DEGREE PROJECT IN MATHEMATICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2019

Loan Default Prediction using Supervised Machine Learning Algorithms

DARIA GRANSTRÖM

JOHAN ABRAHAMSSON

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Loan Default Prediction using Supervised Machine Learning Algorithms

DARIA GRANSTRÖM

JOHAN ABRAHAMSSON

Degree Projects in Mathematical Statistics (30 ECTS credits)
Master's Programme in Applied and Computational Mathematics (120 credits)
KTH Royal Institute of Technology year 2019
Supervisor at Nordea; Aron Moberg, Lee MacKenzie Fischer
Supervisor at KTH: Tatjana Pavlenko
Examiner at KTH: Tatjana Pavlenko

TRITA-SCI-GRU 2019:073
MAT-E 2019:30

Royal Institute of Technology
School of Engineering Sciences
KTH SCI
SE-100 44 Stockholm, Sweden
URL: www.kth.se/sci

Abstract

It is essential for a bank to estimate the credit risk it carries and the magnitude of exposure it has in case of non-performing customers. Estimation of this kind of risk has been done by statistical methods through decades and with respect to recent development in the field of machine learning, there has been an interest in investigating if machine learning techniques can perform better quantification of the risk. The aim of this thesis is to examine which method from a chosen set of machine learning techniques exhibits the best performance in default prediction with regards to chosen model evaluation parameters. The investigated techniques were Logistic Regression, Random Forest, Decision Tree, AdaBoost, XGBoost, Artificial Neural Network and Support Vector Machine. An oversampling technique called SMOTE was implemented in order to treat the imbalance between classes for the response variable. The results showed that XGBoost without implementation of SMOTE obtained the best result with respect to the chosen model evaluation metric.

Referat

Det är nödvändigt för en bank att ha en bra uppskattning på hur stor risk den bär med avseende på kunders fallissemang. Olika statistiska metoder har använts för att estimera denna risk, men med den nuvarande utvecklingen inom maskininlärningsområdet har det väckt ett intesse att utforska om maskininlärningsmetoder kan förbättra kvaliteten på riskuppskattningen. Syftet med denna avhandling är att undersöka vilken metod av de implementerade maskininlärningsmetoderna presterar bäst för modellering av fallissemangprediktion med avseende på valda modellvaldieringsparametrar. De implementerade metoderna var Logistisk Regression, Random Forest, Decision Tree, AdaBoost, XGBoost, Artificiella neurala nätverk och Stödvektormaskin. En översamplingsteknik, SMOTE, användes för att behandla obalansen i klassfördelningen för svarsvariabeln. Resultatet blev följande: XGBoost utan implementering av SMOTE visade bäst resultat med avseende på den valda metriken.

Acknowledgements

We would like to devote an extra gesture of gratitude towards: our supervisors Tatjana Pavlenko for professional academic guidance, Aron Moberg and Lee MacKenzie Fischer for enabling the research and helpful mentoring at Nordea.

Contents

List of Figures

List of Tables

List of Abbreviations

AdaBoost	Adaptive Boosting
ANN	Artificial Neural Network
AUC	Area Under the Curve
CV	Cross-validation
DC	Dummy Classifier
DT	Decision Tree
EAD	Exposure At Default
EBA	European Banking Authority
EL	Expected Loss
FN	False Negatives
FP	False Positives
FPR	False Positive Rate
IRB	Internal Ratings Based approach
PD	Probability of Default
ReLU	Rectified Linear Unit
RF	Random Forest
RFE	Recursive Feature Elimination
ROC	Receiver Operator Characteristic
SME	Small and Medium-sized Enterprises
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machines
TN	True Negatives
TP	True Positives

TPR	True Positive Rate
XGBoost	eXtreme Gradient Boosting

Chapter 1

Introduction

In this chapter an overview of what aim of the thesis is provided. The topics discussed within this chapter are the thesis' background, purpose and scope.

1.1 Background

A recent development of machine learning techniques and data mining has led to an interest of implementing these techniques in various fields [33]. The banking sector is no exclusion and the increasing requirements towards financial institutions to have robust risk management has led to an interest of developing current methods of risk estimation. Potentially, the implementation of machine learning techniques could lead to better quantification of the financial risks that banks are exposed to.

Within the credit risk area, there has been a continuous development of the Basel accords, which provides frameworks for supervisory standards and risk management techniques as a guideline for banks to manage and quantify their risks. From Basel II, two approaches are presented for quantifying the minimum capital requirement such as the standardized approach and the internal ratings based approach (IRB) [3]. There are different risk measures banks consider in order to estimate the potential loss they may carry in future. One of these measures is the expected loss (EL) a bank would carry in case of a defaulted customer. One of the components involved in EL-estimation is the probability if a certain customer will default or not. Customers in default means that they did not meet their contractual obligations and potentially might not be able to repay their loans [43]. Thus, there is an interest of acquiring a model that can predict defaulted customers. A technique that is widely used for estimating the probability of client default is Logistic Regression [44]. In this thesis, a set of machine learning methods will be investigated and studied in order to test if they can challenge the traditionally applied techniques.

1.2 Purpose

The objective of this thesis is to investigate which method from a chosen set of machine learning techniques performs the best default prediction. The research question is the following

- *For a chosen set of machine learning techniques, which technique exhibits the best performance in default prediction with regards to a specific model evaluation metric?*

1.3 Scope

The scope of this paper is to implement and investigate how different supervised binary classification methods impact default prediction. The model evaluation techniques used in this project are limited to precision, sensitivity, F-score and AUC score. The reasons for choosing these metrics will be explained in more detail in section 2.10. The classifiers that will be implemented and studied are:

- Logistic Regression
- Artificial Neural Network
- Decision Tree
- Random Forest
- XGBoost
- AdaBoost
- Support Vector Machine

The project will be performed at Nordea and thus the internal data of Nordea's customers will be used in order to conduct the research. With the regards to this fact, the results presented in this thesis will be biased towards the profile of Nordea's clients, specifically their location and other behavioral factors. The majority of Nordea's clients are situated in Nordic countries, and thus it will be impacted mainly by the behaviour of clients from these countries.

Chapter 2

Theory

In the theory section, relevant theory behind the chosen classification methods is explained. Background needed for understanding the implemented variable selection techniques and chosen model validation methods are also provided in this chapter.

2.1 Formulation of a Binary Classification Problem

Binary classification refers to the case when the input to a model is classified to belong to one of two chosen categories. In this project, customers belong either to the non-default category or to the default category. The categories can therefore be modeled as a *binary* random variable $Y \in \{0, 1\}$, where 0 is defined as non-default, while 1 corresponds to default. The random variable Y_i is the target variable and will take the value of y_i, where i corresponds to the ith observation in the data set. For some methods, the variable $\bar{y}_i = 2y_i - 1$ will be used, since these methods require the response variable to take the values $\bar{y}_i \in \{-1, 1\}$.

The rest of the information about the customers, such as the products the customers posses, account balances and payments in arrears can be modeled as the input variables. These variables are both real numbers and categories and are often referred to as $features$ or *predictors*. Let $X_i \in \mathbb{R}^p$ denote a real valued random input vector and an observed feature vector be represented by $\mathbf{x}_i = [x_{i1}, x_{i2}, ..., x_{ip}]^\top$, where p is the total number of features. Then the observation data set with N samples can be expressed as $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$.

With this setup, it makes it feasible to fit a *supervised* machine learning model that relates the response to the features, with the objective of accurately predicting the response for future observations [14]. The main characteristic of supervised machine learning is that the target variable is *known* and therefore an inference between the target variable and the predictors can be made. In contrast, *unsupervised* machine learning deals with the challenge where the predictors are measured but the target variable is *unknown*.

The chosen classification methods in this project are Logistic Regression, Artificial Neural Network, Decision Tree, Random Forest, XGBoost, AdaBoost and Support Vector Machine. The theory for these classifiers will be explained in more detail in the sections below.

2.2 Logistic Regression

Logistic Regression aims to classify an observation based on its modelled posterior probability of the observation belonging to a specific class. The posterior probability for a customer to be in the default class with a given input $\mathbf{x}_i$ can be obtained with the logistic function as [15]

$$P(Y_i = 1|X_i = \mathbf{x}_i) = \frac{e^{\beta_0 + \boldsymbol{\beta}^\top \mathbf{x}_i}}{1 + e^{\beta_0 + \boldsymbol{\beta}^\top \mathbf{x}_i}}, \tag{2.1}$$

where the parameters β_0 and $\boldsymbol{\beta}$ are parameters of a linear model with β_0 denoting an intercept and $\boldsymbol{\beta}$ denoting a vector of coefficients, $\boldsymbol{\beta} = [\beta_1, \beta_2, ..., \beta p]^\top$. The logistic function from Equation (2.1) is derived from the relation between the *log-odds* of $P(Y_i = 1|X_i = \mathbf{x}_i)$ and a linear transformation of $\mathbf{x}_i$, that is

$$\log \frac{P(Y_i = 1|X_i = \mathbf{x}_i)}{1 - P(Y_i = 1|X_i = \mathbf{x}_i)} = \beta_0 + \boldsymbol{\beta}^\top \mathbf{x}_i. \tag{2.2}$$

The class prediction can then be defined as

$$\hat{y}_i = \begin{cases} 1, & \text{if } P(Y_i = 1|X_i = \mathbf{x}_i) \geq c \\ 0, & \text{if } P(Y_i = 1|X_i = \mathbf{x}_i) < c \end{cases}, \tag{2.3}$$

where c is a threshold parameter of the decision boundary which is usually set to $c = 0.5$ [10]. Further, in order to find the parameters β_0 and $\boldsymbol{\beta}$, the maximization of the log-likelihood of Y_i is performed. After some manipulation of Equation (2.1), the expression can be rewritten as

$$p(\mathbf{x}_i; \beta_0, \boldsymbol{\beta}) = P(Y_i = 1|X_i = \mathbf{x}_i; \beta_0, \boldsymbol{\beta}) = \frac{1}{1 + e^{-(\beta_0 + \boldsymbol{\beta}^\top \mathbf{x}_i)}}. \tag{2.4}$$

Since $P(Y_i = 1|X_i = \mathbf{x}_i; \beta_0, \boldsymbol{\beta})$ completely specifies the conditional distribution, the multinomial distribution is appropriate as the likelihood function [18]. The log-likelihood function for N observations can then be defined as

$$
\begin{aligned}
l(\beta_0, \boldsymbol{\beta}) &= \sum_{n=1}^{N} [y_n \log p(\mathbf{x}_n; \beta_0, \boldsymbol{\beta}) + (1 - y_n) \log(1 - p(\mathbf{x}_n; \beta_0, \boldsymbol{\beta}))] \\
&= \sum_{n=1}^{N} [y_n(\beta_0 + \boldsymbol{\beta}^\top \mathbf{x}_n) - \log(1 + e^{(\beta_0 + \boldsymbol{\beta}^\top \mathbf{x}_n)})]. \qquad (2.5)
\end{aligned}
$$

Let $\theta = \{\beta_0, \boldsymbol{\beta}\}$ and assume that $\mathbf{x}_n$ includes the constant term 1 to accommodate β_0. Then, in order to maximize the log-likelihood, take the derivative of l and set to zero

$$
\frac{\partial l(\theta)}{\partial \theta} = \sum_{n=1}^{N} \mathbf{x}_n (y_n - p(\mathbf{x}_n; \theta)) = 0. \qquad (2.6)
$$

Equation (2.6) generates $p + 1$ equations nonlinear in θ. To solve these equations, the *Newton-Rahpson* method can be used. In order to use this method, the second-derivative must be calculated

$$
\frac{\partial^2 l(\theta)}{\partial \theta \partial \theta^\top} = -\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^\top p(\mathbf{x}_n; \theta)(1 - p(\mathbf{x}_n; \theta)). \qquad (2.7)
$$

A single Newton-Rahpson update will then be performed as

$$
\theta^{new} = \theta^{old} - \left(\frac{\partial^2 l(\theta)}{\partial \theta \partial \theta^\top} \right)^{-1} \frac{\partial l(\theta)}{\partial \theta}. \qquad (2.8)
$$

2.3 Decision Trees

A decision tree algorithm binary splits the feature space into subsets in order to divide the samples into more homogeneous groups. This can be implemented as a tree structure, hence the name decision trees. An example of a two-dimensional split feature space and its corresponding tree can be seen in Figure 2.1.

