Transparency in black box models

An investigation in interpretation methods for machine learning models in a probability to default setting

Sebastiaan Berendsen
2537690

Master Thesis MSc Business Analytics

Vrije Universiteit Amsterdam
Faculty of Science
De Boelelaan 1081a
1081 HV Amsterdam

Supervised by:
Sandjai Bhulai(VU)
Bojidar Ignatov(Deloitte)

Second Reader:
Mark Hoogendoorn(VU)

July, 2019

Preface

This thesis is written for the Master Project Business Analytics. The project is the graduation project for the MSc Business Analytics. This master is a multidisciplinary program with three subprograms. These subprograms are computational intelligence, financial risk management and business process optimization. This thesis is a combination of computational intelligence and financial risk management. Furthermore, the thesis is a six months internship. During this internship, the goal is to combine academic research with real-world problems.

The internship has taken place at the Financial Risk Management department of Deloitte. The department assists banks and insurers in developing and validating their risk models. An example of such a model is a probability to default model. Such a model can be used for client acceptance and capital estimation. In the past, statistical models were used for this type of modeling. However, with the rise of machine learning, statistical models are challenged by machine learning models. In this research, both types of such models will be created. Unfortunately, interpretation of a machine learning model is more challenging due to its opaque nature. Therefore, interpretation techniques of machine learning models will be investigated in a probability to default setting.

I would like to thank the team of Financial Risk Management, for giving me the opportunity to write my thesis at their department. In particular, I would like to thank Bojidar Ignatov for guiding and assisting me during my internship. Furthermore, I would like to thank Erik Wolters for helping me and coaching me during my internship. Finally, I would like to thank my university supervisor Sandjai Bhulai for all the guidance he has given me during the project.

Executive Summary

In recent years, machine learning models have shown the ability to be highly accurate in several applications. Such models are able to handle nonlinear relationships and can be used to boost efficiency of companies. Furthermore, machine learning models have shown to outperform traditional models in different fields. However, the improvements they are able to bring, comes at a cost of interpretability. Machine learning models are very complex and can be seen as a black box. In a black box model, the input is mapped to an output. This mapping process is opaque and therefore difficult to understand. The understanding of this mapping is key in fields like model development and model validation. In other words, creating transparency in a black box model is necessary. In this research, transparency in black box models will be created through model interpretation.

The goal of this research is to investigate and generate knowledge about interpretation methods for black box models. In interpretation methods some properties are defined. The first is the difference between local and global model interpretation. Local model interpretation refers to the explanation of a model's prediction. Global model interpretation is the process of understanding the behavior of a model. In this research the focus will be on local model interpretation. However, the possibility of creating global interpretation with multiple local interpretations is also investigated.

Second, a difference must be made between model specific and model agnostic interpretation methods. Model specific methods leverage the inner workings of a model. That means, that such a method will only be applicable, for example, a neural network. Model agnostic methods treat the model truly as a black box model. These types of methods will not look at a model's inner workings and can therefore be used for any type of models. The focus will be on such methods since the ability that they work on any type of model is very useful.

Without proper models, proper analysis on interpretation methods is not possible. Therefore, the first part of this research will be developing these models. In this research, several probability to default acceptance models are developed. The type of models that will be developed, are a logistic regression model, a random forest, a gradient boosting model and a neural network. From these models, the gradient boosting model performed the best. However, using machine learning for probability to default modelling introduces some challenges. One of them is class imbalance. One of the solutions to solve this problem is balancing the data set and creating a rating system. Furthermore, using tree ensembles can introduce tail risk in a probability to default curve.

On interpretability methods, the main focus will be on local, model agnostic methods. The two most used methods are the SHAP and the LIME methods. Therefore, a theoretical and practical analysis is performed in this research. First, an analysis on the settings of these methods is done. Furthermore, the methods are judged on their capability to detect influential features. Finally, the methods will be investigated on their capability to provide global model interpretation.

After this research, the conclusion is made that both methods are able to generate useful explanations and therefore can create local interpretation. However, the LIME method is subject to a large space of parameters. The setting of these parameters have influence on the generated explanation. Furthermore, the data must be standardized if one plans to use the LIME method. Finally, the SHAP method is able to provide global interpretation, where the LIME method has

more difficulty with providing global interpretation. All this makes that the SHAP method has preference above the LIME method. However, it is wise to use both methods next to each other, as methods could behave different in other environments. In this research, a test is proposed to see which method is able to detect influential variables better in different settings.

Contents

List of Abbreviations

ANN	Artificial Neural Network
AUC	Area Under the Curve
DFS	Deep Feature Synthesis
EI	Expected Improvement
FN	False Negatives
FP	False positives
FPR	False Positive Rate
GDPR	General Data Protection Regulation

LIME	Local Interpretable model agnostic Explanations
MLP	Multilayer Perceptron
PD	Probability to Default
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
SHAP	Shapley Additive Explanations
TN	True negatives
TP	True positives
TPE	Tree Parzen Estimator
TPR	True Positive Rate

List of Symbols

β	Parameter vector in a logistic regression
γ	Weight matrix in Gradient Boosting
B	Number of trees in a tree ensemble
f	Black box model to be explained
G	Number of bins in the credit rating system
g	Local interpretable model
M	Number of features
w	Weight vector, where w_i is the weight i
$W^{[l]}$	Weight matrix for layer l
X	Set of predictor variables
x_i	Feature i
Y	Set of target variables
y	Target variable
Z	Set of instances sampled around x_i

1 Introduction

Machine learning models, such as neural networks and ensemble models, have shown to be highly accurate in several applications. These models are able to handle multiple nonlinear relationships and can model interaction effects between variables. Due to the complexity of these models, they are often treated as black box. In a black box model, the input is mapped to an output. However, this mapping process is opaque and difficult to understand, meaning that these models are difficult to analyze and validate. However, model interpretation plays an important role in developing and validating models. In general, users are uncomfortable using models if they are opaque. Therefore, interpretability of models is necessary to create insight, transparency and develop trust in black box models.

1.1 Problem Statement

One of the applications where machine learning can outperform traditional models is credit risk. At the Financial Risk Management (FRM) team of Deloitte Risk Advisory, they develop such models. The model that will be developed in this research is a probability to default (PD) model. A PD model is used for client acceptance, capital calculations and serves several other goals. As a PD model is used for several goals, it is also subject to different optimization targets. When a PD model is used for client acceptance, the model must have accurate scores and must be able to distinguish clients correctly. In other words, the model should have sufficient performance on a client level. However, when a PD model is used for capital estimations, performance on client level is less important. Instead, the model must be able to estimate the PD on a portfolio level correctly such that enough capital is retained by a lending company.

Therefore, it is important to understand and trust the model deployed. Furthermore, if such a model is used for client acceptance, clients have the right for explanation according to the General Data Protection Regulation (GDPR). These regulations describe that a client has the right for an explanation of why the application was denied.

All this means that model transparency is needed and can be achieved through interpretation of the model. Therefore, model interpretation possibilities of black box models are investigated in this study. The goal of this study is to provide an analysis, theoretical and practical, and generate knowledge on interpretation methods for black box models.

Model interpretation can be divided into two subcategories. The first is global model interpretation. Global model interpretation is the process of interpreting the general model behavior. As an example, the well-known feature importance method for tree ensembles can be seen as global interpretation. The second is local model interpretation. The process of local model interpretation refers to explaining decision or predictions of a black box model. In this research, the emphasis will be on such interpretations.

Several model interpretation techniques leverage the inner workings of a black box model, such techniques are called model specific. However, if every type of black box model needs its own explanation technique, the field of interpretation techniques will be very large and complex. Furthermore, if a team needs to review many models, learning and using all different kinds of methods is time consuming and expensive. Therefore, model agnostic methods are created. Furthermore, model agnostic methods have more practical value. For example, if a model is only accessible externally. In model agnostic methods, the inner workings of a model will not be leveraged. This means that the model will be treated completely as black box. In this research, the emphasis will be on model agnostic methods.

1.2 Related Work

In this section some related work in the subject will be discussed. First, some articles will be discussed that used machine learning for a PD model. Second, some interpretation methods will be discussed.

For a long time, PD modeling is done by using logistic regression. However, with the rise of machine learning models, the traditional models were challenged by the machine learning models. In "Machine learning in credit risk modelling" [1], the authors explore models like Random Forest and Gradient Boosting. They show that these models perform significantly better than the traditional methods. Furthermore, deep learning can also be used to improve current models [2]. However, both articles focus on predictive performance. In PD modeling, outcomes are often calibrated using a rating system. If a rating system is used, machine learning is still able to outperform traditional models and a rating system can even increase the robustness of a model [20].

The field of interpretation techniques has been rising over the past years. A lot of research went into developing model-specific methods for deep learning models, e.g., computer vision and natural language processing. However, these fields are outside of the scope of this research and therefore left out. The two main methods that are discussed in this research are the LIME [22] and SHAP [14] frameworks. These methods will be further explained in Section 6. Unfortunately, not a lot of research has been done on these frameworks. However, both frameworks have robustness issues [4]. Although, the authors raised the question if it is reasonable to judge the frameworks on robustness if the underlying model is not robust.

The data set used in this research has been taken from a website called Kaggle, which will further be explained in Section 2.1. After some research into other projects that used this data, it became clear that a gradient boosting model was performing the best on this data set. Further knowledge about the data and models could also be taken from this competition. However, this could lead to insufficient understanding in the data, developed models and use case. Understanding these parts are from great importance to generate the knowledge needed during the inspection of transparency methods. For example, understanding the data and the use case can help confirming transparency outcomes by checking if they align with human intuition.

1.3 Outline

In various machine learning studies, the focus lies on achieving the highest possible predictive performance. In this study, that will be different as the focus will be on interpretation techniques. However, proper interpretation is not possible without proper models. Therefore, this study consists of two parts. The first will be the development of several models. Thereafter, the two transparency methods will be investigated.

In Chapter 2, the data used in this research will be explored. In this section, an overview of the data will be given. Furthermore, all the pre-processing steps required are described in that section. In Chapter 3, the model development will be discussed. Besides the theoretical explanation of the used models, several other methods are discussed. Such methods concern hyperparameter optimization, rating system calibration and model evaluation. In Chapter 4, the experimental setup is discussed. In Chapter 5, the model performance is shown and two models are selected. These two models are used to assess the transparency methods. In Chapter 6, a theoretical analysis of the transparency is given. Whereafter, in Chapter 7, an analysis is done on several properties using the created models and the created data set. In Chapter 8, a conclusion of the analysis will be given. Finally, in Chapter 9, recommendations and opportunities for further research are given.

2 Data Description and Preparation

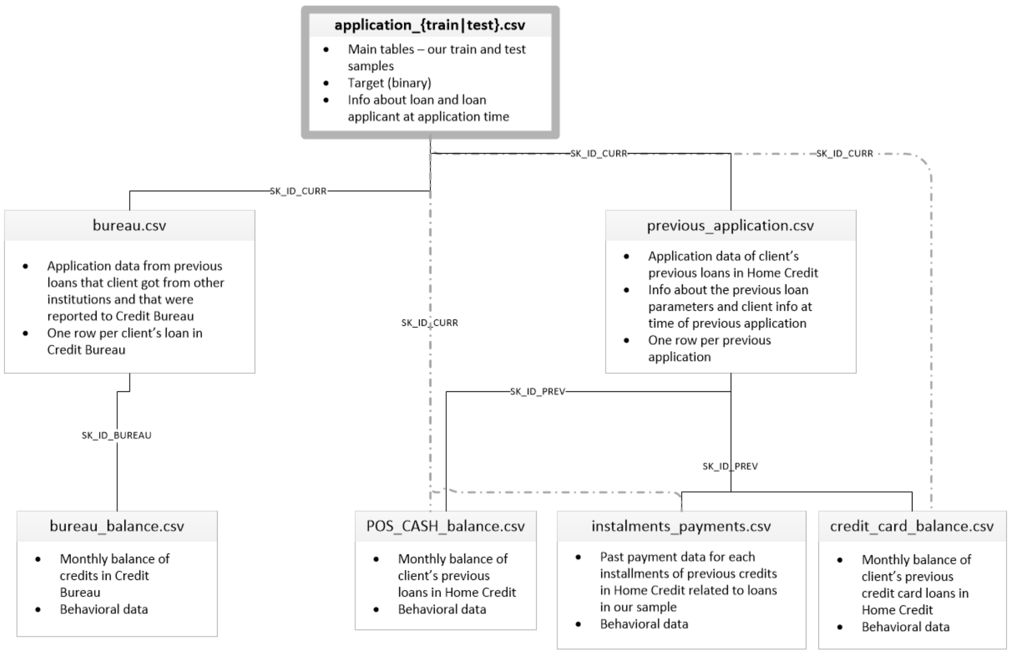
In this section, the data used in this research will be introduced. The data will be described and some insights in the data will be given. Furthermore, the procedure of data aggregation is explained. At last, the methods used to select features will be explained.

The availability of data is crucial for this research. However, choices in data engineering can highly affect performance of future models. Important is, that achieving high performance does not have a high priority. Therefore, a generic approach is taken to create a data set that can be used in this research. However, the goal is to achieve reasonable models that are able to capture patterns present in the data set.

2.1 Data Description

The data used for this study is taken from a data science competition site named Kaggle. The competition is called the Home Credit Default Risk Competition[1]. Home Credit is a non-bank financial institution focusing on people with little or no credit history. Home Credit mostly operates in Asia and the Commonwealth of Independent States (old Soviet Union states). A couple of years ago, they started operating in the USA. Unfortunately, the origin of the data set is not known.

In Figure 1, an overview of the data can be found. In this figure, all available tables and their relationships are shown. Furthermore, some initial information on the data in each table is present.



application_{train|test}.csv
Main tables – our train and test samples
Target (binary)
Info about loan and loan applicant at application time
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
bureau.csv
Application data from previous loans that client got from other institutions and that were reported to Credit Bureau
One row per client's loan in Credit Bureau
previous_application.csv
Application data of client's previous loans in Home Credit
Info about the previous loan parameters and client info at time of previous application
One row per previous application
SK_ID_BUREAU
SK_ID_PREV
SK_ID_PREV
bureau_balance.csv
Monthly balance of credits in Credit Bureau
Behavioral data
POS_CASH_balance.csv
Monthly balance of client's previous loans in Home Credit
Behavioral data
instalments_payments.csv
Past payment data for each installments of previous credits in Home Credit related to loans in our sample
Behavioral data
credit_card_balance.csv
Monthly balance of client's previous credit card loans in Home Credit
Behavioral data

Figure 1: Data overview

[1]https://www.kaggle.com/c/home-credit-default-risk

On Kaggle, companies host a competition with a data set and a question. The question of Home Credit is, "Can you predict how capable each applicant is of repaying a loan?" The answer to this question can be defined by the following target:

$$TARGET = \begin{cases} 1, & \text{client with payment difficulties,} \\ 0, & \text{else.} \end{cases} \tag{1}$$

A client has payment difficulties if he or she had a late payment more than X days on at least one of the first Y installments on the loan[2]. Unfortunately, it is not available what these X and Y are. However, the target should be a good description whether the new applicant is creditworthy. To simplify the definition given in (1), a client with payment difficulties will be labelled as "Default" and "No Default" otherwise.

There are two versions of the main application table, a train and test version. Only the train version contains the target variable. Therefore, the test version and all related data in other tables will be discarded in this research. The test version could be used to assess the performance of the model in an unbiased way. However, predictive performance is not in the scope of this research and such an analysis is left out.

In the main table, new applications are listed. In other tables, information about previous credits can be found. Here, a distinction is made between credits at other bureaus or previous credits at Home Credit. For both, information is available on payment behavior of the new applicant. This information helps assessing the creditworthiness of an applicant. An overview of the information can be found in Table 1. For a detailed description of all attributes available in the tables, one can check Appendix A.

Table	**Shape**	**Information**
Application	(307,511; 122)	Information about loan and applicant
Bureau	(1,465,325; 17)	Info of previous loans at other institutions reported to Credit Bureau
Bureau balance	(14,701,612; 3)	Monthly balance of credits in Credit Bureau
Prev. Application	(2,354,993; 37)	Info of previous loans at Home Credit
POS CASH balance	(10,572,221; 8)	Monthly balance of previous loans in Home Credit
Installments payments	(8,251,754; 8)	Past payment data for each installment of previous credits at Home Credit
Credit card balance	(1,413,701; 23)	Monthly balance of client's previous credit card loans in Home Credit

Table 1: Information on all the tables

In Table 1, it can be seen that the tables are from different sizes. Therefore, the additional tables need to be aggregated with the main table. Besides in Figure 1, the relationships between tables can be found in Table 2.

[2]Note that these X and Y do not relate to those stated in the list of symbols and are only used here.

Relationship	**Description**
Application → Bureau	One row in Application can have zero or more in bureau
Application → Prev. application	One row in Application can have zero or more in Prev. application
Bureau → Bureau balance	One row in Bureau can have zero or more in Bureau balance
Prev. application → POS CASH	One row in Prev. application can have zero or more in POS CASH balance
Prev. application → Installments	One row in Prev. application can have zero or more in Installments payments
Prev. application → Credit card	One row in Prev. application can have zero or more in Credit card balance

Table 2: Relationships between tables

2.2 Exploratory Data Analysis

In this section, some exploratory insights into the data will be given. Due to the size of the data, not all attributes will be thoroughly investigated. Instead, some general insights into the data will be given. Furthermore, the target variable will be analyzed in order to create understanding in the target. At last, some interesting aspects that were found during the analysis are shown.

In Figure 2, the percentage of missing values for several attributes can be seen. Attributes that do not contain missing values are omitted in this picture. In total, 103 attributes contain missing values. Several algorithms, e.g., Random Forest, can handle missing values. Unfortunately, not all algorithms have that property, e.g., a Neural Network. Therefore, all missing values need to be replaced. The exact missing value treatment will be discussed in Section 2.4.1.

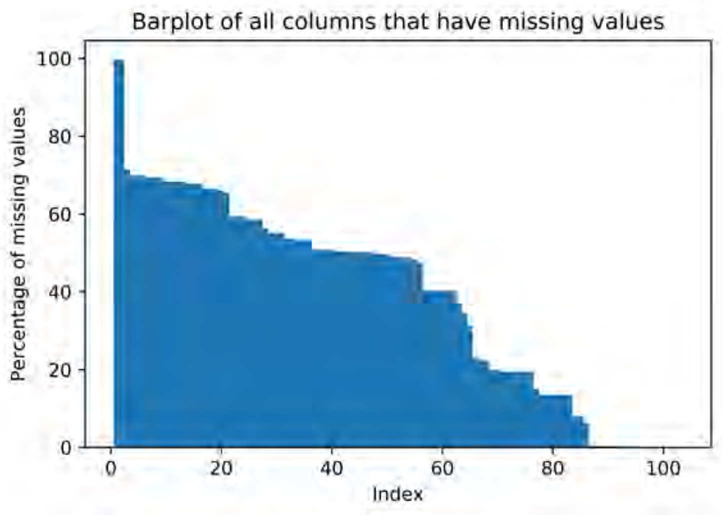


Figure 2: Percentage of missing values, columns without missing values are omitted

Notable is that 99.64% of the attributes RATE_INTEREST_PRIVILEGED and RATE_INTEREST_PRIMARY are missing. These attributes can be found in the previous application table. Attributes with that number of missing values barely contain any information.

Each attribute can be of numerical or categorical type. In Table 3, the distribution of numerical and categorical between the different tables can be seen. In these numbers, client IDs are omitted. Client IDs are not informative features and can only be used as as a relationship indicator between tables.

Column	**Numerical**	**Categorical**
Application	76	44
Bureau	12	3
Bureau balance	1	1
Prev. Application	17	19
POS CASH balance	5	1
Installments payments	6	0
Credit card balance	20	1

Table 3: Number of numerical and categorical features for each table

2.2.1 Target variable analysis

In this section, an analysis of the target variable is done. As described in Section 2.1, the target is defined by:

$$\text{TARGET} = Y = \begin{cases} 0, & \text{No Default,} \\ 1, & \text{Default.} \end{cases} \tag{2}$$

The target variable is binary, which means that the learning goal is binary classification. In binary classification, the balance between the classes has a large influence. In Table 4, it can be seen that the target is skewed towards one class, the majority class. In other words, the data set is highly imbalanced, where the class Default is the minority class. Models created on imbalanced data can be biased towards the majority class. Such bias can occur when a model learns mostly classification rules for the majority class.

Percentage No Defaults	**Percentage Defaults**
91.93%	8.07 %

Table 4: The target variable is highly imbalanced

2.3 Feature Engineering

Feature engineering is an important step in the process of developing a predictive model. The quality and quantity of features will influence model performance. In this section, the process of feature engineering and the techniques used are explained.

As explained in Section 2.1, the data consists of several tables. Therefore, all these tables need to be aggregated into one. Such an aggregation can be done by performing several calculations, e.g., taking the sum or the mean. By combining these methods of aggregation, multiple features for each attribute can be created. Since the dimensionality of the data set is already high, i.e., 207 columns, this will be the only method used.

Deep Feature Synthesis (DFS) [11] will be used for the data aggregation. DFS is an automated feature engineering method that can handle data sets with multiple tables and relationships. The "Deep" in Deep Feature Synthesis represents the ability to stack multilevel relationships. DFS can aggregate tables when all relationships between tables are known. It is also possible to perform data transformations for feature engineering within one table, e.g., taking the absolute of a numeric value. However, such transformations are not performed in order to keep the dimensionality of the data set manageable. Upfront, it is impossible to know which aggregation method is going to be effective. Therefore, multiple aggregation primitives are used. The aggregation primitives used are listed in Table 5.

Data aggregations
Sum
Min
Max
Mean
Standard dev.
Count

Table 5: Aggregation primitives

Some examples of created features are listed below. In these examples, words in front of the parentheses are the aggregation primitives. The words in small caps are the table in which the attribute is present. The attribute itself is always present after a dot.

- MEAN(bureau.AMT-CREDIT-SUM-DEBT), AMT-CREDIT-SUM-DEBT is the amount of credit in debt by a previous credit agency. DFS calculates the mean over all debts known at other agencies, as well as all the other primitives. The depth of this feature is one, since AMT-CREDIT-SUM-DEBT is present in a table directly related to the main table.
- MEAN(previous-application.SUM(install-payments.AMT-INSTALMENT), AMT-INSTALMENT is the prescribed installment payment on a previous credit. This feature creates a sum of this amount for each previous application. Whereafter, the mean is taken over each previous application of one client. The depth of this feature is two.

After all aggregations are performed, the total data set contains 1,130 features. At this stage all missing values were left missing. Due to this fact, a large number of the 1,130 features contain missing values.

