VILNIUS UNIVERSITY

FACULTY OF MATHEMATICS AND INFORMATICS

MASTER’S STUDY PROGRAMME

MODELLING AND DATA ANALYSIS

ANALYSIS OF STATISTICAL AND MACHINE LEARNING TECHNIQUES FOR PREDICTING INDIVIDUAL CREDIT RATING BASED ON LITHUANIAN DATA

Master’s thesis

Author: Airidas Neifaltas

VU email address: airidas.neifaltas@mif.stud.vu.lt

Supervisor: dr. Viktor Medvedev

Vilnius

2023

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	*Definition*
MLR	*Multinomial Logistic Regression*
DT	*Decision Tree*
RF	*Random Forest*
XGBoost	*Extreme Gradient Boosting*
NN	*Neural Network*
MLP	*Multilayer Perceptron*
DBN	*Deep Belief Network*
SVM	*Support Vector Machine*
ML	*Machine Learning*
ANN	*Artificial Neural Network*
AUC	*Area under Receiver Operating Characteristic*
ROC	*Receiver Operating Characteristic*
LR	*Logistic Regression*
LDA	*Linear Discriminant Analysis*
NB	*Naïve Bayes*
DNN	*Deep Neural Network*
TDNN	*Time Delay Neural Network*
RBFN	*Radial Basis Function Neural Network*
CNN	*Convolutional Neural Network*
K-S Statistic	*Kolmogorov-Smirnov Statistic*
FN	*False Negative*
FP	*False Positive*
TP	*True Positive*
TN	*True Negative*
TPR	*True Positive Rate*
FPR	*False Positive Rate*
SMOTE	*Synthetic Minority Oversampling Technique*
ROS	*Random Oversampling*

Abstract

Credit risk assessment is one of the most crucial elements in the financial sector. The study is based on one Lithuanian Loan-comparison platform's applicants' data. Loan-comparison platforms are innovative financial intermediaries that allow consumers to compare several loan offers and choose the most self-favourable. This study presents a comprehensive analysis of statistical and machine learning techniques for predicting individual credit scores using Lithuanian data. Five algorithms, i.e., Logistic Regression, Support Vector Machine, Random Forest, XGBoost and Artificial Neural Networks, are included in our analysis. To deal with the high class imbalance problem the resampling techniques Random Oversampling and SMOTE are also applied. In addition, the analysis is done using binary, 3-Class and 5-Class classification problems. It is the first study in the credit scoring field, to the best of the author's knowledge, that combines the results from both binary and multiclass classification problems. The performance is assessed considering different performance evaluation metrics (Accuracy, AUC, Training time, Precision (Macro Precision), Recall (Macro Recall), False Positive rate and False Negative rate). The final scoring method was proposed to combine the results of different classification experiments. The Random Forest and XGBoost were the best performing algorithms in predicting individual credit scores based on the final scoring method. Empirical results revealed that the best algorithms perform comparatively better in 3-Class classification problem than in binary classification case. Furthermore, the resampling techniques help to predict the minority (risky) class significantly better, where the XGBoost algorithm with the Random Oversampling technique applied reaches an impressive 0.804 Accuracy in predicting the Riskiest E class in the 3-Class classification problem. Finally, this study provides recommendations for applying the credit scoring system in the Lithuanian Loan-comparison platform. Implementing a credit scoring system could help reduce the possibility of lending to a risky customer and avoid additional expenses for external credit scoring agencies that may lack complete information about the customer.

Key words: Credit score prediction, Machine learning, Risk analysis, Resampling techniques, Binary classification, Multiclass classification

1. Introduction

Credit scoring is one of the main determinants of sustainability in the financial world. Due to its importance after the 2007-2008 Financial crisis, many researchers tried to maximize the prediction rates to classify potential risks and defaults correctly. Even though traditional banks have developed their individual credit risk assessment processes, technological innovations have opened new opportunities for customers to access financial services. The scope of this research will be based on one Lithuanian Loan-comparison platform's applicants' data. Loan-comparison platforms are innovative financial intermediaries that allow consumers to compare several loan offers and choose the most self-favourable. However, improper lending could cause enormous losses for creditors. The individual credit scoring system could help the Loan-comparison platform to better distinguish between risky and reliable customers by using privately obtained data about each customer. That would allow company to avoid the possible bias from external credit rating agencies that may lack complete information about the borrower and, in turn, to save some financial resources.

The aim of this study is to analyse the potential of statistical and machine learning techniques for predicting individual credit ratings using the newly introduced Lithuanian credit dataset and to propose a methodology that allows combining the results of different classification methods based on performance evaluation metrics. To achieve it, the following objectives must be solved:

- Extensive literature review;
- Classification in both binary and multiclass problems;
- Comparison of different resampling techniques;
- Evaluation of statistical and machine learning techniques for predicting individual credit ratings performance using a variety of performance evaluation metrics;
- Suggest the methodology, that allows the results of different classification methods to be combined based on the final score of the performance evaluation metrics.

Based on the **extensive literature review**, it was decided to use five statistical and machine learning techniques in our work. Traditional classification techniques like Logistic Regression (LR) and Support Vector Machine (SVM) are used, together with tree-based ensemble methods Random Forest (RF) and eXtreme Gradient Boosting (XGBoost). Finally, we will implement different Artificial Neural Network (ANN) architectures to study the potential of neural networks on credit scoring problem.

The unique detail of our research is the implementation and comparison of statistical and machine learning techniques in both **binary and multiclass classification problems**. The majority of past authors used only binary classification to study the performance of statistical and machine learning techniques in credit scoring. The multiclass type classification was not extensively studied and was mostly used in corporate credit risk assessment problems. Additionally, none of our analysed authors used both binary and multiclass classification problems to compare the performance of classifiers. The nature of the Lithuanian credit dataset allows us to distribute credit ratings into Two, Three and Five classes. Such different environments will allow us to compare the potential of classifiers more precisely by analysing their performance in different types of problems using the same dataset.

Usually, credit scoring datasets face serious class imbalance problem. Naturally, only a minority of customers can be considered risky and probable to default. To deal with the class imbalance problem, our analysis will use **different resampling techniques** like Random Oversampling (ROS) and Synthetic Minority Oversampling Technique (SMOTE). The potential of resampling techniques will be compared with the performance of non-resampled data.

The results of statistical and machine learning techniques will be obtained using the **extensive list of performance evaluation metrics**. Simple accuracy may not be the primary goal in credit risk assessment problem since the misclassification of the risky class customers causes the most significant losses. On the other hand, by misclassifying a reliable customer, the cost of potential borrower and revenue appears. Having different sources of potential losses, it is necessary to compare accuracy in general and the performance of classifiers for each class of credit rating. This is why we will use various performance evaluation metrics like Accuracy, AUC, Training time, Precision (Macro Precision), Recall (Macro Recall), False Positive rate and False Negative rate in our study.

The **final scoring formula** is also presented based on the author's expert knowledge of the market specifics. This score would allow us to rank the statistical and machine learning techniques combining the results from different performance evaluation metrics through all the classification problems using various resampling techniques. To the best of the author's knowledge, none of the previous studies created such a final scoring formula to combine the results from various experiments, which adds additional novelty to our study.

The experimental results show that ensemble classifiers (Random Forest and XGBoost) performed impressively well compared with other statistical and machine learning techniques. Algorithms performed better when resampling techniques (ROS and SMOTE) were applied. The most visible improvements of resampling techniques were visible with respect to the False Positive rate and the classification Accuracy of the Riskiest customers. This aspect is the most crucial part of the credit scoring field, and even slight improvements allow us to avoid possible losses. Based on these results, a unique methodology for the Lithuanian loan comparison platform can be suggested. The use of both Random Forest and XGBoost algorithms, together with minority class oversampling techniques, must be considered in implementing risk assessment systems in real-life scenarios.

The paper is organized as follows. Section 2 reviews the related studies on credit rating classification. Section 3 introduces the dataset used in our research and all the pre-processing steps. Section 4 presents the methodology of statistical and machine learning techniques, together with an extensive description of performance evaluation metrics. Section 5, presents empirical results and findings. Finally, Section 6 draws the main conclusions, suggests possible implementations and provides future work potential.

2. Literature review

The appropriate analysis of statistical and machine learning techniques requires an understanding of what has already been done in this field. Extensive literature analysis of techniques in predicting and classifying credit scores was one of the primary tasks of this work. It would allow us to understand the most successful algorithms so far and what measures can be employed to evaluate the performance of those techniques.

The majority of the recent studies have concentrated on binary classification problems (e.g., individual will default or not). However, such classification does not account for the situation in the real world. Individuals are not only "Black and White", the intermediate levels of riskiness are also possible. Therefore, we will analyse three types of classification problems (binary, 3-class, and 5-class). Below, in Table 1, the type of classification problems analysed in the previous studies are presented.

Table 1. A list of the past studies in credit scoring field

Research	Binary Problem	Multiclass
Ampountolas et al. (2021)		Yes
Dastile and Celik (2021)	Yes	
Gunnarsson et al. (2021)	Yes	
Mahbobi et al. (2021)	Yes	
Moscato et al. (2021)	Yes	
Tripahi et al. (2021)	Yes	
Golbayani et al. (2020)		Yes
Li and Chen (2020)	Yes	
Trivedi (2020)	Yes	
Munkhdalai et al. (2019)		Yes
Sariannidis et al. (2019)	Yes	
Wallis et al. (2019)		Yes
Cao et al. (2018)	Yes	
Hamori et al. (2018)	Yes	
Namvar et al. (2018)	Yes	
Zhu et al. (2018)	Yes	
Barboza et al. (2017)	Yes	
Luo et al. (2016)		Yes
Lessmann et al. (2015)	Yes	
Wu et al. (2014)		Yes
Brown and Mues (2012)	Yes	
Bellotti and Crook (2009)	Yes	
Lee (2007)		Yes
Baesens et al. (2003)	Yes	

Table 1 confirms that most researchers concentrated on binary classification problems, and none of the studies analysed both binary and multiclass problems simultaneously. The nature of our dataset allows us to separate customers either in binary or multiclass problems (Details will be presented in the next section).

2.1. Multiclass classification problem

Having such a background, Table 2 provides a detailed overview of the studies done in multiclass classification field.

Table 2. An overview of studies done for multiclass classification problem

Multiclass Data	**Individuals/ Corporate credit scores**	**Number of classes**	**MLR**	**Ensembles**	**DT**	**RF**	**NN**	**SVM**
Ampountolas et al. (2021)	Individuals	3		AdaBoost, XGBoost	Yes	Yes	MLP	
Golbayani et al. (2020)	Corporate	19			Bagged	Yes	MLP	Yes
Wallis et al. (2019)	Corporate	6	Yes	Gradient Boosted Machines		Yes		Yes
Munkhdalai et al. (2019)	Individuals	8	Yes	XGBoost		Yes	MLP	Yes
Luo et al. (2016)	Corporate	3	Yes				MLP and DBN	Yes
Wu et al. (2014)	Corporate	9		Bagging	Yes		MLP	Yes
Lee (2007)	Corporate	5						Yes
Total			**3**	**4**	**3**	**4**	**5**	**6**

In Table 2, only those machine learning techniques that were repeated more than twice through all the studies are included. Machine learning techniques like Support Vector Machines and Multilayer Perceptron were the most popular methods in classifying multiclass credit scores. Random Forest and other Ensemble methods received more attention in recent studies. Traditional statistical and machine learning techniques like Multinomial Logistic Regression and Decision Tree were also popular in past studies.

Across all reviewed studies, only Ampountolas et al. (2021) and Munkhdalai et al. (2019) have devoted their analysis to classifying individual (People) credit scores. All other authors have been analysing the performance of statistical and machine learning techniques in the corporate (companies' riskiness) field. Even though the objective to classify the level of riskiness for companies and individuals is the same, the features that describe individuals differ a lot and, in turn, may affect the whole classification accuracy. A shortage of studies in Multiclass classification for individuals implies a considerable novelty to our study and highlights its importance.

Furthermore, the previous studies have chosen a specific number of classes they used in their analysis. For example, Wallis et al. (2019) have assigned Moody's Ratings {Aaa, Aaa1, Aa2, Aa3} to the “High Grade” category, {Baa1, Baa2, Baa3} to “Upper Medium Grade” and so on. Luo et al. (2016) have also aggregated rating categories to A = {A, AA, AAA}, B = {B, BB, BBB} and C = {C, CC, CCC}. Having such an aggregation of credit scores does not allow us to correctly compare the results of these studies since, by nature, higher dimension (19 classes) problems are more complex and are associated with poorer accuracy than smaller dimension (3 or 5 classes) problems. Our study will face this problem and compare classifiers' performance via two different multiclass problems (3-class and 5-class). It is presumed that by doing so, we will be able to obtain more stable results of statistical and machine learning techniques’ performances.

Ampountolas et al. (2021) and Golbayani et al. (2020) found that tree-based methods classify credit scores with the highest accuracy across all the techniques. Ampountolas et al. (2021) highlight that machine learning algorithms (Random Forest, XGBoost and AdaBoost) can perform well without full credit history information. These techniques achieved a very similar accuracy of 80%. Our study also lacks complete information about an individual’s financial history, but the results of these three ensembles are very promising. Golbayani et al. (2020) also show that tree-based methods can achieve at least 5% higher accuracy than MLP and SVM techniques.

Wallis et al. (2019) and Munkhdalai et al. (2019) approve the findings of the previous authors. Wallis et al. (2019) summarize that non-parametric techniques like Random Forest or SVM outperform parametrical techniques. Especially because financial data does not follow traditional assumptions, which are needed for parametrical techniques. Munkhdalai et al. (2019) find the superior performance of MLP and XGBoost techniques against traditional techniques like SVM or Logistic Regression. The authors also highlight the fact that the Neural Network approach (MLP) may be useless in the credit scoring field due to its “Black-Box” nature, which makes it impossible to understand the relationship between the attributes and credit score itself.

Summarizing the studies done in the multiclass credit scores classification field, there is no conclusive answer to which method is significantly superior. Statistical and machine learning techniques should be compared individually on a given credit dataset to optimally decide which method yields the highest accuracy. However, we do have the first signals that ensemble methods and mostly Random Forest can be expected to be the best performing algorithm based on the most recent studies.

Similar to Table 2, it is necessary to compare what performance evaluation metrics were the most popular in the past studies. A single evaluation metric can be misleading or hide some valuable information (e.g., False Negative cases in credit scoring problems are more dangerous than False Positive cases. False Negative predictions lead to approving a loan to a risky person who will probably default, and that causes financial loss for the lender). Table 3 presents evaluation metrics used in past multiclass classification studies. It can be seen that every author used a fundamental performance evaluation metric Accuracy rate, which will also be included in our research. Additionally, we will use the AUC measure, which optimize the trade-off between sensitivity and specificity by finding the optimal threshold. To be more precise, AUC is the evaluation of a specific classifier as the threshold varies across all possible values.

Table 3. List of accuracy measures used in the past literature for multiclass classification algorithms

Binary Data	**Accuracy**	**AUC**	**Measures obtained from Confussion matrix**	**Confussion matrix**	**Prediction Distance measures**
Ampountolas et al. (2021)	Yes	Yes	Precision and Recall	Yes	
Golbayani et al. (2020)	Yes				Notch Distance
Munkhdalai et al. (2019)	Yes	Yes	TPR and FPR		
Wallis et al. (2019)	Yes				
Luo et al. (2016)	Yes	Yes	FN and FP	Yes	
Wu et al. (2014)	Yes				1-Away accuracy
Lee (2007)	Yes				
Total	7	3	3	2	2

2.2. Binary classification problem

Similar to Table 1, an overview of studies that analysed binary credit scoring problems is presented in Table 4.

Table 4. An overview of studies done for Binary classification problem

Binary Data	Domain of interest	LR	RF	DT	SVM	LDA	KNN	NN	Ensembles	NB
Moscato et al. (2021)	P2P (individuals)	Yes	Yes					MLP		
Mahbobi et al. (2021)	individuals				Yes		Yes	ANN; DNN		
Tripathi et al. (2021)	individuals		Yes	Yes	Yes	Yes	Yes	MLP;TDNN; RBFN		
Gunnarsson et al. (2021)	individuals	Yes	Yes	Yes				MLP; DBN	XGboost	
Dastile and Celik (2021)	individuals							CNN		
Li and Chen (2020)	individuals	Yes	Yes	Yes	Yes			NN	AdaBoost,XGboost	Yes
Trivedi (2020)	individuals		Yes	Yes	Yes					Yes
Sariannidis et al. (2019)	individuals	Yes	Yes	Yes			Yes			Yes
Cao et al. (2018)	individuals	Yes	Yes	Yes	Yes	Yes		MLP	XGboost	
Namvar et al. (2018)	P2P (individuals)	Yes	Yes			Yes				
Hamori et al. (2018)	individuals		Yes					NN and DNN	Bagging, Boosting	
Zhu et al. (2018)	individuals	Yes	Yes					CNN		
Barboza et. al. (2017)	Companies	Yes	Yes		Yes	Yes		ANN	Bagging, Boosting	
Lessmann et al. (2015)	individuals	Yes	Yes	Yes	Yes	Yes	Yes	Yes (various)	Yes (various)	Yes
Brown and Mues (2012)	individuals	Yes	Yes	Yes	Yes	Yes	Yes	MLP	Gradient boosting	
Bellotti and Crook (2009)	individuals	Yes			Yes	Yes	Yes			
Baesens et at. (2003)	individuals	Yes		Yes	Yes	Yes	Yes	MLP		Yes
Total		12	13	9	10	8	7	13	7	5

Differently than in the multiclass classification case, these authors analysed individuals' credit riskiness rather than companies. To be more specific, the majority of those studies considered "default" status as the main dependent variable. Based on Table 4, we can confirm that most statistical and machine learning techniques coincide with those analysed in the multiclass case.

