UPTEC F 22016

Examensarbete 30 hp

Juni 2022

Prediction of Credit Risk using Machine Learning Models

Philip Isaac

Civilingenjörsprogrammet i Teknisk Fysik

UPPSALA UNIVERSITET

Prediction of Credit Risk using Machine Learning Models

Philip Isaac

Abstract

This thesis aims to investigate different machine learning (ML) models and their performance to find the best performing model to predict credit risk at a specific company. Since granting credit to corporate customers is a part of this company's core business, managing the credit risk is of high importance. The company has of today only one credit risk measurement, which is obtained through an external company, and the goal is to find a model that outperforms this measurement.

The study consists of two ML models, Logistic Regression (LR) and eXtreme Gradient Boosting. This thesis proves that both methods perform better than the external risk measurement and the LR method achieves the overall best performance. One of the most important analyses done in this thesis was handling the dataset and finding the best-suited combination of features that the ML models should use.

Teknisk-naturvetenskapliga fakulteten

Uppsala universitet, Utgivningsort Uppsala/Visby

Ämnesgranskare: Ping Wu

Examinator: Tomas Nyberg

Populärvetenskaplig sammanfattning

Att ett företag skall kunna bedriva en verksamhet där majoriteten av verksamheten bygger på att låna ut pengar medför att en viktig princip är att man ska få tillbaka pengarna man lånat ut. Utan en välfungerande riskanalys hade hela principen med utlåningen av likvida medel falllit samman.

Man kan med en viss sannolikhet förutse utfallet av en ansökan genom att använda olika metoder. En metod som tas upp i det här arbetet är att förutse kreditrisken med hjälp av två maskininläningsmodeller, Logistisk Regression (LR) och eXtreme Gradient Boosting. För att kunna kunna avgöra om en låntagare kommer kunna betala tillbaka lånet med hjälp av de ovanstående nämna maskininlärningsmodeller krävs data som innehåller en viss typ av information. Denna information måste bland annat vara historisk data om låntagaren och själva lånet, tillsammans med utfallet av ansökan. Med utfallet av ansökan i detta projekt, menas huruvida låntagaren har blivit 90 dagar sen, eller mer, med en betalning inom de första 12 månaderna efter ansökan.

Den slutgiltiga metoden som visade sig nå bäst resultat i mån om noggrannhet via det såkallade ROC-AUC måttet var en optimerad modell som använde LR tillsammans med binning. Binning är en metod som delar in datavärden i olika grupper, som behållare, där man kan tänka sig att värden hamnar i olika behållare beroende på i vilken kategori värdet hamnar. Om man tänker sig att det finns tre stycken behållare med värden mellan 1-30 så kan exempelvis alla värden mellan 1-10 hamna i behållare 1, alla värden mellan 11-20 hamna i behållare 2 och slutligen alla värden mellan 21-30 hamna i den sista behållaren.

Contents

1 Introduction

1.1 Background

The possibility that a borrower fails to make due payments of a loan resulting in a loss for the company is defined as credit risk. Credit risk has been relevant ever since lending itself was introduced, where every loan comes with a risk that the money is not going to be returned. To be able to minimize risks and losses, different kind approaches have been used in the banking industry. It is practically impossible to be 100% sure that someone will be able to repay a loan, but by managing credit risk, the loss could be minimized.

Risk management is one of the most important areas in banking and the way credit risk is being managed has changed over the years. Today, there are much more effective and more accurate ways of handling credit risk in a shorter period. Instead of going through customers' applications one by one by analyzing the individual data, more efficient methods could be used. One of these methods is credit scoring combined with machine learning (ML) models, which will be explored in this project.

The main task in this project will be to predict if a customer defaults on a corporate loan. More specifically, to default on a loan indicates that the customer will be 90 days late, or more, with payment within 12 months after application. Since Company X is currently using an external credit information company, which will be referred to as Company Y for the rest of this thesis, they are not fully independent when predicting the credit risk. However, the information from Company Y is currently Company X's only risk measure. It would be beneficial to be able to manage credit risk without any external companies, which also is one of the main reasons why this model plays an important role for the company.

The way ML could be used to predict if a customer will default on a loan and help an organization decide if the customer's application should be approved will be through the following way. The first assumption will be that some kind of data is available that has been stored during a period. This data needs to include relevant information and data from the applicant of the loan, together with the outcome, if the applicant did default or not. When the data is available, the ML model would be able to predict the outcome of future applications by using the collected data. The reason why ML is highly relevant for companies using these kinds of predictions is not only because ML is an efficient way of predicting outcomes, but also because models often could be made more accurate than current prediction methods.

The data used in this thesis is provided by a real company in Sweden, Company X. Company X's principal business is granting credit to corporate customers. Information from Company Y is now combined with a manual credit evaluation done by an administrator to determine to which corporate clients the bank would give credits. The bank is interested in augmenting today's credit evaluation with its internal credit rating model in the form of an application score. It is important to become a more data-driven business and be able to make more accurate and faster lending decisions. The application score must estimate the likelihood that a potential borrower would fail and hence not meet its commitments within its first year on the bank's books. The goal is to explore which characteristics drive risk using the bank's historical data and statistical methodologies and to construct a mathematical model whose output is an application score that predicts the likelihood of bankruptcy for a possible borrower at Company X.

1.2 Purpose and Goals

This project aims to investigate different ML techniques and find and apply such that is best for predicting credit risk, more specifically, for predicting if a potential corporate loan customer is going to default or not. The goals of the project include

1. Investigating ML techniques and finding the best suited for credit risk modeling;
2. Developing the algorithms for the selected techniques, that should be able to achieve a ROC-AUC score of higher than 65%;
3. Finding the best use of the available dataset.

1.3 Tasks and Scope

The specific tasks for the project include

1. Studying the modeling of credit risk;
2. Investigating which variables are statistically significant when predicting credit risk;
3. Investigating various ML techniques;
4. Investigating the available datasets and applying the ML models;

5. Developing a credit risk model that predicts the probability that a potential corporate loan customer at Company X will be 90 days late with payment within 12 months of application.

1.4 Outline

The rest of the report is arranged as follows. Initially, chapter 2 presents the technical background. Chapters 3 and 4 present both the theory, method, and the implementation of the ML models. These chapters also contain the handling of the given dataset, which is important to do before using the dataset in the models. At the end of chapter 4, the models will be optimized and different features will be investigated. Chapters 5 and 6 are presenting and analyze the results together with introduce future work on the project.

2 Technical Background

2.1 Credit Risk and its Modeling

Credit risk is the possibility of a debt default resulting from a borrower's failure to make due payments. Building a credit risk model that estimates the probability that a customer will be able to pay back a loan is one of the most important mathematical problems in today's society. This is done by using a model on the data collected about a customer who applies for a loan. The credit risk modeling is not only limited to loans associated with credit cards but can also be applied to any other loans. There are different models for predicting credit risk or evaluating the level of risk associated with customers. One example of such a model is credit risk scorecards (or credit scorecards), which will be explained in Section 3.1.

