UNIVERSITY OF MACEDONIA
GRADUATE PROGRAM
DEPARTMENT OF APPLIED INFORMATICS

CREDIT RISK ANALYSIS VIA MACHINE LEARNING METHODS: CLIENT SEGMENTATION BASED ON PROBABILITY OF DEFAULT

Master Thesis

of

Grigoriou Kassiani

Thessaloniki, June 2021

CREDIT RISK ANALYSIS VIA MACHINE LEARNING METHODS: CLIENT SEGMENTATION BASED ON PROBABILITY OF DEFAULT

Kassiani Grigoriou

Bachelor degree in Mathematics, Aristotle University of Thessaloniki, 2017

Master Thesis

A thesis submitted in fulfilment of the requirements for the degree of

MASTER OF SCIENCE IN APPLIED INFORMATICS

Supervising Professor
Dasilas Apostolos

Approved by the three-member Examining Committee dd/mm/yyyy

Dasilas Apostolos	Refanidis Ioannis	Steiakakis Emmanouil
.....	.....	.....

Kassiani Grigoriou

.....

Acknowledgements

First of all, I would like to thank Dr. Apostolos Dasilas for giving me the opportunity to fulfill this dissertation as well as providing me a great guidance throughout this period. His contribution means a lot to me. In addition, I would like to mention that I am grateful to University of Macedonia for giving me the chance to be a postgraduate student there. The knowledge that I acquired through my attendance was highly valuable, whereas it helped me finding my professional path. Last but not least, I would like to thank my family, my friends and my classmates for the continuous support during these two years.

Περίληψη

Η ραγδαία εξέλιξη της τεχνολογίας, το ανταγωνιστικό περιβάλλον, καθώς και ο τεράστιος όγκος δεδομένων που είναι διαθέσιμος σήμερα, κρίνουν επιτακτική την ανάγκη των επιχειρήσεων να στραφούν προς τη νέα ψηφιακή πραγματικότητα. Η αυτοματοποίηση των διαδικασιών αλλά και η λήψη αποφάσεων μέσω της αξιοποίησης των δεδομένων με τη χρήση νέων μεθόδων, όπως η τεχνητή νοημοσύνη και η μηχανική μάθηση αποτελούν πρωταρχικό στόχο των οργανισμών. Αυτό το ενδιαφέρον παρουσιάζεται έντονο και στον τομέα των τραπεζών. Η ανάλυση του μεγάλου όγκου των δεδομένων που διαθέτουν συνυπολογίζοντας την ιδιαιτερότητα αυτών με βάση τον προσωπικό τους χαρακτήρα αποτελεί μία πρόκληση για αυτές. Η ανάλυση και η αξιολόγηση του πιστωτικού κινδύνου συνιστά μία από τις πιο σημαντικές διαδικασίες των χρηματοπιστωτικών ιδρυμάτων. Στην παρούσα διπλωματική εργασία, αναπτύχθηκαν 3 μοντέλα εποπτευόμενης μηχανικής μάθησης, με τα οποία γίνεται ταξινόμηση των πελατών μίας τράπεζας σε «καλούς» ή «κακούς» με βάση την πιθανότητα αθέτησης των υποχρεώσεών τους. Οι αλγόριθμοι που χρησιμοποιήθηκαν είναι οι Random Forest, KNN και Decision Trees.

Λέξεις - κλειδιά: πιστωτικός κίνδυνος, μηχανική μάθηση, πιθανότητα αθέτησης, αξιολόγηση πιστωτικού κινδύνου, χρεωκοπία, εντοπισμός απάτης

Abstract

The rapid evolution of the technology, the competitive environment, as well as the huge amount of data that is available today, lead businesses to switch to the new digital reality. Automation of processes and decision-making through the use of data using new methods such as artificial intelligence and machine learning are a primary objective of organizations. This interest is strongly present in the banking sector, too. The analysis of the large volume of data that is available, whereas taking into account their personal nature is a huge challenge for financial institutions. Credit risk analysis and assessment is one of the most important processes for this kind of business. In this dissertation, 3 models of supervised machine learning were developed, which classify bank's customers into "good" or "bad" based on the probability of default on their obligations. The algorithms used are Random Forest, KNN and Decision Trees.

Keywords: credit risk, probability of default, machine learning, credit scoring, bankruptcy, fraud detection

Contents

Contents of figures

Contents of tables

Credit Risk Analysis with machine learning

1.Introduction

For over two hundred years, in the field of prediction of the bankruptcy of an organization most evaluations were done subjectively (Bellovary, Giacomino, and Akers, 2007; Li and Miu, 2010; de Andrés, Landajo, and Lorca, 2012). It was not until the twentieth century that more quantitative procedures came gained some attention; like the seminal univariate analysis work of Beaver (1966) and multiple discriminant analysis work of Altman in the 1960s. Their work exhibited the capacity to predict an organization's failure even five years before it happens. Such data is a resource not only to the organization or investors, but also to numerous different partners, for example, suppliers and employees (Wilson and Sharda, 1994).

To get a better understanding of the importance and the potential effects of a bankruptcy of an organization on everyone, we have to remember what happened to Lehman Brothers Holdings Inc. If they had a reliable forecasting algorithm, they could have seen the patterns and prevent this catastrophy from happening. Companies all over the world would love to have a system like that to help them predict possible crisis and take actions to avoid it.

According to McKinsey & Co, the risk functions in banking institutions should be very different by 2025. The expansion of regulations, growing customer expectations and the evolution of risk types are expected to lead to new products, services and risk management techniques. Machine learning can enable the creation of more accurate risk models by locating complex, non-linear patterns in large data sets. It is expected that machine learning will be applied in many areas within a bank, especially in the area of risk management function.

The study seeks to examine the extent to which machine learning, which has become a very important factor for businesses, has been researched in the context of risk management in the banking industry. The aim of the dissertation is to analyze and evaluate the machine learning techniques applied in risk management in financial institutions and to examine possible problems that have been identified. The dataset that is used in this thesis can be found on Kaggle, in which 32,581 borrowers' data is included. It provides information about age, income, home status, employment length, loan intent, loan amount, loan grade, interest rate, loan to income ratio, historical default and loan status of each borrower.

The next chapter presents the literature review of a part of the extraordinary volume of published work about credit risk analysis with machine learning. The third chapter offers a brief analysis of the main ML techniques such as logistic regression, support vector machines, decision trees used in machine learning in the field of credit risk analysis and a presentation of their advantages and disadvantages. In chapter 4, we develop three different models with real data and discuss the results and their efficacy on improving business decisions. Chapter 5 follows with the conclusions of the study.

2. Literature review

2.1. Introduction

According to studies on the subject of credit risk analysis with machine learning, it seems that ML techniques are superior to traditional statistical models (Malhotra, 2003). The study by Malhotra showed that neural networks have better results than traditional statistical techniques. However, Huang and Day (2013) showed that Support Vector Machine (SVM) models have better accuracy rates among the 17 classification models surveyed, in terms of credit score. This is supported by Khemakhem and Boujelbene (2017), who conducted a study on credit risk assessment for Tunisian banks and compared traditional models with modern Artificial Neural Networks (ANN) and SVM. Nwulu and Nnamdi (2011) performed a comparative analysis of SVMs and ANNs for credit rating and concluded that ANNs performed slightly better than SVMs.

Credit risk assessment is done through the development of classification models, in order to distinguish between reliable and unreliable customers (Dima et al., 2009). A common approach to credit risk assessment is to apply some sort of classification technique to previous customer data so that we can find some relationship between customer characteristics and loan repayment failure. There seems to be a growing research interest in assessing credit risk through machine learning techniques.

Recent studies have found that artificial intelligence (AI) techniques, such as SVM and neural networks, perform better than traditional statistical models and optimization techniques for assessing credit risk due to the weight coordination flexibility. The following pages cover a part of the surveys related to each type of credit risk separately.

2.2. Credit scoring

Credit scoring using machine learning is generally done using some kinds of classifier that differentiates between trusted and unreliable customers using previous customer data. The ML techniques used by researchers for credit scoring are neural networks, SVM, Naive Bayes, Bayesian Networks, Decision Tree, Hybrid models and Ensemble models. Neural networks have become increasingly popular with researchers in recent years. Li et al. (2002) proposed a model

based on the Back-Propagation (BP) algorithm to determine good versus bad creditors. Hu and Tang (2006) proposed an informed credit risk assessment based on artificial neural network (ANN), which measures the applicant's credit score. The most suitable candidates for this model are the commercial banks that have incomplete data. Dima et al. (2010) proposed an ANN model for corporate credit risk assessment to classify good creditors from bad ones. In their document, they assess the risk of the company defaulting on an international sample of 3,000 companies applying for credit at an international bank operating in Romania. The sample includes the general population of companies in Romania. Based on their past credit history, they have divided companies into seven categories. They made their estimates first using logit regression and then ANN (Artificial Neural Networks) and compared the results with Standard & Poor's.

Tomczak and Zieba (2014) in their study, proposed a new machine learning technique which utilizes the Classification Restricted Boltzmann Machine (ClassRBM) to construct the credit scoreboard. Scoreboards are the simplest models to interpret and can be easily applied to any banking system. Unlike standard methods, their approach uses the powerful classifier (ClassRBM), deals with the unequal class distribution problem, and constructs an extremely understandable and easy-to-apply scoring model. Baesens et al. (2003) analyzed three real-life data sets and presented the results. The analysis was performed using neural network rule extraction techniques. It was concluded that neural rule extraction techniques can be used for credit risk analysis. As can be seen, the researchers are moving to hybrid systems with neural networks. Huang et al. (2005) proposed the classification of state commercial bank loan applicants using fuzzy neural networks.

Oreski et al. (2012) proposed a hybrid system with Genetic Algorithm (GA) and ANN for applicants' creditworthiness. In this model, the selection of features is done using GA and sorting using ANN. The proposed hybrid system was found to be competitive with other models. This article presents an advanced new heuristic algorithm, the Neural Network Hybrid Genetic Algorithm (HGA-NN), which is used to increase the classification accuracy in credit risk assessment by identifying an optimal feature subset. The performance of the proposed classifier is evaluated using a set of credit data collected at a Croatian bank, and the results are further validated in another set of real-world credit data selected from a UCI database. The classification accuracy is compared to that presented in the literature. The findings were very promising for feature selection and classification in retail credit risk assessment and show that the HGA-NN is a

promising addition to existing data mining techniques. Djemaiel et al. (2016) studied a hybrid neural network model created using a combination of the Radial basis function (RBF) neural network and the Elman neural network. The data framework was defined using big data. The proposed model proved effective when used to classify customers as "good" or "bad" based on their credit scores. Therefore, the proposed hybrid model may be a good choice when choosing a grading technique for credit scoring. SVM is a widely researched classification technique for creditworthiness for many reasons. SVMs provide an excellent generalization capability. It is also relatively easy to train. SVMs respond relatively well to multidimensional data. Many have used SVM to score credit. Farquad et al. (2011) in their article, proposed a hybrid SVM model for customer relationship management (CRM) purposes. The approach consists of three phases. In the first phase the SVM-RFE (SVM-recursive feature elimination) is used to reduce the feature set. The reduced data set is then used in the second phase to obtain an SVM model. The rules are then created using the Naive Bayes Tree (NBTree) in the final phase. The dataset analyzed in this study is about the prediction of Churn to the bank card customers (Business Intelligence Cup 2004) and is extremely unbalanced with 93.24% loyal and 6.76% churned customers. From the empirical results it is observed that the proposed hybrid surpassed all the other techniques tested. Feng et al. (2009) suggested the PCA-based SVM classification model for dimensional reduction for commercial banks. It is similar to the PCA-SVM model proposed by Farquad et al (2011). A comparison with the backpropagation neural network (BP) showed that the raw SVM method is more accurate and effective than this. Gestel et al. (2003) proposed a Least Squares SVM classifier for a credit score that surpassed traditional SVM classifiers. This method proved to be better than the traditional Linear Discriminant Analysis (LDA) and Logistic Regression models.

In addition to neural network and SVM-based approaches, several other classification techniques are proposed for creditworthiness. Although not a popular classification model for credit score, the Naive Bayes approach has also been suggested. Vedala and Kumar (2012) proposed a Naive Bayes rating for credit rating. This rating is mainly done on e-lending platforms which uses social networks to expand its database. Okesola et al. (2017) also studied a Naive Bayes classification model for creditworthiness. The input variables in this method are the demographic and material indicators. A modern approach to creditworthiness is the decision tree method (Hand et al., 1997). Szwabe and Misiorek (2018) proposed a decision tree model for credit decision making.

Bayraci and Susuz (2019) in their study, applied a Deep Neural Networks (DNN) with multiple hidden layers to assess the risk profiles of loan clients on datasets taken from a Turkish commercial bank. They compared the predictive ability of deep learning method vs Logistic Regression (LR), Decision Tree, Naïve Bayes and Support Vector Machines (SVM). The results indicated that, the DNN model improves the performance of a credit scoring system in terms of balanced accuracy when compared to the other models.

2.3. Non-Performing Asset (NPA)

Another type of credit risk assessment technique is the NPA. This has to do with predicting which loan is likely to become default, so that appropriate measures can be taken to address the situation. Baruah (2018) studied the applications of artificial intelligence in 4 leading Indian banks and concludes that the use of Machine Learning in customer data leads to better service and provides the customers with a better experience, both in terms of speed and in quality.

D'Monte (2018) concluded that machine learning algorithms can connect users to various banking services, track their spending behavior and their behavior pattern so that it can identify any transaction that is questionable because it does not match the customer profile.

Ahmad and Ariff (2007) studied the credit risks in the developed economies of Australia, USA, Japan, France and the emerging economies of India, Malaysia, Thailand, Mexico. The study concluded that the credit risk in the banks of emerging economies is higher than that of developed economies. The ML techniques used for default prediction are different types of neural networks, SVMs and hybrid models. Zhang (2011) suggested a default early warning risk model based on the BP Neural Network algorithm. A BP Neural Network is trained in data samples to determine the default risk. Makrygiannis and Markopoulos (2016) proposed the default provision using the feedforward ANN, which takes into account the financial and personal information of the loan applicant. The proposed model was found to give really good accuracy. Feki et al. (2012) proposed methods of distinguishing banks by the percentage of non-performing loans (NPLs). It was performed using different approaches of multiclass SVM and Gaussian models.

2.4. Fraud detection

Fraud in financial transactions can jeopardize the reputation of financial institutions among their customers as well as cause great losses. Banks and financial institutions invest in refining fraudulent machine learning algorithms and fraud detection systems (Abakarim et al., 2018). Fraud detection is a binary classification problem. The idea is to apply an appropriate classifier to the problem, which will be trained in an appropriate data set. The main approaches to tackling credit card fraud are ML techniques such as SVM and decision tree.

Zareapoor and Shamsolmoali (2015) published an article on fraud detection using different techniques, such as Naïve Bayes, KNN and SVM. In their work, they raise concerns about the availability of real data as financial institutions do not disclose their data because it is confidential and sensitive. So, investigations end to work on fake data. Another problem is that, most of the time the data is imbalanced as the number of fraudulent transactions is only 2% and 98% of the transactions are legal. They refer to their concerns about the large amount of data and the computation time required for an algorithm to run in these cases. One of the major challenges frequently mentioned in many research papers is that machine learning algorithms need to be updated regularly so that malicious attempts can be recorded in real time.

AdaBoost is used in the work of Randhawa et al. (2018) on credit card fraud detection and examines many different machine learning models, such as Naïve Bayes, Random Forest, etc. The study states that AdaBoost is very sensitive to anomalies and extreme values. Boltzmann (RBM) machines can be used to reconstruct data in an uncontrolled learning environment. Pumsirirat et al. (2018) and Yan et al.'s paper (2018) used python's Keras framework to implement a high-level Neural Network. They used the H2O package to calculate the Mean Square Error. In Chouiekha and El Haj [10], Convolutional Neural Networks (CNN) are used to detect fraud. A database of 18,000 artificial images was created, depicting 300 customer activities for over 60 days. A CNN (Convolutional Neural Network) has been applied to the images to detect fraudulent activity.

Gyamfi and Abdulai (2018) used SVM with Spark (SVM-S) to process data streams representing good and bad customer behavior and then use them to assess the validity of new transactions. Kotsiantis et al. (2006) predicted fraudulent financial statements using a decision tree. The decision tree proved to achieve the best performance among all the classifiers examined. Ravishankar et al.

(2011) conducted an analysis to detect financial statement fraud using data mining techniques. The Probabilistic Neural Network (PNN) surpassed all others.

2.5. The research trends in credit risk assessment

As structural changes in the global financial market have taken place as well as an increase in the overall level of risk, it has become imperative to study credit risk assessment. Over the last 20 years, much progress has been made in the area of credit risk assessment. Credit rating models are made up of two basic and still popular statistical tools: Linear Discriminant Analysis (LDA) and logistic regression (LR). As times change, new methods have arrived such as Neural Networks, SVMs, k-NNs and Decision Trees. There are many other methods as described above. However, Hybrid models and ensemble models are becoming increasingly popular. Primary research conducted in the field of credit risk assessment uses non-linear classification algorithms, such as neural networks and SVMs. SVM has received a lot of attention in the machine learning community. Few attempted to score credit using the Naive Bayes classification (Vedala et al., 2012). For all three types of credit risk assessment techniques, researchers have also proposed several hybrid models that combine parts of two or more algorithms.

2.6. Machine Learning

Machine learning is considered as a tool that finds application in various problems, especially in fields that require data processing and interpretation (Awad and Khanna, 2015). It enables data patterns to be searched for in order to extract important information from data analysis. Machine learning programs can be continuously trained and improved and can be applied to problems that require complexity and adaptability (Shalev-Shwartz and Ben-David, 2014).

The machine learning programs used in search engines and motor vehicles can be adopted and applied in the financial sector as well. The financial sector, with the help of technological developments, has been able to extract and analyze a very large volume of data on markets and consumers. Machine learning is increasingly being adopted by economists every day, with the aim of reducing costs, increasing productivity and managing risk. New needs and new regulations pushed banks to automate (Financial Stability Board, 2017). Although machine learning

algorithms are considered more effective in dealing with complex nonlinear relationships, they are considered difficult to interpret in these cases (Galindo and Tamayo, 2000). There has been a huge increase in the volume of data in financial institutions (FI) in recent years, and one of the reasons this has happened is the digitization process. This data comes from a variety of sources, such as consumer applications, consumer interactions, and other external sources.

Organizations want to automate and develop their analytical capabilities in various areas, such as risk management, thereby increasing interest in machine learning and artificial intelligence in FI (Van Liebergen, 2017). Machine learning is able to influence every aspect of the FI business model, improving understanding of customer preferences and needs, risk management, fraud detection, behavior monitoring, customer support, and even automated verification identity when combined with biometric elements.

Van Liebergen (2017) explains the use of machine learning in financial institutions. It refers to credit risk modeling, credit card fraud detection and money laundering. He also emphasizes that machine learning algorithms are trained in some data and their effectiveness is tested in different data. This could complicate the development and evaluation of the models.

Machine learning is also used in the Securities and Exchange Commission (SEC) in the risk assessment process. The same algorithms can serve as a guide for a bank on how they can be applied to fraud detection (Bauguess, 2015).

One of the disadvantages of machine learning, it is argued, is that the results it provides are often difficult to interpret. It is also argued that they are sensitive to extreme values, resulting in conflicting forecasts (Bacham et al., 2017)

Neural networks, SVM models, classification trees and random forest seem to be the most researched algorithms in the credit risk analysis area.

2.6.1. Support Vector Machines (SVMs)

The "Support Vector Machine" (SVM) is a supervised machine learning algorithm used in classification problems. SVMs are based on the idea of finding a hyperplane that best divides a dataset into two classes, as shown in the image below.

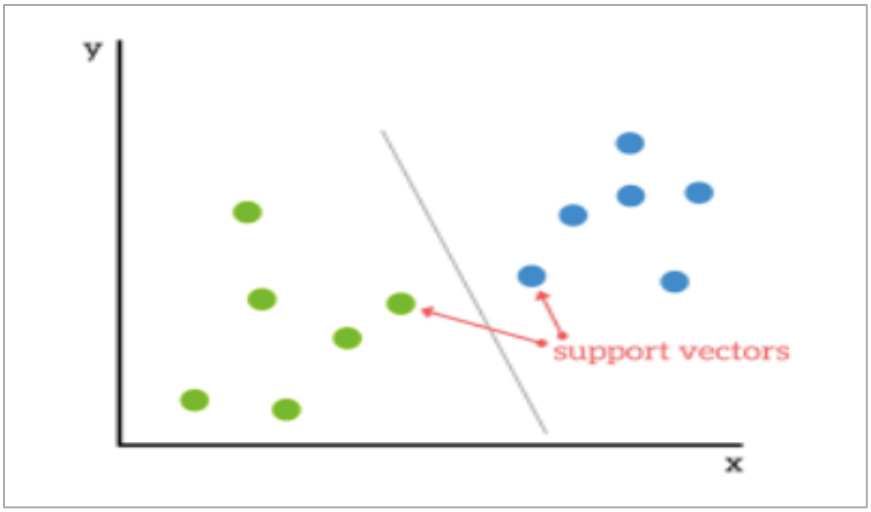

Figure 1: *support vector machines representation*

SVM has been applied to the design of credit risk and credit rating models (Bellotti and Crook, 2009, Cao et al., 2013, Van Gestel et al., 2003, Huang et al., 2007, Lai et al., 2007). Wang et al. (2005) present a weighted SVM model with promising results in credit risk analysis. Huang et al. (2007) developed a credit rating model to assess an applicant's credit score from input attributes based on a hybrid SVM.

Yeh and Lien (2009) recognized that predicting the likelihood of a customer's default is a challenge faced by researchers and needs further studying. Raei et al. (2016) investigated a new hybrid model for estimating the likelihood of corporate customers defaulting on a commercial bank. The research combines a two-stage approach, i.e., combining the predictive power of logit models and nonlinear techniques such as neural networks. The overall accuracy of this hybrid model proved to be superior to both basic models (Brown and Mues, 2012).

