Classification of Banking Clients According to their Loan Default Status using Machine Learning Algorithms

Suveshnee Reddy

(210510172)

UNIVERSITY OF KWAZULU-NATAL

SCHOOL OF MATHEMATICS, STATISTICS AND COMPUTER SCIENCE

WESTVILLE CAMPUS, DURBAN, SOUTH AFRICA

Classification of Banking Clients According to their Loan Default Status using Machine Learning Algorithms

by

Suveshnee Reddy
(210510172)

Thesis Supervisors: Dr. Retius Chifurira
Prof. Temesgen Zewotir

A thesis submitted to the University of KwaZulu-Natal in fulfilment

of the requirements for the degree

of

MASTER OF SCIENCE

in

STATISTICS

Disclaimer

This document describes work undertaken as a Masters programme of study at the University of KwaZulu-Natal (UKZN). All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the institution.

[redacted]

Suveshnee Reddy

29 - 06 - 2022

Date

[redacted]

Dr. Retius Chifurira

12/09/2022

Date

Abstract

Loan lending has become crucial for both individuals and companies. For lending institutions, although profitable, it can be very risky due to clients defaulting on their loan agreement. Credit risk assessment is a critical process which is carried out by most lending institutions; it reduces the possibility of lending to clients who will default on their loan repayment, however, it does not eliminate the problem. Thus, a collections process which aims to retrieve unpaid debt is also necessary. With South Africa facing another recession, which was only worsened by the lockdown during the covid-19 pandemic, lending institutions can expect an increase in the number loan defaulters. To counter this increase, changes will have to be made to their policies and processes. Changes can be made to either the loan application procedures (e.g. credit risk assessment, affordability assessment et cetera) or the post disbursal procedures (e.g. collections processes). The aim of this study is to predict whether a client will default on his/her loan, using machine learning algorithms, in order to enhance the collection process of the financial institution under study, where default is defined as missing at least three payments in the first 12 months of the loan being granted. The logistic regression model, decision tree, random forest, support vector machine, Naïve Bayes classifier, k-nearest neighbours algorithm and the artificial neural network were fitted to the balanced dataset. In the researcher's analysis, loan data from a South African financial institution were used for the period August 2019 to December 2019. Variables related to a client's demographics, income, expenses and debt, as well as loan information, were included in the dataset. Exploratory data analysis (EDA) was utilised in order to analyse the dataset and summarise their main characteristics. To reduce the dimensionality of the dataset, two techniques were used, namely principal component analysis (PCA), which is also used to correct the data for multicollinearity, and feature selection (i.e., recursive feature elimination). Each model was fitted to the dataset using these two techniques, and the confusion matrix and metrics such balanced accuracy, true positive ratio, true negative ratio, AUC score and the Gini coefficient were used to evaluate the different models in order to determine which model performed the best and was most suited for this application problem. The results show that when using the PCA approach, the random forest model, which obtained a balanced accuracy score, true positive ratio and AUC score of 0.69, 0.74 and 0.74, respectively, performed the best. The random forest model also performed the best when using the feature selection technique, obtaining a balanced accuracy score, true positive ratio and AUC score of 0.69, 0.74 and 0.75, respectively. When comparing the random forest model using PCA to the random forest model using feature selection, the results showed a marginal difference between each performance metric analysed. The random forest model using PCA utilised 48 variables, whereas the random forest model using feature selection utilised only 18 variables and thus seemed to be more suitable for the classification problem under study. The results of this study are expected to benefit analysts and data scientists in financial institutions who would like to identify the robust machine learning algorithms for classifying defaulting clients. This study is also of significance to policy makers who would want to identify the risk factors associated with loan defaulting clients.

Keywords: Loan Default; Machine learning; logistic regression; decision trees; random forest; k-nearest neighbours; Naïve Bayes algorithm; support vector machines; artificial neural networks; principal component analysis; feature selection; exploratory data analysis

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr Retius Chifurira for the continuous support, advice and motivation. Thank you for guiding me and sharing your invaluable knowledge during my research and writing of thesis.

I sincerely acknowledge the support and inspiration that I received from Prof Delia North throughout my postgraduate studies. Her willingness to assist was always greatly appreciated and has made my journey a lot less stressful.

I would also like to thank Prof Temesgen Zewotir for his advice and assistance.

To my family and friends, thank you for your motivation and support during this journey. A special thank you to my parents for being a pillar of strength, and for their continuous encouragement and love throughout all my endeavours.

Contents

List of Figures

List of Tables

Chapter 1

1 Introduction

In this chapter the background of the study, the literature review, the problem statement, the research questions, the aim and objectives, and the significance of the study are discussed.

1.1 Background

Over time, credit has become crucial in the lives of both individuals and companies, such that it is almost unavoidable for many of them today (Perera & Premaratne, 2016). Individuals throughout the world require loan facilities in order for them to overcome their financial constraints and thus achieve their personal goals. Some individuals depend on loans for basic needs, whereas others require them for luxuries. Companies, both large and small, often require loan facilities in order to function smoothly when faced with financial constraints (Aslam et al., 2019). Loaning of money is beneficial to both the borrower and lender. For lending institutions, although profitable, it can be very risky. This risk involves the inability of the borrower to pay back the loan amount within the agreed-upon time during the loan origination stage (Kwofie et al., 2015). This is often referred to as a loan default. Clients may fail to fulfil their loan obligation for various reasons; for example, some clients cannot afford payments due to mismanagement of funds or additional unexpected costs, whereas others avoid paying their debt, even if they can afford the instalment.

On a daily basis, many individuals, as well as organisations, apply for loans; however, not all loans are approved. The financial institution decides whether the applicants are likely to default on their instalments before granting the loan (Aphale & Shinde, 2020). There is great difficulty in distinguishing between clients who are creditworthy and those who are likely to default on their loan repayment (Marqués et al., 2012). Credit risk assessment, which is used to evaluate the probability of a client defaulting on his or her loan, helps the institution determine whether to grant the client a loan. Effective and thorough evaluation of credit risk reduces possible losses incurred by the financial institution by reducing the possibility of lending to clients who will default on their loan repayment (Sudhamathy, 2016), however, it does not eliminate the problem of clients defaulting on their loan repayment. Although most financial institutions have a process in place to determine whom to lend to and how much to lend, these institutions still expect that a portion of clients will not fulfil their loan obligation. Financial institutions thus need to have a good collections procedure which includes a debt recovery process that aims to retrieve as much unpaid debt as possible.

A collections procedure is a statement detailing the steps that should be taken regarding the collection of due debts. The absence of a good collections process will result in losses for the company due to delinquent accounts. Each lending institution has its own collections process, but all procedures need to follow all laws concerned. The South African financial institution under study currently starts the debt recovery process after the first missed payment by

informing the client telephonically or in writing. If debt in arrears is still not paid after a certain period, debt collectors get involved in the process. The current debt recovery process has proven to be beneficial to the financial institution under study, as a large sum of unpaid debt is retrieved on a monthly basis once the debt recovery process on delinquent accounts begins.

With the adverse conditions affecting the South African economy such as the current recession, aggravated by the lockdown during the covid pandemic, many more companies are closing down while others are down-sizing, resulting in many individuals losing their jobs or working fewer hours. This leads to a loss of income for many households and as a result, lending institutions can expect an increase in the number loan defaulters. Changes can be made to either the loan application procedures (e.g. credit risk assessment, affordability assessment et cetera) or the post disbursal procedures (e.g. collections processes) to counter this increase. The financial institution under study has made numerous changes to their loan application policies and procedures to cater for the increase in defaulters, however, no improvements were made to the post disbursal procedures. Thus, the financial institution now aims to enhance its collections process by sending through a reminder Short Message Service (SMS) or email to clients at the beginning of each month, starting from the month in which the first instalment is due. This enhanced process will likely result in the financial institution retrieving more unpaid debt and reducing the number of clients who miss payments on their loan. The financial institution was initially considering sending reminder emails to all clients who received a loan as it is not costly, however, the institution decided to use machine learning techniques to target a specific population instead.

Machine learning algorithms identify patterns in data in order to build predictive models. Recently, classification methods that use machine learning algorithms have become more popular amongst researchers and institutes; these methods are used by analysts in financial institutions to identify clients who are likely to default on payment of loans by predicting an individual's credit score, using historical data (Aslam et al., 2019). A similar method can be used to improve the collections process by identifying clients who are likely to default on their loan repayment using information that is available at the time when the loan is disbursed (The financial institution under study defines default as missing at least three payments in the first 12 months of the loan being granted). According to Ereiz (2019), often predicting that clients will default when they actually do not (i.e., a false positive), is not as costly as predicting that clients will not default when they actually do (i.e., a false negative). Since the additional step which the financial institution wants to add to their collections process is not costly as it involves sending emails and SMSs to the client, the financial institution is not too concerned with misclassifying some non-defaulters as defaulters; the institutions main concern is correctly classifying clients who default

1.2 Literature review

Financial institutions have become a crucial part of our daily lives in the digital era. The number of individuals wanting loans has increased in recent years (Radhika et al., 2021). As a result, the demand for loans from financial intuitions has increased. According to Madaan et al. (2021), loan lending is a vital source of income for financial institutions; however, it is also their main source of financial risk. Most of a bank's assets are obtained by using the profits

earned from granting loans (Purswani et al, 2021). It is thus important for the financial institution to assess the credit risk of clients before lending to them (Kwofie et al., 2015). An applicant's credit risk is assessed in order to assign them into one of two classes, namely good or bad (i.e., not default or default) (Ince & Aktan, 2009), which indicates how the bank should treat the client (e.g., whether the bank should approve or decline the application). Sudhamathy (2016) opines that this will assist banks to minimise their losses. This chapter review provides a brief summary of previous work done on predicting whether clients will default on their loan, as well as outlines important features that are commonly used to predict a client's default status and touches on the common problem of imbalance in default datasets. It also provides a review on several studies relating to the application of classification algorithms. In this study, machine learning algorithms will be used to identify clients who will likely default on their loan so that an enhanced collections method can be used.

Marqués et al. (2012) states that it is difficult to determine which clients are likely to default and which of them will be reliable borrowers. The features included in a model have a significant influence on how well the model performs. Bayraci and Susuz (2019), Kadam et al. (2021) and Kwofie et al. (2015), among others, mention variables included in their default models (e.g., age, gender, income, and credit information). The variables mentioned in these papers were commonly used in multiple other research papers that aimed to predict a client's default status. Chen and Zhang (2021) and Radhika et al. (2021) utilise feature selection methods to sift out important features for their credit default models. Chen and Zhang (2021), Radhika et al. (2021), Sudhamathy (2016) and Zhou and Wang (2012), among others, mention the common problem of imbalance in the datasets when predicting loan default and discuss and explore possible solutions to the problem. The difference in misclassification costs of false positives and false negatives when predicting a client's default status is explained by Ince and Aktan (2009). Mwangi (2016) discusses the importance of lending institutions having a good collections process in place in order to collect overdue debt from its borrowers.

Bayraci and Susuz (2019) constructed a neural network model to determine whether an applicant is good or bad and compared the model to several other classification models, namely logistic regression, decision tree, Naïve Bayes, and the support vector machine. The explanatory variables used to predict the clients' default status were grouped into four groups, namely demographical characteristics, employment characteristics, credit characteristics, and credit history. These authors concluded that the deep learning model, namely the neural network, performs better than the other models on bigger datasets.

Kadam et al. (2021) mention that features such as gender, marital status, education, employment status, number of dependants, income, loan amount, and credit history were utilised when predicting whether clients would default on their loan. A comparison between the Naïve Bayes algorithm and the Support Vector Machine (SVM) was performed, and results showed that the Naïve Bayes classifier performed best.

Chen and Zhang (2021) reviewed the artificial neural network, k-nearest neighbour, decision tree, support vector machine and logistic regression. They aimed to predict automobile credit defaulters. Feature selection was utilised in order to identify important features for their models. Of the features selected, it was found that date of birth, employment type, disbursed amount and asset cost were ranked most important when predicting the 'default' target variable. The authors used the SMOTE method to solve the imbalanced dataset problem; however, they

suggest that the model's performance did not improve. It was concluded that all six models could be used to predict the default of automobile and although the decision tree obtained the highest accuracy score of 0.79, the SVM had the best overall performance.

Zhou and Wang (2012) mention that many loan default datasets are highly skewed, with the majority of cases falling within the same class. They propose an improved random forest algorithm in which weights are allocated to decision trees in the random forest during tree aggregation for prediction. Previous performance, that is, out-of-bag errors during training, is used to compute the weights. The weighted majority in the ensemble of trees in the random forest is used to make predictions. The proposed model was compared to the original random forest, the SVM, the K-Nearest Neighbours (K-NN) and the decision tree; the results indicate that the proposed random forest algorithm obtained a better overall accuracy score, as well as balanced accuracy score, than all the other models mentioned.

Chang et al. (2015) adopted the logistic regression model, SVM and Naïve Bayes classifier to build a loan default prediction model. They discuss sensitivity, specificity, accuracy and precision. These authors noted that accuracy would not reflect the model's true performance, as the dataset was imbalanced. From the results obtained, the Naïve Bayes classifier with Gaussian performed the best, obtaining a sensitivity score of 80.1%.

Financial institutions are faced with many classification problems on a daily basis. Credit scoring, which can be used to determine whether a loan application should be approved or declined is one type of classification problem. Ince and Aktan (2009) mention other classification problems related to decision–making in business, for example, financial forecasting, fraud detection, marketing strategy, and process control. The classification problem under study involves allocating clients to one of two classes i.e., default or not default, in order to determine whether the enhanced collections method should be used.

According to Lee et al. (2002), it is possible to use techniques such as statistical methods and artificial intelligence algorithms to solve classification problems. Common statistical methods such as linear discriminant analysis, logistic regression, and their variations, though, have several limitations when applied to credit scoring problems. Ince and Aktan (2009) are of the opinion that limitations associated with these techniques are the following: they are ineffective when high-dimensional inputs are present, and the sample size is small; these methods assume linear separability and that the normality assumption is met; and there is difficulty in automating the process and designing a continuous update flow. Yang (2007) mentions that statistical models are often unable to adapt to population changes over time; as a result, these models may have to be reconstructed. Artificial intelligence techniques, which include machine learning, can be used instead of discriminant analysis and logistic regression when the dependent and independent variables display complex nonlinear relationships (Ince & Aktan, 2009). According to Madaan et al. (2021), many researchers and bank authorities have recently chosen to train classifiers based on numerous machine learning and deep learning algorithms in order to automatically predict an applicant's credit score, as it makes the process significantly easier.

There are several steps involved when building a machine learning model. (Sudhamathy, 2016) discusses these steps, which include data selection, pre-processing, treatment of outliers, imputations removal, splitting the dataset between the training and test set, and balancing the training set; the features selection step, building the classification model, predicting class labels

of the test set, and evaluating predictions are also discussed. The author constructed a decision tree to predict whether the client is likely to default.

Madaan et al. (2021) used tree-based machine learning algorithms to predict whether or not new clients are likely to default on their loan in order to determine whether to lend to the clients. A comprehensive and comparative analysis between the decision tree algorithm and random forest algorithm was done. The results showed that the random forest algorithm performed better than the decision tree algorithm. The random forest obtained an accuracy score of 80%, whereas the decision tree algorithm obtained an accuracy score of 73%. The authors mention that the random forest model, which is an ensemble of several decision trees, has the following advantages over other machine learning algorithms: it is immune to overfitting; it can produce accurate classification or regression results; and it is more efficient on large datasets.

Ince and Aktan (2009) analysed the performance of credit scoring models in order to identify clients who are either "good" or "bad", using both traditional and artificial intelligence methods. They compared the results that were obtained by using discriminant analysis, logistic regression, neural network, classification, and regression trees (CART). The authors noted that misclassification costs associated with false positives were much lower compared to those of false negatives. From the results, the CART model obtained a higher accuracy score in comparison with discriminant analysis, logistic regression, and neural networks. However, the neural network credit scoring model obtained the lowest percentage of false negatives, which is associated with higher misclassification costs. Therefore, it was concluded that the neural network has better credit-scoring capabilities overall.

Radhika et al. (2021) aimed to predict a client's creditworthiness by using the following algorithms: the K-NN classifier, random forest classifier, decision tree and logistic regression. The SMOTE and NearMiss techniques were utilised to cater for the data imbalance. From the results, it was found that the random forest model performed the best. To improve performance and solve the problem of overfitting, the authors combined all models into a single model by utilising the voting method. All the model's votes were considered and the class with the maximum votes was the final model prediction.

Breeden (2020) discusses multiple machine learning methods that are available, including random forest, neural networks, logistic regression, k-nearest neighbours, support vector machines, Naïve Bayes, stochastic gradient boosting et cetera and indicates that it is impossible to declare a single best method. Breeden (2020) states that methods have strengths and weaknesses, depending on the application problem, and that the best method for an application problem is usually a combination of elements from different methods.

This study departs from the study of Chen and Zhang (2021) by comparing the PCA approach to the feature selection approach (recursive feature elimination) for dimensionality reduction, when selecting the best model among several classification algorthims. To the best of the researchers knowledge, there is limited use of machine learning models in financial institutions which help enhance the collections process by identifying clients who are likely to default on their loan as soon as the loan is granted. In the literature, we could not find an application showing a comparison of machine learning models using the PCA and feature selection approaches for dimensionality reduction on a dataset from a financial institution which aimed to enhance their collections process, thus this study will fill the gap in literature.

1.3 Problem statement

Although the financial institution under study does have a collection process for unpaid debt in place, this process only begins once the client has missed a payment. The sooner clients who are likely to default on their debt obligation are identified and contacted, the greater the chances of them not actually defaulting on their loan. Therefore, the researcher aims to identify a 'bad' population who will likely default on their loan obligation by using machine learning algorithms. This 'bad' population, who the researcher will refer to as the 'default' population, is defined as clients who fail to make payment for at least three months during the first 12 months of the loan being granted. This identification needs to take place at the time the loan is granted, as it will provide the institution with the opportunity to enhance the collection process by sending friendly reminders to these clients on a monthly basis starting from the first month the loan is granted. This enhanced process will likely result in the financial institution retrieving more unpaid debt and reducing the number of clients who default on their loans.

1.4 Research questions, aim and objectives

In this study, answers to the following research questions are provided:

1) Which classification algorithms are able to identify correctly a sufficient proportion of clients who defaulted on their loan?
2) Which classification model is most appropriate?
3) Which features are of most importance when predicting the default status of a client?

The aim is to identify the robust machine learning algorithm from logistic regression, the decision tree, random forest, support vector machine, Naïve Bayes classifier, k-nearest neighbours and the artificial neural network, using the PCA technique and the feature selection technique for dimensionality reduction, to predict which clients will miss at least three payments in the first 12 months of the loan being granted.

This is achieved by:

1) Exploratoratory data analysis to review important features of the applicants
2) Dimensionality reduction using two techniques, namely, principal component analysis (PCA), which is also used to correct the data for multicollinearity, and feature selection.
3) Fitting classification algorithms namely logistic regression, decision tree, random forest, support vector machines, the Naïve Bayes classifier, k-nearest neighbours and the artificial neural network to the dataset using both dimensionality reduction approaches
4) Selecting the best fitting classification method for each dimensionality reduction approach using metrics such as balanced accuracy, true positive ratio, true negative ratio, precision, negative predictive value, Area under the ROC curve (AUC score) and the Gini coefficient

5) Selecting the overall best fitting model using the following metrics: balanced accuracy, true positive ratio, true negative ratio, precision, negative predictive value, Area under the ROC curve (AUC score) and the Gini coefficient
6) Identifying the most robust variables for classifying default clients.

1.5 Significance of the study

The results of this study are expected to benefit analysts and data scientists in financial institutions who would like to identify the robust machine learning algorithms for classifying defaulting clients. This study is also of significance to policy makers who would want to identify the risk factors associated with loan defaulting clients.

1.6 Theoretical Framework

Machine learning is a subset of artificial intelligence that allows systems to automatically learn patterns in the data and improve outcomes through experience without being explicitly programmed. Machine learning problems can be divided into different categories. The two main categories of machine learning are supervised learning and unsupervised learning. In supervised learning, the model learns patterns from a labelled dataset and the trained model is used to make predictions on unseen data. Unsupervised learning (e.g., clustering) does not require a labelled dataset. The aim of unsupervised learning is to find structure, hidden relationships and patterns from the input data. In this study, the researcher focuses on supervised learning. There are two types of supervised learning, namely classification and regression. A classification problem has a categorical target variable, for example a client's default status, whereas a regression problem has a real value target variable, for example, house prices. The aim of this study is to build classification models, using labelled data, to predict a client's default status.

The variable information in this study was obtained during the application and offer process, which includes information captured by clients, bureau information, bank statement and payslip information, as well as internal calculations used to determine the final offer. Variables used in a machine learning model can be categorised as either numerical or categorical. A numerical variable is quantitative and can either be discrete, (i.e., a whole number) for example, number of dependants, or it can be continuous, (i.e., it can take any value in a given range) for example, weight. A categorical variable is qualitative and can either be nominal, (i.e., unordered) for example, male and female, or it can be ordinal, (i.e., ordered) for example, small, medium and large. When a large number of variables are included in a dataset, principal component analysis (PCA) and feature selection can be used to reduce the dimensionality of a dataset. PCA is used to transform a larger set of variables into a smaller set of variables whilst retaining most information. By using PCA, one can include the minimum number of principal components needed to explain a certain percentage of the variance. The new principal components formed are uncorrelated. Feature selection is a dimensionality reduction technique

that aims to select a subset of features from the original set of features. This is achieved by removing features that are redundant, irrelevant, and noisy. There are several feature selection methods that can be used to reduce the number of features in the model. The three main feature selection categories are the filter method, wrapper method, and embedded method.

In this study, several classification machine learning algorithms, namely logistic regression, decision trees, random forest, k-nearest neighbours, the Naïve Bayes algorithm, support vector machines and artificial neural networks, are fitted to the default dataset using the PCA and feature selection approaches. A summary of each of these classification models is provided in Chapter 3. Evaluation metrics, such as accuracy, balanced accuracy, recall, specificity, precision, the negative predictive value, Gini, and the auc score, are used to evaluate and compare the performance of the machine learning models in order to identify the model which performed the best. Each of these evaluation metrics are discussed in Chapter 3.

1.7 Project layout

This thesis comprises six chapters. Chapter 1 provides an overview of this study and the literature review. The dataset, provided by a South African financial institution, is discussed and exploratory data analysis, used to analyse and investigate the dataset, is presented in Chapter 2. The theory of the classification models and the evaluation metrics is summarised in Chapter 3. Chapter 4 provides the empirical results obtained when fitting the classification models to the dataset using the PCA approach. Chapter 5 presents the empirical results obtained when fitting the classification models to the dataset using feature selection. Chapter 6 provides a summary of the main findings and concludes the study.

In this chapter the researcher discussed the background of the study, provided a literature review, the problem statement, the research questions, the aim and objectives, and the significance of the study. From the literature, the researcher was unable to find an application which showed a comparison of machine learning models using the principal component analysis and feature selection approaches for dimensionality reduction on a dataset from a financial institution which aimed to enhance their collections process. This study will fill the gap in literature. It is important to have an in depth understanding of the dataset used in this study. In the next chapter the dataset is analysed using exploratory data analytics. The characteristics of the dataset will inform the researcher on which models should be considered when predicting whether a client will default on his/her loan.

CHAPTER 2

2 Exploratory Data Analytics

In this chapter, the dataset used in this study is discussed, all variables under study which are possibly associated with the 'default' target variable are listed and a description is provided, and exploratory data analytics, used to analyse and investigate the dataset, are presented.

2.1 Data

The data used in this study were obtained from a South African financial institution. Applications that were disbursed during the period August 2019 to December 2019 were included in the dataset, which consists of 48 338 disbursed applications and 48 variables. The variable information was obtained during the application and offer process, which includes information captured by clients, bureau information, bank statement and payslip information, as well as internal calculations used to determine the final offer. Of the 48 variables, 32 variables are categorical and 16 are numerical. Using these variables, the researcher aims to predict whether clients will default on their loan. A client is classed as a default if the client missed at least three payments in the first 12 months of the loan being disbursed.

The independent variables used to predict the default class have been grouped into sub-categories, namely demographics, client information, loan information, income, expenses, and debt. In Table 2.1, the researcher indicates and outlines the description of the demographic variables and categorises them as either numerical or categorical.

Table 2. 1: Description of demographic variables under study

Variable	Description	Categorical/Numerical
Age	Age at time of application	Numerical
Number of dependants	How many dependants does the applicant have?	Numerical
Years with current employer	Number of years the applicant has been working for his/her current employer at the time of application	Numerical
Gender	Gender of applicant – male or female	Categorical
Married	Indicates whether applicant is married or unmarried	Categorical
Property owner	Does the applicant own a property? (Yes/No)	Categorical

Table 2.2 presents the client information variables, provides a description for each variable, and indicates whether these variables are numerical or categorical.

Table 2. 2: Description of client information variables under study

Variable	Description	Categorical/Numerical
Client type	A client is either a new, reload or multiple loan client. New – Applicant is new to financial institution. Reload – Applicant chooses to consolidate a current loan with the financial institution. Current – Applicant currently a has loan with the financial institution and is taking an additional loan.	Categorical
External subsequent lending	Did the applicant take up a loan with another company less than 45 days before the loan included in our dataset? (Yes or No)	Categorical
Int/Ext client	A client is either Internal or External. Internal client – main bank is financial institution under study External client – main bank is not financial institution under study	Categorical
Salary bank	Bank which applicant's salary is paid into	Categorical
Staff member	Does the applicant work for the financial institution under study? (Yes or No)	Categorical
Weekly/Monthly	The applicant either earns a weekly wage or a monthly salary (fortnightly earners do not qualify for a loan).	Categorical

Table 2.3 presents the loan information variables, provides a description for each variable and classifies them as either numerical or categorical.

Table 2. 3: Description of loan information variables under study

Variable	Description	Categorical/Numerical
Limiting rule	Rule that limits a client's affordability	Categorical
Loan purpose	The reason the applicant needed a loan (consolidate debt, a family crisis, housing and related, other emergency, et cetera)	Categorical
Lower offer	Did the applicant receive a lower offer once information was verified compared to the initial offer received during the application process? (Yes or No)	Categorical
Product taken	The type of product taken: home loan (HL), personal loan (PL), staff, vehicle loan (VL)	Categorical
Taking max	Did the applicant take the maximum amount offered? (Yes or No)	Categorical
% instalment to income allowed	The maximum ratio of instalment to income that the client is allowed	Numerical
% instalment to income taken	Loan instalment taken as a percentage of income	Numerical
% total taken up	The amount taken as a percentage of the amount offered	Numerical
Instalment /Disposable income	New instalment amount divided by cashflow at the end of the month (Cashflow = income minus expenses minus debt)	Numerical
Max offer	The maximum amount offered to the applicant by the financial institution	Numerical

In Table 2.4, the income, expenses and debt variables are indicated, outlined, and categorised as either numerical or categorical variables.

Table 2. 4: Description of income, expenses and debt variables under study

Variable	Description	Categorical/Numerical
Arrears	Is the client in arrears for any of his/her loans? (Yes or No)	Categorical
Credit card	Does the client have a credit card? (Yes or No)	Categorical
Credit inactive	The client does not have a credit history (Yes or No)	Categorical
Home loan	Does the client have a home loan? (Yes or No)	Categorical
Instalment loan	Does the client have an instalment loan? (Yes or No)	Categorical
Insurance	Did the client capture any insurance expenses? (Yes or No)	Categorical
IntConsol	Is the client consolidating a loan from the financial institution? (Yes or No)	Categorical
Internal living expenses rule	Was the internal living expenses rule the final living expenses amount used in the affordability calculation? (Yes or No)	Categorical
Medical aid	Did the client capture any medical aid expenses? (Yes or No)	Categorical
Overtime	Did the client capture any overtime? (Yes or No)	Categorical
Payslip expenses	Did the client capture any expenses found on payslip? (Yes or No)	Categorical
Pensionprovident	Did the client capture any pension/provident contribution? (Yes or No)	Categorical
Permanent allowances	Did the client capture any permanent allowances? (Yes or No)	Categorical
Personal loan	Does the client have a personal loan? (Yes or No)	Categorical
Revolving credit	Does the client have revolving credit? (Yes or No)	Categorical
Union fees	Did the client capture any union fees? (Yes or No)	Categorical
Unpaids	Does the client have any unpaid debts? (Yes or No)	Categorical
Vehicle loan	Does the client have a vehicle loan? (Yes or No)	Categorical
Calc disposable income/Net income	Cashflow at the end of the month before consolidations (income minus	Numerical

	expenses minus debt) as a percentage of net income	
Debt to income ratio	Debt as a percentage of gross income	Numerical
Debt/Net income	Debt as a percentage of net income	Numerical
Disposable income /Basic	Cashflow at the end of the month (income minus expenses minus debt) as a percentage of basic salary	Numerical
External consolidations/Amount Taken	The total external loan amount consolidated (loan from another financial institution) divided by the new loan amount disbursed	Numerical
Final disposable income/Net income	Final cashflow at the end of the month after consolidations (income minus expenses minus debt plus consolidations) as a percentage of net income	Numerical
Internal consolidations/Amount Taken	The total internal loan amount consolidated (loan from the same financial institution) divided by the new loan amount disbursed	Numerical
Total consolidations/Amount Taken	The total amount consolidated divided by the new loan amount disbursed	Numerical

2.2 Data exploration

The main aim of the study (which was discussed in Chapter one) is to predict whether clients will default on their loan. The variable default is derived from whether a client misses at least three payments in the first 12 months of the loan being disbursed. Figure 2.1 shows the percentage of clients who did and did not default on their loan.

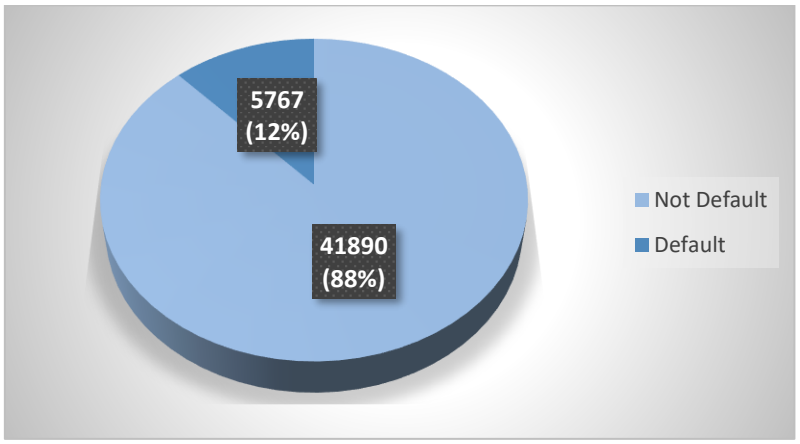

Figure 2. 1: Percentage of clients in each default class

From Figure 2.1, 12% of the clients under study defaulted whereas 88% of clients did not default. This indicates that the majority class (i.e., not default) is significantly larger than the minority class (i.e., default). Therefore, there seems to be an imbalance in the dataset. When a dataset is imbalanced, a high accuracy can be obtained by only predicting the majority class. Since many machine learning algorithms are designed to maximise accuracy by reducing errors, the algorithm is often biased towards the majority class and is more likely to misclassify the minority class than the majority class. In this study, the main focus is on the default class, which is the minority class. The researcher will thus have to cater for the imbalance in the dataset for the models to be able to produce satisfactory results. Balanced weighting and Synthetic Minority Oversampling Technique (SMOTE) are popular techniques used to cater for the imbalance in the dataset. The "balanced" class weight method adjusts weights automatically by using the class values (y), such that the weights are inversely proportional to the input data's class frequencies. SMOTE aims to balance class distribution by creating new synthetic objects in the minority class (i.e., default). Using either of these methods should improve the results obtained when the dataset is imbalanced.

The researcher then investigates whether the demographics, client information, income, expenses, debt and loan information variables have an influence on the 'default' target variable. Figures 2.2 to 2.7 present the distribution of each variable and the percentage of clients who defaulted in each category of each variable.

The distribution of variables under study are explored in order to identify variables that comprise of categories which include a negligible portion of clients. Ideally, the researcher wants each category in a variable to include a meaningful portion of clients under study. If a variable has two categories and a majority of clients fall within one of the categories, whereas a negligible portion of clients fall within the other category, excluding the variable may be considered, as this variable will likely add little value to the model, unless the variable has an exceptional influence on the 'default' target variable. If a variable has more than two categories and at least one of the categories comprise a negligible portion of clients under study, the researcher may consider combining that category with another category within the variable that has a similar default rate.

The percentage of clients who default is analysed in order to determine whether there is a significant difference in the default rate between the different categories in a variable. The default rate indicates how well a group of clients performed. The lower the default rate, the better the performance, whereas the higher the default rate, the worse the performance. If there is a significant difference in default rate (performance) between clients in different categories of a variable, it can be concluded that the variable is likely to be associated with default.

Figure 2.2 shows the distribution and percentage of clients defaulting in each category within each variable in the demographics subgroup, which includes the following variables: gender, married and property owner.

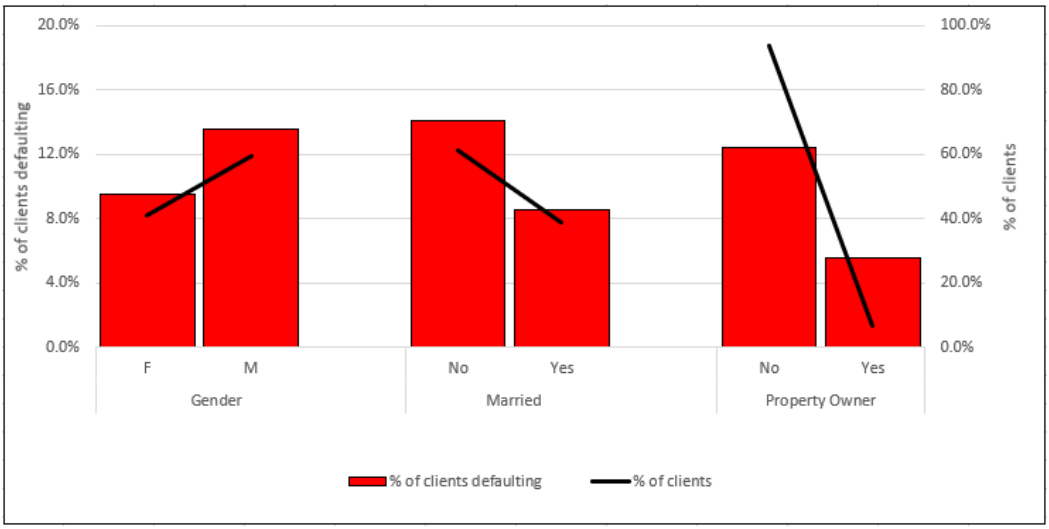

Figure 2. 2: Distribution and percentage of clients defaulting for variables in the demographics subgroup

From the gender variable in Figure 2.2, 59.3% of the clients under study are males, whereas 40.7% of the clients are females; 13.6% of males default, whereas 9.5% of females default. Therefore, there seems to be a significant difference between the percentage of males and females defaulting. There is thus a possibility of gender being associated with default.

Figure 2.2 displays the married variable which shows that 61.1% of clients are not married and the remaining 38.9% are married. The average default rate of unmarried and married clients is 14.1%, and 8.5%, respectively. This suggests variation in the default rates between the married and unmarried clients under study. Thus, there is possibly a relationship between the married variable and default.

From the property owner variable presented in Figure 2.2, 6.5% of clients own a property. The default rate of clients who own a property is 5.5%, whereas clients who do not own a property have a default rate of 12.4%. The 6.9% difference in default rate indicates that the performance of clients who own a property and clients who do not own a property seem to vary substantially. Thus, property owner may have an influence on the 'default' target variable.

Client information variables were then explored. Figure 2.3 shows the distribution and percentage of clients defaulting in each category within each variable in the client information subgroup, which includes the following variables: weekly/monthly, int/ext client, client type, staff member, salary bank, and external subsequent lending.

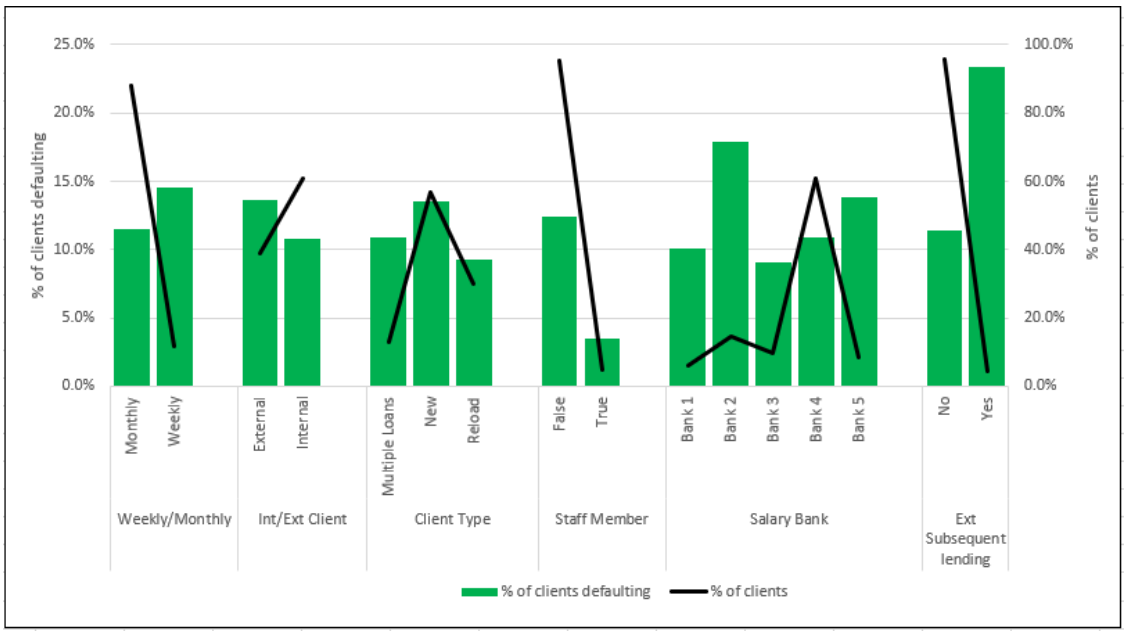

Figure 2. 3: Distribution and percentage of clients defaulting for variables in the client information subgroup

The weekly/monthly variable in Figure 2.3 shows that 88.1% of clients are monthly earners, whereas 11.9% are weekly earners. Monthly earners have a default rate of 11.6% and weekly earners have a default rate of 14.6%. As there seems to be a difference in performance between monthly earners and weekly earners, it is possible that a relationship between the weekly/monthly variable and the 'default' target variable exists.

From the int/ext client variable in Figure 2.3, 60.9% of clients are internal clients, whereas 39.1% of clients are external. Figure 2.3 also indicates that on average, 10.8% of internal clients default, whereas 13.6% of external clients default. This suggests that internal clients tend to default on their loan less often than external clients. Therefore, the variable int/ext client may likely be associated with default.

Figure 2.3 displays the client type variable, which shows that the default rates of current, new, and reload clients are 10.9%, 13.6% and 9.2%, respectively. This suggests that new clients, who comprise 56.8% of the total number of clients, perform worse than current and reload clients. Hence, it can be concluded that the client type variable possibly has an influence on default.

The staff member variable presented in Figure 2.3 indicates that 4.9% of clients are staff members and 95.1% are not. Figure 2.3 also shows that staff members perform exceptionally well, as indicated by their default rate of 3.5%, whereas a significantly higher percentage of non-staff members, that is 12.4%, default. Thus, the staff member variable likely has a relationship with the 'default' target variable. Although the percentage of staff members is small, we may not want to exclude the variable as this population performs significantly better than the remaining population.

The salary bank variable depicted in Figure 2.3 comprises five categories. Clients in Bank 1, Bank 2, Bank 3, Bank 4 and Bank 5 have a default rate of 10.1%, 17.9%, 9.1% 10.8% and 13.8%, respectively. This provides evidence of a substantial difference in default rate between

the salary bank categories. Thus, there is a possibility of salary bank having an influence on default.

The ext subsequent lending variable in Figure 2.3 shows that 4.5% of clients have external subsequent lending and 23.4% of these clients default, whereas 95.5% of clients do not have external subsequent lending and 11.4% of these clients default. This indicates that clients with external subsequent lending seem to perform significantly worse than clients without external subsequent lending. Although the percentage of clients with external subsequent lending is only 4.5%, the variable may still influence the model as there seems to be a very strong association between ext subsequent lending and default.

The loan information variables are then examined. Figure 2.4 presents the distribution and percentage of clients defaulting in each category within each variable in the loan information subgroup, which includes the following variables: limiting rule, lower offer, taking max, product taken and loan purpose.

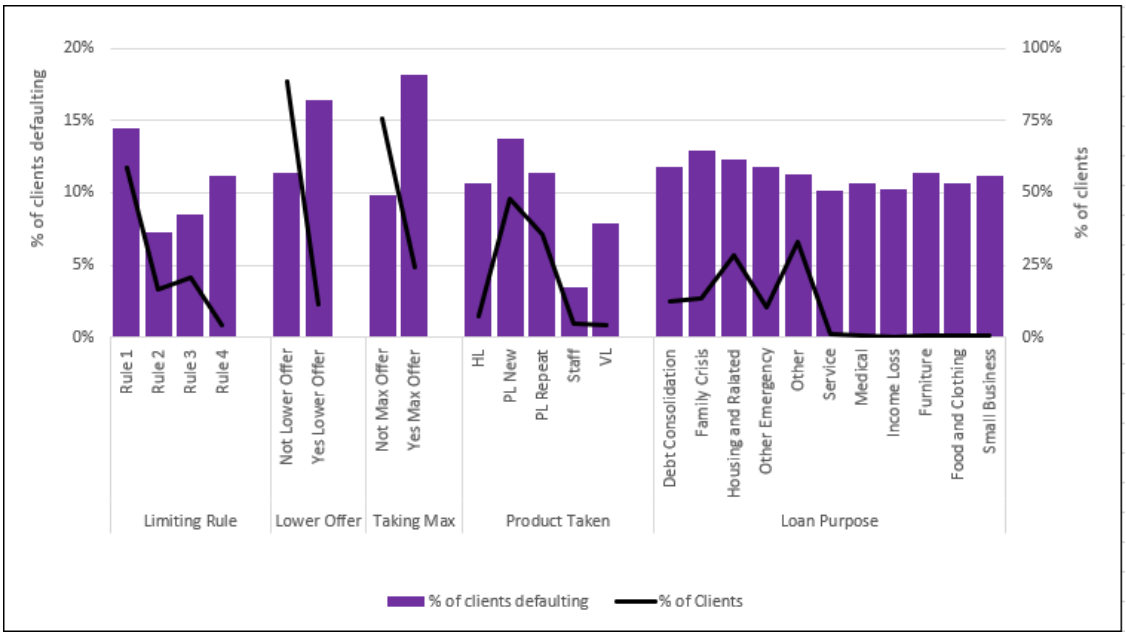

Figure 2. 4: Distribution and percentage of clients defaulting for variables in the loan information subgroup

The limiting rule variable presented in Figure 2.4 shows that the percentage of clients under study in the Rule 1, Rule 2, Rule 3 and Rule 4 categories is 58.7%, 16.5%, 20.6% and 4.2%, respectively, and their corresponding default rates are 14.5%, 7.3%, 8.5% and 11.2%, respectively. As only 4.2% of clients under study fall within the Rule 4 category, grouping this category with another category may be considered. Clients falling within the Rule 1 category perform significantly worse than clients in all other categories; therefore, it is likely that the limiting rule variable and the 'default' target variable are associated.

From the lower offer variable in Figure 2.4, 11.3% of clients under study receive a lower offer and 16.4% of these clients default on their loan, whereas 88.7% of clients do not receive a lower offer and only 11.3% of these clients default on their loan. As there seems to be a

considerable difference in performance between clients who receive a lower offer and clients who do not, it is possible that a relationship between lower offer and default exists.

From Figure 2.4, the taking max variable shows that 24.4% of clients under study take the maximum amount offered and 75.6% of clients do not. Figure 2.4 also shows that clients who take the maximum amount offered perform poorly, as indicated by their default rate of 18.2%, whereas clients who do not take the maximum amount perform significantly better, indicated by their default rate of 9.9%. Hence, it can be concluded that the taking max variable likely has an influence on default.

Figure 2.4 displays the product type variable, which shows that the portion of clients under study falling within the hl, pl new, pl repeat, staff and vl categories is 7.4%, 48.1%, 35.5%, 4.9% and 4.1%, respectively, with average default rates of 10.7%, 13.8%, 11.4%, 3.5% and 7.9%, respectively. This suggests variation in the default rates between these categories. Thus, there is a possibility of a relationship between product type and default. From Figure 2.4, we observe that a small portion of clients under study fall within the vl and staff categories. Therefore, grouping categories in this variable may be considered, however, it is worth noting that clients in these categories perform better than clients in all other categories within the variable.

