University of Twente

Financial Engineering and Management
Master Thesis

Improving a Risk-Based Revision Model for SME Loans

Author:
T. Otten

Internal Supervisors:
W.J.A. van Heeswijk
B. Roorda

Publication date:
July 18, 2022

External Supervisor:
E. van Wijhe

Abstract

In this research, we study how we can improve the risk-based revision model of a Dutch bank that signals the need for measures that should prevent clients from going into default based on the current financial situation of clients. We test various machine learning models, namely logistic regression, decision trees and random forests, to obtain a new improved classification model. Most of the clients of the bank are in a healthy financial situation and do not need measures to prevent a default. Therefore, we implement imbalanced dataset techniques such as alternative cut-off strategies and the synthetic minority oversampling technique (SMOTE) to boost the performance of the models. Also, we apply hyperparameter tuning by implementing a grid search. Higher precision is preferred over lower recall by management in this study. Therefore, we measure model performance with the $F_{0.33}$ score. We found that by using a random forest model and applying an alternative cut-off point, we could improve the $F_{0.33}$ score from 7 percent to 71 percent, indicating that the new model can be seen as an improvement over the old model. Underlying here is that precision increased from 6 percent to 84 percent but the recall performance decreased from 80 percent to 30 percent. Besides suggesting a configuration for the new model, we show what configurations are possible in terms of different precision versus recall trade-offs to provide insight to management on what performance levels can be achieved.

Keywords: *Forbearance measures, risk-based revisions, logistic regression, decision trees, random forests, imbalanced dataset techniques, alternative cut-off, SMOTE, grid search*

Executive Summary

The main goal of this study was to improve the risk-based revision (RBR) model used at the business department of the Volksbank. The RBR model is used to signal clients on a monthly basis that appear to be going into default and should get a revision in which they could get forbearance measures that should prevent a default. The problem with the current RBR model that uses trigger conditions to signal clients is that it is not precise enough according to employees directly involved in the RBR process. The model signals a lot of clients that do not need a revision. Besides that we create a new improved model, this study provides insight into the trade-off that exists for the configuration of a new model with respect to false positive and false negative predictions from a new model.

We analyzed the current RBR process and evaluated the performance of the current model. Relevant performance metrics to measure the performance of the current model are precision, recall and the F score. Precision reflects the ratio of correct signals versus incorrect signals and recall reflects the proportion of actual positives that was identified correctly. Generally, these two metrics are competing with each other, the higher the precision, the lower the recall and vice versa. The F1 score combines the performance in terms of precision and recall into one metric with equal weights. After consulting the management directly supervising this study, we established that for the new model, the improvement in terms of precision is more important than the recall because other processes in the bank such as arrears management and Unlikely to Pay triggers could also signal clients with financial difficulties. Therefore, we chose to use the $F_{0.33}$ score variant of the F1 score in this study to decide on which new model performs best because this variant gives a larger weight to the precision performance than the recall performance. We found that the current model achieves a precision of only 6% with a recall of 80% and a $F_{0.33}$ score of 7%, thereby confirming that the current model is not precise.

By testing various machine learning (ML) models and techniques, we created a new RBR model. We found that of the type of ML models we tested, we could achieve the best performance with a random forest model. We were restricted in the type of ML models we could use as regulatory requirements do not allow for black box type of models to be used. The challenge in this study is that we had to work with a limited-sized dataset which was also very imbalanced, two factors that make it more difficult to create a well-performing model.

We configured the new RBR model such that it has high precision, which is also reflected in the performance of the new model. The new model achieves a $F_{0.33}$ score of 71% versus 7% for the old model. Precision increased to 84% versus the 6% of the old model. The recall decreased as a result of the configuration of the model we chose. The recall is 30% for the new model versus 80% for the old model. An interesting insight we found is that recall increases to 53% if the signal from the RBR model can also come 1 month later from the next monthly run of the model. Important predictors we decided to use in the new RBR model are the probability of default calculated by another risk model within the bank and whether there is payment traffic going in and out of the current account of the client.

We acknowledge that although the new model is an improvement over the old model, it is still not extremely good. We say that this would be when both precision and recall are larger than 90%. We identified four potential reasons why performance is not perfect with the first being inconsistency in the evaluation of the shortlist of the current model that we use to train the new model. The current model produces a shortlist of clients that it signals, this list is manually evaluated by an employee. Manual evaluation leaves room for arbitrariness and so inconsistencies with which a ML model has troubles. The second reason could be the small size of the dataset available to us. As the Volksbank has currently a relatively small number of business clients, there are very few examples of risk-based revisions (only 97). It could be that this number of examples is too limited to be able to achieve better performance. Subsequently, it could be that we did not define the appropriate

features that could capture the patterns in the data that is available to us. Potentially, features could be constructed that could grasp the patterns in the available data better. The last reason could be the lack of patterns in the data. It is possible that with the current set of predictors, the dataset does not contain sufficiently strong patterns. Other factors currently not in our dataset could also influence whether a revision is needed or not. It is also the question of whether there exist patterns that could be captured by any set of predictors or whether the outcome is intrinsically noisy.

This study showed that the current RBR model can be improved. We recommend implementing the new model as it is clear that it is an improvement over the old model. We would recommend updating the model after some years when more examples of risk-based revisions are available. Also, we suggest performing follow-up research on in which type of situation clients should get a revision and which not, currently there is no defined policy on this within the bank. Next, we would recommend quantifying the costs of false positive predictions and false negative predictions of the RBR model. This can justify the configuration of the RBR model in terms of precision versus recall. Lastly, we suggest experimenting with more (advanced) predictors than used in the dataset used by us in this study. It could be that other predictors better capture the patterns in predicting risk-based revisions.

Table of Contents

Glossary

Decision tree	= A machine learning model that makes predictions based on how a set of questions is answered. This set of questions is reflected in a Christmas tree shape.
Forbearance measures	= These are measures in which a bank helps a client by changing some of the conditions on the loan it provided to prevent a client from going into default. Examples of these are postponing repayments or changing the duration of the loan.
Logistic regression	= A classification machine learning model that is used to estimate the probability of an observation belonging to a class.
LTV ratio	= The Loan-To-Value ratio is the total loan amount divided by the value of the underlying collateral of the loan.
Majority class	= In a dataset with two labels, the class (group of observations from the same category) that contains the majority of the observations is called the majority class.
Minority class	= In a dataset with two labels, the under-represented class (group of observations from the same category) of the observations is called the minority class.
Non-performing	= A loan becomes non-performing when there are indications that the borrower is very unlikely to repay the loan, or if more than 90 days have passed without the borrower paying the agreed installments.
Overdraft	= An allowed negative balance in the account. Overdraw.
Performing	= A loan is performing when the obligor met his payment obligations on time. But also loans for which payment arrears are fewer than 90 days late are considered performing.
Random forest	= A machine learning model that combines multiple decision trees to make predictions.
Revision	= A revision is a reassessment of the credit risk and the associated customer strategy of/for a client.
SMOTE	= Synthetic Minority Oversampling TEchnique, a technique that oversamples the minority class of a dataset by creating synthetic but very similar new observations of the minority class instead of oversampling with replacement.
Special Asset Management	= Department at a financial institution that takes care of financially distressed clients (Dutch = Bijzonder Beheer).
UtP Trigger	= Triggers that cause a client to be declared in default. Some UtP triggers are mandatory, as they are defined by regulators. Non-mandatory UtP triggers are more subjective/refutable signals that a client will not repay a loan.

Abbreviations

ACC	= Accuracy
AIC	= Akaike Information Criterion
BIC	= Bayesian Information Criterion
DT	= Decision Tree
EAD	= Expose at Default
EBA	= European Banking Authority
EL	= Expected Loss
FN	= False Negative
FP	= False Positive
LOCF	= Last Observation Carried Forward
LR	= Logistic Regression
LTV	= Loan-To-Value
MAR	= Missing At Random
MCAR	= Missing Completely At Random
ML	= Machine Learning
NMAR	= Not Missing At Random
NOCB	= Next Observation Carried Backward
NPV	= Negative Predictive Value
PD	= Probability of Default
RBR	= Risk-Based Revision
RF	= Random Forest
RSS	= Residual Sum of Squares
SMOTE	= Synthetic Minority Oversampling TEchnique
SVMs	= Support Vector Machines
TN	= True Negative
TP	= True Positive
UtP	= Unlikely to Pay

List of Figures

List of Tables

1. Introduction

This chapter introduces the research and the research context. In Section 1.1, we introduce the Volksbank, the company at which this study is performed. Also, we briefly introduce the business process of a risk-based revision. Section 1.2 presents the problem statement and research goal of this study. In Section 1.3, we provide the research questions and sub-questions of this study. Finally, Section 1.4 presents the research approach of our study.

1.1. Background

Nowadays, the banking industry is complex. Banks have to comply with a large number of regulations and in a bank's day-to-day operations mathematical models are used. This also holds for the Volksbank, the bank at which this study is performed. The Volksbank, from this point also referred to as the bank, is the bank behind brands like SNS, ASN Bank, BLG Wonen and RegioBank. One of the processes at the Volksbank is the monitoring of business loans. The bank takes precautionary action when clients appear to become in default in the near future due to temporary financial stress. This monitoring is even compulsory under legislation coming from the European Banking Authority (EBA). For the specific regulation, we refer to European Banking Authority (2020, pp. 60-67)

Measures banks can take in these situations are called forbearance measures. These are measures in which a bank is lenient to a client. The bank supports a client by changing some of the conditions on the loan it provided. Examples of changes to conditions on a loan are allowing clients to postpone repayments or extending the duration of the loan. Also, the client can be transferred to the Special Asset Management department, which takes care of clients with very high-risk profiles or clients that are already in default.

Reevaluating a client is called a revision. A revision is a reassessment of the credit risk and the associated customer strategy of/for a client. A revision is carried out on the basis of the current (financial) information of a client. The Volksbank has a model in place that pre-screens its portfolio of small and medium-sized enterprise clients each month. Based on the risk the model determines for a client, it decides whether an employee should evaluate that client to take measures if needed. Therefore, this entire process is referred to as the risk-based revision (RBR) process. Within the business department of the Volksbank, there is the wish to improve the current RBR model.

1.2. Problem statement

The current RBR model that determines which clients are eligible for further inspection is not working satisfactorily. The current model is based on triggering conditions (if-then statements on various financial ratios of a client). If one of the conditions set is triggered, the model marks a client as up for revision. However, the performance of this model is insufficient according to management. Estimates from practice are that +- 90% of clients marked by the model as up for revision turn out to be false positives. This means that after a risk manager evaluated the client, 90% of the time the decision was made that a forbearance measure was not needed. Also, the number of false negatives resulting from the current model is unknown. That is, the number of clients that need a forbearance measure but are not marked by the model is unknown.

Because of the perceived poor performance of the current RBR model, management requests to have the model improved, possibly by advanced data science techniques. The improvement should focus on significantly reducing the number of false positive signals from the model. Besides, management wants to gain more insight into what are the important characteristics when predicting the need for a forbearance measure as this is currently not clear.

The RBR process can be described as a process that provides early warning messages when problems are occurring around loans. Based on these warning messages, adequate actions can be taken to prevent a loan from going into default. In this study, the effectiveness of these measures is out of scope. We assume that these actions, the forbearance measures work and therefore we will not cover the effect of such measures on the Probability of Default (PD). The RBR model only predicts whether a forbearance measure is needed, not which specific measure is needed. The potential for value creation of the process is plentiful. Firstly, if fewer clients default, more principals are paid back to the bank. Secondly, the process helps to retain business and potentially creates new business. If clients improve their financial situation because of applied forbearance measures, the bank keeps the earnings from the business they bring in. Furthermore, if clients fully recover and after some time even decide to expand their business, they are likely to come back to the bank that was lenient to them initially, creating even more business for the bank and in the end growing the assets under management of the bank. Lastly, fewer clients in default will result in the bank needing to make fewer capital provisions. A prerequisite for overall value creation of the RBR process is that the benefits outweigh the costs generated by the process. Currently, it is questionable whether this is the case as a lot of costs are generated in the process because of the evaluation of all the false positive signals. Therefore, having a well-functioning RBR model is essential.

There is a feeling in the department that the current model is not working well enough because it is very hard to predict whether measures should be taken. The outcome of whether a loan to a client becomes non-performing is stochastic. A certain financial situation will end up in a default the one time and a very comparable situation will not the other time. The presumption is that future macroeconomic factors/fluctuations that are unpredictable have a large effect on the outcome of a certain situation. This study will have to show whether this suspicion could be true.

Considering all requests of the management, the main goal of this study is to create a new RBR model with better performance than the current model. Creating an improved model will have the direct effect that the labor effectiveness in the RBR process is increased. Employees will lose less time inspecting false positive output from the model and can work on more cases that can create value for the bank. In the long term, an improved model has the potential to recover more principals, retain more business and even generate new business in the future as employees will waste less time inspecting false positive signals of the model. Besides delivering a new RBR model, we will give insight in this study on how the new model can be tweaked after we complete this study. The reason for this is that an implementation procedure of a new model at a bank, which is out of the scope of this study, is lengthy and in which often new requirements or issues arise. By delivering insight, we preempt changing requirements and performance preferences.

Although there is a conjecture that future macroeconomic factors have a significant impact on the RBR process, we decided not to elaborately explore these factors in this study because of time reasons. We expect that even without implementing future macroeconomic factors, the time needed to create a new RBR model will be significant. When we are not able to create a considerably better performing RBR, we can conclude that the suspicion about macroeconomic factors within the department could be correct and this could then be a topic for further research.

Our first conjecture is that applying machine learning techniques using internal client data of the bank is an effective strategy to achieve our goal of a new and better performing RBR model as machine learning (ML) is a great tool to make predictive models. To be able to improve the model, insight into the RBR process is needed. Besides, important characteristics of clients can be derived from a new model (depending on the type of ML algorithm used). This matches the request of the management with the research goal.

1.3. Research setup

As the goal of this study is to improve the performance of the risk-based revision model, our main research question is: “*How can the risk-based revision model be improved using machine learning?”*

To answer the main research question, we formulate 5 sub-research questions. Each of these corresponds to one chapter of this report. If needed, we broke down a question into more sub-questions to go into more detail about what we will research in that phase of our study. Answering all these questions will result in answering the main research question. We identified that we first need to study the current situation. This is reflected in Question 1. Next, we need to find out what ML models and techniques we can apply which is reflected in Question 2. Thereafter in Question 3, we need to establish what design decisions we make for our new model(s) and how we bring the literature into practice, for example, which parameters we set for the ML model(s) we implement. Then we will compare our new model(s) with the old model (Question 4). Finally, we have to determine which conclusion we can make about our work and identify what topics for further research could be which is reflected in Question 5.

1. What are the current risk-based revision process and model?
    a. What is the current working procedure using the output of the risk-based revision model?
    b. How does the current risk-based revision model work?
    c. What are relevant performance metrics to evaluate the performance of the risk-based revision model?
    d. What is the performance of the current risk-based revision model?
    e. How does the available data on the risk-based revision process look like?
2. What machine learning models and techniques are most suitable to use to create a new risk-based revision model?
    a. Why use machine learning techniques to create a new risk-based revision model?
    b. What machine learning techniques can we use to validate our model?
    c. What machine learning models are relevant for creating a new risk-based revision model?
    d. What machine learning techniques are useful to improve the performance of the selected machine learning model(s)?
3. How do we implement relevant machine learning theory into a new risk-based revision model?
4. How does the new model compare with the initial risk-based revision model?
    a. What are the scores of the new risk-based revision model(s) on the earlier identified performance metrics?
    b. What are the important features for the prediction of clients needing a forbearance measure?
5. What are the conclusions and points for further research of our study?

1.4. Research approach

In this section, we will elaborate on the steps we take in this study. Firstly, we will familiarize ourselves more with the problem using a systematic approach. This involves contacting stakeholders of the RBR process and getting to know the procedures in place at the Volksbank by interviewing employees. We will thoroughly analyze the current model, understand how it works and measure its performance. For performance measurement, data on past outcomes of the revision process is needed and the performance metric(s) to be used will have to be identified. We will have to gather, analyze and clean this data to be able to perform performance measurements and to create our new model at a later stage. The results of these steps we cover in Chapter 2.

Secondly, in Chapter 3, we will perform the theoretical part of this study. We will have to establish the theoretical framework of this research. Starting with identifying whether applying ML is the way to go in this study. This includes whether we can improve the model in simpler ways than applying ML models. We will identify relevant model validation methods, ML models and ML techniques that can be applied to our problem context.

Subsequently, in Chapter 4, we will create new models based on our findings from the previous step. We will establish and explain the design decisions of the models we will create. An important step in this phase is the verification of our models.

Next, we will evaluate the results that our models produce and determine which type of ML model has the best performance in Chapter 5. A comparison between the new and the old models we will make. Also, we will identify what are the important variables in the new RBR model.

Finally, in Chapter 6, we will draw our conclusion about the study. In this last step, we will present our findings to the management and state which ML model we would recommend implementing. Also, we will present topics of future research related to our study.

2. Context Analysis

In this chapter, we will describe the context of this study in more depth. In Section 2.1, we will cover the current work procedure of the RBR process. In Section 2.2, we will describe how the current RBR model works. Following that, we will introduce the performance metrics we will use to measure the performance of the RBR model in Section 2.3. From that, in Section 2.4, we introduce some performance measurement metrics and calculate the performance of the current model. Subsequently, in Section 2.5, we will present the characteristics of the dataset that is available to us. Lastly, in Section 2.6, we summarize this chapter.

2.1. Work procedure of the risk-based revision process

In this section, we will describe the current work procedure of the RBR process. We will cover the context of the RBR process, for which type of clients the RBR process is in place, the chronological steps that are taken in the process and describe decision rules we identified in the process.

2.1.1. Context

Clients that have a loan from a bank can be classified into two states. A client can be performing or non-performing. Performing clients are clients that met their payment obligations on time. But also clients for which payment arrears are fewer than or equal to 90 days late are considered performing. Non-performing clients are borrowers for which there are indications that the borrower is very unlikely to repay the loan, these indications are called Unlikely to Pay triggers. Also, clients are non-performing when more than 90 days have passed without the borrower paying the agreed installments. If one of these conditions is met, a client is considered as in default.

The Volksbank wants to prevent that clients go into default. To prevent a default, the bank can give forbearance measures to clients. These are measures in which a bank helps a client by changing some of the conditions on the loan it provided if the client is in temporary financial stress. Examples of these are postponing repayments or changing the duration of the loan. Clients will only receive these measures when it is clear they have a viable business in the long term and have a good financial relationship with the bank. In our study, we assume that forbearance measures have a significant positive effect on the probability of default. To decide whether a forbearance measure is needed, a revision is needed. A revision is a process in which an employee reevaluates a client. Labor capacity does not allow to revise every client periodically. Therefore, the RBR is used. The RBR model determines which clients likely need a forbearance measure, producing a dropout list of clients that should be manageable with the available labor capacity. The current RBR model flags approximately 1 percent of the clients. The goal of the RBR process is to identify clients that are not yet in the picture but do need forbearance measures to prevent default. Of course, some clients also contact the bank themselves when they run into financial difficulties. When this occurs, an employee makes a quick scan of the situation and decides which steps to take next. This is also a very interesting process within the bank but we will not cover it in the scope of this research. We focus on the RBR process and the clients that are flagged by the RBR model.

2.1.2. Type of clients

The RBR process is not applied to every type of client. A distinguishment is made based on the exposure of a client and the Loan-To-Value (LTV) ratio of a client. The LTV is the total loan amount divided by the value of the underlying collateral of the loan. Clients with performing loans with an exposure larger than **€E_1** always get a periodical individual revision (figures in this section are anonymized because of confidentiality). Clients with an exposure between **€E_1** and **€E_2** that also have a LTV ratio larger than **X**% also get a periodical individual revision (**€E_1** > **€E_2**). The high outstanding amounts of these types of clients make the costs a periodical individual revision justified. Clients with performing loans with an exposure of less than **€E$_3$** are revision-free (**€E$_1$** > **€E$_2$** > **€E$_3$**). For these types of clients, a revision process is not rewarding from a business perspective because of the small exposure involved with the loan. On all remaining clients with performing loans, the RBR process is applied. Finally, non-performing clients will always get an individual revision. An overview of the applied methodology per client type can be found in Table 1. This study only focuses on the group of clients that fall into the risk-based revision methodology.

Table 1, Overview of Status, Exposure and Applied Methodology.

Status	Exposure	Applied Methodology
Performing	Exposure > €E$_1$	Individual revision
	€E$_1$ >Exposure > €E$_2$ & LTV ratio > X%	Individual revision
	Remaining exposures > €E$_3$	Risk-Based revision
	Remaining exposures < €E$_3$	Revision free
Non-performing	All exposures	Individual revision

2.1.3. Steps in the process

The RBR process is a recurring process. Every month, the RBR model creates a shortlist of clients that appear to be going into default. Evaluation of the shortlist by an employee can have three outcomes. The first outcome is that there is not enough financial stress that a revision is needed in the opinion of the employee, no action is necessary. The second outcome is that there is financial stress but with forbearance measures, a client can be helped to continue to meet his obligations in the opinion of the employee. Therefore, a revision is needed. The last outcome is that there is financial stress but a client will even with forbearance measures not be able to meet his obligations, resulting in the client being declared in default and/or will be transferred to the Special Asset Department. Such an outcome can be the result of the opinion of the employee on the situation of a client, a client not wanting to meet his obligations, or (mandatory) unlikely to pay (UtP) triggers. For more information on (mandatory) UTP triggers, see ECB (2017).