Banks seek to develop effective models that can assess the likelihood of default of their customers. (Barboza et al., 2017) try out machine learning models to predict bankruptcy one year before it happens, comparing the performance with the results of the traditional methods. They report significant prediction accuracy and also suggest that ML techniques can be easily applied for substantial classification accuracy compared to traditional methods. Despite concerns about the model's explanatory capacity ML models could be a significant help. A bank could benefit from the ability of a model to select the financial ratios that are most relevant to the forecasting process and also from the high level of their accuracy.

2.6.2. Decision trees

The decision tree is a structure of nodes and edges - based on which the population is classified using an explanatory variable (xi) at each node and making a decision about the different options (Breiman et al., 1984). The top of the tree is the root node, the next levels of nodes are the child nodes and at the bottom of the tree are the terminal nodes which describe the final classification (Anderson, 2007).

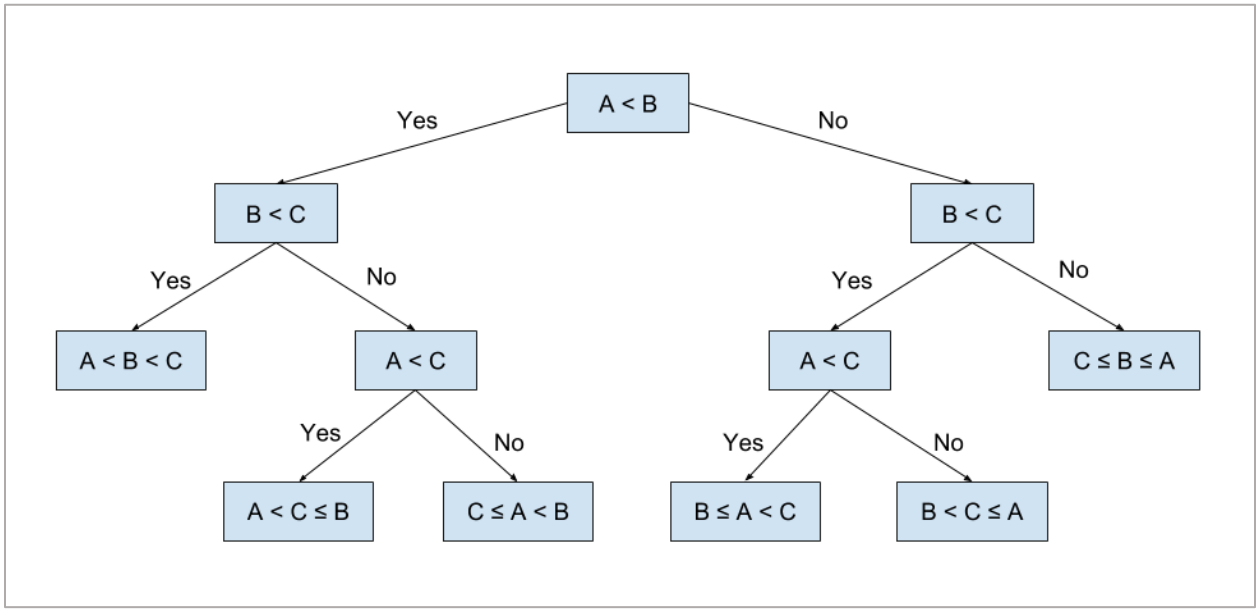

Figure 2: *Decision trees representation*

The way in which the decision is made and the criteria in which the breaking will take place, determines the type and complexity of the decision tree (Anderson, 2007). In its simplest form, especially when there is insufficient data available for analysis (e.g., medical data), the set of decisions and rules is determined empirically by a team of experts. Usually, however, analytical tools are used to make decisions and set the rules.

The best-known methods for applying credit risk are (Li & Zhong, 2012) CART (Classification and Regression Trees) by Breiman et al. (1984). The aim is to define homogeneous classes of the population, in terms of risk level, while at the same time to maximize the difference in risk levels between classes.

The most important advantages of using decision trees are mentioned below (Anderson, 2007 and Genriha and Voronova, 2012):

- As a non-parametric method, not many assumptions are required for its use.
- The calculations are usually simple and the selection of variables and breaks are done with a specific statistical measure.
- For simple trees, the results are transparent, interpretable and easy to implement.
- They can be a quick and easy way to identify very low or very high-risk borrowers.

On contrary, some main disadvantages of the decision trees are:

- They generally do not give as good estimates as regression models. In fact, they are selected mainly when there is a limited data availability.
- Often associated with overfitting issues.
- It is not as flexible as other methods, for example, neural networks.

2.6.3. Random forests

Random forests are the generalization of decision trees, where the estimate for each node is derived as the average of the estimates given for this node by a large set of random trees (Breiman, 2001).

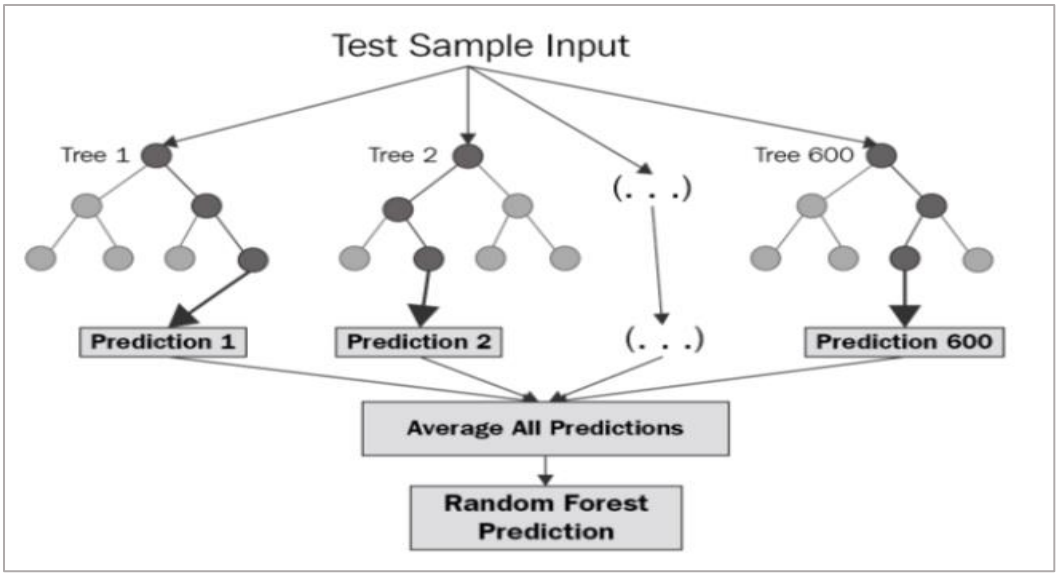

Figure 3: *Random forest representation*

In order to increase the degree of accuracy of the estimation, the individual trees that make up the forest should be as unrelated as possible. In random forests, the following three parameters (hyperparameters) should usually be defined (Beutel et al., 2019):

- The number of decision trees for the forest.
- The number of randomly selected variables to be examined in each breaking.
- N The minimum number of observations that each terminal node should have, which also determines the complexity of the trees.

Some of the crucial random forests' advantages are:

- Reduction of overfitting compared to decision trees.
- Modeling linear and non-linear relationships.
- Quite good and accurate estimates.
- High dimensionality.

On the other side, the main disadvantages are:

- There is no transparency and control over how the model works, except for the definition of parameters.

2.6.4 K-Nearest neighbors (KNN)

The K-Nearest neighbors (KNN) method (Cover and Hart, 1967) is a simple non-parametric technique which is based on machine learning and data mining and is used for classification and regression purposes. The aim is to categorize each observation in a group. To achieve this, the algorithm examines the k nearest neighbors of a new observation (test sample) that has not yet been categorized and then assigns this observation to the class that is most common to these neighbors.

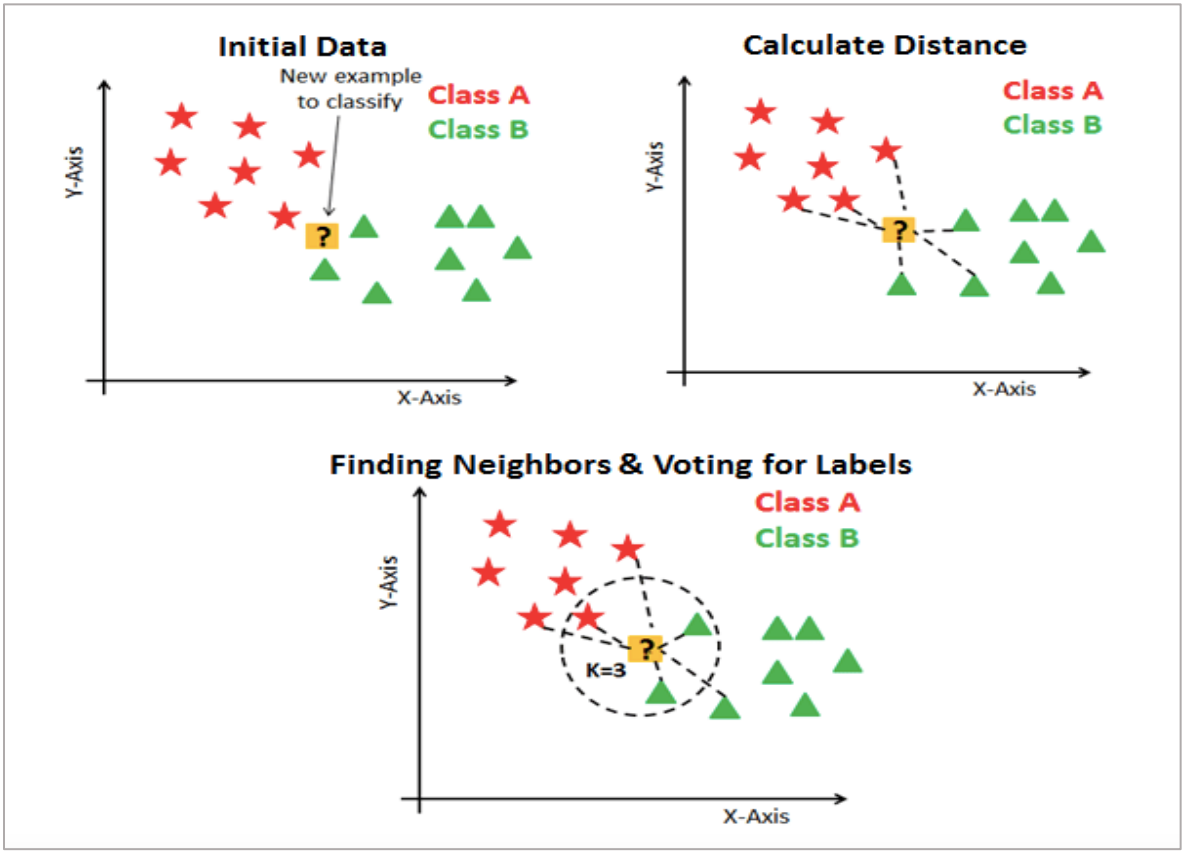

Figure 4: *KNN representation*

The distance between observations is defined by a metric (e.g., Euclidean distance, Minkowski, Manhattan, Euclidean, Cosine and Hamming). The method is simple and also a good approach in case the analyst wants to add new observations to the training sample quickly and easily. The main drawbacks of the method are (Anderson, 2007):

- The algorithm only does the classification but does not calculate final probabilities.
- There is no transparency in how decisions are made.
- The computational times may be long.
- Finally, the KNN algorithm is very much subject to the phenomenon known as 'curse of dimensionality' and as reported by Shalev-Shwartz & Ben-David (2014), the sample size is necessary to achieve a relatively small estimation error increases exponentially with the number of the predictive variables.

3. Data and Methodology

3.1. The dataset

The dataset that is going to be used for the purpose of this thesis can be found on Kaggle and it contains data for 32,581 borrowers and 11 variables (Age, Annual income, Home ownership, Employment length, Loan intent, Loan grade, Loan amount, Interest rate, Loan status, Percent income, Historical default, Credit history) related to each borrower.

The following table presents these variables and a simple description of each one of them.

Table 1: *Dataset variables*

Feature Name	Description
person_age	Age in years
person_income	Annual Income in dollars
personhomeownership	Home ownership
personemplength	Employment length (in years)
loan_intent	Loan intent
loan_grade	Loan grade
loan_amnt	Loan amount in dollars
loanintrate	Interest rate
loan_status	Loan status (0 is non default 1 is default)
loanpercentincome	Percent income
cbpersondefaultonfile	Historical default
cbpresoncredhistlength	Credit history length

Table 2: *Descriptive Statistics of Numerical variables*

Numerical variable	Mean	Min	Max
Age	27.7 (years)	20 (years)	84 (years)
Annual income	66,125.2 ($)	4,000 ($)	948,000 ($)
Employment length	4.8 (years)	0 (years)	41 (years)
Loan amount	9,655.4 ($)	500 ($)	35,000 ($)
Interest rate	11	5.4	23.2

Table 3: *Target variable*

Numerical variable	Non default	Default
Loan status	0	1

Table 4: *Categorical variables*

Categorical variable	Value
Home status	"rent", "mortgage", or "own"
Loan intent	"education", "medical", "venture", "home improvement", "personal" or "debt consolidation"
Loan grade	"A", "B", "C", "D", "E", "F" or "G"
Historical default	"Y" or "N"

3.2. Methodology

3.2.1. Steps

There are three categories of machine learning, which are supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, an input and an output variable are necessary, whereas a training set based on predefined inputs and outputs is used for teaching the models to predict the correct output in the future. In unsupervised learning, patterns are discovered from unlabeled data, while in reinforcement learning the learning system tries to learn through direct interaction with the environment. In this thesis, supervised learning algorithms were used. The steps for this classification problem are described below.

Data Acquisition: The most important and key element in analytics is the data that is used. The volume as well as the quality of data makes the analysis challenging, as the results and their accuracy are strongly affected by them.

Data Preprocessing: The data preparation refers to the transformation of raw data or even encoding in order to make it machine readable. It plays an extremely significant role in learning procedure. Handling missing or inconsistent values and outliers, encoding categorical variables, dimensionality reduction and features selection are some of the requirements. In this step, the creation of several graphs is very helpful, as meaningful insights can be provided.

Model selection: There is a variety of algorithms used in machine learning and each of them is more or less suitable across different problems. The scope is to identify the model that will lead to the highest accuracy. Another key question that needs to be answered before implementing an algorithm is whether the problem corresponds to classification or clustering. In this study, classification techniques were used.

Training: The original dataset is split into training and test data. The first is used to train a model, where the second is used to evaluate how accurate is the output prediction.

Evaluation: This step is very critical, as it depicts the model's performance in new cases that they were not part of the educational process. For this scope, several evaluation metrics are used according to the machine learning category that is used each time.

Hyperparameter Tuning: This process refers to determining which configuration of hyperparameters leads to the best performance. Except from the model parameters, there is another kind, which called "hyperparameters", that is not possible to be learned during the training process and they contain useful information regarding the model's complexity or how fast is its ability to learn. For example, a hyperparameter is the k in the KNN algorithm.

Prediction: After all the above-mentioned processes, the trained model is now ready to predict the most likely output value given a specific input using real time data.

3.2.2. Python and scikit-learn library

In the present work, the data analysis and the model's development are done using Python as the programming language and the scikit-learn library for the models.

Machine learning continues to grow rapidly. Artificial Intelligence makes it possible to create innovative solutions to real problems, such as fraud detection, personal assistants, spam filters. The need for intelligent solutions to real-world problems leads to the further development of artificial intelligence in order to automate tasks that are difficult for programming without AI. The Python programming language is considered the best algorithm for automating such tasks and offers greater simplicity than other programming languages.

Scikit-learn library is the best python's free machine learning library that can be used to simplify the task of coding and implementing Machine Learning algorithms. Scikit-learn includes a number of different machine learning algorithms, such as random forest, SVM and KNN.

It is a collection of the most effective tools for statistical modeling and machine learning. Some of these tools include regression, classification, clustering, dimensionality reduction. It is primarily written in Python and is based on SciPy, NumPy and Matplotlib libraries. It has been developed by David Cournapeau in 2007 as part of Google Summer Code. Subsequently, Gael Varoquaux, Fabian Pedregosa, Alexandre Gramfort and Vincent Michel, from the French Institute for Computer Science and Automation Research, released a beta version of v0.1 in 2010. Since then, newer versions have been released. Scikit-learn is a community-based project where everyone can contribute to their development.

Scikit-learn has some main advantages:

- The library is distributed free of charge with minimal legal restrictions and licensing restrictions.
- It is easy to use.
- It is very flexible and serves real purposes such as predicting consumer behavior.
- It is supported and updated by many partners in the international online community.

3.2.3. Data preprocessing

As the first step of the analysis, the dataset is being checked for missing values:

person_age	0
person_income	0
person_home_ownership	0
person_emp_length	895
loan_intent	0
loan_grade	0
loan_amnt	0
loan_int_rate	3116
loan_status	0
loan_percent_income	0
cb_person_default_on_file	0
cb_person_cred_hist_length	0

In the dataset there are two fields with null values. Employment length contains 895 null values and Interest rate contains 3116 null values.

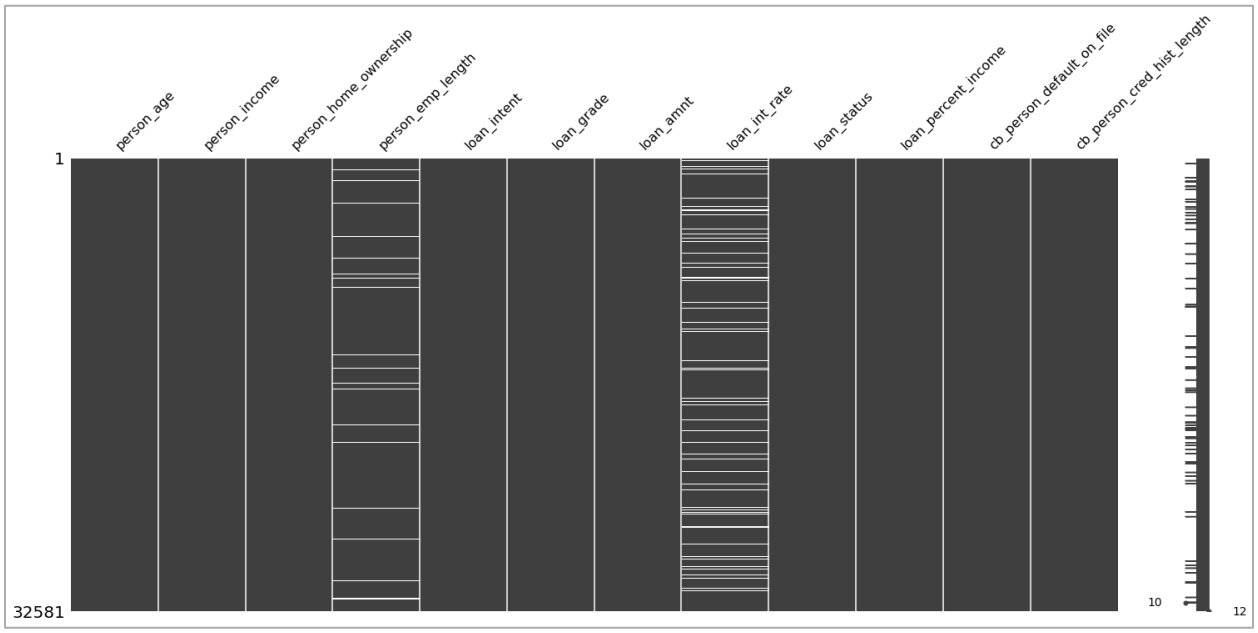

Figure 5: *NA values check*

After identifying the missing values that are contained in the dataset, we are replacing them with the mean value of each variable.

3.2.4. Exploratory Data Analysis

In the past few years, a new sector of Data Analysis has emerged. Exploratory Data Analysis (EDA) is the part of Data Analysis that explores raw or structured data and visualizes it through multiple kind of plots, graphs and techniques. It is essential in a real-life machine learning or data analytics projects to investigate the characteristics and the information a dataset gives, with the ultimate goal of knowing the form, the type, the distribution, the correlations and the basic statistics of the data from an analyst.

- Numerical Data

For the numerical data of the dataset, histograms are created. A histogram is a graph that displays numerical data in groups of bars with different height. Grouping is based on the different values of the variable – feature is designed and its range. Except of being a method to plot the values of a variable, histograms help gaining information about the distribution of it, as well as the frequency of exact several values.