2.2 Dataset

2.2.1 Description of the Dataset

The data in the dataset that is used in this project is data that has been collected by Company X over the years to be able to decide whether the customer should grant credit or not. The features investigated in this project that are received from Company X are classified and will therefore remain unspecified during this thesis. The initial data given by Company X is given in an excel sheet of 2857 rows and 189 columns, where each row represents one customer application and each column contains represents one feature of this exact customer. This means that the dataset contains information about 2857 applications. A visualisation of the structure of the dataset can be seen in Table 2.1, where the entities $x_{1,1}$ to $x_{m,n}$ represents the data.

Table 2.1: Structure of the given dataset.

Application	Feature 1	Feature 2	$\cdots$	Feature n
Application 1	$x_{1,1}$	$x_{1,2}$	$x_{1,*}$	$x_{1,n}$
Application 2	$x_{2,1}$	$x_{2,2}$	$x_{2,*}$	$x_{2,n}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Application m	$x_{m,1}$	$x_{m,2}$	$x_{m,*}$	$x_{m,n}$

The initial dataset has many difficulties where one of them is that data are missing in many locations, even up to over 95% missing values for some features. Another difficulty is that the data is class imbalanced. This is a common problem that occurs in ML classification problems since the algorithms assume that data is evenly distributed. This causes the algorithm to be biased toward predicting the majority class. In this project, the majority class will be the non-defaulting faculty. Class imbalance represents a dataset where one class is heavily populated and the other class only is populated by a few points [1]. In this dataset, the output variable y only consists of 13% ones, while the remaining 87% are zeros.

To get a better understanding of how much of the data is missing, a good indication is the ratio of the total number of missing values to the total number of elements (which equals rows*columns) as follows

$$\frac{\text{missing values}}{\text{rows} * \text{columns}} = \frac{250528}{2857 * 189} = 0.464 \approx 0.47, \tag{2.1}$$

or 47% of the data is missing initially.

2.2.2 Missing Values

Most of the collected data in the financial world often contain missing values for various reasons. One reason could be that some data that were irrelevant earlier, therefore not collected, may be of higher importance today. Since logistic regression (LR) requires a dataset without missing values, this is a problem that has to be dealt with before running the model. There are mainly four ways to deal with missing values [2], which are to

1. Exclude all data with missing values;
2. Exclude characteristics that have 70% (or more) missing values;
3. Include characteristics with missing values, but treat them as a separate attribute;
4. Impute missing values using statistical techniques.

Option 1 will not be considered in this project due to two reasons. The first reason is that the dataset already is small and the second reason is that the missing values are in different characteristics for different companies. This would make it difficult to find a good training set for the model and to find the most relevant characteristics. Option 2 will be considered since some columns are missing more than 70% (or even up to 95+% in some cases) of the values. The columns with such a high percentage of missing values are removed since they are assumed to have no, or negligible, impact on the model. Options 3 and 4 are going to be used, these will be explained in detail in Section 4.2.

3 Theory

3.1 Credit Risk Scorecard

Credit scorecards are mathematical models that calculate a probability threshold that may be used to assess risk tolerance. They offer a powerful solution to measure business risk in an environment where risk managers need to identify customers with low and high risks. The scoring provides an objective and consistent technique to measure risk, as long as minimizing system overrides. Credit scorecards do not identify good or bad customers on an individual basis, where the customer who did not default will be referred to as "good" customers, and the ones who defaulted as "bad" customers for the rest of this report. Credit scorecards will provide probabilities that applications with a specific score are good or bad customers [2].

Credit scorecards will not be implemented in this project, but since ML methods like LR can be used to build risk prediction scorecards, this is still of interest. An example of how scores could be assigned in credit scorecards is shown in Table 3.1.

Table 3.1: The functionality of a scorecard.

Characteristic	Attribute	Score
Income	$< 10,000$	100
Income	$10,000 - 19,999$	80
Income	$20,000 - 49,999$	65
Income	$50,000 - 89,999$	50
Income	$\geq 90,000$	30
Age	< 20	60
Age	$20 - 39$	50
Age	$40 - 59$	60
Age	$60 - 79$	80
Age	≥ 80	100

The two characteristics, or attributes, that are brought up in this example are income and age, the table shows that higher income might lead to lower risk. These scores are assigned after statistical analyses have been done. By combining all the scores for each application's different attributes, a final score will be set. This score will then be compared to statistical data and this will help the institution to either categorize a customer as good or bad.

One way of transforming the independent variables in the dataset is by using the Weight of Evidence (WoE) method. This method determines the strength of each individual or group of attributes to distinguish between good and bad customers. In each attribute, the WoE approach calculates the difference between the proportion of good and bad customers

$$WoE = \left[ln\left(\frac{\textit{Distr. Goods}}{\textit{Distr. Bads}}\right)\right] \times 100. \tag{3.1}$$

One example could be that it calculates the odds that a client with that exact attribute will be either good or bad [2]. Another relevant measure in credit scorecards is the Information Value (IV) that can be calculated in the following way

$$IV = \sum_i ((\textit{Distr. Goods})_i - (\textit{Distr. Bads})_i) * WoE_i. \tag{3.2}$$

WoE and IV are two metric divergence measures that can be calculated after using a binning process which will be described in Section 4.2.2 [3].

3.2 Machine Learning

ML is a research branch, that is part of artificial intelligence and is dedicated to building models that use data, also called training data, to improve the performance of some tasks. ML is used in a broad range of areas, including face recognition, image recognition, email filtering, and credit risk. The use of ML has also increased rapidly in many industries to recognize patterns in data that could be difficult to find.

There are three categories that ML is broadly classified into, which are *supervised-*, *unsupervised-* and *reinforcement* learning. Supervised learning is when data *is* labeled and the aim is to make a classification or a prediction of an output value. Unsupervised learning is when then the data *is not* labeled, and the aim is to identify patterns in a given dataset. The two ML methods, that will be used and compared in this project, are the LR model and the eXtreme Gradient Boosting (XGBoost) model. They both are a part of the supervised learning category. Reinforcement learning algorithms are algorithms that learn patterns by trial and error. The reason why these exact models are being investigated is since the LR model is the most commonly used ML model in the banking industry due to its advantageous features like its robustness. XGBoost is a highly efficient ML algorithm that has become one of the most popular algorithms quickly. It has a unique default behavior for handling missing data, which will be of interest to this thesis.

To be able to use ML on a model, some kind of data needs to exist, such that the model could learn patterns from the old data and apply these patterns to predict feature outcomes [4]. This dataset is, as mentioned before, given by Company X and ML will be used in this thesis to predict the probability that a company will default on their loan.

The LR model is the most commonly used ML model in the banking industry. The reason for this is its advantageous features like robustness and transparency. Some of the main disadvantages of using support vector machines and neural networks are that they do not require information regarding the functional [5].