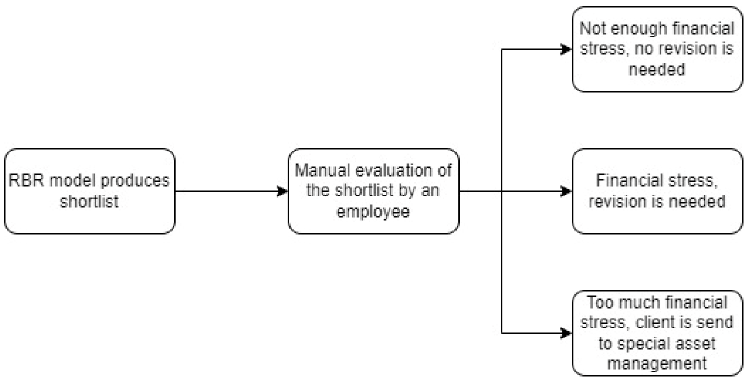
RBR model produces shortlist
Manual evaluation of the shortlist by an employee
Not enough financial stress, no revision is needed
Financial stress, revision is needed
Too much financial stress, client is send to special asset management

Figure 1: Schematic overview of the steps in the process.

The manual part of the RBR process starts with an employee evaluating the shortlist produced by the RBR model. The employee verifies whether a revision of each client is indeed needed. The employee does this by checking factors such as overdraft of a client, use of the limit on the current account, the trend in the use of the current account, payment arrears and whether payment traffic is still going in and out on the bank account of the client in the various systems the bank has to monitor its clients. For unclear reasons we could not identify why these factors, except for the payment traffic, are not yet in the current model, making them appealing to put into our new model as they would be indicators if action is needed or not. The reason for not implementing payment traffic was a practical issue. When a client has a loan at the bank but the account with payment traffic at another bank this data is not available. The next step in the process is that the employee presents his findings to another employee from the risk management department. This other employee must give his approval for the new list of clients that need a revision. When approval is given, the first employee will send revision tasks to the back office of the bank. At the back office, the revision tasks will be distributed over the pool of account managers. An individual account manager is responsible for the execution of a revision task assigned to him. On sending a revision task to the back office, a revision date is set. This is the date before which the revision should be performed.

2.1.4. Decision rules

The critical step in the RBR process is the employee verifying whether a revision is needed for the clients on the shortlist. After interviewing the employees directly involved in the RBR process, we identified two reasons when an employee determines a revision is needed:

1. A negative trend in the current account in combination with a decreasing turnover.
2. No payment traffic over the current account.

The first reason is when there is an obvious negative trend in the use of the current account in combination with a decreasing turnover in the account. A negative trend is when the loan amount outstanding steadily increases, this might indicate a problem. A threshold for the increase of the outstanding amount is not decided on in a policy of the bank but the trend should at least be obvious. Also, solely a negative trend does not have to indicate financial difficulties. This could also indicate that the client is expanding his business. Therefore, it is a requirement that there should be a negative trend in combination with decreasing turnover.

The second reason for revision is when there is no longer payment traffic going in and out over the current account. This indicates that the current account is used as an additional loan by the client. A current account should not be used as a permanent loan, this is not allowed by the bank. We identify that these decision rules are susceptible to randomness. No hard descriptions of these decision rules exist or are documented. Also, as this part of the process is executed by people, it is clear that the current process leaves room for inconsistent decision-making.

2.2. The current risk-based revision model

In this section, we will describe the current RBR model. We discuss the methodology of the model and explain the financial ratios that are used in the model. Some of the ratios mentioned are anonymized because of confidentiality.

The current model uses triggering conditions on the financial ratios of a client to determine which clients need a revision and puts these clients on a shortlist. If one of the conditions set is triggered, the model marks a client as up for revision. But as described in Section 2.1.3, this shortlist is checked on its correctness by an employee in the current way of working. The first ratio that is used for triggering conditions is the Probability of Default (PD) score. The bank uses a credit risk model to determine a new PD score class each month. The higher the PD score, the larger the PD. It can be seen as a black box score as it is not clear what the exact difference is between scores to the employees directly involved in the RBR process. If the PD score of a client is above a certain threshold, the client will be marked as up for revision. Also, if a PD score of a client worsened more than X classes compared to last month(s) and the PD scores of these months are above a certain threshold, the client will be marked as up for revision.

The next ratio that is used is Expected Loss (EL) in combination with the Exposure at Default (EAD). If a client's EL is larger than $\mathbf{X_1}$% of the EAD, the client will be marked as up for revision. The last ratio that is used in the current model is the LTV ratio. If the LTV ratio of a client is lower than $\mathbf{X_2}$% percent, a client will not be marked as up for revision. In these cases, no losses for the bank will occur at default because the underlying collateral has enough value.

2.3. Performance metrics

In this section, we define what performance metrics we will use to measure the performance of the RBR model. We will introduce the concepts of True Positive, False Positive, False Negative and True Negative. But we will also cover the performance metrics of accuracy, precision, recall, specificity, negative predictive value, F1 score and F_β score.

The RBR model can be referred to as a classification model. The model predicts the qualitative outcomes '*revision needed*' or '*no revision needed*'. We define '*revision needed*' as a positive classification and '*no revision needed*' as a negative classification. Positives or positive observations are observations of the label '*revision needed*' and negatives or negative observations are observations of the label '*no revision needed*'.

When classifying, we have the actual class and the predicted class. The actual class is the real group an observation belongs to. In our case, this is whether a client actually needed a revision or not. The predicted class is the forecasted group an observation belongs to. In our case, this is the output of the RBR model that labels clients on whether a revision is needed or not. This results in four combinations of outcomes being possible.

The first combination is actual class positive with predicted class positive. This combination is known as a True Positive (TP). Secondly, we have the combination of actual class negative and predicted class positive. This combination is known as a False Positive (FP). Next, we have the combination of actual class positive and predicted class negative. This combination is known as a False Negative (FN). Lastly, we have the combination of actual class negative and predicted class negative. This combination is known as a True Negative (TN). These combinations are illustrated in

Figure 2.

Also illustrated in

Figure 2, are performance metrics we define that can be derived from the confusion matrix. The most intuitive metric is accuracy. We define accuracy as the percentage of observations that are classified correctly. With balanced positive and negative observations, accuracy is quite a good performance metric. However, in the RBR process, we deal with imbalanced classes. Most of the clients do not need a revision. When we would solely use accuracy, the performance would be pretty high because most TNs will likely indeed be predicted as negative cases. So only using the performance metric accuracy would give an incorrect image of the actual performance. Accuracy is not a good metric for reflecting the performance of predicting a minority class.

		Predicted Class		
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN)	**Recall** $\frac{TP}{TP + FN}$
	Negative	False Positive (FP)	True Negative (TN)	**Specificity** $\frac{TN}{TN + FP}$
		Precision $\frac{TP}{TP + FP}$	**Negative Predictive Value** $\frac{TN}{TN + FN}$	**Accuracy** $\frac{TP + TN}{TP + TN + FP + FN}$

Figure 2: After An (2020), Illustration of a confusion matrix with performance metrics.

Alternatives to using accuracy as a performance metric are metrics based on subsets of the TP, FP, FN and TN classes. We will discuss these in this paragraph. The first metric we will cover is precision. We define precision as the true number of positive predicted observations divided by the total number of positive predicted observations. Precision is an important metric to us as the precision of the current process is very low as we can derive from our problem statement. Management estimates the precision of being around 10 percent. Furthermore, we have the metric specificity. We define specificity as the true number of negative predicted observations divided by the total number of true negative observations. Also, we have the metric negative predictive value (NPV). We define the negative predictive value as all true negative observations divided by all negative predicted observations. Both specificity and the negative predictive value are less important metrics in this study because they reflect the performance considering negative cases (*no revision needed*). These cases are less relevant to us as they do not require a revision.

Lastly, we have the performance metric recall (also known as sensitivity). We define recall as the number of true predicted positive observations divided by the total number of true positive observations (Zhu et al., 2010). Recall reflects how good the model is at determining all truly positive observations. Therefore, recall is another important performance metric for us. It reflects the percentage of clients that need a revision that is correctly identified by the RBR model.

We conclude that precision and recall are relevant metrics to us, with precision being the most important. Management prefers high precision over a high recall as other processes in the bank, such as arrears management and UTP triggers, could be safety nets that reduce the costs of a false negative prediction of the model. Preferably, we would quantify the cost of a false positive and a false negative to justify the preferences for higher precision over higher recall. However, we found this to be a very large and difficult task. Also, a quantification of especially a false negative would be very questionable. It would be hard to determine the PD decreases for a client would they have been signaled and received forbearances measures. For every situation of a client, the PD decrease would be different. On top of that, every client has a different loss given default. Therefore you could get a mean cost of a false negative but for every individual client, the cost will be different, likely with a significant standard deviation. Besides, as the number of false negatives in our dataset is limited, our ability to draw precise statistical conclusions from the data is limited. Therefore, we chose to not include quantifying the costs of a false negative or false positive in the scope of this study.

Conveniently, there is a metric that combines precision and recall. This is the F1 score. The F1 score represents the harmonic mean performance in precision and recall of a model in one number. Just as all earlier mentioned performance metrics, the F1 score can range from 0 to 1, with 0 the worst score possible and 1 the best score possible (Lipton et al., 2014).

$$F1 = \frac{2TP}{2TP + FP + FN} = \frac{2 * precision * recall}{precision + recall}$$

The F1 score can be adjusted to give unequal weights to precision and recall. The concept used for giving precision and recall different weights is the F_β-measure. In this measure, a $\beta < 1$ gives more importance to precision, with a lower β resulting in more relative importance for precision. When $\beta > 1$, more weight is given to recall, with a higher β resulting in more relative importance for recall (Chinchor, 1992). As precision is more important than recall according to management, we choose to use $\beta = 1/3$ in this study.

$$F_\beta = \frac{(1 + \beta^2) * TP}{(1 + \beta^2) * TP + \beta^2 * FN + FP} = \frac{(1 + \beta^2) * precision * recall}{(1 + \beta^2) * precision + recall}$$

Summarizing, recall and precision are relevant metrics in this study. The F1 score combines both recall and precision into one metric. Using the F_β-measure, we can adapt the F1 score to take into account the larger importance of precision over recall. Therefore, we will use the $F_{0.33}$ measure to decide which model performs best in this study.

2.4. Performance existing risk-based revision model

In this section, we discuss how we make the distinction between positive observations and negative observations. Also, we implement these definitions and measure the current performance of the RBR model.

2.4.1. Defining positive and negative observations

In the performance of the existing RBR model we define a positive case as '*revision needed'*. We define a negative case as '*no revision needed'*. However, the underlying question here is what determines whether a revision is actually needed. We found that there is currently no precise description or working instruction available at the bank of which type of clients should actually be revised and which not. Therefore we defined these cases ourselves.

In consultation with management, we decided that we define a *'revision needed'* for the true positive observations by taking a reference date and looking into the future whether a risk-based revision occurred within the subsequent three months after the observation was signaled by the model. For example, if the model signaled a client on 1 March 2020, then we look between 1 March 2020 and 1 June 2020 for the occurrence of a risk-based revision. When this is the case, we say the observation is true positive. If a risk-based revision did not occur in this period, we say the observation is false positive. A disadvantage of this methodology is the potential for incorrect classification of false positives as due to limited staff capacity, the bank is not able to perform a large number of revisions. Potentially this resulted in clients that were on the shortlist and not revised but which should have been, resulting in more false positives due to the methodology that are actually true positives.

We found that it is not straightforward to properly define/identify false negative observations. Therefore, we chose to identify the false negative observations by taking a reference date and looking into the future whether a default occurred within the subsequent three months with an additional filter. The additional filter excludes defaults resulting from life events of clients or other unpredictable events, such as the disease of a client. These types of events cannot be predicted by the bank or occur at random so therefore it would not be logical to label these clients as positives. We chose this method because it is not feasible to evaluate every negative prediction of the current RBR to whether a forbearance measure would be needed using the same manual procedure currently used by employees evaluating the clients on the shortlist produced by the RBR model. We thought of the possibility of detecting false negatives by also including clients that received a revision but were not signaled by the model. However, this turned out to be not feasible as a revision is executed in multiple processes of the bank. We found it was not technically possible with the data available to us to make distinctions between revisions due to a bad financial situation at a client and due to other reasons. Summarizing, we identify false negatives using a three-month lookahead on whether a default occurred.

We found that the method we implemented does not work perfectly; as for most of the observations identified by this method, it is questionable or unclear whether they would have required a revision. Also, the false negative group of clients that did not go into default but nearly did, that would have received forbearances measures would they have been signaled by the current model, is not identified. The question here is whether a default is a proper criterion as for a client that did not default it does not necessarily hold that forbearance measures could not have been applied. This is especially important as the Volksbank profiles itself as the bank that goes the extra mile for its clients. However, a more appropriate method could not be identified without a significantly larger time investment. We decided not to make this time investment as the time available for this study is limited. Instead, we acknowledge the recall score of the current model is likely lower than identified using this method.

Table 2: Explicit definitions of TP, FP, FN and TN.

Type	Definition
True positive	*An observation is true positive when an observation was signaled by the model and in the subsequent three months, a risk-based revision took place.*
False positive	*An observation is false positive when an observation was signaled by the model and in the subsequent three months, no risk-based revision took place.*
False negative	*An observation is false negative when an observation was not signaled by the model but defaulted in the subsequent three months without the events of a decease, an enforcement order, the direct effect of the Covid-19 pandemic, the being under the supervision of special asset management, business termination without financial problems, a too high LTV ratio or other data quality issues occurring.*
True negative	*An observation is true negative when an observation was not signaled by the model and not defaulted in the subsequent three months.*

2.4.2. Numeric performance

After implementing our definitions of revision needed and performing data cleaning activities (see Appendix A), we calculated the performance metrics of the current model (we discuss the dataset in Section 2.5). The calculated performance metrics can be found in Table 3. Also, we visualized performance with a confusion matrix, which can be found in Figure 3. As expected, the precision of the current model is very low, even lower than the 10% estimated by management. The recall score of 80% reflects an estimation based on our applied definition for false negatives observations of which we expect to be lower in reality. The combination of both the precision and recall score result in a low $F_{0.33}$ of 7%. The confusion matrix shows the distribution of the observations in our dataset over the TPs, FPs, FNs and TNs. The confusion matrix clearly indicates that there is a class imbalance. We will need to take this fact into consideration when creating a new model.

Table 3: Performance metrics of the current RBR model. The recall reflects an estimation based on our chosen method to identify the number of false negatives.

Metric	Value
Accuracy	97.5%
Precision	6.0%
Specificity	97.6%
Negative Predictive Value	99.9%
Recall	79.5%
F1 Score	11.1%
$F_{0.33}$ Score	6.6%

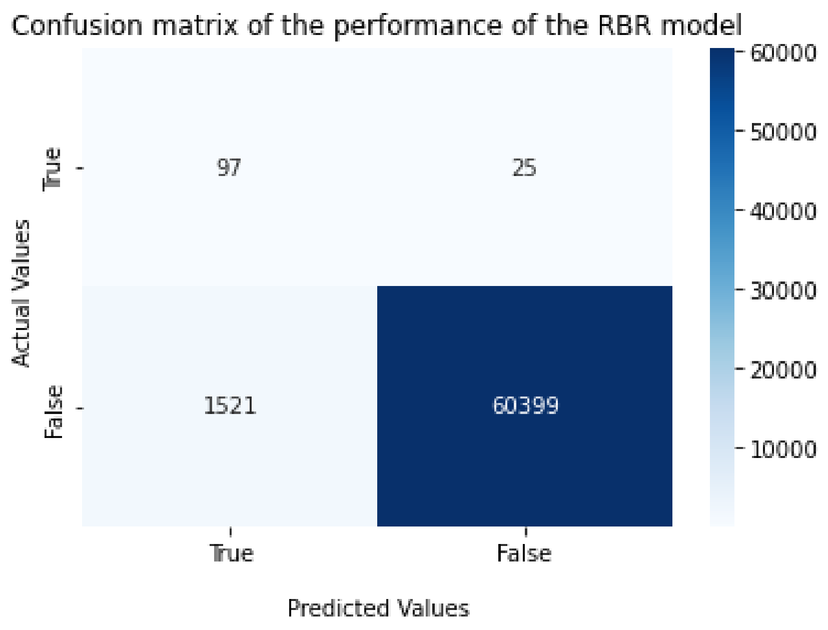

Figure 3: Confusion matrix of the performance of the current RBR model. Clockwise starting top-left, the number of true positives, false negatives (estimation), true negatives and false positives.

2.5. Characteristics of the available data

The data we use in this study we retrieve from the internal data warehouse of the business department of the bank. This limits the number of potential features we can use and create because the data stored in the data warehouse is limited. The data from the bank is highly confidential, therefore we cannot elaborate in the fullest detail on the data, but we disclose what we can.

Each instance in the dataset we retrieved is an observation of one client in a certain month. Giving multiple entries of one client but at different moments in time. Therefore, instances of the same client are related and can be seen as time-series observations. This makes it possible to use feature engineering to derive new features based on the change between variables of the previous month(s) of the same client. The timeframe of the data ranges from when the initial RBR model was taken into use to the most recent data available. So our dataset only comprises of instances when the RBR model was active. The total number of instances in our dataset is 62042, making it at first glance a reasonably large dataset. However, the number of positive observations, the observations we want to predict, is small, resulting in limited relevant observations for us to work with. We performed data cleaning to make the retrieved data suitable for model development. See Appendix A for the description of these steps.

The features in our dataset consist of numerical as well as categorical variables. In total, we have 10 numerical features that we directly retrieve from the data warehouse. We retrieve 4 categorical features from the data warehouse, such as the legal entity of the client. The numerical variables are a mixture of variables being the output of other (credit risk) models, such as a PD or an EL, and variables that are not an output of other models, such as a LTV ratio. We identified that the current PD model, which output we use, the PD, as a feature in our dataset, makes use of some macroeconomic factors, factors on which we cannot elaborate in this report. This means that the effect of current macroeconomic factors is already present in our dataset. It is reflected in the PD. This finding further supports our decision that we will not further look into the impact of future macroeconomic factors as macroeconomic factors already have a role in the PD. We investigated whether it was possible to obtain the underlying drivers of the PD model as data for our model. We found that this would not be straightforward to achieve for various technical reasons. Therefore we decided not to include the underlying drivers in our dataset.

Next to the features directly retrieved from the data warehouse, we also derived/computed new features based on other features, for example, changes in ratios between months. This action was taken because these features have the potential to unlock information that is not accurately represented in the features we can directly retrieve from the data warehouse. Also, the current model makes use of these features as it looks at for example the PD ratio development over the last months. The decision on which features to derive ourselves was taken in consultation with management and data engineers of the bank. After one-hot encoding the categorical variables of our dataset, the process that splits up every category of a categorical variable into a new Boolean feature which is needed to apply most ML learning algorithms, we have a dataset containing a total of 68 features.

We analyzed how the variables in our dataset relate to each other. We tested on multicollinearity of our numerical variables that we directly retrieved from the data warehouse. Multicollinearity is the linear relation/correlation between variables (Fox & Monette, 1992). The variance inflation factor (VIF) is a metric to measure multicollinearity between variables. For the interpretation of the VIF, see Table 4.

Table 4: After (Daoud, 2017), Interpretation of the variance inflation factor.

VIF = 1	Conclusion
VIF = 1	Not correlated
1 < VIF ≤ 5	Moderately correlated
5 < VIF ≤ 10	Highly correlated
VIF > 10	Very highly correlated

Table 5: The variance inflation factor of the numeric variables of our dataset.

Feature	1	2	3	4	5	6	7	8	9	10
VIF	6.4562	1.61512	2.24593	1.66965	4.79141	1.57493	1.06661	1.05741	1.5231	2.77923

We measured multicollinearity because high levels of correlation between variables can result in certain ML algorithms to not perform well. A high correlation between a group of variables makes it difficult to determine the importance of a single variable in that group. Variables that we derived from other variables we left out of this analysis, as these naturally will have a high correlation with the variables they are derived from. Also, we left out the multicollinearity analysis of our categorical variables because we estimate that the time investment this would need would not justify the benefit we would receive from this. The VIFs we obtained on our dataset can be found in

Table 5.

We observe that feature 1 already has a high VIF score. We expect that VIF scores would even be higher when we added the variables derived from other variables. Therefore, we conclude that our dataset contains some highly correlated variables. This causes difficulties when implementing certain ML models as some do not perform well when variables are correlated.

2.6. Summary

In this chapter, we discussed the context in which this study is performed. The RBR process has the goal to identify clients that are not yet in the picture but do need forbearance measures to prevent default. The process is not applied to all clients but only to those with exposure between certain levels (€$\mathbf{E_2}$ and €$\mathbf{E_3}$). There are two decision rules currently used by employees reviewing the shortlist produced by the current RBR model. The first rule is an increasing amount outstanding in the current account in combination with a decreasing turnover. The second rule is no payment traffic over the current account. We established that we should investigate how to implement aspects of these rules in a new model. Next, we presented the financial ratios that are used in the current RBR model. These are the probability of default, expected loss, exposure at default and the loan to value ratio.