2.4 Pre-processing of data

Before the data can be fed to the models, the data must be pre-processed. First, all missing values will be deleted or replaced. Whereafter, categorical variables will be transformed so that all models are able to process them. At last the solution regarding the imbalanced data set will be described.

2.4.1 Missing value treatment

Some models are able to handle missing values. However, not all models possess this ability. In order to work with one data set, all missing values need to be treated upfront. At first, features with a lot of missing values will be eliminated. This is done using some threshold, for numerical variables the threshold is set on 70% and 50% for categorical variables. These thresholds were set on the number of features that got erased afterward. Due to the size of the data set more strict thresholds are possible. However, fully optimizing the performance is not in the scope of this research.

After the elimination, missing values need to be imputed. Missing values in categorical features will be replaced with the category "Unknown". With numerical features, a distinction is made between the original features and the one made in Section 2.3. The engineered features are imputed with zero. A missing value in this type of feature indicates that there was no information available. Imputing these features with zero should imply that there was no information. Original features are imputed with their mean. An overview of the missing value treatment can be seen in Table 6.

Variable type	**Treatment**
Categorical	Set to unknown
Engineered numerical	Imputed with zero
Original numerical	Imputed with mean

Table 6: Overview of missing value treatment

2.4.2 Encoding categorical variables

As described in Section 2.2, several categorical variables are present in the data set. The algorithms used in this research can cope with categorical variables in different ways. However, to simplify and prevent the use of multiple data sets, one universal technique is applied. The technique used is called one-hot encoding, in statistics referred to as dummy variables.

If the categorical variables consist of L categories, $L-1$ dummy variables are created. In that dummy variable, an indicator value represents one if the category is present in that instance and zero otherwise. The L^{th} variable is encoded in the combination of all dummy variables being zero.

2.4.3 Balance data set

In Section 2.2.1, the imbalanced nature of the data set was described. If models are trained on highly imbalanced nature, the probability exists that a model will only learn majority class classification rules. Furthermore, the risk exists that the model will be biased towards the majority class. In this research, data is sampled in order to create a more balanced data set. Sampling can be done by oversampling the minority class or undersampling the majority class.

Oversampling is the process of creating artificial data with the data that already exists. This can amplify data biases which can cause a larger generalization error. This can be caused by several issues, for example through a relative lack of data or an inappropriate learning bias [26]. Undersampling removes instances from the majority class and thus data is thrown away. The desired ratio is 1 : 2. This ratio was decided after some discussion. With a full balancing of the data set to a 1 : 1 ratio, too much data would be thrown away. On the other side, a lower ratio will cause a biased acceptance model.

This means that eventually, for each minority class instance, two majority class instances are present. This is done by taking all minority class instances and sample random majority class instances, without replacement, until the desired ratio is achieved.

Unfortunately, undersampling brings another challenge. The technique chosen has a warping effect on the posterior probability [21]. That is, the probabilities predicted cannot be interpreted as a PD anymore. It should rather be interpreted as a credit score. The results are good client classifications, needed for client acceptance, but bad client and portfolio PD estimations, needed for capital calculations. Therefore, the credit score needs to be translated to a PD. This can be done using credit rating calibration. This will be further explained in Section 3.7.

2.4.4 Standardization of the data

The data set consists of hundreds of features. These features have different type of units, e.g., dollars, m^2, days or are even unitless. The problem of these different types of units is that not all models are able to cope with that. It is not a problem for tree ensembles since these type of models depend on making splits in the data. However, other models, like neural networks, will have problems with data that has different units. One disadvantage is that the optimization process will be slow. Furthermore, weights in the network can be extremely high if one unit is in thousand dollars and another unit as a range between zero and one. Large weights in a network can cause a model that is non-robust. Therefore, the data needs to be standardized, or normalized. The data will be standardized by removing the mean and scaling to the unit variance. In mathematical terms, the following operations are done:

$$\bar{x}_i = \frac{x_i - \mu_i}{\sigma_i}, \tag{3}$$

where $\bar{x}_i$ is the standardized feature, x_i is the unstandardized feature, μ_i is the mean of that feature and σ_i is the variance of that feature.

2.5 Feature Selection

After feature engineering, the feature space is enormous. In total, there are more than 1,000 features. Therefore, features need to be eliminated. Multiple methods are used for feature selection. As explained in Section 2.4.1, features with a large number of missing values are already removed. Features with a lot of missing values will not contain a lot of information that can be exploited by a model.

In some sense, features that are highly correlated with each other contain the same information. Therefore, one of these features will become redundant. This means that one of these features can be removed. Features are highly correlated if their correlation exceeds a certain threshold. In this research, the threshold is set on 0.9. If this value was set higher, the number of features erased by this feature was too high. Lowering the threshold could the learning process of some models, e.g., a neural network. However, as explained, in Section 1.2, a gradient boosting model is expected to perform the best. Such a model is not affected by correlation between features.

Furthermore, some features contain only one value. Features with one single unique value cannot be of use for any model, since it has zero variance. Therefore, the features with only one value are removed.

At last, feature selection is performed through feature importance. In a tree-based model, the importance of a feature can be calculated. If this importance is zero, the feature is not used by the model. Since the dimensionality of the data is still high, such features are assumed to have little predictive power and therefore removed from the data. As a tree-based model, Gradient Boosting is used. In the creation of such a model, data is sampled at random. This can result in different models for several runs. Therefore, five Gradient Boosting models are created. The features that have an importance of zero, in all models, will be removed. Gradient Boosting models will be further explained in Section 3.4.

After all operations, the feature space is reduced from more than 1,000 features, to a final data set with 431 features. An overview of all the feature selection methods can be found in Table 7.

Mechanisms	**Explanation**	**Features removed**
Missing value	Erase under certain threshold, whereafter the values are imputed	373
Correlation	Erase features with higher correlation than 0.9	475
Single unique value	Features with only one value are erased	5
Feature importance	Remove features with zero importance in a Gradient Boosting Model	8

Table 7: Overview of feature selection

Finally, three variable exists that are external credit scores. Financial institutions can buy such scores from external companies. One of the goals of this research is to show that machine learning can replace traditional models using such scores. Therefore, the external scores will not be used in the model development and will therefore be dropped from now on. However, they will be used later in further analysis into transparency methods. Upfront, it is reasonable to assume that these scores will be strong predictors. Therefore, they can be used to investigate transparency methods to see if they are able to pick up such predictors. More on transparency will be explained in Section 6.

3 Model Development

In this section, the models used in this research will be explained. At first, the baseline model will be explained. As a baseline model, logistic regression is used since it is a widely adopted model for PD estimation. Second, a decision tree will be explained, because two of the four models are tree ensembles. A tree ensemble combines multiple decision trees. Third, the methods Random Forest and Gradient Boosting will be explained. These methods are widely used and known for their predictive power. Last, an Artificial Neural Network (ANN) will be explained.

After the explanation of all models, Bayesian optimization will be discussed. With Bayesian optimization, hyperparameters are tuned. After this, the credit rating system will be explained. At last some model evaluation methods will be discussed.

3.1 Logistic Regression

Logistic regression, sometimes referred to as logit, is one of the most used statistical methods for classification. A big advantage of the logit model is that it computes a probability given some predictor variables. The logistic regression model squeezes the output of a linear function between zero and one by using the logit function. The logit function is defined by:

$$logit(\eta) = \frac{1}{1 + exp(-\eta)}. \tag{4}$$

In logistic regression η is defined by a linear function. This means that η can be defined as:

$$\eta = \beta_0 + \sum_{i=1}^{M} \beta_i x_i, \tag{5}$$

where M is the number of features, x_i is feature i and β a vector of parameters. The vector of parameters β contains an intercept and a coefficient for each feature x_i. Note that Equation (5) relates to the linear regression model. In logistic regression, the linear regression model is transformed with the logit function. The transformation enables the model to handle classification problems and the output of probabilities. The combination of the two equations above leads to the full model for classification:

$$P(Y = Default|X) = \frac{1}{1 + exp(-(\beta_0 + \sum_{i=1}^{M} \beta_i x_i))}. \tag{6}$$

The parameters need to be estimated using data. This is done using the limited memory BFGS method [13]. The Limited memory BFGS method is an optimization algorithm often used in machine learning. It is a robust method and uses limited computer memory. Furthermore, to avoid overfitting, L2 regularization is used. L2 regularization penalizes large weights and therefore shrinks them. If large weights are penalized, features that seem over important in the data set will not get a too large weight. Important to note is that it does not shrink weights to zero. Hence, it is no feature selection method.

3.2 Decision Tree

A decision tree is a commonly used machine learning model. It uses a tree structure to predict the target variable y using some predictor variables X. An example of a simple decision tree can be found in Figure 3.

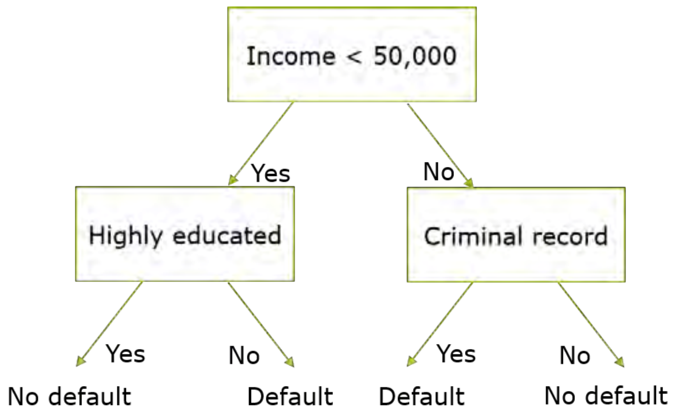


Figure 3: Example of a simple decision tree

This decision tree predicts whether one would default based on his income. When someone's income is lower than 10,000, this decision tree will predict that the person will default. One way to interpret a decision tree, is to see it as a set of rules. For this example, such a rule could be; if $Income < 10,000$ then $Default$ else $No\,Default$.

The tree above uses only three predictor variables. Therefore, it is easy to build. However, when multiple predictor variables are used, building a tree can be more complex. Several algorithms exist for building trees. A commonly used algorithm is the Classification And Regression Tree(CART) algorithm [7]. The goal of CART is to develop a binary tree that minimizes the error in each terminal node. A terminal node is a last node in the tree and represents the prediction.

A common approach for building trees is a greedy approach called recursive binary splitting. In each iteration, a new node will be designed. Such a node will split the data set into two subsets, as in Figure 3. In order to minimize the error, the splitting will be done according to a splitting criterion. The criterion quantifies how well a split separates the target variable. In the CART algorithm, this is done according to the Gini Impurity. The Gini Impurity is defined by:

$$I_G(p) = 1 - p_{Default}, \tag{7}$$

where $p_{Default}$ is the fraction of instances labeled as going into default in the set. The split with the lowest impurity will perform the best. Another criterion is the information gain. In the next iteration, the same calculations will be made on a subset of the data. The subset is created with the instances split by the node above.

As the algorithm is a recursive method, stopping criteria are needed. First, a node is a terminal node if there are no new splits possible that improve the splitting criterion. Second, if a new node covers only a small subset of the data set, e.g., 5 or 10 instances, the node will not be added. This stopping criterion is also used to prevent overfitting.

One of the biggest problems with decision trees is that they can create overly complex models. This happens when the model uses too many splits and the model will overfit to the training set, also known as high variance. Furthermore, decision trees are known to be non-robust. The model is highly sensitive to small changes in the training set. These small changes can lead to different models while the same relationships are present in the data.

3.3 Random Forest

An ensemble method is a form of machine learning where, multiple models are combined into one larger model. Bagging and boosting are the two most known ensemble methods. Boosting will be explained in Section 3.4. The most used algorithm of bagging for decision trees is the Random Forest algorithm developed by Breiman [6].

Bagging, or bootstrap aggregation, tries to overcome the problems of a single decision tree. By building multiple trees, a model is created that is more stable and that can generalize better. All trees are built on random subsets of the data. The predictions of all trees are combined in one final prediction. A simplified example of a Random Forest can be seen in Figure 4.

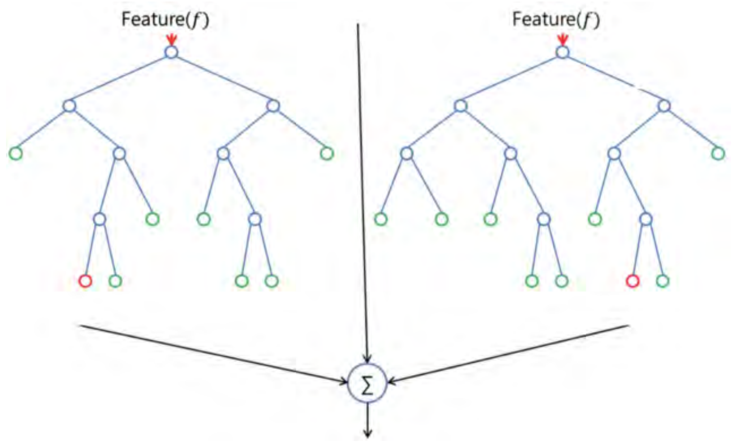


Figure 4: A simplified example of a Random Forest.
Source: `https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd`

The general algorithm for bagging with trees can be described in two steps. The number of trees to build, B is defined upfront. The steps for constructing an ensemble model using bagging can be seen in Algorithm 1.

Algorithm 1: General algorithm for bagging decision trees

```
for $b$ = 1, ..., $B$ do
    Sample, with replacement, $n$ training instances. Creating subsets $X_b$ and $Y_b$;
    Train a decision tree using the subset $X_b$ and $Y_b$;
end
Predictions $\leftarrow$majority vote of all trees;
```

A danger in bagging is the creation of similar trees. Often, the same features are found to be important in a decision tree. If this happens, trees will get correlated and bagging them will not be of any advantage. This results in the fact that the model will still have a high variance.

Therefore, a Random Forest does not consider all features when building a tree. When a tree is constructed, a random subset of X_b is selected as candidates for a split. Therefore, trees will be different since different features are used while building the random forest. For classification, the authors of the paper recommend to create a subset of $\sqrt{M}$ features for each split [6].

If the Random Forest algorithm is used for classification, the model is an ensemble of B classification trees. Each tree predicts whether the client will default or not by assigning a default or no default. However, by combining all trees, a probability can be determined. The probability that a client will default is defined by the fraction of trees that predicts a default. An implication is that probabilities close to zero and one rarely occur. This happens because the decision trees are trained on different data sets. This process causes that the trees will become uncorrelated. That causes that it is very common that all trees will agree that a person will go in default. Therefore, the probabilities are centered around the average probability.

3.4 Gradient Boosting

Boosting is another famous ensemble method, just as bagging. Where bagging builds several trees independently, boosting builds trees sequentially. Trees in the ensemble are dependent on the performance of other trees in the ensemble. In general, subsequent base predictors, e.g., a decision tree, learns from the mistakes of the previous predictors. The approach taken for gradient boosting is based on the paper from Friedman(1999) [9]. A naive formalization for boosting can be found in Algorithm 2.

Algorithm 2: General algorithm for boosting decision trees

```
Fit initial model $F_1(x)$;
for $b$ = 1, ..., $B$ do
    Fit a model on the residuals $h_b(x) = y - F_b(x)$;
    Create new model $F_{b+1}(x) = F_b(x) + h_b(x)$;
end
```

B is again the number of trees used in the ensemble, this number can best be set using a holdout set. On this holdout set, the performance of the ensemble is tracked. When this performance stops increasing, the number of trees used should not increase. If trees will still be added, it is likely that the final model overfits to the training data.

The gradient boosting algorithm is an additive model. This means that a recursive function can be defined as followed:

$$F_{b+1}(x) = F_b(x) + h_b(x) = y, \text{ for } b \geq 0, \tag{8}$$

this equation can be rewritten as:

$$h_b(x) = y - F_b(x) = r_b, \tag{9}$$

where $h_b(x)$ is the to be created model where the target are residuals r_b.

Gradient boosting differs from Algorithm, 2 in several steps. First, in Gradient Boosting new trees are not fitted on the residuals. However, they are fitted on pseudo-residuals called r_b. These pseudo-residuals are computed by the gradient of a loss function. Second, in classification, the

unit of the predicted value is the log odds. Furthermore, a learning rate ν is introduced.
As input, Gradient Boosting needs training data and a differentiable loss function $L(y_i, F(x_i))$. Any differentiable loss function can be optimized. In most classification problems, the log loss function is used. The log loss function is defined by:

$$L(y_i, F(x_i)) = -(y_i \text{log}(p) + (1 - y_i)\text{log}(1 - p)), \tag{10}$$

where p is equal to the predicted probability to default.

The first step in the algorithm, is to initialize a model with a constant value. This model is called the initial leaf and is defined by $F_0(x)$. The model $F_0(x)$ can be seen as the base value. The initial model F_0 is computed by solving:

$$F_0(x) = \underset{\gamma_0}{\text{argmin}} \sum_{i=1}^{N} L(y_i, \gamma_0), \tag{11}$$

where γ_0 relates to the log(odds) for the base prediction.

Step two of Gradient Boosting consists of four phases. These phases are repeated until B trees are created. The first phase is to compute the pseudo-residual r_b. The pseudo-residual is computed by taking the gradient of the loss function:

$$r_{ib} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x_i)=F_{b-1}(x_i)}. \tag{12}$$

With knowledge of the derivative for the log loss function, the pseudo-residual can be rewritten as:

$$r_{ib} = -\bigg(- y_i + \frac{exp(\gamma)}{1 + exp(\gamma)}\bigg) = \bigg(y_i - \frac{exp(\gamma)}{1 + exp(\gamma)}\bigg) = y_i - p, \tag{13}$$

where γ is the most recent predicted log(odds). The pseudo-residuals are then used as the target variable when a new tree is built. This is the second phase. In this phase, a regression tree is fitted to r_b. This creates terminal nodes R_{jb}, where $j = 1, \ldots, J_b$. The variable J_b represents the number of terminal nodes in tree b. All pseudo-residuals end up in one of these terminal nodes.

The third phase consists of computing the new log(odds) values γ_{jb}. The value of γ_{jb} represents the output value for that terminal node. For each terminal node, γ_{jb} is computed by solving:

$$y_{jb} = \underset{\gamma}{\text{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{b-1}(x_i) + \gamma), \tag{14}$$

where x_i are the instances that fall in terminal node R_{ij}. When all output values are computed, the fourth phase updates $F_b(x)$. This means that new predictions are created for each sample. The new predictions are generated by the following:

$$F_b(x) = F_{b-1}(x) + \nu \sum_{j=1}^{J_b} \gamma_{jb} I(x \in R_{jb}). \tag{15}$$

In this equation, the learning rate ν is present. The learning rate is usually small, e.g., 0.1. With the learning rate, the residuals are reduced step by step. In this way, Gradient Boosting reduces overfitting. The four phases are repeated until the desired number of trees B is reached.

The final approximation $F_B(x)$ is related to the log-odds. This can be inverted to yield probability estimates according to:

$$P(y = 1|x) = \frac{exp(F_B(x))}{1 + exp(F_B(x))} \tag{16}$$

Besides the learning rate ν, several options exist to prevent overfitting. The first is taking a subsample when creating a tree. This procedure looks like that subsample for splitting in a random forest algorithm as explained in Section 3.3. If sampling is performed, the same problem arises as with a Random Forest. Due to the fact that trees will predict contradictory outcomes and probabilities close to zero and one will not occur. Second, the same regularization techniques for trees can be used, as controlling the maximum depth or the minimum number of observations present in a leaf.

3.5 Artificial Neural Network

An Artificial Neural Network (ANN) is a complex mathematical model. This model consists out of fully connected neurons. These neurons are mimic the neurons in the human brain. An ANN consists out of three parts, the input layer, one or more hidden layers and the output layer. Every layer is fully connected, these connections mimic the synapses in our brain. An example of an ANN can be seen in Figure 5. In this example, the input layer consists out of four nodes. There are two hidden layers with respectively five and seven nodes. Finally, there is the output layer consisting out of three nodes. Each layer is connected with a weight matrix. This weight matrix is combined with the neuron values to compute the neuron values in the next layer.

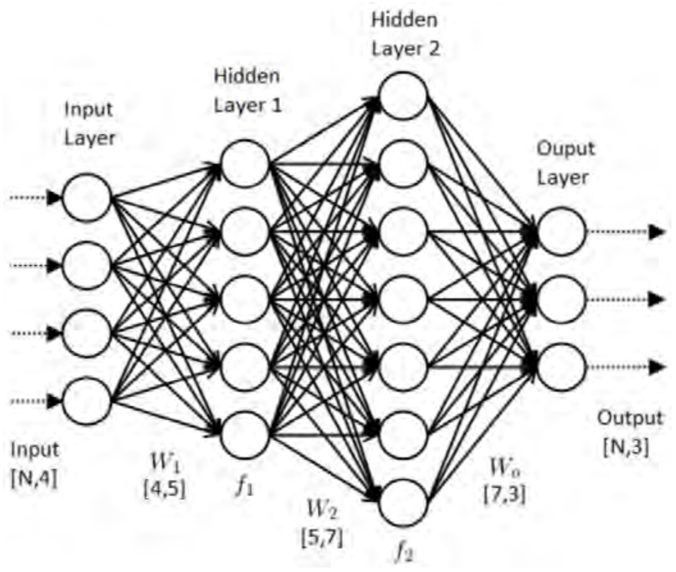


Figure 5: An example of an Artificial Neural Network.
Source: `https://medium.com/coinmonks/the-artificial-neural-networks-handbook-part-1-f9ceb0e376b4`

The neural network in Figure 5 is called a Multilayer Perceptron (MLP). An MLP is a class of feedforward neural networks and consists of multiple linear perceptrons in combination with activation functions. A perceptron consists of one neuron and can be seen in Figure 6.

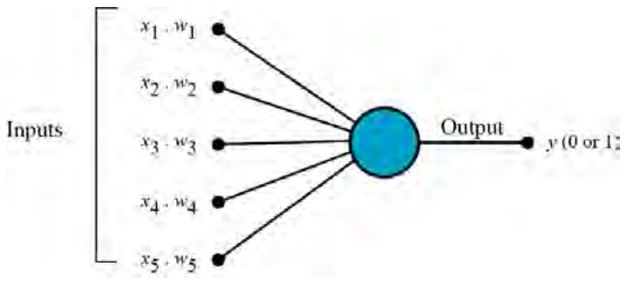


Figure 6: An example of a linear perceptron.
Source: `https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53`

In a linear perceptron, no activation function is used. If this is the case, an MLP is not able to capture nonlinear relationships. The standard activation function is the logistic activation function:

$$\phi(z) = \frac{1}{1 + exp(-z)} \tag{17}$$

An advantage of the sigmoid activation function is the ability to bound the output between zero and one. This is an important property if an MLP is used for classification. However, if z is large, either positive or negative, the gradient of this function will be very small. If this occurs, it is called vanishing gradients. The problem of vanishing gradients is that the gradient is so small, or even zero, that the perceptron or MLP does not learn or optimizes slowly. Therefore, other activation functions were created. One of these activation functions is the Rectified Linear Units (ReLU) [18]. It is a popular replacement for the sigmoid activation function. The ReLU function can be described as:

$$f(z) = z^+ = max(0, z). \tag{18}$$

There is one problem with ReLU. In Equation (18), it can be seen that all values equal to or smaller than zero are treated the same. Therefore, LeakyReLU was invented [15]. LeakyReLU can be mathematically defined as:

$$f(z) = \begin{cases} z, & \text{if } z \geq 0, \\ \alpha \times z, & \text{otherwise.} \end{cases} \tag{19}$$

Here α is a scaling factor. The scaling factor is often smaller than one. LeakyReLU allows small gradients where z is equal to zero, where ReLU returns a gradient of zero.

