samples according to the month.
2. Group data samples on the person number and on the type of feature.
3. Create rolling mean with $window = 3$.
4. Add default columns.

The resulting data includes the original values and their rolling mean of the six features for each period and for each borrower. A total of 12 periods are considered for the supporting model. This creates a total of 935865 data samples in the data set for the supporting model. An additional question is how the indicators change with the number of periods before the occurrence of the default. Therefore, three data sets with the targets {default in 1, default in 2, default in 3} are created. The target of the supporting model considers defaults in **exactly** {1, 2, 3} months. Contrarily, the data sets used for the prediction of future defaults consider the **next** {3, 12} months. The number of defaults as well as the default frequency of the supporting model data sets in are shown in Table 3.1.

Name	Number of defaults	Default frequency
default in 1	274	0.029%
default in 2	302	0.032%
default in 3	306	0.033%

Table 3.1: Default statistics of the supporting model.

On each of the three data sets, a `RandomForestClassifier` (RF), as implemented in `sklearn.ensemble`, was trained using a 3-fold CV. The importance of a feature is determined by the location of the nodes, which use the corresponding feature as data split. Data tests of a feature closer to the root of a decision tree are considered rather important.

Feature	type	mean	STD
1	ORIG	**0.200**	0.036
2	ORIG	0.046	0.005
3	ORIG	0.057	0.008
4	ORIG	0.046	0.005
5	ORIG	0.091	0.007
6	ORIG	0.087	0.014
1	RM	**0.217**	0.019
2	RM	0.047	0.009
3	RM	0.053	0.004
4	RM	0.044	0.011
5	RM	0.050	0.012
6	RM	0.062	0.008

Table 3.2: Feature importances of the supporting model. Over the three data sets, we show the average importance as mean and the standard deviation as STD.

In Table 3.2 the overall results of the supporting model are shown. Out of the twelve total features, two features stick out. Both the important features are of Feature 1. Interestingly, the RM is even more important than the original values. Additionally, the RM has a lower STD than the original value. Hence, the historical trend of the feature over the time is an important and rather stable indicator for a future default. Notice that Feature 1 is the only one with an importance ≥ 0.2 and all other feature importances are <0.1. Hence, only the minority of the features is indicating a future default well. The second most important feature is Feature 6. Both, Feature 1 and Feature 6 contain overdraw information.

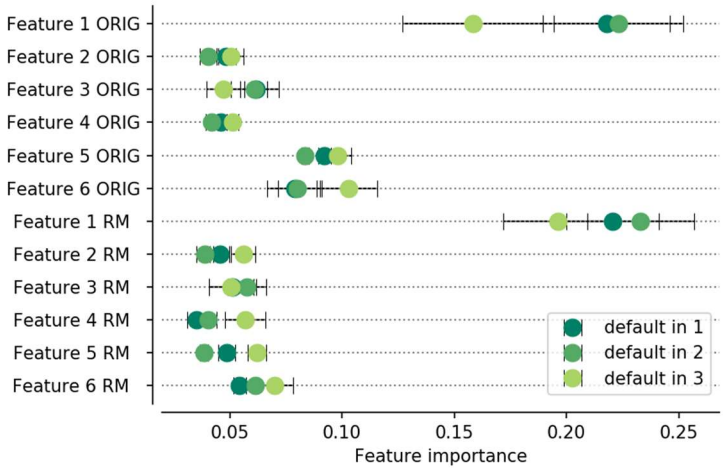

Figure 3.2: Feature importances of the supporting model by periods to the default. The error bars present the STD of the feature importance over 3-fold CV.

Now the change in feature importances separated by different periods to the default's occurrence is investigated. Therefore, the results of the supporting model are shown in Figure 3.2. On the y-axis the features and their transformations are listed, as shown in Table 3.2. The x-axis shows the importance of the feature according to the RF. The different colours present the different data sets, hence different periods to the occurrence of the default. To see the variance of the feature importance, the STD over the 3-fold CV is shown as error bars of the data points. Interestingly, the *default in 3* values do not have

a larger variance of the feature importances and show the least extreme values. Therefore, a default further in the future has less concrete indicators, than a close default.

Concluding, of the investigated features, the amount of overdraw is the most important indicator for a future default. Hence, the focus should be put on features containing overdraw information during the feature extraction.

3.3. Evaluation of the presented techniques

In this chapter models are applied to predict a credit default in the next 3 months. Resulting from the supporting model, a grasp of the important features has been established. For the data set we consider all three tables mentioned in Chapter 3.1. The data in the tables are snapshots of information on a given time. Hereafter, these snapshots are named *account data* and *person data*. Additionally, we enrich the snapshots with the historical data of specific continuous variables. The historical data includes the data of the snapshot and reaches nine months back from the time of the snapshots.

3.3.1. Data processing

To provide the model as much information as possible to predict a future default, all available information is combined. We apply feature aggregation techniques to extract information on the historical data of *account* and *person*. The aggregated information can be merged with the corresponding tables. Afterwards, the tables are merged on the borrower's level. This creates the data set used for the prediction. Figure 3.3 illustrates the relation of the data processing steps to obtain the data set for the predictions.

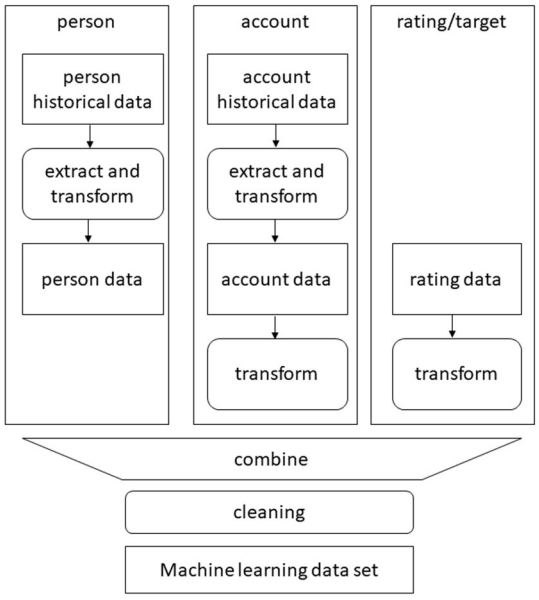

Figure 3.3: Schematic design of the preprocessing of the 3-month data set. We show data as shapes with edges and activities as rounded shapes.

Specifically, the aggregation functions applied to the historical data are shown below. Assume $\boldsymbol{x}$ to be the historical data of one feature and one borrower or account. u is the number of periods, hence $|\boldsymbol{x}| = u$ and $\boldsymbol{x} = (x_1, \dots, x_u)$. The applied aggregations are:

1. Median: $x_{med} = \begin{cases} \frac{x_{u/2}+x_{1+u/2}}{2} \; if \; n \; mod(2) = 0 \\ x_{(u+1)/2} \; otherwise \end{cases}$
   is the value which separates the data into two equally sized subsets.
2. Mean: $\mu = \frac{1}{u}(\sum_{i=1}^{u} x_i)$
   is the sum of the sample values divided by the number of samples. For this task we consider the arithmetic mean.
3. Standard deviation (STD): $\sigma = \sqrt{\frac{1}{u}\sum_{i=1}^{u}(x_i - \mu)^2}$
   with μ as mean as arithmetic mean as defined in the previous function. We use the STD to measure the amount of variation in the dataset. A low STD value indicates that that the data samples are concentrated around the mean, on the opposite a large value indicates a spread over a larger range of values.
4. Min: $x_{min} = argmin(\boldsymbol{x})$
5. Max: $x_{max} = argmax(\boldsymbol{x})$

Additionally, these standard statistical functions we apply more advanced function to each historical vector.

6. Slope: The slope of a linear regression fit $\boldsymbol{x}$. We are interested in the trend of $\boldsymbol{x}$, since the previous methods are unable to capture this behaviour. We fit a linear regression to $\boldsymbol{x}$. The slope of the linear regression presents the trend and defines the name of function. The linear regression is defined as $y_i = \alpha + \beta x_i + \varepsilon_i$, where β is the slope and α is the y-intercept. ε_i is the error term defined by the difference between the actual and predicted targets. To achieve the best fit of the linear regression, we try to minimize the sum of the squared residuals.

The introduced functions should deliver a good understanding of the historical features. We consider the values, the spread and the trend. However, a question is not solved by these functions. In what time do we expect a balance to reach zero, if the customer continuous to charge the account like it has been done in the historical data? Answering this requires the combination of two variables, the value of the latest historical account balance and the mentioned slope. A linear regression is applied to the historical account data. This new feature is called **t**ime-**t**o-**e**mpty-**p**ockets ($ttep$) and is defined as $ttep = \frac{-b}{\beta}$, where b is the account balance x_u in the last available period and β is the slope of the applied linear regression.

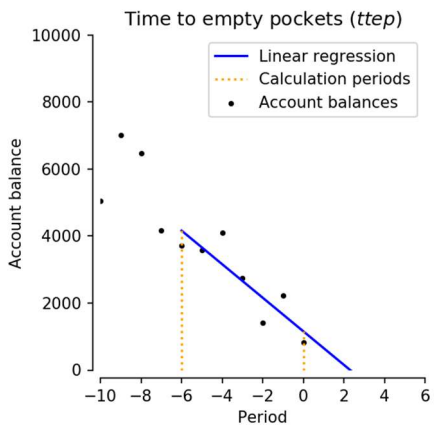

Figure 3.4: Visualization of the time-to-empty-pockets feature using dummy data. We consider the six recent periods for the calculation of the ttep, the first and the last considered periods are highlighted in the plot by the yellow dotted line.

Figure 3.4 plots the account balance of a sample account against the periods. Historical account balances are plotted as black point. In this example it would be $ttep = 2.3$. Hence, the balance is expected to reach zero in 2.3 periods.

To merge *account* with *person*, the categorical features in *account* need to be dummied, as described in Chapter 2.8. This allows aggregating the *account* data by borrower by sum. Hence, the two tables can be merged. The resulting dummy features can have values larger than one, which may be confusing for humans. However, for the machine learning algorithm this does not create an issue, since the relation is correctly presented in the data.

After transforming and merging the data set, we need to handle missing values in the data. SCHAFER, GRAHAM (2002) suggest some methods on how to handle the missing values in the dataset. Most of the machine learning algorithms are not designed to process the missing data itself, hence the missing values need to be taken care of upfront. An example is the maximum likehood imputation, where data is imputed which is most likely to be the missing data according to some predefined metric. This is especially appealing when the missing data is not randomly distributed but related to some patterns in the data.

During the processing of missing values, a preselection of features is conducted. Only features which have no more than 5% of their data missing are considered. All other features are dropped beforehand. Merely overdrafts are considered in a special matter. Here we impute the number zero to missing data, which implies that the overdraft is not existent. In a second step we drill down on the perspective and look at the data samples instead of the features. The data samples are not considered if there is any missing data in the sample. This does not remove many samples, since the preselection on feature level removed most of the missing data already. Handling missing data by imputing would have been possible as well. However, imputing on an imbalanced dataset, like the one present for this thesis, may have a rather large impact on the outcome. Consider the small number of defaults in the dataset. The maximum likehood function would refer to a value common in the majority class, resulting in suboptimal classification performance on the minority class.

After applying the mentioned processes, the resulting data set contains the aggregated information of all available tables. However, the categorical features are not dummied.

3.3.2. Data set

Transforming the original data according to the mentioned process generates the 3-month data set used for the application of machine learning tasks. The target equals 1, if the borrower is currently not defaulted but defaults within the next three months, otherwise the target is set to 0. During the model selection phase, we evaluate different models with various parameters. As mentioned in Chapter 2.2, the data set is split in two parts. Resulting from the transformation, we obtain a total of 242 883 data samples, which are split 80:20 for the train and test set. The resulting train and test sets are shown in Table 3.3:

Attribute	Train set	Test set
Number data samples	194 231	48 652
Number defaults	195	53
Default frequency	0,100%	0,109%

Table 3.3: Default statistics on the 3-month data sets. We consider the train and test set individually.

The dummy transformation of categorical features leads to a strong increase of features. For each unique value of each categorical feature one dummy feature was created. Before applying the dummy transformation to the data set, it consists of 22 categorical features and 192 continuous features. The dataset with dummies consists of a total of 690 features.

3.3.3. Univariate feature analysis

To obtain a better understanding of the features' impacts on the prediction, we perform a univariate analysis for the features. The approach we use is inspired by D' ANGELO (2016). In contrast to the supporting model, all features are evaluated here. We fit a RF to each feature individually and compute the AUC-ROC to determine the univariate discriminatory power of each feature. Hence, 690 AUC-ROC values are obtained for 690 features in the data set. Since we are merely interested in the impact of the feature on the prediction result, we do not apply other metrics to the model's outcome.

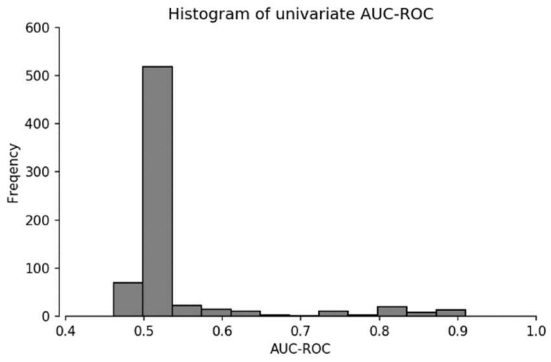

Figure 3.5: Histogram of univariate AUC-ROC values. We consider 12 bins with equal length.

The result of the univariate analysis is shown in Figure 3.5. A histogram is used to visualize the distribution of the univariate AUC-ROC values. Summing the frequencies of all bins would equal the total number of features. It can be observed, that many features are in the second bin, indicating a large number of features with a low discriminatory power. The second bin contains all features with AUC-ROC values in [0.498, 0.536). A total of 519 features are count in this bin. Due to the low AUC-

ROC score, these features can be expected to have a low discriminatory power. There are 69 features with an AUC-ROC value lower than 0.5, and 123 features with an AUC-ROC value of 0.5. However, since this is a univariate analysis no information about the combination of features is obtained. A combination, even of weak features, may result in better results due to the higher discriminatory power.

Combining the last three bins in the histogram we obtain a total of 40 features with AUC-ROC values in [0.798, 0.910], which represent only 5.7% of the total number of features. We consider the features in the last 3 bins to have a high discriminatory power. The univariate analysis implicates that only a model using at least some features with a higher discriminatory power can achieve accurate prediction results. For the present dataset that means that merely every 18th feature has a high discriminatory power.

Metric	Value
Mean	0.532
STD	0.089
Min	0.461
Max	0.910
Median	0.500

Table 3.4: Statistics of the univariate AUC-ROC values.

The statistics of the univariate AUC-ROC results are shown in Table 3.4. As expected from the analysis of the histogram, the mean of the AUC-ROC values is close to 0.5, indicating a poor discriminatory power of most features. The difference between median and mean implies the skew of the distribution.

We continue to investigate two features exemplarily, since the vast number of total features makes an investigation of each feature within the limitation of this thesis impossible. Therefore, we choose one feature with a high and one with a low discriminatory power. We refer to the two features as strong and weak feature. We want to show the differences of the distributions of the non-defaulting and defaulting borrowers. Hence, we choose a conditional plot separated by the target variable. For the weak feature we choose the age of the person, the strong feature is the sum of overdrawn accounts. Both features are continuous and not dummied.

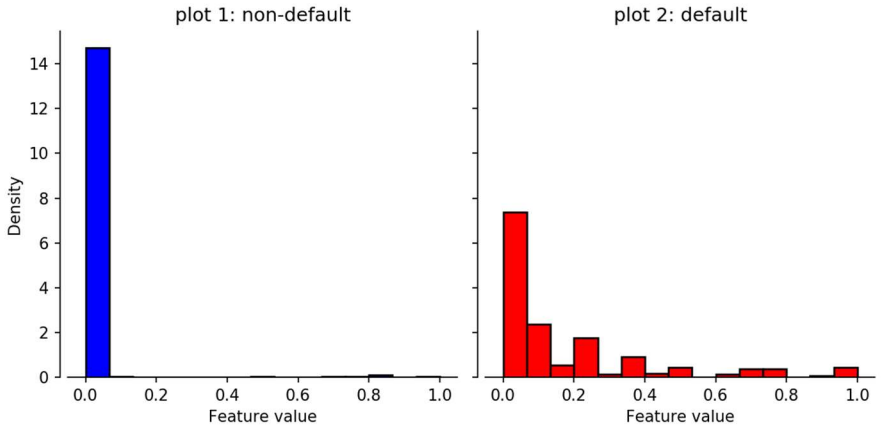

Figure 3.6: Conditional histogram of the strong feature. We consider 15 bins of equal length.