Subsequently, we discussed how we can measure the performance of a RBR model. We introduced the concepts of true positive, false positive, true negative and false negative observations and the performance metrics we can determine from them. Precision and recall are relevant performance metrics in our study. We established that management finds precision more important than recall as there are other safety nets that reduce the costs related to false negative observations. We chose to not quantify the costs of false negatives and false positives due to the challenges of quantifying the costs of false negatives to support the preference of the management. The $F_{0.33}$ score combines the performance in terms of precision and recall into one score, with precision having a larger weight. Therefore, we chose this metric as the main metric to decide on what the performance is of a model in this study.

Next, we presented how we define positive observations. We say that a client needed a revision when in the 3 months after the measuring moment a client underwent a risk-based revision or went into default without a life event. In evaluating the performance of the current model, we found as expected that the performance of the current model is lacking. The 6% precision of the current model we found is even lower than the 10% estimated by the management. Combined with the estimated recall of 80% this resulted in a $F_{0.33}$ score of only 7%.

Lastly, we presented characteristics of the data that is available to us in this study. Although we could not elaborate in the greatest detail on the data because of confidentially, we did investigate and presented how some features of our dataset correlate to each other. Examples of features in our dataset are the probability of default and the expected loss. We found that some features in our dataset are correlated. This can cause difficulties when implementing certain ML models at a later stage of this study because some models do not perform well when variables are correlated. These difficulties we will discuss further in the next chapter when we will introduce ML theory.

3. Theoretical framework

The previous chapter described the context of this study. In this chapter, we establish the theoretical basis needed for this study. In Section 3.1 we discuss the reasons for implementing ML in this study. Next in Section 3.2, we present basic ML concepts before we introduce more advanced concepts in the following sections. In Section 3.3, we elaborate on model validation and in Section 3.4 we cover several ML models. Subsequently, in Section 3.5, we discuss various ML techniques that can be used to overcome dataset issues or improve the performance of ML models introduced in Section 3.4. Lastly, in Section 3.6, we summarize this chapter.

3.1. Justification machine learning

In this section, we present why we implement ML techniques. We present two possible ways in which we could improve the performance of the RBR model and discuss the advantages and disadvantages of each option.

The current situation is that we have a RBR model that does not perform satisfactory. To have a RBR model with better performance, we can improve the current triggering conditions model or we can create a completely new model using ML techniques. We chose the latter. In the following paragraphs, we will motivate this decision.

Improving the current model would require us to adapt, create new, or remove current triggering conditions of the model. This would be a process in which we would have to identify the important attributes manually. We could interview employees about what they think are important variables in the process and analyze the data by hand to identify certain patterns. Next, after we identified the important attributes, we could perform strategies such as fractional factorial design or full factorial design to discover which combination of threshold values would perform best (Law, 2015, pp. 629-692).

In theory, this strategy could work to improve the model. We expect that we could rather quickly achieve improvements in the performance of the model. However, we expect that the improvement would be limited. Interviewing employees could result in a bias on which attributes are important. Besides, employees could indicate that a certain attribute is important but this does not have to be the truth. Validation of the importance of the attributes is hard. Also, the results of analyzing the data to find patterns by hand are limited. It is impossible to find all relations in large datasets by hand for humans. Furthermore, when adding more attributes to the model, we will face the curse of dimensionality in evaluating which combination of attributes to use is best. The curse of dimensionality is the increasing complexity combined with the increasing sensibility to overfitting when adding more attributes. The number of combinations exponentially increases when adding extra attributes. This would result in an unmanageable number of potential models to evaluate. Besides, adding more attributes makes the data sparser and therefore overfitting is more likely. Although, strategies exist to soften this problem to an extent (Law, 2015, pp. 629-692). Lastly, implementing a strategy of improving the current model would not give us insight into the relative importance of attributes of clients which is a request of the management of the bank.

The second option is to create a new RBR model; a model based on ML techniques. The advantages of ML techniques are according to Khanzode & Sarode (2020) that trends and patterns relatively easily can be identified, no human intervention is needed, they allow for continuous improvement, they can handle multi-dimensional and multi-variety data and they allow for wide applications. Disadvantages are data acquisition, time and resources, interpretation of results and high error-susceptibility. For detailed descriptions, see Khanzode & Sarode (2020).

Concluding, if we compare the two options of improving the current model or making a new model, we argued that making a new model is the best alternative. The advantages of making a new model and the disadvantages of improving the current model outweigh the disadvantages of making a new model and the advantages of improving the current model. The decisive factors in this decision are that by making a new model we can fulfill the request of the management to give insight into the most important factors in predicting which clients need forbearance measures and we are likely to create a more accurate model.

3.2. Introduction Machine Learning

In this section, we will introduce and explain several basic concepts in ML to establish a solid base before introducing specific models and techniques. ML is a branch of artificial intelligence that uses data to predict future outcomes, to cluster observations, or to detect patterns. The field of machine learning is the study of algorithms that allow computer programs to automatically improve through experience (Mitchell, 1997). With ML, the goal is to estimate output based on one or more inputs. Dividing ML into subdomains, supervised learning is performed when outputs are known and unsupervised learning is performed when outputs are unknown. In unsupervised learning, the computer algorithms detect patterns in data without having the output. In supervised learning, we provide computer algorithms examples from which input resulted in what output from which the algorithms can learn and detect patterns. We speak of a regression problem if the output variable is continuous or quantitative. When the output variable is categorical or qualitative we speak of a classification problem (James et al., 2021). The set of input variables in a dataset we know as features, predictors, or independent variables. The output variable is often referred to as the label, the target, the outcome, or the dependent variable. A set of measurements in a dataset that belongs to one observation/occurrence we know as an instance (Hoogendoorn & Funk, 2017) (James et al., 2021). Applying ML techniques can generally have two goals. The goal can be to determine relationships between features and the response, which we call inference. The goal of forecasting a response based on the features we call prediction.

3.2.1. Reducible and irreducible error

In prediction, we deal with two quantities influencing accuracy, the reducible error and the irreducible error (Loeffel, 2017). The reducible error is the error we have because we are not applying the most fitting machine learning method. The irreducible error is the error that remains after applying the most fitting machine learning method. Real-word data contains random noise or missing values, such that there will always be some error in our prediction (James et al., 2021).

3.2.2. Accuracy versus interpretability trade-off

Over the years, several types of ML models have been defined and studied. In ML, there does not exist a type of model that is best in every situation (Kotthoff, 2016). Depending on the problem, different models will perform better or worse (Lee & Shin, 2019). Each model has its strengths and weaknesses. Examples are interpretability of the model, speed, accuracy and size of the dataset required (Akinsola, 2017).

The general challenge in deciding which ML method to use is in the trade-off between accuracy and interpretability. Some flexible ML methods can achieve very high accuracy but are not interpretable. These flexible methods can be seen as black boxes (Sarkar et al., 2016). However, often interpretability of the model output is needed, as in this study. Interpretability can be interpreted in two ways, understanding how the model works (global interpretability) or knowing what caused a certain decision (local interpretability). In this study, we refer to local interpretability when we speak of the interpretability of a model. Interpretability is the capability to explain to the user how a decision or response is made. A ML algorithm is interpretable if its classification can be explained by conditional statements about the data (Valdes, et al., 2017). Regulators in for example the financial services industry often require reasons why certain decisions are made. Therefore, it is not always possible to use black-box type ML models (European Banking Authority, 2021).

3.2.3. Overfitting & underfitting and the bias-variance trade-off

Flexibility of a ML method is the degree to which the behavior of a method is influenced by the characteristics of the data. The most flexible ML method does not have to result in the most accurate predictions. A less flexible method can outperform a very flexible method. This counterintuitive phenomenon is the high potential for overfitting of flexible methods (James et al., 2021). Overfitting is the problem that due to the high degree of flexibility of the ML method, the noise in the training data is followed too strongly when creating a model (Hoogendoorn & Funk, 2017). The danger is that noise in the data will be seen as an underlying pattern in the data. If then new test data with other noise is given to the model this results in a worse performance of the model. However, the opposite could also happen for very inflexible ML methods. When the method is too restrictive, a result can be that the underlying data patterns are not fully captured by a model. Which also results in a worse performance of the model than what could be achievable. This phenomenon we call underfitting (James et al., 2021).

The overfitting & underfitting problem is partly the result of the bias-variance trade-off (James et al., 2021). The approximation error of ML models can be decomposed into three factors: The variance of the estimation, the bias of the estimation and the irreducible error. As we cannot decrease the irreducible error, the approximation error can be reduced by reducing the combination of the bias and the variance (Hastie et al., 2009). The bias refers to the error that is introduced by approximating real-life problems when there is not sufficient training data available to capture all the patterns in the data. Real-life problems are often complicated but can be estimated by simpler models making assumptions about the problem and therefore underfit the problem. These assumptions make it possible to make estimations but they do have the effect that there we will systemically prejudice our estimations (James et al., 2021). The variance refers to how our estimations would change if we would use another training set of data from the process we are predicting. High variance indicates that our estimations would change a lot when other training set data would be used. The variance is linked to the overfitting problem. The danger of high variance is that we get a significantly different model due to randomness in our training set data (Hastie et al., 2009). The bias and variance are competing quantities. Less flexible ML methods have a high bias and a low variance. Flexible ML methods have a low bias and a high variance. The challenge is to choose a ML method that minimizes the prediction error resulting from the bias and the variance.

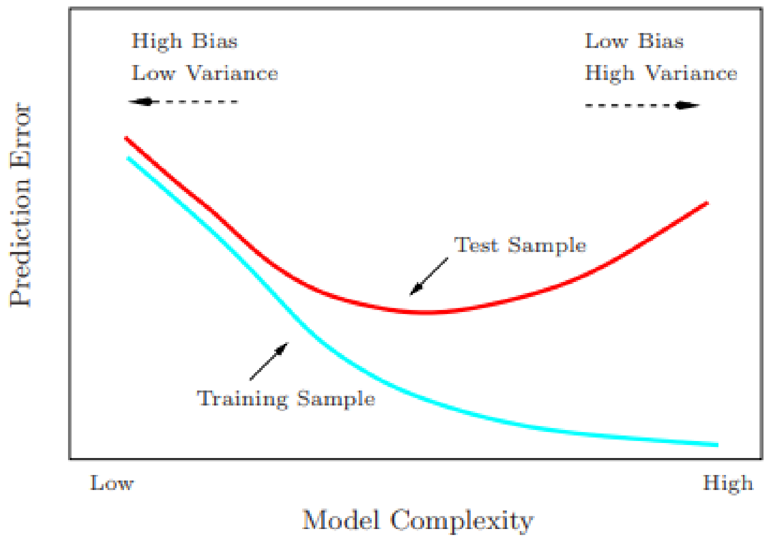

Figure 4: From Hastie et al. (2009), test and training error as a function of model complexity.

3.2.4. Scope

In this study, we will follow the methodology suggested by Lee & Shin (2019) for selecting which ML algorithm to use. The key driving factors in this methodology are whether we will apply unsupervised or supervised ML algorithms and whether interpretability is required. The methodology can be observed in Figure 5.

We are facing a supervised binary classification problem in this study. Also, an interpretable model is required. Regulators require that the prediction the RBR model makes is interpretable. Consequently, we should choose a ML algorithm with the highest accuracy with an acceptable level of interpretability. Hence, we will limit our discussion of ML models in this chapter to models that have an acceptable degree of interpretability.

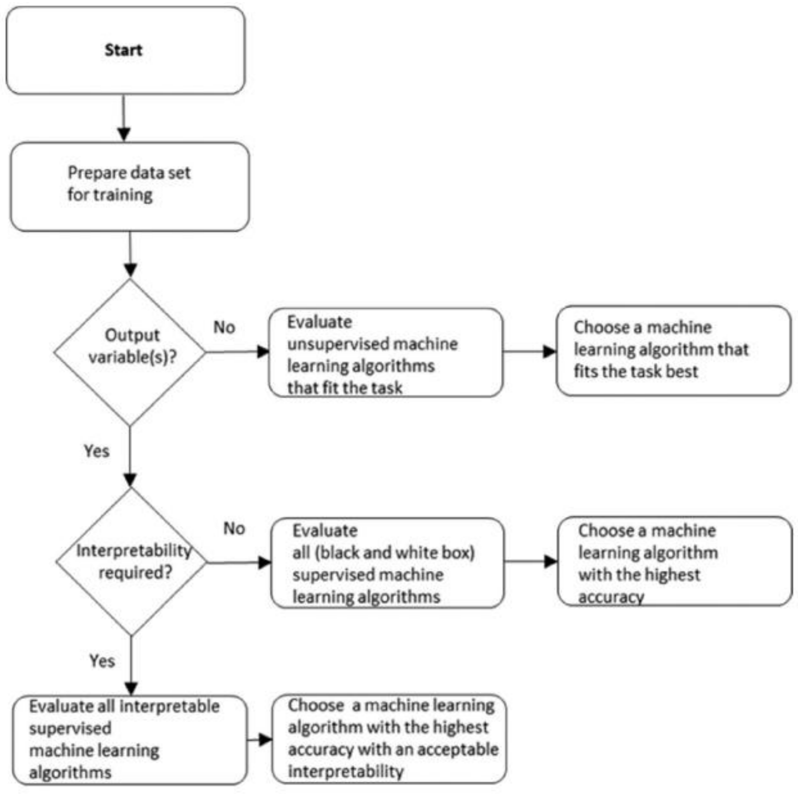
Start
Prepare data set for training
Output variable(s)?
No
Evaluate unsupervised machine learning algorithms that fit the task
Choose a machine learning algorithm that fits the task best
Yes
Interpretability required?
No
Evaluate all (black and white box) supervised machine learning algorithms
Choose a machine learning algorithm with the highest accuracy
Yes
Evaluate all interpretable supervised machine learning algorithms
Choose a machine learning algorithm with the highest accuracy with an acceptable interpretability

Figure 5: From Lee & Shin (2019), workflow for choosing the proper machine learning algorithm to use.

3.3. Model validation

Before we elaborate on specific ML models and techniques in Section 3.4 and Section 3.5, we first discuss model validation in ML. We do this because a grasp of some model validation techniques is needed to properly explain various ML models and techniques.

3.3.1. Validation set approach

In ML, model validation is the process of verifying whether a model performs as expected. A common procedure is the validation set approach. That is, evaluating a trained model with a test dataset. The test dataset is a separate part of data on the same process not used to train the model. A test dataset is used to test the generalization ability of a trained model (Alpaydin, 2020).

The validation set approach gives an unbiased approximation of the performance of a ML model to unseen data. So it estimates how the model would perform in real-world situations (Vabalas et al., 2019). However, the use of a Train/Test split approach comes at a cost. A drawback of the validation set approach is that the test error can highly depend on the instances that are in the training set and the instances that are in the test set. A solution for this problem can be to perform multiple iterations of the validation set approach and use the average performance. Another drawback of the approach is that sacrificing a portion of a dataset for validation has the result that fewer data will be available to perform statistical learning on (James et al., 2021). A commonly used Train/Test split is 80/20. However, also other ratios can be used, for example, 50/50, 60/40, 70/30, or 90/10. Which ratio works best depends on the size of the dataset. Generally, if the dataset is small, you want the percentage of training data to be larger to extract more information from the small number of observations you have (Vabalas et al., 2019).

3.3.2. K-Fold Cross-Validation

The K-Fold cross-validation (CV) approach is a widely used approach for estimating the performance of a model. The approach is to randomly divide the data into K equal-sized parts. Next, we train a model on the data using K–1 parts. Thereafter we use the part that we did not use to train the model as a test set. We then train another model with a different part left out and repeat the procedure K times. In the end, we measure performance by taking the average performance of all K models.

K-Fold CV is very efficient in the use of data as it allows data to be used for training and validation. Theoretically, K-Fold CV gives a more accurate test error estimate than the Validation set approach (Vabalas et al., 2019). A good choice for the number of folds K is 5 or 10. Also, any subset of predictors ML technique, such as Forward stepwise selection discussed in Section 3.5.1, should not be performed before applying CV (James et al., 2021). Figure 6 provides a visual representation of the validation set approach and K-Fold CV. In this figure, ACC is short for accuracy.

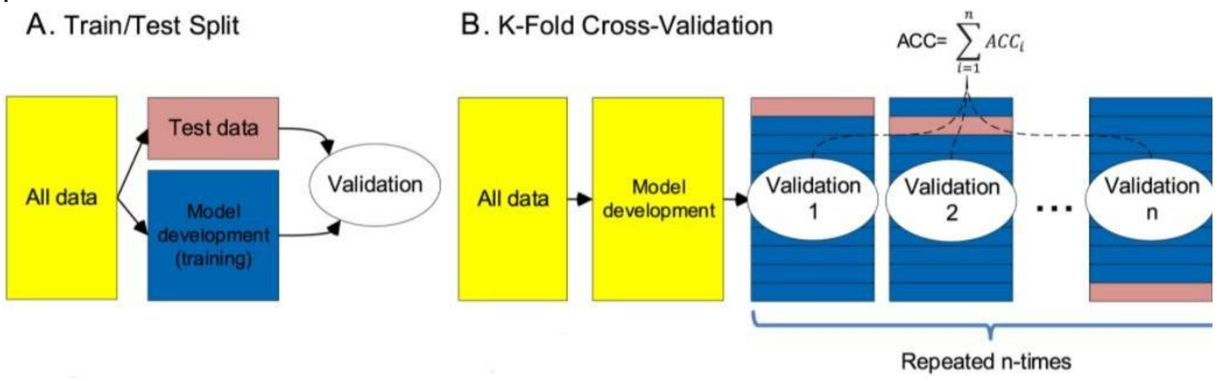

Figure 6: After Vabalas et al. (2019), Visualization of Train/Test Split versus K-Fold Cross-Validation.

3.4. Machine learning models

Various types of supervised classification ML models exist, for example, Support vector machines, Random forests, (Deep) neural networks, Logistic regression, Naïve Bayes, K-nearest neighbors and Decision trees (James et al., 2021). An overview of some of these algorithms in terms of interpretability versus accuracy can be found in Figure 7. We establish that Support Vector Machines (SVMs) and deep neural networks ML models do not have the acceptable level of interpretability needed for this research. Regulators demand that the reason the model signals a client should be derivable from the model. SVMs and deep neural networks do not meet this requirement. This requirement could limit the potential performance we could achieve as black box models in general have a better ability to capture non-linearity and interaction between features.

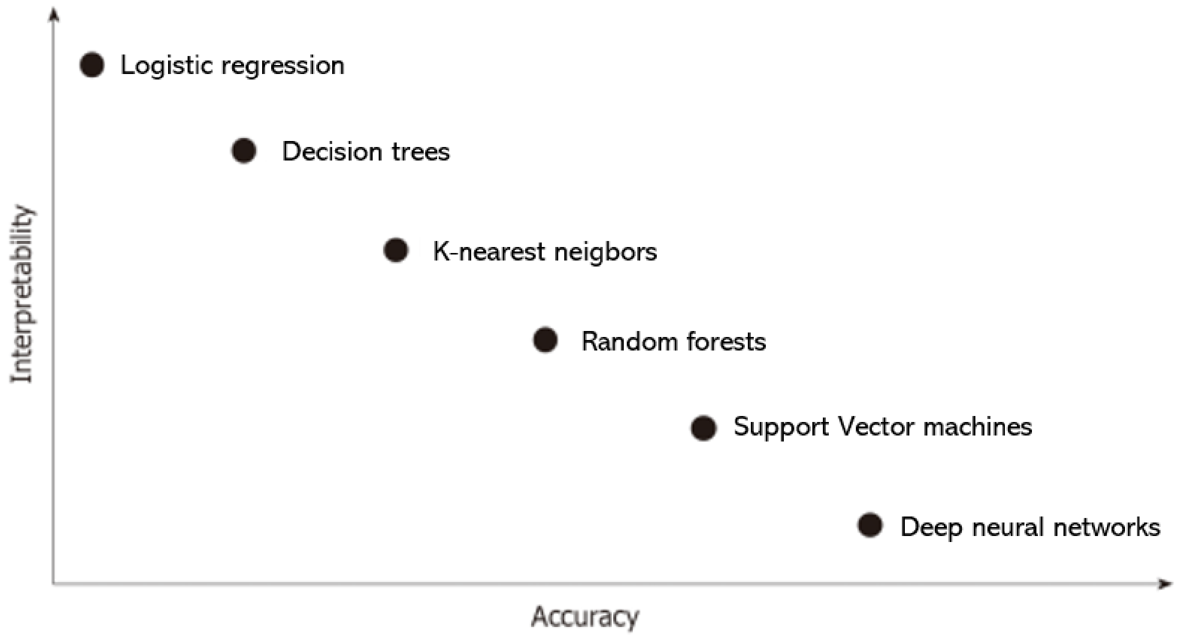

Figure 7: After Yang & Bang (2019), interpretability versus accuracy of various machine learning algorithms.