The terminal nodes in the tree in Figure 2.1 are called leaves and are the predictive outcomes. In this particular example, a regression tree which predicts quantitative outcomes has been used. However, classification trees that predict qualitative outcomes rather than quantitative will be used in this project.

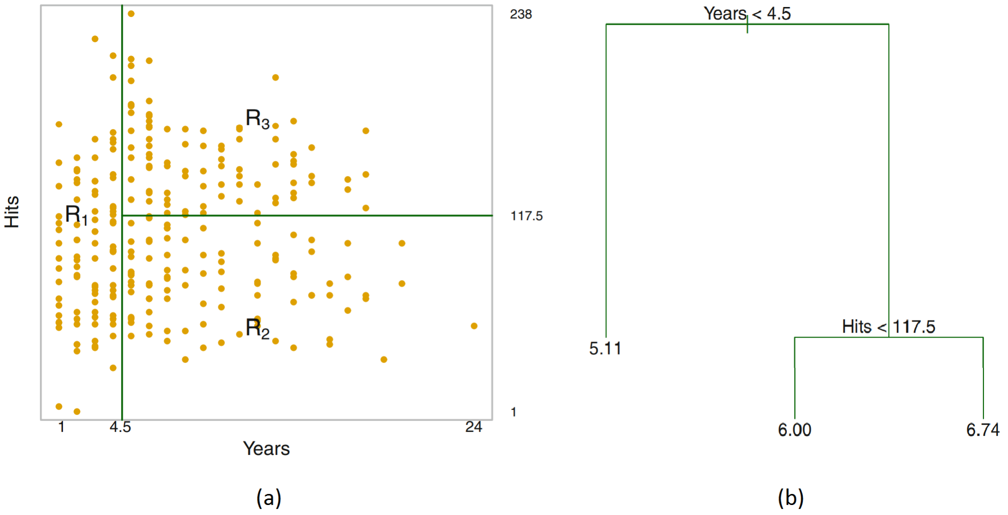


Figure 2.1: (a) Two dimensional feature space split into three subsets. (b) Corresponding tree to the split of the feature space. Source of figure: [16].

In a subset of the feature space, represented by the region R_m with N_m number of observations, let the indicator function be $I(\cdot)$ and

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{\mathbf{x}_i \in R_m} I(y_i = k) \tag{2.9}$$

be the fraction of class k observations in R_m [19]. Then the observations lying in R_m will be predicted to belong to class $k(m) = \arg\max_k \hat{p}_{mk}$. Since the *Gini index*, defined by

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}), \tag{2.10}$$

is amenable for numerical optimization [20], it will be chosen as the criterion for binary splitting.

2.4 Random Forest

Before describing the random forest classifier, the notions of *bootstrapping* and *bagging* will be introduced. Bootstrapping is a resampling with replacement method and is used for creating new synthetic samples in order to make an inference of an analyzed population. An example could be to investigate the variability of the mean of a population. This is done by resampling the original sample B times with replacement, then compute a sample mean for each of the B new samples, and lastly compute the variance for the sample means.

Bagging is an abbreviation for bootstrap aggregation and its purpose is to reduce the variance of a statistical learning method. The method bootstraps the original data set, fits separate models for each bootstrapped data set and takes the average of the predictions made by each model. For a given data set z, the method can be expressed as

$$\hat{f}_{bag}(z) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(z), \tag{2.11}$$

where B is the total amount of bootstrapped data sets and $\hat{f}^{*b}(z)$ is a model used for the bth bootstrapped data set. In a classification setting, instead of taking the average of the models, a majority vote is implemented. When applying bagging to decision trees, the following should be considered. If there is one strong predictor in the data set along with moderately strong predictors, most of the top splits will be done based on the strong predictor. This leads to fairly similar looking trees that are highly correlated. Averaging highly correlated trees does not lead to a large reduction of variance.

The *random forest classifier* has the same setup as bagging when building trees on bootstrapped data sets but overcomes the problem of highly correlated trees. It decorrelates the trees by taking a random sample of m predictors from the full set of p predictors at each split and uses randomly one among the m predictors to split [17]. An example of a classification for a random forest classifier is shown in Algorithm 1.

2.5 Boosting

Boosting works in a similar way as bagging regarding combining models and creating a single predictive model, but it does not build trees independently, it builds trees *sequentially* [17]. Building trees sequentially means that information from the previous fitted tree is used for fitting the current tree. Rather than fitting separate trees on separate bootstrapped data sets, each tree is fit on a modified version of the original data set [17].

2.5.1 AdaBoost

AdaBoost stands for Adaptive Boosting and combines weak classifiers into a strong classifier. A weak classifier refers to a model that is slightly better than classifying

```
Algorithm 1: Random Forest Algorithm
Input: A training data set $z$ and an observation from a test set, $\mathbf{x}_{test}$.
The original training data $z$ is bootstrapped into $B$ different data sets, such
    that $z_1^*, z_2^*, ..., z_B^*$ represents the bootstrapped data sets.
$B$ submodels are fitted on the bootstrapped data sets, such that
    $\hat{f}_1(z_1^*), ..., \hat{f}_B(z_B^*)$.
For each split a submodel $\hat{f}_b(z_b^*)$ does, $m$ predictors are chosen randomly out
    of the $p$ predictors and one of the $m$ predictors is used for splitting.
For each submodel, an unseen observation from the test set, $\mathbf{x}_{test}$, is used for
    prediction, such that $\hat{f}_1(\mathbf{x}_{test}) \in \{0,1\}, ..., \hat{f}_B(\mathbf{x}_{test}) \in \{0,1\}$.
Create the final model $\hat{f}_{final}(\mathbf{x}_{test}) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(\mathbf{x}_{test})$.
if $\hat{f}_{final}(\mathbf{x}_{test}) \geq 0.5$ then
    $\hat{y}_{test} \leftarrow 1$
else
    $\hat{y}_{test} \leftarrow 0$
Output: $\hat{y}_{test}$
```

by flipping a coin, i.e., the accuracy is a bit higher than 0.5. Combining the weak classifiers can be defined as the following linear combination

$$C_{(m-1)}(\mathbf{x}_i) = \alpha_1 k_1(\mathbf{x}_i) + \cdots + \alpha_{m-1} k_{m-1}(\mathbf{x}_i), \tag{2.12}$$

where $\mathbf{x}_i$ is associated with the class $\bar{y}_i \in \{-1, 1\}$ and $k_j(\mathbf{x}_i) \in \{-1, 1\}$ is a weak classifier with its weight α_j [39]. At the mth iteration there is an added weak classifier, k_m with weight α_m, that enhances the boosted classifier

$$C_m(\mathbf{x}_i) = C_{(m-1)}(\mathbf{x}_i) + \alpha_m k_m(\mathbf{x}_i). \tag{2.13}$$

In order to determine the best k_m and its weight α_m, a loss function that defines the total error E of C_m is used and takes the following form

$$E = \sum_{i=1}^{N} e^{-\bar{y}_i C_m(\mathbf{x}_i)}, \tag{2.14}$$

where N is the total sample size. Letting $w_i^{(1)} = 1$ and $w_i^{(m)} = e^{-\bar{y}_i C_{m-1}(\mathbf{x}_i)}$ for $m > 1$, the following is obtained

$$E = \sum_{i=1}^{N} w_i^{(m)} e^{-\bar{y}_i \alpha_m k_m(\mathbf{x}_i)}. \tag{2.15}$$

Correctly classified data points takes the form of $\bar{y}_i k_m(\mathbf{x}_i) = 1$ and misclassified $\bar{y}_i k_m(\mathbf{x}_i) = -1$. This can be used to split the summation into

$$\begin{aligned} E &= \sum_{\bar{y}_i = k_m(\mathbf{x}_i)} w_i^{(m)} e^{-\alpha_m} + \sum_{\bar{y}_i \neq k_m(\mathbf{x}_i)} w_i^{(m)} e^{\alpha_m} \\ &= \sum_{i=1}^{N} w_i^{(m)} e^{-\alpha_m} + \sum_{\bar{y}_i \neq k_m(\mathbf{x}_i)} w_i^{(m)} (e^{\alpha_m} - e^{-\alpha_m}). \end{aligned} \tag{2.16}$$

The only part that depends on k_m in Equation (2.16) is $\sum_{\bar{y}_i \neq k_m(\mathbf{x}_i)} w_i^{(m)}$, it can also be realized that the k_m that minimizes $\sum_{\bar{y}_i \neq k_m(\mathbf{x}_i)} w_i^{(m)}$ also minimizes E. To determine the desired weight α_m that minimizes E with the k_m that was just determined, take the derivative of E with respect to the weight and set to zero

$$\frac{dE}{d\alpha_m} = \frac{d(\sum_{\bar{y}_i = k_m(\mathbf{x}_i)} w_i^{(m)} e^{-\alpha_m} + \sum_{\bar{y}_i \neq k_m(\mathbf{x}_i)} w_i^{(m)} e^{\alpha_m})}{d\alpha_m} = 0 \tag{2.17}$$

$$\implies \alpha_m = \frac{1}{2} \ln \left(\frac{\sum_{\bar{y}_i = k_m(\mathbf{x}_i)} w_i^{(m)}}{\sum_{\bar{y}_i \neq k_m(\mathbf{x}_i)} w_i^{(m)}} \right). \tag{2.18}$$

With the mth calculated weight α_m and the weak classifier k_m, the mth combined classifier can be obtained as in Equation (2.13). The output of the algorithm is then

$$\hat{y}_i = \begin{cases} 1, & \text{if } C_m(\mathbf{x}_i) \geq 0 \\ 0, & \text{if } C_m(\mathbf{x}_i) < 0 \end{cases}. \tag{2.19}$$

2.5.2 XGBoost

XGBoost is an abbreviation of eXtreme Gradient Boosting. One of the evident advantages of XGBoost is its scalability and faster model exploration due to the parallel and distributed computing [7]. In order to understand XGBoost's algorithm, some basic introduction to how gradient tree boosting methods works will be presented.

Let N be a number of samples in the data set with p features, $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ ($|\mathcal{D}| = N$, $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \{0, 1\}$). To predict the output, M additive functions are being used

$$\phi(\mathbf{x}_i) = \sum_{k=1}^{M} f_k(\mathbf{x}_i), \quad f_k \in \mathcal{S}, \quad \mathcal{S} = \{f(\mathbf{x}) = \mathbf{w}_{q(\mathbf{x})})\}, \tag{2.20}$$

where $\mathcal{S}$ is the classification trees' space, q is the structure of a tree and $q : \mathbb{R}^p \rightarrow T$, $\mathbf{w} \in \mathbb{R}^M$ [7]. Further, T is the number of leaves, f_k is an independent tree structure of q and leaf weights $\mathbf{w}$, which can also be viewed as a score for ith leaf, w_i. Learning is being executed by minimization of the regularized objective and is derived as the following equation

$$\mathcal{L}(\phi) = \sum_{i=1}^{N} l(y_i, \phi(\mathbf{x}_i)) + \sum_{k=1}^{M} \Omega(f_k), \tag{2.21}$$

where $\Omega(f)$ is defined as follows

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j}^{T} w_j^2. \tag{2.22}$$

The function $\Omega(f)$ penalizes the complexity of the model by the parameter γ, which penalizes the number of leaves, and λ which penalizes the leaf weights. The loss function l measures the difference between the prediction $\phi(\mathbf{x}_i)$ and the target y_i [7]. Further, let $\phi(\mathbf{x}_i)^{(t)}$ be the prediction of the ith observation at the t-th iteration, then f_t is needed to add in order to minimize the following objective

$$\mathcal{L}^{(t)} = \sum_{i=1}^{N} l(y_i, \phi(\mathbf{x}_i)^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t), \tag{2.23}$$

where f_t is chosen greedily so that it improves the model the most. Second-order approximation can be used to quickly optimize the objective in the general setting [7]

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{N} [l(y_i, \phi(\mathbf{x}_i)^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t), \tag{2.24}$$

where $g_i = \partial_{\phi(\mathbf{x}_i)^{(t-1)}} l(y_i, \phi(\mathbf{x}_i)^{(t-1)})$ and $h_i = \partial^2_{\phi(\mathbf{x}_i)^{(t-1)}} l(y_i, \phi(\mathbf{x}_i)^{(t-1)})$. Simplification of the function can be made by removing a constant term $l(y_i, \phi(\mathbf{x}_i)^{(t-1)})$. and by expanding the Ω function the following expression can be obtained

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{N} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j}^{T} w_j^2. \tag{2.25}$$

Let $I_j = \{i | q(\mathbf{x}_i) = j\}$ be the instance of leaf j. Further, the equation is being simplified to

$$\tilde{\mathcal{L}}^{(t)} = \sum_{j}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2)] + \gamma T. \tag{2.26}$$

Now, the expression for the optimal weight w_j^* can be derived from Equation (2.26)

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda}. \tag{2.27}$$

Thus, the optimal value is given by

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T. \tag{2.28}$$

The final classification is then

$$\hat{y}_i = \begin{cases} 1, & \text{if } \phi(\mathbf{x}_i) \geq c \\ 0, & \text{if } \phi(\mathbf{x}_i) < c \end{cases}, \tag{2.29}$$

where c is a chosen decision boundary and $\phi(\mathbf{x}_i) \in (0,1)$. Further, in order to find the split the Exact Greedy Algorithm is used [7]. There are also other algorithms that can be used as alternatives for split finding such as the Approximate Algorithm and the Sparsity Aware Algorithm [7].

