University of Essex

School of Mathematics, Statistics and Actuarial Science

MA981: DISSERTATION

Enhancing Financial Decision-Making through ML-Driven Credit Score Classification

Nithyashree Velayutham
2310618

Supervisor: Dr. Guo Wenxing

September 18, 2024
Colchester

Abstract

In the ever-evolving landscape of financial decision-making, the ability to accurately predict credit scores has become increasingly important for the financial sector. This study investigates the use of machine learning models to improve the accuracy and efficiency of credit score classification, addressing the limitations of traditional manual methods. By leveraging techniques such as data preprocessing, feature engineering, class balancing, and hyperparameter tuning, the research aims to enhance model performance and provide more accurate credit assessments. The study introduces two novel approaches: the use of Adjusted Outlyingness (AO) for outlier detection, which effectively handles skewness in the data, and a feature creation concept to derive potentially influential factors. These techniques, along with comprehensive data preprocessing, set the foundation for robust model development. A range of machine learning models were implemented, including ensemble methods like Extra Trees and Bagging, as well as K-Nearest Neighbors (KNN) and Multi-Layer Perceptron (MLP). Ensemble models, particularly Extra Trees, showed superior performance, achieving an accuracy of 91% after applying Random Over-Sampling (ROS) to address the class imbalance. The study also emphasizes the impact of hyperparameter tuning in enhancing the performance of models such as KNN and MLP, which initially underperformed. Feature importance analysis revealed that financial behaviors, such as interest rates and outstanding debt, are key factors influencing credit scores, while personal attributes like occupation had little impact. Despite some limitations, such as the use of a single dataset and limited exploration of alternative balancing techniques, the findings provide valuable insights into how machine learning can transform credit score prediction. Future research should explore advanced machine learning algorithms, emerging AI technologies, and real-time credit scoring systems to further enhance prediction accuracy and adaptability. This dissertation advances the field of financial risk assessment by showcasing how machine learning models can offer more accurate and flexible credit scoring systems. These advancements hold significant implications for improving decision-making processes in financial institutions and ensuring more equitable access to credit for consumers.

Keywords: Credit Score Prediction; Machine Learning; Adjusted Outlyingness (AO); Feature Creation; Random Over-Sampling (ROS); Hyperparameter Tuning; Feature Importance

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Background of the Research

Credit score evaluation has undergone a significant transformation since its inception by Fair Isaac Corporation (FICO) in 1989. FICO calculates credit scores for customers using data from their credit records, which financial institutions use to estimate credit risk and make lending decisions. Before the introduction of credit scores, lending decisions were made manually by bank managers or loan officers. This system had several limitations, including limited access to borrower information, inefficient decision-making, time-consuming processes, and restricted access to credit for many individuals. With the introduction of the FICO score, credit evaluation underwent a revolutionary transformation, implementing a data-driven approach that considers multiple factors. The FICO Score Model assesses credit scores by considering factors like payment history, credit utilization, the length of credit history, and the variety of credit types [1]. Credit scoring has played a crucial role in the financial market by improving access to credit for individuals while enabling lenders to assess risk with greater accuracy.

Figure 1.1 illustrates the range of credit scores and their corresponding classifications. The scores are categorized into five levels: Poor (300 to 579), Fair (580 to 669), Good (670 to 739), Very Good (740 to 799), and Excellent (800 to 850). The image also features a sample credit card, illustrating the link between credit scores and financial products. This modern system enables faster, more efficient, and fairer lending decisions while minimizing human error.

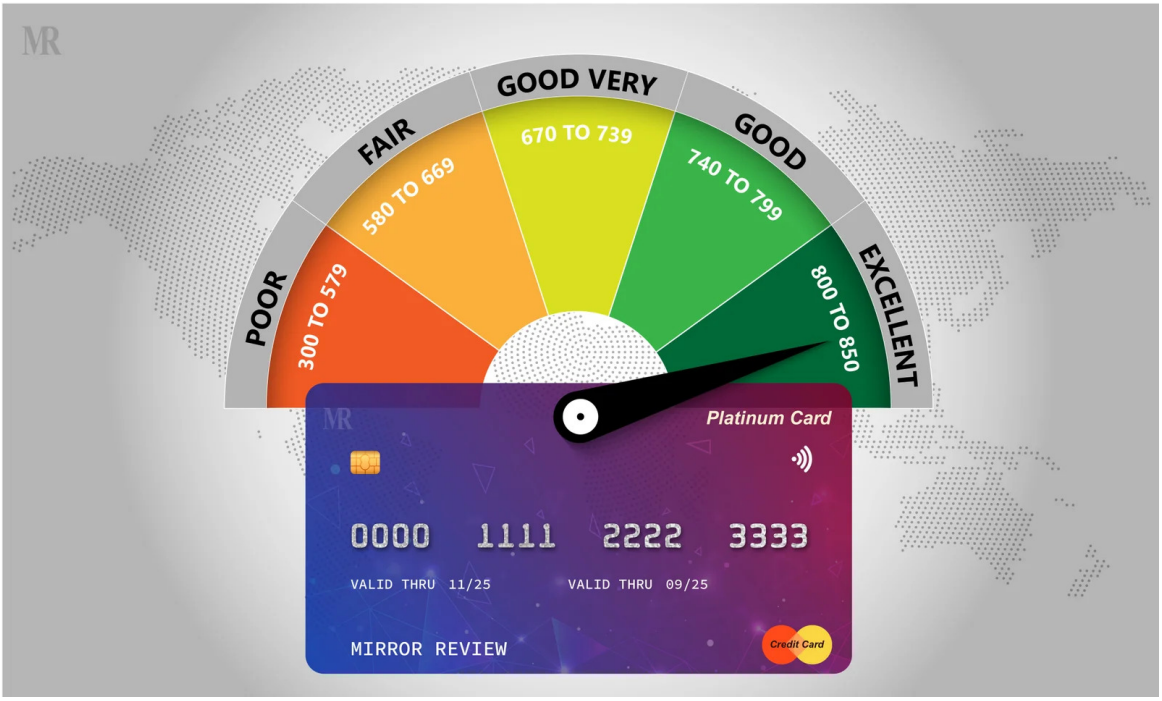

Figure 1.1: Understanding of credit score ranges [2]

In the UK, credit scores are primarily assessed through three credit reference agencies: Experian, Equifax, and TransUnion. Recent trends have shown significant changes in credit card usage and debt repayment patterns. During the COVID-19 pandemic, credit card spending in the UK initially saw a decline. However, by December 2023, according to the FICO analyst team, the average credit card spend reached £850, with outstanding balances hitting a record high of £1,780 [3]. Although credit card spending has stabilized, a growing number of customers are struggling to repay their debts, resulting in lower credit scores. This situation has heightened credit risk for financial institutions, while the number of credit card users continues to grow. These factors underscore the need for a more sophisticated, modern approach to assessing credit risk and managing the large volume of credit information.

1.2 Motivation

The growing challenges in credit scoring, including the increasing number of credit card users and rising credit risks, have made manual methods of predicting credit scores for large populations both impractical and time-consuming. These methods not only consume significant time but also struggle to capture the complexity of modern financial behavior, often leaving financial institutions with inadequate risk assessments. A recent survey conducted

by Fintech Futures [4] on the future of credit highlights how Artificial Intelligence (AI) and Machine Learning (ML) are being used to evaluate credit scores by incorporating additional data, particularly for individuals who do not qualify through traditional procedures. AI and ML offer significant advantages in processing vast datasets, identifying patterns that manual methods may miss. Additionally, an Australian startup, Rich Data Co, is using AI to detect borrower behavior and guide lending decisions. Research from Omdia predicts that the use of AI in financial companies will grow by 16% between 2023 and 2028. To help financial institutions address these challenges, this study proposes the development of a machine learning model for credit score prediction. The aim is to deliver accurate, time-efficient, and risk-sensitive results, assisting both lenders and borrowers in making informed decisions. By exploring advanced algorithms, this research seeks to enhance credit scoring systems and improve financial decision-making in today's increasingly complex credit environment.

1.3 Research Objectives

The objective of this study is to improve the accuracy and effectiveness of credit score prediction for financial institutions by implementing machine learning models, addressing the challenges of manual evaluation for large populations. The research focuses on:

- Examining how **preprocessing and high-quality data** contribute to increased model accuracy, reduce noise and errors in the data, and maintain data consistency.
- Determining how **handling outliers** using statistical approaches impacts model predictions and reduces data skewness.
- Implementing **feature engineering** concepts to identify new features that affect credit scores.
- Exploring how **balancing techniques** provide better model evaluation in terms of recall, precision, and F1 score.
- Identifying the **most important factors** influencing credit scores and their effective **integration** into predictive models.
- Comparing the results of various **machine learning approaches**, including **supervised, ensemble, and artificial neural network** techniques, in credit score prediction.

- Understanding **feature importance** to determine which variables contribute most to the model's predictions.
- Analyzing the **effectiveness of hyperparameter tuning strategies** on model performance to improve lower-performing models.

1.4 Contribution

In addressing the research objectives outlined above, this study makes several key contributions to the field of credit score prediction by incorporating data science strategies, ranging from basic to advanced analysis. The process encompasses the following steps and innovations:

1. **Data Preprocessing:** The study began with exhaustive data cleaning and handling of missing values through imputation methods. Outliers were precisely detected and controlled using Adjusted Outlyingness (AO) to avoid unwanted skewness in the data.
2. **Exploratory Data Analysis:** A comprehensive analysis was conducted to understand the nature of both numerical and categorical variables in the dataset, providing insights into their relationships with credit scores.
3. **Feature Engineering:** New features were derived using advanced techniques, and their impact on credit scores was evaluated, focusing on identifying the key predictors of creditworthiness.
4. **Data Encoding and Standardization:** Label and one-hot encoding were applied to categorical variables, and data was standardized for consistency across all features.
5. **Data Balancing:** The Random Oversampling (ROS) technique was investigated to address class imbalance, and its effects on model performance were carefully analyzed.
6. **Model Implementation:** Four machine learning models Extra Trees, Bagging, K-Nearest Neighbors (KNN), and Multi-Layer Perceptron (MLP) were implemented and compared for credit score prediction, providing a comprehensive evaluation of different algorithmic approaches.