Random Forest, Logistic Regression and Support Vector Machines are among the most popular algorithms in classifying binary credit riskiness problems. Neural Networks and Decision tree methods are also very popular, as they were in studies analysing multiclass problems. Linear

Discriminant Analysis and Naïve Bayes are losing popularity in recent studies. At the same time, Ensembles and K-Nearest Neighbours approaches often deal with binary credit riskiness problems.

Recent improvements and advantages of Deep Learning techniques, especially Convolutional Neural Networks (CNN), have affected a lot of real-life classification tasks. However, only 2 of our analysed authors employed Convolutional Neural Networks to classify binary problems, which implies that these methods are needed to be researched further.

Baesens et al. (2003) is the oldest study included in our review. However, this is one of the most famous studies in the whole credit classification field that compares the performance of various algorithms. Authors found that SVM and NN methods yield great performance, but simple classifiers like LR and LDA perform comparatively well. Having such a finding, it was summarized that credit data is only weakly non-linear. Authors concluded that further research on ensemble algorithms should be up-and-coming, comparing with traditional methods.

Bellotti and Crook (2009) approve the findings of Baesens et al. (2003) and confirm that SVM is competitive with LR and LDA classifiers, while the KNN technique showed the poorest results.

Naturally, credit datasets imply a high “class imbalance” problem due to a small proportion of risky (defaulted) customers. Brown and Mues (2012) checked the performance of statistical and machine learning techniques in the case of a high class imbalance environment. It was found that Random Forest performs very well in such a scenario. Similar to the previous authors, it was found that LR and LDA methods were relatively competitive, but Decision Tree and SVM were not beneficial.

Lessmann et al. (2015) study could be called an update on a benchmarking study of classification algorithms in the credit scoring field made by Baesens et al. (2003). The authors found that some of the classifiers performed significantly better than LR. Especially, ensemble methods were found to be outperforming the previous research-standard techniques. It was concluded that RF could be recommended as a new benchmark algorithm.

A later study by Barboza et al. (2017) confirms that ensemble techniques boosting, bagging and RF provide better classification accuracy. It is even highlighted that Random Forest may produce 20% higher accuracy than LR or LDA.

In the last decade, deep learning algorithms have been increasingly popular in many real-world applications. Credit scoring is not an exception. Hamori et al. (2018) compare the performance of already successful ensemble techniques with various deep learning architectures. It was found that boosting (ensemble) is still superior to other deep learning methods.

Later studies by Trivedi (2020), Li and Chen (2020), Moscato et al. (2021) and Gunnarsson et al. (2021) also confirm that RF and other ensemble techniques could be called the best methods for binary credit scoring classification.

Li and Chen (2020) also found that ensemble learning (especially RF and XGBoost) outperforms individual classifiers. However, authors have also found that LR outperforms all other counterparts in comparison to individual classifiers.

Moscato et al. (2021) devoted their study to analyzing the class imbalance problem and comparing various resampling techniques. It was found that RF, together with the Random Undersampling technique, yielded the best results. It can be seen that even in the studies analyzing feature selection or class imbalance problems, Random Forest is dominating technique.

Gunnarson et al. (2021), similar to Hamori et al. (2018), compare ensembles with deep learning methods (MLP and Deep Belief Network (DBN)). It was concluded that ensemble XGBoost is the best-performing algorithm, and deep learning algorithms cannot outperform their counterparts (with one layer) while being more computationally expensive. However, a few studies have concluded with different outcomes. Tripathi et al. (2021) found that Time Delay Neural Network (TDNN) outperformed the RF algorithm, while feature selection algorithms were also implied. Furthermore, Mahbobi et al. (2021) found that SVM was the most successful technique (against various Neural Networks and KNN) when resampling techniques were also considered.

Similar to the multiclass case, we need to compare the performance evaluation metrics used in the binary credit riskiness classification field. Table 5 shows that accuracy and AUC metrics were the most popular performance evaluation metrics in the past literature and almost every study used these metrics. Sensitivity/Specificity, which was also very popular in past studies, helps to understand more correctly what type of mistakes algorithms make and what enhances the understanding of possible problems of our classifiers. Brier score, G-measure, F-measure and K-S statistic were also used in the literature. However, their use was considerably rare.

Table 5. List of accuracy measures used in the past literature for Binary classification algorithms

Binary Data	**Accuracy**	**AUC**	**Measures obtained from Confussion matrix**	**G-measure**	**Brier Score**	**F-Measure**	**K-S statistic**
Moscato et al. (2021)	Yes	Yes	Sensitivity, Specificity and FPR				
Mahbobi et al. (2021)	Yes	Yes	Sensitivity and Specificity	Yes	Yes	Yes	
Tripathi et al. (2021)	Yes		Sensitivity and Specificity	Yes			
Gunnarsson et al. (2021)		Yes			Yes		
Dastile and Celik (2021)	Yes	Yes			Yes		
Li and Chen (2020)	Yes	Yes			Yes		Yes
Trivedi (2020)	Yes		FP and FN			Yes	
Sariannidis et al. (2019)	Yes						
Cao et al. (2018)	Yes	Yes					
Namvar et al. (2018)	Yes	Yes	Sensitivity, Specificity and FPR	Yes			
Hamori et al. (2018)	Yes	Yes				Yes	
Zhu et al. (2018)	Yes	Yes					Yes
Barboza et al. (2017)	Yes	Yes	TP, TN, FP, FN, Type I error, Type II error				
Lessmann et al. (2015)	Yes	Yes			Yes		Yes
Brown and Mues (2012)		Yes					
Bellotti and Crook (2009)		Yes					
Baesens et al. (2003)	Yes	Yes	Sensitivity and Specificity				
Total	14	14	7	3	5	3	3

2.3. Literature review summary

Many studies were already done in the field of credit scoring, which shows the topic's relevance. Finding the best algorithm through all the studies was hard in the multiclass classification problem. There are some signals that ensemble methods and mostly Random Forest can be expected to be the best performing algorithm. On the other hand, in a binary classification problem, most studies confirm that Random Forest and other ensemble techniques could be called a benchmark in the credit scoring field.

To the best of the author’s knowledge, none of the previous authors had compared the performance of algorithms in both binary and multiclass credit score classification problems. Our analysis will contribute to the research field by combining the results from binary, 3-class and 5-class classification tasks.

The extensive literature review helps to decide which algorithms need to be included in the analysis of statistical and machine learning techniques in predicting individual credit scores. Based on the popularity and performance of past studies, five algorithms are selected: Logistic Regression (Multinomial Logistic Regression for multiclass case), Support Vector Machine with Radial basis kernel function, Random Forest, XGBoost and Artificial Neural Networks. Additionally, based on the literature review, it can be decided that more than one performance evaluation metric is needed to be included for the comprehensive analysis of classifiers.

3. Data Description

This study uses unique and privately observed data from one Lithuanian loan-comparison platform. The Lithuanian credit dataset is collected from January 2020 to September 2022. The loan-comparison platform operates as follows. Each individual (applicant can fill out the application form for a loan. There he needs to submit all the personal data (age, gender, marital status and similar). Afterward, the customer must authenticate his identity and submit the required documents. Later, some additional data, which further describes the potential borrower, is downloaded from various legal institutions. A total number of 29 dependent variables are used in this study and the complete list of variables with descriptions is presented in Appendix 1.

It can be seen from Appendix 1, that the dataset includes three sources of data.

1. **Data from legal institutions and registers**. Different financial characteristics of applicants are obtained from various legal institutions. Official income from The State Social Insurance Fund Board under the Ministry of Social Security and Labour (SODRA), active or past debts and credits, employment duration and similar. (Sources cannot be fully disclosed due to confidentiality). Some financial characteristics obtained from legal institutions overlap with data filled in by customers. This allows us to analyze a novel feature in credit scoring – a comparison of self-evaluation against official data. Maybe it is the case that clients hope to get a better loan offer by lying about their financial situation. Alternatively, maybe they even cannot correctly evaluate their financial burden. All these aspects may be new significant features in estimating credit scores.
2. **Data from the application form**. In this category, all the answers from the application, filled in by the individual, are included. We are obtaining some general demographic characteristics like gender, age, city and similar. Additionally, clients submit their evaluation of their financial situation: income, financial obligations and income source.
3. **Applicant's behavior data**. Each individual fills out the application form differently, which may be a significant factor considering the borrower's reliability. When desperately searching for a loan, one may use the "Copy-Paste" method and quickly fill in the whole application. Desperation is related to riskiness and having information about an individual's behavior while filling out the application form allows us to test whether it has significant power in predicting the riskiness of the customer.

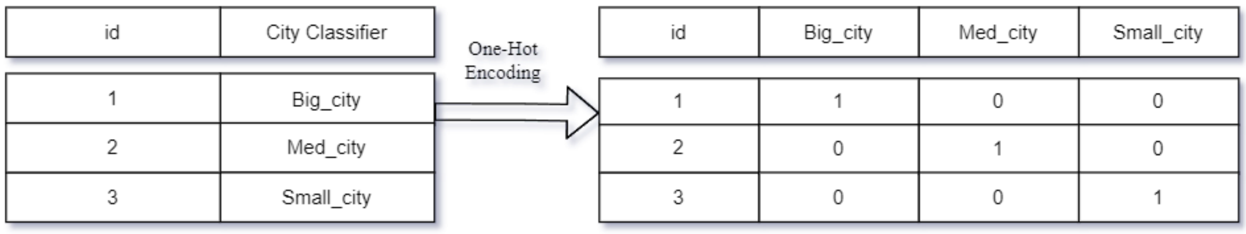
3.1. Data preprocessing

Before implementing statistical and machine learning techniques, it is necessary to prepare the dataset for analysis correctly. Firstly, all the variables that do not have any predictive power were removed from the dataset. Such variables are *application_id* and various timestamps, which were used to calculate the duration of application filling and authentication. Later, observations that had missing or erroneous values were also removed. Finally, we are left with data of 29289 individuals and 30 features (including the target variable). Removing features with a higher than 0.6 correlation was decided to deal with a possible multicollinearity problem. The full correlation matrix is presented in Appendix 2. Using the correlation results, it was decided to exclude variables *DSTI_official* and *Filled_income* from further analysis. *DSTI_official* has an almost perfect positive correlation with the *active_credit_monthly_sum* variable and it is obvious because both represent the amount of monthly financial commitments. *Filled_income* variable was used in the formula to calculate *Income_fill_difference* (as explained in Appendix 1).

The ranges of the features used in our study differ a lot. To appropriately use some statistical and machine learning techniques, it is necessary to unify the ranges of variables. For that reason, we normalized all numerical variables using the following formula:

$$x_{normalized\ i} = \frac{x_i - x_{min}}{x_{max} - x_{min}}.$$

Here $x_{normalized\ i}$ is the normalized value of variable *i*, x_{min} is the minimum value of x_i and x_{max} is the maximum value of x_i. Using this method, we rescale the range of all numerical features in the interval between 0 and 1. Similarly, it is necessary to prepare all the categorical data before using statistical and machine learning techniques. One-hot encoding method was used to convert all the categorical or boolean variables into numerical expressions of 0 or 1. Below, the representation of One-hot encoding for *City_classifier* is presented.



id	City Classifier
1	Big_city
2	Med_city
3	Small_city

id	Big_city	Med_city	Small_city
1	1	0	0
2	0	1	0
3	0	0	1

Figure 1. Example of One-Hot Encoding procedure

3.2. Risk Classes

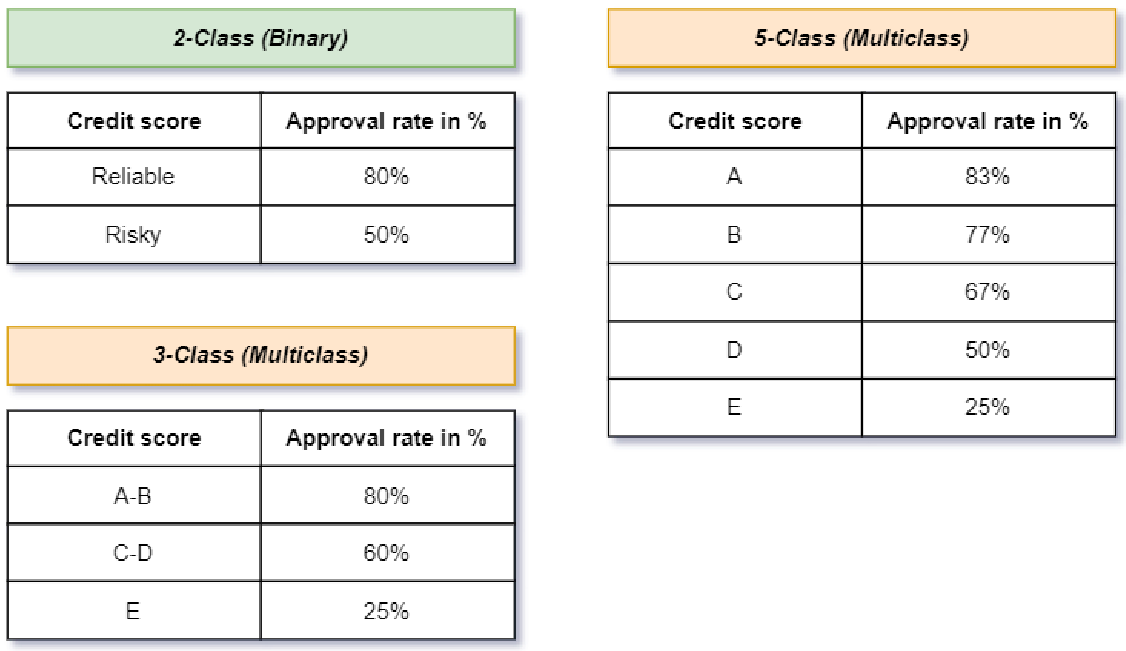
As presented in the introduction section, the unique detail of our study is the implementation and comparison of statistical and machine learning techniques in both binary and multiclass classification problems. The credit score is obtained from one credit rating agency, which implies additional expenses for the loan-comparison platform and a lack of complete information about the borrower's recent behavior. Ratings in 5 classes, namely {A, B, C, D, E} are obtained. However, assigning them to different class groupings based on individual information and market experience is possible. Our research and analysis of statistical and machine learning techniques for predicting individual credit scores will be done using the following classification problems:

- Binary (2-class) classification problem
- Multiclass (3-class) classification problem
- Multiclass (5-class) classification problem

Multiclass (5-class) classification problem will be performed using all five obtained credit scores separately. The possible target values for this problem will be *{A; B; C; D; E}*. Multiclass (3-class) classification problem will use A-B credit scores as first-class, C-D scores as a second class, and E as the third class. So, the possible target values for this problem will be *{A-B; C-D; E}*. Furthermore, the binary classification problem will use A-B credit ratings as a "reliable" customer and C-E ratings as "risky". The reasoning for separating credit scores into different classes can be explained by using the approval rate metric. This metric shows what percentage of clients received at least one loan offer, and the formula is presented below.

$$Approval\ rate = \frac{Offers}{Applications}$$

Where *Offers* is the number of applications where at least one loan offer is generated, and *Applications* is the number of total applications. Below, in Figure 3, you can see what approval rates are associated with each type of grouping. It can be found that there exist significant differences across these groups. Tables of approval rates show that in the credit market, a high difference between reliable and risky customers exists and a proper risk assessment system based on statistical and machine learning techniques may allow credit company to distinguish between these customers more efficiently.



2-Class (Binary)

Credit score	Approval rate in %
Reliable	80%
Risky	50%

3-Class (Multiclass)

Credit score	Approval rate in %
A-B	80%
C-D	60%
E	25%

5-Class (Multiclass)

Credit score	Approval rate in %
A	83%
B	77%
C	67%
D	50%
E	25%

Figure 2. Tables of approval rates for different groupings of target variable

3.3. Test and Train splits

The filtered and preprocessed data was split into training and testing sets using the "80:20" method. 80% of all data was used as a training set and 20% was left for testing. In the case of Artificial Neural Networks, a validation set is necessary additionally. We kept our primary testing set untouched to allow comparison of all methods using the same testing set. However, we split our primary training set into 80% of the training set for Artificial Neural Networks and 20% for validation.

3.4. Data Balancing

High-class imbalance is a typical problem for most credit scoring datasets, as only a minority of customers belong to the riskiest segment (defaulters). There is no exception for the Lithuanian credit dataset as the riskiest E-rating customers consist of only 5% of all the applicants. In comparison, B-rating customers consist of 47% of the dataset. To deal with the class imbalance problem, resampling techniques like Random Oversampling (ROS) (Moscato et al. (2021)) and Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al. (2011), Mahbobi et al. (2021), Moscato et al. (2021)) are implemented for the training set and results are compared with non-resampled (Normal) data. The idea of Random Undersampling technique was rejected because removing the majority class observations leaves us with a tiny amount of data, which makes the training set smaller than the testing set. Resampling techniques were not applied for the testing set because the interest of our study is to analyse the performance of statistical and machine learning techniques on a real-life dataset and applications flow.

3.5. Summary statistics

Table 6 presents the summary statistics of the numerical features used in our analysis. Most financial variables face a mean higher than the median and a high standard deviation. The reason behind these numbers is that majority of risky customers have very high amounts of debt or active credits, which significantly affect the higher mean. Since there are considerably more reliable customers, the median is affected downward, which makes it less than the mean. Table 6 also supports the idea that ranges of numerical variables differ significantly and that data normalization is highly recommended. For categorical variables, we present each category's proportion in Appendix 3.