The model will have an output, or a target variable, that indicates whether or not a potential corporate loan customer will default or not. For simplicity, this target variable y can be set to either 0 or 1. For $y = 0$, the customer is not going to default, whereas when $y = 1$, the customer will default.

The first part will be regarding making the dataset relevant and useful for this model. One necessary condition is that the dataset has to be divided into one train data and one test data. The training data is the sample of data that is used to fit the ML model, and the test data is the sample of data that is used to provide an unbiased evaluation of a final model fit on the training data.

3.2.1 Logistic Regression

In ML, LR is a multivariate statistical model that is appropriate to use when the criterion measure is dichotomous. However, the model can be extended such that three or more categories could be used to measure the criteria [6]. The *criterion validity* is a measurement of how well the method is predicting the outcome of the next measurement.

One of the most commonly used methods to estimate logistic models is the maximum likelihood estimation. LR can be used with either a categorical target variable or a continuous target variable. The categorical has to be either a binary or a dichotomous variable, whereas the categorical target variable represents values in the range of 0.0 to 1.0 [1].

LR is provided by implementing a linear equation that includes explanatory or independent variables. These variables are used to predict a response value, that for example could be the result of a study [7]. The linear equation can be written as

$$z = \alpha_0 + \alpha_1 x_1, \tag{3.3}$$

where x_1 is the explanatory variable, z is the predicted response variable and the coefficients α_1 and α_2 are the parameters of the model. If there exist multiple explanatory variables, the expression of the linear equation in Equation (3.3), can then be written as

$$z = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n, \tag{3.4}$$

where $\alpha_{1,2,\ldots,n}$ are the parameters of the model.

To convert the predicted response value, z, to a probability that is between $0-1$, the sigmoid function is used. This function is often used in ML to map predictions to probabilities, and it is defined as

$$\theta(z) = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}}. \tag{3.5}$$

The plot of the Sigmoid function is shown in Figure 3.1.

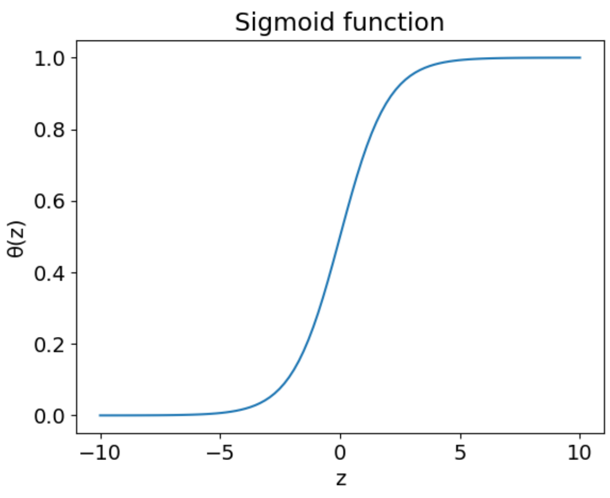


Figure 3.1: Visualisation of the sigmoid function.

Since the given dataset is small, both *lbfgs* and *liblinear* are two solvers that could perform well for the LR model. The *lbfgs* solver was chosen for this thesis since it was performing a bit better than *liblinear*. Two other solvers that could be beneficial to use if the dataset would have been larger would be *sag* and *saga* since they are faster for larger datasets in general.

3.2.2 Gradient Boosting

The methodology of a gradient boosting algorithm is to sequentially build the final models. By combining a series of for example regression trees, which also are called weak learners, the boosting algorithm can build a stronger final model. Gradient boosting is a commonly used and powerful ML technique that aims aiming to find an approximation $\hat{F}(\boldsymbol{x})$, of a function $F^*(\boldsymbol{x})$. $F^*(\boldsymbol{x})$ is mapping $\boldsymbol{x}$ to the output values y and the way gradient boosting can find this approximation is by taking the expected value of a specified loss function $L(y, F(\boldsymbol{x}))$ and minimizing it in the following way

$$F^*(\boldsymbol{x}) = \underset{F(\boldsymbol{x})}{\arg\min}\, E_{y,\boldsymbol{x}} L(y, F(\boldsymbol{x})). \tag{3.6}$$

An additive approximation of $F^*(\boldsymbol{x})$ is built by gradient boosting, which will be a weighted sum of functions

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \rho_m h_m(\boldsymbol{x}), \tag{3.7}$$

where $h_m(\boldsymbol{x})$ will be the m^{th} function and ρ_m will be the weight of this exact function [8, 9, 10, 11]. A recent high performing ensemble model is the XGBoost model, which will be discussed in more detail in the upcoming section, Section 3.2.3 [8].

3.2.3 eXtreme Gradient Boosting

XGBoost is a scalable machine learning method that uses an open-source software package to offer regularized gradient boosting for tree boosting. It is scalable in all scenarios and runs more than a factor of ten faster than other popular solutions, one of the reasons why it is faster than other methods is since handles sparse data by using a novel tree learning algorithm [12].

It is a highly effective ML algorithm that has become one of the most popular algorithms quickly. XGBoost is a scalable end-to-end tree boosting system, which has a unique default behavior for missing data. It does not only know how to handle missing data, it is expecting it. The model identifies the missing values and makes sure that they are set to zero. The model will be able to handle these zeros, even if the dataset already contains zeros.

3.2.4 Data Normalisation

When columns with a large variety of data are used, something to might be worth taking into consideration is to normalize the data before using it. This ensures that every column in the dataset is similar. Equation (3.8) is used to normalize the data by calculating the x_{norm},

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}. \tag{3.8}$$

This normalisation is implemented in python by using the *normalize()* function from the *sklearn* library.

3.3 Model Validation - ROC-AUC Curve

The performance of the used methods is measured using Receiver Operating Characteristic (ROC) Area Under the Curve (AUC) analysis. The ROC-AUC measure is frequently used to evaluate the accuracy of predictors, in fields like engineering and ML. The ROC curve's derivatives must be greater than or equal to zero, and the curve is devoid of parametric assumptions. By using the notations CP (Correctly Positive), IP (Incorrectly Positive), CN (Correctly Negative), and IN (Incorrectly Negative), two fractions could be defined. The first fraction is the *Correct Negative Fraction* (CNF) and the second fraction is the *Correct Positive Fraction* (CPF). A perfect ROC curve could now be defined, that goes through the point when $CPF = CNF = 1$, which is equivalent to the AUC score of 1.0. This also refers to a 100% accurate prediction. The ROC curve is not helpful when the AUC score is under 0.5, since the value of 0.5 is equivalent to a $50-50$ guess [13, 14].