To show the difference in in the distributions we use a histogram again. Since the difference of the distributions for non-defaulters and defaulters is especially interesting, we plot the conditional distribution according to $\boldsymbol{y}$. Hence, two figures are resulting. Figure 3.6 shows the conditional distribution of the strong feature. Plot 1 shows the distribution for non-defaulting borrowers, and plot 2 shows the distribution for defaulting borrowers. Shared x and y-axis to ensure an intuitive comparison of both distributions. Since the density of the distributions is plot, the area of the bars will sum up to 1. Both distributions use the same values for creating the bins, resulting in comparable density values for every bin. Intuitively, the two distributions differ for non-defaults and defaults. The non-defaulting distribution has nearly all the data samples in the first bin, indicated by the height of the first bar. Compared to the distribution of defaulters, visualized in plot 2, such a dominating first bin is not present. The first bin of the default distribution shows most of the data being in the first bin, but in contrast to the non-defaulting histogram, other bins contain a considerable number of data samples as well. The significant difference in the distributions for non-defaulters and defaulters gives the strong feature its high discriminatory power.

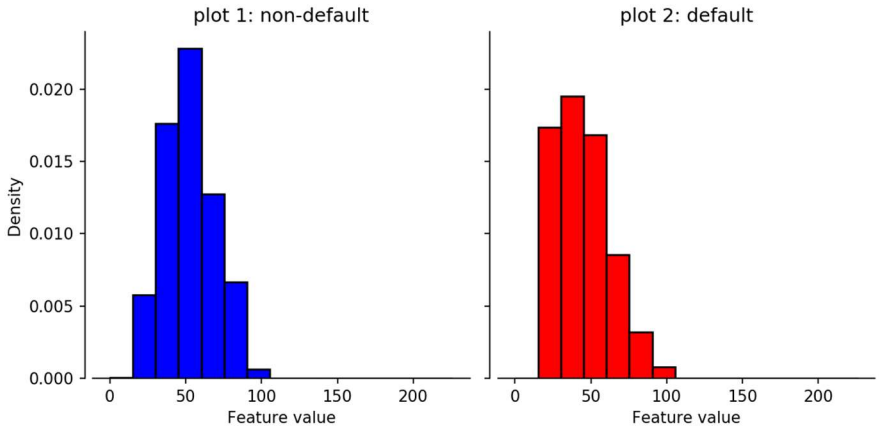

Figure 3.7: Conditional histogram of the weak feature. We consider 15 bins of equal length.

The conditional distribution for the weak feature is plot in Figure 3.7. Again, plot 1 shows the distribution for non-defaults, the distribution for defaulters is displayed in plot 2. There is a difference in the distributions, although they are more similar, compared to the strong feature. 15 bins were considered for the weak feature as well, however only 6 bins for non-defaulter and 7 bins for defaulters are visible. This is likely due to some outlier with a high feature value, resulting in empty or nearly empty bins. In the non-defaulting histogram, we can see that the distribution has its peak in the middle of the visible bins. Compared to the weak feature, we can see a slight difference in the shape of the distribution. This is reasonable since the AUC-ROC of the selected weak feature is still above 0.5. The defaulting distribution shows a skew towards larger feature values. The peak is located towards lower features values. Hence, borrowers with a lower feature value are slightly more likely to default. However, the conditional distributions of the weak feature look much more alike, than what was observed with the strong feature.

Metric	Strong feature	Weak feature
AUC-ROC	0.741	0.579
Mean	0.011	52.579
STD	0.086	16.540
25%	0.000	40.000
50%	0.000	51.000
75%	0.000	64.000
min	0.000	0.000
max	1.000	226.000

Table 3.5: Comparison of statistics for the strong and weak feature.

The statistics for the two features are shown in Table 3.5. The difference in AUC-ROC values quantifies the difference in discriminatory power of the features. Both features are skewed, indicated by the difference between mean and median. The upper quartile of the strong feature is the same as the median and the lower quartile. Hence, only few data samples have values >0. The high AUC-ROC of the strong feature, and the shape conditional distribution, indicate that most of the samples >0 belong to the defaulting class. Concluding, we can agree with the AUC-ROC values of the strong and the weak feature by what is visible in their conditional histograms.

3.3.4. Model and Grid Search

Knowing which models perform well on a given data set is crucial. However, investigating every model with all its parameters is much effort. In this chapter we want to decide which model is investigated in more detail in the next chapter.

We consider the machine learning models as implemented in the library `sklearn` in **python**. Various models are trained with an exhaustive grid search, which trains each model with every possible combination of defined parameters. Hence, not every parameter needs to be investigated manually. We use a 3-fold CV to obtain predictions on all data samples in the train set.

Model	F1	STD F1	Fit time [s]
RandomForestClassifier	**0.472**	0.068	355
ExtraTreesClassifier	0.439	**0.017**	**838**
DecisionTreeClassifier	0.420	0.060	12
GradientBoostingClassifier	0.391	0.067	459
AdaBoostClassifier	0.366	0.091	139
LogisticRegression	0.273	0.066	10

Table 3.6: The best performing models on the 3-month train set. We consider the best results from the exhaustive grid search according to the F1 score.

Since models using ensemble methods were expected to perform well on the task of credit scoring, we choose the ensemble methods `RandomForestClassifier` (RF), `GradientBoostingClassifier` (Grad), `ExtraTreesClassifier` (ET) and the `AdaBoostClassifier` (Ada) from the module `sklearn.ensemble` for the grid search. Additionally, a simpler `DecisionTreeClassifier` (Tree) and `LogisticRegression` (LR) were selected as well. For each of these models we select various parameters and show the best F1

score per type of model. The results are sorted according to the F1 score in a descending manner nad shown in Table 3.6. Also, the STD of the F1 score over the 3-fold CV is displayed. Clearly, the RF has an advantage in F1 score over the other models. The ET has a slightly lower F1 score, but also a much lower STD. Hence, this model provides excellent results when a low variance is desired from the prediction. However, since the time needed for fitting is rather long for the ExtraTreesClassifier, the ET is not further considered. The same issue applies to the Grad, where the time for fitting is rather long. The fitting time of the RF is found to decrease a lot when investigating its parameters in more detail.

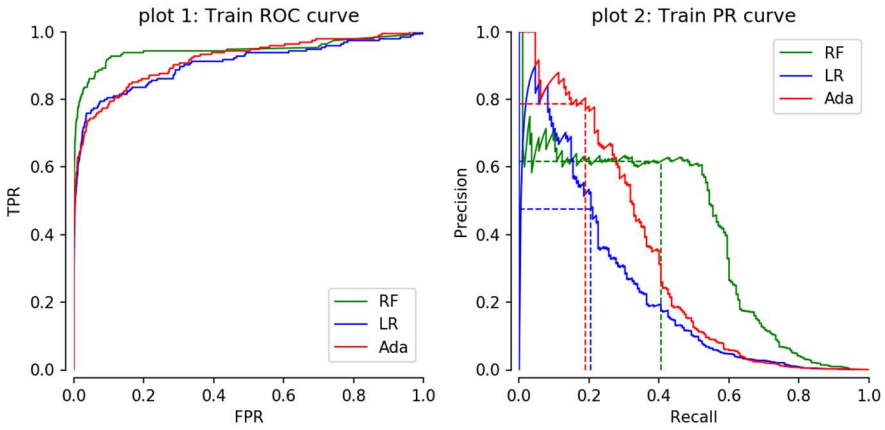

Figure 3.8: ROC and PR curves of best grid search models on the 3-month train set. We show $c = 0.5$ by the dotted lines in the PR plot.

We show the ROC curves in plot 1 and PR curves in plot 2 of three selected models in Figure 3.8, which are RF, Ada and LR. They were selected since fitting times are relatively short and the F1 score is high (for RF and Ada), the LR is interesting since it is widely used for credit scoring. Considering plot 1, all models show a ROC curve much better than random guessing. Remember that the imbalanced class distribution of the dataset leads automatically to good ROC curves, if the model classifies most of the data samples as majority class. Hence, the small differences in the ROC curve must be interpreted as significant differences in prediction quality. The PR curve in plot 2 shows the PR curves of the same probabilistic predictions. Clearly, the RF curve dominates in PR space, except for precision values higher than 0.6. The precision and recall values of a threshold of 0.5 are displayed by the dotted lines in the corresponding colour. Notice, that the LR shows the worst performance in PR space. This implies that the given credit scoring task is complex, since RF and Ada are more complex models than the LR.

The plot confirms that the RF is the best performing model, with the best performance in almost all areas of the PR curve. Hence, we want to further investigate the behaviour of different parameters on the classification performance of the RF in the following chapter.

3.3.5. Investigation of the Random Forest

Since the Random Forest implementation offers the best results on the train set within a reasonable duration for computation, the impact of different parameters on the prediction quality of the model is further investigated. Again, the implementation of the RF in `sklearn` is considered. Contrarily to the original implementation by BREIMAN (2001), the consensus is not achieved by a majority vote of the individual trees. The `sklearn` implementation of the RF uses the average of the scores of the decision trees to combine the results of the base models (scikit-learn 2017a). Notice, that we use the impurity measure of the Gini index (Chapter 2.4.2) for all models, that are based on decision trees, as well as

decision trees themselves. The documentation of the RF (scikit-learn 2017b) has been used as a source for this chapter.

Conducting prediction tasks on an imbalanced data set, like the one of credit scoring, naturally leads to the idea of putting a larger weight on the minority class. The used implementation of the RF provides the parameter `class_weight`. Adjusting this parameter will put different weights on the incorrectly predicted data samples for different classes. Since credit scoring is a binary classification task, we use the parameter to set the `class_weight` of the minority class to an arbitrary multiple of the majority class. Hence, the value 5 for the `class_weight` means we put 5 times the majority class weight on the minority class. We create a Random Forest using each value in $class\_weight\_list$ as `class_weight`:

$$class\_weight\_list = \{1, 5, 9, 13, 17, 21, 25, 29, 33, 37\}.$$

Since we are interested in the impact of different depths of the Random Forest on the prediction quality, we use the values in max_depth_list as value for the parameter `max_depth` of the same models:

$$max\_depth\_list = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19\}.$$

This results in 180 RF's with different parameters. All other parameters of the model are considered fixed. A high number of base models (`n_estimators`) is chosen to achieve a rather low variance of the model. The fixed parameters for the first evaluation are:

`n_estimators` $= 80$
`max_features` $= \sqrt{n\_features} \approx 27$
`min_samples_split` $= 2$

The precision and recall scores are plot against the `class_weight` in Figure 3.9. Additionally, the `max_depth` of each Random Forest as a different colour. The goal of the plots is to show the general impact of both parameters on the prediction quality, measured through precision and recall and to reveal a potential correlation of the parameters. Hence, we consider lines instead a scatter plot.

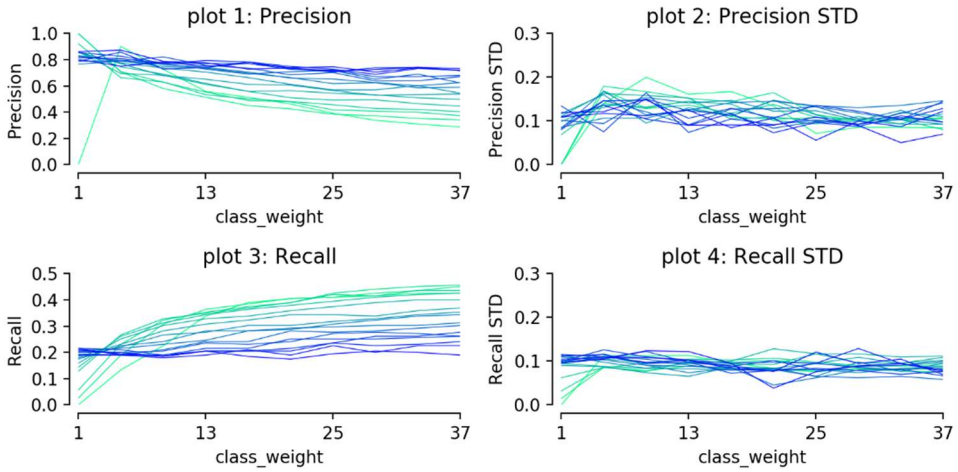

Figure 3.9: Precision, recall the STD's against `class_weight` on the 3-month train set. Deeper models are shown in blue, shallower models are shown in green.

Figure 3.9 reveals that deeper models are more resistant to the increase of `class_weight`, observable by the small change in precision and recall over different values of `class_weight` for deeper models. As we can see in plot 1, the precision of more shallow models is lower compared to deeper models. This difference increases with the increase of the class weight. Contrarily, the recall of more shallow models is generally higher than the one of deeper models, the difference in recall increases with higher values of `class_weight`. We notice in plot 2 and plot 4 that the STD of both scores does not alter significantly over different values for the parameters. However, they are rather high, which may be due to the small number of defaults in the data set. The ideal selection of parameters should deliver a high precision, as well as a high recall score. Therefore, we also plot the F1 score for the same parameters. Since the F1 score is the harmonic mean of precision and recall, it suits the task of finding the ideal trade-off.

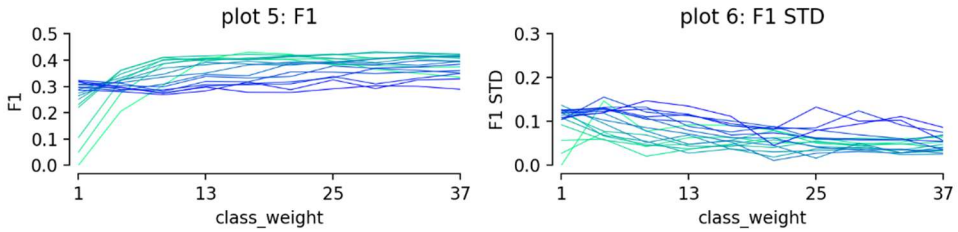

Figure 3.10: F1 scores and STD's against `class_weight` the 3-month train set. Deeper models are shown in blue, shallower models are shown in green.

Figure 3.10 consists of the F1 scores and the corresponding STD's over different values of `class_weight` and `max_depth`. In plot 5 we can observe the F1 scores of the shallower RF's to be higher than deeper models. Therefore, a shallow model should be chosen for the present task, where a high precision and a high recall is important. Interestingly, in plot 6 the STD's of the F1 score are smaller for shallower models. This adds to the choice of a rather shallow RF. The class_weight is determined in a way where precision and recall are balanced and the STD of the F1 scores is rather small. The selected parameters of the evaluation are:

$$\texttt{class\_weight} = 10$$
$$\texttt{max\_depth} = 6$$

The investigation is continued with the impact of `max_features` on the prediction quality of the RF. The parameter determines how many features the underlying decision trees consider for finding the feature/value combination offering the best split. Like in the previous investigation, multiple RF's are created, where `max_features` is selected from $max\_features\_list$:

$$max\_features\_list = \{3, 5, 7, 9, 13, 18, 26, 36, 50, 70, 97, 134, 186, 258, 358, 497, 690\}.$$

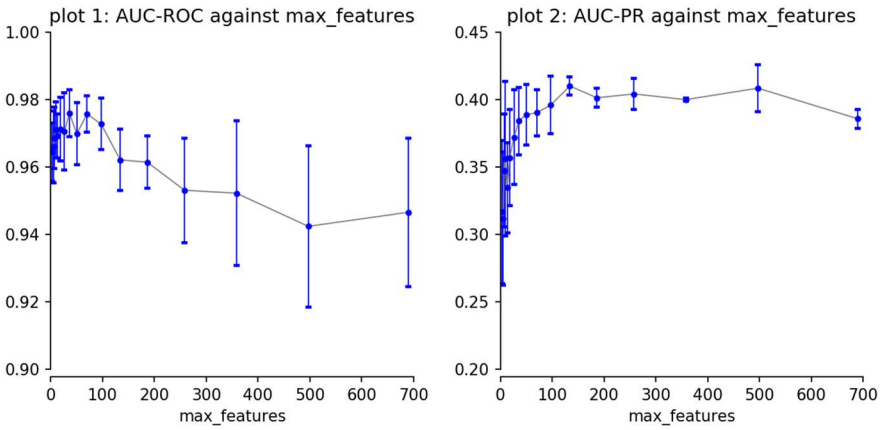

Figure 3.11: AUC-ROC and AUC-PR against max_features on the 3-month train set. The error bars show the STD of the metrics over the 3-fold CV.