The linear perceptron of Figure 6 can be mathematically written as:

$$z = b + \sum_{i=1}^{5} w_i \times x_i. \tag{20}$$

New in this equation is the bias b. With the bias b, the activation function can be shifted such that it will better fit the data. If multiple perceptrons are combined into an MLP, the weights

w_i in Equation (20), combine into one matrix $W^{[l]}$. This matrix represents all the connections between two layers. When training an MLP, these matrices are optimized. For this optimization, the matrices need to be initialized. The initialization process that is used is called the Glorot Uniform initialization process [10].

Such an optimization is done using backpropagation. Backpropagation stands for backward propagation of errors. It is an iterative process that optimizes the weights matrices $W^{[l]}$, for each layer l. At each iteration, the error is propagated backwards through the network using gradient descent. However, the search space is not always convex. Therefore, several implementations exist. These implementations use some form of stochastic optimization so that backpropagation will not get stuck in local optima. In this research, the Adam optimizer is used [12].

Like any other machine learning model, an ANN can overfit. A way to prevent this is by using dropout [25]. The idea is that large weights in an ANN are a sign of overfitting. Therefore, shrinking weights will prevent overfitting.

3.6 Bayesian Optimization of Hyperparameters

The goal of machine learning algorithms is to find a model that maximizes the performance on a training set. However, these algorithms have settings that influence the performance of a model. Such settings are called hyperparameters. Examples of hyperparameters are the number of trees, B, used in a Gradient Boosting model or the number of hidden layers in a neural network.

The aim of hyperparameter optimization is to find hyperparameters of an algorithm that maximizes performance on the validation set. In this research, cross validation is used in order to create a good estimation of the model's generalization on the data. However, this makes that evaluating a set of hyperparameters is computationally expensive. Every time hyperparameters are evaluated, several new models are trained. Methods like grid and random search are often used to find good hyperparameters. These methods are uninformed and do not know the performance of previous evaluations. Therefore, these methods try a lot of hyperparameter settings that will perform well.

An example of an informed search is Bayesian optimization [24]. The idea behind Bayesian optimization is spending more effort in deciding which hyperparameters to test in the next evaluation. Therefore, Bayesian optimization is more efficient since the method chooses the next hyperparameters in an informed manner.

In the optimization process, a probabilistic model is formed that maps hyperparameters to a probability of a score on the objective function. This function is defined by $P(score|hyperparameters)$. This model is called the surrogate for the objective function. With this knowledge, the algorithm can be written as in Algorithm 3.

Algorithm 3: General algorithm for Bayesian optimization

```
Build a surrogate probability of the objective function;
while $i \leq MaxIter$ do
    Search hyperparameters that perform best on the surrogate;
    Apply hyperparameters;
    Update surrogate model with new results;
end
```

There are two remaining challenges in this algorithm. The first is the search of the hyperparameters that perform well. In this research, this is done according to the Expected Improvement(EI). The second challenge is how the surrogate model can be built. A commonly used method is the Tree Parzen Estimator(TPE) [5].

In order to search the hyperparameter space efficiently, a possible distribution of each hyperparameter is defined. Such a distribution can be specific for each hyperparameter. In this research, the distributions are based on the model documentation and on domain knowledge.

In the optimization process, the optimization target has a big influence on the results. In this research, it is important that a good distinction can be made between good and bad clients. Therefore, the AUC score has been chosen as optimization metric and will be maximized. This score will be further explained in Section 3.8.2.

3.7 Rating system calibration

As explained in Section 2.4.3, balancing the data set has a warping effect on the posterior probability. Therefore, the models are predicting the likeliness someone will default instead of the PD. In order to transform the score into the PD, a rating system will be created [20].

A credit rating system is based on a ranking system. A rank in this system corresponds to some level of creditworthiness. A high credit rating is equal with a low PD where a low credit rating corresponds to a high PD. This means that a credit score can relate to the PD.

The calibration of such a system, is the mapping process under which each score from a model is matched to a credit rating. The mapping process consists of several steps. At first, all clients in the training set are ranked on their predictions of a model. Second, these clients are divided over G bins. All these bins are equal in size. Suppose, 1,000 clients are available and 10 bins are used, i.e., $G = 10$. This means that each bin contains $\frac{1000}{10} = 100$ clients when constructing the bins. After the division is made, the lowest and the highest score in a bin will form the boundaries for that bin.

In practice, the last bin is often reserved for a PD of one even if there are no clients that have such a PD. In this research that will be omitted since it does not affect any analysis.

When all clients are divided, the actual PD in each bin will be calculated according to the formula:

$$PD = \frac{Observed\,Defaults}{All\,Observations}. \tag{21}$$

All G probabilities will be used as estimation for the clients that land in a specific bin. All probabilities can be combined into one PD curve.

Finally, the quality of the rating system can be evaluated. First, the clients in the test set are divided over all bins. However, they are not ranked but the binning boundaries created on the training set are used. Then again, the PD is calculated for each bin according to (21). This will result in another PD curve, this time an out-of-sample curve. The difference between the in-sample and the out-of-sample curve can then be used as a performance measure of the quality of the PD estimations. As a metric, the Root Mean Squared Error(RMSE) is used. This will be further explained in Section 3.8.3.

3.8 Model Evaluation

In order to make a fair comparison between several models, proper model evaluation is necessary. One of the fundamentals of model evaluation for classification is the confusion matrix. The confusion matrix enables the visualization of the model's performance in a table. An example of a confusion matrix can be found in Table 8. In this table, TP stands for True Positives, FN for False Negatives, FP for False positives and TN for True Negatives.

		Predicted label	
		No Default	Default
True label	No Default	TP	FN
	Default	FP	TN

Table 8: Template of a confusion matrix

A lot of performance measurements can be taken out of the confusion matrix. Two important ones are precision and recall. Precision gives the proportion of clients that are predicted to default that actually defaulted. Mathematically, this can be defined as:

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{22}$$

Recall measures the proportion of clients that defaulted were actually predicted as a default. This can be defined by the following:

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{23}$$

Recall gives the information on how many defaults were missed. Sometimes, the recall is referred to as the True Positive Rate(TPR). So, predicting only defaults will lead to a recall of 1. On the opposite, precision measures how many of these predicted defaults were actually defaults.

Another useful measure that can be calculated out of the confusion matrix is the specificity. The specificity measures the proportion of the correctly predicted defaults from the population of defaults. The measure is defined by:

$$\text{Specificity} = \frac{FP}{TN + FP}. \tag{24}$$

With the specificity, the False Positive Rate(FPR) can be defined by taking $1 - \text{Specificity}$.

One very useful performance metric for a PD model takes the cost of default into account. Such a metric gives a perfect insight in the risk of model. Such a metric can be used if the cost of default is known for each instance. With this knowledge, a model can take a decision with taking the likeliness of a default and the costs of it into account. Unfortunately, such data was not available in this research.

3.8.1 F1-Score

It is clear that one wants both precision and recall as high as possible. However, optimizing on two measures is not practical. Therefore, a combination of precision and recall is desired. Such a combination is called the F1 score. The F1 score combines the precision and recall using the harmonic mean and can be defined as follows:

$$\text{F1 score} = \frac{2 \times \text{Precision} \times \text{Recall}}{(\text{Precision} + \text{Recall})}. \tag{25}$$

The score calculates the accuracy for a certain label, instead of the total accuracy. This means that the F1 score can be calculated for both labels yielding different results. As it is desirable to catch as many defaults, the F1 score for the label default will be given.

3.8.2 AUC-ROC Curve

Together, the FPR and the TPR (recall) combine into the AUC-ROC curve. The AUC-ROC curve tells how much the model is capable of distinguishing between classes. The curve represents a trade-off between the TPR and the FPR, an example of the curve can be found in Figure 7.

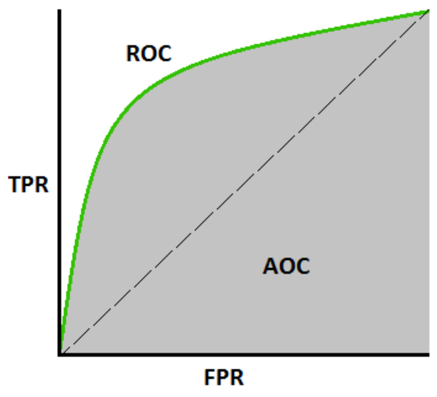


Figure 7: An example of the AUC-ROC curve.
Source: `https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5`

However, a curve is not a useful metric. Fortunately, the area under the curve (AUC) can be calculated. The higher the AUC, the better the model distinguishes the classes. An AUC of 0.5 is the worst situation. It happens when a model is not able to learn anything. The AUC can be smaller than 0.5, this only happens if the model learns the opposite of what it should.

3.8.3 RMSE

With the procedure described in Section 3.7, it can be computed how well the rating systems, and therefore the models, perform. This will be done using the Root Mean Squared Error (RMSE). The RMSE can be defined as follows:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{G}(\widehat{PD}_i - PD_i)^2}{G}}, \tag{26}$$

where G is again the number of bins. Furthermore, $\widehat{PD}_i$ is the PD estimated on bin i with the training set and PD_i is the observed PD on the test set.

It is obvious that mistakes are undesirable. However, big mistakes have larger consequences than small mistakes. The RMSE penalizes large mistakes more than small mistakes. Therefore, the RMSE is considered a good measurement to compute the differences between the PD curves.

4 Experimental Setup

In this section, the experimental setup will be discussed. First, the training and the test setup are discussed. Thereafter all settings of the hyperparameters are given. All this research is conducted in the programming language Python.

4.1 Training and test set up

In order to train and evaluate the model, a split in the data set is necessary. At first, the data set is split into a training and test set. The training data is used to train and tune the models. The test set is only used to evaluate the models.

The training data contains 85% of the data, whereas the test data consists of the other 15% of the data. The train-test ratio is rather high. However, the training data still needs to be balanced after this step. Therefore, the initial split ratio is set high. However, after balancing the train-test ratio is small, i.e., $58\% - 42\%$. Unfortunately, the initial train-test ratio cannot be adapted. If the initial ratio is changed, proper testing of the rating system will be difficult.

4.1.1 Model setup

In model training, the training data is utilized. At first, the training data is sampled in order to balance the data set as described in Section 2.4.3. After this operation, the hyperparameters will be tuned using cross-validation as described in the sections below. After all hyperparameters are tuned, the models will be trained using the full balanced training set. These models are the final models and the final performance is tested using the test set. Note that the test set is not balanced and represents the true distribution.

4.1.2 Credit rating setup

In the development of the credit rating system, the whole training set is utilized. The training set is not balanced. In the creation of the system, the training data should represent the true distribution. After the system is created, the PD estimations for each bin are evaluated with the observed PD in the test set.

4.2 Logistic Regression

The implementation used for Logistic Regression, is the scikit-learn implementation [19]. Scikit-learn is one of the largest open-source libraries for machine learning in Python. It implements regularized Logistic Regression using the Limited memory BFGS method.

As regularization method, L2 regularization is used. Furthermore, to control computational time but still have convergence of the optimization method, a maximum number of iterations is used. An overview of the settings can be seen in Table 9.

Hyperparameter	**Value**
Regularization	L2
Max iterations	750

Table 9: Settings for Logistic Regression

4.3 Random Forest

The most used Random Forest implementation in Python is the algorithm of scikit-learn [19]. Scikit-learn is one of the largest open-source libraries for machine learning in Python.

As with any other algorithm, Random Forest is sensitive to its hyperparameters. Not all hyperparameters are optimized to reduce the complexity of the optimization process. The hyperparameters that are optimized can be found in Table 10. The q in front of the distribution means that the distribution is quantized and a step size is used in order to reduce computational times.

Hyperparameter	**Influence**	**Distribution**	**Range**	**Step size**
Number of estimators	Number of trees	QUniform	(50, 750)	25
Max features	Max features considered when splitting a node	QUniform	(40, 400)	20
Max depth	Max depth of a tree	QUniform	(1, 150)	5
Min samples split	Min number points placed in a split	QUniform	(2, 50)	5
Min samples leaf	Min number points in a lead	QUniform	(1, 50)	5

Table 10: Settings for Bayesian optimization for LGBM

In general, the Random Forest will learn better if the first three hyperparameters are large and the last two small. However, this can increase computational time drastically or cause overfitting. Therefore, these hyperparameters are optimized following the procedure explained in Section 3.6, with a 3-fold cross validation and 100 iterations. The best performing settings found are shown in Table 11.

Hyperparameter	**Value**
Number of estimators	500
Max features	160
Max depth	50
Min samples split	15
Min samples leaf	5

Table 11: Hyperparameters found by Bayesian optimization for a Random Forest

4.4 Gradient Boosting

In this research the Light Gradient Boosting Machine (LGBM) implementation of Microsoft [16] is used. LGBM is a gradient boosting algorithm designed for faster training speed and high accuracy. It has a low memory usage. This is of great use due to the size of the data set, i.e., 802 Megabyte.

The hyperparameters are optimized according to the procedure in Section 3.6, with a 3-fold cross validation and 100 iterations. The hyperparameters are optimized, their influence on the model and their range can be found in Table 12. If there is a q in front of the distribution, the distribution is quantized and a step size is used in order to reduce computational times. If that is the case, the step size columns will indicate what the step size is.

Hyperparameter	**Influence**	**Distribution**	**Range**	**Step size**
Number of leaves	Maximum number of leaves	QUniform	(5, 50)	5
Learning rate	Regularization factor	LogUniform	(log(0.005), log(0.5))	-
Subsample for bin	Sample size for binning	QUniform	(20,000, 240,000)	20,000
Reg α	Regularization factor	Uniform	(0, 1)	-
Reg λ	Regularization factor	Uniform	(0, 1)	-
Colsample by tree	Which features are used	Uniform	(0.6, 1)	-
Subsample	Subsample size of the data	Uniform	(0.5, 1)	-
Min child samples	Minimal amount of data in a leave	QUniform	(20, 500)	5

Table 12: Settings for Bayesian optimization for LGBM

The hyperparameter settings that performed best in the Bayesian optimization can be found in Table 13.

Hyperparameter	**Value**
Number of estimators	3077
Number of leaves	20
Learning rate	0.007770
Subsample for bin	160,000
Reg alpha	0.715208
Reg lambda	0.532382
Colsample by tree	0.617681
Subsample	0.900419
Min child samples	345

Table 13: Hyperparameters found by Bayesian optimization by LGBM

4.5 Neural Network

As Python implementation, the Keras framework is used [8]. An ANN is highly sensitive to its hyperparameters. A distinction is made between the structure of the ANN and the optimization method(Adam). In order to keep the hyperparameter space manageable, the default parameters of the Adam optimizer are used. Furthermore, not all components in the ANN's structure are optimized. The activation functions for the hidden layers are static. All the hyperparameters that are optimized and their distributions can be found in Table 14.

Hyperparameter	**Influence**	**Distribution**	**Range**	**Step size**
Dropout	Regularization factor	Uniform	(0, 1)	-
Number of nodes	Nodes for layer l	QUniform	(1, 500)	10
Activation function	For the first $l-1$ layers	-	Leaky ReLU	-
Activation function	For the output layer	-	Sigmoid	-

Table 14: Settings for Bayesian optimization for an ANN

For LeakyRelu, the α is set on 0.25. The hyperparameters found after 100 iterations with 3-fold cross validation can be seen in Table 15.

Hyperparameter	**Value**
Number of epochs	20
Dropout	0.6475
Hidden layers	3
Nodes layer 1	300
Nodes layer 2	160
Nodes layer 3	200

Table 15: Neural network results

5 Model performance

In this section, the performance of all models is reported. A distinction is made between two types of performance. The first is the classification performance. The second is the quality of the PD estimation. Finally, the performance of all models are compared and the best model will be picked.

5.1 Classification performance

The classification performance measures the client discrimination ability of the model. The client discrimination ability tells how well a model is able to distinguish good clients from bad clients. If the classification performance is acceptable, a model can be used as a client acceptance model. A confusion matrix allows visualization of the performance of a classification model. The visualization is a matrix that shows the correct and incorrect labeled instances. The confusion matrices can be seen in Figure 8.

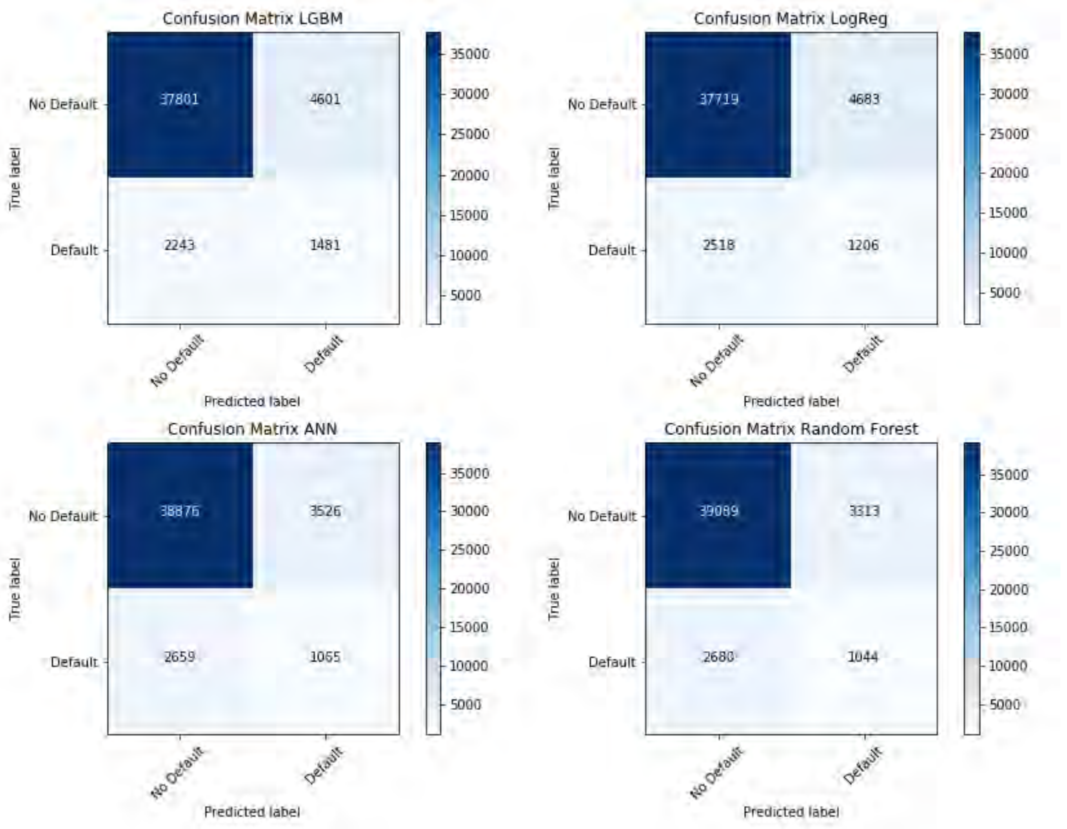


Figure 8: Confusion matrices of all models

The numbers in the confusion matrix above are absolute. Therefore, it is not easy to compare them quickly. In Figure 9, the normalized confusion matrices can be seen for all models.

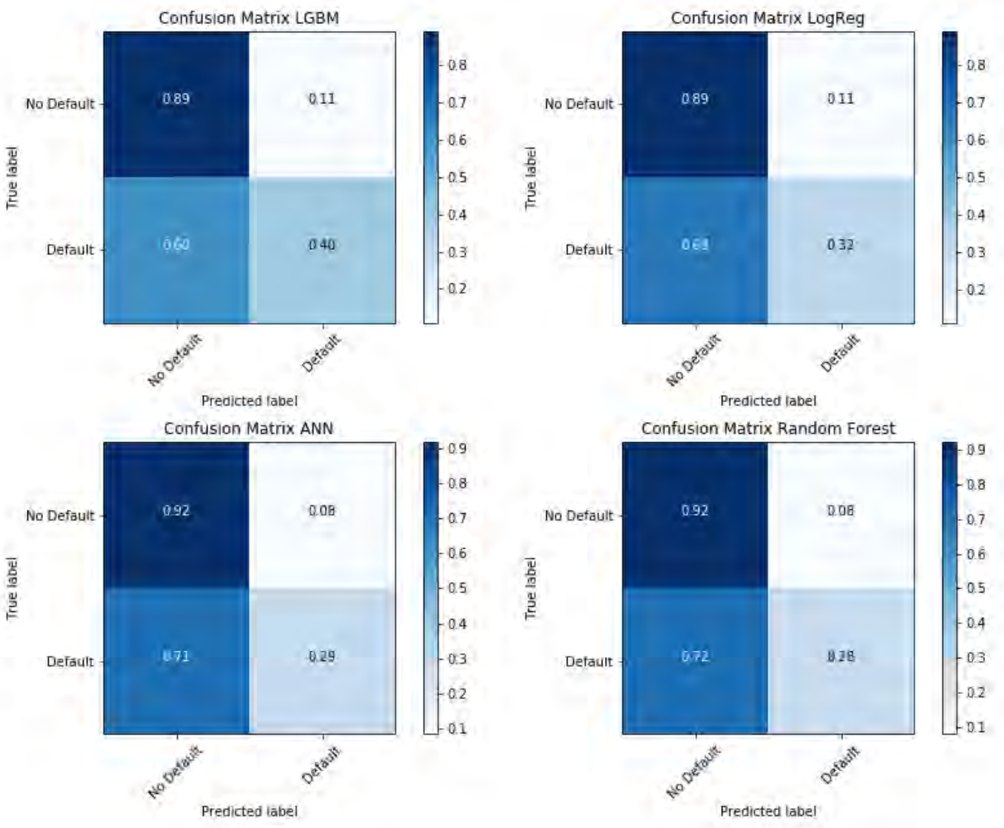


Figure 9: Normalized confusion matrices of all models

The performance metrics used for the classification performance are the AUC and the F1 score. In Table 16, the metrics are shown for all models.