3.4.1. Logistic regression

Logistic regression is a ML classification algorithm very related to linear regression. We assume that the reader is familiar with linear regression. Otherwise, see Montgomery et al. (2021). Logistic regression uses mostly the same techniques as linear regression but instead of predicting quantitative variables, we predict qualitative variables with logistic regression. In a binary classification problem with multiple predictors, the logistic function is:

$$p(X) = \frac{e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}}{1+e^{\beta_0+\beta_1 x_1+\cdots+\beta_p x_p}}$$

The coefficients β_0 up to β_p are the regression coefficients, of which β_0 is the intercept. Generally, these coefficients are estimated using the maximum likelihood method. The logistic function gives an output between 0 and 1 and the output can be interpreted as a probability. If we have a binary dependent variable that can be 0 or 1, p(X) is the probability that the dependent variable is 1, given all independent variables represented in the vector X. In mathematical notation:

$$p(X) = \Pr(Y = 1 \,|X)$$

It is common to make the prediction $Y = 1$ if $p(X) > 0.5$, thereby predicting $Y = 1$ if the conditional probability of $Y = 1$ is larger than that of $Y = 0$. However, also other thresholds could be chosen if one wants to be more conservative in predicting a positive or negative outcome. The p-value of an independent variable indicates the likelihood that the relation between the independent variable and the response is by chance. A low p-value indicates a very high probability that there is a relationship between the variables.

For logistic regression, the scale of the variables is not very relevant. When the scale of a predictor x_j would be multiplied by a constant c, the resulting coefficient β_j of that predictor x_j will simply be estimated by a factor of $1/c$ compared to the coefficient of the variable when the variable would not be scaled. The coefficients in logistic regression are scale equivariant. This means that regardless of how a variable is scaled, the product $x_j\beta_j$ will be the same and thereby the impact of a certain predictor in the prediction (James et al., 2021).

Logistic regression performs well with small datasets (Juárez-Orozco et al., 2018). Also, it is a real advantage that its output can be interpreted as a probability as we discussed in the previous paragraphs of this section. Disadvantages of the method are that it can only provide linear solutions and that data assumptions are needed (Juárez-Orozco et al., 2018). Namely, observations should be independent of each other. That is, there should be no overlap between observations. Also, there should be little or no multicollinearity between features, there should be no influential outliers and there should be linearity in the logit for any continuous independent variables, meaning that a linear relationship exists between the logit-transformed outcomes and each (potentially transformed) independent variable (Stoltzfus, 2011). With respect to multicollinearity, such an issue could be solved by dropping features from the data that have a very high correlation with other features. The problem of influential outliers in the data can be solved by quantile-based capping and/or quantile-based flooring. Quantile-based capping is a technique that replaces data points that are greater than the **X**th percentile (e.g. 90th) with the 90th percentile value. Quantile-based flooring replaces the data points that are smaller than the **X**th (e.g. 10th) percentile with the 10th percentile.

3.4.2. Decision trees

A decision tree is an intuitive method to split into segments based on certain decision rules. A decision tree can both be used for regression and classification problems. A decision tree consists of nodes and branches. The node on top of a tree, the location at which the tree starts, is referred to as the root node or decision node. The nodes on the bottom of a tree are called leaf nodes. These represent the prediction the tree makes. All nodes that are not leaf nodes are internal nodes, including the root node. Internal nodes represent choices on which path an observation should follow. Each path that can be followed in a decision tree is called a branch. Each choice or question in the tree is referred to as a split (Song & Lu, 2015).

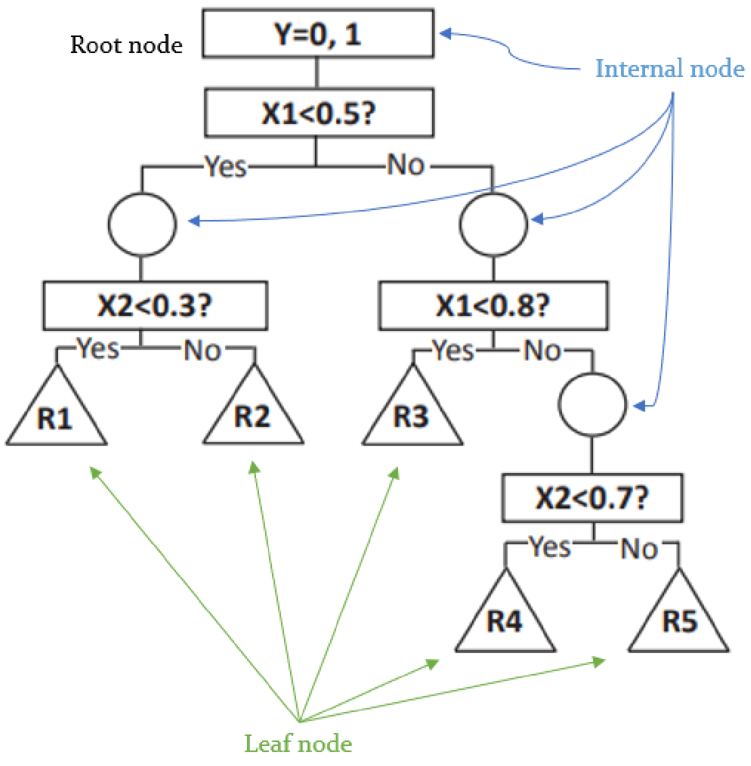

Figure 8: After Song & Lu (2015), an example of a decision tree with binary dependent variable Y.

In a classification tree, we make predictions on new observations based on which leaf node an observation ends up in. The most occurring outcome in a leaf node of the training observations is the prediction we make for new observations. Therefore, the degree of purity in a node should be optimized. That is, all observations in one node should belong to one group as much as possible

(James et al., 2021). The most common measures used to indicate purity in a classification decision tree are the Gini index and entropy. The Gini index is defined as:

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

Here, $\hat{p}_{mk}$ represents the ratio of training instances in the in node m that are from class k (Hastie et al., 2009). A small value indicates a lot of observations from one class.

Entropy is defined as:

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

Just as the Gini index, entropy takes on small values if a lot of observations are from the same class (James et al., 2021).

To build a decision tree, we start at the top of the tree with all observations and make a split based on a characteristic of the observations. Which split we make is based on the split that creates the smallest purity value. We continue the splitting until a pre-determined stopping criterion is met (Song & Lu, 2015). This method is seen as a top-down, greedy approach. Greedy because we only select splits that are best at that particular step, rather than looking further to future splits that could create a better tree after future splits. Top-down because we begin at the top of the tree. This method is also known as recursive binary splitting (James et al., 2021).

A stopping criterion could for example be that no leaf node contains more than 5 observations. Such a criterion would create a very large tree for a reasonably sized dataset and result in good predictions but is likely to overfit the data. To avoid overfitting and underfitting, a rule-of-thumb is that the proportion of observations in a leaf node should be between 0.25% and 1.00% of the training dataset (Berry & Linoff, 1999).

Another solution for this problem is to grow a very large tree and shorten the tree backward to create a smaller tree. We refer to this as tree pruning. To determine how much a tree should be pruned, cost complexity pruning is commonly used (James et al., 2021). The parameter used with cost complexity pruning is alpha and a higher value for alpha results in a more pruned tree.

The use of decision trees has several advantages (Juárez-Orozco et al., 2018; James et al., 2021; Song & Lu, 2015).

1. Performs well in datasets with a lot of features
2. Few parameters to tune
3. Can handle categorical data
4. Decision trees are simple and have very good interpretation
5. Deriving the relative importance of variables is easy
6. Mirrors human decision making
7. Can be displayed graphically

The main disadvantage of the method is that it is a less competitive type of supervised learning. More advanced methods or more flexible methods often result in better performance (James et al., 2021).

3.4.3. Random forest

We ended the previous section with the main disadvantage of decision trees, the fact that their performance is lacking compared to other supervised learning methods. However, there is a method to overcome this problem. This method aggregates multiple decision trees into one model. Such a model we know as a random forest model. The random forest model uses more trees, which results in improved performance. However, the improved performance comes at the cost of reduced interpretability (James et al., 2021).

Bagging

Bootstrap aggregation or bagging is an ensemble method. That is, a method that combines multiple independent models. This method has the goal to reduce the variance of statistical learning. The method is frequently used with decision trees (James et al., 2021). In this method, multiple training sets are created using the bootstrap technique. In the bootstrap technique, multiple samples are drawn of a dataset with replacement (Biau & Scornet, 2016). Each of these multiple samples is then used to create its own decision tree. The class that is eventually predicted is the most frequently occurring prediction (majority vote) of all decision trees combined. The multiple decision trees are not pruned afterward, as would introduce more bias in the model and by applying bagging, the variance of the model is already decreased (James et al., 2021).

From bagging to random forests

A random forest is an improved version of the bagging technique. By using a small tweak in the technique, it decorrelates the trees, which reduces the variance. With random forests, the same procedure as bagging is used but instead of using all predictors, only a random subset of m predictors is used to create a tree. This random subset of predictors is changed for every tree (James et al., 2021).

Parameters

Random forests require two not earlier parameters to be set. The number of m predictors in the random subset and the number of trees. As a rule-of-thumb, the number m is set to the square root of the number of predictors in the dataset (Hastie et al., 2009). The cost of more trees is a longer computation time. Also, more trees will not have to result in a better model. At a certain threshold of trees, a random forest model will no longer significantly improve. Oshiro et al. (2012) found that the number of trees should be between 64 to 128.

Interpretability

Conveniently, most modern programming languages have a built-in measure that represents feature importance and so the interpretability of random forest models. The degree of importance of a feature is represented in a feature importance score. The importance scores of the features are calculated on the mean and standard deviation of the accumulation of the impurity decrease within the trees. So a higher score of a feature reflects that the feature is better at separating the two classes in our trees compared to the other features. From this, we can conclude that a feature is more important. An alternative is to calculate the permutation feature importance scores. The permutation feature importance score is computationally intensive but can be used regardless of the type of ML model. The basic idea of the technique is to observe what happens to the accuracy of the model when feature values are randomly shuffled. If the decrease in performance is low, the feature has low importance. If the decrease in performance is high, the feature has high importance (Orlenko & Moore, 2021).

3.5. Machine learning techniques

In this section, we discuss various machine learning techniques that can be used to improve the performance of ML models or overcome dataset issues. We first cover variable subset selection. Then we discuss alternative fitting procedures with logistic regression in Section 3.5.2. Next, we cover in Section 3.5.3 techniques for handling imbalanced datasets. In Section 3.5.4, we discuss techniques to handle missing data. Finally, in Section 3.5.5, we discuss hyperparameter tuning.

In Section 3.4.1, we discussed the logistic regression model. In this model, a maximum likelihood fitting procedure is used to determine the coefficients indicating the weights of the independent variables of the model. However, including all available variables in a model does not have to result in the highest accuracy and interpretability (James et al., 2021). Often, a subset of the independent variables can achieve higher accuracy and better interpretability. Also, using slightly other fitting procedures could result in better models. Therefore, we will discuss techniques for variable subset selection and alternative fitting procedures in Section 3.5.1 and Section 3.5.2.

3.5.1. Variable subset selection

Intuitively it makes sense to select a subset of variables from a dataset that can make the best model. However, implementing the best subset strategy is not as straightforward as it seems. This is due to computational limitations. We have the problem that as the number of predictors *p* increases, the number of subset combinations grows exponentially (Hastie et al., 2009). When we have p number of predictors, we have a number of 2^p possible subset combinations. If we then would for example have 30 predictors, we would have over 1 billion combinations. This would make it computationally impossible to determine the subset that would result in the best model. We will discuss two methods to overcome the computational problem. Namely, forward stepwise selection and backward stepwise selection

Forward Stepwise Selection

In forward stepwise selection, we start with a model containing no predictors and check by adding which predictor of our dataset would give the largest improvement in performance. After having established this, we add that predictor to the model and repeat the process. We will check which additional predictor would have the largest additional improvement and add this predictor to our model. We repeat this procedure until we have a model containing all predictors of our dataset (Hastie et al., 2009). Applying forward stepwise selection results in only having to fit $1+p(p+1)/2$ models (James et al., 2021). Which for p=30 would result in 466 models compared to over 1 billion.

Backward Stepwise Selection

Backward stepwise selection is very similar to forward stepwise selection. However, in backward stepwise selection, we start with a model containing all predictors and check removing which predictor from the model would result in the best model and take again iterative steps from here (Hastie et al., 2009). This procedure fits the same number of models as forward stepwise selection.

It is important to realize that forward and backward selection do not have to result in the same subsets. Also, both strategies do not guarantee to find the best subset possible. Evaluating which subset is best could be performed using cross-validation, Akaike information criterion (AIC), Bayesian information criterion (BIC), or adjusted R^2 (Pereira et al., 2015). The AIC, BIC and adjusted R^2 are measures that indicate in some form the relative quality of a statistical model for a given set of data (Taddy, 2019). When using CV to evaluate the subset, it is common to use data not yet seen by the model. That is, use the first 50% of the data to find the subsets of features using forward or backward stepwise selection. Then use the other 50% of the data to evaluate the performance of the subsets (James et al., 2021).

3.5.2. Other fitting procedures in regression

Conventional logistic regression maximizes the likelihood function to fit its coefficients. Alternatively of using subsets of predictors for increasing performance, we can apply shrinkage methods. Shrinkage methods use all predictors but the estimated coefficients are shrunken towards zero compared to estimates using least squares (Pereira et al., 2015). The advantage of shrunk coefficients is that it reduces model variance (James et al., 2021).

Ridge regression

In conventional logistic regression, the coefficients are obtained by maximizing the log-likelihood function:

$$l(\beta) = \sum_{i=1}^{n} [y_i x_i \beta - \log(1 + e^{x_i \beta})]$$

Ridge regression introduces an additional tuning parameter λ, which is determined separately, that places a penalty on the size of the coefficients in the log-likelihood function (Pereira et al., 2015). This L_2 penalty is added to the conventional function and results in the following function where p is the number of predictors (Duffy & Santner, 1989; Cessie & van Houwelingen, 1992):

$$l(\beta) = \sum_{i=1}^{n} [y_i x_i \beta - \log(1 + e^{x_i \beta})] - \lambda \sum_{j=1}^{p} \beta_j^2$$

The effect of the shrinkage penalties is that the estimates of $\beta_1, \ldots, \beta_p$ are closer to zero. A value for λ of zero results in the shrinkage penalty having no effect and as λ becomes larger all coefficients will shrink towards zero but to typically remain larger than zero. A good value for λ is selected using cross-validation (James et al., 2021).

Lasso regression

An alternative to ridge regression is lasso regression. The log-likelihood function that is maximized has the form (Hastie et al., 2009):

$$l(\beta) = \sum_{i=1}^{n} [y_i x_i \beta - \log(1 + e^{x_i \beta})] - \lambda \sum_{j=1}^{p} |\beta_j|$$

The advantage of lasso regression over ridge regression is that some coefficients can go to zero, thereby reducing the number of predictors. This has the effect that the interpretability of a lasso model is better than that of a ridge model (James et al., 2021). In lasso regression, the penalty term used is a L_1 penalty (Pereira et al., 2015).

Scale of predictors

For both lasso regression and ridge regression, the scale equivariant property does not hold due to the penalty placed on larger coefficients, for example, when the scale of a predictor would be increased with a factor c, in conventional logistic regression the coefficient of that predictor would be c times smaller than the coefficient of the predictor when the scale of the predictor would not be increased. However, now due to the penalty placed on larger coefficients, the smaller coefficient of this predictor will be less impacted by this penalty because the coefficient is smaller, resulting that the final model will be different. Concluding, the value $x_j \beta_{j,\lambda}$ can depend on the scaling of predictors. Lasso and ridge models generally perform best when predictors are of the same scale. To achieve that predictors are of the same scale, standardization by standard deviation can be used (James et al., 2021). Alternative standardization and transformation techniques for predictors can also be used, for example, log transformations or min-max normalization.

3.5.3. Techniques for imbalanced datasets

In this section, we will discuss several strategies to handle class imbalances. We speak of imbalanced datasets if the different classes in a dataset are not evenly distributed. In binary classification, The class that contains the majority of the observations is called the majority class. The under-represented other class is called the minority class (Nanni et al., 2015). With imbalanced datasets, most of the classification algorithms applied will be biased toward the majority class, which results in bad results in the prediction of the minority class. Possibly, it could even happen that an algorithm predicts every observation as a majority class, as this yields the correct prediction for most of the observations (Longadge et al., 2013).

Alternative cut-off

As we already covered in Section 3.4.1, the output of a logistic regression model can be interpreted as a conditional probability. Generally, we use 0.5 as the cut-off point on which class we assign to the instance. However, also other thresholds could be chosen if one wants to be more conservative in predicting a positive or negative outcome (James et al., 2021). Suppose we assign the number 0 to the majority class and the number 1 to the minority class. If we then lower the cut-off point, we will more often label an instance as the minority class. Tweaking the cut-off point could improve the performance of the model.

We can also use an alternative cut-off for random forest models. A standard random forest for classification makes its prediction by a majority vote of all its trees. In a binary classification problem, the prediction 1 or 0 is made when at least 50 percent of the trees make this prediction. Instead of using the 50 percent as a cut-off, a lower percentage could be used. This will result in fewer false negatives but more false positives. Therefore, the optimal cut-off percentage depends on the trade-off between the performance of a model in terms of precision and recall.

Undersampling

Undersampling is a method in which only a fraction of the instances of the majority class is used. By dropping a random fraction of records of the majority class, the dataset becomes more balanced (Longadge et al., 2013). A drawback of this procedure is that useful information in the dropped instances could be lost (Nanni et al., 2015).

Oversampling

Oversampling is a method in which instances of the minority class are randomly duplicated, which creates a more balanced dataset (Nanni et al., 2015). With implementing oversampling, one should take care not to compromise the validation of a model. It should be prevented that the same instance is both in the train and test set because this will cause overoptimism and overfitting (Santos et al., 2018).

SMOTE

SMOTE (Synthetic Minority Oversampling TEchnique), is a more advanced oversampling technique. With this technique, the minority class is oversampled by creating synthetic but very similar new observations of the minority class instead of oversampling with replacement (Chawla et al., 2002). Undersampling, oversampling and SMOTE can also be combined (Longadge et al., 2013).

Cost-sensitive learning techniques

Cost-sensitive learning is another technique to handle the imbalanced classes problem. In this technique, a different cost is assigned to false positives and false negatives. This results in a model that is more steered towards preventing false positives and false negatives (Nanni et al., 2015). Cost-sensitive learning cannot be implemented in every ML model (Longadge et al., 2013).

3.5.4. Techniques for handling missing data

More often than not, we face the problem of missing data in ML. Frequently some instances in a dataset have entries missing. This is a problem as some operations or ML models require or perform better with no missing entries. In this section, we discuss techniques to take care of the missing data problem.

Little & Rubin (2002) suggest that missing data can be divided into three classes. Batista & Monard (2003) define these classes as follows:

1. *Missing Completely At Random (MCAR).* We say that a datapoint is MCAR if the probability of a missing value of an instance does not depend on the known values of the instance. The interpretation of this is that the missing value occurred completely at random. In such a case, we can take care of the missing data point without introducing bias into our dataset.
2. *Missing At Random (MAR).* We say that a datapoint is MAR if the probability of a missing value of an instance may depend on the known values of the instance. However, not on the missing value itself.
3. *Not Missing At Random (NMAR).* We say that a datapoint is NMAR if the probability of a missing value of an instance may depend on what would be the value of that attribute.

If a missing datapoint is MAR or NMAR, implementing data treatment methods could result in introducing bias in a dataset. Therefore, it is clear that the classes (MCAR, MAR or NMAR) of the missing data points should be identified before implementing data treatment methods. If the decision is made to perform data treatment methods on datapoint that are MAR or NMAR, one should take into account that this can impact the performance of a ML model.

Generalizing, we can deal with missing data in two ways. We can delete the instances or features of the missing data points or we can use imputation, replacing missing values with estimates based on the entire dataset. Deletion is an applicable method when data points are MCAR. Listwise deletion is the removal of instances that have one or more missing values. However, the problem with deletion is that we often do not know whether the data points are MCAR, resulting that removal would reduce the statistical power of our later applied ML model or introduce bias into our data (Hippel, 2012).

There are various methods of imputation. A common method is mean, median or mode imputation. If an instance has one predictor value missing, we use the mean, median, or mode of the known observations of that predictor. This method works best when the number of missing values is small (Little & Rubin, 2020).

When a dataset holds time series data, another strategy is to use earlier or later observations of the same process, an individual, or in our case from the client. We impute the missing data points with measurements of earlier or later observations. We can use the methods Next Observation Carried Backward (NOCB) or Last Observation Carried Forward (LOCF). As both names suggest, NOCB uses a future observation and LOCF uses a past observation. A danger of such a strategy is that it could introduce bias into a dataset (Little & Rubin, 2020).

A more advanced method is to look at the K-Nearest neighbors of an instance. By looking at the average value of the missing datapoint of the nearest neighbors an estimate can be made for the missing datapoint (Little & Rubin, 2020). For all of the above-discussed imputation methods hold that the data points should not be NMAR as then the missing of the datapoint can hold valuable statistical information. When data points are NMAR, it is best to keep this information in the dataset by for example adding one additional category to a categorical feature or creating a derived Boolean feature based on the missing data point(s).

3.5.5. Hyperparameter tuning

Every ML model has hyperparameters. Hyperparameters are the explicitly specified factors that control the training process of a ML learning model (e.g. the number of trees in a random forest model). Intuitively, changing these hyperparameters changes the performance of the ML model. Hyperparameter tuning, which is also referred to as hyperparameter optimization, is the process of optimizing the settings of the hyperparameters to increase the performance of the ML model. A grid search is the most straightforward method to perform hyperparameter tuning. This method calculates the performance for all combinations of settings for the hyperparameters that are in the specified hyperparameter space. The advantage of the method is that it is easy to implement but the method has the drawback that it is not computationally efficient (Feurer & Hutter, 2019). For a grid search holds that the number of combinations grows exponentially when the set of parameter values that are checked grows. The effect is that often a large number of combinations exist which all are checked, although most combinations do not produce relevant results.