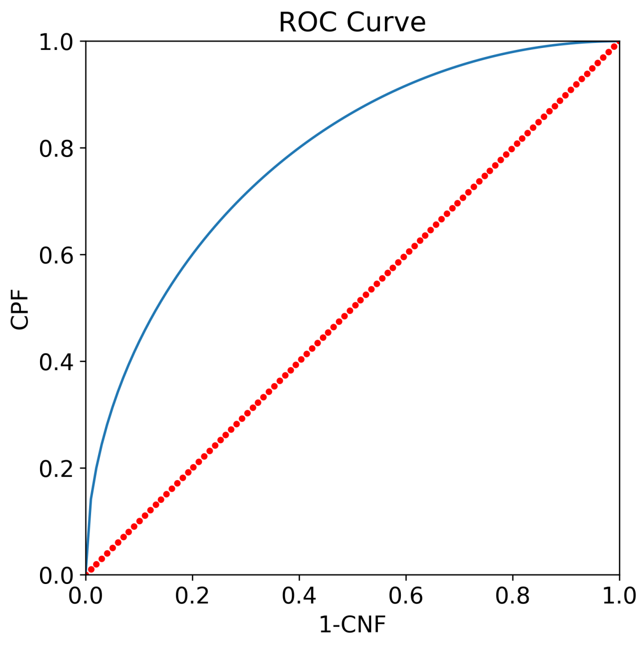


Figure 3.2: The ROC curve in blue and the diagonal line in red.

The red diagonal dotted line in Figure 3.2 illustrates where the curve is equal to an AUC score of 0.5. The degree of separability is represented by the AUC and it indicates how well the model can discriminate between classes [15]. Since AUC is the area under the curve, it can be defined as

$$\text{AUC} = \int_0^1 f(x)dx, \tag{3.9}$$

where $f(x)$ is the function of the ROC curve.

3.4 Cross-Validation

Cross-validation is a data resampling approach that tests and trains a model on multiple iterations using different sections of the data, which often is used to prevent overfitting [16]. Regular *k fold cross-validation* divides a dataset into k folds, where k is a set value. *Stratified k fold cross-validation* on the other hand is an improved version of the k fold cross-validation. It ensures that there exists the same proportion of observations with a given label in each fold of the dataset and it also ensures that there are not two overlapping test sets.

The stratified k fold cross-validator has an in-parameter called random_state. This option determines the randomization of each fold for each class by affecting the ordering of the indices. When passing an integer as the parameter, the result of the model can vary.

3.5 Feature Engineering - Combinations

Combinations without repetition, where the order is irrelevant, could be described with the formula

$$C_{n,k} = \binom{n}{k} = \frac{n!}{k!(n-k)!}, \tag{3.10}$$

where n is the total amount of elements and k is the number of elements taken each time. For example if the following set A *={a, b, c}* is considered, then n = 3, and if k is chosen to be 2, then the combinations without repetition would be *{ab, ac, bc}*. Since the columns should be tested in different cases, this means that k has to change. This increases the number of combinations for each column that is added. The formula for the total amount of iterations would be described as

$$i = \sum_{m=1}^{n} \frac{n!}{m!(n-m)!}, \tag{3.11}$$

where n in this thesis will be the number of columns used in the dataset.

3.6 Correlation

The correlation (r) between different variables is a statistical method that is widely used to be able to determine if the variables are related. The correlation between two variables does not only show if they are related but how strong this relation is [17]. To determine this correlation, Pearson Correlation Coefficient is used, which is a measurement of both strength and direction of a linear correlation between two variables x and y. The correlation can be defined as

$$r = \frac{\sum(x-\overline{x})(y-\overline{y})}{\sqrt{\sum(x-\overline{x})^2 \sum(y-\overline{y})^2}}, \tag{3.12}$$

where $\overline{x}$ and $\overline{y}$ are the mean of x, respectively y [18].

In Table 3.2, two examples α and β, are showing what the correlation r could look like between the variables x_1 and y_1, and the variables x_2 and y_2.

Table 3.2: Visualising two examples (α and β) of the correlation between x and y.

	α		β	
i	x_1	y_1	x_2	y_2
1	0	0	0	0
2	1	2	1	2
3	2	4	2	10
4	3	6	3	6
r	1.0		0.757	

4 Method and Implementation

4.1 Workflow

The workflow chart in Figure 4.1 shows how the process of the thesis was structured. It shows the process, from when the data was collected until the results from the models were achieved and optimized.

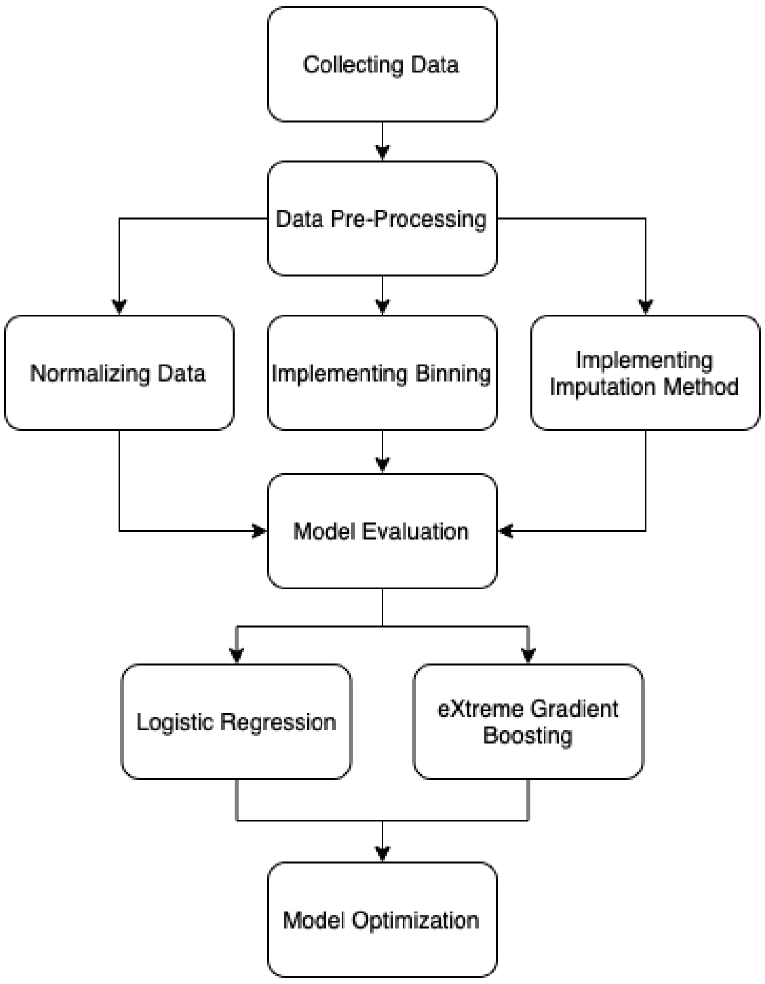


Figure 4.1: Workflow for the project.

4.2 Data Pre-Processing

4.2.1 Dataset

To be able to use the data some adjustments have to be made. The first adjustment that has to be taken into consideration is that the data is consisting of both booked and non-booked applications. The non-booked applications are not providing any relevant information since there are no y values for these applications. Hence all 1752 unbooked applications are removed from the dataset, and there remain 1105 applications. Features that are assumed to have no, or low, overall impact on the performance of the model. One example of features like this could be the application number. The second thing that has to be taken into consideration before using the ML models is the handling of missing data. The different approaches of handling the missing data are being brought up in Section 2.2.2.