Model	**AUC**	**F1**
LGBM	0.7565	0.3021
LogReg	0.7208	0.2509
Neural network	0.7278	0.2562
Random forest	0.7241	0.2584

Table 16: Performance of all models

5.2 Probability to Default estimation

In this section, the quality of the rating system is evaluated. The PD can be evaluated for the whole portfolio, but also for each group in the rating system. In Figure 10, the PD curves can be found. In this analysis, the last bin where the PD should be one is left out. Since it does not influence performance of the model.

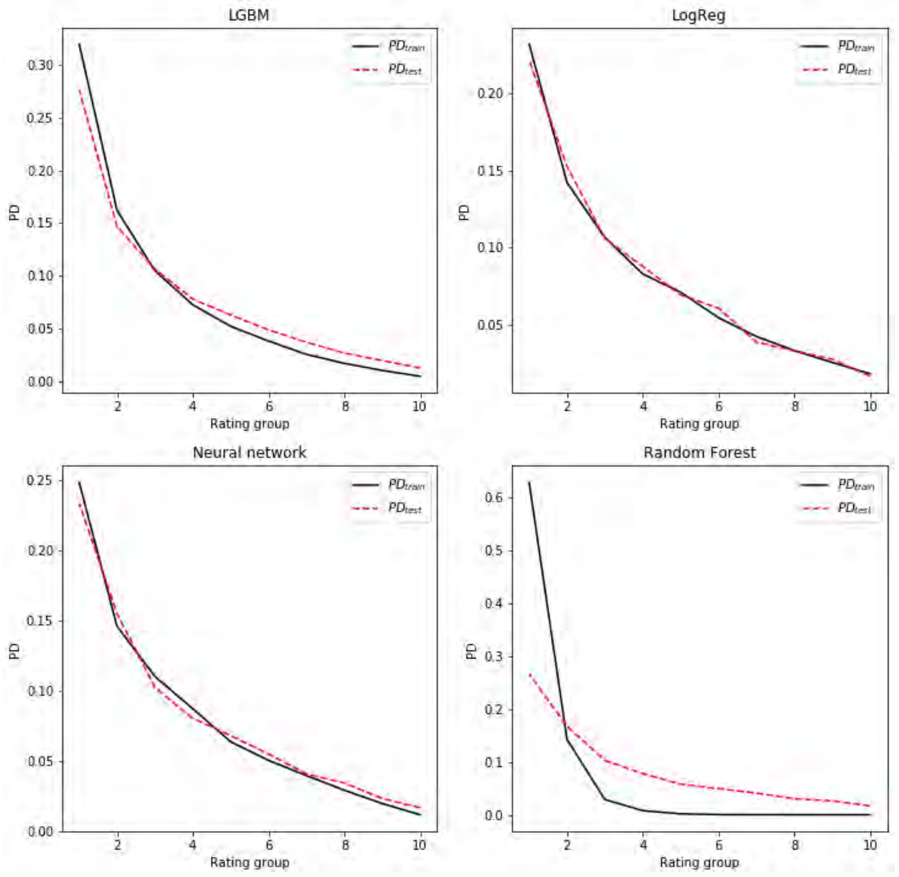


Figure 10: PD curves of all models

The RMSE between these curves can express the quality of the calibration. Since the PD_{train} represents the predicted curve and PD_{test} represents the observed curve. The RMSE of all curves can be found in Table 17.

Model	**RMSE**
LGBM	0.01649
LogReg	0.00551
Neural network	0.00711
Random forest	0.12268

Table 17: RMSE of the curves for all models

If the PD estimations for all bins are combined, the portfolio PD estimation can be calculated. The estimations on portfolio level can be seen in Table 18.

Model	PD_{train}	PD_{test}
Observed	8.0730%	8.0735%
LGBM	8.0732%	8.1599%
LogReg	8.0722%	8.1272%
Neural network	8.0726%	8.1041%
Random forest	8.0734%	8.3340%

Table 18: PD estimation on portfolio level for all models

In Appendix B, the resulting credit rating systems for every model can be found.

5.3 Interpretation of performance

In this section, the results shown in the previous two sections will be discussed. For this discussion, it is important to keep the business in mind. As a reminder, a PD model is mostly used as a client acceptance model and a capital calculation model. For an acceptance model, the discriminatory power is important. In other words, the model should be able to distinguish good from bad clients. For capital calculations, it is important that PD estimations are accurate on a portfolio level. Therefore, the results in the previous sections must be combined in order to determine the best model.

In Section 5.1 the classification performance is shown. It can be seen that the LGBM clearly outperforms all other models. Looking at both metrics and the confusion matrix, it can be concluded that this model is the best in distinguishing good from bad clients. Furthermore, all other models perform similar. Therefore, it cannot be determined yet which models to discard yet.

In Section 5.2, the performance on portfolio level can be seen. It can be seen that the neural network and the logistic regression are well performing on portfolio level. The gradient boosting model slightly worse. However, the random forest performs the worst of all and is therefore discarded from this point on.

As the gradient boosting model is performing well on both business tasks, this model is deemed as the best performing model and should therefore be used. Both the logistic regression and the neural network are performing similar on both domains. Therefore, the conclusion can be made on which of these two models are performing better. However, because the complexity of the logistic regression is much lower than the complexity of the neural network, the logistic regression is preferred above the neural network.

A remarkable observation in the rating systems, are the tails from the tree ensembles. These tails occur because of the structure of the models. During the learning process, subsets of the data are taken. This leads to uncorrelated trees. In other words, trees that predict the opposite. Because of this, predictions close to zero or one rarely occur. This leads to the predictions being centered around the average probability in the data set.

6 Transparency

In this section, two transparency methods are discussed. An important distinction in transparency methods is the difference between model agnostic and model specific methods. model agnostic methods can be used for any type of models. model specific can only be used for a specific model, e.g., a neural network. First, a well-known global method for tree ensembles will be discussed. This method will sometimes be used as a reference method. After this, model agnostic frameworks like LIME and SHAP will be discussed. The focus will be on these two frameworks as they are the most used methods.

6.1 Tree ensembles

A decision tree can be easily interpreted. If one starts at the top of the tree and follows the path until the terminal node, each node can be connected by an "AND" statement. When trees will get larger such interpretation will be much more complex. Furthermore, an ensemble of trees cannot be interpreted this way. However, there are other ways to interpret a decision tree, such as feature importance method described below.

In a decision tree, the overall feature importance can be determined. There are several ways to calculate a feature's importance. The idea of feature importance is to compute the attribution of features to the model. In general, two methods to determine feature importance are used. The first is the split method. This method counts the occurrence of a feature over all splits. The second method is the gain method. This method calculates for each split the gain in accuracy. A disadvantage of these methods, is that no information is given if the influence is positive or negative. The only interpretation achieved with feature importance, is that the identification of important features.

6.2 LIME

One of the most used model agnostic transparency methods is Local Interpretable model agnostic Explanations (LIME) [22]. LIME creates a local interpretable surrogate model. A surrogate model is an approximation of the underlying black box model. A surrogate model can approximate the black box model globally or locally. LIME creates such an approximation locally, i.e., around a specific instance. The interpretable surrogate model is turned to an explanation for that specific instance. This is done by interpreting, in the case of a linear model, its coefficients. With these explanations, LIME tries to provide local transparency. By combining multiple instances across the input space, LIME tries to provide global transparency.

An explanation is a description of which features influences the model's prediction on a certain instance x_i. The process of generating an explanation is developing an interpretable model on a locally defined data set. Examples of interpretable models are linear models, short decision trees and decision rules. The current LIME interpretation focuses on the development of linear models.

Mathematically, LIME creates an explanation by solving the following equation:

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \, \mathcal{L}(f, g, \pi_x) + \Omega(g), \tag{27}$$

where f is the black box model to be explained, G is a class of local models, i.e., linear regression models. Furthermore, π_x is a proximity measure, sometimes referred to as the kernel, between instances. This proximity measure defines the locality of the explanation. $\Omega(g)$ is a complexity measure of g. This complexity measure is static and defined upfront. With linear surrogate models, the complexity of the model relates to the number of variables the model is allowed to use. By setting this value small, LIME generates sparse surrogate models.

In words, Equation (27) can be described as the search for an interpretable model while achieving local accuracy and keeping the complexity of the linear model low. The definition of the loss function is important and can be described as the locally weight square loss:

$$\mathcal{L}(f, g, \pi_x) = \sum_{z \in Z} \pi_x(z)(f(z) - g(z))^2. \tag{28}$$

In this equation Z is the set of instances in the locally defined neighborhood and is sampled around the instance x_i. Since the explanation is a local model, errors close to x_i need to be penalized harder. The proximity measure π_x takes care of this and thereby defines the locality of the surrogate model. Important to note is that the locality of the surrogate model has a large influence on the explanation. The proximity measure is defined by the exponential kernel:

$$\pi_x(z) = exp(\frac{-D(x,z)^2}{\sigma^2}), \tag{29}$$

where D is a distance metric and σ is the kernel width. The distance metric that is often used is the Euclidean distance:

$$D(x,z) = \sqrt{\sum_{i}^{M}(x_i, z_i)^2}, \tag{30}$$

where M is the number of features. However, using the Euclidean distance metric could have some implications. For example, the Euclidean distance has difficulties in distinguishing close by instance from far away instances in a high dimensional space [3]. However, the impact of this is unclear. Furthermore, unstandardized data can corrupt a distance metric. The impact of this will be investigated in Section 7.3.

In Algorithm 4, the complete process of creating an explanation can be seen. Important to note is that Z is a set. Therefore, the addition of the vector $\langle z_i, f(z_i), \pi_x(z_i)\rangle$ can be interpreted as the addition of that vector at the end of the set. Meaning that the elements of Z will contain of N elements and the elements will look like $\langle z_i, f(z_i), \pi_x(z_i)\rangle$.

Algorithm 4: Algorithm for sparse explanation generation with LIME

```
$Z \leftarrow \{\}$;
for $i$ = 1, ..., $N$ do
    $z_i \leftarrow$ sample around $x_i$;
    $Z \leftarrow Z + \langle z_i, f(z_i), \pi_x(z_i)\rangle$ ;
end
$g \leftarrow K$-Lasso($Z$, $K$) ;
return $w_g$
```

One of the steps in the algorithm, is the sampling around the instance x_i. LIME treats numerical and categorical features differently. Numerical features are sampled around a Gaussian distribution, where the mean and standard deviation is taken out of a training set. Categorical features are sampled around their frequency, again this statistic is taken out of a background data set.

A feature selection method is used to comply with the complexity $\Omega(g)$. An example of such a method is the K-Lasso regression. The training pairs for the regression are of the form $(z_i, f(z_i))$ and they are weighted according to the proximity measure π_x. The number of features of z_i that are allowed to be used is defined by K. With this number, LIME is able to produce sparse explanations even if the dimensionality of the underlying data set is high. The number K needs to be specified at the upfront.

For the explanation to be true, the resulting linear model should be locally faithful. Fortunately, goodness-of-fit statistics can be calculated on Z. The model created by LIME can produce several statistics. The most important statistic is the R^2 score. The R^2 score is the proportion of variance in the target variable that is accounted for by the model. However, in LIME the weighted R^2 is used. This means that the score of an explanation, and the explanation itself, are dependent on the locality of the regression. Therefore, the kernel width σ should be chosen carefully. However, choosing σ while optimizing on the R^2 score can be dangerous. In such optimization process, the intercept of the regression could take over the model. If this happens, the coefficients in the model will become very small. This leads to an explanation that is not useful. An interpretation of the intercept is not possible. Unfortunately, no other approach to finding a correct σ is known. That leads to the fact that σ should be carefully evaluated by the user until the explanations make sense and some reasonable R^2 is achieved.

A conclusion drawn from this, is that the correct definition of the neighborhood Z is unclear and could be different for different models and other data sets. The implications of this can be seen in Figure 11. In this figure, the lines show the local models for each kernel width. It can be seen that the local models, and therefore the explanations, are very different. A solution to avoid the kernel width problem can be to change the kernel. This will be further explained in Section 6.3.

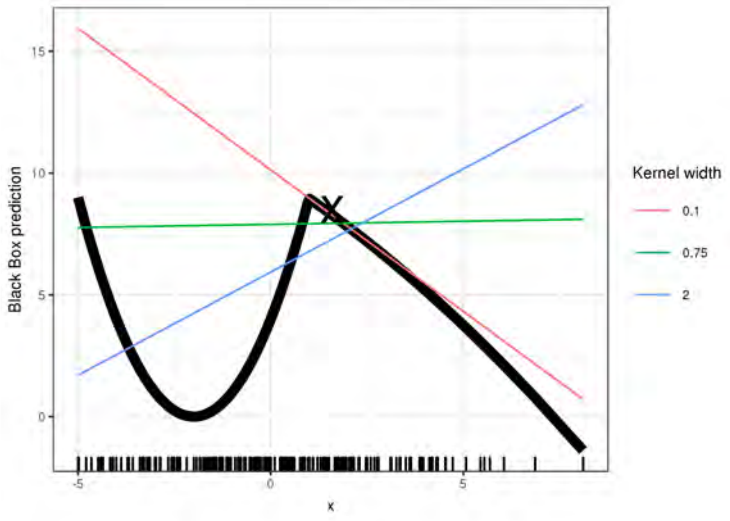


Figure 11: The quality of the local model is highly dependent on the kernel width
Source: `https://christophm.github.io/interpretable-ml-book/lime.html`

6.3 SHAP

Shapley Additive Explanations(SHAP) is a unified approach for interpreting model predictions [14]. SHAP assigns each feature an attribution value for a certain prediction. With these attribution values, explanations are generated. These explanations can be generated for every model, so SHAP can be called model agnostic. However, calculating such explanations can be computationally expensive. Therefore, several model specific implementations exist to speed up the computational time. In order to make a fair comparison between SHAP and LIME, the model specific versions were left out of scope.

Just as LIME, SHAP is in principal a local method, however SHAP can also provide global interpretation. This will be explained later in this section. Local methods often work with simplified inputs x' that maps to x. This mapping is done with a mapping function $h_x(x') = x$. This mapping value can be defined as:

$$h(x'_i) = \begin{cases} 1 & \text{if original feature value } x_i \text{ is included,} \\ 0 & \text{if } x_i \text{ is toggled off.} \end{cases} \tag{31}$$

Local methods try to ensure $g(z') \approx f(h_x(z'))$, where $z' \approx x'$, meaning that the local model g should be accurate to the black box model f.

SHAP is a framework of additive feature attribution methods. Such methods have a local model that is a linear function of binary variables:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i. \tag{32}$$

In this function, M is the number of simplified features and $z' \in \{0,1\}^M$. The symbol ϕ is the attribution effect to each feature and is defined as $\phi_i \in \mathbb{R}$. An important variable in this equation is ϕ_0. This variable represents when all the simplified inputs are toggled off, i.e., missing. This can be interpreted as the base value of the model and is predicted if none of the features are known, e.g., taking the average as predicted value.

An important property of SHAP is local accuracy. Local accuracy ensures that the explanation model matches the original model. This can be expressed by:

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i \tag{33}$$

Other properties the SHAP framework has are missingness and consistency. The missingness property states that if a feature is not present in the simplified input, it should not achieve any value. Mathematically, this can be written as:

$$x'_i = 0 \longrightarrow \phi_i = 0 \tag{34}$$

The last property is called consistency. Consistency states that if a model changes so that a feature contribution increases, then the feature's attribution should increase too.

The attribution effect ϕ_i is called the SHAP value. These values are the Shapley values of a conditional expectation function of the original model. Shapley values are taken from game theory [23]. In a game with a payout, the Shapley value gives the information how to distribute the payout fairly among the players. In feature attribution, the payout is linked to the prediction and the players are linked to the features. This means that the goal of SHAP is to distribute the attribution to a prediction fairly over all features. An advantage of this, is that the SHAP framework benefits from the research that has been done in the field of Shapley values. This makes that the framework has a strong theoretical foundation.

As mentioned before, the SHAP framework is able to provide more than local transparency. As said before, the SHAP value is unique for a certain instance. Meaning that, the SHAP value for a specific variable is dependent on the value of all other variables in the instance. This property brings a useful addition to the framework. Consider a set of 1,000 instances with 10 variables, x, and generate explanations with SHAP for all instances. The result is a matrix containing 1,000 rows and 10 columns with a SHAP value for each situation. If the global behavior of a variable is desired, one can visualize all the SHAP values for that variable. Such a visualization is called a SHAP Dependence Plot. In such a figure, the variable value is plotted against the SHAP value. In this visualization, several situations can occur. The first is horizontal dispersion of variable values. This contains the information how a feature influences the prediction. The second is vertical dispersion, vertical dispersion occurs when equal variable values have different SHAP values. This means that the variable is interacting with other variables in that instance. Investigating vertical dispersion can give information about the interaction effect between two variables and helps discovering relationships in the data set. With such combinations of SHAP values, local explanations can be combined to give global transparency.

The model agnostic version is the KernelSHAP. KernelSHAP combines the Shapley values with the LIME method described in Section 6.2. According to Lundberg et al. [14], LIME should have different settings in order to satisfy the properties described above. The settings should be the following:

$$\begin{aligned}\Omega(g) &= 0,\\ \pi_x(z') &= \frac{(M-1)}{\binom{M}{|z'|}|z'|(M-|z'|)},\\ \mathcal{L}(f,g,\pi_x) &= \sum_{z'\,\in\,Z} \pi_x(z')(f(z')-g(z'))^2,\end{aligned} \tag{35}$$

where $|z'|$ is the number of non-zero elements in $|z|$. Important to note is that if $|z'|$ is equal to zero or to M, $\pi_x(z')$ is not defined. However, two defined properties will help dealing with this problem: $\phi_0 = f_x(\emptyset)$ and $f(x) = \sum_{i=0}^{M}\phi_i$. According to the authors, the undefined proximity weights can be avoided by analytically eliminating two variables using these constraints.

Furthermore, the complexity measure $\Omega(g)$ is defined as zero. As the SHAP method fairly distributes the difference between the prediction and the average over all features. This is a property that it inherits from the classic Shapley values [17]. This means that the complexity of the resulting explanation is not taken into account during the optimization process

The main difference with LIME is the proximity measure $\pi_x(z')$, which is one of the largest theoretical issues in LIME. Furthermore, the sampling method around the instance of interest differs.

The sampling method consists of several steps. The first step is toggling off features using the mapping function described in Equation (34). A feature is off when its value differs from the value in the instance of interest. The value is set to unknown. However, many models cannot handle unknown feature values. Therefore, the value of the feature is sampled from a background data set. This process mimics the unknown feature value.

Furthermore, the proximity measure does not use a distance metric or a kernel width. Instead, it uses the fact that features are toggled off and it will take that into account.

6.4 Theoretical comparison

In this section, a theoretical comparison between SHAP and LIME is made. SHAP and LIME are very similar methods. However, they do differ in several ways. The main difference is the proximity measure. In SHAP the proximity measure is also referred to as the SHAP kernel. In LIME the proximity measure is defined as the exponential kernel. Due to the exponential kernel, LIME is subject to more hyperparameters. Besides the kernel itself as a hyperparameter, the exponential kernel has two additional hyperparameters. The first is the distance metric between two data points. As a distance metric, the Euclidean distance is often used with tabular data. However, the Euclidean metric is known to behave badly with high-dimensional data [3]. As a second hyperparameter, the exponential kernel uses a kernel width to define the size of the local neighborhood. Both the distance and the kernel width adds extra hyperparameters to the LIME method in comparison to SHAP. This makes it more difficult to generate reasonable explanations.

Furthermore, the LIME method has no theoretical justification for the exponential kernel. However, the SHAP kernel connects model interpretation with game theory and thereby gaining theoretical advantage. This also makes that SHAP benefits from research already done in the classic Shapley values.

Second, SHAP does not penalizes complexity as LIME does. The SHAP method looks for a full explanation. In this full explanation, the method tries to identify all features that contributed to the prediction. In LIME, the method looks for sparse explanations. This makes that the number of features desired in the explanation method is another hyperparameter that must be set. Furthermore, with making the explanation sparse, feature selection must be used. The feature selection method used, has big influence on both the generated explanation as the computational time.

Another difference, is the fact that LIME creates a surrogate model that must hold for the locally defined neighborhood. This makes that LIME can give a global overview of the model's behavior by generating multiple explanations for different instances across several local neighborhoods. However, changing hyperparameters can lead to different explanations, this makes it difficult to detect patterns with the explanations. Because different hyperparameters lead to different explanations in the same local region. SHAP does not create a model for a local neighborhood. Instead, it creates an explanation which must be true for that instance according to the theory behind Shapley values. Therefore, these explanations can be combined to show global patterns discovered by the model more easily than LIME.

7 Transparency Analysis

In this section, several analyses on the transparency methods are performed. First, the parameters of the methods and their influence on the explanations is investigated. Second, the impact of standardizing or not standardizing the data will be investigated. Third, the strong predictors that were left out, i.e. the external credit scores, will be added to the model in order to see if the methods are able to detect their importance. Fourth, the ability to detect the most influential variables is investigated. Finally, the methods will be investigated on their ability to give global transparency.

Important to note is that the Random Forest is not taken into account in this analysis. In Section 5, it can be seen that the LGBM outperforms the Random Forest. Since both are tree ensembles, the Random Forest model will not be used in this section.

7.1 LIME parameters

The parameters used in the LIME method have a large influence on the generated explanation. The explanation can change significantly for an instance if the parameters are changed. Several parameters exist, the parameters investigated in this research are the number of features used in the explanation, the kernel width used in Equation (29) and the feature selection method. At first, the feature selection method will be investigated in combination with the number of features and a static kernel width. Second, the kernel width will be adjusted in combination with the number of features and the feature selection method will be static.

A LIME explanation can be assessed in three different ways. First, it can be checked if the explanation is locally faithful. This means that the prediction of the created local model should be close to the prediction of the complex model. This can be defined as the absolute error where the prediction of the complex model is defined as the true value. Furthermore, as a local model should hold for the local neighborhood, the model's performance in this neighborhood can be investigated. This performance is measured by taking the R^2 metric. Finally, the difference between the local model's prediction and its intercept is taken. This difference tells how many importance is actually assigned to the features. Therefore, a small difference is undesired since such an explanation will not give any useful insights. Important is that the intercept is compared with the local model prediction and not the black box model prediction.

7.1.1 Feature selection method analysis

In the LIME implementation several options exist for feature selection. The following list is a summation of these options.

