Quantification of credit risk models using machine learning and statistical analysis

by

Seshni Govender (14569337)

submitted in accordance with the requirements for the degree of

Master of Commerce

in the subject

Quantitative Management

at the

UNIVERSITY OF SOUTH AFRICA

SUPERVISOR: DR MB SEITSHIRO

CO-SUPERVISOR: PROF CJ SWANEPOEL

May 21, 2024

DECLARATION

Name: Seshni Govender

Student number: 14569337

Degree: Master of Commerce in Quantitative Management

Quantification of credit risk models using machine learning and statistical analysis.

I declare that the above dissertation is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references.

I further declare that I submitted the dissertation to originality checking software and that it falls within the accepted requirements for originality.

I further declare that I have not previously submitted this work, or part of it, for examination at Unisa for another qualification or at any other higher education institution.

SIGNATURE

21-05-2024

DATE

Acknowledgements

I extend my deepest gratitude to Dr. MB Seitshiro, whose dedication and unwavering support have been instrumental in the successful completion of this research. Dr. Seitshiro's passion and enthusiasm for model risk have not only been a source of inspiration but have also significantly contributed to my personal and academic growth. His guidance, patience, and expert advice have been invaluable throughout this journey. I am immensely thankful for his willingness to share his vast knowledge and for his continuous encouragement and assistance.

I also extend my heartfelt thanks to Prof. CJ Swanepoel for his meticulous attention to detail and invaluable assistance throughout this process. Prof. Swanepoel's expertise and insightful feedback have greatly enhanced the quality of my work.

Abstract

The main aim of this study is to compare machine learning models with traditional statistical models in predicting credit risk for a commercial bank. Furthermore, the evaluation is conducted on varying levels of data balancing to determine the impact of data balancing on the performance of the models under study. The Logistic Regression is considered the statistical baseline model, while the machine learning techniques in relation to the literature reviewed are k-NN, SVM, Decision Tree, MLP, and RBFNN. Logistic Regression showed consistent AUC values around 0,72, while SVM excelled at higher balance levels with an AUC of 0,73. The MLP model was superior in a fully balanced dataset, achieving a 0,78 AUC. However, Decision Tree and k-NN's performance varied with dataset balance, and RBFNN underperformed. The analysis concludes that no single model is universally superior. Therefore, the choice of credit risk models by financial institutions should be based on the specifics of the data and predictive requirements, considering prediction errors' financial impacts.

Keywords · Credit Risk · Machine Learning · Data Imbalance · Logistic Regression · SVM · MLP · k-NN · Decision Tree · RBFNN · AUC · Predictive Analytics · Chi-square Statistics · SMOTE

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
AUC	Area Under the Curve
AUC-H	Area Under the H Curve
AUC-ROC	Area Under the Receiver Operating Characteristic Curve
AUROC	Area Under Receiver Operating Characteristic
CART	Classification and Regression Tree
CFS	Correlation-Based Feature Selection
CNN	Convolutional Neural Network
FCBF	Fast Correlation-Based Filter
FDA	Fischer Discriminant Analysis
FN	False Negatives
FP	False Positives
FOA	Fruit Fly Optimization Algorithm
GRU	Gated Recurrent Unit
IG	Information Gain
IV	Information Value
k**-NN**	k-Nearest Neighbours
K-S	Kolmogorov-Smirnov
LASSO	Least Absolute Shrinkage and Selection Operator
LDA	Linear Discriminant Analysis
LSTM	Long Short-Term Memory
MLE	Maximum Likelihood Estimation

MLP	Multi-Layer Perceptron
MOE	Mixture-of-Experts
mRmR	Minimum Redundancy, Maximum Relevance
NN	Neural Network
OLS	Ordinary Least-squares
PCC	Percentage Correctly Classified
QDA	Quadratic Discriminant Analysis
RBF	Radial Basis Function
RBFNN	Radial Basis Function Neural Network
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
ROS	Random Over-sampling
RUS	Random Under-sampling
SMOTE	Synthetic Minority Over-sampling Technique
SMOTEN	Synthetic Minority Over-sampling Technique – Nominal
SU	Symmetrical Uncertainty
SVM	Support Vector Machines
Tanh	Hyperbolic Tangent Function
TN	True Negatives
TP	True Positives
WoE	Weight of Evidence
VDM	Value Difference Metric

Chapter 1

Introduction

This chapter introduces the study by providing background context, reviewing relevant literature, stating the problem being addressed, outlining the study's objectives, and describing the overall structure of the study.

1.1 Background

Credit risk poses a significant concern in commercial banking as it directly impacts the stability and performance of a developing country's economic growth. This risk revolves around the occurrence of a *default event*, which happens when a debtor fails to fulfill the legal obligations outlined in the debt contract (Zhang, 2009). To mitigate the risks associated with credit, it is essential to analyse customer behavior beforehand. The global financial crisis that began in 2007 prompted a reevaluation of previous credit risk management methods to minimise potential losses and allocate resources effectively (Mačerinskienė et al., 2014). Credit risk assessment is crucial for banks as it assists in determining the likelihood of a borrower defaulting on a loan. This is crucial for banks, as loan defaults can have serious financial consequences. To evaluate credit risk, banks commonly examine a borrower's credit history, financial records, and pertinent details. Additionally, factors like the borrower's employment track record, debt-to-income ratio, and general financial standing are taken into account. Based on this information, banks can assign a credit score to the borrower, which helps to indicate the level of risk involved in approving a loan. Through careful evaluation of credit risk, banks can make well-informed decisions regarding which borrowers to extend loans to and under what conditions, thereby reducing the likelihood of defaults and safeguarding the financial stability of the institution.

Banks and other financial firms typically use a quantitative model for credit risk assessment. A *quantitative model* is a system of variables. It is formulated based on a set of assumptions and relies on statistical or mathematical theories. *Quantitative model risk* arises when a model fails to perform as required and produces the incorrect outcome.

Evaluating a quantitative credit risk model in the banking industry is important for a number of reasons according to Hull and Suo (2002). Firstly, a model that functions correctly helps financial institutions to accurately assess the likelihood that a borrower will default on a loan. Secondly, a robust credit risk model enables banks to detect potential credit risks and implement suitable measures to reduce those risks.. This can include implementing risk management strategies such as setting higher interest rates on higher risk loans or requiring collateral (an asset pledged as security for the repayment of credit). Banks are also often required to reserve funds for risk arising from inadequate quantitative models. Thirdly, evaluating a credit risk model allows banks to ensure that the model is accurately reflecting the current credit risk environment. This is important because credit risk can change over time, and a model that is not regularly evaluated and updated may not accurately reflect the current risk profile of borrowers.

Machine learning, particularly Neural Networks (NN) and Artificial Intelligence (AI), have emerged as powerful tools in credit risk modelling. Machine learning algorithms can effectively handle large and diverse datasets, identify complex patterns, and improve predictive accuracy. Algorithms such as Decision Trees, Random Forests, Support Vector Machines (SVM), and NNs are commonly employed in credit risk modelling (Shi et al., 2022, Bastos, 2022, Bhoge, 2019, Fantazzini and Figini, 2022, Moula et al., 2017, Liu and Huang, 2020). These algorithms enable the inclusion of multiple variables and capture nonlinear relationships that may be missed by traditional models (Shi et al., 2022). However, these methods can be computationally costly, complex and often lack transparency (interpretability and explainability) (Zednik and Boelsen, 2021). Interpretability and explainability of credit risk models are crucial for building stakeholder trust and understanding in the banking sector.

Addressing data imbalance is another important aspect to consider when modelling credit risk using machine learning and statistical models. Data imbalance occurs when the difference in the number of observations across different classes (e.g., default and non-default) is significantly uneven, leading to biased model performance (Alam et al., 2020). In credit risk modelling, data imbalance is a common issue as defaults are relatively rare compared to non-default cases (García and Sánchez, 2013). This imbalance presents a challenge for machine learning algorithms, as they tend to prioritise the majority class, leading to potential difficulties in accurately predicting the minority class (Rawat and Mishra, 2022). Consequently, models may exhibit high overall accuracy while demonstrating limited proficiency in identifying default cases.

This research is sought to explore an opportunity to enhance accuracy, improve risk management practices, and make more informed decisions in the banking sector. By overcoming the constraints of traditional models, leveraging machine learning algorithms, and considering interpretability, data imbalance, model validation and risk quantification, this research aims to contribute to the advancement of credit risk modelling and provide insights for practitioners and researchers in their pursuit of effective credit model risk assessment and management strategies.

1.2 Literature Review

In a study by Yap et al. (2011), the application of the Weight of Evidence (WoE) technique, Credit Scorecard Model, Logistic Regression, and Decision Trees was explored for credit scoring. The researchers employed WoE as a means to transform independent variables, with the aim of enhancing the predictive power of the models. The results showed that no model performs better than the other with accuracy rates of approximately 72% for all models, however the Credit Scorecard Model is concluded to be the simplest to deploy in banks.

Blanco et al. (2013) investigated the application of Multi-Layer Perceptron (MLP) NNs using a credit dataset comprising around 5500 borrowers from a microfinance institution in Peru. They compared the performance of this model against traditional statistical methods namely: Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), and Logistic Regression. The results showed that the NNs outperformed all traditional models in terms of area under curve (AUC) and misclassifiation costs.

Yang et al. (2015) explored a Logistic Regression model using WoE applied to a credit risk dataset of 690 observations. The Percentage Correctly Classified (PCC) goodness-of-fit measure was used to evaluate the model which gave a percentage of above 84% on the test dataset. The analysis went further to construct a scorecard model to determine the credit score and accuracy of the model. The study has shown that Logistic Regression with WoE is a good model but a comparative analysis could have been done for performance evaluation with and without using WoE on the same Logistic Regression model. Additionally, only the PCC measure was used to evaluate performance.

A study done by Xia et al. (2017) focused on using a sequential ensemble credit scoring model based on an XGBoost gradient boosting machine, together with Bayesian hyper-parameter optimisation. The outcome was to show that the proposed method outperforms baseline models using accuracy, error rate, area under the curve H measure (AUC-H) and Brier score.

A mixed credit scoring model with WoE and Logistic Regression was constructed by Chen et al. (2018) applied to a large credit dataset of 100 000 observations. Ordinary Least-squares (OLS) was used to determine parameters in the regression model. The traditional Logistic Regression model was compared to the hybrid model with WoE. It was concluded that the hybrid model outperformed the traditional model by analysing the coefficient values for each model. Other methods of evaluation could have been explored in this study. The method also makes reference to NN models in outperforming the traditional Logistic Regression model. However, no NN model was constructed and compared to the hybrid model to confirm the improvement in predictability when using WoE.

Nehrebecka (2018) used an approach of comparing WoE applied to both Logistic Regression and SVM models (Linear, Gaussian and Laplace kernel) on credit risk data of non-financial firms. The models were assessed based on GINI statistics, Kolmogorov-

Smirnov (K-S) and Area Under Receiver Operating Charactistic (AUROC). WoE was applied to all models and it was concluded that the Logistic Regression model produced the best accuracy amongst the models.

An assessment of five different machine learning models was conducted by Wang et al. (2020) with the application to credit scoring. The study made use of a Naive Bayesian Model, Logistic Regression Analysis, Random Forest, Decision Tree and the k-NN classifier. The results showed through an evaluation using precision, recall, AUC and accuracy that the best performing model was the Random Forest. This research focused on a comparative performance analysis using accuracy, sensitivity and specificity, and the limitations of each model were not assessed and outlined.

A study completed by Alam et al. (2020) focused on addressing the challenge of imbalanced datasets and explores the effectiveness of different data resampling methods. The study was on credit risk prediction in commercial banks using various machine learning techniques and used three different datasets. The results demonstrated that the proposed models significantly improved accuracy compared to traditional statistical approaches. The Gradient Boosted Decision Tree method outperformed other classifiers, particularly when combined with the k-means SMOTE oversampling method. No NNs were explored in this paper.

A study on the effect of WoE on the discriminatory power of credit scoring models was produced by Persson (2021). The study showed that using WoE with Logistic Regression decreased the discriminatory power through evaluation across multiple metrics, in comparison to models that did not use WoE to transform features. In addition, the study also shows that using the feature selection technique, Information Value (IV), did not provide any benefit over using backward selection techniques. However, the results were different when applied to the SVM models in both WoE and IV where certain metrics proved that these methods improved the performance of the models.

Aranha and Bolar (2023) developed a study on comparing the performance of machine learning models to a traditional Logistic Regression model in credit risk prediction. The Classification Tree, Random Forest, NN, Naive Bayes, Logistic Regression, and AdaBoost, were used. Undersampling was applied to balance the dataset containing 10 000 records by addressing the majority class. After assessing five parameters across six techniques, including AUROC, accuracy, F1 Score, precision, and recall, the study concluded that Random Forest modelling stood out for its superior performance in accurately predicting outcomes and distinguishing between classes. Following closely, the NN technique also showed competitive results. By utilising changes in operating expenses as a measure of the Covid-19 pandemic's impact, the research revealed that machine learning methods (NN, AdaBoost, and Random Forest) outperformed Logistic Regression in predictive accuracy. Additionally, integrating Covid-19-related uncertainty into the analysis improved the model's predictive ability.

Abdou et al. (2008) achieved promising results in classification accuracy using various techniques on an credit dataset from an Egyptian bank. Discriminant analysis yielded

an average correct classification rate of 86,75%, while employing a stepwise approach increased it to 86,92%. Probit analysis achieved 87,78% accuracy, which revealed 87,26% after eliminating insignificant variables. Logistic regression attained a higher average correct classification rate of 88.30%, or 87.95% after eliminating insignificant variables. Remarkably, probabilistic NNs achieved an outstanding average correct classification rate of 96.21%, surpassing the performance of conventional methods. Multi-layer feed-forward nets achieved rates of 94,84% (five nodes) and 93,98% (four nodes). The models demonstrated superior performance compared to the existing system, which achieved a correct classification rate of 74.50% for approved loans that did not result in default. Additionally, the study explored misclassification costs, taking into account the varying consequences of type I and type II errors. No data balancing was performed on the dataset.

In a study conducted by West (2000), credit scoring accuracy was investigated utilising various NN models. In particular, the study examined Mixture-of-Experts (MOE), MLP, Radial Basis Function (RBF), Learning Vector Quantization, and Fuzzy Adaptive Resonance models. The performance evaluation utilised 10-fold cross-validation with two real-world datasets. The NN models were compared to traditional approaches like Linear Discriminant Analysis (LDA), Logistic Regression, k-NN, Kernel Density Estimation, and Decision Trees.

The findings suggested that the MLP may not be the most accurate NN model for credit scoring applications. Nevertheless, both the Radial Basis Function Neural Network (RBFNN) and MOE models demonstrate promising performance and warrant consideration. Among traditional approaches, Logistic Regression emerged as the most precise method. These findings enhance understanding of credit scoring accuracy and offer insights into the applicability of different models in real-world scenarios. Additionally, the paper explores two methods for handling data imbalance to assist NNs in identifying the minority class. It is noteworthy that neither of these training strategies resulted in a more accurate classifier when the primary goal was minimising overall error rate. However, both strategies proved effective when considering the significant costs associated with misclassifications, which vary considerably. These results shed light on the importance of considering disparate misclassification costs when designing credit risk models.

Pearson et al. (1995) investigated the suitability of RBFNNs for credit vetting systems. The study compared the performance of the RBFNN model in classifying good and bad loan cases with a manually configured MLP and a genetic algorithm-designed counterpart. The findings indicate that the RBFNN model performs similarly to other methods, yielding comparable prediction accuracy. The RBF approach's simplicity results in faster computation without sacrificing performance, potentially aiding decision-making processes by simplifying information retrieval.

This study addresses gaps in current literature by conducting a comprehensive comparative analysis of credit risk models. While previous studies have individually explored models like Logistic Regression, NN, SVM, and Decision Trees, this research consolidates

and expands upon these investigations within a unified framework. By encompassing a diverse range of models, from a traditional statistical approach (Logistic Regression) to modern machine learning techniques (k-NN, SVM, Decision Trees, MLP and RBFNN), this study seeks to provide a more holistic understanding of their relative effectiveness in credit risk contexts.

One additional gap in existing research is the limited exploration of how different data balancing techniques impact model performance. While data imbalance is a well-known challenge in credit risk, prior studies have often overlooked its significance. This research aims to bridge this gap by systematically investigating the effects of various data balancing levels on model performance.

Furthermore, by employing a wide range of performance metrics, including accuracy, precision, recall, and AUC, log loss, Jaccard Scores and F1 Scores, this study aims to provide a thorough evaluation of each model's strengths and weaknesses.

1.3 Problem Overview and Motivation

Credit risk is a key problem in commercial banking because it impacts the stability and performance of a developing country's economic growth. By modelling and managing credit risk effectively and reducing the impact of model risk, banks can maintain a healthy loan portfolio. This will ensure capital adequacy, support lending capacity, foster confidence in the financial institution's system and its models, and facilitate economic development (Siddique et al., 2021).

A variety of approaches have been employed for this task, from traditional statistical modelling to machine learning and AI. Many machine learning models necessitate extensive computational power to handle intricate and voluminous datasets. Machine learning and AI techniques can be complex and usually require statisticians or mathematicians to interpret them. Simpler statistical techniques can provide excellent results while requiring fewer computational resources. Simplified methods have also been adjusted and integrated to enhance the efficacy of models. In the financial sector, models are constantly adapted and improved to account for changing circumstances. The problem statement of this research is to determine if machine learning models can provide better results than traditional statistical modelling tools when used for credit risk assessment.

Evaluating the performance of the model requires careful consideration of model risk. This evaluation allows the modeller to understand which aspects of the model are deficient and why. It is also important for the customer who has received an undesirable credit score to understand the reasons as to why. Therefore, to guarantee the reliability and robustness of credit risk models, comprehensive evaluation and validation procedures are essential. Evaluation criteria like accuracy, precision, recall, and area under the Receiver Operating Characteristic (ROC) curve are frequently employed to gauge the effectiveness of credit risk models. Model validation techniques, including out-of-sample testing, cross-validation, and back-testing, help verify that the models

accurately reflect the current credit risk environment (Bhattacharya et al., 2023).

This research aims to conduct a comparative analysis of different statistical and machine learning methods of classification on a credit dataset from a commercial bank, as well as optimising various NN techniques to enhance performance. To further test the robustness, the performance of the models will be evaluated under various levels of data imbalance. This is to understand how a real-world dataset, which inherently exhibits data imbalance, can impact model performance and risk. The primary motivation is to gain a better understanding of the limitations of the different models and the dataset's features. Furthermore, to ascertain whether machine learning techniques are superior to optimised or non-optimised statistical analysis. This will be accomplished by employing various performance metrics to quantify the model risk.

1.4 Research Objectives

The main objective of this research is to determine the best performing model between the traditional statistical techniques and machine learning classifiers, by using various evaluation methods. The following objectives apply to this research in achieving the main objective:

1. To determine the features of the credit registry dataset contribute the most to credit risk.
2. To model the credit registry dataset using a traditional statistical technique (Logistic Regression), and various modern techniques (k-Nearest Neighbours (k-NN), SVM, Decision Trees and NNs).
3. To employ optimisation techniques for the statistical model and machine learning classifiers with the intent of improving the credit risk models.
4. To determine the limitations of each statistical and machine learning model employed in objective 2.
5. To assess if more modern statistical methods based on machine learning outperform traditional techniques for credit risk prediction on a real-world dataset.
6. To assess the performance of both traditional statistical models and machine learning models in handling imbalanced datasets commonly encountered in credit risk prediction.
7. To use various techniques to evaluate and quantify model risk with the application to credit risk.

1.5 Tools and Techniques

This section outlines the various tools and techniques that will be utilised in this research for credit risk assessment. These tools and techniques encompass statistical and

exploratory analysis, data pre-processing and balancing, feature selection, parameter estimation, machine learning models, hyper-parameter tuning, and model evaluation. The following is a summary of the tools and techniques:

1. Statistical and exploratory analysis using Python including, but not limited to, the following packages: *sklearn*, *matplotlib*, *pandas* and *seaborn*.
2. Data pre-processing and balancing through techniques such as Synthetic Minority Over-sampling Technique (SMOTE), Random Under-Sampling (RUS) or Random Over-Sampling (ROS).
3. Feature Selection through the use of dimensionality reduction. This can be achieved through extraction of important variables by techniques such as *filter*, *wrapper* and *embedded* methods.
4. Parameter Estimation techniques for statistical analysis such as Maximum Likelihood Estimation.
5. Machine learning such as k-NN, SVM, Decision Trees and NNs, as well as statistical learning using Logistic Regression through Python packages such as: *sklearn*, *tensorflow*, *pytorch*, *matplotlib*, *numpy*, *imblearn*, *pandas* and *seaborn*.
6. Hyper-parameter tuning of the machine learning models, where necessary, through the use of grid and random search, and other applicable methods.
7. Model evaluation and risk quantification through the use of specificity, sensitivity and accuracy methods.

1.6 Research Layout

The paper is structured as follows: Chapter 2 presents the statistical model and machine learning techniques. Chapter 3 outlines the methodology used in the study. Chapter 4 presents the results and evaluates the models employed. Finally, Chapter 5 concludes the findings and offers recommendations for future research.

Chapter 2

Statistical Modelling and Machine Learning Techniques

This chapter provides an overview of feature selection methods, outlines data balancing techniques, explains relevant machine learning and statistical models, and outlines different model evaluation methods.

2.1 Feature Selection Methods

Features represent the predictor variables used in a quantitative model to classify the predicted variable. Having too many, and particularly irrelevant, features can negatively impact model performance, computing power, model complexity and interpretation (Beniwal and Arora, 2012). Feature selection is a process whereby unnecessary predictor variables are removed or transformed so that the final dataset is more attributable to the modelling process and problem statement. There are different techniques that can be used for feature selection in supervised learning, *filter*, *wrapper* and *embedded* (Wah et al., 2018), which are highlighted in this section.

2.1.1 Filter Methods

The filter methods use a process of ranking the predictor variables and thus filtering out low-scoring features. These methods are simple, fast and are independent of the classifier. However, a threshold must be determined for filtered features since all features can be selected through this process (Sánchez-Marono et al., 2007). There are various techniques that fall under the *filter* category, but the methods applicable to classification are outlined in this sub-section.

- Chi-square statistics: the Chi-square statistics involves measuring how independent one categorical variable is from another. If the dataset consists of numerical variables, then these must be discretised to form groups or levels. The purpose

is to determine if the class variable is independent of the feature, in which case the feature is disregarded. On the contrary, if the feature and class variable are dependent, then the feature is important. The Chi-square statistic is given in Equation 2.1 (Liu et al., 2002),

$$\chi^2 = \sum_{i=1}^{m}\sum_{j=1}^{k}\left(\frac{A_{ij} - E_{ij}}{E_{ij}}\right), \tag{2.1}$$

where m is the number of levels or groups, k is the number of classes, A_{ij} is the number of observations in the interval i and class j. E_{ij} is the expected frequency of A_{ij} and is determined from Equation 2.2,

$$E_{ij} = R_i \times \frac{C_j}{N}, \tag{2.2}$$

where R_i is the number of observations in the interval i, C_j is the number of observations in the class j and N is the total number of observations. The larger the chi-square value, the more important the feature is.

- Correlation Coefficient: A correlation coefficient can be used on a dataset to determine how linearly correlated a feature is to the class variable. A threshhold for correlation is set to filter out features that are insignificant to the target class and are correlated with other features. This method is used for numerical features and cannot be used for categorical or nominal features. Most commonly, the Pearson correlation coefficient, r, is used and is defined as follows (Biesiada and Duch, 2007),

$$r = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum(x_i - \overline{x})^2 \sum(y_i - \overline{y})^2}}, \tag{2.3}$$

where x_i is value of the independent variable, $\overline{x}$ is the average of the independent variables, y_i is value of the class variable and $\overline{y}$ is the average of the class variables in the dataset. The correlation coefficient can take on values in the range [-1,1]. A value close to 1 means the feature is positively correlated with the target class, a value close to -1 means the feature is negatively correlated with the target class, and a value of 0 means there exists no correlation between the feature and the target class.

- Information Gain: this is a measure based on the theory of entropy which is a study of the disorderliness of a process. The information gain determines the reduction in entropy before and after a feature is included in the dataset. The purpose is

to determine how relevant features are for classification. The information gain between two random variables x and y is determined using Equation 2.4 (Jadhav et al., 2018),

$$IG(y|x) = H(y) - H(y|x), \tag{2.4}$$

where $H(y)$ is the binary entropy of event Y (target class) which can have two possible outcomes, and $H(y|x)$ is the conditional entropy of the events X (feature) and Y when $X = x$. The higher the information gain, the more relevant a feature is and the more discriminatory power it has.

- Symmetrical Uncertainty: The symmetrical uncertainty (SU) measures feature redundancy as follows using the feature entropy and information gain value (Piao et al., 2019),

$$SU(X,Y) = 2 \times \frac{IG(Y|X)}{H(X) + H(Y)}, \tag{2.5}$$

where $SU(X,Y)$ is the symmetrical uncertainty between features X and Y, $IG(Y|X)$ is the information gain, and $H(Y)$ and $H(X)$ are the entropies of features X and Y. $SU(X,Y)$ can take on a value in the range [0,1], where a value of 1 indicates strong predictability of one feature to others, and a value of 0 indicates that two features are uncorrelated (Potharaju and Sreedevi, 2017).

- Minimum Redundancy, Maximum Relevance (mRmR): this method serves the purpose of reducing feature redundancy among selected features and maximising relevance to the the class variable (Peng et al., 2005). Search methods can also be used with the mRmR to further obtain the best features. The process is iterative whereby at each iteration, the feature that has maximum relevance to the class and minimum redundancy with respect to all the features selected in preceding iterations, is selected. For continuous features, a score is determined at each iteration by using the F-Statistic for relevance and Pearson Correlation for redundancy.

- Correlation-Based Feature Selection (CFS): this method deals with feature redundancy. The method selects features which are highly correlated with the predicted variable , but have correlation among the independent variables. This is done through the use of a correlation coefficient. The idea is to find feature subsets that have high feature to class correlation to improve prediction and low feature to feature correlation to avoid redundant features. For a subset of features, the metric (Sánchez-Marono et al., 2007) is given in Equation 2.6,

$$M_S = \frac{k\overline{r}_{cf}}{\sqrt{k + k(k-1)\overline{r}_{ff}}}, \tag{2.6}$$

where M_s is the heuristic merit of the feature subset S with k features, $\overline{r}_{cf}$ is the average feature to class correlation and $\overline{r}_{ff}$ is the average feature to feature correlation. In general, the numerator of Equation 2.6 indicates the predictability of the feature subset to the class. The denominator indicates the feature to feature redundancy within the subset.

- Fast Correlation-Based Filter (FCBF): this method makes use of the symmetrical uncertainty (SU) value, represented in Equation 2.4, on a full set of features to determine correlation between features. It further makes use of backward selection to identify a final subset of features together with a sequential search method (Senliol et al., 2008). The purpose is to filter out insignificant features until there are no features to eliminate. It will select features correlated with the class variable above a specified SU threshold. This method is ideal for a multivariate dataset.

- Fisher Score: the Fisher score is an approach which aims to find a feature subset that it maximises the distances between observations from different classes and minimises the distances between observations of the same class (Lin et al., 2021). The Fisher Score of on an iteration for the i^{th} feature can be calculated as follows (Aggarwal, 2015),

$$S_i = \frac{\sum n_j(\mu_{ij} - \mu_i)}{\sum n_j \rho_{ij}^2}, \tag{2.7}$$

where μ_{ij} and ρ_{ij} are the mean and the variance of the i^{th} feature in the j^{th} class, respectively, n_j is the number of instances in the j^{th} class and μ_i is the mean of the i^{th} feature.