In Figure 3.11 we show the AUC-ROC and the AUC-PR over the `max_features`. Looking at the AUC-ROC values in plot 1, an increase of the score at a `max_features` value of around 50 can be observed. This is a rather low value considering the maximum number of features. Increasing the parameter further results in a drop of the score, as well as a significant increase in the AUC-ROC STD. Both effects are obviously not desired. Like the AUC-ROC, the AUC-PR shown in plot 2 has an increasing score for increasing values of `max_features` up to a value of `max_features` of about 130. However, the score stays on a more stable and does not drop as much as the AUC-ROC. The increasing AUC-ROC STD for an increasing value for `max_features` indicates that the individual trees in the RF are less diverse, since underlying the underlying decision trees select more similar features at each split. Therefore, the variance of the resulting RF increases with the number of `max_features`. Another disadvantage when increasing `max_features` is the linear increase of computational time for the fitting of the Random Forest (Louppe 2014). For the further investigation the recommended value for `max_features` on classification tasks is selected `max_features`$= \sqrt{n\_features} = \sqrt{690} \approx 27$, where $n\_features$ is the number of features in the data set (scikit-learn 2017a).

So far, we have chosen a large number for $n\_estimators$ to ensure a low variance of the RF. Now we investigate how many decision trees are necessary to achieve a rather low variance, and at which number of trees the metrics converge. Hence, the AUC-ROC and the AUC-PR are plot against `n_estimators`. Again, one RF is created for each value in $n\_estimators\_list$.

$$n\_estimators\_list = \{1, 2, 3, 4, 5, 6, 9, 12, 16, 21, 27, 36, 48, 64, 84, 111, 147, 194\}$$

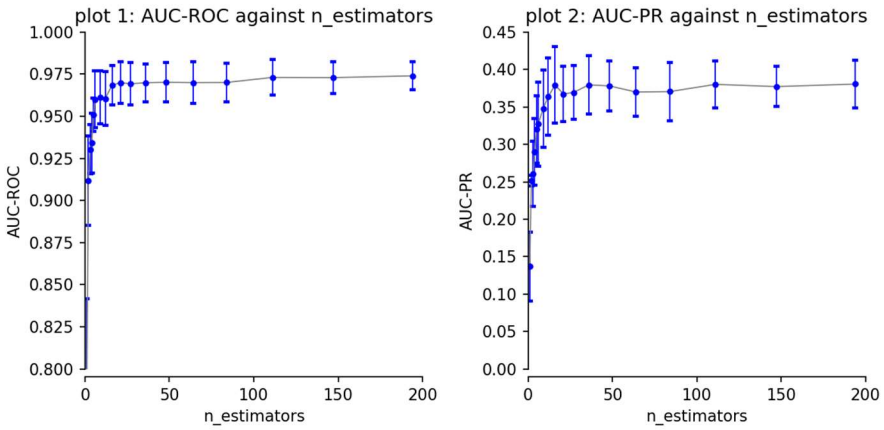

Figure 3.12: AUC-ROC and AUC-PR against n_estimators on the 3-month train set. The error bars show the STD of the metrics over the 3-fold CV.

The relation between the AUC-ROC and F1 score to the number of base models is shown in Figure 3.12. In plot 1 the AUC-ROC score is shown, plot 2 shows the AUC-PR score. An early convergence of the AUC-ROC score over the `n_estimators` can be noticed. The STD of the AUC-ROC is slightly decreasing even for high numbers of decision trees. The AUC-PR shows a similar behaviour. However, the AUC-PR scores show a higher STD in general. Again, this may be due to the higher sensitivity of the AUC-PR to the small number of defaulters in the data set. Hence, many `n_estimators` should be considered for a stable prediction with low variance. Notice, that more base models means the ensemble takes more computational effort. We choose `n_estimators` to be 100, which offers a good trade-off between the low variance and the required computational effort.

The investigation of the RF has shown the impacts of different parameters on the model's prediction quality. Such an investigation should be considered for all models that are used to understand which parameters should be tuned to achieve the desired result. However, this would exceed the limits of this thesis. Therefore, the investigation of the RF should be considered an example of such an analysis.

3.3.6. Over- and under-samplingWe have seen in Chapter2.7 there are techniques to balance out an data set with skewed class distribution. The data set we use throughout Chapter 3.3, uses a 3-month period as target, resulting in a target distribution of 1000:1. Hence, we want to investigate the impact of applying the over- and under-sampling techniques to the dataset. We consider the following oversampling methods {ROS, SMOTE, SMOTE Borderline 1, SMOTE Borderline 2, ADASYN}. Additionally, the following under-sampling methods were applied {RUS, NearMiss}. Each of the over- and under-sampling techniques was applied to a RF with `max_depth` = {2, 5} and a LR. The `class_weight` was not changed, since each sampling technique provides a balanced data set already. Notice, sampling techniques must not be applied to the test set. The sampling techniques are merely used to refine the decision boundaries of the model during model fitting. Again, a 3-fold CV is considered to obtain scores for all data samples in the train set.

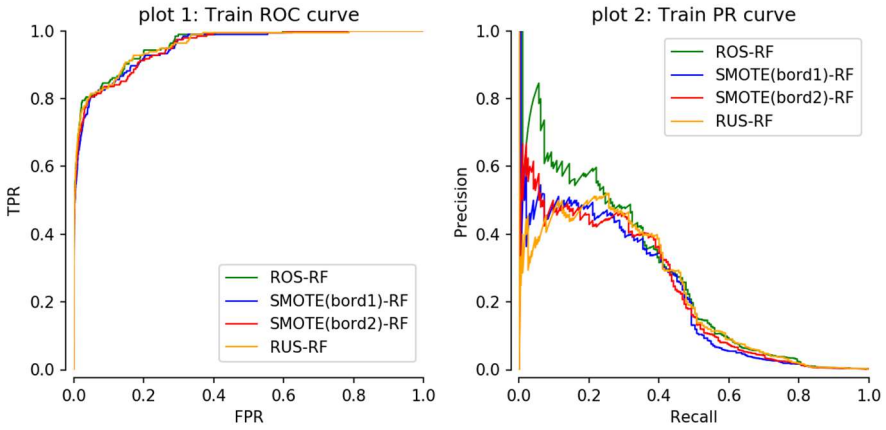

Figure 3.13: ROC and PR curves of sampling techniques on the 3-month train set. We show the curves of the top 4 combinations of over- and undersampling techniques and models, according to the AUC-PR.

In Figure 3.13 we plot the ROC and PR curves for the top 4 performing sampling techniques on the train set, as it is shown in Table 3.7. In plot 1 the ROC curves are shown, plot 2 shows the PR curves. We notice similar curves for all sampling techniques in the ROC plot. However, the PR curves in plot 2 exploit different performances on the defaulting borrowers. Unexpectedly, the ROS technique deliver the best PR curve. Especially for values of a high precision, the recall is above any other shown curve. This is not expected, since ROS is a very simple technique compared to the SMOTE techniques. We have potentially overfit the training data with the ROS technique.

Method	AUC-ROC	AUC-PR	Brier
ROS-RF	0.960	0.275	0.078
SMOTE(bord1)-RF	0.953	0.223	0.028
SMOTE(bord2)-RF	0.954	0.228	0.032
RUS-RF	0.958	0.226	0.085

Table 3.7: Results of sampling techniques on the 3-month train set. We show the metrics of the top 4 combinations of over- and undersampling techniques and models, according to the AUC-PR.

Table 3.7 shows the metrics of the best 4 combinations of sampling technique and machine learning model according to the F1 score. We have tried different models on top of the sampling techniques

and found the RF to deliver the best results with `max_depth`= 5. The oversample techniques generally lead to a high recall score, but trade in a low precision score. This implies that the decision boundaries of the models trained on oversampled data do not have the necessary detail and predict too many samples as positives. This increases the recall, since many defaults predicted as such, but the large number of false-positive samples reduces the precision score dramatically. A reason is this behaviour is that the artificial data samples of the positive class are too close to the negative class. Hence, the algorithm cannot distinguish between the two classes.

Due to the advantage in AUC-PR, we use the ROS oversampling approach in combination with a RF as the best model of the oversampling and undersampling approaches. Putting the focus more on the brier score and less on the AUC scores, may make the use of a SMOTE model more appropriate.

3.3.7. Stacked model

As introduced in Chapter 2.5.3, stacking can be used to combine the outcomes of different models. We use stacking to combine the outcomes of different, heterogenous models which were trained on the original data. Also, this allows to create a prediction using not only machine learning models, but also the internal scores from FraSpa. Hence, the stacking model would not present a competitive alternative to the currently implemented internal model, but rather act as a symbiosis of both. We expect the stacking model to be more likely to be accepted by the people working with the scores. Hence, the proposed stacking model is a combination of machine learning models and FraSpa scores.

The models and parameters from Chapter 3.3.4, which provided the best F1 score, are selected as base models of the stacked ensemble. Additionally, the FraSpa scores are used as another input for the meta-model as well. To provide an equally weighted data set to the meta-model, the base models' scores are scaled in [0, 1]. This is necessary for the meta models that are sensible to the relative size of the features.

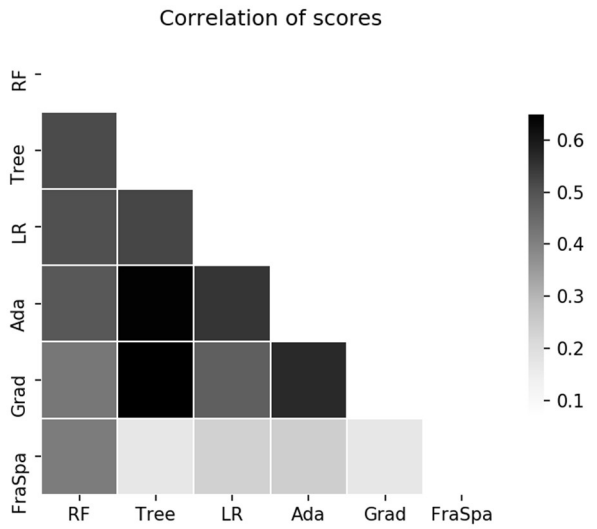

Figure 3.14: Correlation of base model scores on the 3-month train set. We use the Pearson correlation to determine linear relations between the scores from the base models before applying a meta model.

Stacking takes its advantage from diverse models which perform differently in different areas of the data set (Zhou 2012). This allows the meta model to learn and choose the scores of the correct model if a certain pattern occurs. Hence, the scores the meta-model considers as input should be diverse.

Figure 3.14 plots the correlation matrix of the base models' scores using the *Pearson* correlation, which identifies linear relations between the scores of different models. Interestingly, the simple decision tree shows a relatively high correlation to both boosting algorithms. Also, the correlation of the Ada and Grad is rather high, however this can be expected since both models use boosting. Another interesting observation is that the RF scores show the highest correlation to the FraSpa scores. We have not shown a comparison with the FraSpa scores so far, however, this indicates that the scores of the RF are the ones most like the FraSpa scores. Since the strongest correlation is just above 0.6, we continue the investigation of the stacked model without removing any of base-models.

Again, it is not obvious which model works best as a meta-model on the given data set. Hence, four models are considered as meta-models and the best performing one should be selected. Since, the data set for the meta-models is rather simple (6 features), we consider rather simple types of models. The simplicity makes using kernel-methods in combination with a support vector machine possible. On the original data set kernel methods cannot be used due to their high computational costs. Hence, we introduce the **s**upport **v**ector machine for **c**lassification (SVC) with an 'rbf' kernel. Since we will decide not to use the SVC, the method is not further explained. The types of models considered as meta-models are {SVC, RF, LR, Tree}.

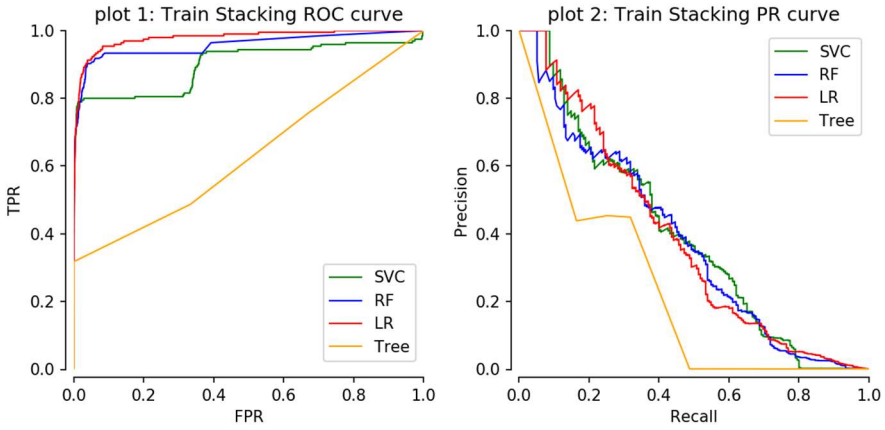

Table 3.8: ROC and PR curves of different meta models on the 3-month train set. We show the best parameters for each type of model.

In the Figure 3.9 the ROC curves (plot 1) and PR curves (plot 2) for each meta model are shown. Clearly, the decision tree has the worst performance of the different meta-models. The other three models show a rather high discriminatory power, with the LR dominating the ROC curves.

	AUC-ROC	AUC-PR
LR	0.978	0.378
RF	0.953	0.355
SVC	0.936	0.385
Tree	0.619	0.200

Table 3.9: AUC results the stacked approach with different meta models on the training data. For each type of model, we determine the parameters with the best results and show the resulting metrics. The meta-models are sorted according to the AUC-ROC value.

The AUC results of the ROC and PR curves are shown in Table 3.9. Notice that the AUC-ROC value of the SVC is the lowest compared to the LR and RF, however the SVC has the highest AUC-PR score. The LR is selected as best model for the stacking approach, since it offers the highest AUC-ROC and second

highest AUC-PR. Thus, we expect the LR to provide the highest discriminatory power of the investigated meta-models.

3.3.8. Calibration

The interest in using the model scores as default probabilities leads to this Chapter. Here, it is investigated how well the scores can be interpreted as default probabilities. Also, the impact of the calibration techniques presented in Chapter 2.7 are applied and investigated. Since the RF has proven to perform well on the data set, the calibration techniques are applied to the RF with the parameters found in Chapter 3.3.5. To compare the effects of the calibration, the techniques are applied to a LR as well. Generally, the calibration techniques aim to provide a better interpretability of the scores as probability. For an improved readability of the two calibration techniques, sigmoid function is encoded as *SIG* and the isotonic regression as *ISO*.