2.6 Artificial Neural Networks

Artificial neural networks (ANN) is originally inspired by how a human brain works and is intended to replicate its learning process [23]. A neural network consists of an input layer, an output layer and a number of hidden layers (see Figure 2.2).

The input layer is made of p predictors $x_1, x_2, ..., x_p$ and $\mathbf{x}_i$ is an arbitrary ith observation such that $\mathbf{x}_i = [x_{i1}, x_{i2}, ..., x_{ip}]^\top$. For K-class classification problems, K is the number of target measurements Y_k and $k = 1, ..., K$ represented in as a binary variable of either 0 or 1 for the kth class [21]. The target Y_k is a function derived

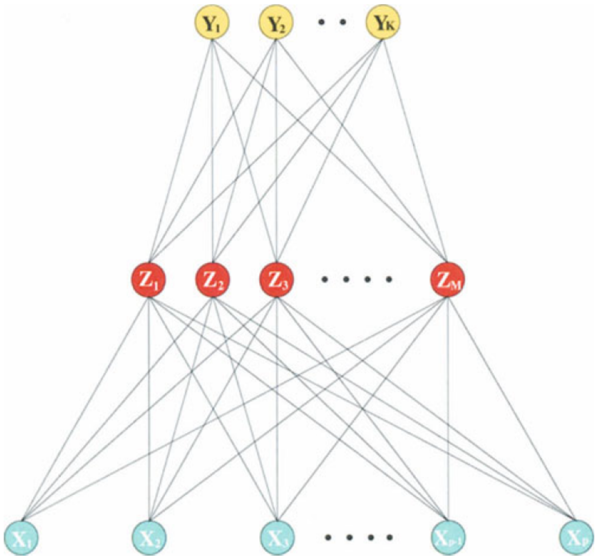


Figure 2.2: Structure of a single hidden layer, feed-forward neural network. Source of figure: [21].

from linear combinations of a variable Z_m, which in turn is originated from linear combinations of the inputs

$$Z_m = f(\alpha_{0m} + \boldsymbol{\alpha}_m^\top \mathbf{x}_i), \quad m = 1, ..., M, \tag{2.30}$$

where $\boldsymbol{\alpha}_m = [\alpha_{m1}, ..., \alpha_{mp}]^\top$ and M is the total number of hidden units. The activation function $f(\cdot)$ in Equation (2.30) is a Rectified Linear Unit (ReLU) function $f(v) = max(0, v+\epsilon)$ with $\epsilon \sim N(0, \sigma_l(v))$[34], where $\sigma_l(v)$ is the variance of ϵ. Then, a linear transformation of Z is performed

$$H_k = \beta_{0k} + \boldsymbol{\beta}_k^\top Z, \quad k = 1, ..., K, \tag{2.31}$$

where $Z = [Z_1, Z_2, ..., Z_M]^\top$ and $\boldsymbol{\beta}_k = [\beta_{k1}, ..., \beta_{km}]^\top$. Further, the final transformation of the vector $H = [H_1, H_2, ..., H_K]^\top$ is being done by a sigmoid function $\sigma_k(H)$

$$g_k(\mathbf{x}_i) = \sigma_k(H) = \frac{1}{1 + e^{-H_k}}. \tag{2.32}$$

The complete set of weights θ is $\{\alpha_{0m}, \boldsymbol{\alpha}_m; m = 1, ..., M\}$, $M(p+1)$, and $\{\beta_{0m}, \boldsymbol{\beta}_k; k = 1, ..., K\}$, $K(M+1)$. They are unknown and the intention is to find the most optimal values for them, so that the the model fits the training data well. For classification problems, the cross-entropy error function is defined as follows

$$R(\theta) = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} \log g_k(\mathbf{x}_i), \tag{2.33}$$

with the respective classifier $G(\mathbf{x}_i) = \arg\max_k g_k(\mathbf{x}_i)$. In this project, $K = 2$, where $k = 1$ is defined as non-default and $k = 2$ corresponds to default. In order to find the most optimal solution, the back-propagation algorithm is used, which in turn makes use of gradient descent for finding the global minimum.

The back-propagation algorithm works in the following way. The derivatives of the error function with respect to the weights should be found. Let $z_{mi} = f(\alpha_{0m}+\boldsymbol{\alpha}_m^\top \mathbf{x}_i)$ derived from Equation (2.30) and $\mathbf{z}_i = [z_{1i}, z_i, ..., z_{Mi}]$. Then the cross-entropy error can be expressed as

$$\begin{aligned} R(\theta) &\equiv \sum_{i=1}^{N} R_i \\ &= -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} \log g_k(\mathbf{x}_i), \end{aligned} \tag{2.34}$$

with derivatives

$$\frac{\partial R_i}{\partial \beta_{km}} = -\frac{y_{ik}}{g_k(\mathbf{x}_i)}\sigma_k'(\boldsymbol{\beta}_k^\top \mathbf{z}_i) z_{mi}, \tag{2.35}$$

$$\frac{\partial R_i}{\partial \alpha_{ml}} = -\sum_{k=1}^{K} \frac{y_{ik}}{g_k(\mathbf{x}_i)}\sigma_k'(\boldsymbol{\beta}_k^\top \mathbf{z}_i)\beta_{km} f'(\boldsymbol{\alpha}_m^\top \mathbf{x}_i) x_{il}. \tag{2.36}$$

The gradient descent update consequently takes the following form at the $(r+1)$th iteration

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}}, \tag{2.37}$$

$$\alpha_{ml}^{(r+1)} = \alpha_{ml}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{ml}^{(r)}}, \tag{2.38}$$

where γ_r is the learning rate. In order to avoid overfitting, the stopping criterion should be introduced before the global minimum is being reached [21]. As mentioned above, at the start of training, the model is highly linear due to the starting point of weights and, thus, it might lead at the end might lead to the shrinkage of the model. In this case, the penalty is being introduced to the the error function

$$R(\theta) + \lambda J(\theta), \tag{2.39}$$

where λ is a positive tuning parameter and $J(\theta)$ is either

$$J(\theta) = \sum_{km} \beta_{km}^2 + \sum_{ml} \alpha_{ml}^2, \tag{2.40}$$

considering the weight decay method of regularization [21] or

$$J(\theta) = \sum_{km} \frac{\beta_{km}^2}{1 + \beta_{km}^2} + \sum_{ml} \frac{\alpha_{ml}^2}{1 + \alpha_{ml}^2}, \tag{2.41}$$

if the weight elimination penalty is being used instead. The Adam algorithm will be implemented to perform a stochastic optimization in order to find the optimal weights [28]. The more detailed description of the algorithm can be found in Algorithm 4 in Appendix B.

One of the critiques ANN algorithm has received is that it is relatively slow to train. Further, relying on the back-propagation algorithm, sometimes it is quite unstable due to the tuning parameter should be adjusted in a way that the algorithm reaches the final solution and not oversteps it [25].

2.7 Support Vector Machine

The Support Vector Machine (SVM) is an algorithm that involves creating a hyperplane for classification. In order to classify an object, a set of features is used. Thus, if there are p features, the hyperplane will lie in p-dimensional space [41]. The hyperplane is created by the optimization performed by an SVM, which in turn maximizes the distance from the closest points, also called support vectors. Let $\mathbf{x}_i = [x_{i1}, ..., x_{ip}]^\top$ be an arbitrary observation feature vector in the training set, $\bar{y}_i$ the corresponding label to $\mathbf{x}_i$, $\mathbf{w}$ a weight vector $\mathbf{w} = [w_1, ..., w_p]^\top$ with $||\mathbf{w}||^2 = 1$, and b a threshold. Then following constraints are defined for the classification problem [8]:

$$\mathbf{w}^\top \mathbf{x}_i + b > 0 \quad \text{for} \quad \bar{y}_i = +1 \tag{2.42}$$

$$\mathbf{w}^\top \mathbf{x}_i + b < 0 \quad \text{for} \quad \bar{y}_i = -1. \tag{2.43}$$

Let $f(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i + b$, then the output of the model $\hat{y}_i$ is defined follows

$$\hat{y}_i = \begin{cases} 1 & \text{for } f(\mathbf{x}_i) \geq 0 \\ 0 & \text{for } f(\mathbf{x}_i) < 0 \end{cases}. \tag{2.44}$$

For margin maximization, instead of using $||\mathbf{w}||^2 = 1$, the lower bound on the margin and the optimization problem can be defined for minimization of $||\mathbf{w}||^2$. The constraints for the optimization problem are derived from the inequalities (2.42) and (2.43) can be presented as follows

$$\bar{y}_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1. \tag{2.45}$$

In some cases, it is relevant to implement a soft margin, which allows some points to lie on the wrong side of the hyperplane or between the support vector and the origin in order to provide a more robust model. A cost parameter C may also be introduced, which plays a role of assigning penalty to errors, where $C > 0$. Then the objective function to minimize takes the following form

$$||\mathbf{w}||^2 + C \sum_i \xi_i, \tag{2.46}$$

where ξ_i is a slack variable. The constraints to the optimization problem now are as follows [8]

$$\bar{y}_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \tag{2.47}$$

where $\xi_i \geq 0$. For non-linear SVM, a kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ can be introduced. In this project, a radial basis function will be used as a kernel. Thus, a hypersphere with radius R_{SVM} and center a is introduced, such that the minimization problem takes the following form [31]

$$R_{SVM}^2 + C \sum_i \xi_i, \tag{2.48}$$

where all patterns are grouped by

$$||\mathbf{x}_i - a||^2 \leq R_{SVM}^2 + \xi_i. \tag{2.49}$$

Lagrangian dual problem is used for solving this optimization problem, where the maximization of following objective with respect to parameter α_i is done

$$L = \sum_i \alpha_i (\mathbf{x}_i^\top \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i^\top \mathbf{x}_i), \tag{2.50}$$

where $0 \leq \alpha_i \leq C$ for all i. Thus, the center of the hypershpere is estimated by

$$a = \sum_i \alpha_i \Phi(\mathbf{x}_i), \tag{2.51}$$

where $\Phi(\mathbf{x}_i)$ is a mapping to a new space.

2.8 Feature Selection Methods

In the data given by Nordea, features are presented as both continuous and categorical variables. Thus, in order to understand how features correlate with each other and the response variable, implemented methods for feature selection should process both continuous and categorical variables simultaneously. That is why it is been decided to use following methods for feature selection: *Feature selection with Kendall's Tau Coefficient Analysis* and *Recursive Feature Elimination.*

2.8.1 Correlation Analysis with Kendall's Tau Coefficient

The Kendall's Tau rank correlation coefficient measures the ordinal association between two measured variables, thus, it is a univariate method of correlation analysis [13]. The ordinal association between two variables is done by calculating the proportion of concordant pairs minus the proportion of discordant pairs in a sample [11]. Any pair of observations (x_{ki}, y_i) and (x_{kj}, y_j), for $i < j$, are concordant if the product $(x_{ki} - x_{kj})(y_i - y_j)$ is positive, and discordant if this product is negative. The Kendall's Tau coefficient can then be defined as follows [35]

$$\tau = \frac{2}{n(n-1)} \sum_{i<j} \text{sgn}(x_{ki} - x_{kj}) \text{sgn}(y_i - y_j). \tag{2.52}$$

The Kendall's Tau rank correlation is a non-parametric test which means that it does not rely on any assumptions of distributions between the analyzed variables [42]. In a statistical hypothesis test, the null hypothesis implies an independence of X and Y and for large data sets, the distribution of the test can be approximated by a normal distribution with mean zero and variance [1]

$$\sigma_\tau^2 = \frac{2(2n+5)}{9n(n-1)}. \tag{2.53}$$

Further, for samples where $N > 10$, the transformation of τ into a Z value can be done for the null hypothesis test, such that Z value has a normal distribution with zero mean and standard deviation of 1, such that

$$Z_\tau = \frac{\tau}{\sigma_\tau} = \frac{\tau}{\sqrt{\frac{2(2n+5)}{9n(n-1)}}}. \tag{2.54}$$

When Z value is being computed, it should be compared with the chosen significance α-level, such that the null hypothesis can be rejected or not. In this project, α-level has chosen to be 0.1.