Table 6. Summary Statistics of numerical features used in our analysis

	Mean	Median	Standard Deviation	1st Quartile	3rd Quartile
Official_income	1000.9	903.3	623.62	642.7	1248.4
Quer_by_90d	4.27	2	6.67	1	5
Open_debt_count	0.4032	0	0.286	0	0
Open_Debt_sum	25.08	0	381.53	0	0
Past_Debt_count	6.36	0	19.78	0	3
Past_Debt_sum	1389.4	0	7675.459	0	227.9
Employment_duration_months	55.56	31	63.6	12	75
Active_credit_sum	18945	2751	34565	0	20990
Active_credit_monthly_sum	178.7	85.7	3033.603	0	21.4
Req_amount	10960	10000	6680.348	5000	15000
Age	38.02	37	10.13	30	45
Children	0.8187	1	0.9144	0	1
Filled_obligations	137.4	60	207.44	0	220
application_filling_duration	197.1	121	440.1467	82	189
Income_fill_difference	240.87	31.29	857.31	−61.37	284.34
Auth_duration	56	68	130.42	37	85
Auth_Number	1	1	0.54	1	1

4. Methodology

The experimental procedure is shown in Figure 4. The complete analysis of statistical and machine learning techniques to predict credit scores can be separated into six different phases.

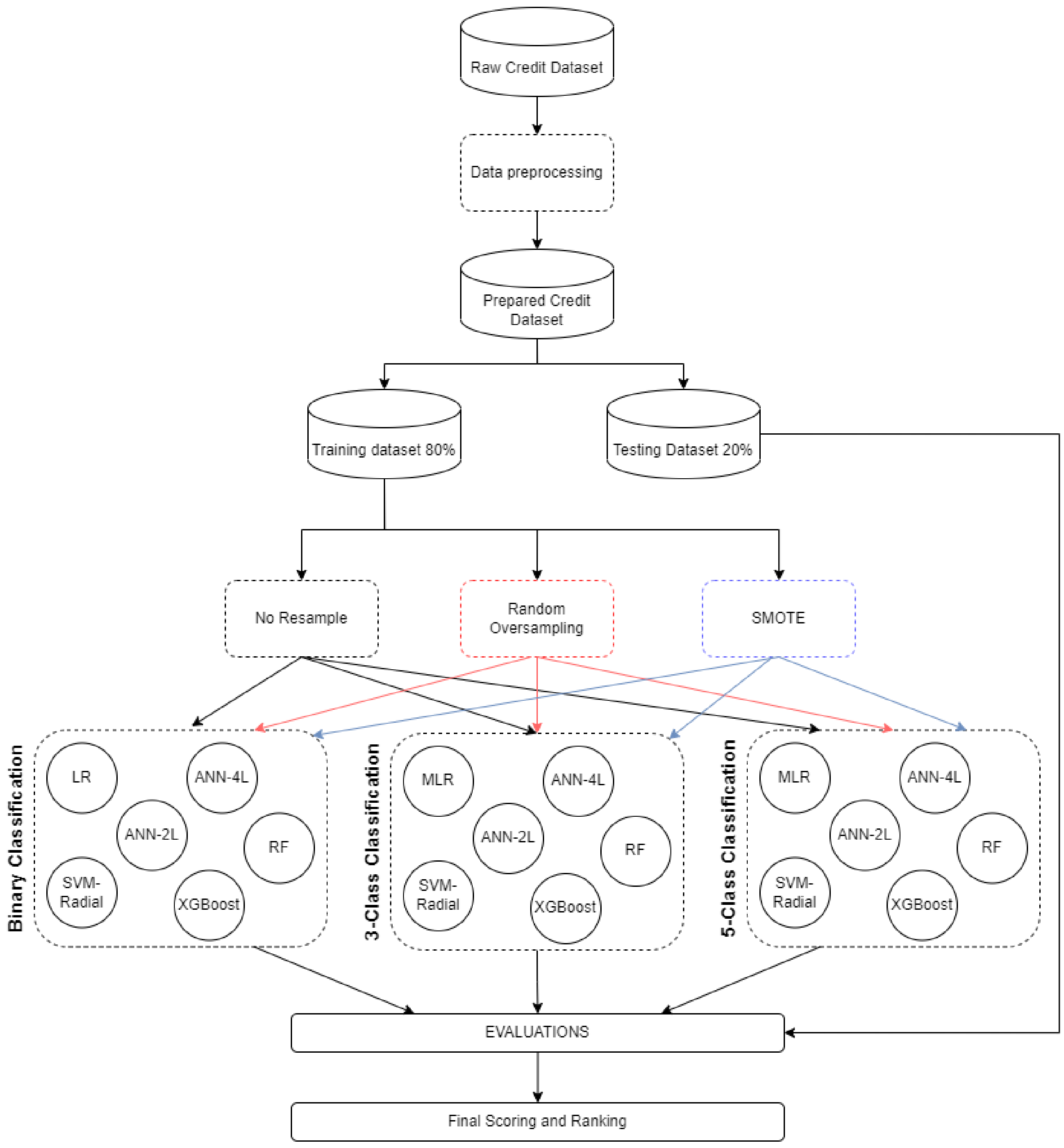


Figure 3. The process of analysis

Firstly, the raw credit dataset must be properly preprocessed before analyzing statistical and machine learning techniques, as presented in the 3.1 section. Later the preprocessed dataset is split into training and testing samples based on the 80:20 rule (3.3 section). The credit dataset faces the class imbalance problem. Therefore, it is necessary to conduct experiments using various sampling techniques (3.4 section). The performance of six statistical and machine learning techniques is analysed in our study. Each of the algorithms will be more explicitly presented in the 4.1 section.

In addition, three different classification problems are included in our research (binary, 3-class and 5-class classification, as presented in the 3.2 section). Finally, the results of six classifiers using different sampling techniques and three distinct classification tasks are evaluated using various performance evaluation metrics, which will be presented in the 4.2 section. The final scoring function will be applied based on the performance evaluation metrics. Final scores will determine which algorithms are the best performing statistical and machine learning technique in predicting credit scores, taking into account class imbalance problem, a different dimensions of target variable and various performance evaluation metrics. The final score methodology is presented in the 4.2.3 section.

4.1. Statistical and Machine Learning Techniques

4.1.1. Logistic Regression

In recent decades, logistic regression has been considered a benchmark method for many classification tasks. It encouraged us to include this statistical technique in our analysis to compare whether machine learning techniques can overperform their benchmark. It is assumed that independent variables are linearly related to the log odds (logit), which is the dependent variable of logistic regression.

$$\log(odds) = \ln\left(\frac{P}{1-P}\right).$$

The linear relationship between independent variables and log odds can expand to:

$$\ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X.$$

Where P is the probability of specific event happening, β_0 is the intercept, β_1 is the coefficient associated with the independent variable X. By taking logs out of both sides in the second equation, we are left with:

$$\left(\frac{P}{1-P}\right) = e^{\beta_0 + \beta_1 X}.$$

By converting odds to a simple probability function, we are left with the following simple probability function, that could be named a logistic function:

$$P = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}.$$

Expanding the expression to include more dependent variables and coefficients associated with this is possible. However, due to simplicity, it is shown the case with only one dependent variable. The maximum likelihood estimation is usually used to estimate logistic regression coefficients. The likelihood function can be expressed as follows:

$$l(\beta_0, \beta_1) = \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}.$$

Here x_i is the value of dependent variable and y_i is the binary outcome for observation *i*, *n* is number of total observations, and the probability of dependent variable is *p(x)*, if y = 1 and 1-*p*(*x*), if *y* = 0. The estimates of β_0 and β_1 are chosen to maximize this likelihood function

4.1.2. Multinomial Logistic Regression

Multinomial logistic regression (MLR) is the extension of previously discussed logistic regression for the analysis of multiclass classification problems rather than binary problem. MLR produces probability for more than 2 classes for each input example. Let's assume again that we have a set of data with *n* observations $\{x_i\ ,\ y_i\}$ where $x_i\ \in R^M$ and $y_i \in \{1, \dots, C\}$, for *i* = 1,...,*n*. Using the previous function for logistic regression, we can express the probability for *i*-th observation belonging to the *j*-th class by:

$$p_{ij} = P(y_i = j) = \frac{e^{\beta_j \cdot x_i}}{1 + \sum_{c=1}^{C-1} e^{\beta_c \cdot x_i}}.$$

The coefficients are again estimated using the maximum likelihood estimation method as in logistic regression case. Because the sum of probabilities of all classes must be equal to 1, the last class probability is expressed as follows:

$$p_{iC} = 1 - \sum_{j=1}^{C-1} p_{ij}.$$

4.1.3. Support Vector Machine

Based on the Literature review presented in this study, Support Vector Machine (SVM) is another popular technique used in credit ratings classification tasks. SVM creates hyperplanes that can divide data into different classes. Those hyperplanes are chosen to maximize the distance between the closest data points of different classes. Since financial data cannot often be linearly separated, SVM analysis will include a radial basis kernel modification in this work. Based on Li et al. (2013), the classifier function of SVM for binary classification can be expressed as:

$$f(x) = sign(\sum_{i=1}^{M} \alpha_i y_i K(x, x_i) + b),$$

using training dataset $D = \{x_i, y_i\}$, where $x_i \in R^m$ is the independent variable and $y_i \in \{-1{,}1\}$ is the target class, α_i is Lagrange multiplier, $K(x, x_i)$ is kernel function of the two vectors. In our work, radial basis function will be used as a kernel function, which expression is $K(x, x_i) =$

$e^{(-\gamma\|x-x_i\|^2)}$. The SVM algorithm is discussed in more detail in Boser et al. (1992), Cortes and Vapnik (1995) and Li et al. (2013), while Boser et al. (1992) additionally presents the SVM extension to multiclass classification tasks.

4.1.4. Random Forest

Random Forest (RF) is the first algorithm in our analysis based on the ensemble of classification trees. In the Literature review section, it was noted that RF is among the most popular and successful techniques used in binary and multiclass credit score classification problems. This ensemble method was developed by Breiman (2001). Random Forest is an extension of another ensemble method called bagging. As in bagging, Random Forest builds a number of decision trees on bootstrapped training samples. Instead of using all available feature variables, the RF algorithm randomly chooses a subset of *m* predictors from the full set of *p* predictors ($m < p$). In the case of one powerful predictor in the dataset, the bagged decision trees could look very similar because all of the trees will use this strongest predictor in the top split. Therefore, the predictions from separate bagged trees would be highly correlated. Random Forest overcomes this issue by taking only a subset of predictors, and this process is called *decorrelation* of the trees. After building numerous uncorrelated decision trees in the forest, the algorithm uses majority voting to decide which category a given observation belongs to.

4.1.5. XGBoost

XGBoost (extreme gradient boosting) is the second algorithm used in our study based on the ensemble of classification trees. Firstly it was proposed by Chen and Guestrin (2016) and received a lot of popularity and success in the credit scoring field in recent years (based on the Literature review presented in this study). Both Random Forest and XGBoost build a model based on multiple decision trees. However, the process of "building" is the main difference between those algorithms. Random Forest uses bagging to build all decision trees at once. On the other hand, XGBoost constructs an ensemble of decision trees using a gradient boosting algorithm to build trees to minimize the loss sequentially. In addition to gradient boosted decision trees, Chen and Guestrin (2016) suggested adding a regularization (penalty) term to the loss function to avoid possible overfitting:

$$L(f) = \sum_{i=1}^{n} \Psi(\widehat{y_i}\ , y_i) + \sum_{k=1}^{K} \Omega(\delta_k).$$

Here $\hat{y}_i$ is the prediction of *i*-th observation at *K*-th boost (tree), $\Psi(*)$ is the lost function to measure the difference between prediction and the reference label and $\Omega(\delta_k)$ is the regularization term. This regularization term can be expressed as:

$$\Omega(\delta) = \gamma T + \frac{1}{2}\lambda\|\omega\|^2.$$

Here γ is complexity term, *T* is the of classification's tree number of leaves, λ is penalty parameter and $\|\omega\|^2$ is the output of each leaf node. Additionally, XGBoost applies second-order Taylor approximation in the loss function and this is another difference from gradient boosted decision trees. More details about this algorithm can be found in Chen and Guestrin (2016).

4.1.6. Artificial Neural Networks

Artificial Neural Networks (ANN) are another class of models used in our study. Most of the past studies in credit scoring field had used this technique, what highlights the importance of Neural Network models. ANNs consist of three main layers: input, hidden and output layers. The more hidden layers are used, the more complex relationships may be modelled. A feed-forward neural network will be used in our analysis using two different architectures (with two and with four hidden layers). In feed-forward network, the information is carried forward from input variables through connected neurons in the middle (hidden) layers and finally to the specified output layer. Each neuron processes its inputs and transfers its output value to the neurons in the next layer. Initially, these neural connections are assigned with random weights, and then, during the training process, the model adjusts the weights. The output value of hidden neuron *i* is calculated by applying activation function $f^{(1)}$ (ReLU activation function is used in our study based on Munkhdalai et al. (2019), Mahbobi et al. (2021)) to weighted inputs and bias term $b_i^{(1)}$:

$$h_i = f^{(1)}\left(b_i^{(1)} + \sum_{j=1}^{n} W_{ij}x_j\right).$$

Here *W* is the weight matrix, W_{ij} denotes the weight connecting input *j* to hidden neuron *i*. In a similar way, the output of the output layer is computed by:

$$y = f^{(2)}\left(b^{(2)} + \sum_{j=1}^{n_h} v_j h_j\right).$$

Here n_h is the number of hidden neurons and v denotes the weight vector, where v_j is the weight that connects hidden neuron *j* to the output neuron. In the case of binary classification, $f^{(2)}$ is sigmoid activation function, and for multiclass classification, softmax activation function was used. More details about ANNs can be found in Bishop (1995).

4.2. Performance evaluation metrics

The choice of performance evaluation metrics was based on extensive literature review and expert knowledge. Classifying a risky customer as reliable is the most hazardous error in the credit scoring field. For that reason, some additional metrics are presented to cover that kind of error.

4.2.1. Binary Classification metrics

We use Accuracy, AUC, Training time, False Positive rate (FPR), Recall and Precision to evaluate the performance of binary classification algorithms. For a better explanation of these metrics, let us use Table 7, which presents the example of a binary classification confusion matrix. Below all the formulas and explanations are presented for each evaluation metric.

Table 7. Example of binary classification confusion matrix

		Actual values	
		Good	**Risky**
Predicted values	**Good**	*TP*	*FP*
	Risky	*FN*	*TN*

- $Accuracy = \frac{TP+TN}{TP+FP+FN+TN}$, it shows the proportion of all correctly classified examples. Accuracy may not be the most correct classifiers evaluation metric because it assumes equal misclassification costs for false negative and false positive cases. In the credit scoring field, the false positive is more expensive and more attention must be paid to this possible error.
- AUC is the measure of the model's ability to distinguish between classes. AUC is equal to probability that the classifier will rank randomly chosen good observation higher than that of randomly chosen risky observation. AUC uses the true positive rate and false positive rate of the model across all possible thresholds, what allows us to deeper understand the predictive power of the classifier.
- Training time was included in this study for observing the duration of how long model trains.
- $FPR = \frac{FP}{FP+TN}$, it is the rate, which shows what percentage of risky customers was actually predicted as good ones. In credit industry this is the rate that causes the most losses. The goal of this analysis will attempt to minimize FPR as much as possible.

- $Recall = \frac{TP}{TP+FN}$, it shows the percentage of correct positive predictions made out of all possible positive predictions that could have been made by the classifier.
- $Precision = \frac{TP}{TP+FP}$, it shows the quality of good predictions made by the model.

4.2.2. Multiclass classification metrics

Similar to the presentation of evaluation metrics for binary classification, the measures for multiclass classification performance are presented below. Table 8 shows how the confusion matrix for 3-class classification would look and all the performance evaluation metrics are explained.

Table 8. Example of multiclass classification confusion matrix

		Actual values		
		A-B	**C-D**	**E**
Predicted values	**A-B**	*AA*	*AC*	*AE*
	C-D	*CA*	*CC*	*CE*
	E	*EA*	*EC*	*EE*

- Accuracy = $\frac{AA+CC+EE}{Sum(all\ matrix)}$, similar to the binary classification case, it is the percentage of correctly classified observations.
- AUC, differently than in binary classification, must be calculated considering all distinct AUCs for each class. One-versus-rest approach is used to calculate each class AUC value and then by averaging those values, it is possible to obtain the final AUC for the multiclass classification model.
- Training time – similar to the binary classification task, training time for each model will be observed to account for the rapidness of each classifier.
- The Riskiest class (E) recall = $\frac{EE}{AE+CE+EE}$. From the economic point of view, this metric is equivalent to 1-FPR in binary classification case. Recall of E class shows how many E class observations were predicted as E class. In the credit rating field, it is highly important to account for that type of mistake. For that reason, it was highlighted.
- Macro Recall - Similar to AUC, this metric was calculated by averaging the Recall of each class. Macro Recall was included in our research instead of Weighted Recall because the Weighted average of metric includes individual sample sizes. Having a situation where the majority of observations are assigned to the “good” class, it was decided not to weigh the

metric based on the majority class, but do the simple arithmetic mean and treat all the classes equally.