- Relief: the Relief method is an algorithm that calculates statistical weights in relation to relevance to the class variable. These weights, $W[X]$ (Weight of variable X), can take on values from [−1,1] with −1 being the worst and 1 being the best in relation to quality and relevance to the class variable. The algorithm uses feature value difference through k-nearest neighbours to determine the nearest hit H and nearest miss M for an instance pair. If a feature value difference exists for a neighbouring pair of the same class, then the weight score decreases. Conversely, if a feature value difference exists between a neighbouring pair of a different class, then the weight score increases (Urbanowicz et al., 2018).

2.1.2 Wrapper Methods

Wrapper methods are based on machine learning algorithms, known as black boxes, for example SVM or Naive Bayes, fitted on the given dataset. The feature selection follows

an iterative process on the dataset which eliminates or adds features and the final subset is one based on maximum accuracy. Wrapper methods are slower than filter methods as the algorithms require more computational resources to fit, train and predict using the model, but produce subsets that have better performance (Jović et al., 2015). The final dataset is also biased towards the particular algorithm used and hence should be validated with a secondary algorithm for maximum accuracy.

- Forward Selection: this feature selection method is a type of stepwise regression and greedy search strategy which starts with an empty dataset and adds features one by one on each iteration. Features are added based on maximising model performance using an evaluation function (Marcano-Cedeño et al., 2010).
- Backward Elimination: this method is similar to forward selection and is also a type of step-wise regression and greedy search strategy which starts with a full dataset and works backward to eliminate redundant features to maximise model performance based on evaluation criteria.
- Standard Step-wise Regression: this method combines forward selection and backward elimination whereby features are added or removed on each iteration based on improving the R-square score.
- Recursive Feature Elimination: unimportant features are eliminated using an algorithm in the recursive feature elimination method to improve model performance. The features are removed such that there is mimimal impact on training errors and is thus ideal for small sample classification datasets (Chen and Jeong, 2007).

2.1.3 Embedded Methods

Embedded methods are attributed to selecting features during execution of the model and are known to combine qualities of filter and wrapper methods. These methods are known for good interpretability and prediction performance.

- Least Absolute Shrinkage and Selection Operator (LASSO) L1 Regularisation: the LASSO Regularisation method was introduced by Tibshirani (1996) and selects features through 'shrinking' regression coefficients and reducing some to zero. Features with coefficients that are greater than zero are selected to be used in the final dataset. This method is ideal for reducing errors when predicting and is best used in datasets which have a low number of observations and a high number of features.
- Random Forest: this method is an algorithm that uses a series of several decision trees established over a randomised selection of observations and features (Nguyen et al., 2013). Since each tree does not select all features, the model is not prone to over-fitting. At each point in a tree, also known as a node, the algorithm fits the observations into two buckets which classify those which are similar to one another but different to the observations in the second bucket. Feature importance is then based on how pure a bucket is using a measure such as the Gini Impurity Measure

(the probability of misclassifying an observation) or Information Gain which has been described in Section 2.1.1. The more important a feature is, the more it reduces the impurity measure. The decrease in impurity is averaged out for each feature across all trees and aids in determining how important a feature is for the final dataset.

- Classification and Regression Tree (CART): this method is useful for non-linear datasets and consists of a binary decision tree. It is constructed by determining the best feature that reduces impurity in the subsequent two tree nodes (Questier et al., 2005). This reduction in impurity is measured by the Gini Impurity Measure or Entropy as discussed in Section 2.1.1. The tree can be grown until a user-specified threshold. The difficulty is in selecting an optimal threshold for which performance is maximised but can be aided using cross-validation techniques.

2.2 Data Balancing

In the real-world, credit risk datasets are often imbalanced. In classification, an imbalanced dataset is one in which the forecast variable has an uneven distribution of observations. In other words, one group of the forecast variable contains more or less observations than the alternative group (s) of the forecast variable. A crucial step in data analysis is balancing. Problems with imbalanced data arise when modelling because a model's performance can be biased toward a specific group of the forecast variable and affect prediction accuracy. The purpose of balancing a dataset is to ensure that the model accurately predicts the minority and majority class. There are a few techniques which can be used to balance a dataset:

- Random Under-sampling (RUS): this method is a technique whereby the number of observations from the majority class are randomly removed until equal to the minority class (Mohammed et al., 2020). Figure 2.1 shows a simple diagram of how RUS works in modelling. Because observations are removed when under-sampled, valuable features and patterns that can be beneficial to one, are also removed. This technique is therefore not suitable for small datasets.
- Random Over-sampling (ROS): this technique is opposite to the RUS method, whereby observations from the minority class are duplicated to achieve the same number of observations in the majority class. Figure 2.1 shows a diagram of how ROS works in modelling.

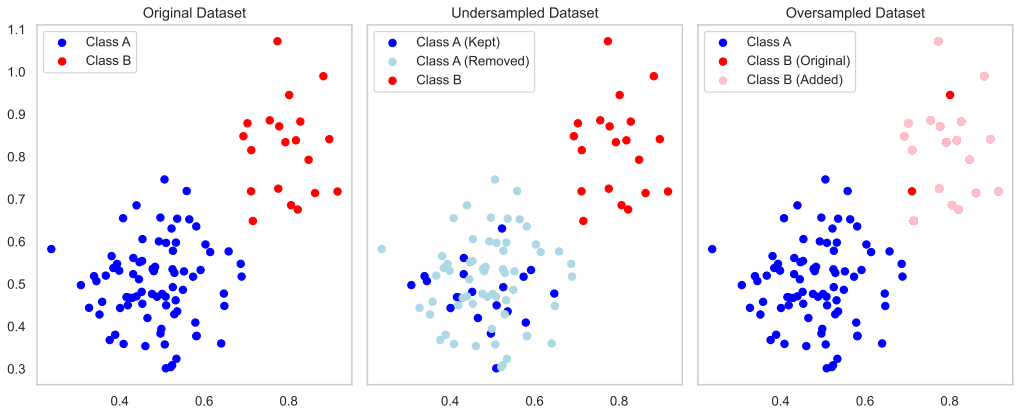

Figure 2.1: Random Under-sampling and Over-sampling differences for imbalanced datasets (Towards Data Science, 2019).

- Synthetic Minority Over-sampling Technique (SMOTE): this method is an extension of the ROS method developed by Chawla et al. (2002). The purpose is to create 'synthetic' observations of the minority class in place of duplicating samples to avoid model over-fitting (Elhassan et al., 2016). The technique works by selecting random observations from the minority class and determining the k-nearest neighbours for this data point. Thereafter, 'synthetic' observations are added between the chosen observation and its neighbours. SMOTE should be applied after selecting a test set, onto the training set. This is because the test set may contain the same points as in the training set if the SMOTE technique is applied beforehand and the model may over-fit the data (Mishra, 2017). Figure 2.2 shows a diagram of how SMOTE works.

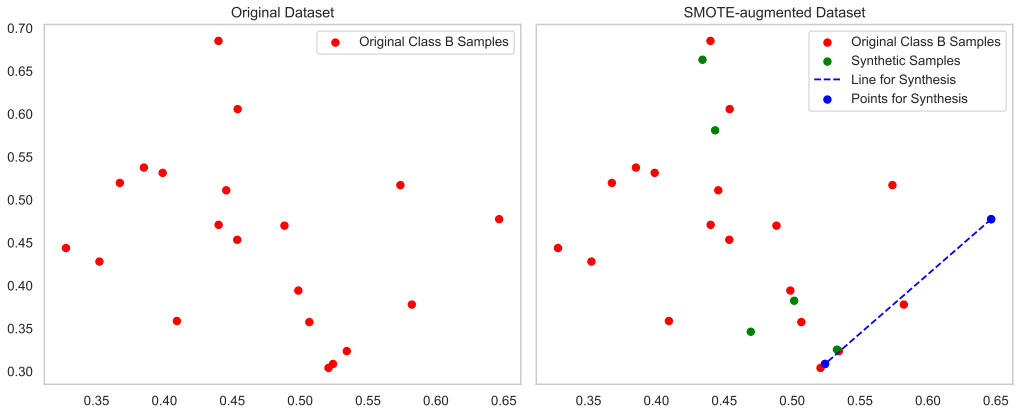

Figure 2.2: Synthetic Minority Over-sampling Technique for imbalanced datasets (Ashes Das, 2019).

The original SMOTE technique will not work for datasets that are fully categorical or mixed between categorical and quantitative, and must be adapted. This is because the method interpolates between points to create a 'synthetic' point. Depending on how the data is encoded, the 'synthetic' points produced have the possibility of having no meaning. For example, if feature A has categories 1 and 2, there is a possibility of achieving a 'synthetic' point of 0,5 which has no meaning to the feature. There are a few adaptations of the SMOTE technique to cater for these types of datasets:

1. Synthetic Minority Over-sampling Technique – Nominal Continuous: this method is a variant of SMOTE applied to a mixed dataset of quantitative and categorical observations. The method works as follows (Chawla et al., 2002):
   (a) Determine the median: the median of the standard deviations of the quantitative or continuous observations in the minority class is calculated.
   (b) Compute the nearest neighbour: the Euclidean distance for the sample set for which the k-nearest neighbours are being evaluated and other feature observations in the minority class.
   (c) Populate the 'synthetic' sample: this step involves the same approach used in the SMOTE technique where the 'synthetic' sample feature is given a value that occurs the most in the k-nearest neighbours evaluated for the sample set.
2. Synthetic Minority Over-sampling Technique – Nominal: this method is an extension of the SMOTE technique applied to nominal features only. The k-nearest neighbours are computed using an adapted Value Difference Metric (VDM) (Stanfill and Waltz, 1986). VDM considers feature value overlapping over all feature subsets and then a generates distance matrix which can be used to determine the nearest neighbours.

2.3 Statistical and Machine Learning Classifiers

There are numerous statistical and machine learning methods applied to model classification and prediction. The following sections will cover the most common methods used for classification and supervised learning.

2.3.1 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA), also known as Fischer Discriminant Analysis (FDA), is a popular statistical technique used for classification and dimensionality reduction. The technique is advantageous for handling within-class imbalance (Balakrishnama and Ganapathiraju, 1998). The aim is to find a linear combination of independent

variables that maximises the separation of two or more target classes. LDA assumes features are linearly separable and that the features are normally distributed.

Steps in Analysis

1. The first step in an LDA classification is to find the distance between the means of the different classes known as the between-class matrix.
2. The distance between the mean and the sample observations are computed and this is known as the within-class matrix.
3. The next step is to construct a reduced dimensional space such that the between-class variance is maximised and the within-class variance is minimised (Tharwat et al., 2017).

The reduced dimensional space can then be used to construct a discriminant using Bayes' rule. This determines the posterior probability or the probability that a sample point is associated with a certain target class, given the attribute of the sample point.

LDA outperforms Logistic Regression in cases where there are more than two target classes. It falls short when linear decision boundaries are not effective in separating classes which are non-linearly separable. LDA is also performance sensitive to datasets where the number of independent variables are less than the number of observations. In this case, regularisation is needed.

2.3.2 Logistic Regression

Logistic Regression is a type of traditional statistical model used in classification and prediction. It is widely used in credit risk models due to its mathematical flexibility and ease of interpretation (Satchidananda and Simha, 2006). The method, in contrast to Linear Regression, does not assume that the distributions of characteristics in the feature space are normal. Over-fitting can be a disadvantage for model performance in datasets where the number of observations are less than the number of features. The concept behind Logistic Regression is to find a relationship between a dependent variable and one or more independent variables. The independent variables can be categorical, continuous or both, in nature. The method can be used as a classification technique to predict a dichotomous target variable by determining the probability that an observation belongs to a particular class. The functional form of this probability can be expressed as follows (Dreiseitl and Ohno-Machado, 2002):

$$P(y|x) = f(x,\beta), \tag{2.8}$$

where $P(y|x)$ is the probability that an observation belongs to a class y given that the observation takes on a specific value x, modelled by some functional form $f(x,\beta)$. The parameters for β can be determined through maximum likelihood estimation on the given dataset.

The mathematical model for the functional form of Logistic Regression for one independent variable is as follows (Korkmaz et al., 2012) and is characterised as a sigmoid function shown in Figure 2.3:

$$P(Y = 1) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}. \tag{2.9}$$

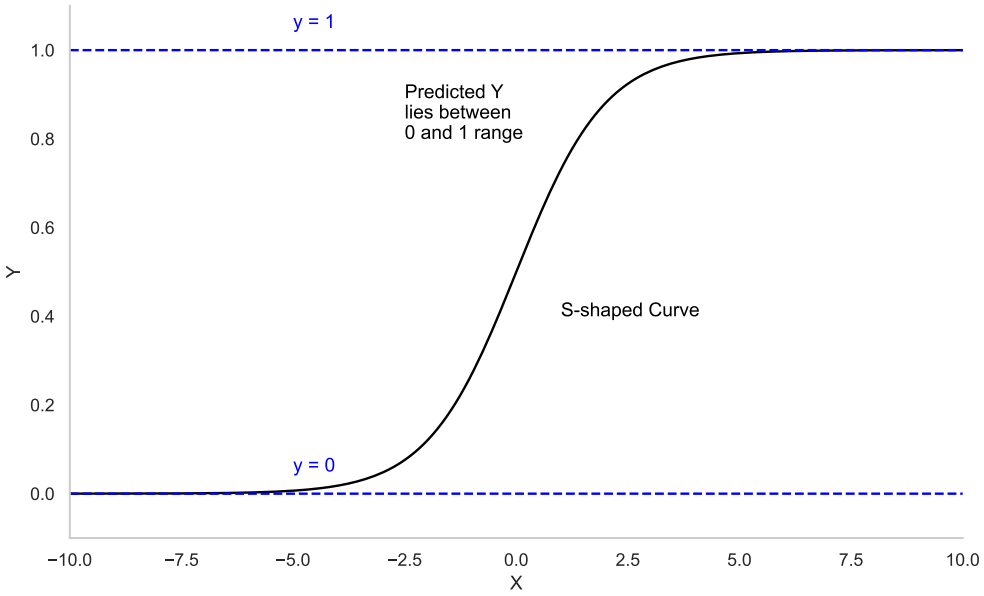

Figure 2.3: Graphical form of the Logistic Regression Model – sigmoid function (Kanade, 2022).

The model can also be extended to multiple independent variables as follows (Yang et al., 2015):

$$P(n) = \frac{e^{\beta_0+\beta_1 X_{n,1}+\beta_2 X_{n,2}+\cdots+\beta_k X_{n,k}}}{1 + e^{\beta_0+\beta_1 X_{n,1}+\beta_2 X_{n,2}+\cdots+\beta_k X_{n,k}}}, \tag{2.10}$$

where $P(n)$ is the probability of the the outcome of the class, $X_{n,i}$ is the value of observation n for a specific category i and β_i is the regression coefficient of the model. Since the functional form is a probability distribution, the values of β_i can be determined using maximum likelihood estimation.

Assumptions of Logistic Regression

- The independent variables are linearly correlated to the log-odds $\log \frac{P(n)}{1-P(n)}$ of the class variable (Tu, 1996).
- No multicollinearity exists between independent variables.
- Observations in the dataset are independent.
- There are no outliers in the dataset as the model is sensitive to large changes in observation values.

Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE) is a parameter estimation technique used for a Logistic Regression model. The MLE is the value that maximises the predicted probability of an observation belonging to a particular class. Referring to Equation 2.10, the parameters to be estimated are the β vectors. The parameters are estimated such that the product of all probability $P(n)$ is as close to 1 as possible for a given observation for the target class y equal to 1. For the target class y equal to 0, the parameters are then estimated such that the product of the probability $1 - P(n)$ is as close to 1 as possible.

The product of all probability for $P(n)$ or $1 - P(n)$ is called the likelihood function $L(\beta)$ (Czepiel, 2002).

$$\begin{aligned} L(\beta) &= \prod_{\text{n in } y_i = 1} P(n) \times \prod_{\text{n in } y_i = 0} (1 - P(n)) \\ &= \prod_{n} P(n)^{y_i} \times (1 - P(n))^{1-y_i}. \end{aligned} \tag{2.11}$$

To simplify solving Equation 2.11, the log of the function is taken to get the log likelihood function $l(\beta)$.

$$\begin{aligned} l(\beta) &= \ln \left[\prod_{n} P(n)^{y_i} \times (1 - P(n))^{1-y_i} \right] \\ &= \sum_{i=1}^{N} y_i \ln \left(P(n) \right) + (1 - y_i) \ln(1 - P(n)). \end{aligned} \tag{2.12}$$

Weight of Evidence

WoE is a type of variable transformation that can be used with Logistic Regression to maximise the predictive ability of a feature. WoE determines the predictive power of an independent variable in relation to the target or dependent variable. The method

works by assigning a weight to each binned category in the feature dataset. Logistic Regression assumes that the independent variables have a linear relationship with the log odds. Knowing this, a WoE is formulated so that a higher weighting is assigned to more relevant categories, and a lower weighting is assigned to less relevant categories. This allows the method to satisfy the log-odds assumption and is a good variable transformation for Logistic Regression. The WoE is calculated as follows (Yuan, 2018):

$$WoE_i = \ln \left[\frac{\frac{G_i}{G}}{\frac{B_i}{B}} \right], \tag{2.13}$$

where i represents an integer from 1 to the number of categories in the dataset, G_i is the number of observations representing good risk given the i^{th} category, in terms of credit risk, G is the total number observations classed as good risk, B_i is the number of observations representing bad risk given the i^{th} category, in terms of credit risk and B is the total number observations classed as bad risk.

The transformed variables are then used instead of the original values in the Logistic Regression model. It must be noted that before using WoE, feature selection should be performed for optimal performance.

2.3.3 Naive Bayes

Naive Bayes is a classification algorithm that can be used for dichotomous or polytomous dependent variable classification. The classification algorithm is based on Baye's rule and assumes that the features are conditionally independent of one another, given the class variable (Krichene, 2017). The *naive* aspect comes from the fact that the hypothesis probabilities used in the classification, are simplified.

The basis of Naive Bayes comes from Bayes' rule and hypothesis testing where the conditional probability of each target variable given the feature space is calculated. For example, consider a dataset that consists of the feature space A_1,A_2,A_3,...,A_k with a target variable that has a binary output B_1 or B_2. The hypothesis h can then be defined as the target class, B_1 or B_2, to which the given feature sample space is assigned. Then the following relationship is used (Frank et al., 2000):

$$P(h|A_1,A_2,A_3,\ldots,A_k) = \frac{P(A_1,A_2,A_3,\ldots,A_k|h) \times P(h)}{P(A_1,A_2,A_3,\ldots,A_k)}, \tag{2.14}$$

where,

- $P(h|A_1,A_2,A_3,\ldots A_k)$ is the posterior probability or the probability of hypothesis h given the data A_1,A_2,A_3,...,A_k.

- $P(A_1,A_2,A_3,\ldots,A_k|h)$ is the probability of feature space $A_1,A_2,A_3,\ldots,A_k$ given that the hypothesis h was true.
- $P(h)$ is the prior probability or the probability that the hypothesis h was true.
- $P(A_1,A_2,A_3,\ldots,A_k)$ is the probability of the feature space $A_1,A_2,A_3,\ldots,A_k$.

The Naive Bayes classifier is advantageous for its speed, accuracy and minimal resource usage. The model is used in many areas for classification, however, performance is adversely affected when the assumption of feature independence is violated (Wang et al., 2020). This can be a disadvantage in real-world modelling since most datasets have features which are correlated.

2.3.4 Decision Tree

The Decision Tree is a supervised learning method that can be used for classification and regression. The method is based on a tree structure, where the decision nodes of the tree represent the deciding features of the data, branches represent the decision-based algorithms and the ending nodes or leaves represent the outcome. The algorithm begins at the root node and compares a feature's attribute value to that of population dataset and makes a decision to which node it branches to. A graphical representation of the tree structure is shown in Figure 2.4.

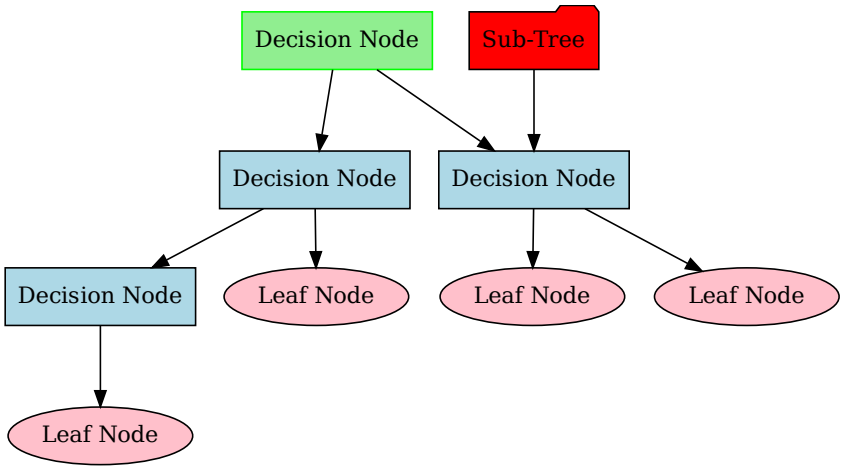

Figure 2.4: Structure of the Decision Tree classifier (Arain, 2022).

Algorithms based on impurity measures form the foundation of constructing a Decision Tree from a dataset. Impurity is a measure of the homogeneity of the labels at a particular node. The most common impurity measures used in Decision Trees are Entropy,

Gini Index and Information Gain. Entropy is defined as the amount of information required to describe data. If the data is homogeneous, then it has an entropy of 0, while less homogeneous data will contain an entropy moving toward 1. The Gini Index measures how impure a node is. The metric can take on a value between 0 and 1 and it is calculated as follows (Muchai and Odongo, 2014):

$$\text{Gini Index} = 1 - \sum_{i=1}^{n} p_i^2, \tag{2.15}$$

where p_i is the probability of each class. The lower the index, the more pure a node is. Decision Trees measure homogeneity in data, and if the data is homogeneous then it is binned into the same class. This is how a Decision Tree will branch its nodes. At every decision node, the best feature used to branch the tree and its best feature value or category, are selected. These attributes are based on reducing the impurity at each node using the impurity measures.

While the Decision Tree classifier can be easily understood, is computationally fast and similar to human-based decision-making patterns, it can contain many branches if a dataset is large with many features, rendering it complex. Decision Trees are also sensitive to noisy data (Podgorelec et al., 2002).

2.3.5 k-Nearest Neighbours

k-NN is a supervised machine learning method that uses proximity to predict outcomes or classify data. The method is non-parametric in nature which means that it does not require the population feature space to be normally distributed. A probability density function is developed to estimate the probability that a sample feature space belongs to a certain target class by using the proportions of the target class determined by the k most similar points (Henley and Hand, 1996). A graphical and high-level overview of this algorithm is shown in Figure 2.5.

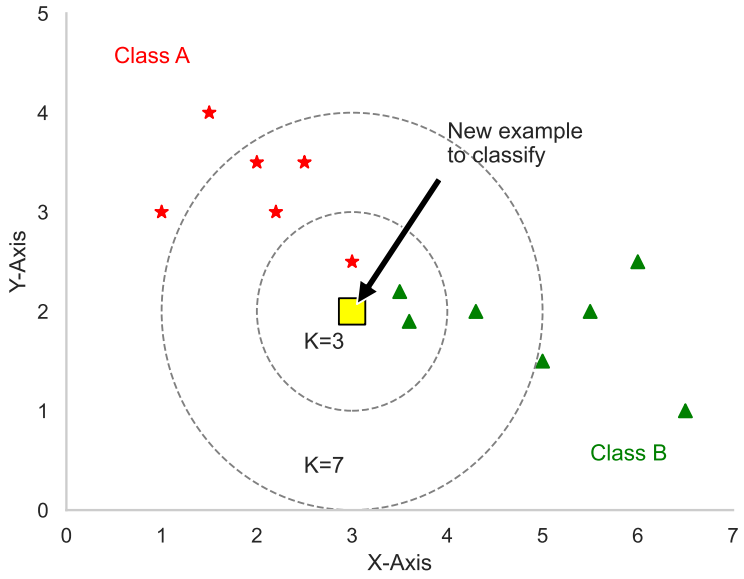

Figure 2.5: Graphical representation of the k-NN classifier (Al-Rahman Al-Serw, 2021).

The k-NN algorithm uses one of a variety of distance metrics to calculate the distance or proximity between a observation and its neighbours. The most popular distance metric used is the Euclidean distance which is a variant of the Minkowski distance measures. The Euclidean distance formula is computed as follows (Gu and Dong, 2018):

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}, \tag{2.16}$$

where $d(\mathbf{x},\mathbf{y})$ is the measure of similarity between vectors $\mathbf{x}$ and $\mathbf{y}$, given that $\mathbf{x} = (x_1,x_2,\ldots,x_n)$ and $\mathbf{y} = (y_1,y_2,\ldots,y_n)$.

There are several other distance metrics which can be used such as Manhattan, Chebyshev, Sorensen, Hassanat and Cosine distance to name a few. A full study on the effect of distance measures on the performance of the k-NN classifier was completed by Surya et al. (2019). The study showed that k-NN model performance is significantly affected by the type of distance metric, however, the performance is also dependent on the nature of the dataset used, for example, noisy data. This means there is no best distance metric to use and it is up to one to decide depending on the dataset. The model performance is also affected by the parameter k. There are many methods to choose the value of k, however, it is suggested to use different values and select the best performing k-NN model (Paryudi, 2019).

k-NN can be beneficial in datasets where the target variable is polytomous. The method

is also known to be easily interpreted. As with most machine learning models, k-NN is sensitive to imbalanced datasets and requires a balanced distribution on the target variable for better performance (Hand and Henley, 1997). Hence, data balancing is of interest to the current study.

2.3.6 Support Vector Machines

Support Vector Machines (SVM) is a supervised learning technique used for classification and regression modelling. However, it is more commonly used in classification. The SVM algorithm is known to be robust, accurate and simple. A major advantage of the model is that it is not prone to over-fitting.

The purpose of the SVM algorithm is to find an optimal decision boundary called a hyperplane in an n-dimensional space, where n is the number of features in the dataset, such that the plane separates the data points by target class. Since there can be several planes to correctly separate the data points, the maximum margin is calculated to find the optimal hyperplane. This is the maximum distance between the support vectors of each class. The support vectors are the points closest to the hyperplane. The reason the maximum distance is required is because the support vectors represent data points that are the most difficult to classify since they are close to the boundary separating the classes. Hence, the further away these points are from one another, the more accurately the algorithm can classify a data point. A graphical representation of the SVM algorithm is shown in Figure 2.6.

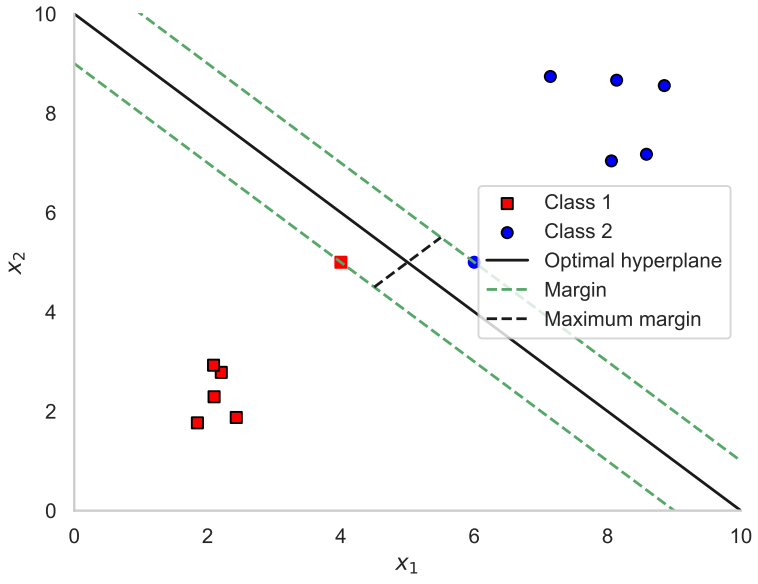

Figure 2.6: Graphical representation of the SVM classifier (Gandhi, 2022).