7. **Performance Evaluation:** Multiple evaluation metrics, including accuracy, recall, precision, F1 score, and AUROC, were used to assess model performance, along with measuring training and testing times to evaluate computational efficiency.

8. **Feature Importance Analysis:** The most influential features were extracted from the best-performing models, offering insights into the key determinants of credit scores.

9. **Hyperparameter Tuning:** Advanced optimization techniques were applied to fine-tune model parameters, enhancing predictive performance for the least performing model.

10. **Comparative Analysis:** A comprehensive comparison of all analyses was conducted, synthesizing the findings and identifying the most effective approaches.

11. **Future Directions:** The study outlines limitations and suggests areas for future research, paving the way for further advancements in the field.

These contributions collectively enhance the understanding and utilization of machine learning in credit score prediction, providing both theoretical insights and practical approaches for more accurate and efficient financial risk assessment.

1.5 Outline of the Structure

This dissertation's remaining sections are arranged as follows: Chapter 2 provides a comprehensive literature review, examining current research on credit scoring techniques, the role of machine learning in credit scoring, and recent advancements in the field. Chapter 3 covers dataset preprocessing and exploratory data analysis, including data cleaning, handling missing values, and detecting outliers. Chapter 4 focuses on feature engineering techniques such as feature creation, encoding, and standardization to enhance model performance. Chapter 5 outlines the methodology, describing the machine learning models utilized, hyperparameter tuning strategies, and the metrics used for evaluation. Chapter 6 presents the results of the analysis, comparing model performance on imbalanced and balanced datasets, discusses the outcomes, identifies limitations, and suggests future research directions. Finally, Chapter 7 summarizes the key findings from this study on credit score prediction using machine learning and concludes the dissertation.

CHAPTER 2

Literature Review

The primary aim of this research is to predict credit scores and assess credit risk using automated methods. Classical manual calculations for large datasets are not only time-consuming but also insufficient, making automated credit score prediction essential. Numerous researchers have attempted to address these challenges, employing various approaches from basic statistical methods to advanced techniques such as artificial neural networks and deep learning. This literature review explores their contributions to this topic, analyzing the evolution of credit scoring methods and their effectiveness in automating and improving the prediction process.

2.1 Early Approaches to Credit Risk Assessment

The research **"Credit Risk Assessment Using Statistical and Machine Learning: Basic Methodology and Risk Modeling Applications"** by J. Galindo and P. Tamayo [5] published in 2000 addressed the financial crises of the 1980s and 1990s by predicting credit risk factors for individuals. Using a mortgage loan dataset, the researchers applied both statistical analyses and machine learning models to forecast credit risk. The target variable in the dataset was "default," which indicated whether a customer had defaulted based on their payment behavior. To analyze the dataset's error curve, they employed several methodologies, including Probit Regression, Decision-Tree CART (Classification and Regression Trees), Neural Networks, and KNN. The Decision-Tree CART model was the most effective, achieving an error

rate of 8.31% on a training dataset of 2,000 records. Other models, such as Neural Networks and KNN, performed slightly worse, with error rates of 11% and 14.95%, respectively. The Probit Regression model had an error rate of approximately 15.13%, and while it performed well in identifying non-defaulting customers, it struggled with defaulting ones. This study highlighted the importance of analyzing error curves and conducting complexity analysis to understand the strengths and limitations of each model. However, the researchers faced challenges, including a lack of financial datasets for comparative studies and difficulties in managing data bias, model complexity, and noise. They proposed that future research should focus on utilizing more extensive datasets and additional algorithms to improve credit risk predictions.

In another significant study, **"Evaluating Credit Risk Models"** (2000), J.A. Lopez and M.R. Saidenberg [6] addressed the growing need for accurate credit risk evaluation in increasingly complex financial markets. Their research introduced default models and multi-state models for estimating credit losses based on rating information. To overcome the limited availability of time-series data, they developed cross-sectional resampling techniques to generate estimates for simulated credit portfolios. The study proposed several evaluation methods, including expected loss estimates, critical value assessments, and full loss distribution estimations. For these, they recommended statistical tools such as Mincer-Zarnowitz regressions for expected loss estimates, the binomial method (commonly used in Value-at-Risk models) for critical value assessments, and tests for uniformity and independence of observed quantiles for full distribution estimates. Their work resulted in a novel approach for evaluating credit risk models despite limited historical data, enabling quantitative comparisons between models and providing a flexible framework for both credit portfolio managers and supervisors to monitor model performance. However, the study faced challenges such as dataset limitations and the static nature of the evaluation, which assessed models' performance on a fixed set of credits without accounting for dynamic portfolio management. Additionally, the dataset's dependence on economic conditions affected accuracy due to the lack of extensive data. The authors concluded by recommending further exploration of specific parameters to enhance evaluation methodologies and support comparative studies using different datasets.

2.2 Machine Learning Strategies in Credit Risk Analysis and Scoring

Research on credit risk estimation typically involves categorizing individuals as either good or bad loan applicants through the use of multiple classifiers. In his 2010 study, **"Multiple Classifier Application to Credit Risk Assessment"** [7], B. Twala evaluated the performance of various classifiers under different levels of noise. The classifiers included Artificial Neural Networks (ANN), Decision Trees (DT), Naïve Bayes Classifier (NBC), kNN, and Logistic Discrimination (LgD), with kNN and LgD being used for statistical pattern recognition. Twala's study reviewed the predictive accuracy of individual classifiers and explored whether combining these classifiers into ensembles could improve accuracy. To test this, four credit datasets were used, introducing different noise levels, and creating 20 ensemble methods by combining the five classifiers in various ways. The results showed that as noise levels increased from 5% to 50%, the predictive accuracy of all individual classifiers decreased. However, ensemble classifiers consistently outperformed single classifiers. Among the ensembles, those combining NBC and DT produced the best results compared to other combinations. The study employed multiple classifier system architectures, such as Static Parallel, Multi-stage, and Dynamic classifier selection methods, integrating various techniques like voting, rank-based, and probabilistic methods. Notably, the Static Parallel architecture demonstrated the best results. In the final phase of the study, Twala evaluated several sampling and combination techniques, including bagging, boosting, stacking, feature selection, and randomization, all of which led to statistically significant improvements over individual classifiers. Despite the promising results, challenges persisted in handling noise and managing the computational complexity of ensemble methods. To address these challenges, the author recommended future research on developing more robust methods to handle noise and reduce computational complexity, while also emphasizing the need for larger, balanced datasets to improve credit risk prediction accuracy.

In 2016, R. E. Turkson, E. Y. Baagyere, and G. E. Wenya presented their paper, **"Machine Learning Approach for Predicting Bank Credit Worthiness"** [8], at the *Third International Conference on Artificial Intelligence and Pattern Recognition (AIPR)*. The goal was to develop an effective algorithm using real-world bank data to evaluate customers' ability to repay credit

in the subsequent month. Initially, the dataset contained 23 features related to customer creditworthiness, from which the most influential factors were identified. The researchers applied various machine learning techniques to both the full dataset and the subset of important features, comparing the results to determine the most effective approach. Their analysis covered a wide range of methodologies, from supervised learning to neural networks, and included unsupervised learning and discriminant analysis. Of the approximately 15 algorithms used, boosting methods demonstrated stronger performance, with bagging emerging as the top performer, achieving a 98% accuracy score. To refine their approach further, they focused on the five most important features from the original 23. These features were trained using algorithms such as Extra Trees, CART, AdaBoost, Random Forest, and logistic regression. In this more focused analysis, AdaBoost outperformed the other models, it achieved an accuracy of 82%. In the final step, the researchers employed linear regression models to predict credit scores using only the top three features. Although this feature reduction did not show a noticeable improvement over using five features, it confirmed the efficacy of the original dataset for credit score prediction. The authors concluded by stating their intention to develop a hybrid machine learning model in the future to better understand the influential features impacting customer credit scores.

In 2020, S.K. Trivedi conducted a study titled **"A Study on Credit Scoring Modelling with Different Feature Selection and Machine Learning Approaches"** [9], aiming to evaluate individual credit risk by implementing various feature selection and machine learning models. The study applied feature selection techniques such as chi-squared, information gain, and gain ratio to a German bank dataset, which consisted of 20 features. These selected features were then trained separately using Naïve Bayes, Decision Tree (C5.0), Random Forest, and SVM (Support Vector Machine) with the Radial Basis Function (RBF) kernel. The results varied across feature selection techniques: with the chi-squared method, Random Forest achieved the highest accuracy of 76.18%; in the gain ratio method, SVM and Random Forest both performed well, scoring around 77.40%; and with the information gain method, SVM (RBF) outperformed the others, achieving an accuracy of 78.24%. Notably, 10-fold cross-validation produced better results compared to a 66-35% data split, with Random Forest achieving accuracies of 93.12% and 91.90% for the chi-squared and gain ratio methods, respectively. The study also considered training and testing times for model efficiency. While the study successfully identified significant predictors for credit scoring, Trivedi acknowledged the

limitations of using only the German dataset and called for more diverse datasets to enhance the generalizability of the results. In future research, the author proposed exploring additional feature selection techniques and experimenting with new ways of splitting datasets to better understand model performance.