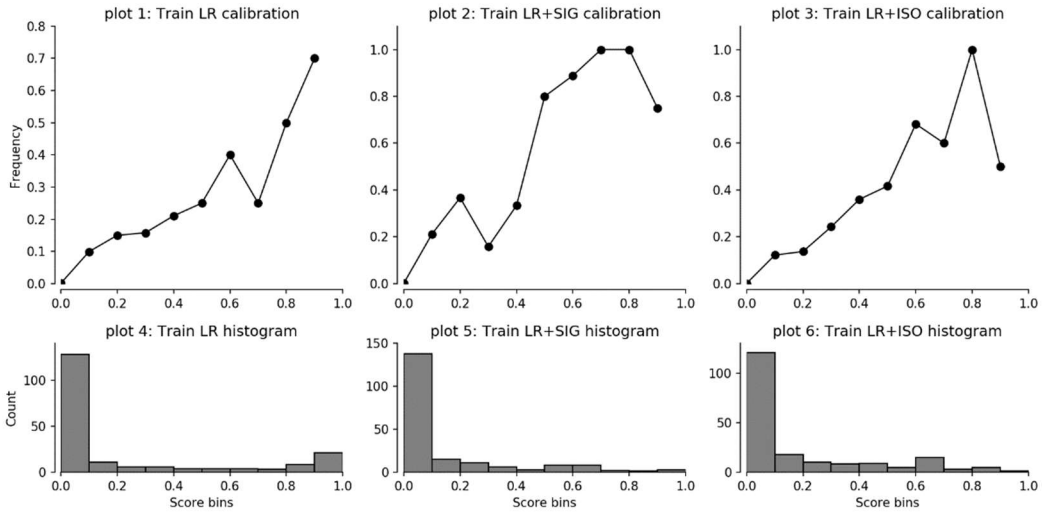

Figure 3.15: Calibration plot of the LR scores on the 3-month train set. We show the default frequency and the absolute number of defaults for 10 bins.

The calibration scores are shown in Figure 3.15. We consider 10 bins of equal length for the scores and compute the default frequency, as it has been shown in Chapter 2.7, and the absolute number of defaults for each bin. The plot is split in 2 rows and 3 columns, where the columns display the applied calibration method, while the rows show the default frequency and the absolute defaults. Note that in column 1 the original scores are shown. The linear relation can be observed the best from the original scores in plot 1. The last bin in plot 2 and plot 3 shows a significantly lower default frequency. The absolute number of defaults shown in plot 4 – plot 6 show that most of the defaults are in the first bin. This means that these defaults are not correctly classified by the model and are many false-negatives. Concluding, the linear relation of the default frequency and the score is hardly visible for the LR from the calibration plot, and the application of the calibration techniques did not provide improved results.

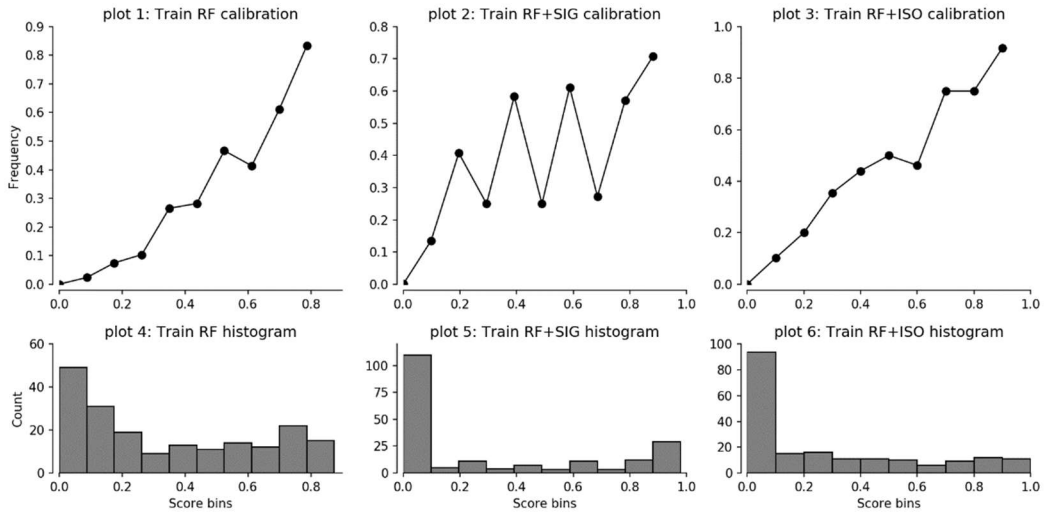

Figure 3.16: Calibration plot of the RF scores on the 3-month train set. We show the default frequency and the absolute number of defaults for 10 bins.

Applying the same techniques for calibration to the RF scores results in Figure 3.16. The original data, in plot 1, shows a rather good linear relation of default frequency over the scores. Clearly, the sigmoid function in plot 2 does not improve the linearity. The scores are rather distributed in a less linear manner. Additionally, the default frequency is more volatile over adjacent bins, which is certainly not an improvement towards a better interpretability. The use of the isotonic regression, as shown in plot 3, offers an improvement to the sigmoid values. The default frequency is aligned in an almost linear manner in [0, 1]. Looking at the absolute numbers in plot 6, we notice that the ISO scores have more defaulters in the first bin and fewer in bins of higher score, compared to the original scores. This pattern can also be observed with the SIG scores.

Model	Calibration	AUC-ROC	AUC-PR	Brier
LR	Orig	0.904	0.225	0.00092
LR	SIG	0.903	0.232	0.00087
LR	ISO	0.908	0.232	0.00086
RF	Orig	0.970	0.377	0.00095
RF	SIG	0.962	0.368	0.00080
RF	ISO	0.969	0.379	0.00076

Table 3.10: Scores of the calibration by metric of the on the 3-month train set. We define the model and the used calibration technique. The original scores are labelled 'Orig'.

For each of the probabilistic predictions of the LR and RF in combination with the calibration technique, we calculate the brier score to quantify the suitability of the scores as default probabilities. Also, the AUC-PR is shown, as well as the AUC-PR for each of the model and calibration technique combinations. The results are shown in Table 3.10. Remember that the Brier score is defined as a loss metric. Hence a smaller brier value means a better interpretability of the scores as default probabilities. For the LR and the RF, the Brier score is the highest for the original scores. Hence, the calibration of the scores has improved the interpretability of the scores as default probabilities. Also, the AUC-values have hardly changed over the calibration procedure. We notice the lowest overall brier score at the combination of the RF and the ISO calibration technique. Hence, this combination should be considered when the most linear alignment of the default frequency and the score is desired.

4. Results

In this chapter the results of the application are shown. The chapter is split into two subsections. The results of the comparison of different classification models and techniques are shown in Chapter 4.1. Here, the 3-month data set is used, as it results from Chapter 3.1. Out of each approach, the best performing technique and its corresponding parameters are selected. These are then compared to each other. For this purpose, the classification techniques are used, that were introduced in Chapter 2.9 of this thesis. Also, a recursive feature elimination is performed on the same data set, to evaluate how many features are necessary for a good prediction quality. The results of this evaluation are shown in Chapter 4.2.

In the second section of this chapter, the method that was found ideal in the previous comparison is selected and compared to the internal scores of the Frankfurter Sparkasse. Additionally, a LR is applied to the same data set to obtain a benchmark to validate the performance. For the comparison, the data set had to be altered and a 12-month target had to be considered. The results of this comparison are shown in Chapter 4.3. A short conclusion of the comparison is given in Chapter 4.4

4.1. Comparisons of different models

This chapter compares the models and techniques we have experimented with in Chapter 3.3. The purpose is to select the best performing model for the comparison with the FraSpa scores. Remember we are using the 3-month target, which includes the information extracted from the historical features.

We consider three models in their original state and add the best performing model from calibration, sampling and stacking. The models that are left original are labelled according to their original name. Models from special process are labelled according to the process. Out of the sampling methods, the over-sampling methods showed the best results on the train set. Hence, the best sampling method is labelled *over-sampling*. To see which model is taken for the individual process, consider the corresponding section in Chapter 3.3.6.

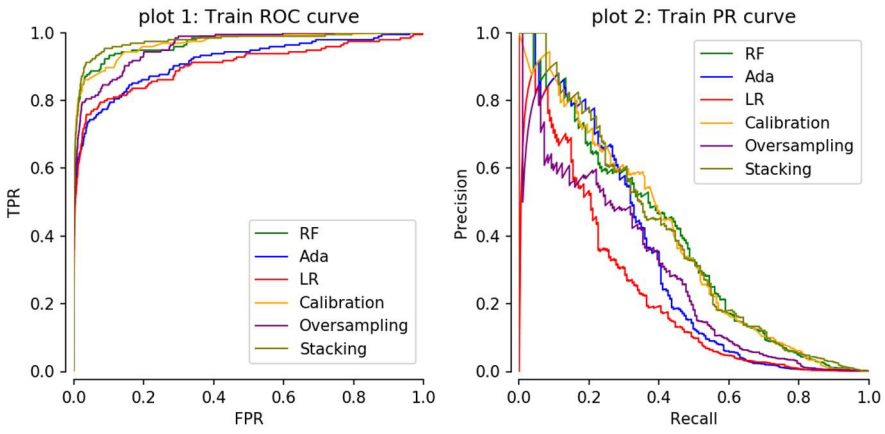

Figure 4.1: ROC and PR curves for the model comparison on the 3-month train set.

In Figure 4.1 the ROC and PR curves from the three original models and the three processed models are plot. Clearly, the ROC curves in plot 1 show a good discriminating power. However, the ROC does not consider the imbalances of the data set. Although, we can separate two models with a worse

performance already on the ROC curves. The Ada and the LR are not able to hold the performance of the other models. In plot 2 we show the corresponding PR curve, which makes the differences more obvious. Interestingly, the Over-sampling approach does not show a good performance for lower thresholds, as it can be observed for the other models. This indicates a poor prediction quality, and a bad interpretability of the scores as default probabilities. Even for high thresholds, where the model is more confident the borrower will default, the precision is rather low. Remember that the ROS method chosen for Over-sampling is the best sampling method, as we have seen in Chapter 3.3.6. Hence, we must conclude that the data set is not suitable for sampling methods. A reason might be the low total number of defaults, which gives little statistical structure to create artificial data samples. Looking at the best PR curves, we see that the RF, Calibration and Stacking approaches have delivered the best prediction quality on the minority class over different thresholds. The Ada shows a high recall for high precision values, but the relative performance decreases for lower precision values.

	AUC-ROC	AUC-PR	Brier
RF	0.970	0.377	0.0010
Ada	0.917	0.318	0.0008
LR	0.904	0.225	0.0009
Calibration	0.969	0.379	0.0008
Oversampling	0.961	0.275	0.0756
Stacking	0.977	0.385	0.0020

Table 4.1: AUC's and Brier of the model comparison on the 3-month train set.

The AUC values of the ROC and PR curves, as well as the corresponding Brier of the scores are displayed in Table 4.1. The results confirm the good performance of RF, Calibration and Stacking with the highest AUC-PR scores. As we expected, the Brier of the Over-sampling approach is the highest of all models considered, indicating a bad calibration. Except for Stacking, which has the second highest Brier, all other Brier scores are relatively close to each other. The Calibration approach using the RF with ISO improves the Brier slightly and should be considered, if the interpretability of the scores is the main goal.

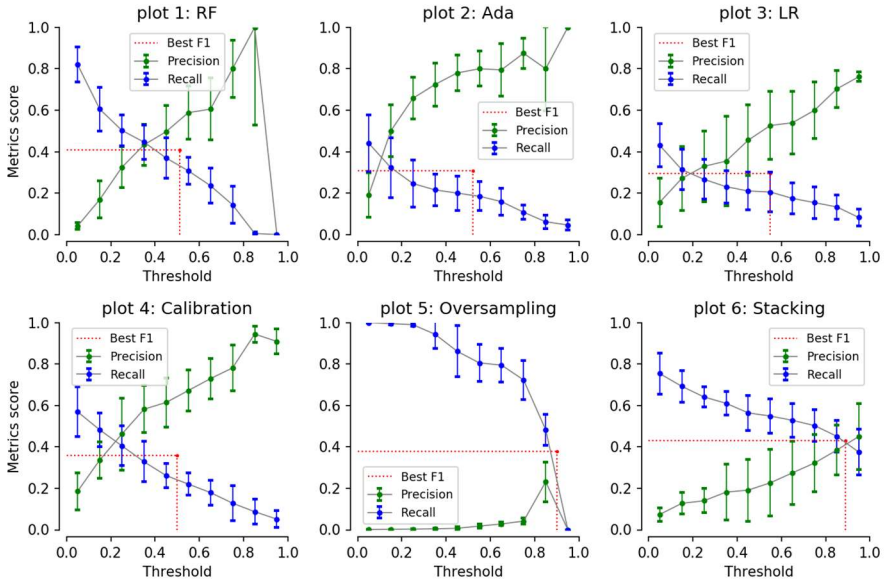

Figure 4.2: Precision and recall against c for the model comparison on the 3-month train set. The error bars present the STD of the corresponding metric over a 3-fold CV. The maximum F1 and the corresponding c is shown by the red dotted line.

In Figure 4.2 the Precision and recall of each model is plot against the threshold c. This plot makes the precision and recall of different models comparable for different c's. Hence, additionally to the PR curve, we obtain the information of the specific c. Generally, recall is high for small values of c and decreases with larger values of c. This relation can be expected, since a low c predicts many data samples as positive class, hence defaulters are predicted correctly. Precision on the other hand, has an inverse relation to the threshold c. The precision is low for small values of c and rises with an increase of c. Also, this relation to c is expected, since for a low c predicts many data samples as positive, resulting in the low ratio of true-positives and positive predictions. Hence, the c offering the best trade-off must be found. Naturally, the F1 score comes into mind, which incorporates both scores in the harmonic mean. However, the highest F1 score is not necessarily at the intersection of precision and recall. Therefore, we compute the F1 score for all unique thresholds and select the c resulting in the highest F1 score. This is the optimal c, which can be used to determine other classification metrics, depending on the threshold.

Interestingly, the plots in Figure 4.2 show a high variance. The RF scores, using the parameters we determined in Chapter 3.3.5, show the expected relation between precision and recall. A c of 0.5 yields about the same precision as recall, which can be interpreted as a good balance of the two. Since a high precision and high recall is desired, the 0.5 threshold would be a good choice for the RF. In plot 5 we show the same curves for the best over-sampling method we found in Chapter 3.3.6, where we selected the combination of ROS and RF. The recall is high over almost the entire range of c's. This implies, that the over-sampling technique creates artificial samples of the minority class leading to rather wide decision boundaries. Hence, the model classifies many data samples incorrectly as minority class, resulting in the high recall and low precision score.

	c	Precision	Recall	F1
RF	0.510	0.542	0.328	0.409
Ada	0.520	0.804	0.190	0.307
LR	0.550	0.526	0.205	0.295
Calibration	0.500	0.658	0.246	0.358
Oversampling	0.900	0.421	0.344	0.379
Stacking	0.890	0.416	0.446	0.431

Table 4.2: Results of the model comparison using an optimal c on the 3-month train set. We use the threshold c which results in the maximum F1 score for each model.

The exact values of the metrics using the optimized threshold can be seen in Table 4.2. We notice not only the large differences in the scores of the metrics, but of the chosen values of c as well. The highest F1 score is reached by the stacking approach.

The confusion matrixes of the model comparison on the train set are shown in the appendix in Table 7.1. Here, we can see the absolute numbers of the classified data samples. The good results of the stacking model show its downside, when investigating the absolute numbers in the confusion matrix. c had to be set high, hence less borrowers are predicted as defaulters, compared to the RF with its c.

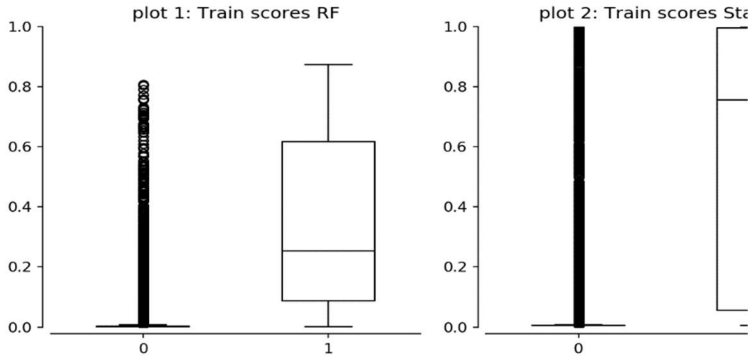

Figure 4.3: Boxplot comparison of the RF and Stacking scores on the 3-month train set.