The aim of the technique is to find a hyperplane $g(x)$ that has the following mathemat-

ical formula (Awad and Khanna, 2015):

$$g(x) = \mathbf{w^T x} + \mathbf{b}. \tag{2.17}$$

This is so that the plane correctly separates the data points per class variable. In mathematical terms, given the set of data points $\mathbf{x}$, categorised by two linearly separable classes y_1 and y_2, SVM aims to find $\mathbf{w}$ and $\mathbf{b}$ such that $g(x)$ is equal to 1 for the support vectors belonging to class y_1 and -1 for the support vectors belonging to class y_2 (Awad and Khanna, 2015).

2.4 Neural Networks

NNs represent a class of machine learning algorithms applicable in tasks such as classification, clustering, and regression analysis. They comprise intricate networks that loosely mimic the interconnected structure of neurons in the human brain. In supervised machine learning, where datasets are labelled, NNs are utilised for classification tasks. Input nodes are interconnected with coefficients known as weights, which are aggregated and processed through an activation function to generate an output. This mechanism bears resemblance to the activation of neurons in the human brain in response to stimuli. Activation functions are mathematical functions that introduce non-linearity into the neuron's output. They aid in representing intricate relationships and capturing non-linear patterns within the data.

There are many variants of NNs that can be used for both unsupervised and supervised machine learning. This section will discuss the most common NN techniques used in modelling data for credit risk classification.

2.4.1 Multilayer Perceptron

The Multilayer Perceptron (MLP) is a type of feedforward NN model comprising an input layer, followed by one or more hidden layers, and finally an output layer (Ramchoun et al., 2016). In the context of NNs, *feedforward* refers to the process of information propagation through the network in a one-way direction, starting from the input layer and progressing towards the output layer, without any feedback loops. In an MLP network architecture, the input layer comprises artificial neurons representing input variables, while the output layer predicts the desired outcome. Situated between these layers, there may exist one or multiple hidden layers, aiding the network in discerning intricate patterns and correlations within the dataset. The structure of the MLP network is depicted in Figure 2.7.

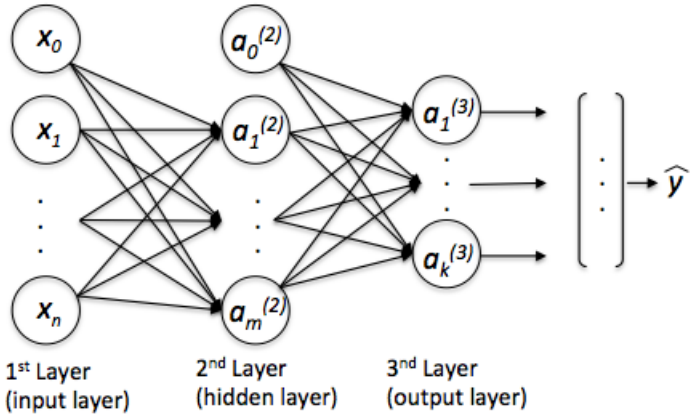

Figure 2.7: MLP architecture (Sebastian Raschka, 2023).

Throughout training, the MLP network fine-tunes the connections, termed weights, between its neurons to minimise the disparity between its predictions and the true outcomes within the training dataset. This iterative adjustment process is referred to as *backpropagation* (Taud and Mas, 2018). The network then, through iteration, updates the weights based on the produced error, gradually improving its prediction ability.

Weighted Sum

Weights can be thought of as coefficients for the equation modelled by the data which is required to be solved. During training, the weights are adjusted to minimise the loss, which represents the difference between the model's output and the desired target.

The weighted sum of a neuron's inputs is computed by multiplying each input by its respective weight and then summing up all these products. This is shown visually in Figure 2.8.

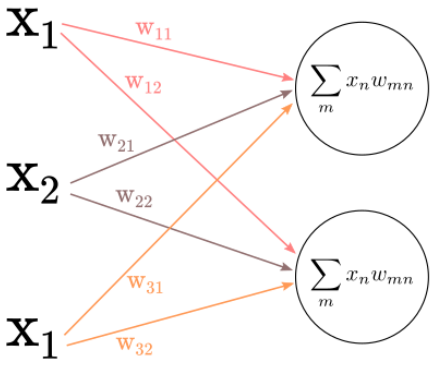

Figure 2.8: Weights in the MLP network (Pablo Caceres, 2020).

For the neuron j in a single layer perceptron, the weighted sum (h) is computed by (Heidari et al., 2019):

$$h_j = \sum(w_{ij} \times x_i) + b_j, \tag{2.18}$$

where x_i is the i^{th} input unit, w_{ij} is the weight connecting input x_i to neuron j, and b_j is the bias term. Following this summation, the weighted sum undergoes an activation function to produce the output for neuron j. The outcome y_i after subjecting the weighted sum to the activation function is computed as follows:

$$y_i = a_i^{l-1} = f_j(\sum(w_{ij} \times x_i) + b_j), \tag{2.19}$$

where a_i^{l-1} represents the activation of neuron i in the preceding layer $l-1$ and f_j denotes the activation function applied to the neuron.

Regarding many hidden layers, for the neuron j in layer l, the weighted sum (h_j^l) is computed by:

$$h_j^l = \sum(w_{ij}^l \times a_i^{l-1}) + b_j^l. \tag{2.20}$$

Activation Functions in MLP

There are several functions used to activate a neuron in the MLP network. The following functions are the most commonly used in MLP.

- Sigmoid Function (Logistic Function): This is a smooth, S-shaped curve that maps the weighted sum of a neuron's inputs, h, to value between 0 and 1. The equation for the sigmoid activation function is (Aljarah et al., 2018):

$$f(h) = \frac{1}{1 + e^{-h}}. \tag{2.21}$$

- Hyperbolic Tangent Function (Tanh): Much like the sigmoid function, the tanh function also converts the weighted sum to a value within the range of -1 and 1 using the following equation (Roy et al., 2022):

$$f(h) = \frac{e^{h} - e^{-h}}{e^{h} + e^{-h}}. \tag{2.22}$$

  The tanh function is more commonly used in the hidden layers of the MLP network as they allow for negative output values and can capture a larger range of patterns within data compared to the sigmoid function.

- Rectified Linear Unit (ReLU): This function operates in a piece-wise linear manner. It returns the input unchanged if the input is positive, but yields 0 if the input is negative. The function is defined by the following equation:

$$f(h) = \max(0,h). \tag{2.23}$$

  The ReLU function is frequently employed within the hidden layers of the MLP network due to its computational efficiency, especially in deep neural networks. Deep NNs are those with many hidden layers.

- Leaky ReLU: This function is an adaptation of the ReLU function that introduces a small slope for negative inputs. This prevents 'dead' or zero output neurons during training that addresses issues such as a neuron becoming stuck at zero (Yu et al., 2020). The equation for this is given by:

$$f(h) = \max(ah,h), \tag{2.24}$$

  where a is some small positive constant.

2.4.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are those designed for sequential data, for example sentences, time series data or even music. Because each piece of data is dependent on the data which precedes it, the order of the data is important. RNNs have a memory based structure to allow information to be persisted throughout the network structure. This allows the network to understand dependencies and context within a data sequence. Using time steps, RNNs process sequential data by going through each element in the data sequence. At each iteration, the network receives an input and adjusts its memory using information from the current input as well as the memory retained from the preceding step.

The RNN network comprises three layers: input, recurrent hidden, and output. The input layer can have n many units and is fed through the network in a sequence through time t. The hidden state $h(t)$ at time step t is calculated by (Salehinejad et al., 2017):

$$h(t) = f(w_{xh} \times x(t) + w_{hh} \times h(t-1) + b), \tag{2.25}$$

where the weight w_{hy} links the input $x(t)$o the hidden state $h(t)$, while w_{hh} connects the current hidden state $h(t)$ to the previous hidden state $h(t-1)$. The function $f()$ applied here represents the activation function for the hidden layer, and b denotes the bias term. Then, the output $y(t)$ is calculated as follows:

$$y(t) = f_y(w_{hy} \times h(t) + b_y), \tag{2.26}$$

where w_{hy} denotes the weight parameter linking the hidden state to the output layer, while b_y represents the bias term associated with the output layer. The function $f_y()$ is the applied activation function for the output layer.

It is crucial to note that these function computations are recurrent, meaning the network will repeat the process at each time step to revise the weights accordingly. This allows information to be integrated throughout training to make accurate predictions.

Activation Functions in RNNs

The activation functions used in RNNs are the same as for the MLP network which include the sigmoid, hyberbolic tangent and ReLU functions described earlier. Moreover, the Softmax activation function is frequently applied to the output layer in datasets involving multi-class classification. This function transforms the outputs of the RNN into a probability distribution across the multiple classes, ensuring that the probabilities sum up to 1. The equation is as follows (Wang et al., 2018a):

$$f(h_i) = \frac{e^{h_i}}{\sum_{j=1}^{n} e^{h_j}} (i = 1,2,...,n), \tag{2.27}$$

where h_1,h_2,...,h_n are the inputs to the softmax layer and $f(h_i)$ is the probability that the observation sample belongs to class i.

2.4.3 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a type of RNN that is particularly effective in modelling sequential or time series data. In credit risk, LSTMs can be used to model credit transactional data. The LSTM model architecture consists of one or more LSTM layers, followed by one or more fully connected (dense) layers for classification. It uses memory cells and gates to capture long-term dependencies to make accurate predictions. The input gate determines which information should be updated. The *forget* gate determines what to forget from previous memory (Toharudin et al., 2023). The output gate regulates the information that is to be emitted as output. The fundamental unit of the LSTM is the *memory cell* or *cell state*, denoted as $C(t)$, which retains information over time. This can be thought of as a conveyor belt that takes in new information at each time step and updates its internal state.

The process that information flows through in an LSTM network can be covered across a few steps. All equations used are referenced from Ala'raj et al. (2021). The high-level architecture is shown in Figure 2.9.

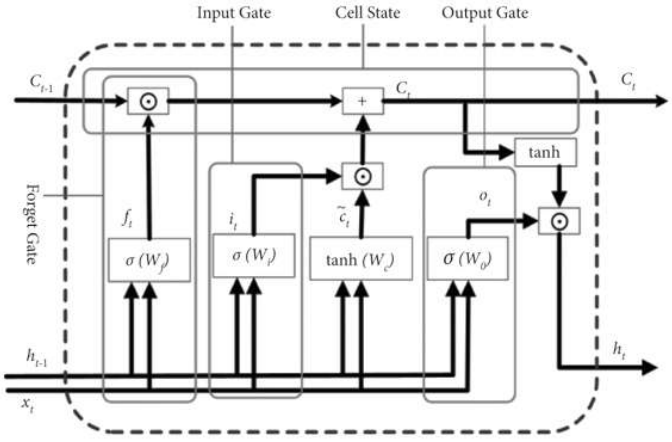

Figure 2.9: Architecture of LSTMs (Liu et al., 2022).

LSTMs process inputs and generate outputs in vector form. Input data is fed into the network sequentially, one element at a time. At each time step, the LSTM network

determines which information from the previous hidden state to retain. The forget gate, functioning as a sigmoid function, evaluates the current input alongside the previous hidden state, producing values between 0 and 1 for each element within the hidden state. This mechanism enables the forget gate to determine which information should be discarded from the previous hidden state. The forget gate $f(t)$, where t represents the time step, is calculated as follows:

$$f(t) = \sigma(W_f \cdot [h(t-1),x(t)] + b_f), \tag{2.28}$$

where σ represents the sigmoid activation function, W_f denotes the weight associated with the forget gate, $h(t-1)$ represents the hidden state output at time step $t-1$, $x(t)$ denotes the input at time step t, and b_f signifies the bias term of the forget gate.

Subsequently, an input gate is employed to determine which information should be modified in the hidden state. This gate considers both the input at the current time step and the previous hidden state. Similar to the forget gate, it produces a value between 0 and 1 for each element within the hidden state. The equation for the input gate $i(t)$ is:

$$i(t) = \sigma(W_i \cdot [h(t-1),x(t)] + b_i), \tag{2.29}$$

where W_i denotes the weight parameter, while b_i represents the bias parameter associated with the input gate. A candidate vector of new information is created that could be added to the hidden state. This process is done using the hyperbolic tangent function that condenses the values between -1 and 1.

$$\tilde{C}(t) = tanh(W_C \cdot [h(t-1),x(t)] + b_C), \tag{2.30}$$

where W_C and b_C are the weight and bias of the cell state, respectively. In the next step, the LSTM updates the current hidden state by integrating information from both the previous hidden state and the newly calculated candidate information, guided by the forget and input gates as follows:

$$C(t) = f(t) \cdot C(t-1) + i(t) \cdot \tilde{C}(t), \tag{2.31}$$

Finally, the LSTM decides what part of the hidden state to output as the current output through the output gate. The equation for the output gate is:

$$o(t) = \sigma(W_o \cdot [h(t-1),x(t)] + b_o), \tag{2.32}$$

where W_o represents the weight parameter, while b_o denotes the bias parameter associated with the output gate. The output gate is applied through a tanh function to get the updated hidden state which is then used to get the final output:

$$h(t) = o(t) \cdot C(t). \tag{2.33}$$

Activation Functions in LSTM

LSTMs use specific activation functions within their gates and cell state computations. The following are the most frequently used activation functions in LSTM:

- Sigmoid Function: The sigmoid function is used in the input gate, forget gate, and output gate to condense the values between 0 and 1. It manages the information flow by adjusting the activation of the gates.
- Hyperbolic Tangent Function: The hyperbolic tangent function is used to compute the candidate values that can be added to the cell state. It compresses the values between -1 and 1, capturing the potential updates to the memory.

2.4.4 Gated Recurrent Unit

Gated Recurrent Units (GRUs) are another variant of RNNs that can perform similarly to LSTMs while being computationally more efficient. They are used for prediction in scenarios where LSTM models might be too resource-intensive. Like LSTMs, GRUs are predominantly used for sequential data. GRUs are known for capturing long-term dependencies in sequential data in comparison to traditional RNNs (Wang et al., 2018b). The network also uses fewer parameters compared to complex RNNs like LSTMs, and are thus easier to train due to their simpler architecture.

The architecture of the GRU network is shown in Figure 2.10.

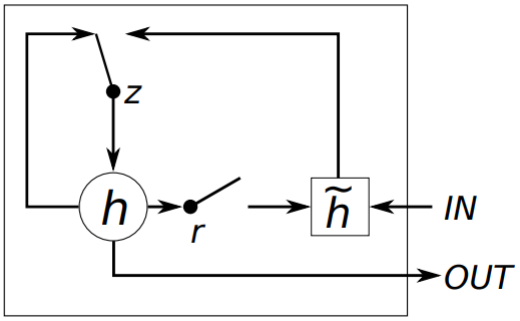

Figure 2.10: Architecture of GRUs (Chung et al., 2014).

The process GRUs follow is determined by gates which combine and filter information. All equations used are referenced from Dey and Salem (2017).

Initially, the update gate determines the degree to which the previous hidden state is merged with the candidate hidden state to generate the new current hidden state. This is mathematically defined as follows:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z). \tag{2.34}$$

This gate receives the previous hidden state h_{t-1} and the current input x_t, then subjects them to a linear transformation denoted as W_z. The parameter b_z is the bias and U_z is a parameter matrix of the update gate. The applied activation function σ is the sigmoid function.

Next, the *reset* gate r_t regulates the extent to which the previous hidden state should be disregarded when calculating the candidate hidden state. The equation for the reset gate is:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r). \tag{2.35}$$

The candidate hidden state is then computed by the following equation:

$$\tilde{h}_t = tanh(W_h x_t + U_h(r_t \cdot h_{t-1}) + b_h). \tag{2.36}$$

The reset gate regulates the extent to which the previous hidden state is taken into account, and the outcome is then processed through a hyperbolic tangent (tanh) function to form the candidate hidden state.

Ultimately, the current hidden state is computed by combining together the previous hidden state and the candidate hidden state. The hidden state h_t at time t can be determined using the previous hidden state h_{t-1} using the following equation:

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t, \tag{2.37}$$

where z_t is denoted as the update gate and $\tilde{h}_t$ is the candidate hidden state.

Activation Functions in GRUs

- Sigmoid: This activation function compresses the input values within the range of 0 to 1, enabling the gates to manage the flow of information.

- Hyperbolic Tangent (tanh): This activation function compresses the input values within the range of -1 to 1, aiding in controlling the values of the candidate hidden state.

2.4.5 Radial Basis Function Neural Network

RBFNNs can be thought of as a feedforward network of artificial neurons that uses radial basis functions to make predictions. Each neuron in an RBFNN takes input values and calculates its output based on a radial basis function. RBFNNs consist of three main layers: input layer, hidden layer, and output layer. The input layer consists of neurons that represent the input variables. The hidden layer is where the radial basis functions are applied. The output layer is where prediction outcomes are determined. This layer aggregates the outputs of the hidden layer neurons using weights to generate the final output. The architecture of a typical RBFNN is shown in Figure 2.11.

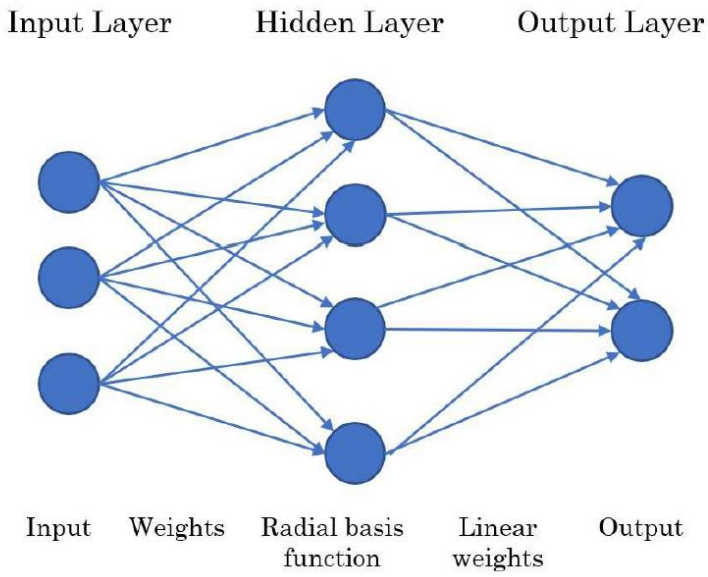

Figure 2.11: Architecture of RBFNNs (Chai et al., 2019).

The radial basis function calculates the similarity between the input values and a set of reference points called centroids. The similarity is commonly assessed using a distance metric such as the Euclidean distance. The output of the neuron is a weighted combination of the similarity values, where the weights represent the importance of each centroid (Bekhet and Eletter, 2014).

For an input x_i, the output $y(x_i)$ of the RBF network is given by the following equation (Wu et al., 2012):

$$y(x_i) = \sum w_i \Phi(||x - c_i||), \tag{2.38}$$

where $\Phi()$ is the applied radial basis function, $||x - c_i||$ is the Euclidean distance between x and c_i, w_i is the weight associated with the $i - th$ hidden neuron and c_i is the center point associated with the RBF.

The hidden layer output h_i is calculated using a radial basis function using the following equation:

$$h_i = \Phi(||x - c_i||). \tag{2.39}$$

Activation Functions in RBFNN

The main activation function used in RBFNNs is the radial basis function itself. It calculates the similarity or distance between the input values and the centroids (reference points) in the hidden layer. The RBF function assigns higher values to inputs that are closer to the centroids and lower values to inputs that are farther away. There are a few types of radial basis functions that can be used. All functions used are referenced from Satapathy et al. (2019).

- Gaussian: This is the most widely used radial basis function for RBFNN in practice. The Gaussian RBF is defined as a bell-shaped curve centered around a specific point. The equation is defined as follows:

$$\Phi(x_i) = \exp\left\{\frac{x - c_i^2}{\sigma^2}\right\}, \tag{2.40}$$

  where $||x - c_i||$ is the Euclidean distance between x and c_i and the spread of the Gaussian RBF is changed through the size of σ.

- Multiquadric: The multiquadric RBF is frequently employed and is calculated as the square root of the sum of squared differences between the input x and the centre point c_i augmented by σ^2 which governs the curve's shape. The equation is defined as follows:

$$\Phi(x_i) = \sqrt{\sigma^2 + x - c_i^2}. \tag{2.41}$$

- Inverse Multiquadric: The inverse multiquadric RBF is the inverse of the square root of the sum of the squared differences between the input x and the centre point c_i. This is given as:

$$\Phi(x_i) = \frac{1}{\sqrt{\sigma^2 + x - c_i^2}}. \tag{2.42}$$

2.4.6 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of NN specifically designed for processing structured grid-like data, such as images (Sharan et al., 2021). They are composed of various layers, such as pooling layers, convolutional layers, and fully connected layers. CNNs learn to automatically extract meaningful features from the input data through the convolution operation, which involves sliding small filters, known as *kernels*, over the input and capturing patterns or features. Pooling layers decrease the size of the output, reducing spatial dimensions while retaining essential features. These extracted features are subsequently fed into fully connected layers for tasks such as classification or regression.

The way CNNs work can be explained by the following steps.

1. The first step is the convolution operation that consists of sliding a kernel over the input to extract different attributes. The kernel performs element-wise multiplication with the local regions of the input and then sums up the results to create a map of features. The equation for this operation to give the output A_j is given as follows at a location on the map (Li et al., 2014):

$$A_j = f(\sum_{i=1}^{N} I_i \times K_{i,j} + B_j), \tag{2.43}$$

   where I_i is the value of the input, $K_{i,j}$ is the value of the filter or kernel, B_j represents the bias term and f denotes an activation function applied to introduce non-linearity to the output, allowing the network to capture more complex patterns.

2. The next step is the pooling operation which includes reducing the spatial dimensions of the feature map while keeping important information. Max-pooling is the most commonly used pooling which takes the maximum value within a local area on the feature map then discards the rest (Wang et al., 2012). The other type is average pooling which then takes the average value within a local area on the feature map.

3. Following multiple convolutional and pooling layers, the network produces one or more fully connected layers. These layers process the reduced and abstracted features and then perform classification or regression operations based on the learned representations.

Activation Functions in CNN

The activation functions commonly used in CNNs are:

- ReLU: ReLU is the predominant activation function utilised in CNNs. It effectively converts all negative input values to zero while keeping positive values

unchanged. ReLU helps introduce non-linearity, allowing CNNs to learn complex relationships and speeding up the training process. These enable enhancement of the discriminative capability of the CNN network. (Nair and Hinton, 2010).

- Sigmoid Function: The sigmoid activation function transforms the input into a value ranging from 0 to 1. It is often used in the final layer of a CNN for binary classification tasks, where the output represents a probability of belonging to one of the classes (LeCun et al., 1998).
- Softmax Function: this activation function is typically applied in the final layer of NNs for multi-class classification purposes. It transforms the output values into a probability distribution, guaranteeing that the sum of probabilities for all classes equals 1. Softmax helps in selecting the most likely class among multiple choices.

2.5 Methods of Model Evaluation

Evaluation of a model's performance is a crucial part of prediction and analysis to understand the limitations of performance and to determine how well a model can predict an outcome. This section covers the most common methods used in evaluating statistical and machine learning models for classification.

2.5.1 Confusion Matrix

The Confusion Matrix, developed as a contingency table by Pearson (1904), is a performance evaluation tool for statistical classification models particularly for supervised learning. It consists of a contingency grid containing information about actual and predicted values of a classification model as shown in Figure 2.12. The matrix allows the modeller to calculate metrics such as recall, accuracy, precision, specificity and others (Bénédict et al., 2021).

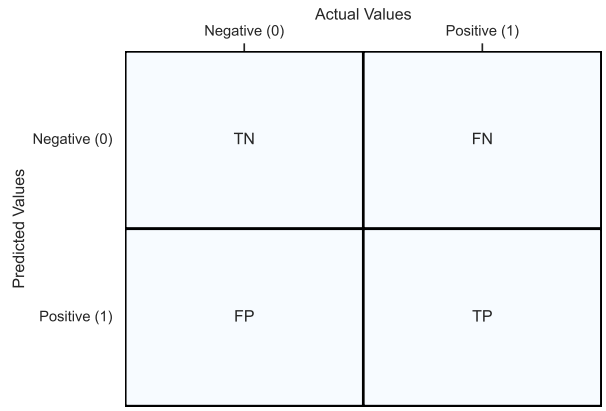

Figure 2.12: Confusion Matrix representation for classification (Narkhede, 2018).

For binary classification, a target variable can have a positive and negative class. Then the following definitions exist:

- True Positives (TP): the number of correctly predicted observations belonging to the positive class.
- False Positives (FP): the number of incorrectly predicted observations belonging to the positive class.
- True Negatives (TN): the number of correctly predicted observations belonging to the negative class.
- False Negatives (FN): the number of incorrectly predicted observations belonging to the negative class.

Using a Confusion Matrix, there are various performance metrics that can be calculated to mathematically describe the performance of a classification model. The equations used in this section are referenced from Kohavi (1998).

Accuracy

Accuracy is defined as the proportion of correct predictions as a percentage of the total number of predictions made by a model. It can be calculated as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \times 100. \tag{2.44}$$

Precision

Precision is the proportion of true positives as a percentage of the total predictions made such that the predictions were for a specific class. The purpose is to determine to what degree a model can correctly predict a specific class in relation to its total predictions for that class. It is calculated using the following equation:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \times 100. \tag{2.45}$$

Recall

Recall, also known as model sensitivity, is defined as the proportion of true positives as a percentage of the total actual positives predicted. For example, if a dataset contains two target classes, *good* and *bad*, recall allows one to determine how many other observations were predicted to be of the class *good* but actually belonged to class *bad*. The formula for recall is as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \times 100. \tag{2.46}$$

Specificity

Specificity is the proportion of true negatives as a percentage of the total actual negatives predicted. It is computed by the following equation:

$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \times 100. \tag{2.47}$$

F1 Score

The F1 Score combines precision and recall by taking harmonic the mean of both metrics. This is used to understand accuracy for a model in cases where the metrics vary. For example, one model could have a higher recall but another comparative model could have a higher precision. In this case, the F1 score combines both metrics such that both are taken into consideration. The calculation of the metric is as follows:

$$\text{F1 Score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{2.48}$$

All of the aforementioned measures of evaluation should be as high as possible for a good model.

2.5.2 Receiver Operating Characteristic (ROC)

The ROC curve is defined as a graph of the recall or sensitivity (true positive rate) versus one minus the specificity (false positive rate) and is used to evaluate the diagnostic ability of a classification model.