Finally, in the *Journal of Information Systems and Informatics*, T. Mokheleli and T. Museba published the article **"Machine Learning Approach for Credit Score Predictions"** [10] in 2023. The paper addresses the challenges faced by banks and financial institutions in managing the growing number of credit products. The authors aimed to develop a dynamic heterogeneous ensemble credit model that combined machine learning and boosting methods, with a focus on improving results for imbalanced datasets. To balance the data, they applied the Synthetic Minority Oversampling Technique (SMOTE). Their methodology centered on the Adaptive and Dynamic Heterogeneous Ensemble (ADHE), which integrates XGBoost and SVM for credit score prediction. Recognizing the frequent changes in customer datasets, the authors applied a classifier pool generation technique to ensure the selection of diverse and accurate models. They also used the Selection by Accuracy and Diversity (SAD) algorithm for classifier selection and combined SVM and XGBoost to maximize diversity, which led to better results and identified key model parameters. To address dimensionality issues and reduce computational costs, they introduced the Particle Swarm Optimization (PSO) algorithm for parameter optimization. Their methodology was evaluated using five datasets. The Random Forest algorithm emerged as the best-performing individual classifier on the Japanese dataset, achieving an accuracy of 87.9%. Notably, ADHE outperformed other ensemble models, achieving a 93.4% accuracy on the same dataset. While the study successfully addressed class imbalance and yielded accurate results, it encountered challenges related to cross-validation and grid search techniques, particularly on larger datasets due to their time-consuming nature. Moving forward, the authors proposed exploring more efficient feature selection techniques and improving approaches to handle imbalanced datasets in future research.

2.3 Deep Learning and Explainable AI Approaches in Credit Scoring

In 2023, Peng Du and Hong Shu conducted a study titled **"Exploration of Financial Market Credit Scoring and Risk Management and Prediction Using Deep Learning and Bionic**

Algorithm" [11], aiming to minimize financial market risk by exploring advanced deep learning models, particularly Recurrent Neural Networks (RNN) and Bidirectional Recurrent Neural Networks (BRNN). The authors also applied bionic-based optimization techniques to enhance path analysis. Their methodology featured an Integrated Financial Risk Management System that trained and integrated multiple base models simultaneously. The study applied parameter tuning across datasets, focusing on factors like classification learning rate, node weight, maximum depth, random sampling ratio, and feature ratio to improve performance through a Gradient Boosting classifier. In their analysis, XGBoost achieved the highest AUC scores across three datasets (93%, 84%, and 93%), with BRNN also performing strongly. When BRNN was integrated with LR and XGBoost, the model achieved an accuracy of 89% on the Australian dataset and 77% to 87% accuracy on the German and Japanese datasets, respectively. The study also explored various optimization methods, such as Stochastic Gradient Descent (SGD), Momentum, and AdaBound, finding that BRNN fine-tuned with SGD yielded 87% accuracy on the Australian dataset, while AdaBound achieved the best results for the German dataset at 78%. Although the integrated models generally outperformed single models, the researchers noted limitations, such as the lack of balancing techniques and a focus on significant parameters. They suggested future research should adopt all features for a more comprehensive analysis and implement Big Data approaches.

In the same year, F.M. Talaat, A. Aljadani, M. Badawy, and colleagues published **"Toward Interpretable Credit Scoring: Integrating Explainable Artificial Intelligence with Deep Learning for Credit Card Default Prediction"**[12], addressing the challenge posed by the "black box" nature of traditional machine learning and deep learning models. Their study introduced Explainable Artificial Intelligence (XAI) to improve transparency in deep learning credit scoring models. Using a real-world dataset, the authors implemented feature selection techniques like Recursive Feature Elimination and Lasso, followed by training the model with CreditNetXAI. The model configuration included parameters such as the number of layers, activation functions, node distribution, loss functions, and learning rate. To enhance performance, hyperparameter tuning was applied using grid search and random search, and dropout regularization was used to avoid overfitting. A key component of the study was the use of Shapley Additive exPlanations (SHAP) to provide insights into feature importance and individual prediction scores. Compared to other models like XGBoost, SVM, and Random Forest, CreditNetXAI outperformed them, achieving an accuracy of 83.50%. The feature

importance analysis was instrumental in identifying the most influential factors driving the model's performance. However, the authors expressed concerns that CreditNetXAI could produce biased results in real-world data, given that such data frequently changes. They proposed integrating CreditNetXAI with other cutting-edge techniques, such as One-Class Neural Networks (OCNN) and You Only Look Once (YOLO), to improve adaptability and efficiency in future studies.

The paper by Qian et al. (2023) titled **"Soft Reordering One-Dimensional Convolutional Neural Network for Credit Scoring"** [14] introduces the Soft Reordering One-Dimensional Convolutional Neural Network (SR-1D-CNN), a novel deep learning model specifically designed for credit scoring. This model addresses the challenge of applying Convolutional Neural Networks (CNNs) to tabular data, which lacks the spatial local correlation typically found in image or text data. The authors propose a soft reordering mechanism that transforms tabular data into a format that better suits CNN learning, allowing the model to capture complex feature interactions. Through comprehensive experiments on real-world credit scoring datasets, SR-1D-CNN outperformed traditional machine learning models such as Logistic Regression, Decision Trees, and ensemble methods like LightGBM and XGBoost, particularly when applied to larger datasets. For instance, SR-1D-CNN improved the AUC score on the Polish dataset by 29.14% compared to the vanilla 1D CNN, with further improvements of 2.50%, 0.62%, and 1.07% on the A-share, LendingClub, and HomeCreditDefaultRisk datasets, respectively. Compared to other deep learning models such as DeepFM, DCN-V2, and TBN, SR-1D-CNN demonstrated exhibited higher accuracy and computational efficiency, especially as the size of the data increased. In brief the paper highlights the importance of hyperparameter tuning for optimal performance. This work makes a notable contribution to advancing credit scoring methodologies by adapting CNNs for more effective use with tabular data. By making CNNs more applicable to tabular data, the study marks notable advancements in credit scoring methodologies.

CHAPTER 3

Dataset Preprocessing and EDA

3.1 Dataset Description

The dataset used for this research, **"Enhancing Financial Decision-Making through ML-Driven Credit Score Classification"** was collected from Kaggle [15], a data science community offering resources and data. The credit score classification dataset contains 100,000 rows and 28 columns. This raw data includes individuals' credit records. The dataset has some anomalies such as irrelevant entries, missing values, negative values, and datatype mismatches. Using this dataset, this study explores concepts such as data preprocessing, outlier handling, and model development. Table 3.1 shows the 28 features with descriptions. Among them, *Occupation*, *Credit_Mix*, *Payment_of_Min_Amount*, *Payment_Behaviour*, and *Credit_Score* are categorical columns in the dataset, which are also important factors in credit score prediction. *Credit_Score* is the target variable in the dataset, which has "Good," "Bad," and "Standard" categories. "Good" indicates a customer with a good credit score, "Bad" indicates a poor credit score, and "Standard" represents an average credit score. The *Occupation* column contains the job titles of the customers. *Credit_Mix* has the same categories as *Credit_Score*. *Payment_of_Min_Amount* has attributes such as "Yes" and "No". *Payment_Behaviour* has 6 entries with combinations of "high", "low", "large", "small", and "medium". Figure 3.1 illustrates the distribution of credit scores in the dataset. Standard accounts for 53.2% of the dataset, Poor is 29%, and Good is around 17.8%. This distribution indicates that the class is imbalanced, and there is a need for balancing techniques.

Column	Description
ID	A distinct identifier assigned to each individual record
Customer_ID	A distinct identifier assigned to each customer
Month	Denotes the month of the data collection process
Name	Customer's name
Age	Customer's age
SSN	The individual's personal Social Security Number
Occupation	The job title held by the customer
Annual_Income	The yearly earnings of the customer
Monthly_Inhand_Salary	Monthly salary received
Num_Bank_Accounts	The count of bank accounts the customer holds.
Num_Credit_Card	The count of credit cards held by the customer
Interest_Rate	Interest rate on credit accounts
Num_of_Loan	Number of current loans
Type_of_Loan	Types of loans taken
Delay_from_due_date	Days late in making payments
Num_of_Delayed_Payment	The total instances of overdue payments
Changed_Credit_Limit	Indicator if credit limit has changed
Num_Credit_Inquiries	The count of credit inquiries
Credit_Mix	Range of credit types held
Outstanding_Debt	Current debt owed
Credit_Utilization_Ratio	Ratio of credit usage to available credit
Credit_History_Age	Length of credit history
Payment_of_Min_Amount	Indicator if minimum payment was made
Total_EMI_per_month	Total monthly EMI
Amount_invested_monthly	Monthly investment amount
Payment_Behaviour	Description of payment patterns
Monthly_Balance	Balance after monthly transactions
Credit_Score	Categorized credit score

Table 3.1: Description of columns in the credit score classification dataset

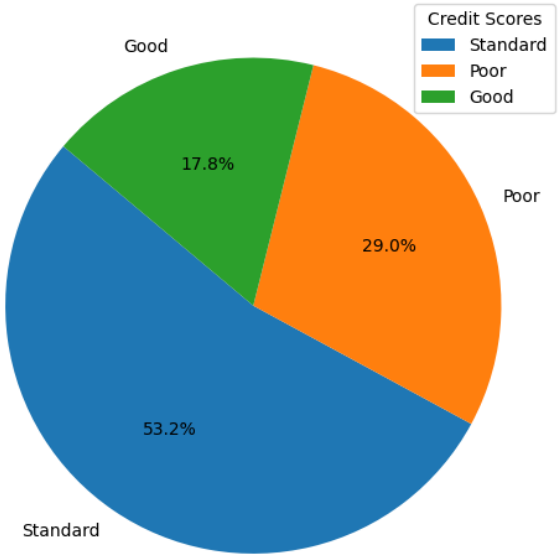

Figure 3.1: Distribution of Credit Scores in the Dataset

3.2 Dataset Preprocessing

To address the dataset's anomalies, the study begins with preprocessing to ensure high quality model building. Before starting the preprocessing steps, unnecessary columns that were not useful for the analysis were removed. These include *ID*, *Customer_ID*, *Month*, *SSN*, *Type_of_Loan*, *Name*, *Monthly_Inhand_Salary*, and *Credit_History_Age*. Afterward, the dataset was checked for duplicate entries, and none were found.