2.8.2 Recursive Feature Elimination

Recursive Feature Elimination (RFE) is a multivariate method of variable selection [38], which performs a backward selection of predictors [30]. The algorithm works in the following way. It starts with computing an importance score for each predictor in the whole data set. Let S be a sequence of the number of variables to include in the model. For each iteration, the number of S_i predictors which have been top-ranked are retained in the model [9]. Further, the importance scores are computed again and the performance is reassessed. The tuning parameter for RFE is the subset size and the subset of S_i predictors with the highest importance score is then used for fitting the final model. Thus, the performance criteria is optimized by the subset size with regards to the performed importance ranking. A more detailed description of the procedure can be found in Algorithm 2.

Further, it is also relevant to highlight that improvement for some models may be seen when applying RFE, while for others no remarkable difference in performance could exhibit. For example, random forest is one of the models that might benefit when RFE is applied [30]. One of the reasons is caused by the nature of model ensembles. Random forest tends not to exclude irrelevant predictors when a split is made, which requires a prior irrelevant feature elimination.

Thus, the aim is to test how RFE will impact on the implemented models. In contrast, when applying logistic regression for RFE, it is relevant to consider that

Algorithm 2: Recursive Feature Elimination

```
The model including all predictors is trained
The model performance is evaluated
Ranking of variables is computed
for Every subset size $S_i$,$i = 1, ..., S$ do
    The $S_i$ number of most relevant variables are retained
    The model using $S_i$ predictors is tuned on the training set
    The model performance is assessed
The performance profile for $S_i$ is computed
The number of predictors is defined
The model with the optimal $S_i$ is implemented
```

logistic regression is sensitive to class imbalances [2], which in fact exists in the given data set. Further, the given data is not particularly linear and that is why linear regression has not been considered either as a method for RFE. Therefore, the intention is to build RFE based on a random forest classifier. In order to choose the optimal number of variables, the F-score will be used as the evaluation score for RFE.

2.9 Treatment of Imbalanced Data with SMOTE Algorithm

In the data provided, a heavy class imbalance exhibits. An imbalanced data set contains of observations where the classes of the response variable are not approximately equally represented. In fraud detection for example, it is prevalent that the minority class is in the order of 1 to 100 [36]. In many studies there have been cases with orders of 1 to 100,000. The imbalance causes a problem when training machine learning algorithms since one of the categories is almost absent, hence poor predictions of new observations of the minority class are expected. In order to increase the performance of the algorithms there are different sampling techniques that can be used. One of them is called SMOTE and will be explained in the next paragraph.

SMOTE (Synthetic Minority Over-sampling Technique) is a sampling algorithm that creates synthetic data points for the minority class rather than sampling existing observations with replacement [36]. The method is applied on the continuous parameters for each sample in the minority class. A detailed description of the technique can be found in Algorithm 3 and a visual example of the synthetic data points can be seen in Figure 2.3. There are also some specifics that should be considered when implementing SMOTE with cross-validation, they will be discussed in section 2.11.1.

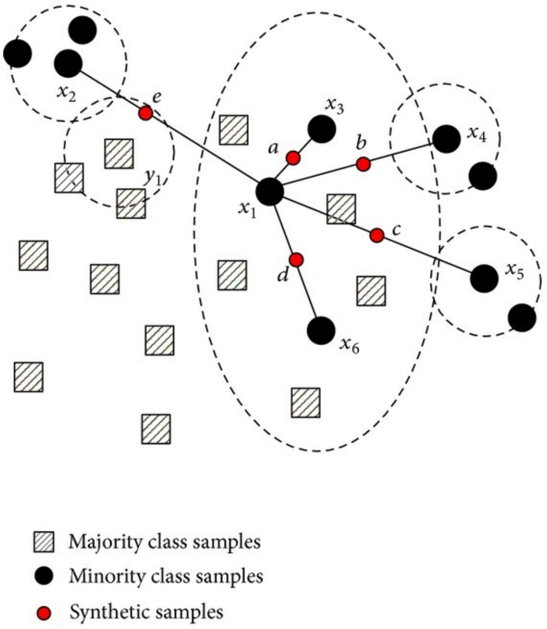


Figure 2.3: An example of a result using the SMOTE algorithm. Source of figure: [24].

Algorithm 3: SMOTE algorithm

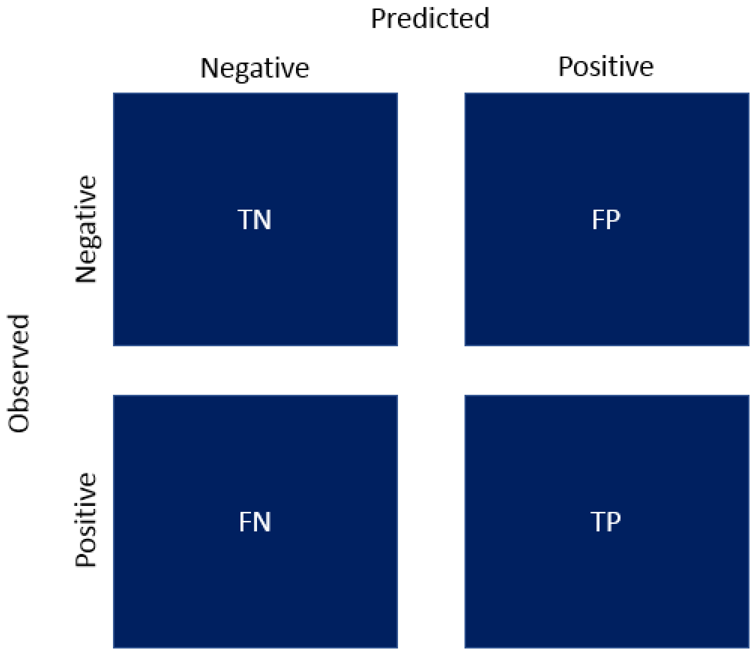
```
Take the $k$ nearest neighbors (that belong to the same class) of the
  considered sample
Randomly choose $n$ samples of these $k$ neighbors. The number $n$ is based on
  the requirement of the over-sampling, if 200% over-sampling is required,
  then $n = 2$
Compute differences between the feature vector of the considered sample and
  each of the $n$ neighbor feature vectors.
Multiply each of the differences with a random number between 0 and 1
Add these numbers separately to the feature vector of the considered sample
  in order to create $n$ new synthetic samples
Return new synthetic samples
```

2.10 Model Evaluation Techniques

2.10.1 Confusion Matrix

One common way to evaluate the performance of a model with binary responses is to use a confusion matrix. The observed cases of default are defined as positives and non-default as negatives [10]. The possible outcomes are then true positives (TP) if defaulted customers have been predicted to be defaulted by the model. True negatives (TN) if non-default customers have been predicted to be non-default.

False positives (FP) if non-default customers have been predicted to be defaulted, and false negatives (FN) if defaulted customers have been predicted to be non-default. A confusion matrix can be presented as in the Figure 2.4.



		Predicted	
		Negative	Positive
Observed	Negative	TN	FP
	Positive	FN	TP

Figure 2.4: Confusion matrix

From a confusion matrix there are certain metrics that can be taken into consideration. The most common metric is accuracy which is defined as the fraction of the total number of correct classifications and the total number of observations. It is mathematically defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}. \tag{2.55}$$

The issue with using accuracy as a metric is when applying it for imbalanced data. If the data set contains 99% of one class it is possible to get an accuracy of 99%, if all of the predictions are made for the majority class. A metric that is more relevant in the context of this project is specificity. It is defined as [6]

$$Specificity = \frac{TN}{FP + TN}, \tag{2.56}$$

and will be used for explaining the theory behind receiver operator characteristic curve and its area under the curve in section 2.10.2.

In terms of business sense, the aim is to achieve a trade-off between loosing money on non-performing customers and opportunity cost caused by declining of a potentially performing customer. Thus, there is a high relevance to analyze how sensitivity and precision are affected by various methods applied, as sensitivity is a measure of how many defaulted customers are captured by the model, while precision relates to the potential opportunity cost. Sensitivity and precision are defined in Equation (2.58) and (2.57) [12].

$$Sensitivity = \frac{TP}{TP + FN}, \tag{2.57}$$

$$Precision = \frac{TP}{TP + FP}. \tag{2.58}$$

Since sensitivity and precision are of equal importance in this project, a trade-off between these metrics is considered. The F-score is the weighted harmonic average of precision and sensitivity [12]. The definition of F-score can be expressed as

$$\begin{aligned} F &= (1+\beta^2)\frac{Precision \cdot Sensitivity}{Sensitivity + \beta^2 \cdot Precision} \\ &= \frac{(1+\beta^2)TP}{(1+\beta^2)TP + \beta^2 FN + FP}, \end{aligned} \tag{2.59}$$

where β is a weight parameter. As mentioned before, both measures of precision and sensitivity are equally relevant and therefore the weight is set to $\beta = 1$. Further, the F-score takes both of these measures into consideration, and thus performance of every method will be primarily evaluated and compared with the regards to this metric.

2.10.2 Area Under the Receiver Operator Characteristic Curve

Another way to evaluate results from the models is to analyze the Receiver Operator Characteristic (ROC) curve and its Area Under the Curve (AUC). In this section, the definition of ROC will be provided, followed by the explanation of AUC.

Let V_0 and V_1 denote two independent random variables with cumulative distribution functions F_0 and F_1 respectively. The random variables V_0 and V_1 describe the outcomes predicted by a model if a customer has defaulted or not. Let c be a threshold value for the default classification such that if the value from the model is greater or equal to c, a customer is classified as default and non-default otherwise.

Further, in this setting, sensitivity and specificity are defined then in the following way [37]

$$Sensitivity(c) = P(V_1 \geq c) = 1 - F_1(c), \tag{2.60}$$

$$Specificity(c) = P(V_0 < c) = F_0(c). \tag{2.61}$$

The ROC curve uses the false positive fraction in order to describe the trade-offs between sensitivity and (1-specificity). Let m express $1 - F_0(c)$, then the following definition for the ROC curve is obtained

$$ROC(m) = 1 - F_1\{F_0^{-1}(1 - m)\}, \tag{2.62}$$

where $0 \leq m \leq 1$ and $F_0^{-1}(1 - m) = \inf\{z : F_0(z) \leq 1 - m\}$. A ROC curve can also be summarized by AUC score, which represents an index for ROC curve and is defined in following way [6]

$$AUC = \int_0^1 ROC(m)dm. \tag{2.63}$$

2.11 Cross-validation

In order to prevent using the same information in the training phase and the evaluation phase of models, which makes the results less reliable, the data is divided into training set, validation set and test set [40]. The training set and validation set are used for finding the best model and the test set is only used for calculating the prediction performance of the best model. The test data will therefore be held out until the best model is obtained, this is called the *holdout method* [40]. Choosing the best model from a set of models can be done by a method called K-fold cross-validation (CV).

K-fold CV involves the procedure where the data set is divided in K roughly equal-sized sets or folds. One of them is set to be a validation set and the rest are the training set that a model is being fitted on [22]. The procedure is repeated K times and the validation error is being estimated for each time. For example, in Figure 2.5, K has been assigned the numerical value of 5, which means that there have been 5 iterations of the procedure.

Let $\kappa : \{1, ..., N\} \mapsto \{1, ..., K\}$ be a mapping function that shows an index of the partition to which observation i is assigned by the randomization and $k = 1, 2, ..., K$.