- Forward selection
- Auto
- None
- Lasso path

Forward selection method is a well-known method. This method is an iterative process where, in each iteration, a variable is added that results in the best performing model. The process stops if the number of desired features are reached. Lasso is a linear model with sparse coefficients. In the auto selection method, forward selection is used until the number of desired features exceeds six. After this, a regression model is trained on all features and the variables with the highest coefficients are selected. These variables are then used for the final surrogate model. The none option uses all features and ignores the number of desired features.

In this analysis, 500 explanations are created for each feature selection method. These 500 explanations are divided over 10 different numbers of desired features. This is divided in a range starting at 5 and that ends at 50 with a step size of 5. This analysis has both be done for the LGBM and the developed neural network.

LGBM results

In Figures 12, 13 and 14 the results of the analysis can be seen. One conclusion drawn from the figures is that if more features are used, the quality of the prediction will rise. However, the quality of the explanations will have more variance, according to the rising standard deviation. Making it more uncertain if an explanation will be of good quality. Furthermore, the conclusion can be drawn that the lasso and the forward selection method are the best performing methods. In Figure 15, the computational times can be seen. The forward selection method is computationally expensive in comparison with the Lasso method. Therefore, from now on, the lasso method is used.

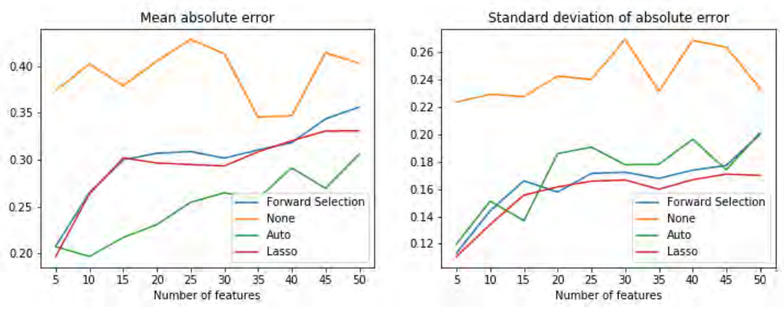


Figure 12: The mean absolute error on 50 explanations using the LGBM as complex model with different feature selection methods

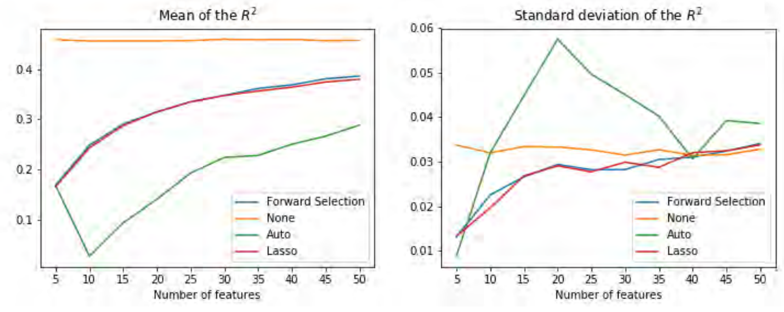


Figure 13: The mean R^2 on 50 explanations using the LGBM as complex model with different feature selection methods

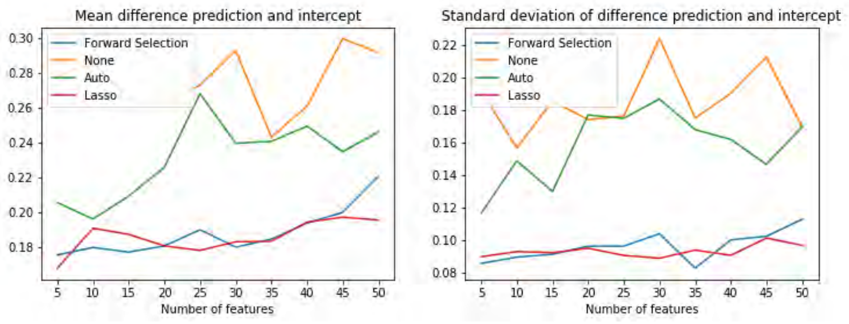


Figure 14: The mean difference using 50 explanations and the LGBM as complex model

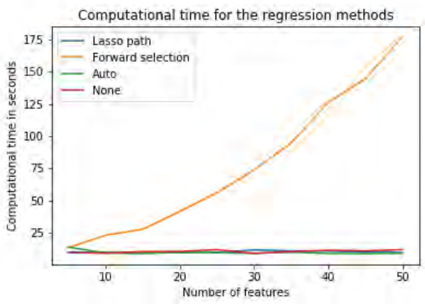


Figure 15: Difference between prediction and intercept

Neural network results

In Figures 16, 17 and 18 the results of the analysis can be seen. The conclusions for the neural network are very similar to the LGBM results. However, it can be seen that the lasso method does not perform similarly to the forward selection method anymore. It can be seen that the lasso method performs significant better on the local neighborhood, with similar R^2 performance. However, it can be seen the difference between the intercept and the prediction is very low.

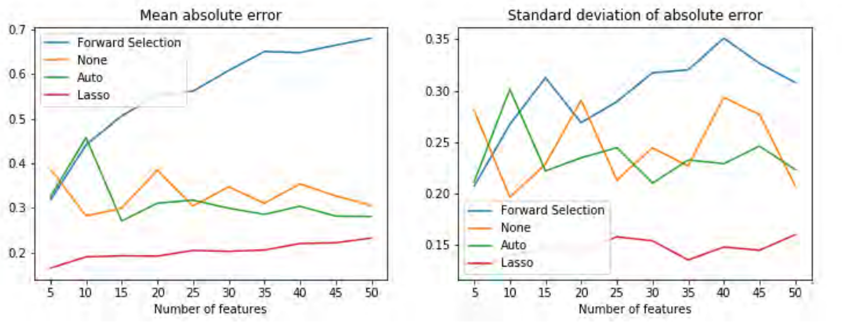


Figure 16: The mean absolute error on 50 explanations using the neural network as complex model with different feature selection methods

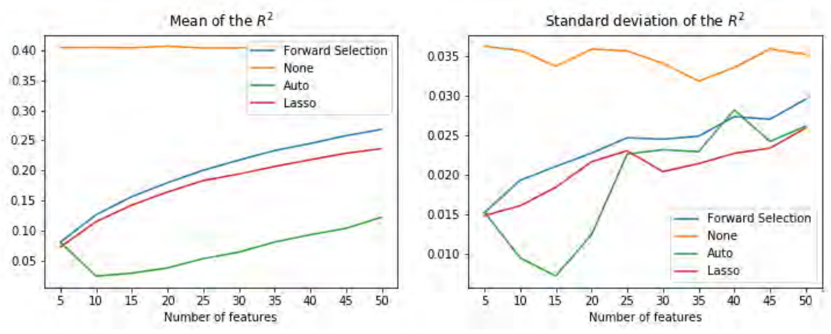


Figure 17: The mean R^2 on 50 explanations using the LGBM as complex model with different feature selection methods

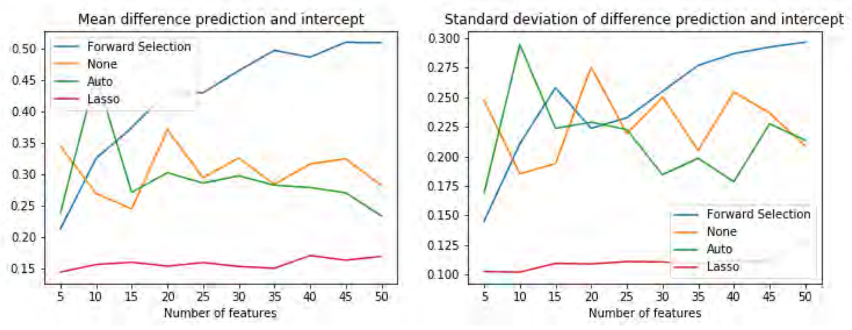


Figure 18: The mean difference between the prediction and the intercept using 50 explanations using the neural network

In Figure 19, the computational times can be seen. The trend seen in the computational times for the LGBM can also be seen in this figure. From now on, the lasso method will be used. One of the reasons is the cheap computational power, this will be useful in other analysis on the method.

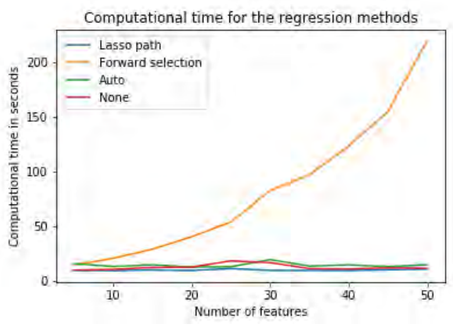


Figure 19: Computational time of all regression methods in seconds

7.1.2 Kernel width analysis

For this analysis, 50 explanations are generated for 50 different settings. These settings consist out of 5 different kernel widths, i.e. 5, 10, 15, 20, 25. Where a kernel width of 15 approximates the default setting of the LIME implementation, i.e., $\sqrt{\text{Number of columns}} \times 0.75$. The number of columns refers to the number of variables used in the model. The range for the number of features used in the explanation starts at 5 and ends at 50 with a step size of 5. Larger values than 50 are undesired, since it increases the complexity of the local model. An increasing complexity means a decrease in interpretability of the local model. Important to note is, that the size of the kernel width did not affect the computational times significantly.

LGBM results

From Figure 20, a trend can be seen. This trend shows that the mean error and its standard deviation is increasing when the number of features used is increasing. In Figure 21, it can be seen that the R^2 score is increasing while the standard deviation stays the same. The conclusion that can be drawn from this, is that there exists a trade-off. This is a trade-off between accuracy on the instance or on the local neighborhood. It shows that if one desires a high accuracy on the instance, a small number of features should be chosen. On the other hand, if one desires a good explanation for the locally defined neighborhood, more features should be added to the explanation. In Figure 22, there is also no relationship between the kernel width and the difference between the intercept and the prediction.

Unfortunately, there is not a clear difference in performance of the different kernel widths. However, a different kernel width highly affects the explanation. This is an issue because there is no clear indication on a reasonable value of the kernel width. From now on, the kernel width will be kept at the default value.

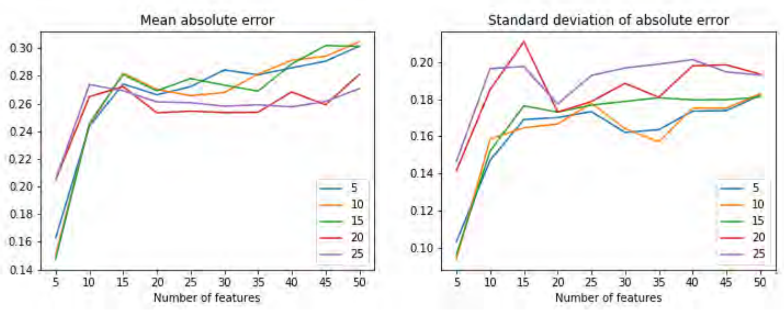


Figure 20: The mean absolute error on 50 explanations using the LGBM as complex model using different kernel widths

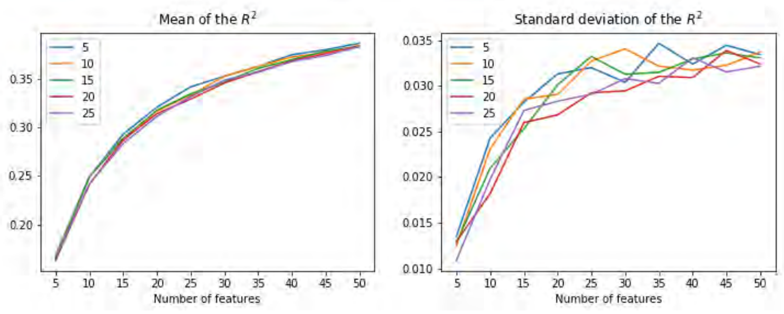


Figure 21: The mean R^2 on 50 explanations using the LGBM as complex model using different kernel widths

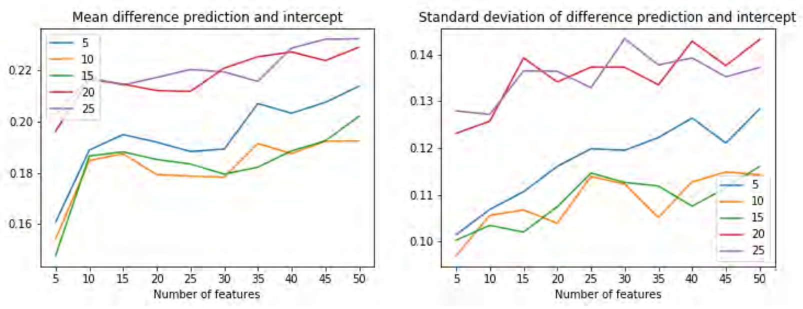


Figure 22: The mean difference between the prediction and the intercept using 50 explanations using the LGBM as complex model

Neural network Results
In Figures 23, 24 and 25, the results of the neural network can be seen. Again, the results of the neural network look very similar to the results of the LGBM. The same conclusions can be drawn from the figures below. However, from Figure 25, it can be seen that a small kernel width causes a larger difference between the prediction and the intercept. However, this conclusion is uncertain since the standard deviation is also higher.

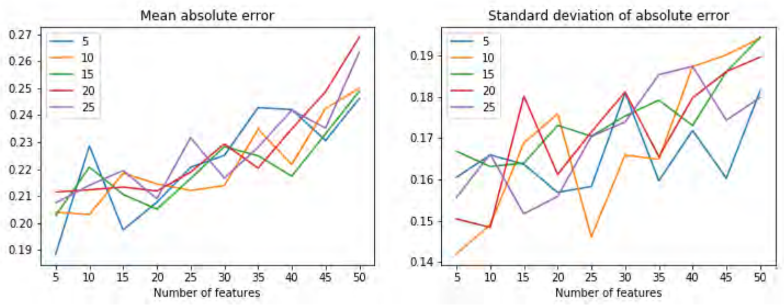


Figure 23: The mean absolute error on 50 explanations using the neural network as complex model using different kernel widths

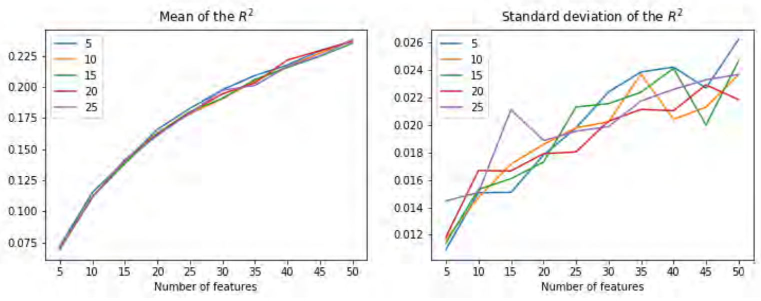


Figure 24: The mean R^2 on 50 explanations using the neural network as complex model using different kernel widths

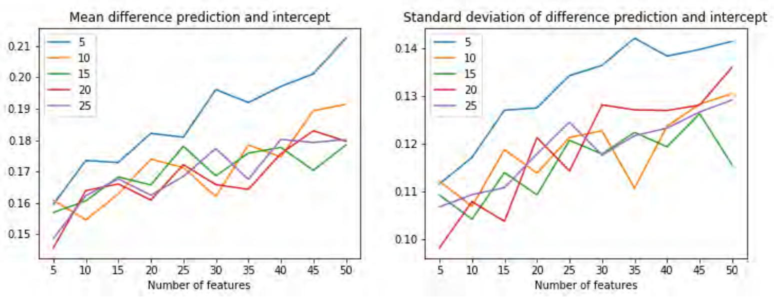


Figure 25: The mean difference between the prediction and the intercept using 50 explanations using the neural network

7.2 SHAP parameters

The most important parameter in the SHAP method is the size of the background data set. The background set is used for sampling the toggled off variables. The larger the data set, the better the approximations of the SHAP values will be. However, a large background set will lead to a longer computational time. The effect of the size of the background set is analyzed for both the LGBM and the neural network.

In the experiment, the 10 highest SHAP values will be shown. A development curve of these 10 features will be created over 12 settings. The setting available is the size of the background set. This can be set on 10, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1,000, using a background set of more than a 1,000 features will lead to unreasonable computational times.

LGBM Results
In Figure 26, the estimated SHAP values can be seen. It can be seen that the algorithms have difficulties to converge. It can be seen that after, approximately, 700 instances. However, from now on, the size of the background will be set on 400. This in order to manage computational times. The computational times can be seen in Figure 28.

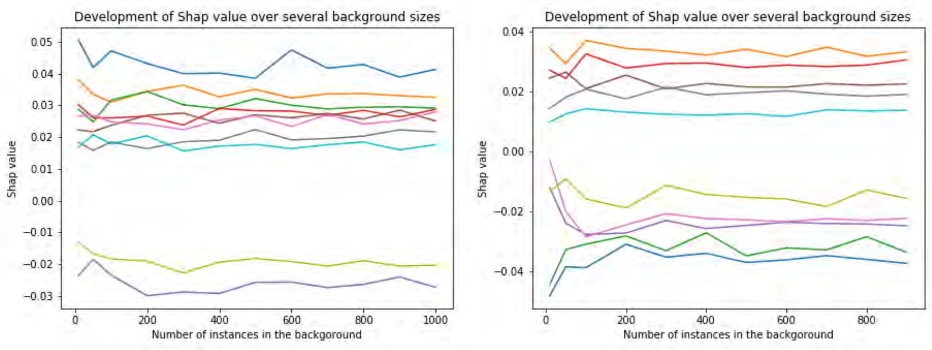


Figure 26: The R^2 on 100 explanations using the LGBM as complex model

Neural network Results
In Figure 27, the development of the SHAP values can be seen. If the size of the background data set is small, the approximations are not stable yet meaning the approximations will not be correct. When the number of instances used increases, the approximation becomes more stable and oscillates between a certain range. From now on, a background set of 600 will be used in SHAP explanations for the neural network.

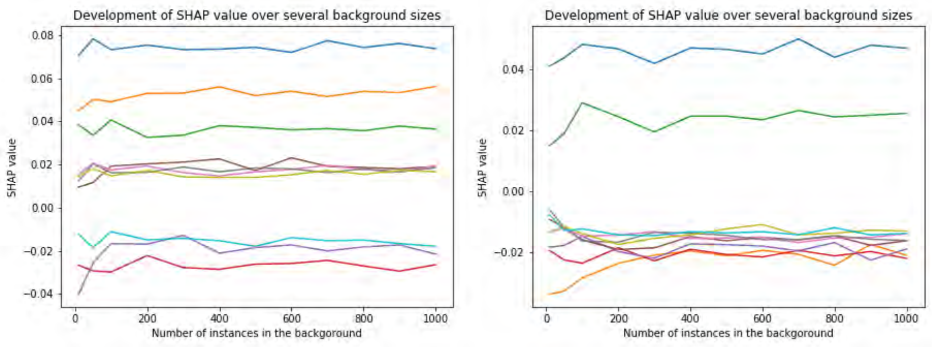


Figure 27: Development of SHAP values for two instances and the LGBM

Computational Time
In Figure 28, the computational times for both models can be seen. It is observed that the computational time behaves linearly with the size of the background set. Furthermore, the computational times for the LGBM is much longer than the computational time for the neural network. The computational time for generating a prediction with the LGBM is three times longer than generating a prediction with the neural network.

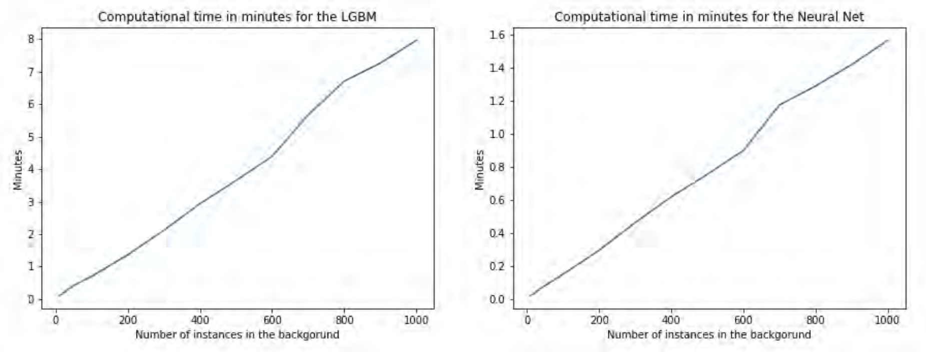


Figure 28: Development of SHAP values for two instances and the neural network

7.3 Standardization

In machine learning, standardizing, or normalizing, the data has a lot of impact on model performance. In deep learning, data with different units can lead to no or slower convergence. Second, if features have a unit difference of several thousands, e.g., a person's income and the person's credit score, a neural network has to learn large weights to compensate which leads to a highly unstable network. However, if one builds a tree ensemble, e.g., an LGBM, standardizing the data has little to no impact.

First, it will be shown that an LGBM finds the same relationships if the data is standardized. This will be not be done for a neural network since the model convergence is dependent on standardizing the data. Therefore, discovered relationships can be different causing comparisons irrelevant. Second, the impact on the explanation methods will be shown by comparing several explanations.

7.3.1 Similar models

In order to investigate the impact of standardizing the data, a new model needs to be trained. This new model is trained on data which is created according to the process described in Section 2.4.4. Furthermore, the new standardized LGBM will be trained with the same setting as described in Section 4.4.

In order to make a comparison of the transparency methods, it is necessary that they discovered the same relationships. To ensure this, a comparison is made of the performance of the models. Furthermore, the feature importance method described in Section 6.1 is used to draw similarities between the models.

In Table 19, it can be seen that the performance metrics are the same. In the third column, the correlation between the predicted probabilities is very high. These results show that the two models are very likely to be similar.

AUC standardized	**AUC Standard**	**Correlation Probabilities**
0.75648	0.75647	0.99656

Table 19: Performance metric is the same and the predicted probabilities are highly correlated

Furthermore, in Figure 29, it can be seen that the performance of the models is very similar.