3.6. Summary

In this chapter, we discussed the theoretical framework of this study. The reason for implementing machine learning in this study is that we then are very likely to create a more accurate model than with other methods. Also, with machine learning, we can give insight into the most important factors in predicting which clients need forbearance measures. When deciding which machine learning model to use, the accuracy versus interpretability trade-off exists. Generally, the higher the accuracy of a model, the lower the interpretability of the model, with interpretability being the capability to explain to the user how a decision or response is made. The new RBR model needs a certain level of interpretability due to regulation. We identified 3 appealing machine learning models that have the desired level of interpretability. These are the logistic regression model, the decision tree model and the random forest model. A machine learning model should be validated. That is, the model should work for unseen data. Validation of the model can be performed by the validation set approach or K-Fold cross-validation. There exist various machine learning techniques that can improve the performance of machine learning models. With subset selection techniques, only a subset of the variables in a dataset is used which could result in increased performance. Also, when dealing with imbalanced datasets, techniques such as using an alternative cut-off point or the Synthetic Minority Oversampling Technique can be used to achieve better performance. When there are missing data points in a dataset, deletion or imputation can be used. To optimize the performance of a certain machine learning model, hyperparameter tuning can be used, for example, by performing a grid search. The design decisions we make about implementing the theory we introduced in this chapter we discuss in Chapter 4.

4. Model construction

In this chapter, we discuss how we construct a new RBR model. We will first state the steps we take in our model construction in Section 4.1. Subsequently, we cover in Section 4.2 how and why we focus on true positive observations and what design decisions we make specific to individual ML models we found to be appealing to construct. This section also covers our implementation of imbalanced dataset techniques and model validation methods.

4.1. Steps in model construction

The procedure we used to construct our new RBR model can be summarized into 8 steps. These steps can be found in Figure 9. Based on our study of ML literature, we identified 3 appealing ML models to construct: the logistic regression model, the decision tree model and the random forest model. The first step we take is to construct a basic logistic regression model. The second step is that we use the basic model to implement various more advanced techniques/variants of the logistic regression model. These same steps are applied for the decision tree model and the random forest model, creating first a basic variant of the model which we then make various variants of. Performing these steps gives us a good impression of which models and techniques have a good performance and which do not. This obtained knowledge we will use in the next step to determine which variant(s) of which model(s) we will further tune. We will select the variant(s) of the model(s) that have the highest performance on the $F_{0.33}$ score. Computational speed is the limiting factor in the number of models we select for hyperparameter tuning. We will select the number of models such that the total computational time of our hyperparameter tuning will not exceed 48 hours. The last step is then to perform hyperparameter tuning for the selected model(s) to determine whether we can further optimize the performance and decide on which specific model we recommend to become the new RBR model.

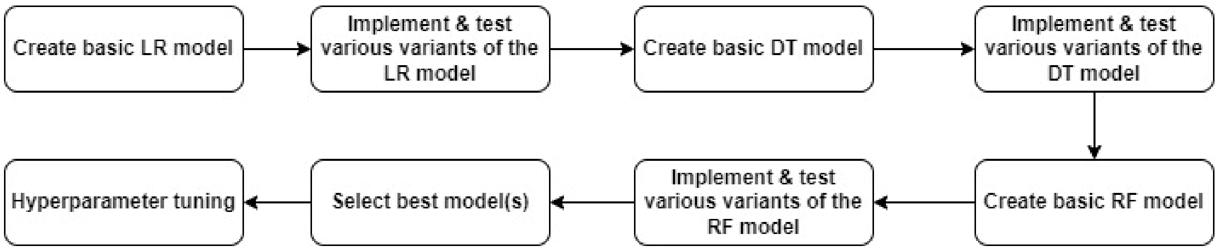
Create basic LR model
Implement & test various variants of the LR model
Create basic DT model
Implement & test various variants of the DT model
Create basic RF model
Implement & test various variants of the RF model
Select best model(s)
Hyperparameter tuning

Figure 9: Overview of steps in our model construction, LR = Logistic Regression, DT = Decision Tree, RF = Random Forest.

4.2. Design decisions

This section states the design decision we make in the construction of our models. We start in Section 4.2.1 with a decision considering focusing on true positives. Thereafter, in Sections 4.2.2, 4.2.3 and 4.2.4, we state our design decisions with respect to our logistic regression, decision tree and random forest models. We present which variants of these models we cover and explain the reasons for investigating certain variants. Next, in Section 4.2.5, we discuss how we implement imbalanced dataset techniques. Subsequently, we state how we implement model validation in Section 4.2.6. Finally, in Section 4.2.7, we close this section off with a statement about our design decisions.

4.2.1. Focus on true positives

Before we elaborate on design decisions related to specific ML models, we explain a decision we made related to the input for all models. As stated in Section 2.4.1, it is hard to identify false negative observations, causing that the method we decided on to identify these observations with is a non-perfect estimation. Data exploration showed that the observations that are false negative with our chosen method are significantly different than the observations that are true positive with respect to their feature values. We also found that our models performed worse when we included the false negatives in the set of positive observations we train the models with. The true positives are less often signaled by the new models. Because of these reasons we created the new models using only the true positives as positive observations in our dataset. We rather create a model based on only true positive observations and predict these observations better than also include doubtful false negative observations. The consequence of this decision is that when our model would be used in practice, it will not be as good at spotting false negative observations as when we would include the false negative observations. However, the current model does not spot the false negative observations at all, although the current model produces a lot of false positive observations. So compared to the old model, our new model will not perform worse with respect to signaling the false negative observations.

4.2.2. Design decisions of logistic regression models

The first appealing ML algorithm we identified is logistic regression. For a logistic regression model to work properly, there should be no influential outliers. Some numerical features in our data contain severe right-tailed outliers. Therefore, we implement quantile-based capping. We cap the features with outliers to the 99.5 percentile because this percentile results in no longer having very influential outliers but it maintains significant variance in the data to perform statistical analysis on. The effect of this action is that we expect the logistic regression model to perform better. Quantile-based flooring is not needed as the numerical features in consideration are already floored by the number zero.

As discussed, we can add a shrinkage penalty term to the log-likelihood function to be maximized for fitting the coefficients of the model. A L1 penalty represents a lasso model and a L2 penalty a ridge model, see Section 3.5.2. We investigate both options in addition to the basic regular model with no shrinkage penalty because more often than not a logistic regression model with a L1 penalty or a L2 penalty has better performance than without an added penalty. Also, we test whether standardizing the numerical variables in our dataset benefits the lasso and ridge model. Besides, we still investigate the model with no added penalty to establish a baseline performance.

In Section 2.5, we found that some variables in our dataset are correlated. However, the logistic regression model assumes that variables are not correlated. A solution is to drop correlated variables from our dataset. Conveniently, we can combine dropping variables from our dataset with implementing forward stepwise selection. Hence, we will implement forward stepwise selection to a maximum of 23 features, 23 features are 33 percent of the total 68 features of our dataset. We chose this maximum number of features because of the otherwise unmanageable computational time. We do not implement backward stepwise selection because of again the significant computational time it requires.

4.2.3. Design decisions of decision tree models

For constructing a decision tree model, we have 3 factors to decide on: the measure with which we calculate impurity, the stopping criterion used and the pruning of the tree. We will use the Gini index to calculate impurity as performance compared to using entropy is very similar but training the model with the Gini index is faster which is convenient as we will create a significant number of variants of the model. Also, we can still find out the effect of using the entropy criterion in a later grid search. As a stopping criterion for the leaf nodes, we will study multiple options. We use the very small numbers of 1, 3, 5 and 10 as well as the 0.25%, 0.50%, 0.75% and 1% of the number of observations as suggested by literature sources. Lastly, we will implement cost complexity pruning and also test multiple options for the tuning parameter alpha. For the decision tree model, outlier cleaning is not needed because decision trees are not sensitive to outliers. Since we will test multiple variants for the stopping criterion of the leaf nodes and the parameter alpha applied in the pruning of the tree, we have a significant number of models to test. However, the computation effort of the decision tree algorithm is manageable so the total computation time of all these models is not an issue.

4.2.4. Design decisions of random forest models

As the random forest model is basically multiple decision tree models combined, we have some overlap in our design choices. We will also use the Gini index in our basic random forest models for the same reasons as with the decision tree model; the faster computation time. We will use a number of 128 trees in our random forest models. We chose to be in the top of the 64 to 128 range suggested by Oshiro et al. (2012) as we prefer the potential higher performance of 128 trees to the shorter computation time of choosing fewer trees. For each tree in the basic models, we will only use m number of random features from the dataset. Here, we follow the rule-of-thumb and set m to the square root of the number of features in the dataset, resulting in that we will use 5 features per tree. We found that the random forest model had the best performance compared to logistic regression and the decision tree model, resulting in that we are performing hyperparameter tuning on the random forest model. We included using the Gini index versus entropy and the number of features used in a single tree as parameters in our hyperparameter tuning procedure.

We decided not to test forward or backward subset selection for the random forest model because of the significant computation time such procedures require. However, we will remove the subset of features that are not relevant to our random forest model to increase performance. We decide which features to include on the basis of the feature importance scores.

4.2.5. Implementation of imbalanced dataset techniques

Imbalanced dataset techniques have the goal to improve model performance when datasets are imbalanced. To achieve a better performance, we implement SMOTE combined with undersampling of the majority class to create balanced datasets. As a basic setup, we first oversample the minority class to obtain a ratio of 1:10 (1 positive observation per 10 negative observations). Then we undersample the majority class to obtain a ratio of 1:2. For the random forest model, we also implement the alternative cut-off strategy. We do not implement the cost-sensitive learning technique because of the time constraints in which this study is performed. We focus on the implementation the SMOTE and the alternative cut-off strategy.

4.2.6. Implementation of model validation

Model validation is needed to determine whether a ML model also performs as expected for new unseen observations. Therefore, we will perform the validation set approach as well as K-Fold CV, with K = 5, to perform validation of our models. Initially, we used the regular validation set approach with an 80/20 split. We found that performance drastically depended on instances that are in the train or test set. Therefore we perform multiple repetitions of the split, with each repetition having different random subsets of training and test data. The reason for not simply using K-Fold CV only is that CV for the random forest model with an alternative cut-off was not possible in the ML module we use to program the ML models in this study. The combination of implementing the validation set approach and K-Fold CV is a good solution to overcome this issue in our opinion. To make sure that the multiple repetitions of the validation set approach are comparable for different models, we will make use of the same random number stream. Thereby, eliminating that different train test set combinations cause a difference in performance between models.

4.2.7. Closing statement

This section presented various design decisions we make. We presented these with the goal to explain why we make certain design decisions, why we test certain variants of models and why we make certain trade-offs. We made a selection of the type of models and variants we test but if we would have had more time, more options could be explored. If possible, we made our decisions on the basis of recommendations from literature. Also, when provided with the same dataset as us, these statements in this section allow reproducing the models and results we found.

In the end, after testing and evaluating multiple types of models, we found that a random forest model with the criterion entropy, using 4 features per tree (resulting from the log 2 rule) and a cut-off of 0.35 has the best performance considering the preferences of management for the new model. The performance of this model and the performance of the intermediate models that we constructed to obtain this model we discuss in the next chapter.

5. Results

The previous chapter provided an overview of how we constructed our ML models. In this chapter, we present the results of the models we created. In Section 5.1, we discuss the numeric performance of the models we created and state which models and techniques we found to be effective. Next, in Section 5.2, we compare the new RBR model with the old model. In Section 5.3, we state what are the important features to predict the risk-based revisions we identified.

5.1. Numeric performance

As stated in Section 4.1, we first constructed basic logistic regression, decision tree and random forest models and resulting variations of those models to get an indication of which models and techniques have good performance and which do not. In Table 6, an overview of the performance basic models can be found. Here we tested the performance of the basic LR model, the basic LR model with outlier cleaning applied and/or the SMOTE applied, the basic DT model with or without the SMOTE and the basic RF model with or without the SMOTE. It is interesting that the $F_{0.33}$ score and the precision of the standard random forest model are the highest. The random forest model with the SMOTE has the highest recall but it has significantly lower precision and $F_{0.33}$ score than the random forest model without the SMOTE.

Table 6: Performance of the basic Logistic Regression model, basic Decision Tree model and basic Random Forest model with or without outlier cleaning or/and the SMOTE (with added undersampling) applied. Validation is performed by 5-fold cross-validation. The highest performance in terms of precision, recall, F1 score and $F_{0.33}$ score are highlighted.

Model	Accuracy		Precision		Recall		F1 Score	F0.33 Score
	Mean	Std	Mean	Std	Mean	Std	Mean	Mean
Logistic Regression	0.9982	0.0002	0.0222	0.0497	0.0100	0.0224	0.0138	0.0198
Logistic Regression + outlier cleaning	0.9983	0.0002	0.1000	0.2236	0.0100	0.0224	0.0182	0.0526
Logistic Regression + SMOTE	0.9513	0.0207	0.0288	0.0161	0.7458	0.1733	0.0550	0.0319
Logistic Regression + SMOTE + outlier cleaning	0.9478	0.0191	0.0248	0.0112	0.7458	0.1886	0.0479	0.0275
Decision Tree	0.9977	0.0002	0.2663	0.0524	0.2874	0.0923	0.2736	0.2683
Decision Tree + SMOTE	0.9940	0.0006	0.1514	0.0285	0.6074	0.0744	0.2421	0.1637
Random Forest	0.9986	0.0001	0.8400	0.1497	0.1753	0.0249	0.2874	0.6090
Random Forest + SMOTE	0.9968	0.0005	0.2791	0.0550	0.6284	0.0782	0.3833	0.2955

5.1.1. Performance logistic regression models

In this section, we discuss the performance of the LR models in detail. We observe that the most basic LR model performs very poorly with a mean precision of 2% and an extremely low mean recall of 1%. The basic LR model with outlier cleaning performs better but we see that the standard deviation of the precision is very high, indicating a lot of instability in the performance of the model. The basic LR with the SMOTE has a low precision of 3% but the recall of 75% is decent. We observe that the outlier cleaning has a minimum impact on the performance of the model when the SMOTE is applied. For more advanced variants of the LR model without the SMOTE (e.g. lasso or forward subset selection), we found similar very low recall scores. Therefore, we will not discuss the performance of these models further in this report because we now establish that these models are clearly not performing well enough.

The performance of more advanced variants of the LR model, the lasso regression model, the ridge regression model, and the model with forward stepwise selection applied, all with the SMOTE and outlier cleaning applied, can be found in Appendix B. We observe here that all more advanced variants outperform the basic LR model with the SMOTE applied. Forward stepwise selection performs best, although lasso regression comes close. Using forward stepwise selection, a precision of +- 8% can be achieved with a high recall of +- 90%, resulting in a $F_{0.33}$ score of +- 10%. This is an improvement over the current model but it is not the improvement we aim for. Concluding, using a logistic regression model is not a satisfactory solution. Unexpectedly, the performance of the lasso regression and ridge regression models was worse when we applied standardization but as we decided it was not worthwhile to further investigate this observation because the general performance of logistic regression was lacking.

5.1.2. Performance decision tree models

In this section, we discuss the performance of the DT models in detail. We observe in Table 6, that the basic DT model with or without the SMOTE applied significantly outperforms the basic LR models with the SMOTE in terms of precision (15% and 27% versus 3%), although the recall in both cases is lower (61% and 29% versus 75%) and the precision achieved is still rather low. However, the $F_{0.33}$ scores are higher compared to the basic LR models with the SMOTE, 27 and 16% versus 3%. The DT model without the SMOTE applied achieves a higher precision but a lower recall that its counterpart with the SMOTE.

The performance of more advanced variants of the DT model, variants with cost complexity pruning applied or with a different stopping criterion, can be found in Appendix C. Changing the stopping criterion could not significantly improve the basic DT model, both with and without the SMOTE applied. For the variant without the SMOTE, setting the minimum number of instances in a leaf up to and over the 155 even resulted in a precision of zero. This can be explained by the fact that there are significantly fewer than 155 positive observations given to the model, causing that the observations of the majority class in every node will always be with more than those observations from the minority class.

The DT models with cost complexity pruning applied could also not produce significantly better performance than the DT models without cost complexity pruning. We also observe here for the variant without the SMOTE that when the value of alpha, the parameter used in cost complexity pruning, becomes larger, the precision becomes zero. The larger value of alpha results in more tree pruning, which causes the majority class to always be with more observations in the leaf nodes. Overall we conclude that variants of a DT model are not a satisfactory solution for a new RBR model. The combinations of precision versus recall performance (27% vs. 29% or 15% vs. 61%) we can achieve with a DT model are not good enough to be able to improve the current RBR model.

5.1.3. Performance random forest models

In this section, we discuss the performance of the random forest models in detail. We observe in Table 6, that the basic RF model achieves a very high precision but a low recall, resulting in a significant increase in the $F_{0.33}$ score. This result signals that the RF model with an alternative cut-off probabilities could be interesting to explore as with this technique we can increase the recall by decreasing the precision. Possibly, we can find an optimum of high precision and high recall. The RF model with the SMOTE achieves a much lower precision, 28% versus 84%, but a higher recall, 63% versus 18%, with the result that the F1 score is higher for SMOTE variant of the model; 38% versus 29%. However, the $F_{0.33}$ score is higher for the RF model without the SMOTE; 61% versus 30%.

We created more advanced variants of the RF model by experimenting with alternative cut-offs because the basic RF model outperforms the other models and has more potential for further exploration. We tested alternative cut-off probabilities from 0.0125 to the regular 0.5 for both the RF model with and without the SMOTE. For the variant with the SMOTE, we changed our approach slightly. Instead of combining oversampling and undersampling to obtain a 1:2 ratio of positive versus negative observations, we chose to only use oversampling to obtain a 1:1 ratio. We also used only a subset of the features of our dataset. By identifying and thereafter not using the features that had a zero feature important score in the basic RF model. This resulted in decreasing the number of features from 68 to 27.

Table 7: Performance of the random forest model with various different cut-offs. The different cut-offs illustrate the precision versus recall combinations possible. Validation is performed by 20 repetitions of the validation set approach with a training size of 80%. The highest performance over the 20 repetitions in terms of precision, recall, F1 score and $F_{0.33}$ score are highlighted.

	Random forest									
Cut-off	**Accuracy**		**Precision**		**Recall**		**F1 Score**	**F0.33 Score**	**Grid Search**
	Mean	Std	Mean	Std	Mean	Std	Mean	Mean	
0.0125	0.9797	0.0013	0.0661	0.0135	0.9201	0.0607	0.1230	0.0728	No
0.025	0.9881	0.0011	0.1137	0.0301	0.9059	0.0505	0.2008	0.1246	No
0.0375	0.9898	0.0006	0.1186	0.0228	0.8701	0.0848	0.2080	0.1298	No
0.05	0.9928	0.0009	0.1635	0.0395	0.8184	0.0738	0.2701	0.1777	No
0.075	0.9951	0.0008	0.2218	0.0594	0.7688	0.0849	0.3406	0.2388	No
0.1	0.9965	0.0006	0.2788	0.0731	0.6768	0.0852	0.3898	0.2962	No
0.125	0.9972	0.0005	0.3298	0.0842	0.6229	0.0762	0.4235	0.3461	No
0.15	0.9979	0.0004	0.4131	0.1066	0.5572	0.0862	0.4654	0.4240	No
0.175	0.9983	0.0003	0.4864	0.1035	0.5257	0.0890	0.4955	0.4900	No
0.2	0.9985	0.0004	0.5509	0.1403	0.4734	0.0654	0.5017	0.5421	No
0.225	0.9986	0.0003	0.6015	0.1259	0.4337	0.0732	0.4958	0.5791	No
0.25	0.9987	0.0004	0.6630	0.1402	0.4146	0.0853	0.5026	0.6255	No
0.275	0.9987	0.0003	0.6866	0.1452	0.3506	0.0929	0.4565	0.6265	No
0.3	0.9987	0.0004	0.7111	0.1696	0.3290	0.0821	0.4423	0.6371	Yes
0.325	0.9986	0.0004	0.7290	0.1670	0.2942	0.0913	0.4110	0.6351	Yes
0.35	0.9986	0.0003	0.7556	0.1536	0.2661	0.0651	0.3882	0.6382	Yes
0.375	0.9987	0.0003	0.8096	0.1526	0.2505	0.0817	0.3730	0.6619	Yes
0.4	0.9986	0.0004	0.8357	0.1749	0.2135	0.0761	0.3311	0.6471	Yes
0.425	0.9986	0.0004	0.8441	0.1791	0.1879	0.0743	0.2977	0.6256	No
0.45	0.9986	0.0003	0.8299	0.1730	0.1866	0.0675	0.2969	0.6172	No
0.475	0.9986	0.0004	0.8905	0.1539	0.1765	0.0675	0.2869	0.6341	No
0.5	0.9986	0.0004	0.9207	0.1275	0.1538	0.0843	0.2525	0.6143	No

Table 8: Performance of the random forest model with the SMOTE (1:1 ratio) with various different cut-offs. The different cut-offs illustrate the precision versus recall combinations possible. Validation is performed by 20 repetitions of the validation set approach with a training size of 80%. The highest performance over the 20 repetitions in terms of precision, recall, F1 score and $F_{0.33}$ score are highlighted.