- Macro Precision – Similarly to Macro Recall, it is average of Precision of each class.
- False Positive rate – This metric shows what percentage of all observations was predicted higher credit score than it actually is. The formula for False Positive rate is presented below:

$$False\ Positive\ rate = \frac{AC + AE + CE}{Sum(all\ matrix)}.$$

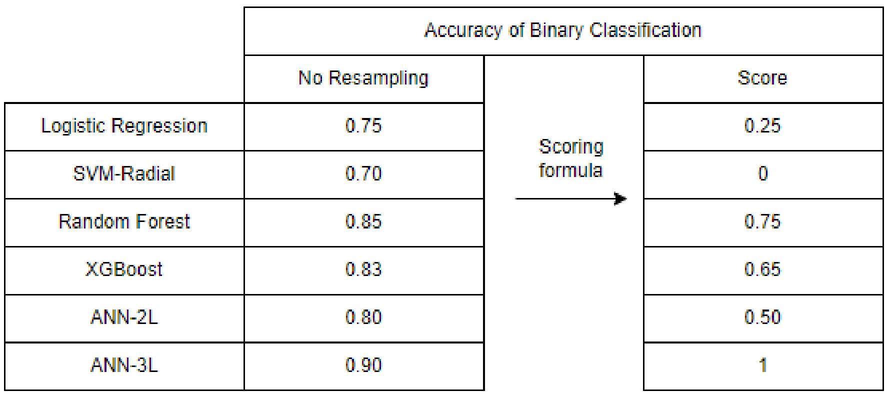
- False Negative rate – Similar to the False Positive rate, this metric shows what percentage of all observations was predicted lower credit score that it actually is. And the formula is:

$$False\ Negative\ rate = \frac{CA + EA + EE}{Sum(all\ matrix)}.$$

Accuracy, False Positive and False Negative rates add up to 1. That was the reason why such rates were included in our study, and this helps us to understand more deeply what type of errors our model makes. The logic and formulas for the 5-class classification problem are the same and can be extended from examples of the 3-class classification environment.

4.2.3. Final performance evaluation formula

A novel aspect of this research will be the final performance evaluation formula presented in this section. The reason to use this formula is the need to unify the results of different classification tasks (binary, 3-class, 5-class), performance evaluation metrics and resampling techniques. A final performance evaluation metric would allow us to decide which statistical and machine learning techniques perform best in predicting individual credit ratings. This formula will include the score from each performance evaluation formula. The example for score calculation for accuracy metric having a binary classification task and performing no resampling technique is presented in Figure 4.



	Accuracy of Binary Classification		
	No Resampling		Score
Logistic Regression	0.75	Scoring formula →	0.25
SVM-Radial	0.70		0
Random Forest	0.85		0.75
XGBoost	0.83		0.65
ANN-2L	0.80		0.50
ANN-3L	0.90		1

Figure 4. The example of score calculation for Accuracy, in binary classification and no resampling applied

The scoring formula is based on "min-max" normalization technique:

$$Accuracy_{Score_{i,j,z}} = \frac{x_{i,j,z} - x_{min,j,z}}{x_{max,j,z} - x_{min,j,z}}.$$

Here i is the algorithm's name, j stands for the type of classification problem ($j \in \{binary,\ 3 - class,\ 5 - class\}$), z is the resampling technique ($z \in \{no\ resampling,$ $Random\ Oversampling,\ SMOTE\}$). This scoring technique allows us to rank the classifiers in a given environment while simultaneously capturing the size of the difference between the performance. If the difference in Accuracy is tiny between the two best algorithms relative to the poorer classifiers' Accuracy, then the difference in score will also be minimal. By summing scores for Accuracy through all classification problems and all resampling techniques, we can obtain the total accuracy score for each of the algorithms:

$$Total\_Accuracy\_Score_i = \sum_j \sum_z Accuracy\_Score_{i,j,z}\ .$$

Such scores will be calculated for other performance evaluation metrics: AUC, Training time, the Riskiest class classification accuracy (1-FPR for the binary classification and E class Recall for the multiclass classification), Recall (Macro Recall for multiclass classification), Precision (Macro Precision for multiclass classification), False Negative rate (only for multiclass classification) and False Positive rate (only for multiclass classification). Having many total scores of different performance evaluation metrics, it is necessary to combine all of them for the final score of the research. However, the final score must adjust each metric's importance to the final evaluation of the classifier. Training time cannot have the same impact on the ranking of classifiers as the False Positive rate does. Additionally, the False Positive rate is more expensive than the False Negative rate. Based on author's expert knowledge of the field of analysis, it was decided to adjust the weights (coefficients) of each performance evaluation metric's total score to the final score function as:

$$\begin{aligned} Final\_Score_i = {} & Total\_Accuracy\_score_i + Total\_AUC\_score_i + 0.1 \\ & * Total\_TrainTime\_score_i\ + 1.5 * Total\_RiskiestClassAccuracy\_score_i \\ & + Total\_Recall\_score_i + Total\_Precision\_score_i\ + 1.5 \\ & * Total\_FalsePositive\_score_i + Total\_FalseNegative\_score_i. \end{aligned}$$

Based on that final score value, the ranking of the classifiers will be made. As it can be seen from the formula, the final score formula adjusts by weighting Training time to be a less impactful factor. Furthermore, all expensive errors (the Riskiest class classification accuracy and False Positive rate) are weighted to be more important factors for the final decision and ranking of the statistical and machine learning techniques.

5. Results

5.1. Empirical Set-Up

To avoid contingency in comparing the performance of statistical and machine learning techniques, the experiments were repeated three times with different sampling into training and testing sets, keeping the 80:20 ratio as presented in the 3.3 section. The full analysis was coded in R studio (R version 4.1.3) using the R statistical software packages like *randomForest*, *e1071*, *xgboost*, *keras and nnet* on the computer system with Windows 11 operating system, 16 GB RAM and AMD Ryzen 5 processor (program code for binary classification is presented in Appendix 4).

The selection of hyperparameters may influence the performance of algorithms. Classifiers SVM-Radial, RF, XGBoost and ANNs will have hyperparameters that need to be determined. Grid search was used as a hyperparameters optimization method, and the search space is presented in Table 9. Other, not introduced parameters are kept as default values as provided in R statistical software packages documentation. It was decided to find optimal parameters for binary and multiclass problems separately to optimize the performance of algorithms as much as possible.

Table 9. Searching space of hyper parameters

Model	Hyperparameters	Symbol	Search Space	Optimal parameter Binary	Optimal parameter Multiclass
SVM-Radial Basis	*Gamma*	γ	*0.001, 0.01, 0.05, 0.1*	*0.05*	*0.01*
	Cost	*C*	*1, 2, 5, 10*	*2*	*1*
Random Forest	*Number of Tree*	*ntree*	*100, 200, 500*	*200*	*200*
	Number of features	*mtry*	*[1;15]*	*5*	*5*
XGBoost	*Maximum tree depth*	*max_depth*	*2, 4, 5, 6, 8*	*6*	*5*
	Learning rate	*eta*	*0.01, 0.1, 0.2, 0.5*	*0.01*	*0.01*
	Column subsample ratio	*colsample_bytree*	*[0;1]*	*0.5*	*0.5*
	Number of Boost	*nrounds*	*100,200,500,1000*	*1000*	*1000*
	Minimum Child Weight	*min_child_weight*	*1, 2, 3, 4*	*2*	*2*

The Radial basis kernel function was selected for the Support Vector Machine and two hyperparameters (Gamma and Cost) needed to be optimally selected. These hyperparameters control the trade-off between the accuracy on the training data and the risk of possible overfitting. For binary classification, it was found that Gamma = 0.05 and Cost = 2 were the values of the most optimal parameters, while for the multiclass, Gamma = 0.01 and Cost = 1 yielded the highest accuracy.

As presented in the Random Forest description (see section 4.1.4), the number of trees and the number of randomly sampled predictors used to build a tree are needed to be set. It was found that

200 trees and five randomly selected features for each tree were the most optimal parameters to build the Random Forest algorithm.

For the XGBoost algorithm, maximum tree depth was optimized at 6 for binary classification and 5 for multiclass classification problems. The learning rate used to prevent overfitting was optimal at 0.01, and the column subsample ratio used in our analysis is set to 0.5 in both binary and multiclass classification problems. The number of boosts is set to be equal to 1000, and minimum child weight was found to be optimal at value 2.

In section 4.1.6, it was mentioned that two ANN architectures will be used in our study. ANN with two and ANN with four hidden layers are implemented to compare how the network's complexity may affect the performance in predicting credit scores. The number of neurons in hidden layers was decided by the grid search, where possible values were 5, 10, 15, 20, and 25. For the Binary classification case, the optimal number of neurons in each hidden layer was found to be 20; for the multiclass case, this number was higher and set to 25. Using a trial and error process, the ReLU was decided over sigmoid as an activation function in hidden layers. Softmax activation function was used in the last (output) layer for multiclass classification, and sigmoid was used in the binary classification case. Adam was used as an optimization algorithm with a learning rate of 0.0001. To prevent overfitting, the network's training procedure used a dropout of value 0.1. The number of epochs was set to 300, and batches with 32 instances at each iteration were used. The results and analysis of the experiments are discussed in the following section.

5.2.1. Accuracy Results

The results of the analysis of statistical and machine learning techniques in predicting individual credit scores start from comparing the performance of the classifiers based on the proportion of correctly classified observations. Table 10 summarizes the results of our experiments, which were repeated three times with respect to the Accuracy metric. The technique achieving the highest Accuracy in a given environment is underlined and bolded.

Table 10. Results of Accuracy measure

	Accuracy								
	Binary Results			**3-Class Results**			**5-Class Results**		
	Normal	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**
LR/MLR	0.754	0.735	0.733	0.735	0.660	0.653	0.571	0.495	0.496
SVM	0.748	0.730	0.741	0.734	0.674	0.662	0.570	0.502	0.496
RF	**<u>0.769</u>**	**<u>0.767</u>**	0.747	**<u>0.750</u>**	**<u>0.749</u>**	**<u>0.695</u>**	**<u>0.601</u>**	**<u>0.598</u>**	**<u>0.542</u>**
XGBoost	0.768	0.750	**<u>0.761</u>**	0.748	0.685	0.690	0.592	0.512	0.534
ANN-2L	0.759	0.757	0.753	0.737	0.686	0.652	0.577	0.498	0.482
ANN-4L	0.756	0.756	0.751	0.737	0.683	0.674	0.572	0.499	0.471

It can be clearly seen that methods based on ensembles of classification trees (RF and XGBoost) dominate with respect to Accuracy in all types of classification problems and through all the resampling techniques. RF performs as the best algorithm in eight out of nine classification experiments. The highest Accuracy of 76.9% is achieved in binary classification problem and using a not-resampled dataset for model training. Alternatively, in the case of a 5-class classification problem and using SMOTE resampling technique for the training dataset, the best performing classifier reaches only 54.2% of Accuracy. It is necessary to highlight that our testing set faces a class imbalance problem to test the performance of the algorithms in a real-life scenario. Therefore, the Accuracy measure may not be the most correct way of measuring the model's performance since some classifiers may naively predict the majority class and provide high Accuracy. Comparing two resample techniques, namely Random Oversampling (ROS) against SMOTE technique, it can be included that the ROS technique helps classifiers to reach higher Accuracy than using SMOTE. Additionally, by increasing the dimension of the classification problem (dimension of the target variable), the Accuracy decreases, meaning that binary classification is the easiest for the algorithms, while 5-class classification is the most difficult.

Table 10 provides the scoring table (based on the score function described in the 4.2.3 section) to evaluate and rank statistical and machine learning techniques.

Table 11. Score table from Accuracy measure

	Score from Accuracy										
	Binary Results			3-Class Results			5-Class Results				
	Normal	ROS	SMOTE	Normal	ROS	SMOTE	Normal	ROS	SMOTE	Total Score	Rank
LR/MLR	0.26	0.13	0.00	0.08	0.00	0.02	0.06	0.00	0.35	0.89	6
SVM	0.00	0.00	0.30	0.00	0.15	0.23	0.00	0.07	0.35	1.10	5
RF	**1.00**	**1.00**	0.50	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	8.50	1
XGBoost	0.98	0.55	1.00	0.85	0.27	0.89	0.72	0.17	0.89	6.33	2
ANN-2L	0.51	0.74	0.73	0.16	0.29	0.00	0.25	0.03	0.16	2.87	3
ANN-4L	0.39	0.71	0.64	0.16	0.26	0.51	0.07	0.04	0.00	2.79	4

It can be found that the Total Accuracy Score formula approves the findings from Table 9, that RF and XGBoost are the best and the second-best algorithms with respect to Accuracy. ANNs techniques perform slightly worse (both architectures), which could have been expected based on the extensive literature review. LR/MLR and SVM are associated with the poorest performance of all the algorithms used in this study. The main idea of our scoring function was the ability to capture the gaps between the algorithms. Based on Table 10, the gap between ensemble techniques and all other algorithms is significant. The highest gaps can be spotted in multiclass classification cases, where MLR, SVM and ANNs received a score from accuracy higher than 0.5 only once. The result shows that ensemble techniques outperform all other algorithms (especially in multiclass classification problems) in terms of Accuracy. However, an extensive comparison between many performance evaluation metrics is necessary for the more robust evaluation and ranking of statistical and machine learning techniques in predicting individual credit scores.

5.2.2. AUC Results

Similar to Table 10, Table 12 presents the experimental results using the AUC performance evaluation metric.

Table 12. Results of AUC measure

	AUC								
	Binary Results			**3-Class Results**			**5-Class Results**		
	Normal	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**
LR/MLR	0.812	0.812	0.807	0.839	0.866	0.862	0.831	0.845	0.843
SVM	0.797	0.797	0.794	0.764	0.818	0.810	0.775	0.830	0.822
RF	**0.829**	0.826	0.825	**0.861**	0.872	**0.878**	**0.858**	0.865	**0.865**
XGBoost	0.828	**0.827**	**0.826**	0.859	**0.881**	0.873	0.854	**0.867**	0.862
ANN-2L	0.812	0.812	0.800	0.846	0.867	0.865	0.842	0.849	0.852
ANN-4L	0.811	0.810	0.796	0.845	0.856	0.860	0.833	0.838	0.844

Based on AUC results, RF and XGBoost outperform their competitor algorithms and strengthen the position of ensembles of classification trees. Differently than in the Accuracy results, it can be seen that AUC in the multiclass classification case is not smaller than the AUC in binary classification results. Additionally, XGBoost dominates when the Random Oversampling technique is applied, and RF dominates in the case of the Normal training dataset. In every multiclass classification experiment, the resampling techniques (ROS and SMOTE) are associated with higher AUC than in the case of the Normal training dataset. However, in the binary classification case, we cannot observe such differences, and the results of AUC are considerably similar through all the resampling scenarios. Table 13 presents the scores obtained from the AUC measure to compare the difference across classification techniques better.

Table 13. Score table from AUC metric

	Score from AUC										
	Binary Results			**3-Class Results**			**5-Class Results**				
	Normal	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**	**Total Score**	**Rank**
LR/MLR	0.46	0.49	0.41	0.77	0.76	0.75	0.67	0.41	0.47	5.20	4
SVM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6
RF	**1.00**	0.97	0.98	**1.00**	0.86	**1.00**	**1.00**	0.95	**1.00**	8.75	1
XGBoost	0.96	**1.00**	**1.00**	0.97	**1.00**	0.92	0.95	**1.00**	0.92	8.73	2
ANN-2L	0.48	0.50	0.19	0.85	0.76	0.81	0.80	0.51	0.69	5.61	3
ANN-4L	0.45	0.43	0.05	0.83	0.60	0.72	0.70	0.22	0.50	4.52	5

The results from Table 13 show that RF and XGBoost dominate through all the experiments with respect to the AUC measure. Nevertheless, the difference between the two ensembles is minimal. LR and both ANNs architectures performed similarly, as seen in the "Total Score" value. Interestingly, SVM is the poorest classification algorithm in all the experiments based on the AUC performance evaluation metric. In the case of Accuracy, we obtained the results where ensemble methods dominated even more in the case of multiclass classification. With respect to the AUC measure, such dominance cannot be found.

5.2.3. Training Time Results

Training time is another dimension that needs to be evaluated and compared in analyzing statistical and machine learning techniques in predicting individual credit scores. The rapid algorithm should have an advantage over another, slower classifier if the prediction rates are similar. Table 14 presents the Training time in minutes of the algorithms used in our study.

Table 14. Results of Training Time

	Training time (in minutes)								
	Binary Results			**3-Class Results**			**5-Class Results**		
	Normal	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**
LR/MLR	**0.016**	**0.026**	**0.012**	**0.044**	**0.083**	**0.064**	**0.099**	**0.278**	**0.152**
SVM	9.570	10.483	7.508	2.142	8.130	1.750	3.414	14.360	3.983
RF	3.080	2.588	1.972	0.289	0.600	0.243	0.285	0.778	0.339
XGboost	1.310	0.619	0.690	0.432	0.578	0.405	0.689	1.099	0.770
ANN-2L	6.696	8.263	8.014	7.500	15.610	6.412	8.445	17.570	10.913
ANN-4L	7.133	10.638	9.097	8.346	16.573	7.054	9.087	18.533	11.267

LR is the fastest algorithm in our study that runs in seconds. On the other hand, the more sophisticated classifiers like SVM and ANNs train significantly longer, which is an important aspect when comparing those classifiers. It is also noted that the most computationally expensive experiments were ANNs in the multiclass classification case and using resampling techniques. The duration of training time reached 18 minutes, while faster algorithms like LR, RF, and XGBoost trained within one minute. Such a result gives us a dimension of the algorithm's rapidness in analyzing statistical and machine learning techniques. The scores to classifiers from Training time are presented in Table 15.