The loan purpose variable depicted in Figure 2.4 shows that each of the following categories consist of less than 1% of the total clients under study: service, medical, income loss, furniture, food and clothing, and small business. Therefore, grouping these categories with the ‘other’ category may be considered as the percentage of clients falling within each of these categories is negligible and there is no significant difference in default rates between these categories. Figure 2.4 shows that the default rates of clients falling within the remaining categories, namely, consolidate debt, family crisis, housing and related, other and other emergency categories are 11.8%, 12.9%, 12.4%, 11.8%, and 11.3%, respectively. This indicates that there does not appear to be a significant difference in the default rates between these categories. Thus, it is unlikely that there is a strong relationship between loan purpose and the ‘default’ target variable.

The income and expense variables were then analysed. Figure 2.5 displays the distribution and percentage of clients defaulting in each category within each variable in the income and expenses subgroup, which includes the following variables: payslip expenses, internal living expenses rule, permanent allowances, overtime, union fees, insurance, and pensionprovident.

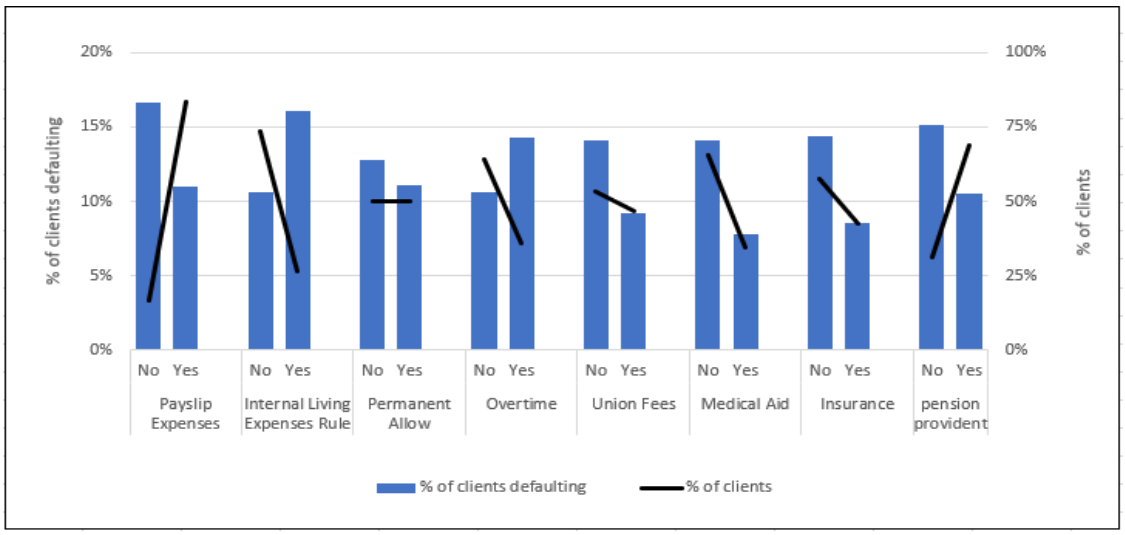

Figure 2. 5: Distribution and percentage of clients defaulting for variables in the income and expenses subgroup

From the payslip expenses variable displayed in Figure 2.5, 83.5% of clients have payslip expenses, whereas 16.5% of clients do not have any payslip expenses. 11.0% of clients who have payslip expenses default, whereas 16.6% of clients who do not have payslip expenses default. This indicates that clients with payslips expenses have a remarkably lower default rate compared to clients with no payslip expenses. Hence, the payslip expenses variable is likely associated with default.

The internal living expenses variable in Figure 2.5 shows that 26.4% of clients utilise the internal living expense rule and 16.1% of these clients default, whereas 73.6% of clients do not utilise the internal living expense rule and 10.6% of these clients default. As there seems to be a considerable difference in performance between clients who utilise the rule and clients who do not, it is possible that a relationship between the internal living expense rule and default exists.

Figure 2.5 displays the permanent allowances variable, which shows that the number of clients who have and clients who do not have permanent allowances is equally distributed between the two categories, with a default rate of 11.1% and 12.8%, respectively. This indicates that there is little variation in performance between clients who have and clients who do not have permanent allowances. Hence, it can be concluded that it is unlikely that the permanent allowances variable has a strong influence on default.

From the overtime variable in Figure 2.5, 35.8% of clients under study earn overtime. Clients earning overtime have a default rate of 14.3%, whereas clients who do not earn overtime have a default rate of 10.6%. Therefore, overtime earners seem to perform worse than clients who do not earn overtime. Thus, overtime and default seem to be associated.

The union fees variable in Figure 2.5 shows that 46.7% of the clients under study pay union fees and 9.2% of these clients default on their loan, whereas 14.1% of clients who do not pay union fees default on their loan. Therefore, there could be a relationship between default and union fees, as clients paying union fees seem to perform noticeably better than clients who do not pay union fees.

The medical aid variable presented in Figure 2.5 indicates that 34.5% of clients pay medical aid, whereas 65.5% do not pay medical aid. From Figure 2.5, clients who pay medical aid have a default rate of 7.8%, which is substantially lower than the 14.1% default rate of clients who do not pay medical aid. Therefore, an association between default and the medical aid variable seems to exist.

From the insurance variable in Figure 2.5, 42.5% of clients under study pay for insurance. Clients who pay insurance have a default rate of 8.6%, whereas clients who do not pay insurance have a default rate of 14.4%. Hence, the difference in default rate between clients who pay and clients who do not pay insurance is 5.4%, which is significant. Thus, there is a possibility that the insurance variable has an influence on default.

From the pensionprovident variable displayed in Figure 2.5, 68.9% of clients under study contribute towards their pension/provident fund. These clients have a default rate of 10.5%, whereas clients who do not contribute, have a default rate of 15.1%. Since there seems to be a significant different between clients who contribute towards their pension/provident fund and clients who don't (i.e., 4.6%), the pensionprovident variable is likely associated with default.

The debt variables under study are then explored. Figure 2.6 shows the distribution and percentage of clients defaulting in each category within each variable in the debt subgroup, which includes the following variables: personal loan, home loan, vehicle loan, instalment loan, credit card, and revolving credit.

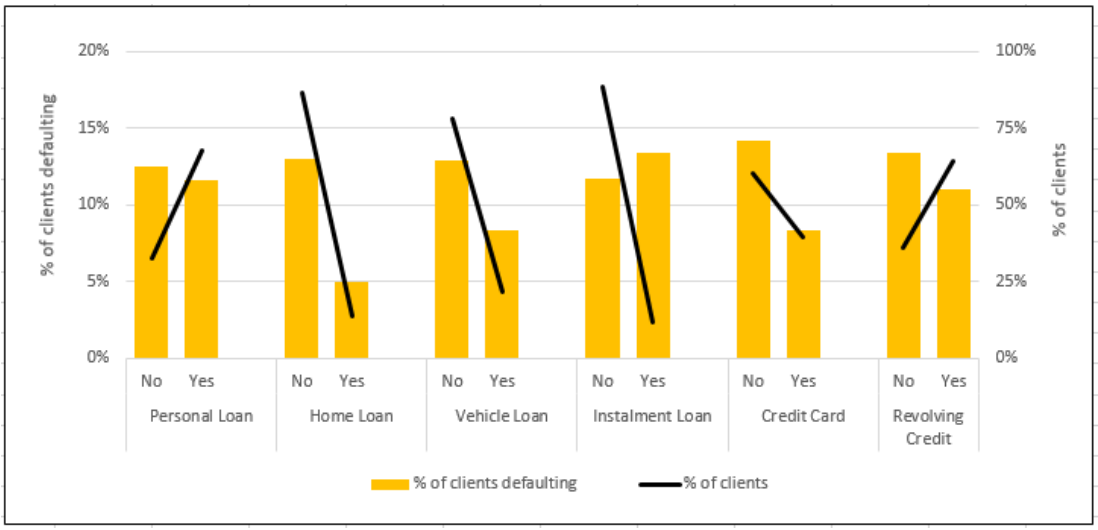

Figure 2. 6: Distribution and percentage of clients defaulting in the debt subgroup

From the personal loan variable in Figure 2.6, 67.6% of clients under study have a personal loan, whereas 32.4% of clients do not have a personal loan. The default rate of clients who have a personal loan is 11.6%, whereas clients who do not have a personal loan have a default rate of 12.5%. Therefore, there seems to be little variation in performance between clients who have and clients who do not have a personal loan. It is thus unlikely that there is a strong association between personal loan and the 'default' target variable.

The home loan variable presented in Figure 2.6 shows that 13.6% of clients have a home loan and only 5.0% of these clients default on their loan, whereas 86.4% of clients do not have a home loan and 13.0% of these clients default on their loan. This indicates that clients who have a home loan perform considerably better than clients without a home loan. Hence, a strong relationship between default and the home loan variable seems to exist.

Figure 2.6 displays the vehicle loan variable, which shows that 21.7% of clients have a vehicle loan and 78.3% of clients do not have a vehicle loan. The default rate of clients with and without a vehicle loan is 8.4% and 12.9%, respectively. Therefore, there seems to be a noticeable difference in the default rate between clients who have a vehicle loan and clients who do not have a vehicle loan (i.e., 4.1%). It can thus be concluded that the vehicle loan variable may have an influence on default.

From the instalment loan variable shown in Figure 2.6, 11.6% of clients under study have an instalment loan. The default rate of clients who have an instalment loan is 11.8%, whereas clients who do not have an instalment loan have a default rate of 13.3%. This suggests that there is little variation in the default rate between clients who have and clients who do not have an instalment loan. Thus, it is unlikely that there is a strong relationship between default and the instalment loan variable.

The credit card variable shown in Figure 2.6 indicates that 39.6% of clients have a credit card and these clients have a default rate of 8.4%, whereas 60.4% of clients do not have a credit card and these clients have a default rate of 14.2%. As there seems to be a material difference in performance between clients who have and clients who do not have a credit card, it is likely that a relationship exists between the credit card variable and the ‘default’ target variable.

Figure 2.6 displays the revolving credit variable, which shows that 64.1% of clients under study have a revolving credit facility. 11.1% of clients who have revolving credit, default, whereas 13.4% of clients without a revolving credit facility, default. Hence, there seems to be variation in performance between clients with and clients without a revolving credit facility. Thus, there is a possibility that revolving credit may have an influence on the ‘default’ target variable.

Figure 2.7 displays the distribution and percentage of clients in each category within each variable in the debt related subgroup, which includes the following variables: unpaids, internal consolidation, credit inactive and arrears.

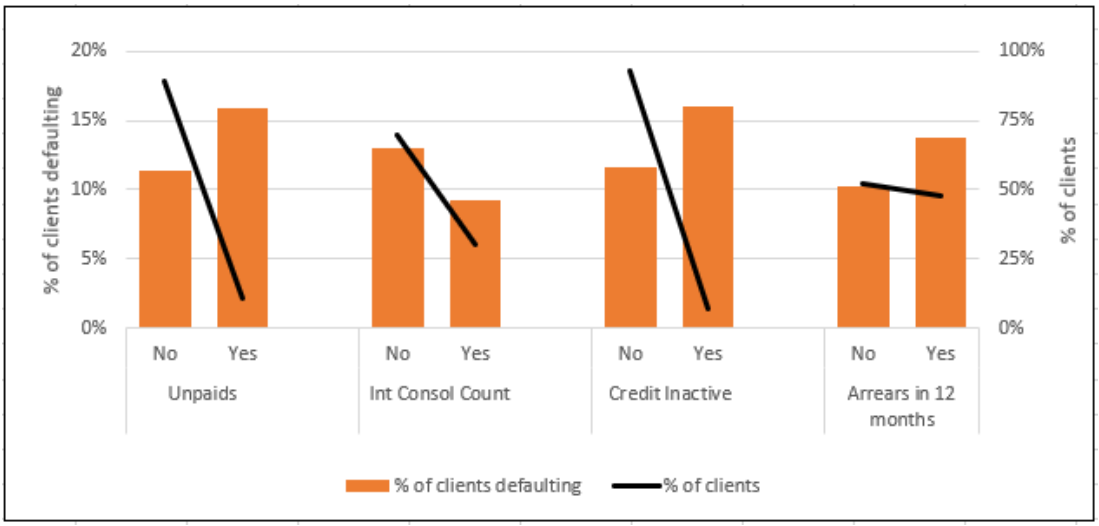

Figure 2. 7: Distribution and percentage of clients defaulting for variables in the debt related subgroup

The unpaids variable in Figure 2.7 shows that 10.5% of clients under study have unpaids, whereas 89.5% do not have unpaids. Figure 2.7 also shows that the default rates of clients who have unpaids and clients who do not have unpaids are 16.0% and 11.4%, respectively. This suggests a significant difference in the default rate between clients with and clients without unpaids. Hence, it can be concluded that a relationship between default and unpaids seems to exist.

In Figure 2.7, the int consol variable shows that 30.2% of clients have an internal consolidation. Clients who have an internal consolidation have a default rate of 9.2%, whereas clients with no internal consolidation have a default rate of 13.1%. Therefore, there appears to be variation in performance between clients who have and clients who do not have an internal consolidation. Thus, there is a possibility that the int consol variable has an influence on default.

Figure 2.7 also displays the credit inactive variable, which shows that a minority of clients under study (i.e., 6.7%) are credit inactive. The default rate of clients who are credit inactive is 16.1%, which is substantially worse than the 11.6% default rate of clients who are not credit inactive. This suggests that there is likely an association between default and credit inactive.

The arrears variable presented in Figure 2.7 indicates that approximately half of the clients under study have been in arrears in the last 12 months. Clients who were in arrears in the last 12 months have a default rate of 13.8%, whereas clients who have not been in arrears in the last 12 months perform better, as indicated by their default rate of 10.2%. Therefore, there may be a relationship between arrears and default.

In Figure 2.2 to Figure 2.7, the relationship between the ‘default’ target variable and the categorical variables was explored. The relationship between the independent categorical variables is then examined in order to identify any strong association between these variables, using Cramer’s rule. The Cramer’s V coefficient ranges from 0 to 1. The researcher considers a Cramer’s value of 0.51 or more as a strong association. Table 2.5 provides a list of the 10 pairs of variables under study with the highest Cramer coefficient values.

Table 2. 5: Pairs of variables with the 10 highest Cramer coefficient values

Variable 1	Variable 2	Cramer coefficient
Int/Ext Client	Salary bank	1.000
Staff Member	Product Taken	1.000
Client type	IntConsol	1.000
PayslipExpenses	Internal Living Expenses Rule	0.439
Union fees	PayslipExpenses	0.415
Medicalaid	Insurance	0.406
CreditCard	Limiting rule	0.393
Union fees	Insurance	0.390
Internal Living Expenses Rule	Medicalaid	0.389
PersonalLoan	Credit Inactive	0.385

From Table 2.5, three pairs of variables with a Cramer's coefficient value greater than 0.51 were reported, which indicates a strong association. The pairs of variables are Int/Ext Client and Salary bank, Staff Member and Product taken and Client type and IntConsol. All three pairs have a Cramer's coefficient value of 1. After further investigation, the researcher discovered that the strong association between these variables was due to one variable being a subcategory of the other variable. For example, Staff, which is one of the subcategories in the Product Taken variable, is also included as a separate variable in the dataset as Staff Member. Therefore, we can remove Int/Ext Client, Staff Member and IntConsol, as their information is contained in other variables.

The discrete variables under study are then examined. Visual representation methods such as histogram plots and line graphs are used to explore the discrete variables under study. The dataset includes three discrete variables, namely age, years with current employer and number of dependants, which are shown in Figures 2.8 to 2.10.

Figure 2.8 presents the volume of clients by age group and the percentage of clients defaulting in each age group.

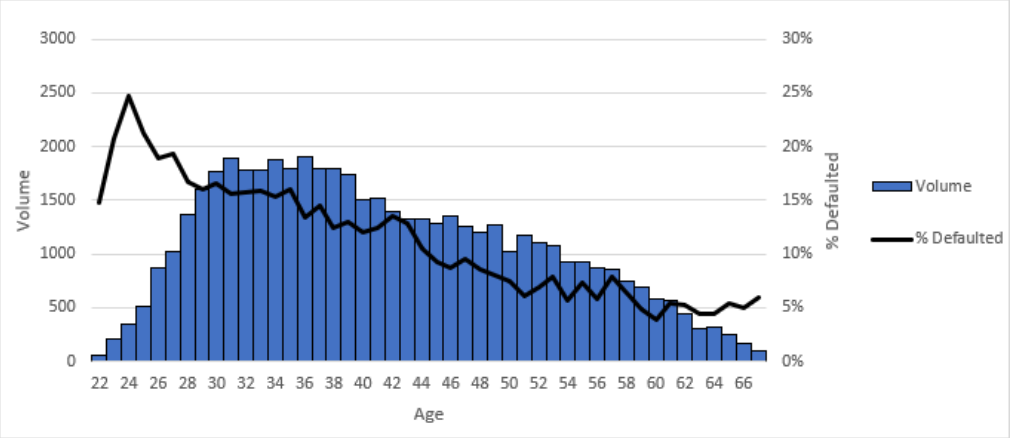

Figure 2. 8: Volume of clients by age group and percentage of clients who defaulted in each group

Figure 2.8 shows that the age of clients ranges from 22 to 67 years. Initially, as age increases, the volume of clients increases. Clients who are in their 30s apply for a loan at the financial institution most frequently. From the age of 40 years, the volume of clients then starts to decrease. Figure 2.8 also shows that as age increases, the percentage of clients defaulting tends to decrease and levels off after 60 years. Thus, there seems to be a negative relationship between age and the 'default' target variable.

Figure 2.9 displays the volume of clients by the number of years with their current employer and the percentage of clients defaulting in each group.

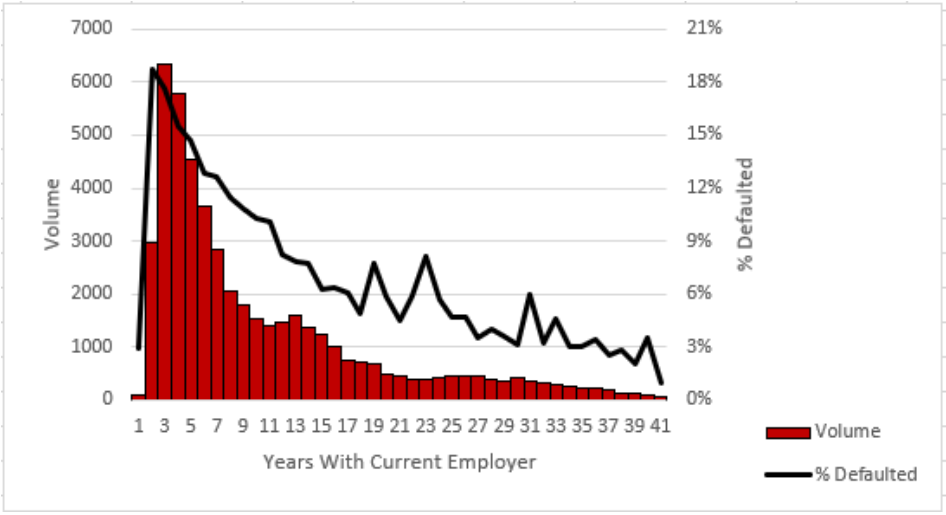

Figure 2. 9: Volume of clients by number of years with their current employer and percentage of clients who defaulted in each group

From Figure 2.9, we observe that the years with current employer variable ranges from 1 to 41 years. The most frequent number of years with the current employer is 3 to 5 years. Clients falling within this range seem to perform the worst, as indicated by the default rate, which ranges from approximately 15% to 18%. The overall trend shows that as the number of years with the current employer increases, the percentage of clients defaulting decreases. Therefore,

it is likely that the number of years with the current employer has a negative relationship with default.

Figure 2.10 shows the volume of clients and the percentage of clients defaulting by the number of dependants.

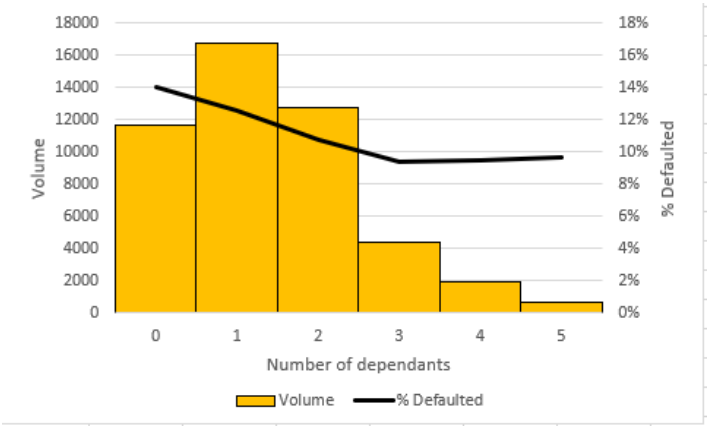

Figure 2. 10: Percentage of clients who defaulted by number of dependants and the volume of clients in each group

Figure 2.10 indicates that the number of dependants ranges from 0 to 5. A majority of clients taking a loan have 1 or 2 dependants or no dependants at all. The percentage of clients defaulting decreases from 0 dependants to 2 dependants and thereafter remains approximately 10%; clients with no dependants perform, on average, the worst. It can therefore be concluded that there is possibly an association between the number of dependants and the 'default' target variable.

The continuous variables are then analysed using the box and whisker plots displayed in Figures 2.11 and 2.12. The box and whisker plot is a method used to represent continuous data visually when doing explanatory data analysis. The plot shows the minimum value (excluding outliers), the first quartile (Q1), the median (Q2), the third quartile (Q3), the maximum value (excluding outliers), and the presence of any possible outliers in the dataset. The box and whisker plot is also used to visualise the distribution of the data as well as variability in the data. In this study, these plots will be used to compare clients who defaulted and clients who did not default.

Figure 2.11 displays box and whisker plots for loan information variables, namely instalment/disposable income, % instalment to income allowed, % instalment to income taken, and max offer. The dataset is divided according to the client's default status in order to compare the box and whisker plots of clients who default and clients who do not default.

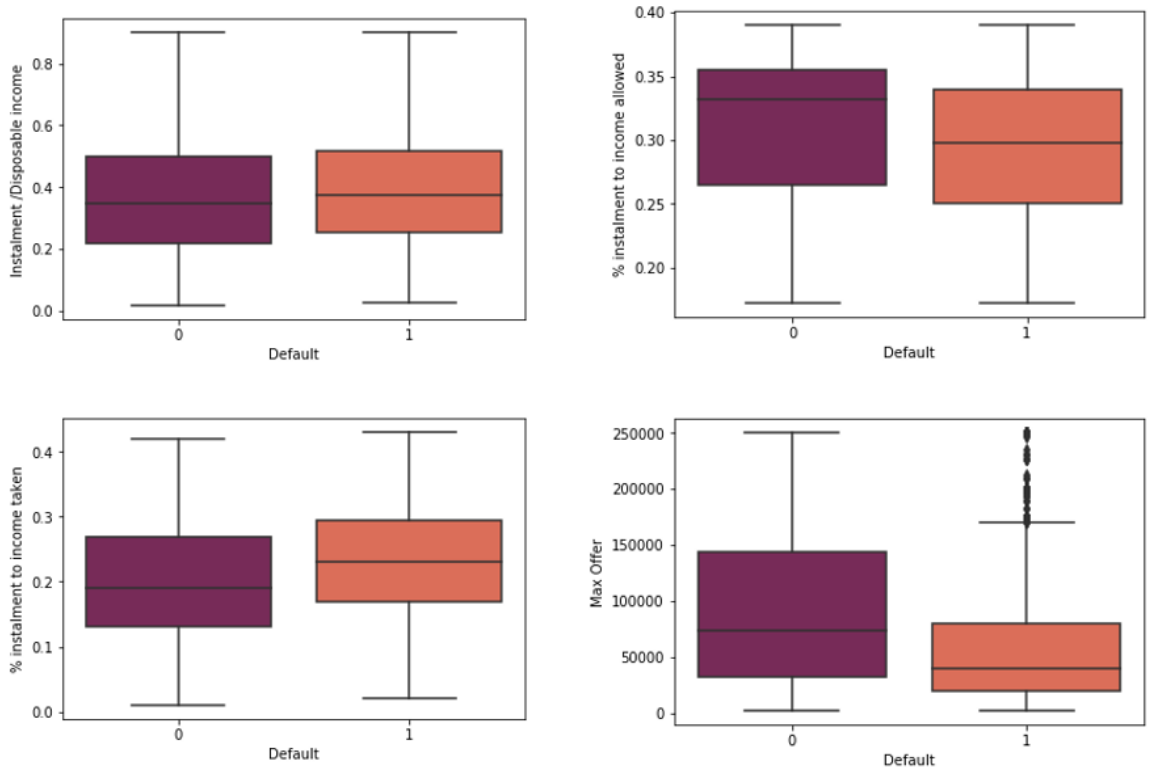

Figure 2. 11: Box and Whisker plots for variables in the loan information subgroup

From Figure 2.11, the instalment/disposable income variable ranges from 0.02 to 0.9 for clients who do not default and from 0.03 to 0.9 for those who default. The Q1, Q2 and Q3 values for clients who do not default are 0.22, 0.35 and 0.50, respectively, whereas the Q1, Q2 and Q3 values for clients who default are 0.25, 0.37 and 0.52, respectively. Thus, there seems to be no significant difference in instalment/disposable income between the clients who default and those who do not default. Therefore, from the box and whisker plots in Figure 2.11, there does not seem to be a strong relationship between instalment/disposable income and the 'default' target variable.

The % instalment to income allowed variable shown in Figure 2.11 ranges for both classes, default and not default, from 0.17 to 0.39. The box plot for clients who do not default shows a negatively skewed distribution, whereas the box plot for clients who default shows a more symmetric distribution. 50% of clients who do not default have a % instalment to income allowed value greater than 0.33, whereas only approximately 25% of clients who default have a % instalment to income allowed of more than 0.33,. This suggests that clients who do not default have, on average, larger % instalment to income allowed values compared to clients who default. Thus, % instalment to income allowed possibly has an influence on the 'default' target variable.

Figure 2.11 presents the box plot for the % instalment to income taken variable. % instalment to income taken ranges from 0.01 to 0.42 for clients who do not default, and from 0.02 to 0.43 for clients who default. Both default and non-default plots show a relatively symmetric distribution. The Q1, Q2 and Q3 values for clients who do not default are 0.13, 0.19 and 0.27, respectively, and the Q1, Q2 and Q3 values for clients who default are 0.17, 0.23 and 0.30, respectively. This indicates that the % instalment to income taken values seem slightly higher

for clients who default compared to clients who do not default. Hence, there is a possibility that % instalment to income taken has a relationship with the 'default' target variable.

From Figure 2.11, the max offer box plots for both default and not default indicate that the distribution is positively skewed, with noticeably greater variability in the maximum offer for clients who do not default. One can also observe that approximately 50% of clients who do not default, have a maximum offer value of more than R75 000, whereas only approximately 25% of clients who default have a maximum offer value of more than R75 000, which suggests that clients who do not default tend to receive higher offers. This provides evidence that maximum offer is likely associated with default.

The income and debt variables under study are explored next. Figure 2.12 presents box and whisker plots for income and debt variables, namely debt/net income, debt to income ratio, disposable income/basic, final disposable income/net income and calculated disposable income/net income.

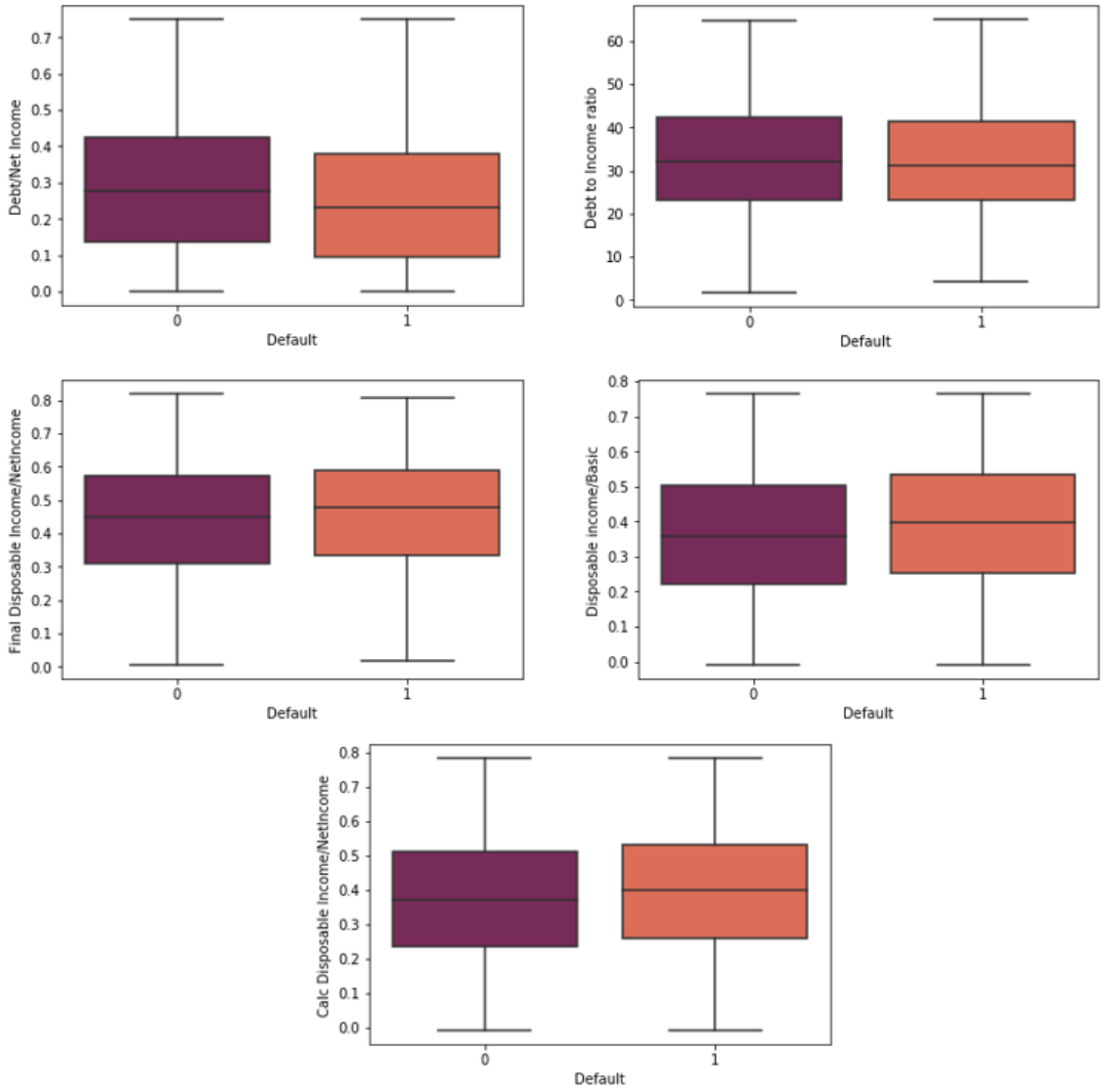

Figure 2. 12: Box and Whisker plots for variables in the Income and debt subgroup

The box plot for the debt/net income variable displayed in Figure 2.12 ranges from 0 to 0.75 for both non-defaulters and defaulters. Both plots for this variable show values which tend towards the lower end of the scale. The Q1, Q2 and Q3 values for clients who do not default are 0.14, 0.27 and 0.42, respectively, whereas these values for clients who default are 0.10, 0.23 and 0.38, respectively. This suggests that debt/net income values are higher for clients who do not default compared to clients who default. Thus, the debt/net income variable is possibly associated with default.

From the debt to income ratio variable in Figure 2.12, debt to income ratio ranges from 1.7 to 65.0 for the non-default class and from 4.2 to 65.0 for the default class. The Q1, Q2 and Q3 values for the non-default class are 23.2, 32.3 and 42.3, respectively, whereas the Q1, Q2 and Q3 values for the default class are 23.0, 31.1 and 41.2, respectively. The debt to income ratio variable also appears to be equally dispersed for the non-default and default classes. Therefore, from Figure 2.12 there does not seem to be significant variation in the debt to income ratio between clients who default and clients who do not default. A strong relationship between the debt to income ratio and default is thus unlikely.

The final disposable income/net income variable shown in Figure 2.12 seems to follow a normal distribution. This variable ranges from 0.01 to 0.82 for clients who do not default and from 0.02 to 0.81 for those who default. Q1, Q2 and Q3 values for the non-default class are 0.31, 0.45 and 0.57, respectively, and these values for the default class are 0.33, 0.48 and 0.59, respectively. Thus, there seems to be little difference between the default and non-default classes regarding the final disposable income/net income variable. Association between the final disposable income/net income variable and the ‘default’ target variable therefore seems improbable.

The box and whisker plot for the disposable income/basic variable displayed in Figure 2.12 show that the Q1, Q2 and Q3 values for the non-default class are 0.22, 0.36 and 0.50, respectively, whereas Q1, Q2 and Q3 values for the default class are 0.25, 0.39 and 0.53, respectively. Therefore, from the box and whisker plot in Figure 2.12, there seems to be little variation in disposable income/basic between clients who default and clients who do not default. Thus, it seems unlikely that disposable income/basic has an influence on default.

From Figure 2.12, the box plots for the calculated disposable income/net income range from approximately 0 to approximately 0.8 for both non-defaulters and defaulters. Both plots also show a normal distribution for this variable. The Q1, Q2 and Q3 values for the non-default class are 0.23, 0.37 and 0.51, respectively, whereas these values for the default class are 0.25, 0.40 and 0.53, respectively. Thus the researcher concludes that it seems unlikely that calculated disposable income/net income has a strong relationship with the target variable, ‘default’.

After gaining further insight and understanding of the numerical variables under study and their relationship with the ‘default’ target variable in Figures 2.11 and 2.12, the researcher analyses the distribution, presence of outliers and correlation between the numerical variables. Histogram plots of the numerical variables, namely instalment/disposable income, % instalment to income allowed, % instalment to income taken, max offer, debt/net income, debt to income ratio, disposable income/basic, final disposable income/net income and calc disposable income/net income, are explored in Figure 2.13 to determine whether they follow a normal distribution. These plots are often used to show the frequency distribution of a variable; the histogram plot of a variable that follows a normal distribution will have a bell-shaped curve.

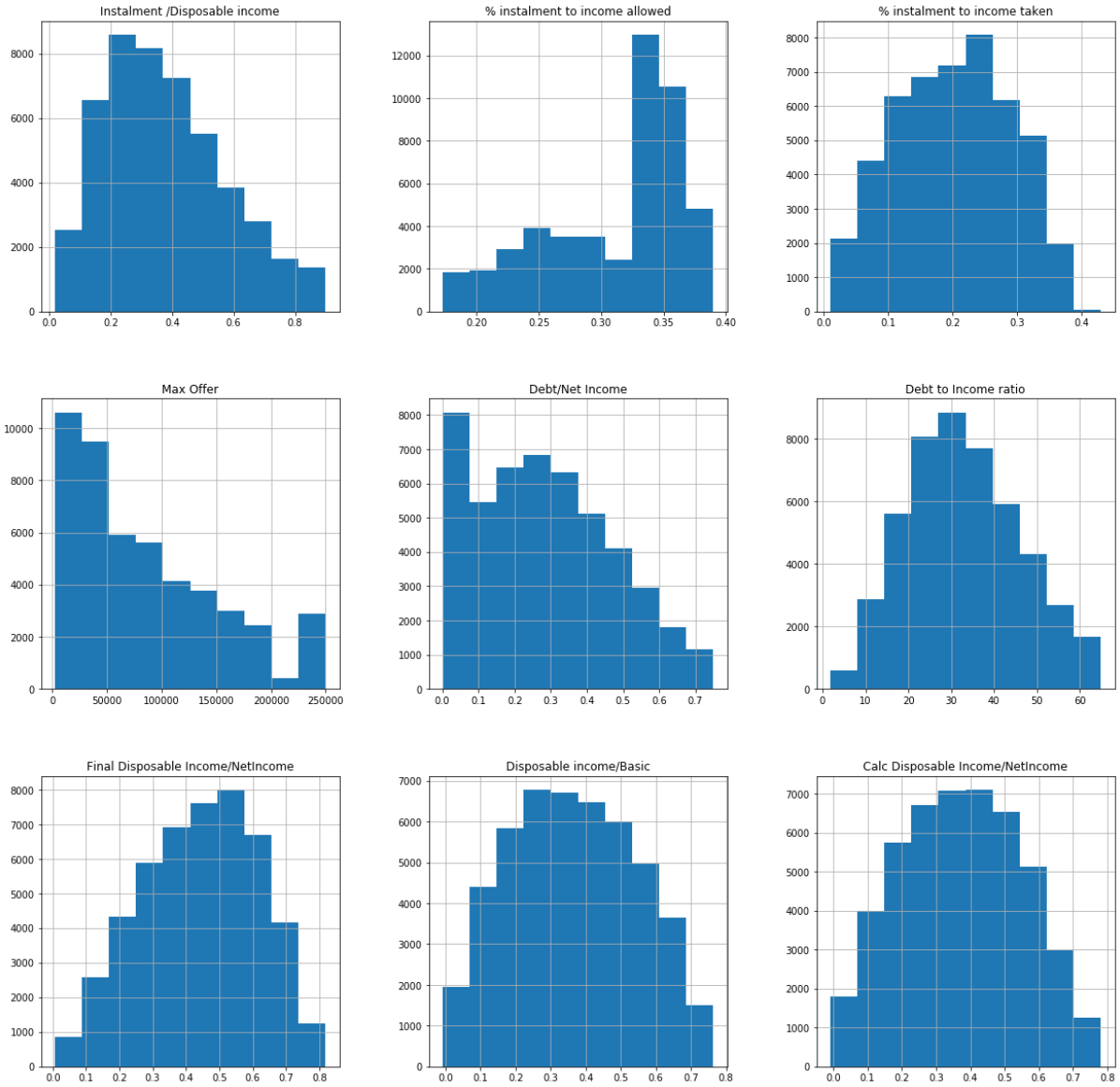

Figure 2. 13: Histogram plots of numerical variables

The histogram plots in Figure 2.13 show that the following variables seem to deviate from normal distribution: % instalment to income allowed, max offer and debt/net income. The Box-Cox transformation technique was used to normalise the data; it suggested the natural log transformation for the variables % instalment to income allowed and debt/net income, and the square root transformation for the variable max offer as the lambda values for % instalment to income allowed, max offer and debt/net income were 0.105 (approximately 0), 0.307 (approximately 0.5) and 0.218 (approximately 0), respectively.

To identify the presence of outliers, the interquartile range technique is used; the results are reported in Table 2.6. An outlier is identified as a point that falls below the lower limit Q1 - 1.5 * IQR, or above the upper limit Q3 + 1.5 * IQR, where Q1 is the lower quartile, Q3 is the upper quartile, and IQR is the interquartile range, that is, Q3-Q1.

Table 2. 6: Outlier detection using the interquartile range technique for numerical variables

	Quartile 1	Quartile 3	IQR	Lower limit	Upper limit	Min	Max	Outliers
Instalment/Disposable income	0.22	0.50	0.28	-0.19	0.92	0.02	0.90	0
% instalment to income allowed	0.27	0.35	0.09	0.13	0.48	0.17	0.39	0
% instalment to income taken	0.13	0.27	0.14	-0.08	0.48	0.01	0.43	0
Max offer	29970	130832	100862	-121978	282780	2005	250000	0
Debt/Net income	0.13	0.42	0.29	-0.30	0.84	0.00	0.75	0
Debt to income ratio	23.13	42.20	19.07	-5.12	70.45	1.73	64.97	0
Final disposable income/Net income	0.31	0.58	0.27	-0.08	0.97	0.01	0.82	0
Disposable income/Basic	0.22	0.51	0.28	-0.20	0.93	-0.01	0.76	0
Calc disposable income/Net income	0.24	0.51	0.27	-0.17	0.92	-0.01	0.78	0

From Table 2.6, there were no significant outliers present in the dataset.

Multicollinearity between numerical variables is then explored by using the correlation matrix. Correlated variables in a model may affect the model's performance negatively. The correlation matrix shows the Pearson correlation coefficient for each combination of the different features. The matrix is used to identify which features are correlated, the degree of correlation, and the direction. A correlation heatmap is a visual representation of the correlation matrix. The correlation values in the heatmap range from -1 to 1, where values close to -1 and 1 represent high correlation and 0 represents no correlation. Different colours are used to show the degree of correlation and the direction. The cells with correlation values closer to -1 are shaded in darker colours, whereas cells with correlation values closer to +1 are shaded in light colours. Cells shaded in orange and pink are associated with low correlation values. Pairs of variables that have an absolute correlation of 0.61 or more are considered highly correlated in this study. Figure 2.14 presents a heatmap for all numerical variables.

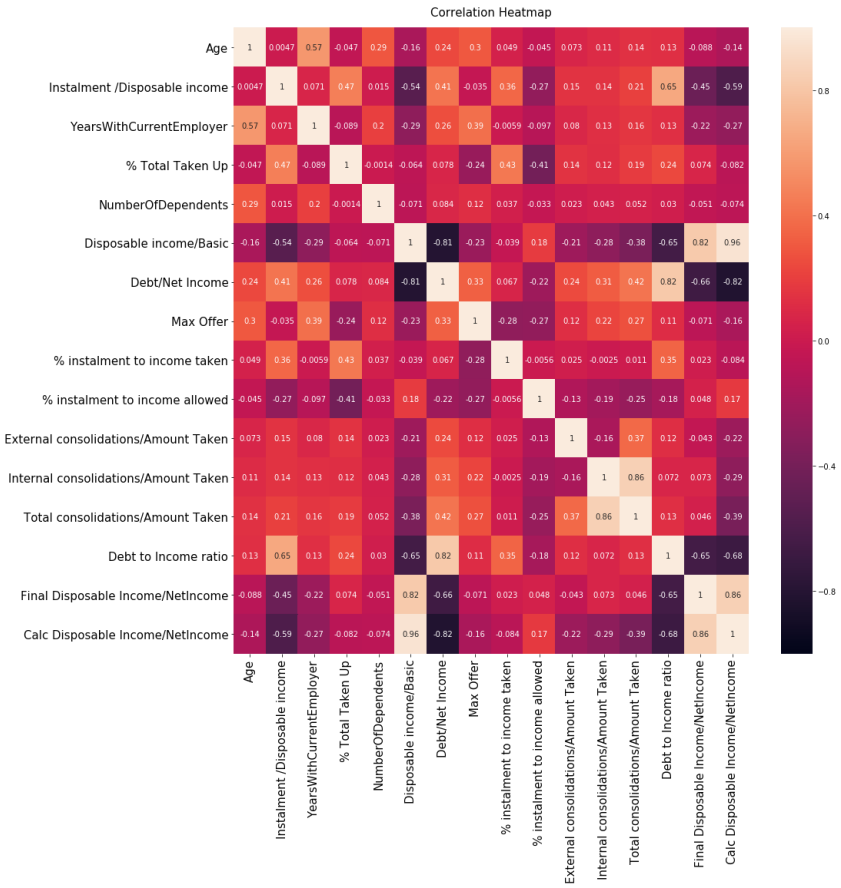

Figure 2. 14: Heatmap showing numerical variables

The heatmap in Figure 2.14 shows that 12 pairs of variables are highly correlated. The following variables are highly correlated with at least one other variable: disposable income/basic, final disposable income/net income, calculated disposable income/net income, instalment/disposable income, debt/net income, internal consolidations/amount taken, total consolidations/amount taken and debt to income ratio. Multicollinearity among variables may affect the model negatively. Methods such as principal component analysis, discussed in Chapter 2.3, or simply removing one of the correlated variables can remove multicollinearity in the dataset.

2.3 Principal Component Analysis (PCA)

PCA is a dimensionality reduction method. It is used to transform a larger set of variables into a smaller set of variables whilst retaining most information. By using PCA, one can include the minimum number of principal components needed to explain a certain percentage of the variance. The new principal components formed are uncorrelated. PCA will therefore solve the problem observed in section 2.2 in which some of the numerical variables are highly correlated. An alternative method to solve the problem of highly correlated variables is to drop one of the correlated features; however, this may result in loss of information.

When using PCA, the number of principal components needs to be selected. The cumulative percentage of variance method was used to select the number of principal components. Figure 2.15 displays the cumulative percentage of variance explained by number of principal components.

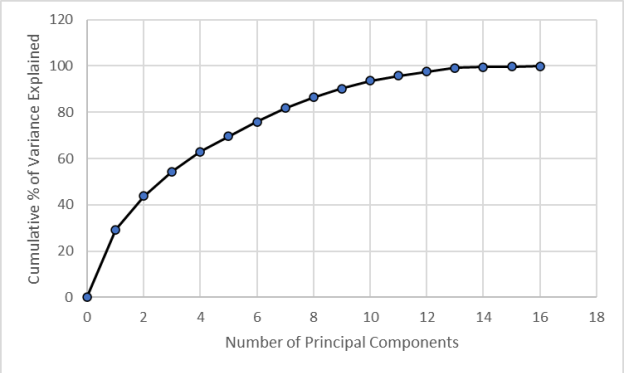

Figure 2. 15: Cumulative percentage of variance explained by the number of principal components

From Figure 2.15, in order to explain 80% of variance, which was chosen by the researcher, 7 principal components are required. This will reduce the number of numerical variables from 16 to 7 principal components, while still explaining 81.8% of variance, and the principal components will be uncorrelated. Figure 2.16 represents a heatmap for the 7 principal components.