4.2.2 Binning

Binning is a process that categorizes continuous variables into small sets of bins or groups. This method is commonly used in finance, particularly in credit scoring, and when ML is used [3]. There are two functions in the *pandas* library in python, *cut()* and *qcut()*, that sorts and divide the data values into bins. The *cut()* function is binning the values into discrete intervals. Since the given data is not equally distributed among the values of the column, without taking the number of elements in each bucket in mind. This could result in an unoptimized partition of the data being unevenly spread throughout the interval of the data. One of the columns in the dataset is for example shown in the histogram in Figure 4.2, where a clear majority of the data is between the values 0 and 3. This would result in an uneven amount of data in each bucket, using *cut()*, which can be seen in Table 4.1. Here the first column shows the number of elements in each bucket by using *cut()*, and the second column shows the number of elements in each bucket for the *qcut()* function. As described above, one can see that *cut()* does not care about the spread of the data. The first bucket contains 760 elements, whereas the last bucket only contains 18 elements. On the other hand, the *qcut()* function is discretizing the variable into equal-sized buckets based on quantiles. The second column in Table 4.1 shows that there are exactly 221 elements in each bucket. This is why *qcut()* will be used, rather than *cut()*, to get a more accurate partition for the prediction.

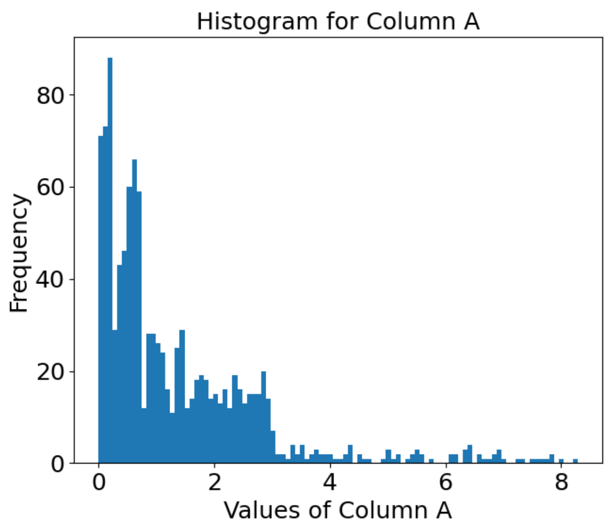


Figure 4.2: Histogram showing the distribution of the values in Column A.

Table 4.1: The different distributions of elements in buckets by using *cut()* and *qcut()* on Column A.

cut()			*qcut()*		
Bucket	Label	#Elements	Bucket	Label	#Elements
$(-0.00829, 1.658]$	1	760	$(0.999, 221.8]$	1	221
$(1.658, 3.316]$	2	264	$(221.8, 442.6]$	2	221
$(3.316, 4.974]$	3	35	$(442.6, 663.4]$	3	221
$(4.974, 6.632]$	4	28	$(663.4, 884.2]$	4	221
$(6.632, 8.29]$	5	18	$(884.2, 1105.0]$	5	221

The binning is not only used with the *qcut()* method, particularly due to one reason. Since the binning is attending a label to each bucket, the bucket with the lowest interval will obtain the lowest label and the bucket with the highest interval will obtain the highest label. In the case of missing values, they will be replaced with a 0, this means that all missing values will get in the category of the lowest bucket label, even though the value might have been in the highest label if it existed. Hence, a new column is created only containing zeros and ones, where the column is created for each feature. The entity in this column will be 0 if the same entity in the original column has a value, but a 1 if the value is missing. This will be displayed in Table 4.2.

Table 4.2: Shows the bucket labels with and without missing values.

A		B		C		
Value	$Label_A$	Value	$Label_B$	Value	$Label_{C_1}$	$Label_{C_2}$
6.0	2	6.0	2	6.0	2	0
2.3	1	2.3	1	2.3	1	0
10.2	3	-	0	-	0	1
25.1	4	25.1	4	25.1	4	0
10.3	3	10.3	3	10.3	3	0

In the three examples A, B, and C, two different value columns are used to show how the dataset binning is handled. Example A represents a complete dataset column with no missing values, and therefore the labels go from $1-4$. In examples B and C, there exists one missing value which is denoted by '-'. According to example A, the missing value in examples B and c should be equal to 10.2, and the label for this missing value should have been 3. But since this method gives the lowest label 0, Then $Label_B$ and $Label_{C_1}$ would give an inaccurate representation of the missing value. Hence, the method in B is used, but with another feature included. If there is a missing value, $Label_{C_1}$ will still obtain the label 0. But as mentioned earlier, another column $Label_{C_2}$ is now created and has the value 1, in the position where the missing value is located. In order to be able to use both columns $Label_{C_1}$ and $Label_{C_2}$, they are considered as one column.

4.2.3 Imputation Methods

Imputation methods are methods that are filling out missing values with numeric values such that there are no missing values or NaN values in the dataset. This is helpful, especially in ML, since it prevents dropping columns with missing values for some ML models that cannot handle missing values by themselves. There are different ways of imputing the missing values, some ways of imputing missing values are to replace the missing values with the mean of the column, the median of the column, the most frequent value of the column, or just a set constant. The method that is used in this thesis is to impute the missing values with the mean of the columns and it is imputed by using the *SimpleImputer* estimator in the *sklearn* library in python.

4.3 Cross-Validation

To reduce the variance of the performance, the model is run multiple times in a for-loop from $i = 1$ to $i = 100$ where the accuracy of each iteration is saved into a list. The random state for each iteration will be i, hence the method would calculate the accuracy for all random states from $1 - 100$. After the loop, a mean value of the accuracies in the list will be calculated.

4.4 Feature Engineering

To find the combination of columns or parameters, that results in the highest ROC-AUC score, a few things have to be taken into consideration. The goal is to try different combinations of parameters and compare the results to find the final combinations that result in the highest scores. To try as many sets of combinations as possible, combinations without repetition were used.

The number of iterations is a problem since every added iteration is an increase in computational time. If two datasets are considered, one with 20 columns and one dataset with 25 columns. Then, by setting $n = 20$, respectively $n = 25$ in Equation (3.11) gives $i = 1.048.575$ and $i = 33.554.431$.

Since the testing dataset in Section 2.2.1 includes 42 columns, this would result in $i = 4.4 * 10^{12}$. The runtime could be predicted by taking the runtime for 10 columns into consideration this would approximately give a 42-column dataset a runtime of over a year, which is not doable.