Table 15. Score table from Training Time

	Score from Training Time										
	Binary Results			**3-Class Results**			**5-Class Results**				
	Normal	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**	**Total Score**	**Rank**
LR/MLR	**<u>1.00</u>**	**<u>1.00</u>**	**<u>1.00</u>**	**<u>1.00</u>**	**<u>1.00</u>**	**<u>1.00</u>**	**<u>1.00</u>**	**<u>1.00</u>**	**<u>1.00</u>**	9.00	1
SVM	0.00	0.01	0.17	0.75	0.51	0.76	0.63	0.23	0.66	3.72	4
RF	0.68	0.76	0.78	0.97	0.97	0.97	0.98	0.97	0.98	8.07	3
XGboost	0.86	0.94	0.93	0.95	0.97	0.95	0.93	0.96	0.94	8.44	2
ANN-2L	0.30	0.22	0.12	0.10	0.06	0.09	0.07	0.05	0.03	1.05	5
ANN-4L	0.26	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.26	6

Logistic regression earns maximum points of Total Score from the Training time because it is the fastest algorithm used in our study. Interestingly, ensembles of classification trees are also comparatively fast, which shows that RF and XGBoost in predicting credit scores are impressive, combined with the previous results of Accuracy and AUC. ANNs are the most computationally expensive techniques in our analysis, especially with four hidden layers executing exceptionally slowly. However, the Training time cannot be equally important as the predictive power of the classifier. For that reason, the weights for each measure of predictive performance are adjusted based on author's expert knowledge as presented in section 4.2.3.

5.2.4. The Riskiest Class Classification Results

The Riskiest class classification is arguably the most crucial aspect of credit scoring. Misclassifying the riskiest segment of people as reliable may lead to substantial losses for the lender. For that reason, the Riskiest class classification accuracy is excluded as a separate performance evaluation metric. For the binary classification case, it is simply 1-False Positive rate (1-FPR), and for the multiclass classification case, it is equivalent to the Recall of the Riskiest (E) class. Table 16 presents the classification results based on the accuracy of the Riskiest class segment.

Table 16. Results of the Riskiest class classification accuracy

	(1-FPR) for Binary case and the riskiest class (E) RECALL for Multiclass case								
	Binary Results			**3-Class Results**			**5-Class Results**		
	Normal	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**
LR/MLR	0.513	**<u>0.705</u>**	0.694	0.420	**<u>0.822</u>**	0.804	0.466	0.695	0.678
SVM	0.441	0.683	0.622	0.372	0.789	0.757	0.429	0.682	0.678
RF	0.559	0.574	**<u>0.709</u>**	0.460	0.529	0.661	**<u>0.538</u>**	0.582	0.675
XGBoost	**<u>0.566</u>**	0.702	0.644	**<u>0.472</u>**	0.804	0.658	0.519	**<u>0.742</u>**	0.683
ANN-2L	0.550	0.569	0.614	0.465	0.660	0.779	0.513	0.335	**<u>0.716</u>**
ANN-4L	0.543	0.526	0.596	0.465	0.575	**<u>0.871</u>**	0.521	0.328	0.704

The results of this performance evaluation metric differ almost in every experiment performed in our analysis. Additionally, it is hard to compare the results between binary classification and multiclass classification, since the riskiest class differs in both of these environments.

In binary classification case, the RF algorithm using SMOTE resampling techniques achieves the highest value of 1-FPR at 0.709. It can be clearly seen that both ROS and SMOTE resampling techniques significantly improve the 1-FPR rate, except for ANNs case. ANNs perform impressively when no resampling techniques are applied, however after the resampling, other classifiers improved by more.

In multiclass classification case, the similar trends are also observed. The ANN with four hidden layers provides the highest Recall of E class using 3-Class classification problem and applying SMOTE resampling technique at the value of 0.871. In the case of 5-Class classification problem, the XGBoost algorithm with ROS resampling technique achieves 0.742 of Recall of E class. The results of the Riskiest class classification accuracy shows that evaluation and comparison of a performance evaluation metric through all the experiments of our study could be hardly interpretable. For that reason, the scores formula allows us to compare the performance of all statistical and machine learning techniques more easily, by taking into account all different classification problems and different resampling techniques.

Table 17. Score table from the riskiest class accuracy metric

	Score from the Riskiest class accuracy										
	Binary Results			**3-Class Results**			**5-Class Results**				
	Normal	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**	**Total Score**	**Rank**
LR/MLR	0.58	**1.00**	0.86	0.48	**1.00**	0.69	0.34	0.89	0.07	5.90	2
SVM	0.00	0.88	0.23	0.00	0.89	0.47	0.00	0.85	0.07	3.38	6
RF	0.94	0.27	**1.00**	0.88	0.00	0.01	**1.00**	0.61	0.00	4.71	4
XGBoost	**1.00**	0.98	0.43	**1.00**	0.94	0.00	0.82	**1.00**	0.20	6.37	1
ANN-2L	0.87	0.24	0.17	0.93	0.45	0.57	0.77	0.02	**1.00**	5.01	3
ANN-4L	0.81	0.00	0.00	0.93	0.15	**1.00**	0.85	0.00	0.70	4.45	5

Table 17 allows us to compare and rank the classifiers with respect to the Riskiest class accuracy. Based on the Total Score value, differences across the algorithms are not very high. However, the XGBoost algorithm is the best performing algorithm on average through all the experiments performed. Interestingly, XGBoost with the SMOTE resampling technique performs relatively poorly in all the classification problems. In addition, RF is also not impressive in multiclass classification problems using SMOTE. Based on such results, ensembles of classification trees perform poorly with respect to E-class Recall using SMOTE. On the other hand, ANNs perform comparatively much better when SMOTE is applied in the multiclass classification case.

5.2.5. Recall Results

In this section results from Recall (for binary classification) and Macro Recall (for multiclass classification) are compared.

Table 18. Results from the Recall metric

	Recall for binary case and Macro Recall for multiclass case								
	Binary Results			3-Class Results			5-Class Results		
	Normal	ROS	SMOTE	Normal	ROS	SMOTE	Normal	ROS	SMOTE
LR/MLR	0.886	0.751	0.755	0.587	0.690	0.682	0.448	0.547	0.542
SVM	**0.918**	0.757	0.805	0.562	0.694	0.680	0.423	0.559	0.551
RF	0.885	0.873	0.768	0.615	0.650	0.684	**0.498**	0.542	**0.582**
XGBoost	0.880	0.777	0.825	**0.618**	**0.710**	0.678	0.488	**0.586**	0.567
ANN-2L	0.874	0.861	0.830	0.605	0.675	0.683	0.468	0.535	0.564
ANN-4L	0.874	**0.883**	**0.836**	0.602	0.653	**0.720**	0.456	0.526	0.553

Based on Table 18, SVM is the best performing algorithm in the case of binary classification when no resampling technique was applied. However, that result must be considered with caution. SVM in that experiment provided the poorest 1-FPR results, which informs us that it was highly biased towards the majority class, which showed up in a high Recall of 0.918. In all other experiments, Neural network architectures and ensemble methods provide the best results. ANN with four hidden layers combined with ROS method reaches 0.883 of Recall in binary classification, and with SMOTE method in 3-Class classification case reaches Recall equal to 0.720. In 5-Class classification, ensemble methods dominate the other techniques, with XGBoost reaching 0.586 Recall when ROS was applied. Table 19 presents the scores obtained using Recall as the performance evaluation metric.

Table 19. Score table from Recall metric

	Score from Recall										
	Binary Classification			3-Class Results			5-Class Results				
	Normal	ROS	SMOTE	Normal	ROS	SMOTE	Normal	ROS	SMOTE	Total Score	Rank
LR/MLR	0.28	0.00	0.00	0.44	0.67	0.09	0.33	0.35	0.00	2.17	6
SVM	**1.00**	0.04	0.62	0.00	0.73	0.06	0.00	0.55	0.22	3.21	5
RF	0.25	0.92	0.16	0.93	0.00	0.15	**1.00**	0.28	**1.00**	4.68	2
XGBoost	0.15	0.19	0.86	**1.00**	**1.00**	0.00	0.86	**1.00**	0.61	5.67	1
ANN-2L	0.00	0.83	0.92	0.76	0.42	0.11	0.59	0.15	0.56	4.35	4
ANN-4L	0.01	**1.00**	**1.00**	0.71	0.05	**1.00**	0.43	0.00	0.27	4.47	3

Based on the results from the previous table, one more time, XGBoost reaches the highest total score from the Recall metric. Additionally, the Random Forest and both ANNs architectures provide very similar results, while LR/MLR and SVM performed not impressively. These results approve the idea of the scoring formula. The difference between RF and ANNs architecture is minimal. Thus, the results from the Recall metric will contribute to the final ranking with a similar scores regarding RF and ANNs.

5.2.6. Precision Results

This section presents the last performance evaluation metric, where binary classification case was included. Table 20 illustrates the Precision (for binary classification) and Macro Precision (for multiclass classification) results.

Table 20. Results from Precision metric

	Precision for binary case and Macro Precision for multiclass case								
	Binary Results			**3-Class Results**			**5-Class Results**		
	Normal	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**
LR/MLR	0.767	0.822	0.817	0.740	0.540	0.535	0.594	0.428	0.428
SVM	0.748	0.812	0.799	0.755	0.556	0.549	0.643	0.444	0.437
RF	0.784	0.788	**0.827**	**0.782**	**0.752**	**0.620**	**0.656**	**0.627**	**0.502**
XGBoost	**0.786**	**0.825**	0.807	0.758	0.572	0.617	0.633	0.462	0.494
ANN-2L	0.778	0.783	0.796	0.740	0.600	0.543	0.601	0.557	0.430
ANN-4L	0.776	0.771	0.789	0.733	0.625	0.610	0.597	0.568	0.420

Similar to the cases of Accuracy and AUC, we can see a dominance of the ensemble methods. RF performed with the highest rate of Precision in all the multiclass classification experiments and binary classification when SMOTE was applied. In Table 21, it can be seen how dominant the RF was compared with other statistical and machine learning techniques. None of the LR/MLR, SVM and ANNs obtained a total score of at least 3. That shows that RF was performing significantly better than the competitor algorithms with respect to Precision and Macro Precision.

Table 21. Scores table from Precision metric

	Score from Precision										
	Binary Results			**3-Class Results**			**5-Class Results**				
	Normal	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**	**Total Score**	**Rank**
LR/MLR	0.51	0.93	0.73	0.14	0.00	0.00	0.00	0.00	0.10	2.42	6
SVM	0.00	0.75	0.26	0.46	0.07	0.17	0.80	0.08	0.20	2.79	3
RF	0.95	0.30	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	8.25	1
XGboost	**1.00**	**1.00**	0.49	0.51	0.15	0.96	0.63	0.17	0.91	5.82	2
ANN-2L	0.81	0.22	0.17	0.15	0.28	0.09	0.11	0.65	0.13	2.61	5
ANN-4L	0.73	0.00	0.00	0.00	0.40	0.88	0.06	0.70	0.00	2.77	4

5.2.7. False Positive Rate Results

A False Positive rate is one of the most crucial performance evaluation metrics for the multiclass classification problem. The importance of this measure can be compared only with the

E class Recall metric. The False Positive rate for the multiclass type of problem shows what proportion of all the observations was classified as more reliable than they actually are. Table 22 presents the results obtained by using this performance evaluation metric only for the multiclass classification experiments because FPR for binary classification was already used in the comparison of the Riskiest class accuracy.

Table 22. Results of false positive rate for the multiclass classification

	False Positive Rate only for the multiclass case					
	3-Class Results			**5-Class Results**		
	Normal	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**
MLR	0.197	0.095	0.095	0.275	**0.233**	**0.238**
SVM	0.217	0.097	0.095	0.290	0.255	0.254
RF	0.183	0.151	0.091	**0.250**	0.249	0.246
XGBoost	**0.178**	**0.090**	0.095	0.262	0.255	0.254
ANN-2L	0.189	0.096	**0.087**	0.267	0.285	0.252
ANN-4L	0.193	0.097	0.089	0.267	0.291	0.248

In the 3-Class classification case, resampling techniques ROS and SMOTE significantly improve the results of classifiers with respect to the False Positive rate. However, in the 5-Class classification case, the False Positive rate is not changing much when resampling techniques are applied. ANN with two hidden layers reaches the best False Positive rate of 0.087 in the 3-Class classification case when SMOTE was applied. In the 5-Class classification case, applying the MLR with ROS technique shows the best results, where the False Positive rate reaches 0.233. Nevertheless, the constantly best-performing algorithm cannot be seen in Table 22. For that reason, it is necessary to check what are the scores from the False Positive rate, shown in Table 23.

Table 23. Scores table from the False Positive rate metric

	Score from the False Positive Rate							
	3-Class Results			**5-Class Results**				
	Normal	**ROS**	**SMOTE**	**Normal**	**ROS**	**SMOTE**	**Total Score**	**Rank**
MLR	0.53	0.92	0.03	0.37	**1.00**	**1.00**	3.85	1
SVM	0.00	0.89	0.02	0.00	0.62	0.04	1.57	6
RF	0.88	0.00	0.45	**1.00**	0.72	0.49	3.54	2
XGBoost	**1.00**	**1.00**	0.00	0.72	0.63	0.00	3.35	4
ANN-2L	0.73	0.91	**1.00**	0.57	0.10	0.14	3.45	3
ANN-4L	0.61	0.90	0.76	0.57	0.00	0.37	3.21	5

The results from Table 23 suggest that only SVM is not competitive with respect to the False Positive rate in the multiclass classification case. However, all other techniques are associated with quite a similar total scores of False Positive rate across all the experiments. MLR is ranked first, but the total sum of scores for the fifth place is only lower by 0.64. Meaning that none of these algorithms would get an advantage over the other in the final score calculation based on the False Positive results.

5.2.8. False Negative Rate Results

Unlike in the previous section, the False Negative rate shows what percentage of all observations were predicted as riskier than they actually are. The full results of the False Negative rate are presented in Table 24.

Table 24. Results of False Negative rate

	False Negative rate only for multiclass case					
	3-Class Results			5-Class Results		
	Normal	ROS	SMOTE	Normal	ROS	SMOTE
MLR	0.068	0.244	0.253	0.153	0.272	0.266
SVM	**0.049**	0.229	0.243	**0.140**	0.243	0.250
RF	0.067	**0.100**	**0.214**	0.149	**0.153**	**0.212**
XGBoost	0.074	0.225	0.214	0.146	0.233	0.212
ANN-2L	0.074	0.218	0.261	0.155	0.217	0.266
ANN-4L	0.070	0.220	0.238	0.161	0.210	0.281

The SVM algorithm is associated with the lowest False Negative rate in multiclass classification problems when no resampling technique was applied. Having applied ROS and SMOTE techniques, RF dominates over other algorithms in each experiment. Interestingly, the differences in False Negative rates across the 5-Class and 3-Class classification results are minimal. In the case of Accuracy and False Positive rate, we faced a situation when the increasing dimension of the target variable was associated with the decreased performance of classifiers. However, a constant False Negative rate in both multiclass classification problems shows that worse performance in the 5-class classification case was mainly driven by the increasing False Positive rate. Such a situation is hazardous in the credit scoring field and may imply high losses for the credit company. Table 25 presents the scores obtained from the False Negative rate.

Table 25. Scores table of the False Negative rate metric

	Score from the False Negative Rate							
	3-Class Results			5-Class Results				
	Normal	ROS	SMOTE	Normal	ROS	SMOTE	Total Score	Rank
MLR	0.25	0.00	0.19	0.36	0.00	0.21	1.01	6
SVM	**1.00**	0.10	0.38	**1.00**	0.25	0.44	3.17	3
RF	0.29	**1.00**	**1.00**	0.60	**1.00**	**1.00**	4.88	1
XGBoost	0.03	0.13	0.98	0.71	0.32	1.00	3.17	2
ANN-2L	0.00	0.18	0.00	0.28	0.46	0.21	1.13	5
ANN-4L	0.18	0.17	0.50	0.00	0.52	0.00	1.36	4

Ensembles of classification trees are performing quite impressive in multiclass classification cases with respect to the False Negative rate. However, the SVM received an equal amount of score as the XGBoost algorithm from all the experiments. But the main dominance of the SVM algorithm was observed only in the case when no resampling techniques were applied.

5.2.9. Final Scores

Having analysed all the results from distinct performance evaluation metrics, it is necessary to combine these results and see what final conclusions can be drawn. The final score formula summarizes all the distinct (total) scores obtained from different classification evaluation metrics. As already presented, some scores must be weighted based on the author's expert knowledge. Training time is weighted with a coefficient of 0.1 since it is not the most critical aspect of credit scoring. In contrast, the Riskiest class classification accuracy (1-FPR for the binary case and the E class Recall for the multiclass case) and False Positive rate are weighted with a coefficient of 1.5 because these rates are the most crucial in the credit scoring field. The final scores and ranking are presented in Table 26. Each column represents the total scores obtained from that performance evaluation metric across all the experiments. The last two columns show the final score of the classifier and the final rank.