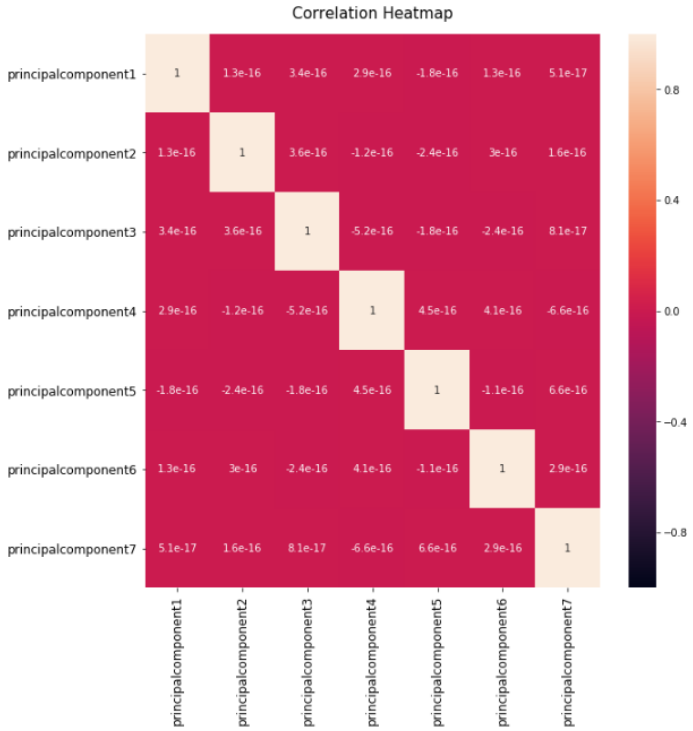

Figure 2. 16: Heatmap for principal components

From Figure 2.16, correlation coefficients for all pairs of principal components are close to zero, which indicates that all 7 principal components are uncorrelated.

2.4 Summary

In this study, the researcher's aim was to predict whether clients will default or not on their loan. A client is classed as a default if he/she missed at least three payments in the first 12 months of the loan being disbursed. 12% of clients under study defaulted, whereas 88% of clients did not default, which indicates that the dataset is imbalanced. This often leads to models that produce unsatisfactory results, as the models are more likely to misclassify the minority class (i.e., default), which is of more interest to the researcher than the majority class (i.e., non-default). The imbalance in the dataset must thus be catered for by using techniques such as balanced weighting and SMOTE. A combination of 32 categorical and 16 numerical variables are included in the dataset; these variables are potentially associated with the 'default' target variable. They were grouped together in the following subcategories: demographics, client information, loan information, income, expenses, and debt for analysis purposes.

Each variable in the dataset was explored and the analysis and key findings are summarised below:

- The categorical variables were analysed using bar graphs and line graphs in order to gain insight into the distribution of clients for each variable and to understand the relationship between the 'default' target variable and the independent variables.
- Variables such as Ext Subsequent Lending and staff member comprise of two categories, where one of the categories have the majority of clients and the other category, a negligible portion of clients (i.e., less than 5% of total clients). It is unlikely that such variables will be of importance to the model, unless the variable is exceptionally good at differentiating between clients who default and those who do not. The difference in default rates between the two categories within the Ext Subsequent Lending and staff member variables were 12% and 8.9%, respectively, which is considerably large and therefore, these variables may still influence the model.
- Variables such as loan purpose, limiting rule and product type each comprise of more than two categories, with at least one category including a minor portion of clients. Combining the minority category with another category within the variable may be considered. A general practice is to combine the minority category with a category that has the closest default rate, alternatively, where numerous categories within a variable include a negligible portion of clients, grouping all of these categories into an 'other' category may be considered. Loan purpose comprised of 11 categories; 6 of these categories each included less than 1% of the population. Since the portion of clients falling in each of these categories was negligible and there was no significant difference in default rates between these categories, the researcher grouped these 6 categories with the category 'other'. The researcher chose not to group categories within the limiting rule variable and product type variable as (for both variables) the default rate of the clients in the minority category was noticeably different compared to the other categories within the variable.
- The following categorical variables showed little difference in default rates between the categories within each variable: permanent allowances, personal loan, loan purpose and instalment loan. Therefore, these variables do not seem to be strongly associated with

the ‘default’ target variable. All other categorical variables seem to have a relationship with default.

- Cramer’s rule was then used in order to identify pairs of categorical variables that were strongly associated. A Cramer’s value of 0.51 or more was considered as a strong association. Three pairs of variables, namely Int/Ext Client and Salary bank, Staff Member and Product taken and Client type and IntConsol, were identified as being strongly associated, with all pairs having a Cramer’s coefficient value of 1. After further investigation, it was observed that the strong association between these variables was due to one variable being a subcategory of the other variable. Therefore, Int/Ext Client, Staff Member and IntConsol were removed from the dataset, as their information is contained in other variables.
- The relationship between default and the following discrete variables was also explored: age, years with current employer and number of dependants. All three discrete variables seem to have a negative relationship with default; as age, years with current employer and number of dependants increased, the percentage of clients defaulting decreased and eventually levelled off.
- To analyse the continuous variables, box and whisker plots were used. The continuous variables were grouped into subgroups, namely loan information, and income and debt. In order to explore each variable’s association with default, the dataset was grouped according to the client’s default status.
- The box plots for instalment/disposable income, debt to income ratio, disposable income/basic, final disposable income/net income and calculated disposable income/net income showed that there was no significant difference between clients who defaulted and clients who did not default. Therefore, it is unlikely that these variables have a strong relationship with the target variable, default. Box plots for max offer, % instalment to income allowed, % instalment to income taken and debt/net income did show a difference between clients who defaulted and clients who did not. Thus, these variables are possibly associated with the ‘default’ target variable.
- The distributions, presence of outliers and the correlation between the numerical variables were analysed to gain further insight into the models’ inputs.
- Using histogram plots, the researcher observed that % instalment to income allowed, max offer and debt/net income seemed to deviate from normal distribution. The Box-Cox transformation was used to normalise the data. This technique suggested the natural log transformation for the variables % instalment to income allowed and debt/net income, and the square root transformation for the variable max offer.
- To identify outliers, the interquartile range method was used; there were no significant outliers identified in the dataset.
- Correlation heatmaps were then used to identify multicollinearity between numerical variables. Pairs of variables with an absolute correlation value of more than 0.61 were considered highly correlated. Disposable income/basic, final disposable income/net income, calculated disposable income/net income, instalment/disposable income, debt/net income, internal consolidations /amount taken, total consolidations/amount taken and the debt to income ratio were identified as being highly correlated with at

least one other variable. The principal component analysis (PCA) method in section 2.3 can be used to solve the problem of highly correlated variables by creating new variables (principal components) that are uncorrelated. An alternative method is to drop one of the correlated variables; however, this may result in a loss of information.

The researcher notes that before the model building step, the data needs to be pre-processed. The number of variables in the processed dataset increased to 57, prior to using dimensionality reduction techniques (the original dataset included 48 variables). The increase in the number of variables was a result of categorical variables being encoded using the dummy variable method. Dummy variables are used to represent categorical variables and can only take values 0 and 1 where 0 represents the absence of a condition and 1 represents the presence of it. Categorical variables, which have more than two categories, are represented by a set of dummy variables. Encoding categorical variables is critical as most machine learning algorithms require numeric input and output variables.

From the findings in this chapter, the researcher is also aware of the large number of variables in the dataset and is mindful that not all variables will be important in the model. Principal component analysis, which is also used to correct the data for multicollinearity, and feature selection, which aims to remove irrelevant and redundant features, are dimensionality reduction techniques which can be used to reduce the number of variables used in the model.

In Chapter 3, machine learning classification algorithms are discussed, in Chapter 4, the classification algorithms are fitted to the default dataset using PCA and in Chapter 5, the classification algorithms are fitted to the dataset using feature selection; the feature selection technique used is recursive feature elimination.

Chapter 3

3 Classification algorithms and evaluation metrics

In this chapter, the researcher provides a brief introduction to machine learning, explains the theory of the classification algorithms used in this study and presents the performance metrics which the models will be evaluated on.

3.1 Brief introduction to machine learning

Machine learning has become increasingly popular amongst researchers and institutes over a short period of time; it is a subset of artificial intelligence that allows systems to learn patterns in the data automatically and improve outcomes through experience without being explicitly programmed. There are two main categories of machine learning, namely supervised learning and unsupervised learning. In supervised learning, the model learns patterns from a labelled dataset and the trained model is used to make predictions on unseen data. Unsupervised learning does not require a labelled dataset. The aim of unsupervised learning is to find structure, hidden relationships, and patterns from the input data. This study focuses on supervised learning. There are two types of supervised learning, namely classification and regression. A classification problem has a categorical target variable, for example a client's default status, whereas a regression problem has a real value target variable. The process, for both classification and regression problems, include the following steps; collection of data, data cleaning and feature engineering, selection of machine learning algorithm and model building, model evaluation, model improvement and lastly, model deployment. The objective of this study is to build classification models, using labelled data, to predict a client's default status. The researcher explores several classification machine learning algorithms, namely logistic regression, decision trees, random forest, k-nearest neighbours, the Naïve Bayes algorithm, support vector machines and artificial neural networks. Evaluation metrics, such as accuracy, balanced accuracy, recall, specificity, precision, the negative predictive value, Gini, and the AUC score, are used to evaluate and compare the performance of the machine learning models.

3.2 Logistic regression

Generalized linear models (GLM) is an extension of linear models which allows for non-normal response distributions (Venables & Ripley, 1999). Linear regression, logistic regression and Poisson regression are all examples of GLMs. GLMs consists of 3 components namely, the response component, the systematic component and the link function. The distribution of the response variable belongs to the exponential family (e.g., Poisson, binomial, gamma distribution), the systematic component is the linear predictor, and the link function is the

connection between the mean of the response and the linear predictor (Naufal, Devila, & Lestari, 2019). The link function is often based on the response distribution. It maps the range of the mean response on to the whole real line (McCullagh & Nelder, 1989). Examples of link functions are the identity link function (normal distribution), log link function (Poisson distribution) and the logit link function (binomial distribution). To estimate the parameters for the GLM, the maximum likelihood estimation method is used (Naufal et al., 2019). In this study, the GLM which the researcher focuses on, is logistic regression.

Logistic regression is used to analyse the relationship between a categorical dependent variable, for example a client's default status, and a set of independent variables that affects the dependent variable (Park, 2013). In most cases, the target variable (i.e., response variable) has only two possible outcomes, namely the event occurs (Y=1, e.g., the client defaults) or the event does not occur (Y=0, e.g., the client does not default). However, the model can be modified to cater for a dependent variable with more than two categories, which is referred to as multinomial logistic regression (Hintze, 2007). This study will focus on binary target variables.

The logistic regression model assumes that the target variable is categorical. However, the target variable is not modelled directly; instead, logistic regression analysis is based on probabilities that are associated with the target variable's values. Logistic regression estimates the probability that an event will occur, and these estimated probabilities are used to assign an observation to a class, based on the threshold selected (Dayton, 1992).

Consider a binary output Y_i that is equal to one when the event occurs (e.g., the clients default on their loan) and zero when the event does not occur (e.g., the clients do not default on their loan); for each observation i = 1, 2, …., n, with k explanatory variables $x_{i1}, x_{i2}, x_{i3}, \ldots, x_{ik}$. Let p be the probability of Y_i =1 such that p = $P(Y_i = 1)$ for a given observation. In logistic regression, the log-odds (or logit) are modelled as a linear combination of the intercept and explanatory variables, as shown below:

$$logit(y_i) = \ln\left(\frac{p}{1-p}\right) = \alpha + \beta_1 x_{i1} + \ldots + \beta_k x_{ik}, \qquad (3.2.1)$$

where $\frac{p}{1-p}$ is referred to as the odds of an event and denotes the likelihood of the event occurring, $\ln\left(\frac{p}{1-p}\right)$ is known as the log odds or logit, α is the Y intercept and β_j's denote coefficients of the explanatory variables $x_{i1}, x_{i2}, x_{i3}, \ldots, x_{ik}$ (Park 2013; Peng et al., 2002).

The estimated probability of an event occurring can be derived by taking the antilog and rearranging the equation (3.2.1).

This is given by

$$p = P(Y_i = 1)$$

$$= \frac{e^{\alpha+\beta_1 x_1 + \ldots + \beta_k x_k}}{1 + e^{\alpha+\beta_1 x_1 + \ldots + \beta_k x_k}} \qquad (3.2.2)$$

$$= \frac{1}{1 + e^{-(\alpha+\beta_1 x_1 + \ldots + \beta_k x_k)}}.$$

In logistic regression, the aim is to obtain the strongest linear combination of independent variables such that we maximise the probability (likelihood) of predicting the correct class for each training observation (Stoltzfus, 2011). The maximum likelihood function is often used to estimate the function's coefficients.

For a sample size of n where each observation has a vector of features X and a target variable Y_i, $Y_i = 1$ if the event occurred and $Y_i = 0$ if the event did not occur, and the probability is p when $Y_i = 1$ and 1- p when $Y_i = 0$.

The likelihood function is given by

$$\begin{aligned} L &= \prod_{i=1}^{n} p(y|x)^{Y_i}\big(1 - p(y|x)\big)^{1-Y_i} \\ &= p(y|x)^{\sum_{i=1}^{n} Y_i}\,(1 - p(y|x))^{n-\sum_{i=1}^{n} Y_i}. \end{aligned} \tag{3.2.3}$$

The logarithm of the likelihood equation is often utilised, as it is mathematically easier to work with. It is known as the log likelihood and is given by the following equation:

$$l = \log(L) = \sum_{i=1}^{n} Y_i \log[p(y|x)] + (n - \sum_{i=1}^{n} Y_i) \log[1 - p(y|x)], \tag{3.2.4}$$

where $p(y|x) = \frac{e^{\alpha+\beta_1 x_1+\cdots+\beta_k x_k}}{1+ e^{\alpha+\beta_1 x_1+\cdots+\beta_k x_k}}$.

The maximum likelihood estimates are obtained by computing the first derivative of the log likelihood and solving for α and β (Park, 2013; Shalizi, 2019).

A predefined threshold value is selected in order to classify each observation into a class. The threshold is defaulted to 50%; however, this value can be adjusted. When using the default threshold of 50%, if $p > 0.5$, the model predicts that the event will occur (the client will default) and if $p < 0.5$, the model will predict that the event will not occur (the client will not default) (Shalizi, 2019).

Interpretation

Consider a binary dependent variable y with values 0 and 1 (0 = not default and 1 = default) and one explanatory variable x where the logistic regression equation is given by

$$\ln \frac{p}{1-p} = \alpha + \beta_1 x. \tag{3.2.5}$$

If x increases by one unit, the logistic regression equation is

$$\ln\frac{p'}{1-p'} = \alpha + \beta_1(x+1) = \alpha + \beta_1 x + \beta_1. \tag{3.2.6}$$

β_1 can be obtained by taking the difference between equations (3.2.5) and (3.2.6), as shown below:

$$\begin{aligned}\beta_1 &= (\alpha + \beta_1 x + \beta_1) - (\alpha + \beta_1 x)\\ &= (\ln\tfrac{p'}{1-p'}) - (\ln\tfrac{p}{1-p})\\ &= \ln((\tfrac{p'}{1-p'})/(\tfrac{p}{1-p}))\\ &= \ln(\tfrac{odds'}{odds}).\end{aligned} \tag{3.2.7}$$

Thus, the logistic regression coefficient β_i associated with the explanatory variable x_i can be interpreted as the change in log-odds of the event (e.g., a client defaulting on his/her loan) per unit change in x. By exponentiating both sides of equation (3.2.7), the following equation is obtained:

$$e^{\beta_1} = \frac{odds'}{odds}. \tag{3.2.8}$$

When multiple independent variables are present, the regression coefficients are interpreted in a similar manner while holding all other independent variables constant (Hintze, 2007).

Overall model evaluation:

Deviance Test: To assess the goodness-of-fit of a model, the deviance statistic can be used. The deviance statistic compares the log-likelihood of the fitted model to the log-likelihood of the saturated model. A saturated model has the same number of estimated parameters as the number of observations.

The deviance statistic is given by

$$D = (-2logL\ of\ fitted\ model) - (-2\ logL\ of\ saturated\ model). \tag{3.2.9}$$

Equation 3.2.9 can be written as

$$D = -2\log(\frac{likelihood\ of\ fitted\ model}{likelihood\ of\ saturated\ model}). \tag{3.2.10}$$

The test statistic D asymptotically follows a X^2 distribution with n-p degrees of freedom i.e., the degrees of freedom is equal to the number of parameters in the saturated model (n) minus the number of parameters in the fitted model (p) (Badi, 2017).
The rule of thumb for the Deviance test is that the D statistic should be less than the degrees of freedom i.e., n-p. A model is acceptable if the D statistic is within 1.5 times the degrees of freedom (Chifurira, 2018).

Residual Analysis

The analysis of residuals to assess the adequacy of a fitted model is an important step in the model building process. Residual analysis can be used to identify unusual observations i.e., outliers, influential observations and observations which have high leverage. Observations with values that deviate from the expected range, resulting in very large residuals, may be considered outliers; outliers in a dataset can lead to incorrect inferences. An observation is considered influential if the estimate of coefficients changes significantly once the observation is removed. An observation with an extreme value on a covariate is said to have high leverage and can have an uncommonly large impact on the coefficient estimates (Sarkar, Midi, & Rana, 2011). The Pearson residual, deviance residual and the hat diagonal are commonly used for logistic regression diagnostics.

Given a binary response variable Y, the logistic regression model can be written as:

$$Y_i = \pi_i(x) + \varepsilon_i \quad where\ i = 1,2, \dots, n. \tag{3.2.11}$$

Since the response variable Y can only take on the value 0 and 1, the ordinary residual is given by:

$$\hat{\varepsilon}_i = \begin{cases} 1 - \hat{\pi}_i & if \quad Y_i = 1 \\ -\hat{\pi}_i & if \quad Y_i = 0 \end{cases}. \tag{3.2.12}$$

Therefore, the error variance can be defined as:

$$V(Y|X) = (\hat{\pi}_i)(1 - \hat{\pi}_i). \tag{3.2.13}$$

To obtain the Pearson residual, the ordinary residual is divided by the estimated standard error of Y_i and is given by:

$$\begin{aligned} r_{P_i} &= \frac{\hat{\varepsilon}_i}{\sqrt{(\hat{\pi}_i)(1 - \hat{\pi}_i)}} \\ &= \frac{Y_i - \hat{\pi}_i(x)}{\sqrt{(\hat{\pi}_i)(1 - \hat{\pi}_i)}}. \end{aligned} \tag{3.2.14}$$

Pearson residuals can be defined as the standardized distance between the observed and predicted responses.

Since the Pearson residuals do not have unit variance, they are standardized further by their estimated standard deviation which results in the studentized Pearson Residuals. This is defined as:

$$r_{SP_i} = \frac{Y_i - \hat{\pi}_i}{\sqrt{(\hat{\pi}_i)(1 - \hat{\pi}_i)(1 - h_{ii})}} = \frac{r_{P_i}}{\sqrt{(1 - h_{ii})}}. \quad (3.2.15)$$

Where h_{ii} (often called hat diagonal in logistic regression) is the i-th element on the diagonal of the nxn estimated hat matrix H, which is given by:

$$H = \widehat{W}^{\frac{1}{2}} X \left(\acute{X}\widehat{W}X\right)^{-1} \acute{X}\widehat{W}^{\frac{1}{2}}, \quad (3.2.16)$$

where $\widehat{W}$ is the nxn diagonal matrix with elements $(\hat{\pi}_i)(1 - \hat{\pi}_i)$ (Sarkar, Midi, & Rana, 2011). The hat diagonal measures the leverage of an observation.

Deviance residuals are also often analysed when using logistic regression. Deviance residuals are defined as the difference between the log-likelihood of the saturated model and the log-likelihood of the fitted model. They represent the contribution of individual observations to the deviance (Feng, Longhai, & Sadeghpour, 2020).

The deviance residuals are calculated based on the deviance statistic and is given by

$$D = 2\sum_{i=1}^{n} \left\{ Y_i \log\left(\frac{Y_i}{\hat{\pi}_i}\right) + (1 - Y_i) \log\left(\frac{1 - Y_i}{1 - \hat{\pi}_i}\right) \right\}^2, \quad (3.2.17)$$

where Y_i is the response variable which has two possible outcomes for logistic regression i.e., 1 and 0 with probabilities $\hat{\pi}_i$ and 1-$\hat{\pi}_i$, respectively (Ahmad, 2011).

The deviance residual for the i-th individual component is defined as

$$r_{D_i} = sign(Y_i - \hat{\pi}_i)\sqrt{-2[Y_i \log(\hat{\pi}_i) + (1 - Y_i) \log(1 - \hat{\pi}_i)]}. \quad (3.2.18)$$

If $y_i = 0$ then $r_{D_i} = -\sqrt{-2[\log(1 - \hat{\pi}_i)]}$ and if $y_i = 1$ then $r_{D_i} = \sqrt{-2 \log(\hat{\pi}_i)}$.

When assessing the model fit, the residuals are often plotted against the predicted probabilities. Standardized residuals greater than |2| are potential outliers and should be further investigated.

If the LOWESS smooth (Locally Weighted Scatterplot Smoothing or LOWESS is a technique used in regression analysis to produce a smooth curve through a scatterplot) of the plot of the residuals against the predicted probability approximates a line which has a zero slope and intercept, one can conclude that there does not seem to be significant model inadequacy and no influential outliers seem to be present (Sarkar, Midi, & Rana, 2011).

3.3 Decision tree (ID3, C4.5)

Decision trees are one of the most popular classification methods used in machine learning. They are non-parametric, as there are no distributional assumptions made and a predefined relationship is not required between the dependent and independent variables (Kuhn et al., 2013). A decision tree is similar to a flowchart that takes on the form of a tree structure. Decision trees consist of three types of nodes, namely the root node, the internal nodes (decision nodes), and the leaf nodes.

The root node is the node found at the top of the tree; it has outgoing edges but no incoming edges. The feature that classifies the data the best is used as the test feature for the root node. Internal nodes are the nodes found between the root node and the leaf node. They have an incoming edge and outgoing edges. At each root node and internal node, a certain feature is tested and each of the node's branches represents an outcome of the test, such that the space is split into two or more sub-spaces. The leaf nodes are the final nodes of each branch, and these nodes indicate which class an observation belongs to. Leaf nodes have an incoming edge but no outgoing edges (Patel & Upadhyay, 2012; Singh & Giri, 2014).

Figure 3.1 displays an example of a simple decision tree structure.

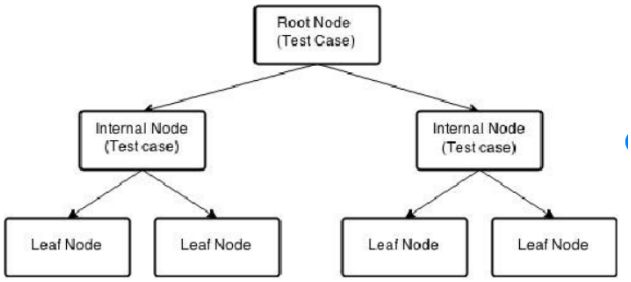

Figure 3. 1: **Structure of a simple decision tree**

Most decision tree algorithms use the standard top-down approach. At each node, starting from the root node, a feature is selected by using a certain splitting criterion; the chosen feature is used to split the data into subsets. Some of the common splitting criteria are entropy, information gain, gain ratio and the Gini index. Most splitting criteria are defined in terms of how much the impurity reduces by after the split, that is, from parent to child node. The splitting/partitioning process is repeated until the nodes cannot be split further, or a stopping criterion is met, and all leaf nodes are present (Singh & Giri, 2014). To avoid overfitting, post-pruning may be used once the tree has been grown.

Once the decision tree is constructed, new observations can be classified. For each new observation, one starts at the top of the tree, which is the root node, and makes one's way down the tree by moving along the internal nodes and selecting the branch that presents the correct answer to the question asked at each node. The answer to each question determines the next question. This continues until the leaf node is reached. The leaf node will represent the class to which the new observation is assigned to, according on the model's prediction (Podgorelec et al., 2002).

There are many decision tree algorithms available. In this study, the C4.5 algorithm is used, which is an improved version of ID3. Quinlan (1986) developed the ID3 algorithm. This is often considered one of the simplest decision tree algorithms. Like many other decision tree algorithms, it uses the top-down approach when building the decision tree (Singh & Giri, 2014). The ID3 algorithm utilises information gain (which is based on entropy) to determine the best split (Murthy & Salzberg, 1995).

Shannon entropy is one of the most popular splitting criteria used in decision tree algorithms. Entropy is the measure of randomness or impurity in a dataset and ranges from zero to one. If the outcome of an event, such as clients defaulting on their loan, is certain, the entropy will equal to zero. If the probability of an event occurring is 50%, the entropy will equal one.

The equation for entropy is given by

$$Entropy = \sum_{i=1}^{c} p_i \log p_i \,, \tag{3.3.1}$$

where c = the number of classes and p_i is the proportion of values that fall within the class i, that is, p_i is the probability of the event (Fakir et al., 2020; Gulati et al., 2016).

Entropy before the split minus entropy after the split is known as information gain. Information gain measures homogeneity in a dataset. The higher the information gain, the more informative the feature (Gulati et al., 2016).

The equation for information gain is given by

$$Gain(S, A) = Entropy(S) \; - \sum_{i=1}^{n} \frac{|S_i|}{|S|} * Entropy(S_i), \tag{3.3.2}$$

where

$|S|$ = total number of observations in S;
$|S_i|$ = number of observations in the subset $S_i|$;
n = number of attribute A.

When choosing the test feature at each test node, the entropy of the parent node and the entropy of each category in each feature is calculated. This is used to calculate the information gain for each feature. The feature which maximises the information gain is selected as the splitting criterion for that specific node. This process starts at the root node and runs in a recursive manner by treating the new child node as a parent node. The splitting process continues until

the new node is a leaf node, that is, all observations in the node belong to the same class, all observations have identical feature values, or a predefined stopping criterion is met. A tree is generated once all branches end with leaf nodes. This method is often referred to as the divide-and-conquer approach (Fakir et al., 2020; Murthy & Salzberg, 1995).

Stopping criteria are utilised during the learning process. They are used to reduce over-fitting by preventing insignificant branches from being generated; however, it may also result in loss of information. Common pre-pruning/stopping conditions that may be utilised when building a decision tree are listed below:

- All observations in a node belong to the same class.
- The maximum pre-specified tree depth is reached.
- The decision node has fewer observations than the (pre-specified) minimum number of observations allowed in a parent node.
- After the node is split, the number of observations in at least one child node is less than the pre-specified minimum number of observations allowed in a child node (Singh & Giri, 2014).
- A statistical test can also be utilised to determine whether splitting the data on a certain attribute is statistically significant. If the null hypothesis cannot be rejected, the split does not occur and the splitting stops at that node (Patel & Upadhyay, 2012).

Using a stopping criterion may result in leaf nodes consisting of observations that belong to different classes. The majority vote rule can be utilised to determine the class of that leaf node (Quinlan, 1986).

The main disadvantage of ID3 is that it only supports categorical data and not continuous data. Using the same method on continuous data will result in very small subsets. This will lead to continuous attributes with unrealistically high information gain, which will likely result in a poor model (Fakir et al., 2020). Another issue that arises when using the ID3 algorithm is that it uses the greedy approach. ID3 chooses the best locally optimal split at each node without allowing for backtracking. Using a strategy that chooses locally optimal splits may result in a suboptimal tree (Murthy & Salzberg, 1995).

C4.5 is an extension of ID3 to overcome disadvantages of the latter and was also developed by Quinlan (Adhatrao et al., 2013). The C4.5 algorithm utilises the same method as ID3 when handling categorical features; however, unlike ID3, C4.5 can handle continuous attributes.

Figure 3.2 displays a decision tree that consists of both categorical and continuous variables.

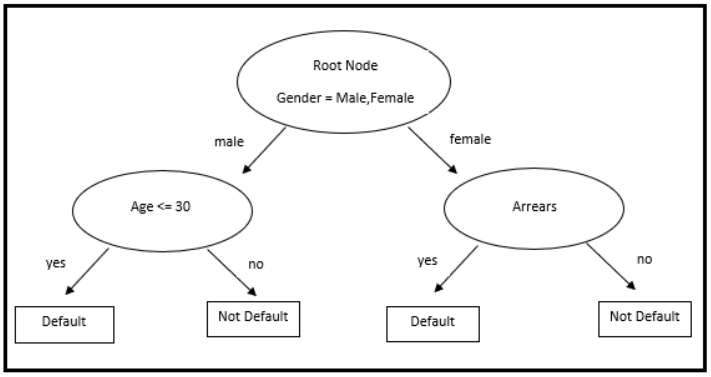

Figure 3. 2: Example of a decision tree that includes both categorical and continuous variables

In order to handle features that are continuous, the C4.5 algorithm converts the continuous features into nominal features by creating a threshold that is used to split the data into two subsets. When selecting the threshold value, all values of the continuous feature are considered as an option. For each possible threshold value, the dataset is split into two subsets: those whose feature value is less than or equal to the possible threshold value, and those whose values are greater. The entropy and information gain are then calculated by using each possible threshold value; the one selected is the threshold value that offers maximum gain (Fakir et al., 2020; Gulati et al., 2016).

Another advantage that the C4.5 algorithm has over the ID3 algorithm is that it allows for post-pruning. Decision tree classifiers aim to split the training set into subsets, such that each subset includes observations that all belong to the same class. Often, fitting a decision tree until all leaf nodes consist of observations that belong to the same class results in over-fitting (Podgorelec et al., 2002). The latter occurs when the learning algorithm continues to develop hypotheses, such that the training set error decreases, but the test set error increases (Patel & Upadhyay, 2012). By overfitting, the decision tree is classifying the training set instead of the overall population and the algorithm becomes too specific to the training set (Podgorelec et al., 2002).

To avoid overfitting, many decision tree algorithms use a method called ‘pruning’. Pruning helps to optimise the computational efficiency and classification accuracy of a decision tree. In most cases, pruning leads to a reduction of the tree size (Patel & Upadhyay, 2012). Post-pruning allows the tree to grow to its maximum size and thereafter, insignificant branches are removed. Examples of post-pruning methods are given by the authors, namely reduced error pruning, error complexity pruning, minimum error pruning, pessimistic pruning and cost-based pruning (Patel & Upadhyay, 2012).

3.4 Random forest

Ensemble classification is a method which uses multiple classifiers that work together in order to assign a new observation to a class. Generally, the ensemble of classifiers has a higher accuracy than the individual classifiers in the ensemble. Random forest classification is an example of an ensemble classification algorithm in which the base learners are decision trees (Fawagreh et al., 2014). Breiman (2001) developed the random forest algorithm and Breiman et al. (1984) proposed using the classification and regression trees (CART) method to build the individual trees in the random forest. As its name suggests, CART can be used to build both classification and regression trees. If the target variable is categorical (e.g., defaulting or not defaulting), a classification tree is used and if the target variable is continuous, a regression tree is used (Kuhn et al., 2013). The CART algorithm builds binary decision trees, in other words, it splits the data at each test node into two subsets.

The splitting criteria used in CART to determine the feature that best splits the data is the Gini index, which is a measure of impurity (Singh & Giri, 2014).

The Gini index is defined by

$$Gini(D) = 1 - \sum_{j=1}^{n} (P_j)^2 , \tag{3.4.1}$$

where P_j denotes the probability of an observation being classified as class j.

If the features are categorical, the feature with the minimum Gini index is chosen as the splitting feature at each test node. If the features are continuous, a strategy similar to the one used by the C4.5 algorithm for information gain (discussed in section 3.3) is used for the Gini index. The point of a continuous feature that minimises the Gini index is used to split the data (Han et al., 2006; Lin & Fan, 2019). When building a decision tree, CART (like C4.5) uses the recursive partitioning method, also known as the divide-and-conquer method, until all nodes cannot be split further or one of the stopping criteria is met.

To construct a random forest, Breiman (2001) combined the CART approach, the bagging sampling approach, also known as bootstrap aggregation, which he developed in 1996 (Breiman, 1996), and random feature selection. As a result, individual trees in the forest are trained by using different datasets and features to make predictions. This reduces correlation between the individual decision trees in the model.

During the building process of the random forest model, a training set needs to be created. The bootstrap random sampling method is used to draw N samples from the original dataset. The samples are created with replacement and each of the bootstrap samples should include the same number of observations, which is usually about two-thirds the size of the original dataset (Ali et al., 2012; Gao, Wen, & Zhang, 2019). This method helps to reduce instability in the model (Tyralis et al., 2019).

A decision tree for each bootstrap training set is then constructed. These N trees form the random forest. Breiman (2001) uses the CART method when building decision trees for the random forest. Each tree should be grown to its full size, that is, until the leaf nodes are pure, or until the leaf node contains a specified number of observations, or another stopping criterion is met. These trees should not be pruned (Ali et al., 2012).

When building a single decision tree that is not part of a random forest, all possible features are considered and the feature that results in the largest decrease in impurity is chosen as the splitting feature for that node. In a random forest, each tree selects m features randomly from all the features and only these m features are considered when selecting a feature to split the data at each node. This method adds more variation between the trees and further randomness to the model, which reduces correlation between the individual trees, resulting in the reduction of variance of the prediction (Gao, Wen, & Zhang, 2019; Tyralis et al., 2019).

The final prediction for the classification problem is determined by using the majority vote rule based on each individual tree's vote, that is, each decision tree in the random forest makes a prediction and the class with the most votes is the random forest model's final prediction (Gao, Wen, & Zhang, 2019). The random forest will only predict incorrectly when a higher portion of the individual trees predict the incorrect class.

According to Breiman (2001), the error rate of a random forest depends on the strength of the trees and the correlation between the trees. An increase in correlation between two trees in the

random forest will result in an increase in the error rate of the random forest. An increase in the strength of the trees in the random forest will result in a decrease in the error rate of the random forest (Fawagreh et al., 2014; Tyralis et al., 2019).

To compute the error rate, out-of-bag observations can be used. Out-of-bag samples are those ones that were not included in the bootstrap training set (Chen & Ishwaran, 2012). These observations can also be used for parameter tuning. Some of the parameters that are often tuned are the number of trees, the number of observations in each tree, the number of randomly selected predictor variables, and stopping criteria such as the number of observations in each leaf node (Tyralis et al., 2019).

3.5 Support vector machines

The support vector machine, also known as SVM, was introduced by Vapnik (1982). SVM is a supervised learning algorithm that can be used for both linear and nonlinear classification as well as regression (Jakkula, 2006). For classification problems, which will be focused on in this study, support vector machines separate the observations into classes, based on the observation's features, by identifying the optimal hyperplane (Rampisela & Rustam, 2018). Vapnik initially described an optimal hyperplane as "a linear decision function with maximal margin between the vectors of the two classes", but extended this method to cater for non-linearly separable training data (Vapnik, 1995).

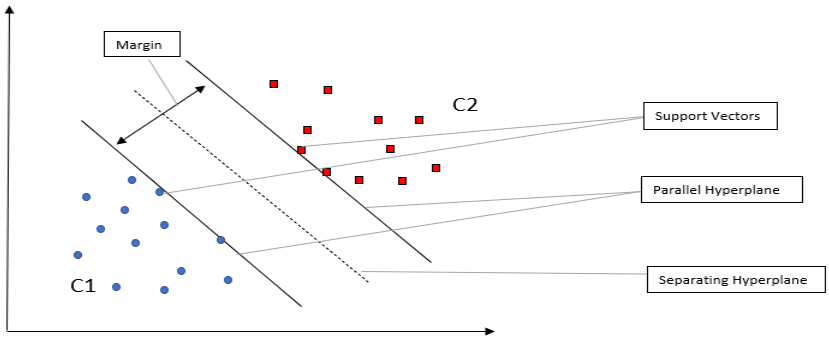

Figure 3. 3: Example of support vector machine structure

Figure 3.3 illustrates the structure of a support vector machine with two linearly separable classes, namely C1 and C2. In this study, the objective is to separate clients into the default (C1) and not default (C2) classes. Figure 3.3 shows the separating hyperplane, parallel hyperplanes, support vectors, and the margin. Hyperplanes are the decision boundaries that help to classify the observations, support vectors are the data points from each class that are situated closest to the separating hyperplane, and the margin is the distance between the parallel hyperplanes associated with the support vectors of the two classes. The number of hyperplanes which could separate the data into the two classes are infinite. The aim is to identify the separating hyperplane that is as far as possible from the support vectors of both classes, thus maximising the margin between the support vectors (while trying to minimise the error when the data are not fully linearly separable) so that the training data are classified correctly and the

model works well on unseen data. The assumption is that the larger the distance between the two parallel hyperplanes, the smaller the classification error will be (Bhavsar & Panchal, 2012; Pal & Mather, 2005). During the training phase, all observations in the training set need to be available when the SVM model's parameters are obtained. Once these parameters are obtained, only the support vectors are required when predicting the unseen data (Awad & Khanna, 2015).

There are different types of support vector machines, namely a hard margin SVM, a soft margin SVM and a kernel method SVM. The SVM utilised depends on the complexity of the classification problem.

A **hard margin SVM** is an SVM which can fully linearly separate the data into the correct classes by identifying the hyperplane with the maximum margin (Fletcher, 2008). Figure 3.3 represents a hard margin SVM.

Consider a problem with N number of observations where each input x_i has D features and belongs to one of two classes, namely y_i = -1 or y_i = 1; in other words, the training data used to build the model is in the form

$\{x_i, y_i\}$, where i= 1, 2,.., N, x E R^D and y_i E {-1,1}.

Now consider the following function:

$$g(x) = \omega^T x + b. \tag{3.5.1}$$

The separating hyperplane is given by the equation

$$\omega^T x + b = 0, \tag{3.5.2}$$

and the equations for the parallel hyperplanes, which are the planes that the support vectors lie on, are given below:

$$\omega^T x + b = -1, \tag{3.5.3}$$
$$\omega^T x + b = +1, \tag{3.5.4}$$

where ω E R^n is a p-dimensional vector that is perpendicular to the separating hyperplane, b E R is a parameter that relates to the closest distance between the origin of coordinates and the separating hyperplane, and 1 and -1 relate to the two classes. The hyperplane will pass through the origin in the absence of b (Bhavsar & Panchal, 2012; Rampisela & Rustam, 2018).

ω and b are selected such that the training data can be described by

$$x_i\omega + b \leq -1 \text{ when } y_i = -1 \tag{3.5.5}$$
$$\text{and } x_i\omega + b \geq +1 \text{ when } y_i = +1. \tag{3.5.6}$$

Combining equations (3.5.5) and (3.5.6) will result in the following:

$$y_i\,(x_i\omega + b) - 1 \geq 0 \text{ for all } i\text{'s}. \tag{3.5.7}$$

The distance from each observation to the hyperplane is $\frac{|g(x)|}{||\omega||}$ (Fletcher, 2008). To find ω and b such that $g(x)$ equals to 1 and -1 for the closest point belonging to each of the two classes, ω_1 and ω_2, respectively, the margin can be described as follows:

$$\frac{1}{||\omega||} + \frac{1}{||\omega||} = \frac{2}{||\omega||}\ , \tag{3.5.8}$$

where $\omega^T x + b = +1$ for xE ω_1 and $\omega^T x + b = -1$ for xE ω_2.

In order to maximise the margin and solve the optimisation problem, the following objective function (in primal form) will need to be minimised:

$$J(w) = \frac{1}{2}||\omega||^2, \tag{3.5.9}$$

s.t. $y_i(\omega_i^T x + b) \geq 1$ where i =1,2,…,N (Awad & Khanna, 2015; Nayak et al., 2015).

In an optimisation problem where the variables being optimised have constraints, the constraints, multiplied by the Lagrange multipliers, are added to the error function in order to augment it. With support vector machines, the Lagrangian function is attained by augmenting the objective function with a weighted sum of the constraints.

$$L(\omega, b, \lambda) = \frac{1}{2}\omega^T\omega - \sum_{i=1}^{N} \lambda_i[\, y_i(\omega^T x_i + b) - 1], \tag{3.5.10}$$

where λ_i's are the Lagrange multiplies and ω and b are referred to as primal variables.

The Karush-Kuhn-Tucker conditions, also known as KKT conditions, are used to generalise the Lagrange multipliers when inequality constraints are present (Awad & Khanna, 2015). Using the KKT conditions and partially differentiating with respect to w, b and λ, the following equations are obtained:

$$\omega = \sum_{i=1}^{N} \lambda_i\, y_i x_i, \tag{3.5.11}$$

subject to

$$\sum_{i=1}^{N} \lambda_i\, y_i = 0. \tag{3.5.12}$$

The optimisation problem can be solved in primal form or in dual form. The primal form is in terms of ω and b, whilst the dual form is in terms of λ (Srivastava & Bhambhu, 2010). According to these authors, the primal problem can be written in the dual form by substituting equation (3.5.11) into the Lagrange function.

The problem in dual form for the hard margin SVM is given by

$$max_{\lambda}\,(\sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i\,j}^{N}\lambda_i\,\lambda_j y_i y_j x_i x_j), \tag{3.5.13}$$

subject to the following:

$$\sum_{i=1}^{N}\lambda_i\,y_i = 0, \quad \lambda_i \geq 0\ \forall i. \tag{3.5.14}$$

This problem can be solved by using a quadratic optimisation solver.

The dual form is often used, as it always has a unique optimal solution and is often quicker and more efficient compared to the primal form, since the number of primal variables can be considerably more than the number of dual variables (Awad & Khanna, 2015; Nayak et al., 2015).

In the real world, data are often not fully linearly separable. A **soft margin SVM** may be utilised to solve this problem. Figure 3.4 illustrates an example of an SVM structure in which some observations are misclassified. The SVM objective function can be modified by adding slack variables, also referred to as error variables, ξ_i, which allow for small misclassifications (errors), that is, if the data points are on the incorrect side of the separating plane by a short distance, the data points can be misclassified without violating the constraints (Jakkula, 2006). Data points on the incorrect side of the separating plane are penalised according to their distance away from the 'correct side' (Fletcher, 2008). Instead of finding a hyperplane that separates the data into two classes correctly and without any errors, the soft margin SVM algorithm now searches for a hyperplane that maximises the margin while trying to minimise the error (Pal & Mather, 2005).

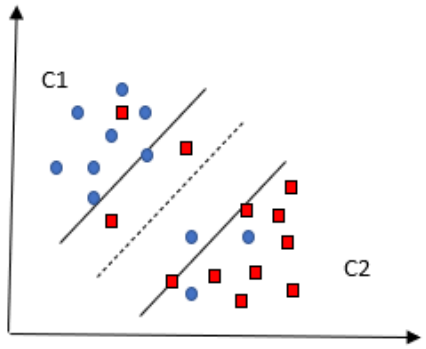

Figure 3. 4: Example of support vector machine structure with misclassifications

When including the slack variable, ω and b are selected such that the training data can be described by

$$x_i\omega + b \leq -1 + \xi_i \quad \text{when } y_i = -1 \tag{3.5.15}$$

$$\text{and } x_i\omega + b \geq +1 + \xi_i \quad \text{when } y_i = +1. \tag{3.5.16}$$

where ξ_i >= 0 for all i's (Fletcher, 2008).

Combining equations (3.5.15) and (3.5.16) results in the following:

$$y_i\ (x_i\omega + b) - 1 + \ \xi_i\ \geq 0 \text{ where } \ \xi_i \geq 0 \text{ for all } i\text{'s.} \tag{3.5.17}$$

In order to maximise the margin and solve the optimisation problem, the following objective function (in primal form) will need to be minimised (Awad & Khanna, 2015; Nayak et al., 2015):

$$J(\omega, b, \xi) = \frac{1}{2}\left|\left|\omega\right|\right|^2 + C\sum_{i=1}^{N}\xi_i, \tag{3.5.18}$$

subject to the following constraints:

$$y_i({\omega_i}^T x_i + b) \geq 1 - \xi_i \quad \text{where } i\ = 1,2,3,\ldots,\text{N}, \tag{3.5.19}$$

$$\xi_i \geq 0 \qquad \text{where } i\ = 1,2,3,\ldots,\text{N}. \tag{3.5.20}$$

The parameter C is the regularisation term that is based on the optimisation goal. It controls the trade-off between the margin size and misclassification error. As C increases, the margin gets smaller (SVM prioritises the minimisation of the number of misclassification errors). As C decreases, the number of misclassifications allowed increases (SVM prioritises the maximisation of the margin between the classes) (Rampisela & Rustam, 2018).

The problem in dual form for the soft margin SVM is given by

$$max_\lambda\ (\sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i\ j}^{N}\lambda_i\,\lambda_j y_i y_j x_i x_j), \tag{3.5.21}$$

subject to the following constraints:

$$\sum_{i=1}^{N}\lambda_i\, y_i = 0, \tag{3.5.22}$$

$$0 \leq \lambda_i \leq C \quad i = 1,2,3,\ldots,N.$$

The difference between the hard margin dual problem and the soft margin dual problem is that the dual variable for the soft margin is upper bounded by C (Awad & Khanna, 2015; Nayaket al., 2015).