Algorithm 1 Finding the Highest ROC-AUC

```
1. Divide the dataset of 42 features (columns) into three separate datasets containing
   15, 15, and 12 features.
2. Find the two sets of features, for all of the three new datasets, that gives the highest
   ROC-AUC (resulting in $3 \times 2 = 6$ sets of features).
3. Find all unique features in the 6 sets of features from the previous step (resulting
   in 18 unique features).
4. Redo step 2 with the 18 best features.
5. Find the mean accuracy of each set of features when calculating the accuracy for
   each random state between $1 - 100$.
6. (Optional) Optimize the binning for the set of features giving the highest accuracy,
   by optimizing how many splits the buckets for each variable should have.
```

4.5 Correlation

To be able to find the correlation between the different parameters in the dataset, a few steps had to be taken before being able to proceed. First, since the dataset includes missing variables, the imputation method described in Section 4.2.3 was used, where the missing values were imputed using the mean value of the column. Subsequently, the NumPy function *corrcoef()* was used to find the correlation between the different columns.

Since the correlation of all columns wanted to be analyzed, a loop was written such that for every combination of two different columns, all correlations over 60% was saved.

5 Results and Discussion

To achieve the first set of results noted in column *ROC-AUC* in Table 5.1, Algorithm 1 (excluding the optional step) was used. The table shows the best combinations of features and the corresponding ROC-AUC values for the random state 10. Combination $1-6$ presents the two best combinations of ROC-AUC for each of the three subsets of the dataset.

Table 5.1: ROC-AUC for each combination of features.

Combination	Features	ROC-AUC
1	A2, A3, A4, A10 & A16	61.72%
2	A2, A3, A10 & A16	61.59%
3	A18, A19, A20, A22, A24, A27 & A31	60.73%
4	A20, A21, A22, A24, A27 & A31	60.19%
5	A32, A38, A40 & A41	55.27%
6	A37 & A40	55.27%
7	A2, A3, A16, A20, A22, A24, A27, A32, A38, A40 & A41	65.57%
8	A2, A3, A16, A20, A22, A24, A27, A32, A37, A40 & A41	65.45%
9	A3	57.08%

Combinations 7 and 8 demonstrate the best ROC-AUC for the new subset by taking the new subset of characteristics and combining all unique features from combinations $1-6$. Combination 9 is presenting the ROC-AUC score for Company X's current risk available measurement given by Company Y, to get a comparison between how much better the ROC-AUC score could get by the new model.

The algorithm could achieve better results by using parallel computing which would allow the computer to execute calculations simultaneously. This would allow the program to run the algorithm on a larger subset of the dataset since the computation time would decrease. This means that more feature combinations would be possible and also a higher ROC-AUC score.

result in more possible combinations and therefore possibly achieve higher results.

By using binning and picking the ROC-AUC of the best combination above, combination 7, Figure 5.1 is plotted. This plot shows the ROC-AUC values for each random state between 0 – 100.

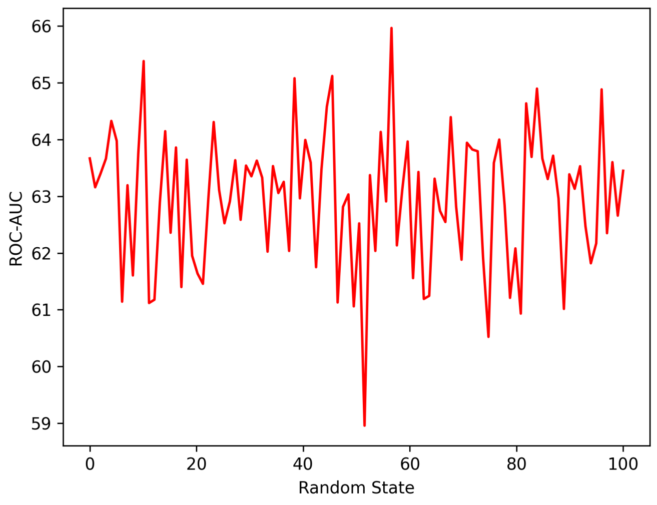


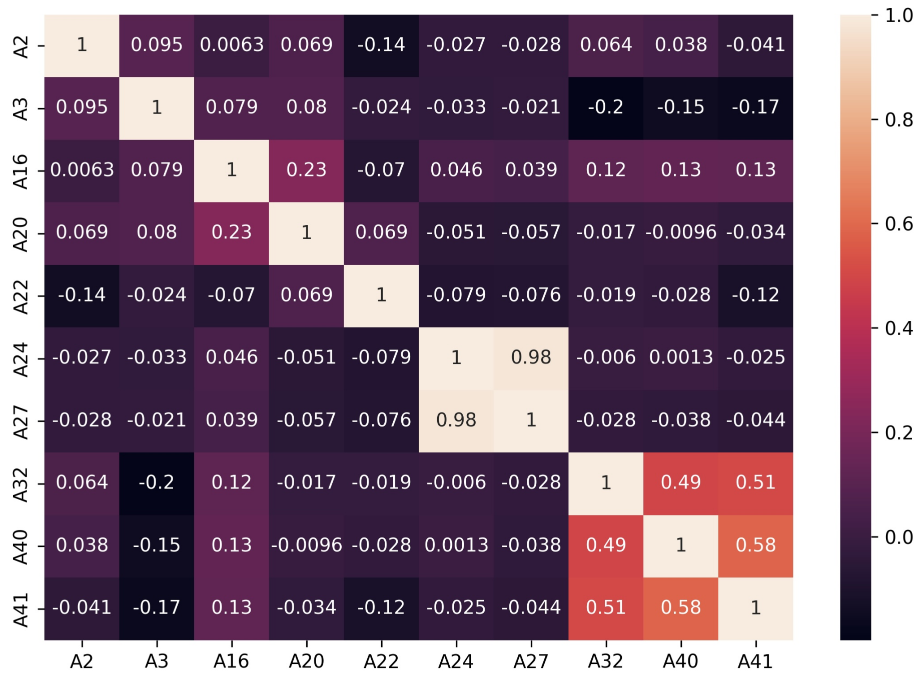
Figure 5.1: Random state check (Note that the y-axis has a range of values between 59 – 66).

By calculating the mean of the ROC-AUC for all random states between 0 – 100 for every combination in Table 5.1, the mean values are noted in the *Mean ROC-AUC* column in Table 5.2. Even though the variation seems to be high, the lowest value is around 59%, which still is higher than the lowest Mean ROC-AUC value.

Table 5.2: ROC-AUC and the mean of ROC-AUC for different random states.

Combination	ROC-AUC	Mean ROC-AUC
1	61.72%	59.36%
2	61.59%	59.43%
3	60.73%	57.55%
4	60.19%	57.63%
5	55.27%	53.15%
6	55.27%	51.89%
7	65.57%	62.95%
8	65.45%	62.49%
9	57.08%	56.34%

The features in the combination that results in the highest mean ROC-AUC (combination 7) are analyzed by plotting the heat map in Figure 5.2, showing the correlation between the features used to achieve the result.