Table 26. Final Scores table

	Total Scores									
	Accuracy	**AUC**	**Training time (0.1 coef.)**	**Accuracy of the riskiest class (1.5 coef.)**	**Recall**	**Precision**	**False positive rate (1.5 coef.)**	**False Negative rate**	**FINAL SCORE of Classifier**	**RANK of Classifier**
LR/MLR	0.89	5.20	9.00	5.90	2.17	2.42	3.85	1.01	**27.21**	**5**
SVM	1.10	0.00	3.72	3.38	3.21	2.79	1.57	3.17	**18.07**	**6**
RF	8.50	8.75	8.07	4.71	4.68	8.25	3.54	4.88	**48.27**	**1**
XGBoost	6.33	8.73	8.44	6.37	5.67	5.82	3.35	3.17	**45.15**	**2**
ANN-2L	2.87	5.61	1.05	5.01	4.35	2.61	3.45	1.13	**29.37**	**3**
ANN-4L	2.79	4.52	0.26	4.45	4.47	2.77	3.21	1.36	**27.43**	**4**

As seen in the analysis of different performance evaluation metrics, the ensembles of classification trees dominate in predicting individual credit ratings across all the algorithms in our study. RF collected just a few points more than XGBoost, showing that both ensembles are highly capable of predicting the riskiness of customers using a Lithuanian credit dataset. In addition, the ANN with two layers stays at the third position, showing that the neural network of simpler architecture performs relatively better than the more sophisticated neural network with four hidden layers. LR/MLR performs similarly to ANN with four layers, which approves the idea of Baesens et al. (2003) that credit scoring datasets are only weakly non-linear. Lastly, SVM with the Radial basis function produced the poorest results across all the statistical and machine learning techniques used in our study and all the experiments. Having found that RF and XGBoost algorithms performed as the best classifiers, we can take a deeper look into the performance of only these algorithms. To make an easier comparison of these two algorithms, only four performance evaluation metrics are used in Table 27. Based on the author's expert knowledge, the four most representative metrics are Accuracy, AUC, the Riskiest class classification accuracy and the False Positive rate (for multiclass classification only).

Additionally, the average values ("avg." in Table 27) are included in columns and rows for each classification problem and each performance evaluation metric. In the column, the average shows how the given algorithm performed across all the resampling techniques. Rows present the average performance of both algorithms for a given resampling technique. Based on Table 27, we can compare how the two best performing algorithms in our study perform with respect to four of the most crucial performance evaluation metrics. Additionally, it could be evaluated how the performance differs when resampling techniques are applied and when different classification problems are of interest. The best value with respect to a given performance evaluation metric across all classification problems and resampling techniques is bolded and underlined.

Table 27. Performance of RF and XGBoost across all the experiments

	Binary Classification				**3-Class classification**				**5-Class Classification**			
	Normal	**ROS**	**SMOTE**	*avg.*	**Normal**	**ROS**	**SMOTE**	*avg.*	**Normal**	**ROS**	**SMOTE**	*avg.*
Accuracy												
RF	**<u>0.769</u>**	0.767	0.747	*0.761*	0.750	0.749	0.695	*0.731*	0.601	0.598	0.542	*0.580*
XGboost	0.768	0.750	0.761	*0.760*	0.748	0.685	0.690	*0.708*	0.592	0.512	0.534	*0.546*
avg.	*0.769*	*0.758*	*0.754*		*0.749*	*0.717*	*0.693*		*0.597*	*0.555*	*0.538*	
AUC												
RF	0.829	0.826	0.825	*0.827*	0.861	0.872	0.878	*0.871*	0.858	0.865	0.865	*0.863*
XGboost	0.828	0.827	0.826	*0.827*	0.859	**<u>0.881</u>**	0.873	*0.871*	0.854	0.867	0.862	*0.861*
avg.	*0.828*	*0.827*	*0.826*		*0.860*	*0.877*	*0.876*		*0.856*	*0.866*	*0.863*	
The riskiest class classification accuracy												
RF	0.559	0.574	0.709	*0.614*	0.460	0.529	0.661	*0.550*	0.538	0.582	0.675	*0.598*
XGboost	0.566	0.702	0.644	*0.637*	0.472	**<u>0.804</u>**	0.658	*0.645*	0.519	0.742	0.683	*0.648*
avg.	*0.562*	*0.638*	*0.677*		*0.466*	*0.667*	*0.660*		*0.528*	*0.662*	*0.679*	
False Positive rate												
RF	-	-	-		0.183	0.151	0.091	*0.142*	0.250	0.249	0.246	*0.249*
XGboost	-	-	-		0.178	**<u>0.090</u>**	0.095	*0.121*	0.262	0.255	0.254	*0.257*
avg.					*0.181*	*0.121*	*0.093*		*0.256*	*0.252*	*0.250*	

Having such a broad spectrum of results, allows us to compare the advantages of each of the algorithms in terms of various types of classification problems and across different resampling techniques.

Regarding Accuracy, we can see a constant trend across all classification problems, where RF with no resampling technique reaches the highest Accuracy. In binary classification, Accuracy reached 0.769, in 3-Class classification – 0.750 and in 5-Class classification – 0.601. Using ROS resampling technique, the Accuracy rate decreases just by hundredths, suggesting that ROS may also be competitive to the case when no resampling techniques were applied. The Accuracy may decrease when the target variable's dimension increases and the classification problem becomes more difficult. Nevertheless, the decrease in Accuracy is minimal in the case of 3-class classification, comparing it with the binary classification case. On the other hand, the Accuracy decreased drastically when 5-class classification results were compared with the 3-class classification case results. Such findings suggest that the 5-class classification problem is much harder for the RF and XGBoost, and there is no clear distinction between the five classes. Using the 3-Class classification, the customers can be classified into three risk categories with an Accuracy similar to the binary classification case and 15% higher Accuracy than in the 5-class classification.

Next, Table 27 presents the classification results with respect to the AUC value. AUC uses the True Positive and False Positive rates across different thresholds, what should be of interest when classification algorithms are evaluated, besides the Accuracy rate. The RF and XGBoost algorithms perform similarly across all the experiments with respect to the AUC measure. In binary classification, the RF with no resampling technique reaches the highest AUC of 0.829. However, in the multiclass classification case, the best performances can be obtained when applying the ROS technique. The XGBoost combined with the ROS achieves the highest AUC of 0.881 in the 3-class classification problem and 0.866 in the 5-class classification case. Having such results, 3-class classification problem stands out as the most promising classification problem using the newly obtained Lithuanian credit dataset.

It is highly important to test the algorithm’s performance in classifying the Riskiest class. For binary classification, 1-FPR is used as a measure, while the riskiest class (E) Recall is used to evaluate the model’s performance in the multiclass classification case. In the binary classification case, the RF achieves a 0.709 rate of Accuracy of the Riskiest class classification (1-FPR) with SMOTE technique applied. Nevertheless, the XGBoost algorithm achieves an impressive 0.804 Accuracy of the Riskiest class classification (the Recall of the riskiest E class) in a 3-class classification problem when ROS resampling technique was used. Similarly, in the 5-class classification case, the XGBoost with ROS technique reaches 0.742 Accuracy in predicting the riskiest class. These results one more time show that 3-class classification problem, combined with the ROS technique, allows our two best algorithms to reach the best possible results.

Lastly, the False Positive rate for only multiclass classification problems is included in our comparison. Here, drastically different results across the classification problems are obtained, where in the 3-class classification case, XGBoost with ROS technique applied minimizes the False

Positive rate to only 0.09. In contrast, the best performing algorithm in the 5-class classification problem Random Forest with SMOTE is associated with a False Positive rate of 0.246. Such results additionally shows that 5-Class classification problem is much more difficult to the algorithms and no clear distinction between all five classes cannot be found.

6. Conclusions and Recommendations

A proper credit risk management system could help lenders to make more informed decisions in lending processes and avoid the risk of default. This study comprehensively analyzed statistical and machine learning techniques in predicting individual credit scores. To carefully evaluate the performance of the algorithms, the analysis was done using three different classification problems: binary, 3-class and 5-class. Additionally, it was crucial to check whether the classifiers were not biased towards a specific group of predictions (due to the class imbalance problem). To avoid that, resampling techniques ROS and SMOTE were used across the experiments, together with a wide variety of performance evaluation techniques. The analysis was done using the newly introduced Lithuanian credit dataset obtained from one Lithuanian Loan-comparison platform.

The general conclusions of the research carried out are presented below.

- Based on the results obtained in this work, the final scoring method was proposed to combine the results of different classification problems, resampling techniques and various performance evaluation metrics. The proposed scoring technique allows to rank the classifiers, while simultaneously capturing the size of the difference between the performance. If the difference in predictive performance between the two best algorithms is relatively small, then the difference in the final score will also be minimal.
- **The Random Forest and XGBoost were found to be the best performing techniques in predicting individual credit scores based on the proposed final scoring method**. The difference between the predictive performance of ensembles of classification trees and other algorithms was significant. Random Forest outperforms its counterpart when the Accuracy measure is of interest to compare the classifiers. However, the XGBoost shows predictive superiority with respect to the Recall of the Riskiest E class. The Recall of the Riskiest class can be considered as the classification Accuracy of the Riskiest class, which is one of the most crucial elements in the credit scoring field.
- **Empirical results show that the best algorithms perform comparatively better in 3-Class classification problem than in binary classification case**. The Accuracy in binary classification using Random Forest and no resampling technique was 0.769, compared to the same algorithm's performance in the 3-class classification problem, where Accuracy was 0.750. The difference of Accuracy between these experiments is minimal. However, XGBoost with ROS technique applied in 3-Class classification case reaches impressive classification Accuracy of the Riskiest class equal to 0.804. In contrast, the same algorithm in binary classification problem with the ROS technique applied classifies the Riskiest class with an Accuracy of only 0.702. In such a situation, where the Accuracy differs only

by the hundredths, but the classification Accuracy of the Riskiest class is higher by 0.102, it is concluded that the 3-class classification problem is more effective. Figure 2 in section 3.2 shows that the Approval rate for the E-class customers is only 25%. Combining the results of Approval rate and the classification Accuracy for the Riskiest class, it can be concluded that 3-Class classification helps to distinguish the riskiest customers, who rarely receive a loan offer, impressively well.

- It was also concluded that **resampling techniques help to predict the minority (risky) class better.** In the case of the 3-class classification problem, the XGBoost algorithm predicts the Riskiest E class with an Accuracy of 0.804 when ROS technique was applied. In comparison, when no resampling techniques were applied, the same algorithm reached only 0.472 Accuracy of the Riskiest class classification rate. This result demonstrates how severe the class imbalance problem is in the credit scoring field and that Random Oversampling technique helps to significantly improve the performance in classifying the Riskiest segment.

The study used the newly introduced credit dataset from one Lithuanian Loan-comparison platform. Having analysed the performance of statistical and machine learning techniques in predicting individual credit scores, the recommendations for applying the credit scoring system in the Loan-comparison platform can be presented. Firstly, it was found that Random Forest and XGBoost techniques should both be considered in the implementation. When the predictions from these algorithms contradict each other, an expert evaluation must be done. Additionally, 3-class classification with the Random Oversampling technique is highly recommended. The implementation of an individual credit scoring system would allow the Lithuanian Loan-comparison platform to reduce the possibility of lending to a risky customer, improve the efficiency of credit granting processes and avoid additional expenses for external credit scoring agency.

Future research in that field must pay more attention to the development of Neural Network architectures. It is possible to convert the tabular data into the picture format and perform Convolutional Neural Network analysis, which is popular in various research fields. Additionally, it would be recommended to perform analysis using more resampling techniques since the improvements in predictive performance by using Random Oversampling are clear. Nevertheless, extra feature variables would allow us to investigate additional aspects of the customer that could hide some valuable information about his riskiness. Last but not least, the implementation of the results from this study will be performed in a Lithuanian Loan-comparison platform, which would allow the author to investigate how successfully these algorithms can contribute to the performance of that company in real-life scenarios.

References

1. A. Ampountolas, T.J. Nde, P. Date, C. Constantinescu. A Machine Learning Approach for Micro-Credit Scoring, *Risks 9:50*, 2021, p.p. 1-20.
2. B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens, and J. Vanthienen. Benchmarking state-of-the-art classification algorithms for credit scoring, *Journal of the Operational Research Society*, 2003, p.p. 627-635.
3. F. Barboza, H. Kimura, E. Altman. Machine Learning Models and Bankruptcy Prediction, *Expert Systems with Applications*, 2017, Volume (83), p.p. 405-417.
4. T. Bellotti, J. Crook. Support vector machines for credit scoring and discovery of significant features, *Expert Systems with Applications*, 2009, Volume (36), p.p. 3302-3308.
5. C.M. Bishop. *Neural Networks for Pattern Recognition*, Oxford, Clarendon Press, 1995, 498 p.
6. B.E. Boser, I.M. Guyon, V.N. Vapnik. A Training Algorithm for Optimal Margin Classifiers, *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, 1992, p.p. 144-152.
7. L. Breiman. Random Forests, *Machine Learning*, 2001, Volume (45), p.p. 5-32.
8. I. Brown, C. Mues. An experimental comparison of classification algorithms for imbalanced credit scoring data sets, *Expert Systems with Applications*, 2012, Volume (39), p.p. 3446-3453.
9. A. Cao, H. He, Z. Chen, W. Zhang. Performance Evaluation of Machine Learning Approaches for Credit Scoring, *International Journal of Economics, Finance and Management Sciences*, 2018, 6(6), p.p. 255-260.
10. N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique, *Journal of Artificial Intelligence Research*, 2002, Volume (16), p.p. 321-357.
11. T. Chen, C. Guestrin. XGBoost: A Scalable Tree Boosting System, *KDD'16:Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, p.p. 785-794.
12. C. Cortes, V. Vapnik. Support-Vector Networks, *Machine Learning*, 1995, Volume (20), p.p. 273-297.
13. X. Dastile, T. Celik. Making Deep Learning-Based Predictions for Credit Scoring Explainable, *IEEE Access*, 2021, Volume (9), p.p. 50426-50440.
14. P. Golbayani, I. Florescu, R. Chatterjee. A Comparative study of forecasting corporate credit ratings using neural networks, support vector machines and decision trees, *North American Journal of Economics and Finance*, 2020, Volume (54), p.p. 1-16.

15. B.J. Gunnarsson, S. vanden Broucke, B. Baesens, M. Oskarsdottir, W. Lemahieu. Deep learning for credit scoring: Do or don't, *European Journal of Operational Research*, 2021, Volume (295), p.p. 292-305.

16. S. Hamori, M. Kawai, T. Kume, Y. Murakami, C. Watanabe. Ensemble Learning or Deep Learning? *Application to Default Risk Analysis, Journal of Risk and Financial Management*, 2018, 11(1):12, p.p. 1-14.

17. Y.C. Lee. Application of support vector machines to corporate credit rating prediction, *Expert Systems with Applications*, 2007, Volume (33), p.p. 67-74.

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

19. Y. Li, W. Chen. A Comparative Performance Assessment of Ensemble Learning for Credit Scoring, *Mathematics*, 2020, Volume (8), p.p. 1-19.

20. S. Li, M. Wang, J. He. Prediction of Banking Systemic Risk Based on Support Vector Machine, *Mathematical Problems in Engineering*, 2013, p.p. 1-5.

21. C. Luo, D. Wu, D. Wu. A deep learning approach for credit scoring using credit default swaps, *Engineering Applications of Artificial Intelligence*, 2016, Volume (65), p.p. 465-470.

22. M. Mahbobi, S. Kimiagari, M. Vasudevan. Credit risk classification: an integrated predictive accuracy algorithm using artificial and deep neural network, *Annals of Operations Research*, 2021, p.p. 1-29.

23. V. Moscato, A. Picariello, G. Sperli. A benchmark of machine learning approaches for credit score prediction, *Expert Systems With Applications*, 2021, Volume (165), p.p. 1-8.

24. L. Munkhdalai, T. Munkhdalai, O.E. Namsrai, J.Y. Lee, K.H. Ryu. An Empirical Comparison of Machine-Learning Methods on Bank Client Credit Assesments, *Sustainability*, 2019, 11(3), p.p. 1-23.

25. N. Sariannidis, S. Papadakis, A. Garefalakis, C. Lemonakis, T. Kyriaki-Argyro. Default avoidance on credit card portfolios using accounting, demographical and exploratory factors: decision making based on machine learning (ML) techniques, *Annals of Operations Research*, 2019, Volume (294), p.p. 715-739.

26. D. Tripathi, D.R. Edla, A. Bablani, A.K. Shukla, B.R. Reddy. Experimental analysis of machine learning methods for credit score classification, *Progress in Artificial Intellifence*, 2021, Volume (10), p.p. 217-243.

27. S.K. Trivedi. A study on credit scoring modelling with different feature selection and machine learning approaches, *Technology in Society*, 2020, Volume (63), p.p. 1-9.