The SVM method can be extended to cater for cases where a nonlinear boundary exists by using a **kernel function** (Jakkula, 2006). A kernel function is used to project the input data to a higher dimensional space by the function ϕ. The SVM will then try to obtain the optimal linear separating hyperplane in the higher dimensional space (Srivastava & Bhambhu, 2010). When kernels are utilised, all computations are carried out in the input space and no computations are performed in the higher dimensional space. The hyperplane in the input space corresponds to a nonlinear decision function, which is based on the kernel used (Hearst et al, 1998). Mercer's conditions need to be satisfied when using the kernel method.

The kernel function is defined below:

$$\mathrm{K}(\mathrm{x}_i, \mathrm{x}_j) = \varphi(\mathrm{x}_i)\varphi(\mathrm{x}_j), \tag{3.5.23}$$

where the $\varphi(x)$ maps x E R^n to a higher dimensional feature space and belongs to the Hilbert space. $\varphi(\mathrm{x}_i)\varphi(\mathrm{x}_j)$ can be replaced by $\mathrm{K}(\mathrm{x}_i, \mathrm{x}_j)$ in the classification algorithm (Rampisela & Rustam, 2018).

The primal form for the problem is given below:

$$min_{\omega,\xi}\, \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{N}\xi_i, \tag{3.5.24}$$

subject to the following constraints:

$$y_i(\omega^T\varphi(\mathrm{x}_i) + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0\ \forall i. \tag{3.5.25}$$

The KKT conditions will also need to be satisfied when using the kernel method to separate data (Awad & Khanna, 2015).

The dual form for the problem is shown below:

$$max_{\lambda}\,(\sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i\ j}^{N}\lambda_i\,\lambda_j y_i y_j x_i x_j), \tag{3.5.26}$$

subject to the following constraints:

$$\sum_{i=1}^{N}\lambda_i\, y_i = 0. \tag{3.5.27}$$

The kernel function is chosen, based on the problem. Equations (3.5.28), (3.5.29) and (3.5.30) display kernel functions that are commonly utilised:

- Linear kernel

$$\mathrm{K}(\mathrm{x}_i, \mathrm{x}_j) = \mathrm{x}_i^T \mathrm{x}_j + \mathrm{C}. \tag{3.5.28}$$

- Polynomial

$$\mathrm{K}(\mathrm{x}_i, \mathrm{x}_j) = (\gamma \mathrm{x}_i^T \mathrm{x}_j + \mathrm{r})^d \quad , \gamma > 0. \tag{3.5.29}$$

- Radial basis function

$$\mathrm{K}(\mathrm{x}_i, \mathrm{x}_j) = e^{-\gamma \left|\left|\mathrm{x}_i - \mathrm{x}_j\right|\right|^2} \quad , \gamma > 0. \tag{3.5.30}$$

where $\gamma, r \; and \; d$ are kernel parameters.

3.6 Naive Bayesian algorithm

The Naive Bayesian algorithm is a statistical classification method that was proposed by Thomas Bayes, a British scientist (Wibawa et al., 2019). It is a Bayesian network that is based on the Bayesian theorem, combined with the assumption of independence among features (Ginting et al., 2018). The Naïve Bayesian classifier classifies data by identifying the class with the maximum posterior probability given a set of features that were observed. The new observation is then assigned to this class (Demichelis et al., 2006).

The formula for the Bayesian theorem on which the Naïve Bayesian classifier is based, is given by

$$P(Y|X) = \frac{P(X|Y) * P(Y)}{P(X)}, \tag{3.6.1}$$

where
X: A feature vector for an observation with n features values, i.e., X = (x_1, x_2,…, x_n).
Y: The hypothesis that an observation belongs to a specific class.
P(X): Prior probability of X, regardless of Y.
P(Y): Prior probability of Y, regardless of X.
$P(X|Y)$: Conditional probability of X, given hypothesis Y.
This is referred to as the posterior probability of X or the conditional probability.
$P(Y|X)$: Probability of hypothesis Y, given the feature vector X (posterior probability of Y) (Demichelis et al, 2006; Gangrade & Patel, 2012).

The Bayesian theorem uses P(X), P(Y) and $P(X|Y)$ to determine the posterior probability $P(Y|X)$ (Gao, Zeng, & Yao, 2019). The Naïve Bayes classifier calculates the probability of an observation belonging to a specific class by using the Bayes formula (Liu et al., 2014).

The Naïve Bayesian algorithm is often referred to as naïve, as it is based on a very strong assumption, namely the class independence assumption. This means that the effect of a particular feature in a class is assumed to be independent of the other features in the class. In our study, we aim to classify an applicant as default or not default, based on the applicant's age, salary, credit history et cetera. These features will be considered independently, even if they are interdependent. The independence assumption is often not true in the real world; however, the model is often still able to generate reliable results for classification (Jang et al., 2015). The assumption is made in order to simplify the computation of $P(X|Y)$, as discussed below (Gao, Zeng, & Yao, 2019).

Consider a sample of observations where each observation has a feature vector X with n features, that is, x_1, x_2,…, x_n, and each observation needs to be assigned to a class Y.

Since the Naive Bayes classifier assumes independence of class conditions, the formula for the Naïve Bayes theorem can be written as

$$P(y|x_1, \dots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)\dots P(x_n)}. \tag{3.6.2}$$

Since $P(X)$ is constant for all classes, only the prior probability $P(y)$ and the conditional probability $P(x_i|y)$ will need to be computed (Berrar, 2018).

That is,

$$P(y|x_1, \dots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y). \tag{3.6.3}$$

We can estimate $P(Y = y)$ by using the following formula:

$$P(Y = y) = \frac{N_y}{|D|}, \tag{3.6.4}$$

where
N_y : number of observations in class Y and
$|D|$: number of observations in the training set D (He et al., 2012).

The method used to calculate $P(x_i|y)$ depends on whether the data are categorical or continuous. If the data are categorical, equation (3.6.5) can be used to estimate the conditional probability

$$P(X_i = x_i|Y = y) = \frac{N_{yi}}{N_y}, \tag{3.6.5}$$

where
N_y : number of observations in class Y and
N_{yi} : the number of observations in class Y where $X_i = x_i$ (He et al., 2012).

When the data is numeric, kernel density estimation can be used to estimate $P(X_i = x_i|Y = y)$, which is given by

$$P(X_i = x_i|Y = y) = \frac{1}{N_y h_i^y} \sum_{k=1}^{N_y} K(\frac{x_i - x_{i,k}}{h_i^y}), \tag{3.6.6}$$

where

N_y is the number of observations in class Y
K : kernel function,
h_i^y : smoothing parameter for the i-th feature in class Y, and
$x_{i,k}$: i-th feature in the k-th sample for Class Y (He et al., 2012).

According to He et al. (2012), the standard Gaussian function is often used as the kernel function and is given by

$$K(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right). \tag{3.6.7}$$

The Naïve Bayes Classifier utilizes the equation; $P(y|x_1, \dots, x_n) = P(y) \prod_{i=1}^{n} P(x_i|y)$ to estimate the probability of an observation belonging to each class. During the classification phase, the Naïve Bayes classifier, given by equation (3.6.8) (Krichene, 2017), is used to identify the class with the highest probability and the new observation will be assigned to it:

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y). \tag{3.6.8}$$

3.7 K-nearest neighbours

The k-nearest neighbours (K-NN) algorithm was initially proposed by Fix and Hodges (1951) and was later modified by Cover and Hart (1967). K-NN is a non-parametric algorithm that can be used to solve both classification and regression problems. It has gained popularity due to its simplicity and easy implementation and is often used as a baseline classifier in classification problems (Hu et al, 2016).

K-NN is a distance-based classification algorithm that assigns a new observation to a class by calculating the distance between the new observation and the observations in the training set in order to find the new observation's nearest neighbours. The new observation is then classified according to its k-nearest neighbours by using the majority rule (Ali et al., 2019). This classifier is often referred to as a lazy learner. Lazy learners are memory-based algorithm in which the learning phase is usually quite simple, whereas most of the work is done in the classification phase. A K-NN classifier stores the training data during the learning phase and

only uses them in the classification phase when new observations need to be classified (Mazinani & Fathi, 2015).

There are four main steps in the K-NN process. During the first step, the number of k-nearest neighbours used in the algorithm will need to be selected. The 'k' in k-nearest neighbours is a parameter used to define the number of nearest neighbours that are included in the voting process (Ali et al., 2019). The value of k may significantly impact the performance of the K-NN algorithm, as it may alter the prediction.

From Figure 3.5, we observe that if the value of k was set to 5, the model would predict that the new data point belonged to Class B. However, if the value of k was set to 12, the model would predict that the new data point belonged to Class A.

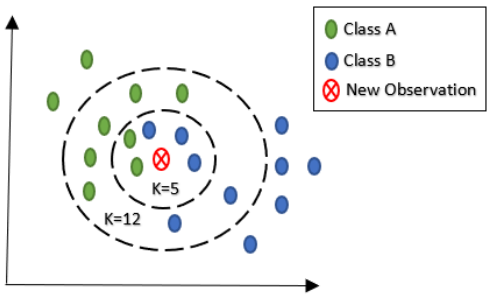

Figure 3. 5: Impact of selected k value on model's prediction

A k-value that is too small will result in overfitting, whereas one that is too large will result in underfitting. Common methods utilised when selecting the k-parameter are discussed below.

- An odd value is usually chosen to prevent ties in the voting process when the output is binary.
- K = 1 is the simplest form of the K-NN rule and is often used as a benchmark, as it provides reasonable results (Hu et al., 2016).
- When assigning k, a fixed value method across all test samples may be used (Zhang et al., 2018).
- Lall and Sharma (1996) mentioned using k = N^(0.5) as the fixed optimal k-value across different test samples when the number of training samples N > 100.
- The alternative to assigning a fixed optimal k-value for all test samples is to assign different optimal k-values for each test sample (Zhang et al., 2018). There are different methods available that may be used. For example, the algorithm can be run multiple times, using different values for k each time. The k-value that results in the best performance is the one that should be chosen (Guo et al., 2003).

During the second step, a distance metric needs to be selected. There are numerous methods available that may be used to compute the distance between two points. Some of the common distance calculations used in K-NN to measure the distance between two vectors x and y, where $x = \{x_1, x_2, \ldots, x_n\}$ and $y = \{y_1, y_2, \ldots, y_n\}$, are given below:

- Minkowski distance measure

$$D_{Mink}(x, y) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}. \quad (3.7.1)$$

This is a generalised metric where p is a positive value, x_i is the ith value in the vector x, and y_i is the ith value in the vector y (Alfeilat et al., 2019).

- Manhattan distance measure

$$MD(x, y) = \sum_{i=1}^{n} |x_i - y_i|. \quad (3.7.2)$$

This is a special case of the Minkowski distance measure where p = 1 (Alfeilat et al., 2019).

- Euclidean distance measure

$$ED(x, y) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}. \quad (3.7.3)$$

This is a special case of the Minkowski distance measure where p = 2. The Euclidean distance measure is an extension to the Pythagorean theorem (Alfeilat et al., 2019).

- Normalised Euclidean distance measure

$$NED(x, y) = \sqrt{\frac{\sum_{i=1}^{n} (x_i - y_i)^2}{n}}. \quad (3.7.4)$$

- Hamming distance metric

$$HamD(x, y) = \sum_{i=1}^{n} 1_{x_i \neq y_i}. \quad (3.7.5)$$

The Hamming distance metric measures how many mismatches there are between two vectors (Hamming, 1958).

- Pearson correlation distance measure

$$PeaD(x,y) = 1 - \frac{\sum_{i=1}^{n}(x_i - \bar{x}) * (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 * \sum_{i=1}^{n}(y_i - \bar{y})^2}}. \quad (3.7.6)$$

  Pearson correlation distance measures the linear relationship between two vectors. The Pearson correlation coefficient is used to derive the Pearson correlation distance (Alfeilat et al., 2019).

According to Hu et al. (2016), the Euclidean distance method is the most popular distance metric used in literature.

During the third step, the distance between the new data point and each training data point is calculated (using the chosen distance metric) (Ali et al., 2019). The final step involves sorting the data points based on their distance away from the new observation and identifying the k-nearest neighbours (where k was selected during step one). The majority voting rule is then used to assign the new observation to a class, that is, the new observation is assigned to the most common class among those k data points (Ali et al., 2019).

3.8 Artificial neural network

In 1943, Warren McCulloch, a neurophysiologist, and Walter Pitts, a mathematician, introduced a simple neural network, which was the first step towards the artificial neural network (ANN) (McCulloch & Pitts, 1943). The latter is a modelling tool used for pattern classification problems. It has non-parametric, non-linear and adaptive learning properties (Hu et al., 1999). ANNs aim to replicate the human brain, which is made up of interconnected cells called neurons. The neurons in the artificial network strive to copy the structure and behaviour of these human cells (Shiruru et al., 2016).

The artificial neural network comprises different layers of interconnected cells; the complexity of this network depends on the complexity of the problem. ANN models used for classification problems are either feed-forward or feed-back neural networks, which are also known as back-propagation neural networks. In feed-forward neural networks, information flows in a single direction. The simplest artificial neural network is called a single-layer perceptron, which only has an input layer and an output layer, whereas a multiple layer perceptron has an input layer, at least one hidden layer, and an output layer. A feed-back neural network has at least one hidden layer and one feedback loop (Shahid et al., 2019).

This study focuses on the feed-back neural network, which uses an iterative process in order to train the model. Figure 3.6 illustrates the architecture of a feed-back neural network with one hidden layer, such that the network includes three layers, namely the input layer, the hidden layer and the output layer. The feed-back neural network consists of two phases i.e., forward-propagation and back-propagation. During the forward-propagation phase, information moves forward through the network, from input layer to hidden layer and then to output layer. During

this phase, the network produces an output value, which is a prediction. During the back-propagation phase, information is propagated backward through the network and used to adjust the weights in order to minimise the error between the predicted output and actual value.

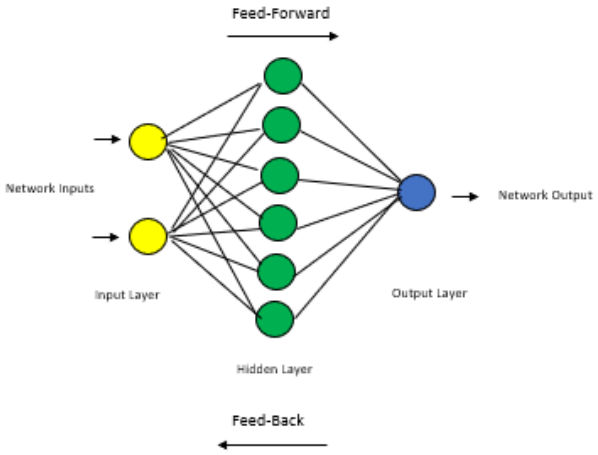

Figure 3. 6: Structure of a feed-back neural network

All artificial neural networks comprise the following components: the input signals, weights, bias, activation function, and the output. Figure 3.7 represents the architecture of a simple neural network:

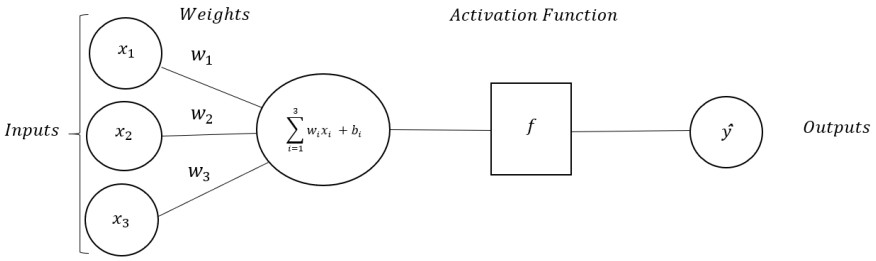

Figure 3. 7: Structure of a simple neural network

The feed-back neural network is more complex in comparison to the architecture of the neural network shown in Figure 3.7, as it includes at least one hidden layer and one feedback loop; however, the components within the input layer, hidden layer/s and output layer are similar.

In feed-back neural networks, the input data from the external environment are received by the input layer**.** The data do not undergo any transformations in this layer. The input information is multiplied by weights and sent through to the processing components of the hidden layer in the artificial neural network (Lallahem & Mania, 2002). Each input in the neural network has a corresponding weight. These weights are initially randomly assigned (Pasini, 2015); they may be positive, negative, or zero. Inputs multiplied by the corresponding weights produce the strength of a signal (Shiruru et al., 2016).

The weighted inputs from the input layer (assuming one hidden layer only) are received by the hidden layer; the latter is required when there are complex patterns in the data. A feed-back neural network contains one or more hidden layers, which are found between the input layer

and the output layer (Hu et al., 1999). Generally, by adding more hidden layers, we can improve the performance of the model; however, adding too many hidden layers may result in overfitting (Moon et al., 2019). In the hidden layer, an activation function is applied to the weighted sum of the inputs and the bias term in order to transform the inputs. This is given by

$$y_i = \sigma(\sum_{i=1}^{n} w_i x_i + b_i), \tag{3.8.1}$$

where x_i is representative of the inputs, w_i is the weight assigned to each input, b_i is the bias term that allows one to shift the activation function, σ is the activation function, and y_i is the output value of the neuron. The activation function uses the strength of the signal to determine whether the neuron should be activated and whether the signal from the hidden layer will be passed on to the next neuron (Oken, 2017).

In the absence of an activation function, the output of each layer can range from anything between -∞ and +∞, and each layer's outputs will be a linear function of the layer's inputs. The activation function adds non-linearity to the outputs and allows the output to range from 0 to 1 or from -1 to 1. It also allows for back propagation, where the gradients and error information are used to adjust weights and biases. There are two types of activation functions, namely linear and non-linear activation functions. Non-linear activation functions are more popular in comparison to the linear functions, as they allow the model to generalise easily as well as differentiate between outputs (Feng & Lu, 2019). The sigmoid function, tangent hyperbolic function (tanh), and rectified linear unit function (ReLU) are commonly used non-linear activation functions and are given by the following equations:

Sigmoid function:

$$\mathrm{f}(\mathrm{x_i}) = \frac{\mathrm{e}^{\mathrm{x_i}}}{1 + \mathrm{e}^{\mathrm{x_i}}} \tag{3.8.2}$$

Tangent hyperbolic function:

$$\tanh(x_i) = \frac{\sinh(x_i)}{\cosh(x_i)} \tag{3.8.3}$$

Rectified linear unit function (ReLU):

$$\mathrm{f}(x_i) = \max(0, x_i) = \begin{cases} x_i, & x_i > 0 \\ 0, & x_i < 0 \end{cases} \tag{3.8.4}$$

where x_i is the input of the activation function f (Feng & Lu, 2019).

According to Feng and Lu (2019), ReLU, which was proposed by Nair and Hinton (2010), is one of the most popular activation functions used in the hidden layer in ANN, as it can correct the vanishing gradient problem that the Sigmoid function and Tanh function face. The disadvantage of ReLU is that it suffers from the dying ReLU problem, where weights cannot be adjusted during back propagation when the gradient is 0; this happens when inputs are negative. New activation functions were proposed to improve the ReLU function, for example, the Leaky ReLU function, Parametric ReLU function and randomised ReLU function (Feng & Lu). Bing et al. (2015) compared the performance of the different ReLU functions and found that the modified ReLU functions all performed better than the original ReLU function.

The activation function in the hidden layer determines whether the neuron will fire or not. The output from the hidden layer is then multiplied by a second set of weights and passed to the output layer, where it is summed together along with a bias term and transformed by using another activation function (assuming there is one hidden layer in the network). According to Feng and Lu (2019), the sigmoid function is one of the most commonly used activation functions in the outer layer for a binary classification problem where the output is either 0 or 1. Since the sigmoid function outputs a value that ranges from 0 to 1, the sigmoid function output value can be easily interpreted as a prediction of 0 if the output value is less than 0.5, and 1 if the output value is greater or equal to 0.5.

The initial output value from the output layer determines the model's initial prediction. The predicted value is then compared to the actual value in order to calculate the error. The smaller the error, the closer the prediction is to the actual observation. The aim is to construct a function such that the predicted value will be equal to the actual target value, that is, $y_i = t_i$ where y_i is the predicted value and t_i the actual target value. To obtain the function, an error function can be used and the parameters that minimise the error function will need to be identified. A common error function is the sum of squared error between the actual and predicted value and is given by

$$\mathrm{E}(w_i, b_i) = \frac{1}{2}\sum_{i=1}^{p} \left|\left|t_i - y_i\right|\right|^2, \tag{3.8.5}$$

where t_i is the actual target value and the output, y_i, is dependent on the weights, w_i, and the bias term, b_i. In order to minimise the error function, w_i and b_i will therefore need to be adjusted appropriately (Oken, 2017). The information obtained from the error calculation is propagated backwards into the neural network, which is then used to update and change the weights and bias terms in the model. Thus, in training the model, you are adjusting the weights assigned to the signals (Hu et al., 1999). The weights can be adjusted via gradient descent. These new weights are then used in the neural network to recalculate the model's output and thereafter, by using the new predicted value, the error is recalculated; this information is then, once again, propagated backwards into the neural network and the weights and bias terms are adjusted again. The cycle of moving from input to output and then from output to input is known as an epoch. This process is repeated until the neural network learns the training data and the error is within a certain threshold. The backward process is called 'back propagation' and the back-propagation algorithm is used to reduce the error (Shiruru et al., 2016).

Gradient descent optimisation is one of the most popular algorithms used to optimise an artificial neural network by minimising the error function (Feng & Lu, 2019). In order to minimise the error by adjusting the weights using the gradient descent method, the derivatives of the error with respect to the weights and the bias in the artificial neural network need to be calculated. For each iteration, the weights and biases are updated in the opposite direction of the error function's gradient, in other words, a positive derivative implies that the error function will increase if the weights are increased; therefore, the weights should be decreased. A negative derivative implies that the error function will decrease if the weights are increased; therefore, the weights should be increased. If the derivative is zero, the weights should not be changed, as this shows that the model has reached a stable point (Tawfiq & Thirthar, 2013). There are three variants of gradient descent, namely stochastic gradient descent, batch gradient descent and mini-batch gradient descent. When stochastic gradient descent is used, weights are adjusted after each observation goes through the neural network; when batch gradient descent is used, weights are adjusted after all observations go through the neural network; and when mini-batch gradient descent, which is a combination of both methods, is used, small batches of data run through the neural network each time (Ruder, 2017).

3.9 Evaluation metrics

Sections 3.2 to 3.8 outlined the following machine learning models: logistic regression, decision trees, random forest, k-nearest neighbours, the Naïve Bayes algorithm, support vector machines and artificial neural networks. We train each model by using labelled data and the trained models are used to make predictions on unseen data. The model's ability to make correct predictions is evaluated in order to understand how well the model has performed. In this section, the confusion matrix as well as evaluation metrics such as accuracy, balanced accuracy, precision, recall, Area under ROC curve (AUC score) and the Gini coefficient are discussed.

The **confusion matrix** is a visual representation that summarises a classification algorithm's performance. It comprises four possible outcomes, namely true positive (TP), false positive (FP), true negative (TN), and false negative (FN). In this study, a positive case represents a client who defaulted on his or her loan at least three times in the first 12 months of the loan being granted, whereas a negative case represents a client who did not default on his or her loan at least three times in the first 12 months of the loan being disbursed. Thus, TP, FP, TN and FN can be defined as follows:

True positive (TP): TP represents the number of times the classification algorithm correctly predicted that the clients defaulted on their loan.

False positive (FP): FP represents the number of times the classification algorithm incorrectly predicted that the clients defaulted on their loan.

True negative (TN): TN represents the number of times the classification algorithm correctly predicted that the clients did not default on their loan.

False negative (FN): FN represents the number of times the classification algorithm incorrectly predicted that the clients did not default on their loan.

Table 3.1 displays the structure of a confusion matrix. TP, FP, TN and FN can be used to derive evaluation metrics such as accuracy, balanced accuracy, true positive ratio, true negative ratio, positive predictive value, negative predictive value and false positive rate.

Table 3. 1: Structure of a confusion matrix that illustrates the TP, FP, TN and FN

	Predicted client defaulted	Predicted client did not default
Client defaulted	True positive (TP)	False negative (FN)
Client did not default	False positive (FP)	True negative (TN)

Accuracy: This represents how often the model classifies observations correctly. It can be computed as $\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$. Accuracy is not very reliable when the dataset is imbalanced.

Balanced accuracy: The balance accuracy ratio describes the overall accuracy of the model while taking into consideration the imbalance in the dataset. The balance accuracy is computed as $\frac{\text{TPR + TNR}}{2}$ where $TPR = \frac{\text{TP}}{\text{TP+FN}}$ and $TNR = \frac{\text{TN}}{\text{TN+FP}}$.

True positive ratio (Sensitivity/Recall): Recall describes the proportion of positive cases that the classification algorithm identified correctly. True positive ratio is computed as $\frac{\text{TP}}{\text{TP+FN}}$.

True negative ratio (Specificity): Specificity describes the proportion of negative cases that the classification algorithm identified correctly. True negative ratio is computed as $\frac{\text{TN}}{\text{TN+FP}}$.

Positive predictive value (Precision): Precision represents the proportion of observations that the classification algorithm correctly identified as positive cases of the total number of observations classified as positive cases. Precision is computed as $\frac{\text{TP}}{\text{TP+FP}}$.

Negative predictive value: This value represents the proportion of observations that the classification algorithm correctly identified as negative cases of the total number of observations classified as negative cases. Negative predictive value is computed as $\frac{\text{TN}}{\text{TN+FN}}$.

False positive rate: This rate describes the proportion of negative cases that the classification algorithm incorrectly identified as positive cases. False positive rate is computed as $\frac{\text{FP}}{\text{FP+TN}}$ (Arjaria et al., 2021).

ROC curve: The receiver operating characteristic curve (ROC curve) is used to analyse a classification algorithm's performance, based on its ability to discriminate between classes. It is a plot of the true positive rate (TPR) against the false positive rate (FPR) under different thresholds (Schechtman & Schechtman, 2019). Each point on the graph is a different threshold

and together, all the points form the curve. The curve shows how the performance of the classifier changes as the threshold is changed. Figure 3.8 illustrates an example of a ROC curve.

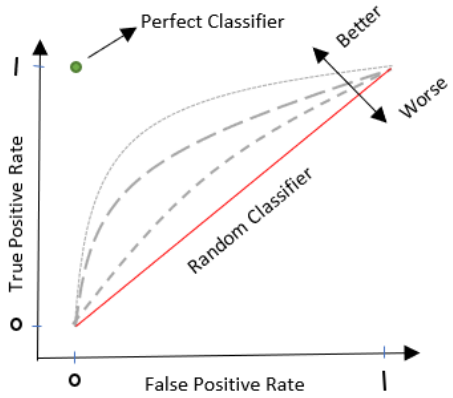

Figure 3. 8: Example of a ROC Curve

AUC score (area under the ROC curve): This is a single score that summarises the ROC plot. It is used to measure the ability of a classification algorithm to distinguish between positive and negative classes and can be used to compare the different models. The AUC value ranges from 0 to 1. A model will have an AUC value of 0 if all its predictions are wrong and a model that predicts 100% correctly will have an AUC value of 1. An AUC value of 0.5 is just as good as a random guess (Singpurwalla & Lai, 2020).

Gini coefficient: The Gini can be computed as (2*AUC) – 1 (Hand & Till, 2001). The relationship between AUC and Gini is linear. The Gini coefficient ranges from -1 to 1 where a random classifier has a score of 0 and a perfect classifier has a score of 1.

In this chapter several machine learning classification algorithms and evaluation metrics were discussed. In the next chapter, empirical results obtained when fitting these classification models to the default dataset using the PCA approach are presented.

Chapter 4

4 Classification with PCA

Each classification algorithm discussed in Chapter 3 was fitted to the balanced default dataset using the PCA approach, discussed in Chapter 2, to reduce the dimension of the dataset and to correct the dataset for multicollinearity. In this chapter, the confusion matrix and evaluation metrics, discussed in Chapter 2, are presented and examined for each model in order to identify the one which was most appropriate for the classification problem under study when using the PCA approach.

4.1 Logistic regression

Logistic regression models the probability of a discrete target variable given a set of input variables. In this study, the target variable is the default status of a client. The logistic regression model was fitted to the balanced default dataset and the results are examined.

The researcher assesses the fit of the model using the deviance test (discussed in Chapter 2) which is presented in Table 4.1; thereafter, the researcher analyses the maximum likelihood parameter estimates, p-values and odds ratios associated with each variable.

Table 4. 1: Deviance test for logistic regression model

Deviance statistic (D)	dof (n-p)	Deviance statistic/dof
18591	14454	1.286

Table 4.1 reports a deviance statistic/dof of 1.286 where the deviance statistic is 18591 and degrees of freedom (dof) are 14454. Since D/(n-p) = 1.286 < 1.5, we conclude that the model fits the data well.

The maximum likelihood parameter estimates and p-values for the fitted logistic regression model are presented in Table 4.2a.

Table 4. 2a: Maximum Likelihood Parameter estimates, and p-values for the fitted logistic regression model

	Parameter estimate	p-value
Intercept	0.617	<0.001***
principalcomponent1	-0.089	<0.001***
principalcomponent2	-0.517	<0.001***
principalcomponent3	0.182	<0.001***
principalcomponent4	-0.074	<0.001***
principalcomponent5	0.056	0.003***

principalcomponent6	0.063	<0.001***
principalcomponent7	0.186	<0.001***
Gender_M	0.388	<0.001***
LowerOffer_Yes	0.037	0.406
Married_Yes	-0.148	<0.001***
Weekly/Monthly_Weekly	0.019	0.612
Internal Living Expenses Rule_Yes	0.108	<0.001***
Unpaids_Yes	0.301	<0.001***
Taking Max_Yes	0.56	0.002***
PayslipExpenses_Yes	-0.034	0.368
Permanent Allowances_yes	-0.004	0.891
Overtime_yes	0.022	0.486
Union fees_yes	-0.089	0.011**
Pensionprovident_yes	-0.021	0.565
Medicalaid_yes	-0.26	<0.001***
Client Type_New	-0.112	0.044**
Client Type_Reload	-0.286	<0.001***
Insurance_yes	-0.175	<0.001***
Salary Bank_2	-0.068	0.334
Salary Bank_3	-0.129	0.104
Salary Bank_4	-0.276	<0.001***
Salary Bank_5	0.007	0.832
PersonalLoan_yes	0.084	0.074*
VehicleLoan_yes	-0.047	0.374
HomeLoan_yes	-0.292	<0.001***
Product Taken_PL New	0.400	<0.001***
Product Taken_PL Repeat	0.169	0.009***
Product Taken_Staff	-0.964	<0.001***
Product Taken_VL	0.005	0.902
InstallmentLoan_yes	0.073	0.104
Limiting Rule_2	0.003	0.952
Limiting Rule_3	0.148	0.006***
Limiting Rule_4	0.444	<0.001***
External Subsequent Lending_Yes	0.69	<0.001***
CreditCard_yes	-0.119	0.019**
RevolvingCredit_yes	-0.167	<0.001***
Credit Inactive_yes	0.039	0.546
Loan Purpose_Family Crisis	0.047	0.414
Loan Purpose_Housing and Ralated	-0.026	0.617
Loan Purpose_Other	-0.043	0.38
Loan Purpose_Other Emergency	-0.018	0.77
PropertyOwner_yes	0.016	0.845
Arrearrs_1	0.173	<0.001***

Note: ***, ** and * indicate significance at 1%, 5% and 10% level of significance respectively

From Table 4.2a, principalcomponent1, principalcomponent2, principalcomponent3, principalcomponent4, principalcomponent5, principalcomponent6, principalcomponent7, Gender_M, Married_Yes, Internal Living Expenses Rule_Yes, Unpaids_Yes, Taking Max_Yes, Medicalaid_yes, Client Type_Reload, Insurance_yes, Limiting Rule_3, Salary Bank_4, HomeLoan_yes, Product Taken_PL New, Product Taken_PL Repeat, Product Taken_Staff, External Subsequent Lending_Yes, Limiting Rule_4, RevolvingCredit_yes and Arrearrs_1 are significant at 1% level of significance, Union fees_yes, Client Type_New, and CreditCard_yes are significant at 5% level of significance and PersonalLoan_yes is significant at 10% level of significance. Thus, these 29 variables influence the prediction of loan defaulting clients; the estimates indicate how much they influence it. The researcher exponentiated the maximum likelihood parameter estimates (β_i coefficients) to obtain the odds ratio estimates, as the odds ratios can be interpreted more easily. Table 4.2b lists the 29 significant variables in the fitted logistic regression model and provides an interpretation of the odds ratio estimates for each variable.

Table 4.2b: Interpretation of the odds ratio estimates for the 29 significant variables in the fitted logistic regression model

Variable	Odds ratio estimates	Interpretation
principalcomponent1	0.915	One unit increase in principalcomponent1 is associated with an 8.5% ((1 – 0.915) x 100%) reduction in the odds of a client defaulting when all other variables are held constant
principalcomponent2	0.596	Increasing principalcomponent2 by one unit, decreases the odds of a client defaulting on their loan by 40.4% ((1 – 0.596) x 100%) when all other variables are held constant
principalcomponent3	1.200	One unit increase in Principalcomponent3 is associated with a 20% ((1.2-1)*100%) increase in the odds of a client defaulting, holding all other variables constant
principalcomponent4	0.929	Each unit increase in principalcomponent4 reduces the odds of a client defaulting by 7.1% ((1 – 0.929) x 100%), holding all other variables constant
principalcomponent5	1.057	Each unit increase in principalcomponent5 increases the odds of a client defaulting by 5.7% ((1.057-1)*100%), when all other variables are held constant
principalcomponent6	1.065	One unit increase in Principalcomponent6 is associated with a 6.5% ((1.065-1)*100%) increase in the odds of a client defaulting when all other variables are held constant
principalcomponent7	1.204	A single unit increase in principalcomponent7 increases the odds of a client defaulting on their loan by 20.4% ((1.204-1) x 100%), holding all other variables constant
Gender_M	1.474	Gender_M (male) is associated with a 47.4% ((1.474 – 1) x 100%) increase in the odds of a client

		defaulting when all other variables are held constant (Gender_M is equal to 1 if male and 0 if female)
Married_Yes	0.863	Married is associated with a 13.7% ((1-0.863)*100%) decrease in the odds of a client defaulting when all other variables are held constant (Married_Yes is equal to 1 if married and 0 if not married)
Internal Living Expenses Rule_Yes	1.114	The odds of a client defaulting increases by 11.4% ((1.114-1)*100%) if the internal living expenses rule is used, holding all other variables constant
Unpaids_Yes	1.351	The odds of a client defaulting increases by 35.1% ((1.351 – 1) x 100%) if the client has unpaids, holding all other variables constant
Taking Max_Yes	1.751	Taking Max is associated with a 75.1% ((1.751 – 1) x 100%) increase in the odds of a client defaulting when all other variables are held constant
Union fees_yes	0.915	The odds of a client defaulting reduces by 8.5% ((1-0.915)*100%) if the client pays union fees, holding all other variables constant
Medicalaid_yes	0.771	When all other variables are held constant, the odds of a client defaulting reduces by 22.9% ((1-0.771)*100%) if the client has medical aid
Client Type_New	0.894	Client Type New is associated with a 10.6% ((1-0.894)*100%) decrease in the odds of a client defaulting when all other variables are held constant
Client Type_Reload	0.751	There is a 24.9% ((1-0.751)*100%) decrease in the odds of a client defaulting when client type is reload, holding all other variables constant
Insurance_yes	0.839	If a client pays insurance, the odds of defaulting decreases by 16.1% ((1-0.839)*100%), when all other variables are held constant
Salary Bank_4	0.759	The odds of a client defaulting reduces by 24.1% ((1-0.759)*100%), if the client's main bank is Salary Bank 4, holding all other variables constant
PersonalLoan_yes	1.087	If the client has a personal loan, the odds of defaulting increases by 8.7% ((1.087 – 1) x 100%) holding all other variables constant
HomeLoan_yes	0.747	The odds of a client defaulting decreases by 25.3% ((1-0.747)*100%) if the client has a home loan, when all other variables are held constant
Product Taken_PL New	1.492	The odds of a client defaulting increases by 49.2% ((1.492-1)*100%), if the product taken is PL new, holding all other variables constant
Product Taken_PL Repeat	1.184	If product taken is PL Repeat, the odds of a client defaulting increases by 18.4% ((1.184-1)*100%), when all other variables are held constant
Product Taken_Staff	0.382	If product taken is Staff, the odds of a client defaulting reduces by 61.8% ((1-0.382)*100%), holding all other variables constant
Limiting Rule_3	1.159	The odds of defaulting increases by 15.9% ((1.159-1)*100%) if the client is limiting by rule 3, holding all other variables constant

Limiting Rule_4	1.559	If a client is limiting by rule 4, the odds of defaulting increases by 55.9% ((1.559-1)*100%), when all other variables are held constant
External Subsequent Lending_Yes	1.893	External subsequent lending is associated with an 89.3% ((1.893-1)*100%) increase in the odds of a client defaulting when all other variables are held constant
CreditCard_yes	0.888	If a client has a credit card, the odds of defaulting reduces by 11.2% ((1-0.888)*100%), holding all other variables constant (CreditCard_yes is equal to 1 if the client has a credit card and 0 if he/she doesn't)
RevolvingCredit_yes	0.846	When all other variables are held constant, the odds of a client defaulting reduces by 15.4% ((1-0.846)*100%) if the client has revolving credit
Arrearrs_1	1.189	Arrears is associated with an 18.9% ((1.189 – 1) x 100%) increase in the odds of a client defaulting, holding all other variables constant (Arrearrs_1 is equal to 1 if the client was in arrears and 0 if he/she wasn't in arrears)

Once the significance of the variables in the model and the impact the variables have on the 'default' target variable is examined, the fitted logistic regression model's performance is analysed. Table 4.3 presents the confusion matrix for the fitted logistic regression model.

Table 4. 3: Confusion matrix for the logistic regression model

		PREDICTION		
ACTUAL CLASS		**Default**	**Not default**	**Total**
	Default	1265 (8.7%)	546 (3.8%)	1811 (12.5%)
	Not default	4735 (32.7%)	7956 (54.9%)	12691 (87.5%)
	Total	6000 (41.4%)	8502 (58.6%)	14502 (100%)

Table 4.3 shows that 1811 clients under study defaulted on their loan, whereas 12691 did not default. From Table 4.3 the fitted logistic regression model classified 1265 out of 1811 defaulters correctly; however, out of 6000 clients whom the model classified as defaulters, 4735 clients were actually non-defaulters (false positives). Thus, although the fitted logistic regression model identified a significant number of clients who defaulted correctly (1265 out of 1811), the model also misclassified many non-defaulters as defaulters. This can be attributed to the imbalance in the dataset. Table 4.3 also shows that 7956 out of 12691 non-defaulters were classified correctly; however, out of 8502 clients who were classified as non-defaulters, 546 clients were actually defaulters (false negatives). This indicates that although the model misclassified a significant number of non-defaulters, the majority of clients who were classified as non-defaulters were classified correctly. Overall, the fitted logistic regression model classified 9221 out of 14502 clients correctly, resulting in an accuracy score of 63.6%. As explained in Chapter 3, since the dataset is imbalanced and the class of interest is the minority class (i.e., default), the accuracy score may not be reliable when evaluating the model's performance. Table 4.4 shows several performance metrics the researcher explored to gain further insight into the fitted logistic regression model's performance.

Table 4. 4: Performance metrics for the logistic regression model

Performance metric	
Accuracy	0.636
Balanced accuracy	0.663
Sensitivity/True positive ratio/recall	0.699
Specificity/True negative ratio	0.627
Precision/Positive predictive value	0.211
Negative predictive value	0.936
AUC	0.720
Gini	0.440

From Table 4.4, a balanced accuracy score of 0.663 is reported, which is close to 70%, and therefore acceptable. Since the main focus of this study is to identify clients who default, the researcher then examines the true positive ratio to determine whether the model performs well when identifying defaulters. The true positive ratio (sensitivity) of 0.699 suggests that the model correctly classified approximately 70% of clients who defaulted. Thus, the model produces good results when identifying clients who default. Table 4.4 also shows a true negative ratio of 0.627, which indicates that the model correctly identified 62.7% of non-defaulters. This suggests that the model did not perform as well when identifying clients who did not default. However, since the cost of misclassifying non-defaulters as defaulters is very low, the researcher considers a true negative rate of about 65% and above acceptable. From Table 4.3, out of the clients who were classified as defaulters, a substantial portion of them did not default on their loan, whereas only a small portion of the clients who were classified as non-defaulters, did default on their loan; this is indicated by the low positive predictive value of 0.211 and high negative predictive value of 0.936. The low positive predictive value (precision) is presumably due to the imbalance in the dataset; the researcher focuses on the balanced accuracy, true positive ratio and true negative ratio when assessing the model's performance.

From the confusion matrix results in Table 4.3 and performance metrics in Table 4.4, the researcher concludes that although the logistic regression model did misclassify a considerable portion of non-defaulters, i.e., 37.3% (1-62.7%), the model was able to correctly identify a significant portion of defaulters (approximately 70%). Thus, the logistic regression model's performance is acceptable as the researcher's main focus is identifying clients who default, and the costs associated with misclassifying non-defaulters is low.

4.2 Decision tree

Figure 4.1 shows the structure of the decision tree used to predict the 'default' target variable under study.

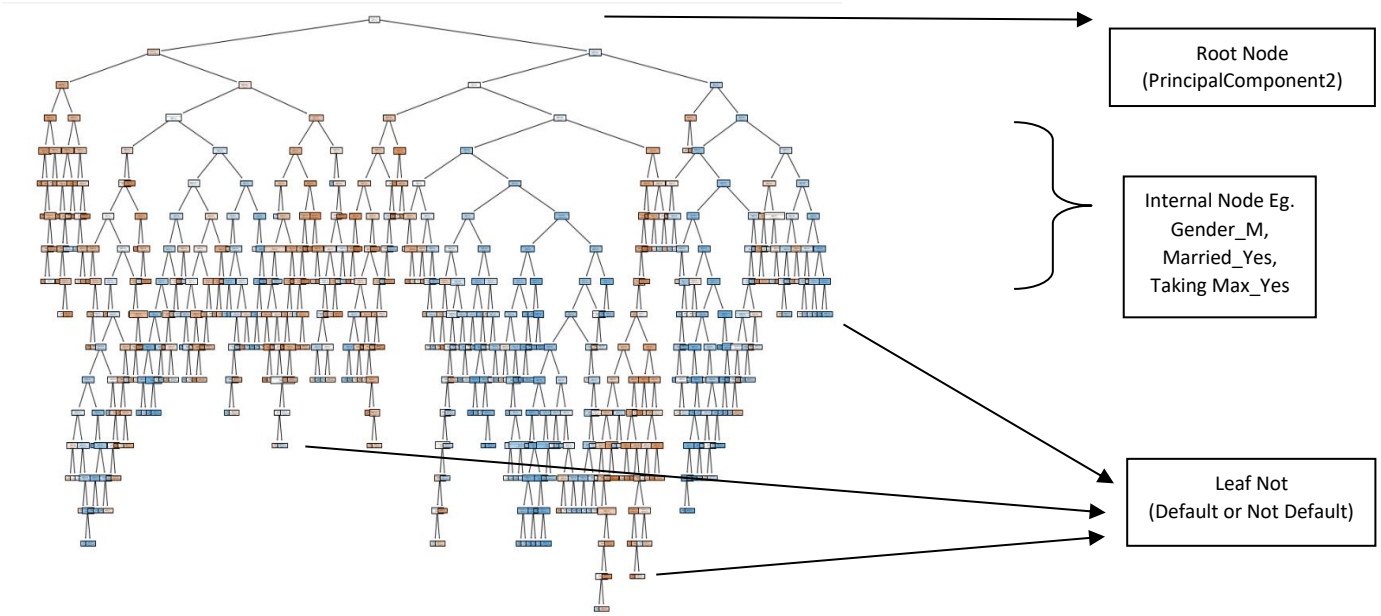

Figure 4. 1: Decision tree structure

From Figure 4.1, the test feature of the root node, which is the feature that best splits the data, is PrincipalComponent2. The internal nodes are all nodes found between the root node and the leaf node. Features such as Gender_M, Married_Yes, Taking Max_Yes et cetera, are tested at the internal nodes. Figure 4.1 also shows the leaf nodes that indicate which class the client is assigned to according to the model's prediction (i.e., default or not default).

To understand the importance of each feature in the decision tree, SHapley Additive exPlanations (SHAP) values are used. SHAP is a technique used in game theory to quantify how much each player in a team has contributed to the team's success. In machine learning, SHAP values measure the contribution each feature makes to the model's prediction, while taking into consideration the other features in the model. Every feature for every observation has a SHAP value. Figure 4.2 presents the mean absolute value of the SHAP values across all observations for each feature included in the decision tree, ranked in ascending order of importance.