Below are displayed the histograms of numerical the following features: person_age, person_income, person_emp_length, loan_amnt, loan_int_rate, loan_status, loan_percent_income and cb_person_cred_hist_length:

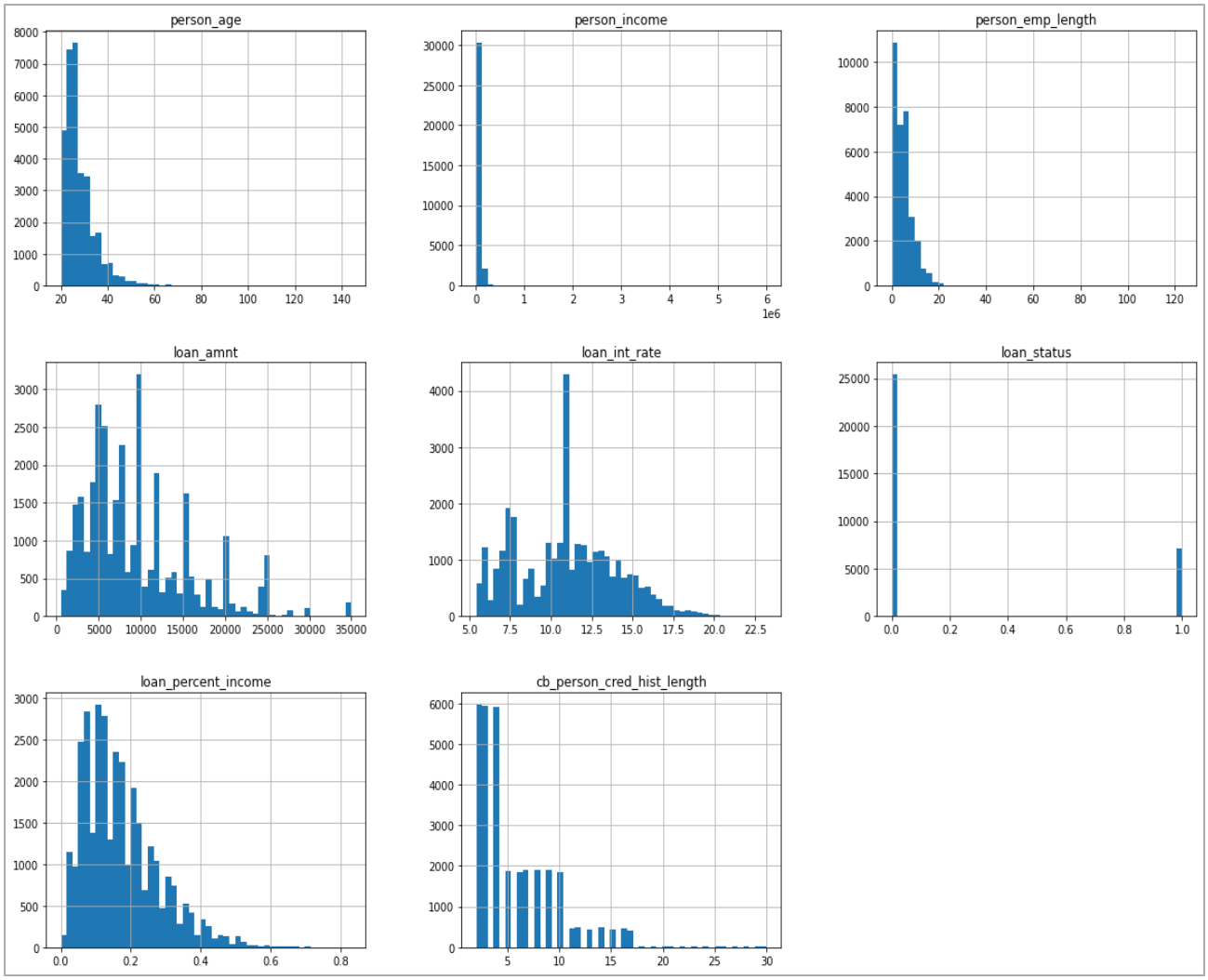

Figure 6: *Histograms of Numerical Data (Distribution and values)*

- Categorical Data

Following the analysis of the categorical data, we are creating graphs that have similar meaning and practical application to histograms for numeric features. These graphs are named count plots and are bar graphs. Basically, they count the values of the categories for each one of them, so again the height of a bar diverges as in the histogram. Each bar represents one category of the categorical variable.

A count plot of the categorical variable person_home_ownership is presented below, where it is obvious that the two main types of customers who request a loan, have a rent or mortgage.

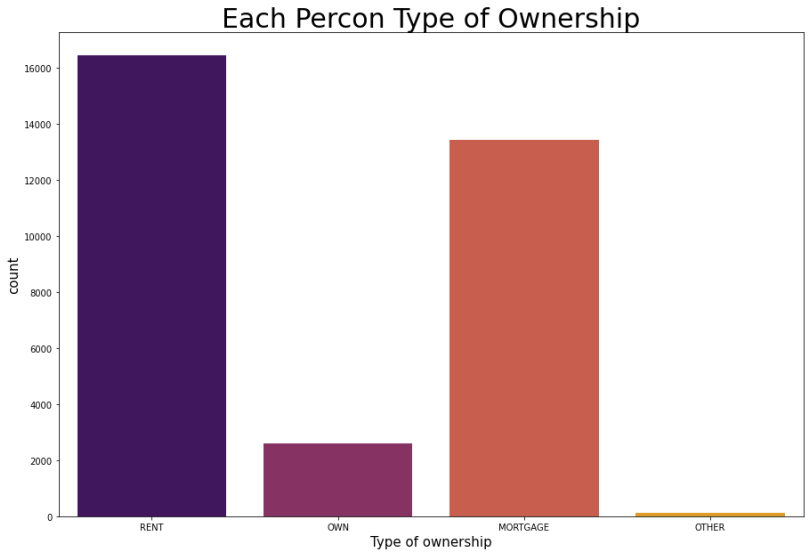

Figure 7: *Count plot – Bar graph: Type of ownership*

Furthermore, the following graph is a count plot of the categorical variable loan_intent, which reveals that the customers request a loan mainly for educational and medical purposes, while the rest reasons are coming next with no such a big difference.

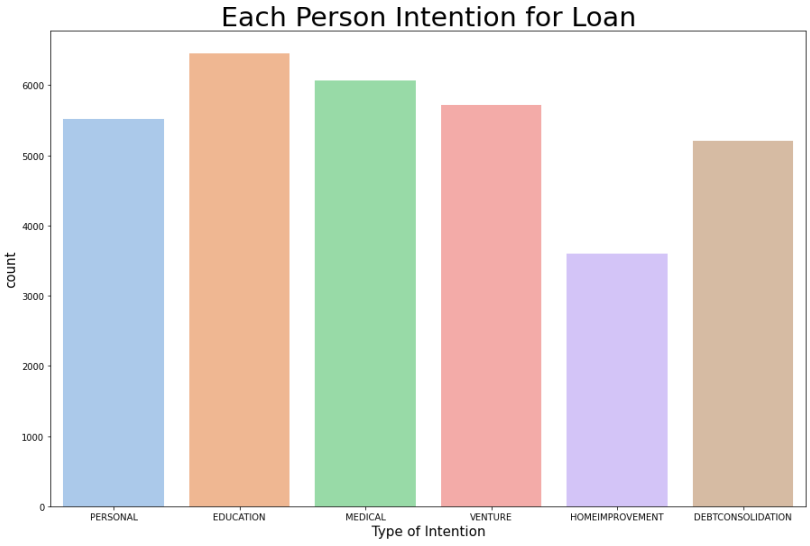

Figure 8: *Count plot – Bar graph: Type of intention*

In addition, the grade of loan per person (loan_grade) is displayed with the following count plot, in which, customers with grades B and D are greater than 20,000 of the totals.

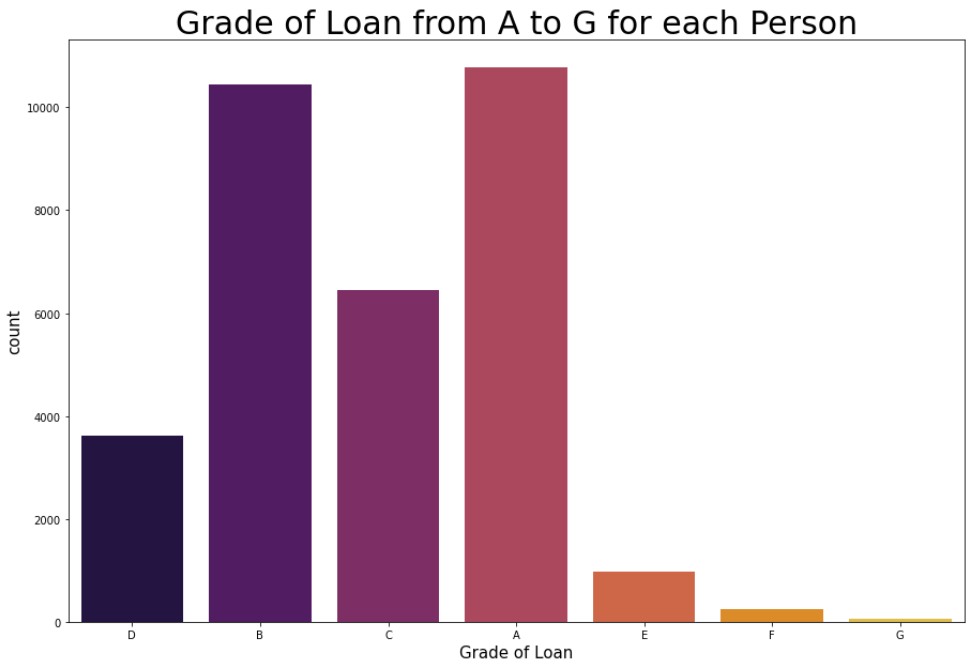

Figure 9: *Count plot – Bar graph: Grade of loan*

A count plot of the categorical variable cb_person_default_on_file is also presented below.

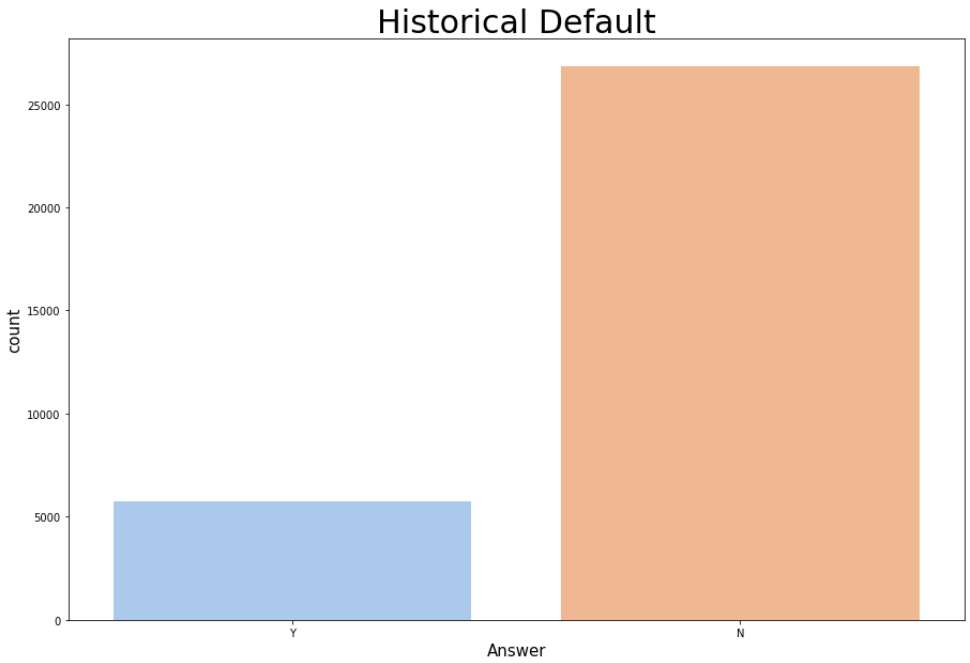

Figure 10: *Count plot – Bar graph: Historical default*

- Mixed Plots

The following charts are comparative bar graphs (like count plots) of the categorical data, but this time compared to the 2 classes (0-1) of the target.

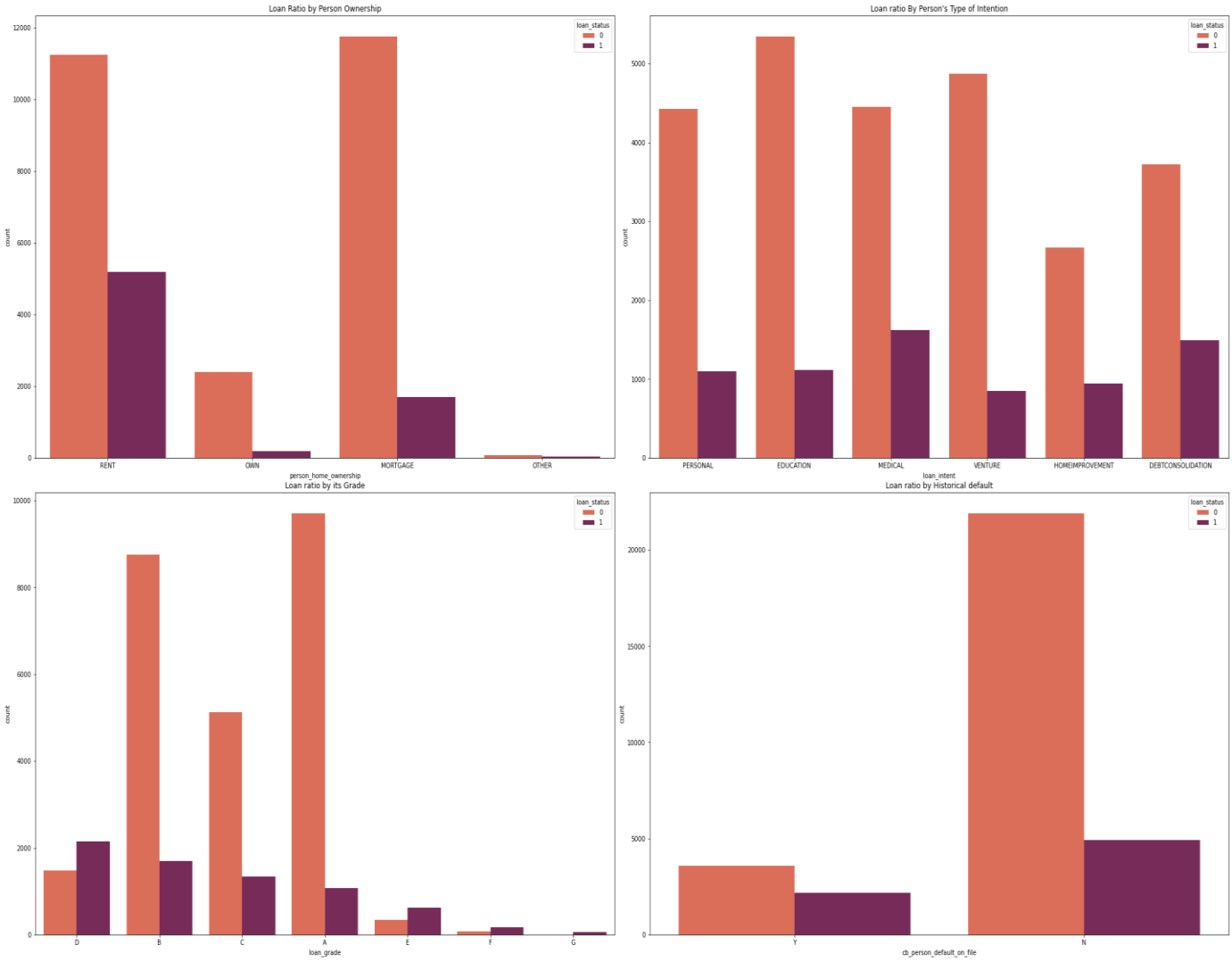

Figure 11: *Bar graph – Comparison of Class variable values in categorical features*

In the continue, three violin plots are shown. Violin plots are quite useful, because they give us information about the values, the distribution and the range of a categorical variable compared to the values and distribution of a numeric variable.

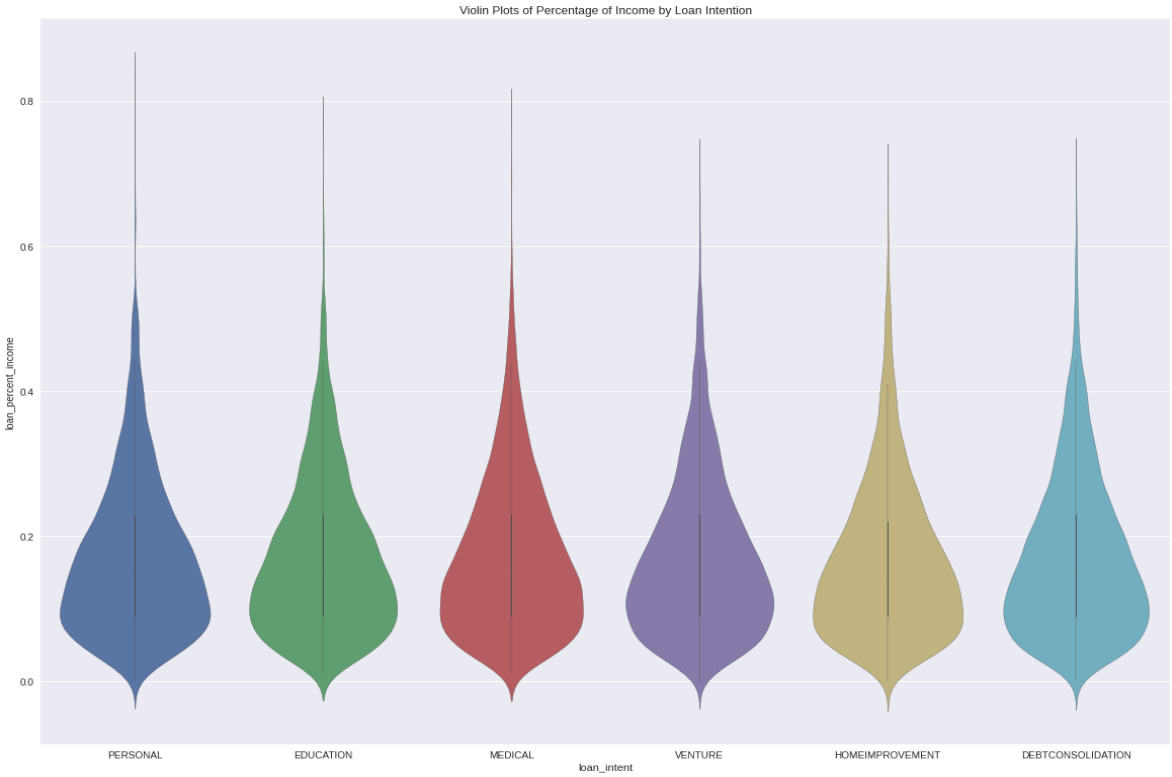

Figure 12: *Violin plot of the Percentage of personal Income by the type of Loan intention*

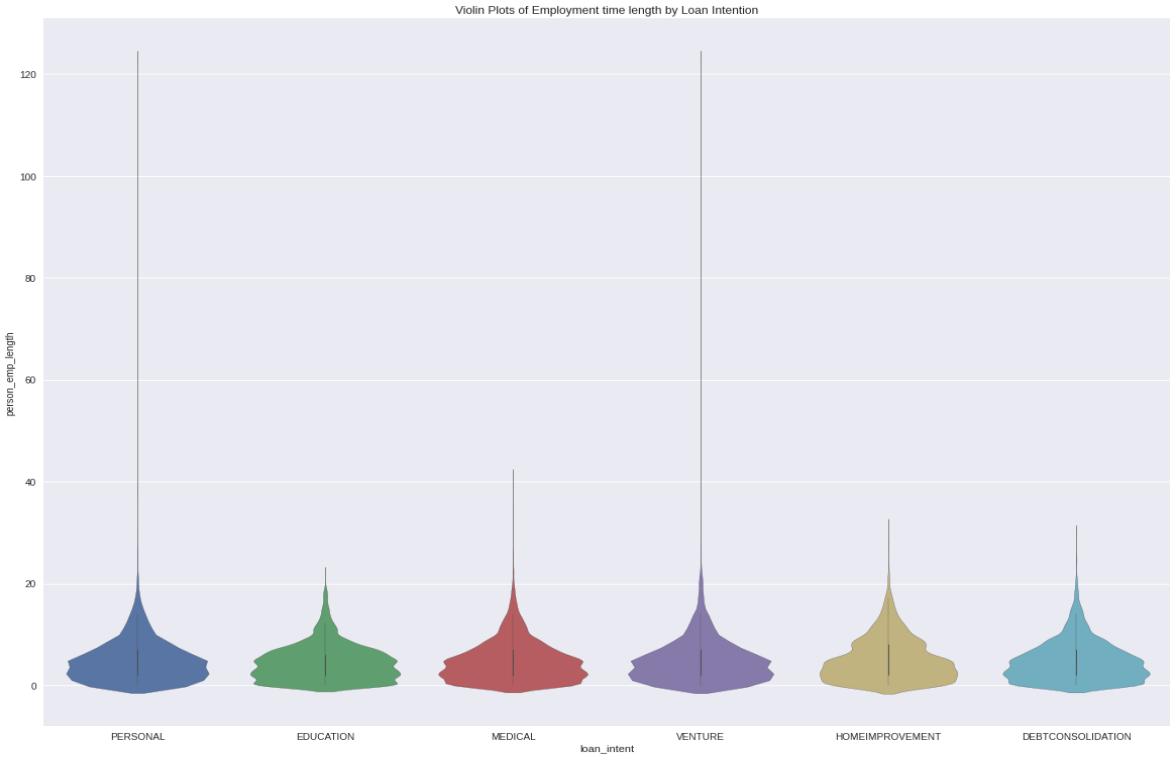

Figure 13: *Violin plot of the Employment Length (in Years) by the type of Loan Intention*

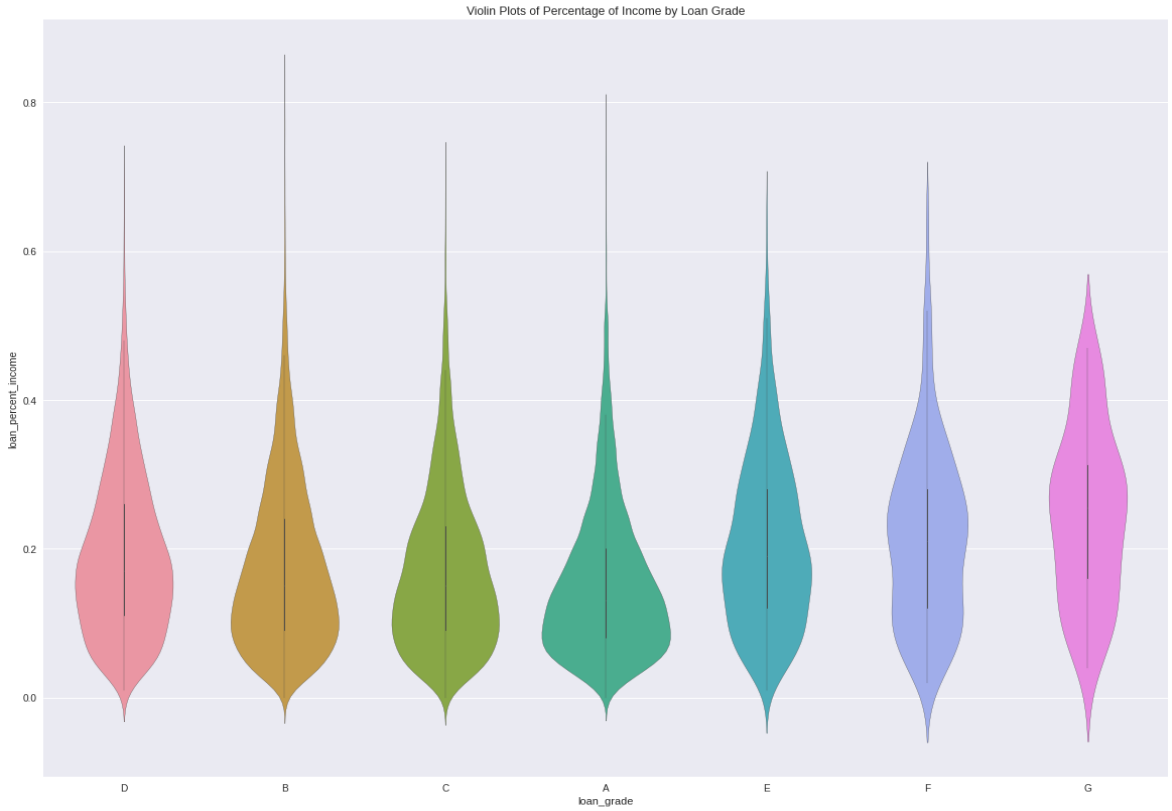

Figure 14: *Violin plot of the Percentage of personal Income by the type of Loan Grade*

- Correlation Plots

In order to see if the variables in the dataset correlate with each other, the creation of two last graphs is necessary. The first refers to a correlation matrix that has been created in a heatmap plot. The second is a pair plot, which is a very useful graph for observing distribution of each variable in the diagonal and correlations on all other cells, both in comparison with the 2 groups of class variable.