3.2.1 Data Cleaning

The dataset contains some unique and unrealistic values. Underscores were found in columns such as *Age*, *Annual_Income*, *Occupation*, *Num_of_Delayed_Payment*, *Changed_Credit_Limit*, and *Amount_invested_monthly*. Unrealistic values, such as 500 for age, were also present. The underscore values were replaced with NaN values. The *Payment_of_Min_Amount* column had entries like "NM" instead of "No", which were alternated with "No". Entries such as "@9#%8" were found in the *Payment_Behaviour* column and were removed by filling with the mode value. "Low_spent_Small_value_payments" appears to be the mode. *Credit_Mix*, a categorical column, also had underscores which were swapped by the mode. Columns

like *Age*, *Num_Bank_Accounts*, *Num_of_Loan*, *Delay_from_due_date*, *Num_of_Delayed_Payment*, *Changed_Credit_Limit*, and *Monthly_Balance* contained negative values. These were handled by replacing them with absolute values, except for *Num_of_Loan*. For *Num_of_Loan*, negative values were set to zero because the number of loans should not be represented by absolute values. The *Age* column was fixed by setting the range to a maximum of 100 years.

3.2.2 Handling Missing and Mismatched Data

The missing values in the dataset can be addressed using several techniques, such as Mean, Mode, and Median Imputation, or more advanced methods like the KNN Imputation.

- The **Median Imputation Method** involves replacing missing values with the median, which is the middle value of the non-missing data in a particular column. This method is straightforward and commonly used for handling missing data in numerical columns.
- **Mode Imputation** substitutes missing values with the most frequently occurring value (the mode) from the non-missing data in the column. This approach is especially useful for categorical columns where the mode reflects the most common entry.

Figure 3.2 shows the details of missing values in the dataset for *Num_of_delayed_payment*, *Changed_credit_limit*, *Num_of_credit_enquiries*, *Amount_invested_monthly*, and *Monthly_balance*. The highest number of missing values in the dataset seems to be in *Num_of_delayed_payment*. With missing values in the dataset, machine learning algorithms won't perform well. To resolve this issue, the dataset needs to be treated with a proper approach. The paper [16] used mean, mode, median, and KNN imputation for handling missing values and compared them. The mean, median, and mode showed results equal to KNN imputation. They also stated that while modern approaches are available, they are complex, and this simple approach shows equal results. Before applying imputation, the dataset had datatype mismatches. This issue was addressed by changing the appropriate datatype for the columns. Then, median imputation was applied to numeric columns and mode imputation to object columns because it is reasonable to fill in the most occurring value. After applying these techniques, the dataset has no missing or mismatched data.

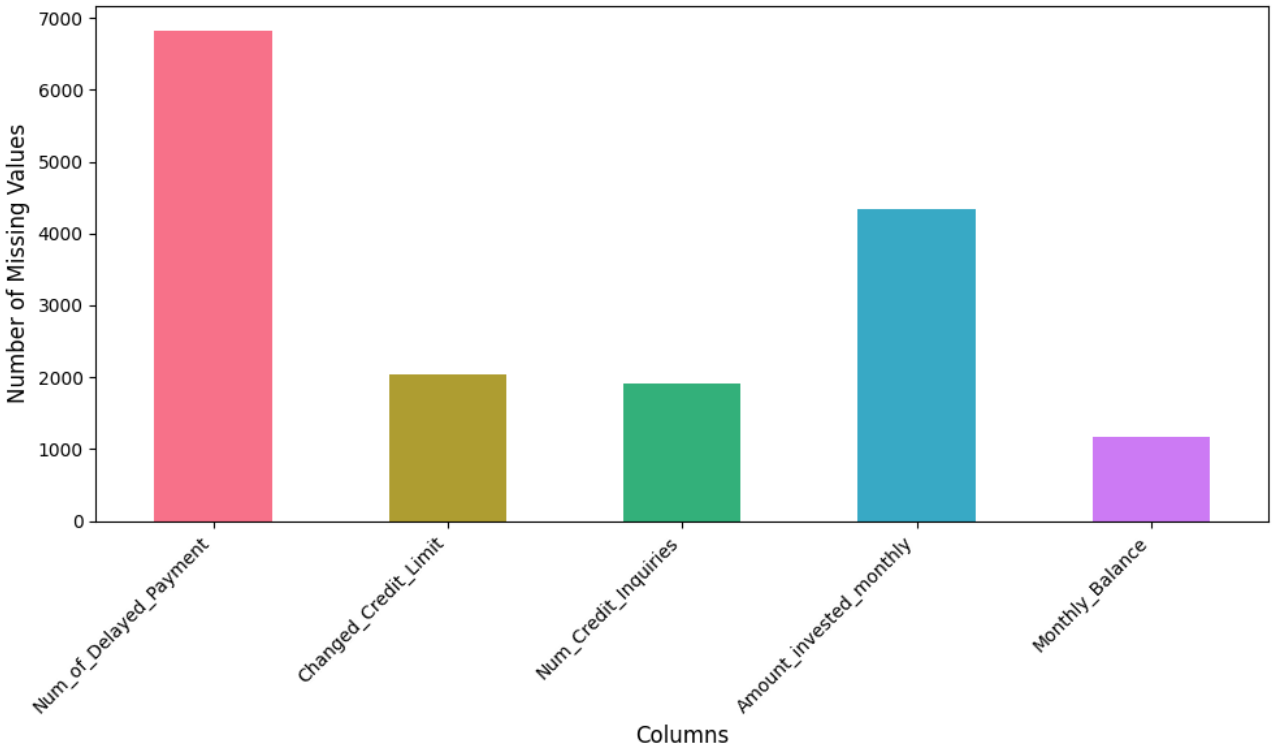

Figure 3.2: Distribution of Missing Values in the Credit Score Dataset

3.2.3 Outlier Detection and Handling Methodology

This study implements an outlier detection and handling strategy to address the challenges presented by skewed data distributions. The research "Outlier Detection for Skewed Data" [17] introduces the modern approach of Adjusted Outlyingness (AO), which this study adopted to detect and handle outliers based on skewness. AO showed better results compared to Traditional methods for detecting outliers. Skewness refers to the asymmetry in data distribution, where the distribution has a long tail on one side. The skewness values can be positive, negative or zero. If it's positive, it shows the right-skewed distribution with the right tail longer compared to the left tail. If it is negative, the left tail has larger distribution than the right tail, and zero indicates the perfectly symmetrical distribution.

Summary statistics were calculated to identify skewed variables in the dataset's numeric columns. Table 3.2 shows that several variables, including *annual_income*, *interest_rate*, *num_of_loan*, and *num_of_delayed_payment*, are highly skewed, as evidenced by their mean values being significantly greater than their median values. The interpretation of skewness involves considering the relationship between mean and median values. When mean and

median are roughly equal or only slightly different, it indicates no skew (e.g., age). A slightly higher mean than median suggests mild skew (e.g.,delay_from_due_date). An extremely higher mean than median indicates extreme positive skew(e.g.,monthly_balance). This analysis revealed that most columns in the dataset are highly skewed, highlighting the importance of using AO to handle them effectively. By implementing this approach, the study aims to improve the accuracy of outlier detection and enhanced data analysis, ensuring more reliable results.

Column Name	**Mean**	**Median (50th Percentile)**	**Skewness Indicator**
Age	33.32	33	No significant skew
Annual_Income	176,841	37,579	Positive skew
Num_Bank_Accounts	17.13	6	Positive skew
Num_Credit_Card	22.45	5	Positive skew
Interest_Rate	72.59	13	Positive skew
Num_of_Loan	10.71	3	Positive skew
Delay_from_due_date	21.11	18	Mild positive skew
Num_of_Delayed_Payment	29.69	14	Positive skew
Changed_Credit_Limit	10.44	9.41	Mild positive skew
Num_Credit_Inquiries	27.43	6	Positive skew
Outstanding_Debt	1426.29	1166.08	Mild positive skew
Credit_Utilization_Ratio	32.28	32.31	No significant skew
Total_EMI_per_month	1396.54	69.27	Positive skew
Amount_invested_monthly	614.01	136.01	Positive skew
Monthly_Balance	3.09×10^{22}	336.68	Extreme positive skew

Table 3.2: Summary Statistics of Skewness in Credit Score dataset's column

Skewness Categorization

Adjusted Outlyingness (AO) is a statistical tool used to measure how much a data point stands out as an outlier in a dataset. It's particularly effective for skewed datasets. Unlike traditional methods that calculate outliers using the mean or median, AO considers the skewness of the data using different factors, such as median as a measure of central tendency and the median absolute deviation (MAD) to assess variability. The process begins by categorizing the skewed

data into high and low skewness using thresholds of -0.5 and 0.5. If a column's skewness falls within this range, it suggests that the distribution is approximately symmetric or has low skewness. This implies that the data isn't heavily skewed in either direction. Values outside this range indicate high skewness. More specifically, a skewness value less than -0.5 indicates negative skewness, while a skewness value greater than 0.5 indicates positive skewness. After categorizing the skewness, AO uses the median and MAD to calculate separate measures for the lower and upper parts of the distribution. These measures are based on the median and the MAD above and below the median, adjusted according to the direction of skewness.