	A2	A3	A16	A20	A22	A24	A27	A32	A40	A41
A2	1	0.095	0.0063	0.069	-0.14	-0.027	-0.028	0.064	0.038	-0.041
A3	0.095	1	0.079	0.08	-0.024	-0.033	-0.021	-0.2	-0.15	-0.17
A16	0.0063	0.079	1	0.23	-0.07	0.046	0.039	0.12	0.13	0.13
A20	0.069	0.08	0.23	1	0.069	-0.051	-0.057	-0.017	-0.0096	-0.034
A22	-0.14	-0.024	-0.07	0.069	1	-0.079	-0.076	-0.019	-0.028	-0.12
A24	-0.027	-0.033	0.046	-0.051	-0.079	1	0.98	-0.006	0.0013	-0.025
A27	-0.028	-0.021	0.039	-0.057	-0.076	0.98	1	-0.028	-0.038	-0.044
A32	0.064	-0.2	0.12	-0.017	-0.019	-0.006	-0.028	1	0.49	0.51
A40	0.038	-0.15	0.13	-0.0096	-0.028	0.0013	-0.038	0.49	1	0.58
A41	-0.041	-0.17	0.13	-0.034	-0.12	-0.025	-0.044	0.51	0.58	1

Figure 5.2: Heat map for the correlation

By looking at Figure 5.2, two parameters with a high correlation of 0.98 could be noticed, which are the parameters $A24$ and $A27$. The parameter $A24$ is translated to Ax_1, which includes the characteristics x_1 and x_4. The parameter $A27$ is translated to Ax_2, which includes the characteristics x_1, x_2, x_3, and x_4. Hence, the difference between $A27$ and $A24$ will be Ax_3 in Equation (5.3)

$$Ax_1 = \frac{x_1}{x_4}, \tag{5.1}$$

$$Ax_2 = \frac{x_1 + x_2 + x_3}{x_4}, \tag{5.2}$$

$$Ax_3 = Ax_2 - Ax_1 = \frac{x_2 + x_3}{x_4}. \tag{5.3}$$

By replacing the A27 parameter (or Ax_2) with Ax_3 in the parameter combinations 3, 4, 7 and 8, the following results are achieved.

Table 5.3: Old and new ROC-AUC values for the new set of parameters.

Combination	Old ROC-AUC	New ROC-AUC
3	57.55%	54.27%
4	57.63%	54.83%
7	62.95%	60.47%
8	62.49%	60.22%

The results are not as expected since a high correlation between variables could cause problems when the model is fitted and the results are interpreted. This is why similar, or higher, results are expected for the new ROC-AUC values. To validate that the analysis in Equation (5.1)-(5.3), the Ax_2 is replaced with Ax_3 and the new heat map is plotted in Figure 5.3

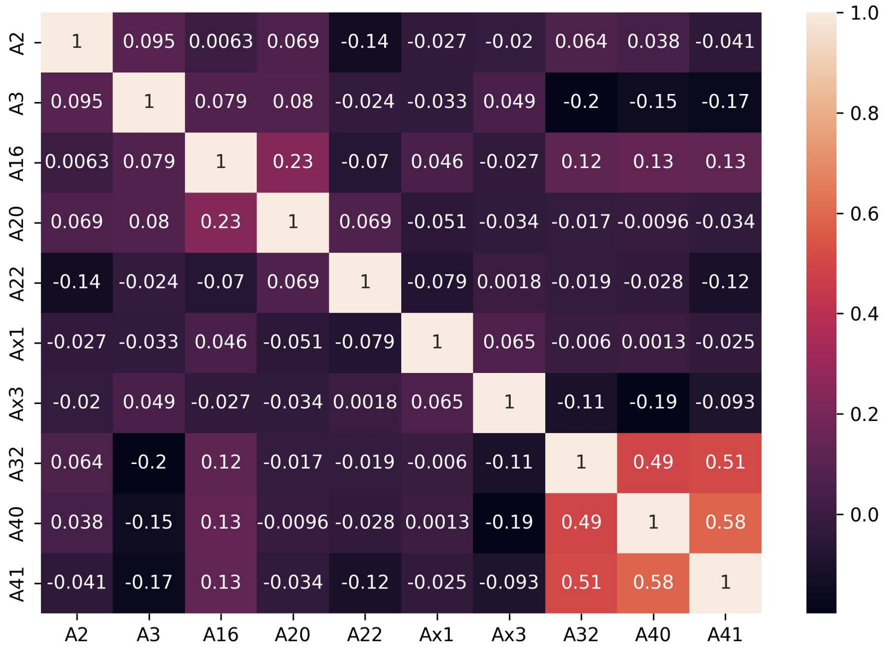


Figure 5.3: Heat map for the correlation with replaced A24 and A27

As the heat map shows, the analysis was done correctly, and the correlation of 0.98 is now gone.

To check how the model performs without the current risk measure *A3*, *A3* is removed from every combination that includes *A3* in Table 5.2. The results of the ROC-AUC mean values including *A3* are compared to the results of ROC-AUC mean values without *A3* in Table 5.4

Table 5.4: ROC-AUC and the mean of ROC-AUC for different random states.

Combination	Mean ROC-AUC	Mean ROC-AUC (excl. A3)
1	59.36%	56.92%
2	59.43%	56.35 %
3	57.55%	57.55 %
4	57.63%	57.63 %
5	53.15%	53.15%
6	51.89%	51.89 %
7	62.95%	60.72 %
8	62.49%	60.25 %

As shown in Table 5.4, the ROC-AUC results are lower when A3 is excluded. But there are still four combinations that perform better than the ROC-AUC score for combination 9 which only has the A3 parameter.

The comparison between the results of the three main methods used in this project are given in Table 5.5.

Table 5.5: Mean ROC-AUC Values for the different combinations for different models.

Combination	Binning (LR)	Imputation Method (LR)	XGBoost
1	59.36%	59.68%	57.02 %
2	59.43%	59.81%	55.75 %
3	57.55%	52.32%	52.66 %
4	57.63%	52.19%	53.37 %
5	53.15%	50.38%	54.38 %
6	51.89%	52.66%	50.40 %
7	62.95%	57.78%	57.94 %
8	62.49%	57.94%	57.21 %
9	56.34%	59.52%	53.27 %

The results in Table 5.5 show that the LR model using the binning algorithm is performing better than both the LR model using the imputation method and the XGBoost model. The best combination is given by the LR model using binning for feature combination 7. By comparing the results from Table 5.5, the imputation method is outperforming the binning method in feature combination 9. The reason behind this is since combination 9 is Company X's current risk measure, it has no missing values. Hence, it is more reasonable that the imputation method will work better since the LR model can use the data without losing information. The binning method will divide the column into intervals, and will therefore contain less information than a column full of relevant information.