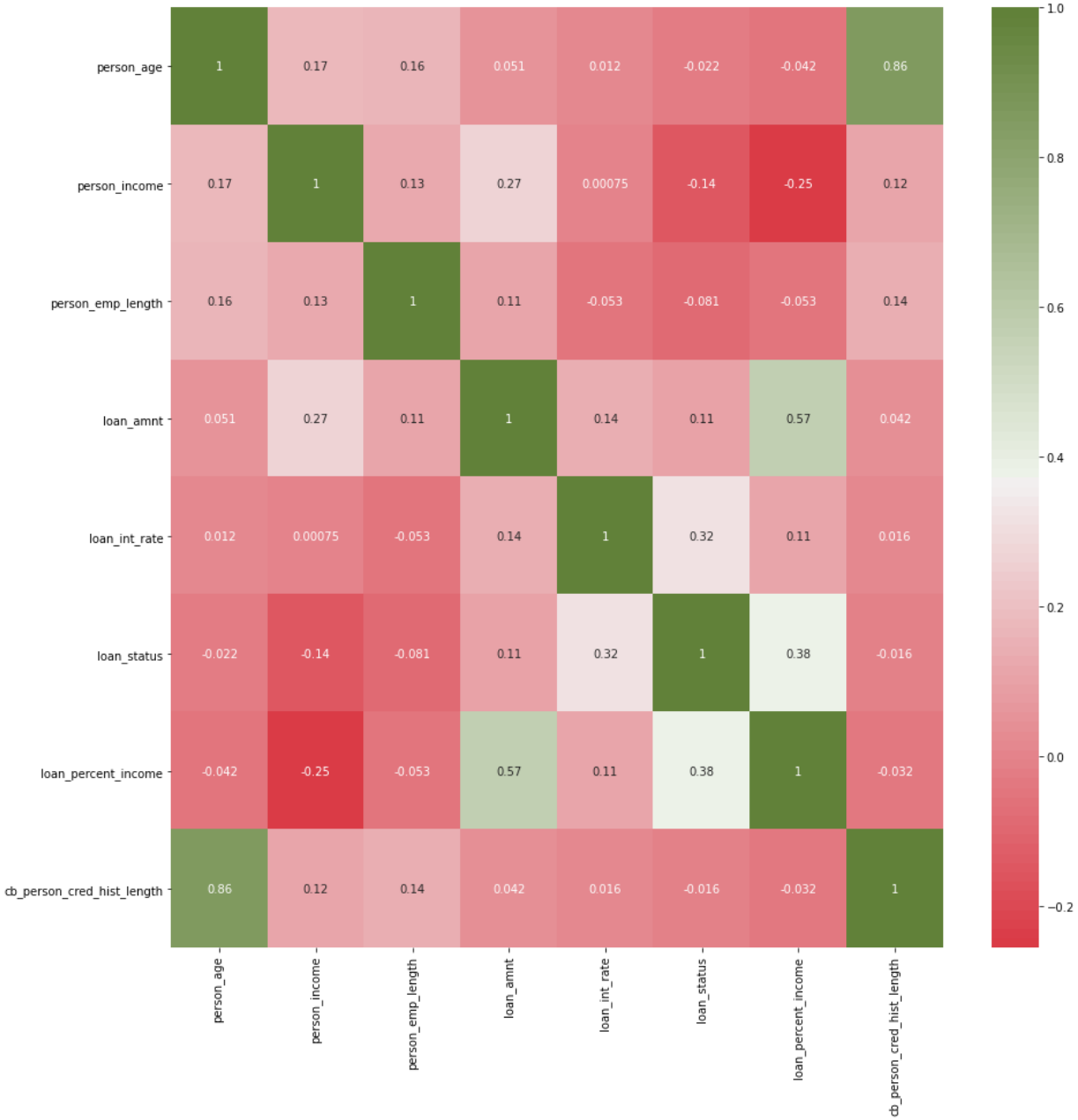

Figure 15: *Correlation matrix (Heatmap plot)*

Each square shows the correlation between the variables on each axis. Correlation ranges from -1 to +1. Values closer to zero means there's no linear trend between the 2 variables. The closer to 1 the correlation is, the more positively correlated they are and the stronger their relationship is. This means that if one increases, the other will increase too. A correlation closer to -1 is similar, but instead of both variables, if one variable increases, the other will decrease. The diagonals are all equal to 1 because those squares are correlating each variable to itself. The correlation between two variables is higher when the number in the square is larger and the color is darker. The plot is also symmetrical about the diagonal since the same two variables are being paired together in those squares. Thus, the correlation matrix shows that there is not any significant correlation between the variables. There is a low correlation between loan amount and personal income which is not surprising.

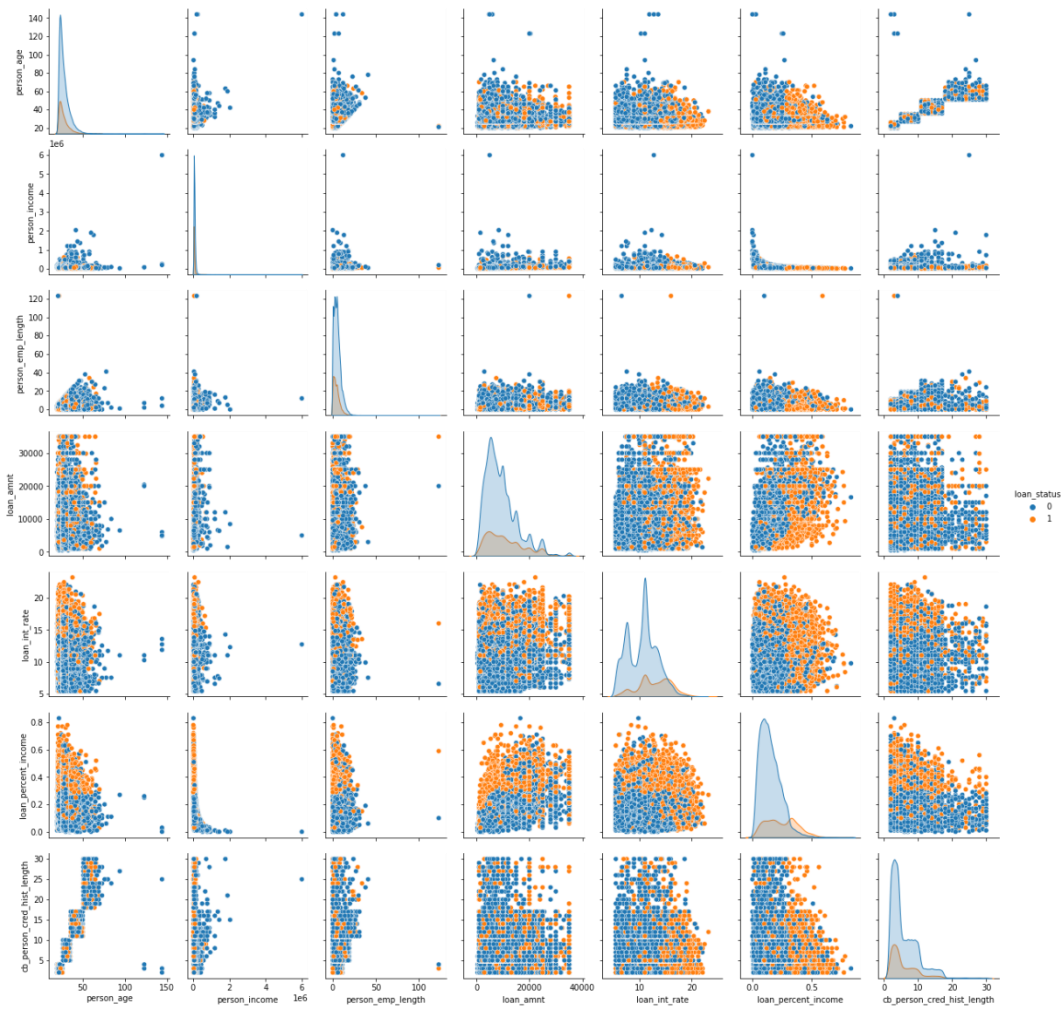

Figure 16: *Pair plot Graph*

The most critical factor affecting the success of machine learning procedure is the training and testing process. An effective training process improves the quality of the model created. Datasets divide into two parts for training and testing according to some rules. After the machine learning model is trained based on the training data, it has to be tested using the testing data. In the case that is being examined in this thesis, before the training process, the categorical variables have to be converted – labeled into numerical, and that can be done by using the label encoding method. Label encoding is a machine learning encoding technique that allows us to convert categorical and text data (columns) to numerical, so that they can be used in the machine learning models. Label encoder just encodes categorical data to a number for each of these values from the columns. Python and scikit-learn have multiple easy ways to do this to a dataset automatically. Another famous encoder is One-Hot-Encoder. After the last conversion we make, we get the last form of the dataset for use which is described perfectly by following heatmap – information map and the pair plot graph.

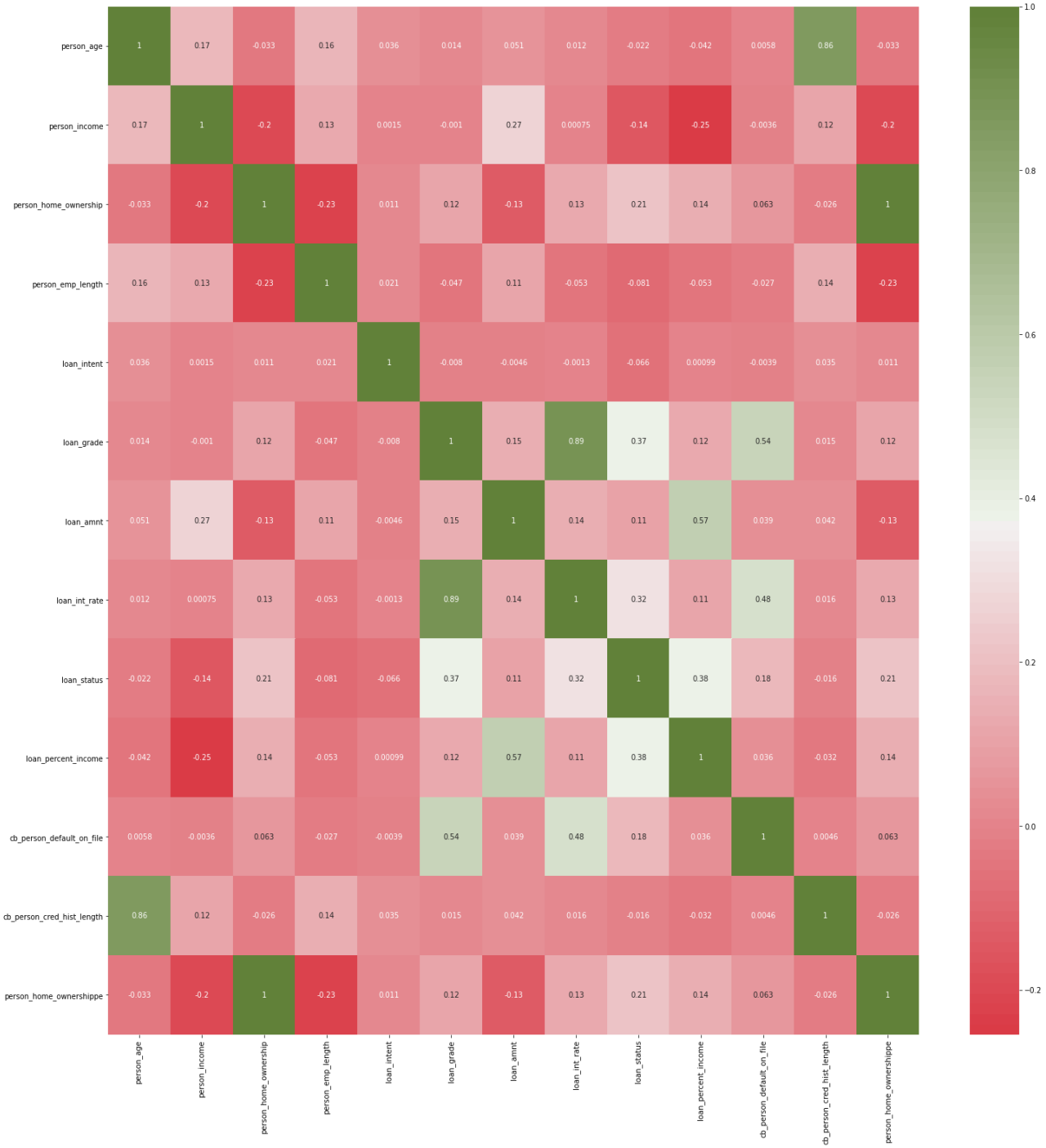

Figure 17: *Final Heatmap (Correlation Matrix)*

From the heatmap above that corresponds to the final dataset, we can make some conclusions about the correlations between the variables and the information gained from them. Some of these conclusions are described below:

- In the diagonal of the heatmap, the only values we get equal to 1. This happens because correlation between a variable and the same (2 times) is 100%.
- In the squares with light green color, there is a very good relationship and correlation between the two characteristics. For example, loan_amnt and loan_percent_income have high correlation equals to 0.57, which means that a good information is provided about the data regarding these two variables. Another example of same pair of variables is loan_int_rate with loan_grade. There are many other pairs of good correlations as it can be observed in this graph.
- In the squares with values less than 0, a useful outcome cannot be retrieved. Negative correlation means that the correlation between the two variables is bad and there is no information given from them. For example, person_income with loan_percent_income have a value equal to -0.25.
- Finally, for the light pink squares it can be highlighted that they are not bad, but not good enough as well to draw a conclusion about the information might be given from these variables.

The pair plot that corresponds to the final dataset is shown below.

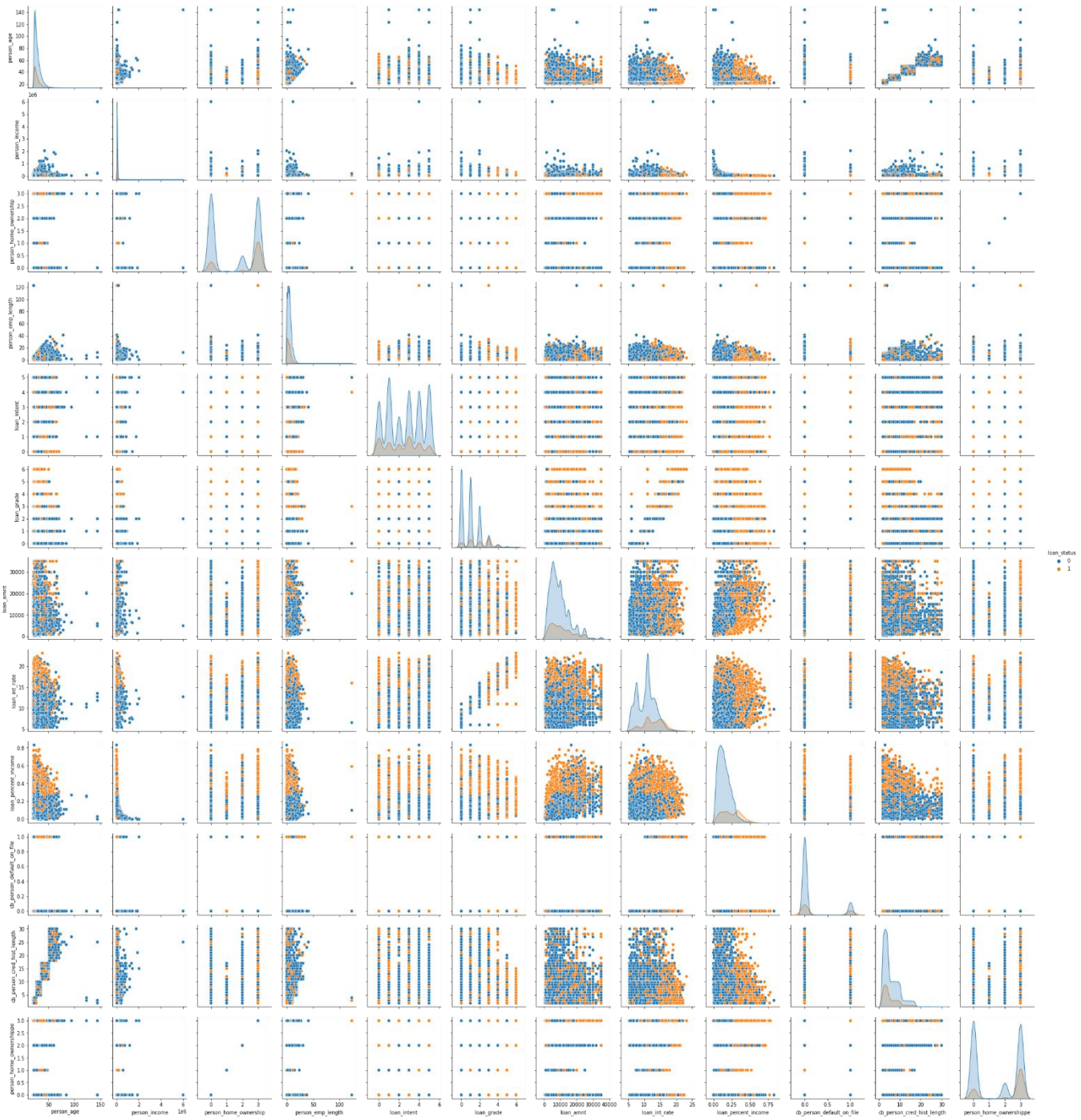

Figure 18: *Final Pair plot Graph*

Last but not least, before creating and evaluating the machine learning models, the dataset is split in two parts, the train set which contains the 80% of the initial dataset and the test set which contains the rest 20% of the observations. Using the function stratify (stratify = y) in training, we take the advantage of the usage of keeping the same rate of each group of the class variable in the training and test sets. As a result, it is guaranteed that the percentage of values from the two classes (0-1) of the target group is the same in both training and test set.

3.2.5. Evaluation – Metrics

In this chapter, the results of the machine learning models that have been created before are analyzed. In order to do this, it is necessary to describe briefly the evaluation metrics that the analysis is based on.

- **Confusion Matrix**

			Predicted values		
			Positive	**Negative**	Totals
	Actual Values	**Positive**	TP	FN	**P = (TP + FN) = Actual Total Positives**
		Negative	FP	TN	**N = (FP + TN) = Actual Total Negatives**
		Totals	Predicted Total Positives	Predicted Total Negatives	

Figure 19: *Confusion matrix*

Confusion Matrix is by far the most important tool for evaluating classification machine learning models. It is a matrix that represents the actual values (the real values of the dataset) in y axis in comparison with the predicted values (the values that were predicted by the models) in x axis. Both actual and predicted values are separated to positive and negative class (1-0). The meaning is that

the class variable is binary and takes two values, 0 for negative fact-class and 1 for positive fact-class. As a result, the matrix that is created has four squares and it contains four groups of values. These values are the following:

TP (True positive): Actual value and predicted value are positive (1-1), meaning that model's prediction was correct (correct classification).
FP (False Positive): Actual value is negative, but predicted value is positive (0-1), meaning that model's prediction was wrong (wrong classification).
FN (False Negative): Actual value is positive, but predicted value is negative (1-0), meaning that model's prediction was wrong.
TN (True Negative): Actual value and predicted value are negative (0-0), meaning that model's prediction was correct.

TP, FP, FN and TN values can be written either as a specific number of all Test set inputs, for example the number of TP predictions in Test set, or as a percentage of each square.
Confusion matrix is usually represented as a heatmap.

- **Precision and Recall**

The mathematical formulas of these two metrics are:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} = \frac{True\ Positive}{Total\ Predicted\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ negative} = \frac{True\ Positive}{Total\ Actual\ Positive}$$

Precision is a metric that shows how accurate the model is. In other words, it measures how many of the observations are actual positive out of the predicted positive. It is a good measure to determine, when the costs of false positive are high.

Recall measures how many of the actual positives the model predicts it as positive (True Positive). This evaluation metric should be chosen as the model's metric, when there is a high cost associated with false negative.

- **F1 Score**

Below is the mathematical formula for F1 score:

$$\text{F1 Score } = 2 * \frac{Precision * Recall}{Precision + Recall}$$

F1 score provides a single score to measure both precision and recall.

- **ROC – AUC score and ROC curve**

The Receiver Operator Characteristic Curve or simpler ROC curve is an evaluation metric curve for binary classification problems (classification problems where class variable is binary). It is a probability curve that plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold values and essentially separates the 'signal' from the 'noise'. The Area Under the Curve (AUC) or ROC-AUC score is a probability that defines the measure of the ability of a classifier to distinguish between the 2 classes and is used as a summary of the ROC curve. This is also known as ROC-AUC score and is another powerful accuracy score for evaluating Classification Machine Learning Models.