	Random forest + SMOTE							
Cut-off	**Accuracy**		**Precision**		**Recall**		**F1 Score**	**F0.33 Score**
	Mean	Std	Mean	Std	Mean	Std	Mean	Mean
0.0125	0.9751	0.0025	0.0573	0.0137	0.9358	0.0553	0.1075	0.0632
0.025	0.9846	0.0015	0.0888	0.0141	0.9113	0.0594	0.1615	0.0976
0.0375	0.9862	0.0014	0.0a954	0.0210	0.9049	0.0794	0.1716	0.1047
0.05	0.9895	0.0011	0.1228	0.0195	0.8867	0.0614	0.2149	0.1344
0.075	0.9918	0.0009	0.1455	0.0252	0.8312	0.0861	0.2467	0.1586
0.1	0.9932	0.0008	0.1696	0.0267	0.8148	0.0941	0.2796	0.1842
0.125	0.9943	0.0007	0.1939	0.0346	0.7816	0.0849	0.3092	0.2096
0.15	0.9953	0.0006	0.2204	0.0475	0.7377	0.1176	0.3373	0.2370
0.175	0.9959	0.0005	0.2404	0.0483	0.7167	0.1142	0.3584	0.2575
0.2	0.9963	0.0004	0.2679	0.0492	0.7219	0.1172	0.3886	0.2858
0.225	0.9967	0.0005	0.2843	0.0570	0.6892	0.1170	0.3996	0.3020
0.25	0.9969	0.0004	0.3020	0.0617	0.6824	0.1164	0.4159	0.3198
0.275	0.9973	0.0005	0.3422	0.0678	0.6273	0.0897	0.4373	0.3585
0.3	0.9975	0.0005	0.3568	0.0697	0.5929	0.0931	0.4413	0.3716
0.325	0.9977	0.0005	0.3793	0.0772	0.5692	0.0958	0.4507	0.3924
0.35	0.9977	0.0004	0.3845	0.0705	0.5502	0.0964	0.4472	0.3964
0.375	0.9979	0.0004	0.4115	0.0845	0.5215	0.1096	0.4539	0.4204

0.4	0.9980	0.0004	0.4314	0.0879	0.4875	0.1151	0.4521	0.4364
0.425	0.9981	0.0004	0.4499	0.0890	0.4795	0.1119	0.4583	0.4527
0.45	0.9983	0.0003	0.4834	0.0987	0.4613	0.1051	0.4648	0.4811
0.475	0.9983	0.0003	0.4969	0.1064	0.4416	0.1191	0.4608	0.4908
0.5	0.9984	0.0003	0.5188	0.1184	0.4152	0.1163	0.4543	0.5062

The results of these variants of the RF model can be found in Table 7 and Table 8. In both tables, we highlighted the models with the highest performance in terms of precision, recall, F1 score and $F_{0.33}$ score. Table 7 indicates the effect of changing the cut-off probability of the RF model without the SMOTE. As expected, lowering the cut-off point of the model results in a lower precision but a higher recall. By changing the cut-off, we can illustrate the level of precision versus the level of recall that can be achieved to management, for example, Table 7 suggests that we could create a model with a precision of around 60% and a recall of around 40% at a cut-off of 0.25. However, if higher precision is desired we could also create a model with a precision of around 73% and a recall of around 30%. In these examples, we do note that the high standard deviations of the performance of our models suggest that performance fluctuates significantly depending on the observations in the train or test set of our models, resulting in that the real-world performance of these models could significantly differ.

Table 8, with the results of the RF model with the SMOTE, shows similar findings to the RF model without the SMOTE. The results indicate that with the SMOTE, we cannot achieve a precision and $F_{0.33}$ score as high as without the technique. However, with the technique, it is possible to achieve higher levels of recall for some of the same levels of precision compared to the model without the SMOTE. We also see that with this variant of the technique (1:1 ratio without undersampling), we can achieve higher precision levels than with our initial RF model with the SMOTE.

At this point, it is clear that the RF model is the best performing model; it has the highest $F_{0.33}$ scores. Therefore, we apply hyperparameter tuning to the RF model. We will explore what maximum performance we can extract from this model. In Table 7, the column Grid Search indicates whether we investigate this setting of the RF model further by a grid search. We made this selection based in consultation with management. We selected the models with a $F_{0.33}$ score above 63.5 percent. These models have the desired level of precision but still an acceptable level of recall taking into account the other safety nets to identify clients with financial difficulties. We do not look further into RF models with the SMOTE applied as these have lower $F_{0.33}$ scores.

5.1.4. Hyperparameter tuning of the random forest model

In this section, we discuss the results of our grid search. We graphically represented the results in Figure 10 in the form of a scatterplot. We performed 50 repetitions in our grid search instead of the 20 repetitions used per setup for the results presented in Section 5.1.3 to get a more accurate mean performance measurement. For most combinations we checked, the entropy criterion outperforms the Gini index criterion. We also see that for the number of features used in a tree, 8 or 10 features per tree perform worse than using 4 or 5 features resulting from the log 2 or the square root of the number of features. The numeric results of the grid search can be found in Appendix D.

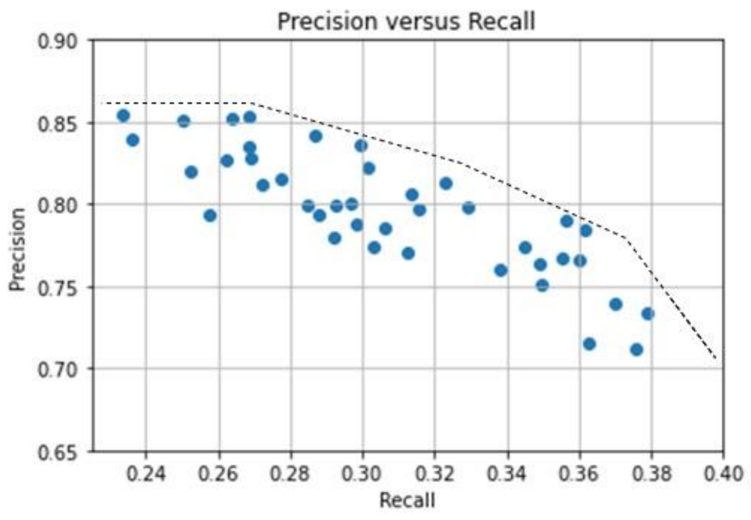

Figure 10: Precision versus recall performance of the models of the grid search reflected in a scatterplot. The dotted line reflects the optimal combination of precision versus recall achievable with the random forest model.

Figure 10 provides a graphical representation of the precision versus recall combinations that are possible, for example, we can calibrate our model such that we have a precision of +- 0.85 with a recall of +- 0.26. or a combination of a precision of +- 0.84 with a recall of +- 0.30. An approximation of the optimal trade-off of precision versus recall we can achieve with our model is reflected in the black dotted line in the scatterplot. The line can be seen as the efficient frontier of precision versus recall. It reflects how much precision we potentially can achieve with our current model given a level of recall. This information can be used to tweak the model if at a later stage when implementing our model a slightly other preference in precision versus recall is desired. Still, based on the current preferences of the management of the bank and the model that has the highest $F_{0.33}$ score, the configuration of the model that uses entropy as the criterion, 4 features per tree (from the log 2 rule) and a cut-off of 0.35 best suits the current wishes.

5.1.5. Impact of having more data

In this section, we discuss the results of what giving the new model more data does to the performance of the model. Generally, the performance of a ML model improves if given more data. We checked whether this is also the case for our new model. The result of this experiment is striking. We found that when we would only train the new model with 50% of the data, the precision was only 49% and the recall was only 14.3%. We suspect that this difference is such large because we go from using very few positive observations to almost too few observations. Nevertheless, this comparison does show that by adding more data, the performance of the model can be improved.

5.2. Comparison with the old model

When we compare the old and the new model we recommend, we can clearly see the strengths and weaknesses of the new model. The performance of the old model and the new model we recommend can be found in Table 9. The new model has significantly higher precision and $F_{0.33}$ score than the old model (84% and 71% versus 6% and 7%). The drawback of the new model is the lower recall score (30% versus 80%). Although the recall of the old model is an estimation and is likely lower in reality, the new model does perform less well for this metric. Meaning, that the new model results in more false negatives. Still, the F1 score of the new model is higher than the old model (43% versus 11%). From this observation, we can derive that the combined performance of precision and recall with both having equal weight is higher for the new model.

*Table 9: Performance of the old model versus the new model. *Recall of old model is an imperfect estimation and is likely lower in reality.*

Metric	Old model	New model
Accuracy	97.5%	99.9%
Precision	6.0%	83.6%
Specificity	97.6%	99.9%
Negative Predictive Value	99.9%	99.9%
Recall	79.5%*	30.0%
F1 Score	11.1%	43.4%
$F_{0.33}$ Score	6.6%	70.9%

Besides the direct comparison of the old and new model, we examined how the new model signals false negative observations at a later time. That is, we checked whether a client that needs a revision but is not marked will be marked by the model the next month. We found this was the case for 33 percent of the false negative observations. As 33 percent of the 70% of observations not identified is +- 23 percent of the total positive instances, we can state that the new model signals over 50 percent of the clients that it should signal if we also include signals that come 1 month later.

With respect to the $F_{0.33}$ score, we can conclude that the new model is an improvement over the old model. The improvement in terms of precision is especially very large. The weak point of the new model is the lower recall compared to the old model. The fundamental difference between the two models is that the new model is configured for high precision and the old model in configured for high recall. Precision and recall are competing performance metrics so a trade-off has to be made. The new configuration better suits the preference of management that precision is more important than recall. The signals of the new model will be correct much more often than the signals of the old model but the new model will signal fewer total clients that need a risk-based revision.

5.3. Important features

Getting insight into the important features for the prediction of risk-based revisions was a demand from management. As the new RBR model we recommend is a random forest model, we give this insight by presenting the feature important scores and the permutation feature important scores that can be derived from our random forest model. As earlier stated in Section 3.4.3, the feature importance score is computed on the mean and standard deviation of the accumulation of the impurity decrease within the trees of a random forest model. So a higher score of a feature reflects that the feature is better at separating the two classes in the trees compared to the other features. From this, we can conclude that a feature is more important. The permutation feature importance score is calculated by the effect on performance if all the values of one feature in the dataset are randomly shuffled over all the observations in the dataset. This method disconnects the relation between the actual feature value of an observation and the dependent variable as the feature value is replaced by a random value from the dataset. If performance is still about the same, the conclusion is that the feature has little importance. Also here a higher score of a feature indicates higher importance.

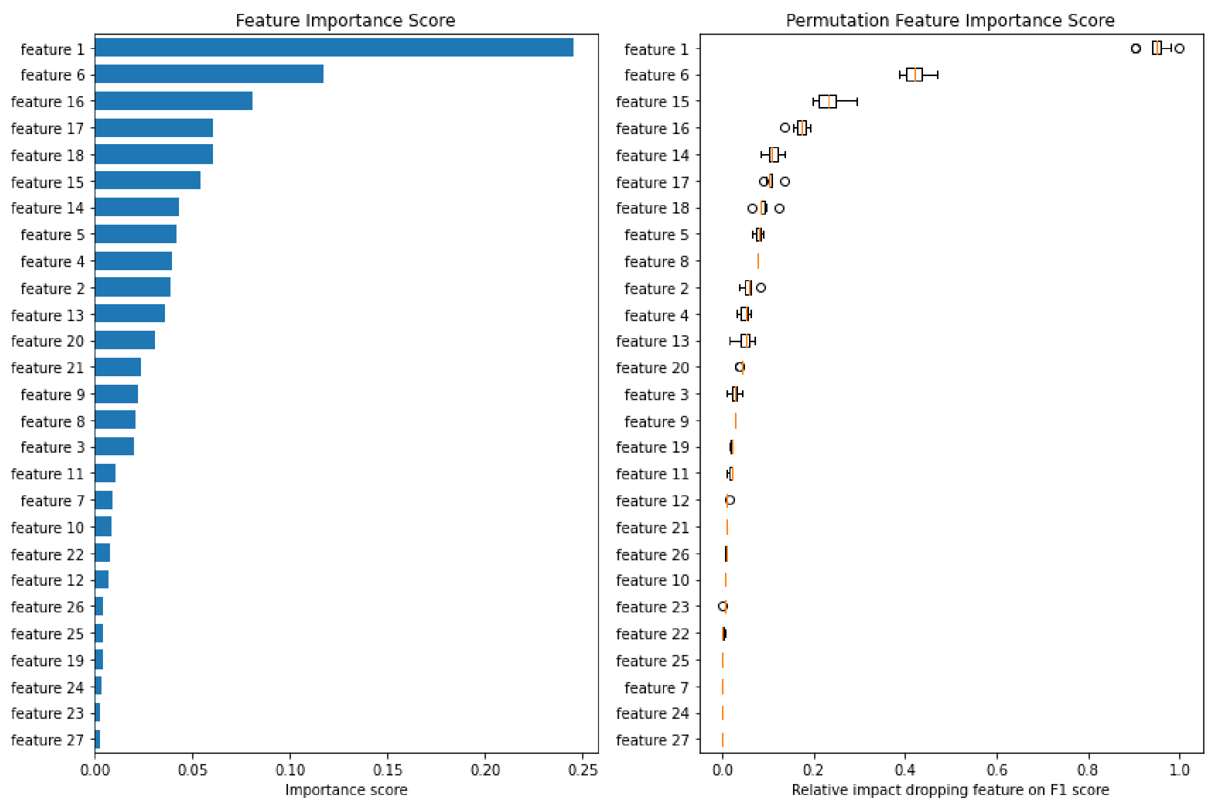

Figure 11: Feature importance scores and permutation feature important scores.

The feature important scores can be found in the left part of Figure 11. We anonymized the names of the features for confidentiality reasons. What we can disclose is that the PD ratio calculated by another model of the bank is a very important predictor in our model. The combination of the PD ratio itself and derived predictors that reflect the PD ratio development over the last months have a combined score of over 50 percent. From this, we can conclude that the PD ratio is heavily used in our model to make a distinguishment between clients that need a risk-based revision and clients that do not. Clients with high PD scores more often need a revision. However, it is not a one-to-one relation, we found that a high PD score does not have to imply the need for a risk-based revision, this is also the main reason that the current RBR model does not function properly.

In Section 2.1, we identified that the current RBR model does not use predictors related to the payment traffic of the client. It was not possible to add it to the current model because when a client has a loan at the Volksbank but its account with payment traffic at another bank this data is not available. However, we found that adding relatively basic payment traffic data that is available within the Volksbank helps to distinguish between clients that need a risk-based revision and clients that do not. We see that clients more often need a revision when there is less money going in and out of their current account.

What is left to be said is that we found that a single feature does not solely predict on itself the need for a risk-based revision. A high PD ratio does not automatically mean that a revision is needed. The key is to combine multiple factors to determine whether a revision is needed.

6. Conclusion & Recommendations

In this chapter, we present our conclusions in Section 6.1 and our recommendations to the Volksbank in Section 6.2. Subsequently, in Section 6.3, we state limitations of our study. Finally, we present topics for further research in Section 6.4.

6.1. Conclusions

Currently, the risk-based revision (RBR) process at the Volksbank does not function properly. A revision is a reassessment of the credit risk and the associated customer strategy of/for a client. In the RBR process, clients that appear to go into default are revised. During a revision, it is determined whether measures could be taken that could prevent the client from defaulting. These measures are called forbearance measures. Not all clients are revised, a risk-based procedure is used. The current RBR model that uses trigger conditions to signal clients that need a revision has very low precision, the model produces a lot of false positive signals. Therefore the main goal of this study was to create a new RBR model with better performance than the current model using machine learning. We posed the following main research question:

"*How can the risk-based revision model be improved using machine learning?*"

To answer this question, we analyzed the current RBR process/model, researched methods that we could use to create a new model and thereafter constructed and tested various new models. We established that the current RBR model indeed performs poorly, achieving only a precision of 6% and a $F_{0.33}$ score of 7%. We could not identify the exact recall performance because there currently does not exist a hard definition of which clients need to be revised and which do not in the bank. However, we approximated that the current model identifies 80% of the clients that would need a revision with the note that we expect this performance to be lower in reality. We conclude that the old model can be characterized as throwing enough mud at the wall and observing that some of it will stick.

After consultation of management, we conclude that a model with fewer false positives (clients that do not need a revision that are signaled by the model) is preferred over a model with fewer false negatives (clients that should have been signaled by the model but were not). Other processes in the bank, such as arrears management and UTP triggers, could be safety nets that reduce the costs of a false negative prediction of the model. Therefore, we created a new RBR model with a focus on increasing precision and use the $F_{0.33}$ metric to determine which new type of RBR model performs best.

For the construction of a new model, we tested various models and techniques. Due to regulatory requirements, we were not allowed to implement black box type models, such as support vector machines or neural networks. Out of the type of models that were available and tested, we found that a random forest model performs best. With this type of model, we can achieve a precision that is higher than 75 percent, which is a very large increase compared to the current model. However, we found that added precision comes at a cost because we find that with a random forest model, we cannot achieve the same recall score as the current model. This all means that we can make a model of which its signals will be correct much more often than the signals of the old model but the new model will signal fewer total clients that need a risk-based revision.

The model that we recommend to management achieves a $F_{0.33}$ score of 71%, a precision of 83.6% and a recall of 30.0%. An interesting insight we found is that recall increases to 53% if the signal from the RBR model can also come 1 month later from the next monthly run of the model. Based on the $F_{0.33}$ score, the new model is a significant improvement over the current model but it is not perfect as the number of false negatives of this model is considerable. We are able to reduce the number of false negatives of the model but this has the effect that the number of false positives will increase. For the insight into what combinations of precision versus recall can be achieved we refer back to Section 5.1.3 and Section 5.1.4. In the new model, predictors related to the PD and payment traffic in the current account of the client are most important, clients with a high PD and low payment traffic in the current account more often need a revision.

We identified 4 reasons that individually or combined could have resulted in the new model not being perfect:

- Inconsistency in the evaluation of the shortlist produced the RBR model
- Small size of the dataset
- No appropriate features were defined that grasp the available patterns in the data
- Lack of patterns in the data

The first potential reason is inconsistency in the evaluation of the shortlist produced by the current RBR model. Currently, an employee manually verifies the need for a revision by checking factors such as the use of the limit of the current account or payment traffic in the account of the client in the various systems the bank has to monitor its clients. As this is a manual process and currently no pre-set criteria of the clients that need a revision exist, this process is susceptible to randomness and inconsistent decision making. Also, the decision-making of an employee could be affected by the limited staff capacity for risk-based revisions of the bank. If the employee already confirmed that some clients on the shortlist need to get a revision, he might be unconsciously tempted to not confirm more cases as capacity does not allow these clients to be reviewed. We use the evaluation of the shortlist as input for our model to learn from, so our model could be affected by inconsistent decision-making.

Secondly, the size of our dataset is limited, especially the number of instances that are labeled as '*revision needed'* which is only 97. When also taking into account that we use 20% of our instances to test our models and that due to the characteristic of corporate clients that every client is slightly different, the limited number of examples for the algorithm to learn from could be too limited.

Next, it could be that we did not define the appropriate features that could capture the patterns in the data that is available to us. Potentially, using feature engineering, other (more advanced) features could be constructed that could grasp the patterns in the outcome whether a risk-based revision is needed.

The last reason could be the lack of patterns in the data. It is possible that the dataset does not contain sufficiently strong patterns. Other factors currently not in our dataset could also influence whether a revision is needed or not. It is also the question of whether there are patterns that could be captured by any set of predictors or whether the outcome is intrinsically noisy, for example, unpredictable future macroeconomic factors are an example of patterns that cannot be captured by predictors

In the old model, the features derived from the PD are mostly used to determine whether a client needs a risk-based revision. In the new model we created, the PD ratio is also an important driver. Besides the PD ratio, the new model also makes use of the payment traffic data of the clients. We found that making use of the relative basic payment traffic data that is available within the Volksbank helps to distinguish between clients that need a risk-based revision and clients that do not in a new model. Potentially, more advanced payment traffic features that also take into account whether the client has his payment traffic at the Volksbank or another bank could further increase the performance of the model.

Summarizing the findings of this study, the risk-based revision model can be improved by implementing a random forest model. The configuration of the model we suggest drastically improves the $F_{0.33}$ score and the precision of the model but the recall is reduced. However, the higher precision is preferred over the lower recall by management. Predictors related to the PD and payment traffic in the current account of the client are most important in the model we suggest.

6.2. Recommendations

In this section, we present our recommendations to the Volksbank based on the findings of our study. We will list these recommendations below after we will elaborate on and explain each recommendation.

- Implement the new RBR model.
- Update the model after some years when more data is available.
- Perform follow-up research on in which type of situation clients should get a revision and which not.
- Quantify the costs of false positives and false negatives.
- Examine how payment traffic can more elaborately be implemented in the model.
- Investigate if the underlying drivers of the PD can be used to further improve the model.