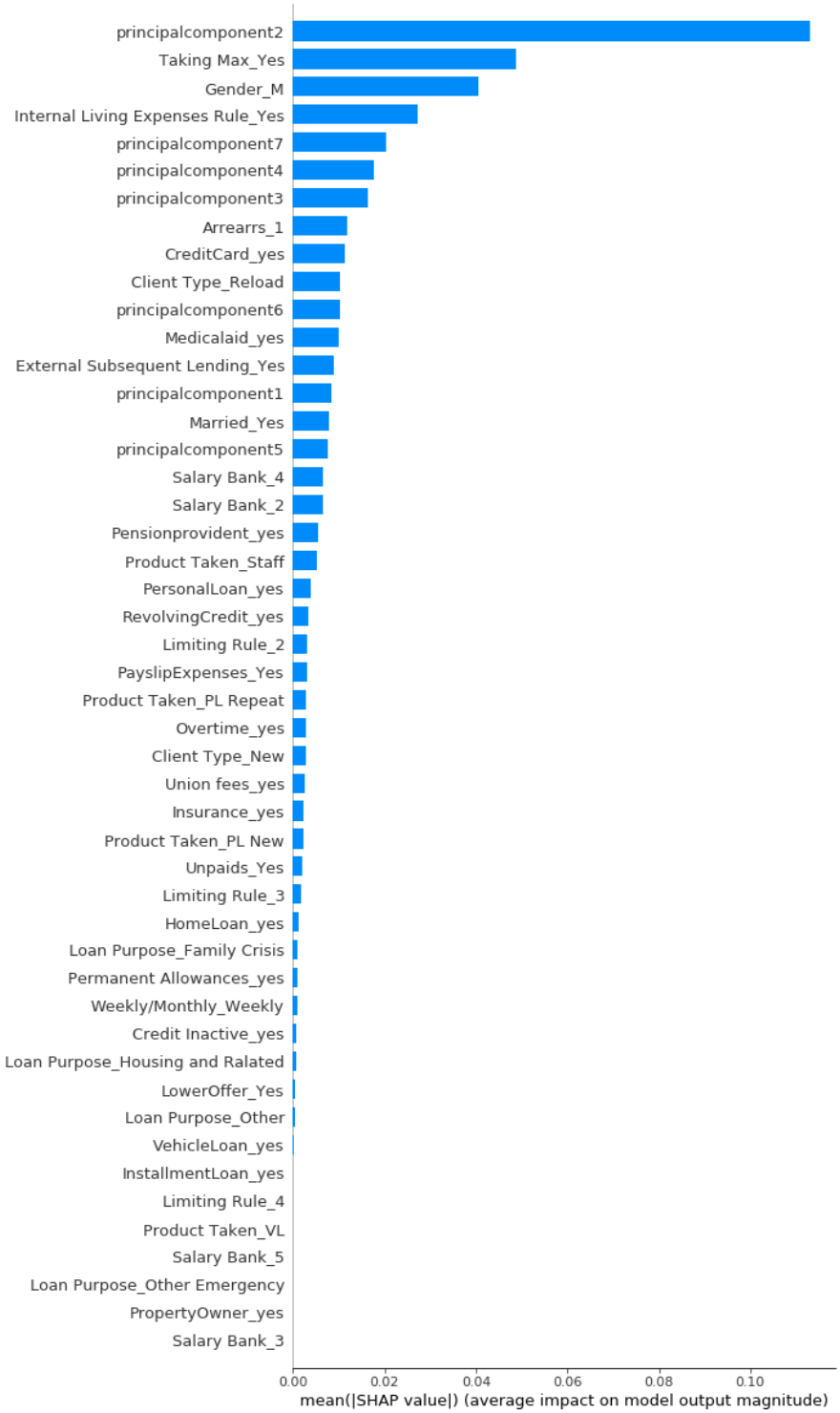

Figure 4. 2: Mean absolute SHAP value for each feature in the decision tree

Figure 4.2 shows that principalcomponent2 has the highest mean absolute SHAP value, which indicates that it contributes the most to the model's prediction. Taking Max_Yes is the second highest ranked feature, followed by Gender_M. A gradual decrease in mean absolute SHAP values for all remaining features in the model is then observed.

The researcher then examines the model's performance. Table 4.5 reports the confusion matrix which was obtained by fitting the decision tree algorithm to the balanced default dataset using the PCA approach.

Table 4. 5: Confusion matrix for the decision tree algorithm

		PREDICTION		
		Default	**Not default**	**Total**
ACTUAL CLASS	**Default**	1325 (8.7%)	486 (3.8%)	1811 (12.5%)
	Not default	5285 (32.7%)	7406 (54.9%)	12691 (87.5%)
	Total	6000 (41.4%)	8502 (58.6%)	14502 (100%)

The results in Table 4.5 indicate that the decision tree algorithm correctly classified 1325 out of 1811 defaulters and 7406 out of 12691 non-defaulters. This indicates that the decision tree seems to classify a significant number of defaulters correctly, however the model appears to misclassify a considerable number of non-defaulters. Overall, the model classified 8731 out of 14502 clients under study correctly, resulting in an accuracy score of 60.2% (8731/14502). Evaluation metrics in Table 4.6 are then explored in order to obtain a better understanding of the model's performance.

Table 4. 6: Performance metrics for the decision tree algorithm

Performance metric	
Accuracy	0.602
Balanced accuracy	0.658
Sensitivity/True positive ratio/Recall	0.732
Specificity/True negative ratio	0.584
Precision/Positive predictive value	0.200
Negative predictive value	0.938
AUC	0.705
Gini	0.409

Table 4.6 reports a balanced accuracy score of 0.658 which is noticeably higher than the accuracy score of 0.602, suggesting that the model performed significantly better when predicting the minority class (i.e., default). This is confirmed when observing the true positive ratio and true negative ratio of 0.732 and 0.584, respectively, in Table 4.6. The true positive ratio of 0.732 indicates that 73.2% of defaulters were correctly identified, whereas the true negative ratio of 0.58 indicates that the model only classified 58.4% of non-defaulters correctly. Thus, although the model performed well when identifying the default class, which is the focus of the study, the model seems unsuitable as it performed poorly when identifying non-defaulters.

4.3 Random forest

A random forest is an ensemble classification algorithm in which the base learners are decision trees. Figure 4.3 displays the structure of the first five decision trees in the random forest under study.

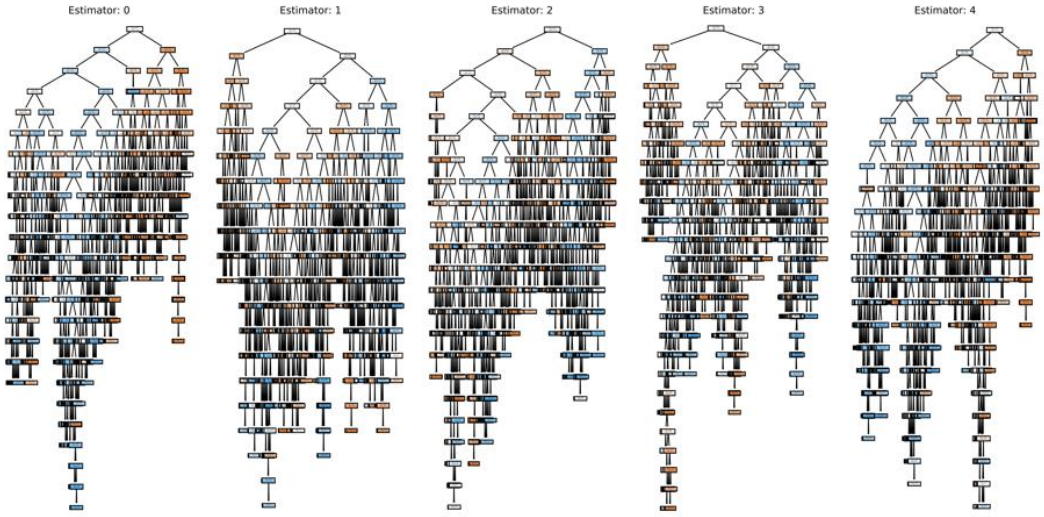

Figure 4. 3: Structure of the first five decision trees in the random forest

Figure 4.3 shows that the structures of the first five decision trees in the random forest are significantly different. Each tree is constructed independently and trained on a subset of features and subset of samples selected with replacement. There are 50 trees in the random forest under study. Observations are assigned to the class that receives the majority votes.

The importance of each feature in the random forest is examined by using SHAP values. Figure 4.4 shows the mean absolute value of the SHAP values across all observations, for each feature, ranked in ascending order.

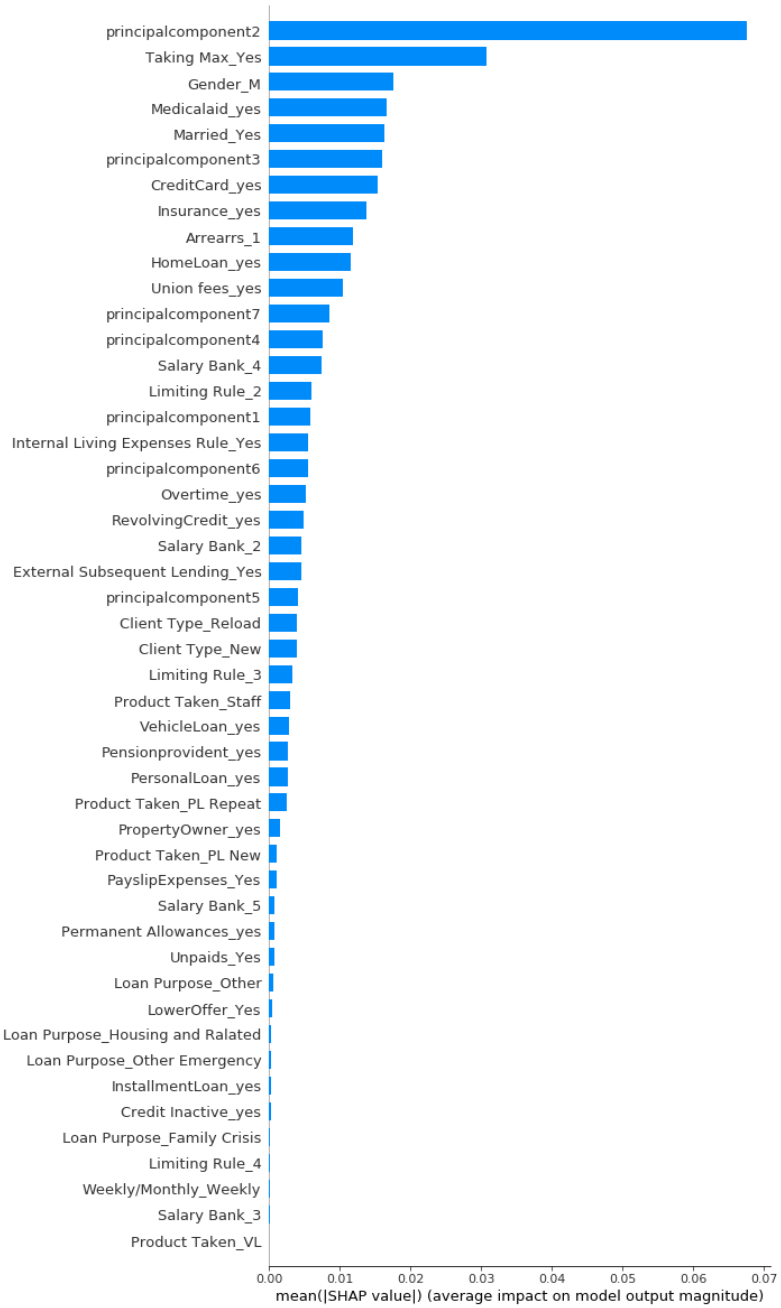

Figure 4. 4: Mean absolute SHAP value for each feature in the random forest

From Figure 4.4, PrincipalComponent2 ranked the highest which suggests that it contributed the most to the model's prediction. PrincipalComponent2 contributes significantly more than all other features in the model. The feature ranked second highest is Taking Max_yes followed by Gender_M. A gradual decrease in the mean absolute SHAP value for all remaining features is observed thereafter.

To gain further understanding of the importance of the features in the dataset and the impact they have on the 'default' target variable, the researcher examines Figure 4.5, which displays the SHAP values of every feature for every observation i.e., each dot on the plot represents the SHAP value associated with a feature for a particular observation. The vertical axis shows the features included in the model; the features are ranked in ascending order, from least important

to most important. The colour indicates whether the feature's value was large or small. Larger values of a feature are indicated by red dots, whereas smaller values are indicated by blue dots. The horizontal axis shows the SHAP values which indicate whether the effect of a value, for a particular feature, resulted in a higher or lower prediction. A positive SHAP value is associated with a higher prediction whilst a negative SHAP value is associated with a lower prediction, and when the SHAP value is approximately zero, the feature has negligible or no impact on the model itself. Since the default target variable is equal to 1 (higher prediction) when 'default' and 0 (lower prediction) when 'not default', a positive SHAP value is associated with 'default' and a negative value is associated with 'not default'.

Figure 4. 5: SHAP values of every feature for every observation in the random forest model

From Figure 4.5, Principalcomponent2 contributes the most to the model's prediction. Smaller Principalcomponent2 values, represented by the colour blue, increase the chance of defaulting, as indicated by the positive SHAP values, whereas large Principalcomponent2 values, represented by the colour red, decrease the chance of defaulting, as indicated by negative SHAP

values. The second most important feature is Taking Max_Yes. Large values of Taking Max_Yes imply that the client took the maximum amount offered, whereas small values imply that the client did not take the maximum amount offered as Taking Max_Yes is equal to 1 when the maximum amount offered is taken and 0 when the maximum amount offered is not taken. Figure 4.5 shows that high values of Taking Max_Yes (i.e., maximum amount taken) increase the chance of defaulting, as indicated by positive SHAP values, whereas low values of Taking Max_Yes (i.e., maximum amount not taken) decreases the chance of defaulting, as indicated by negative SHAP values. A similar method can be used to interpret the importance of all other features in Figure 4.5.

The results obtained from the fitted random forest algorithm are then analysed. Table 4.7 reports the confusion matrix for the random forest algorithm, which shows the number of true positives, true negatives, false positives, and false negatives.

Table 4. 7: Confusion matrix for the random forest algorithm

		PREDICTION					
		Default		**Not default**		**Total**	
ACTUAL CLASS	**Default**	1336	(9.2%)	475	(3.3%)	1811	(12.5%)
	Not default	4510	(31.1%)	8181	(56.4%)	12691	(87.5%)
	Total	5846	(40.3%)	8656	(59.7%)	14502	(100%)

The results in Table 4.7 show that the random forest model classified 1336 out of 1811 defaulters correctly and out of 5846 clients whom the model classified as defaulters, 4510 clients were actually non-defaulters (false positives). Therefore, although the random forest model identified a significant number of clients who defaulted correctly (1336 out of 1811), the model also misclassified many non-defaulters as defaulters. Given the nature of the dataset, the researcher expects a considerable number of false positives. From Table 4.7, 8181 out of 12691 non-defaulters were classified correctly, and out of 8656 clients whom the model classified as non-defaulters, only 475 clients were actually defaulters (false negatives). Thus, although the model misclassified a relatively large number of clients who did not default, most of the clients who were classified as non-defaulters, did not actually default on their loan. Overall, the random forest model classified 9517 out of 14502 clients correctly, resulting in an accuracy score of 65,6%. Table 4.8 summarises performance metrics that were explored to gain further insight into the random forest model's performance.

Table 4. 8: Performance metrics for the random forest algorithm

Performance metric	
Accuracy	0.656
Balanced accuracy	0.691
Sensitivity/True positive ratio/Recall	0.738
Specificity/True negative ratio	0.645
Precision/Positive predictive value	0.229
Negative predictive value	0.945
AUC	0.744
Gini	0.489

From Table 4.8, a balanced accuracy score of approximately 70% is reported which suggests that the model performed well overall. The balance accuracy score is greater than the accuracy score of 0.656; this indicates that the model correctly identified a larger portion of clients in the minority class i.e., the default class. The true positive ratio of 0.738 is then analysed; the model correctly identified approximately 74% of clients who defaulted. This suggests that the model performs very well when identifying clients who default. Table 4.8 shows a true negative ratio of 0.645, which indicates that the model correctly identified approximately 65% of clients who did not default. This implies that the model did not perform as well when identifying clients who did not default, however, the true negative ratio of 65% is still acceptable as misclassification costs related to false negatives is very low. From Table 4.8, a positive predictive value (precision) of 0.229 and negative predictive value of 0.945 is also reported. The low precision is presumably influenced by the imbalance in the dataset; the researcher concentrates on the balanced accuracy, true positive ratio and true negative ratio when evaluating the model's performance.

From the confusion matrix in Table 4.7 and the evaluation metrics in Table 4.8, the researcher concludes that the random forest model seems suitable for the classification problem under study as the model was able to correctly identify a significant portion of defaulters (approximately 74%), which is the class that the researcher is more interested in, and the model's performance was acceptable when classifying non-defaulters (i.e., a true negative ratio of approximately 65%)

4.4 Support vector machines

In classification problems, the support vector machine is used to separate the observations into classes, by identifying the optimal hyperplane with the largest margin. The SVM method can be extended to cater for cases where a nonlinear boundary exists by using a kernel function. In this study, the RBF kernel was used in order to separate the data and classify clients according to their default class. The SVM model was fitted to the balanced default dataset. Table 4.9 reports the confusion matrix results for the fitted SVM model.

Table 4. 9: Confusion matrix for the fitted SVM model

		PREDICTION					
		Default		**Not default**		**Total**	
ACTUAL CLASS	**Default**	1079	(7.4%)	731	(5.0%)	1811	(12.5%)
	Not default	3605	(24.9%)	9086	(62.7%)	12691	(87.5%)
	Total	4684	(32.3%)	9818	(67.7%)	14502	(100%)

The results in Table 4.9 show 1079 true positives and 731 false negatives, which indicates that the fitted SVM model seemed to misclassify a considerable portion of defaulters as non-defaulters. Table 4.9 also reports 9086 true negatives and 3605 false positives; this suggests that relative to the number of actual non-defaulters (i.e., 12691), the fitted SVM model classified a substantial number of them correctly (i.e. 9086). Overall, the model assigned 10165

clients to the correct class out of 14502 clients under study, resulting in an accuracy score of 0.701. Since our dataset is imbalanced, accuracy score is not the most reliable metric when examining a model's performance. We then analyse several evaluation metrics presented in Table 4.10.

Table 4. 10: Performance metrics for the fitted SVM model

Performance metric	
Accuracy	0.701
Balanced accuracy	0.656
Sensitivity/True positive ratio/Recall	0.596
Specificity/True negative ratio	0.716
Precision/Positive predictive value	0.230
Negative predictive value	0.925
AUC	0.720
Gini	0.440

From Table 4.10, the researcher discusses the balanced accuracy score, true positive ratio and true negative ratio. Table 4.10 reports a balanced accuracy score of 0.656 which is close to 70% and therefore acceptable. This table shows that the true positive ratio for the fitted SVM model is 0.596, which indicates that the model only correctly classified 59.6% of clients who defaulted, whereas the true negative ratio of 0.716 suggests that the model correctly classified 71.6% of non-defaulters. Thus, the fitted SVM model seems to perform better when predicting the non-default class. Overall, we conclude that although the SVM model performed well when classifying non-defaulters, the model does not seem fitting for the classification problem under study, as the SVM's ability to correctly identify defaulters, which is the main focus of this study, was below the acceptable level.

4.5 Naïve Bayes Classifier

The Naïve Bayes Classifier is a Bayesian network that is based on the Bayesian theorem, combined with the assumption of independence among features. It classifies data by identifying the class with the maximum posterior probability given a set of features. The Naïve Bayes Classifier was fitted to the balanced default dataset and the performance was then examined. The confusion matrix for the fitted Naïve Bayes classifier is presented in Table 4.11.

Table 4. 11: Confusion matrix for the fitted Naïve Bayes classifier

	PREDICTION					
ACTUAL CLASS		**Default**		**Not default**	**Total**	
	Default	1220	(8.4%)	591 (4.1%)	1811	(12.5%)
	Not default	4755	(32.8%)	7936 (54.7%)	12691	(87.5%)
	Total	5975	(41.2%)	8527 (58.8%)	14502	(100%)

From Table 4.11, 1220 true positives and 591 false negatives are reported. This suggests that a substantial portion of clients who defaulted were incorrectly classified as non-defaulters (591/1811). Thus, the model's ability to identify defaulters seems unsatisfactory. Table 4.11 shows 7936 true negatives and 4755 false positives; this indicates that the model also misclassified many non-defaulters. Overall, the model correctly classified 9156 clients out of the 14502 clients under study, resulting in an accuracy score of 63.1%. Evaluation metrics are then analysed in Table 4.12 to further understand the model's performance.

Table 4. 12: Performance metrics for the fitted Naïve Bayes classifier

Performance metric	
Accuracy	0.631
Balanced accuracy	0.649
Sensitivity/True positive ratio/Recall	0.674
Specificity/True negative ratio	0.625
Precision/Positive predictive value	0.204
Negative predictive value	0.931
AUC	0.703
Gini	0.406

From Table 4.12, the researcher discusses the true positive ratio and the true negative ratio. The true positive ratio of 0.674 does not meet the researcher's expectation as it suggests that a large portion of defaulters were misclassified (i.e., 32.6%); the cost associated with misclassifying defaulters is high. The true negative ratio of 0.625 in Table 4.12 is then analysed; the model did not perform very well when identifying clients who did not default, however, the true negative ratio is still acceptable as it is close to 65%. From the confusion matrix in Table 4.11 and the performance metrics in Table 4.12, we conclude that the Naïve Bayes classifier does not seem appropriate for the classification problem under study as it did not perform well when identifying both the default and non-default class.

4.6 K-nearest neighbours (K-NN)

K-NN is a distance-based classification algorithm that assigns a new observation to a class by calculating the distance between the new observation and the observations in the training set in order to find the new observation's k-nearest neighbours. The new observation is then classified based on the k-nearest neighbours by using the majority vote rule. In the application problem, the Manhattan distance formula was used to measure the distance between new observations and observations in the training set and 121 nearest neighbours were considered when classifying the new observations. The K-NN model was trained on the balanced dataset; Tables 4.13 and 4.14 summarises the results obtained by the model. The confusion matrix for the fitted K-NN model is presented in Table 4.13.

Table 4. 13: Confusion matrix for the K-NN model

		PREDICTION		
		Default	**Not default**	**Total**
ACTUAL CLASS	**Default**	1140 (7.9%)	671 (4.6%)	1811 (12.5%)
	Not default	4187 (28.9%)	8504 (58.6%)	12691 (87.5%)
	Total	5327 (36.7%)	9175 (63.3%)	14502 (100%)

From the results in Table 4.13, the fitted K-NN model correctly identified 1140 out of 1811 clients who defaulted. Thus, since there seems to be a significant number of defaulters who were misclassified as non-defaulters (i.e.,671), the K-NN model is likely unsuitable for the classification problem under study, as the main objective is to identify clients who defaulted. Table 4.13 shows that 8504 out of 12691 clients who did not default, were correctly classified, which suggests that a large number of clients who did not default were also misclassified (i.e., 4187). Overall, 9644 out of the 14502 clients under study were correctly classified, which results in an accuracy score of 66.5%. Table 4.14 displays several evaluation metrics that provide us with more clarity on the K-NN model's performance.

Table 4. 14: Performance metrics for the fitted K-NN model

Performance metric	
Accuracy	0.665
Balanced accuracy	0.650
Sensitivity/True positive ratio/Recall	0.629
Specificity/True negative ratio	0.670
Precision/Positive predictive value	0.214
Negative predictive value	0.927
AUC	0.701
Gini	0.403

From Table 4.14, the researcher discusses the balanced accuracy score, the true positive ratio and the true negative ratio. The balanced accuracy score of 0.65 is close to 70% and thus acceptable. The researcher then examines the true positive ratio to determine whether the model performs well when classifying defaulters. Table 4.14 shows that the fitted K-NN model attained a true positive ratio of 0.629; this indicates that the model correctly identified only 62.9% of clients who defaulted which is below the acceptable level of 70%. The true negative ratio shown in Table 4.14 is then analysed to gain insight into the model's performance when identifying non-defaulters; the true negative ratio of 0.67 is acceptable as misclassification costs associated with false positives is low.

Since the K-NN model performed poorly when identifying clients who defaulted, this model seems unsuitable for the classification problem under study, whose main focus is to identify clients who defaulted on their loan.

4.7 Artificial neural network

Artificial neural networks consist of multiple interconnected nodes and activation functions. Figure 4.6 represents the architecture of the artificial neural network in this study, when using the PCA approach. A feed-back neural network was used with two hidden layers.

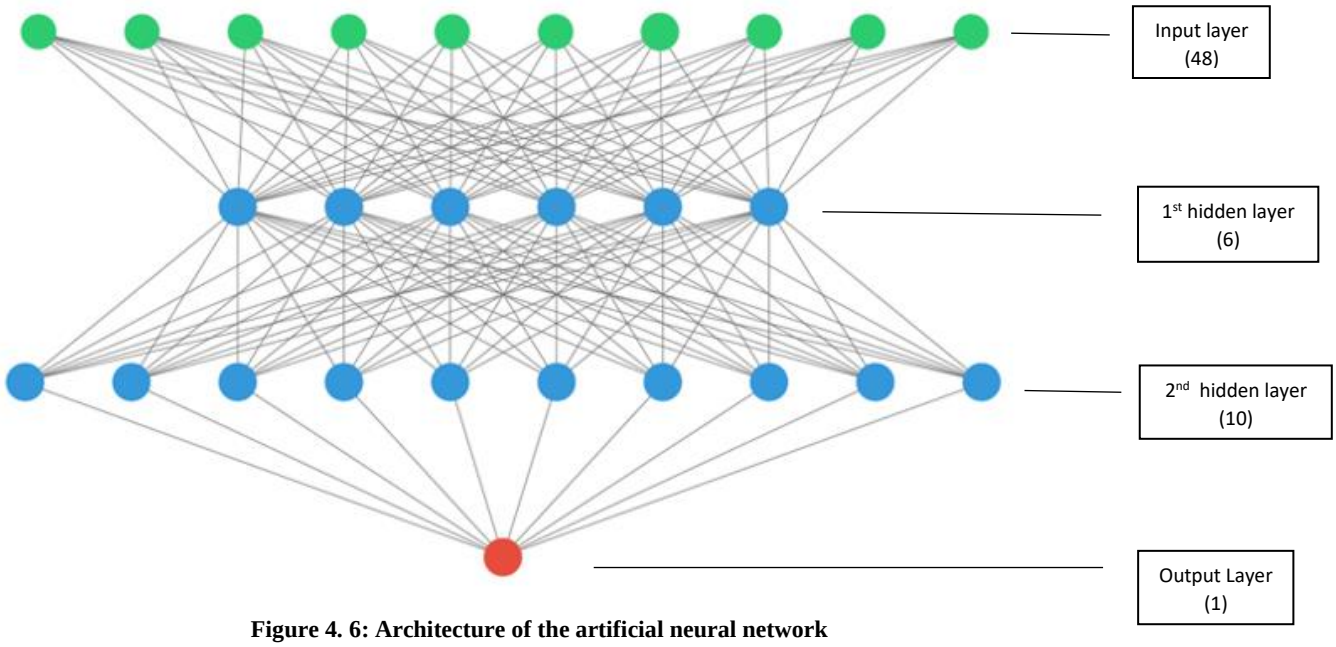

Figure 4. 6: Architecture of the artificial neural network

Figure 4.6 shows that the input layer comprises 48 nodes (i.e., one node for each feature); the first hidden layer includes six nodes, the second hidden layer includes ten nodes and the output layer includes one node that outputs the model's prediction (i.e., default or not default). The activation function used in the first and second hidden layer is the RELU function, whereas the activation function used in the output layer is the sigmoid function. The confusion matrix for the ANN model fitted to the balanced dataset is displayed in Table 4.15.

Table 4. 15: Confusion matrix for the fitted ANN model

		PREDICTION					
		Default		**Not default**		**Total**	
ACTUAL CLASS	**Default**	1338	(9.2%)	473	(3.3%)	1811	(12.5%)
	Not default	4597	(31.7%)	8094	(55.8%)	12691	(87.5%)
	Total	5935	(40.9%)	8567	(59.1%)	14502	(100%)

From Table 4.15, the results show that the ANN model correctly classified 1338 out of 1811 defaulters; however, out of 5935 clients whom the model classified as defaulters, 4597 clients did not default (false positives). Therefore, although the ANN model correctly identified a large portion of clients who defaulted, the model also misclassified many non-defaulters as defaulters. Given the nature of the dataset, a relatively large number of false positives can be

expected. From Table 4.15, 8094 out of 12691 non-defaulters were correctly classified and out of 8567 clients who were classified as non-defaulters, 473 clients were actually defaulters (false negatives). This suggests that although the model misclassified a substantial number of clients who did not default, most of the clients who were classified as non-defaulters, did not default. Overall, the ANN model correctly classified 9432 out of 14502 clients, resulting in an accuracy score of 65%. From Chapter 3, the researcher has learned that the accuracy score is not a reliable evaluation measure when the dataset is imbalanced and the class of interest is the minority class (i.e., default). Several performance metrics in Table 4.16 are then explored to further understand each model's performance.

Table 4. 16: Performance metrics for the ANN model

Performance metric	
Accuracy	0.650
Balanced accuracy	0.688
Sensitivity/True positive ratio/Recall	0.739
Specificity/True negative ratio	0.638
Precision/Positive predictive value	0.225
Negative predictive value	0.945
AUC	0.745
Gini	0.490

From Table 4.16, a balanced accuracy score of approximately 0.69 is reported. The balanced accuracy score is good, given the imbalance in the dataset and the difficulty in identifying whether a client will default at the time the loan is granted, when information is restricted. We then examine the true positive ratio (sensitivity) of 0.739; it suggests that the model correctly identified approximately 74% of clients who defaulted. Thus, the model performs well when identifying clients who default. Table 4.16 reports a true negative ratio of 0.638, which indicates that the model correctly identified approximately 64% of clients who did not default; although the score is acceptable given the classification problem under study, the model did not seem to perform as well when identifying clients who did not default. A positive predictive value (precision) and negative predictive value of 0.225 and 0.945, respectively, is also shown in Table 4.16. As previously stated, the low precision is presumably influenced by the imbalance in the dataset; therefore, the researcher focuses on the balanced accuracy, true positive ratio and true negative ratio when evaluating the model's performance.

The researcher therefore concludes that, although the model did misclassify a considerable portion of non-defaults, that is, 36.2% (1-63.8%), the model was able to correctly identify a significant portion of defaulters (approximately 74%), which is the class that the researcher is more interested in. Thus, the ANN model seems appropriate for the classification problem under study.

4.8 Summary of model performance using PCA

In Chapter 3, multicollinearity was identified and the researcher was also aware of the large number of features in the dataset. In this chapter, the researcher fitted the logistic regression model, decision tree, random forest, support vector machines, Naïve Bayes classifier, k-nearest neighbours, and artificial neural network to the balanced default dataset, corrected for multicollinearity by using the PCA approach, which is also a dimensionality reduction technique. The number of features in the dataset decreased from 57 to 48, and the multicollinearity problem was solved when using this approach. In this section of Chapter 4, the results obtained by these models are compared by using the confusion matrix and following evaluation metrics: accuracy, balanced accuracy, true positive ratio, true negative ratio, positive predictive value, negative predictive value, the AUC score and the Gini coefficient.

Table 4.17 displays the confusion matrix for each algorithm under study using the PCA approach

Table 4. 17: Confusion Matrix for each model under study using the PCA approach

			PREDICTED	
			DEFAULT	**NOT DEFAULT**
LR	**ACTUAL**	**DEFAULT**	1265	546
		NOT DEFAULT	4735	7956
DT		**DEFAULT**	1325	486
		NOT DEFAULT	5285	7406
RF		**DEFAULT**	1336	475
		NOT DEFAULT	4510	8181
SVM		**DEFAULT**	1079	731
		NOT DEFAULT	3605	9086
NB		**DEFAULT**	1220	591
		NOT DEFAULT	4755	7936
K-NN		**DEFAULT**	1140	671
		NOT DEFAULT	4187	8504
ANN		**DEFAULT**	1338	473
		NOT DEFAULT	4597	8094

From Table 4.17, the fitted ANN model attained the highest number of true positives, closely followed by the random forest model and then the decision tree algorithm; the number of defaulters correctly identified by these models were 1338, 1336 and 1325, respectively. This is an important factor to consider when selecting the most suitable model as the main aim of this study is to identify clients who default.

Evaluation metrics for each model under study using the PCA approach are reported in Table 4.18.

Table 4. 18: Evaluation metrics for each model under study using the PCA approach

	Logistic regression	Decision tree	Random forest	SVM	Naïve Bayes	K-NN	ANN
Accuracy	0.636	0.602	0.656	0.701	0.631	0.665	0.650
Balanced accuracy	0.663	0.658	0.691	0.656	0.649	0.650	0.688
True positive ratio (Sensitivity)	0.699	0.732	0.738	0.596	0.674	0.629	0.739
True negative ratio (Specificity)	0.627	0.584	0.645	0.716	0.625	0.670	0.638
Positive predictive value (Precision)	0.211	0.200	0.229	0.230	0.204	0.214	0.225
Negative predictive value	0.936	0.938	0.945	0.925	0.931	0.927	0.945
AUC score	0.720	0.705	0.744	0.720	0.703	0.701	0.745
Gini	0.440	0.409	0.489	0.440	0.406	0.403	0.490

From Table 4.18, the SVM model obtained the highest accuracy score (i.e., 0.701). In Chapter 3, the researcher ascertained that the dataset is imbalanced as 88% of clients under study did not default on their loan, whereas only 12% of the clients under study defaulted. Thus, since the accuracy score is biased toward the majority class (i.e., not default), and since the researcher's focus is on the minority class (i.e., default), accuracy is not a reliable metric for the imbalanced dataset under study. The balanced accuracy score provides a better indication of the model's overall performance in comparison to the accuracy score, as it takes into consideration the imbalance in the dataset. Table 4.18 shows that the random forest model obtained the highest balanced accuracy score of 0.691, which was marginally higher than the balanced accuracy score attained by the ANN model of 0.688; the researcher considers these to be good scores as they are both very close to 70%.

To understand each model's ability to predict the individual classes (i.e., default and not default), the researcher then examined the true positive ratio and the true negative ratio. The true positive ratio represents the percentage of clients who were correctly classified as defaulters of those who actually defaulted. It is one of the most important metrics in this study as the main aim of this study is to identify clients who default on their loan. The random forest and ANN model both obtained a true positive ratio of approximately 0.74, which was the highest true positive ratio obtained across the different models. In this study, a true positive ratio above 70% indicates that a model performs well when identifying clients who default. The researcher then analyses the true negative ratio; this represents the percentage of clients who were correctly classified as non-defaulters, of the actual non-defaulters. Ideally, the researcher would also want a good true negative ratio, however, since the cost associated with misclassifying a non-defaulter is very low for the classification problem under study, the researcher is not as concerned about misclassifying non-defaulters as defaulters; a true negative ratio of about 65% and above is considered acceptable. The SVM model obtained the highest true negative ratio of 0.716, however, the SVM attained the worst true positive ratio (i.e., 0.596). The random forest and ANN models, which achieved the best true positive ratios, obtained true negative ratios of approximately 0.65 and 0.64, respectively, which are both acceptable scores.

The positive predictive values and negative predictive values were then analysed. The positive predictive value, which represents the portion of clients who were correctly classified as

defaulters, of all clients who were classified as defaulters, was exceptionally low across all models; the researcher presumes that the large imbalance in the dataset had an influence on this score. As previously stated, since the costs associated with misclassifying non-defaulters is negligible, the low precision values are not a concern to the researcher. The random forest, SVM and ANN models all obtained a positive predictive value of approximately 0.23, which was the highest value obtained across all models. The negative predictive value was then analysed; it represents the portion of clients who were correctly classified as non-defaulters of all clients who were classified as non-defaulters. The negative predictive value was remarkably high across all models which indicates that the majority of clients who were classified as non-defaulters, did not default on their loan. The random forest and ANN models both attained the highest negative predictive value of 0.945. Lastly, the AUC score and Gini were examined; the random forest and ANN model both attained an AUC score of approximately 0.74 and a Gini of approximately 0.49 which were the highest values across all classifiers under study. The researcher considers these to be good scores given the nature of the classification problem under study.

Table 4.18 also reports the AUC score and Gini for each model; the random forest and ANN model both attained an AUC score of approximately 0.74 and a Gini of approximately 0.49, respectively, which were the highest values across all classifiers under study.

Since the large imbalance in the dataset presumably had an influence on the accuracy, positive predictive value and negative predictive value, the researcher focuses on the confusion matrix, balanced accuracy, true positive ratio, true negative ratio, AUC score and Gini coefficient when identifying the most appropriate classification algorithm for the problem under study. Overall, the random forest and ANN models seemed to achieve the best scores across most metrics. The SVM model attained the highest true negative ratio, however, it did not perform well when identifying clients who defaulted. Since the ANN model has a significantly longer training time, has several parameters that need to be tuned, and has little interpretability, the random forest model seems to be the most suitable model when predicting the default status of clients under study when using the PCA approach.

In this chapter, the dataset was corrected for multicollinearity by using the PCA approach which is also a dimensionality reduction technique; thereafter, several models were constructed. The importance and contribution of features across the various models were examined and the researcher noticed that numerous features were of little importance to the model. In Chapter 5, the aim is to build new models that only include relevant features by utilising a feature selection method.

Chapter 5

5 Classification with feature selection

Each classification algorithm discussed in Chapter 3 was fitted to the balanced default dataset using the feature selection approach (recursive feature elimination). In this chapter, the features selected using the recursive feature elimination technique are listed, and the confusion matrix and evaluation metrics, discussed in Chapter 2, are presented and examined for each model in order to identify the one which was most appropriate for the classification problem under study. A comparison is also made between the model which performed the best when using the PCA approach (from Chapter 4) and the feature selection approach in order to identify the best model overall.

5.1 Brief introduction to feature selection

Feature selection is a dimensionality reduction technique that aims to select a subset of features from the original set of features. This is achieved by removing features that are redundant, irrelevant, and noisy. Feature selection often leads to models with improved accuracy, lower computational costs, and improved model interpretability (Wang et al., 2016). This technique can also simplify models, make implementation easier and reduce the risk of data errors, since reducing the number of features reduces the risk of errors during data collection and storage. There are several feature selection methods that can be used to reduce the number of features in the model. The three main feature selection categories are the filter method, wrapper method, and embedded method (Venkatesh & Anuradha, 2019).

Filter methods do not consider the induction algorithm when selecting features (Kohavi & John, 1997); instead, they use characteristics of the features in order to select relevant features and are thus model agnostic, in other words, they can be used for any machine learning model. This method ranks features based on the scores computed for each feature that is independent of the induction algorithm. Either a predefined number of features with the highest scores are selected, or all features that attain a score above a certain threshold are included in the subset of selected features (Bommert et al., 2020). The chi-square test, variance threshold, Fisher score and correlation coefficient are examples of types of filter methods (Venkatesh & Anuradha, 2019). This method is generally less computationally expensive than wrapper methods and embedded methods; however, this method often result in lower prediction performance.

The wrapper method utilises a machine learning algorithm within its feature selection process. The feature selection algorithm exists as a wrapper around the selected machine learning algorithm (Kohavi & John, 1997). Wrapper methods are often referred to as greedy algorithms, as they try to find the best subset of features that will lead to the best performing model (Belete & Manjaiah, 2020). During the feature selection process, a subset of features is first chosen from the available features by employing a search strategy. There are different search strategies that can be used, for example, sequential forward feature selection and sequential backward

feature selection. Forward selection is an iterative method in which the model initially has no features in the subset. At each iteration, a single feature is added, one at a time. In backward elimination, all features are initially included in the model and one feature is removed at each iteration, one at a time (Panthong & Srivihok, 2015). Once the subset of features is selected, a chosen machine learning algorithm is trained on this subset and the model performance is evaluated by using a selected evaluation metric. This process is repeated by using a new subset of features, which depends on the search strategy selected, and continues until pre-defined stopping criteria is met. Once the stopping criteria is met, the best subset of features is chosen (Li et al., 2016). A disadvantage of the wrapper method is that it is computationally more complex compared to the filter and embedded methods (Venkatesh & Anuradha, 2019).

The embedded method is, as its name suggests, a feature selection method that is embedded in the algorithm, that is, feature selection is performed as part of the model-building process (Bommert et al., 2020). By combining feature selection with model building, the embedded method considers the interaction between the classification model and the features. When using an embedded method, a machine learning model is trained and feature importance is derived from the model for each feature; thereafter, the irrelevant features are removed. LASSO regularisation and tree-based methods such as CART, random forests and gradient boosting are examples of embedded methods (Bommert et al.). This method has the advantage of considering the classification algorithm in the process while being less computationally intensive compared to the wrapper method (Mwadulo, 2016).

The researcher then discusses feature selection methods that do not fall into a single category; instead, a combination of methods from different categories is used. These methods are known as hybrid methods which often aim to combine the advantages of different feature selection methods. The hybrid method used in this study is recursive feature elimination; this method, like the embedded method, uses the importance derived from a machine learning algorithm, and like the wrapper method, features are removed one at a time and the model performance is evaluated at each iteration. This method is often better than embedded methods and quicker than wrapper methods. Some studies, however, consider recursive feature elimination as a wrapper method and not a hybrid method.

To understand the recursive feature elimination technique, the researcher discusses and explains it by using the random forest algorithm. The random forest algorithm is initially trained using the original dataset with the full set of features. Thereafter, the feature importance values of all features used during the training process is derived by using, for example, tree-based models, lasso or logistic regression. In this study, a tree-based feature importance method was used for the random forest model. Figure 5.1 shows the feature importance of all features in the random forest model, ranked in ascending order (from the least important to the most important feature).

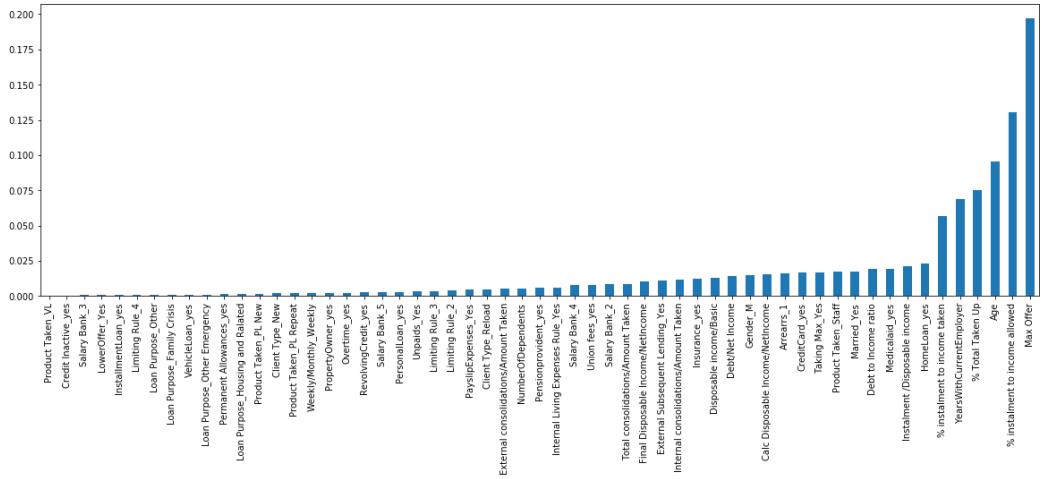

Figure 5. 1: Feature importance for all variables in the random forest model

From Figure 5.1, Max Offer is ranked the most important feature, followed by % instalment to income allowed. We also observe that several features are of very little importance to the model.

Features are then removed one by one, from the least important to the most important feature, as per the feature importance value associated with each feature. Figure 5.1 shows that Product_Taken_VL was ranked the least important feature in the random forest model. Therefore, Product_Taken_VL is the first feature to be removed; thereafter, a new random forest model is constructed by using the remaining features. An evaluation metric is then selected, and the performance of the new model is analysed and compared to the performance of the initial random forest model. In this study, the AUC score was the selected evaluation metric. Therefore, the change in AUC score between the models is compared to an arbitrary threshold and if the change is greater than the threshold, the feature will not be removed; if the change is less than the threshold, though, the feature will be removed. In this study, 0.001 was selected as the threshold for the random forest model. This process is repeated such that at each iteration, the next least-important feature is removed. Thus, in the second iteration, Credit Inactive_yes is removed. The process ends once all features have been 'tested'. The researcher then builds the final model, which only includes the features that were deemed important.

Table 5.1 lists each feature under study, shows the drop in ROC AUC score when the feature is removed from the random forest model, and indicates whether the feature should be removed from the model.