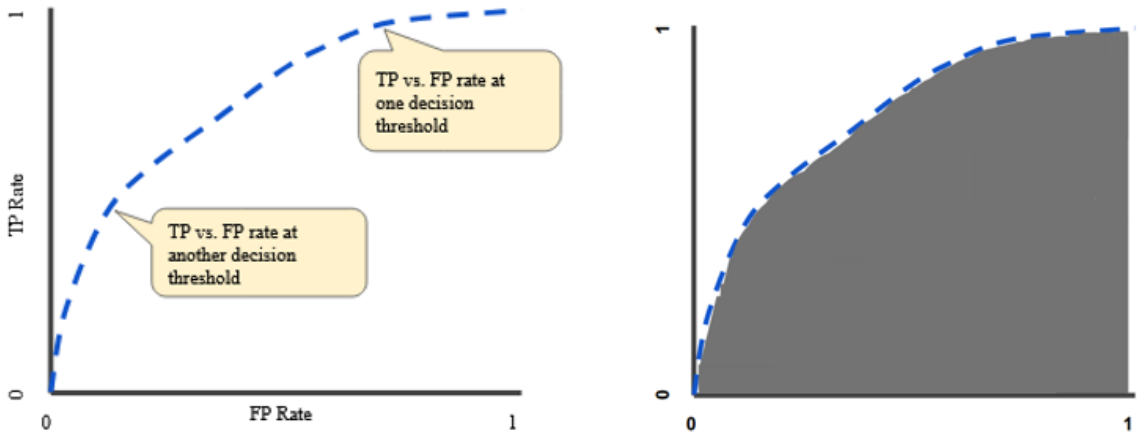

Figure 20: *ROC curve and ROC-AUC*

Based on the evaluation metrics that were used in this thesis, the Random Forest model performs better than the others on the given imbalanced dataset. In the case that is examined, recall is a more important metric to take into account given that the problem contains false negatives values, meaning that the model predicts that someone is not going to default but they do. Having said that, the decision tree performs better in this metric, but the Random Forest has an overall better performance in all the metrics.

3.2.6. Hyperparameter tuning

The best way to think about hyperparameters is like the settings of an algorithm that can be adjusted to optimize performance, just as the knobs of an AM radio are turned to get a clear signal. While model parameters are learned during training, hyperparameters must be set by the data scientist before training.

To improve the KNN's model performance grid search is the most common approach. It exhaustively searches through all possible combinations of hyperparameters during training the phase. For example, consider a KNN model. We can specify a grid of number of neighbors (K = 1, 2, 3) and two metrics (p=1, 2). The grid search starts training a model of K = 1 and p=1 and calculates its accuracy score. In the sequence, it continues to train models of (K = 2, p = 1), (K = 3, p = 1), (K = 1, p = 2), ..., and (K = 3, p = 2) and obtains their score values. Based on the accuracy

scores, the grid search will rank the models and determine the set of hyperparameter values that give the highest accuracy score.

Using *grid search*, it can be seen that the best number of neighbors is 13, while the optimal distance metric is *Euclidean* and p=2.

For the random forest's classifier, the two main hyperparameters are:

- *n_estimators:* The n_estimator parameter controls the number of trees inside the classifier. Using many trees to fit a model is not always the best case. It may not cause any overfitting but it can certainly increase the time complexity of the model. The default number of estimators is 100 in scikit-learn.
- *max_depth:* It governs the maximum height up to which the trees inside the forest can grow. It is one of the most important hyperparameters when it comes to increasing the accuracy of the model. It is important to set its value appropriately to avoid overfitting. The default value is set to None.

In the Random Forest model, the default parameters are performing better than any other case.

For the decision tree's classifier, the main hyperparameters are:

- *Criterion:* The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.
- *max_depth:* The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- *max_features:* The number of features to consider when looking for the best split:
- *min_samples_leaf:* The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model.

3.2.7. SMOTE: Synthetic Minority Oversampling Technique

Resampling data is one of the most common approaches to use when there is an imbalanced dataset. There are two main methods for this: Undersampling and Oversampling. Most of the times, oversampling is preferred over undersampling techniques. The reason is that, in undersampling, instances that may be carrying some important information tend to be removed from the data.

SMOTE method is an oversampling technique where the synthetic samples are generated for the minority class. This algorithm helps to overcome the overfitting problem caused by random oversampling. It focuses on the features to generate new instances with the help of interpolation between the positive instances that lie together.

In order to see if oversampling gives a better machine learning model, the SMOTE implementation is used, provided by the imbalanced-learn Python library in the SMOTE class. The SMOTE class acts like a data transform object from scikit-learn and the process follows the steps: first it must be defined and configured, then it fits on a dataset, and finally it applies to the old dataset and creates a new transformed version of the dataset.

More specifically:

- A SMOTE instance is defined, with default parameters that will balance the minority class and then create a transformed version of the dataset.
- Then the same models and the same evaluation methods will be tried, using a SMOTE transformed version of the dataset.

3.2.8. Feature Importance Techniques

Nowadays, in machine learning projects is essential to extract information for the features of the data. Sometimes, some features have no contribution in predictions and modelling and they are practically useless. On the contrast, some features may contain the biggest amount of information for the dataset given. In this case, we have to construct models training only these essential characteristics. In this dissertation, we used permutation importance technique for KNN algorithm and we implemented the default model's importance for Decision Tree and Random Forest. As a conclusion, most informative features are described and summarized in the following table:

Table 5: *Feature Importance*

Feature	Permutation Importance (KNN)	Decision Tree Importance	Random Forest model Importance
person_age	X		
person_income	X	X	X
person_home_ownership	X	X	X
person_emp_length	X		
loan_intent		X	
loan_grade		X	X
loan_amnt	X		
loan_int_rate			X
loan_percent_income		X	X
cb_person_default_on_file			
cb_person_cred_hist_length			

From the table above it can be concluded that the variables – characteristics which are the most important and carry the biggest amount of information of this dataset are the person_income, person_home_ownership, loan_grade and loan_percent_income.

In the continue, the machine learning procedure that is followed in this study is described. 12 machine learning classification models have been created using 3 different algorithms: Decision Trees, Random Forest and KNN. In each algorithm, four different models have been developed and the order that has been followed, is displayed below.

1) Initial Model - For each machine learning algorithm, an initial model is created and the parameters that are used are only the default.
2) Model with best parameters (after hyper tuning) - After creating the initial model, the next step is to search for the best parameters that maximizing the performance of it via Randomized Search CV technique of hyper tuning procedure. After finding these hyperparameters, they are used to create and evaluate a new second model.

3) Model after feature importance implementation - The third model of each algorithm is a model being trained, tested and evaluated resulting to which features are best for this algorithm.
4) Model using SMOTE method - The final model of each algorithm is a model being trained, tested and evaluated to SMOTE – oversampled dataset, considering that the original data is imbalanced in class variable. This means that the volume of the data points of each class is not the same.

4. Empirical Results

4.1 Decision Tree Classifier Results

The best decision tree model that concluded from this analysis was the one that was built on Smote dataset. The best parameters which were identified are the following:

1) criterion='gini'
2) min_samples_leaf=1
3) min_samples_split=2

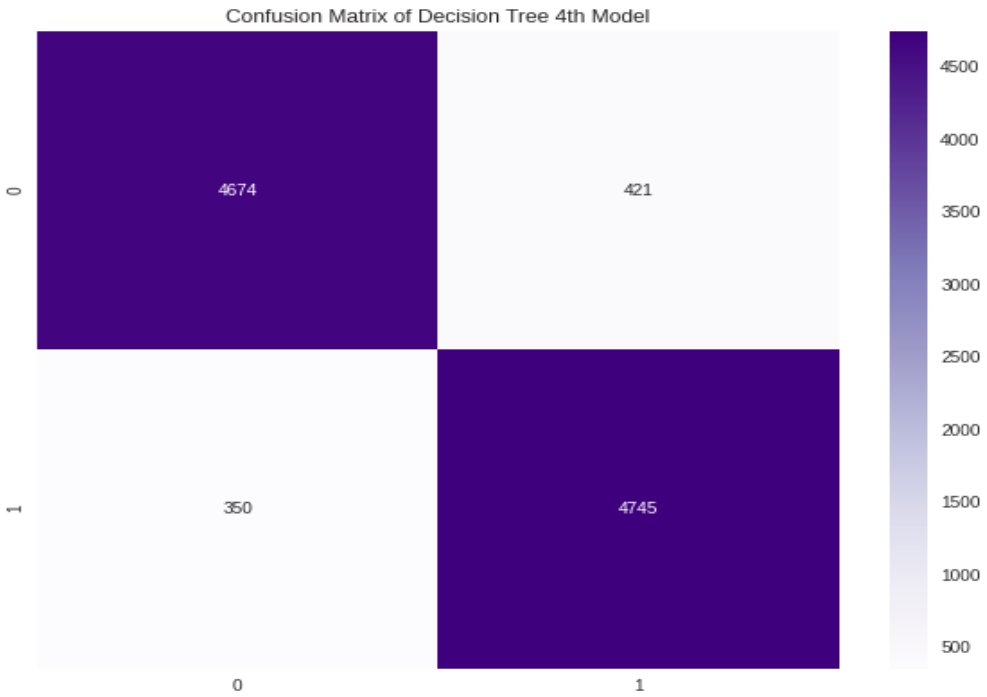

Figure 21: *Confusion Matrix of Decision tree Smote model*

From the confusion matrix above, it can be concluded that from 10,190 people, **9,419** were predicted in the correct class. Furthermore, **4,674** people were classified in class 0, which means that they would take a loan and that was correct as they did take the loan, while **4,745** classified in class 1, which means they would not take a loan and actually they did not take it.

Table 6: *Full Classification Report – Decision Tree*

	Precision	Recall	F1-score	Support
0	0.93	0.92	0.92	5,095
1	0.92	0.93	0.92	5,095
Accuracy			**0.92**	10,190
macro avg	0.92	0.92	0.92	10,190
weighted avg	0.92	0.92	0.92	10,190

Table 7: *Accuracy score vs ROC-AUC score – Decision Tree*

Accuracy score	ROC-AUC score
92.4337%	92.4337%

Table 8: *Comparison Results for All Decision Tree Classification Models*

Scores	Initial model	Hypertuned model	Feature Selected model	Smote Data model
Accuracy	88.8752%	**91.1155%**	89.0900%	**92.4337%**
ROC-accuracy	84.6470%	80.3764%	84.6070%	924337%

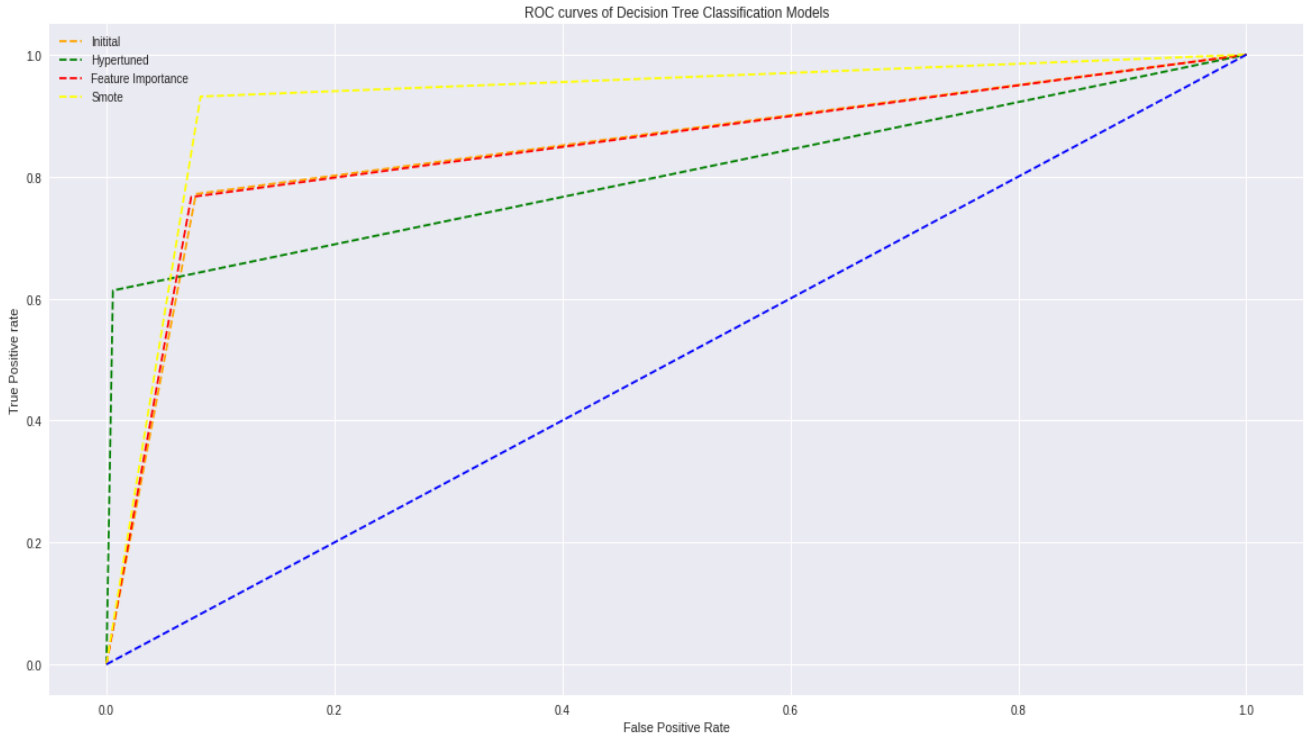

Figure 22: *Comparison of ROC curves for all Decision Tree Models*

Overall, from decision tree algorithm two models have been created with very high accuracy – over 90%, which means that decision tree classification models are quite efficient for credit risk analysis with machine learning modelling.

4.2 Random Forest Classifier Results

The best random forest model of the four that have been created was the one that was built on Smote dataset. The parameters which led to the best performance were the following:

1) criterion='gini'
2) min_samples_leaf=1
3) min_samples_split=2
4) n_estimators=100

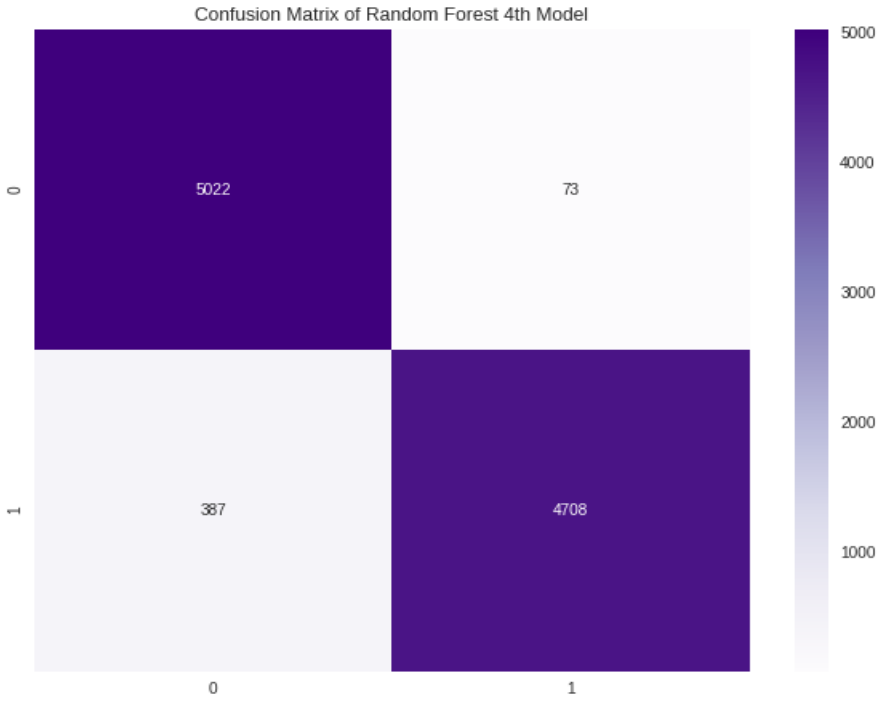

Figure 23: *Confusion Matrix of Random Forest Smote model*

From the above confusion matrix, it can be resulted that from 10,190 people, **9,730** were predicted in the correct class. Furthermore, **5,022** people were classified in class 0, which means that they would take a loan and that was correct as they did take it, whereas **4,708** classified in class 1, which means they would not take a loan and actually they did not take it.

Table 9: *Full Classification Report – Random Forest*

	Precision	Recall	F1-score	Support
0	0.93	0.99	0.95	5,095
1	0.98	0.92	0.95	5,095
Accuracy			**0.95**	10,190
macro avg	0.96	0.95	0.95	10,190
weighted avg	0.96	0.95	0.95	10,190

Table 10: *Accuracy score vs ROC-AUC score – Random Forest*

Accuracy score	ROC-AUC score
95.4858%	95.4858%

Table 11: *Comparison Results for All Random Forest Classification Models*

Scores	Initial model	Hypertuned model	Feature Selected model	Smote Data model
Accuracy	93.2484%	**90.1028%**	90.7320%	**95.4858%**
ROC-accuracy	85.5467%	79.5260%	83.7812%	95.4858%

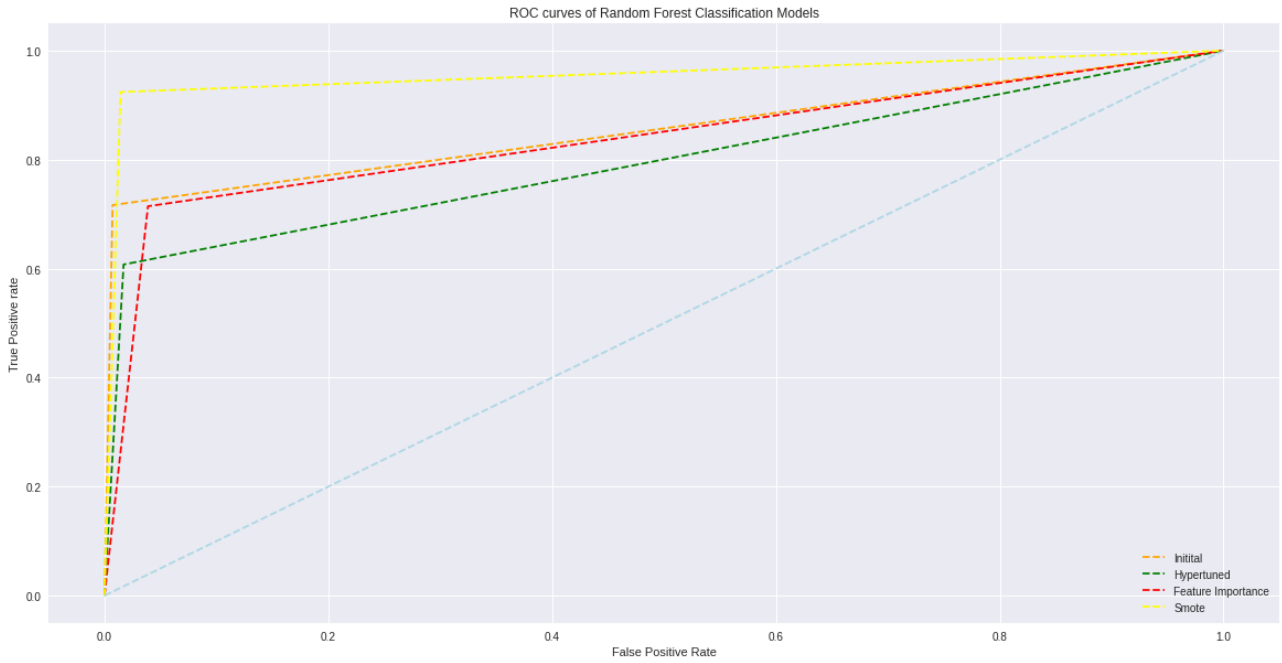

Figure 24: *Comparison of ROC curves for all Random Forest Models*

Overall, from random forest algorithm all the four models that have been created, had very high accuracy – over 90%, which means that random forest models are the most efficient for credit risk analysis using machine learning techniques.

4.3 KNN Classifier Results

The best KNN model of the four that have been created was the one that was built with the best parameters after hyper tuning implementation. These parameters are the following:

1) metric='euclidean'
2) n_neighbors=13
3) p=2

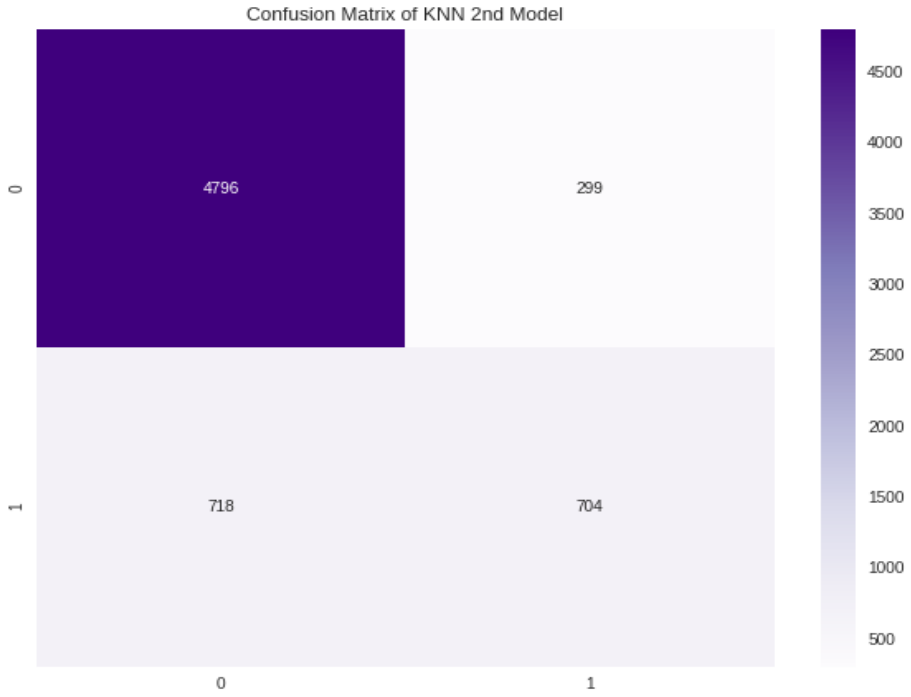

Figure 25: *Confusion Matrix of KNN Hypertuned model*

From the above confusion matrix, it is concluded that from 6,517 people, **5,500** were predicted in the correct class. Furthermore, **4,796** people were classified in class 0, which means that they would take a loan and that was correct as they did take it, while **704** classified in class 1, which means that they would not take a loan and actually they did not take it.