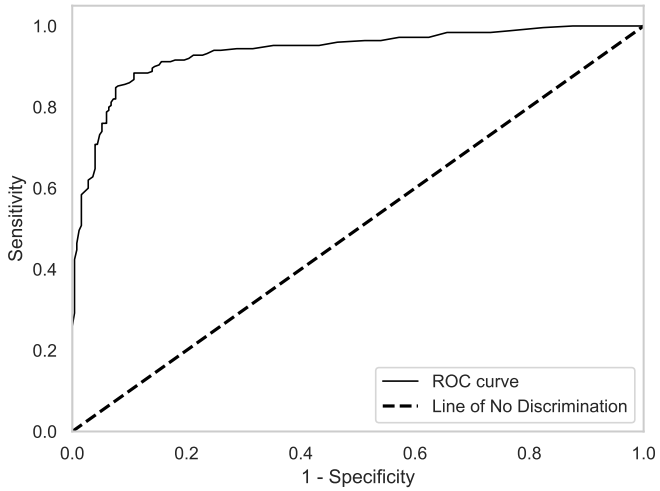

Figure 2.13: Receiver Operating Chararacteristic curve (Bewick et al., 2004).

Referring to Figure 2.13, the ideal curve would be a vertical line along the sensitivity axis and a horizontal line connecting the vertical line, parallel to the one minus specificity axis. This would mean a model can accurately predict observations belonging to the positive class correctly. In reality, models will not follow this pattern. An unsatisfactory model would have a curve lying below the diagonal dotted line. Model performance will thus improve if the curve is skewed towards the upper left of the graph.

Area Under the Curve (AUC)

The area under the ROC curve (AUC-ROC) can also be evaluated for performance. As stated, if the ROC curve is skewed toward the upper left of the graph in Figure 2.13, the the model performance is better than that of one that lies toward the diagonal random predictor line. The ideal area would then be equal to one. Therefore, the larger AUC-ROC, the better the model performance in predicting outcomes.

2.5.3 Log Loss

Log loss is an accuracy metric used for models that determine the probability of an observation belonging to a particular class such as Logistic Regression. The metric indicates how far away each predicted probability is from the actual class. It is calculated using the following equation (Aggarwal et al., 2021):

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \ln p_{ij}, \tag{2.49}$$

where N is the number of observations in the test set, M is the number of class variables, y_{ij} is equal to 1 if the observation i belongs to class j and is 0 otherwise, and p_{ij} is the predicted probability that the observation i belongs to class j.

The lower the log loss score is, the higher a model's accuracy will be.

2.5.4 Jaccard Index

The Jaccard Index is an intersection over union ratio of the number of correctly predicted values to the sum of the wrongly predicted values and the total actual positives. The higher the Jaccard Index is, the higher the model performance will be. The measure is calculated as follows (Eelbode et al., 2020):

$$J(y,\hat{y}) = \frac{y \cap \hat{y}}{y \cup \hat{y}}, \tag{2.50}$$

where y represents the actual values and $\hat{y}$ represents the predicted values.

Chapter 3

Methodology

This chapter outlines the dataset selected for the study and the corresponding data pre-processing procedures. Additionally, it presents the findings of exploratory data analysis. Furthermore, the methods utilised to model the data are explained in detail.

A summary of the methodology which is used in this study is shown in Figure 3.1.

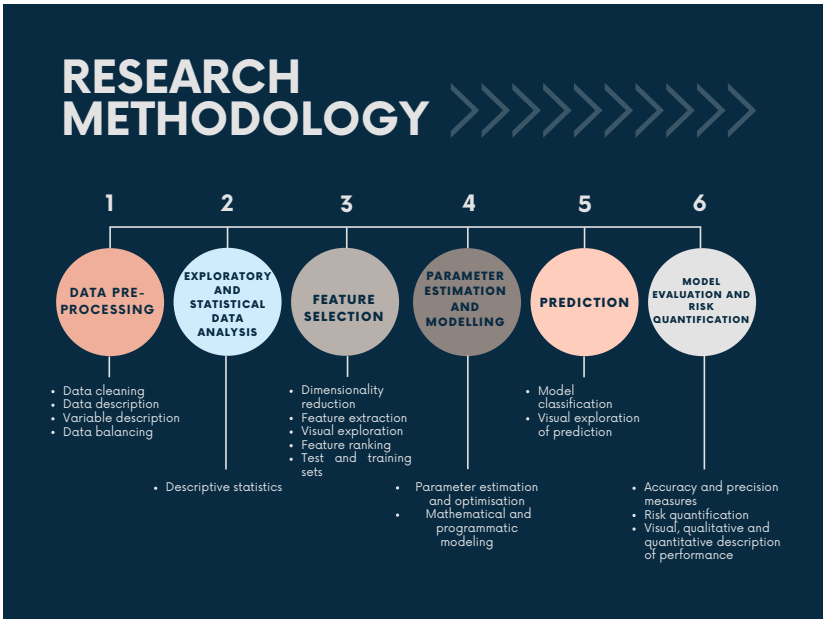

Figure 3.1: Research methodology applied to the analysis and prediction of credit registry data.

3.1 Data Pre-processing

The dataset used for modelling in this study was sourced from Society for Data Science (2020), a platform known for hosting published datasets applicable in the field of data science for purposes such as modelling and prediction. Specifically, the dataset consists of credit registry data from a bank in South Germany.

3.1.1 Background of Data

The dataset originates from a commercial bank in South Germany and comprises 1 000 credit observations spanning from 1973 to 1975. Initially contributed by German professor Hans Hofmann in 1994 through the European Stratlog project, the dataset was found to have coding errors, which were rectified as detailed by Groemping (2019). The dataset contains 700 observations classified as representing good risk customers and 300 observations classified as representing bad risk customers, indicating an imbalance in the dataset. No missing values were identified within any of the records. Good risk customer profiles are defined as those who adhered to the conditions outlined in the credit contract, while bad risk customer profiles are characterised by non-compliance with these conditions.

3.1.2 Description of Data

The dataset is labelled and comprises 20 explanatory predictor variables. Among these variables, 7 are quantitative, with 4 being discretised into ordinal levels and the remaining 3 available in their original units. Additionally, there are 13 categorical variables. The target variable, denoted as *credit_risk* in the dataset, represents the credit risk and is a binary variable indicating whether the contract has been complied with (good risk) or not (bad risk). The dataset was imported into Python as a comma-separated value file and loaded into a *dataframe* using the *pandas* package. Prior to analysis, the variable names were translated from German to English, as the original dataset was in German.

The list of predictor variables are given as follows:

- *status*: status of the debtor's checking account with the bank (categorical).
- *credit_history*: history of compliance with previous or concurrent credit contracts (categorical);
- *purpose*: purpose for which the credit is needed (categorical);
- *savings*: debtor's savings (categorical);
- *employment_duration*: duration of debtor's employment with current employer (ordinal; discretised quantitative);
- *installment_rate*: credit installments as a percentage of debtor's disposable income (ordinal; discretised quantitative);

- *personal_status_sex*: combined information on sex and marital status. Sex cannot be recovered from the variable, because male singles and female non-singles are coded with the same code (2). Female widows cannot be easily classified because the code table does not list them in any of the female categories (categorical);
- *other_debtors*: specifies if there is another debtor or a guarantor for the credit (categorical);
- *property*: the debtor's most valuable property, i.e. the highest possible code is used. Code 2 is used if codes 3 or 4 are not applicable and there is a car or any other relevant property that does not fall under the variable *savings* (ordinal);
- *other_installment_plans*: installment plans from providers other than the credit-giving bank (categorical);
- *housing*: type of housing the debtor lives in (categorical);
- *number_credits*: number of credits including the current one the debtor has (or had) at this bank (ordinal, discretised quantitative);
- *job*: quality of debtor's job (ordinal);
- *people_liable*: number of persons who financially depend on the debtor or are entitled to maintenance (binary, discretised quantitative);
- *telephone*: specifies if is there a telephone landline registered on the debtor's name (binary);
- *foreign_worker*: specifies if the debtor a foreign worker (binary);
- *duration*: credit duration in months (quantitative);
- *amount*: credit amount in Deutsche Mark (DM) (quantitative);
- *age*: age in years (quantitative).

Table 3.1 presents a comprehensive list of variables along with their respective categories. Among these variables, three are quantitative and retain their original units: *duration*, *amount* and *age*. The remaining variables are categorical. In the raw dataset, categories are represented by codes rather than textual descriptions, which are provided in the table. The final variable, *credit_risk*, serves as the target variable and is encoded as 1 for good risk and 0 for bad risk. It is noteworthy that certain categories include monetary values denominated in the German currency Deutsche Mark (DM), which was in use in Germany from 1948 to 1990.

Table 3.1: Different categories for the list of variables in the credit dataset.

Variable	**Code**	**Category**
status	1	no checking account
	2	...<0 DM
	3	0≤...<200 DM
	4	≥200 DM/salary for at least 1 year
credit_history	0	delay in paying off in the past
	1	critical account/other credits elsewhere
	2	no credits taken/all credits paid back duly
	3	existing credits paid back duly till now
	4	all credits at this bank paid back duly
purpose	0	others
	1	car (new)
	2	car (used)
	3	furniture/equipment
	4	radio/television
	5	domestic appliances
	6	repairs
	7	education
	8	vacation
	9	retraining
	10	business
savings	1	unknown/no savings account
	2	...<100 DM
	3	100≤...<500 DM
	4	500≤...<1000 DM
	5	≥1000 DM
employment_duration	1	unemployed
	2	<1 year
	3	1≤...<4 years
	4	4≤...<7 years
	5	≥7 years
installment_rate	1	≥35
	2	25≤...<35
	3	20≤...<25
	4	<20
personal_status_sex	1	male: divorced/separated
	2	female: non-single or male: single
	3	male: married/widowed
	4	female: single
other_debtors	1	none
	2	co-applicant
	3	guarantor

present_residence	1	<1 year
	2	1≤...<4 years
	3	4≤...<7 years
	4	≥7 years
property	1	unknown/no property
	2	car or other
	3	building soc. savings agr./ life insurance
	4	real estate
other_installment_plans	1	bank
	2	stores
	3	none
housing	1	for free
	2	rent
	3	own
number_credits	1	1
	2	2-3
	3	4-5
	4	≥6
job	1	unemployed/unskilled - non-resident
	2	unskilled - resident
	3	skilled employee/official
	4	manager/self-empl./highly qualified employee
people_liable	1	0-2
	2	≥3
telephone	1	no
	2	yes (under customer name)
foreign_worker	1	no
	2	yes
duration	N/A	N/A
amount	N/A	N/A
age	N/A	N/A
credit_risk	0	bad risk
	1	good risk

3.2 Exploratory and Statistical Data Analysis

3.2.1 Descriptive Statistics

The *value_counts()* function in Python was used to determine the distribution of categories for each categorical feature. These values are given in Table 3.2. Using the *describe()* function, the mean, standard deviations and percentiles were produced for

the quantitative variables *amount*, *duration* and *age*. These values are provided in Table 3.3.

The distribution of categories for each categorical feature was assessed in Python, and the results are summarised in Table 3.2 Additionally, descriptive statistics including mean, standard deviation, and percentiles were computed for the quantitative variables *amount*, *duration* and *age*. These statistics are presented in Table 3.3.

Table 3.2: Percentage of each category for categorical features as a total of the observations and good or bad credit risk.

			Credit Risk (Percent)	
Variable	**Code**	**Count (Percent)**	**Bad (0)**	**Good (1)**
status	1	27,4	45,0	19,9
	2	26,9	35,0	23,4
	3	6,3	4,7	7,0
	4	39,4	15,3	49,7
credit_history	0	4,0	8,3	2,1
	1	4,9	9,3	3,0
	2	53,0	56,3	51,6
	3	8,8	9,3	8,6
	4	29,3	16,7	34,7
purpose	0	23,4	29,7	20,7
	1	10,3	5,7	12,3
	2	18,1	19,3	17,6
	3	28,0	20,7	31,1
	4	1,2	1,3	1,1
	5	2,2	2,7	2,0
	6	5,0	7,3	4,0
	7	0,0	0,0	0,0
	8	0,9	0,3	1,1
	9	9,7	11,3	9,0
	10	1,2	1,7	1,0
savings	1	60,3	72,3	55,1
	2	10,3	11,3	9,6
	3	6,3	3,7	7,4
	4	4,8	2,0	6,0
	5	18,3	10,7	21,6
employment_duration	1	6,2	7,7	5,6
	2	17,2	23,3	14,6
	3	33,9	34,7	33,6
	4	17,4	13,0	19,3
	5	25,3	21,3	27,0

personal_status_sex	1	5,0	6,7	4,3
	2	31,0	36,3	28,7
	3	54,8	48,7	57,4
	4	9,2	8,3	9,6
installment_rate	1	13,6	11,3	14,6
	2	23,1	20,7	24,1
	3	15,7	15,0	16,0
	4	47,6	53,0	45,3
other_debtors	1	90,7	90,7	90,7
	2	4,1	6,0	3,3
	3	5,2	3,3	6,0
present_residence	1	13,0	12,0	13,4
	2	30,8	32,3	30,1
	3	14,9	14,3	15,1
	4	41,3	41,3	41,3
property	1	28,2	20,0	31,7
	2	23,2	23,7	23,0
	3	33,2	34,0	32,9
	4	15,4	22,3	12,4
other_installment_plans	1	13,9	19,0	11,7
	2	4,7	6,3	4,0
	3	81,4	74,7	84,3
housing	1	17,9	23,3	15,6
	2	71,4	62,0	75,4
	3	10,7	14,7	9,0
number_credits	1	63,3	66,7	61,9
	2	33,3	30,7	34,3
	3	2,8	2,0	3,1
	4	0,6	0,7	0,6
job	1	2,2	2,3	2,1
	2	20,0	18,7	20,6
	3	63,0	62,0	63,4
	4	14,8	17,0	13,9
people_liable	1	84,5	84,7	84,4
	2	15,5	15,3	15,6
telephone	1	59,6	62,3	58,4
	2	40,4	37,7	41,6
foreign_worker	1	96,3	98,7	95,3
	2	3,7	1,3	4,7

The following descriptive statistics were observed for each feature:

- *status*: For the *status* feature, 45% and 35% of the total bad risk make up customers in the category "no checking account" and "...<0DM". This is expected as customers who have no debit account or a negative balance currently, would not have any liquid capital to pay back credit and would be considered risky to the bank. On the contrary, 49,7% of the total good risk make up customers who had more than 200DM or at least a one year's salary. However, it is unusual to note that only 7% of the total good risk make up customers who had between 0DM and 200DM, in comparison to 19,9% and 23,4% who make up customers who had a negative account balance or no checking account, respectively. Also a similar pattern is observed for the bad risk totals where 15,3% of the total make up customers who have more than 200DM on account, where only 4,7% of the total had between 0DM and 200DM. It must be noted that only 6,3% of the total between good and bad risk consist of customers that had between 0DM and 200DM on account. Of the entire total, 39,4% of the dataset make up customers who have more than 200DM or at least one year's salary on account. It is worthwhile to state that the percentage totals cannot be fairly compared between good and bad risk since the dataset is imbalanced with the good risk totals making up the major class.

- *credit_history*: 56,3% of the total bad risk make up customers who had taken no credit or had paid back all credits in full. This is expected as the customer would have no history or current credit for the bank to assess. 51,6% of the total bad risk make up customers who had taken no credit or had paid back all credits in full. Although these percentages are not fairly comparable, it is worthwhile to note that these classes make up majority for both bad and good risk. 34% of the good risk are customers that had paid back all credits duly in the current bank. Conversely, 16,7% of the bad risk total make up customers who ha paid up all credit in full at the current bank, which also indicates that the bank used other information to assess the customer's risk and not solely on whether the customer had paid back all credit or not.

- *purpose*: For this feature, the categories that make up most of the customers between both bad and good risk are "others", "car(used)" and "furniture/equipment" of 23,4%, 18% and 28,0%, respectively. The category "others" is not very descriptive of purpose of the credit and could represent a variety of credits for which the contract was taken.

- *savings*: The category for this variable that makes up the majority of observations between good and bad risk is "unknown/no savings account". Of the total bad risk observations 72,3% fall within this category and of good risk, 55,1% fall within the same category.

- *employment_duration*: For this feature, the statistics show a middle ground category making up 33,9% of the total observations of a customer being employed

between one and four years. In this category, the distributions for good and bad risk are somewhat equal indicating that between one and four years, a customer was equally likely to be a good risk or a bad risk to the bank. In the categories below this, specifically for good, a trend is noticed that the less time a customer is employed, the lower the chances of being a good risk to the bank. On the categories above, the percentage distribution increases for good risk as the years a customer is employed increases. This represents positive correlation between the good risk target variable and the *employment_duration* feature. This positive correlation does not exist for the bad risk target variable and distributions vary. For example, the category of "<1 year" has a distribution of 23,3% and similar distribution of 21,3% exists for the category of having an employment duration of more than seven years.

- *personal_status_sex*: The categories in this feature incorporate multiple descriptions per category. For example, the code 2 represents non-single females but also single males. It is difficult to attribute a particular customer description to the type of risk. In the total dataset, 54,8% of all observations make up customers who are male and who are married or widowed.

- *installment_rate*: For this feature, more than half of the total observations which have a classification of bad risk were within in the installment rate category of "< 20". The same pattern is followed for the good risk category.

- *other_debtors*: For both good and bad risk, 90,7% of observations make up the category of "none". In other words, the vast majority of customers classified took the credit contract individually. This is an indication that this feature is not a good classifier to the target class.

- *present_residence*: The normalised distributions between good and bad risk as a percentage of individual totals, can be seen to be fairly balanced between the two target classes. This may indicate the this feature is not a good classifier to the target class as the same patterns occur for both good and bad risk, and this is seen across all categories in Table 3.4. For example, a customer residing in the current residence for more than seven years was equally as likely to have bad risk as the customer was to having good risk since the distributions across both are 41,3%.

- *property*: The distributions per risk class for this feature are fairly balanced for categories 2 and 3, which are "car or other" and "building society. savings agr./life insurance", respectively. For the "real estate" category, the percentage of observations making up the bad risk category is 22,4% which is more than the 12,4% making up the good risk category. This may indicate that customers who owned real estate properties had a higher chance of defaulting than those who did not. For the "unknown/no property" category, 31,7% make up the distribution of observations in the good risk class, which is 10% more than the distribution in the bad risk class. This means the likelihood of defaulting on credit was higher if the

customer owned no property.

- *other_installment_plans*: The category in this feature that makes up the largest distribution between both good and bad risk is "none". This means that the customers who have no other installment plans other than at the current bank made up the major class for this feature. The distribution of bad risk for the same category is 74,7% and the distribution of good risk is approximately 10% more. The "stores" category make up a small percentage for this feature and are similar for both target classes. For the "bank" category, it is noticed that the distribution for bad risk is approximately 8% higher than the distribution for good risk. This means the likelihood of defaulting against credit at the current bank was higher if the customer had an installment plan taken out at another bank.

- *housing*: The major category for this feature is "rent" with a total percentage distribution of 71,4%. The distribution is similar between good and bad risk with a difference of approximately 13%. For good risk, this means that a customer was more likely to have been a good risk to the bank if they were renting rather than owning a residence or living at no cost. The same pattern, however, also occurred for bad risk customers.

- *number_credits*: For this feature, customers who had only one credit contract taken out made up the most for both good and bad risk. The distributions across each category are similar which also indicate this feature may not be a good predictive variable to credit risk for this dataset.

- *job*: The category that makes up the major distribution for this feature is "skilled employee/official" with 63,0% and the minor distribution, "unemployed/unskilled - non-resident" with 2,2%. This is expected as a customer would need a form of income in order to pay off credit. The distributions are similar across all categories for good and bad risk also indicating that this feature is not a good predictive variable to credit risk for this dataset.

- *people_liable*: This feature has only two categories, where "0-2" make up 84,5% across both target classes, and 15,5% make up the category "≥ 3". The distributions are similar across both categories for good and bad risk again indicating that this feature is not a good predictive variable to credit risk for this dataset.

- *telephone*: This feature has two categories, where 59,6%, across both target classes, are made up of observations for customers who did not own a telephone, and 40,4% make up the category "≥ 3". The distributions are similar across both categories for good and bad risk with approximately a 4% difference, indicating that this feature is not a good predictive variable to credit risk for this dataset.

- *foreign_worker*: 96,3% of the observations are made up of customers who were not foreign workers. The distribution is similar across both target classes for those who were not foreign workers. Only 3,7% of observations are made up of customers who were foreign workers. The distributions also indicate that if a customer was

a foreign worker, then he or she was more likely to have good risk.

Table 3.3: Descriptive statistics for quantitative variables in the credit risk dataset.

Variable	**Mean**	**25%**	**50%**	**75%**	**Min**	**Max**
duration (months)	20,9	12,0	18,0	24,0	4,0	72,0
amount (DM)	3271,2	1365,0	2319,5	3972,3	250,0	18424,0
age (years)	35,5	27,0	33,0	42,0	19,0	75,0

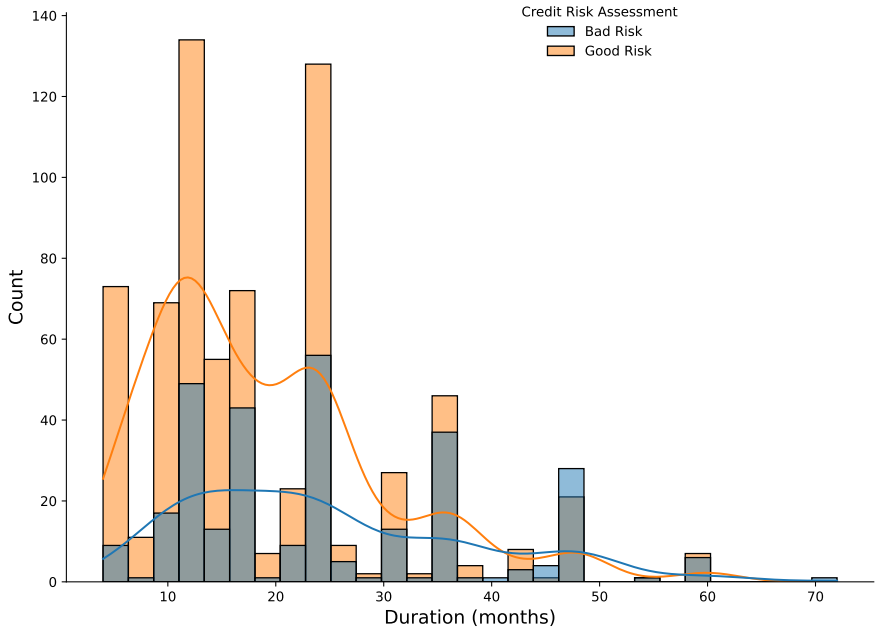

Figure 3.2: Distribution of the *duration* variable for the credit risk dataset between good (1) and bad (0) risk.

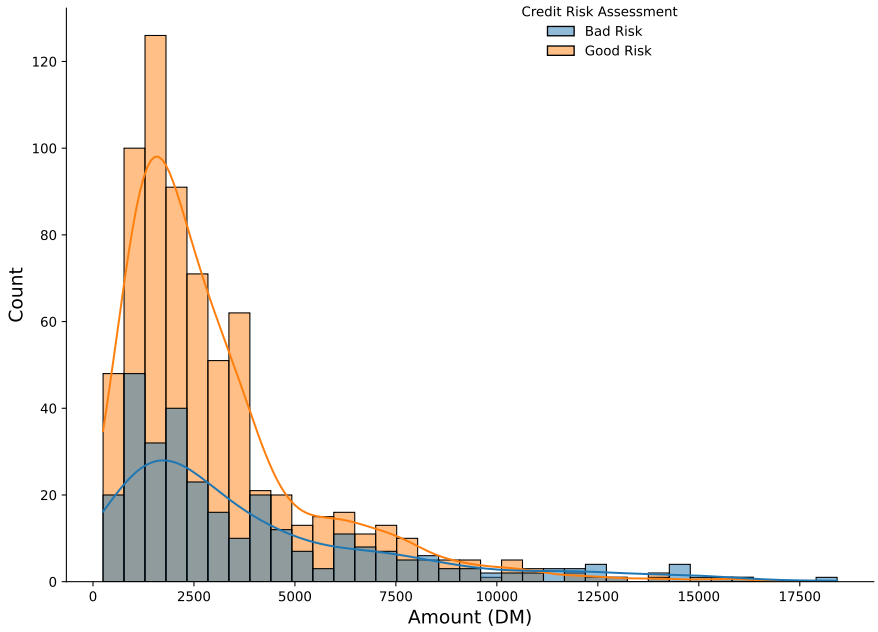

Figure 3.3: Distribution of the *amount* variable for the credit risk dataset between good (1) and bad (0) risk.

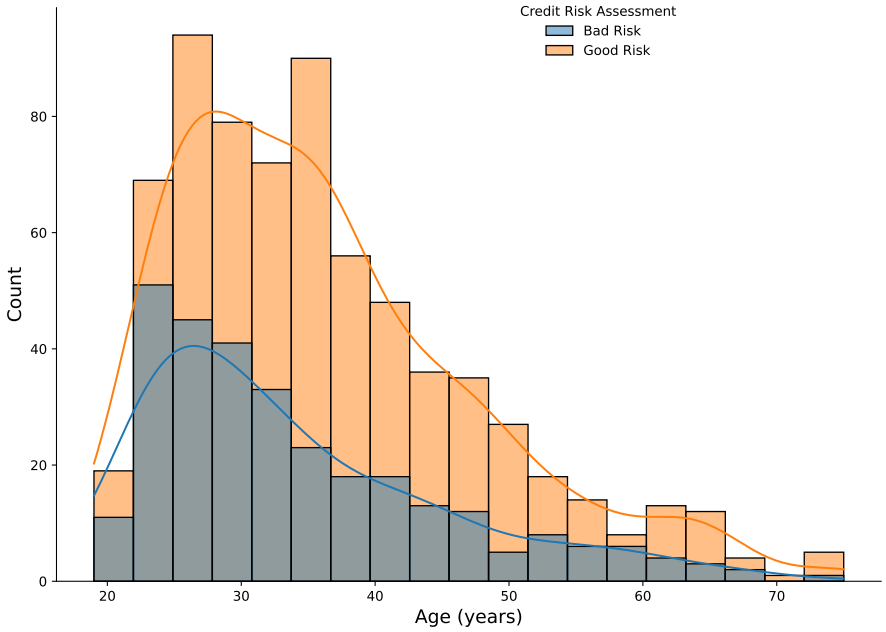

Figure 3.4: Distribution of the *age* variable for the credit risk dataset between good (1) and bad (0) risk.

Referring to Table 3.3, the distribution for the quantitative variable *duration* indicates that 50% of the customers in the dataset had a credit duration of 18 months or less, while the other 50% had a credit duration of 18 to 72 months. The average duration was around 20,9 months. Referring to individual distributions for the good and bad risk classes, Figure 3.2, the distributions for bad risk are more spread out across the duration values but is also concentrated around the 15 to 20 month duration. Conversely, for good risk, there is a skewness towards lower duration values. This indicates that customers who were classified as good risk also took out a credit contract with a duration that was lower. The distributions for both classes do have noise, which makes it difficult to find correlations between this variable and the target class.

In Table 3.3, the statistics show that for the *amount* variable, 50% of the customers took out a credit contract with an amount of 2319,5DM or lower. The distribution curve in Figure 3.3 is smoother and more distinct in comparison to the *duration* variable. It can be seen that customers were more likely to have good risk at lower credit amounts and the likelihood decreases significantly when the amount was increased. A similar pattern occurs for bad risk, and as expected, higher credit amounts were associated with a customer having bad risk.

For the *age* variable, 50% of the distribution are customers that were below the age of 33 with a minimum age of 19 years, the other 50% were above the age of 33 with a maximum age of 75. Thus, the variation in age is spread out evenly about an average age of 35,5. Referring to Figure 2.13, the distribution lies toward the age of 27 for both bad and good risk. The distribution declines with an increase in age.