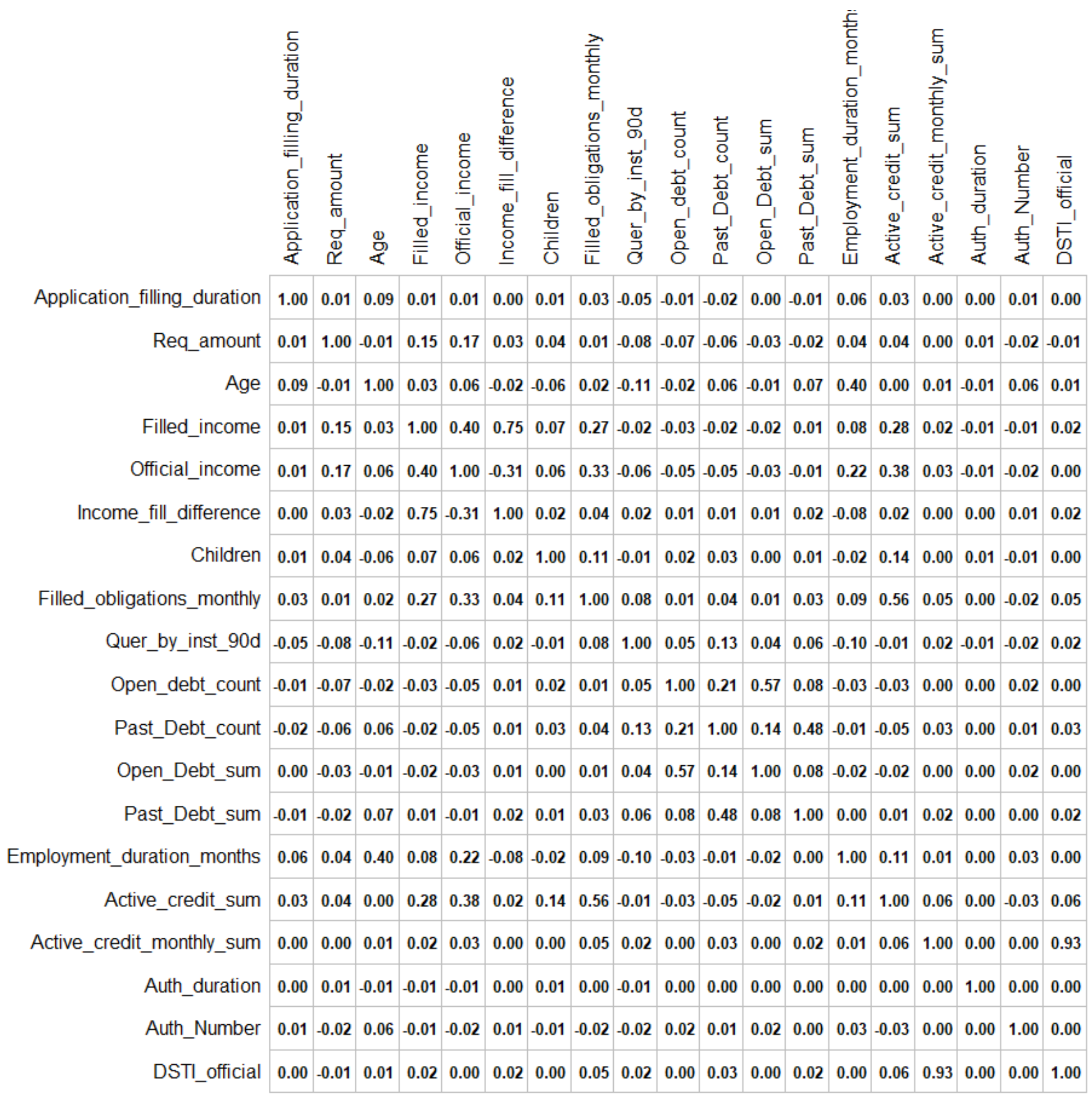
28. M. Wallis, K. Kumar, A. Gepp. Credit Rating Forecasting Using Machine Learning Techniques, *Intelligent Big Data Analytics*: A Managerial Perspective, 2019, p.p. 180-199.
29. H.C. Wu, Y.H. Hu, Y.H. Huang. Two-stage credit rating prediction using machine learning techniques, *Kybernetes*, 2014, Volume (43), p.p. 1098-1113.

Appendices

Appendix 1 – The list of variables included in the study

Variable	Description	Category	Format
Application_rating5	*5 classes of applicant's credit score (A. B. C. D. E)*	*Legal institutions*	*Dependent*
Application_rating3	*3 classes of applicant's credit score (A-B. C-D. E)*	*Legal institutions*	*Dependent*
Application_rating2	*2 classes of applicant's credit score (Reliable (A-B) and Risky (C-E)*	*Legal institutions*	*Dependent*
Official_income	*Income obtained from The State Social Insurance Fund Board under the Ministry of Social Security and Labour (SODRA)*	*Legal institutions*	*Numerical*
Quer_by_inst_90d	*How many querries (requests) were made for financial institutions via 90 days for this applicant*	*Legal institutions*	*Numerical*
Has_real_est	*Has applicant real estate?*	*Legal institutions*	*Boolean*
Has_open_debt	*Has applicant open debt?*	*Legal institutions*	*Boolean*
Open_debt_count	*The amount of debts (in units). a certain indiviual has now*	*Legal institutions*	*Numerical*
Open_Debt_sum	*The amount of debts (in EUR). a certain indiviual has now*	*Legal institutions*	*Numerical*
Has_debt_Past	*Had applicant any debts in the past?*	*Legal institutions*	*Boolean*
Past_Debt_count	*The amount of debts (in units). a certain indiviual had in the past*	*Legal institutions*	*Numerical*
Past_Debt_sum	*The amount of debts (in EUR). a certain indiviual had in the past*	*Legal institutions*	*Numerical*
Employment_duration_months	*Employment duration at current workplace in months*	*Legal institutions*	*Numerical*
Has_activ_credit	*Has applicant current credit?*	*Legal institutions*	*Boolean*
Active_credit_sum	*The amount of credit (in EUR). a certain indiviual has now*	*Legal institutions*	*Numerical*
Active_credit_monthly_sum	*Monthly amount of payments for credit*	*Legal institutions*	*Numerical*
DSTI_official	*Official Debt-to-Income ratio*	*Legal instututions*	*Numerical*
Req_amount	*Requested loan amount*	*Filled by applicant*	*Numerical*
Marital_status	*Marital status (Married/single)*	*Filled by applicant*	*Class*
Age	*Age*	*Filled by applicant*	*Numerical*
Gender	*Gender (Male/Female)*	*Filled by applicant*	*Class*
City_classifier	*Classifier of cities by its population (Big city>100k citizens; 30k<Medium city<100k; and Small city<30k). Data taken from worldpopulationreview.com*	*Filled by applicant*	*Class*
Filled_income	*Income amount filled by individual*	*Filled by applicant*	*Numerical*
Income_Source	*Income source filled by individual (Labour in Lithuania. individual activity. state officer and so on.)*	*Filled by applicant*	*Class*
Children	*Number of children*	*Filled by applicant*	*Numerical*
Filled_obligations_monthly	*Amount of current monthly financial commitments. filled by customer in EUR*	*Filled by applicant*	*Numerical*
Application_filling_duration	*How long customer fills the application form in seconds*	*Filled by applicant*	*Numerical*
Returning_30_days	*Has customer already filled application during the last 30d*	*Filled by applicant*	*Boolean*
Income_fill_difference	*It is calculated by such formula: filled income - official income.*	*Behavioral*	*Numerical*
Doc_Type	*What type of document was used to confirm identity (passport. identity card. etc.)*	*Behavioral*	*Class*
Auth_duration	*What was duration of identity authentication process in seconds*	*Behavioral*	*Numerical*
Auth_Number	*How many times. customer tried to authenticate his identity*	*Behavioral*	*Numerical*

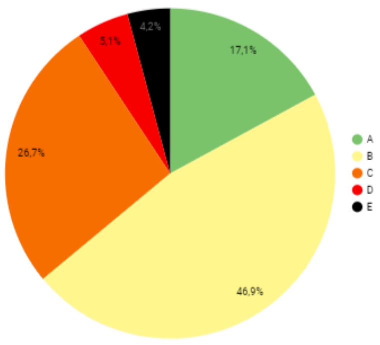
Appendix 2 – Correlation matrix of numerical variables



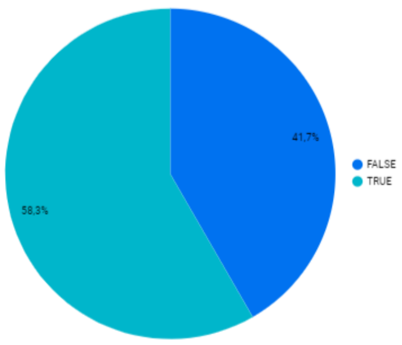
	Application_filling_duration	Req_amount	Age	Filled_income	Official_income	Income_fill_difference	Children	Filled_obligations_monthly	Quer_by_inst_90d	Open_debt_count	Past_Debt_count	Open_Debt_sum	Past_Debt_sum	Employment_duration_month	Active_credit_sum	Active_credit_monthly_sum	Auth_duration	Auth_Number	DSTI_official
Application_filling_duration	1.00	0.01	0.09	0.01	0.01	0.00	0.01	0.03	-0.05	-0.01	-0.02	0.00	-0.01	0.06	0.03	0.00	0.00	0.01	0.00
Req_amount	0.01	1.00	-0.01	0.15	0.17	0.03	0.04	0.01	-0.08	-0.07	-0.06	-0.03	-0.02	0.04	0.04	0.00	0.01	-0.02	-0.01
Age	0.09	-0.01	1.00	0.03	0.06	-0.02	-0.06	0.02	-0.11	-0.02	0.06	-0.01	0.07	0.40	0.00	0.01	-0.01	0.06	0.01
Filled_income	0.01	0.15	0.03	1.00	0.40	0.75	0.07	0.27	-0.02	-0.03	-0.02	-0.02	0.01	0.08	0.28	0.02	-0.01	-0.01	0.02
Official_income	0.01	0.17	0.06	0.40	1.00	-0.31	0.06	0.33	-0.06	-0.05	-0.05	-0.03	-0.01	0.22	0.38	0.03	-0.01	-0.02	0.00
Income_fill_difference	0.00	0.03	-0.02	0.75	-0.31	1.00	0.02	0.04	0.02	0.01	0.01	0.01	0.02	-0.08	0.02	0.00	0.00	0.01	0.02
Children	0.01	0.04	-0.06	0.07	0.06	0.02	1.00	0.11	-0.01	0.02	0.03	0.00	0.01	-0.02	0.14	0.00	0.01	-0.01	0.00
Filled_obligations_monthly	0.03	0.01	0.02	0.27	0.33	0.04	0.11	1.00	0.08	0.01	0.04	0.01	0.03	0.09	0.56	0.05	0.00	-0.02	0.05
Quer_by_inst_90d	-0.05	-0.08	-0.11	-0.02	-0.06	0.02	-0.01	0.08	1.00	0.05	0.13	0.04	0.06	-0.10	-0.01	0.02	-0.01	-0.02	0.02
Open_debt_count	-0.01	-0.07	-0.02	-0.03	-0.05	0.01	0.02	0.01	0.05	1.00	0.21	0.57	0.08	-0.03	-0.03	0.00	0.00	0.02	0.00
Past_Debt_count	-0.02	-0.06	0.06	-0.02	-0.05	0.01	0.03	0.04	0.13	0.21	1.00	0.14	0.48	-0.01	-0.05	0.03	0.00	0.01	0.03
Open_Debt_sum	0.00	-0.03	-0.01	-0.02	-0.03	0.01	0.00	0.01	0.04	0.57	0.14	1.00	0.08	-0.02	-0.02	0.00	0.00	0.02	0.00
Past_Debt_sum	-0.01	-0.02	0.07	0.01	-0.01	0.02	0.01	0.03	0.06	0.08	0.48	0.08	1.00	0.00	0.01	0.02	0.00	0.00	0.02
Employment_duration_months	0.06	0.04	0.40	0.08	0.22	-0.08	-0.02	0.09	-0.10	-0.03	-0.01	-0.02	0.00	1.00	0.11	0.01	0.00	0.03	0.00
Active_credit_sum	0.03	0.04	0.00	0.28	0.38	0.02	0.14	0.56	-0.01	-0.03	-0.05	-0.02	0.01	0.11	1.00	0.06	0.00	-0.03	0.06
Active_credit_monthly_sum	0.00	0.00	0.01	0.02	0.03	0.00	0.00	0.05	0.02	0.00	0.03	0.00	0.02	0.01	0.06	1.00	0.00	0.00	0.93
Auth_duration	0.00	0.01	-0.01	-0.01	-0.01	0.00	0.01	0.00	-0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
Auth_Number	0.01	-0.02	0.06	-0.01	-0.02	0.01	-0.01	-0.02	-0.02	0.02	0.01	0.02	0.00	0.03	-0.03	0.00	0.00	1.00	0.00
DSTI_official	0.00	-0.01	0.01	0.02	0.00	0.02	0.00	0.05	0.02	0.00	0.03	0.00	0.02	0.00	0.06	0.93	0.00	0.00	1.00