Skewness Handling in Outlier Detection

Once the skewness of each column is categorized, different approaches are applied to handle outliers based on the degree of skewness. For highly skewed columns, the process replaces outliers with the median value. The process for each highly skewed column involves calculating AO values to identify outliers, determining upper and lower bounds using the Interquartile Range (IQR) method, which is used to understand the spread and dispersion of data, and then replacing any values whose AO falls outside these bounds with the column's median. For columns with low skewness, the process also uses AO but takes a different approach. It determines upper and lower boundaries for outliers based on the IQR method and identifies data points with AO values beyond these boundaries as outliers. However, instead of removing or replacing outliers with a central value, it caps them at the boundary values. Outliers above the upper boundary are set to the upper boundary value, while those below the lower boundary are set to the lower boundary value. This method effectively reduces the impact of extreme values while preserving the overall shape of the data distribution. As a result of this process, the majority of columns become free from outliers. The effectiveness of this approach can be visualized in Figure 3.3, which presents box plots comparing the data before and after handling outliers. A total of 15 columns contain outliers, as shown in Table 3.2. Five of these columns have outliers, while the remaining columns are free from extreme values. The box plots typically show a tighter data range and spread in the "After" plots, indicating a reduction or removal of extreme values of outliers from the dataset. This approach provides a more nuanced and effective way of dealing with outliers, addressing the skewness of each column and applying appropriate methods to maintain data integrity while managing extreme values.

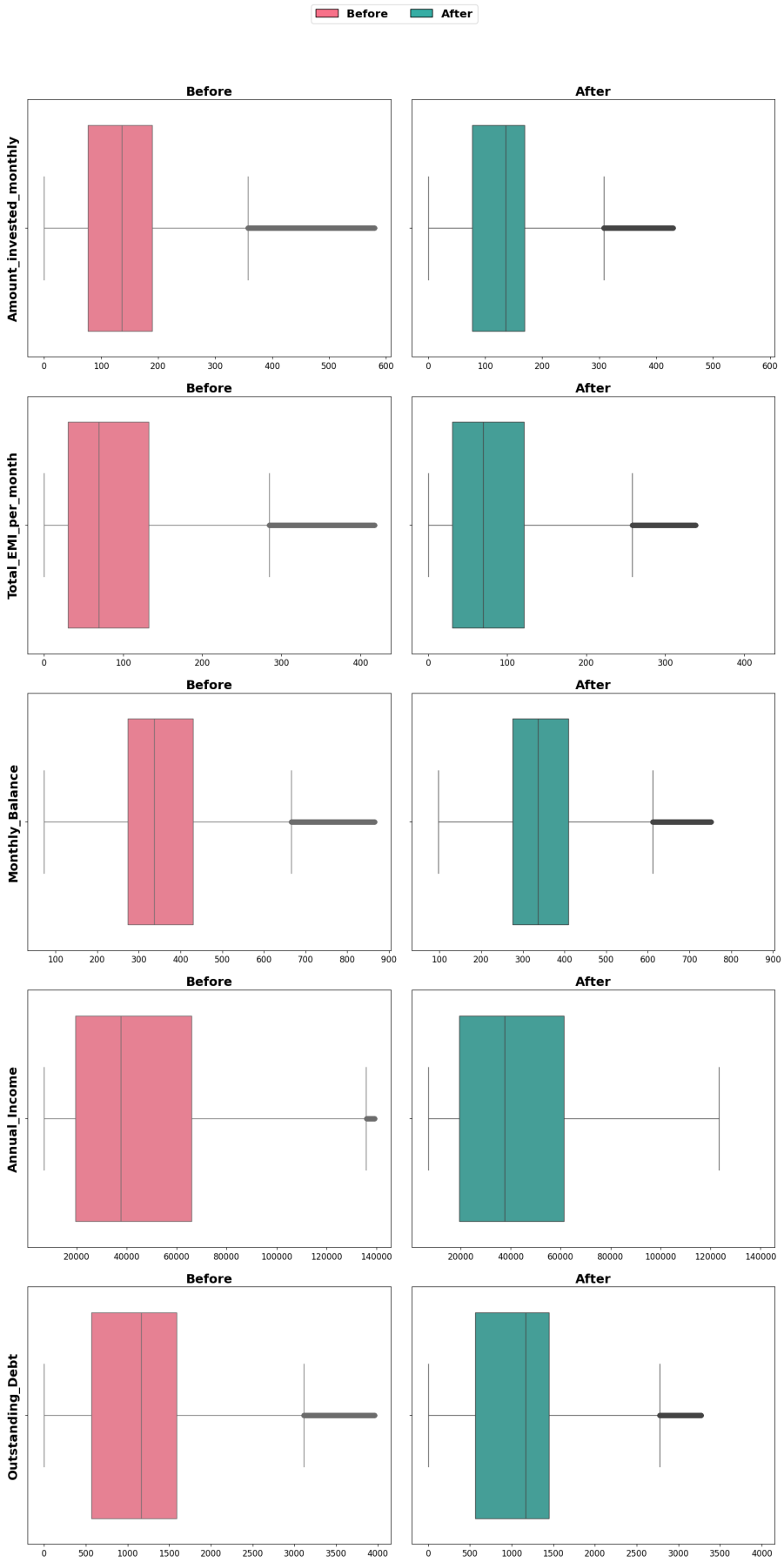

Figure 3.3: Feature Distributions: Before and After Outlier Removal

3.3 Dataset Exploration

The credit score dataset was explored using correlation and bivariate analysis to identify the influencing columns that contribute to model performance. This analysis helps in better understanding the factors that affect credit scores.

3.3.1 Correlation analysis

Correlation analysis measures the strength of relationships between variables, using coefficients ranging from -1 to +1. A perfect positive correlation is +1, a perfect negative correlation is -1, and a value of 0 indicates no correlation. A correlation matrix displays these coefficients for multiple variables, with the diagonal elements showing the correlation of each variable with itself, which is always +1. Heatmaps are often used to visually represent correlation matrices.

The correlation matrix in Figure 3.4 provides a clear visualization of the relationships between the numerical variables in the dataset. Each cell represents the correlation coefficient between two variables. For example, *Num_Bank_Accounts* and *Num_Credit_Card* have a positive correlation of 0.57, indicating that individuals with more bank accounts tend to have more credit cards, likely due to managing multiple financial accounts. Conversely, the strongest negative correlation, -0.77, is observed between *Credit_Utilization_Trend* and *Changed_Credit_Limit*, suggesting that higher credit utilization is associated with fewer changes in credit limits. Additionally, *Age* shows weak correlations with most other variables, indicating that it does not strongly influence financial behaviors or metrics within this dataset. The color gradient in the heatmap visually enhances the interpretation, with darker shades representing stronger positive correlations and lighter shades or yellow tones representing weaker or negative correlations. This analysis is important because it helps identify which variables are closely related, allowing us to understand underlying patterns in the dataset. Strong correlations between variables can provide insights into which features are likely to influence model predictions, while weak correlations may suggest that certain features are less relevant. This helps in feature selection, improving the efficiency and accuracy of machine learning models by focusing on the most impactful variables. Additionally, understanding the relationships between variables can aid in addressing multicollinearity issues, which can

negatively affect model performance.