Since the scores of the RF and the stacking approach have resulted in the highest F1 scores, the distributions of the scores is investigated. As a demonstrational example we show the analysis of these scores, since the analysis over all the models to of too much volume for this thesis. The conditional boxplots of the RF and Stacking scores are shown in Figure 4.3. The boxplot is conditioned according to the true target, where non-defaulting borrowers are shown as 0 and defaulting borrowers are shown as 1. Starting with the RF scores in plot 1, the difference of the non-defaulters and defaulters can be clearly observed. The boxes of non-defaulters and defaulters do not overlap, which means that the interquartile range of the two distributions does not overlap. This implies a good discriminatory power of the model already. On the non-defaulting side, we observe some outliers with a high score. These are the data samples which are potentially predicted as false-positives, depending on the chose c.

Switching to the scores of Stacking in plot 2, using a Logistic Regression as meta-model, as determined in Chapter 3.3.7, the distributions are significantly different as well. At first sight, the larger range and rather high score of the defaulters stands out. The median and upper quartile of the defaulters are much higher, compared to the RF scores. Interestingly, the lower quartile is lower than for the RF, which results, together with the higher median, in a wider range of the IQR. Hence, the scores are less

compact distributed. This less compact distribution, together with a larger number of positive outliers for the non-defaulting borrowers, results in the lower discriminatory power of Stacking. Also, the boxplot of Stacking shows why the c must be chosen rather high. Using a lower c would result in many false-positives, which is also indicated by the complementary plot in Figure 4.2.

Concluding, the original RF provides the best predictions on the 3-month train set. The calibration of the models results in a slight improvement of the Brier and a loss for all other metrics. Since the main goal of the task is to provide the model with the best prediction quality to charge the internal model of FraSpa, the RF is chosen as the ideal model.

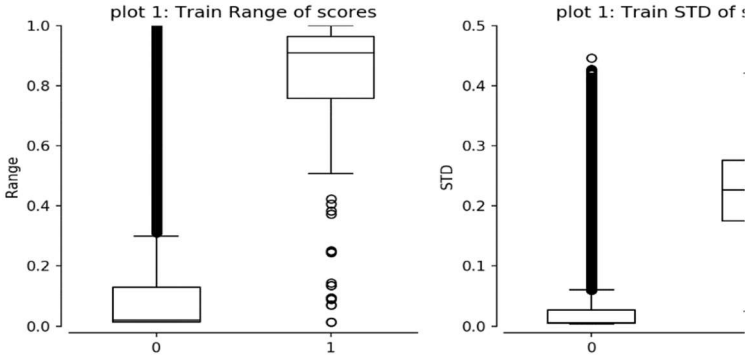

Figure 4.4: Range and STD of the RF scores on the 3-month train set.

Hence, the scores of the RFs base models are investigated in more detail, according to D' ANGELO (2016). We plot the range and the STD of the scores over the trees in Figure 4.4. Since, we are interested in the differences of non-defaulters to defaulters, we use a conditional plot once more. The distributions of the ranges and STD's are displayed as boxplots. In plot 1, it can observe that the trees show a large range of scores for defaulters, while the non-defaulters show lower ranges and some positive outliers. Interestingly, the ranges of the individual trees scores show a high discriminatory power in this plot. A reason may be that some trees always generate a low score and cannot predict a default at all, due to the generation of the individual data sets using the bootstrap method. The STD's over the trees shown in plot 2 show a similar relation of non-defaulters to defaulters with lower values. Notice that the defaulter distribution of the STD's is not skewed, in contrast to the defaulter distribution of the STD's. By comparing the skewness of the two distributions for defaulters we can interpret that the high values for the range may come from outliers, instead of a wide spread of all scores. Otherwise the corresponding STD distribution would show the same skew.

After conducting the model comparison on the 3-month train set, it is interesting to evaluate the performance on the test set for the first time. Here, it can be observed if the models overfit the data, which may result from the intensive model selection on the train set.

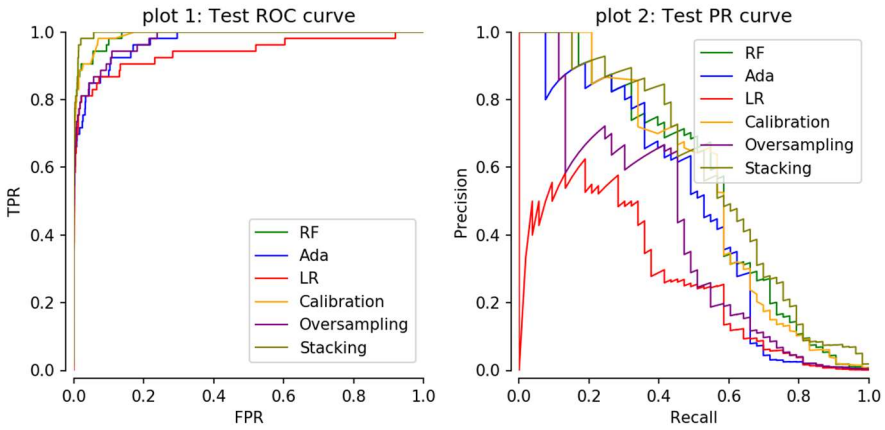

Figure 4.5: ROC and PR curves of the model comparison on the 3-month test set.

For the model comparison on the test set, the models are trained on the train set and score the test set. The ROC and PR curves of the model comparison are shown in Figure 4.5. Due to the small number of data samples in the test set, fewer unique scores exist and the ROC and PR curve have fewer data points to plot. Hence, the less smooth appearance of the curves. Nonetheless, the curves are analysed the as previously. In plot 1 we generally observe better ROC curves, than we have on the train set. The best curves have an almost rectangular shape, which implies a perfect prediction quality. The PR curves in plot 2 show the bad performance of the LR and Over-sampling approach, as observed on the train set already. The other models have similar curves, with the Ada having a slightly worse performance.

	AUC-ROC	AUC-PR	Brier	Precision	Recall	F1	Defaults
RF	0.990	0.545	0.001	0.688	0.415	0.518	32
Ada	0.971	0.476	0.001	0.867	0.245	0.382	15
LR	0.940	0.259	0.001	0.600	0.170	0.265	15
Calibration	0.990	0.540	0.001	0.846	0.208	0.333	13
Oversampling	0.976	0.402	0.072	0.621	0.340	0.439	29
Stacking	0.996	0.577	0.001	0.719	0.434	0.541	32

Table 4.3: Results of the model comparison on the 3-month test set. For the calculation of the threshold depending metrics the individual c was used as determined on the train set.

We show the results of the metrics of the model comparison using the 3-month test set in Table 4.3. Additionally, we show the number of predicted defaults in the last column. Interestingly, all metrics improved compared to the train set, except for the LR. This improvement of the scores is usually not expected, but we do not consider it as an issue. Rather it is an indicator, that the models did not overfit the data. Otherwise, the metrics would be worse on the test set than on the train set. A reason for the improvement might be that the models are trained on more data, when predicting the test set. Remember that a 3-fold CV is used to obtain scores for all samples in the train set, hence the train data to obtain scores on the train set is $\frac{2}{3}$ of the train data to obtain scores on the test set. Although, we use a stratified CV, it is still possible that the CV with the random seed splits the train set in folds of different clusters. The confusion matrixes for the test set are shown again in the appendix in Table 7.2.

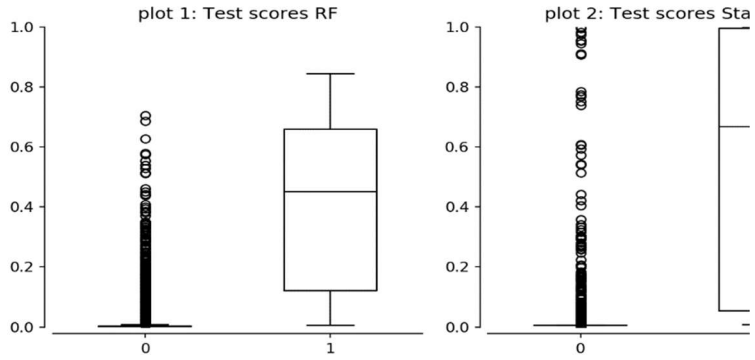

Figure 4.6: Boxplot comparison of the RF and Stacking scores on the 3-month test set.

As previously, the conditional distributions of the scores on the 3-month test set are shown in Figure 4.8. Again, the RF scores are displayed in plot 1 and the scores of Stacking in plot 2. The distributions show only small differences compared to the distributions from the train set. The boxes, as well as the whiskers are located at similar values. This is the desired behaviour of the distributions, since only then the model generalizes well on the unseen data of the test set. The major observable difference is the smaller number of positive outliers for the non-defaulters for RF and Stacking. This is due to the fewer data samples being in the test set than in the train set. Generally, both distributions show a good discriminatory power, since the IQR is not overlapping.

Concluding, we have identified the RF as our model of choice for the application using the 3-month data set. The approaches we have considered for an improvement of the prediction quality did not have the expected results. The calibrated models showed smaller values of AUC for ROC and PR curves, although the brier score could be slightly improved. The comparison of train and test scores has proven that we did not overfit the data and have created a robust model, which generalizes well. Hence, using the RF is recommended on the given data set. For the comparison with the FraSpa scores we have set the focus on the use of models without any further processing of the data.

4.2. Recursive feature elimination

The large number of features with low univariate discriminatory power (Chapter 3.3.3) leads to the assumption, that a model does not need to consider all features when predicting an outcome. Rather the model should evaluate the strong features and leave all other features unconsidered. Features with a low discriminatory power may decrease the prediction quality, since they add noise to the data. To test if this holds true on the given data set, a recursive feature elimination is performed on the 3-month data set. Weak features are recursively eliminated from the train set. 20 runs are considered, where in each run the ≈27% weakest features are removed from the data set. Since we have 690 features in the data set, dropping the mentioned percentage each run results in the following list:

$$dropped\ features = \{0, 183, 317, 416, 489, 542, 581, 610, 631, 647, 659, 667, 673, 678, 681, 684, 685, 687, 688, 689\}.$$

The values in the list are sorted in an ascending manner, since we need to start by fitting the model to all features. Thus, the first element in the list is 0. For each number of dropped features d in the list, we fit a model to the remaining features after dropping the d least important features, according to the feature importance of the RF.

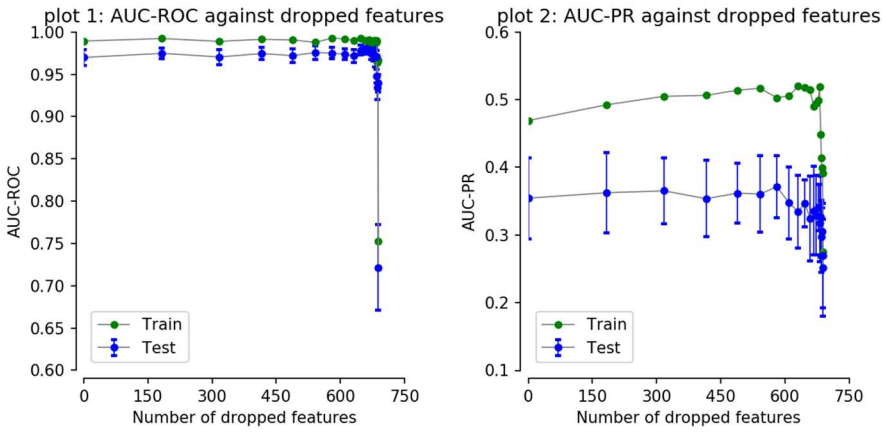

Figure 4.7: Results of the recursive feature elimination on the 3-month train and test set. We plot the AUC-ROC and AUC-PR against the number of dropped features. Train scores are shown in blue, test scores are shown in green. The STD of the train metrics over a 3-fold CV is displayed by the blue error bars.

The results of the recursive feature elimination are shown in Figure 4.7. In plot 1 the AUC-ROC of each run is shown, plot 2 shows the AUC-PR scores. The number of features dropped in each successive run is decreasing, hence the distance on the x-axis between the data samples is decreasing as well. As we expected from the univariate analysis of the features, the performance of the model stays on a good level until most of the features are dropped. This indicates that a prediction based on the few strongest features gives already a good discriminatory power to the model. The train score for the AUC-ROC value is even increasing just before the score drops. This may result from the removal of less important features creating noise in the data. Judging by the AUC-ROC score, using only the strongest features would improve the prediction quality. Interestingly, the metrics on the test data are much higher compared to the metrics from the train data. This behaviour has been observed before, when comparing train and test metrics. Again, it is not considered an issue, but rather an indicator for not overfitting the data. The reason may be that the model is trained on more data when predicting the test set.

Note that the dropped features present a change in the complexity in the data set. Therefore, the model should be adapted to each data set, by tuning the model's parameters. However, since the focus of this thesis is not the feature elimination, we use the same model for each data set.

4.3. Comparison with the FraSpa scores

In this chapter we compare the internal scores of FraSpa to the scores created with the RF and similar parameters as determined in Chapter 3.3.5. For the comparison of the scores we need to adjust the data set and the FraSpa scores. The following aspects must be kept in mind for this comparison chapter:

- The FraSpa scores present the default probability of a certain group of customers. These groups are created to fit the corresponding default probability. The scores produced by the machine learning models present the score of an individual customer. Hence, we need to change the meaning of the FraSpa scores and consider the FraSpa default probability as individual scores. This is not the intended use of the FraSpa scores and might lead to biased results.

- Due to the nature of the FraSpa scores, a transformation is necessary to make them comparable to the machine learning scores Since the FraSpa scores have a maximum PD of 45%, we need to scale the scores in [0, 1]. Using the unscaled scores with a threshold of 0.5 would results in no predictions for the positive class. For the normalization of the FraSpa scores we use the same procedure as do for feature scaling, as presented in Chapter 2.8. Hence, we lose the probabilistic meaning in the FraSpa scores. Note that for creating the calibration plots, the non-scaled FraSpa PD's are considered.
- The FraSpa scores use a default of a credit within a 12-month period as target. Therefore, we need to adjust to the target of the data set used for the predictions accordingly. Unfortunately, this leads to the loss of the described data transformations in Chapter 3.3.1, since the historical data is not available anymore. Hence, the informational advantage of the historical features is lost. It can be expected, that using the same historical data as it has been done for the 3-month data set, would result in better results. Nevertheless, we use the available snapshot data to predict a default within the next 12 months.
- The data used for the credit scoring is not necessarily the same as we use for the application of the machine learning methods.

Rating	PD [%]	Rating	PD [%]
1 (AAA)	0.01	7	0.90
1 (AA+)	0.02	8	1.30
1 (AA)	0.03	9	2.00
1 (AA-)	0.04	10	3.00
1 (A+)	0.05	11	4.40
1 (A)	0.07	12	6.70
1 (A-)	0.08	13	10.00
2	0.12	14	15.00
3	0.17	15	20.00
4	0.30	15 (B)	30.00
5	0.40	15 (C)	45.00
6	0.60	16-18	Default

Table 4.4: Master scale of the Sparkassen. (Sparkasse KoelnBonn 2016).

The *master scal*e translates the internal rating of the FraSpa of a borrower in a PD. Table 4.4 illustrates the ratings and the corresponding PD's, as used by the Frankfurter Sparkasse. A borrower that is rated with a 16 or worse is considered defaulted by FraSpa.

Since we change the target of the data, as well as the period of the snapshot, we check the number of defaults in the 12-month data set. Again, 22 categorical features and 142 continuous features are considered before applying the dummy transformation. Afterwards, the total number of features is 656. The decrease number in continuous features arises from the missing historical features, which were added to the 3-month data set. Again, the data set is split randomly 80:20 for train and test set.

Attribute	Training data	Test data
Number data samples	196 295	49 182
Number defaults	648	139
Default frequency	0,330%	0,283%

Table 4.5: Default statistics on the 12-month data sets.