Table 5. 1: Recursive feature elimination results for the random forest model

Feature	Drop in ROC AUC score	Decision
Product Taken_VL	0.0006	Remove
Credit Inactive_yes	-0.0009	Remove
Salary Bank_3	0.0005	Remove
LowerOffer_Yes	0.0005	Remove
InstallmentLoan_yes	0.0007	Remove
Limiting Rule_4	0.0006	Remove
Loan Purpose_Other	-0.0003	Remove

Loan Purpose_Family Crisis	-0.0013	Remove
VehicleLoan_yes	0.0001	Remove
Loan Purpose_Other Emergency	-0.0004	Remove
Permanent Allowances_yes	-0.0009	Remove
Loan Purpose_Housing and Ralated	0.0008	Remove
Product Taken_PL New	0.0002	Remove
Client Type_New	0.0003	Remove
Product Taken_PL Repeat	0.0001	Remove
Weekly/Monthly_Weekly	0.0004	Remove
PropertyOwner_yes	-0.0003	Remove
Overtime_yes	-0.0001	Remove
RevolvingCredit_yes	0.0002	Remove
Salary Bank_5	-0.0002	Remove
PersonalLoan_yes	0.0002	Remove
Unpaids_Yes	0.0004	Remove
Limiting Rule_3	0.0003	Remove
Limiting Rule_2	0.0008	Remove
PayslipExpenses_Yes	0.0006	Remove
Client Type_Reload	0.0017	Keep
External consolidations/Amount Taken	0.0008	Remove
NumberOfDependents	0.0002	Remove
Pensionprovident_yes	0.0012	Keep
Internal Living Expenses Rule_Yes	0.0012	Keep
Salary Bank_4	0.0005	Remove
Union fees_yes	0.0019	Keep
Salary Bank_2	0.0009	Remove
Total consolidations/Amount Taken	0.0013	Keep
Final Disposable Income/NetIncome	0.0005	Remove
External Subsequent Lending_Yes	0.0001	Remove
Internal consolidations/Amount Taken	0.0007	Remove
Insurance_yes	0.0010	Keep
Disposable income/Basic	0.0002	Remove
Debt/Net Income	0.0020	Keep
Gender_M	0.0024	Keep
Calc Disposable Income/NetIncome	0.0007	Remove
Arrearrs_1	0.0007	Remove
CreditCard_yes	0.0016	Keep
Taking Max_Yes	0.0010	Keep
Product Taken_Staff	0.0008	Remove
Married_Yes	0.0025	Keep
Debt to Income ratio	0.0001	Remove
Medicalaid_yes	0.0021	Keep
Instalment /Disposable income	0.0005	Remove
HomeLoan_yes	0.0009	Remove
% instalment to income taken	0.0028	Keep
YearsWithCurrentEmployer	0.0032	Keep
% Total Taken Up	0.0038	Keep
Age	0.0044	Keep
% instalment to income allowed	0.0024	Keep
Max Offer	0.0033	Keep

From Table 5.1, we observe that 39 out of 57 features can be removed from the dataset when using the recursive feature elimination technique in combination with the random forest model, in which the AUC score was chosen as the evaluation metric and 0.001 was selected as the threshold.

The researcher uses a similar feature selection method for the logistic regression model, decision tree, support vector machines, Naïve Bayes classifier, k-nearest neighbours and the artificial neural network models. Sections 5.2 to 5.8 provide and compare results obtained by the model fitted to the full set of features and the model fitted to the selected features using the recursive feature elimination technique, for each classifier, in order to identify the better model.

5.2 Logistic regression

In this section, the recursive feature elimination technique outlined in section 5.1 was utilised in conjunction with the logistic regression model to select features deemed important. The following 24 features (out of 57) were selected as input variables: % instalment to income allowed, % instalment to income taken, Age, % Total Taken Up, Gender_M, CreditCard_yes, Debt/Net Income, External Subsequent Lending_Yes, Arrearrs_1, Client Type_Reload, Insurance_yes, Internal consolidations/Amount Taken, Internal Living Expenses Rule_Yes, Limiting Rule_4, Married_Yes, Max Offer, Medicalaid_yes, Product Taken_PL Repeat, Product Taken_Staff, RevolvingCredit_yes, Salary Bank_4, Taking Max_Yes, Union fees_yes, YearsWithCurrentEmployer.

Thereafter, two logistic regression models were trained, one using the full set of features and the other only the selected features. To compare the performance of these two models, the confusion matrix is analysed in Table 5.2.

Table 5. 2: Confusion matrix for the fitted logistic regression model using feature selection and the fitted model using the full set of features

			Predicted	
			Default	Not Default
Selected features	**Actual**	Default	1280	531
		Not Default	4784	7907
Full set of features		Default	1253	558
		Not Default	4772	7919

From Table 5.2, 1280 true positives and 7907 true negatives were reported for the fitted logistic regression model that used the feature selection approach, whereas 1253 true positives and 7919 true negatives were reported for the fitted model that used the full set of features. This indicates that the model using feature selection correctly predicted more clients who defaulted on their loan compared to the model that included the full set of features; the latter predicted more non-defaulters. We then examine several evaluation metrics (listed in Table 5.3) to ascertain which of these two models performed better overall.

Table 5. 3: Performance metrics for the logistic regression model using feature selection and the model using the full set of features

Performance metric	Feature selection	Full set of features
Accuracy	0.633	0.632
Balanced accuracy	0.665	0.658
Sensitivity/True positive ratio/Recall	0.707	0.692
Specificity/True negative ratio	0.623	0.624
Positive predictive value (Precision)	0.211	0.208
Negative predictive value	0.937	0.934
AUC	0.726	0.717
Gini	0.453	0.434

From Table 5.3, an accuracy score of approximately 0.63 was reported for both fitted logistic regression models. This table shows that the model using feature selection attained a better balanced accuracy score, true positive ratio, AUC score, and Gini in comparison to the model that utilised the full set of features. From Table 5.3, the true negative ratio obtained by both models was approximately 0.62. The researcher thus concludes that overall, the logistic regression model trained on selected features seemed to perform better than the model trained on the full set of features.

The researcher assesses the fit of the model that used feature selection, using the deviance test (discussed in Chapter 2) which is presented in Table 5.4; thereafter, the researcher analyses the maximum likelihood parameter estimates, p-values and odds ratios associated with each variable.

Table 5. 4: Deviance test for logistic regression model using feature selection

Deviance statistic (D)	dof (n-p)	Deviance statistic/dof
18689	14478	1.291

Table 5.4 displays the deviance statistic/dof of 1.291, where the deviance statistic is 18689 and the degrees of freedom (dof) are 14 478. Since $D/(n-p) = 1.291 < 1.5$, the researcher concludes that the model fits the data well.

The maximum likelihood parameter estimates and p-values for the fitted logistic regression model are presented in Table 5.5a.

Table 5. 5a: Maximum Likelihood Parameter estimates, and p-values for the fitted logistic regression model

	Parameter estimate	p-value
Intercept	0.749	<0.001***
% instalment to income allowed	- 0.253	<0.001***
% instalment to income taken	0.150	<0.001***
Age	- 0.203	<0.001***
% Total Taken Up	0.211	<0.001***
Gender_M	0.393	<0.001***

CreditCard_yes	- 0.167	0.014**
Debt/Net Income	0.114	0.035**
External Subsequent Lending_Yes	0.583	<0.001***
Arrearrs_1	0.133	0.006***
Client Type_Reload	- 0.264	<0.001***
Insurance_yes	- 0.139	0.002***
Internal consolidations/Amount Taken	- 0.152	0.021**
Internal Living Expenses Rule_Yes	0.129	<0.001***
Limiting Rule_4	0.380	0.008***
Married_Yes	- 0.161	0.001***
Max Offer	- 0.371	<0.001***
Medicalaid_yes	- 0.235	<0.001***
Product Taken_PL Repeat	0.116	<0.001***
Product Taken_Staff	- 1.052	<0.001***
RevolvingCredit_yes	- 0.145	0.009***
Salary Bank_4	- 0.263	<0.001***
Taking Max_Yes	0.451	<0.001***
Union fees_yes	- 0.111	0.001***
YearsWithCurrentEmployer	- 0.153	<0.001***

Note: ***, ** and * indicate significance at 1%, 5% and 10% level of significance respectively

From Table 5.5a, % instalment to income allowed, % instalment to income taken, Age, % Total Taken Up, Gender_M, External Subsequent Lending_Yes, Arrearrs_1, Client Type_Reload, Insurance_yes, Internal Living Expenses Rule_Yes, Limiting Rule_4, Married_Yes, Max Offer, Medicalaid_yes, Product Taken_PL Repeat, Product Taken_Staff, RevolvingCredit_yes, Salary Bank_4, Taking Max_Yes, YearsWithCurrentEmployer and Union fees_yes are significant at 1% level of significance and CreditCard_yes, Debt/Net Income and Internal consolidations/Amount Taken are significant at 5% level of significance Thus, all 24 variables influence the prediction of loan defaulting clients; the estimates indicate how much they influence it. The researcher exponentiated the maximum likelihood parameter estimates (β_i coefficients) to obtain the odds ratio estimates, as the odds ratios can be interpreted more easily. Table 5.5b lists the 24 significant variables in the fitted logistic regression model and provides an interpretation of the odds ratio estimates for each variable.

Table 5.5b: Interpretation of the odds ratio estimates for the 24 variables in the fitted logistic regression model using feature selection

Variable	Odds ratio estimates	Interpretation
% instalment to income allowed	0.776	One unit increase in % instalment to income allowed is associated with a 22.4% ((1 – 0.776) x 100%) reduction in the odds of a client defaulting when all other variables are held constant

% instalment to income taken	1.162	Increasing % instalment to income taken by one unit, increases the odds of a client defaulting on their loan by 16.2% ((1.162-1) x 100%) when all other variables are held constant
Age	0.816	Each unit increase in Age reduces the odds of a client defaulting by 18.4% ((1 – 0.816) x 100%), holding all other variables constant
% Total Taken Up	1.235	One unit increase in % Total Taken Up is associated with a 23.5% ((1.235-1)*100%) increase in the odds of a client defaulting, holding all other variables constant
Gender_M	1.482	Gender_M (male) is associated with a 48.2% ((1.482 – 1) x 100%) increase in the odds of a client defaulting when all other variables are held constant (Gender_M is equal to 1 if male and 0 if female)
CreditCard_yes	0.846	The odds of a client defaulting decreases by 15.4% ((1-0.846)*100%) if the client has a credit card, holding all other variables constant. (CreditCard_yes is equal to 1 if the client has a credit card and 0 if he/she does not)
Debt/Net Income	1.121	One unit increase in Debt/Net Income is associated with a 12.1% ((1.121-1)*100%) increase in the odds of a client defaulting, holding all other variables constant
External Subsequent Lending_Yes	1.791	External subsequent lending is associated with a 79.1% ((1.791-1)*100%) increase in the odds of a client defaulting when all other variables are held constant
Arrearrs_1	1.142	Arrears is associated with a 14.2% ((1.142 – 1) x 100%) increase in the odds of a client defaulting, holding all other variables constant (Arrearrs_1 is equal to 1 if the client was in arrears and 0 if he/she wasn't in arrears)
Client Type_Reload	0.768	There is a 23.2% ((1-0.768)*100%) decrease in the odds of a client defaulting when client type is reload, holding all other variables constant
Insurance_yes	0.870	If a client pays insurance, the odds of defaulting decreases by 13.0% ((1-0.870)*100%), when all other variables are held constant
Internal consolidations/Amount Taken	0.859	Increasing Internal consolidations/Amount Taken by one unit, decreases the odds of a client defaulting on their loan by 14.1% ((1-0.859) x 100%) when all other variables are held constant
Internal Living Expenses Rule_Yes	1.138	The odds of a client defaulting increases by 13.8% ((1.138-1)*100%) if the internal living expenses rule is used, when holding all other variables constant
Limiting Rule_4	1.462	If a client is limiting by rule 4, the odds of defaulting increases by 46.2% ((1.462-1)*100%), when all other variables are held constant
Married_Yes	0.851	Married is associated with a 14.9% ((1-0.851)*100%) decrease in the odds of a client defaulting when all other variables are held constant (Married_Yes is equal to 1 if married and 0 if not married)

Max Offer	0.690	One unit increase in Max Offer is associated with a 31.0% ((1 – 0.690) x 100%) reduction in the odds of a client defaulting, holding all other variables constant
Medicalaid_yes	0.791	When all other variables are held constant, the odds of a client defaulting reduces by 20.9% ((1-0.791)*100%) if the client has medical aid
Product Taken_PL Repeat	1.123	If product taken is PL Repeat, the odds of a client defaulting increases by 12.3% ((1.123-1)*100%), when all other variables are held constant
Product Taken_Staff	0.349	If product taken is Staff, the odds of a client defaulting reduces by 65.1% ((1-0.349)*100%), holding all other variables constant
RevolvingCredit_yes	0.865	When all other variables are held constant, the odds of a client defaulting reduces by 13.5% ((1-0.865)*100%) if the client has revolving credit
Salary Bank_4	0.769	The odds of a client defaulting reduces by 23.1% ((1-0.769)*100%), if the client's main bank is Salary Bank 4, holding all other variables constant
Taking Max_Yes	1.570	Taking Max is associated with a 57.0% ((1.570 – 1) x 100%) increase in the odds of a client defaulting when all other variables are held constant
Union fees_yes	0.895	The odds of a client defaulting reduces by 10.5% ((1-0.895)*100%) if the client pays union fees, holding all other variables constant (Union fees_yes is equal to 1 if the client pays union fees and 0 if he/she does not)
YearsWithCurrentEmployer	0.858	One unit increase in YearsWithCurrentEmployer is associated with a 14.2% ((1 – 0.858) x 100%) reduction in the odds of a client defaulting when all other variables are held constant

5.3 Decision tree

The recursive feature elimination technique was used in combination with the decision tree algorithm in order to remove features that contributed little to the model. 27 out of 57 features were removed; the 30 features included in the decision tree when using the feature selection approach are % instalment to income allowed, % instalment to income taken, % Total Taken Up, Age, Arrearrs_1, Client Type_Reload, CreditCard_yes, Debt/Net Income, External consolidations/Amount Taken, External Subsequent Lending_Yes, Final Disposable Income/NetIncome, Gender_M, Internal consolidations/Amount Taken, Internal Living Expenses Rule_Yes, Limiting Rule_2, Debt to Income ratio, Limiting Rule_3, Married_Yes, Max Offer, Medicalaid_yes, NumberOfDependents, Product Taken_PL Repeat, Pensionprovident_yes, PersonalLoan_yes, Product Taken_PL New, HomeLoan_yes, Product Taken_Staff, Taking Max_Yes, Unpaids_Yes and YearsWithCurrentEmployer.

The confusion matrix for the decision tree trained on selected features and the decision tree trained on the full set of features is reported in Table 5.6.

Table 5. 6: Confusion matrix for the decision tree using feature selection and the decision tree using the full set of features

			Predicted	
			Default	Not Default
Selected features	**Actual**	Default	1310	501
		Not Default	5267	7424
Full set of features		Default	1210	601
		Not Default	5064	7627

Table 5.6 shows that the decision tree trained on selected features correctly predicted 1310 clients who defaulted and 7427 clients who did not default, whereas the decision tree trained on the full set of features correctly predicted 1210 defaulters and 7627 non-defaulters. Thus, the decision tree using feature selection performed better when predicting the default class and the decision tree using all features under consideration performed slightly better when predicting the non-default class. Performance metrics for both decision trees are shown in Table 5.7 and the results are analysed to gain further insight into each model's performance.

Table 5. 7: Performance metrics for the decision tree using feature selection and the decision tree using the full set of features

Performance metric	**Feature selection**	**Full set of features**
Accuracy	0.602	0.609
Balanced accuracy	0.654	0.635
Sensitivity/True positive ratio/Recall	0.723	0.668
Specificity/True negative ratio	0.585	0.601
Positive predictive value (Precision)	0.199	0.193
Negative predictive value	0.937	0.927
AUC	0.698	0.687
Gini	0.396	0.375

Table 5.7 indicates that the accuracy score for the decision tree using feature selection is 0.602, which is marginally lower than the accuracy score of 0.609 obtained by the decision tree using all features. Since the dataset is imbalanced, accuracy is not the most reliable evaluation metric. From Table 5.7, the balanced accuracy score, true positive ratio, AUC score and Gini attained by the decision tree that used selected features showed better results compared to the decision tree trained on the full set of features in the dataset. The true negative ratio of 0.585 obtained by the decision tree using feature selection was lower than the true negative ratio score of 0.601 obtained by the decision tree that utilised all features. Since the model using feature selection obtained better scores for all metrics other than the true negative ratio (which was marginally lower than the score obtained by the model using the full set of features), it can be concluded that this decision tree seems more appropriate for the classification problem under study, whose main focus is to identify clients who defaulted on their loan.

5.4 Random forest

In section 5.1, the researcher discussed in detail the recursive feature elimination technique using the random forest algorithm and indicated that 39 features could be removed. The following 18 features were selected using this approach: % instalment to income allowed, % instalment to income taken, % Total Taken Up, Age, Client Type_Reload, CreditCard_yes, Debt/Net Income, Gender_M, Insurance_yes, Internal Living Expenses Rule_Yes, Married_Yes, Max Offer, Medicalaid_yes, Pensionprovident_yes, Taking Max_Yes, Total consolidations/Amount Taken, YearsWithCurrentEmployer and Union fees_yes.

One random forest model was trained on the full set of features and another on only the selected features. In order to establish how many true positives and true negatives were attained by these two models, the confusion matrix in Table 5.8 is examined.

Table 5. 8: Confusion matrix for the random forest model using feature selection and the model using the full set of features

			Predicted	
			Default	Not Default
Selected features	**Actual**	Default	1332	479
		Not Default	4501	8190
Full set of features		Default	1317	494
		Not Default	4391	8300

Table 5.8 reports 1332 true positives and 8190 true negatives for the random forest model using the recursive feature elimination method, whereas 1317 true positives and 8300 true negatives are reported for the model using the full set of features. This indicates that the model using only selected features correctly predicted more clients who defaulted on their loan compared to the model that used the full set of features; the latter predicted more non-defaulters. In order to determine which model performed better overall, evaluation metrics (listed in Table 5.9) are examined.

Table 5. 9: Performance metrics for the random forest using selected features and the model using the full set of features

Performance metric	**Feature selection**	**Full set of features**
Accuracy	0.657	0.663
Balanced accuracy	0.690	0.691
Sensitivity/True positive ratio/Recall	0.736	0.727
Specificity/True negative ratio	0.645	0.654
Positive predictive value (Precision)	0.228	0.231
Negative predictive value	0.945	0.944
AUC	0.745	0.744
Gini	0.491	0.488

From Table 5.9, there was no significant difference between the accuracy score and balanced accuracy score obtained by the two random forest models. Table 5.9 shows that the model using feature selection obtained a higher true positive ratio (i.e., 0.736) compared to the model that included all features under consideration (i.e., 0.727). The true negative ratio of 0.645 attained by the model with only selected features was marginally lower than the true negative ratio of 0.654 obtained by the model that used the full set of features. It is not a concern, as the researcher is more interested in identifying clients who default. From this table, the AUC score and Gini coefficient of 0.745 and 0.491, respectively, which were attained by the model using feature selection, were both slightly higher than the AUC score and Gini coefficient of 0.744 and 0.488, respectively, obtained by the model that included all features. Therefore, from Tables 5.8 and 5.9, the researcher concludes that the random forest model trained on 18 features seems to be the more favourable model.

5.5 Support vector machine

Recursive feature elimination was utilised in order to select important features to include in the SVM model's building process; the following 25 features were chosen using this approach: % instalment to income allowed, % instalment to income taken, % Total Taken Up, Age, Arrearrs_1, Client Type_New, CreditCard_yes, Debt/Net Income, Gender_M, Insurance_yes, Internal consolidations/Amount Taken, Internal Living Expenses Rule_Yes, Limiting Rule_2, Married_Yes, Max Offer, Medicalaid_yes, NumberOfDependents, Pensionprovident_yes, Union fees_yes, Product Taken_PL New, Product Taken_PL Repeat, RevolvingCredit_yes, Taking Max_Yes, Unpaids_Yes and YearsWithCurrentEmployer.

Two SVM models were trained thereafter, one using the full set of features and the other only the selected features. A confusion matrix for these two models is reported in Table 5.10.

Table 5. 10: Confusion matrix for the SVM model using feature selection and the model using the full set of features

			Predicted	
			Default	Not Default
Selected features	**Actual**	Default	1269	542
		Not Default	4431	8260
Full set of features		Default	1103	708
		Not Default	3655	9036

From Table 5.10, the SVM model using the feature selection approach correctly predicted 1269 defaulters and 8260 non-defaulters, whereas the model using all features under consideration correctly predicted 1103 defaulters and 9036 non-defaulters. Thus, the model using selected features seems to perform better when predicting the default class. However, this model misclassified more non-defaulters than the model trained on the full set of features. The researcher then examines the evaluation metrics (listed in Table 5.11) to ascertain which of these two models performed better overall.

Table 5. 11: Performance metrics for the SVM model using feature selection and the SVM model using the full set of features

Performance metric	Feature selection	Full set of features
Accuracy	0.657	0.699
Balanced accuracy	0.676	0.661
Sensitivity/True positive ratio/Recall	0.701	0.609
Specificity/True negative ratio	0.651	0.712
Positive predictive value (Precision)	0.223	0.232
Negative predictive value	0.938	0.927
AUC	0.733	0.720
Gini	0.466	0.440

From Table 5.11, the balanced accuracy score for the SVM model using feature selection is 0.676, which is higher than the score of 0.661 obtained by the model using all features. Table 5.11 also shows that the true positive and true negative ratio for the SVM model trained on selected features are 0.701 and 0.651, respectively, and for the SVM model using the full set of features, the ratios are 0.609 and 0.712, respectively. Thus, the model using the feature selection approach performed significantly better when identifying defaulters, which is the study's main focus; however, it performed worse when identifying non-defaulters. From Table 5.11, it can be observed that the AUC score and Gini obtained by the model using selected features are 0.733 and 0.466, respectively; these values are higher than the AUC score and Gini obtained by the model using all features, namely 0.720 and 0.440, respectively. Therefore, from Tables 5.10 and 5.11, the researcher concludes that the SVM model trained on the 25 selected features appears to be the better model for the application problem under study.

5.6 Naïve Bayes classifier

The Naïve Bayes classifier, in conjunction with the feature selection technique discussed in section 5.1, was used in order to select features which were most relevant. The model using the feature selection approach was trained on the following 19 features: % instalment to income allowed, % instalment to income taken, % Total Taken Up, Age, Arrearrs_1, External Subsequent Lending_Yes, Gender_M, Limiting Rule_2, Married_Yes, Max Offer, PayslipExpenses_Yes, Product Taken_Staff, RevolvingCredit_yes, Salary Bank_2, Union fees_yes, Unpaids_Yes, Taking Max_Yes, YearsWithCurrentEmployer and Insurance_yes.

The confusion matrix results for the Naïve Bayes classifier trained on selected features only and the Naïve Bayes classifier trained on the full set of features are reported in Table 5.12.

Table 5. 12: Confusion matrix for the Naïve Bayes classifier using feature selection, and the model using the full set of features

			Predicted	
			Default	Not Default
Selected features	**Actual**	Default	1012	799
		Not Default	3103	9588
Full set of features		Default	1186	625
		Not Default	4658	8033

From Table 5.12, the Naïve Bayes classifier using the feature selection approach correctly classified 1012 defaulters and 9588 non-defaulters, whereas the model using all features under consideration correctly classified 1186 defaulters and 8033 non-defaulters. Thus, the Naïve Bayes classifier using feature selection performed significantly better when identifying non-defaulters; however, the classifier using the full set of features correctly identified more defaulters.

A summary of the performance metrics for the Naïve Bayes classifier that used selected features and the one that used the full set of features are presented in Table 5.13.

Table 5. 13: Performance metrics for the Naïve Bayes classifier using feature selection, and using the full set of features

Performance metric	**Feature selection**	**Full set of features**
Accuracy	0.731	0.636
Balanced accuracy	0.657	0.644
Sensitivity/True positive ratio/Recall	0.559	0.655
Specificity/True negative ratio	0.755	0.633
Positive predictive value (Precision)	0.246	0.203
Negative predictive value	0.923	0.928
AUC	0.730	0.700
Gini	0.460	0.400

From Table 5.13, all reported performance metrics (other than the true positive ratio) showed better results for the Naïve Bayes classifier trained on selected features in comparison to the classifier trained on all features. The true positive ratio attained by the Naïve Bayes classifier using feature selection was 0.559, whereas the true positive ratio attained by the classifier that included the full set of features was significantly higher at 0.655. Although the classifier using feature selection obtained a better score for all other performance metrics except for the true positive ratio, it is not necessarily the better suited model, since the study's main focus is to identify clients who default. From Table 5.13, the overall performance of the Naïve Bayes classifier that used the full set of features in the model building process seems unsatisfactory. Since the Naïve Bayes classifier that used only selected features performed poorly when identifying clients who defaulted, this model is also unsuitable.

5.7 K-nearest neighbours

The recursive feature elimination method, in combination with the K-NN model, was used to identify important features in the model. The following 28 features (out of 57) were deemed important: % instalment to income allowed, % instalment to income taken, % Total Taken Up, Age, Arrearrs_1, Client Type_Reload, CreditCard_yes, External Subsequent Lending_Yes, Gender_M, HomeLoan_yes, Insurance_yes, Internal consolidations/Amount Taken, Internal Living Expenses Rule_Yes, Limiting Rule_3, LowerOffer_Yes, Married_Yes, Max Offer, Product Taken_PL Repeat, Medicalaid_yes, NumberOfDependents, PayslipExpenses_Yes, Salary Bank_2, RevolvingCredit_yes, Taking Max_Yes, Unpaids_Yes, Union fees_yes, Weekly/Monthly_Weekly and YearsWithCurrentEmployer.

Two K-NN models were trained thereafter, one using the full set of features and the other only the selected features. The confusion matrix results for these two models are presented in Table 5.14.

Table 5. 14: Confusion matrix for the K-NN model using selected features and the model using the full set of features

			Predicted	
			Default	Not Default
Selected features	**Actual**	Default	1163	648
		Not Default	4114	8577
Full set of features		Default	1192	619
		Not Default	4493	8198

From Table 5.14, the number of true positives attained by the K-NN model using the feature selection approach and the model using the full set of features is 1163 and 1192, respectively. This suggests that the model using feature selection misclassified a few more defaulters compared to the model trained on all features under consideration. Table 5.14 also shows that the number of true negatives obtained by the K-NN model using feature selection and the model using the full set of features is 8577 and 8192, respectively, indicating that the model using feature selection correctly predicted significantly more non-defaulters compared to the model using all features. The researcher then examines Table 5.15, which shows a summary of performance metrics for both models.

Table 5. 15: Performance metrics for the K-NN model using feature selection and the model using the full set of features

Performance metric	**Feature selection**	**Full set of features**
Accuracy	0.672	0.647
Balanced accuracy	0.659	0.652
Sensitivity/True positive ratio/Recall	0.642	0.658
Specificity/True negative ratio	0.676	0.646
Positive predictive value (Precision)	0.220	0.210
Negative predictive value	0.930	0.930
AUC	0.711	0.702
Gini	0.422	0.404

From Table 5.15, it can be observed that the K-NN model using feature selection obtained a true positive ratio of 0.642, whereas the K-NN model using the full set of features obtained a slightly higher true positive ratio of 0.658. Table 5.15 also reports that the true negative ratio attained by the K-NN model trained on selected features is 0.676, which is higher than the true negative ratio of 0.646 that was attained by *the* model trained on the full set of features. All other evaluation metrics shown in Table 5.15 indicate better performance for the K-NN model that only included selected features. Thus, from Tables 5.14 and 5.15, the K-NN model with 29 fewer features appears to be the more appropriate model for the classification problem under study.

5.8 Artificial neural network

The feature selection technique discussed in Chapter 5.1 was utilised in order to remove features which contributed little to the model. The following 24 features (out of 57) were chosen as input variables for the ANN model: % instalment to income allowed, % instalment to income taken, % Total Taken Up, Age, Arrears_1, Client Type_Reload, CreditCard_yes, Debt/Net Income, Gender_M, Insurance_yes, Internal Living Expenses Rule_yes, Limiting Rule_2, Married_yes, Max Offer, Medicalaid_yes, PayslipExpenses_Yes, Taking Max_yes, Pensionprovident_yes, Product Taken_PL Repeat, Salary Bank_2, Union fees_yes, Unpaids_yes, Total consolidations/Amount Taken and YearsWithCurrentEmployer,.

The confusion matrix results for the ANN model trained using feature selection and the model trained on the full set of features are reported in Table 5.16.

Table 5. 16: Confusion matrix for the ANN model using selected features and the model using the full set of features

			Predicted	
			Default	Not Default
Selected features	**Actual**	Default	1308	503
		Not Default	4403	8288
Full set of features		Default	1346	465
		Not Default	4848	7843

From Table 5.16, the ANN model using selected features correctly classified 1308 out of 1811 defaulters, whereas the model using all features under consideration correctly classified 1 346 defaulters. This indicates that the model using feature selection misclassified a few more defaulters compared to the model using all features. Table 5.16 also indicates that the ANN model that used selected features correctly identified 8288 out of 12691 non-defaulters and the model using the full set of features correctly identified 7843 non-defaulters. This suggests that the model using feature selection correctly identified a significantly larger portion of non-defaulters compared to the model trained on the full set of features.

In order to ascertain which model performed better overall, evaluation metrics (listed in Table 5.17) are examined.

Table 5. 17: Performance metrics for the ANN model using selected features and the model using the full set of features

Performance metric	Feature selection	Full set of features
Accuracy	0.662	0.634
Balanced accuracy	0.688	0.681
Sensitivity/True positive ratio/Recall	0.722	0.743
Specificity/True negative ratio	0.653	0.618
Positive predictive value (Precision)	0.229	0.217
Negative predictive value	0.943	0.944
AUC	0.745	0.737
Gini	0.490	0.475

From Table 5.17, the ANN model trained on selected features obtained better results for all evaluation metrics (other than the true positive ratio) in comparison to the ANN model that included the full set of features in the model-building process. The true positive ratio obtained by the model using only selected features was 0.722, whereas the true positive ratio attained by the model including all features was 0.743. Although the ANN model using feature selection attained a lower true positive ratio, the true positive ratio of 0.722 is still very good. Thus, the ANN model that included 24 features is more favourable.

5.9 Summary of model performance using feature selection

In this chapter, recursive feature elimination was utilised in order to select a subset of the most relevant features in the dataset for each machine learning algorithm outlined in Chapter 3. Each algorithm was trained twice, first using the full set of features and then using only the selected features, and a comparison was done between the two models for each algorithm. From sections 5.2 to 5.8, the researcher observed that the machine learning algorithms that utilised the recursive feature elimination technique generally seemed more suitable for the classification problem under study compared to the models that utilised the full set of features. In section 5.9, a comparison of results obtained by each classification algorithm (discussed in Chapter 3), when using the feature selection approach, is performed. Table 5.18 shows the confusion matrix for each algorithm and Table 5.19 provides a summary of several performance metrics used to evaluate the models, namely accuracy, balanced accuracy, true positive ratio, true negative ratio, positive predictive value (precision), negative predictive value, AUC score, and the Gini coefficient.

Table 5. 18: Confusion matrix for each model under study using feature selection

			PREDICTED	
			DEFAULT	**NOT DEFAULT**
LR	**ACTUAL**	**DEFAULT**	1280	531
		NOT DEFAULT	4784	7907
DT		**DEFAULT**	1310	501
		NOT DEFAULT	5267	7424
RF		**DEFAULT**	1332	479
		NOT DEFAULT	4501	8190
SVM		**DEFAULT**	1269	542
		NOT DEFAULT	4431	8260
NB		**DEFAULT**	1012	799
		NOT DEFAULT	3103	9588
K-NN		**DEFAULT**	1163	648
		NOT DEFAULT	4114	8577
ANN		**DEFAULT**	1308	503
		NOT DEFAULT	4403	8288

From Table 5.18, the random forest model attained the highest number of true positives, followed by the decision tree algorithm and then the ANN model; the number of defaulters correctly identified by these models were 1332, 1310 and 1308, respectively. Since the main aim of this study is to identify clients who default, the number of defaulters correctly identified by each algorithm is a significant factor when selecting the most suitable model. To compare the overall performance of the logistic regression model, decision tree, random forest, support vector machine, Naïve Bayes classifier, k-nearest neighbours algorithm and the artificial neural network when using the feature selection approach, Table 5.19 is examined.

Table 5. 19: Performance metrics for each model under study using the feature selection approach

	Logistic regression	**Decision tree**	**Random forest**	**SVM**	**Naïve Bayes**	**K-NN**	**ANN**
Accuracy	0.633	0.602	0.657	0.657	0.731	0.672	0.662
Balanced accuracy	0.665	0.654	0.69	0.676	0.657	0.659	0.688
True positive ratio (Sensitivity)	0.707	0.723	0.736	0.701	0.559	0.642	0.722
True negative ratio (Specificity)	0.623	0.585	0.645	0.651	0.755	0.676	0.653
Positive predictive value (Precision)	0.211	0.199	0.228	0.223	0.246	0.22	0.229
Negative predictive value	0.937	0.937	0.945	0.938	0.923	0.93	0.943
AUC score	0.726	0.698	0.745	0.733	0.73	0.711	0.745
Gini	0.453	0.396	0.491	0.466	0.46	0.422	0.49

Table 5.19 shows that the Naïve Bayes classifier obtained the highest accuracy score of 0.731. In Chapter 2, we established that there was an imbalance in the dataset, as only 12% of the clients under study defaulted on their loan, whereas 88% of clients under study did not default on their loan. Thus, accuracy is not the most reliable metric in this study since the minority class, default, is the class that the researcher is more interested in, and accuracy tends to be

biased towards the majority class (i.e., not default). The balanced accuracy score is then examined. Table 5.19 shows that the random forest model obtained the highest balanced accuracy score of 0.690, which is marginally higher than the ANN classifier's balanced accuracy score of 0.688; the researcher considers these scores to be good as they are both very close to 70%.

The true positive ratios and true negative ratios are then analysed. From Table 5.19, the random forest attained the highest true positive ratio of 0.736, which corresponds to the researcher's findings from the confusion matrix in Table 5.18 which indicated that the random forest correctly classified the most number of defaulters. The decision tree classifier and ANN classifier both attained a true positive ratio of approximately 0.72, which was the second highest. A true positive ratio above 70% is considered to be a good score by the researcher. The true negative ratios are examined next. Table 5.19 shows that the Naïve Bayes classifier attained the highest true negative ratio of 0.755; however, this classifier obtained the worst true positive ratio of 0.559, which is one of the most significant metrics in the study. Since the misclassification costs associated with false positives is negligible, the researcher considers a true negative ratio of about 65% and higher as acceptable. The random forest and ANN model both obtained a true negative ratio of approximately 65%.

The positive predictive values and negative predictive values are then analysed. The positive predictive value (precision) was low across all models which was presumably influenced by the significant imbalance in the dataset. The Naïve Bayes classifier obtained the highest positive predictive value of 0.246; the ANN and random forest model attained the second and third highest values of 0.229 and 0.228, respectively. As previously stated, the low positive predictive value (precision) is not a concern to the researcher as the main focus of this study is to identify defaulters and the costs associated with misclassifying non-defaulters is very low. The negative predictive value was high across all models; the random forest and ANN models attained the highest and second highest negative predictive values of 0.945 and 0.943, respectively. Table 5.19 also reports the AUC score and Gini for each model; the random forest and ANN model both attained an AUC score of 0.745 and a Gini of approximately 0.49, respectively, which were the highest values across all classifiers under study.

Since the large imbalance in the dataset likely had an influence on the accuracy, positive predictive value and negative predictive value, the researcher focuses on the balanced accuracy, true positive ratio, true negative ratio, AUC score and Gini when identifying the most appropriate classification algorithm for the problem under study. From the confusion matrix in Table 5.18 and the performance metrics in Table 5.19, the random forest classifier seemed to have the best overall performance, followed by the ANN classifier. Since the ANN model's training time is longer and there are several parameters that need to be tuned, the random forest classifier is the most favourable one.

The researcher then compares the models that were most suitable for the classification problem under study when using the PCA approach in Chapter 4 and the feature selection approach in Chapter 5 in order to identify the most appropriate model overall. In both cases, the random forest algorithm seemed to be the most suitable model. Therefore, the performance metrics associated with the random forest model using the PCA approach is compared to the performance metrics associated with the random forest model that utilised feature selection. The results are presented in Table 5.20.

Table 5. 20: Performance metrics for the random forest model using the PCA approach and the random forest model using feature selection

Random Forest	PCA	Feature Selection
Accuracy	0.656	0.657
Balanced accuracy	0.691	0.690
True positive ratio	0.738	0.736
True negative ratio	0.645	0.645
Positive predictive value (Precision)	0.229	0.228
Negative predictive value	0.945	0.945
AUC	0.744	0.745
Gini	0.489	0.491

Table 5.20 shows a marginal difference between each performance metric listed, when comparing the random forest model using the PCA approach to the random forest model using feature selection. The AUC value is a single scaler value that can be used to compare the overall performance of the models; from Table 5.20, the AUC scores are similar across both models. The ROC curves for the random forest classifier using feature selection and the random forest classifier using PCA are then analysed in Figure 5.2.

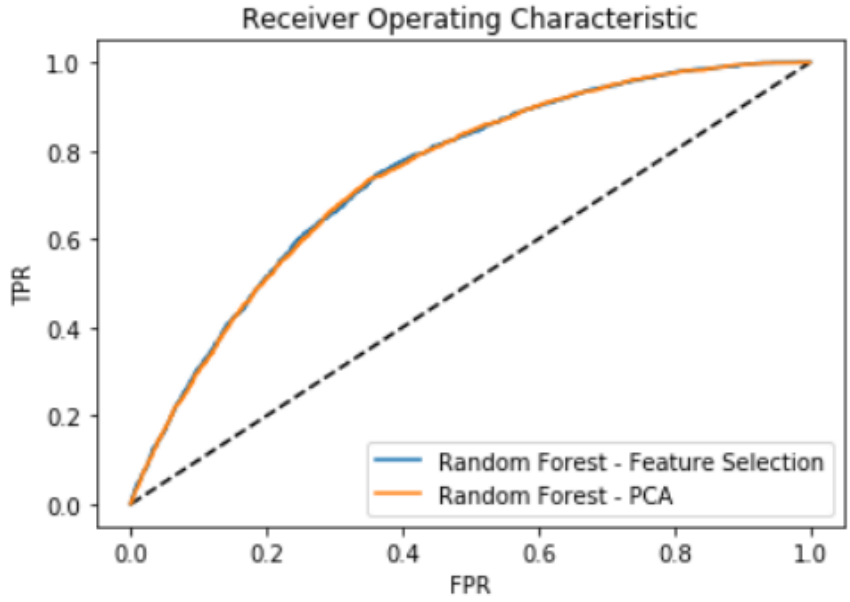

Figure 5. 2: ROC curve for the random forest model using PCA and the random forest model using feature selection

From Figure 5.2, remarkably similar results for both models are shown. This supports the results reported in Table 5.20. Thus, from Table 5.20 and Figure 5.2, both classifiers seem to have attained similar results. Since the random forest model using feature selection included 18 features, whereas the random forest model using PCA included 48 features, the random forest model using feature selection seems to be the more suitable model. Fewer features in a model reduce the computational costs of modelling as well as the risk of data errors. Figure 5.3 shows the feature importance of the 18 features included in the random forest model using feature selection, ranked in ascending order of importance.

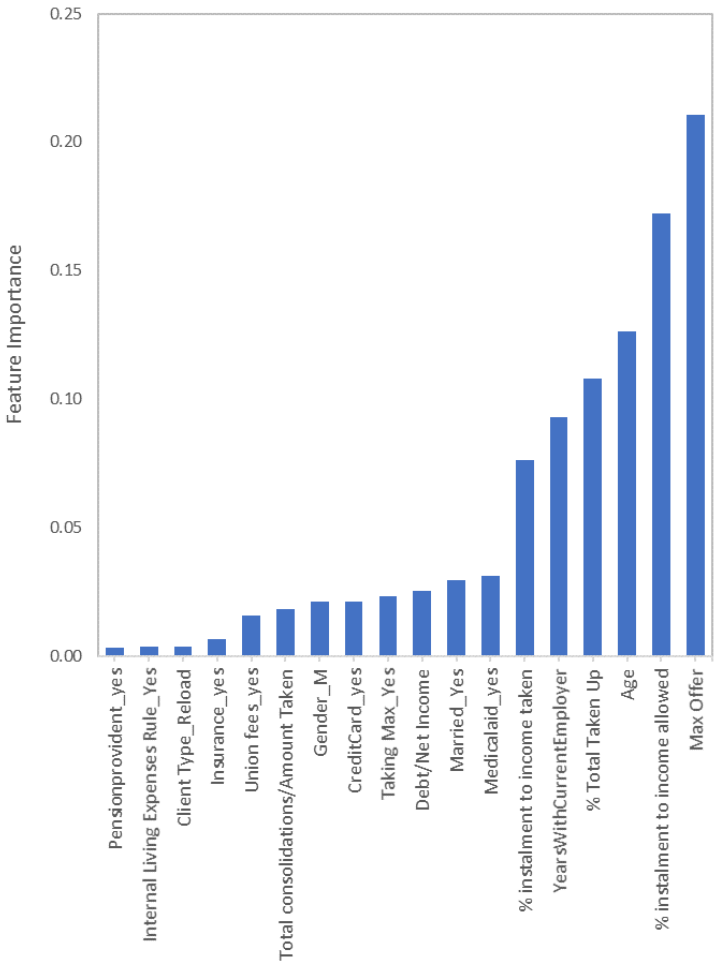

Figure 5. 3: Feature importance of variables in the random forest model that used feature selection

From Figure 5.3, Max Offer, % instalment to income allowed, Age, % Total Taken Up, % instalment to income taken and YearsWithCurrentEmployer were the most important features in the random forest model which used the feature selection approach, when predicting the default status of clients. Features toward the lower end of the tail (e.g., Pensionprovident_yes, Internal Living Expenses Rule_Yes and Client Type_Reload) seem to be considerably less important compared to features such as Max Offer and % instalment to income allowed. The objective of this study was to obtain the best model; however, more features may be removed by adjusting the threshold used in the feature selection process, to simplify the model. This results in a tradeoff between model performance and model simplicity.

Each classification algorithm discussed in Chapter 3 was fitted to the balanced default dataset under study. In chapter 4 and chapter 5 we discussed the results obtained for each model when using the PCA approach and feature selection approach, respectively. The next chapter concludes this study and provides answers to each of the research questions listed in Chapter 1.

Chapter 6

6 Conclusions, limitations and recommendations

This chapter provides the conclusions and limitations to this study, and recommendations for further study.

6.1 Conclusions

Financial institutions have rules and regulations in place that dictate whom to lend to and how much the institution is willing to lend, depending on the client's affordability and risk levels, among other factors. Lending institutions, however, still expect a percentage of clients to default on their payments. Clients miss payments on their loans for multiple reasons. Some cannot afford payments due to mismanagement of funds, additional unexpected costs et cetera, whereas others choose not to pay. It is necessary for financial institutions to have a good collections process in place in order reduce the number of clients who default on their debt obligation and to collect as much unpaid debt as possible. The financial institution under study currently starts the debt recovery process once a client misses a payment. The institution now wants to enhance the collection process by identifying clients who are more likely to miss payments, as soon as the loan is granted, in order for the institution to send through reminder SMS messages and emails to this population at the beginning of each month, starting from the month in which the first instalment is due. This enhanced process will likely result in the financial institution retrieving more unpaid debt and reducing the number of clients who default on their loans.

Thus, in this study, machine learning algorithms were used to predict whether a client will default on his/her loan (i.e., miss at least three payments in the first 12 months of the loan being granted) by using information available at the time the loan is granted so that an enhanced collections process can be used on clients who were classified as defaulters. The researcher fitted the logistic regression model, decision tree, random forest, support vector machines, Naïve Bayes classifier, k-nearest neighbours and artificial neural network to the balanced default dataset. To reduce the dimensionality of the dataset, two techniques were used, namely, principal component analysis (PCA), which is also used to correct the data for multicollinearity, and feature selection i.e., recursive feature elimination, which aims to remove irrelevant and redundant features. The researcher compared the results obtained by the different models in order to identify the model which was most suitable for the problem under study. Since the additional step in the collections process involving the sending of emails and SMS's is not costly, misclassifying non-defaulters as defaulters (i.e., false positives) was not a major concern; the financial institution was more concerned with identifying clients who default.

The first research question aimed to identify which classification algorithms were able to correctly classify a sufficient proportion of clients who defaulted on their loan. From the results in the study, it can be concluded that, where the PCA approach was used, the random forest, ANN and logistic regression model classified a significant proportion of clients who defaulted

correctly; where feature selection was used, the random forest, ANN, SVM, decision tree and logistic regression models were able to correctly classify a substantial portion of defaulters by using the reduced subset of features. Chen and Zhang (2021) reviewed the artificial neural network, k-nearest neighbour, decision tree, support vector machine and logistic regression and concluded that all six models, including the K-NN model, could be used to predict the default of automobile. In contrary, in this study, the K-NN model obtained results which were unsatisfactory when using both, the PCA approach and the feature selection approach. By fitting the K-NN algorithm to various default datasets, one can gain further insight into the algorithm's ability to predict the default status of clients.

The second research question aimed to identify which classification algorithm was most appropriate when using the PCA approach and the feature selection approach, and which model was most appropriate overall. The evaluation metrics used to analyse and compare the model's performance were accuracy, balanced accuracy, true positive ratio, true negative ratio, positive predictive value, negative predictive value, AUC score, and the Gini coefficient. From the results presented in Chapters 4 and 5, where the PCA and feature selection methods were utilised, respectively, both methods showed that the random forest and ANN models seemed to have the best overall performance when considering the classification problem under study. When using the PCA approach, the random forest model obtained a balanced accuracy score, true positive ratio, true negative ratio and AUC score of 0.69, 0.74, 0.65 and 0.74, respectively; these values for the fitted ANN model were 0.69, 0.74, 0.64 and 0.75, respectively. When using the feature selection approach (i.e., recursive feature elimination), the random forest model attained a balanced accuracy score, true positive ratio, true negative ratio and AUC score of 0.69, 0.74, 0.65 and 0.75, respectively, whereas these values were 0.69, 0.72, 0.65 and 0.75, respectively for the fitted ANN model. Since the ANN model has a significantly longer training time, has several parameters that need to be tuned, and has little interpretability, the random forest model seemed to be the most suitable model when predicting the default status of clients under study when using both PCA and feature selection techniques. In line with the results of this study, Bayraci and Susuz (2019), Madaan et al. (2021), Ince and Aktan (2009) and Radhika et al. (2021) concluded that either the random forest model or the neural network performed the best when comparing various models used to predict the default status of clients.