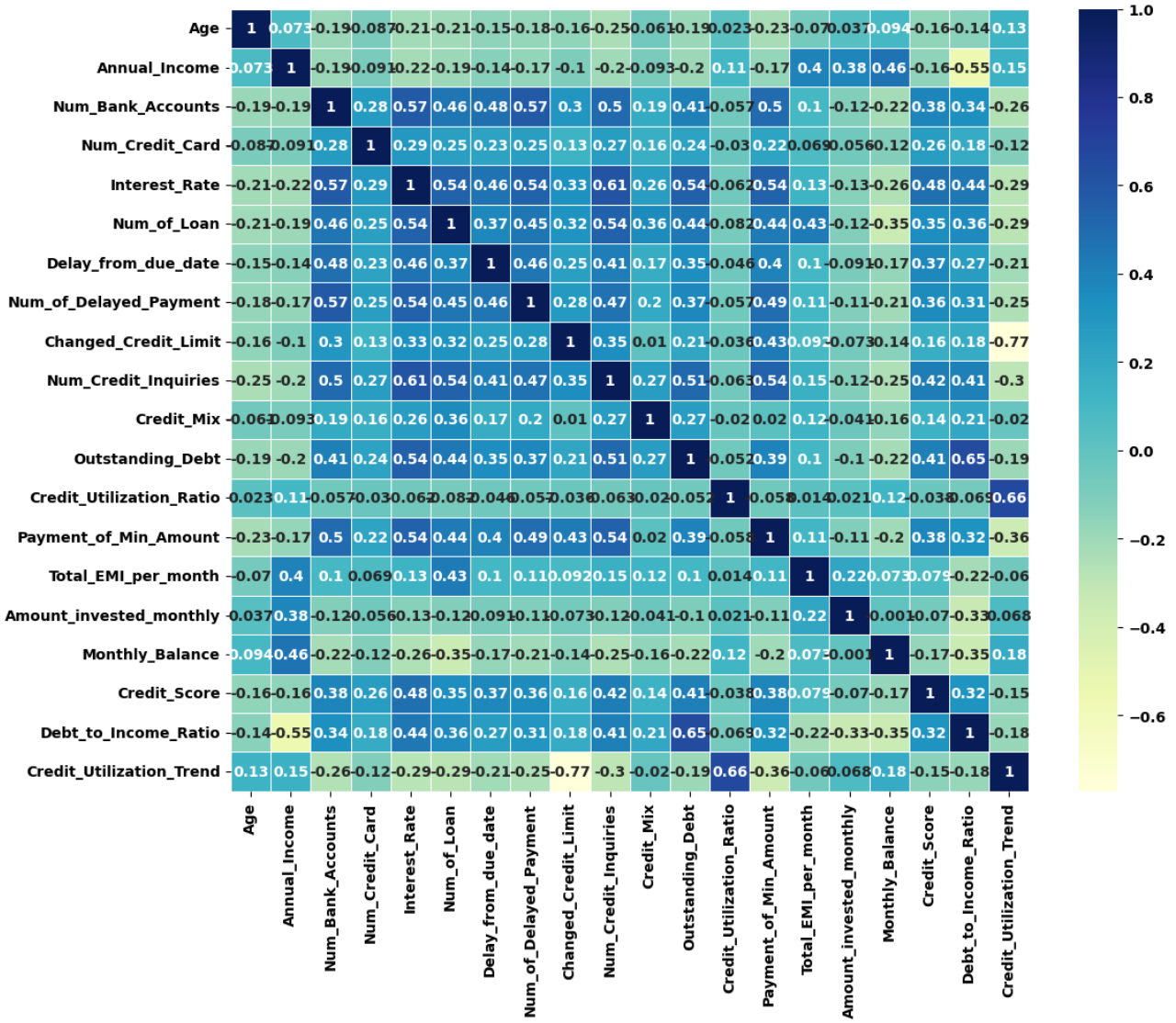

Figure 3.4: Correlation Matrix for Credit Score Dataset

3.3.2 Bivariate Analysis

Bivariate analysis helps to compare the relationship between two variables. In this analysis, a credit score is compared with numerical and categorical columns to identify the influencing features responsible for a good credit score, those that indicate a standard score, and columns with lower values that have no impact on the credit score, resulting in a poor score. In this study, violin plots are used for numerical columns versus credit score, while stacked bar charts are used for categorical columns. By analyzing, this research intended to find the most significant predictors of credit scores and understand the complex relationships between

various factors and creditworthiness.

Numerical Variables vs. Credit Score

The violin plots in Figure 3.5 compare various numerical columns in the dataset against the target variable, credit score. These plots illustrate the distribution of credit score categories good, standard, and poor across different numerical features. Features such as *Annual_Income*, *Credit_Utilization_Ratio*, *Amount_Invested_Monthly*, and *Monthly_Balance* show a strong association with good credit scores. This suggests that individuals with higher incomes, lower credit utilization, consistent investment patterns, and well-managed balances tend to achieve better credit scores. On the other hand, poor credit scores are linked to higher values in variables like *Outstanding_Debt*, *Num_Credit_Inquiries*, *Num_of_Delayed_Payment*, and *Interest_Rate*. These factors reflect greater financial strain, including larger debt amounts, frequent credit inquiries, delayed payments, and higher interest rates, all of which signal reduced financial stability. Standard credit scores generally fall between these extremes, with moderate values in most of these features. Interestingly, variables like *Num_Bank_Accounts*, *Num_Credit_Card*, and *Num_of_Loan* show less differences across the credit score categories, suggesting that the relationship between these variables and credit scores may be more complex or influenced by other factors. The violin plots analysis provides valuable insights for feature selection and engineering, helping to identify which variables are most likely to improve model performance and which may require further transformation.

Categorical Variables vs. Credit Score

The stacked bar charts illustrate the relationships between categorical variables and credit scores. Figure 3.6 shows the distribution of good, standard, and poor credit scores for each categorical variable. Features like *Credit_Mix* and *Payment_of_Min_Amount* demonstrate strong correlations with credit scores. A "Good" credit mix and paying more than the minimum amount are associated with higher proportions of good credit scores, indicating that diverse credit portfolios and responsible payment behavior contribute to better creditworthiness. *Payment_Behaviour* also shows notable variations, with "High_spent_Small_value_payments" linked to better credit scores, while "Low_spent_Large_value_payments" correlates with poorer scores. *Occupation*, while showing some variations, doesn't exhibit significant differences in credit score distributions across categories, suggesting a less direct influence on

credit scores. The "Others" occupation category shows a slightly higher proportion of poor credit scores. These patterns reveal that certain financial behaviors and credit management strategies have a more significant impact on credit scores than occupational categories. This analysis provides useful information for feature importance in modeling and understanding key factors in credit score determination, highlighting the complex interplay between categorical variables and creditworthiness.