The resulting number of data samples, as well as the number of defaults in each set are shown in Table 4.5. Due to the extended target period of 12-months, the number of defaults in the data set increases. Interestingly, the total number of defaults is lower than expected. Since the target period is now 4 times the 3-month target period, we would expect 4 times more defaults. However, the 12-month data set has just above 3 times the number of defaults. This indicates that the number of defaults in the 3-month period is unusually high, compared to the 12-month data set. Also, the number of default frequencies of the train and test set are not equal. This is possible due to the random split of the data set. Nevertheless, we continue using the data set for the comparison of the FraSpa and machine learning scores.

We select the LR and the RF as machine learning models for this comparison. The LR is considered, since the Logistic Regression is commonly used for credit scoring (see Chapter 1.2). Moreover, the RF is selected, which offered the best prediction quality on the 3-month data set comparison in Chapter 4.1, as well as over a grid search on the 12-month data set. We refer to the scores of the Frankfurter Sparkasse as FraSpa.

Changing the data set in an elementary manner like it is done for the comparison, makes an adjustment of the model's parameters necessary. Therefore, we compute again RF with various parameters, to find the best parameters for the task. The selected parameters are:

$$\texttt{n\_estimators} = 80$$
$$\texttt{max\_depth} = 5$$
$$\texttt{max\_features} = \sqrt{n\_features} \approx 26$$
$$\texttt{min\_samples\_split} = 2$$

First, the performance of the models on the train set is investigated. Again, a 3-fold CV is used to obtain predictions on the entire train set. In a second step, the models are fit to the train set to predict the data samples in the test set. To compare the performances of the RF to another machine learning model, the results of the LR are plot as well. The parameters of the LR were optimized with an exhaustive grid search as well.

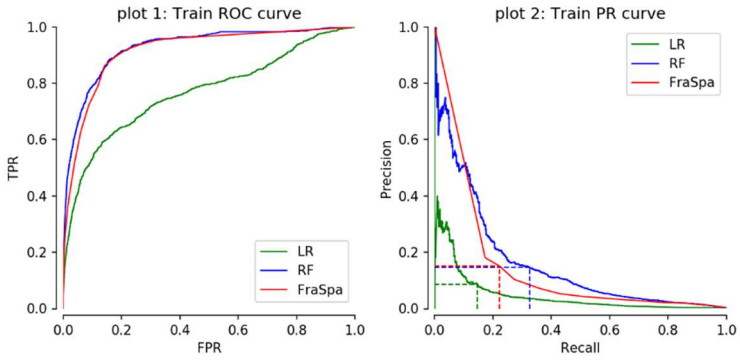

Figure 4.8: ROC and PR curves for LR, RF and FraSpa on the 12-month train set. We show $c = 0.5$ by the dotted lines in the PR plot.

The ROC and PR curves of LR, RF and FraSpa on the train set are plot in Figure 4.8. In plot 1 the ROC curves are shown, in which the RF and FraSpa show a high discriminatory power. The two curves are hardly distinguishable, due to rather similar values for TPR and FPR over the various thresholds c. The LR ROC curve shows a much worse performance. In plot 2 the PR curves are shown. Here we can easily observe the different performances for precision and recall. For high precision values, the FraSpa has an advantage over the RF. This is expressed by the larger recall values for the same precision values. For lower precision values, the RF has an advantage. Note that both FraSpa curves show multiple buckles, due to the discrete distribution of scores, as we have seen in the master scale in Table 4.4. We have seen in Chapter 2.9.2 that for both, the ROC and the PR curve, metrics are plot for all probabilistic predictions. Since the FraSpa scores are discrete, the number of unique thresholds is lower compared to a complex machine learning algorithm like the RF. Hence, the buckets in the FraSpa curves. Additionally, this decreases the information received from both the ROC and PR curves of FraSpa, since the data points are interpolated. However, in interpolated sections of the curve, it is not known, if the model would perform that way. An example of good visibility is the section of the FraSpa PR curve with recall values in [0, 0.1], where it has an advantage over the RF PR curve.

	AUC-ROC	AUC-PR	Brier
LR	0.767	0.046	0.0077
RF	0.927	0.159	0.0082
FraSpa	0.916	0.145	0.0083

Table 4.6: Metrics for LR, RF and FraSpa scores on the 12-month train set.

Since c has not been chosen yet, we first compute metrics independent from any c. Hence, Table 4.6 shows the comparison of the three models on the AUC and the Brier. We see the AUC-ROC and AUC-PR are as we expected from Figure 4.8. The LR performs the worst on the train set, while the RF scores obtain the largest AUC scores on the ROC and PR curves. Hence, we conclude that the RF provides the highest discriminatory power. Interestingly, the Brier scores show a different pattern. Here, the LR shows the lowest score, which indicates the best interpretability of the scores as default probabilities. However, considering the AUC values, the classification performance is far from good. The second-best Brier was obtained by the RF, the worst Brier was obtained by the FraSpa scores. The FraSpa and RF show only little difference in all metrics. Generally, the results of the metrics much lower compared to the 3-month data set, which we have investigated in Chapter 4.1. One factor for the worse scores is that defaults within a 12-month period are more diverse. Some borrowers default within the next month, some others only default in a year. As a result, the indicators are more diverse, leading to less

solid predictions and a worse performance of the model compared to the 3-month target. Another factor is the absence of the historical features in the data set, which we could not use due to the limitation of the data source.

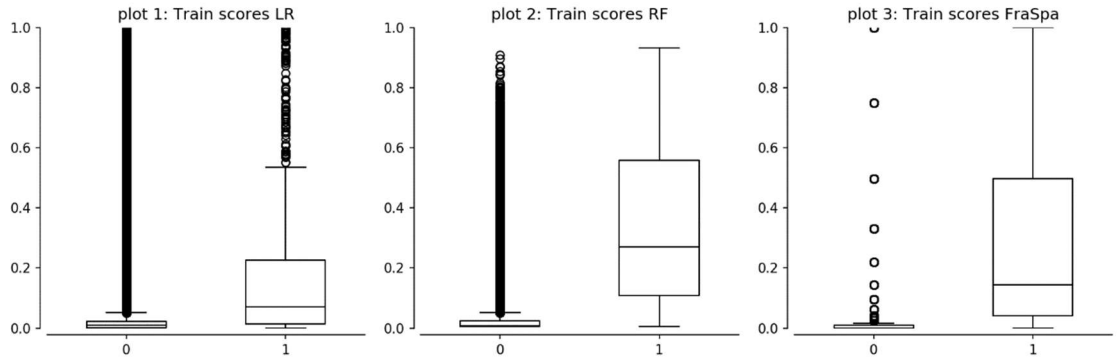

Figure 4.9: Boxplot comparison for LR, RF and FraSpa scores on the 12-month train set. Non-defaults are encoded as 0, defaults are encoded as 1.

To compare the distribution of the models' scores, we display conditional boxplots which show the scores separated by the true target in Figure 4.9. The first plot shows the LR scores, plot 2 contains the RF scores and plot 3 displays the FraSpa scores. All are separated by default and non-default. On the first sight, all scores separate the non-defaults and defaults relatively well, since the boxes of non-defaulters and defaulters hardly overlap. Clearly, the LR scores show the lowest discriminatory power, due to the generally low score for defaulters and many positive outliers for non-defaulters, compared to the other models. This results in a slight overlap of the IQR. Therefore, the LR would predict relatively many false-negatives and many false-negatives, which is obviously not desired.

The good discriminatory power of the RF can be observed by the difference in the conditioned boxplots. The upper quartile of the non-defaults is lower than the lower quartile of the defaults. Hence the IQR does not overlap. The non-defaults have outliers with a high score. This is reasonable, since the RF does classify credits incorrectly as defaults. However, there are fewer scores with a high score > 0.8 than we have seen for the LR scores for non-defaulters. The scores of the defaulters are generally higher than the LR scores, since quartiles and median, as well as the upper whisker are situated at higher values. The upper whisker of the LR is at the same score value as the upper quartile of the RF scores, indicating a huge difference in discriminatory power.

The boxplot of the FraSpa scores shows a good discriminatory power as well. Since it was necessary to scale the FraSpa scores, the scores fill the entire range in [0, 1]. Compared to the RF scores, the FraSpa scores show lower scores, as well as fewer positive outliers, for non-defaults. This means the RF tends to predict more non-defaults incorrectly as defaulters, resulting in the prediction of more false-positives than the FraSpa model. The median and quartiles of the FraSpa non-defaults are lower compared to the RF scores. The scores of the defaulters show generally a lower score compared to the RF scores, which can be observed by the median and the lower quartile of the default boxplots. Hence, the FraSpa model tends to classify more borrowers as false-negatives. Comparing the median to the upper quartile, we observe the range to be larger for FraSpa than RF. This indicates a stronger skew of the FraSpa score distribution, which adds to the assumption that the FraSpa scores have more false-negatives than the RF scores. Therefore, we conclude that the FraSpa scores are generally lower for both, the non-defaulters and the defaulters, compared to the RF scores.

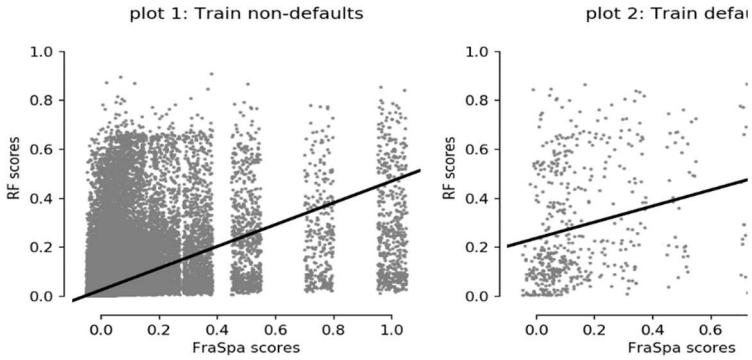

Figure 4.10: Scatter comparison RF against FraSpa scores on the 12-month train set. The FraSpa scores are scaled and jittered on the x-axis due to the discrete manner. We plot a linear regression over the non-jittered data samples.

For a visual investigation of the individual scores, we consider a scatter plot, where the RF scores are plot against the FraSpa scores. For each data sample, the FraSpa score is plotted on the x-axis and the RF score is shown on the y-axis. This should illustrate how much the scores of the individual data samples differ from each other. Again, the plots are separated by the actual default of the data sample, but for this plot we change to two separated plots, since the visualization in one plot would be too confusing. In plot 1 non-defaulted borrowers are shown, plot 2 shows the defaults. Unfortunately, the FraSpa scores are discrete, which justifies the data points being arranged in vertical lines. Both plots provide the same general alignment of the data points. The less compact arrangement of data samples in plot 2 is due to the imbalanced ratio of 0.3% between non-defaulters and defaulters on the 12-month target.

A linear relation of the scores would result in an angle bisecting formation of the scores. Since such a relation is hard to see with the discrete FraSpa scores, we consider a linear regression to illustrate the relation. In plot 1 the linear regression starts in [0, 0], which indicates that the scores of RF and FraSpa are similar for lower scores. The linear regression ends at a much lower value of the RF scores, than FraSpa scores. This indicates that many samples are higher scored by FraSpa than the RF. Since plot 1 shows non-defaulting borrowers, a low score is desired. For defaulters, the start of the linear regression is at about [0, 0.2], which indicates that the RF scores are higher than the rather low FraSpa scores for defaulters. This implies that the RF predicts less false-negatives.

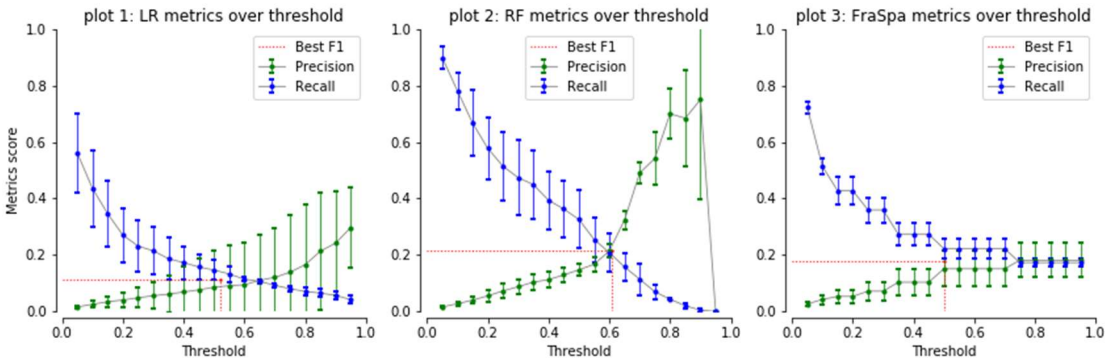

Figure 4.11: Precision and Recall for LR, RF and FraSpa scores against c on the 12-month train set. The error bars present the STD over a 3-fold CV of the metric. We compute the highest F1 score depending on c. The optimum c with the corresponding F1 score is plot as red dotted line.

The trade-off between precision and recall against the threshold c is shown in Figure 4.11. Again, we consider the scores we obtained from the LR, RF and FraSpa. The general interpretation is similar as in Chapter 4.1. Again, the F1 score is computed for all unique scores and the maximum F1 score is displayed as the red dotted line. The value of the highest F1 score can be read from the y-axis and the corresponding c from the x-axis. As expected, the RF scores provide the highest value for the F1 score, followed by the FraSpa scores and lastly the LR scores. Looking at the STD's of the metrics, we observe that precision and recall scores with larger values have a larger STD as well. For the LR scores and the RF scores, the optimal c also provides a good trade-off between the STD's of precision and recall. This adds to the choice of selecting the threshold according to the maximum F1 score. The found values for c can now be used to investigate the results by metrics, which are depending on the prediction $\hat{\boldsymbol{y}}$.

	c	Precision	Recall	F1
LR	0.52	0.090	0.145	0.111
RF	0.61	0.232	0.205	0.218
FraSpa	0.50	0.151	0.222	0.180

Table 4.7: Metrics using an optimal threshold on the 12-month train set. We show the optimum threshold c and the corresponding results of the metrics, which depend on the chosen c.

The metrics for each of the models using the c with the maximum F1 score are shown in Table 4.7. We notice that the chosen c of the RF scores is higher compared to the other models' c's. This might be the reason for the precision score of the RF being higher than the recall score, which is a difference compared to the other two scores. However, since this c provides the highest F1 score, we continue without adjusting the c's.

		LR Prediction		RF Prediction		FraSpa Prediction	
		0	1	0	1	0	1
Actual	0	194695	952	195206	441	194838	809
	1	554	94	515	133	504	144

Table 4.8: Confusion matrix of the scores using the optimized c on the 12-month train set. Non-defaults are encoded as 0, defaults are encoded as 1. The confusion matrixes of all three models are shown next to each other.

The shown plots provided insights on the distribution of the scores. Nevertheless, reading absolute number of predictions from the distributions is hard, also since c was not determined. Hence, we consider the confusion matrix as a mean to obtain the absolute numbers of the predictions in Table 4.8, as we have introduced in Chapter 2.9. Notice that the non-defaults and defaults are encoded as 0 and 1, due to a better readability of the table. For the same reason, the confusion matrixes of the models' predictions are shown next to each other. $\hat{\boldsymbol{y}}$ is created for each model with its individual c. Clearly, the 12-month target is a difficult task to predict, as we can see by the large numbers of false-positives and false-negatives, as well as by the comparably small number of true-positives for all three models. Notice that the FraSpa scores have predicted the highest number of true-positive data samples. Comparing the number of false-positives, all three models do not differ a lot, besides a slightly higher number of misclassified negatives for the LR. The most significant difference resides in the number of false-positives. While the FraSpa predictions and the LR predictions have just under two times their number of false-negatives, the RF has less false-negatives than false-positives. Hence the relatively high precision score of the RF.