The researcher then identified which model was the most appropriate overall. Since the random forest model was most suitable when using both PCA and feature selection, a comparison between the random forest model using PCA and the random forest model using feature selection was made in order to ascertain the most suitable model overall. When comparing both models, the results (presented in Chapter 5) showed a marginal difference between each performance metric analysed. The ROC AUC curves also showed great similarity between the two models. The random forest model using feature selection utilised 18 features, whereas the random forest model using PCA utilised 48 features. Fewer features help to simplify models, make implementation easier, and may reduce the risk of data errors, since reducing the number of features reduces the risk of errors during data collection and storage. Therefore, the random forest model using feature selection seemed most appropriate for the classification problem under study.

The third research question aimed to identify the key risk factors associated with loan defaulting clients; based on the recommended model (i.e., the random forest using feature selection), these factors are the client's age, the number of years the client has been with their current company, the maximum amount offered to the client, the maximum ratio of instalment to income that the client is allowed, the percentage of the total offer taken up by the client and the instalment amount taken by the client as a percentage of their income. Chen and Zhang (2021), who aimed to predict automobile credit defaulters, used feature selection and found that date of birth, employment type, disbursed amount and asset cost were ranked most important when predicting the 'default' target variable. Bayraci and Susuz (2019), Kadam et al. (2021) and Kwofie et al. (2015), among others, mention that variables such as age, gender, income, and credit information were included in their default models. Thus, features related to a client's age, income and loan amount often seem to be important in models which aim to predict the default status of clients.

6.2 Limitations to the study

This study entailed fitting machine learning classification algorithms to loan data from a South African financial institution in order to predict a client's default status. There were a few limitations and challenges experienced that were related to the dataset and classification algorithms. Firstly, the data used in this study was limited to banking clients from only one South African financial institution whose name is withheld due to the nondisclosure agreement. Secondly, this study focused on clients who defaulted at least three times in the first 12 months of the loan being granted; clients who defaulted outside of this condition could not be identified and were included in the non-defaulting class. Thirdly, all results from this study are pertaining to data from the period August 2019 to December 2019. Fourthly, many machine learning algorithms are often considered black boxes. It is difficult to attain a comprehensive understanding of how the models work once they have been trained; therefore, it is difficult to understand and explain the behaviour of the models.

6.3 Recommendations for further study

For future research, alternate techniques used to handle imbalanced datasets can be investigated to help improve the model performance, focusing on the reduction of false positives; the SMOTE method and weightings method, among others, were investigated in this study, however, whilst it did improve each model's performance when identifying clients who defaulted, the models still attained many false positives. Secondly, machine learning algorithms are often considered black boxes; there is little understanding of how a specific prediction is made for many of these models. Further research into tools and methods which help with the interpretability of these algorithms can be explored.

References

Adhatrao, K., Gaykar, A., Dhawan, A., Jha, R., & Honrao, V. (2013). Predicting students' performance using ID3 And C4.5 classification algorithms. *International Journal of Data Mining & Knowledge Management Process (IJDKP), 3*(5), 39-52.

Ahmad, S. (2011). Diagnostic for Residual Outliers using Deviance Component in Binary Logistic Regression. *World Applied Sciences Journal 14(8)*, 1125-1130.

Ahmed, M. S., & Rajaleximi, P. R. (2019). An empirical study on credit scoring and credit scorecard for financial institutions. *International Journal of Advanced Research in Computer Engineering & Technology, 8*(7), 2278-1323.

Alfeilat, H. A., Hassanat, A. B., Lasassmeh, O., Tarawneh, A. S., Alhasanat, M. B., Salman, H. S., & Prasath, V. S. (2019). Effects of distance measure choice on k-nearest neighbor classifier performance: a review. *Big Data, 7*(4).

Ali, J., Khan, R., Ahmad, N., & Maqsood, I. (2012). Random forests and decision trees. *International Journal of Computer Science Issues, 9*(5), 272-278.

Ali, N., Neagu, D., & Trundle, P. (2019). Evaluation of k-nearest neighbour classifier performance for heterogeneous datasets. *SN Applied Sciences, 1,* Article 1559. https://doi.org/10.1007/s42452-019-1356-9

Aphale, A. S., & Shinde, S. R. (2020). Predict loan approval in banking system machine learning approach for cooperative banks loan approval. *International Journal of Engineering Research & Technology, 9*(8), 991-995.

Arjaria, S. K., Rathore, A. S., & Cherian, J. S. (2021). Kidney disease prediction using a machine learning approach: a comparative and comprehensive analysis. In N. Pradeep, S. Kautish, & S.-L. Peng (Eds.), *Demystifying big data, machine learning, and deep learning for healthcare analytics* (pp. 307-333). Academic Press.

Aslam, U., Aziz, H. I., Sohail, A., & Batcha, N. K. (2019). An empirical study on loan default prediction models. *Journal of Computational and Theoretical Nanoscience, 16,* 3483-3488.

Awad, M., & Khanna, R. (2015). *Efficient learning machines: theories, concepts, and applications for engineers and system designers.* Apress.

Badi, N. H. (2017). Asymptomatic Distribution of Goodness-of-Fit Tests in Logistic Regression Model. *Journal of Statistics, vol 7, No.3*, 434-445.

Bayraci, S., & Susuz, O. (2019). Deep neural network (DNN) based classification model in application to loan default prediction. *Theoretical and Applied Economics : GAER Review, 26*, 75-84.

Belete, D. M., & Manjaiah, D. H. (2020). Wrapper based feature selection techniques on EDHS-HIV/AIDS dataset. *European Journal of Molecular & Clinical Medicine, 7*(8), 2642-2657.

Berrar, D. (2018). Bayes' theorem and Naive Bayes classifier. In S. Ranganathan, M. Gribskov, K. Nakai, & C. Schonbach (Eds.), *Encyclopedia of Bioinformatics and Computational Biology* (Vol 1, 403-412). Elsevier.

Bhavsar, H., & Panchal, M. H. (2012). A review on support vector machine for data classification. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 1*(10), 185-189.

Bing, X., Naiyan, W., Tianqi, C., & Mu, L. (2015). Empirical evaluation of rectified activations in convolutional network. arXiv:1505.00853v2[cs.KG], https://doi.org/10.48550/arXiv.1505.00853

Bommert, A., Sun, X., Bischl, B., Rahnenführer, J., & Langa, M. (2020). Benchmark for filter methods for feature selection in high-dimensional classification data. *Computational Statistics & Data Analysis, 143*.

Breeden, J. (2020). *A survey of machine learning in credit risk.* ResearchGate. DOI:10.13140/RG.2.2.14520.37121

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*(2), 123-140.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5-32.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees.* Wadsworth International Group.

Chang, S., Kim, S. D.-O., & Kondo, G. (2015). Predicting default risk of lending club loans. *CS229: Machine Learning*. http://cs229.stanford.edu/proj2015/199_report.pdf

Chen, X., & Ishwaran, H. (2012). Random forests for genomic data analysis. *Genomics, 99*(6), 323-329.

Chen, Y., & Zhang, R. (2021). Default prediction of automobile credit based on support cector machine. *Journal of Information Processing Systems, 17*(1), 75-88.

Chifurira, R. (2018). Modelling Mean Annual Rainfall for Zimbabwe, PhD thesis. University Of The Free State, Bloemfontein.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13*(1), 21-27.

Dayton, C. M. (1992). *Logistic Regression Analysis.* http://www.econ.upf.edu/~satorra/dades/M2012LogisticRegressionDayton.pdf

Demichelis, F., Magni, P., Piergiorgi, P., Rubin, M. A., & Bellazzi, R. (2006). A hierarchical Naïve Bayes model for handling sample heterogeneity in classification problems: an application to tissue microarrays. *BMC Bioinformatics, 7*(1), 1-12.

Ereiz, Z. (2019, November 26-27). Predicting default loans using machine learning (OptiML). *Conference: 27th Telecommunications Forum (TELFOR 2019).* Belgrade, Serbia.

Fagerland, M. W., & Hosmer, D. W. (2012). A generalized Hosmer–Lemeshow goodness-of-fit test for multinomial logistic regression models. *Stata Journal 12(3),* 447-453

Fakir, Y., Azalmad, M., & Elaychi, R. (2020). Study of The ID3 and C4.5 Learning Algorithms. *Journal of Medical Informatics and Decision Making, 1*(2), 29-43.

Fawagreh, K., Gaber, M. M., & Elyan, E. (2014). Random forests: from early developments to recent advancements. *Systems Science & Control Engineering, 2*(1), 602-609.

Feng, C., Longhai, & Sadeghpour, A. (2020). A comparison of residual diagnosis tools for diagnosing regression models for count data. *BMC Medical Research Methodology 20(175)*, 1-21.

Feng, J., & Lu, S. (2019). Performance analysis of various activation functions in artificial neural networks. *Journal of Physics: Conference Series, 1237*(2).

Fix, E., & Hodges, J. (1951). *Discriminatory analysis, nonparametric discrimination: consistency properties.* (Technical Report No. 4). USAF School of Aviation Medicine, Randolph Field.

Fletcher, T. (2008, December 23). *Support vector machines explained.* www.cs.ucl.ac.uk/staff/T.Fletcher/

Gangrade, A., & Patel, R. (2012). Privacy preserving Naive Bayes classifier for horizontally distribution scenario using un-trusted third party. *IOSR Journal of Computer Engineering, 7*(6), 4-12.

Gao, H., Zeng, X., & Yao, C. (2019). Application of improved distributed naive Bayesian algorithms in text classification. *The Journal of Supercomputing, 75*, 5831-5847.

Gao, X., Wen, J., & Zhang, C. (2019). An improved random forest algorithm for predicting employee turnover. *Mathematical Problems in Engineering, 4,* 1-12. https://doi.org/10.1155/2019/4140707

Ginting, S. L., Adler, J., Ginting, Y. R., & Kurniadi, A. H. (2018, September). The development of bank applications for debtors' selection by using Naive Bayes classifier technique. *IOP Conference Series Materials Science and Engineering, 407*(1), 012090. DOI:10.1088/1757-899X/407/1/012090

Gulati, P., Sharma, A., & Gupta, M. (2016). Theoretical study of decision tree algorithms to identify pivotal factors for performance improvement: a review. *International Journal of Computer Applications, 141*(14), 19-25.

Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). K-NN model-based approach in classification. *International Conference on Ontologies, Databases and Applications of Semantics* (pp. 986-996). Catania, Sicily, Italy: Springer.

Hamming, R. (1958). Error detecting and error correcting codes. *Bell System Technical Journal, 131*(1), 147-160.

Han, J., Pei, J., & Kamber, M. (2006). *Data mining, Southeast Asia edition* (2nd ed.). Elsevier.

Hand, D. J., & Till, R. J. (2001). A simple generalisation of the area under the ROC. *Machine Learning, 45,* 171-186.

He, J., Zhang, Y., Li, X., & Shi, P. (2012). Learning Naive Bayes classifiers from positive and unlabelled examples with uncertainty. *International Journal of Systems Science, 43*(10), 1805-1825 .

Hearst, M. A., Dumais, S., Osman, E., Platt, J. C., & Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems, 13*(4), 18-28.

Hintze, J. L. (2007). *NCSS User's Guide 3.* NCSS.

Hu, L.Y., Huang, M.W., Ke, S.W., & Tsai, C.F. (2016). The distance function effect on k-nearest neighbor classification for medical datasets. *SpringerPlus, 5*(1), 1-9.

Ince, H., & Aktan, B. (2009). A comparison of data mining techniques for credit scoring in banking: a managerial perspective. *Journal of Business Economics and Management, 10*(3), 233-240.

Jakkula, V. (2006). Tutorial on support vector machine (SVM). *School of EECS, Washington State University.* https://course.ccs.neu.edu/cs5100f11/resources/jakkula.pdf

Jang, W., Lee, J. K., Lee, J., & Han, S. H. (2015). Naïve Bayesian classifier for selecting good/bad projects during the early stage of international construction bidding decisions. *Mathematical Problems in Engineering* 2015(10), 1-12.

Kadam, A. S., Nikam, S. R., Aher, A. A., Shelke, G. V., & Chandgude, A. S. (2021). Prediction for loan approval using machine learning algorithm. *International Research Journal of Engineering and Technology, 8*(4), 4089-4092.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence. 97*(1-2), 273-324.

Krichene, A. (2017). Using a Naive Bayesian classifier methodology for loan risk assessment: evidence from a Tunisian commercial bank. *Journal of Economics Finance and Administrative Science, 22*(42), 3-24.

Kuhn, L., Page, K., Ward, J., & Worrall-Carter, L. (2013). The process and utility of classification and regression tree methodology in nursing research. *Journal of Advanced Nursing, 70,* 1276-1286.

Kwofie, C., Ansah, C. O., & Boad, C. (2015). Predicting the probability of loan-default: an application of binary logistic regression. *Research Journal of Mathematics and Statistics 7*(4), 46-52.

Lall, U., & Sharma, A. (1996). A nearest neighbor bootstrap for resampling hydrologic time series. Water Resour. Res., 32(3), 679-693.

Lallahem, S., & Mania, J. (2003). A nonlinear rainfall-runoff model using neural network technique: example in fractured porous media. *Mathematical and Computer Modelling, 37,* 1047-1061.

Lavrač, N., Todorovski, L., & Jantke, K. P. (Eds.) (2006, October 7-10). *Proceedings of the Discovery Science: 9th International Conference, (DS 2006), Barcelona, Spain.* Springer.

Lee, T.-S., Chiu, C.-C., Lu, C.-J., & Chen, I.-F. (2002). Credit scoring using the hybrid neural discriminant technique. *Expert Systems with Applications, 23*(3), 245-254.

Li, J., Wang, S., Cheng, K., & Morstatter, F. (2016). Feature selection: a data perspective. *ACM Computing Surveys, 50*(6).

Lin, C., & Fan, C. (2019). Evaluation of CART, CHAID, and QUEST algorithms: a case study of construction defects in Taiwan. *Journal of Asian Architecture and Building Engineering, 18*(6), 539-553.

Liu, Q., Lu, J., Chen, S., & Zhao, K. (2014). Multiple Naïve Bayes classifiers ensemble for traffic incident detection. *Mathematical Problems in Engineering, 4,* 1-16. Article ID 383671. DOI:10.1155/2014/383671

Ma, C.M., Yang, W.S., & Cheng, B.W. (2014). How the parameters of k-nearest neighbor algorithm impact on the best classification accuracy: in case of Parkinson dataset. *Journal of Applied Sciences, 14*(2), 171-176. DOI: 10.3923/jas.2014.171.176

Madaan, M., Kumar, A., Keshri, C., Jain, R., & Nagrath, P. (2021). Loan default prediction using decision trees and random forest: a comparative study. In *IOP Conference Series Materials Science and Engineering* (Vol. 1022, No. 1, p. 012042). IOP Publishing. Pp 1-12

Marqués, A., García, V., & Sánchez, J. (2012). Exploring the behaviour of base classifiers in credit scoring ensembles. *Expert Systems with Applications, 39,* 10244-10250.

Mazinani, S. M., & Fathi, K. (2015). Combining K-NN and decision tree algorithms to improve intrusion detection system performance. *International Journal of Machine Learning and Computing, 5*(6), 476-479.

McCullagh, P., & Nelder, J. A. (1989). *Generalized Linear Models 2nd Edition.* London: Chapman and Hall.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics, 5,* 115-133.

Moon, J., Park, S., Rho, S., & Hwang, E. (2019). A comparative analysis of artificial neural network architectures for building energy consumption forecasting. *International Journal of Distributed Sensor Networks.* https://doi.org/10.1177%2F1550147719877616

Murthy, S. K., & Salzberg, S. (1995). *Decision tree induction: how effective is the greedy heuristic?* KDD https://dblp.org/db/conf/kdd

Mwadulo, M. W. (2016). A review on feature selection methods for classification tasks. *International Journal of Computer Applications Technology and Research, 5*(6), 395-402.

Mwangi, M. C. (2016). Effect of loan collection procedures and loan default in microfinance institutions in Kirinyaga County. *Global Journal of Management and Business Research,16*(8), Version 1.0, 36-42.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. *Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML 10)*, 807-814.

Naufal, N., Devila, S., & Lestari, D. (2019). Generalized Linear Model (GLM) to Determine Life Insurance Premiums. *AIP Conference Proceedings 2168 (020036)*, Pp 1-8.

Nayak, J., Naik, B., & Behera, H. (2015). A comprehensive survey on support vector machine in data mining tasks: applications and challenges. *International Journal of Database Theory and Application, 8*(1), 169-186.

Oken, A. (2017, May 5). *An introduction to and applications of neural networks*. https://www.whitman.edu/Documents/Academics/Mathematics/2017/Oken.pdf

Pal, M., & Mather, P. M. (2005). Support vector machines for classification in remote sensing. *International Journal of Remote Sensing, 26*(5).

Panthong, R., & Srivihok, A. (2015). Wrapper feature subset selection for dimension reduction based on ensemble learning algorithm. *Procedia Computer Science, 72*, 162-169.

Park, H.A. (2013). An introduction to logistic regression: from basic concepts to interpretation with particular attention to nursing domain. *Journal of Korean Academy of Nursing, 43*(2), 154-164.

Pasini, A. (2015). Artificial neural networks for small dataset analysis. *Journal of Thoracic Disease, 7*(5), 953-960.

Patel, N., & Upadhyay, S. (2012). Study of various decision tree pruning methods with their empirical comparison in WEKA. *International Journal of Computers and Applications, 60*(12), 20-25.

Peng, J., So, T.-S. H., Stage, K., & John, E. P. (2002). The use and interpretation of logistic regression in higher education journals 1988-1999. *Research in Higher Education, 43*(3), 259-293.

Perera, H., & Premaratne, S. (2016, September). An artificial neuralnetwork approach for the predictive accuracy of payments of leasing customers in Sri Lanka. *Full Paper Proceeding BESSH-2016, 285*(2), 1-11.

Podgorelec, V., Kokol, P., Stiglic, B., & Rozman, I. (2002). Decision trees: an overview and their use in medicine. *Journal of Medical Systems, 26*(5), 445-463.

Purswani, R., Verma, S., & Jaiswal, Y. (2021). Loan approval prediction using machine learning: a review. *International Research Journal of Engineering and Technology, 8*(6), 3252-3255.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*, 81-106.

Radhika, R., Dharani, G. V., & Thanuja, G. (2021). Loan risk prediction using machine learning. *International Research Journal of Engineering and Technology, 8*(3).

Rampisela, T., & Rustam, Z. (2018). Classification of schizophrenia data using support vector machine (SVM). *Journal of Physics: Conference Series 1108 012044*.

Ruder, S. (2017, June 15). *An overview of gradient descent optimization algorithms*. Sebastian Ruder. https://ruder.io/optimizing-gradient-descent/

Sarkar, S., Midi, H., & Rana, S. (2011). Detection of Outliers and Influential Observations in Binary Logistic Regression: An Empirical Study. *Journal of Applied Sciences*, 11, 26-35.

Schechtman, E., & Schechtman, G. (2019). The relationship between Gini terminology and the ROC curve. *Metron, 77*(6). DOI:10.1007/s40300-019-00160-7

Shahid, N., Rappon, T., & Berta, W. (2019). Applications of artificial neural networks in health care organizational decision-making: a scoping review. *PLOS One, 14*(2)*: e0212356. https://doi:10.1371/journal.pone.0212356*.

Shalizi, C. R. (2019). *Advanced data analysis from an elementary point of view.* https://freecomputerbooks.com/Advanced-Data-Analysis-from-an-Elementary-Point-of-View.html

Shiruru, K., Kukreja, H., Bharath, N., & Siddesh, C. (2016). An introduction to artificial neural network. *International Journal Of Advance Research And Innovative Ideas In Education, 1*(5), 27-30.

Singh, S., & Giri, M. (2014). Comparative study Id3, Cart And C4.5 decision tree algorithm: a survey. *International Journal of Advanced Information Science and Technology (IJAIST), 3*(7), 47-52.

Singpurwalla, N. D., & Lai, B. (2020, August 30). *The dinegentropy of diagnostic tests*. Retrieved from arXiv:2008.13127v1: https://arxiv.org/abs/2008.13127v1

Srivastava, D., & Bhambhu, L. (2010). Data classification using support vector machine. *Journal of Theoretical and Applied Information Technology*, *12*(1), 1-7.

Stoltzfus, J. C. (2011). Logistic regression: a brief primer. *Academic Emergency Medicine, 18(10)*, 1099-1104.

Sudhamathy, G. (2016). Credit risk analysis and prediction modelling of bank loans using r. *International Journal of Engineering and Technology*, *8*(5), 1954-1966.

Tawfiq, L. N., & Thirthar, A. A. (2013). Improving gradient descent method for training feed forward neural networks. *International Journal of Modern Computer Science & Engineering*, *2*(1), 12-24.

Tyralis, H., Papacharalampous, G., & Langousis, A. (2019). A brief review of random forests for water scientists and practitioners and their recent history in water resources. *Water, 11*(5), 910-946.

Vapnik, V. N. (1982). *Estimation of dependences based on empirical data* (S. Kotz, Trans.). Springer.

Vapnik, V. N. (1995). *The nature of statistical learning theory.* Springer.

Venables, W. N., & Ripley, B. D. (1999). *Generalized Linear Models. In: Modern Applied Statistics with S-PLUS. Statistics and Computing.* New York: Springer.

Venkatesh, B., & Anuradha, J. (2019). A review of feature selection and its methods. *Cybernetics and Information Technologies, 19*(1), 3-26.

Wang, S., Tang, J., & Liu, H. (2016). Feature Selection. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning and Data Mining.* Springer Science+Business Media.

Wibawa, A., Kurniawan, A. C., Adiperkasa, R. P., Murti, D. M., Putra, S. M., Kurniawan, S. A., & Nugraha, Y. R. (2019). Naïve Bayes classifier for journal quartile classification. *International Journal of Recent Contributions from Engineering Science & IT, 7*(2), 91-99.

Yang, Y. (2007). Adaptive credit scoring with kernel learning methods. *European Journal of* Operational Research, 183(3), 1521-1536.

Zhang, G., Hu, M. Y., Patuwo, B. E., & Indro, D. C. (1999). Artificial neural networks in bankruptcy prediction: general framework and cross-validation analysis. European Journal of Operational Research, 116(1), 16-32.

Zhang, S., Li, X., Zong, M., Zhu, X., & Wang, R. (2018). Efficient K-NN Classification With Different Numbers of Nearest Neighbors. IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 5, 1774-1785.

Zhou, L., & Wang, H. (2012). Loan default prediction on large imbalanced data using random forests. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, *10*(6), 1519-1525.

Appendix 1 – Principal Component Analysis

```
#########Packages#########

import pandas as pandas
from pandas.plotting import scatter_matrix
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
import scipy.stats
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
#from sklearn.datasets import make_classification
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import GridSearchCV
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.svm import SVC
import tensorflow
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.constraints import maxnorm
from keras.layers import Dropout
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import matthews_corrcoef, precision_score, accuracy_score, recall_score,
f1_score
from sklearn.metrics import roc_auc_score, r2_score
from sklearn.metrics import roc_curve




#########PCA#########


###import data###
url = "C:/Users/suered/Desktop/masters/MastersData.csv"
dataset = pandas.read_csv(url,low_memory=False)


###First select number of principal components###

PCAfeatures = ['Age',
'Instalment /Disposable income',
'YearsWithCurrentEmployer',
'% Total Taken Up',
'NumberOfDependents',
'Disposable income/Basic',
'Debt/Net Income',
'Max Offer',
'% instalment to income taken',
'% instalment to income allowed',
'External consolidations/Amount Taken',
'Internal consolidations/Amount Taken',
```

```
'Total consolidations/Amount Taken',
'Debt to Income ratio',
'Final Disposable Income/NetIncome',
'Calc Disposable Income/NetIncome',
]


PCAFeat = dataset.loc[:, PCAfeatures].values
PCAFeat = StandardScaler().fit_transform(PCAFeat)

pca = PCA(n_components=16)
principalComponents = pca.fit_transform(PCAFeat)

var = pca.explained_variance_ratio_
var1 = np.cumsum(np.round(pca.explained_variance_ratio_, decimals = 3)*100)
var1


###Create 7 principal components###
pca = PCA(n_components=7)
principalComponents = pca.fit_transform(PCAFeat)
principalDf = pandas.DataFrame(data = principalComponents
             , columns = ['principalcomponent1'
                         ,'principalcomponent2'
                          ,'principalcomponent3'
                          ,'principalcomponent4'
                          ,'principalcomponent5'
                          ,'principalcomponent6'
                          ,'principalcomponent7'

                          ])

pca.explained_variance_ratio_


###Check correlation of principal components###
plt.figure(figsize=(20,20))

heatmap = sns.heatmap(principalDf.corr(),
            vmin=-1,vmax=1,annot=True)

heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':15},pad=15);




#########Logistic Regression - PCA#########



###import data###

url = "C:/Users/suered/Desktop/masters/MastersData.csv"
dataset = pandas.read_csv(url,low_memory=False)


###categorical variables - get_dummies###

X= dataset.drop('Default', axis=1)
Y = dataset['Default']
X_transformed = pandas.get_dummies(X, drop_first=True)
dataset = pandas.concat([Y,X_transformed],axis=1)
###PCA###

correlatedfeatures = ['Age',
'Instalment /Disposable income',
'YearsWithCurrentEmployer',
'% Total Taken Up',
'NumberOfDependents',
'Disposable income/Basic',
'Debt/Net Income',
```

```
'Max Offer',
'% instalment to income taken',
'% instalment to income allowed',
'External consolidations/Amount Taken',
'Internal consolidations/Amount Taken',
'Total consolidations/Amount Taken',
'Debt to Income ratio',
'Final Disposable Income/NetIncome',
'Calc Disposable Income/NetIncome',
]
xcorr = dataset.loc[:, correlatedfeatures].values
xcorr = StandardScaler().fit_transform(xcorr)
from sklearn.decomposition import PCA
pca = PCA(n_components=7)
principalComponents = pca.fit_transform(xcorr)

principalDf = pandas.DataFrame(data = principalComponents
             , columns = ['principalcomponent1'
                         ,'principalcomponent2'
                          ,'principalcomponent3'
                          ,'principalcomponent4'
                          ,'principalcomponent5'
                          ,'principalcomponent6'
                          ,'principalcomponent7'
                          ])

pca.explained_variance_ratio_

dataset.reset_index(drop=True, inplace=True)
principalDf.reset_index(drop=True, inplace=True)
dataset = pandas.concat([dataset,principalDf ],axis=1)

dataset.drop(['Age',
'Instalment /Disposable income',
'YearsWithCurrentEmployer',
'% Total Taken Up',
'NumberOfDependents',
'Disposable income/Basic',
'Debt/Net Income',
'Max Offer',
'% instalment to income taken',
'% instalment to income allowed',
'External consolidations/Amount Taken',
'Internal consolidations/Amount Taken',
'Total consolidations/Amount Taken',
'Debt to Income ratio',
'Final Disposable Income/NetIncome',
'Calc Disposable Income/NetIncome',
],axis=1,inplace=True)


###Splitting Data###

X = dataset.iloc[:, 1:]
Y = dataset.iloc[:, 0]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 0,)


###Standardisation###

# set up the scaler
scaler = StandardScaler()
# fit the scaler to the train set
scaler.fit(X_train)
# transform train and test sets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
# transform NumPy arrays to dataframes
X_train_scaled = pandas.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled = pandas.DataFrame(X_test_scaled, columns=X_test.columns)
# standardisation
```

```
X_train = X_train_scaled
X_test = X_test_scaled



### Getting class weight###

from sklearn.utils.class_weight import compute_class_weight
cw = compute_class_weight('balanced', np.unique(y_train), y_train)
class_weight = {c: w for c, w in zip(np.unique(y_train), cw)}


###Classifier###

classifier = LogisticRegression(n_jobs=-1,
                                solver = 'newton-cg',
                                C = 10,
                                penalty = 'l2',
                                class_weight=class_weight,
                                random_state=0)
classifier.fit(X_train, y_train)


###Evaluation###

# Gini function
def gini(y_true, y_pred, sample_weight=None):
    y_true = np.squeeze(y_true)
    y_pred = np.squeeze(y_pred)
    return 2.*roc_auc_score(y_true, y_pred, sample_weight=sample_weight) - 1


#Training Set

y_train_pred = classifier.predict(X_train)
y_train_pred_prob = classifier.predict_proba(X_train)
y_train_pred_adj = (y_train_pred_prob[:,1] >= 0.5).astype('int')

print('Confusion Matrix: ')
print(confusion_matrix(y_train, y_train_pred))
print()
print('Model Results: ')
print(classification_report(y_train,y_train_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_train,y_train_pred), 4))
print("Gini:",gini(y_train.astype(int), y_train_pred_prob[:,1]))
#Test Set

y_pred = classifier.predict(X_test)
y_pred_prob = classifier.predict_proba(X_test)
y_pred_adj = (y_pred_prob[:,1] >= 0.5).astype('int')

print('Confusion Matrix: ')
print(confusion_matrix(y_test,y_pred))
print()
print('Model Results: ')
print(classification_report(y_test,y_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_test, y_pred), 4))
print("Gini:",gini(y_test.astype(int), y_pred_prob[:,1]))



#########Decision Tree - PCA#########

###import data###

url = "C:/Users/suered/Desktop/masters/MastersData.csv"
dataset = pandas.read_csv(url,low_memory=False)


###categorical variables - get_dummies###
```

```
X= dataset.drop('Default', axis=1)
Y = dataset['Default']
X_transformed = pandas.get_dummies(X, drop_first=True)
dataset = pandas.concat([Y,X_transformed],axis=1)


###PCA###

correlatedfeatures = ['Age',
'Instalment /Disposable income',
'YearsWithCurrentEmployer',
'% Total Taken Up',
'NumberOfDependents',
'Disposable income/Basic',
'Debt/Net Income',
'Max Offer',
'% instalment to income taken',
'% instalment to income allowed',
'External consolidations/Amount Taken',
'Internal consolidations/Amount Taken',
'Total consolidations/Amount Taken',
'Debt to Income ratio',
'Final Disposable Income/NetIncome',
'Calc Disposable Income/NetIncome',
]
xcorr = dataset.loc[:, correlatedfeatures].values
xcorr = StandardScaler().fit_transform(xcorr)
from sklearn.decomposition import PCA
pca = PCA(n_components=7)
principalComponents = pca.fit_transform(xcorr)

principalDf = pandas.DataFrame(data = principalComponents
             , columns = ['principalcomponent1'
                         ,'principalcomponent2'
                          ,'principalcomponent3'
                          ,'principalcomponent4'
                          ,'principalcomponent5'
                          ,'principalcomponent6'
                          ,'principalcomponent7'
                          ])

pca.explained_variance_ratio_

dataset.reset_index(drop=True, inplace=True)
principalDf.reset_index(drop=True, inplace=True)
dataset = pandas.concat([dataset,principalDf ],axis=1)

dataset.drop(['Age',
'Instalment /Disposable income',
'YearsWithCurrentEmployer',
'% Total Taken Up',
'NumberOfDependents',
'Disposable income/Basic',
'Debt/Net Income',
'Max Offer',
'% instalment to income taken',
'% instalment to income allowed',
'External consolidations/Amount Taken',
'Internal consolidations/Amount Taken',
'Total consolidations/Amount Taken',
'Debt to Income ratio',
'Final Disposable Income/NetIncome',
'Calc Disposable Income/NetIncome',
],axis=1,inplace=True)


###Splitting Data###

X = dataset.iloc[:, 1:]
Y = dataset.iloc[:, 0]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 0,)
```

```
###Standardisation###

# set up the scaler
scaler = StandardScaler()
# fit the scaler to the train set
scaler.fit(X_train)
# transform train and test sets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
# transform NumPy arrays to dataframes
X_train_scaled = pandas.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled = pandas.DataFrame(X_test_scaled, columns=X_test.columns)
# standardisation
X_train = X_train_scaled
X_test = X_test_scaled



### Getting class weight###

from sklearn.utils.class_weight import compute_class_weight
cw = compute_class_weight('balanced', np.unique(y_train), y_train)
class_weight = {c: w for c, w in zip(np.unique(y_train), cw)}


###Classifier###

classifier = DecisionTreeClassifier(criterion='entropy',
                                    max_depth = 60,
                                    max_features = 'auto',
                                    min_samples_leaf = 200,
                                    splitter = 'best',
                                    min_impurity_split = 0.6,
                                    class_weight=class_weight,
                                    random_state=0)
classifier.fit(X_train, y_train)


###Evaluation###

# Gini function
def gini(y_true, y_pred, sample_weight=None):
    y_true = np.squeeze(y_true)
    y_pred = np.squeeze(y_pred)
    return 2.*roc_auc_score(y_true, y_pred, sample_weight=sample_weight) - 1


#Training Set

y_train_pred = classifier.predict(X_train)
y_train_pred_prob = classifier.predict_proba(X_train)
y_train_pred_adj = (y_train_pred_prob[:,1] >= 0.5).astype('int')

print('Confusion Matrix: ')
print(confusion_matrix(y_train, y_train_pred))
print()
print('Model Results: ')
print(classification_report(y_train,y_train_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_train,y_train_pred), 4))
print("Gini:",gini(y_train.astype(int), y_train_pred_prob[:,1]))


#Test Set

y_pred = classifier.predict(X_test)
y_pred_prob = classifier.predict_proba(X_test)
y_pred_adj = (y_pred_prob[:,1] >= 0.5).astype('int')

print('Confusion Matrix: ')
print(confusion_matrix(y_test,y_pred))
print()
print('Model Results: ')
```

```
print(classification_report(y_test,y_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_test, y_pred), 4))
print("Gini:",gini(y_test.astype(int), y_pred_prob[:,1]))



#SHAP Graphs

import shap
shap.initjs()

explainer = shap.TreeExplainer(classifier)
shap_values = explainer.shap_values(X_train)
shap.summary_plot(shap_values[1], X_train, plot_type="bar",max_display=57)

explainer = shap.TreeExplainer(classifier)
shap_values = explainer.shap_values(X_test)
shap.summary_plot(shap_values[1], X_test, plot_type="bar",max_display=57)




#########Random Forest – PCA#########

###import data###

url = "C:/Users/suered/Desktop/masters/MastersData.csv"

dataset = pandas.read_csv(url,low_memory=False)


###categorical variables - get_dummies###

X= dataset.drop('Default', axis=1)
Y = dataset['Default']
X_transformed = pandas.get_dummies(X, drop_first=True)
dataset = pandas.concat([Y,X_transformed],axis=1)


###PCA###

correlatedfeatures = ['Age',
'Instalment /Disposable income',
'YearsWithCurrentEmployer',
'% Total Taken Up',
'NumberOfDependents',
'Disposable income/Basic',
'Debt/Net Income',
'Max Offer',
'% instalment to income taken',
'% instalment to income allowed',
'External consolidations/Amount Taken',
'Internal consolidations/Amount Taken',
'Total consolidations/Amount Taken',
'Debt to Income ratio',
'Final Disposable Income/NetIncome',
'Calc Disposable Income/NetIncome',
]
xcorr = dataset.loc[:, correlatedfeatures].values
xcorr = StandardScaler().fit_transform(xcorr)
from sklearn.decomposition import PCA
pca = PCA(n_components=7)
principalComponents = pca.fit_transform(xcorr)

principalDf = pandas.DataFrame(data = principalComponents
             , columns = ['principalcomponent1'
                         ,'principalcomponent2'
                          ,'principalcomponent3'
                          ,'principalcomponent4'
                          ,'principalcomponent5'
                          ,'principalcomponent6'
```

```
                        ,'principalcomponent7'
                        ])

pca.explained_variance_ratio_

dataset.reset_index(drop=True, inplace=True)
principalDf.reset_index(drop=True, inplace=True)
dataset = pandas.concat([dataset,principalDf ],axis=1)

dataset.drop(['Age',
'Instalment /Disposable income',
'YearsWithCurrentEmployer',
'% Total Taken Up',
'NumberOfDependents',
'Disposable income/Basic',
'Debt/Net Income',
'Max Offer',
'% instalment to income taken',
'% instalment to income allowed',
'External consolidations/Amount Taken',
'Internal consolidations/Amount Taken',
'Total consolidations/Amount Taken',
'Debt to Income ratio',
'Final Disposable Income/NetIncome',
'Calc Disposable Income/NetIncome',
],axis=1,inplace=True)


###Splitting Data###

X = dataset.iloc[:, 1:]
Y = dataset.iloc[:, 0]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 0,)


###Standardisation###

# set up the scaler
scaler = StandardScaler()
# fit the scaler to the train set
scaler.fit(X_train)
# transform train and test sets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
# transform NumPy arrays to dataframes
X_train_scaled = pandas.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled = pandas.DataFrame(X_test_scaled, columns=X_test.columns)
# standardisation
X_train = X_train_scaled
X_test = X_test_scaled



### Getting class weight###

from sklearn.utils.class_weight import compute_class_weight
cw = compute_class_weight('balanced', np.unique(y_train), y_train)
class_weight = {c: w for c, w in zip(np.unique(y_train), cw)}
###Classifier###

classifier = RandomForestClassifier(criterion='gini',
                                    bootstrap = True,
                                    max_depth = 40,
                                    max_features = 'auto',
                                    min_samples_leaf = 50,
                                    min_samples_split = 550,
                                    n_estimators = 50,
                                    n_jobs=-1,
                                    oob_score=True,
                                    class_weight=class_weight,
                                    random_state=0)
```

```
classifier.fit(X_train, y_train)

###Evaluation###

# Gini function
def gini(y_true, y_pred, sample_weight=None):
    y_true = np.squeeze(y_true)
    y_pred = np.squeeze(y_pred)
    return 2.*roc_auc_score(y_true, y_pred, sample_weight=sample_weight) - 1


#Training Set

y_train_pred = classifier.predict(X_train)
y_train_pred_prob = classifier.predict_proba(X_train)
y_train_pred_adj = (y_train_pred_prob[:,1] >= 0.5).astype('int')
print('Confusion Matrix: ')
print(confusion_matrix(y_train, y_train_pred))
print()
print('Model Results: ')
print(classification_report(y_train,y_train_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_train,y_train_pred), 4))
print("Gini:",gini(y_train.astype(int), y_train_pred_prob[:,1]))


#Test Set

y_pred = classifier.predict(X_test)
y_pred_prob = classifier.predict_proba(X_test)
y_pred_adj = (y_pred_prob[:,1] >= 0.5).astype('int')

print('Confusion Matrix: ')
print(confusion_matrix(y_test,y_pred))
print()
print('Model Results: ')
print(classification_report(y_test,y_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_test, y_pred), 4))
print("Gini:",gini(y_test.astype(int), y_pred_prob[:,1]))


#SHAP Graphs

import shap
shap.initjs()

explainer = shap.TreeExplainer(classifier)
shap_values = explainer.shap_values(X_train)
shap.summary_plot(shap_values[1], X_train, plot_type="bar",max_display=57)
shap.summary_plot(shap_values[1], X_train,max_display=57)

explainer = shap.TreeExplainer(classifier)
shap_values = explainer.shap_values(X_test)
shap.summary_plot(shap_values[1], X_test, plot_type="bar",max_display=57)
shap.summary_plot(shap_values[1], X_test,max_display=57)




#########SVM - PCA#########

###import data###

url = "C:/Users/suered/Desktop/masters/MastersData.csv"

dataset = pandas.read_csv(url,low_memory=False)
###categorical variables - get_dummies###

X= dataset.drop('Default', axis=1)
Y = dataset['Default']
X_transformed = pandas.get_dummies(X, drop_first=True)
```

```
dataset = pandas.concat([Y,X_transformed],axis=1)
###PCA###

correlatedfeatures = ['Age',
'Instalment /Disposable income',
'YearsWithCurrentEmployer',
'% Total Taken Up',
'NumberOfDependents',
'Disposable income/Basic',
'Debt/Net Income',
'Max Offer',
'% instalment to income taken',
'% instalment to income allowed',
'External consolidations/Amount Taken',
'Internal consolidations/Amount Taken',
'Total consolidations/Amount Taken',
'Debt to Income ratio',
'Final Disposable Income/NetIncome',
'Calc Disposable Income/NetIncome',
]
xcorr = dataset.loc[:, correlatedfeatures].values
xcorr = StandardScaler().fit_transform(xcorr)
from sklearn.decomposition import PCA
pca = PCA(n_components=7)
principalComponents = pca.fit_transform(xcorr)

principalDf = pandas.DataFrame(data = principalComponents
             , columns = ['principalcomponent1'
                         ,'principalcomponent2'
                          ,'principalcomponent3'
                          ,'principalcomponent4'
                          ,'principalcomponent5'
                          ,'principalcomponent6'
                          ,'principalcomponent7'
                          ])

pca.explained_variance_ratio_

dataset.reset_index(drop=True, inplace=True)
principalDf.reset_index(drop=True, inplace=True)
dataset = pandas.concat([dataset,principalDf ],axis=1)

dataset.drop(['Age',
'Instalment /Disposable income',
'YearsWithCurrentEmployer',
'% Total Taken Up',
'NumberOfDependents',
'Disposable income/Basic',
'Debt/Net Income',
'Max Offer',
'% instalment to income taken',
'% instalment to income allowed',
'External consolidations/Amount Taken',
'Internal consolidations/Amount Taken',
'Total consolidations/Amount Taken',
'Debt to Income ratio',
'Final Disposable Income/NetIncome',
'Calc Disposable Income/NetIncome',
],axis=1,inplace=True)


###Splitting Data###

X = dataset.iloc[:, 1:]
Y = dataset.iloc[:, 0]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 0,)


###Standardisation###

# set up the scaler
```

```
scaler = StandardScaler()
# fit the scaler to the train set
scaler.fit(X_train)
# transform train and test sets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
# transform NumPy arrays to dataframes
X_train_scaled = pandas.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled = pandas.DataFrame(X_test_scaled, columns=X_test.columns)
# standardisation
X_train = X_train_scaled
X_test = X_test_scaled



### Getting class weight###

from sklearn.utils.class_weight import compute_class_weight
cw = compute_class_weight('balanced', np.unique(y_train), y_train)
class_weight = {c: w for c, w in zip(np.unique(y_train), cw)}


###Classifier###

classifier = SVC(kernel = 'rbf',
                 gamma = 0.005 ,
                 C = 21,
                 class_weight = 'balanced',
                 probability=True)

classifier.fit(X_train, y_train)


###Evaluation###

# Gini function
def gini(y_true, y_pred, sample_weight=None):
    y_true = np.squeeze(y_true)
    y_pred = np.squeeze(y_pred)
    return 2.*roc_auc_score(y_true, y_pred, sample_weight=sample_weight) - 1


#Training Set

y_train_pred = classifier.predict(X_train)
y_train_pred_prob = classifier.predict_proba(X_train)
y_train_pred_adj = (y_train_pred_prob[:,1] >= 0.5).astype('int')

print('Confusion Matrix: ')
print(confusion_matrix(y_train, y_train_pred))
print()
print('Model Results: ')
print(classification_report(y_train,y_train_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_train,y_train_pred), 4))
print("Gini:",gini(y_train.astype(int), y_train_pred_prob[:,1]))
#Test Set

y_pred = classifier.predict(X_test)
y_pred_prob = classifier.predict_proba(X_test)
y_pred_adj = (y_pred_prob[:,1] >= 0.5).astype('int')

print('Confusion Matrix: ')
print(confusion_matrix(y_test,y_pred))
print()
print('Model Results: ')
print(classification_report(y_test,y_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_test, y_pred), 4))
print("Gini:",gini(y_test.astype(int), y_pred_prob[:,1]))
```

```
#########NB - PCA#########

###import data###

url = "C:/Users/suered/Desktop/masters/MastersData.csv"
dataset = pandas.read_csv(url,low_memory=False)


###categorical variables - get_dummies###

X= dataset.drop('Default', axis=1)
Y = dataset['Default']
X_transformed = pandas.get_dummies(X, drop_first=True)
dataset = pandas.concat([Y,X_transformed],axis=1)


###PCA###

correlatedfeatures = ['Age',
'Instalment /Disposable income',
'YearsWithCurrentEmployer',
'% Total Taken Up',
'NumberOfDependents',
'Disposable income/Basic',
'Debt/Net Income',
'Max Offer',
'% instalment to income taken',
'% instalment to income allowed',
'External consolidations/Amount Taken',
'Internal consolidations/Amount Taken',
'Total consolidations/Amount Taken',
'Debt to Income ratio',
'Final Disposable Income/NetIncome',
'Calc Disposable Income/NetIncome',
]
xcorr = dataset.loc[:, correlatedfeatures].values
xcorr = StandardScaler().fit_transform(xcorr)
from sklearn.decomposition import PCA
pca = PCA(n_components=7)
principalComponents = pca.fit_transform(xcorr)

principalDf = pandas.DataFrame(data = principalComponents
             , columns = ['principalcomponent1'
                         ,'principalcomponent2'
                          ,'principalcomponent3'
                          ,'principalcomponent4'
                          ,'principalcomponent5'
                          ,'principalcomponent6'
                          ,'principalcomponent7'
                          ])


pca.explained_variance_ratio_

dataset.reset_index(drop=True, inplace=True)
principalDf.reset_index(drop=True, inplace=True)
dataset = pandas.concat([dataset,principalDf ],axis=1)

dataset.drop(['Age',
'Instalment /Disposable income',
'YearsWithCurrentEmployer',
'% Total Taken Up',
'NumberOfDependents',
'Disposable income/Basic',
'Debt/Net Income',
'Max Offer',
'% instalment to income taken',
'% instalment to income allowed',
'External consolidations/Amount Taken',
'Internal consolidations/Amount Taken',
'Total consolidations/Amount Taken',
'Debt to Income ratio',
```

```
'Final Disposable Income/NetIncome',
'Calc Disposable Income/NetIncome',
],axis=1,inplace=True)


###Splitting Data###

X = dataset.iloc[:, 1:]
Y = dataset.iloc[:, 0]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 0,)


###Standardisation###

# set up the scaler
scaler = StandardScaler()
# fit the scaler to the train set
scaler.fit(X_train)
# transform train and test sets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
# transform NumPy arrays to dataframes
X_train_scaled = pandas.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled = pandas.DataFrame(X_test_scaled, columns=X_test.columns)
# standardisation
X_train = X_train_scaled
X_test = X_test_scaled


### Getting class weight###

from sklearn.utils.class_weight import compute_class_weight
cw = compute_class_weight('balanced', np.unique(y_train), y_train)
class_weight = {c: w for c, w in zip(np.unique(y_train), cw)}


###Classifier###

classifier = GaussianNB()

classifier.fit(X_train, y_train)



###Evaluation###

# Gini function
def gini(y_true, y_pred, sample_weight=None):
    y_true = np.squeeze(y_true)
    y_pred = np.squeeze(y_pred)
    return 2.*roc_auc_score(y_true, y_pred, sample_weight=sample_weight) - 1



#Training Set

y_train_pred = classifier.predict(X_train)
y_train_pred_prob = classifier.predict_proba(X_train)
y_train_pred_adj = (y_train_pred_prob[:,1] >= 0.5).astype('int')

print('Confusion Matrix: ')
print(confusion_matrix(y_train, y_train_pred))
print()
print('Model Results: ')
print(classification_report(y_train,y_train_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_train,y_train_pred), 4))
print("Gini:",gini(y_train.astype(int), y_train_pred_prob[:,1]))
```

```
#Test Set

y_pred = classifier.predict(X_test)
y_pred_prob = classifier.predict_proba(X_test)
y_pred_adj = (y_pred_prob[:,1] >= 0.5).astype('int')

print('Confusion Matrix: ')
print(confusion_matrix(y_test,y_pred))
print()
print('Model Results: ')
print(classification_report(y_test,y_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_test, y_pred), 4))
print("Gini:",gini(y_test.astype(int), y_pred_prob[:,1]))




#########KNN - PCA#########

###import data###

url = "C:/Users/suered/Desktop/masters/MastersData.csv"

dataset = pandas.read_csv(url,low_memory=False)


###categorical variables - get_dummies###

X= dataset.drop('Default', axis=1)
Y = dataset['Default']
X_transformed = pandas.get_dummies(X, drop_first=True)
dataset = pandas.concat([Y,X_transformed],axis=1)


###PCA###

correlatedfeatures = ['Age',
'Instalment /Disposable income',
'YearsWithCurrentEmployer',
'% Total Taken Up',
'NumberOfDependents',
'Disposable income/Basic',
'Debt/Net Income',
'Max Offer',
'% instalment to income taken',
'% instalment to income allowed',
'External consolidations/Amount Taken',
'Internal consolidations/Amount Taken',
'Total consolidations/Amount Taken',
'Debt to Income ratio',
'Final Disposable Income/NetIncome',
'Calc Disposable Income/NetIncome',
]
xcorr = dataset.loc[:, correlatedfeatures].values
xcorr = StandardScaler().fit_transform(xcorr)
from sklearn.decomposition import PCA
pca = PCA(n_components=7)
principalComponents = pca.fit_transform(xcorr)

principalDf = pandas.DataFrame(data = principalComponents
             , columns = ['principalcomponent1'
                         ,'principalcomponent2'
                          ,'principalcomponent3'
                          ,'principalcomponent4'
                          ,'principalcomponent5'
                          ,'principalcomponent6'
                          ,'principalcomponent7'
                          ])

pca.explained_variance_ratio_
```

```
dataset.reset_index(drop=True, inplace=True)
principalDf.reset_index(drop=True, inplace=True)
dataset = pandas.concat([dataset,principalDf ],axis=1)

dataset.drop(['Age',
'Instalment /Disposable income',
'YearsWithCurrentEmployer',
'% Total Taken Up',
'NumberOfDependents',
'Disposable income/Basic',
'Debt/Net Income',
'Max Offer',
'% instalment to income taken',
'% instalment to income allowed',
'External consolidations/Amount Taken',
'Internal consolidations/Amount Taken',
'Total consolidations/Amount Taken',
'Debt to Income ratio',
'Final Disposable Income/NetIncome',
'Calc Disposable Income/NetIncome',
],axis=1,inplace=True)


###Splitting Data###

X = dataset.iloc[:, 1:]
Y = dataset.iloc[:, 0]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 0,)


###Standardisation###

# set up the scaler
scaler = StandardScaler()
# fit the scaler to the train set
scaler.fit(X_train)
# transform train and test sets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
# transform NumPy arrays to dataframes
X_train_scaled = pandas.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled = pandas.DataFrame(X_test_scaled, columns=X_test.columns)
# standardisation
X_train = X_train_scaled
X_test = X_test_scaled



###Smote###

from collections import Counter
from sklearn.datasets import make_classification
from imblearn.over_sampling import SMOTE
print(Counter(y_train))
# sampling strategy
sample = SMOTE()
X_train, y_train = sample.fit_resample(X_train, y_train)
print(Counter(y_train))


###Classifier###

classifier = KNeighborsClassifier(metric= 'manhattan',n_neighbors= 121, weights = 'uniform')
classifier.fit(X_train, y_train)


###Evaluation###

# Gini function
def gini(y_true, y_pred, sample_weight=None):
    y_true = np.squeeze(y_true)
    y_pred = np.squeeze(y_pred)
```

```
    return 2.*roc_auc_score(y_true, y_pred, sample_weight=sample_weight) - 1
#Training Set

y_train_pred = classifier.predict(X_train)
y_train_pred_prob = classifier.predict_proba(X_train)
y_train_pred_adj = (y_train_pred_prob[:,1] >= 0.5).astype('int')

print('Confusion Matrix: ')
print(confusion_matrix(y_train, y_train_pred))
print()
print('Model Results: ')
print(classification_report(y_train,y_train_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_train,y_train_pred), 4))
print("Gini:",gini(y_train.astype(int), y_train_pred_prob[:,1]))
#Test Set

y_pred = classifier.predict(X_test)
y_pred_prob = classifier.predict_proba(X_test)
y_pred_adj = (y_pred_prob[:,1] >= 0.5).astype('int')

print('Confusion Matrix: ')
print(confusion_matrix(y_test,y_pred))
print()
print('Model Results: ')
print(classification_report(y_test,y_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_test, y_pred), 4))
print("Gini:",gini(y_test.astype(int), y_pred_prob[:,1]))




#########ANN - PCA#########

###import data###

url = "C:/Users/suered/Desktop/masters/MastersData.csv"
dataset = pandas.read_csv(url,low_memory=False)


###categorical variables - get_dummies###

X= dataset.drop('Default', axis=1)
Y = dataset['Default']
X_transformed = pandas.get_dummies(X, drop_first=True)
dataset = pandas.concat([Y,X_transformed],axis=1)



###PCA###

correlatedfeatures = ['Age',
'Instalment /Disposable income',
'YearsWithCurrentEmployer',
'% Total Taken Up',
'NumberOfDependents',
'Disposable income/Basic',
'Debt/Net Income',
'Max Offer',
'% instalment to income taken',
'% instalment to income allowed',
'External consolidations/Amount Taken',
'Internal consolidations/Amount Taken',
'Total consolidations/Amount Taken',
'Debt to Income ratio',
'Final Disposable Income/NetIncome',
'Calc Disposable Income/NetIncome',
]
xcorr = dataset.loc[:, correlatedfeatures].values
xcorr = StandardScaler().fit_transform(xcorr)
from sklearn.decomposition import PCA
```

```
pca = PCA(n_components=7)
principalComponents = pca.fit_transform(xcorr)

principalDf = pandas.DataFrame(data = principalComponents
             , columns = ['principalcomponent1'
                         ,'principalcomponent2'
                          ,'principalcomponent3'
                          ,'principalcomponent4'
                          ,'principalcomponent5'
                          ,'principalcomponent6'
                          ,'principalcomponent7'
                          ])

pca.explained_variance_ratio_

dataset.reset_index(drop=True, inplace=True)
principalDf.reset_index(drop=True, inplace=True)
dataset = pandas.concat([dataset,principalDf ],axis=1)

dataset.drop(['Age',
'Instalment /Disposable income',
'YearsWithCurrentEmployer',
'% Total Taken Up',
'NumberOfDependents',
'Disposable income/Basic',
'Debt/Net Income',
'Max Offer',
'% instalment to income taken',
'% instalment to income allowed',
'External consolidations/Amount Taken',
'Internal consolidations/Amount Taken',
'Total consolidations/Amount Taken',
'Debt to Income ratio',
'Final Disposable Income/NetIncome',
'Calc Disposable Income/NetIncome',
],axis=1,inplace=True)


###Splitting Data###

X = dataset.iloc[:, 1:]
Y = dataset.iloc[:, 0]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 0,)


###Standardisation###

# set up the scaler
scaler = StandardScaler()
# fit the scaler to the train set
scaler.fit(X_train)
# transform train and test sets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
# transform NumPy arrays to dataframes
X_train_scaled = pandas.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled = pandas.DataFrame(X_test_scaled, columns=X_test.columns)
# standardisation
X_train = X_train_scaled
X_test = X_test_scaled



### Getting class weight###

from sklearn.utils.class_weight import compute_class_weight
cw = compute_class_weight('balanced', np.unique(y_train), y_train)
class_weight = {c: w for c, w in zip(np.unique(y_train), cw)}


###Classifier###
```

```
classifier = Sequential()
classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 'relu', input_dim
= 48))
classifier.add(Dense(units = 10, kernel_initializer = 'uniform', activation = 'relu'))
classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
classifier.fit(X_train, y_train, batch_size = 128, epochs = 200, class_weight=class_weight)


###Evaluation###

# Gini function

def gini(y_true, y_pred, sample_weight=None):
    y_true = np.squeeze(y_true)
    y_pred = np.squeeze(y_pred)
    return 2.*roc_auc_score(y_true, y_pred, sample_weight=sample_weight) - 1.
#Training Set
y_train_pred = classifier.predict(X_train)
y_train_pred = (y_train_pred > 0.5)

print('Confusion Matrix: ')
print(confusion_matrix(y_train, y_train_pred))
print()
print('Model Results: ')
print(classification_report(y_train,y_train_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_train,y_train_pred), 4))


#Test Set
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)

print('Confusion Matrix: ')
print(confusion_matrix(y_test,y_pred))
print()
print('Model Results: ')
print(classification_report(y_test,y_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_test, y_pred), 4))
```