Appendix 3 – Categorical variables distribution

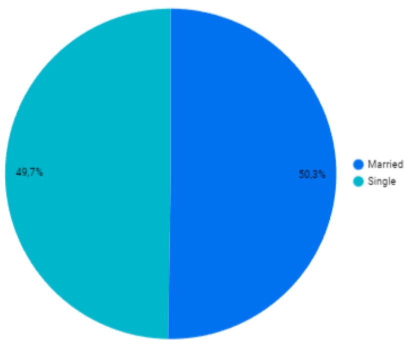
1. Credit scores distribution



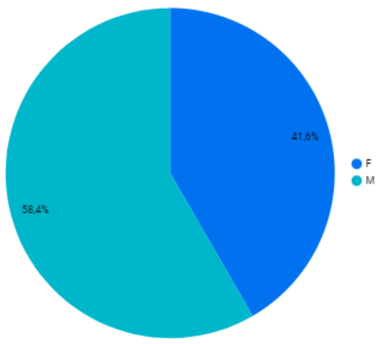
2. Has_real_est distribution



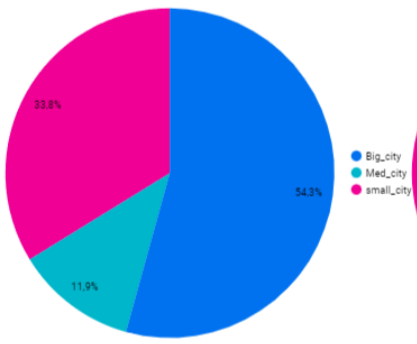
3. Marital_status distribution



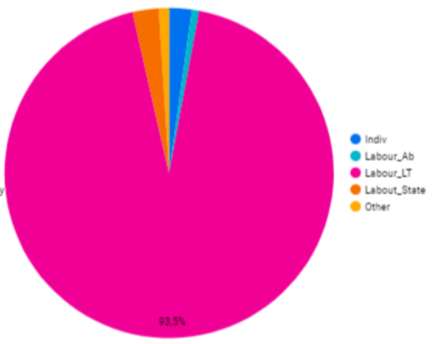
4. Gender distribution



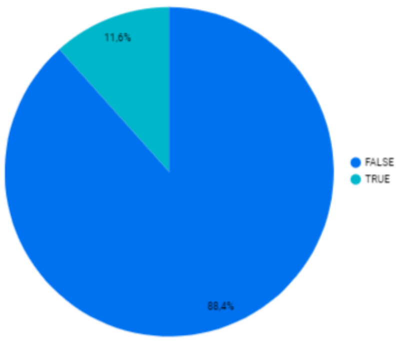
5. City_classifier distribution



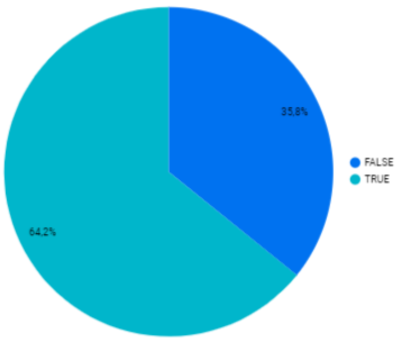
6. Income_source distribution



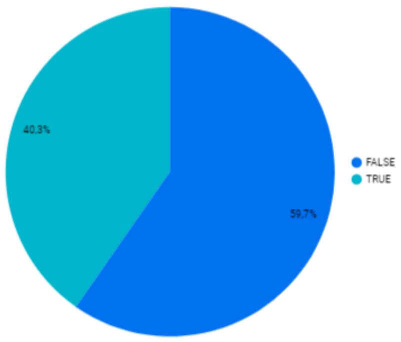
7. Returning_30d distribution



8. Has_active_credit distribution



9. Has_past_debt distribution



Appendix 4 – The program code for binary classification.

```
#Code for binary classification
#---------------------------------------------------------------------------
#Libraries
Sys.setenv(LANG="EN")
library(readxl)
#---------------------------------------------------------------------------
#uploading and cleaning dataset
Final_dataset <- read_excel("Final_dataset.xlsx")
Final_dataset=as.data.frame(Final_dataset)
Final_dataset$Application_rating15=as.factor(Final_dataset$Application_rating
15)
Final_dataset$Application_rating5class=as.factor(Final_dataset$Application_ra
ting5class)
Final_dataset$Application_rating3class=as.factor(Final_dataset$Application_ra
ting3class)
Final_dataset$Application_rating2class=as.factor(Final_dataset$Application_ra
ting2class)
Final_dataset$Returning_30_days_cat=as.factor(Final_dataset$Returning_30_days
_cat)
Final_dataset$Doc_Type=as.factor(Final_dataset$Doc_Type)
Final_dataset$Marital_status=as.factor(Final_dataset$Marital_status)
Final_dataset$Gender=as.factor(Final_dataset$Gender)
Final_dataset$City_classifier=as.factor(Final_dataset$City_classifier)
Final_dataset$Income_Source=as.factor(Final_dataset$Income_Source)
Final_dataset$Has_real_est=as.factor(Final_dataset$Has_real_est)
Final_dataset$Has_open_debt=as.factor(Final_dataset$Has_open_debt)
Final_dataset$Has_debt_Past=as.factor(Final_dataset$Has_debt_Past)
Final_dataset$Employment_duration_months=as.numeric(Final_dataset$Employment_
duration_months)
Final_dataset$Has_activ_credit=as.factor(Final_dataset$Has_activ_credit)
Final_dataset$Email=as.factor(Final_dataset$Email)
Final_dataset$Auth_Meth=as.factor(Final_dataset$Auth_Meth)

str(Final_dataset)
colnames(Final_dataset)
#checking NA
apply(Final_dataset,2, function(x) any(is.na(x)))

#===========================================================================
#Normalizing Variables
min_max_norm <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

dataset_normalized=as.data.frame(lapply(Final_dataset[c("Application_filling_
duration","Req_amount","Age","Filled_income",
"Official_income","Income_fill_difference",
"Children","Filled_obligations_monthly",
"Filled_DSTI","Quer_by_inst_90d","Open_debt_count",
"Past_Debt_count","Open_Debt_sum","Past_Debt_sum",
"Employment_duration_months","Active_credit_sum",
"Active_credit_monthly_sum","Auth_duration","Auth_Number")],
                                        min_max_norm))
dataset_normalized$Application_rating2class=Final_dataset$Application_rating2
class
dataset_normalized$Application_rating2_numerical=Final_dataset$Application_ra
ting2_numerical
dataset_normalized$Returning_30_days_num=Final_dataset$Returning_30_days_num
dataset_normalized$ID_Card=Final_dataset$ID_Card
dataset_normalized$Passport=Final_dataset$Passport
```

```
dataset_normalized$Married=Final_dataset$Married
dataset_normalized$Male=Final_dataset$Male
dataset_normalized$Big_City=Final_dataset$Big_City
dataset_normalized$Med_city=Final_dataset$Med_city
dataset_normalized$Labour_LT=Final_dataset$Labour_LT
dataset_normalized$Labour_Abroad=Final_dataset$Labour_Abroad
dataset_normalized$Individual_work=Final_dataset$Individual_work
dataset_normalized$State_officer=Final_dataset$State_officer
dataset_normalized$Has_real_est_num=Final_dataset$Has_real_est_num
dataset_normalized$Has_open_debt_num=Final_dataset$Has_open_debt_num
dataset_normalized$Has_debt_Past_num=Final_dataset$Has_debt_Past_num
dataset_normalized$Has_activ_credit_num=Final_dataset$Has_activ_credit_num

#==========================================================================
#Exploratory analysis
#1. Means
str(Final_dataset)
colnames(dataset_normalized)
numerical_dataset=Final_dataset[,c("Application_rating2class","Application_fi
lling_duration",

"Req_amount","Age","Filled_income","Official_income","Income_fill_difference"
,                       "Children","Filled_obligations_monthly","Filled_DSTI",
"Quer_by_inst_90d","Open_debt_count","Past_Debt_count",
"Open_Debt_sum","Past_Debt_sum","Employment_duration_months",
"Active_credit_sum","Active_credit_monthly_sum","Auth_duration",
                                   "Auth_Number","DSTI_official")]
summary(numerical_dataset)
#--------------------------------------------------------------------------
#Checking means:
aggregate(numerical_dataset[,2:21],
list(numerical_dataset$Application_rating2class), mean)

##########################################################################
#Correlation and interdependence
##########################################################################

myvars<-c("Application_filling_duration",

"Req_amount","Age","Filled_income","Official_income","Income_fill_difference"
,
          "Children","Filled_obligations_monthly",
          "Quer_by_inst_90d","Open_debt_count","Past_Debt_count",
          "Open_Debt_sum","Past_Debt_sum","Employment_duration_months",
          "Active_credit_sum","Active_credit_monthly_sum","Auth_duration",
          "Auth_Number","DSTI_official")

d=Final_dataset[myvars]
#Generate correlation matrix
cor(d, use="pairwise.complete.obs")
#Visualising correlation
library(corrplot)
M <- cor(d, use="pairwise.complete.obs")
corrplot(M, method = "number")
(order.AOE <- corrMatOrder(M, order="AOE"))
(order.FPC <- corrMatOrder(M, order="FPC"))
(order.hc <- corrMatOrder(M, order="hclust"))
(order.hc2 <- corrMatOrder(M, order="hclust", hclust.method="ward"))
corrplot(M, method="number", col = "Black",
         cl.cex = 1, number.cex = 0.8, tl.col="Black")
help("corrplot")
M.AOE <- M[order.AOE,order.AOE ]
M.FPC <- M[order.FPC,order.FPC ]
M.hc <- M[order.hc, order.hc ]
```

```
M.hc2 <- M[order.hc2,order.hc2]
par(ask=FALSE)
png(height=1200, width=1500, pointsize=10, file="overlap.png")
corrplot(M.FPC, tl.cex=2, tl.col='black', method='number',
         addCoef.col = "grey") # Original order
dev.off()
corrplot(M.FPC, tl.cex=1, tl.col='black') # The first principal component order

#----------------------------------------------------------------------------
#spliting cleaned data
library(ROCR)
library(sjPlot)
#library(caret)
set.seed(65450)
#set.seed(65451) - for second split
#set.seed(65452) - for third split
split  <-  sample(c(TRUE,  FALSE),  nrow(dataset_normalized),  replace=TRUE,
prob=c(0.8,0.2))
training=dataset_normalized[split,]
testing=dataset_normalized[!split,]
summary(testing)
colnames(testing)
#-------------------------------------
# Sampling Training data
library(ROSE)
library(randomForest)
library(caret)
library(e1071)
library(performanceEstimation)
summary(training$Application_rating2class)
#Random Oversampling
oversampled_training   <-   ovun.sample(Application_rating2class~.,   data   =
training, method = "over", N = 29836, seed = 12345)$data
summary(oversampled_training$Application_rating2class)
#SMOTE
Smote_training <- smote(Application_rating2class~.,data = training, perc.over
= 0.5, k=10, perc.under = 3)
summary(Smote_training$Application_rating2class)
#----------------------------------------------------------------------------
set.seed(123)

#============================================================================
# Logistic Regression
#============================================================================
#prediction
library(pROC)
library(caret)
library(InformationValue)
library(ISLR)
start_time_Logistic <- Sys.time()
set.seed(1123)


regression=glm(Application_rating2class~Application_filling_duration+Req_amou
nt+Age+Official_income+
                 Income_fill_difference+Children+Filled_obligations_monthly+
                 Quer_by_inst_90d+Open_debt_count+Past_Debt_count+

Open_Debt_sum+Past_Debt_sum+Employment_duration_months+Active_credit_sum+
                 Active_credit_monthly_sum+Auth_duration+Auth_Number+
                 Returning_30_days_num+ID_Card+Married+
                 Male+Big_City+Med_city+Labour_LT+
                Labour_Abroad+Individual_work+State_officer+Has_real_est_num+
                 Has_open_debt_num+Has_debt_Past_num+Has_activ_credit_num,
               data = training, family=binomial(link="logit"))
end_time_Logistic<-Sys.time()
Logistic_duration=end_time_Logistic-start_time_Logistic
Logistic_duration

summary(regression)
# Evaluation of regression
library(car)
vif(regression)
# AUC
probabilities = predict(regression,testing,type="response")
roc_object <- roc( testing$Application_rating2class,probabilities)
plot(roc_object,   main="ROC   Curve   [Logistic   Regression]",print.auc=TRUE,
print.auc.x = 0.5, print.auc.y = 0.3, add=FALSE)
auc(roc_object)

#Confusion matrix
confusionMatrix(testing$Application_rating2class, probabilities)

#===============================================================================
# RANDOM FOREST
#===============================================================================

#Random Forest
library(randomForest)
colnames(training)

?randomForest
#Short Data
start_time_RF <- Sys.time()
start_time_RF
rf=randomForest(Application_rating2_numerical~Application_filling_duration+Re
q_amount+Age+Official_income+
                  Income_fill_difference+Children+Filled_obligations_monthly+
                  Quer_by_inst_90d+Open_debt_count+Past_Debt_count+
    Open_Debt_sum+Past_Debt_sum+Employment_duration_months+Active_credit_sum+
                  Active_credit_monthly_sum+Auth_duration+Auth_Number+
                  Returning_30_days_num+ID_Card+Married+
                  Male+Big_City+Med_city+Labour_LT+
                Labour_Abroad+Individual_work+State_officer+Has_real_est_num+
                  Has_open_debt_num+Has_debt_Past_num+Has_activ_credit_num,
                      ntree=200,mtry=5,
                data = oversampled_training
                )
end_time_RF<-Sys.time()
RF_duration=end_time_RF-start_time_RF
RF_duration

# Confusion
RF2_predictions <- predict(rf, testing, type = "class")
confusionMatrix(testing$Application_rating2class,RF2_predictions)

#ROC
test.predictions_RF <- predict(rf, newdata=testing)
roc.RFtest              <-              roc(testing$Application_rating2class,
as.numeric(test.predictions_RF))
plot(roc.RFtest, add = FALSE,col = "Blue", print.auc=TRUE, print.auc.x = 0.5,
print.auc.y = 0.3)

print(rf)
rf
# Plotting model
plot(rf)
```

```
# Importance plot
importance(rf)
# Variable importance plot
varImpPlot(rf)

#============================================================================
# SVM
#============================================================================
library(e1071)

Start_time_SVM<-Sys.time()
set.seed(123)
svm_fit=svm(Application_rating2_numerical~Application_filling_duration+Req_am
ount+Age+Official_income+
              Income_fill_difference+Children+Filled_obligations_monthly+
              Quer_by_inst_90d+Open_debt_count+Past_Debt_count+
    Open_Debt_sum+Past_Debt_sum+Employment_duration_months+Active_credit_sum+
              Active_credit_monthly_sum+Auth_duration+Auth_Number+
              Returning_30_days_num+ID_Card+Married+
              Male+Big_City+Med_city+Labour_LT+
              Labour_Abroad+Individual_work+State_officer+Has_real_est_num+
              Has_open_debt_num+Has_debt_Past_num+Has_activ_credit_num,
            data = Smote_training, kernel="radial",gamma=0.05,
        cost=2)
end_time_SVM<-Sys.time()
SVM_duration=end_time_SVM-Start_time_SVM
SVM_duration

print(svm_fit)
summary(svm_fit)
svm_fit

# Predict the testing set with the trained model
SVM_predictions <- predict(svm_fit, testing, type = "class")

# Accuracy and other metrics
confusionMatrix(testing$Application_rating2class, SVM_predictions)

svmPrediction=predict(svm_fit,testing)

roc_svm_test  <-  roc(response  =  testing$Application_rating2class,  predictor
=as.numeric(svmPrediction))
plot(roc_svm_test,  add  =  FALSE,col  =  "green",  print.auc=TRUE,  print.auc.x  =
0.5, print.auc.y = 0.3)

#============================================================================
# XGBoost
library(xgboost)
?xgboost
library(pacman)

#prepare training data
colnames(training_short)
trainm=data.matrix(oversampled_training[,c("Application_filling_duration","Re
q_amount","Age","Official_income",
"Income_fill_difference","Children","Filled_obligations_monthly",
"Quer_by_inst_90d","Open_debt_count","Past_Debt_count",
"Open_Debt_sum","Past_Debt_sum","Employment_duration_months","Active_credit_s
um",
"Active_credit_monthly_sum","Auth_duration","Auth_Number",
"Returning_30_days_num","ID_Card","Married","Male","Big_City","Med_city","Lab
our_LT",
"Labour_Abroad","Individual_work","State_officer","Has_real_est_num",
"Has_open_debt_num","Has_debt_Past_num","Has_activ_credit_num")])

train_label = oversampled_training[,"Application_rating2_numerical"]

train_matrix = xgb.DMatrix(data = as.matrix(trainm), label = train_label)

#prepare validation data
testm=data.matrix(testing[,c("Application_filling_duration","Req_amount","Age
","Official_income",
"Income_fill_difference","Children","Filled_obligations_monthly",
                      "Quer_by_inst_90d","Open_debt_count","Past_Debt_count",
"Open_Debt_sum","Past_Debt_sum","Employment_duration_months","Active_credit_s
um",
"Active_credit_monthly_sum","Auth_duration","Auth_Number",
"Returning_30_days_num","ID_Card","Married",
"Male","Big_City","Med_city","Labour_LT",
"Labour_Abroad","Individual_work","State_officer","Has_real_est_num",
"Has_open_debt_num","Has_debt_Past_num","Has_activ_credit_num")])

test_label = testing[,"Application_rating2_numerical"]

test_matrix = xgb.DMatrix(data = as.matrix(testm), label = test_label)

?xgb.train
#parameters
xgb_params = list(objective   = "binary:logistic",
                  eval_metric = "error",
                  max_depth   = 6,
                  eta         = 0.01,
                  gammma      = 1,
                  colsample_bytree = 0.5,
                  min_child_weight = 2)

#model

set.seed(123)
start_time_XGboost<-Sys.time()
XGBoost_model = xgb.train(params = xgb_params, data = train_matrix,
                          nrounds = 500)

end_time_XGboost<-Sys.time()
XGboost_duration=end_time_XGboost-start_time_XGboost
XGboost_duration

#feature importance
imp = xgb.importance(colnames(train_matrix), model = XGBoost_model)
xgb.plot.importance(imp)


# Confusion matrix
XGB2_predictions <- predict(XGBoost_model, newdata = test_matrix, type =
"class")
confusionMatrix(testing$Application_rating2class, XGB2_predictions)

#NORMAL AUC
p_full = predict(XGBoost_model, newdata = test_matrix)
plot.roc(test_label, p_full, col="Orange", print.auc=T, print.auc.y=0.5,
add=FALSE)
```

```
#----------------------
#Keras ANN
#----------------------

library(keras)
library(lime)
library(rsample)
library(recipes)

?keras_model_sequential
model_keras22 <- keras_model_sequential()

model_keras22 %>%

  # First hidden layer
  layer_dense(
    units              = 20,
    kernel_initializer = "uniform",
    activation         = "relu",
    input_shape        = ncol(trainm)) %>%

  # Dropout to prevent overfitting
  layer_dropout(rate = 0.1) %>%

  # Second hidden layer
  layer_dense(
    units              = 20,
    kernel_initializer = "uniform",
    activation         = "relu") %>%

  # Dropout to prevent overfitting
  layer_dropout(rate = 0.1) %>%


  # Output layer
  layer_dense(
    units              = 1,
    kernel_initializer = "uniform",
    activation         = "sigmoid") %>%

  # Compile ANN
  compile(
    optimizer = optimizer_adam(learning_rate = 0.0001),
    loss      = 'binary_crossentropy',
    metrics   = c('accuracy'))
model_keras22


start_time_ANN2L<-Sys.time()
history <- fit(
  object           = model_keras22,
  x                = as.matrix(trainm),
  y                = train_label,
  batch_size       = 32,
  epochs           = 300,
  validation_split = 0.20
)
End_time_ANN2L<-Sys.time()
Duration_ANN2_L=End_time_ANN2L-start_time_ANN2L
Duration_ANN2_L

print(history)

plot(history)
```

```
#-----------------------------------
# 2 Layers results
#-----------------------------------
# Confusion matrix
x = as.matrix(testm)
predictions_keras=model_keras22 %>% predict(x, batch_size=32)
y_pred=round(predictions_keras)
predictions_keras=as.numeric(predictions_keras[,1])
confusion_matrix=table(test_label,y_pred)
confusion_matrix
#Auc
roc_object_Keras <- roc(testing$Application_rating2class, predictions_keras)
plot(roc_object_Keras, main="ROC Curve [Keras Neural Network]",print.auc=TRUE,
print.auc.x = 0.5, print.auc.y = 0.3, add=FALSE)
auc(roc_object_Keras)

#-------------------------
# 4 layers
model_keras4 <- keras_model_sequential()

model_keras4 %>%

  # First hidden layer
  layer_dense(
    units              = 20,
    kernel_initializer = "uniform",
    activation         = "relu",
    input_shape        = ncol(trainm)) %>%

  # Dropout to prevent overfitting
  layer_dropout(rate = 0.1) %>%

  # Second hidden layer
  layer_dense(
    units              = 20,
    kernel_initializer = "uniform",
    activation         = "relu") %>%

  # Dropout to prevent overfitting
  layer_dropout(rate = 0.1) %>%

  # Third hidden layer
  layer_dense(
    units              = 20,
    kernel_initializer = "uniform",
    activation         = "relu") %>%

  # Dropout to prevent overfitting
  layer_dropout(rate = 0.1) %>%

  # Fourth hidden layer
  layer_dense(
    units              = 20,
    kernel_initializer = "uniform",
    activation         = "relu") %>%

  # Dropout to prevent overfitting
  layer_dropout(rate = 0.1) %>%

  # Output layer
  layer_dense(
    units              = 1,
    kernel_initializer = "uniform",
    activation         = "sigmoid") %>%

  # Compile ANN
  compile(
    optimizer = optimizer_adam(learning_rate = 0.0001),
    loss      = 'binary_crossentropy',
    metrics   = c('accuracy'))
model_keras4


start_time_ANN4L<-Sys.time()
history <- fit(
  object           = model_keras4,
  x                = as.matrix(trainm),
  y                = train_label,
  batch_size       = 32,
  epochs           = 300,
  validation_split = 0.20
)

End_time_ANN4L<-Sys.time()
Duration_ANN4_L=End_time_ANN4L-start_time_ANN4L
Duration_ANN4_L

print(history)

plot(history)

#-----------------------------------
# 4 Layers results
#-----------------------------------
# Confusion matrix
x = as.matrix(testm)
predictions_keras=model_keras4 %>% predict(x, batch_size=32)
y_pred=round(predictions_keras)
predictions_keras=as.numeric(predictions_keras[,1])
confusion_matrix=table(test_label,y_pred)
confusion_matrix
#Auc
roc_object_Keras <- roc(testing$Application_rating2class, predictions_keras)
plot(roc_object_Keras, main="ROC Curve [Keras Neural Network]",print.auc=TRUE,
print.auc.x = 0.5, print.auc.y = 0.3, add=FALSE)
auc(roc_object_Keras)
```