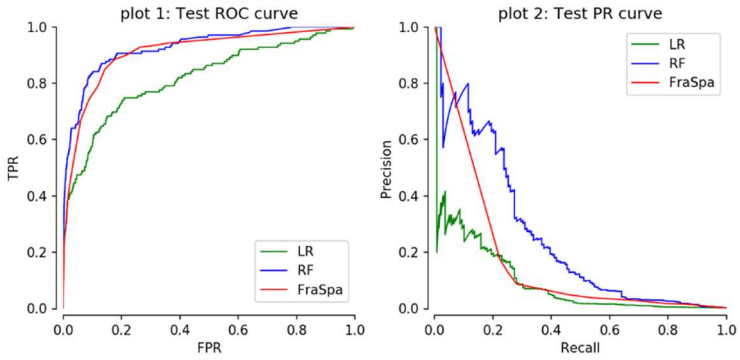

Figure 4.12: ROC and PR curves for LR, RF and FraSpa scores on the 12-month test set.

After finishing the investigation of the scores on the train set, the same procedure is applied to the test set, except for the investigation of the optimal c. Selecting the optimal c on the test set would make us loose the independence to the test set and therefore create biased results.

The results of the RF on the test set are more important than on the train set. Since we have kept the independence of the test set, the model is not biased, and the test set presents an unseen data set, which would be the situation in a real-world environment as well. The ROC and PR curves are plot for the comparison on the test set in Figure 4.12. Again, the ROC plot does not differ much for the FraSpa and RF scores. Yet, the PR curves show significant differences of the FraSpa and RF scores, like the PR curves on the train set shown in Figure 4.8. The PR curve of the RF has increased compared to the train set, while the FraSpa does not differ a lot. This is reasonable, since the FraSpa scores are fixed and not dependent on our distribution of data samples on the train and test set. The RF does take advantage from the increased number of data samples for fitting, which may be a reason for the better curves.

	AUC-ROC	AUC-PR	Brier
LR	0.820	0.091	0.0054
RF	0.928	0.249	0.0076
FraSpa	0.911	0.164	0.0085

Table 4.9: Comparison of the LR, RF and FraSpa scores on the 12-month test set.

The AUC values and the brier scores of the scores are shown in Table 4.9. We observe an improvement for the LR and RF metrics compared to the train set. The LR shows the worst performance of the three models. However, the brier is better than any other model. As expected, the FraSpa scores do not change from the train to the test set. Since the scores are obtained from an external source upfront, the FraSpa scores are entirely independent of our splitting approach.

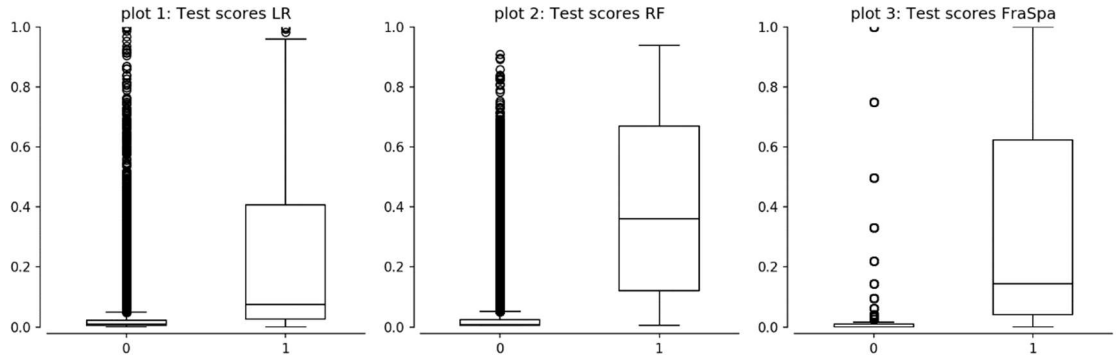

Figure 4.13: Boxplot comparison the LR, RF and FraSpa scores on the 12-month test set.

In Figure 7. the conditional boxplots of the scores on the test set is plotted. The LR and RF scores show a slight improvement in discriminatory power, compared to the train set. The median and quartiles of the defaulters are higher on the test set, which allows for a better separation of the classes. The FraSpa scores do not show a difference compared to the train set. Again, the improvement from the train to the test set proofs our RF model to generalize well on unseen data.

The scatter plot of the scores and the complementary linear regression are shown in the appendix in Figure 7.1, since the changes compared to the same plot of the train set in Figure 4.10, are merely minor.

	Precision	Recall	F1
LR	0.202	0.180	0.190
RF	0.276	0.324	0.298
FraSpa	0.128	0.252	0.169

Table 4.10: Results of the scores using an optimal threshold on the 12-month test set. The c was determined on the train set.

The metrics from the 12-month test use the *c* determined on the train set and are shown in Figure 4.10. The improvement of the metrics of the models we applied (LR and RF) from the train to the test set, makes the FraSpa scores to have the worst precision and F1 score. However, the FraSpa recall score is still higher than the LR recall. Again, the RF has the highest F1 score, as well as the highest precision and recall score. Hence, for the given task, the RF offers the best prediction quality of the considered models.

		LR Prediction		RF Prediction		FraSpa Prediction	
		0	1	0	1	0	1
Actual	0	48944	99	48925	118	48804	239
	1	114	25	94	45	104	35

Table 4.11: Confusion matrix of each models' scores on the 12-month test set. The c was determined on the train set. Non-defaults are encoded as 0, defaults are encoded as 1.

For the absolute numbers of the predictions, consider the confusion matrixes in Table 4.11. Again, the matrixes of all three models are shown next to each other for a better readability. The RF is delivering the highest number of true-positives and the lowest number of false-negatives. Only the LR has a lower number of false-positives.

Now, the interpretation of the scores as default probabilities is further investigated. This is partly done by the previously shown Brier scores already. However, for a visualization the calibration plots of the scores is considered. Therefore, we split the scores into 10 bins of equal length and compute the default frequency, as explained in Chapter 2.6. Since the scaling of the FraSpa scores takes the probabilistic meaning out of the scores, the non-scaled FraSpa scores are considered for the calibration plots. We start with the train scores.

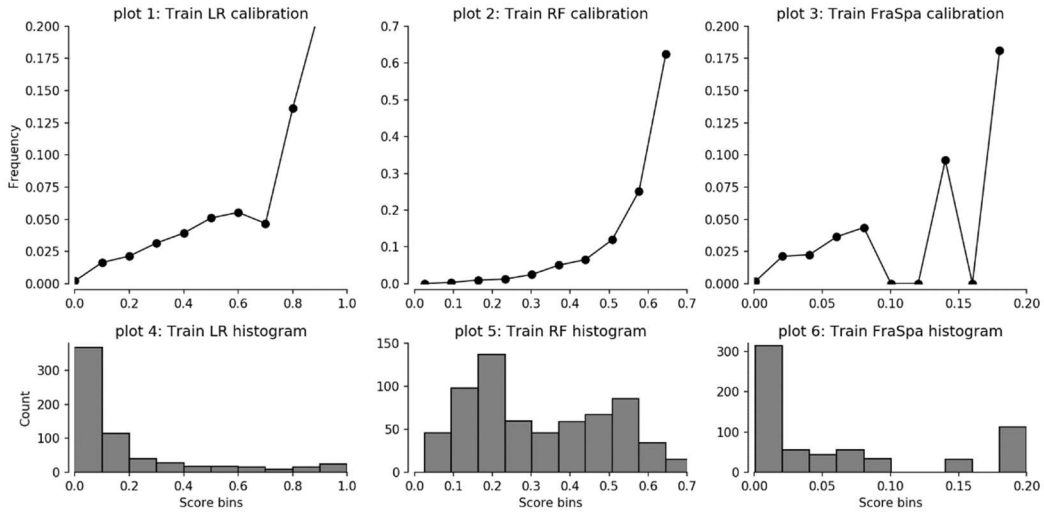

Figure 4.14: Calibration plot of LR, RF and FraSpa scores on the 12-month train set. We consider 10 bins of equal length. The non-scaled FraSpa scores are considered for this comparison.

In Figure 4.14 we show the default frequency as well as the absolute number of defaults for each bin of the train scores. In plot 1-3 the default frequencies are shown, plot 4-6 show the absolute number of defaults in each bin. The x-axis is shared by column of the plots, since we want to compare the default frequency with the corresponding absolute number of defaults for each model. Ideally, a proportional relation of the scores and the default frequencies is visible. This would mean the scores of the model can be used as default probabilities. Such a relation does not exist for any model. For the RF, the default frequency increases exponentially with an increase of the bins. Note that the RF maximum score is just about 0.7. In the FraSpa default frequency shown in plot 3, as well as the number of absolute defaults in plot 6, 3 bins without any scores can be observed. Again, the discrete nature of the FraSpa scores makes it hard to compare the scores here. For the bins without scores, the FraSpa does simply not have a possible score. Consider the master scale in Table 4.4, which shows that no borrower can obtain a score belonging to the 3 empty bins. However, considering the non-empty bins, the default frequency shows a rather proportional relation over the score bins. Also notice the range of the FraSpa scores of [0.0, 0.2], which is much smaller than the RF range. Additionally, the non-scaled scores exploit that no credit of FraSpa has observed an internal scoring larger than 0.2. The linear relation of the FraSpa scores allows us to consider them as default probabilities. However, this can be expected, since the scores are given to a group of borrowers and adjusted if necessary, to provide the proportional relation.

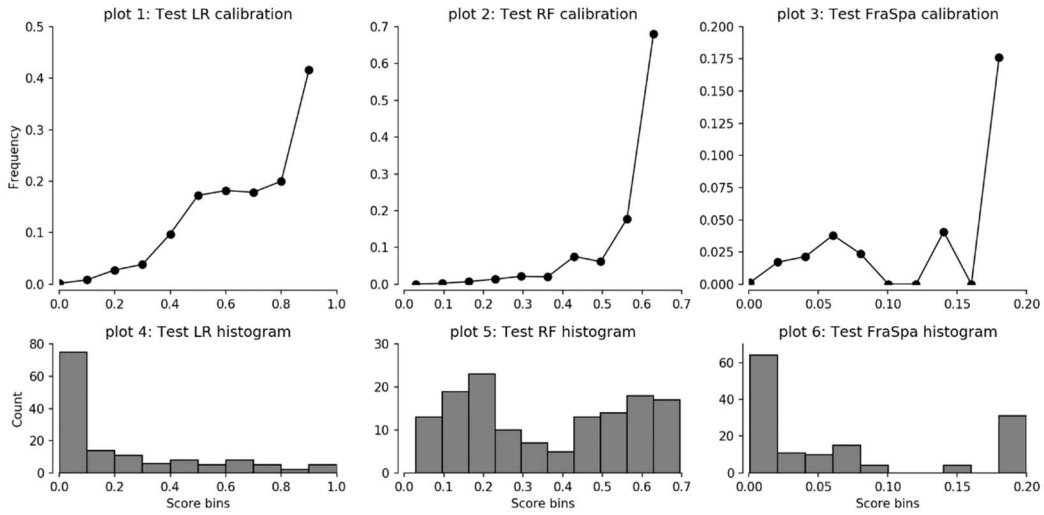

Figure 4.15: Calibration plot of LR, RF and FraSpa scores on the 12-month test set. We consider 10 bins of equal length. The non-scaled FraSpa scores are considered for this comparison.

Like the previous plot, the default frequencies and the absolute number of defaults for the 12-month test set are shown in Figure 4.15. Again, an exponential increase of the default frequency over the score bins for the RF scores can be observed. Naturally, the FraSpa scores, shown in plot 2, display the same 3 empty bins as on the train set. A difference between the train and test set is the increase of variance in the frequencies. The smooth shape of the default frequency curve has vanished, but the general tendency remains unchanged. A reason may be the fewer number of defaults in the test set, compared to the train set. Looking at the absolute number of defaults, similar distributions as on the train set can be found. Note that the calibration techniques from Chapter 2.7 have been applied to the LR and RF scores as well. However, the calibration was not improved, and the plots are not shown.

As a last comparison, the values of the metrics of the models are compared to each other. Since the RF is the best performing machine learning model, we show the change of the RF over the other models' metrics. We calculate the change as

$$change_{metric} = \frac{\mathrm{RF}_{metric} - model_{metric}}{model_{metric}} * 100\%,$$

with $model \epsilon \{$LR, FraSpa$\}$ and $metric \epsilon \{$precision, recall, F1$\}$. The change for precision, recall and F1, is calculated separated by the train and test set.

	Train [%]	Test [%]
Precision	157.84	36.93
Recall	41.49	80.00
F1	96.14	56.75

Table 4.12: Change by metric of RF and LR scores on the 12-month train and test set. We show the difference of the RF metrics to the LR metrics in percent of the LR metric for train and test set.

The change of the RF metrics over the LR metrics are shown in Table 4.12. The RF has achieved better results of on all metric on the train and test set, hence all changes are positive. However, we observe that the F1 metric improvement is lower on the test than on the train set. Hence, the LR metrics have

improved even more on the test set, compared to the RF metrics. The application of the LR might result in relatively good performance in a real-world scenario as well.

	Train [%]	Test [%]
Precision	53.35	116.13
Recall	-7.64	28.57
F1	21.01	75.83

Table 4.13: Change by metric of RF and FraSpa scores on the 12-month train and test set. We show the difference of the RF metrics to the FraSpa metrics in percent of the FraSpa metric for train and test set.

In Table 4.13 the change of the RF metrics compared to the FraSpa metrics are shown. We notice the strong increase of the precision score, which can be expected since the RF is stronger on precision than on recall, inversely to the FraSpa. Consider that a model with high precision can be considered more trustable, since its positive prediction are more often correct. Also, the change on all metrics increases when considering the test set. Similar metrics can be expected when using the RF model in a real-world scenario, where borrowers need to be scored on a regular basis. Since the recall score of the FraSpa is high already, the RF with its relatively balanced precision and recall cannot keep the same level. Thus, the worse recall score of the RF on the train set. The improvement in prediction quality on the test set improves the recall score and results in the positive change against the FraSpa recall.

Concluding the calculated changes in metrics, the applied RF model has a 116% higher precision compared to the FraSpa scores. Hence, a predicted default of RF is more than twice as likely to be an actual default, than a predicted default of FraSpa. The same RF scores have a 29% higher recall than the FraSpa scores. Hence, the RF predicts about $\frac{1}{3}$ more actual defaults correctly, which means less defaults are not found by the model. Since recall and precision have improved, the F1 scores improves as well. Here, we have a change of 76% compared to the FraSpa scores.

4.4. Conclusion of the comparison

In this chapter the machine learning scores of the RF and the LR were compared to the FraSpa scores. Out of all techniques, the original RF delivered the best performance, on the 3-month target data set and the 12-month target data set. The prediction quality was first measured by threshold independent metrics, like AUC-ROC and AUC-PR. Afterwards, the threshold delivering the highest F1 score was selected. When compared to the traditional scoring method of the Frankfurter Sparkasse, we could achieve significant improvement in all metrics, especially on the test set. Hence, the RF separates the non-defaulting borrowers and defaulting borrowers better than the currently implemented scoring system, using a 12-month target. Interestingly, we notice better performance on the test set than on the train set throughout the thesis.

Due to the great flexibility of the RF, features do not have to be scaled, which is an advantage over other methods. However, the transparency of the RF is curtailed. Since each of the trees in the ensemble has a different structure, the decision path of a single borrower is hard to comprehend. An alternative is the evaluation of the importance of the features for the RF. This has drawbacks as well, since the importance does not provide information, whether the feature has a positive or negative impact on the decision.

Notice, that an almost proportional relation was found between the scores of the RF and the default probability on the 3-month data set. However, this was not true on the 12-month data set. Hence, the scores can only partly be interpreted as default probabilities. Calibration techniques could improve the linearity slightly, but led to worse metrics besides the Brier.

Hence, we recommend using the RF as a validation model, which exists in parallel to the current credit scoring method. This way, the high prediction quality of the RF can be used, without the need to understand each decision. Since the conformity of a machine learning model with the IRB-A requirements for credit scoring models is not certain, the RF could be used as a warning system, which internally flags the potentially defaulting borrowers. These borrowers could then be further investigated. Since we have seen the high default probability of borrowers with high scores of the RF, the threshold could be set on a high level to reduce the number of false-positives. This would result in a high level of trust in the prediction of the responsible employees. However, any other threshold can be selected depending on the desired ratio of precision and recall.