3.3 Feature Selection

In this section, a methodology is outlined for extracting the most significant independent variables from the dataset. Additionally, graphical representations will be provided for these selected features to visually investigate key aspects and relationships among variables within the dataset.

3.3.1 Chi-square Statistics

For feature selection in the dataset, the filter method utilising Chi-square Statistics was employed. This method was deemed suitable due to its applicability to datasets containing categorical data. The Chi-square Statistics test aims to assess the degree of dependence between the class and target variables. It was chosen specifically for its ability to operate without requiring homoscedasticity within the data, its minimal assumption requirement, and its classification as a non-parametric method.

Since the Chi-squared Statistics test is non-parametric, there are a few conditions that the dataset must adhere to.

Conditions for Non-parametric Tests

- The data to which the test is applied must be measured in levels which are nominal or ordinal.
- If the dataset contains variables that were originally measured as continuous data, the variables must have violated one of the following assumptions of parametric tests:
  - The data is homescedastistic in nature. This is also known as the assumption of equal variance.
  - The distributions are marginally skewed and assume normal distribution.
  - The data are continuous and not collapsed into a small number of categories.

Due to the dataset's non-parametric nature and the violation of parametric assumptions, the Chi-squared Statistic feature selection method was chosen. An alternative method, the Fischer Score, was not selected due to the risk of producing a sub-optimal subset, as features are independently selected in relation to the target variable. In this scenario, there are seventeen categorical or ordinal independent variables and three quantitative independent variables. The dependent variable, representing credit risk, is also nominal. To meet the conditions of the non-parametric test, the variables *age*, *duration*, and *amount* will be categorised into levels.

Assumptions of the Chi-square Statistics Test

To use the Chi-square Statistics test, there are a few assumptions which the data should follow.

1. The categories of the variables are mutually exclusive. This means the categories cannot overlap.
2. The data must be categorical and any continuous variables must be discretised into levels or categories.
3. The expected frequency of any of the variables should be greater than 5.
4. The data is independently sampled.

Before any processing for modelling is undertaken on the dataset, it is imperative to split the data into training and test subsets. This step is crucial to ensure that the models do not have prior knowledge of the test dataset during prediction, thereby maximising model performance and reducing overfitting. Utilising a splitting ratio defines the percentage of observations allocated to training and test subsets. The 80:20 splitting ratio, where 80% of observations are used for training and 20% for testing, is a commonly employed and standard ratio in practice (Joseph, 2022). While ratios such as 70:30 and 60:40 are also widely used, there is no fixed rule or formula for selecting a ratio, and it typically depends on the dataset and modeller preferences. In this dataset, the 80:20 splitting ratio will be adopted, meaning a random sample of 200 observations will be used for testing, while 800 observations will be allocated for training across all models.

As mentioned earlier, the variables *age*, *duration*, and *amount* will be categorised into levels to suit the non-parametric test conditions. The three variables were extracted into individual dataframes and categorised into bins. The resulting bins or categories, along with their corresponding encoding, are presented in Table 3.4.

Table 3.4: Categorised data for the continuous variables in the credit dataset.

Variable	Code	Category
amount (DM)	0	0 - 1000
	1	1001 - 2000
	2	2001 - 3000
	3	3001 - 4000
	4	4001 - 5000
	5	5001 - 6000
	6	6001 - 7000
	7	$\geq$ 7000
duration (months)	0	0 - 6
	1	7 - 12
	2	13 - 18
	3	19 - 24
	4	25 - 30
	5	31 - 36
	6	$\geq$ 37
age (years)	0	0 - 25
	1	26 - 35
	2	36 - 45
	3	46 - 55
	4	$\geq$ 56

For feature selection, the Python package *sklearn.feature_selection* was employed, alongside the *chi2* scoring function and the *SelectKBest* function. The *SelectKBest* function utilises the *chi2* scoring function and a parameter k, allowing the modeller to specify the number of best features to select. In this case, the value of k was set to 20 to include all features in the credit dataset. The Chi-square score indicates the significance of each feature, with higher scores implying greater significance. However, to select the best features, a threshold must be set on this score. Therefore, hypothesis testing was conducted. A null hypothesis and an alternative hypothesis were formulated, and a p-value test was applied with a threshold of 5%. If the p-value from the Chi-square test exceeded 0,05, the alternative hypothesis was rejected. Conversely, if the p-value was less than 0,05, the null hypothesis could be rejected.

1. Null hypothesis: the feature has no association with the target variable.
2. Alternative hypothesis: the feature is associated with the target variable and is significant.

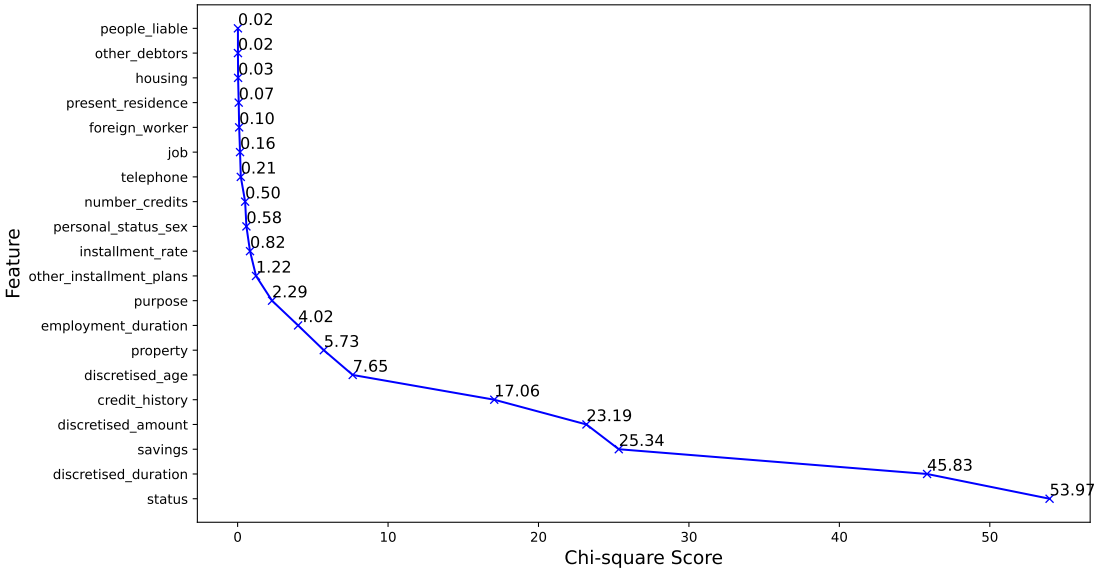

Figure 3.5: Variation in Chi-square values across all features in the credit dataset.

Table 3.5: Chi-square score and p-values for the various features in the credit dataset.

Variable	**Chi-square Score**	***p*-value**
status	53,97	$< 0{,}0001$
discretised_duration	45,83	$< 0{,}0001$
savings	25,33	$< 0{,}0001$
discretised_amount	23,19	$< 0{,}0001$
credit_history	17,06	$< 0{,}0001$
discretised_age	7,65	$< 0{,}0001$
property	5,73	0,02
employment_duration	4,02	0,05
purpose	2,29	0,13
other_installment_plans	1,22	0,27
installment_rate	0,82	0,36
personal_status_sex	0,58	0,44
number_credits	0,50	0,48
telephone	0,21	0,65
job	0,16	0,69
foreign_worker	0,10	0,75
present_residence	0,07	0,79
housing	0,03	0,86
other_debtors	0,02	0,88
people_liable	0,02	0,88

Referring to Figure 3.5, it is evident that numerous variables exhibit low Chi-square scores, corroborating the earlier discussion on descriptive statistics where many features showed similar distributions across all categories. Among the features, *status* and *duration* emerge as the most significant based on the Chi-square test. Conversely, the variables *people_liable* and *other_debtors* are identified as the least significant features according to the Chi-square test.

Table 3.5 presents the p-values obtained from the Chi-square test conducted using Python. These p-values are arranged in descending order based on the Chi-square score. According to the hypothesis testing criteria established, the features listed below, ranked from highest to lowest significance, are considered significant. This determination is based on both the high Chi-square test scores and, more importantly, the p-values being less than 0,05:

1. *status*
2. *discretised_duration*
3. *savings*
4. *discretised_amount*
5. *credit_history*
6. *discretised_age*
7. *property*
8. *employment_duration*

These features will be used in the final dataset for modelling.

3.4 Data Balancing

3.4.1 Synthetic Minority Over-sampling Technique - Nominal

To prevent any bias towards a particular classification within the class variable, it is necessary to balance the data. In this context, the initial dataset comprised 300 instances labelled as bad risk and 700 instances labelled as good risk. Following a split based on an 80:20 ratio, the training dataset encompassed a total of 800 instances, with 564 instances classified as good risk and 236 instances classified as bad risk. The distribution is shown in Figure 3.6.

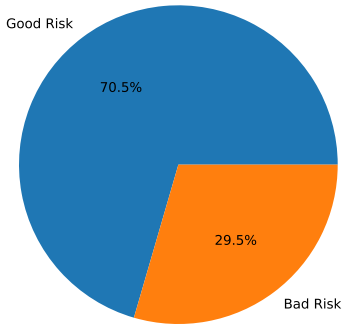

Figure 3.6: Distribution between good and bad risk on the training dataset before balancing.

As all variables were discretised, the entire feature dataset was categorical. Consequently, the Synthetic Minority Over-sampling Technique – Nominal was employed. This technique involves oversampling the minority class (in this case, the bad risk class) to match the size of the majority class, using synthetic data points generated based on the k-nearest neighbours. The rationale behind using oversampling was to ensure that crucial characteristics within the dataset are retained, especially given the dataset's small size. Downsampling the majority target class could risk losing significant information essential for modelling. The *SMOTEN* API from the imbalanced-learn package in Python was utilised to balance the training dataset, with the number of nearest neighbors set to 5 for computing the synthetic samples.

Various imbalance levels were computed as the secondary objective to determine the effect of imbalance on the models to follow. Since the minority class contains 236 observations which is approximately 41% of the majority class (564 observations), balancing in percentage of the majority class was taken from 50% all the way to 100% which is fully balanced. This was to ensure that the minority class is not under-sampled. These percentages are visually represented in Figure 3.7.

Figure 3.7: Distribution between good and bad risk on the training dataset after balancing for various balancing percentages.

A summary of the counts of the minority class observations and total observations are provided in Table 3.6.

Table 3.6: Summary of the various imbalance levels and observation counts.

Imbalance Level	**Percentage of Majority Class**	**Count of Minority Class**	**Total Observations**
1	50%	282	846
2	60%	338	902
3	70%	394	958
4	80%	451	1015
5	90%	507	1071
6	100%	564	1128

3.5 Parameter Estimation, Modelling and Prediction

3.5.1 Traditional Learning: Logistic Regression

This section details the mathematical approach employed in constructing the Logistic Regression model. These techniques serve as the conventional statistical methods for comparison with machine learning models.

Maximum Likelihood Estimation with Gradient Descent Optimisation

The initial stage of modelling involves obtaining the regression coefficients through Maximum Likelihood Estimation coupled with a gradient descent optimisation algorithm. Referring back to Equation 2.12, which represents the log likelihood function that needs to be maximised to acquire the regression coefficients, a new cost function is formulated to be minimised. This is achieved by transforming Equation 2.12 into a negative log likelihood function:

$$J = -\sum_{i=1}^{N} y_i \ln\left(P(n)\right) + (1 - y_i)\ln(1 - P(n)), \tag{3.1}$$

where J denotes the cost function. After obtaining the cost function, its derivative with respect to the parameters (also known as weights) can be computed and equated to zero to determine the parameters. However, solving this function analytically might not be feasible, hence optimisation techniques like gradient descent are employed. The gradient descent method starts by initialising the parameters with initial guesses, followed by updating the parameter values (weights) by calculating the slope or derivative of the cost function (Haji and Abdulazeez, 2021). This process is repeated until the gradient approaches zero. Mathematically, the method can be expressed as follows (all formulas utilised for the optimisation technique are adapted and cited from Cheon et al. (2018)):

$$\omega := \omega - \Delta\omega, \tag{3.2}$$

where ω is the weight and $\Delta\omega$ is the learning rate multiplied by the derivative of the cost function with respect to the weights; also known as the gradient. The weights are also known as the regression coefficients which need to be determined and Equation 3.2 can be written as:

$$\beta_{updated} := \beta - \alpha\frac{\partial J(\beta)}{\partial \beta}, \tag{3.3}$$

where β is the regression coefficient vector, α is the learning rate, $J(\beta)$ is the cost function in terms of the regression coefficient vector and $\beta_{updated}$ is the regression coefficient vector at which the cost function is optimised. The term $\frac{\partial J(\beta)}{\partial \beta}$ can be further expanded as follows:

$$\frac{\partial J(\beta)}{\partial \beta} = \frac{1}{m}\vec{X}^T \cdot (f(\vec{X} \cdot \beta) - \vec{Y}), \tag{3.4}$$

where m is the number of observations in the training feature dataset, $\vec{X}$ is the vector of the training dataset, $f(\vec{X} \cdot \beta)$ is the dot product of the training feature set and the regression coefficient vector, and $\vec{Y}$ is the target vector corresponding to the training feature dataset.

Equation 3.4 was implemented in a Python function using the training feature dataset and training target dataset until the model converged for all levels of data imbalance, indicating the lowest possible value for the cost function. To obtain the coefficients for the Logistic Regression model, 100 iterations with a learning rate of 1 were performed. The initial values for β were set to random values using the Python *numpy.random* package. The optimisation process for the cost function of the Logistic Regression model for a 100% balanced dataset, which serves as the baseline model, is depicted in Figure 3.8.

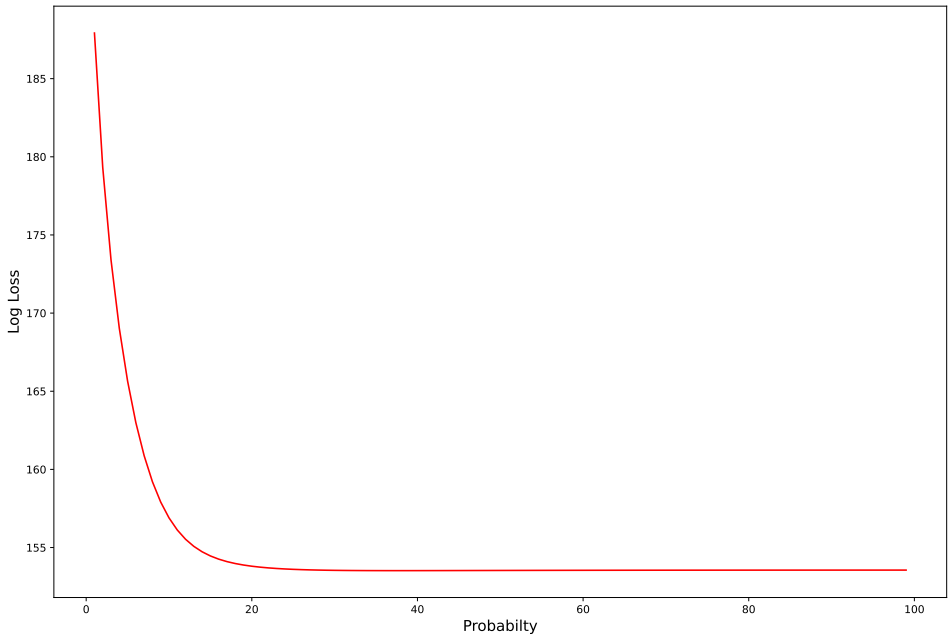

Figure 3.8: Cost function minimisation for the Logistic Regression model for a 100% balance level.

Referring to Figure 3.8, the Logistic Regression baseline model was optimised after approximately 20 iterations.

The coefficients after optimising for Logistic Regression for the different balanced models are provided in Table 3.7.

Table 3.7: Regression coefficients for the various balanced models.

Coefficient	**100%**	**90%**	**80%**	**70%**	**60%**	**50%**
β_0	0,02	0,16	0,31	0,47	0,67	0,88
β_1	0,68	0,69	0,66	0,65	0,67	0,67
β_2	−0,36	−0,39	−0,39	−0,39	−0,41	−0,40
β_3	0,25	0,24	0,23	0,26	0,27	0,22
β_4	0,06	0,05	0,06	0,06	0,05	0,03
β_5	0,49	0,50	0,49	0,46	0,42	0,38
β_6	0,36	0,33	0,31	0,26	0,22	0,20
β_7	−0,25	−0,21	−0,24	−0,25	−0,26	−0,25
β_8	0,17	0.20	0,19	0,17	0,16	0,15

For a Logistic Regression model, the mathematical equation can be written in general as follows:

$$Y = \frac{1}{1 + e^{-f(\beta)}}, \tag{3.5}$$

where Y is the probability that an observation has the outcome of 1.

In terms of the 8 independent variables, Equation 3.5 can be expanded to the following:

$$Y = \frac{1}{1 + e^{-(\beta_0 + X_1\beta_1 + X_2\beta_2 + X_3\beta_3 + X_4\beta_4 + X_5\beta_5 + X_6\beta_6 + X_7\beta_7 + X_8\beta_8)}} \tag{3.6}$$

Then, the equation for the Logistic Regression model baseline model (100% balanced), is given as follows:

$$Y = \left[1 + e^{-(0{,}017 + 0{,}677X_1 - 0{,}359X_2 + 0{,}245X_3 + 0{,}056X_4 + 0{,}493X_5 + 0{,}356X_6 - 0{,}247X_7 + 0{,}175X_8)}\right]^{-1}.$$

The variables $X_1 \ldots X_8$ are associated with the selected independent variables summarised in Table 3.8, and β_0 is the intercept or bias of the model.

Table 3.8: Variable to feature mapping for the Logistic Regression model.

Variable	$X_i(i \in [1,8])$
status	X_1
discretised_duration	X_2
savings	X_3
discretised_amount	X_4
credit_history	X_5
discretised_age	X_6
property	X_7
employment_duration	X_8

After obtaining the coefficients, the $\beta_{updated}$ values were multiplied by the test feature dataset, followed by the application of a sigmoid function. This step yielded the prediction probability necessary to validate the model. However, to make actual predictions, a threshold probability needed to be set. This threshold probability determines whether the predicted outcome is 1 (indicating good risk) or 0 (indicating bad risk). To determine the optimal threshold probability, the log loss of the model was computed. This process involved iterating through 300 threshold values and computing the log loss using the *sklearn.metrics* package in Python. The results were plotted on a graph to identify the threshold value corresponding to the lowest loss, as depicted in Figure 3.9 for the baseline model. This procedure was repeated for all levels of data imbalance, and a summary is presented in Table 3.9. The log loss serves as an indicator of how closely the prediction probability aligns with the actual outcome, with lower log loss values indicating better prediction accuracy.

Table 3.9: Log loss and probability threshold for the various balanced models.

Balanced Model	**Log Loss**	**Threshold Probability**
50%	7,93	0,48
60%	7,57	0,49
70%	7,57	0,44
80%	7,57	0,42
90%	7,93	0,32
100%	7,93	0,30

Referring to Figure 3.9, the threshold probability for the Logistic Regression model baseline model is 0,3 for a log loss of 7,93.

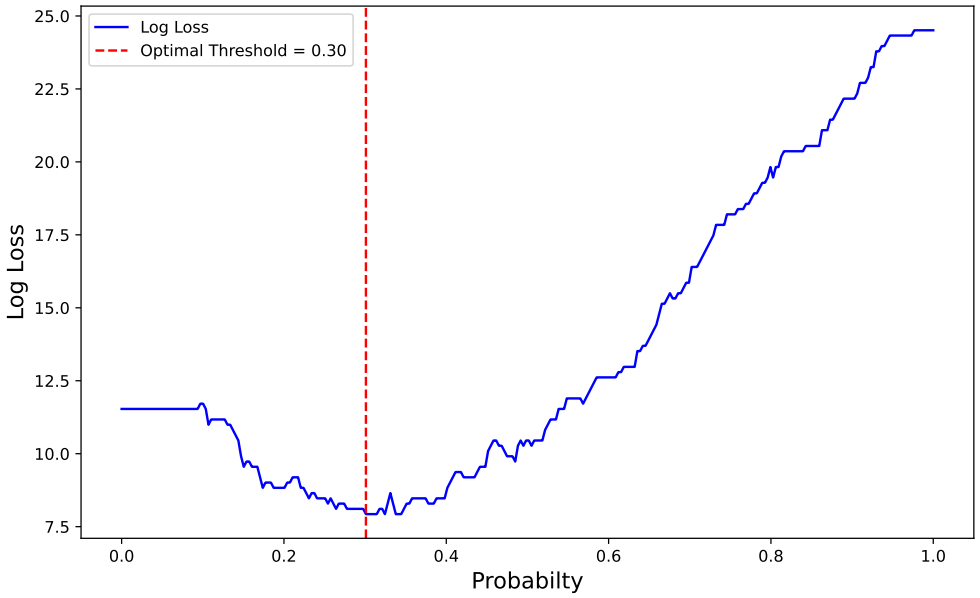

Figure 3.9: Log loss for various threshold probabilities for the Logistic Regression model with 100% balance on the minority class.

3.5.2 Machine Learning Models

k-Nearest Neighbours

For the k-NN model, the *KNeighborsClassifier* class from the Python *sklearn.neighbors* package was employed. However, this classifier requires a specified integer value for the number of neighbors to consider when classifying the data. To determine the optimal number of neighbors (k), a range of values from 1 to 15 was tested, and the accuracy score was computed for each iteration. The results were plotted in Figure 3.10 for the baseline model (100% balance). The graphs illustrate that the accuracy for the training dataset tended to decrease as the number of nearest neighbors increased.

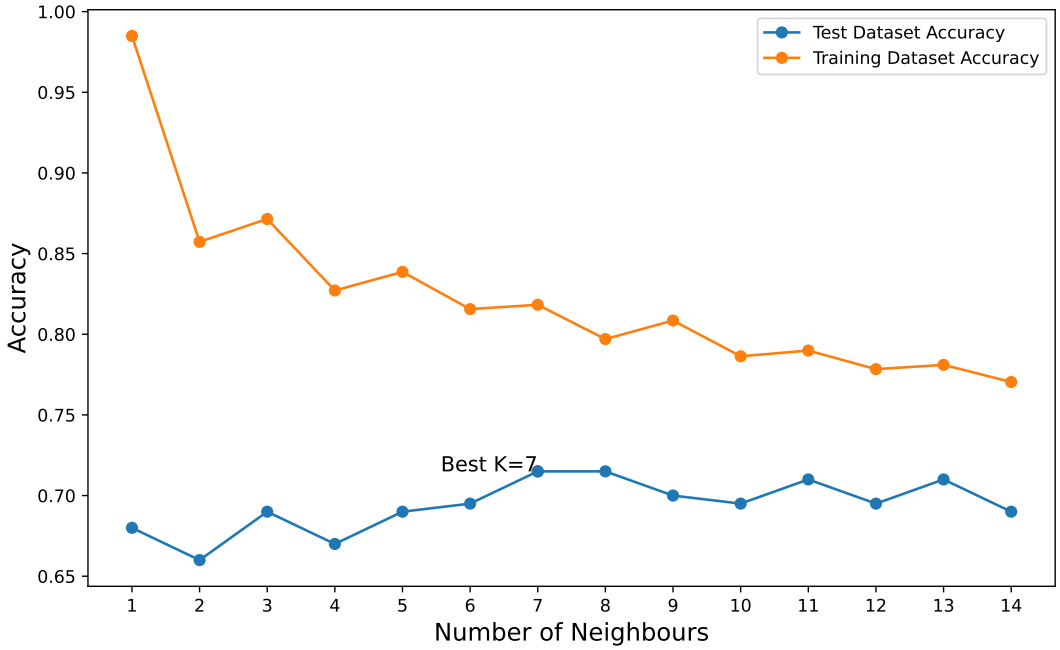

Figure 3.10: Train and testing accuracy of different neighbour values used in the k-NN baseline model.

A summary of the best k for all the balanced k-NN models are provided in Table 3.10.

Table 3.10: Best value of k for the various balanced k-NN models with the corresponding test accuracy.

Balanced Model	**Best k**	**Test Accuracy**
50%	3	0,76
60%	3	0,73
70%	3	0,73
80%	11	0,73
90%	9	0,72
100%	7	0,72

Based on the results depicted in Figure 3.10, the highest accuracy was achieved when using 7 nearest neighbors for classification for the baseline model using the testing dataset. This value was selected for both training and testing the model to strike a balance between maximising test accuracy and maintaining reasonable training accuracy. It is important to note that this optimisation process was performed using a baseline k-NN model without any hyperparameter tuning.

Support Vector Machines

For the Support Vector Machines (SVM) model, Python's *sklearn.model_selection* and *sklearn.svm* packages were utilised, employing the *GridSearchCV* and *SVC* classes for modelling. In SVM classification, three key parameters are typically adjusted: *C*, which denotes the regularisation parameter inversely related to the strength of regularisation; *kernel*, specifying the kernel type chosen from options like 'linear', 'poly', 'rbf', 'sigmoid', 'precompted', or a predefined callable function; and *gamma*, representing the kernel coefficient for 'rbf', 'poly', or 'sigmoid' kernels.

To find the optimal combination of parameters for the SVM model, the *GridSearchCV* class was employed. This class systematically explores a grid of parameter combinations to determine the best set that maximises the accuracy of the model. Since the 'rbf' kernel is commonly used as a baseline for SVM, it was selected as the default kernel. Initial values for the parameters were set, and the *GridSearchCV* class was utilised to perform parameter optimisation. This process typically takes longer compared to other machine learning models due to the exhaustive search over the parameter grid. The initial parameter values were set as follows:

- *C*: [0.01,0.1,1, 10, 100,1000];
- *kernel*: 'rbf';
- *gamma*:[1,0.1,0.01,0.001,0.0001,2,3,10].

A sample of the optimisation process is shown in Figure 3.11.

```
Running model for resampled_data[1]
Fitting 5 folds for each of 72 candidates, totalling 360 fits
[CV 1/5] END .......C=50, gamma=0.1, kernel=rbf;, score=0.696 total time=   0.0s
[CV 2/5] END .......C=50, gamma=0.1, kernel=rbf;, score=0.702 total time=   0.0s
[CV 3/5] END .......C=50, gamma=0.1, kernel=rbf;, score=0.667 total time=   0.0s
[CV 4/5] END .......C=50, gamma=0.1, kernel=rbf;, score=0.783 total time=   0.0s
[CV 5/5] END .......C=50, gamma=0.1, kernel=rbf;, score=0.811 total time=   0.0s
[CV 1/5] END .........C=50, gamma=1, kernel=rbf;, score=0.680 total time=   0.0s
[CV 2/5] END .........C=50, gamma=1, kernel=rbf;, score=0.685 total time=   0.0s
[CV 3/5] END .........C=50, gamma=1, kernel=rbf;, score=0.678 total time=   0.0s
[CV 4/5] END .........C=50, gamma=1, kernel=rbf;, score=0.761 total time=   0.0s
[CV 5/5] END .........C=50, gamma=1, kernel=rbf;, score=0.872 total time=   0.0s
[CV 1/5] END ......C=50, gamma=0.01, kernel=rbf;, score=0.757 total time=   0.0s
[CV 2/5] END ......C=50, gamma=0.01, kernel=rbf;, score=0.696 total time=   0.0s
[CV 3/5] END ......C=50, gamma=0.01, kernel=rbf;, score=0.767 total time=   0.0s
[CV 4/5] END ......C=50, gamma=0.01, kernel=rbf;, score=0.728 total time=   0.0s
[CV 5/5] END ......C=50, gamma=0.01, kernel=rbf;, score=0.806 total time=   0.0s
[CV 1/5] END ....C=50, gamma=0.0001, kernel=rbf;, score=0.735 total time=   0.0s
[CV 2/5] END ....C=50, gamma=0.0001, kernel=rbf;, score=0.702 total time=   0.0s
[CV 3/5] END ....C=50, gamma=0.0001, kernel=rbf;, score=0.778 total time=   0.0s
[CV 4/5] END ....C=50, gamma=0.0001, kernel=rbf;, score=0.717 total time=   0.0s
```

Figure 3.11: A sample of the parameter optimisation for the SVM baseline model.