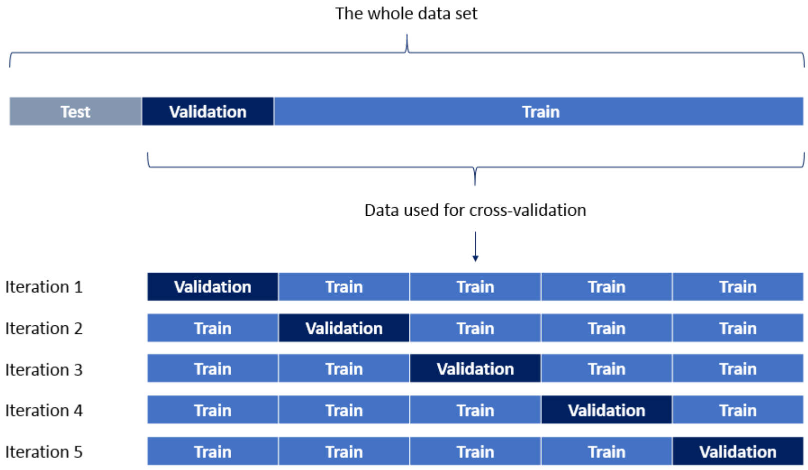


Figure 2.5: K-fold cross validation performed on a data set

For the kth fold, the model is fitted on the $K-1$ parts and the prediction error is calculated for the kth part. The fitted model is denoted as $\hat{f}^{-k}(\mathbf{x})$ with the kth part of the data removed, then the CV error is defined as follows

$$CV = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(\mathbf{x}_i)), \tag{2.64}$$

where $L(\cdot)$ is the loss function for the respective model. Given a set of competing models $f(\mathbf{x}, \alpha)$, with α denoting the index of the model, an exhaustive search for the best αth model can be performed. Let the αth fitted model be $\hat{f}^{-k}(\mathbf{x}_i, \alpha)$ with the kth part of the data removed, then the CV error becomes

$$CV(\alpha) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(\mathbf{x}_i, \alpha)). \tag{2.65}$$

The objective is to find the α that minimizes the validation error, denoted as $\hat{\alpha}$. This is also known as a hyperparameter search and will be implemented by looping through every combination of a chosen set of hyperparameter values for each machine learning method. When the final model $f(\mathbf{x}, \hat{\alpha})$ is obtained, where $\hat{\alpha}$ represents the best combination of hyperparameters, the performance of $f(\mathbf{x}, \hat{\alpha})$ will be calculated when predicting on the test set .

In this project, due to the class imbalance problem, stratified CV will be used in order to preserve proportional representation of the two classes in each fold. In binary classification problem, stratification in cross-validation is a technique which rearrange data that each fold contains roughly the same representation of classes between folds [27].

2.11.1 Implementation of Cross-validation with SMOTE

When using CV and applying an oversampling technique such as SMOTE, the following has to be considered. A simple oversampling technique duplicates observations of the minority class. If the oversampling is performed before CV, then the probability of getting duplicate observations in the validation set and the training set is fairly high [4]. The point of the validation set is to calculate the performance of the method on *unseen* observations. If oversampling is performed before CV there is a possibility that model has already ”seen” observations and therefore cause biased results. SMOTE is an oversampling technique that does not duplicate the observations but rather synthetically creates new ones. Even if the synthetic new data points generated by SMOTE are not duplicates of an observation, they are still based on an original observation and will therefore cause biased results when predicting on them. An example of a correct way of oversampling and use of CV following with an example of an incorrect way is visualized in Figure 2.6.

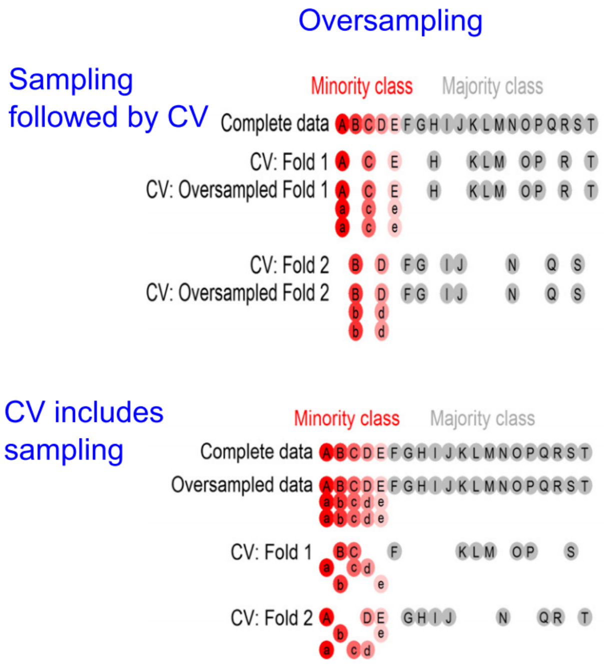


Figure 2.6: Upper: Correct way of oversampling and implementing CV. Lower: Incorrect way of oversampling and implementing CV. Source of figure: [4]

Chapter 3

Data Processing and Variable Selection

In this chapter, procurement of data, preprocessing of data and implementation of variable selection will be discussed.

3.1 Data Description

The data provided by Nordea contains their small and medium enterprise (SME) clients. Because of the clients' geographical locations and different currencies, it was decided to segment them into their respective countries, see Figure 3.1.

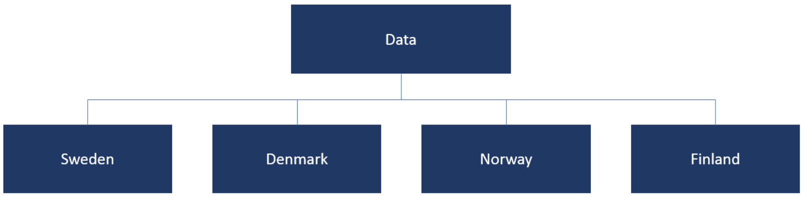


Figure 3.1: Segmentation of customers according to their geographical location

The parameters in the data were the same regardless of geographical location and were in total over 400 variables. Information derived from these variables could be, for example, what types of accounts a customer possessed and how these particular accounts *behaved* over a certain time span. From now, these variables will be denoted as *behavioral variables*. The time span of data was merged with a default flag indicating if the customer was in default or not in the following year. For example, if the observation period of a customer was between December 31, 2015 to December 31, 2016, then the performance period of the customer was of year 2017. This means that the customers who defaulted any time during 2017 were given a default flag.

3.2 Preprocessing of Data

As mentioned in the section 3.1, the data contained different types of accounts that the customers possessed. These accounts can be related to different financial products such as mortgages, loans or credit. Thus, in order to decrease the number of variables, all different types of financial products have been aggregated into one category, see Figure 3.2.

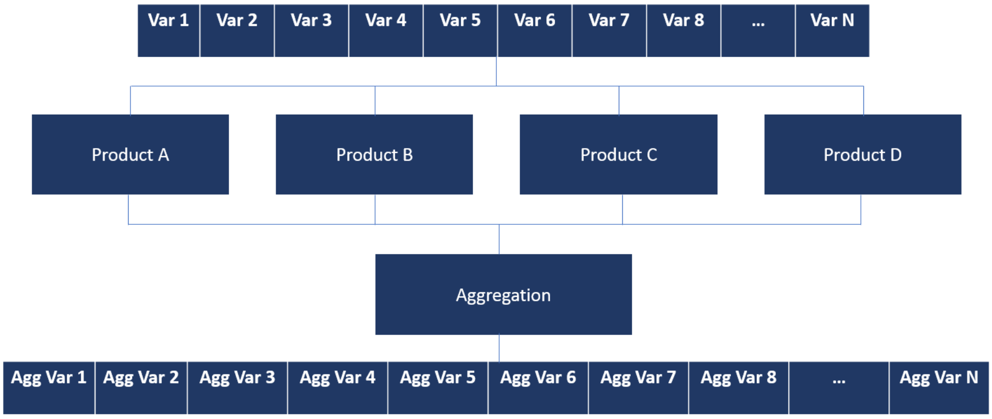


Figure 3.2: Visualization of the aggregation logic

Some of the created variables describe the same feature, but for different time spans. This means that the variables, for example $bf\_ag\_1\_1$, $bf\_ag\_1\_3$, $bf\_ag\_1\_6$ and $bf\_ag\_1\_12$, describe the same feature but represent different time periods.

The input data to all of the models were standardized according to section 3.2.1. Missing values were treated by doing a complete case analysis.

3.2.1 Standardization of Variables

In order to obtain invariant measures and familiarly similar scale between variables, standardization was applied. The standard score was calculated for each x as follows [26]

$$z_{ki} = \frac{x_{ki} - \overline{x}_k}{s_k}, \tag{3.1}$$

where $\overline{x}_k$ is the mean and s_k is the standard deviation of the sample for a specific feature k [29]. The mean of a sample $x_{k1}, x_{k2}, ..., x_{kn}$ is defined as

$$\overline{x}_k = \frac{1}{n}(\sum_{i=1}^{n} x_{ki}), \tag{3.2}$$

and the standard deviation of a sample is calculated in the following way

$$s_k = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_{ki} - \overline{x}_k)^2}. \tag{3.3}$$

The standardization parameters were fit on the training data and applied on the test data.

3.3 Variable Selection by Correlation Analysis with Kendall's Tau

In order to reduce the number of variables and treat multicollinearity, correlation analysis for the features against response and features against features was performed. The relationship between the continuous feature variables did not appear to follow any particular distribution. It was therefore decided to measure the correlation between them with the non-parametric correlation test Kendall's Tau coefficient. For an explanation of a non-parametric test and theory behind Kendall's Tau coefficient, see section 2.8.1.

The hypothesis was made that the feature variable groups that describe the same feature but represent different time periods have high correlation within the group. This is evident in Figure 3.3, where the majority of the groups indeed have a high correlation internally.

Because of the high internal correlation in each group, the decision was made to chose one feature variable from a respective group. The aim is to pick the variable, which correlates the most with the response variable. Thus, from each group the variable with the highest Kendall Tau correlation with the response variable was chosen. Parts of the results are shown in Table 3.1. The whole table can be seen in Appendix A.2.

Table 3.1: Correlation of feature variables against the response variable with Kendall's Tau

Feature name	Correlation with response
bf_ag_4_1	0.382857
bf_ag_7_1	0.381114
bf_ag_1_1	0.360268
bf_ag_10_3	0.356646
bf_ag_4_3	0.351934
bf_ag_7_3	0.346601
bf_ag_6_3	0.340912
bf_ag_1_3	0.340761
bf_ag_2_3	0.340727
⋮	⋮
bf_ag_3_3	-0.324351

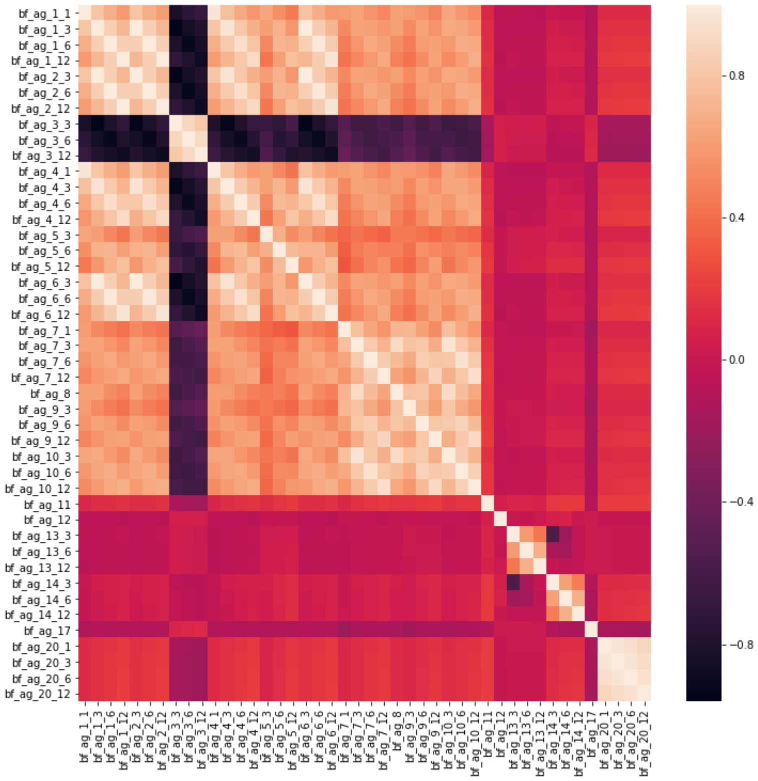


Figure 3.3: Heat map for variables grouped by feature

From Table 3.1 it can be seen that two behavioral variables $bf\_ag\_7\_1$ and $bf\_ag\_7\_3$ are from the same feature group. Since $bf\_ag\_7\_1$ has higher correlation with the response variable, it will be chosen as the only remaining variable from this feature group. After the selection of variables from each group was made, correlation was analyzed again in order to investigate how the chosen variables correlates with each other. Results can be found in Figure 3.4.