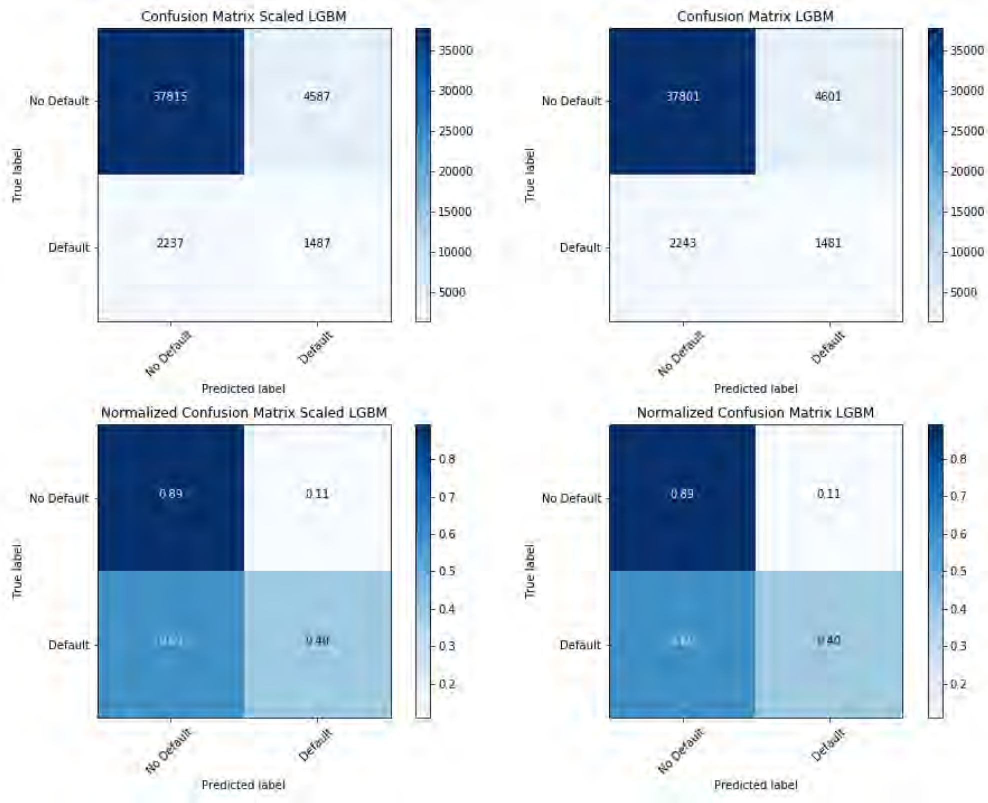


Figure 29: Confusion matrices of the models look very similar

In Table 20, the feature importance of both models are shown. The feature importance shown is the split count. The table shows that the models are very similar. However, the split counts have some small deviations, but they are still close together. Therefore, the assumption is made that both models are similar. This means that explanations should be similar. However small deviations in the explanation are allowed because the models are not exactly the same.

Features	Importance unstandardized	Importance standardized
Amt. Credit	1415	1420
Days Birth	1219	1247
Amt. Annuity	1170	1212
Days Last Phone Change	937	956
Days ID Publish	875	869
Days Employed	846	846
MAX(bureau.Days Credit)	790	818
SUM(bureau.Amt. Credit Sum)	741	746
Days Registration	727	681
MEAN(bureau.Amt. Credit Sum)	694	702

Table 20: Split count feature importance for the top 10 important features for both models

7.3.2 Comparison of explanations

If the explanations are generated for both models, it can be seen whether standardizing the data had any influence on the eventual explanation. In Figure 30, the two SHAP explanations can be seen. The upper explanation is the explanation for the original model and the lower one is the explanation for the standardized model. It can be observed that both explanations are very similar. Important to note, is that only the top 10 SHAP values are shown. Furthermore, all the explanation generated in this section, are generated from the same instance.

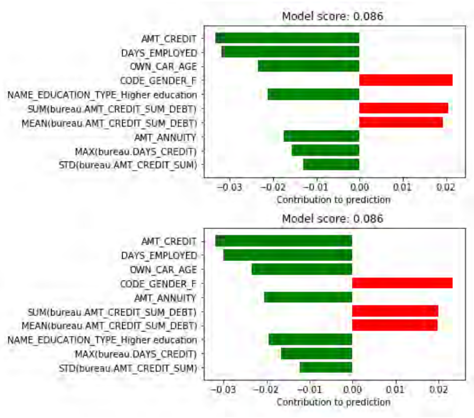


Figure 30: SHAP explanation for the original(above) and standardized(below) model

In Figure 31, the generated LIME explanations can be seen. For this method, it can be observed that the explanations have changed. However, they still have some similarity. The change seen can be caused by the distance metric. As explained in Section 6.2, the distance metric can behave badly if the features are in different units. If a feature has a range of a couple of thousands, it takes over the distance metric compared to features that have a range between zero and one. More comparisons that show the same result can be seen in Appendix C.

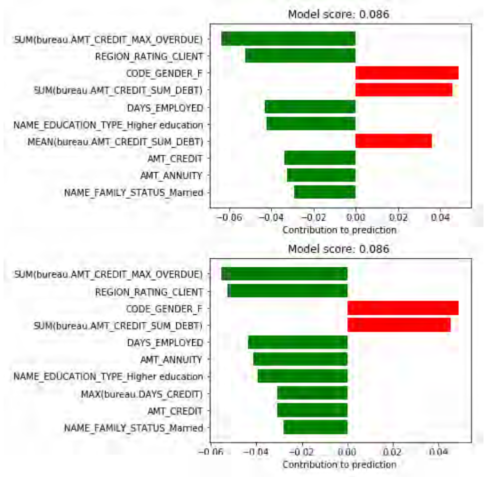


Figure 31: LIME explanation for the original(above) and standardized(below) model

7.4 Identifying important features

As explained in Section 2.2.1, some variables were left out. These variables are external credit scores. Obviously, these scores are good predictors of the creditworthiness of a new client. Therefore, it is expected that if these variables are added to the model the performance will increase significantly. In this section, these variables will be added to the model. Whereafter, it will be checked if both explanation methods are able to detect this addition to the model and observe these strong predictors. From now on, the model with the additional variables will be referred to as the full model. Furthermore, this analysis will be done for both the LGBM and the neural network.

In Table 21, it can be seen that the performance of the model significantly increases when the credit scores are added to the model.

	LGBM	**Neural network**
Original	0.7565	0.7278
Full	0.7881	0.7724

Table 21: ROC AUC score of the original and full models

In Figures 32, 33, 34 and 35, it can be seen that both methods are able to detect the strong predictors. Important to note that the figures for the SHAP method are not the full explanation. Only the variables with the largest SHAP value are displayed. For each figure, the above explanation is the explanation for the original model. The explanation below is an explanation for the full model.

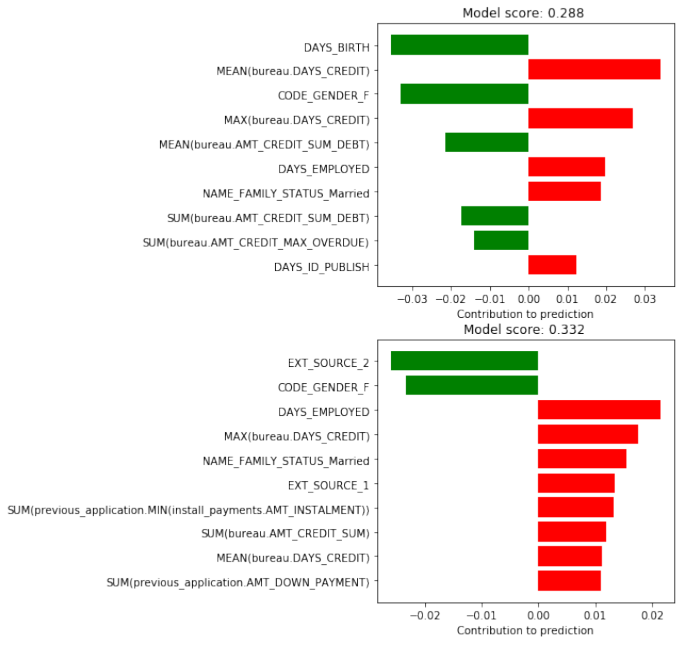


Figure 32: Explanations using the LGBM and the SHAP method

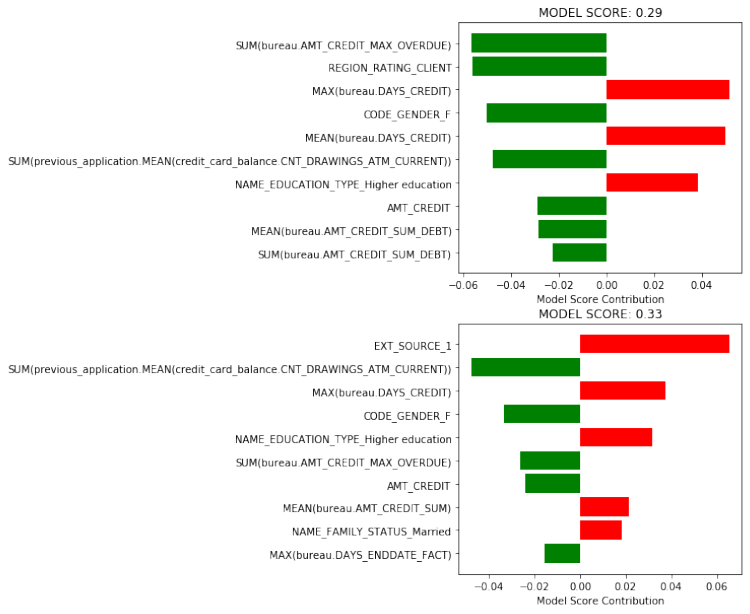


Figure 33: Explanations using the LGBM and the LIME method

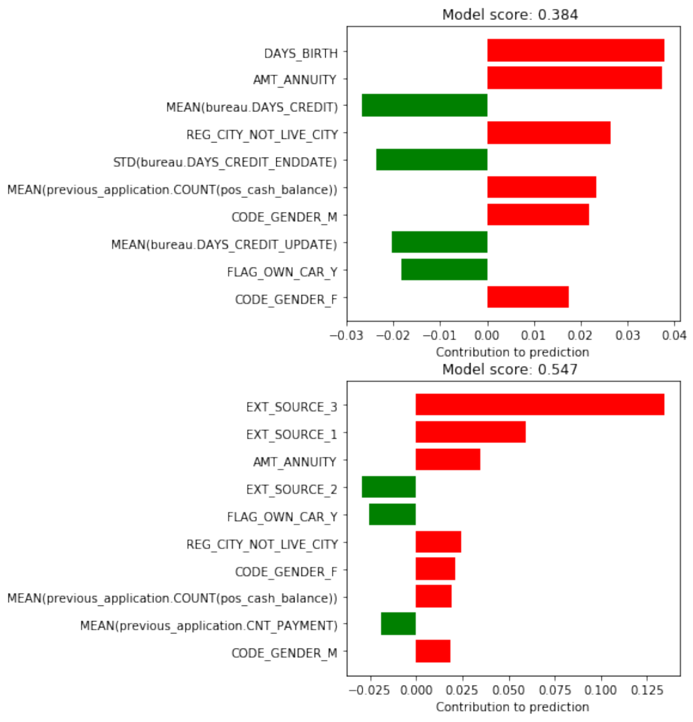


Figure 34: Explanations using the neural network and the SHAP method

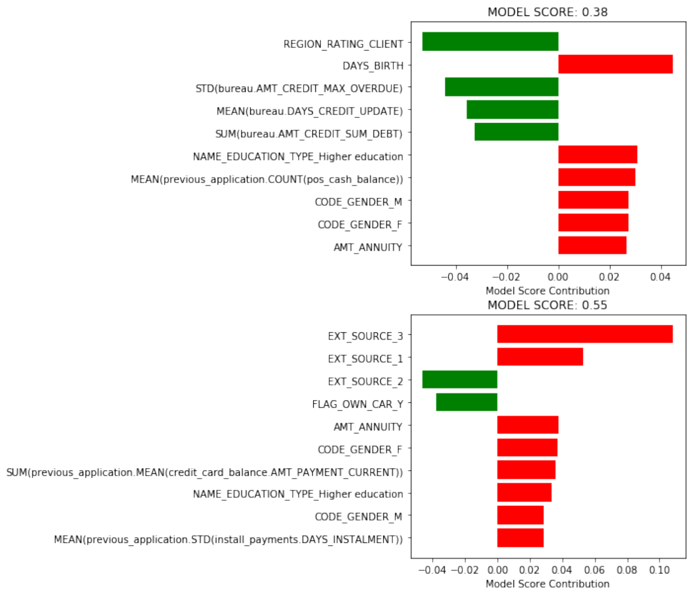


Figure 35: Explanations using the LGBM and the LIME method

More comparisons that show the same result can be seen in Appendix D.

7.5 Changing influential features

For an interpretation method to be of value, it should be able to detect variables that had a large influence on the model prediction. In this section, this will be investigated for both methods on both models. The investigation is done for the most positive and the most negative variable in the explanation.

The general approach for this analysis can be described in several steps. The first step is to generate explanations with both methods for a specific instance. Second, the variable that contributed the most negative in the prediction is identified. In other words, the variable that raises the predicted probability the most according to the explanation. After the selection of this variable, a new value will be assigned to this variable for that particular instance. The new value will be sampled at random from the training set to create this new instance. Finally, a prediction will be made with this new instance. This prediction will be compared with the original prediction. The second approach is the same but with the variable that lowers the predicted probability the most according to the explanation method.

Since the feature with the most negative impact is changed, the prediction should be lower. However, it can happen that the sampled value has an even more negative impact on the model prediction. The effect of this should be small. If a feature has a large negative impact, only a few values should exist that cause an even larger negative impact. Therefore, if the cumulative change in prediction is taken over a large number of explanations, a clear trend can be seen. In this analysis, the number of features in this analysis is set on 200.

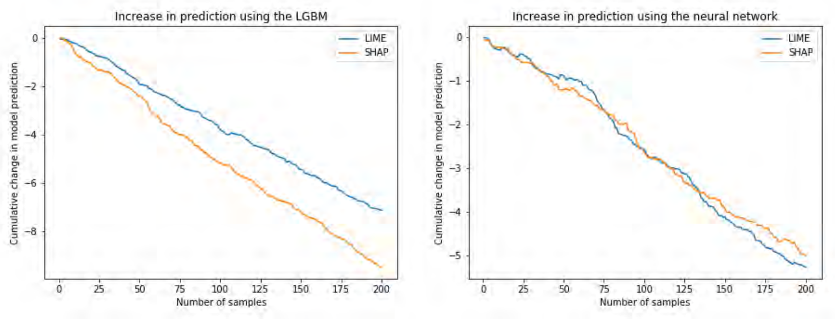


Figure 36: Change in prediction when, according to the method, the feature with the most negative influence was changed

In Figure 36, it can be seen that, in the case of the LGBM, the SHAP method is better in recognizing the feature that influences the prediction in a negative way. Furthermore, it can be seen that no conclusion can be drawn for the neural network. In Figure 37, this is the case for the LGBM. For the LGBM, LIME and SHAP have similar performance in picking the most negative feature. However, in the case of the neural network, it seems that SHAP is better in recognizing the most positive influencing feature.

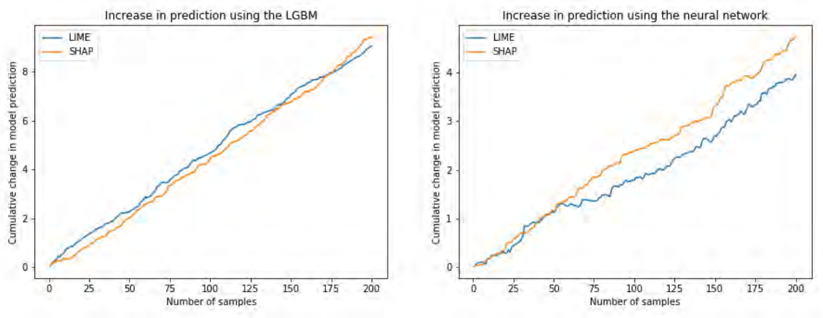


Figure 37: Change in prediction when, according to the method, the feature with the most positive influence was changed

The reason that the highest absolute influential feature is not changed, is that it would be unclear whether the prediction will increase or decrease. One option to solve this problem is to take the absolute value of the change in prediction. However, it could happen that the prediction changes in the opposite way than expected. If the absolute value is taken on this change, the curve in the figures above will increase faster than it actually does. Two reasons exist that the prediction changes in an unexpected way. First, the new sampled value can cause an even higher, or lower, prediction than the current value. Then, the curve will be overconfident because of the analysis design. Second, it can happen that the explanation method wrongly identified a positive influential feature as positive influential, or negative influential. If this happens, the curve will be overconfident by actually the lack of a method being a good identifier of influential features.

7.6 Global behavior

In this section, the ability to express global behavior for both methods will be investigated. This will only be done for the LGBM, because the method described in Section 6.1 can be used as a reference.

As can be seen in Section 6.1, feature importance for trees can be created in several ways. In Table 22, the split count can be seen.

Features	**Total split**
Amt. Credit	1420
Days Birth	1247
Amt. Annuity	1212
Days Last Phone Change	956
Days ID Publish	869
Days Employed	846
MAX(bureau.Days Credit)	818
SUM(bureau.Amt. Credit Sum)	746
Days Registration	681
MEAN(bureau.Amt. Credit Sum)	702

Table 22: Split count feature importance for the top 10 important features

The feature importance can differ greatly when different measurements are used, e.g., split count and total gain. Therefore, the total gain of the features will be shown in Table 23. This also means that the globally important features indicated by SHAP and LIME can be present in of the two tables. Therefore, a method receives one point if a global feature is present in one table and two if it is present in both.

Features	**Split gain**
MEAN(bureau.Days Credit)	66120
Days Employed	59000
Days Birth	44299
MAX(bureau.Days Credit)	33797
Amt. Credit	30248
Education Type Higher Education	28907
SUM(bureau.Amt. Credit Max Overdue)	28734
Region Rating Client	26965
Days Last Phone Change	26315
Code Gender F	24783

Table 23: Total gain feature importance for the top 10 important features

7.6.1 Global interpretation SHAP

As described in Section 6.3, SHAP is able to create global interpretation by combining local interpretations. As SHAP values are unique to an instance, one SHAP value can create an insight locally. However, if these SHAP values are collected for more instances, e.g., 200, a global interpretation can be created. By taking the mean of the absolute value of the SHAP values of the instance, feature importance can be created. This can be seen in Table 24. In total, SHAP was able to detect eight features that were in either the split count or the gain importance. Furthermore, the SHAP method scored 13 points.

Features	**Importance**
MEAN(bureau.Days Credit)	0.0260
Days Birth	0.0244
Days Employed	0.0241
Code Gender F	0.0230
SUM(bureau.Amt. Credit Max Overdue)	0.0196
MAX(bureau.Days Credit)	0.0185
Days Last Phone Change	0.0180
MEAN(bureau.Amt. Credit Sum Debt)	0.0177
Amt. Annuity	0.0164
SUM(bureau.Amt. Credit Sum Debt)	0.0142

Table 24: Mean absolute SHAP value for the top 10 important features

Furthermore, if one does not take the absolute value, the current feature importance can be extended. This extended feature importance can show if the influence on the model is positive or negative for low or high feature values. Such a feature importance plot can be seen in Figure 38.

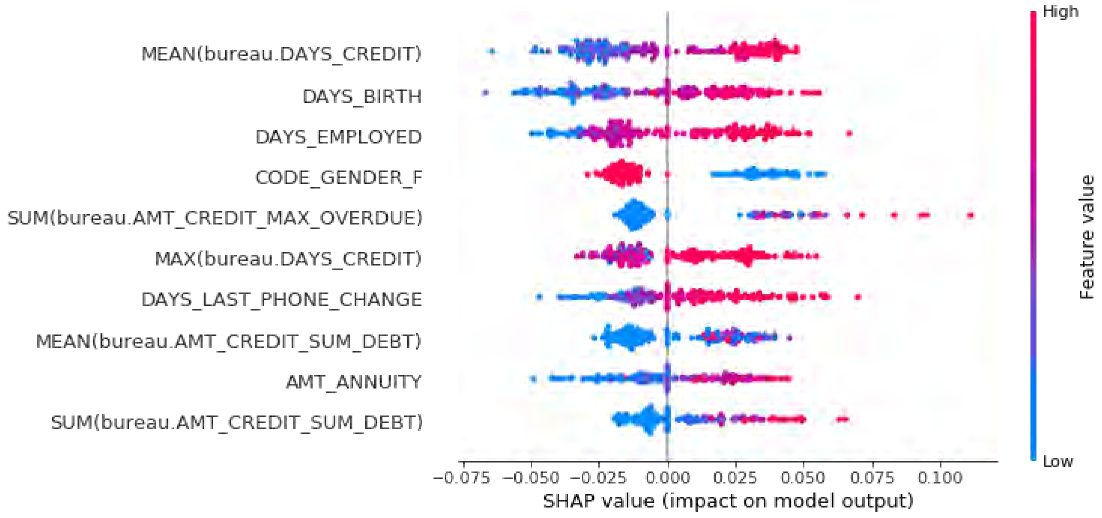


Figure 38: Global feature importance using SHAP values

Third, if one zooms in on a particular variable, in this example Days Birth. If the value of a feature is plotted against its SHAP value, the relationship with the target variable can be observed. Such a SHAP dependence can be seen in Figure 39. In this figure it can be seen that older people or more likely to default according to the model. Furthermore, with such plots, interaction between features can be investigated. In the figure, people that had higher education are colored red. It can be seen that young, high educated people are riskier than young, low educated people. A reason for this could be the fact that young, high educated people often have a student debt making it more difficult to payback their loan.

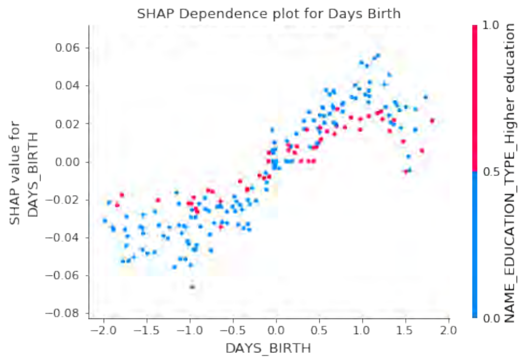


Figure 39: SHAP dependence plots can help investigate discovered relationship

7.6.2 Global interpretation LIME

In this section, a possible global interpretation technique will be discussed for the LIME method. Unfortunately, the coefficients created by the LIME method do not have the same properties as the SHAP values. However, the idea of combining local explanations can still provide some meaningful insights into the global behavior of a model. In order to get this understanding of global behavior, 200 LIME explanations are generated with each ten variables. For each explanation, the coefficients are ranked and scored. The scoring depends on the rank of each variable. The feature with the largest coefficient will receive a scoring of ten, the second largest a scoring of nine and eventually the variable with the smallest coefficient will receive a scoring of one. The intuition behind this, is that important variables occur often in the explanation. Therefore, such a ranking system over a large number of explanations can give insight into the global importance of all the variables.

After all explanations are generated, all features can be ranked according to their importance. In Table 25, the globally important features identified by the LIME method can be seen. Just as the SHAP method, LIME was able to detect eight out of the ten features present in either the gain or split count importance. Furthermore, in total it detected 11 features.