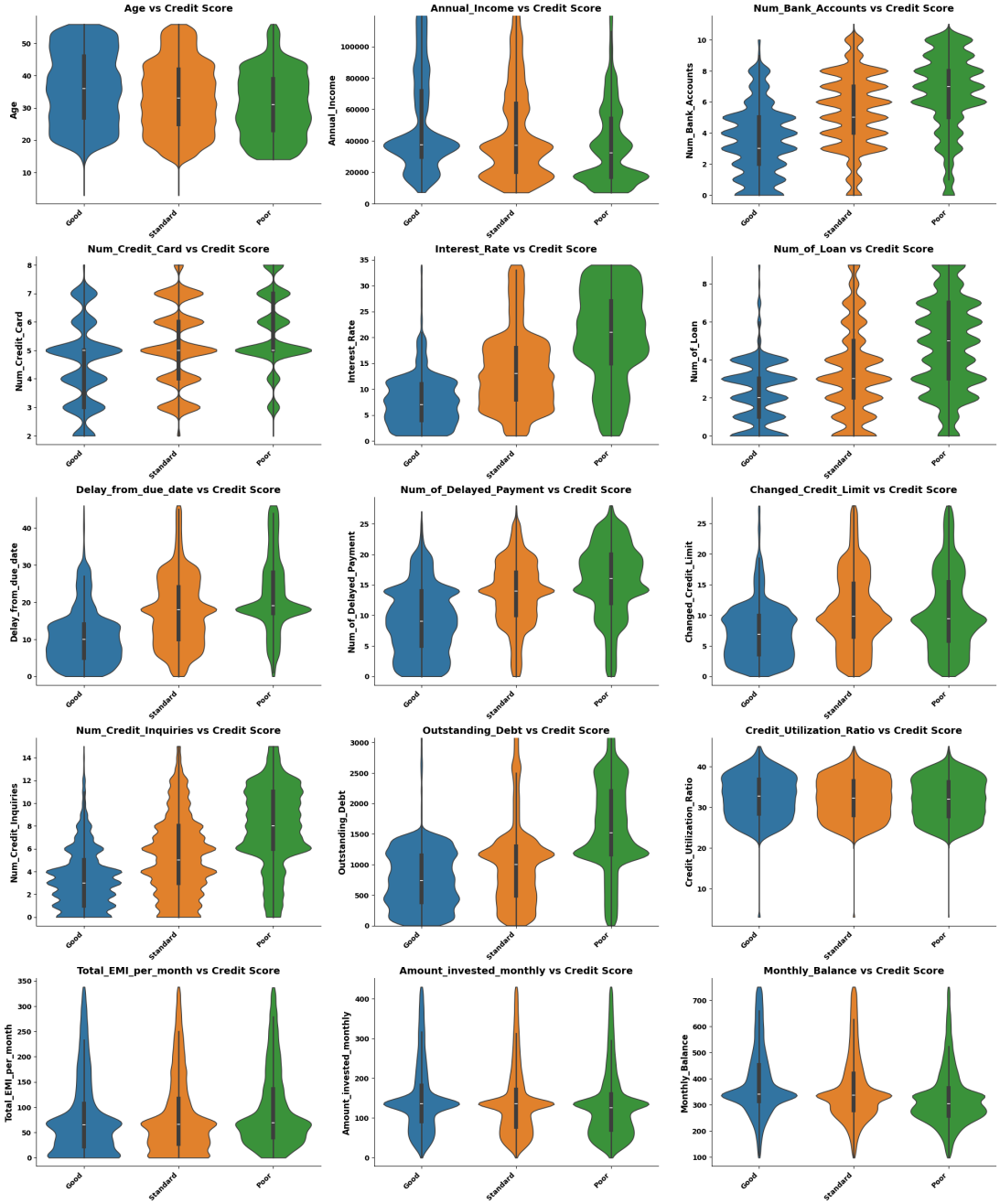

Figure 3.5: Distribution of Numerical Variables by Credit Score Category

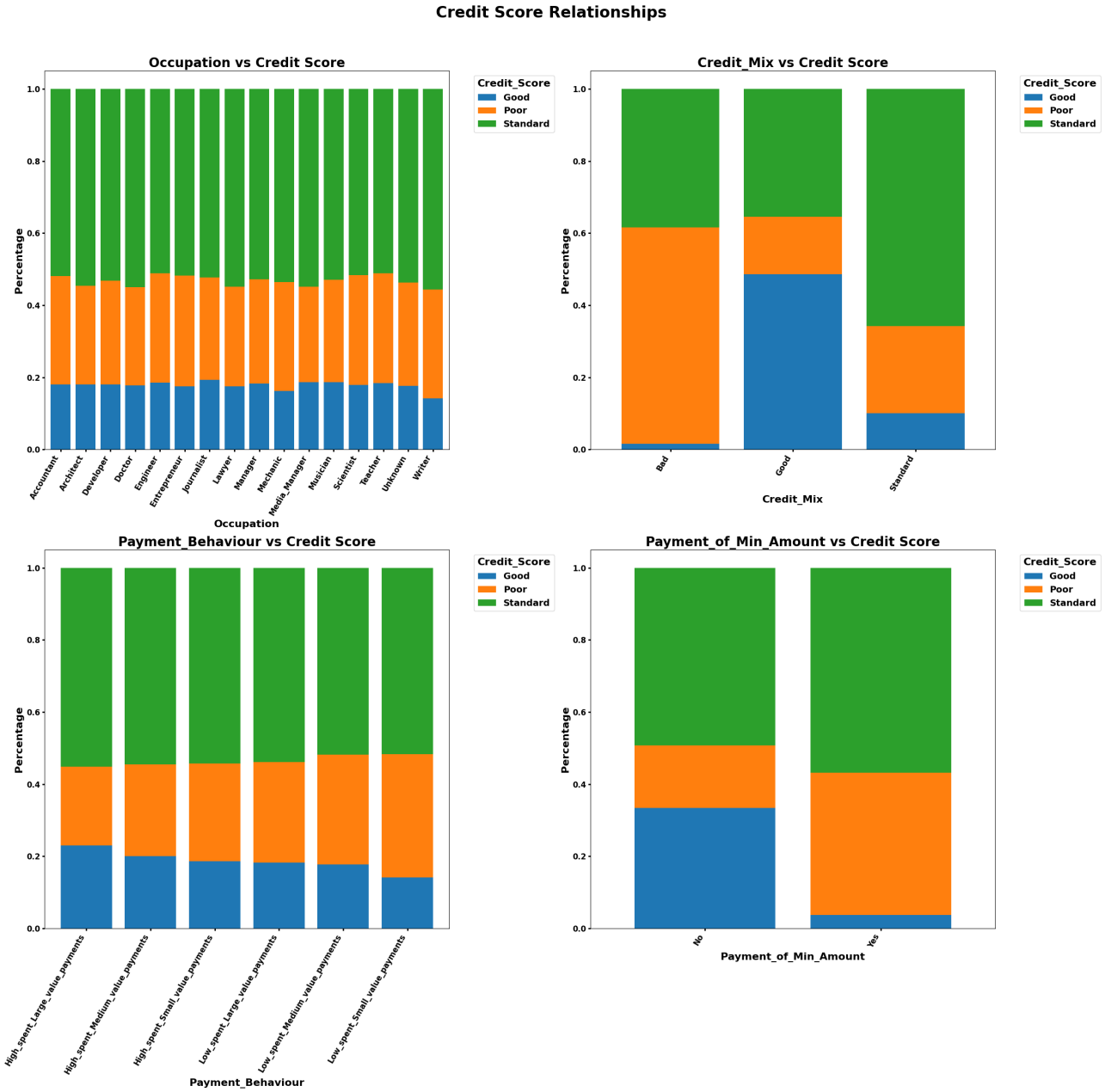

Figure 3.6: Credit Score Relationships with Categorical Features

CHAPTER 4

Feature Engineering

4.1 Feature Creation

The blog [18] outlines eight feature engineering techniques valuable for machine learning. This study aimed to explore feature engineering in the credit scoring, where numerous attributes contribute to the evaluation of credit score. Feature creation was implemented to uncover new factors influencing credit scores. This process involves deriving new features from existing ones using mathematical operations. The focus was on creating two key features:

1. **Debt-to-Income Ratio (DTI)**: This ratio is determined by calculating the sum of all monthly debt obligations and dividing it by the total monthly income before deductions. DTI serves as a key indicator of an individual's financial health, offering insights into their ability to manage monthly payments and take on additional debt.

2. **Credit Utilization Trend**: Calculated by comparing the credit utilization ratio over time. This feature aims to capture the dynamic nature of credit usage, reflecting not just the current state but also the trend in a customer's credit behavior over time.

Figure 4.1 shows two box plots representing the relationship between newly engineered features and credit score categories ("Good," "Standard," and "Poor"). On the left, the DTI shows a slight upward trend in median values as credit scores decline, indicating that individuals with poorer credit scores tend to have higher DTI ratios. On the right, the Credit

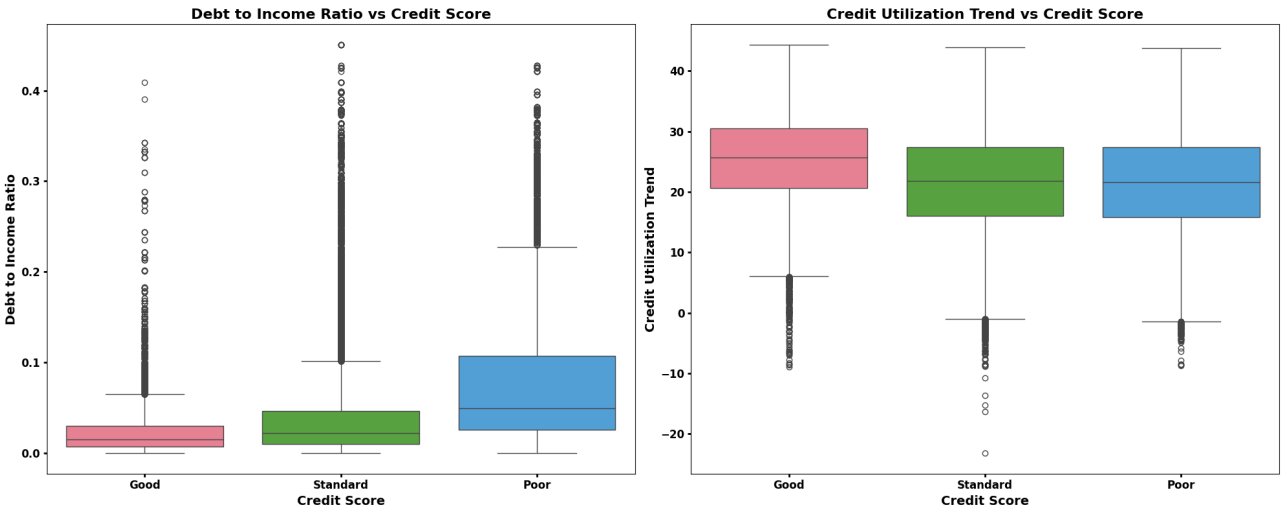

Figure 4.1: Comparison of Newly Created Features Across Credit Score Categories

Utilization Trend displays a more complex relationship, with moderate utilization levels associated with "Good" credit scores, while higher utilization levels are observed in the "Standard" and "Poor" categories. These features did not exhibit strong correlations with credit scores, limiting their potential to improve model performance.

4.2 Data Transformation

In Data Science, encoding is the method of transforming categorical columns into numerical features. The article [19] explains why encoding is crucial. Categorical columns contain text, and if used directly for model building, the model won't perform well and will be unable to process the data effectively. Models can show good and accurate results only with numerical patterns, so data transformation is essential in our study.

4.2.1 Label encoding

Label encoding involves converting categorical values into numerical form by assigning a specific integer to each unique category. This method doesn't create any additional columns in the dataset. The categorical features like *Credit_Score*, *Credit_Mix*, and *Payment_of_Min_Amount* have been chosen from the dataset to undergo data transformation because these features contribute to model performance. Label encoding was selected for these specific features due to their ordinal nature (in the case of *Credit_Score* and *Credit_Mix*) and binary structure

(for *Payment_of_Min_Amount*). This approach preserves the inherent order of categories, maintains dataset dimensionality, and is compatible with various model types, particularly tree-based algorithms. The transformed results displayed in Figure 4.2, where unique values have been mapped for each category. For instance, in *Credit_Score*, 'Good' is mapped to 0, 'Standard' to 1, and 'Poor' to 2, reflecting the ordinal relationship between these categories. This consistent mapping across features helps in model performance and potential prediction accuracy, while transforming the data into a numerical format appropriate for machine learning algorithms.

```
Unique values in Credit_Score: ['Good' 'Standard' 'Poor']
Unique values in Credit_Mix: ['Standard' 'Good' 'Bad']
Unique values in Payment_of_Min_Amount: ['No' 'Yes']
Credit_Score mapping: {'Good': 0, 'Standard': 1, 'Poor': 2}
Credit_Mix mapping: {'Standard': 0, 'Good': 1, 'Bad': 2}
Payment_of_Min_Amount mapping: {'No': 0, 'Yes': 1}
```

Figure 4.2: Label Encoding Results for Categorical Variables

4.2.2 One-hot encoding

One-hot encoding is a technique used to convert categorical data into a format suitable for machine learning algorithms. It works by creating new binary columns for each unique category in the original data. In these new columns, a value of 1 represents the presence of that category for a given data point, while 0 represents its absence. This encoding can be applied to columns with multiple categories. In the dataset, the categorical columns *Occupation* and *Payment_Behaviour* have multiple values. One-hot encoding was applied to these columns, which created new columns in the dataset. Specifically, this process resulted in the creation of 22 new binary columns: 16 for *Occupation* and 6 for *Payment_Behaviour*. Each new column represents a unique category from the original columns, with binary values indicating the presence or absence of that category for each record. This conversion is particularly useful for non-ordinal categorical variables like *Occupation* and *Payment_Behaviour*, as it converts them into a numerical format that machine learning algorithms can process effectively. It preserves all categorical information without implying any ordinal relationships between categories. The resulting encoded features can improve model performance, facilitate feature importance analysis, and enhance model results. However, it's important to note that this transformation can lead to high dimensionality if there are many categories in the original

columns, as evidenced by the addition of 22 new columns from just two original categorical variables in this case

4.3 Standardization

Standardization, also referred to as Z-score normalization, is a crucial preprocessing technique often applied before constructing machine learning models. This method transforms the features in a dataset so that they conform to a specific statistical distribution namely, one with a mean (average) of 0 and a standard deviation of 1. This process can be described mathematically as:

$$z = \frac{x - \mu}{\sigma}$$

Where, z represents the standardized value, x the original value, μ the feature's mean, and σ its standard deviation[20].

4.3.1 Importance of Standardization

Standardization is very important in preparing data for ML models, especially when the variables have different units or scales. Without standardization, features with larger scales can take over the learning process, potentially resulting in skewed or inaccurate predictions. This imbalance can lead to a model that performs poorly or does not generalize well to unseen data. Furthermore, models like KNN, are sensitive to the scale of the data. Standardizing the data verify that all variables responsible equally to the model's learning process, improving both the performance and the speed of convergence in optimization algorithms.