A summary of the best parameters for the various balanced SVM models are provided in Table 3.11.

Table 3.11: Best parameters for the various balanced SVM models.

Balanced Model	**C**	**gamma**
50%	2	0,1
60%	3	0,1
70%	3	0,1
80%	3	0,1
90%	3	0,1
100%	50	1

Following the optimisation of the SVM baseline model, the parameter set yielding the highest overall accuracy was identified as: 'C': 50, 'gamma': 1, 'kernel': 'rbf'. It is noteworthy that these parameter values diverge considerably from those of the models with imbalance levels below 100%.

Decision Tree

For the Decision Tree model, the Python *sklearn.tree* package was employed, utilising the *DecisionTreeClassifier* class. The Gini Index impurity measure was selected as it yielded the best overall accuracy. Additionally, a maximum depth of 4 was imposed on the tree. This maximum depth constraint ensures that the tree does not become overly complex, thus mitigating the risk of overfitting and maintaining model interpretability. Notably, no hyperparameter optimisation was performed. The structure of the resulting tree is illustrated in Figure 3.12, depicting 4 levels below the highest node. At each node, the Gini impurity was calculated, and the corresponding class (0 or 1) was determined.

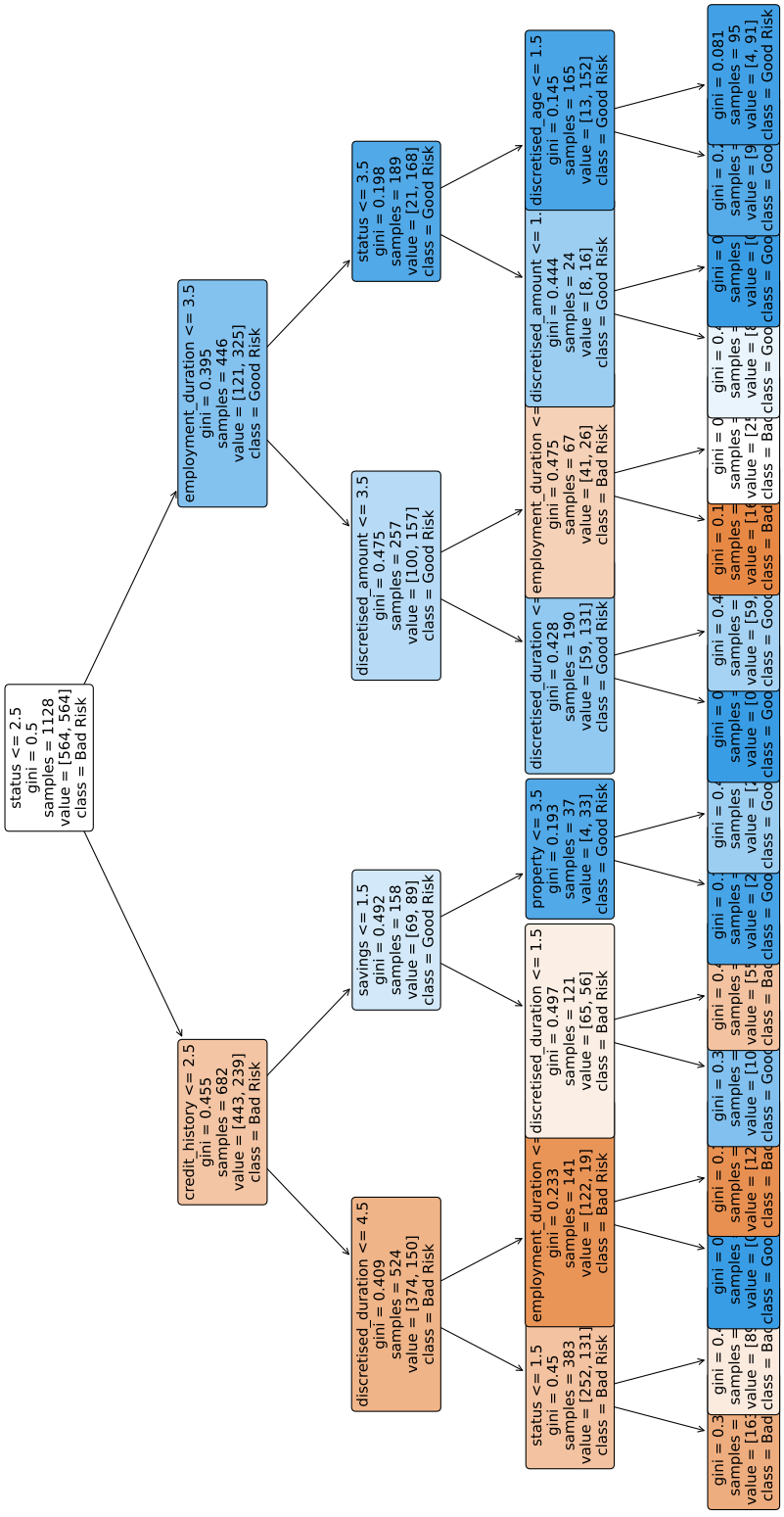

Figure 3.12: Decision Tree model structure for a balance level of 100%.

3.5.3 Neural Network Models

Multi-layer Perceptron

The MLP model was built using the *Keras* package in Python, consisting of three fully connected layers which are also known as *dense* layers. The first two dense layers have 64 and 32 neurons, respectively. The layers use the ReLU activation function. To prevent over-fitting, L2 regularisation was used with a penalty of 0,001. The *Dropout* parameter of 0,5 was also used after each dense layer which aids in over-fitting. Finally, the last dense layer contains 1 neuron using sigmoid activation, which is suitable for binary classification in this case. The architecture for this model is shown in Figure 3.13.

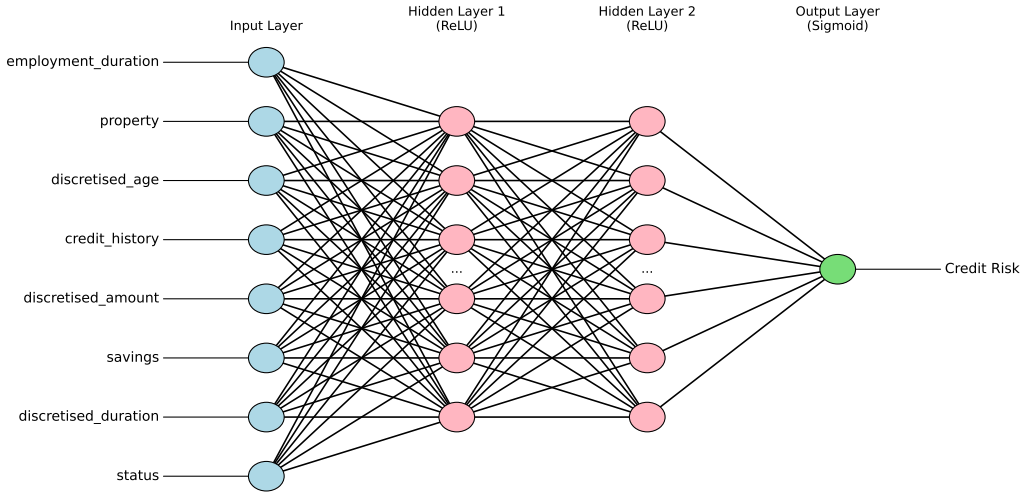

Figure 3.13: MLP model architecture.

The model used the *Adam* and binary cross-entropy function, both frequently employed in binary classification tasks. Validation was used in training with two main parameters through callbacks. The first was *EarlyStopping* which is responsible for stopping the training when the validation loss hasn't improved for the specified number of epochs to prevent over-fitting. In this case, 5 epochs were used. The second callback was *ReduceLROnPlateau* which reduces the learning rate when the validation loss has not improved for the epochs (5), aiding in model convergence. These parameters were set as the same for all the differently balanced models.

The model is trained for up to 100 epochs and 20% of the training data was used for validation. However, the model stops training at 20 epochs. The accuracy and loss was plotted for the baseline model to visualise the training process. These plots are provided in Figure 3.14 and Figure 3.15. For the accuracy plot, the goal is to observe an upward

trend in both the validation and training lines as epochs progress. This indicates that the model performs effectively not only on the training dataset but also on unseen or validation data.

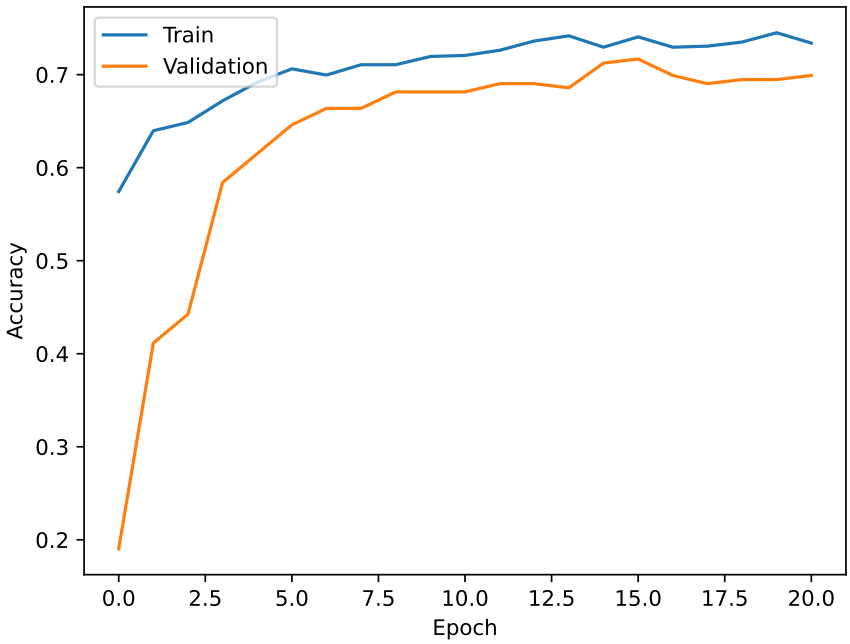

Figure 3.14: MLP model accuracy for the baseline model.

The loss plot in Figure 3.15 illustrates the loss metric, which quantifies how closely the model's predictions align with the actual outcomes. Lower values indicate less error. The aim for this plot is to see the validation and training lines decreasing with the epochs, indicating that the model is learning and improving. Additionally, a converging trend between the training an validation lines is desirable which suggests that the model is not over-fitting. Over-fitting is indicated by an decreasing training loss while the validation loss begins to increase or ceases to decrease.

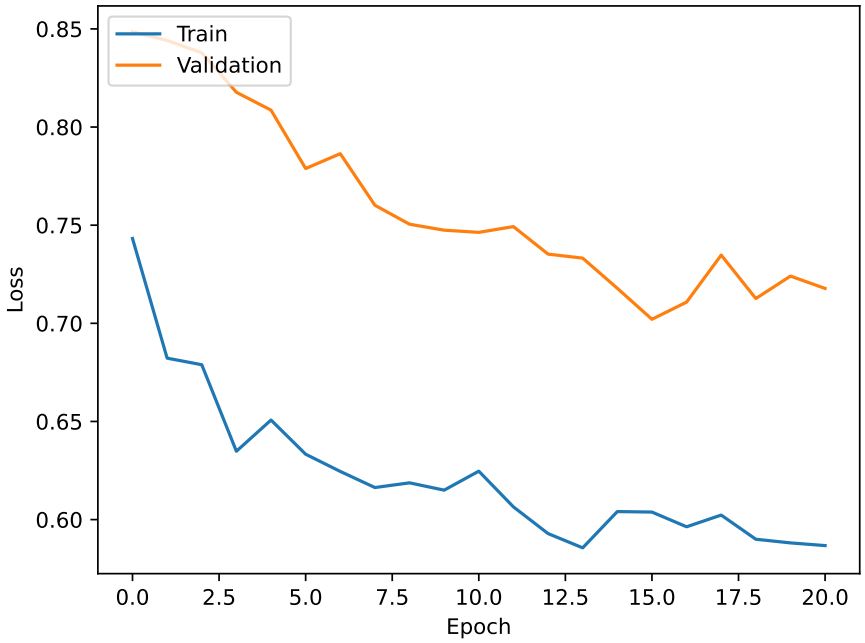

Figure 3.15: MLP model loss for the baseline model.

RBFNN

The RBFNN model contains only three layers. The second layer, called the RBF layer, calculates the distance of the input data from points called centres and applies an exponential function based on these distances. There are two main parameters in the RBF layer which are *units* and *gamma*. Units refer to the number of centres the RBF layer will use. The centres can be thought of as reference points against which the distances of the input data are measured. Gamma is a scaling factor for distances. A higher gamma value indicates increased sensitivity of the RBF layer to small changes in distance. The output layer consists of a dense layer with a sigmoid activation function.

Since there is no pre-defined model in Python for the RBFNN, a custom RBFNN model was created. Before training the model, a k-means clustering was used to find the initial centres. This step is carried out to establish an initial reference point for the RBF layer, which is determined by classifying the input data. This is then provided to the model as a starting point for training.

The model architecture is provided in Figure 3.16.

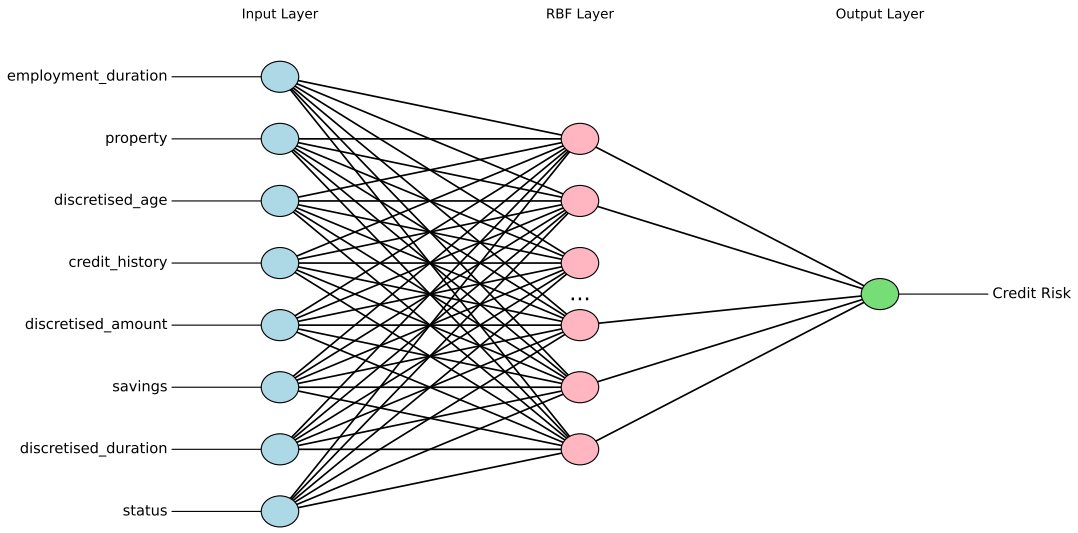

Figure 3.16: RBFNN model architecture.

Hyperparameter Tuning

In this procedure, the objective is to identify the optimal or most effective combination of center points and gamma values. Cross-validation was employed to ascertain these combinations. Cross-validation entails partitioning the training data into multiple subsets, with the model being trained on some of these subsets and tested on the remaining ones. This process is iterated several times, typically 5 times in this instance, to ensure dependable results. The combination offering the highest average accuracy across all splits is selected as the best choice. The highest average accuracy was 72% giving the best number of centres of 15 with a gamma of 0,1.

Chapter 4

Results and Discussion

This chapter will discuss the assessment of various models and balancing levels following predictions made using the test dataset. The test dataset comprised 136 instances of good risk and 64 instances of bad risk. The baseline model is defined as having fully balanced training data, with a 100% balance. It is important to note that the evaluation metrics employ a weighted average approach for precision, recall, and F1 score. When comparing models against each other on an imbalanced dataset, the weighted average of these metrics is typically preferred. This method offers a more realistic assessment of the model's performance by considering the imbalanced nature of the dataset.

4.1 Evaluation Metrics

Table 4.1: Evaluation metrics for the different balanced Logistic Regression models.

Balanced Model	Precision	Recall	F1 Score	Jaccard Index	Accuracy	AUC	Log Loss
50%	0,77	0,78	0,77	0,73	0,78	0,71	7,93
60%	0,79	0,79	0,79	0,74	0,79	0,75	7,57
70%	0,79	0,79	0,79	0,74	0,79	0,75	7,57
80%	0,78	0,79	0,78	0,74	0,79	0,73	7,75
90%	0,77	0,78	0,77	0,73	0,78	0,72	7,93
100%	0,77	0,78	0,77	0,73	0,78	0,72	7,93

Referring to Table 4.1, both precision and recall appear to be relatively stable across the various levels of dataset balancing, ranging from 0,77 to 0,79. This suggests that the Logistic Regression model's ability to correctly predict positive cases (precision) and its ability to find all positive observations (recall) is not highly sensitive to the degree of imbalance in the dataset. The F1 scores also remain fairly stable with slight variations, inhibiting a similar pattern as precision and recall. The highest F1 scores are seen at 60% and 70% balance, indicating that these levels may offer a good trade-off between precision and recall. The Jaccard Index scores, which indicate a measure of

the similarity between the predicted and actual positive observations, are also stable across the different levels of balance, with a slight decrease as the dataset becomes fully balanced. This may suggest that as the balance increases, the model does not necessarily improve in distinguishing the positive class. The accuracy improves slightly from 50% to 60% and remains constant up to 80% before decreasing slightly. This indicates that for the Logistic Regression model, moderate balance may contribute to the overall accuracy without a clear benefit from a fully balanced model. The AUC score, indicative of the model's capability to differentiate between positive and negative classes, is the highest at 60% and 70% balance. This suggests that these levels of balance improve the model's discrimination capability, while balancing fully to 100% does not improve and may slightly diminish capability. The log loss, which measures the error of the probability estimates by the model, is the lowest at 60% and 70% balance, indicating that these models are more confident in their probability estimates. The increase in log loss at higher levels of balance suggests that the model's confidence in its probability estimates decreases as the dataset becomes fully balanced.

Table 4.2: Evaluation metrics for the different balanced k-NN models.

Balanced Model	**Precision**	**Recall**	**F1 Score**	**Jaccard Index**	**Accuracy**	**AUC**
50%	0,75	0,76	0,75	0,70	0,76	0,72
60%	0,74	0,73	0,74	0,67	0,74	0,71
70%	0,74	0,72	0,73	0,65	0,73	0,71
80%	0,75	0,73	0,74	0,65	0,73	0,72
90%	0,75	0,72	0,73	0,63	0,72	0,72
100%	0,76	0,71	0,72	0,62	0,72	0,73

For the k-NN model from Table 4.2, the precision slightly decreases from 50% to full balance, suggesting that a perfectly balanced dataset does not improve the model's ability to predict positive instances correctly. The recall also declines as the model balance increases, indicating that the model becomes less capable of identifying all relevant instances in a fully balanced dataset. The F1 score follows the trend of the precision and recall values, diminishing with increased balance, implying that the balanced dataset does not enhance the balance between precision and recall. The Jaccard Index decreases with more balanced data, suggesting overlap between the model's predictions and the actual positive instances in a fully balanced dataset. The accuracy slightly reduces as the dataset balance increases, with the highest accuracy at 50% balance. This may suggest that the k-NN model is suited to slightly imbalanced datasets for this type of scenario. The AUC also remains fairly stable across the different balance levels, with a marginal increase at the fully balanced model. This stability implies that the model's capability in discriminating between good and bad risk is not heavily influenced by dataset imbalance.

Table 4.3: Evaluation metrics for the different balanced Decision Tree models.

Balanced Model	Precision	Recall	F1 Score	Jaccard Index	Accuracy	AUC
50%	0,71	0,73	0,71	0,69	0,73	0,64
60%	0,71	0,73	0,71	0,69	0,73	0,64
70%	0,75	0,69	0,70	0,59	0,69	0,71
80%	0,71	0,70	0,70	0,63	0,70	0,67
90%	0,77	0,71	0,72	0,61	0,71	0,74
100%	0,78	0,69	0,70	0,57	0,69	0,73

Referring to Table 4.3, the precision on the Decision Tree model remains stable as the dataset is balanced from 50% to 60%, then increases at 70%, and continues to improve as the dataset becomes fully balanced. This indicates that having a balanced dataset enhances the model's accuracy in predicting the positive class. The recall starts higher at 50% and 60% at 0,73, then declines at 70%, thereafter remaining consistent from 80% to 100% balance at 0,69. The initial drop may indicate that the model's sensitivity to identifying all actual positives is affected when the dataset is moderately balanced. The F1 score is stable from 50% to 60%, drops at 70% balance, then slightly improves as the dataset becomes more balanced. This metric indicates that the balance between precision and recall does not uniformly improve with more balanced data. The accuracy and Jaccard Index seem to decrease with increased balance, while AUC improves, indicating a complex relationship between dataset balance and model performance. These results highlight that for Decision Tree models, a fully balanced dataset does not guarantee superior performance across all metrics, and that the optimal balance level might depend on which performance metric is prioritised for the scenario.

Table 4.4: Evaluation metrics for the different balanced SVM models.

Balanced Model	Precision	Recall	F1 Score	Jaccard Index	Accuracy	AUC
50%	0,77	0,77	0,77	0,70	0,77	0,73
60%	0,75	0,74	0,74	0,67	0,74	0,72
70%	0,75	0,74	0,74	0,67	0,74	0,72
80%	0,76	0,74	0,75	0,67	0,75	0,73
90%	0,76	0,74	0,75	0,66	0,74	0,73
100%	0,69	0,71	0,67	0,68	0,71	0,59

Referring to Table 4.4, the performance of SVM models on the dataset demonstrates a trend where certain metrics peak at intermediate levels of dataset balancing rather than at full balance. Initially, with 50% balance, precision, recall, and F1 score are at their highest (0,77), indicating a strong balance between the accuracy of positive predictions and the model's ability to detect all positives. As the dataset becomes more balanced from 60% to 90%, these metrics slightly decline or remain stable (precision and F1 score at 0.75 and recall at 0,74 for 80% and 90% balance), suggesting a slight impact of increased balance on the model's performance. However, at full balance (100%), a

notable drop in precision (0,69) and F1 score (0,67) is observed, along with a moderate decrease in recall (0,71), which could indicate that the model's ability to classify may be compromised when the data is perfectly balanced. Interestingly, while the accuracy remains fairly consistent, peaking at 0,77 for 50% balance and declining to 0,71 at full balance, the AUC drastically falls to 0,59 at 100% balance. This sharp decline in AUC implies a significant deterioration in the model's discriminative capacity when the dataset is fully balanced, potentially due to the SVM model's sensitivity to the overlap of data points near the decision boundary in a fully balanced state. Overall, while some balancing may be beneficial, overbalancing can adversely affect SVM model's predictive power in this credit risk dataset.

Table 4.5: Evaluation metrics for the different balanced MLP models.

Balanced Model	**Precision**	**Recall**	**F1 Score**	**Jaccard Index**	**Accuracy**	**AUC**
50%	0,73	0,74	0,73	0,69	0,74	0,68
60%	0,73	0,74	0,72	0,70	0,74	0,65
70%	0,73	0,74	0,72	0,70	0,74	0,65
80%	0,71	0,73	0,71	0,69	0,73	0,64
90%	0,78	0,78	0,78	0,73	0,78	0,73
100%	0,80	0,80	0,80	0,74	0,80	0,78

For the MLP model in Table 4.5, measures such as recall, precision, and F1 score demonstrate consistent performance at lower levels of balance (50% to 80%), with values around 0,73 for precision and F1 score, and a slight drop in recall to 0,73 at 80% balance. These metrics improve significantly at higher levels of balance, peaking at 0,80 for both precision and recall at full balance (100%), indicating an enhanced ability of the model to correctly identify and predict positive instances when the dataset is fully balanced. The Jaccard Index and accuracy follow a similar trend, showing a marked improvement at 90% and 100% balance, with scores of 0,74 and 0,80, respectively. The AUC increases notably from 0,68 at 50% balance to 0,78 at full balance, confirming that the MLP model's overall performance on predicting credit risk improves as the data approaches full balance. These results suggest that, unlike the previous models, the MLP benefits more distinctly from a balanced dataset, achieving its best performance across all metrics at 100% balance, which could be due to the MLP's ability to learn more complex patterns when the class distribution is even.

Table 4.6: Evaluation metrics for the different balanced RBFNN models.

Balanced Model	**Precision**	**Recall**	**F1 Score**	**Jaccard Index**	**Accuracy**	**AUC**
50%	0,62	0,68	0,57	0,68	0,68	0,51
60%	0,62	0,68	0,57	0,68	0,68	0,51
70%	0,65	0,69	0,59	0,68	0,69	0,52
80%	0,66	0,69	0,60	0,68	0,69	0,53
90%	0,67	0,69	0,62	0,68	0,70	0,54
100%	0,72	0,73	0,69	0,70	0,73	0,61

Referring to Table 4.6, the evaluation metrics for RBFNN models on the dataset demonstrate a clear trend of improvement across all metrics as the level of balance in the training data increases. Precision starts at a lower value of 0,62 for both 50% and 60% balance, then gradually increases, reaching its peak at 0,72 with a fully balanced dataset. Similarly, recall increases marginally from 0,68 to 0,73 as the dataset balance increases from 50% to 100%. The F1 Score, which integrates precision and recall, follows this upward trend, starting at 0,57 and growing to 0,69, indicating an enhanced balance between precision and recall at higher levels of dataset balance. The Jaccard Index remains constant at 0,68 until 80% balance and then slightly improves to 0,70 at full balance. Accuracy improves consistently from 0,68 to 0,73 as the dataset balance increases, suggesting that the model's overall predictive performance benefits from a more balanced dataset. The AUC shows a more pronounced increase from 0,51 to 0,61, confirming a significant enhancement in the model's discriminative capacity with a fully balanced dataset. These trends indicate that the RBFNN model significantly benefits from a balanced dataset, likely due to the enhanced generalisation ability provided by a more uniform distribution of classes, which helps the RBFNN model to better capture the underlying patterns in the data.

4.2 Confusion Matrices for the Models Under Study

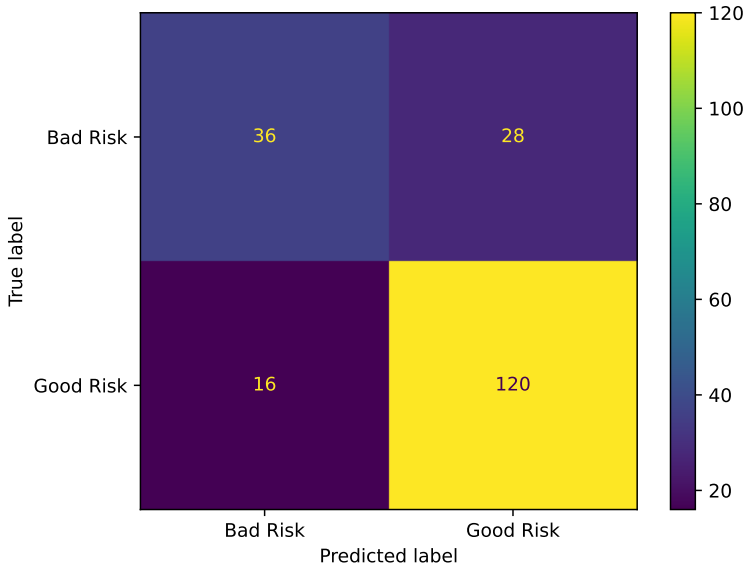

Figure 4.1: Confusion Matrix for the Logistic Regression baseline model.