Table 5.1 shows that ROC-AUC values up to 61.72% are being achieved by combinations $1-6$. By trying to combine the unique features in combination $1-6$, and measuring the two highest ROC-AUC values for the new set of features, the results are indeed higher than expected. The results went from the highest 61.72% to 65.57% in combination 7 and 65.45% in combination 8. The most optimal way to find the best combinations would be to run the Algorithm 1, but without splitting the number of features. This would find all of the best combinations instantly, most likely producing even higher ROC-AUC values. However, using this approach is not an option because the computation time climbs to unreasonable times.

By noticing the two highly correlated parameters in Figure 5.2, a deeper look into the parameters was made. The parameter A27 represents Equation (5.2), whereas A24 represents Equation (5.1). Hence, the conclusion could be made that A27 is excluding two properties compared to A24. To check if this has an impact on the result, a new parameter is created which can be seen in Equation (5.1), which is $(A27-A24)$. The results show that the two highly correlated parameters do not have any bad impact on the performance of the model.

Figure 5.1 shows how much the ROC-AUC could vary for different random states, hence the results could be misleading. If the results in this example are considered, the random state 52 gives a ROC-AUC of approximately 59, whereas the random state 57 gives a ROC-AUC of approximately 66. The highest mean ROC-AUC value, 62.95, is obtained.

By implementing Algorithm 1, including the optional step, the final ROC-AUC is given by the LR model for the features of combination 7, which is a ROC-AUC value of 66.04%. By taking the mean of this value for all scores between random states between $1-100$ gives the final result of a ROC-AUC score of 64.43%.

6 Conclusion and Future Work

The goals of this project are to investigate the best-suited ML techniques for credit risk modeling, to develop an algorithm that achieves a ROC-AUC score higher than 65%, and to find the best use of the currently available dataset for this model.

The two ML techniques that have been investigated are the LR model and the XGBoost model. The best-suited model for predicting credit risk is shown to be the LR model. The best technique combined with the LR model shows to be the binning technique since it produces the highest ROC-AUC score. In general, is the binning technique better than the imputation method for the LR model. The comparison made in Section 5, shows that the LR model using the imputation method would be achieving better results than binning for a column containing relevant information without any missing values.

The dataset is shown to be achieving the best results without combining all the features in the model. The best use of the dataset shows that feature combinations that are resulting in the highest ROC-AUC score is including between 2 to 11 (out of 42) features.

The ROC-AUC score achieved was over the target, with a ROC-AUC score of 66.04%. By using the method of taking the mean score of all random states the final score got to 64.43%, which also is close to the target ROC-AUC score. The score was higher than expected due to the imperfect and small dataset. It is performing better than the current risk measurement which is currently being purchased from Company Y. Even though the model is performing better than the current risk measurement, it would maybe be beneficial to combine the current risk measurement with the other features to achieve even more accurate predictions.

The features of the highest importance in the dataset were disclosed by the best performing ROC-AUC scores. This was one of the most important goals since Company X will from now on be able to be more strict regarding collecting these features when receiving applications. The most important features could for example be mandatory in future credit applications. This together with the earlier mentioned goals, indicates that the goal of this thesis was fulfilled but there are still some improvements or tests that could be made.

One continuation of this thesis could be to investigate further ML techniques and compare the results based on how well they manage to predict the credit risk. It would also be possible to try different methods that the ML models use, for example by imputing the missing values with the most frequent value of each column instead of the mean value.

Another continuation of this work could be to investigate how the ML models perform with different measurement techniques. The only measure that is evaluated in this thesis is the ROC-AUC score. One example could be the *Brier Score*, which measures the accuracy of probabilistic predictions.

It would also be interesting to see how various ML models perform with a larger dataset, as this is most likely why the ROC-AUC values aren't high enough to pass the 65% threshold.

Exploring how parallel computing might alter the results by allowing more features to be performed simultaneously would also be interesting. The subsets would be larger than the current 15 characteristics per subset, allowing for more combinations. A higher ROC-AUC score might be feasible if more feature combinations were available.

References

[1] Thomas Oommen, Laurie G Baise, and Richard M Vogel. “Sampling bias and class imbalance in maximum-likelihood logistic regression”. In: *Mathematical Geosciences* 43.1 (2011), pp. 99–120.

[2] Naeem Siddiqi. *Credit risk scorecards: developing and implementing intelligent credit scoring*. Vol. 3. John Wiley & Sons, 2012.

[3] Guoping Zeng. “A necessary condition for a good binning algorithm in credit scoring”. In: *Applied Mathematical Sciences* 8.65 (2014), pp. 3229–3242.

[4] A. Jeremy Mahone. *Credit Risk Modeling with Machine Learning*. URL: %5C%5Chttps://towardsdatascience.com/credit-risk-%20modeling-with-machine-learning-8c8a2657b4c4. (accessed: 11.1.2022).

[5] Gang Dong, Kin Keung Lai, and Jerome Yen. “Credit scorecard based on logistic regression with random coefficients”. In: *Procedia Computer Science* 1.1 (2010), pp. 2463–2468.

[6] Raymond E Wright. “Logistic regression.” In: (1995).

[7] Joseph M Hilbe. *Logistic regression models*. Chapman and hall/CRC, 2009.

[8] Candice Bentéjac, Anna Csörgő, and Gonzalo Martınez-Muñoz. “A comparative analysis of gradient boosting algorithms”. In: *Artificial Intelligence Review* 54.3 (2021), pp. 1937–1967.

[9] Jerome H Friedman. “Stochastic gradient boosting”. In: *Computational statistics & data analysis* 38.4 (2002), pp. 367–378.

[10] Alexey Natekin and Alois Knoll. “Gradient boosting machines, a tutorial”. In: *Frontiers in neurorobotics* 7 (2013), p. 21.

[11] Yufei Xia et al. “A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring”. In: *Expert Systems with Applications* 78 (2017), pp. 225–241.

[12] 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*. 2016, pp. 785–794.

[13] Alex J Bowers and Xiaoliang Zhou. “Receiver operating characteristic (ROC) area under the curve (AUC): A diagnostic measure for evaluating the accuracy of predictors of education outcomes”. In: *Journal of Education for Students Placed at Risk (JESPAR)* 24.1 (2019), pp. 20–46.

[14] Jin Huang and Charles X Ling. “Using AUC and accuracy in evaluating learning algorithms”. In: *IEEE Transactions on knowledge and Data Engineering* 17.3 (2005), pp. 299–310.

[15] Christopher D Brown and Herbert T Davis. “Receiver operating characteristics curves and related decision measures: A tutorial”. In: *Chemometrics and Intelligent Laboratory Systems* 80.1 (2006), pp. 24–38.

[16] Daniel Berrar. *Cross-Validation.* 2019.

[17] Richard Taylor. “Interpretation of the correlation coefficient: a basic review”. In: *Journal of diagnostic medical sonography* 6.1 (1990), pp. 35–39.

[18] Per Ahlgren, Bo Jarneving, and Ronald Rousseau. “Requirements for a cocitation similarity measure, with special reference to Pearson’s correlation coefficient”. In: *Journal of the American Society for Information Science and Technology* 54.6 (2003), pp. 550–560.