5. Conclusion

In this thesis we pursued the goal to predict if a borrower defaults in a defined future period. It has been shown that modern ensemble methods are able to improve the prediction quality compared to the common logistic regression. In all comparisons the Random Forest showed the best performance.

To understand the motivation and background of this thesis, the idea of credit scoring and its components was introduced in Chapter 1. Furthermore, the suitability of machine learning methods for solving the task were pointed out. In Chapter 2 we investigated the machine learning process and specific types of models for the credit scoring task. Due to the success of ensemble methods in similar applications, we focused on such in Chapter 2.5. Additionally, sampling techniques for imbalanced data sets we investigated. Since the scores should be interpreted as default probabilities, calibration methods and the Brier score were analysed.

The application of the mentioned methods and techniques was shown in Chapter 3. First, the data set and the preprocessing was described. In Chapter 3.2 we used a *supporting model* to determine, whether there are indicators before a credit defaults and how they differ over different periods to the occurrence. We found the amount of overdraw of an account to be the most important of the selected features. The trend of the overdraw was of the same importance. Hence, borrowers that increase their overdrawn amount are more likely to default. The supporting model has shown, that historical features and the trends are relevant for the classification. Therefore, we aggregated the historical features and applied the new feature *time-to-empty-pockets*, which estimates the time until a borrower's account balance reaches zero. As a result, the 3-month data set was obtained. We found the Random Forest to deliver the best performance. Hence, a detailed investigation of the model was conducted in Chapter 3.3.5, where the ideal parameters were determined.

In the comparison in Chapter 4.1 the different machine learning approaches and modes were compared. Again, the Random Forest showed the best results. Hence, a Random Forest was selected for the comparison with the FraSpa scores on a 12-month data set in Chapter 4.3. The Random Forest showed an improvement over the existing scores. Using an objectively optimized threshold, the Random Forest improved the F1 score of the FraSpa scores by 21% on the train set and 76% on the test set.

All in all, the application of machine learning models for credit scoring tasks has shown positive results throughout the thesis. Considering the small number of defaults and the missing historical features in the 12-month data set, the improvement of the machine learning scores over the traditional scoring methods show that machine learning can deal with less than perfect conditions. This demonstrates the huge potential for machine learning models. The performance of the machine learning methods is expected to further improve on larger data sets and more defaulters.

6. References

Abellán, J.; Castellano, J. G. (2017): A comparative study on base classifiers in ensemble methods for credit scoring. In *Expert Systems with Applications* 73, pp. 1–10.

Ala'raj, M.; Abbod, M. F. (2016): Classifiers consensus system approach for credit scoring. In *Knowledge-Based Systems* 104, pp. 89–105.

Altman, E. I. (1968): Financial ratios, discriminant analysis and the prediction of the corporate bankruptcy. In *The Journal of Finance* 23 (4), pp. 589–609.

Angelini, E.; Di Tollo, G.; Roli, A. (2008): A neural network approach for credit risk evaluation. In *The Quarterly Review of Economics and Finance* 48 (4), pp. 733–755.

Baesens, B.; van Gestel, T.; Viaene, S.; Stepanova, M.; Suykens, J.; Vanthienen, J. (2003): Benchmarking state-of-the-art classification algorithms for credit scoring. In *Journal of the Operational Research Society* 54 (6), pp. 627–635.

Batista, G. E.; Prati, R. C.; Monard, M. C. (2004): A study of the behavior of several methods for balancing machine learning training data. In *ACM SIGKDD explorations newsletter* 6 (1), pp. 20–29.

Bauer, E.; Kohavi, R. (1999): An empirical comparison of voting classification algorithms: Bagging, Boosting, and variants. In *Machine Learning* 36 (1/2), pp. 105–139.

BCBS (2005): An Explanatory Note on An Explanatory Note on the Basel II IRB Risk Weight Functions. July 2005.

BCBS (2006): International convergence of capital measurement and capital standards. A revised framework. Basel: Bank for International Settlements.

BCBS (2011): Basel III. A global regulatory framework for more resilient banks and banking systems. December 2010 (rev. June 2011). Basel.

Bishop, C. M. (2006): Pattern recognition and machine learning. New York NY: Springer (Information science and statistics).

Blum, J. M. (2008): Why 'Basel II' may need a leverage ratio restriction. In *Journal of Banking & Finance* 32 (8), pp. 1699–1707.

Breiman, L. (1996): Bagging Predictors. In *Machine Learning* 24 (2), pp. 123–140.

Breiman, L. (2001): Random forests. In *Machine Learning* 45 (1), pp. 5–32.

Breiman, L.; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984): CART: Classification and Regression Trees. Belmont, CA: Wadsworth.

Brier, G. W. (1950): Verification of forecasts expressed in terms of probability. In *Monthly Weather Review* 78 (1), pp. 1–3.

Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; Kegelmeyer, W. P. (2002): SMOTE. Synthetic Minority Over-sampling Technique. In *Journal Of Artificial Intelligence Research* (16), pp. 321–357.

Chen, W.; Ma, C.; Ma, L. (2009): Mining the customer credit using hybrid support vector machine technique. In *Expert Systems with Applications* 36 (4), pp. 7611–7616.

Conway, D.; White, J. M. (2012): Machine learning for hackers. Sebastopol, CA: O'Reilly.

Crook, J. N.; Edelman, D. B.; Thomas, L. C. (2007): Recent developments in consumer credit risk assessment. In *European Journal of Operational Research* 183 (3), pp. 1447–1465.

D' Angelo, P. (2016): The Method of Random Forest for Credit Scoring. Master Thesis. Technische Universität Kaiserslautern, Kaiserslautern. Fachbereich Mathematik.

Deutsche Bundesbank (2018): Lending to domestic employees and other individuals / Long-term / All categories of banks. BBK01.PQ3002. Available online at https://www.bundesbank.de/Navigation/EN/Statistics/, checked on 1/8/2018.

Fawcett, T. (2003): ROC Graphs. Notes and Practical Considerations for Data Mining Researchers. In *Machine Learning* 21 (1), pp. 1–38.

Freund, Y.; Schapire, R. E. (1996): Experiments with a new boosting algorithm. In *ICML* (96), pp. 148–156.

Geman, S.; Bienenstock, E.; Doursat, R. (1992): Neural Networks and the Bias/Variance Dilemma. In *Neural Computation* 4 (1), pp. 1–58.

Goonatilake, S.; Treleaven, P. (1995): Intelligent Systems for Finance and Business. New York: John Wiley & Sons, Inc.

Han, H.; Wang, W.-Y.; Mao, B.-W. (2005): Borderline-SMOTE. A New Over-Sampling Method in Imbalanced Data Sets Learning. In : International Conference on Intelligent Computing. Berlin, Heidelberg: Springer, pp. 878–887.

Han, J.; Kamber, M.; Pei, J. (2012): Data mining. Concepts and techniques. 3. ed. Amsterdam: Elsevier/Morgan Kaufmann.

Hand, D. J.; Henley, W. E. (1997): Statistical Classification Methods in Consumer Credit Scoring. A Review. In *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 160 (3), pp. 523–541.

Harris, T. (2015): Credit scoring using the clustered support vector machine. In *Expert Systems with Applications* 42 (2), pp. 741–750.

Hastie, T.; Tibshirani, R.; Friedman, J. (2009): The Elements of Statistical Learning. Data Mining, Inference, and Prediction. 3. ed. New York, NY: Springer (Springer Series in Statistics).

He, H.; Bai, Y.; Garcia, E. A.; Li, S. (Eds.) (2008): ADASYN. Adaptive synthetic sampling approach for imbalanced learning. IEEE World Congress on Computational Intelligence. IEEE International Joint Conference on Neural Networks.

He, H.; Garcia, E. A. (2009): Learning from Imbalanced Data. In *IEEE Transactions on Knowledge and Data Engineering* 21 (9), pp. 1263–1284.

Horn, D. M. (2016): Credit Scoring Using Genetic Programming. Master Thesis. NOVA IMS, Lisbon. Instituto Superior de Estatística e Gestão de Informação.

Hung, C.; Chen, J.-H. (2009): A selective ensemble based on expected probabilities for bankruptcy prediction. In *Expert Systems with Applications* 36 (3), pp. 5297–5303.

Kearns, M. (1988): Thoughts on Hypothesis Boosting. Machine Learning class project.

Khandani, A. E.; Kim, A. J.; Lo, A. W. (2010): Consumer Credit Risk Models Via Machine-Learning Algorithms. In *Journal of Banking & Finance* 34 (11), pp. 2767–2787.

Laurikkala, J. (2001): Improving Identification of Difficult Small Classes by Balancing Class Distribution. In S. Quaglini, P. Barahona, S. Andreassen (Eds.): Artificial Intelligence in Medicine. Berlin, Heidelberg: Springer, pp. 63–66.

Lessmann, S.; Baesens, B.; Seow, H.-V.; Thomas, L. C. (2015): Benchmarking state-of-the-art classification algorithms for credit scoring. An update of research. In *European Journal of Operational Research* 247 (1), pp. 124–136.

Longstaff, F. A. (2010): The subprime credit crisis and contagion in financial markets. In *Journal of Financial Economics* 97 (3), pp. 436–450.

Louppe, G. (2014): Understanding Random Forests. From Theory to Practice. PhD dissertation. University of Liège, Liège. Department of Electrical Engineering & Computer Science.

Louzada, F.; Ara, A.; Fernandes, G. B. (2016): Classification methods applied to credit scoring. A systematic review and overall comparison. In *Operations Research and Management Science* 21 (2), pp. 117–134.

Luo, C.; Wu, D. (2017): A deep learning approach for credit scoring using credit default swaps. In *Engineering Applications of Artificial Intelligence* 65, pp. 465–470.

Marqués, A. I.; García, V.; Sánchez, J. S. (2012): Exploring the behaviour of base classifiers in credit scoring ensembles. In *Expert Systems with Applications* 39 (11), pp. 10244–10250.

Nanni, L.; Lumini, A. (2009): An experimental comparison of ensemble of classifiers for bankruptcy prediction and credit scoring. In *Expert Systems with Applications* 36 (2), pp. 3028–3033.

Penikas, H. (2015): History of banking regulation as developed by the Basel Committee on Banking Supervision in 1974-2014 (Brief overview).

Quinlan, J. R. (1986): Induction of Decision Trees. In *Machine Learning* 1 (1), pp. 81–106.

Quinlan, R. (1996): Bagging, Boosting, and C4.5. In *Proceedings of the National Conference on Artificial Intelligence* Vol 1., pp. 725–730.

Raschka, S.; Olson, R. S. (2015): Python machine learning. Unlock deeper insights into machine learning with this vital guide to cutting-edge predicitve analytics. Birmingham, Mumbai: Packt Publishing.

Reh, A. (2017): Optimierung von Retouren eines Onlinehändlers mithilfe von Predictive Analytics. Master Thesis. Hochschule Reutlingen, Reutlingen. Fakultät Informatik.

Richert, W.; Coelho, L. P. (2013): Building machine learning systems with Python. 1. ed. Birmingham, Mumbai: Packt Publishing.

Saito, T.; Rehmsmeier, M. (2015): The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. In *PloS one* 10 (3).

Schafer, J. L.; Graham, J. W. (2002): Missing data. Our view of the state of the art. In *Psychological Methods* 7 (2), pp. 147–177.

Schebesch, K. B.; Stecking, R. (2005): Support vector machines for classifying and describing credit applicants. Detecting typical and critical regions. In *Journal of the Operational Research Society* 56 (9), pp. 1082–1088.

scikit-learn (2017a): Ensemble Methods. Available online at http://scikit-learn.org/stable/modules/ensemble.html#parameters, checked on 12/7/2017.

scikit-learn (2017b): RandomForestClassifier (v0.19.1). Available online at http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier, checked on 12/11/2017.

Siddiqi, N. (2012): Credit risk scorecards. Developing and implementing intelligent credit scoring: John Wiley & Sons.

Sparkasse KoelnBonn (2016): Jahresabschluss 2016. Köln. Available online at https://www.sparkasse-koelnbonn.de/content/dam/myif/sk-koelnbonn/work/dokumente/pdf/unternehmen/zahlen-und-fakten/jahresabschluesse/JA2016_final.pdf?n=true, checked on 3/2/2018.

Tian-Shyug L.; I-Fei C. (2005): A two-stage hybrid credit scoring model using artificial neural networks and multivariate adaptive regression splines. In *Expert Systems with Applications* 28 (4), pp. 743–752.

Ting, K. M.; Witten, I. H. (1999): Issues in Stacked Generalization. In *Journal Of Artificial Intelligence Research* (10), pp. 271–289.

Tsymbal, A.; Pechenizkiy, M.; Cunningham, P. (2005): Diversity in search strategies for ensemble feature selection. In *Information Fusion* 6 (1), pp. 83–98.

Weiss, G. M.; Provost, F. (2001): The Effect if Class Distribution on Classifier Learning. An Empirical Study: Technical Report ML-TRr-44, Department of Computer Science, Rutgers University.

West, D. (2000): Neural network credit scoring models. In *Computers & Operations Research* 27 (11-12), pp. 1131–1152.

Wolpert, D. H. (1992): Stacked generalization. In *Neural networks 5* (2), pp. 241–259.

Xia, Y.; Liu, C.; Li, Y.; Liu, N. (2017): A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring. In *Expert Systems with Applications* 78, pp. 225–241.

Xiao, H.; Xiao, Z.; Wang, Y. (2016): Ensemble classification based on supervised clustering for credit scoring. In *Applied Soft Computing* 43, pp. 73–86.

Zadrozny, B.; Elkan, C. (2001): Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *ICML* (1), pp. 609–616.

Zadrozny, B.; Elkan, C. (2002): Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining.*, pp. 694–699.

Zhang, Jianping; Mani, Inderjeet (Eds.) (2003): kNN approach to unbalanced data distributions. A case study involving information extraction. Proceedings of workshop on learning from imbalanced datasets.

Zhao, Z.; Xu, S.; Kang, B. H.; Kabir, M. M. J.; Liu, Y.; Wasinger, R. (2015): Investigation and improvement of multi-layer perceptron neural networks for credit scoring. In *Expert Systems with Applications* 42 (7), pp. 3508–3516.

Zhou, Z.-H. (2012): Ensemble methods. Foundations and algorithms: CRC press.

7. Appendix

	RF		Ada		LR	
	0	1	0	1	0	1
Actual 0	193982	54	194027	9	194000	36
Actual 1	131	64	158	37	155	40

	Calibration		Oversampling		Stacking	
	0	1	0	1	0	1
Actual 0	194011	25	193964	72	193914	122
Actual 1	147	48	136	59	108	87

Table 7.1: Confusion matrix of the model comparison on the train set. For a more compact visualization we plot the confusion matrixes next to each other.

	RF		Ada		LR	
	0	1	0	1	0	1
Actual 0	48589	10	48597	2	48593	6
Actual 1	31	22	40	13	44	9

	Calibration		Oversampling		Stacking	
	0	1	0	1	0	1
Actual 0	48597	2	48588	11	48590	9
Actual 1	42	11	35	18	30	23

Table 7.2: Confusion matrix of the model comparison on the train set. For a more compact visualization we plot the confusion matrixes next to each other.

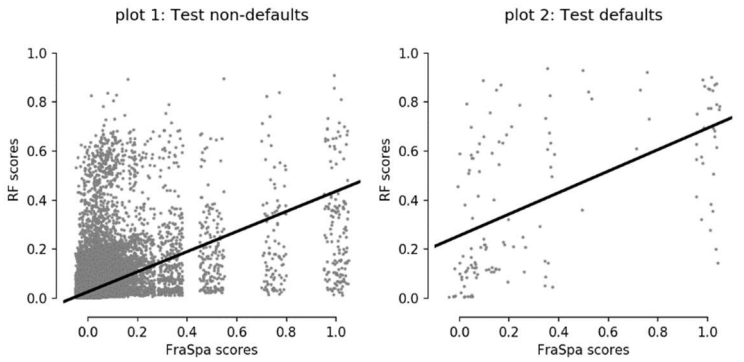

Figure 7.1: Scatter comparison RF against FraSpa scores on the 12-month test set.