The first recommendation we have is to implement the new model we created. The new model better suits the preference of management that fewer false positives (clients that do not need a revision that are signaled by the model) are more important than fewer false negatives (clients that should have been signaled by the model but were not). If during the implementation process of the new model this preference slightly changes, which could be the case because of numerous requirements when implementing a new model in a bank such as approval from the risk department or several committees of the bank, we provided the tools in the form of internal documentation about our model to make these changes and the insight into what effect a different calibration of the model would have in terms of precision versus recall in the form of presentations and this report. Besides, for effectively implementing and maintaining the model in the future, employees with knowledge of these topics are needed. Also, when there would be disagreement on how to configure the model in the future, quantifying the costs of a false positive and a false negative could be a solution, although this is a very difficult and extensive task (see Section 2.3 for a full elaboration on this topic).

Secondly, we recommend updating the model after some time when more data is available. Currently, there is a limited number of cases in which clients received a risk-based revision. resulting in the ML model having limited examples to learn from. If after some time, more examples of risk-based revisions are collected, these added observations could be used to retrain the model. This will very likely create a better-performing model but the quantity of additional data that is needed to reach a certain performance level or whether that level can be reached by adding more data we cannot predict. We confirmed by an experiment of us, see Section 5.1.5, that the performance of the model can improve when more data is used. In that experiment, we only used 50% of the available data. The model produced in this experiment only achieved a precision of 49% compared to 84% of the model with 100% of the available data, thereby proving that adding more data can help to increase the performance of the model. We found that training the current model takes approximately 1 to 2 minutes. Therefore time is not a limiting factor when updating the model. An issue with updating the model is that old data may be less representative because of changing circumstances. Representativeness of data should be taken into account when updating the model with new data and the question should be asked whether the old data still reflects the current process and circumstances.

Next, we would recommend follow-up research on the topic of risk-based revisions. Currently, it is not exactly clear which clients should be revised and which not. Therefore, we were also not able to accurately identify false negative instances in our study. We recommend defining and documenting which clients should be revised, thereby creating a clear risk-based revision policy. Also, we advise research into what happens after a risk-based revision. It is interesting or even fundamental to study how often the effect of a risk-based revision is successful and which forbearance measures are effective. In this study, we simply assumed that forbearance measures have a significant decreasing effect on the probability of default. Potentially, the effect of a forbearance measure could be quantified by a reduction in the probability of default.

Subsequently, we identified in this research that features related to payment traffic are useful for predicting risk-based revisions. We currently implemented features directly available from the data warehouse from the bank. However, these features give limited insight into clients when a client has his payment traffic at another bank. Therefore, we recommend exploring whether it is possible to create new features that incorporate this aspect of the payment traffic of a client.

Lastly, we recommend investigating the underlying drivers of the PD model as potential new features. We tried to implement these drivers ourselves in our dataset but we ran into various practical constraints and therefore decided not to implement these due to time constraints. However, we suspect that after the time investment needed to gather this data, the underlying drivers have to potential to increase the performance of our model.

6.3. Limitations

Some limitations we encountered during this study restricted the methods we could apply and the results we could achieve. These were the availability of data, a restriction on using black box ML models, a missing policy from the bank and data quality. In this section, we go into detail on these limitations and if needed, what action we took to still perform this study.

The first limitation in this study is the availability of data and especially the number of examples of risk-based revisions. This number is limited which reduces the effectiveness of machine learning. Another aspect of the availability of data is limitations in the data warehouses of the bank. Bundling all data sources together turned out to be a challenge and one that could not always be overcome, resulting that we were not always able to extract all the data we preferred to have.

Another limitation in this study is the restriction of using black box machine learning models. Due to regulatory requirements, models like support vector machines and neural networks were not available for us to implement, although these models could have the potential to make better performance than the random forest model we currently recommend. Generally, black box models have a better ability to capture non-linearity and interaction between features.

Next, we have the limitation that there currently is no concrete policy on which clients should receive a risk-based revision and which not. This severely limited us in identifying false negative instances from our dataset. In the end, it also resulted in that we not included false negative instances as positives to train our model with.

The last limitation of this study we mention is data quality issues. The data we have available to us contains missing data points. To still be able to implement ML algorithms, we had to perform several data cleaning operations. These operations could have impacted this study.

6.4. Further research

We identified several topics for further research of which some we already mentioned in our recommendations. The first topic of further research is increasing the number of relevant features of the data. Adding more macroeconomic factors or underlying drivers of features such as the PD could be very interesting to study. Also, feature engineering could be used to extract more information from the current dataset.

Secondly, implementing a complete subset selection technique could be worthwhile. In this study, we used the subset of features that at first grasp contains all the features that have some degree of importance for predicting revisions. Examining other subsets of features has the potential to increase the performance of the model.

Next, despite models like SVMs and neural networks are not allowed in the context of this research, it could be studied what kind of performance could be achieved with these kinds of models. This could indicate a benchmark of the performance that could be achieved with machine learning with the currently available data.

Subsequently, a topic of further research could be to quantify the costs of false positives and false negatives. The costs of false positives could be rather easily identified by establishing the average costs of checking the output of the RBR model. The challenge here is to establish the costs associated with a false negative which could potentially be modeled by multiplying the average decrease in default probability from forbearance measures with the average cost of default. These two factors will be challenging to determine and at least questionable assumptions will be needed. In our study, we assumed that forbearance measures have a significant positive effect on the PD. However, proving this can be tricky, especially given the limited number of data points. Besides, it is questionable if averages could be used as the cost of a false negative is significantly higher when the client has an exposure of 1 million than with an exposure of 0.1 million, resulting in likely large standard deviations of the average costs. Using an average would not correctly display the trade-off of the costs of false positives and false negatives and therefore confidence intervals should be used. Further research should show whether these challenges could be overcome.

Then, when we performed lasso regression and ridge regression, we standardized the numerical features of our dataset as the scale of our variables has an effect on the results of these type of models. Unexpectedly, we found that standardizing did not increase the performance of our models. It could be interesting to establish the reason for this result. Also, other methods of standardization or transformation could be performed. Examples of these are min-max standardizing or log transformations.

Lastly, in our discussion of techniques for imbalanced datasets, we covered cost-sensitive learning (assigning different misclassification costs for false positives or false negatives). We decided due to time constraints to not test this method. However, it could be interesting to observe the effect of implementing this technique. It could be that cost-sensitive learning is a more effective technique than using an alternative cut-off or the synthetic minority oversampling technique.

References

Akinsola, J. E. (2017). Supervised Machine Learning Algorithms: Classification and Comparison. *International Journal of Computer Trends and Technology (IJCTT), 48*, 128-138.

Alpaydin, E. (2020). *Introduction To Machine Learning (4th ed.).* MIT Press.

An, J. (2020, July 27). Retrieved from medium.com: https://medium.com/swlh/how-to-remember-all-these-classification-concepts-forever-761c065be33

Batista, G. E., & Monard, M. C. (2003). An analysis of four missing data treatment methods for supervised learning. *Applied Artificial intelligence, 17*, 519-533.

Berry, M., & Linoff, G. (1999). *Mastering Data Mining: The Art and Science of Customer Relationship Managment.* New York: John Wiley & Sons, Inc.

Biau, G., & Scornet, E. (2016). *A random forest guided tour.*

Cessie, S., & van Houwelingen, J. C. (1992). Ridge estimators in logistic regression. *Applied Statistics, 41*, 191-201.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artifical Intelligence Research, 16*, 321-357.

Chinchor, N. (1992). MUC-4 Evaluation Metrics. *in Proc. of the Fourth Message Understanding Conference*, 22-29.

Daoud, J. I. (2017). Multicollinearity and Regression Analysis. *Journal of physics, 949.*

Duffy, D. E., & Santner, T. J. (1989). On the small sample properties of norm-resticted maximum likelihood estimators for logistic regression models. *Communications in Statistics - Theory and Methods* , 959-980.

ECB. (2017). *Guidance to banks on non-performing loans.* ECB.

European Banking Authority. (2020). *Final Report on Guidelines on loan origination and monitoring.*

European Banking Authority. (2021). *EBA Discussion Paper On Machine Learning For IRB Models.*

Feurer, M., & Hutter, F. (2019). Hyperparameter Optimization. In F. Hutter, L. Kotthoff, & J. Vanschoren, *Automated Machine Learning* (pp. 3-33). Springer.

Fox, J., & Monette, G. (1992). Generalized Collinearity Diagnostics. *Journal of the American Statistical Association, 87*, 178-183.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning (2nd ed.).* Springer.

Hippel, P. T. (2012). Biases in SSS 12.0 Missing Value Analysis. *The American Statistician*, 160-164.

Hoogendoorn, M., & Funk, B. (2017). *Machine Learning for the Quantified Self.* Springer.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning (2nd ed.).* Springer.

Juárez-Orozco, L. E., Manzanera, O. M., Nesterov, S. V., & Kajander, S. A. (2018). The machine learning horizon in cardiac hybrid imaging. *European Journal of Hybrid Imaging*.

Khanzode, C. A., & Sarode, R. D. (2020). Advantages and disadvantages of artificial intelligence and machine learning: a literature review. *International Joural of Library & Information Science, 9(1)*, pp. 30-36.

Kotthoff, L. (2016). Algorithm Selection for Combinatorial Search Problems: A Survey. In C. Bessiere, L. De Raedt, L. Kotthoff, S. Nijssen, B. O'Sullivan, & D. Pedreschi, *Data Mining and Constraint Programming. Lecture Notes in Computer Science.* Cham: Springer.

Law, A. M. (2015). *Simulation Modeling (5th ed.).* McGraw-Hill Education.

Lee, I., & Shin, Y. (2019). *Machine learning for enterprises: Applications, algorithm selection, and challenges.* Elsevier Inc.

Lipton, Z. C., Elkan, C., & Naryanaswamy, B. (2014). Optimal Thresholding of Classifiers to Maximize F1 measure. *Machine Learning and Knowledge Discovery in Databases*, 225-239.

Little, R. J., & Rubin, D. B. (2002). *Statistical Analysis with Missing Data (2th ed.).* Wiley.

Loeffel, P.-X. (2017). *Adaptive machine learning algorithms for data streams subject to concept drifts.* HAL open science.

Longadge, R., Dongre, S. S., & Malik, L. (2013). Class Imbalance Problem in Data Mining: Review. *International Journal of Computer Science and Network, 2.*

Mitchell, T. (1997). *Machine Learning.* McGraw Hill.

Montgomery, D. C., Peck, E. A., & Vinning, G. G. (2021). *Introduction to Linear Regression Analysis (6th ed.).* Wiley.

Nanni, L., Fantozzi, C., & Lazzarini, N. (2015). Coupling different methods for overcoming the class imbalance problem. *Elsevier, 158*, 48-61.

Orlenko, A., & Moore, J. H. (2021). *A comparison of methods for interpreting random forest models of genetic association in the presence of non-additive interactions.* Springer Nature.

Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012). How Many Trees in a Random Forest. *Machine Learning and Data Mining in Pattern Recognition*, 154-168.

Pereira, J. M., Basto, M., & Silva, A. F. (2015). *The logistic lasso and ridge regression in predicting corporate failure.* Elsevier.

Santos, M. S., Abreu, P. H., Soares, J. P., & Araujo, H. J. (2018). Cross-Validation for Imbalanced Datasets: Avoiding Overoptimistic and Overfitting Approaches. *IEEE Computational Intelligence Magazine, 13.*

Sarkar, S., Weyde, T., Garcez, A. d., Slabaugh, G., Dragicevic, S., & Percy, C. (2016). *Accuracy and Interpretability Trade-offs in Machine Learning Applied to Safer Gambling.*

Song, Y.-y., & Lu, Y. (2015). Decision tree methods: applications for classifcation and prediction. *Shanghai Arch Psychiatry, 27(2)*, 130-135.

Stoltzfus, J. C. (2011). *Logistic Regression: A Brief Primer.*

Taddy, M. (2019). *Business Data Science: Combining Machine Learning and Economics to Optimize, Automate, and Accelerate Business Decisions.* McGraw Hill Professional.

Vabalas, A., Gowen, E., Piliakoff, E., & Casson, A. J. (2019). *Machine learning algorithm validation with a limited sample size.* PLOS ONE.

Valdes, G., Luna, J. M., Eaton, E., Simone II, C. B., Ungar, L. H., & Solberg, T. D. (2017). *MediBoost: a Patient Stratification Tool for Interpretable Decision Making in the Era of Precision Medicine.* Nature, 6.

Yang, Y. J., & Bang, C. S. (2019). *Application of artificial intelligence in gastroenterology.* World Journal of Gastroenterology.

Zhu, W., Zeng, N., & Wang, N. (2010). *Sensitivity, Specificity, Accuracy, Associated Confidence Interval and ROC Analysis with Practical SAS ® Implementations.*

Appendices

Appendix A: Data cleaning operations

In this appendix, we describe some issues in our dataset for which we performed data cleaning operations that are worthwhile to mention.

Issue 1

The observations in our dataset are monthly time series of a client. When multiple types of revisions are performed in one month, we have multiple observations of one month in our dataset, which is undesirable.

Solution

When we have multiple observations of one month of a client due to multiple revisions, we select the observation with the revision type that is relevant to us. The other observation we drop from the dataset. Thereby, preventing duplicates.

Issue 2

New clients often have some empty data points in their first occurrence in our dataset.

Solution

Using NOCB, we use the data points of the client in the next month to fill the empty data points in the first month.

Issue 3

Some features in our dataset have as default empty values (a None value), which can give trouble in some operations or ML models.

Solution

We replace the empty values with logical values that are more fitting. In most cases zero.

Issue 4

A risk-based revision is not always performed in the same month as when a client is signaled by the RBR model. This creates trouble with matching a performed revision with a signal of the RBR model.

Solution

When a RBR revision is performed, we look back 3 months to check if the RBR model signaled the client. If this is the case, we say that those signals are true positives.

Issue 5

Disturbing data after a true positive. The observations after a true positive are very similar to the true positive observation because they are of the same client but one or two months later. However, these observations are not true positives. This will make it hard for a ML algorithm to distinguish between the observations.

Solution

After a true positive followed by a not true positive, we drop all future instances of that client.

Issue 6

We have clients in our dataset that are in default at the first observation we have of them. These are not clients the RBR model is applied to.

Solution

We drop those clients from the dataset

Issue 7

We have clients in our dataset that are already undergoing a revision process at the first observation we have of them. These are not clients the RBR model is applied to.

Solution

We drop those clients from the dataset.

Appendix B: Results Logistic Regression

In this appendix, we highlighted the models with the highest performance in terms of precision, recall, F1 score and $F_{0.33}$ score per type of model.

Table 10: Performance of the Lasso regression model with varying lambdas and with the SMOTE + undersampling applied (SMOTE to a 1:10 ratio and undersampling to a 1:2 ratio) without standardizing numerical features of the dataset. Validation is performed using 5-fold cross-validation. The highest performance in terms of precision, recall, F1 score and $F_{0.33}$ score are highlighted.

Model	**Lambda**	**Accuracy**		**Precision**		**Recall**		**F1 Score**	**F0.33 Score**
		Mean	Std	Mean	Std	Mean	Std	Mean	Mean
Lasso regression + SMOTE	0.01	0.9776	0.0019	0.0524	0.0047	0.7811	0.1141	0.0982	0.0578
Lasso regression + SMOTE	0.1	0.9822	0.0019	0.0659	0.0081	0.7816	0.0930	0.1214	0.0725
Lasso regression + SMOTE	0.5	0.9856	0.0014	0.0783	0.0071	0.7611	0.0822	0.1419	0.0860
Lasso regression + SMOTE	1	0.9863	0.0021	0.0849	0.0093	0.7832	0.0774	0.1530	0.0932
Lasso regression + SMOTE	2	0.9863	0.0023	0.0840	0.0087	0.7721	0.0863	0.1511	0.0922
Lasso regression + SMOTE	3	0.9861	0.0026	0.0850	0.0148	0.7826	0.1023	0.1527	0.0933
Lasso regression + SMOTE	4	0.9863	0.0017	0.0853	0.0105	0.7926	0.0937	0.1537	0.0936
Lasso regression + SMOTE	5	0.9867	0.0016	0.0864	0.0129	0.7721	0.0960	0.1553	0.0948
Lasso regression + SMOTE	6	0.9862	0.0018	0.0812	0.0099	0.7516	0.0718	0.1464	0.0892
Lasso regression + SMOTE	7	0.9869	0.0021	0.0890	0.0114	0.7826	0.0920	0.1594	0.0977
Lasso regression + SMOTE	8	0.9860	0.0018	0.0807	0.0113	0.7516	0.0718	0.1455	0.0886
Lasso regression + SMOTE	9	0.9857	0.0020	0.0806	0.0131	0.7716	0.1034	0.1457	0.0886
Lasso regression + SMOTE	10	0.9864	0.0016	0.0854	0.0101	0.7832	0.0774	0.1537	0.0937
Lasso regression + SMOTE	25	0.9864	0.0011	0.0802	0.0106	0.7300	0.0880	0.1445	0.0881
Lasso regression + SMOTE	50	0.9862	0.0014	0.0761	0.0082	0.7000	0.0644	0.1372	0.0835
Lasso regression + SMOTE	100	0.9867	0.0012	0.0770	0.0059	0.6795	0.0447	0.1383	0.0845
Lasso regression + SMOTE	1000	0.9865	0.0014	0.0773	0.0077	0.6984	0.1063	0.1390	0.0848

Table 11: Performance of the Lasso regression model with varying lambdas and with the SMOTE + undersampling applied (SMOTE to a 1:10 ratio and undersampling to a 1:2 ratio) with standardizing numerical features of the dataset. Validation is performed using 5-fold cross-validation. The highest performance in terms of precision, recall, F1 score and $F_{0.33}$ score are highlighted.

Model	Lambda	Accuracy		Precision		Recall		F1 Score	F0.33 Score
		Mean	Std	Mean	Std	Mean	Std	Mean	Mean
Lasso regression + SMOTE	0.01	0.9754	0.0015	0.0560	0.0022	0.9279	0.0416	0.1056	0.0618
Lasso regression + SMOTE	0.1	0.9795	0.0011	0.0627	0.0072	0.8647	0.0926	0.1169	0.0691
Lasso regression + SMOTE	0.5	0.9802	0.0017	0.0642	0.0094	0.8542	0.1132	0.1194	0.0708
Lasso regression + SMOTE	1	0.9807	0.0019	0.0645	0.0086	0.8337	0.1034	0.1196	0.0710
Lasso regression + SMOTE	2	0.9810	0.0012	0.0644	0.0096	0.8237	0.1237	0.1195	0.0710
Lasso regression + SMOTE	3	0.9808	0.0012	0.0653	0.0081	0.8442	0.1063	0.1211	0.0719
Lasso regression + SMOTE	4	0.9814	0.0010	0.0678	0.0100	0.8542	0.1132	0.1256	0.0747
Lasso regression + SMOTE	5	0.9802	0.0017	0.0625	0.0093	0.8332	0.1316	0.1163	0.0689
Lasso regression + SMOTE	6	0.9806	0.0011	0.0653	0.0048	0.8553	0.0619	0.1213	0.0719
Lasso regression + SMOTE	7	0.9814	0.0017	0.0672	0.0069	0.8442	0.0887	0.1244	0.0740
Lasso regression + SMOTE	8	0.9809	0.0020	0.0652	0.0097	0.8342	0.1122	0.1208	0.0718
Lasso regression + SMOTE	9	0.9813	0.0013	0.0662	0.0116	0.8337	0.1274	0.1226	0.0729
Lasso regression + SMOTE	10	0.9812	0.0010	0.0649	0.0059	0.8237	0.0730	0.1204	0.0715
Lasso regression + SMOTE	25	0.9807	0.0019	0.0636	0.0080	0.8242	0.0984	0.1181	0.0701
Lasso regression + SMOTE	50	0.9803	0.0022	0.0642	0.0105	0.8442	0.1109	0.1193	0.0708
Lasso regression + SMOTE	100	0.9812	0.0011	0.0673	0.0071	0.8547	0.0704	0.1247	0.0741
Lasso regression + SMOTE	1000	0.9811	0.0013	0.0670	0.0061	0.8547	0.0704	0.1242	0.0738

Table 12: Performance of the Ridge regression model with varying lambdas and with the SMOTE + undersampling applied (SMOTE to a 1:10 ratio and undersampling to a 1:2 ratio) without standardizing numerical features of the dataset. Validation is performed using 5-fold cross-validation. The highest performance in terms of precision, recall, F1 score and $F_{0.33}$ score are highlighted.