As can be seen in Figure 3.4, some variables have strong correlation, which should be treated. The next step in this case would be to choose a threshold for correlation allowed in the model. The threshold was chosen to 0.7, because pairwise correlation higher than 0.7 leads to unstable estimates and multicollinearity [5]. After the threshold was decided, the variable, which correlates the most with the response variable was kept, see Figure 3.5. For example, the behavioral variable $bf\_ag\_6\_3$ is perfectly correlated with the behavioral variable $bf\_ag\_2\_3$. According to Table 3.1, behavioral variable $bf\_ag\_6\_3$ correlates more with the response variable than the variable $bf\_ag\_2\_3$ and this is why the variable $bf\_ag\_6\_3$ should be retained from this pair.

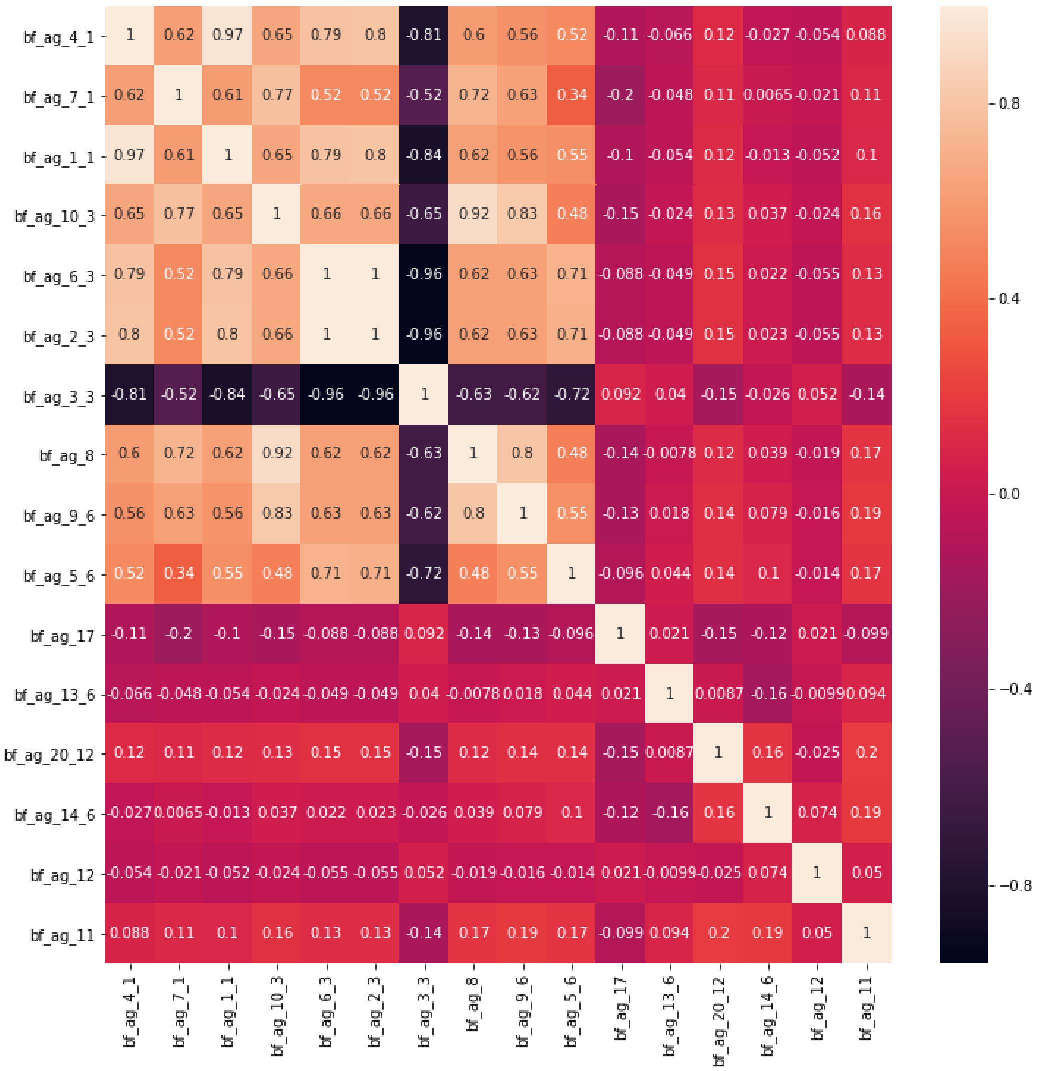


Figure 3.4: Heat map for selected variables

After this selection, a significance test described in section 2.8.1 was executed in order to see if the null hypothesis can be rejected. For the final selection of variables with α-level of 0.1, all variables from Figure 3.5 were tested with regards to their independence with the response variable. The result showed that all tests proved that there is a dependence between the feature variables and the response variable, and therefore the null hypothesis can be rejected. Thus, the variables shown in Figure 3.5 are included in the final model.

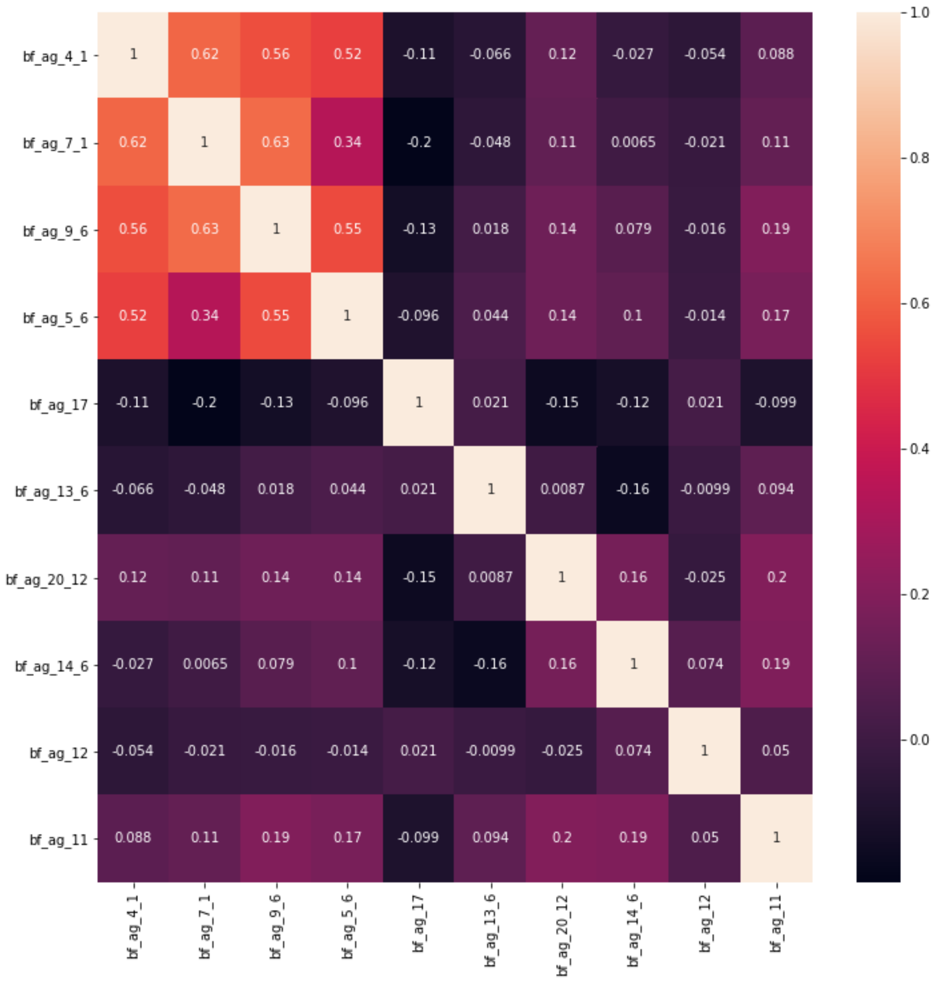


Figure 3.5: Final selected variables

3.4 Variable Selection by RFE

For the sake of getting a multivariate based variable selection, RFE was implemented. The variable selection by RFE was performed only on the behavioral variables with a Random Forest classifier with F-score as a scoring evaluation. Further, the shaded area in Figure 3.6 represents the variability of cross-validation with one standard deviation above and below the mean accuracy score drawn by the graph [9].

As seen in Figure 3.6, the highest score was obtained when the total number of variables is 29. It can be also noticed that after around 19 variables the score varies with small fluctuations and even after 7 variables it can be seen that the score does not vary much either. Thus, experiments were made with 7, 19 and 29 variables selected from the RFE. From now, RFE with 7, 19 and 29 respectively will be denoted as RFE_7, RFE_{19} and RFE_{29}. The aim is although to obtain a model with low complexity such that it will be easy to interpret.

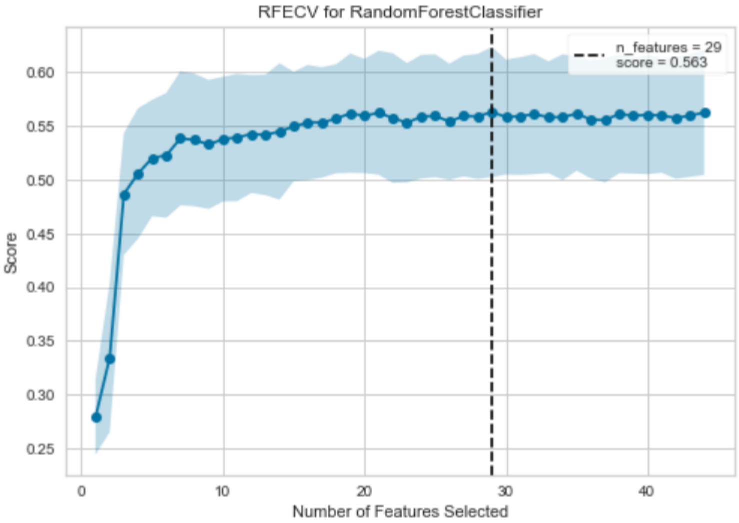


Figure 3.6: Graph over number of features selected with the corresponding F-score.

Chapter 4

Results

In this chapter, results obtained from different models will be presented and discussed

The results were obtained with the data set provided by Nordea. Four different data sets were studied, where one of them was obtained by performing variable selection with correlation analysis with Kendall's Tau and in total contained 13 variables. The other three data sets were generated by RFE and included 7, 19 and 29 features respectively. Half of the models were implemented with oversampling of the minority class with SMOTE. In these cases, the minority class was oversampled until its magnitude corresponded to 60% of the majority class.

4.1 Performance of the Methods

As mentioned in section 2.10, performance of the methods is evaluated by sensitivity, precision, F-score and ROC-score with the F-score as the primary metric. All of the results are presented in Table 4.1 and 4.2.

In Tables 4.1 and 4.2, it can be seen that the highest obtained F-score was for XGBoost without SMOTE and variable selection method of RFE_7 and RFE_{29}. With regards to precision and sensitivity, Artificial Neural Networks without SMOTE and RFE_{29} generated the best precision, while AdaBoost with SMOTE and RFE_7 had the best sensitivity.

On average, if the performance of SMOTE is to be evaluated with regards to F-score, it can be seen in Table 4.3 that the impact of SMOTE on average was marginal on the models performance. It should also be noticed that on SMOTE had a positive impact on the models performance when the number of variables was higher. That trend is seen when RFE_7 and correlation analysis with Kendall's Tau were implemented, where SMOTE had on average a negative impact on the F-score, when comparing to the cases where SMOTE were not implemented, while for RFE_{19} and RFE_{29} the reverse is true.