Table 12: *Full Classification Report – KNN*

	Precision	Recall	F1-score	Support
0	0.87	0.94	0.90	5,095
1	0.70	0.50	0.58	1,422
Accuracy			**0.84**	6,517
macro avg	0.79	0.72	0.74	6,517
weighted avg	0.83	0.84	0.83	6,517

Table 13: *Accuracy score vs ROC-AUC score – KNN*

Accuracy score	ROC-AUC score
84.3947%	71.8196%

Table 14: *Comparison Results for All KNN Classification Models*

Scores	Initial model	Hypertuned model	Feature Selected model	Smote Data model
Accuracy	83.7195%	**84.3947%**	83.2592%	**82.2865%**
ROC-accuracy	72.3510%	71.8196%	71.7525%	82.2865%

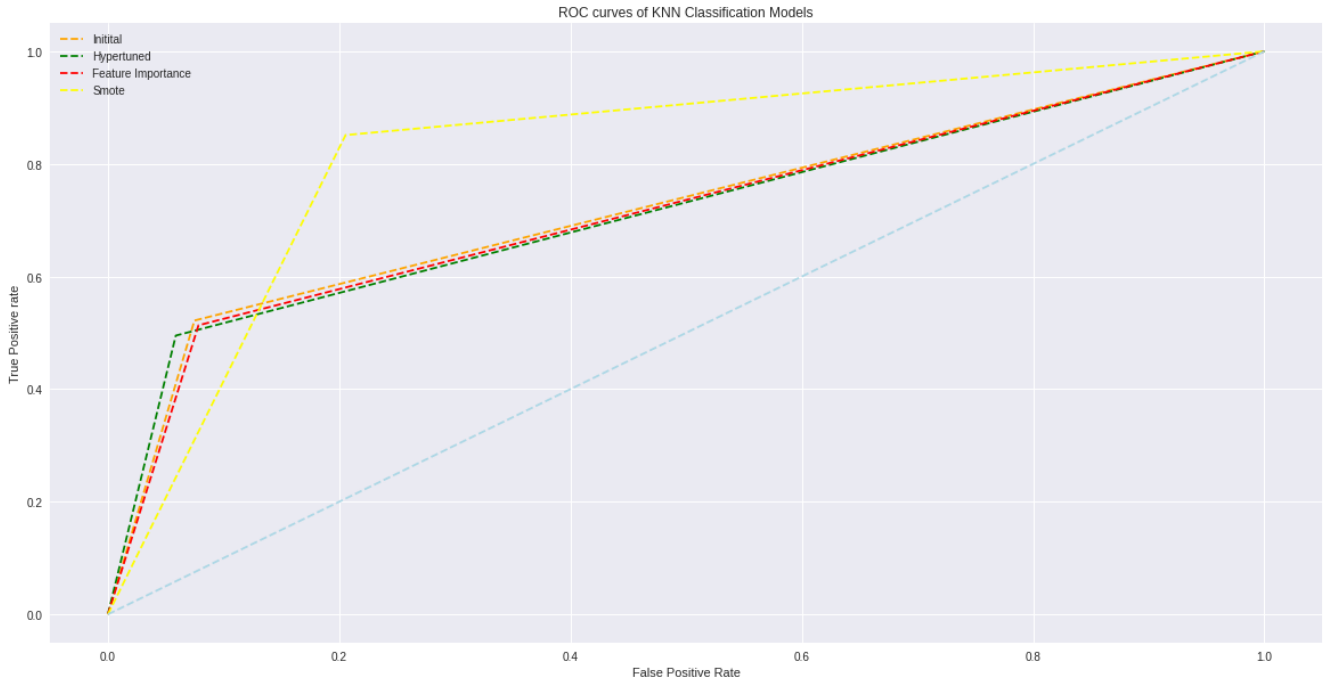

Figure 26: *Comparison of ROC curves for all KNN Models*

Overall, from all the four K-Nearest Neighbors models that have been created, they have very slightly good accuracy – between 80-85%, which means that random forest and decision tree models are much better and efficient for credit risk analysis with machine learning modelling.

4.4 Summary Results

Following the analysis per each model separately, a few final important comparisons of the machine learning results are presented. In the following tables, are shown the best results of each machine learning algorithm in 3 phases:

1) Comparing all 12 models that have been deployed,

2) Comparing all models trained in the original – real dataset, and

3) Comparing all models trained in oversampled – smote dataset.

All comparisons are based on the three most important machine learning evaluation metrics: Accuracy (F-1 score), ROC accuracy (ROC-AUC score) and ROC curves.

Table 15: *Comparison Results for the best model of each machine learning algorithm*

Scores	KNN	Decision Tree	Random Forest
Accuracy	84.3947%	**92.4337%**	**95.4858%**
ROC-accuracy	71.8196%	92.4337%	95.4858%

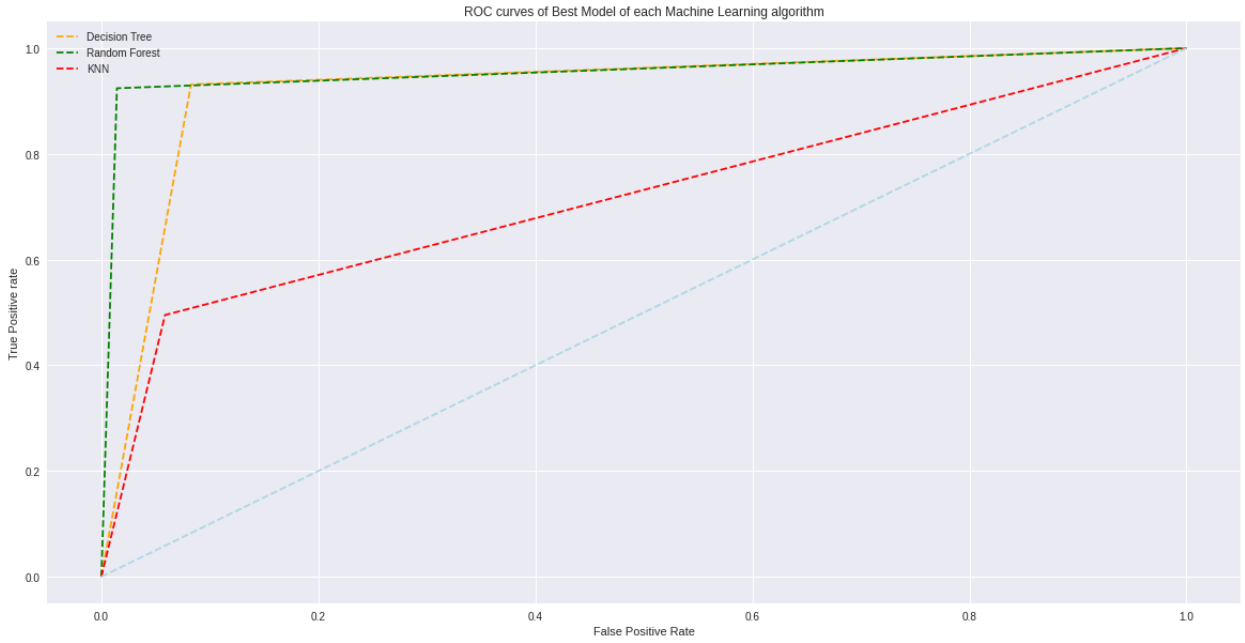

Figure 27: *Comparison of ROC curves for the best models*

Table 16: *Comparison Results for the best model of each machine learning algorithm based on the original dataset*

Model	KNN	Decision Tree	Random Forest
Accuracy	84.3947%	**91.1155%**	**93.2484%**
ROC-accuracy	71.8196%	80.3764%	85.5467%

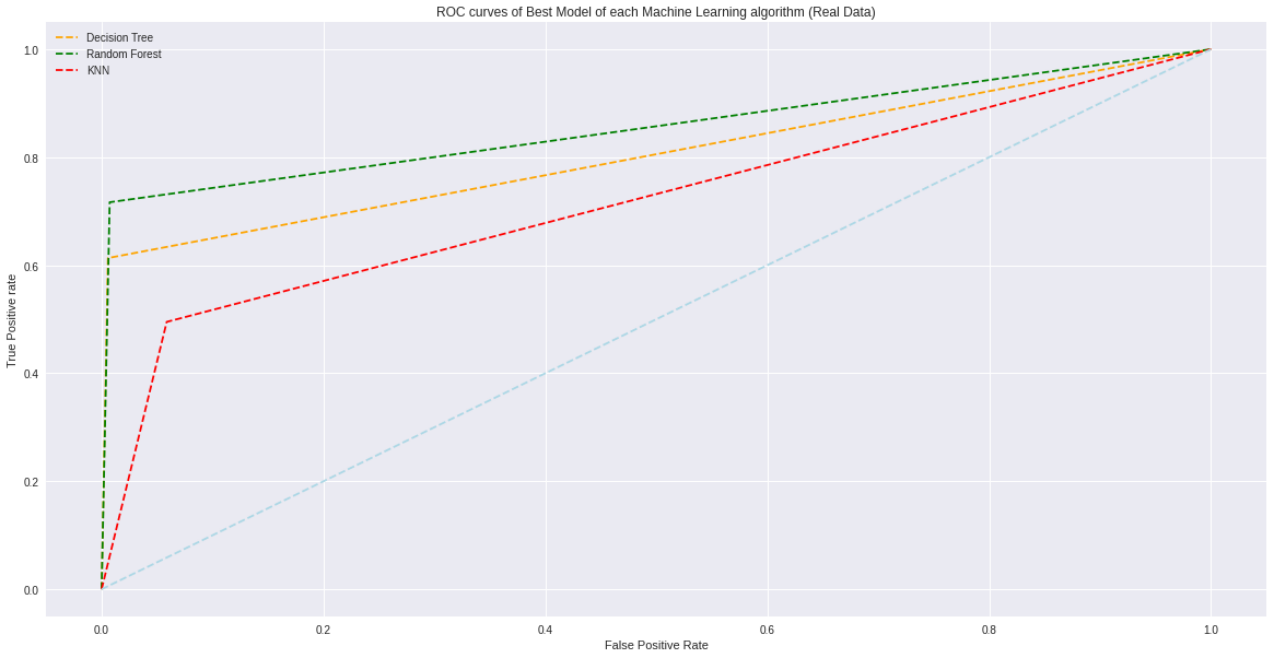

Figure 28: *Comparison of ROC curves for the best models based on the original dataset*

Table 17: *Comparison Results for the best model of each machine learning algorithm based on oversampled data*

Model	KNN	Decision Tree	Random Forest
Accuracy	82.2865%	**92.4337%**	**95.4858%**
ROC-accuracy	82.2865%	92.4337%	95.4858%

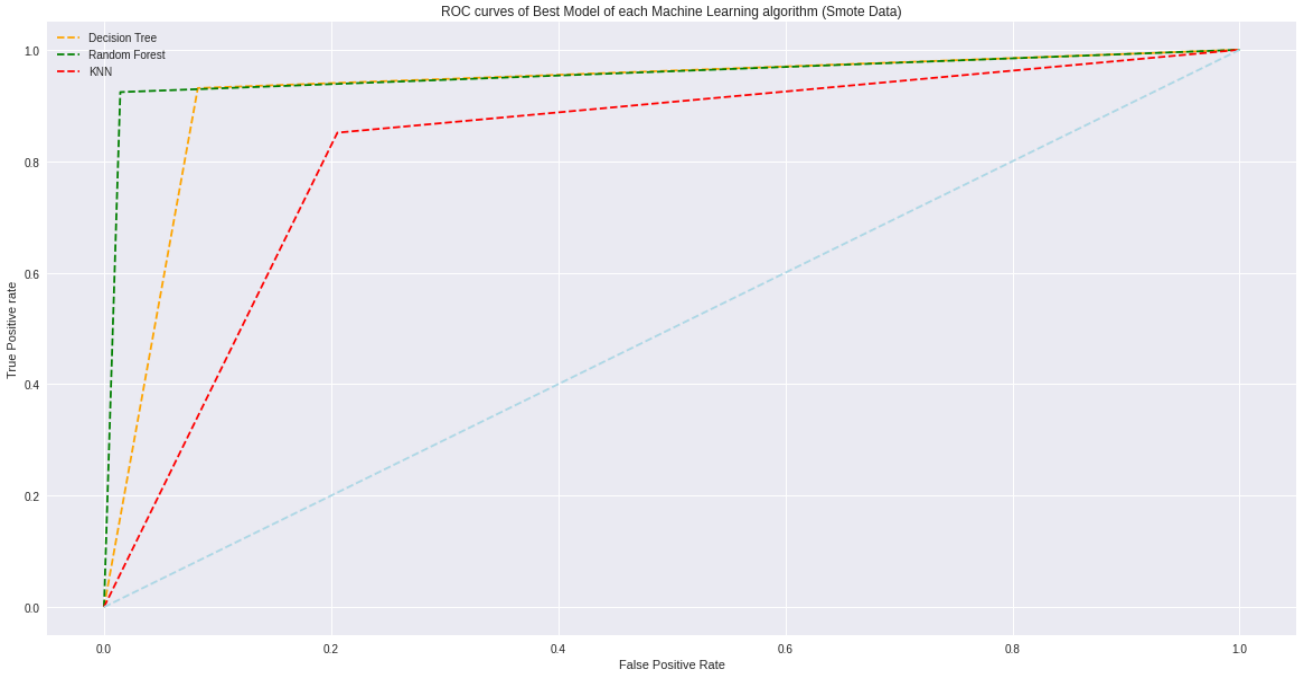

Figure 29: *Comparison of ROC curves for best models on smote data*

As a conclusion, the decision tree, random forest and K-nearest neighbors classification models achieve high-level predictions and a large amount of correct classifications in credit risk classification machine learning problems and in machine learning classification problems in finance in general. Especially, random forest classifier had the best overall performance in this analysis, as best Random Forest model had 95.4858% accuracy and 95.4858% ROC accuracy (ROC-AUC score).

5.Conclusions

The future of machine learning in the banking and financial sector is expected to be great, especially in the field of risk management. Machine learning techniques will be developed and applied to banking data in an effort to improve their operations. The ability of machine learning models to analyze large volumes of data in a relatively easier way and with greater reliability is very important. Machine learning, having important applications in risk management, can enable the creation of more accurate risk models by locating complex, non-linear patterns in large data sets.

In this dissertation an evaluation and analysis of the literature around the application of machine learning in risk management in the financial sector was presented. Most of the research seems to focus on credit risk management. One could attribute this to the fact that credit risk is considered the most important risk for a financial organization. More specifically, the credit risk management problems have been investigated concerned creditworthiness. The advantages and disadvantages of different machine learning techniques in solving specific risk management problems were studied, which can be further studied and evaluated.

The review showed that applying machine learning to risk management, such as credit risk, has been extensively investigated. However, it could be further explored for some areas where large-scale data analysis with complex and non-linear calculations are required.

As banks and financial institutions want to increase their risk management capabilities, it would be useful to explore how machine learning can be applied to the combination of different risks and improve the potential for reducing these risks. Areas such as behavioral risk could also be explored, in other words behavioral and activity monitoring. This improves risk management in financial institutions.

In the present work, twelve machine learning models such as decision tree, random forest and nearest neighbors were also studied and the analysis of the metrics of the models showed that both K-nearest neighbors, decision trees and random forest models are quite accurate in predicting the correct class for each person in our classification credit risk problem. More specifically, the best performance was achieved based on oversampled data, as 82.3%, 92.4% and 95.5% was the accuracy of KNN, Decision tree and Random Forest algorithm respectively. However, if we have to choose one machine learning algorithm for our future work and analysis, it is essential to use Random Forest, because it was examined as the most accurate by 95.5%.

Finally, it is important to highlight that several kinds of transformations in the original dataset contributed to taking the last form of the dataset that finally used for training and testing the models. It is true, that, when it comes to imbalanced datasets, working on the original data or working on oversampled data is a topic that raises a great conversation between researchers and scientists, and depends on many different factors. Having said that, using the SMOTE method of over-sampling the minority (abnormal) class can achieve, in general, better classifier performance in this problem. However, this doesn’t mean that they are better for every use in machine learning tasks.

Although machine learning is considered as a useful tool for credit risk analysis and default prediction, there are several limitations connected to this type of analysis. The most important limitation is data quality and predictive strength. An essential prerequisite to build a good and trustful model is the acquisition of a high-quality volume of representative data. Identifying the characteristics that highly affect defaulting or not is one of the greatest challenges. As economic conditions are continuously and rapidly changing, whereas new customers, new products and new trends are being introduced, new features, variables and correlations are needed to be taken into account. All these limitations are present in this study, as the dataset contains only 32,581 borrowers and their characteristics, which are limited and disclosure only a small part of all these parameters that could contribute to borrowing or not.

Personal data and its use are a very sensitive subject that concerns businesses and financial institutions. As time changes and more data is available, further research can be done taking into account larger and more complex datasets. New features and new techniques can be used in future projects, as machine learning and artificial intelligence in general are rapidly growing. In addition, decisions made by financial institutions using machine learning algorithms and their results can be used as the input for future research, leading in this way to a new type of assessment based on historical records for both right and wrong calls.

Bibliography

Dima, A. and Vasilache, S. (2009). *Ann model for corporate credit risk assessment*. In International Conference on Information and Financial Engineering, pages 94–98

Anderson, R. (2007). *The Credit Scoring Toolkit: Theory and Practice for Retail Credit Risk Management and Decision Automation.* Oxford University Press, UK Oxford.

Andrzej Szwabe and Pawel Misiorek (2018). *Decision trees as interpretable bank credit scoring models.* In Stanisław Kozielski, Dariusz Mrozek, Paweł Kasprowski, Boz˙ena Małysiak-Mrozek, and Daniel Kostrzewa, editors, Beyond Databases, Architectures and Structures. Facing the Challenges of Data Proliferation and Growing Variety, pages 207–219. Springer International Publishing.

Apapan Pumsirirat and Liu Yan (2018). *Credit Card Fraud Detection using Deep Learning based on Auto-Encoder and Restricted Boltzmann Machine.* International Journal of Advanced Computer Science and Applications, Vol. 9, No. 1, 2, pp 18-25.

Apostolik Richard, Christopher Donohue, Peter Went, and Global Association of Risk Professionals (2009). *Foundations of Banking Risk: An Overview of Banking, Banking Risks, and Risk-Based Banking Regulation.* New York: John Wiley.

Asma Feki, Anis Ishak, and Saber Feki (2012). *Feature selection using Bayesian and multiclass support vector machines approaches: Application to bank risk prediction*. Expert Syst. Appl., 39:3087– 3099.

Awad, Mariette, and Rahul Khanna (2015). *Efficient Learning Machines Theories, Concepts, and Applications for Engineers and System Designers.* DOI:10.1007/978-1-4302-5990-9_11.

Bacham, Dinesh, and Janet Zhao (2017). *Machine Learning: Challenges and Opportunities in Credit Risk Modeling.*

Available online: https://www.moodysanalytics.com/risk-perspectives-magazine/managing-disruption/spotlight/machine-learning-challenges-lessons-and-opportunities-in-credit-risk-modeling (accessed on 2 April 2018).

Bauguess, Scott W. (2015). *The Hope and Limitations of Machine Learning in Market Risk Assessment.* Washington, DC: U.S. Securities and Exchange Commission.

Barboza, Flavio, Herbert Kimura, and Edward Altman (2017). *Machine learning models and bankruptcy prediction.* Expert Systems with Applications 83: 405–17.

Bart Baesens, Rudy Setiono, Christophe Mues, and Jan Vanthienen (2003). *Using neural network rule extraction and decision tables for credit - risk evaluation.* Management Science, 49:312–329

Bayraci S., Susuz O. (2019). *Theoretical and Applied Economics Volume XXVI*. Winter, pages 75-84

Beaver, W. (1966). *Financial ratios as predictors of failure.* Journal of Accounting Research, 4, 71-111.

Bellotti, Tony, and Jonathan Crook (2009). *Support Vector Machines for Credit Scoring and Discovery of Significant Features.* Expert Systems with Applications DOI:10.1016 / j.eswa.2008.01.005.

Bellovary, J., Giacomino, D., & Akers, M. (2007). *A review of bankruptcy prediction studies: 1930 to present.* Journal of Financial Education, 33, 87-114.

Bo Huang, Qing-Pu Zhang, and Yun-Quan Hu (2005). *Research on credit risk management of the state-owned commercial bank*. In 2005 International Conference on Machine Learning and Cybernetics, volume 7, pages 4038–4043 Vol. 7.

Breiman, L. (1996). *Bagging predictors.* Machine Learning 24, pages 123–140.

Breiman, L. (2001). *Random Forests.* Machine Learning 45, pages 5–32.

Breiman, L. & Friedman, J. & Olshen, R. & Stone, C. (1984). *Classification and Regression Trees.* Wadsworth.

Brown, Iain, and Christophe Mues (2012). *An experimental comparison of classification algorithms for imbalanced credit scoring data sets.* Expert Systems with Applications 39: 3446–53.

Chartio (2021). *What is Exploratory Data Analysis* [Online]
Available at: https://chartio.com/learn/data-analytics/what-is-exploratory-data-analysis/

Chen, H., Chiang, R., & Storey, V. (2012). *Business intelligence and analytics: From big data to big impact.* MIS Quarterly: Special Issue on Business Intelligence Research, 36(4), pages 1165-1188.

Chen-Guang Yang and Xiao-Bo Duan (2008). *Credit risk assessment in commercial banks based on svm using pca.* In 2008 International Conference on Machine Learning and Cybernetics, volume 2, pages 1207–1211.

Chen, N., Ribeiro, B., & Chen, A. (2016). *Financial credit risk assessment: a recent review.* Artificial Intelligence Review, 45(1), pages 1-23.