The confusion matrix in Figure 4.1 for the Logistic Regression model on a fully balanced credit risk dataset shows that out of the total predictions, 36 instances of 'Bad Risk' were correctly identified (true negatives), and 120 instances of 'Good Risk' were correctly

predicted (true positives). However, there were 28 instances where 'Bad Risk' was incorrectly predicted as 'Good Risk' (false positives), and 16 instances where 'Good Risk' was incorrectly labelled as 'Bad Risk' (false negatives). This gives the model a precision of 0,81 (120/(120+28)) for predicting 'Good Risk' and a recall of 0,88 (120/(120+16)) for the same (note that these are not weighted averages). The ability to accurately predict 'Good Risk' is crucial for approving credit and minimising default risk, while correctly identifying 'Bad Risk' helps in reducing the chances of extending credit to customers who are likely to default. The relatively small count of false positives and negatives suggests that the model is reliable, but the presence of false predictions also points to the inherent risk and uncertainty in credit risk assessment. Hence, while the model performs well, there is still a tangible risk of misclassification that must be accounted for in decision-making processes.

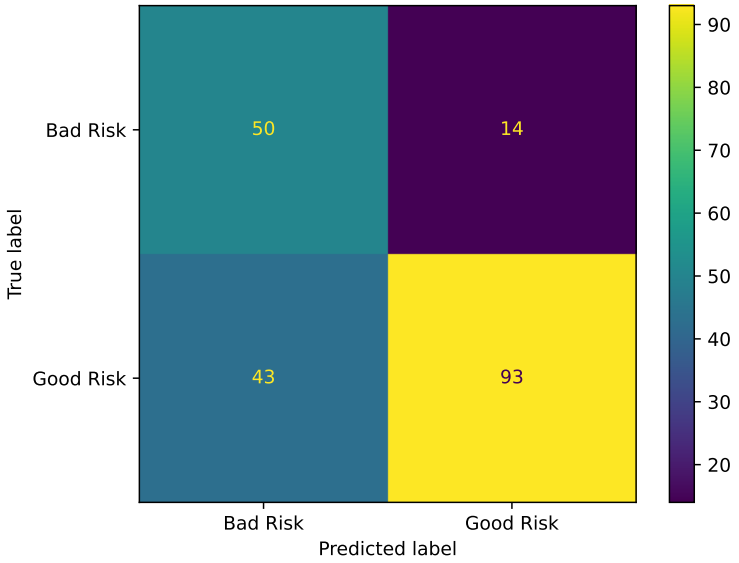

Figure 4.2: Confusion Matrix for the k-NN baseline model.

The confusion matrix for the k-NN model in on a balanced credit risk dataset in Figure 4.2 shows that the model correctly identified 50 'Bad Risk' cases and 93 'Good Risk' cases, which are the true negatives and true positives, respectively. However, it also misclassified 14 'Bad Risk' cases as 'Good Risk' (false positives) and 43 'Good Risk' cases as 'Bad Risk' (false negatives). The model has a precision of 0,87 (93/(93+14)) for 'Good Risk' and a recall of 0,68 (93/(93+43)) for the same. In the context of credit risk prediction in the banking and finance industry, the ability of the k-NN model to accurately classify 'Bad Risk' is vital for minimising the risk of default by identifying potential non-payers. The high number of false negatives (43), where 'Good Risk' is classified as 'Bad Risk', could result in a loss of business by denying credit to potentially good customers. Conversely, the relatively smaller count of false positives (14) suggests the model is more conservative in predicting 'Good Risk', which is prudent for risk

management but could impact customer acquisition and the bank's profitability.

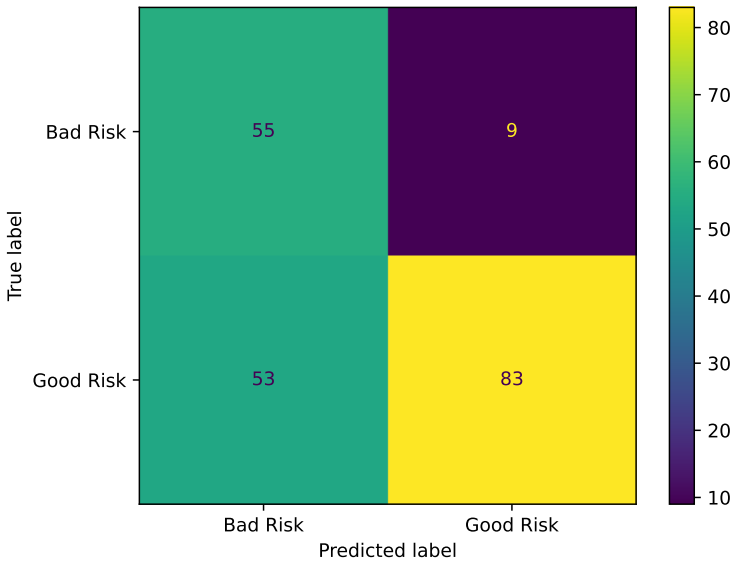

Figure 4.3: Confusion Matrix for the Decision Tree baseline model.

The confusion matrix for the Decision Tree model in Figure 4.3 reveals that the model correctly predicted 55 instances of 'Bad Risk' and 83 instances of 'Good Risk', representing true negatives and true positives, respectively. It incorrectly classified 9 instances of 'Bad Risk' as 'Good Risk' (false positives) and 53 instances of 'Good Risk' as 'Bad Risk' (false negatives). This gives a precision of 0,90 (83/(83+9)) for 'Good Risk' predictions, but the recall is lower at 0,61 (83/(83+53)) for identifying 'Good Risk'. In the banking and finance industry, these results are significant; the high precision indicates that when the model predicts an applicant as a 'Good Risk', it is very likely to be correct, which is crucial for trust in credit approvals. However, the lower recall and substantial number of false negatives mean that the model is conservative, potentially leading to the rejection of creditworthy applicants, which could result in missed opportunities for revenue.

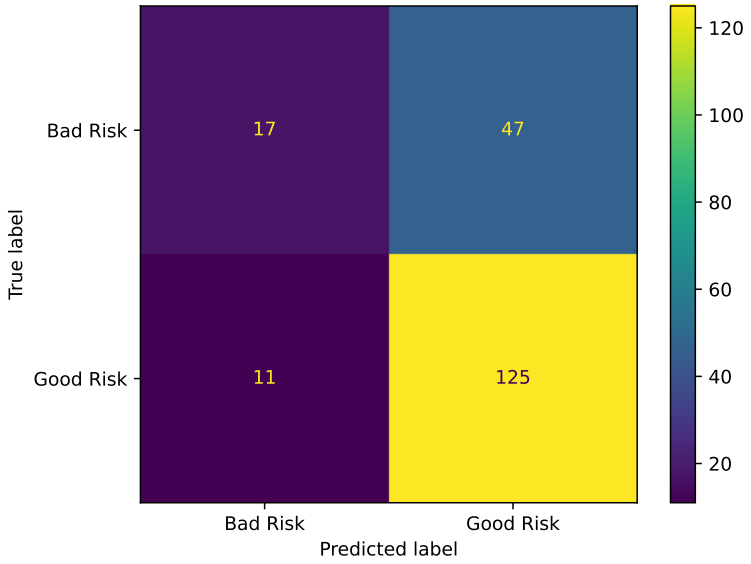

Figure 4.4: Confusion Matrix for the SVM baseline model.

For the Support Vector Machine (SVM) model, Figure 4.4 indicates the model has a high ability to identify 'Good Risk' applicants, with 125 true positives and a low false negative rate of 11. Conversely, the model is less proficient at identifying 'Bad Risk' applicants, with 47 false positives and only 17 true negatives. This results in a high precision rate for 'Good Risk' ($125/(125+47) = 0{,}73$) but a lower recall ($125/(125+11) = 0{,}92$), suggesting the model is conservative when predicting 'Good Risk'. In the context of credit risk prediction within the banking and finance industry, the high number of false positives for 'Bad Risk' is concerning as it indicates a propensity to classify potentially defaulting applicants as creditworthy, which could lead to higher default rates. The high recall for 'Good Risk' implies that the institution would likely approve most creditworthy applicants. However, the precision imbalance between 'Good Risk' and 'Bad Risk' necessitates careful risk management strategies to mitigate potential financial losses if this model is used.

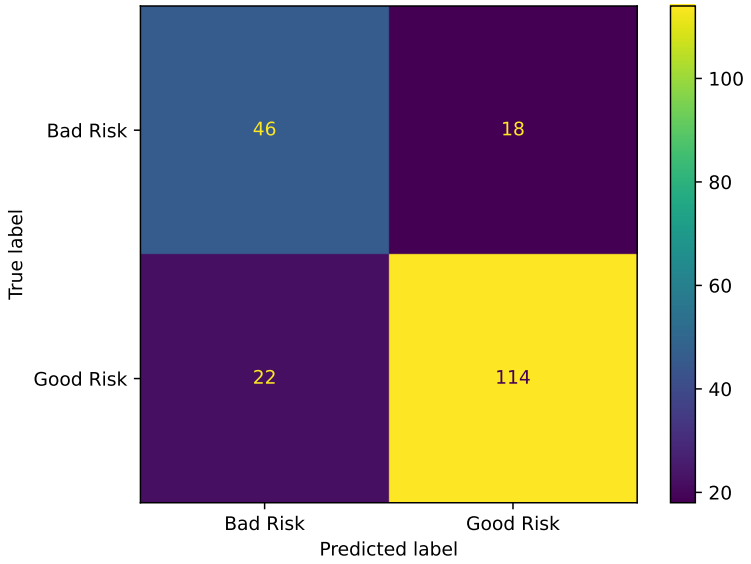

Figure 4.5: Confusion Matrix for the MLP baseline model.

The confusion matrix for the MLP model in Figure 4.5 indicates that the model has correctly predicted 46 'Bad Risk' and 114 'Good Risk' applicants, which are true negatives and true positives, respectively. However, there are 18 false positives and 22 false negatives. This results in a precision of 0,86 (114/(114+18)) for 'Good Risk' predictions, indicating a high likelihood that individuals labelled as 'Good Risk' by the model are creditworthy. The model has a recall of 0,84 (114/(114+22)) for 'Good Risk', showing it is quite proficient at identifying most of the actual 'Good Risk' individuals. A high number of false positives could lead to granting credit to those likely to default, while false negatives could mean denying loans to potential good customers, leading to lost revenue opportunities. Hence, this MLP model's balance between precision and recall suggests it could be a valuable tool for credit risk evaluation, minimising potential defaults while not excessively turning away good customers.

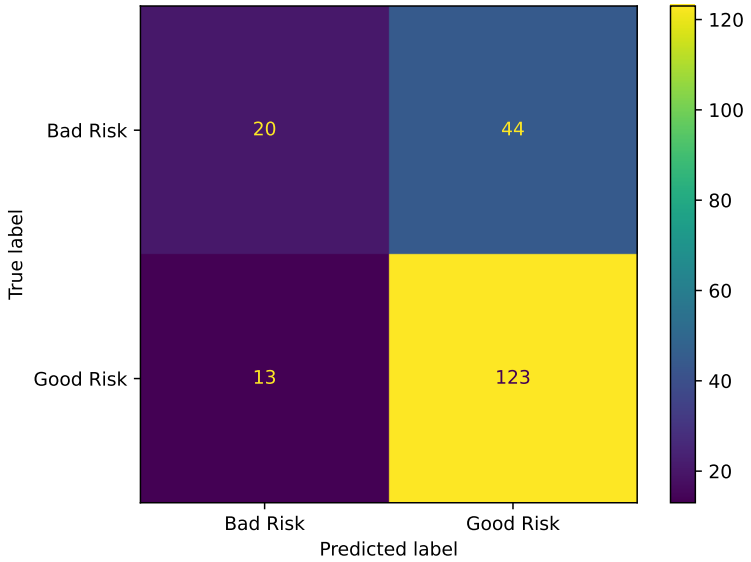

Figure 4.6: Confusion Matrix for the RBFNN baseline model.

The confusion matrix for the RBFNN model in Figure 4.6 indicates only 20 true negatives, and 123 true positives. The model produced 44 false positives and 13 false negatives. This results in a precision for 'Good Risk' of 0,74 (123/(123+44)) and a recall for 'Good Risk' of 0,90 (123/(123+13)). This suggests that while the model is fairly accurate, there may still be a considerable portion of 'Bad Risk' applicants being incorrectly classified as 'Good Risk'. The recall for 'Good Risk' is high at approximately 0,90, indicating the model's strength in identifying the majority of 'Good Risk' applicants.

4.3 AUC-ROC Curve

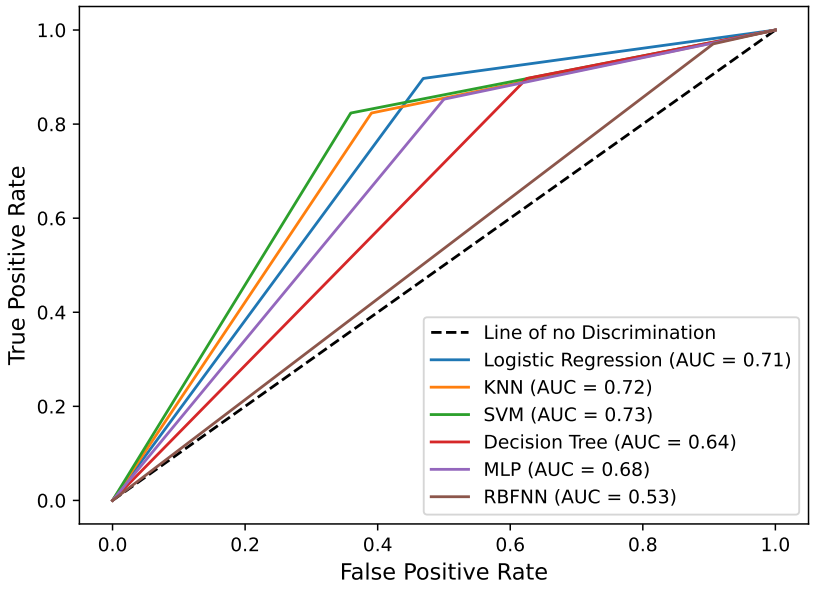

Figure 4.7: AUC-ROC curve for the various models at a 50% balance level.

The AUC-ROC curves in Figure 4.7 showcase how the various models perform on a credit risk dataset that is balanced at 50%, where the true positive rate (sensitivity) is plotted against the false positive rate (1− specificity) at different threshold values. Recall that the AUC quantifies the overall ability of the model to discriminate between the positive ('Good Risk') and negative ('Bad Risk') classes.

The Logistic Regression model (AUC = 0.71) and the SVM model (AUC = 0.73) exhibit curves positioned nearer to the top-left corner. This suggests that they strike a better balance between sensitivity and specificity, implying a more robust capability to distinguish between the two classes. The k-NN model's curve (AUC = 0,72) is slightly below these two, suggesting a slightly lower but comparable discriminative performance.

In contrast, the Decision Tree model's curve (AUC = 0,64) and the MLP model's curve (AUC = 0,68) are further from the top-left corner, which implies a weaker performance in distinguishing between good and bad risks. The RBFNN model, with an AUC of 0.53, shows only slight improvement compared to random chance. This is evident from its curve closely tracking the diagonal line of no discrimination (AUC = 0.5), which signifies a model incapable of distinguishing between the two classes effectively.

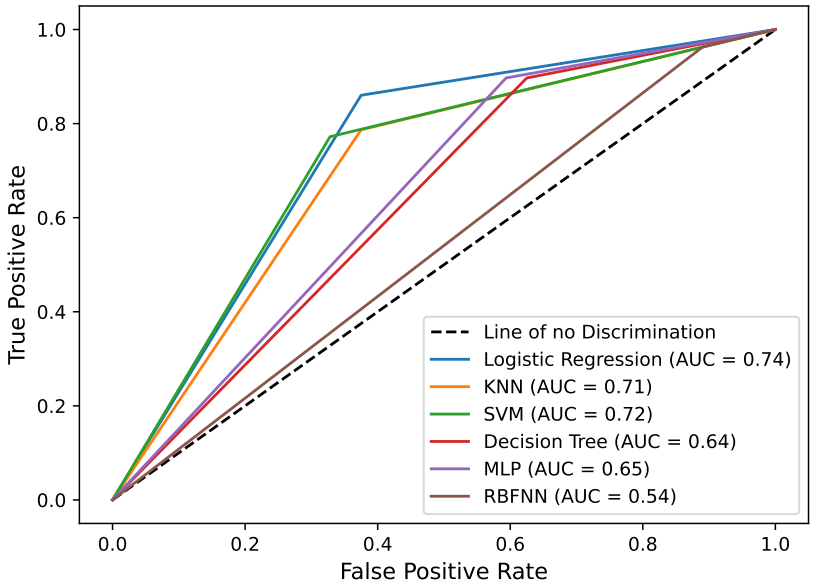

Figure 4.8: AUC-ROC curve for the various models at a 60% balance level.

The AUC-ROC curves for the various models at a 60% balance level is provided in Figure 4.8. The Logistic Regression model achieves the highest AUC of 0,74, indicating a strong ability to discriminate between the two classes. This is followed closely by the SVM model with an AUC of 0,72 and the k-NN model with an AUC of 0,71, both showing good discriminative performance.

The Decision Tree and MLP models have lower AUC values of 0,64 and 0,65, respectively, suggesting that while they are better than random guessing (which would result in an AUC of 0,5), they are less effective at correctly classifying risks compared to Logistic Regression, SVM, and k-NN models. The RBFNN model has the lowest AUC of 0,54, indicating that its performance is not much better than random chance and it struggles to distinguish between the risk classes effectively. This suggests that the RBFNN model is highly impacted by class imbalance.

The Logistic Regression model, with its curve positioned highest towards the top-left, suggests it would be the most effective model in this set for credit risk assessment.

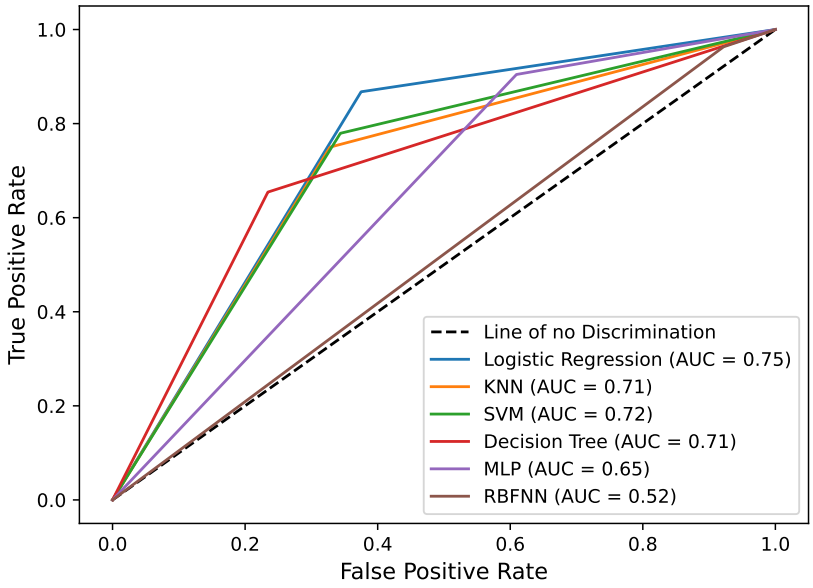

Figure 4.9: AUC-ROC curve for the various models at a 70% balance level.

The AUC-ROC curves for different models trained on a 70% balanced credit risk dataset are provided in Figure 4.9. The Logistic Regression model outperforms others with an AUC of 0,75, indicating a strong discriminative ability between the two classes. The SVM model follows closely with an AUC of 0,72, and both k-NN and Decision Tree models have an AUC of 0,71, which are respectable but slightly lower than Logistic Regression.

The MLP model has a lower AUC of 0,65, showing less discriminative power than the previously mentioned models. Lastly, the RBFNN model has the lowest AUC of 0,52, suggesting its performance is only slightly better than random guessing, as depicted by its proximity to the diagonal line of no discrimination (AUC = 0,5).

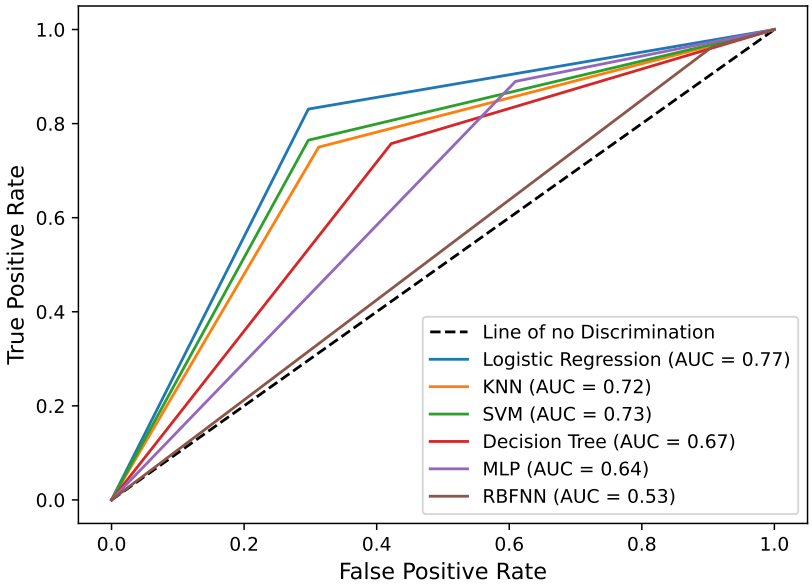

Figure 4.10: AUC-ROC curve for the various models at an 80% balance level.

The AUC-ROC curves at an 80% balanced credit risk dataset are given in Figure 4.10. The Logistic Regression model shows the highest AUC of 0,77, which suggests it has the best predictive capability among the models. The SVM model follows with an AUC of 0,73, indicating a strong ability to differentiate between the classes. The k-NN model has an AUC of 0,72, comparable to SVM, while the Decision Tree model shows a moderate AUC of 0,67.

The MLP model, with an AUC of 0,64, and the RBFNN model, with an AUC of 0,53, lag behind the other models. The low AUC for the RBFNN model indicates a performance near random chance and suggests that this model might be the least suitable for credit risk prediction at this balance level. The position of each curve in relation to the line of no discrimination (diagonal dotted line) and the axes, defined by the true positive rate along the y-axis and the false positive rate along the x-axis, confirms the Logistic Regression model as the most reliable for predicting credit risk, while the RBFNN model is likely to be the least reliable, with the other models offering varied degrees of effectiveness in between.

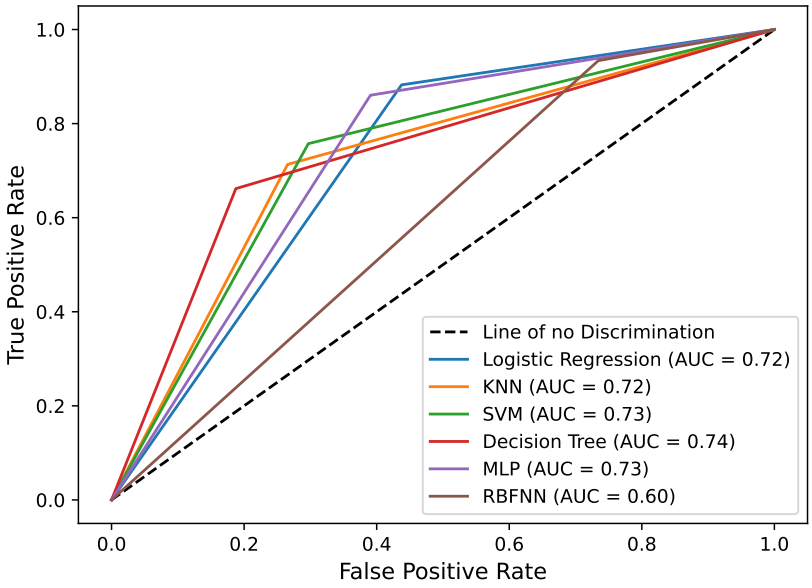

Figure 4.11: AUC-ROC curve for the various models at a 90% balance level.

The AUC-ROC curve for various models at a 90% balance level are illustrated in Figure 4.11. The Decision Tree model exhibits the highest AUC at 0,74, indicating a strong ability to differentiate between 'Good Risk' and 'Bad Risk.' This is closely followed by the SVM and MLP models, both with an AUC of 0,73, and the Logistic Regression and k-NN models with an AUC of 0,72, suggesting these models have a similar capability in distinguishing the two classes at this level of balance. The RBFNN model has a lower AUC of 0,60, implying it is less effective in discrimination. It is worthwhile to note that the MLP model starts to increase in disciminatory power at this balance level.

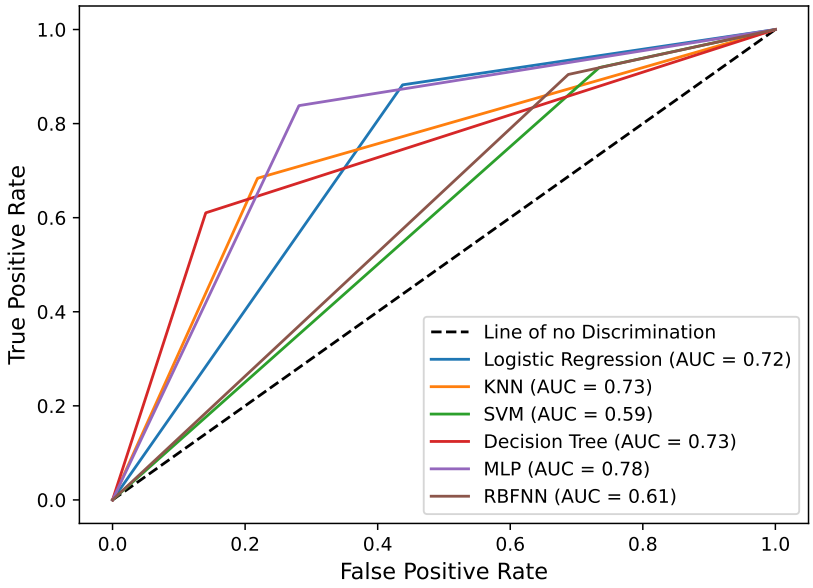

Figure 4.12: AUC-ROC curve for the various models at a 100% balance level.

The AUC-ROC curve for different models on the fully balanced (100%) training dataset is given in Figure 4.12. The MLP model achieves the highest AUC of 0,78, indicating a strong capability to differentiate between 'Good Risk' and 'Bad Risk'. The curve lies further to the top-left and indicates that the model performs the best after being trained on a fully balanced dataset. The Decision Tree and k-NN models are also strong performers with an AUC of 0,73, demonstrating good classification effectiveness. Logistic Regression follows with an AUC of 0,72, suggesting slightly less discriminative power than the Decision Tree and k-NN but still a robust model.