Model	Lambda	Accuracy		Precision		Recall		F1 Score	F0.33 Score
		Mean	Std	Mean	Std	Mean	Std	Mean	Mean
Ridge regression + SMOTE	0.01	0.9683	0.0082	0.0398	0.0137	0.7611	0.1289	0.0754	0.0440
Ridge regression + SMOTE	0.1	0.9676	0.0077	0.0382	0.0077	0.7826	0.1023	0.0727	0.0422
Ridge regression + SMOTE	0.5	0.9608	0.0092	0.0316	0.0094	0.7611	0.1654	0.0605	0.0349
Ridge regression + SMOTE	1	0.9688	0.0056	0.0353	0.0051	0.7016	0.0330	0.0671	0.0390
Ridge regression + SMOTE	2	0.9636	0.0049	0.0306	0.0048	0.7121	0.0477	0.0586	0.0338
Ridge regression + SMOTE	3	0.9689	0.0046	0.0348	0.0048	0.6911	0.0548	0.0661	0.0384
Ridge regression + SMOTE	4	0.9603	0.0052	0.0290	0.0033	0.7426	0.0627	0.0558	0.0321
Ridge regression + SMOTE	5	0.9662	0.0095	0.0361	0.0095	0.7432	0.0401	0.0687	0.0399
Ridge regression + SMOTE	6	0.9659	0.0083	0.0319	0.0054	0.6805	0.0859	0.0608	0.0353
Ridge regression + SMOTE	7	0.9685	0.0076	0.0348	0.0055	0.6911	0.0949	0.0661	0.0385
Ridge regression + SMOTE	8	0.9669	0.0088	0.0343	0.0055	0.7121	0.0910	0.0653	0.0380
Ridge regression + SMOTE	9	0.9662	0.0062	0.0345	0.0053	0.7432	0.0401	0.0659	0.0382
Ridge regression + SMOTE	10	0.9664	0.0070	0.0345	0.0059	0.7332	0.0536	0.0658	0.0382
Ridge regression + SMOTE	25	0.9696	0.0065	0.0389	0.0091	0.7521	0.1225	0.0738	0.0429
Ridge regression + SMOTE	50	0.9648	0.0065	0.0352	0.0106	0.7716	0.1559	0.0673	0.0389
Ridge regression + SMOTE	100	0.9670	0.0051	0.0354	0.0082	0.7421	0.1243	0.0674	0.0391
Ridge regression + SMOTE	1000	0.9619	0.0091	0.0333	0.0120	0.7621	0.1194	0.0637	0.0368

Table 13: Performance of the Ridge regression model with varying lambdas and with the SMOTE + undersampling applied (SMOTE to a 1:10 ratio and undersampling to a 1:2 ratio) with standardizing numerical features of the dataset. Validation is performed using 5-fold cross-validation. The highest performance in terms of precision, recall, F1 score and $F_{0.33}$ score are highlighted.

Model	Lambda	Accuracy		Precision		Recall		F1 Score	F0.33 Score
		Mean	Std	Mean	Std	Mean	Std	Mean	Mean
Ridge regression + SMOTE	0.01	0.9775	0.0016	0.0579	0.0033	0.8763	0.0512	0.1085	0.0638
Ridge regression + SMOTE	0.1	0.9783	0.0019	0.0583	0.0043	0.8453	0.0716	0.1090	0.0643
Ridge regression + SMOTE	0.5	0.9787	0.0019	0.0592	0.0042	0.8453	0.0716	0.1106	0.0653
Ridge regression + SMOTE	1	0.9793	0.0016	0.0603	0.0056	0.8347	0.0751	0.1124	0.0664
Ridge regression + SMOTE	2	0.9798	0.0019	0.0625	0.0057	0.8453	0.0716	0.1163	0.0689
Ridge regression + SMOTE	3	0.9801	0.0015	0.0624	0.0060	0.8353	0.0862	0.1160	0.0687
Ridge regression + SMOTE	4	0.9794	0.0019	0.0600	0.0078	0.8247	0.0879	0.1117	0.0661
Ridge regression + SMOTE	5	0.9799	0.0019	0.0620	0.0072	0.8347	0.0751	0.1154	0.0683
Ridge regression + SMOTE	6	0.9800	0.0020	0.0630	0.0061	0.8453	0.0716	0.1172	0.0695
Ridge regression + SMOTE	7	0.9799	0.0022	0.0614	0.0062	0.8247	0.0879	0.1142	0.0676
Ridge regression + SMOTE	8	0.9801	0.0019	0.0617	0.0050	0.8247	0.0814	0.1148	0.0680
Ridge regression + SMOTE	9	0.9802	0.0021	0.0632	0.0078	0.8342	0.0611	0.1174	0.0697
Ridge regression + SMOTE	10	0.9802	0.0022	0.0603	0.0064	0.7932	0.0854	0.1121	0.0665
Ridge regression + SMOTE	25	0.9802	0.0021	0.0636	0.0056	0.8453	0.0716	0.1183	0.0701
Ridge regression + SMOTE	50	0.9810	0.0017	0.0653	0.0050	0.8353	0.0862	0.1210	0.0719
Ridge regression + SMOTE	100	0.9804	0.0023	0.0637	0.0077	0.8347	0.0751	0.1183	0.0702
Ridge regression + SMOTE	1000	0.9807	0.0018	0.0643	0.0044	0.8353	0.0795	0.1192	0.0708

Table 14: Performance of logistic regression model using subset of features of the available data with the SMOTE + undersampling applied (SMOTE to a 1:10 ratio and undersampling to a 1:2 ratio). The subsets are determined using the forward stepwise selection technique, 50% of the data was used for the feature selection with 3 fold CV and the other 50% of the data was used for performance testing using 3-fold CV. The highest performance in terms of precision, recall, F1 score and $F_{0.33}$ score are highlighted.

Model	**Number of features**	**Accuracy**		**Precision**		**Recall**		**F1 Score**	**F0.33 Score**
		Mean	Std	Mean	Std	Mean	Std	Mean	Mean
Logistic regression forward stepwise	1	0.9705	0.0027	0.0493	0.0039	0.9412	0.0480	0.0936	0.0544
Logistic regression forward stepwise	2	0.9767	0.0029	0.0645	0.0080	0.9804	0.0277	0.1210	0.0712
Logistic regression forward stepwise	3	0.9775	0.0025	0.0616	0.0062	0.9007	0.0260	0.1151	0.0679
Logistic regression forward stepwise	4	0.9823	0.0020	0.0784	0.0060	0.9216	0.0555	0.1443	0.0863
Logistic regression forward stepwise	5	0.9821	0.0012	0.0773	0.0030	0.9216	0.0555	0.1425	0.0851
Logistic regression forward stepwise	6	0.9818	0.0018	0.0763	0.0054	0.9216	0.0555	0.1408	0.0840
Logistic regression forward stepwise	7	0.9830	0.0021	0.0831	0.0084	0.9412	0.0480	0.1525	0.0914
Logistic regression forward stepwise	8	0.9821	0.0012	0.0771	0.0031	0.9216	0.0555	0.1423	0.0849
Logistic regression forward stepwise	9	0.9824	0.0014	0.0769	0.0032	0.9020	0.0734	0.1416	0.0846
Logistic regression forward stepwise	10	0.9835	0.0011	0.0817	0.0036	0.9020	0.0734	0.1497	0.0899
Logistic regression forward stepwise	11	0.9838	0.0018	0.0835	0.0065	0.9020	0.0734	0.1527	0.0919
Logistic regression forward stepwise	12	0.9835	0.0017	0.0785	0.0019	0.8627	0.1000	0.1437	0.0864
Logistic regression forward stepwise	13	0.9837	0.0017	0.0813	0.0042	0.8824	0.0832	0.1487	0.0894
Logistic regression forward stepwise	14	0.9844	0.0017	0.0828	0.0029	0.8627	0.1000	0.1509	0.0910
Logistic regression forward stepwise	15	0.9847	0.0014	0.0825	0.0011	0.8431	0.1109	0.1501	0.0907
Logistic regression forward stepwise	16	0.9850	0.0014	0.0878	0.0034	0.8824	0.0832	0.1595	0.0965
Logistic regression forward stepwise	17	0.9856	0.0017	0.0895	0.0038	0.8627	0.1000	0.1618	0.0983
Logistic regression forward stepwise	18	0.9859	0.0017	0.0878	0.0041	0.8235	0.1271	0.1584	0.0964
Logistic regression forward stepwise	19	0.9863	0.0017	0.0901	0.0105	0.8223	0.1259	0.1621	0.0989
Logistic regression forward stepwise	20	0.9862	0.0013	0.0915	0.0085	0.8419	0.0988	0.1648	0.1004
Logistic regression forward stepwise	21	0.9865	0.0017	0.0952	0.0136	0.8615	0.1101	0.1711	0.1045
Logistic regression forward stepwise	22	0.9857	0.0022	0.0811	0.0038	0.7635	0.1254	0.1462	0.0891
Logistic regression forward stepwise	23	0.9860	0.0018	0.0888	0.0114	0.8223	0.1259	0.1599	0.0974

Appendix C: Results Decision tree

In this appendix, we highlighted the models with the highest performance in terms of precision, recall, F1 score and $F_{0.33}$ score per type of model.

Table 15: Performance of the decision tree model with varying minimum leaf sizes. Validation is performed using 5-fold cross-validation. The highest performance in terms of precision, recall, F1 score and $F_{0.33}$ score are highlighted.

Model	Minimum leaf size	Accuracy		Precision		Recall		F1 Score	F0.33 Score
		Mean	Std	Mean	Std	Mean	Std	Mean	Mean
Decision Tree + Stop Criterion	1	0.9977	0.0002	0.2663	0.0524	0.2874	0.0923	0.2736	0.2487
Decision Tree + Stop Criterion	3	0.9976	0.0002	0.1747	0.0614	0.1537	0.0708	0.1622	0.1723
Decision Tree + Stop Criterion	5	0.9980	0.0004	0.2594	0.1701	0.1558	0.1144	0.1921	0.2432
Decision Tree + Stop Criterion	10	0.9982	0.0002	0.3124	0.1714	0.1126	0.0593	0.1623	0.2653
Decision Tree + Stop Criterion	15	0.9984	0.0001	0.3950	0.2100	0.1026	0.0563	0.1610	0.3074
Decision Tree + Stop Criterion	20	0.9983	0.0001	0.1952	0.1954	0.0516	0.0459	0.0807	0.1527
Decision Tree + Stop Criterion	25	0.9982	0.0002	0.2673	0.1876	0.1021	0.0633	0.1465	0.2301
Decision Tree + Stop Criterion	30	0.9982	0.0003	0.1127	0.0923	0.0516	0.0459	0.0691	0.1008
Decision Tree + Stop Criterion	155	0.9984	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0
Decision Tree + Stop Criterion	310	0.9984	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0
Decision Tree + Stop Criterion	465	0.9984	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0
Decision Tree + Stop Criterion	620	0.9984	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0

Table 16: Performance of the decision tree model with varying minimum leaf sizes and with the SMOTE + undersampling applied (SMOTE to a 1:10 ratio and undersampling to a 1:2 ratio). Validation is performed using 5-fold cross-validation. The highest performance in terms of precision, recall, F1 score and $F_{0.33}$ score are highlighted.

Model	Minimum leaf size	Accuracy		Precision		Recall		F1 Score	F0.33 Score
		Mean	Std	Mean	Std	Mean	Std	Mean	Mean
Decision Tree + Stop Criterion + SMOTE	1	0.9940	0.0006	0.1514	0.0285	0.6074	0.0744	0.2421	0.1608
Decision Tree + Stop Criterion + SMOTE	3	0.9944	0.0006	0.1315	0.0437	0.4737	0.1951	0.2048	0.1417
Decision Tree + Stop Criterion + SMOTE	5	0.9937	0.0007	0.1392	0.0235	0.5874	0.1561	0.2244	0.1507
Decision Tree + Stop Criterion + SMOTE	10	0.9928	0.0004	0.1390	0.0279	0.7005	0.1473	0.2318	0.1511
Decision Tree + Stop Criterion + SMOTE	15	0.9913	0.0012	0.1219	0.0160	0.7337	0.1519	0.2083	0.1330
Decision Tree + Stop Criterion + SMOTE	20	0.9899	0.0015	0.1114	0.0032	0.7832	0.1389	0.1942	0.1218
Decision Tree + Stop Criterion + SMOTE	25	0.9885	0.0008	0.0965	0.0158	0.7632	0.1390	0.1711	0.1057
Decision Tree + Stop Criterion + SMOTE	30	0.9871	0.0019	0.0916	0.0195	0.7942	0.1280	0.1639	0.1005
Decision Tree + Stop Criterion + SMOTE	155	0.9823	0.0044	0.0708	0.0162	0.8053	0.1265	0.1290	0.0779
Decision Tree + Stop Criterion + SMOTE	310	0.9683	0.0071	0.0447	0.0101	0.8984	0.0957	0.0848	0.0494
Decision Tree + Stop Criterion + SMOTE	465	0.9624	0.0036	0.0382	0.0025	0.9484	0.0658	0.0733	0.0422

Decision Tree + Stop Criterion + SMOTE	620	0.9646	0.0030	0.0404	0.0032	0.9484	0.0658	0.0775	0.0447

Table 17: Performance of the decision tree model with varying alphas used in cost complexity pruning. Validation is performed using 5-fold cross-validation. The highest performance in terms of precision, recall, F1 score and $F_{0.33}$ score are highlighted.

Model	Alpha	Accuracy		Precision		Recall		F1 Score	F0.33 Score
		Mean	Std	Mean	Std	Mean	Std	Mean	Mean
Decision Tree + Pruning	0.000000	0.9977	0.0002	0.2663	0.0524	0.2874	0.0923	0.2736	0.2154
Decision Tree + Pruning	0.000025	0.9979	0.0005	0.2850	0.0810	0.1953	0.0566	0.2246	0.2725
Decision Tree + Pruning	0.000050	0.9982	0.0004	0.4167	0.1826	0.1121	0.0571	0.1643	0.3277
Decision Tree + Pruning	0.000075	0.9983	0.0003	0.2364	0.3883	0.0411	0.0503	0.0639	0.1602
Decision Tree + Pruning	0.000100	0.9984	0.0001	0.1000	0.2000	0.0211	0.0421	0.0348	0.0727
Decision Tree + Pruning	0.000125	0.9984	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0
Decision Tree + Pruning	0.000150	0.9984	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0
Decision Tree + Pruning	0.000175	0.9984	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0
Decision Tree + Pruning	0.000200	0.9984	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0

Table 18: Performance of the decision tree model with varying alphas used in cost complexity pruning and with the SMOTE + undersampling applied (SMOTE to a 1:10 ratio and undersampling to a 1:2 ratio). Validation is performed using 5-fold cross-validation. The highest performance in terms of precision, recall, F1 score and $F_{0.33}$ score are highlighted.

Model	Alpha	Accuracy		Precision		Recall		F1 Score	F0.33 Score
		Mean	Std	Mean	Std	Mean	Std	Mean	Mean
Decision Tree + Pruning + SMOTE	0.000000	0.9948	0.0010	0.1598	0.0377	0.5268	0.1314	0.2437	0.1717
Decision Tree + Pruning + SMOTE	0.000025	0.9940	0.0007	0.1363	0.0319	0.5353	0.1500	0.2164	0.1473
Decision Tree + Pruning + SMOTE	0.000050	0.9950	0.0002	0.1690	0.0162	0.5674	0.0681	0.2603	0.1817
Decision Tree + Pruning + SMOTE	0.000075	0.9949	0.0004	0.1636	0.0053	0.5568	0.0695	0.2523	0.1760
Decision Tree + Pruning + SMOTE	0.000100	0.9945	0.0005	0.1547	0.0255	0.5668	0.0976	0.2424	0.1668
Decision Tree + Pruning + SMOTE	0.000125	0.9940	0.0007	0.1425	0.0249	0.5579	0.0956	0.2263	0.1540
Decision Tree + Pruning + SMOTE	0.000150	0.9941	0.0012	0.1497	0.0520	0.5679	0.1399	0.2364	0.1616
Decision Tree + Pruning + SMOTE	0.000175	0.9933	0.0008	0.1350	0.0170	0.6089	0.1216	0.2203	0.1464
Decision Tree + Pruning + SMOTE	0.000200	0.9932	0.0009	0.1283	0.0255	0.5674	0.0892	0.2089	0.1390

Appendix D: Results Grid Search Random Forest

In this appendix, we highlighted the models with the highest performance in terms of precision, recall, F1 score and $F_{0.33}$ score per type of model. We also highlight the model that is best in our opinion based on the preferences of management.

Table 19: Results of grid search of the random forest model. Validation is performed by 50 repetitions of the validation set approach with a training size of 80%. The highest performance over the 50 repetitions in terms of precision, recall, F1 score and $F_{0.33}$ score are highlighted.

Model	**Cut-off**	**Criterion**	**Maximum features**	**Accuracy**		**Precision**		**Recall**		**F1 Score**	**F0.33 Score**
				Mean	Std	Mean	Std	Mean	Std	Mean	Mean
Random forest	0.3	Gini	log2	0.9988	0.0003	0.7744	0.1766	0.3451	0.0844	0.4711	0.6888
Random forest	0.3	Gini	sqrt	0.9988	0.0003	0.7509	0.1811	0.3497	0.0807	0.4686	0.6736
Random forest	0.3	Gini	8	0.9988	0.0002	0.7154	0.1575	0.3625	0.0988	0.4720	0.6519
Random forest	0.3	Gini	10	0.9988	0.0003	0.7122	0.1936	0.3756	0.1066	0.4836	0.6536
Random forest	0.3	entropy	log2	0.9988	0.0003	0.7897	0.1649	0.3564	0.0901	0.4842	0.7041
Random forest	0.3	entropy	sqrt	0.9988	0.0002	0.7846	0.1630	0.3618	0.0900	0.4876	0.7025
Random forest	0.3	entropy	8	0.9988	0.0003	0.7394	0.1905	0.3702	0.1098	0.4857	0.6724
Random forest	0.3	entropy	10	0.9988	0.0002	0.7341	0.1612	0.3791	0.1100	0.4889	0.6712
Random forest	0.325	Gini	log2	0.9988	0.0003	0.7876	0.1976	0.2984	0.0881	0.4248	0.6766
Random forest	0.325	Gini	sqrt	0.9988	0.0002	0.7855	0.1910	0.3065	0.0884	0.4339	0.6793
Random forest	0.325	Gini	8	0.9988	0.0003	0.7601	0.2045	0.3382	0.1040	0.4604	0.6758
Random forest	0.325	Gini	10	0.9988	0.0003	0.7655	0.1890	0.3599	0.1075	0.4787	0.6880
Random forest	0.325	entropy	log2	0.9988	0.0002	0.7971	0.1948	0.3158	0.0897	0.4452	0.6917
Random forest	0.325	entropy	sqrt	0.9988	0.0003	0.8130	0.1750	0.3232	0.0964	0.4556	0.7060
Random forest	0.325	entropy	8	0.9988	0.0003	0.7633	0.1943	0.3489	0.1010	0.4728	0.6823
Random forest	0.325	entropy	10	0.9988	0.0002	0.7671	0.1617	0.3551	0.0875	0.4772	0.6874
Random forest	0.35	Gini	log2	0.9988	0.0002	0.8150	0.1874	0.2775	0.0907	0.4072	0.6827
Random forest	0.35	Gini	sqrt	0.9988	0.0003	0.7930	0.2047	0.2879	0.0871	0.4160	0.6747
Random forest	0.35	Gini	8	0.9988	0.0002	0.7734	0.1921	0.3030	0.0977	0.4275	0.6695
Random forest	0.35	Gini	10	0.9988	0.0003	0.7708	0.2056	0.3125	0.1039	0.4378	0.6722
Random forest	0.35	entropy	log2	0.9988	0.0003	0.8361	0.1998	0.2995	0.0917	0.4343	0.7089
Random forest	0.35	entropy	sqrt	0.9988	0.0003	0.8223	0.1728	0.3018	0.0927	0.4348	0.7013
Random forest	0.35	entropy	8	0.9988	0.0003	0.8065	0.2007	0.3135	0.0917	0.4452	0.6969
Random forest	0.35	entropy	10	0.9988	0.0003	0.7985	0.1644	0.3293	0.0980	0.4592	0.6989
Random forest	0.375	Gini	log2	0.9988	0.0003	0.8523	0.2025	0.2637	0.0891	0.3965	0.6968
Random forest	0.375	Gini	sqrt	0.9987	0.0003	0.8267	0.2037	0.2623	0.0875	0.3924	0.6804
Random forest	0.375	Gini	8	0.9988	0.0003	0.7994	0.1988	0.2848	0.0976	0.4115	0.6771

Random forest	0.375	Gini	10	0.9987	0.0003	0.7794	0.1956	0.2922	0.0917	0.4183	0.6680
Random forest	0.375	entropy	log2	0.9988	0.0003	0.8284	0.2041	0.2689	0.0937	0.3993	0.6858
Random forest	0.375	entropy	sqrt	0.9988	0.0003	0.8422	0.1927	0.2871	0.0873	0.4228	0.7058
Random forest	0.375	entropy	8	0.9988	0.0003	0.8009	0.2000	0.2966	0.0990	0.4263	0.6845
Random forest	0.375	entropy	10	0.9988	0.0002	0.7988	0.1751	0.2926	0.0966	0.4217	0.6810
Random forest	0.4	Gini	log2	0.9987	0.0003	0.8544	0.2014	0.2338	0.0875	0.3609	0.6752
Random forest	0.4	Gini	sqrt	0.9987	0.0003	0.8392	0.2099	0.2364	0.0913	0.3613	0.6687
Random forest	0.4	Gini	8	0.9987	0.0003	0.8201	0.2063	0.2523	0.0945	0.3780	0.6695
Random forest	0.4	Gini	10	0.9987	0.0003	0.7936	0.2138	0.2576	0.0966	0.3821	0.6569
Random forest	0.4	entropy	log2	0.9988	0.0003	0.8515	0.1949	0.2502	0.0934	0.3800	0.6865
Random forest	0.4	entropy	sqrt	0.9988	0.0003	0.8536	0.1905	0.2685	0.0887	0.4029	0.7008
Random forest	0.4	entropy	8	0.9988	0.0003	0.8355	0.2003	0.2688	0.0953	0.3995	0.6900
Random forest	0.4	entropy	10	0.9987	0.0003	0.8118	0.1943	0.2724	0.0897	0.4020	0.6776