Corporate Finance Institute (2021). *What is a Histogram?* [Online]
Available at: https://corporatefinanceinstitute.com/resources/excel/study/histogram/

D. J. Hand and W. E. Henley (1997). *Statistical classification methods in consumer credit scoring: a review.* Journal of the Royal Statistical Society: Series A (Statistics in Society), 160(3):523–541.

de Andrés, J., Landajo, M., & Lorca, P. (2012). *Bankruptcy prediction models based on multinorm analysis: An alternative to accounting ratios.* Knowledge-Based Systems, 30, pages 67-77.

Demyanyk, Y., & Hasan, I. (2010). *Financial crises and bank failures: A review of prediction methods.* Omega, 38, pages 315-324.

Ellis, D. (2008). *Lehman Brothers collapse stuns global markets.* Atlanta, United States: Cable News Network (CNN).

Geeksforgeeks (2020). *Confusion Matrix in Machine Learning* [Online]
Available at: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

Hamori, Shigeyuki, Minami Kawai, Takahiro Kume, Yuji Murakami, and Chikara Watanabe. (2018). *Ensemble Learning or Deep Learning? Application to Default Risk Analysis*. Journal of Risk and Financial Management 11: 12. DOI:10.3390/jrfm11010012.

Hull, John (2012). *Risk Management and Financial Institutions*. New York: John Wiley and Sons, vol. 733.

Ira I. Makrygianni and Angelos P. Markopoulos (2016). *Loan evaluation applying artificial neural networks.* In Proceedings of the Southeast European Design Automation, Computer Engineering, Computer Networks and Social Media Conference, SEEDA-CECNSM ’16, pages 124–128, New York, NY, USA.

Jakub Tomczak and Maciej Zieba (2014). *Classification restricted Boltzmann machine for comprehensible credit scoring model.* Expert Systems with Applications, 42.

James Wilson, H., Mulani, N., & Alter, A. (2016). *Sales gets a machine-learning makeover*. Cambridge, United States: MIT Sloan Management Review.

Jorion, Philippe (2007). *Value at Risk: The New Benchmark for Managing Financial Risk.* New York: McGraw-Hill.

Kuldeep Randhawa, Chu Kiong Loo, Man Jeevan Seera, Chee Peng Lim, Asoka K. Nandi (2018). *Credit card fraud detection using AdaBoost and majority voting.* IEEE Access (Volume: 6).

Lessmann, Stefan, Bart Baesens, Hsin Vonn Seow, and Lyn C. Thomas (2015). *Benchmarking State-of-the-Art Classification Algorithms for Credit Scoring: An Update of Research.* European Journal of Operational Research 247: 124–36. DOI:10.1016/j.ejor.2015.05.030.

Li, M., & Miu, P. (2010). *A hybrid bankruptcy prediction model with dynamic loadings on accounting-ratio-based and market-based information: A binary quantile regression approach.* Empirical Finance,17, pages 818-833.

M. A. H. Farquad, V. Ravi, Sriramjee, and G. Praveen (2011). *Credit scoring using pca-svm hybrid model*. In Vinu V. Das, Janahanlal Stephen, and Yogesh Chaba, editors, Computer Networks and Information Technologies, pages 249–253, Berlin, Heidelberg Springer Berlin Heidelberg.

Masoumeh Zareapoora, Pourya Shamsolmoali (2015). *Application of Credit Card Fraud Detection: Based on Bagging Ensemble Classifier* International Conference on Intelligent Computing, Communication & Convergence (ICCC-2015). Procedia Computer Science 48, pages 679 – 686.

N. K. Gyamfi and J. Abdulai (2018). *Bank fraud detection using support vector machine.* In 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pages 37–41.

Nnamdi I. Nwulu, Shola Oroja, and Mustafa Ilkan (2011). *Credit scoring using soft computing schemes: A comparison between support vector machines and artificial neural networks.* In Ezendu Ariwa and Eyas El-Qawasmeh, editors, Digital Enterprise and Information Systems, pages 275–286, Springer Berlin Heidelberg.

Olatunji Okesola, Kennedy Okokpujie, Adeyinka Adewale, Samuel John, and Osemwegie Omoruyi (2017). *An improved bank credit scoring model: A naive Bayesian approach*. pages 228–233.

P. Ravisankar, V. Ravi, G. Raghava Rao, and I. Bose (2011). *Detection of financial statement fraud and feature selection using data mining techniques.* Decis. Support Syst., 50(2):491–500.

R. Vedala and B. R. Kumar (2012). *An application of naive bayes classification for credit scoring in e-lending platform.* In 2012 International Conference on Data Science Engineering (ICDSE), pages 81–84.

Rashmi Malhotra and D.K. Malhotra (2003). *Evaluating consumer loans using neural networks.* Omega, 31:83–96.

Rong-Zhou Li, Su-Lin Pang, and Jian-Min Xu (2002). *Neural network credit-risk evaluation model based on back-propagation algorithm.* In Proceedings, International Conference on Machine Learning and Cybernetics, pages 1702–1706 vol.4.

S. Huang and M. Day (2013). *A comparative study of data mining techniques for credit scoring in banking.* In 2013 IEEE 14th International Conference on Information Reuse Integration (IRI), pages 684–691.

Saunders, Anthony, Marcia Millon Cornett, and Patricia Anne McGraw (2006). *Financial Institutions Management: A Risk Management Approach.* New York: McGraw-Hill.

Seaborn (2020). *Seaborn count plot.* [Online]
Available at: https://seaborn.pydata.org/generated/seaborn.countplot.html

Sihem Khemakhem and Younes Boujelbene (2017). *Artificial intelligence for credit risk assessment: Artificial neural network and support vector machines.* ACRN Oxford Journal of Finance and Risk Perspectives, 6:1–17.
Shalev-Shwartz, Shai, and Shai Ben-David (2014). *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press. DOI: 10.1017 / CBO9781107298019.

Shell, A. (2009). *Lehman Bros. collapse triggered economic turmoil.* New York City, United States: American Broadcasting Company (ABC).

Sotiris Kotsiantis, Euaggelos Koumanakos, Dimitris Tzelepis, and Vasilis Tampakas (2006). *Predicting fraudulent financial statements with machine learning techniques.* In Grigoris Antoniou, George Potamias, Costas Spyropoulos, and Dimitris Plexousakis, editors, Advances in Artificial Intelligence, pages 538–542, Springer Berlin Heidelberg.

Stjepan Oreski, Dijana Oreski, and Goran Oresˇki (2012). *Hybrid system with genetic algorithm and artificial neural networks and its application to retail credit risk assessment.* Expert Systems with Applications, 39:12605–12617.

Van Gestel, Ir Tony, Bart Baesens, Ir Joao Garcia, and Peter Van Dijcke (2003). *A support vector machine approach to credit scoring.* In FORUM FINANCIER-REVUE BANCAIRE ET FINANCIAIRE BANK EN FINANCIEWEZEN, pages 73-82.

Van Liebergen, Bart (2017). *Machine Learning: A Revolution in Risk Management and Compliance?* Journal of Financial Transformation 45: 60–67.

W. Feng, Y. Zhao, and J. Deng (2009). *Application of svm based on principal component analysis to credit risk assessment in commercial banks.* In 2009 WRI Global Congress on Intelligent Systems, volume 4, pages 49–52.

Wang, Yongqiao, Shouyang Wang, and Kin Keung Lai (2005). *A New Fuzzy Support Vector Machine to Evaluate Credit Risk.* IEEE Transactions on Fuzzy Systems 13: 820–831. DOI: 10.1109 /TFUZZ.2005.859320.

Wang, Hong, Qingsong Xu, and Lifeng Zhou (2015). *Large Unbalanced Credit Scoring Using Lasso-Logistic Regression Ensemble.* PLoS ONE 10: e0117844. DOI: 10.1371 / journal.pone. 0117844.

Wikipedia (2021). *Confusion matrix.* [Online]
Available at: https://en.wikipedia.org/wiki/Confusion_matrix

Wikipedia (2021). *Exploratory data analysis.* [Online]
Available at: https://en.wikipedia.org/wiki/Exploratory_data_analysis

Williams, M. (2010). *Uncontrolled risk: Lessons of Lehman Brothers and how systemic risk can still bring down the world financial system.* New York City, United States: McGraw-Hill Education.

Wilson, R., & Sharda, R. (1994). *Bankruptcy prediction using neural networks.* Decision Support Systems, 11, pages 545-557.

Wójcicka, Aleksandra (2017). *Neural Networks vs. Discriminant Analysis in the Assessment of Default.* Electronic Economy. Pages 339–349. DOI:10.17951/h.2017.51.5.339.

Xin-Yue Hu and Yongli Tang (2006). *Ann-based credit risk identification and control for commercial banks.* Vol.2006, pages 3110 – 3114.

Yacine Djemaiel, Nadia Labidi, and Noureddine Boudriga (2016). *A dynamic hybrid rbf/elman neural networks for credit scoring using big data.* In Witold Abramowicz, Rainer Alt, and Bogdan Franczyk, editors, Business Information Systems, pages 102–113, Springer International Publishing.

Yang, Zijiang, Wenjie You, and Guoli Ji (2011). *Using partial least squares and support vector machines for bankruptcy prediction.* Expert Systems with Applications 38, pages 8336–8342.

Yeh, I. Cheng, and Chehui Lien (2009). *The Comparisons of Data Mining Techniques for the Predictive Accuracy of Probability of Default of Credit Card Clients.* Expert Systems with Applications DOI:10.1016/j.eswa.2007.12.020.

Youness Abakarim, Mohamed Lahby, and Abdelbaki Attioui (2018). *An efficient real time model for credit card fraud detection based on deep learning.* In Proceedings of the 12th International Conference on Intelligent Systems: Theories and Applications, SITA'18, pages 30:1–30:7, New York, NY, USA.

Z. Zhang (2011). *Research of default risk of commercial bank's personal loan based on rough sets and neural network.* In 2011 3rd International Workshop on Intelligent Systems and Applications, pages 1–4.

Appendix

Python code

```
1.  # Import libraries
2.
3.  import pandas as pd
4.  import matplotlib.pyplot as plt
5.  import numpy as np
6.  import scipy
7.  import sklearn
8.  import seaborn as sns
9.  import missingno as msno
10. from sklearn.preprocessing import LabelEncoder
11. from sklearn.model_selection import train_test_split
12. from sklearn.feature_selection import SelectKBest
13. from sklearn.feature_selection import chi2
14. from imblearn.over_sampling import SMOTE
15. from sklearn.inspection import permutation_importance
16. from sklearn.neighbors import KNeighborsClassifier
17. from sklearn import metrics
18. from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score,
    roc_curve, accuracy_score
19. from imblearn import under_sampling, over_sampling
20. from sklearn.ensemble import RandomForestClassifier
21. from sklearn.tree import DecisionTreeClassifier
22. from sklearn import svm
23. from sklearn.model_selection import RandomizedSearchCV
24. from imblearn.pipeline import Pipeline
25. from sklearn.model_selection import RepeatedStratifiedKFold
26.
27. # Upload Dataset
28. import io
29. from google.colab import files
30.
31. uploaded = files.upload()
32.
33. df = pd.read_csv(io.BytesIO(uploaded['credit_risk_dataset.csv']))
34.
35. df
36.
37. """Data cleaning
38.
39. Handling NA values
40. """
41.
42. # Checking NA values in our dataset
43. for i in df.columns:
44.     print('Column:',i)
45.     print('Number of N/As:',df[i].isna().sum())
46.
47. print("We have 895 missing values in 'person_emp_length' feature and 3116 in 'loan_int_rate'
    feature")
48.
49. # Or with plot checking
50. # Visualize missing values as a matrix
51. msno = msno.matrix(df)
52.
```

```
# Fill missing values with the aerage average(mean) of the feature.

df['person_emp_length'].fillna(df['person_emp_length'].mean(),inplace=True)
df['loan_int_rate'].fillna(df['loan_int_rate'].mean(),inplace=True)

# Now check again

for i in df.columns:
    print('Column:',i)
    print('Number of N/As:',df[i].isna().sum())

print("We have not missing values now.")

"""# Exploratory Data Analysis"""

# Dataset statistics
df.describe().T

# Our dataset information
df.info()
print("So we have 4 categorical variables and 8 numeric.")

# Histograms for numerical Data
df.hist(bins=50, figsize=(20,15))
plt.show()

# Check the distribution of our Target variable (0 No - 1 Yes)
plt.hist(df['loan_status'])
plt.title('Distribution of Class variable', fontsize = 30)
plt.xlabel('Loan', fontsize = 15)
plt.ylabel('Count', fontsize = 15)
plt.grid()
plt.show()

print(df.loan_status.value_counts())
print("We have 25473 people on class 0 and 7108 on class 1.")

# checking the Distribution of variable 'person_home_ownership'

plt.figure(figsize=(15,10))
sns.countplot(df['person_home_ownership'], palette = 'inferno')
plt.title('Each Percon Type of Ownership', fontsize = 30)
plt.xlabel('Type of ownership', fontsize = 15)
plt.ylabel('count', fontsize = 15)

plt.show()

 # checking the Distribution of variable 'loan_intent'

 plt.figure(figsize=(15,10))
 sns.countplot(df['loan_intent'], palette = 'pastel')
 plt.title('Each Person Intention for Loan', fontsize = 30)
 plt.xlabel('Type of Intention', fontsize = 15)
 plt.ylabel('count', fontsize = 15)

 plt.show()

 # checking the Distribution of variable 'loan_grade'

 plt.figure(figsize=(15,10))
 sns.countplot(df['loan_grade'], palette = 'inferno')
 plt.title('Grade of Loan from A to G for each Person', fontsize = 30)
 plt.xlabel('Grade of Loan', fontsize = 15)
 plt.ylabel('count', fontsize = 15)

```

```
plt.show()

# checking the Distribution of variable cb_person_default_on_file

plt.figure(figsize=(15,10))
sns.countplot(df['cb_person_default_on_file'], palette = 'pastel')
plt.title('Historical Default', fontsize = 30)
plt.xlabel('Answer', fontsize = 15)
plt.ylabel('count', fontsize = 15)

plt.show()

# Mixed plots

fig, ax = plt.subplots(2,2,figsize = (30,20))
plt.tight_layout(pad = 3)

sns.countplot(data = df, x ='person_home_ownership',hue = 'loan_status',ax =ax[0,0],
              palette = "rocket_r").set_title('Loan Ratio by Person Ownership')
sns.countplot(data = df, x ='loan_intent',hue = 'loan_status',ax =ax[0,1],
              palette = "rocket_r").set_title("Loan ratio By Person's Type of Intention")
sns.countplot(data = df, x ='loan_grade',hue = 'loan_status',ax =ax[1,0],
              palette = "rocket_r").set_title('Loan ratio by its Grade')
sns.countplot(data = df, x ='cb_person_default_on_file',hue = 'loan_status',ax =ax[1,1],
              palette = "rocket_r").set_title('Loan ratio by Historical default')

# fig, axs = plt.subplots(nr,nc)
sns.color_palette("pastel")
plt.figure(figsize=(20,15))
sns.violinplot(x='loan_intent',y='loan_percent_income',data=df,linewidth=0.3)
sns.despine()
plt.title("Violin Plots of Percentage of Income by Loan Intention")
plt.show()

# fig, axs = plt.subplots(nr,nc)
sns.color_palette("pastel")
plt.figure(figsize=(20,15))
sns.violinplot(x='loan_intent',y='person_emp_length',data=df,linewidth=0.3)
sns.despine()
plt.title("Violin Plots of Employment time length by Loan Intention")
plt.show()

# fig, axs = plt.subplots(nr,nc)
sns.color_palette("pastel")
plt.figure(figsize=(20,15))
sns.violinplot(x='loan_grade',y='loan_percent_income',data=df,linewidth=0.3)
sns.despine()
plt.title("Violin Plots of Percentage of Income by Loan Grade")
plt.show()

# Check for outliers
plt.figure(figsize=(80, 30))
df.boxplot()

# Heatmap - information map
plt.figure(figsize=(15,15))
heatmap = sns.heatmap(df.corr(), annot=True ,cmap =sns.diverging_palette(10, 110, n=100))

# Pairplot

pairplot = sns.pairplot(df, hue = 'loan_status')

"""# Data Preparation"""

# Encoding categorical data with LabelEncoder Technique
```

```
183.
184.  # Define label_encoder
185.  label_encoder = LabelEncoder()
186.  # Encode labels in columns 'Type' and 'Method.
187.  df['person_home_ownership'] = label_encoder.fit_transform(df['person_home_ownership'])
188.  df['loan_intent'] = label_encoder.fit_transform(df['loan_intent'])
189.  df['loan_grade'] = label_encoder.fit_transform(df['loan_grade'])
190.  df['cb_person_default_on_file'] =
    label_encoder.fit_transform(df['cb_person_default_on_file'])
191.
192.  df.info()
193.
194.  df
195.
196.  # Let's see now how heatmap and pairplot look
197.
198.  # Heatmap - information map
199.  plt.figure(figsize=(25,25))
200.  heatmap = sns.heatmap(df.corr(), annot=True ,cmap =sns.diverging_palette(10, 110, n=100))
201.
202.  # Pairplot
203.  pairplot = sns.pairplot(df, hue = 'loan_status')
204.
205.  # Final Check
206.  # Which are the unique values for each feature?
207.  for i in df.columns:
208.      print('Column Title',i)
209.      print('# Unique Values:',len(df[i].unique()))
210.      print('Unique Values:',df[i].unique())
211.      print()
212.
213.  """# Feature Importances"""
214.
215.  X = pd.DataFrame(df)
216.  X = X.drop(['loan_status'], axis=1)
217.  y = df[['loan_status']]
218.
219.  X
220.
221.  y
222.
223.  X.info()
224.
225.  # K-best Method
226.  best_features = SelectKBest(score_func=chi2, k=11)
227.  fit = best_features.fit(X,y)
228.  df_scores = pd.DataFrame(fit.scores_)
229.  df_columns = pd.DataFrame(X.columns)
230.  feature_scores = pd.concat([df_columns, df_scores], axis=1)
231.  feature_scores.columns = ["Feature","Score"]
232.  print(feature_scores.nlargest(11,"Score"))
233.
234.  # Permutation Importance method (for KNN)
235.
236.  # define the model
237.  model = KNeighborsClassifier()
238.  # fit
239.  model.fit(X, y)
240.  # perform permutation importance to get the importances of the features
241.  results = permutation_importance(model, X, y, scoring='accuracy')
242.  # get importance
243.  importance = results.importances_mean
244.  # summarize feature importance
245.  for i,v in enumerate(importance):
246.            print('Feature: %0d, Score: %.5f' % (i,v))
```

```
247.  # plot feature importance
248.  plt.bar([x for x in range(len(importance))], importance)
249.  plt.show()
250.
251.  print('Best Features are only 2: person_income and loan_amount, but if we have to select
    5: person_age, person_income,person_home_ownership,person_emp_length,loan_amount.')
252.
253.  # Decision Tree Classifier Feature Importance method
254.
255.  # define the model
256.  model = DecisionTreeClassifier()
257.  # fit
258.  model.fit(X, y)
259.  # get the importances of the features based on decision tree model
260.  importance = model.feature_importances_
261.  # summarize feature importance
262.  for i,v in enumerate(importance):
263.            print('Feature: %0d, Score: %.5f' % (i,v))
264.  # plot feature importance
265.  plt.bar([x for x in range(len(importance))], importance)
266.  plt.show()
267.
268.  print('Based on Decision Tree model importances, best 5 features of dataset are:
    person_income, person_home_ownership, loan_intent, loan_grade, loan_percent_income ')
269.
270.  # Random Forest Classifier Feature Importance method
271.
272.  # define the model
273.  model = RandomForestClassifier()
274.  # fit
275.  model.fit(X, y)
276.  # get the importances of the features based on random forest model
277.  importance = model.feature_importances_
278.  # summarize feature importance
279.  for i,v in enumerate(importance):
280.            print('Feature: %0d, Score: %.5f' % (i,v))
281.  # plot feature importance
282.  plt.bar([x for x in range(len(importance))], importance)
283.  plt.show()
284.
285.  print('Based on Decision Tree model importances, best 5 features of dataset are:
    person_income, person_home_ownership, loan_grade,loan_int_rate, loan_percent_income ')
286.
287.  """# Smote dataset"""
288.
289.  # If we take for granted that our dataset is imbalanced, we have to make an oversampling
    or undersampling technique to make it balanced.
290.  # So we will use Smote technique
291.
292.  # transform the dataset
293.  oversample = SMOTE()
294.  X_smote, y_smote = oversample.fit_resample(X, y)
295.
296.  X_smote= pd.DataFrame(X_smote)
297.  y_smote = pd.DataFrame(y_smote)
298.
299.  X.info()
300.
301.  X_smote.columns =
    ['person_age','person_income','person_home_ownership','person_emp_length','loan_intent','loa
    n_grade','loan_amnt',
302.
    'loan_int_rate','loan_percent_income','cb_person_default_on_file','cb_person_cred_hist_lengt
    h']
303.
```

```
y_smote.columns = ['loan_status']

X_smote

y_smote

"""# Machine Learning models

For each machine learning algorithm, we will create 4 models: 1st in initial dataset, 2nd after hyper tuning (finding best parameters), 3rd after feature importance data and 4th in Smote dataset

### Decision Tree Classifier

1) Initial
"""

# Train, Test, Split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.2, random_state=152, stratify=y)

# Define the model
clf_1 = DecisionTreeClassifier()

# Fit the model
clf_1.fit(X_train,Y_train)

# Make prediction
pred_1 = clf_1.predict(X_test)

# Accuracy score
print('Accuracy score of Decision Tree model 1 is:', accuracy_score(Y_test, pred_1))

# Confusion Matrix
conmat_1 = confusion_matrix(Y_test, pred_1)
print(conmat_1)
plt.figure(figsize=(10,8))
plt.title('Confusion Matrix of Decision Tree 1st Model')
sns.heatmap(conmat_1, annot=True, cmap="Purples", fmt="d",cbar=True)

# Classification Report
print('Classification Report of Decision Tree 1st Model')
print(classification_report(Y_test, pred_1))

# ROC-AUC score
print('ROC-AUC score of 1st Decision Tree Model is:', roc_auc_score(Y_test, pred_1))

"""2) Hypertuning """

# Setup the parameters and distributions to sample from: param_dist
param_dist = {"max_depth": range(1,5),
              "min_samples_split": range(10),
              "min_samples_leaf": range(1,5),
              "criterion": ["gini", "entropy"]}

# Define Hypertuning model
tree_cv = RandomizedSearchCV(estimator=DecisionTreeClassifier(), param_distributions=param_dist, cv=5, return_train_score=False)

# Fit
tree_cv.fit(X_train,Y_train)

# Best parameters
print('Best parameters for Decision Tree Classification are:', tree_cv.best_params_)