Table 4.1: Performance of Methods with regards to chosen performance measurements without SMOTE

Variable Selection	No. of features	Method	Precision (%)	Sensitivity (%)	F-score	AUC score
RFE	7	Logistic Regression	72.4	27.5	0.398	0.88
		Random Forest	76.2	48.9	0.596	0.87
		AdaBoost	68.9	46.6	0.556	0.88
		XGBoost	67.2	56.0	**0.611**	0.89
		Artificial Neural Network	80.0	32.2	0.459	0.88
		Support Vector Machine	81.8	30.0	0.438	0.75
		Decision Tree	59.1	45.9	0.517	0.75
RFE	19	Logistic Regression	78.3	30.3	0.437	0.89
		Random Forest	78.2	48.9	0.601	0.89
		AdaBoost	63.5	50.6	0.563	0.88
		XGBoost	66.5	54.5	0.599	0.90
		Artificial Neural Network	79.3	28.9	0.424	0.89
		Support Vector Machine	82.5	33.1	0.473	0.76
		Decision Tree	55.2	44.3	0.492	0.74
RFE	29	Logistic Regression	78.2	32.8	0.462	0.89
		Random Forest	77.9	48.9	0.601	0.89
		AdaBoost	63.5	50.6	0.563	0.89
		XGBoost	67.3	55.9	**0.611**	0.90
		Artificial Neural Network	**86.8**	25.4	0.393	0.89
		Support Vector Machine	83.8	33.6	0.480	0.77
		Decision Tree	59.1	44.5	0.507	0.75
Correlation analysis with Kendall's Tau	13	Logistic Regression	80.7	27.9	0.414	0.88
		Random Forest	80.4	46.1	0.586	0.88
		AdaBoost	62.2	39.8	0.485	0.88
		XGBoost	70.0	51.0	0.590	0.89
		Artificial Neural Network	75.4	34.3	0.472	0.88
		Support Vector Machine	84.2	32.6	0.470	0.74
		Decision Tree	53.5	43.3	0.478	0.75

Further, in Table 4.3, it is clearly indicated that in terms of variable selection methods, RFE performed better than correlation analysis with Kendall's Tau. Potential causes for this will be discussed in chapter 5.

In Table 4.4, the relationship between the number of variables and mean value of F-score has been studied. The results indicate that on average there is a direct relation between increase in the number of variables and increase in F-score when RFE was used as a variable selection method. It is also relevant to highlight that the increase was marginal, which is one of the arguments to consider when trying to achieve a trade-off between the complexity of the model and a higher F-score.

Another thing to consider is that according to Figure 3.6 in section 3.4, with the

Table 4.2: Performance of methods with regards to chosen performance measurements with SMOTE

Variable Selection Method	No. of features	Method	Precision (%)	Sensitivity (%)	F-score	AUC score
RFE	7	Logistic Regression	43.7	69.2	0.536	0.89
		Random Forest	46.8	59.2	0.522	0.88
		AdaBoost	37.2	**74.4**	0.497	0.89
		XGBoost	42.5	72.3	0.535	0.88
		Artificial Neural Network	40.1	72.5	0.516	0.89
		Support Vector Machine	40.6	72.0	0.519	0.90
		Decision Tree	37.1	52.7	0.436	0.78
RFE	19	Logistic Regression	35.4	69.0	0.468	0.89
		Random Forest	61.5	55.7	0.585	0.89
		AdaBoost	40.2	73.0	0.518	0.90
		XGBoost	58.4	62.9	0.605	0.90
		Artificial Neural Network	33.4	73.2	0.459	0.90
		Support Vector Machine	40.8	68.5	0.511	0.89
		Decision Tree	42.5	49.6	0.458	0.75
RFE	29	Logistic Regression	35.8	64.8	0.461	0.88
		Random Forest	67.2	55.2	0.606	0.89
		AdaBoost	48.7	72.9	0.522	0.90
		XGBoost	60.2	61.1	0.606	0.90
		ANN	35.8	71.1	0.477	0.90
		Support Vector Machine	45.3	65.5	0.536	0.88
		Decision Tree	57.8	47.6	0.477	0.76
Correlation analysis with Kendall's Tau	13	Logistic Regression	38.4	62.9	0.477	0.86
		Random Forest	55.0	54.5	0.548	0.88
		AdaBoost	38.8	66.2	0.489	0.88
		XGBoost	53.9	55.3	0.546	0.86
		Artificial Neural Network	37.7	67.1	0.482	0.88
		Support Vector Machine	37.9	64.6	0.477	0.87
		Decision Tree	52.0	46.1	0.439	0.76

increasing number of variables the F-score will plateau. However, in order to make the conclusion about the relation between the number of variables used in the model and F-score, statistically significant more tests should be executed and analyzed.

Further, the conclusion can be made that tree-based methods showed on average better performance than Artificial Neural Networks. The average F-score has been computed for tree-based methods (AdaBoost, XGBoost, Decision Tree and Random Forest) and Artificial Neural Networks and is shown in Table 4.5.

Table 4.3: SMOTE performance

Variable Selection	Number of variables	Mean value of F-score without SMOTE	Mean value of F-score with SMOTE
RFE	7	0.511	0.509
RFE	19	0.513	0.515
RFE	29	0.517	0.526
Correlation analysis with Kendall's Tau	13	0.499	0.494
Total Average		**0.510**	**0.511**

Table 4.4: RFE's performance

Number of variables	Mean value of F-score
7	0.510
19	0.514
29	0.522

Table 4.5: Tree-based methods and Artificial Neural Networks

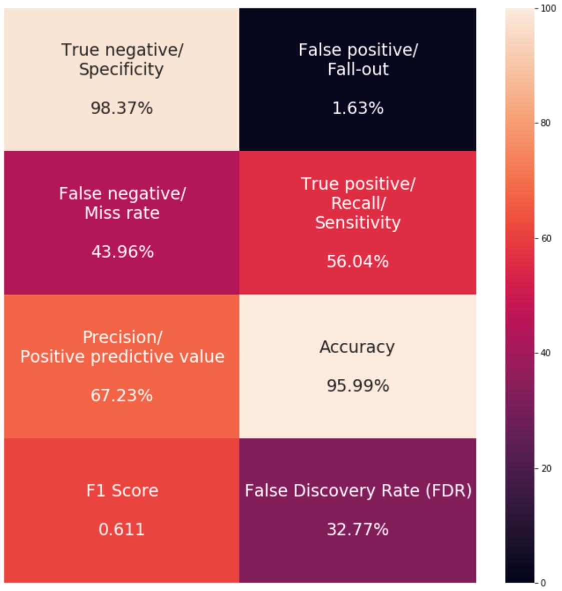
Method	Mean value of F-score
Tree-based methods	**0.542**
Artificial Neural Networks	0.460

4.2 Results of the Best Performing Method

In this section results of the best performing method, XGBoost without SMOTE and RFE$_7$ will be presented. All model evaluation parameters can be studied in Figure 4.1.

In Figure 4.2, ROC curves of XGBoost without SMOTE with different feature selection methods are analyzed. From the figure it can be concluded that there is no remarkable difference between different applied feature selection methods in terms of ROC curves and AUC scores for XGBoost without SMOTE.

In Figure 4.3, ROC curves of different classification methods, when RFE$_7$ was applied as a variable selection method, are presented. The yellow line in Figure 4.3 is a dummy classifier (DC) and represents a model that classifies every observation with a 50% probability to be a default. It can be clearly seen that even though the same variable selection method was applied, ROC curves differed for the studied classifiers.



True negative/ Specificity 98.37%	False positive/ Fall-out 1.63%
False negative/ Miss rate 43.96%	True positive/ Recall/ Sensitivity 56.04%
Precision/ Positive predictive value 67.23%	Accuracy 95.99%
F1 Score 0.611	False Discovery Rate (FDR) 32.77%

Figure 4.1: Performance metrics for XGBoost without SMOTE and RFE_7

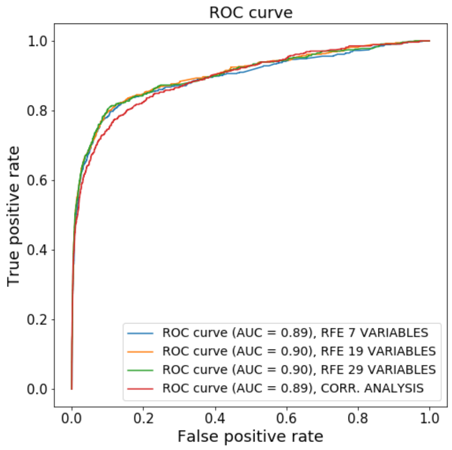


Figure 4.2: ROC curve for XGBoost without SMOTE and all of the different variable selection approaches

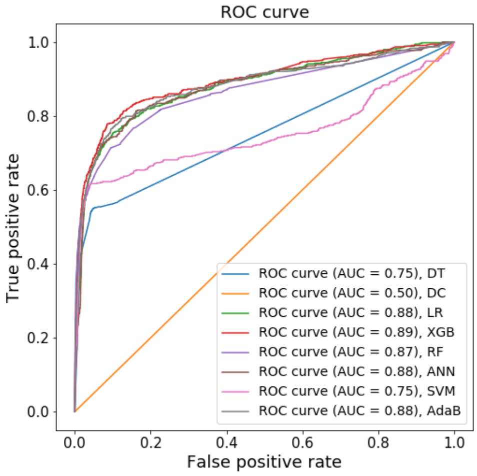


Figure 4.3: Comparison of the method's ROC curves when RFE_7 was applied as a feature selection method

Chapter 5

Discussion

In this chapter, the results from chapter 4 will be analyzed and discussed. From chapter 4, it can be seen that the overall best performance was obtained with the machine learning technique XGBoost. The best precision was shown with ANN and the best sensitivity was obtained with AdaBoost. In this project, F-score was chosen to be the main indicator of how well the model performed. This decision was made, because sensitivity and precision were of equal importance and that is why the F-score was chosen as main metric in order to capture both of the measures. Thus, if the main performance indicator is to be changed (for example sensitivity, accuracy or precision), another conclusions can be made and further analysis should be performed.

5.1 Findings

Generally, results indicated that tree-based models were more stable and showed better performance on average than Artificial Neural Networks. This is aligned with results from another study, which had the same conclusion when comparing performance between ANN and tree-based methods [32]. Other insights from chapter 4 is the impact of SMOTE on the model performance as well as how the number of variables included in the models will be discussed in section 5.1.1 and 5.1.2. However, in order to test if this relation exhibits generally, more tests should be done and different sets of classifiers should be studied.

5.1.1 Impact of SMOTE

On average, SMOTE did not have a significant positive impact on the F-score, but it had an impact on sensitivity and precision. It can be noted that implementation of SMOTE led to an increase in sensitivity and a decrease in precision. The models with SMOTE showed the following trend. The number of TP and FP increased, while FN and TN decreased. Thus, sensitivity was higher when using SMOTE.

Further, since an increase in FP was marginally higher than an increase in TP, precision decreased after SMOTE implementation.

5.1.2 Impact of Variable Selection Methods

After analyzing the results, the conclusion can be made that RFE performed better than correlation analysis with Kendall's Tau. That can be partially explained by the nature of variable selection methods. RFE is a multivariate selection method, which allows the analysis of a set of features simultaneously, while Kendall's Tau provides a pair-wise analysis of variables. In this case, it can be concluded that for this type of project, the multivariate feature selection method is more suitable than the univariate one.

Results indicated also that the number of features included in the model had an impact on the model performance. When using RFE as a variable selection method, there is a direct relation between an increase in F-score and an increase of number of variables used in the model. This is shown in Table 4.1. On the other hand, the increase is marginal. Considering the trade-off between the complexity of the model and obtaining the highest F-score, solution in this case could be to define the highest number of variables allowed in the model or a threshold of the lowest F-score.

5.2 Conclusion

The research question to be answered in this thesis was

- *For a chosen set of machine learning techniques, which technique exhibits the best peroformance in default prediction with regards to a specific model evaluation metric?*

The overall results showed that XGBoost without SMOTE executed with RFE_7 and RFE_{29} showed the best performance with regards to F-score. However, because of model complexity, the data set, which contains 7 variables is more preferable and therefore recommended in the context.

Results also showed that using RFE as a feature selection method led to the better performance than the correlation analysis with Kendall's Tau, which could partially be explained by the multivariate nature of RFE.

It can be concluded that SMOTE did not enhance the performance of the models remarkably in terms of F-score. When SMOTE was applied, sensitivity increased and precision decreased. On average, it was also concluded that the increased number of variables used in the models chosen by RFE had a direct relation with an increase in F-score. However, this increase was marginal and that should be taken into consideration as it is preferably to have a trade-off between the complexity of the model and the F-score. Further, it could be also concluded that tree-based methods showed on average better performance than ANN.