Features	**Importance**
SUM(bureau.Amt. Credit Max Overdue)	1929
Code Gender F	1739
Region Rating Client	1601
Education Type Higher education	1047
MEAN(bureau.Days Credit)	809
SUM(previous application.MEAN(credit card balance.Count Drawings ATM Current))	679
Days Employed	617
Days Birth	474
Max(bureau.Days Credit)	396
Family Status Married	381

Table 25: Occurrences in 200 LIME explanations for the top 10 important features

8 Conclusion

In this section, the LIME and SHAP methods will be compared using the analysis done in Section 7. In total five different analysis have been performed. The first is analyses on the parameters of the methods. Second, the impact of standardizing the data has been investigated. Third, it was investigated if the methods were able to detect strong predictors. Fourth, the impact of changing variables that had a strong influence on the model is observed. Finally, the ability of providing global interpretations was investigated.

In the parameter analysis, it was observed that the parameters of the LIME method have a strong influence on the eventual explanation. Such parameters are the variable selection method, kernel width and the number of variables used in the explanation. It has been shown that different settings lead to different explanations and a different quality of explanations. Furthermore, it was observed that the forward selection and the lasso method were the better performing methods. Of these two methods, the lasso method was more efficient, speaking of computational times. Therefore, the lasso variable selection was preferred.

In general, if more variables are used, the explanations will be of better quality. However, using more features will increase the complexity of the surrogate model. Furthermore, it was shown that a trade-off exists in the LIME method. If one desires high accuracy on the instance, lower accuracy on the neighborhood should be accepted and vice versa. Finally, it was shown that the kernel width had a big influence on the eventual explanation. However, no relationship could be found between the kernel width and the quality of an explanation.

The parameters described above are not all the parameters used by the LIME method. Other parameters are the distance metric, the kernel used and number of instances in the local neighborhood. All these parameters influence the explanation and it is difficult to find correct settings. It means that for each setting, new parameter settings should be investigated until the explanation makes sense. Obviously, all this is a big disadvantage for the LIME method.

The SHAP method does not have such a large parameter space. The only parameter of influence is the size of the background data set. It has been shown that the effect of the size is rather small. A larger background set means better estimations for the SHAP values and a larger computational time.

Second, standardizing the data had impact when the LIME method was used. The distance metric used in this method can corrupt explanations if the data is not in the same unit. Therefore, if the underlying model used unstandardized data, the SHAP method should be used. The SHAP method uses a different proximity measure and the unit of variables does not have any impact on this metric.

Third, when the three strong predictors were added to the model, both methods were able to detect these strong predictors. This makes both models suitable for detecting variables that have a strong influence on the predictions.

Fourth, the ability to detect features that have the largest influence was investigated. The variables with the largest SHAP value or the largest LIME coefficient were changed, both positive and negative. After the value of the variable was changed, the change in the prediction was observed. It was shown that SHAP was slightly better in detecting the most influential variable since the cumulative change in the prediction was larger.

Finally, the ability of giving a global interpretation was investigated. The SHAP method has the ability to give a global interpretation by definition. By generating a lot of explanations, the SHAP method is able to detect global relationships and give an insight into variable relationships. Especially, the dependence plot created by the SHAP method can be really valuable. Unfortunately is the LIME method not able to to create such insights by definition. However, by generating a lot of explanations, a global insight can be created.

To conclude, both SHAP and LIME are able to generate reasonable explanations. Both methods can detect features that had influence on the prediction. However, unstandardized data can corrupt LIME explanations. Furthermore, the use of the explanation should be taken into account. If explanations are used for model development or validation, both methods can be used. However, using the LIME method as explanation method towards the model users can cause problems. The influence of the parameters on the LIME explanation is large and makes it impossible to distinguish the most correct explanation. Therefore, it is recommended to use the SHAP method in such use cases.

9 Discussion

The goal of this research was to investigate the use of machine learning in a probability to default(PD) model and to provide transparency to such models. It has been shown that machine learning models can outperform traditional models if one looks at discriminatory power. However, during the development of the models, it was discovered that the use of machine learning introduces new problems.

An example of such a problem is the problem of imbalanced data. Imbalanced data can create biased models. These biased models are skewed towards the majority class, this results in predicting a lower PD. If such a model is used in production as a client acceptance model, a bank is exposed to more risk in reality than the model is predicting.

In this research, a solution is created for this problem. First the data is balanced. In this way a model is able to learn more about defaults. However, such an operation has a warping effect on the posterior probability. In other words, the probability distribution of the predictive model is not aligned with the true distribution. Therefore, a rating system is introduced. A rating system creates a link between a model prediction and the probability to default. It was shown that the combination of a machine learning model and a rating system creates an accurate and robust PD model.

Furthermore, it was shown in Section 5.2 that using a tree ensemble can introduce tail risks in the PD curve. A tree ensemble often consists of contradictory predicting trees, this means that the predictions will be centered around the mean PD in the training set. The tail risk was even higher in the Random Forest compared to the LGBM. However, it is not clear yet what the business implication is for this problem. This could be a topic for further research.

Second, an analysis of the two most used explanations methods was done. Both methods are able to provide meaningful explanations. Such explanations can be used to provide transparency and interpretation for machine learning models. Furthermore, the methods can be used to gain trust in the inner workings of the models. By generating a large number of explanations, the behavior of a model will become clear. Both methods can be used to provide local interpretation and global interpretation. However, the abilities of the SHAP method exceeds the abilities of the LIME method in global interpretation. The combination of SHAP values can show discovered relationships and help analyzing the working of black box models.

However, the LIME method does have some issues. For example, it has been shown that it is not a correct method if the data set is not standardized. Furthermore, the LIME method uses several parameters that highly affect the explanation. Unfortunately, there are, not yet, no clear procedures for identifying correct values of these parameters. This could be a subject for future research.

A limitation of this research is the focus on one particular data set. As this gave other benefits, like deep understanding in the models, data and use case. However, the methods can behave differently on other data sets and other use cases. Therefore, the main recommendation is to use both methods next to each other. The analysis done in Section 7.5, is able to help deciding which method is more suitable for that use case. The creation of more of such analysis is also a suitable topic for future research.

Furthermore, it was explained that the intended use of the explanation has an influence on which method to use. A future study towards regulations on black box models could help deciding when to use which method. For example, the LIME method could work better for a certain black box model. However, the SHAP explanation is the only explanation that is compliant with the regulation on that use case. In this case, the SHAP method will get preference even if the LIME method will generate better explanations.

A limitation of this research, is the focus on two explanation frameworks. However, focusing on two methods gave the ability investigate the methods more. Besides LIME and SHAP, other methods exist. For example, one could generate counterfactual explanations. A counterfactual explanation provides an actionable explanation in the form of "if not X, then not Y". More information on counterfactual explanations and other interpretation techniques can be found in the book Interpretable Machine Learning by Christoph Molnar [17].

References

[1] Machine learning in credit risk modelling. https://james.finance/static/assets/whitepapers/Machine-Learning-in-Credit-Risk-Modeling-James-white-paper.pdf, 2017.

[2] P. M. Addo, D. Guegan, , and B. Hassani. Credit risk analysis using machine and deep learning models. 2018.

[3] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In *Proceedings of the 8th International Conference on Database Theory*, ICDT '01, pages 420–434, Berlin, Heidelberg, 2001. Springer-Verlag.

[4] D. Alvarez-Melis and T. S. Jaakkola. On the robustness of interpretability methods. *CoRR*, abs/1806.08049, 2018.

[5] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl. Algorithms for hyper-parameter optimization. 2011.

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

[7] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[8] F. Chollet et al. Keras. https://keras.io, 2015.

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

[10] X. Glorot and Y. Bergio. Understanding the difficulty of training deep feedforward neural networks. 2010.

[11] J. M. Kanter and K. Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2015.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. 2015.

[13] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Program.*, 45:503–528, 1989.

[14] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[15] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. 2013.

[16] Microsoft. Lightgbm, light gradient boosting machine. https://github.com/Microsoft/LightGBM, 2016.

[17] C. Molnar. *Interpretable Machine Learning*. 2019. https://christophm.github.io/interpretable-ml-book/.

[18] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. *Proceedings of the 27th International Conference on Machine Learning*, 2010.

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

[20] A. Petropoulos, V. Siakoulis, E. Stavroulakis, and A. Klamargias. A robust machine learning approach for credit risk analysis of large loan level datasets using deep learning and extreme gradient boosting. *Ninth IFC Conference on "Are post-crisis statistical initiatives completed?"*, 2018.

[21] A. D. Pozzolo, O. Caelen, and G. Bontempi. When is undersampling effective in unbalanced classification tasks? In *Proceedings of the 2015th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I*, pages 200–215. Springer, 2015.

[22] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[23] L. S. Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton, 1953.

[24] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. *NIPS*, 2012.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[26] G. M. Weiss. Mining with rarity: A unifying framework. 2004.

A List of all available attributes

Table	Attribute	Description
Application	SK_ID_CURR	ID of loan in our sample
Application	TARGET	Target variable (1 - payment difficulties, 0 - other)
Application	NAME_CONTRACT_TYPE	Identification if loan is cash or revolving
Application	CODE_GENDER	Gender of the client
Application	FLAG_OWN_CAR	Flag if the client owns a car
Application	FLAG_OWN_REALTY	Flag if client owns a house or flat
Application	CNT_CHILDREN	Number of children the client has
Application	AMT_INCOME_TOTAL	Income of the client
Application	AMT_CREDIT	Credit amount of the loan
Application	AMT_ANNUITY	Loan annuity
Application	AMT_GOODS_PRICE	For consumer loans it is the price of the goods for which the loan is given
Application	NAME_TYPE_SUITE	Who was accompanying client when he was applying for the loan
Application	NAME_INCOME_TYPE	Clients income type
Application	NAME_EDUCATION_TYPE	Level of highest education the client achieved
Application	NAME_FAMILY_STATUS	Family status of the client
Application	NAME_HOUSING_TYPE	Housing situation of the client
Application	REGION_POPULATION_ RELATIVE	Normalized population of region where client lives
Application	DAYS_BIRTH	Client's age in days at the time of application
Application	DAYS_EMPLOYED	How many days before the application the client started current employment
Application	DAYS_REGISTRATION	How many days before the application did client change his registration
Application	DAYS_ID_PUBLISH	How many days before the application did client change the identity document with which he applied for the loan
Application	OWN_CAR_AGE	Age of client's car
Application	FLAG_MOBIL	Did client provide mobile phone
Application	FLAG_EMP_PHONE	Did client provide employer phone
Application	FLAG_WORK_PHONE	Did client provide work phone
Application	FLAG_CONT_MOBILE	Was mobile phone reachable
Application	FLAG_PHONE	Did client provide home phone
Application	FLAG_EMAIL	Did client provide email
Application	OCCUPATION_TYPE	What kind of occupation does the client have
Application	CNT_FAM_MEMBERS	How many family members does client have
Application	REGION_RATING_CLIENT	Rating of the region where client lives
Application	REGION_RATING_CLIENT_ W_CITY	Rating of the region where client lives with taking city into account
Application	WEEKDAY_APPR_PROCESS_ START	On which day of the week did the client apply for the loan

Table	Attribute	Description
Application	HOUR_APPR_PROCESS_START	Approximately at what hour did the client apply for the loan
Application	REG_REGION_NOT_LIVE_REGION	Flag if client's permanent address does not match contact address (1=different, 0=same, at region level)
Application	REG_REGION_NOT_WORK_REGION	Flag if client's permanent address does not match work address (1=different, 0=same, at region level)
Application	LIVE_REGION_NOT_WORK_REGION	Flag if client's contact address does not match work address (1=different, 0=same, at region level)
Application	REG_CITY_NOT_LIVE_CITY	Flag if client's permanent address does not match contact address (1=different, 0=same, at city level)
Application	REG_CITY_NOT_WORK_CITY	Flag if client's permanent address does not match work address (1=different, 0=same, at city level)
Application	LIVE_CITY_NOT_WORK_CITY	Flag if client's contact address does not match work address (1=different, 0=same, at city level)
Application	ORGANIZATION_TYPE	Type of organization where client works
Application	EXT_SOURCE_1	Normalized score from external data source
Application	EXT_SOURCE_2	Normalized score from external data source
Application	EXT_SOURCE_3	Normalized score from external data source
Application	APARTMENTS_AVG	Normalized information about building where the client lives[3]
Application	BASEMENTAREA_AVG	Normalized information about building where the client lives[3]
Application	YEARS_BEGIN_EXPLUATATION_AVG	Normalized information about building where the client lives[3]
Application	YEARS_BUILD_AVG	Normalized information about building where the client lives[3]
Application	COMMONAREA_AVG	Normalized information about building where the client lives[3]
Application	ELEVATORS_AVG	Normalized information about building where the client lives[3]
Application	ENTRANCES_AVG	Normalized information about building where the client lives[3]
Application	FLOORSMAX_AVG	Normalized information about building where the client lives[3]
Application	FLOORSMIN_AVG	Normalized information about building where the client lives[3]
Application	LANDAREA_AVG	Normalized information about building where the client lives[3]
Application	LIVINGAPARTMENTS_AVG	Normalized information about building where the client lives[3]
Application	LIVINGAREA_AVG	Normalized information about building where the client lives[3]
Application	NONLIVINGAPARTMENTS_AVG	Normalized information about building where the client lives[3]

Table	Attribute	Description
Application	NONLIVINGAREA_AVG	Normalized information about building where the client lives[3]
Application	APARTMENTS_MODE	Normalized information about building where the client lives[3]
Application	BASEMENTAREA_MODE	Normalized information about building where the client lives[3]
Application	YEARS_BEGINEXPLUATA-TION_MODE	Normalized information about building where the client lives[3]
Application	YEARS_BUILD_MODE	Normalized information about building where the client lives[3]
Application	COMMONAREA_MODE	Normalized information about building where the client lives[3]
Application	ELEVATORS_MODE	Normalized information about building where the client lives[3]
Application	ENTRANCES_MODE	Normalized information about building where the client lives[3]
Application	FLOORSMAX_MODE	Normalized information about building where the client lives[3]
Application	FLOORSMIN_MODE	Normalized information about building where the client lives[3]
Application	LANDAREA_MODE	Normalized information about building where the client lives[3]
Application	LIVINGAPARTMENTS_MODE	Normalized information about building where the client lives[3]
Application	LIVINGAREA_MODE	Normalized information about building where the client lives[3]
Application	NONLIVINGAPARTMENTS_MODE	Normalized information about building where the client lives[3]
Application	NONLIVINGAREA_MODE	Normalized information about building where the client lives[3]
Application	APARTMENTS_MEDI	Normalized information about building where the client lives[3]
Application	BASEMENTAREA_MEDI	Normalized information about building where the client lives[3]
Application	YEARS_BEGINEXPLUATA-TION_MEDI	Normalized information about building where the client lives[3]
Application	YEARS_BUILD_MEDI	Normalized information about building where the client lives[3]
Application	COMMONAREA_MEDI	Normalized information about building where the client lives[3]
Application	ELEVATORS_MEDI	Normalized information about building where the client lives[3]
Application	ENTRANCES_MEDI	Normalized information about building where the client lives[3]
Application	FLOORSMAX_MEDI	Normalized information about building where the client lives[3]

Table	**Attribute**	**Description**
Application	FLOORSMIN_MEDI	Normalized information about building where the client lives [3]
Application	LANDAREA_MEDI	Normalized information about building where the client lives [3]
Application	LIVINGAPARTMENTS_MEDI	Normalized information about building where the client lives[3]
Application	LIVINGAREA_MEDI	Normalized information about building where the client lives[3]
Application	NONLIVINGAPARTMENTS_MEDI	Normalized information about building where the client lives[3]
Application	NONLIVINGAREA_MEDI	Normalized information about building where the client lives[3]
Application	FONDKAPREMONT_MODE	Normalized information about building where the client lives[3]
Application	HOUSETYPE_MODE	Normalized information about building where the client lives[3]
Application	TOTALAREA_MODE	Normalized information about building where the client lives[3]
Application	WALLSMATERIAL_MODE	Normalized information about building where the client lives[3]
Application	EMERGENCYSTATE_MODE	Normalized information about building where the client lives[3]
Application	OBS_30_CNT_SOCIAL_CIRCLE	Observations of client's social surroundings with observable 30 DPD default
Application	DEF_30_CNT_SOCIAL_CIRCLE	Observations of client's social surroundings defaulted on 30 DPD
Application	OBS_60_CNT_SOCIAL_CIRCLE	Observations of client's social surroundings with observable 60 DPD default
Application	DEF_60_CNT_SOCIAL_CIRCLE	Observations of client's social surroundings defaulted on 60 DPD
Application	DAYS_LAST_PHONE_CHANGE	How many days before application did client change phone
Application	FLAG_DOCUMENT_i	Did client provide document i, i = 2, 3, ..., 21
Application	AMT_REQ_CREDIT_BUREAU_HOUR	Number of enquiries to Credit Bureau about the client one hour before application
Application	AMT_REQ_CREDIT_BUREAU_DAY	Number of enquiries to Credit Bureau about the client one day before application (excluding one hour before application)
Application	AMT_REQ_CREDIT_BUREAU_WEEK	Number of enquiries to Credit Bureau about the client one week before application (excluding one day before application)
Application	AMT_REQ_CREDIT_BUREAU_MON	Number of enquiries to Credit Bureau about the client one month before application (excluding one week before application)

Table	**Attribute**	**Description**
Application	AMT_REQ_CREDIT_BUR-EAU_QRT	Number of enquiries to Credit Bureau about the client 3 months before application (excluding one month before application)
Application	AMT_REQ_CREDIT_BUR-EAU_YEAR	Number of enquiries to Credit Bureau about the client one day year (excluding last 3 months before application)
Bureau	SK_ID_CURR	ID of loan in our sample - one loan in our sample can have 0,1,2 or more related previous credits in credit bureau
Bureau	SK_BUREAU_ID	ID of previous Credit Bureau credit related to our loan
Bureau	CREDIT_ACTIVE	Status of the Credit Bureau (CB) reported credits
Bureau	CREDIT_CURRENCY	Recoded currency of the Credit Bureau credit
Bureau	DAYS_CREDIT	How many days before current application did client apply for Credit Bureau credit
Bureau	CREDIT_DAY_OVERDUE	Number of days past due on CB credit at the time of application for related loan in our sample
Bureau	DAYS_CREDIT_ENDDATE	Remaining duration of CB credit at the time of application in Home Credit
Bureau	DAYS_ENDDATE_FACT	Days since CB credit ended at the time of application in Home Credit
Bureau	AMT_CREDIT_MAX_OVERDUE	Maximal amount overdue on the Credit Bureau credit so far
Bureau	CNT_CREDIT_PROLONG	How many times was the Credit Bureau credit prolonged
Bureau	AMT_CREDIT_SUM	Current credit amount for the Credit Bureau credit
Bureau	AMT_CREDIT_SUM_DEBT	Current debt on Credit Bureau credit
Bureau	AMT_CREDIT_SUM_LIMIT	Current credit limit of credit card reported in Credit Bureau
Bureau	AMT_CREDIT_SUM_OVERDUE	Current amount overdue on Credit Bureau credit
Bureau	CREDIT_TYPE	Type of Credit Bureau credit
Bureau	DAYS_CREDIT_UPDATE	How many days before loan application did last information about the Credit Bureau credit changed
Bureau	AMT_ANNUITY	Annuity of the Credit Bureau credit
Bureau balance	SK_BUREAU_ID	Recoded ID of Credit Bureau credit
Bureau balance	MONTHS_BALANCE	Month of balance relative to application date
Bureau balance	STATUS	Status of Credit Bureau loan during the month
POS CASH balance	SK_ID_PREV	ID of previous credit in Home Credit related to loan in our sample
POS CASH balance	SK_ID_CURR	ID of loan in our sample

Table	Attribute	Description
POS CASH balance	MONTHS_BALANCE	Month of balance relative to application date
POS CASH balance	CNT_INSTALMENT	Term of previous credit
POS CASH balance	CNT_INSTALMENT_FUTURE	Installments left to pay on the previous credit
POS CASH balance	NAME_CONTRACT_STATUS	Contract status during the month
POS CASH balance	SK_DPD	DPD during the month of previous credit
Credit card balance	SK_DPD_DEF	DPD during the month with tolerance of the previous credit
Credit card balance	SK_ID_PREV	ID of previous credit in Home credit related to loan in our sample
Credit card balance	SK_ID_CURR	ID of loan in our sample
Credit card balance	MONTHS_BALANCE	Month of balance relative to application date
Credit card balance	AMT_BALANCE	Balance during the month of previous credit
Credit card balance	AMT_CREDIT_LIMIT_ACTUAL	Credit card limit during the month of the previous credit
Credit card balance	AMT_DRAWINGS_ATM_CURRENT	Amount drawing at ATM during the month of the previous credit
Credit card balance	AMT_DRAWING_CURRENT	Amount drawing during the month of the previous credit
Credit card balance	AMT_DRAWINGS_OTHER_CURRENT	Amount of other drawings during the month of the previous credit
Credit card balance	AMT_DRAWINGS_POS_CURRENT	Amount drawing or buying goods during the month of the previous credit
Credit card balance	AMT_INST_MIN_REGULARITY	Minimal installment for this month of the previous credit
Credit card balance	AMT_PAYMENT_CURRENT	How much did the client pay during the month on the previous credit
Credit card balance	AMT_PAYMENT_TOTAL_CURRENT	How much did the client pay during the month in total on the previous credit
Credit card balance	AMT_RECEIVABLE_ PRINCIPAL	Amount receivable for principal on the previous credit
Credit card balance	AMT_RECIVABLE	Amount receivable on the previous credit
Credit card balance	AMT_TOTAL_RECEIVABLE	Total amount receivable on the previous credit
Credit card balance	CNT_DRAWINGS_ATM_CURRENT	Number of drawings at ATM during this month on the previous credit
Credit card balance	CNT_DRAWINGS_CURRENT	Number of drawings during this month on the previous credit