The SVM model, however, has a notably lower AUC of 0,59, indicating a poorer performance in distinguishing between the risk categories. The RBFNN model has an AUC of 0,61, which is better than the SVM model but still indicates limited discriminative ability. Throughout all balance levels, the RBFNN model performed the worst.

4.4 Comparative Analysis with Existing Literature

This section briefly examines how the findings of this study align with or diverge from established literature in the area of credit risk prediction.

Chen et al. (2023) propose a selective learning framework that combines transparent Logistic Regression with a NN to balance accuracy and interpretability in credit risk assessment. The NN utilised in the study is a shallow model comprising only one hidden layer with a limited number of neurons. Training is conducted using the backpropagation optimisation algorithm. The empirical results show that the NN outperforms

Logistic Regression in terms of accuracy and recall for credit risk assessment, particularly in capturing relevant default cases. The dataset used in the study contains 30 000 observations and is imbalanced, with a lower proportion of default cases. The simplicity of Logistic Regression in the current research is particularly beneficial for clear, direct insights, although it may not capture the more subtle, complex patterns present in larger datasets. The larger scale and diversity of dataset used by the authors likely contribute to the observed deviation in performance between the Logistic Regression model in this research and their approach. The extensive dataset could potentially encompass a broader range of credit risk scenarios, thereby enabling the NN to effectively learn and predict more complex patterns that may not be as prevalent or detectable in a smaller dataset like the one used in this research.

Alonso and Carbo (2021) utilise a large and imbalanced dataset from a major Spanish bank for credit default prediction. To address the imbalance, the authors apply SMOTE, resulting in a dataset with a 25% default rate. They compare the performance of traditional Logistic Regression with more advanced machine learning models such as Lasso penalised Logistic Regression, Random Forest, CART, XGBoost, and deep NNs. The results indicate that the advanced machine learning models, particularly XGBoost and Random Forest, outperform Logistic Regression in terms of classification and calibration for credit default prediction. While Logistic Regression and Lasso models exhibit lower false positive rates, the superior true positive rates of the advanced machine learning models offset these differences. The deep NN model did not outperform the Logistic Regression model by a significant amount with an AUC of 0,79 for Logistic Regression and 0,80 for the deep NN. This aligns with the results on a fully balanced dataset in this research, however, for the imbalanced datasets, the MLP and RBFNN models underperform. This again suggests that the dataset may be too small for the NN models to capture complexities.

Chen and Ma (2022) focuses on enhancing an SVM model for predicting credit market risk. They use the SMOTE technique for dataset balancing and optimised the SVM's parameters with the Fruit Fly Optimization Algorithm (FOA). The model demonstrates superior performance, particularly in stability and generalisation, compared to other models enhanced with FOA-SMOTE. The SVM model used by the authors and the SVM model in this research differ primarily in optimisation techniques and dataset performance. The model in this research employs GridSearchCV for optimising parameters like 'C', 'kernel', and 'gamma', and defaulted to the 'rbf' kernel. This led to good performance at higher balance levels but a dip in fully balanced datasets. These methodological differences, including kernel choice and parameter optimisation strategies, could account for the variations in performance between the two models.

Chapter 5

Conclusions and Recommendations

This chapter concludes the findings of the study and provides recommendations for future research.

5.1 Conclusions

Credit risk assessment is a crucial aspect in commercial banking and finance, where institutions face substantial financial risks while evaluating the creditworthiness of customers seeking credit. Various factors like employment status, credit history, income, and duration of employment play a significant role in determining the likelihood of a customer defaulting on credit repayments. In a high-volume operational context, analysing these factors is not only expensive but also time-intensive. Quantitative learning models enable these institutions to perform efficient, cost-effective, and accurate credit assessments using a range of statistical and machine learning algorithms. Despite their effectiveness, certain machine learning models pose challenges due to their computational intensity and complexity. On the other hand, statistical learning models, though less resource-intensive, can potentially offer comparable performance. This research undertook a thorough analytical comparison of six distinct models: the statistical model of Logistic Regression, machine learning models including k-NN, Decision Trees, and SVM, alongside NN models such as MLP and RBFNN. The aim was to explore whether statistical learning models can match or even surpass the performance of machine learning models in credit risk assessment using a real-world dataset while also considering data imbalance.

The comparative analysis of various predictive models on a credit risk dataset at different levels of class balance reveals that no single model consistently outperforms others across all balance levels. However, certain trends can be observed which are highlighted below.

The Logistic Regression model shows strong and stable performance across different balance levels, with AUC values reflecting good discriminative power. It demonstrates robustness and reliability as a baseline model for credit risk assessment.

The k-NN model performs well, particularly at moderate balance levels, but does not exhibit the highest discriminative ability at any specific balance level. Its performance is competitive, yet it is overshadowed by other models depending on the balance of the dataset.

The SVM model generally shows good performance, especially at higher balance levels. However, it does exhibit a dip in performance at the fully balanced level, suggesting that SVM may require careful parameter tuning or may not be as suitable for fully balanced datasets in credit risk prediction.

The Decision Tree model, while simpler and more interpretable than other models, shows variability in its performance, with its highest AUC at a 90% balance level. This indicates that while Decision Trees can be effective, their performance may be sensitive to the balance of the dataset.

The MLP model stands out with the highest AUC at a fully balanced level, suggesting that neural network-based approaches can be very effective for credit risk assessment, particularly when the class distribution is equal. However, its performance at other balance levels suggests that MLP models may require a balanced dataset to perform optimally. This model achieved the highest accuracy of 80% on a fully balanced dataset.

The RBFNN model generally under-performs compared to other models, with lower AUC values. This suggests that RBFNN may not be the best choice for credit risk prediction tasks, or it may require more sophisticated data pre-processing and feature engineering to improve its performance. Additionally, this model is intricate and demands greater computational resources compared to the other models.

In the banking and finance industry, where the consequences of misclassification can be significant, the choice of model can impact both the profitability and the risk profile of the institution. The analysis suggests that while MLP may provide the best performance in a fully balanced scenario, Logistic Regression and SVM are strong contenders across various levels of dataset balance, offering a more consistent performance for credit risk assessment.

Ultimately, the selection of a predictive model for credit risk assessment should be based on the specific requirements of the task, the available data, and the acceptable trade-off between different types of errors. It is also crucial to consider model interpretability, computational efficiency, and ease of integration into existing systems. In practice, a combination of models or an ensemble approach may be employed to leverage the strengths of different models and mitigate their weaknesses.

5.2 Recommendations

Future research in credit risk assessment using predictive models can extend in several directions in an attempt to enhance performance, interpretability, and integration into financial systems. Ensemble methods should be explored in an attempt to achieve more robust predictions by leveraging the strengths of multiple models. Enhanced feature engineering might uncover more complex data relationships, while advanced hyper-parameter optimisation techniques can fine-tune models for better accuracy. The exploration of alternative balancing techniques could offer insights into handling imbalanced datasets more effectively.

Given the complexity of models like NNs, applying explainable AI methods would be crucial for interpretability, especially in a heavily regulated industry like finance. Adjusting classification thresholds to account for the costs of misclassification in credit risk, and validating models against data from different domains and over time, may help ensure robustness and generalisability. Compliance with financial regulations should be a key consideration, ensuring models not only perform well but also align with industry standards. Considering the operational integration of these models will ensure that they contribute value to the existing credit risk management processes, balancing technological advancement with practical business applications.

References

H. Abdou, J. Pointon, and A. El-Masry. Neural nets versus conventional techniques in credit scoring in Egyptian banking. *Expert Systems with Applications*, 35(3):1275–1292, 2008.

A. Aggarwal, S. Kasiviswanathan, Z. Xu, O. Feyisetan, and N. Teissier. Label Inference Attacks from Log-loss Scores. In *International Conference on Machine Learning*, pages 120–129. PMLR, 2021.

C. C. Aggarwal. Data Classification: Algorithms and Applications. page 44. Springer, 2015.

N. Al-Rahman Al-Serw. K-nearest Neighbor: The maths behind it, how it works and an example, 2021. URL https://medium.com/analytics-vidhya/k-nearest-neighbor-the-maths-behind-it-how-it-works-and-an-example-f1de1208546c. [Online; accessed March 1, 2023].

T. M. Alam, K. Shaukat, I. A. Hameed, S. Luo, M. U. Sarwar, S. Shabbir, J. Li, and M. Khushi. An Investigation of Credit Card Default Prediction in the Imbalanced Datasets. *IEEE Access*, 8:201173–201198, 2020.

M. Ala'raj, M. F. Abbod, and M. Majdalawieh. Modelling customers credit card behaviour using bidirectional LSTM neural networks. *Journal of Big Data*, 8(1):1–27, 2021.

I. Aljarah, H. Faris, and S. Mirjalili. Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Computing*, 22:1–15, 2018.

A. Alonso and J. M. Carbo. Understanding the performance of machine learning models to predict credit default: a novel approach for supervisory evaluation. 2021.

F. N. Arain. Decision Tree Classification Algorithm, 2022. URL https://www.devops.ae/decision-tree-classification-algorithm/. [Online; accessed May 5, 2023].

M. Aranha and K. Bolar. Efficacies of artificial neural networks ushering improvement in the prediction of extant credit risk models. *Cogent Economics & Finance*, 11(1): 2210916, 2023.

Ashes Das. Oversampling to remove class imbalance using SMOTE, 2019. URL https://medium.com/@asheshdas.ds/oversampling-to-remove-class-imbalance-using-smote-94d5648e7d35. [Online; accessed July 16, 2023].

M. Awad and R. Khanna. Support Vector Machines for Classification. In *Efficient Learning Machines*, pages 39–66. Springer, 2015.

S. Balakrishnama and A. Ganapathiraju. Linear Discriminant Analysis – A Brief Tutorial. *Institute for Signal and information Processing*, 18(1998):1–8, 1998.

J. A. Bastos. Predicting Credit Scores with Boosted Decision Trees. *Forecasting*, 4: 925–935, 2022.

H. A. Bekhet and S. F. K. Eletter. Credit Risk Assessment Model for Jordanian Commercial Banks: Neural Scoring Approach. *Review of Development Finance*, 4(1): 20–28, 2014.

G. Bénédict, V. Koops, D. Odijk, and M. de Rijke. sigmoidF1: A Smooth F1 Score Surrogate Loss for Multilabel Classification. *arXiv preprint arXiv:2108.10566*, 2021.

S. Beniwal and J. Arora. Classification and Feature Selection Techniques in Data Mining. *International Journal of Engineering Research & Technology (IJERT)*, 1(6):1–6, 2012.

V. Bewick, L. Cheek, and J. Ball. Statistics Review 13: Receiver Operating Characteristic Curves. *Critical Care*, 8(6):1–5, 2004.

A. Bhattacharya, S. K. Biswas, and A. Mandal. Credit risk evaluation: a comprehensive study. *Multimedia Tools and Applications*, 82(12):18217–18267, 2023.

M. Bhoge. Using the artificial neural network for credit risk management. *Oracle Blogs*, 2019.

J. Biesiada and W. Duch. Feature Selection for High-Dimensional Data – A Pearson Redundancy Based Filter. In *Computer Recognition Systems 2*, pages 242–249. Springer, 2007.

A. Blanco, R. Pino-Mejías, J. Lara, and S. Rayo. Credit scoring models for the microfinance industry using neural networks: Evidence from Peru. *Expert Systems with Applications*, 40(1):356–364, 2013.

S. S. Chai, W. K. Wong, K. L. Goh, H. H. Wang, and Y. C. Wang. Radial Basis Function (RBF) Neural Network: Effect of Hidden Neuron Number, Training Data Size, and Input Variables on Rainfall Intensity Forecasting. *International Journal on Advanced Science Engineering Information Technology*, 9(6):1921–1926, 2019.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16: 321–357, 2002.

D. Chen, J. Ye, and W. Ye. Interpretable Selective Learning in Credit Risk. *Research in International Business and Finance*, 65:101940, 2023.

K. Chen, K. Zhu, Y. Meng, A. Yadav, and A. Khan. Mixed Credit Scoring Model of Logistic Regression and Evidence Weight in the Background of Big Data. In *International Conference on Intelligent Systems Design and Applications*, pages 435–443. Springer, 2018.

L. Chen and R. Ma. Market Risk Early Warning Based on Deep Learning and Fruit Fly Optimization. *Mathematical Problems in Engineering*, 2022, 2022.

X. Chen and J. C. Jeong. Enhanced Recursive Feature Elimination. In *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, pages 429–435. IEEE, 2007.

J. H. Cheon, D. Kim, Y. Kim, and Y. Song. Ensemble method for privacy-preserving logistic regression based on homomorphic encryption. *IEEE Access*, 6:46938–46948, 2018.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv:1412.3555*, 2014.

S. A. Czepiel. Maximum Likelihood Estimation of Logistic Regression Models: Theory and Implementation. *Available at czep. net/stat/mlelr. pdf*, 83, 2002.

R. Dey and F. M. Salem. Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.

S. Dreiseitl and L. Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics*, 35 (5-6):352–359, 2002.

T. Eelbode, J. Bertels, M. Berman, D. Vandermeulen, F. Maes, R. Bisschops, and M. B. Blaschko. Optimization for Medical Image Segmentation: Theory and Practice when evaluating with Dice Score or Jaccard Index. *IEEE Transactions on Medical Imaging*, 39(11):3679–3690, 2020.

A. T. Elhassan, M. Aljurf, F. Al-Mohanna, and M. Shoukri. Classification of Imbalance Data using Tomek Link (T-Link) Combined with Random Under-sampling (RUS) as a Data Reduction Method. *Global J Technol Optim S*, 1, 2016.

D. Fantazzini and S. Figini. Random survival forests models for sme credit risk measurement. *SSRN*, 2022.

E. Frank, L. Trigg, G. Holmes, and I. H. Witten. Technical Note: Naive Bayes for Regression. *Machine Learning*, 41:5–25, 2000.

R. Gandhi. Support Vector Machine — Introduction to Machine Learning Algorithms, 2022. URL https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. [Online; accessed May 5, 2023].

V. García and J. S. Sánchez. Improving risk predictions by preprocessing imbalanced credit data. In *Advances in Computational Intelligence*, pages 120–129. Springer, 2013.

U. Groemping. South German Credit Data: Correcting a Widely Used Data Set. *Rep. Math., Phys. Chem., Berlin, Germany, Tech. Rep*, 4:2019, 2019.

Z. Gu and L. Dong. Distance formulas capable of unifying euclidian space and probability space. *arXiv preprint arXiv:1801.01972*, 2018.

S. H. Haji and A. M. Abdulazeez. Comparison of optimization techniques based on gradient descent algorithm: A review. *PalArch's Journal of Archaeology of Egypt/Egyptology*, 18(4):2715–2743, 2021.

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

A. A. Heidari, H. Faris, I. Aljarah, and S. Mirjalili. An efficient hybrid multilayer perceptron neural network with grasshopper optimization. *Soft Computing*, 23:7941–7958, 2019.

W. E. Henley and D. J. Hand. A k-Nearest-Neighbour Classifier for Assessing Consumer Credit Risk. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 45 (1):77–95, 1996.

J. Hull and W. Suo. A methodology for assessing model risk and its application to the implied volatility function model. *Journal of Financial and Quantitative Analysis*, 37 (2):297–318, 2002.

S. Jadhav, H. He, and K. Jenkins. Information Gain Directed Genetic Algorithm Wrapper Feature selection for Credit Rating. *Applied Soft Computing*, 69:541–553, 2018.

V. R. Joseph. Optimal ratio for data splitting. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2022.

A. Jović, K. Brkić, and N. Bogunović. A review of feature selection methods with applications. In *38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1200–1205. IEEE, 2015.

V. Kanade. What Is Logistic Regression? Equation, Assumptions, Types, and Best Practices, 2022. URL https://www.spiceworks.com/tech/artificial-

intelligence/articles/what-is-logistic-regression/. [Online; accessed March 1, 2023].

R. Kohavi. *Special Issue on Applications of Machine Learning and The Knowledge Discovery Process*. Kluwer, 1998.

M. Korkmaz, S. Güney, and Ş. YİĞİTER. The importance of logistic regression implementations in the turkish livestock sector and logistic regression implementations/-fields. *Harran Tarım ve Gıda Bilimleri Dergisi*, 16(2):25–36, 2012.

A. Krichene. 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, 2017.

Y. LeCun, L. Bottou, G. B Orr, K.-R. Müller, et al. Neural networks: Tricks of the trade. *Springer Lecture Notes in Computer Sciences*, 1524(5-50):6, 1998.

Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen. Medical Image Classification with Convolutional Neural Network. In *2014 13th international conference on control automation robotics & vision (ICARCV)*, pages 844–848. IEEE, 2014.

S. Lin, W. Tianxiang, D. Weiping, X. Jiucheng, and L Yaojin. Feature selection using Fisher score and multilabel neighborhood rough sets for multilabel classification. *Information Sciences*, 578:887–912, 2021. ISSN 0020-0255.

H. Liu, J. Li, and L. Wong. A Comparative Study on Feature Selection and Classification Methods Using Gene Expression Profiles and Proteomic Patterns. *Genome Informatics*, 13:51–60, 2002.

X. Liu, R. Zhou, D. Qi, and Y. Xiong. A Novel Methodology for Credit Spread Prediction: Depth-Gated Recurrent Neural Network with Self-Attention Mechanism. *Mathematical Problems in Engineering*, 2022, 2022.

Y. Liu and L. Huang. Supply chain finance credit risk assessment using support vector machine–based ensemble improved with noise elimination. *International Journal of Distributed Sensor Networks*, 16(1), 2020.

I. Mačerinskienė, L. Ivaškevičiūtė, and G. Railienė. The Financial Crisis Impact on Credit Risk Management in Commercial Banks. *KSI Transactions on Knowledge Society*, 7(1), 2014.

A. Marcano-Cedeño, J. Quintanilla-Domínguez, M. G. Cortina-Januchs, and D. Andina. Feature Selection Using Sequential Forward Selection and Classification Applying Artificial Metaplasticity Neural Network. In *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*, pages 2845–2850. IEEE, 2010.

S. Mishra. Handling imbalanced data: Smote vs. random undersampling. *International Research Journal of Engineering and Technology*, 4(8):317–320, 2017.

R. Mohammed, J. Rawashdeh, and M. Abdullah. Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results. In *2020 11th International Conference on Information and Communication Systems (ICICS)*, pages 243–248. IEEE, 2020.

F. E. Moula, C. Guotai, and M. Z. Abedin. Credit default prediction modeling: an application of support vector machine. *Risk Management*, 19:158–187, 2017.

E. Muchai and L. Odongo. Comparison of crisp and fuzzy classification trees using Gini index impurity measure on simulated data. *European Scientific Journal*, 10(18), 2014.

V. Nair and G. E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

S. Narkhede. Understanding Confusion Matrix, 2018. URL https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62. [Online; accessed May 21, 2023].

N. Nehrebecka. Predicting the default risk of companies. Comparison of credit scoring models: LOGIT vs Support Vector Machines. *Ekonometria*, 22(2):54–73, 2018.

C. Nguyen, Y. Wang, and H. N. Nguyen. Random forest classifier combined with feature selection for breast cancer diagnosis and prognostic. *Journal of Biomedical Science and Engineering*, 2013.

Pablo Caceres. The Multilayer Perceptron - Theory and Implementation of the Backpropagation Algorithm, 2020. URL https://pabloinsente.github.io/the-multilayer-perceptron. [Online; accessed June 03, 2023].

I. Paryudi. What Affects K Value Selection In K-Nearest Neighbor. *International Journal of Scientific and Technology Research*, 8(7):86–92, 2019.

D. W. Pearson, N. C. Steele, R. F. Albrecht, A. G. Williamson, and P. Munson. Radial Basis Function Neural Networks in Credit Application Vetting Systems. In *Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference in Alès, France, 1995*, pages 301–304. Springer, 1995.

K. Pearson. *On the theory of contingency and its relation to association and normal correlation*, volume 1. Dulau and Company London, UK, 1904.

Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.

R. Persson. Weight of evidence transformation in credit scoring models: How does it affect the discriminatory power? Master's thesis, Lund University, Lund, Sweden, sep 2021.

M. Piao, Y. Piao, and J. Y. Lee. Symmetrical Uncertainty-Based Feature Subset Generation and Ensemble Learning for Electricity Customer Classification. *Symmetry*, 11(4):498, 2019.

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

S. P. Potharaju and M. Sreedevi. A Novel M-Cluster of Feature Selection Approach Based on Symmetrical Uncertainty for Increasing Classification Accuracy of Medical Datasets. *Journal of Engineering Science and Technology Review*, 10(6), 2017.

R. Questier, F.and Put, D. Coomans, B. Walczak, and Y. Vander Heyden. The use of cart and multivariate regression trees for supervised and unsupervised feature selection. *Chemometrics and Intelligent Laboratory Systems*, 76(1):45–54, 2005.

H. Ramchoun, Y. Ghanou, M. Ettaouil, and M. A. J. Idrissi. Multilayer Perceptron: Architecture Optimization and Training. *International Journal of Interactive Multimedia and Artificial Intelligence*, 4(1):26–30, 2016.

S. S. Rawat and A. K. Mishra. Review of Methods for Handling Class-Imbalanced in Classification Problems. *arXiv preprint arXiv:2211.05456*, 2022.

S. K. Roy, S. Manna, S. R. Dubey, and B. B. Chaudhuri. LiSHT: Non-Parametric Linearly Scaled Hyperbolic Tangent Activation Function for Neural Networks. In *International Conference on Computer Vision and Image Processing*, pages 462–476. Springer, 2022.

H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee. Recent Advances in Recurrent Neural Networks. *arXiv preprint arXiv:1801.01078*, 2017.

N. Sánchez-Marono, A. Alonso-Betanzos, and M. Tombilla-Sanromán. Filter methods for feature selection. A comparative study. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 178–187. Springer, 2007.

S. K. Satapathy, S. Dehuri, A. K. Jagadev, and S. Mishra. Empirical study on the performance of the classifiers in EEG classification. *EEG Brain Signal Classification for Epileptic Seizure Disorder Detection*, pages 45–65, 2019.

S. S. Satchidananda and J. B. Simha. Comparing decision trees with logistic regression for credit risk analysis. *International Institute of Information Technology, Bangalore, India*, 2006.

Sebastian Raschka. MultilayerPerceptron: A simple multilayer neural network, 2023. URL https://rasbt.github.io/mlxtend/user_guide/classifier/MultiLayerPerceptron/. [Online; accessed June 03, 2023].

B. Senliol, G. Gulgezen, L. Yu, and Z. Cataltepe. Fast Correlation Based Filter (FCBF) with a Different Search Strategy. In *2008 23rd International Symposium on Computer and Information Sciences*, pages 1–4. IEEE, 2008.

R. V. Sharan, H. Xiong, and S. Berkovsky. Benchmarking Audio Signal Representation Techniques for Classification with Convolutional Neural Networks. *Sensors*, 21(10): 3434, 2021.

S. Shi, R. Tse, W. Luo, S. D'Addona, and G. Pau. Machine learning-driven credit risk: a systemic review. *Neural Computing and Applications*, 34(17):14327–14339, 2022.

A. Siddique, M. A. Khan, and Z. Khan. The effect of credit risk management and bank-specific factors on the financial performance of the south asian commercial banks. *Asian Journal of Accounting Research*, 7(2):182–194, 2021.

BIT Mesra Society for Data Science. South German Credit Prediction, 2020. URL https://www.kaggle.com/c/south-german-credit-prediction/overview. [Online; accessed January 24, 2023].

C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986.

V. B. Surya, P. Haneen, A. A. Ahmad, B. A. Omar, and L. Ahmad. Effects of Distance Measure Choice on KNN Classifier Performance-A Review. *Mary Ann Liebert*, 2019.

H. Taud and J. F. Mas. Multilayer Perceptron (MLP). *Geomatic approaches for modeling land change scenarios*, pages 451–455, 2018.

A. Tharwat, T. Gaber, A. Ibrahim, and A. E. Hassanien. Linear discriminant analysis: A detailed tutorial. *AI communications*, 30(2):169–190, 2017.

R. Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

T. Toharudin, R. S. Pontoh, R. E. Caraka, s. Zahroh, Y. Lee, and R. C. Chen. Employing long short-term memory and Facebook prophet model in air temperature forecasting. *Communications in Statistics-Simulation and Computation*, 52(2):279–290, 2023.

Towards Data Science. Having an Imbalanced Dataset? Here Is How You Can Fix It., 2019. URL https://towardsdatascience.com/having-an-imbalanced-dataset-here-is-how-you-can-solve-it-1640568947eb/. [Online; accessed July 16, 2023].

J. V. Tu. Advantages and Disadvantages of Using Artificial Neural Networks versus Logistic Regression for Predicting Medical Outcomes. *Journal of Clinical Epidemiology*, 49(11):1225–1231, 1996.

R. J. Urbanowicz, M. Meeker, W. La Cava, R. S. Olson, and J. H. Moore. Relief-based feature selection: Introduction and review. *Journal of Biomedical Informatics*, 85: 189–203, 2018.

Y. B. Wah, N. Ibrahim, H. A. Hamid, S. Abdul-Rahman, and S. Fong. Feature Selection Methods: Case of Filter and Wrapper Approaches for Maximising Classification Accuracy. *Pertanika Journal of Science & Technology*, 26(1), 2018.

M. Wang, S. Lu, D. Zhu, J. Lin, and Z. Wang. A High-Speed and Low-Complexity Architecture for Softmax Function in Deep Learning. In *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pages 223–226. IEEE, 2018a.

T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pages 3304–3308. IEEE, 2012.

Y. Wang, W. Liao, and Y. Chang. Gated Recurrent Unit Network-Based Short-Term Photovoltaic Forecasting. *Energies*, 11(8):2163, 2018b.

Y. Wang, Y. Zhang, Y. Lu, and X. Yu. A Comparative Assessment of Credit Risk Model Based on Machine Learning - a case study of bank loan data. *Procedia Computer Science*, 174:141–149, 2020.

D. West. Neural network credit scoring models. *Computers & Operations Research*, 27 (11-12):1131–1152, 2000.

Y. Wu, H. Wang, B. Zhang, and K.-L. Du. Using Radial Basis Function Networks for Function Approximation and Classification. *International Scholarly Research Notices*, 2012, 2012.

Y. Xia, C. Liu, Y. Li, and N. Liu. A boosted decision tree approach using bayesian hyper-parameter optimization for credit scoring. *Expert Systems with Applications*, 78:225–241, 2017.

X. Yang, Y. Zhu, L. Yan, and X. Wang. Credit Risk Model Based on Logistic Regression and Weight of Evidence. In *3rd International Conference on Management Science, Education Technology, Arts, Social Science and Economics*, pages 810–814. Atlantis Press, 2015.

B. W. Yap, S. H. Ong, and N. H. M. Husain. Using data mining to improve assessment of credit worthiness via credit scoring models. *Expert Systems with Applications*, 38 (10):13274–13283, 2011.

Y. Yu, K. Adu, N. Tashi, P. Anokye, X. Wang, and M. A. Ayidzoe. RMAF: Relu-Memristor-Like Activation Function for Deep Learning. *IEEE Access*, 8:72727–72741, 2020.

Z. Yuan. Research on Credit Risk Assessment of P2P Network Platform:Based on the Logistic Regression Model of Evidence Weight. *Journal of Research in Business, Economics and Management*, 10(2):1874–1881, 2018.

C. Zednik and H. Boelsen. Overcoming Opacity in Machine Learning. *AISB Symposium Proceedings*, 2021.

A. Zhang. *Statistical Methods in Credit Risk Modeling.* PhD thesis, University of Michigan, 2009.