5.3 Future Work

The potential future work for this project will be a further development of the model by deepening analysis on variables used in the models as well as creating new variables in order to make better predictions. Data available for the scope of this thesis has constraints in terms of many years are covered by the data presented as well as geographical breadth of the Nordea's clients. The majority of customers at Nordea's clients are from the Nordic countries, thus, it should be considered that the behaviour of Nordic customers influence the results of this research. It means that the behaviour of clients outside Nordics may or may not follow the same pattern and therefore one should make additional analysis and obtain a geographically-broader data set if the objective is to have a model unbiased of the geographical location. An assumption can also be made that if there is data available for longer time span as well as broader geography of clients, there is an interest to implement marco-economical variables, which in turn might open some new insights about factors impacting default of a customer as well as what machine learning methods are more suitable for this type of a problem. Further, a large part of this project was to make a grounded feature selection such that variables included in the models were valuable for prediction. Variable selection was made by RFE and correlation analysis with Kendall's Tau, but it would be interesting to apply other variable selection methods. An alternative for dimensionality reduction could be Principal Component Analysis (PCA).

It would also be interesting to make a study concerning what metrics are the most relevant for this type of the problem. As mentioned previously, in this project the main metric all evaluations were analyzed by was F-score, because the aim was to achieve a trade-off between the sensitivity and the precision. If a deeper analysis could be performed regarding the most relevant metric for this type of problem, then potentially a weight function could be implemented if one of the metrics explored turned out to be of more importance. The example of a weight function can be to use a weighted F-score, where β is not set to 1, but the value of interest.

Bibliography

[1] H. Abdi. *Kendall rank correlation.* eng. Encyclopedia of Measurement and Statistics. Sage, 2007.

[2] *An overview of correlation measures between categorical and continuous variables.* `https://medium.com/@outside2SDs/an-overview-of-correlation-measures-between-categorical-and-continuous-variables-4c7f85610365`. Accessed: 2019-05-15.

[3] Basel Committee on Banking Supervision. *International Convergence of Capital Measurement and Capital Standards: a Revised Framework,Comprehensive Version.* June 2006.

[4] Rok Blagus and Lara Lusa. “Joint use of over- and under-sampling techniques and cross-validation for the development and assessment of prediction models”. In: *BMC Bioinformatics* 16 (2015).

[5] Michael Bowles. *Machine Learning in Python: Essential Techniques for Predictive Analysis.* eng. 2015. ISBN: 9781118961742.

[6] Camilla Calì and Maria Longobardi. “Some mathematical properties of the ROC curve and their applications”. In: *Ricerche di Matematica* 64 (Oct. 2015). DOI: `10.1007/s11587-015-0246-8`.

[7] Tianqi Chen and Carlos Guestrin. “Xgboost: A scalable tree boosting system”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining.* ACM. 2016, pp. 785–794.

[8] Andreas Christmann. *Support Vector Machines.* eng. Information science and statistics. 2008. ISBN: 1-281-92704-X.

[9] *Feature Analysis Visualizer. Recursive Feature Elimination.* `https://www.scikit-yb.org/en/latest/api/features/rfecv.html`. Accessed: 2019-05-24.

[10] James Finance. “Machine Learning in Credit Risk Modeling: Efficiency should not come at the expense of Explainability”. In: (July 2017).

[11] Jean D. Gibbons. *Nonparametric Measures of Association.* Thousand Oaks, California: SAGE Publications, Inc., URL: `https://methods.sagepub.com/book/nonparametric-measures-of-association`.

[12] Cyril Goutte and Eric Gaussier. “A probabilistic interpretation of precision, recall and F-score, with implication for evaluation”. In: *European Conference on Information Retrieval.* Springer. 2005, pp. 345–359.

[13] Xavier Benoit Gust Lucile D'journo. *The use of correlation functions in thoracic surgery research.* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4387406/. Accessed: 2019-05-17.

[14] Trevor Hastie, Jerome Friedman, and Robert Tibshirani. *An Introduction to Statistical Learning.* eng. Springer Series in Statistics. New York, NY: Springer New York, 2013, p. 26. ISBN: 978-1-4614-7138-7.

[15] Trevor Hastie, Jerome Friedman, and Robert Tibshirani. *An Introduction to Statistical Learning.* eng. Springer Series in Statistics, New York, NY: Springer New York, 2013, p. 132. ISBN: 978-1-4614-7138-7.

[16] Trevor Hastie, Jerome Friedman, and Robert Tibshirani. *An Introduction to Statistical Learning.* eng. Springer Series in Statistics. New York, NY: Springer New York, 2013, pp. 304–305. ISBN: 978-1-4614-7138-7.

[17] Trevor Hastie, Jerome Friedman, and Robert Tibshirani. *An Introduction to Statistical Learning.* eng. Springer Series in Statistics, New York, NY: Springer New York, 2013, pp. 319–320. ISBN: 978-1-4614-7138-7.

[18] Trevor Hastie, Jerome Friedman, and Robert Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* eng. Springer Series in Statistics. New York, NY: Springer New York, 2001, p. 98. ISBN: 9780387216065.

[19] Trevor Hastie, Jerome Friedman, and Robert Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* eng. Springer Series in Statistics. New York, NY: Springer New York, 2001, p. 270. ISBN: 9780387216065.

[20] Trevor Hastie, Jerome Friedman, and Robert Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* eng. Springer Series in Statistics. New York, NY: Springer New York, 2001, p. 271. ISBN: 9780387216065.

[21] Trevor Hastie, Jerome Friedman, and Robert Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* eng. Springer Series in Statistics. New York, NY: Springer New York, 2001, pp. 350–355. ISBN: 9780387216065.

[22] Trevor Hastie, Jerome Friedman, and Robert Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* eng. Springer Series in Statistics. New York, NY: Springer New York, 2001, pp. 214–217. ISBN: 9780387216065.

[23] Simon S Haykin et al. *Neural networks and learning machines.* Vol. 3. Pearson Upper Saddle River, 2009.

[24] Feng Hu and Hang Li. "A Novel Boundary Oversampling Algorithm Based on Neighborhood Rough Set Model: NRSBoundary-SMOTE". In: *Mathematical Problems in Engineering* (2013). URL: https://www.researchgate.net/publication/287601878_A_Novel_Boundary_Oversampling_Algorithm_Based_on_Neighborhood_Rough_Set_Model_NRSBoundary-SMOTE.

[25] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. “Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks”. In: (2004).

[26] Simon James. *An Introduction to Data Analysis using Aggregation Functions in R*. eng. 2016. ISBN: 3-319-46762-X.

[27] *K fold and other cross-validation techniques*. `https://medium.com/datadriveninvestor/k-fold-and-other-cross-validation-techniques-6c03a2563f1e`. Accessed: 2019-05-15.

[28] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[29] Erwin Kreyszig. *Advanced Engineering Mathematics*. eng. 2011. ISBN: 9780470646137.

[30] Kjell Kuhn Max Johnson. *Feature Engineering and Selection: A Practical Approach for Predictive Models*. eng. Feature Engineering and Selection: A Practical Approach for Predictive Models. 2019. ISBN: 9781138079229.

[31] A.E. Lazzaretti and D.M.J. Tax. “An adaptive radial basis function kernel for support vector data description”. In: vol. 9370. Springer Verlag, 2015, pp. 103–116. ISBN: 9783319242606.

[32] Peter Martey Addo, Dominique Guegan, and Bertrand Hassani. “Credit Risk Analysis Using Machine and Deep Learning Models”. In: *Risks* 6 (2018).

[33] Tom M Mitchell. “Machine learning and data mining”. In: *Communications of the ACM* 42.11 (1999).

[34] Vinod Nair and Geoffrey E Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[35] R.B. Nelsen. *Kendall tau metric*. eng. Encyclopedia of Mathematics. 2001. ISBN: 978-1-55608-010-4.

[36] Chawla Nitesh V. et al. “SMOTE: Synthetic Minority Over-sampling Technique”. In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357.

[37] *Nonparametric Bayesian Inference in Biostatistics*. eng. 1st ed. 2015.. Frontiers in Probability and the Statistical Sciences. 2015. ISBN: 3-319-19518-2.

[38] *Recursive Feature Elimination (RFE)*. `https://www.brainvoyager.com/bv/doc/UsersGuide/MVPA/RecursiveFeatureElimination.html`. Accessed: 2019-05-17.

[39] Raúl Rojas. “AdaBoost and the super bowl of classifiers a tutorial introduction to adaptive boosting”. In: (2009).

[40] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. eng. Upper Saddle River, New Jersey 07458: Pearson Education, Inc., 2010.

[41] Bernhard Schölkopf, Chris Burges, and Vladimir Vapnik. “Incorporating invariances in support vector learning machines”. In: *International Conference on Artificial Neural Networks*. Springer. 1996, pp. 47–52.

[42] Peter Sprent and Nigel C Smeeton. *Applied nonparametric statistical methods*. Chapman and Hall/CRC, 2000.

[43] Arthur Sullivan and Steven M. Sheffrin. “Economics: Principles in Action”. In: (2003). ISSN: 0-13-063085-3.

[44] L. C. Thomas, E. N. C. Tong, and C. Mues. “Mixture cure models in credit scoring: If and when borrowers default”. In: *European Journal of Operational Research* 218 (2012), pp. 132–139.

Appendix A

Kendall's Tau Correlation Analysis

Table A.1: Positive correlation of Feature Variables against the Response Variable with Kendall's tau

Feature name	Correlation with response
bf_ag_4_1	0.382857
bf_ag_7_1	0.381114
bf_ag_1_1	0.360268
bf_ag_10_3	0.356646
bf_ag_4_3	0.351934
bf_ag_7_3	0.346601
bf_ag_6_3	0.340912
bf_ag_1_3	0.340761
bf_ag_2_3	0.340727
bf_ag_10_6	0.332280
bf_ag_4_6	0.330372
bf_ag_6_6	0.319739
bf_ag_7_6	0.319038
bf_ag_1_6	0.318593
bf_ag_2_6	0.318559
bf_ag_8	0.315214
bf_ag_10_12	0.313506
bf_ag_4_12	0.311391
bf_ag_9_6	0.306812
bf_ag_7_12	0.300638
bf_ag_6_12	0.300206
bf_ag_9_3	0.298221
bf_ag_2_12	0.296260
bf_ag_1_12	0.296115
bf_ag_9_12	0.289003
bf_ag_5_6	0.193858
bf_ag_5_12	0.193014
bf_ag_5_3	0.192234
bf_ag_20_12	0.053963
bf_ag_20_6	0.048757
bf_ag_20_3	0.045284
bf_ag_20_1	0.037427
bf_ag_11	0.003460

Table A.2: Negative correlation of Feature Variables against the Response Variable with Kendall's tau

Feature name	Correlation with response
*bf_ag*_12	-0.030496
*bf_ag*_14_3	-0.044130
*bf_ag*_14_12	-0.044615
*bf_ag*_14_6	-0.050902
*bf_ag*_13_3	-0.070456
*bf_ag*_13_12	-0.073104
*bf_ag*_13_6	-0.080682
*bf_ag*_17	-0.092303
*bf_ag*_3_12	-0.275970
*bf_ag*_3_6	-0.298557
*bf_ag*_3_3	-0.324351

Appendix B

Adam Algorithm

Algorithm 4: Adam algorithm

```
Choose a stepsize $\gamma$
Determine decay rates for the moment estimates $\psi_1$ and $\psi_2 \in [0, 1)$
Define stochastic objective function with parameters $\theta$
Initialize $\theta_0$
Set first and second moment vectors $m_0$ and $v_o$ to 0
Set timestep $t \leftarrow 0$
Set $\varepsilon \leftarrow 10^{-8}$
while $\theta_t$ has not converged do
    $t \leftarrow t + 1$
    Gradients with regards to stochastic objective is computed $\phi_t \leftarrow \nabla_\theta R_t(\theta_{t-1})$
    First moment estimate is updated $m_t \leftarrow \psi_1 \cdot m_{t-1} + (1 - beta_1) \cdot \phi_t$
    Second moment estimate is updated $v_t \leftarrow \psi_2 \cdot v_{t-1} + (1 - beta_1) \cdot \phi_t^2$
    Estimation of bias-corrected first raw moment estimate is performed $\hat{m}_t \leftarrow \frac{m_t}{(1-\psi_1^t)}$
    Estimation of bias-corrected second raw moment estimate is performed $\hat{v}_t \leftarrow \frac{v_t}{(1-\psi_2^t)}$
    $\theta_t \leftarrow \theta_{t-1} - \gamma \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$
Output: $\theta_t$
```

TRITA -SCI-GRU 2019:073