Table	Attribute	Description
Credit card balance	CNT_DRAWINGS_OTHER_CURRENT	Number of other drawings during this month on the previous credit
Credit card balance	CNT_DRAWINGS_POS_CURRENT	Number of drawings for goods during this month on the previous credit
Credit card balance	CNT_INSTALMENT_MATURE_CUM	Number of paid installments on the previous credit
Credit card balance	NAME_CONTRACT_STATUS	Contract status on the previous credit
Credit card balance	SK_DPD	DPD during the month on the previous credit
Credit card balance	SK_DPD_DEF	DPD during the month with tolerance of the previous credit
Previous application	SK_ID_PREV	ID of previous credit in Home credit related to loan in our sample
Previous application	SK_ID_CURR	ID of loan in our sample
Previous application	NAME_CONTRACT_TYPE	Contract product type of the previous application
Previous application	AMT_ANNUITY	Annuity of previous application
Previous application	AMT_APPLICATION	For how much credit did client ask on the previous application
Previous application	AMT_CREDIT	Final credit amount on the previous application
Previous application	AMT_DOWN_PAYMENT	Down payment on the previous application
Previous application	AMT_GOODS_PRICE	Goods price of good that client asked for (if applicable) on the previous application
Previous application	WEEKDAY_APPR_PROCESS_START	On which day of the week did the client apply for previous application
Previous application	HOUR_APPR_PROCESS_START	Approximately at what day hour did the client apply for the previous application
Previous application	FLAG_LAST_APPL_PER_CONTRACT	Flag if it was last application for the previous contract
Previous application	NFLAG_LAST_APPL_IN_DAY	Flag if the application was the last application per day of the client
Previous application	NFLAG_MICRO_CASH	Flag Micro finance loan
Previous application	RATE_DOWN_PAYMENT	Down payment rate normalized on previous credit
Previous application	RATE_INTEREST_PRIMARY	Interest rate normalized on previous credit
Previous application	RATE_INTEREST_PRIVILEGED	Interest rate normalized on previous credit
Previous application	NAME_CASH_LOAN_PURPOSE	Purpose of the cash loan

Table	**Attribute**	**Description**
Previous application	NAME_CONTRACT_STATUS	Contract status of previous application
Previous application	DAYS_DECISION	Relative to current application when was the decision about previous application made
Previous application	NAME_PAYMENT_TYPE	Payment method that client chose to pay for the previous application
Previous application	CODE_REJECT_REASON	Why was the previous application rejected
Previous application	NAME_TYPE_SUITE	Who accompanied client when applying for the previous application
Previous application	NAME_CLIENT_TYPE	Was the client old or new client when applying for the previous application
Previous application	NAME_GOODS_CATEGORY	What kind of goods did the client apply for in the previous application
Previous application	NAME_PORTFOLIO	Goal of the previous application
Previous application	NAME_PRODUCT_TYPE	Was the previous application x-sell or walk-in
Previous application	CHANNEL_TYPE	Through which channel the client was acquired
Previous application	SELLERPLACE_AREA	Selling area of seller place of the previous application
Previous application	NAME_SELLER_INDUSTRY	The industry of the seller
Previous application	CNT_PAYMENT	Term of previous credit at application of the previous application
Previous application	NAME_YIELD_GROUP	Grouped interest rate into small medium and high of the previous application
Previous application	PRODUCT_COMBINATION	Detailed product combination of the previous application
Previous application	DAYS_FIRST_DRAWING	Relative to application date of current application when was the first disbursement of the previous application
Previous application	DAYS_FIRST_DUE	Relative to application date of current application when was the first due supposed to be of the previous application
Previous application	DAYS_LAST_DUE_1ST_VERSION	Relative to application date of current application when was the first due of the previous application
Previous application	DAYS_LAST_DUE	Relative to application date of current application when was the last due date of the previous application
Previous application	DAYS_TERMINATION	Relative to application date of current application when was the expected termination of the previous application
Previous application	NFLAG_INSURED_ON_APPROVAL	Did the client requested insurance during the previous application

Table	**Attribute**	**Description**
Installments payments	SK_ID_PREV	ID of previous credit in Home credit related to loan in our sample
Installments payments	SK_ID_CURR	ID of loan in our sample
Installments payments	NUM_INSTALMENT_VERSION	Version of installment calendar of previous credit
Installments payments	NUM_INSTALMENT_NUMBER	On which installment the payment was observed
Installments payments	DAYS_INSTALMENT	When the installment of previous credit was supposed to be paid
Installments payments	DAYS_ENTRY_PAYMENT	When was the installments of previous credit paid actually
Installments payments	AMT_INSTALMENT	What was the prescribed installment amount of previous credit on this installment
Installments payments	AMT_PAYMENT	What the client actually paid on previous credit on this installment

Table 26: List of all available attributes

[3]What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor

B Credit Rating systems

(a) LGBM

Group	1	2	3	4	5	6	7	8	9	10
Range	(0, 0.099)	(0.010, 0.135)	(0.136, 0.171)	(0.172, 0.208)	(0.209, 0.250)	(0.251, 0.299)	(0.300, 0.358)	(0.359, 0.434)	(0.435, 0.546)	(0.547, 1)
$\boldsymbol{PD_{train}}$	0.49%	1.03%	1.71%	2.56%	3.83%	5.21%	7.25%	10.48%	16.20%	31.97%
$\boldsymbol{PD_{test}}$	1.28%	1.97%	2.68%	3.70%	4.88%	6.29%	7.79%	10.65%	14.68%	27.67%

(b) LogReg

Group	1	2	3	4	5	6	7	8	9	10
Range	(0, 0.112)	(0.113, 0.155)	(0.156, 0.194)	(0.195, 0.232)	(0.233, 0.273)	(0.274, 0.319)	(0.320, 0.372)	(0.373, 0.438)	(0.439, 0.533)	(0.534, 1)
$\boldsymbol{PD_{train}}$	1.79%	2.53%	3.30%	4.22%	5.43%	7.09%	8.28%	10.64%	14.22%	23.22%
$\boldsymbol{PD_{test}}$	1.63%	2.75%	3.27%	3.86%	6.06%	6.92%	8.77%	10.59%	15.26%	22.17%

(c) Neural network

Group	1	2	3	4	5	6	7	8	9	10
Range	(0, 0.132)	(0.132, 0.181)	(0.182, 0.221)	(0.222, 0.256)	(0.257, 0.289)	(0.290, 0.323)	(0.324, 0.360)	(0.361, 0.410)	(0.410, 0.501)	(0.502, 1)
$\boldsymbol{PD_{train}}$	1.21%	2.00%	2.95%	3.98%	5.07%	6.39%	8.73%	11.03%	14.59%	24.80%
$\boldsymbol{PD_{test}}$	1.69%	2.36%	3.45%	4.10%	5.51%	6.82%	8.06%	10.24%	15.48%	23.32%

(d) Random Forest

Group	1	2	3	4	5	6	7	8	9	10
Range	(0, 0.127)	(0.128, 0.171)	(0.172, 0.208)	(0.209, 0.244)	(0.245, 0.281)	(0.282, 0.322)	(0.323, 0.370)	(0.371, 0.434)	(0.435, 0.539)	(0.540, 1)
$\boldsymbol{PD_{train}}$	0.00%	0.00%	0.00%	0.00%	0.03%	0.17%	0.76%	2.87%	14.21%	62.70%
$\boldsymbol{PD_{test}}$	1.68%	2.62%	3.01%	4.08%	4.92%	5.76%	7.75%	10.25%	16.69%	26.58%

Table 27: Credit rating for all models

C Additional standardized explanations

In this section, additional explanations are provided. These explanations provide extra proof that the SHAP method is robust against unstandardized data, while the LIME methods have issues. Note that the upper explanation is always an explanation of the unstandardized model where the explanation below is the explanation of the standardized model. Some difference is seen in the SHAP explanation, however the difference in the LIME explanation is larger.

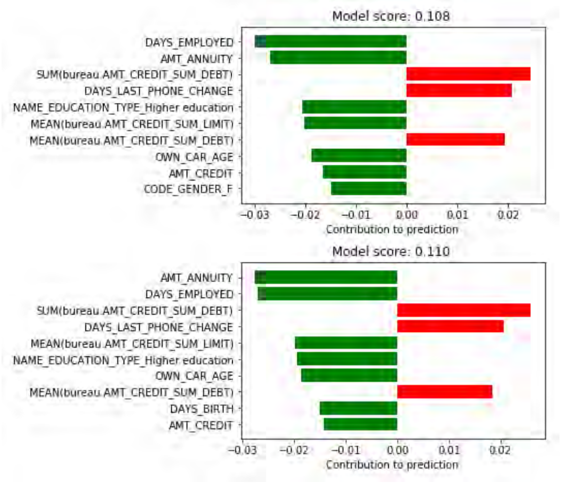


Figure 40: Influence of standardizing the data for the SHAP method

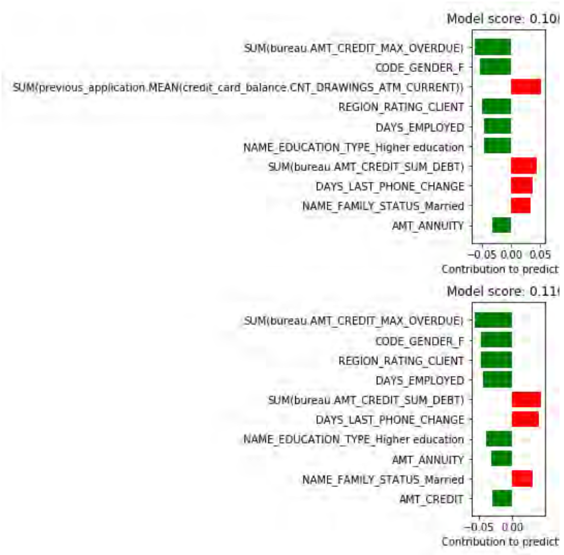


Figure 41: Influence of standardizing the data for the LIME method

D Additional strong predictor explanations

In this section, some additional explanations are shown that provide extra proof that both methods are able to detect the strong predictors. The upper explanation is always an explanation of the original model where the explanation below is the explanation of the full model.

D.1 LGBM results

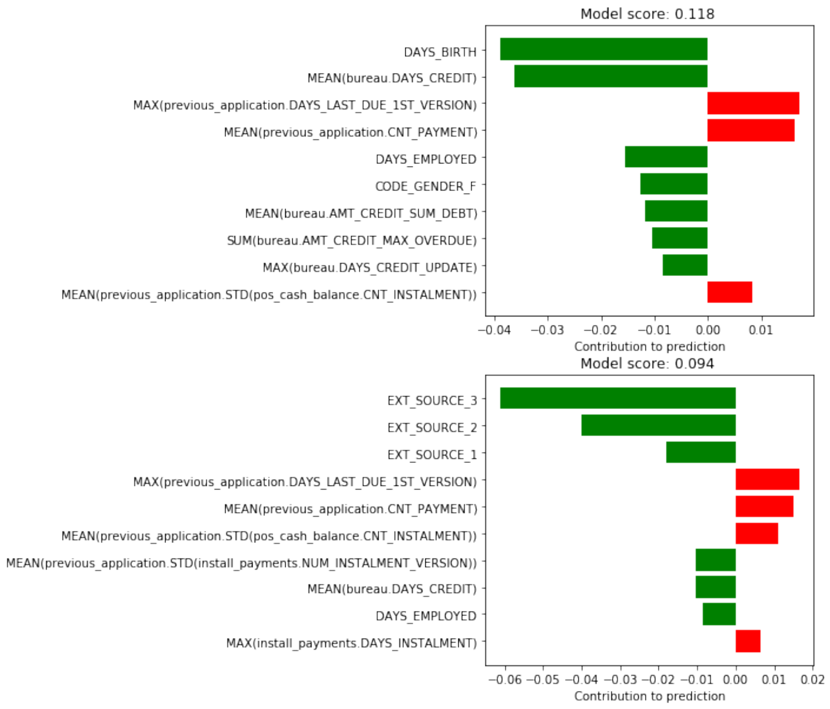


Figure 42: Example of the detection strong predictors for the SHAP method

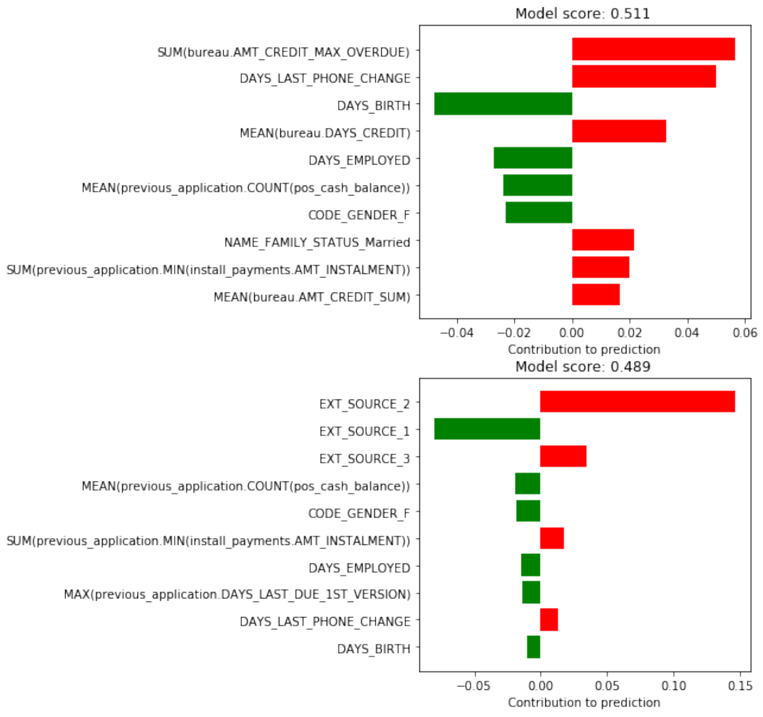


Figure 43: Example of the detection strong predictors for the SHAP method

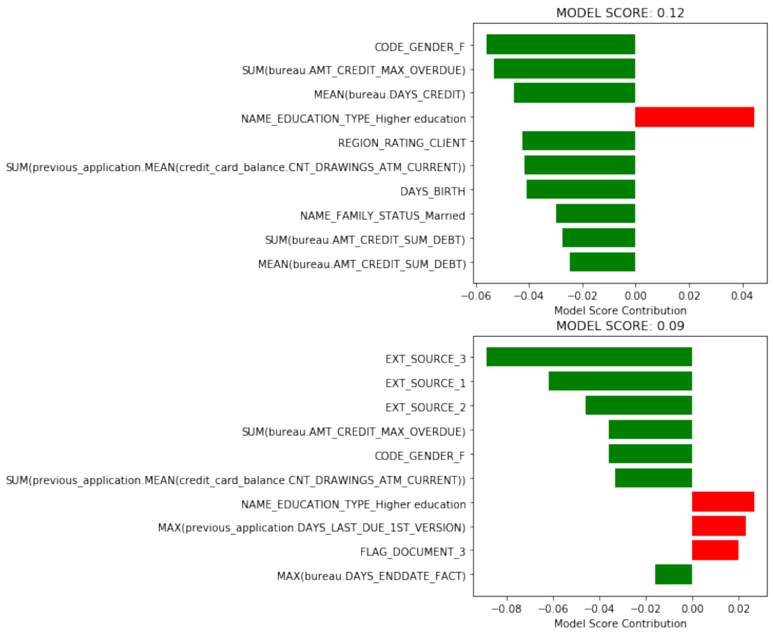


Figure 44: Example of the detection strong predictors for the LIME method

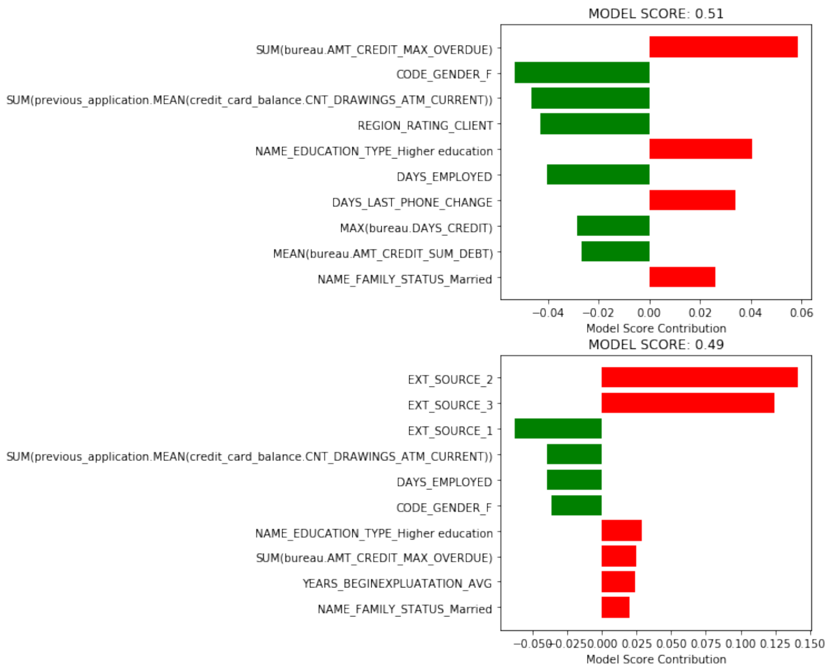


Figure 45: Example of the detection strong predictors for the LIME method

D.2 Neural network results

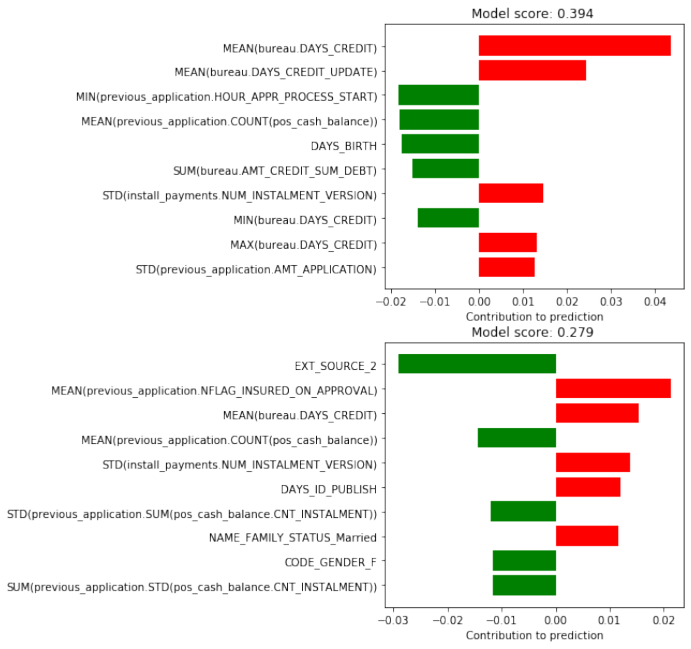


Figure 46: Example of the detection strong predictors for the SHAP method

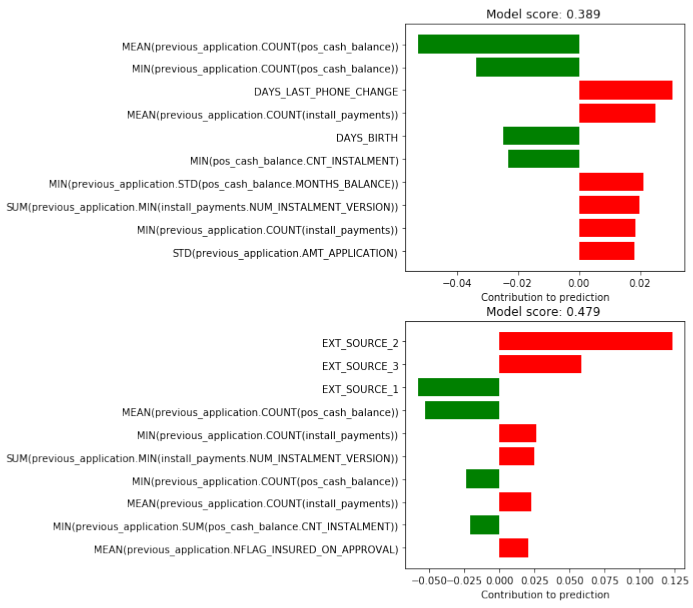


Figure 47: Example of the detection strong predictors for the SHAP method

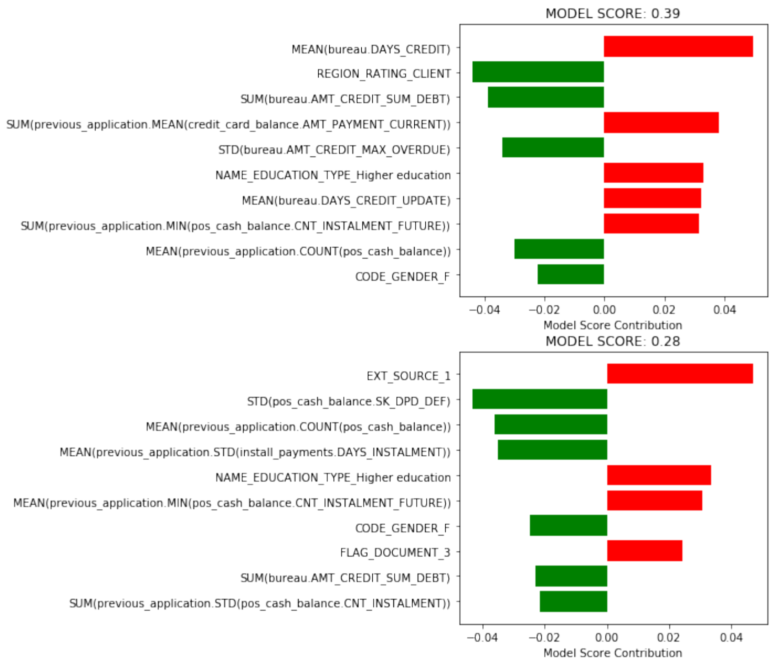


Figure 48: Example of the detection strong predictors for the LIME method

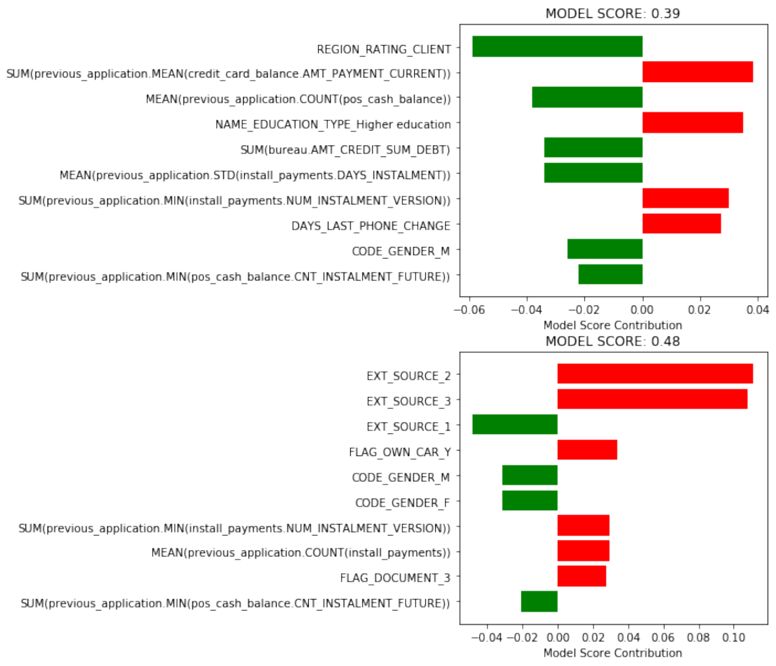


Figure 49: Example of the detection strong predictors for the LIME method