```

```
# Train, Test, Split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.2, random_state=152, stratify=y)

# Define the model
clf_2 = DecisionTreeClassifier(min_samples_split=4, min_samples_leaf=2, max_depth=4, criterion='gini')

# Fit the model
clf_2.fit(X_train,Y_train)

# Make prediction
pred_2 = clf_2.predict(X_test)

# Accuracy score
print('Accuracy score of Decision Tree model 2 is:', accuracy_score(Y_test, pred_2))

# Confusion Matrix
conmat_2 = confusion_matrix(Y_test, pred_2)
print(conmat_2)
plt.figure(figsize=(10,8))
plt.title('Confusion Matrix of Decision Tree 2nd Model')
sns.heatmap(conmat_2, annot=True, cmap="Purples", fmt="d",cbar=True)

# Classification Report
print('Classification Report of Decision Tree 2nd Model')
print(classification_report(Y_test, pred_2))

# ROC-AUC score
print('ROC-AUC score of 2nd Decision Tree Model is:', roc_auc_score(Y_test, pred_2))

"""3) Feature importance"""

X_2 = X[['person_income','person_home_ownership','loan_intent', 'loan_grade', 'loan_percent_income']]

X_2

# Train, Test, Split
X_train_2, X_test_2, Y_train_2, Y_test_2 = train_test_split(X_2, y, test_size = 0.2, random_state=152, stratify=y)

# Define the model
clf_3 = DecisionTreeClassifier()

# Fit the model
clf_3.fit(X_train_2,Y_train_2)

# Make prediction
pred_3 = clf_3.predict(X_test_2)

# Accuracy score
print('Accuracy score of Decision Tree model 3 is:', accuracy_score(Y_test_2, pred_3))

# Confusion Matrix
conmat_3 = confusion_matrix(Y_test_2, pred_3)
print(conmat_3)
plt.figure(figsize=(10,8))
plt.title('Confusion Matrix of Decision Tree 3rd Model')
sns.heatmap(conmat_3, annot=True, cmap="Purples", fmt="d",cbar=True)

# Classification Report
print('Classification Report of Decision Tree 3rd Model')
print(classification_report(Y_test_2, pred_3))

```

```
# ROC-AUC score
print('ROC-AUC score of 3rd Decision Tree Model is:', roc_auc_score(Y_test_2, pred_3))

"""4) Smote"""

# Train, Test, Split
X_train_3, X_test_3, Y_train_3, Y_test_3 = train_test_split(X_smote, y_smote, test_size =
    0.2, random_state=152, stratify=y_smote)

# Define the model
clf_4 = DecisionTreeClassifier()

# Fit the model
clf_4.fit(X_train_3,Y_train_3)

# Make prediction
pred_4 = clf_4.predict(X_test_3)

# Accuracy score
print('Accuracy score of Decision Tree model 4 is:', accuracy_score(Y_test_3, pred_4))

# Confusion Matrix
conmat_4 = confusion_matrix(Y_test_3, pred_4)
print(conmat_4)
plt.figure(figsize=(10,8))
plt.title('Confusion Matrix of Decision Tree 4th Model')
sns.heatmap(conmat_4, annot=True, cmap="Purples", fmt="d",cbar=True)

# Classification Report
print('Classification Report of Decision Tree 4th Model')
print(classification_report(Y_test_3, pred_4))

# ROC-AUC score
print('ROC-AUC score of 4th Decision Tree Model is:', roc_auc_score(Y_test_3, pred_4))

"""5) Combine and Compare all ROC curves"""

# roc curve for models
fpr1, tpr1, thresh1 = roc_curve(Y_test, pred_1)
fpr2, tpr2, thresh2 = roc_curve(Y_test, pred_2)
fpr3, tpr3, thresh3 = roc_curve(Y_test_2, pred_3)
fpr4, tpr4, thresh4 = roc_curve(Y_test_3, pred_4)

# plot roc curves
plt.style.use('seaborn')

plt.figure(figsize=(20,10))

plt.plot(fpr1, tpr1, linestyle='--',color='orange', label='Initital')
plt.plot(fpr2, tpr2, linestyle='--',color='green', label='Hypertuned')
plt.plot(fpr3, tpr3, linestyle='--',color='red', label='Feature Importance')
plt.plot(fpr4, tpr4, linestyle='--',color='yellow', label='Smote')
plt.plot([0,1],linestyle='--', color = 'blue')
# title
plt.title('ROC curves of Decision Tree Classification Models')
# x label
plt.xlabel('False Positive Rate')
# y label
plt.ylabel('True Positive rate')
plt.legend(loc='best')

plt.show()

"""### Random Forest Classifier

```

```
1) Initial
"""

# Train, Test, Split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.2,
    random_state=152, stratify=y)

# Define the model
clf_5 = RandomForestClassifier()

# Fit the model
clf_5.fit(X_train,Y_train)

# Make prediction
pred_5 = clf_5.predict(X_test)

# Accuracy score
print('Accuracy score of Random FOrest model 1 is:', accuracy_score(Y_test, pred_5))

# Confusion Matrix
conmat_5 = confusion_matrix(Y_test, pred_5)
print(conmat_5)
plt.figure(figsize=(10,8))
plt.title('Confusion Matrix of Random FOrest 1st Model')
sns.heatmap(conmat_5, annot=True, cmap="Purples", fmt="d",cbar=True)

# Classification Report
print('Classification Report of Random Forest 1st Model')
print(classification_report(Y_test, pred_5))

# ROC-AUC score
print('ROC-AUC score of 1st Random Forest Model is:', roc_auc_score(Y_test, pred_5))

"""2) Hypertuning"""

# Setup the parameters and distributions to sample from: param_dist
param_dist = {"n_estimators": [50, 150, 200, 250],
              "max_depth": range(1,5),
              "min_samples_split": range(10),
              "min_samples_leaf": range(1,5),
              "criterion": ["gini", "entropy"]}

# Define Hypertuning model
rf_cv = RandomizedSearchCV(estimator=RandomForestClassifier(), param_distributions=
    param_dist, cv=5, return_train_score=False)

# Fit
rf_cv.fit(X_train,Y_train)

# Best parameters
print('Best parameters for Random Forest Classifier are:', rf_cv.best_params_)

# Train, Test, Split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.2,
    random_state=152, stratify=y)

# Define the model
clf_6 = RandomForestClassifier(n_estimators=250, min_samples_split=5, min_samples_leaf=1,
    max_depth=4, criterion='entropy')

# Fit the model
clf_6.fit(X_train,Y_train)

# Make prediction
pred_6 = clf_6.predict(X_test)
```

```
551.
552.  # Accuracy score
553.  print('Accuracy score of Random Forest model 2 is:', accuracy_score(Y_test, pred_6))
554.
555.  # Confusion Matrix
556.  conmat_6 = confusion_matrix(Y_test, pred_6)
557.  print(conmat_6)
558.  plt.figure(figsize=(10,8))
559.  plt.title('Confusion Matrix of Random Forest 2nd Model')
560.  sns.heatmap(conmat_6, annot=True, cmap="Purples", fmt="d",cbar=True)
561.
562.  # Classification Report
563.  print('Classification Report of Random Forest 2nd Model')
564.  print(classification_report(Y_test, pred_6))
565.
566.  # ROC-AUC score
567.  print('ROC-AUC score of 2nd Random Forest Model is:', roc_auc_score(Y_test, pred_6))
568.
569.  """3) Feature Importance"""
570.
571.  X_3 = X[['person_income','person_home_ownership','loan_grade', 'loan_int_rate',
    'loan_percent_income']]
572.
573.  # Train, Test, Split
574.  X_train_4, X_test_4, Y_train_4, Y_test_4 = train_test_split(X_3, y, test_size = 0.2,
    random_state=152, stratify=y)
575.
576.  # Define the model
577.  clf_7 = RandomForestClassifier()
578.
579.  # Fit tme model
580.  clf_7.fit(X_train_4,Y_train_4)
581.
582.  # Make prediction
583.  pred_7 = clf_7.predict(X_test_4)
584.
585.  # Accuracy score
586.  print('Accuracy score of Random Forest model 3 is:', accuracy_score(Y_test_4, pred_7))
587.
588.  # Confusion Matrix
589.  conmat_7 = confusion_matrix(Y_test_4, pred_7)
590.  print(conmat_7)
591.  plt.figure(figsize=(10,8))
592.  plt.title('Confusion Matrix of Random Forest 3rd Model')
593.  sns.heatmap(conmat_7, annot=True, cmap="Purples", fmt="d",cbar=True)
594.
595.  # Classification Report
596.  print('Classification Report of Random Forest 3rd Model')
597.  print(classification_report(Y_test_4, pred_7))
598.
599.  # ROC-AUC score
600.  print('ROC-AUC score of 3rd Random Forest Model is:', roc_auc_score(Y_test_4, pred_7))
601.
602.  """4) Smote"""
603.
604.  # Train, Test, Split
605.  X_train_3, X_test_3, Y_train_3, Y_test_3 = train_test_split(X_smote, y_smote, test_size =
    0.2, random_state=152, stratify=y_smote)
606.
607.  # Define the model
608.  clf_8 = RandomForestClassifier()
609.
610.  # Fit tme model
611.  clf_8.fit(X_train_3,Y_train_3)
612.
```

```
# Make prediction
pred_8 = clf_8.predict(X_test_3)

# Accuracy score
print('Accuracy score of Random Forest model 4 is:', accuracy_score(Y_test_3, pred_8))

# Confusion Matrix
conmat_8 = confusion_matrix(Y_test_3, pred_8)
print(conmat_8)
plt.figure(figsize=(10,8))
plt.title('Confusion Matrix of Random Forest 4th Model')
sns.heatmap(conmat_8, annot=True, cmap="Purples", fmt="d",cbar=True)

# Classification Report
print('Classification Report of Random Forest 4th Model')
print(classification_report(Y_test_3, pred_8))

# ROC-AUC score
print('ROC-AUC score of 4th Random Forest Model is:', roc_auc_score(Y_test_3, pred_8))

"""5) Combine and Compare all ROC Curves"""

# roc curve for models
fpr1, tpr1, thresh1 = roc_curve(Y_test, pred_5)
fpr2, tpr2, thresh2 = roc_curve(Y_test, pred_6)
fpr3, tpr3, thresh3 = roc_curve(Y_test_4, pred_7)
fpr4, tpr4, thresh4 = roc_curve(Y_test_3, pred_8)

# plot roc curves
plt.style.use('seaborn')

plt.figure(figsize=(20,10))

plt.plot(fpr1, tpr1, linestyle='--',color='orange', label='Initital')
plt.plot(fpr2, tpr2, linestyle='--',color='green', label='Hypertuned')
plt.plot(fpr3, tpr3, linestyle='--',color='red', label='Feature Importance')
plt.plot(fpr4, tpr4, linestyle='--',color='yellow', label='Smote')
plt.plot([0,1],linestyle='--', color = 'lightblue')
# title
plt.title('ROC curves of Random Forest Classification Models')
# x label
plt.xlabel('False Positive Rate')
# y label
plt.ylabel('True Positive rate')
plt.legend(loc='best')

plt.show()

"""### KNN Classifier

1) Initial
"""

# Train, Test, Split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.2,
    random_state=152, stratify=y)

# Define the model
clf_9 = KNeighborsClassifier()

# Fit the model
clf_9.fit(X_train,Y_train)

# Make prediction
pred_9 = clf_9.predict(X_test)
```

```
677.
678.  # Accuracy score
679.  print('Accuracy score of KNN model 1 is:', accuracy_score(Y_test, pred_9))
680.
681.  # Confusion Matrix
682.  conmat_9 = confusion_matrix(Y_test, pred_9)
683.  print(conmat_9)
684.  plt.figure(figsize=(10,8))
685.  plt.title('Confusion Matrix of KNN 1st Model')
686.  sns.heatmap(conmat_9, annot=True, cmap="Purples", fmt="d",cbar=True)
687.
688.  # Classification Report
689.  print('Classification Report of KNN 1st Model')
690.  print(classification_report(Y_test, pred_9))
691.
692.  # ROC-AUC score
693.  print('ROC-AUC score of 1st KNN Model is:', roc_auc_score(Y_test, pred_9))
694.
695.  """2) Hypertuning"""
696.
697.  # Setup the parameters and distributions to sample from: param_dist
698.  param_dist = {"n_neighbors": range(1,15),
699.                "metric": ['euclidean', 'manhattan', 'chebyshev', 'minkowski']}
700.
701.  # Define Hypertuning model
702.  knn_cv = RandomizedSearchCV(estimator=KNeighborsClassifier(), param_distributions=
    param_dist, cv=5, return_train_score=False)
703.
704.  # Fit
705.  knn_cv.fit(X_train,Y_train)
706.
707.  # Best parameters
708.  print('Best parameters for KNN Classifier are:', knn_cv.best_params_)
709.
710.  # Train, Test, Split
711.  X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.2,
    random_state=152, stratify=y)
712.
713.  # Define the model
714.  clf_10 = KNeighborsClassifier(n_neighbors=13, metric='euclidean')
715.
716.  # Fit the model
717.  clf_10.fit(X_train,Y_train)
718.
719.  # Make prediction
720.  pred_10 = clf_10.predict(X_test)
721.
722.  # Accuracy score
723.  print('Accuracy score of KNN model 2 is:', accuracy_score(Y_test, pred_10))
724.
725.  # Confusion Matrix
726.  conmat_10 = confusion_matrix(Y_test, pred_10)
727.  print(conmat_10)
728.  plt.figure(figsize=(10,8))
729.  plt.title('Confusion Matrix of KNN 2nd Model')
730.  sns.heatmap(conmat_10, annot=True, cmap="Purples", fmt="d",cbar=True)
731.
732.  # Classification Report
733.  print('Classification Report of KNN 2nd Model')
734.  print(classification_report(Y_test, pred_10))
735.
736.  # ROC-AUC score
737.  print('ROC-AUC score of 2nd KNN Model is:', roc_auc_score(Y_test, pred_10))
738.
739.  """3) Feature Importance"""
```

```
740.
741.  X_4 = X[['person_age', 'person_income', 'person_home_ownership', 'person_emp_length',
    'loan_amnt']]
742.
743.  # Train, Test, Split
744.  X_train_5, X_test_5, Y_train_5, Y_test_5 = train_test_split(X_4, y, test_size = 0.2,
    random_state=152, stratify=y)
745.
746.  # Define the model
747.  clf_11 = KNeighborsClassifier()
748.
749.  # Fit the model
750.  clf_11.fit(X_train_5,Y_train_5)
751.
752.  # Make prediction
753.  pred_11 = clf_11.predict(X_test_5)
754.
755.  # Accuracy score
756.  print('Accuracy score of KNN model 3 is:', accuracy_score(Y_test_5, pred_11))
757.
758.  # Confusion Matrix
759.  conmat_11 = confusion_matrix(Y_test_5, pred_11)
760.  print(conmat_11)
761.  plt.figure(figsize=(10,8))
762.  plt.title('Confusion Matrix of KNN 3rd Model')
763.  sns.heatmap(conmat_11, annot=True, cmap="Purples", fmt="d",cbar=True)
764.
765.  # Classification Report
766.  print('Classification Report of KNN 3rd Model')
767.  print(classification_report(Y_test_5, pred_11))
768.
769.  # ROC-AUC score
770.  print('ROC-AUC score of 3rd KNN Model is:', roc_auc_score(Y_test_5, pred_11))
771.
772.  """4) Smote"""
773.
774.  # Train, Test, Split
775.  X_train_3, X_test_3, Y_train_3, Y_test_3 = train_test_split(X_smote, y_smote, test_size =
    0.2, random_state=152, stratify=y_smote)
776.
777.  # Define the model
778.  clf_12 = KNeighborsClassifier()
779.
780.  # Fit the model
781.  clf_12.fit(X_train_3,Y_train_3)
782.
783.  # Make prediction
784.  pred_12 = clf_12.predict(X_test_3)
785.
786.  # Accuracy score
787.  print('Accuracy score of KNN model 4 is:', accuracy_score(Y_test_3, pred_12))
788.
789.  # Confusion Matrix
790.  conmat_12 = confusion_matrix(Y_test_3, pred_12)
791.  print(conmat_12)
792.  plt.figure(figsize=(10,8))
793.  plt.title('Confusion Matrix of KNN 4th Model')
794.  sns.heatmap(conmat_12, annot=True, cmap="Purples", fmt="d",cbar=True)
795.
796.  # Classification Report
797.  print('Classification Report of KNN 4th Model')
798.  print(classification_report(Y_test_3, pred_12))
799.
800.  # ROC-AUC score
801.  print('ROC-AUC score of 4th KNN Model is:', roc_auc_score(Y_test_3, pred_12))
```

```
802.
803.  """5) Combine and Compare all ROC curves"""
804.
805.  # roc curve for models
806.  fpr1, tpr1, thresh1 = roc_curve(Y_test, pred_9)
807.  fpr2, tpr2, thresh2 = roc_curve(Y_test, pred_10)
808.  fpr3, tpr3, thresh3 = roc_curve(Y_test_5, pred_11)
809.  fpr4, tpr4, thresh4 = roc_curve(Y_test_3, pred_12)
810.
811.  # plot roc curves
812.  plt.style.use('seaborn')
813.
814.  plt.figure(figsize=(20,10))
815.
816.  plt.plot(fpr1, tpr1, linestyle='--',color='orange', label='Initital')
817.  plt.plot(fpr2, tpr2, linestyle='--',color='green', label='Hypertuned')
818.  plt.plot(fpr3, tpr3, linestyle='--',color='red', label='Feature Importance')
819.  plt.plot(fpr4, tpr4, linestyle='--',color='yellow', label='Smote')
820.  plt.plot([0,1],linestyle='--', color = 'lightblue')
821.  # title
822.  plt.title('ROC curves of KNN Classification Models')
823.  # x label
824.  plt.xlabel('False Positive Rate')
825.  # y label
826.  plt.ylabel('True Positive rate')
827.  plt.legend(loc='best')
828.
829.  plt.show()
830.
831.  """Comparison Results"""
832.
833.  # ROC curves for best model of each algorithm
834.
835.  # roc curve for models
836.  fpr1, tpr1, thresh1 = roc_curve(Y_test_3, pred_4)
837.  fpr2, tpr2, thresh2 = roc_curve(Y_test_3, pred_8)
838.  fpr3, tpr3, thresh3 = roc_curve(Y_test, pred_10)
839.
840.
841.
842.  # plot roc curves
843.  plt.style.use('seaborn')
844.
845.  plt.figure(figsize=(20,10))
846.
847.  plt.plot(fpr1, tpr1, linestyle='--',color='orange', label='Decision Tree')
848.  plt.plot(fpr2, tpr2, linestyle='--',color='green', label='Random Forest')
849.  plt.plot(fpr3, tpr3, linestyle='--',color='red', label='KNN')
850.  plt.plot([0,1],linestyle='--', color = 'lightblue')
851.  # title
852.  plt.title('ROC curves of Best Model of each Machine Learning algorithm')
853.  # x label
854.  plt.xlabel('False Positive Rate')
855.  # y label
856.  plt.ylabel('True Positive rate')
857.  plt.legend(loc='best')
858.
859.  plt.show()
860.
861.  # ROC curves for best model of each algorithm
862.
863.  # roc curve for models
864.  fpr1, tpr1, thresh1 = roc_curve(Y_test, pred_2)
865.  fpr2, tpr2, thresh2 = roc_curve(Y_test, pred_5)
866.  fpr3, tpr3, thresh3 = roc_curve(Y_test, pred_10)
```

```
867.
868.
869.
870.  # plot roc curves
871.  plt.style.use('seaborn')
872.
873.  plt.figure(figsize=(20,10))
874.
875.  plt.plot(fpr1, tpr1, linestyle='--',color='orange', label='Decision Tree')
876.  plt.plot(fpr2, tpr2, linestyle='--',color='green', label='Random Forest')
877.  plt.plot(fpr3, tpr3, linestyle='--',color='red', label='KNN')
878.  plt.plot([0,1],linestyle='--', color = 'lightblue')
879.  # title
880.  plt.title('ROC curves of Best Model of each Machine Learning algorithm (Real Data)')
881.  # x label
882.  plt.xlabel('False Positive Rate')
883.  # y label
884.  plt.ylabel('True Positive rate')
885.  plt.legend(loc='best')
886.
887.  plt.show()
888.
889.  # ROC curves for best model of each algorithm
890.
891.  # roc curve for models
892.  fpr1, tpr1, thresh1 = roc_curve(Y_test_3, pred_4)
893.  fpr2, tpr2, thresh2 = roc_curve(Y_test_3, pred_8)
894.  fpr3, tpr3, thresh3 = roc_curve(Y_test_3, pred_12)
895.
896.
897.
898.  # plot roc curves
899.  plt.style.use('seaborn')
900.
901.  plt.figure(figsize=(20,10))
902.
903.  plt.plot(fpr1, tpr1, linestyle='--',color='orange', label='Decision Tree')
904.  plt.plot(fpr2, tpr2, linestyle='--',color='green', label='Random Forest')
905.  plt.plot(fpr3, tpr3, linestyle='--',color='red', label='KNN')
906.  plt.plot([0,1],linestyle='--', color = 'lightblue')
907.  # title
908.  plt.title('ROC curves of Best Model of each Machine Learning algorithm (Smote Data)')
909.  # x label
910.  plt.xlabel('False Positive Rate')
911.  # y label
912.  plt.ylabel('True Positive rate')
913.  plt.legend(loc='best')
914.
915.  plt.show()
916.
917.
918.
919.
920.
921.
922.
923.
924.
925.
```