Appendix 2 – Feature Selection

```
#########Packages#########

import pandas as pandas
from pandas.plotting import scatter_matrix
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
import scipy.stats
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
#from sklearn.datasets import make_classification
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import GridSearchCV
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.svm import SVC
import tensorflow
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.constraints import maxnorm
from keras.layers import Dropout
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import matthews_corrcoef, precision_score, accuracy_score, recall_score,
f1_score
from sklearn.metrics import roc_auc_score, r2_score
from sklearn.metrics import roc_curve




#########Logistic Regression – Feature Selection#########

###import data###

url = "C:/Users/suered/Desktop/masters/MastersData.csv"
dataset = pandas.read_csv(url,low_memory=False)


###categorical variables - get_dummies###
X= dataset.drop('Default', axis=1)
Y = dataset['Default']
X_transformed = pandas.get_dummies(X, drop_first=True)
dataset = pandas.concat([Y,X_transformed],axis=1)


###Splitting Data###
X = dataset.iloc[:, 1:]
Y = dataset.iloc[:, 0]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 0,)

###Standardisation###
# set up the scaler
```

```
scaler = StandardScaler()
# fit the scaler to the train set
scaler.fit(X_train)
# transform train and test sets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
# transform NumPy arrays to dataframes
X_train_scaled = pandas.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled = pandas.DataFrame(X_test_scaled, columns=X_test.columns)
# standardisation
X_train = X_train_scaled
X_test = X_test_scaled



### Getting class weight###
from sklearn.utils.class_weight import compute_class_weight
cw = compute_class_weight('balanced', np.unique(y_train), y_train)
class_weight = {c: w for c, w in zip(np.unique(y_train), cw)}
seed_val = 1000000000
np.random.seed(seed_val)


### feature selection###
### build initial model using all the features###
model_all_features = LogisticRegression(n_jobs=-1,
                                solver = 'newton-cg',
                                C = 10,
                                penalty = 'l2',
                                class_weight=class_weight,
                                random_state=0)

model_all_features.fit(X_train, y_train)
# calculate the roc-auc in the test set
y_pred_test = model_all_features.predict_proba(X_test)[:, 1]
auc_score_all = roc_auc_score(y_test, y_pred_test)
print('Test all features Random Forrest ROC AUC=%f' % (auc_score_all))

###determine order in which features are removed from model ###
feature_order   = RandomForestClassifier(criterion='gini',
                                bootstrap = True,
                                max_depth = 60,
                                max_features = 'auto',
                                min_samples_leaf = 100,
                                min_samples_split = 400,
                                n_estimators = 50,
                                n_jobs=-1,
                                oob_score=True,
                                class_weight=class_weight,
                                random_state=0)

feature_order.fit(X_train, y_train)

# obtaining feature names and importance
features = pandas.Series(feature_order.feature_importances_)
features.index = X_train.columns
# sorting features by importance
features.sort_values(ascending=True, inplace=True)
# plotting feature importance
features.plot.bar(figsize=(20,6))
# show list of ordered features
features = list(features.index)
features



###recursive feature elimination###
tol = 0.001
print('doing recursive feature elimination')
features_to_remove = []
count = 1
```

```
for feature in features:
    print()
    print('testing feature: ', feature, ' which is feature ', count,
          ' out of ', len(features))
    count = count + 1

# build model
    model_int = LogisticRegression(n_jobs=-1,
                                   solver = 'newton-cg',
                                   C = 10,
                                   penalty = 'l2',
                                   class_weight=class_weight,
                                   random_state=0)

    model_int.fit(
             X_train.drop(features_to_remove + [feature], axis=1), y_train)
    y_pred_test = model_int.predict_proba(
        X_test.drop(features_to_remove + [feature], axis=1))[:, 1]

    auc_score_int = roc_auc_score(y_test, y_pred_test)
    print('New Test ROC AUC={}'.format((auc_score_int)))
    print('All features Test ROC AUC={}'.format((auc_score_all)))

    diff_auc = auc_score_all - auc_score_int


    if diff_auc >= tol:
        print('Drop in ROC AUC={}'.format(diff_auc))
        print('keep: ', feature)
        print
    else:
        print('Drop in ROC AUC={}'.format(diff_auc))
        print('remove: ', feature)
        print

        auc_score_all = auc_score_int
        features_to_remove.append(feature)

print('End')


features_to_keep = [x for x in features if x not in features_to_remove]
print('total features to keep: ', len(features_to_keep))

###final model###
final_model = LogisticRegression(n_jobs=-1,
                                   solver = 'newton-cg',
                                   C = 10,
                                   penalty = 'l2',
                                   class_weight=class_weight,
                                   random_state=0)

final_model.fit(X_train[features_to_keep], y_train)
y_pred_test = final_model.predict_proba(X_test[features_to_keep])[:, 1]
auc_score_final = roc_auc_score(y_test, y_pred_test)
print('Test selected features ROC AUC=%f' % (auc_score_final))

### Gini function###
def gini(y_true, y_pred, sample_weight=None):
    y_true = np.squeeze(y_true)
    y_pred = np.squeeze(y_pred)
    return 2.*roc_auc_score(y_true, y_pred, sample_weight=sample_weight) - 1.

###Train###
y_train_pred = final_model.predict(X_train[features_to_keep])
y_train_pred_prob = final_model.predict_proba(X_train[features_to_keep])
y_train_pred_adj = (y_train_pred_prob[:,1] >= 0.5).astype('int')
print('Confusion Matrix: ')
print(confusion_matrix(y_train,y_train_pred))
print()
print('Model Results: ')
```

```
print(classification_report(y_train,y_train_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_train,y_train_pred), 4))
print("Gini:",gini(y_train.astype(int), y_train_pred_prob[:,1]))

###Test###
y_pred = final_model.predict(X_test[features_to_keep])
y_pred_prob = final_model.predict_proba(X_test[features_to_keep])
y_pred_adj = (y_pred_prob[:,1] >= 0.5).astype('int')
print('Confusion Matrix: ')
print(confusion_matrix(y_test,y_pred))
print()
print('Model Results: ')
print(classification_report(y_test,y_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_test, y_pred), 4))
print("Gini:",gini(y_test.astype(int), y_pred_prob[:,1]))




#########Decision Tree- Feature Selection#########

###import data###

url = "C:/Users/suered/Desktop/masters/MastersData.csv"
dataset = pandas.read_csv(url,low_memory=False)


###categorical variables - get_dummies###
X= dataset.drop('Default', axis=1)
Y = dataset['Default']
X_transformed = pandas.get_dummies(X, drop_first=True)
dataset = pandas.concat([Y,X_transformed],axis=1)


###Splitting Data###
X = dataset.iloc[:, 1:]
Y = dataset.iloc[:, 0]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 0,)


###Standardisation###
# set up the scaler
scaler = StandardScaler()
# fit the scaler to the train set
scaler.fit(X_train)
# transform train and test sets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
# transform NumPy arrays to dataframes
X_train_scaled = pandas.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled = pandas.DataFrame(X_test_scaled, columns=X_test.columns)
# standardisation
X_train = X_train_scaled
X_test = X_test_scaled



### Getting class weight###
from sklearn.utils.class_weight import compute_class_weight
cw = compute_class_weight('balanced', np.unique(y_train), y_train)
class_weight = {c: w for c, w in zip(np.unique(y_train), cw)}
seed_val = 1000000000
np.random.seed(seed_val)


### feature selection###
### build initial model using all the features###
model_all_features = DecisionTreeClassifier(criterion='entropy',
                                max_depth = 60,
```

```
                                max_features = 'auto',
                                min_samples_leaf = 200,
                                splitter = 'best',
                                min_impurity_split = 0.6,
                                class_weight=class_weight,
                                random_state=0)

model_all_features.fit(X_train, y_train)
# calculate the roc-auc in the test set
y_pred_test = model_all_features.predict_proba(X_test)[:, 1]
auc_score_all = roc_auc_score(y_test, y_pred_test)
print('Test all features Random Forrest ROC AUC=%f' % (auc_score_all))

###determine order in which features are removed from model ###
feature_order   = RandomForestClassifier(criterion='gini',
                                bootstrap = True,
                                max_depth = 60,
                                max_features = 'auto',
                                min_samples_leaf = 100,
                                min_samples_split = 400,
                                n_estimators = 50,
                                n_jobs=-1,
                                oob_score=True,
                                class_weight=class_weight,
                                random_state=0)

feature_order.fit(X_train, y_train)

# obtaining feature names and importance
features = pandas.Series(feature_order.feature_importances_)
features.index = X_train.columns
# sorting features by importance
features.sort_values(ascending=True, inplace=True)
# plotting feature importance
features.plot.bar(figsize=(20,6))
# show list of ordered features
features = list(features.index)
features
###recursive feature elimination###
tol = 0.001
print('doing recursive feature elimination')
features_to_remove = []
count = 1
for feature in features:
    print()
    print('testing feature: ', feature, ' which is feature ', count,
          ' out of ', len(features))
    count = count + 1

# build model
    model_int = DecisionTreeClassifier(criterion='entropy',
                                max_depth = 60,
                                max_features = 'auto',
                                min_samples_leaf = 200,
                                splitter = 'best',
                                min_impurity_split = 0.6,
                                class_weight=class_weight,
                                random_state=0)

    model_int.fit(
             X_train.drop(features_to_remove + [feature], axis=1), y_train)
    y_pred_test = model_int.predict_proba(
        X_test.drop(features_to_remove + [feature], axis=1))[:, 1]

    auc_score_int = roc_auc_score(y_test, y_pred_test)
    print('New Test ROC AUC={}'.format((auc_score_int)))
    print('All features Test ROC AUC={}'.format((auc_score_all)))

    diff_auc = auc_score_all - auc_score_int
```

```
    if diff_auc >= tol:
        print('Drop in ROC AUC={}'.format(diff_auc))
        print('keep: ', feature)
        print
    else:
        print('Drop in ROC AUC={}'.format(diff_auc))
        print('remove: ', feature)
        print

        auc_score_all = auc_score_int
        features_to_remove.append(feature)

print('End')


features_to_keep = [x for x in features if x not in features_to_remove]
print('total features to keep: ', len(features_to_keep))

###final model###
final_model = DecisionTreeClassifier(criterion='entropy',
                                max_depth = 60,
                                max_features = 'auto',
                                min_samples_leaf = 200,
                                splitter = 'best',
                                min_impurity_split = 0.6,
                                class_weight=class_weight,
                                random_state=0)

final_model.fit(X_train[features_to_keep], y_train)
y_pred_test = final_model.predict_proba(X_test[features_to_keep])[:, 1]
auc_score_final = roc_auc_score(y_test, y_pred_test)
print('Test selected features ROC AUC=%f' % (auc_score_final))

### Gini function###
def gini(y_true, y_pred, sample_weight=None):
    y_true = np.squeeze(y_true)
    y_pred = np.squeeze(y_pred)
    return 2.*roc_auc_score(y_true, y_pred, sample_weight=sample_weight) - 1.

###Train###
y_train_pred = final_model.predict(X_train[features_to_keep])
y_train_pred_prob = final_model.predict_proba(X_train[features_to_keep])
y_train_pred_adj = (y_train_pred_prob[:,1] >= 0.5).astype('int')
print('Confusion Matrix: ')
print(confusion_matrix(y_train,y_train_pred))
print()
print('Model Results: ')
print(classification_report(y_train,y_train_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_train,y_train_pred), 4))
print("Gini:",gini(y_train.astype(int), y_train_pred_prob[:,1]))

###Test###
y_pred = final_model.predict(X_test[features_to_keep])
y_pred_prob = final_model.predict_proba(X_test[features_to_keep])
y_pred_adj = (y_pred_prob[:,1] >= 0.5).astype('int')
print('Confusion Matrix: ')
print(confusion_matrix(y_test,y_pred))
print()
print('Model Results: ')
print(classification_report(y_test,y_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_test, y_pred), 4))
print("Gini:",gini(y_test.astype(int), y_pred_prob[:,1]))
```

```
#########Random Forest- Feature Selection#########

###import data###

url = "C:/Users/suered/Desktop/masters/MastersData.csv"
dataset = pandas.read_csv(url,low_memory=False)


###categorical variables - get_dummies###
X= dataset.drop('Default', axis=1)
Y = dataset['Default']
X_transformed = pandas.get_dummies(X, drop_first=True)
dataset = pandas.concat([Y,X_transformed],axis=1)


###Splitting Data###
X = dataset.iloc[:, 1:]
Y = dataset.iloc[:, 0]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 0,)


###Standardisation###
# set up the scaler
scaler = StandardScaler()
# fit the scaler to the train set
scaler.fit(X_train)
# transform train and test sets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
# transform NumPy arrays to dataframes
X_train_scaled = pandas.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled = pandas.DataFrame(X_test_scaled, columns=X_test.columns)
# standardisation
X_train = X_train_scaled
X_test = X_test_scaled



### Getting class weight###
from sklearn.utils.class_weight import compute_class_weight
cw = compute_class_weight('balanced', np.unique(y_train), y_train)
class_weight = {c: w for c, w in zip(np.unique(y_train), cw)}
seed_val = 1000000000
np.random.seed(seed_val)


### feature selection###
### build initial model using all the features###
model_all_features = RandomForestClassifier(criterion='gini',
                                bootstrap = True,
                                max_depth = 40,
                                max_features = 'auto',
                                min_samples_leaf = 50,
                                min_samples_split = 550,
                                n_estimators = 50,
                                n_jobs=-1,
                                oob_score=True,
                                class_weight=class_weight,
                                random_state=0)

model_all_features.fit(X_train, y_train)
# calculate the roc-auc in the test set
y_pred_test = model_all_features.predict_proba(X_test)[:, 1]
auc_score_all = roc_auc_score(y_test, y_pred_test)
print('Test all features Random Forrest ROC AUC=%f' % (auc_score_all))

###determine order in which features are removed from model ###
feature_order   = RandomForestClassifier(criterion='gini',
                                bootstrap = True,
                                max_depth = 40,
                                max_features = 'auto',
```

```
                                min_samples_leaf = 50,
                                min_samples_split = 550,
                                n_estimators = 50,
                                n_jobs=-1,
                                oob_score=True,
                                class_weight=class_weight,
                                random_state=0)

feature_order.fit(X_train, y_train)

# obtaining feature names and importance
features = pandas.Series(feature_order.feature_importances_)
features.index = X_train.columns
# sorting features by importance
features.sort_values(ascending=True, inplace=True)
# plotting feature importance
features.plot.bar(figsize=(20,6))
# show list of ordered features
features = list(features.index)
features



###recursive feature elimination###
tol = 0.001
print('doing recursive feature elimination')
features_to_remove = []
count = 1
for feature in features:
    print()
    print('testing feature: ', feature, ' which is feature ', count,
          ' out of ', len(features))
    count = count + 1

# build model
    model_int = RandomForestClassifier(criterion='gini',
                                bootstrap = True,
                                max_depth = 40,
                                max_features = 'auto',
                                min_samples_leaf = 50,
                                min_samples_split = 550,
                                n_estimators = 50,
                                n_jobs=-1,
                                oob_score=True,
                                class_weight=class_weight,
                                random_state=0)

    model_int.fit(
             X_train.drop(features_to_remove + [feature], axis=1), y_train)
    y_pred_test = model_int.predict_proba(
        X_test.drop(features_to_remove + [feature], axis=1))[:, 1]

    auc_score_int = roc_auc_score(y_test, y_pred_test)
    print('New Test ROC AUC={}'.format((auc_score_int)))
    print('All features Test ROC AUC={}'.format((auc_score_all)))

    diff_auc = auc_score_all - auc_score_int


    if diff_auc >= tol:
        print('Drop in ROC AUC={}'.format(diff_auc))
        print('keep: ', feature)
        print
    else:
        print('Drop in ROC AUC={}'.format(diff_auc))
        print('remove: ', feature)
        print

        auc_score_all = auc_score_int
        features_to_remove.append(feature)
```

```
print('End')


features_to_keep = [x for x in features if x not in features_to_remove]
print('total features to keep: ', len(features_to_keep))

###final model###
final_model = RandomForestClassifier(criterion='gini',
                                     bootstrap = True,
                                     max_depth = 40,
                                     max_features = 'auto',
                                     min_samples_leaf = 50,
                                     min_samples_split = 550,
                                     n_estimators = 50,
                                     n_jobs=-1,
                                     oob_score=True,
                                     class_weight=class_weight,
                                     random_state=0)

final_model.fit(X_train[features_to_keep], y_train)
y_pred_test = final_model.predict_proba(X_test[features_to_keep])[:, 1]
auc_score_final = roc_auc_score(y_test, y_pred_test)
print('Test selected features ROC AUC=%f' % (auc_score_final))


### Gini function###
def gini(y_true, y_pred, sample_weight=None):
    y_true = np.squeeze(y_true)
    y_pred = np.squeeze(y_pred)
    return 2.*roc_auc_score(y_true, y_pred, sample_weight=sample_weight) - 1.

###Train###
y_train_pred = final_model.predict(X_train[features_to_keep])
y_train_pred_prob = final_model.predict_proba(X_train[features_to_keep])
y_train_pred_adj = (y_train_pred_prob[:,1] >= 0.5).astype('int')
print('Confusion Matrix: ')
print(confusion_matrix(y_train,y_train_pred))
print()
print('Model Results: ')
print(classification_report(y_train,y_train_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_train,y_train_pred), 4))
print("Gini:",gini(y_train.astype(int), y_train_pred_prob[:,1]))

###Test###
y_pred = final_model.predict(X_test[features_to_keep])
y_pred_prob = final_model.predict_proba(X_test[features_to_keep])
y_pred_adj = (y_pred_prob[:,1] >= 0.5).astype('int')
print('Confusion Matrix: ')
print(confusion_matrix(y_test,y_pred))
print()
print('Model Results: ')
print(classification_report(y_test,y_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_test, y_pred), 4))
print("Gini:",gini(y_test.astype(int), y_pred_prob[:,1]))




#########SVM- Feature Selection#########

###import data###

url = "C:/Users/suered/Desktop/masters/MastersData.csv"
dataset = pandas.read_csv(url,low_memory=False)


###categorical variables - get_dummies###
X= dataset.drop('Default', axis=1)
Y = dataset['Default']
```

```
X_transformed = pandas.get_dummies(X, drop_first=True)
dataset = pandas.concat([Y,X_transformed],axis=1)


###Splitting Data###
X = dataset.iloc[:, 1:]
Y = dataset.iloc[:, 0]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 0,)


###Standardisation###
# set up the scaler
scaler = StandardScaler()
# fit the scaler to the train set
scaler.fit(X_train)
# transform train and test sets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
# transform NumPy arrays to dataframes
X_train_scaled = pandas.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled = pandas.DataFrame(X_test_scaled, columns=X_test.columns)
# standardisation
X_train = X_train_scaled
X_test = X_test_scaled



### Getting class weight###
from sklearn.utils.class_weight import compute_class_weight
cw = compute_class_weight('balanced', np.unique(y_train), y_train)
class_weight = {c: w for c, w in zip(np.unique(y_train), cw)}
seed_val = 1000000000
np.random.seed(seed_val)


### feature selection###
### build initial model using all the features###
model_all_features = SVC(kernel = 'rbf',
                 gamma = 0.005,
                 C = 21,
                 class_weight = 'balanced',
                 probability=True)

model_all_features.fit(X_train, y_train)
# calculate the roc-auc in the test set
y_pred_test = model_all_features.predict_proba(X_test)[:, 1]
auc_score_all = roc_auc_score(y_test, y_pred_test)
print('Test all features Random Forrest ROC AUC=%f' % (auc_score_all))

###determine order in which features are removed from model ###
feature_order   = RandomForestClassifier(criterion='gini',
                                 bootstrap = True,
                                 max_depth = 60,
                                 max_features = 'auto',
                                 min_samples_leaf = 100,
                                 min_samples_split = 400,
                                 n_estimators = 50,
                                 n_jobs=-1,
                                 oob_score=True,
                                 class_weight=class_weight,
                                 random_state=0)

feature_order.fit(X_train, y_train)

# obtaining feature names and importance
features = pandas.Series(feature_order.feature_importances_)
features.index = X_train.columns
# sorting features by importance
features.sort_values(ascending=True, inplace=True)
# plotting feature importance
```

```
features.plot.bar(figsize=(20,6))
# show list of ordered features
features = list(features.index)
features



###recursive feature elimination###
tol = 0.001
print('doing recursive feature elimination')
features_to_remove = []
count = 1
for feature in features:
    print()
    print('testing feature: ', feature, ' which is feature ', count,
          ' out of ', len(features))
    count = count + 1

# build model
    model_int = SVC(kernel = 'rbf',
                gamma = 0.005,
                C = 21,
                class_weight = 'balanced',
                probability=True)

    model_int.fit(
             X_train.drop(features_to_remove + [feature], axis=1), y_train)
    y_pred_test = model_int.predict_proba(
        X_test.drop(features_to_remove + [feature], axis=1))[:, 1]

    auc_score_int = roc_auc_score(y_test, y_pred_test)
    print('New Test ROC AUC={}'.format((auc_score_int)))
    print('All features Test ROC AUC={}'.format((auc_score_all)))

    diff_auc = auc_score_all - auc_score_int


    if diff_auc >= tol:
        print('Drop in ROC AUC={}'.format(diff_auc))
        print('keep: ', feature)
        print
    else:
        print('Drop in ROC AUC={}'.format(diff_auc))
        print('remove: ', feature)
        print

        auc_score_all = auc_score_int
        features_to_remove.append(feature)

print('End')


features_to_keep = [x for x in features if x not in features_to_remove]
print('total features to keep: ', len(features_to_keep))

###final model###
final_model = SVC(kernel = 'rbf',
                gamma = 0.005,
                C = 21,
                class_weight = 'balanced',
                probability=True)

final_model.fit(X_train[features_to_keep], y_train)
y_pred_test = final_model.predict_proba(X_test[features_to_keep])[:, 1]
auc_score_final = roc_auc_score(y_test, y_pred_test)
print('Test selected features ROC AUC=%f' % (auc_score_final))


### Gini function###
def gini(y_true, y_pred, sample_weight=None):
    y_true = np.squeeze(y_true)
```

```
    y_pred = np.squeeze(y_pred)
    return 2.*roc_auc_score(y_true, y_pred, sample_weight=sample_weight) - 1.

###Train###
y_train_pred = final_model.predict(X_train[features_to_keep])
y_train_pred_prob = final_model.predict_proba(X_train[features_to_keep])
y_train_pred_adj = (y_train_pred_prob[:,1] >= 0.5).astype('int')
print('Confusion Matrix: ')
print(confusion_matrix(y_train,y_train_pred))
print()
print('Model Results: ')
print(classification_report(y_train,y_train_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_train,y_train_pred), 4))
print("Gini:",gini(y_train.astype(int), y_train_pred_prob[:,1]))

###Test###
y_pred = final_model.predict(X_test[features_to_keep])
y_pred_prob = final_model.predict_proba(X_test[features_to_keep])
y_pred_adj = (y_pred_prob[:,1] >= 0.5).astype('int')
print('Confusion Matrix: ')
print(confusion_matrix(y_test,y_pred))
print()
print('Model Results: ')
print(classification_report(y_test,y_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_test, y_pred), 4))
print("Gini:",gini(y_test.astype(int), y_pred_prob[:,1]))



#########NB - Feature Selection#########

###import data###

url = "C:/Users/suered/Desktop/masters/MastersData.csv"
dataset = pandas.read_csv(url,low_memory=False)


###categorical variables - get_dummies###
X= dataset.drop('Default', axis=1)
Y = dataset['Default']
X_transformed = pandas.get_dummies(X, drop_first=True)
dataset = pandas.concat([Y,X_transformed],axis=1)


###Splitting Data###
X = dataset.iloc[:, 1:]
Y = dataset.iloc[:, 0]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 0,)


###Standardisation###
# set up the scaler
scaler = StandardScaler()
# fit the scaler to the train set
scaler.fit(X_train)
# transform train and test sets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
# transform NumPy arrays to dataframes
X_train_scaled = pandas.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled = pandas.DataFrame(X_test_scaled, columns=X_test.columns)
# standardisation
X_train = X_train_scaled
X_test = X_test_scaled
```

```
### Getting class weight###
from sklearn.utils.class_weight import compute_class_weight
cw = compute_class_weight('balanced', np.unique(y_train), y_train)
class_weight = {c: w for c, w in zip(np.unique(y_train), cw)}
seed_val = 1000000000
np.random.seed(seed_val)


### feature selection###
### build initial model using all the features###
model_all_features = GaussianNB()
model_all_features.fit(X_train, y_train)
# calculate the roc-auc in the test set
y_pred_test = model_all_features.predict_proba(X_test)[:, 1]
auc_score_all = roc_auc_score(y_test, y_pred_test)
print('Test all features Random Forrest ROC AUC=%f' % (auc_score_all))

###determine order in which features are removed from model ###
feature_order   = RandomForestClassifier(criterion='gini',
                                bootstrap = True,
                                max_depth = 60,
                                max_features = 'auto',
                                min_samples_leaf = 100,
                                min_samples_split = 400,
                                n_estimators = 50,
                                n_jobs=-1,
                                oob_score=True,
                                class_weight=class_weight,
                                random_state=0)

feature_order.fit(X_train, y_train)

# obtaining feature names and importance
features = pandas.Series(feature_order.feature_importances_)
features.index = X_train.columns
# sorting features by importance
features.sort_values(ascending=True, inplace=True)
# plotting feature importance
features.plot.bar(figsize=(20,6))
# show list of ordered features
features = list(features.index)
features



###recursive feature elimination###
tol = 0.001
print('doing recursive feature elimination')
features_to_remove = []
count = 1
for feature in features:
    print()
    print('testing feature: ', feature, ' which is feature ', count,
          ' out of ', len(features))
    count = count + 1

# build model
    model_int = GaussianNB()
    model_int.fit(
             X_train.drop(features_to_remove + [feature], axis=1), y_train)
    y_pred_test = model_int.predict_proba(
        X_test.drop(features_to_remove + [feature], axis=1))[:, 1]

    auc_score_int = roc_auc_score(y_test, y_pred_test)
    print('New Test ROC AUC={}'.format((auc_score_int)))
    print('All features Test ROC AUC={}'.format((auc_score_all)))
    diff_auc = auc_score_all - auc_score_int


    if diff_auc >= tol:
        print('Drop in ROC AUC={}'.format(diff_auc))
```

```
        print('keep: ', feature)
        print
    else:
        print('Drop in ROC AUC={}'.format(diff_auc))
        print('remove: ', feature)
        print

        auc_score_all = auc_score_int
        features_to_remove.append(feature)

print('End')


features_to_keep = [x for x in features if x not in features_to_remove]
print('total features to keep: ', len(features_to_keep))

###final model###
final_model = GaussianNB()
final_model.fit(X_train[features_to_keep], y_train)
y_pred_test = final_model.predict_proba(X_test[features_to_keep])[:, 1]
auc_score_final = roc_auc_score(y_test, y_pred_test)
print('Test selected features ROC AUC=%f' % (auc_score_final))

### Gini function###
def gini(y_true, y_pred, sample_weight=None):
    y_true = np.squeeze(y_true)
    y_pred = np.squeeze(y_pred)
    return 2.*roc_auc_score(y_true, y_pred, sample_weight=sample_weight) - 1.

###Train###
y_train_pred = final_model.predict(X_train[features_to_keep])
y_train_pred_prob = final_model.predict_proba(X_train[features_to_keep])
y_train_pred_adj = (y_train_pred_prob[:,1] >= 0.5).astype('int')
print('Confusion Matrix: ')
print(confusion_matrix(y_train,y_train_pred))
print()
print('Model Results: ')
print(classification_report(y_train,y_train_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_train,y_train_pred), 4))
print("Gini:",gini(y_train.astype(int), y_train_pred_prob[:,1]))

###Test###
y_pred = final_model.predict(X_test[features_to_keep])
y_pred_prob = final_model.predict_proba(X_test[features_to_keep])
y_pred_adj = (y_pred_prob[:,1] >= 0.5).astype('int')
print('Confusion Matrix: ')
print(confusion_matrix(y_test,y_pred))
print()
print('Model Results: ')
print(classification_report(y_test,y_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_test, y_pred), 4))
print("Gini:",gini(y_test.astype(int), y_pred_prob[:,1]))



#########KNN - Feature Selection#########

###import data###

url = "C:/Users/suered/Desktop/masters/MastersData.csv"
dataset = pandas.read_csv(url,low_memory=False)


###categorical variables - get_dummies###
X= dataset.drop('Default', axis=1)
Y = dataset['Default']
X_transformed = pandas.get_dummies(X, drop_first=True)
dataset = pandas.concat([Y,X_transformed],axis=1)
```

```
###Splitting Data###
X = dataset.iloc[:, 1:]
Y = dataset.iloc[:, 0]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state = 0,)


###Standardisation###
# set up the scaler
scaler = StandardScaler()
# fit the scaler to the train set
scaler.fit(X_train)
# transform train and test sets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
# transform NumPy arrays to dataframes
X_train_scaled = pandas.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled = pandas.DataFrame(X_test_scaled, columns=X_test.columns)
# standardisation
X_train = X_train_scaled
X_test = X_test_scaled



### Getting class weight###
from sklearn.utils.class_weight import compute_class_weight
cw = compute_class_weight('balanced', np.unique(y_train), y_train)
class_weight = {c: w for c, w in zip(np.unique(y_train), cw)}
seed_val = 1000000000
np.random.seed(seed_val)


###SMOTE###
from collections import Counter
from sklearn.datasets import make_classification
from imblearn.over_sampling import SMOTE
print(Counter(y_train))
# sampling strategy
sample = SMOTE()
X_train, y_train = sample.fit_resample(X_train, y_train)
print(Counter(y_train))
### feature selection###
### build initial model using all the features###
model_all_features = KNeighborsClassifier(metric= 'manhattan',n_neighbors= 121, weights =
'uniform')

model_all_features.fit(X_train, y_train)
# calculate the roc-auc in the test set
y_pred_test = model_all_features.predict_proba(X_test)[:, 1]
auc_score_all = roc_auc_score(y_test, y_pred_test)
print('Test all features Random Forrest ROC AUC=%f' % (auc_score_all))

###determine order in which features are removed from model ###
feature_order   = RandomForestClassifier(criterion='gini',
                                bootstrap = True,
                                max_depth = 60,
                                max_features = 'auto',
                                min_samples_leaf = 100,
                                min_samples_split = 400,
                                n_estimators = 50,
                                n_jobs=-1,
                                oob_score=True,
                                class_weight=class_weight,
                                random_state=0)
feature_order.fit(X_train, y_train)

# obtaining feature names and importance
features = pandas.Series(feature_order.feature_importances_)
features.index = X_train.columns
# sorting features by importance
features.sort_values(ascending=True, inplace=True)
```

```
# plotting feature importance
features.plot.bar(figsize=(20,6))
# show list of ordered features
features = list(features.index)
features



###recursive feature elimination###
tol = 0.001
print('doing recursive feature elimination')
features_to_remove = []
count = 1
for feature in features:
    print()
    print('testing feature: ', feature, ' which is feature ', count,
          ' out of ', len(features))
    count = count + 1

# build model
    model_int = KNeighborsClassifier(metric= 'manhattan',n_neighbors= 121, weights =
'uniform')

    model_int.fit(
             X_train.drop(features_to_remove + [feature], axis=1), y_train)
    y_pred_test = model_int.predict_proba(
        X_test.drop(features_to_remove + [feature], axis=1))[:, 1]

    auc_score_int = roc_auc_score(y_test, y_pred_test)
    print('New Test ROC AUC={}'.format((auc_score_int)))
    print('All features Test ROC AUC={}'.format((auc_score_all)))

    diff_auc = auc_score_all - auc_score_int


    if diff_auc >= tol:
        print('Drop in ROC AUC={}'.format(diff_auc))
        print('keep: ', feature)
        print
    else:
        print('Drop in ROC AUC={}'.format(diff_auc))
        print('remove: ', feature)
        print

        auc_score_all = auc_score_int
        features_to_remove.append(feature)

print('End')

features_to_keep = [x for x in features if x not in features_to_remove]
print('total features to keep: ', len(features_to_keep))

###final model###
final_model = KNeighborsClassifier(metric= 'manhattan',n_neighbors= 121, weights = 'uniform')
final_model.fit(X_train[features_to_keep], y_train)
y_pred_test = final_model.predict_proba(X_test[features_to_keep])[:, 1]
auc_score_final = roc_auc_score(y_test, y_pred_test)
print('Test selected features ROC AUC=%f' % (auc_score_final))

### Gini function###
def gini(y_true, y_pred, sample_weight=None):
    y_true = np.squeeze(y_true)
    y_pred = np.squeeze(y_pred)
    return 2.*roc_auc_score(y_true, y_pred, sample_weight=sample_weight) - 1.
###Train###
y_train_pred = final_model.predict(X_train[features_to_keep])
y_train_pred_prob = final_model.predict_proba(X_train[features_to_keep])
y_train_pred_adj = (y_train_pred_prob[:,1] >= 0.5).astype('int')
print('Confusion Matrix: ')
print(confusion_matrix(y_train,y_train_pred))
print()
```

```
print('Model Results: ')
print(classification_report(y_train,y_train_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_train,y_train_pred), 4))
print("Gini:",gini(y_train.astype(int), y_train_pred_prob[:,1]))

###Test###
y_pred = final_model.predict(X_test[features_to_keep])
y_pred_prob = final_model.predict_proba(X_test[features_to_keep])
y_pred_adj = (y_pred_prob[:,1] >= 0.5).astype('int')
print('Confusion Matrix: ')
print(confusion_matrix(y_test,y_pred))
print()
print('Model Results: ')
print(classification_report(y_test,y_pred))
print('Model Accuracy: ')
print(round(accuracy_score(y_test, y_pred), 4))
print("Gini:",gini(y_test.astype(int), y_pred_prob[:,1]))
```