4.3.2 Standardization in the Data Preprocessing Workflow

In this project, standardization was applied after encoding the categorical features into numerical ones. The one-hot encoding process transformed categorical variables into binary vectors, which then became part of the feature set to be standardized. This sequence is important because standardization should be applied to all numerical features, including those generated from categorical encoding. After the one-hot encoding, the standardization

process was performed on both the training and testing datasets. This involved the following steps:

- **Fit the Scaler on Training Data:** The scaling process initially used the training data to compute each feature's average and spread. This approach ensures the model's learning is based solely on training set characteristics, which is important for avoiding unwanted information transfer from the test set.
- **Transform the Training Data:** The scaler, after being calibrated, was applied to the training dataset. This transformation produced a new set of data where each variable is centered around zero and has a unit variance.
- **Transform the Testing Data:** The identical scaling parameters derived from the training set were applied to the test data. This approach maintains consistency across datasets, enabling accurate model assessment.

By standardizing the features after encoding and before model training, we ensured that all features or variables contribute equally to the model, leading to more reliable results.

4.4 Handling Class Imbalance

In the field of machine learning, class imbalance refers to situations where the categories in a dataset have unequal representation. This imbalance can significantly impact model performance, often leading to bias towards the majority class. It can result in skewed predictions, reduced generalization, and misleading performance metrics, especially important in applications like credit scoring. From Figure 3.1, it is demonstrates that the credit score dataset is imbalanced.

The research paper by [21] explored the challenges of credit score prediction in the context of class imbalance, particularly in Peer-to-Peer (P2P) lending platforms. They utilized various balancing techniques to address these challenges and improve predictive accuracy. Their study benchmarked several machine learning models, underscoring the importance of using appropriate sampling strategies to enhance model performance. Building upon this foundation, this research implements a specific balancing technique, namely Random Over-Sampling (ROS), to tackle the imbalance in the dataset. By focusing on ROS, the study

aims to evaluate its effectiveness in improving the model's ability to predict credit scores accurately.

4.4.1 Initial Class Distribution

The original training dataset showed a clear imbalance among the three credit score classes, as can be seen in Table 4.1. Class '1' (Standard) is the majority class with 41,379 samples, while Class '0' (Poor) is the minority with only 13,836 samples. Credit Score 2 was moderately represented with 22,564 instances. This imbalance could results a model that performs better on the majority class but poorly on the minority classes.

Credit Score	Category	Count
1	Standard	41,379
2	Good	22,564
0	Poor	13,836

Table 4.1: Credit Score Distribution in the Imbalanced Dataset

4.4.2 Random Over-Sampling (ROS)

ROS is a technique used to address class imbalance in datasets. It works by randomly duplicating samples from the minority class or classes. This process continues until the number of instances in each class matches that of the majority class, effectively balancing the dataset. Table 4.2 shows the value counts after balancing the dataset using ROS. It effectively balanced the class distribution by increasing the number of instances in the minority classes (Credit Scores 0 and 2) to match the majority class (Credit Score 1). This technique ensures that the model has enough data to learn from each class equally, reducing the likelihood of bias.

Credit Score	Category	Count
1	Standard	41,379
0	Poor	41,379
2	Good	41,379

Table 4.2: Balanced Credit Score Distribution Using ROS

CHAPTER 5

Methodology

5.1 Train-Test Split Methodology

In this study, the dataset was divided into training and testing subsets to support model development and evaluation. Specifically, 80% of the data was used for training, while the remaining 20% was reserved for testing. This 80-20 split is a standard practice, providing the model with enough data to learn while keeping a significant portion for fair evaluation. To maintain consistent class distribution across both training and testing sets, stratification was applied which is an important step given the class imbalance in the data. The use of stratification in the data splitting process ensures that both the training and testing subsets maintain the same class distribution as the original dataset. This approach is essential for preventing any potential bias towards the majority class and guarantees that both subsets accurately reflect the characteristics of the complete dataset. Additionally, a random state value of 42 was set to ensure the split is reproducible, meaning that each time the code is run, the data is divided in the same way. This consistency is crucial for verifying results and comparing model performance under stable conditions.

5.2 Machine Learning Models Employed

In this study, various machine learning models were employed to predict credit scores. Machine learning models refer to algorithms that are specifically created to detect patterns

in a dataset and generate predictions by utilizing those patterns. These models are trained on relevant datasets, allowing them to learn the connections between input features and output labels, which are then used for predicting new, unseen data. The models utilized in this research fall into three main categories: traditional machine learning algorithm, ensemble models, and ANN. Each category offers distinct advantages in addressing the dataset's complexities, enhancing the overall strength of the prediction models. Finally, their performance was compared to determine which model provided the most accurate predictions.

5.2.1 K-Nearest Neighbors (KNN)

The KNN was used as part of our methodology to classify credit scores. KNN is a flexible classifier that belongs to the family of supervised learning models. It works by grouping individual data points based on their similarity to other points using distance formulas. KNN is versatile and can be applied to both classification and regression tasks, making it a simple yet widely used model for credit scoring [7].

The KNN is based on the principle that data points with similar characteristics or patterns tend to be close to each other in feature space. In this credit score prediction project, 5 data points are chosen as the nearest neighbors. Each data point represents an individual's financial profile, including features like income, credit utilization, payment history, and credit mix. The KNN calculates the distance between these profiles and a new individual's profile to find the nearest neighbors. It then predicts the new individual's credit score category (good, bad, or standard) based on the majority class of these neighbors. This process begins by calculating the distance between an unseen data point and the points already present in the dataset. While various distance metrics are available, the most commonly applied is the Euclidean distance. The Euclidean distance formula used in KNN is:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2} \tag{5.2.1}$$

Where, $d(x, y)$ denotes the Euclidean distance between two points, x and y. In this equation, x_i and y_i represent the values of the i-th feature for points x and y, while n refers to the total number of features. This equation calculates the most direct linear path between two points in a space with multiple dimensions

The Figure 5.1 illustrates the KNN algorithm. It shows two classes of training data (blue triangles and purple squares) and a test sample (red circle). The solid and dashed circles represent decision boundaries for k =1 and k=3, respectively, demonstrating how the algorithm selects neighbors to classify the test point based on proximity.

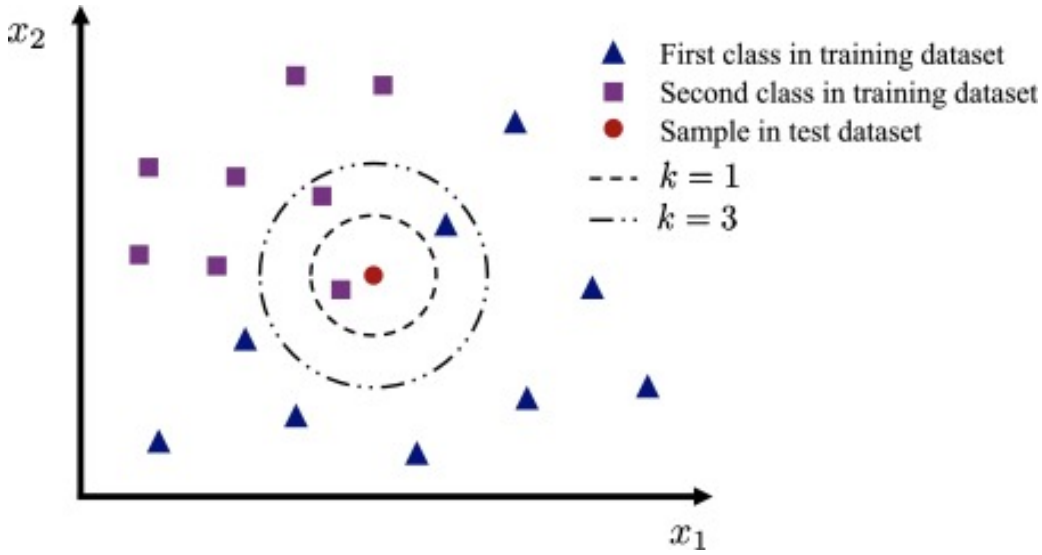

Figure 5.1: KNN Classification with Decision Boundaries [22]

Justification

KNN was selected for credit score classification due to its strengths in multi-class problems, aligning well with our dataset's multiple credit score categories. KNN's non-parametric nature allows it to capture complex relationships in credit-related data without distribution assumptions. Its interpretability is valuable in credit risk assessment, where decision clarity is often required. Although sensitive to class imbalance, KNN showed significant improvement when the dataset was balanced using ROS. This effectiveness, combined with KNN's adaptability to new patterns and computational efficiency during predictions, makes it an ideal choice for our credit scoring task.

5.3 Ensemble Learning Model

Ensemble learning models train multiple models on different subsets of the data, which are then combined to give better predictive results. These models also combine weak learners to produce good outcomes. Ensemble learning aims to reduce bias, variance, and overfitting,

as well as improve overall performance. In our experiment of ensemble learning model, Bagging and Extra Trees are used to predict credit score.

5.3.1 Bagging (Bootstrap Aggregating)

Bagging is a powerful ensemble method that works by creating diverse subsets of the initial dataset. It does this through a process called random sampling with replacement. This means that data points can be selected more than once for each subset, allowing for variability in the training data used for individual models within the ensemble. These subsets are called bootstrap samples, and each sample is the exact size as the original dataset. Bagging works by training separate models (classifiers) on each of these bootstrap samples [23]. The outputs of the individual models are then combined to make a final prediction through majority voting, which is commonly used for classification tasks, as depicted in Figure 5.2. In our credit score dataset, a Decision Tree classifier is used as the base model for each bootstrap sample. It is chosen as the estimator because it tends to have high variance, which benefits from the bagging process. In our implementation, an ensemble of 100 Decision Tree classifiers is created, each trained on a different bootstrap sample of the original credit score dataset. This specific number of estimators balances computational efficiency with the benefits of ensemble learning. It performed well in predicting the credit score and showed extremely good results when the dataset was balanced using ROS.

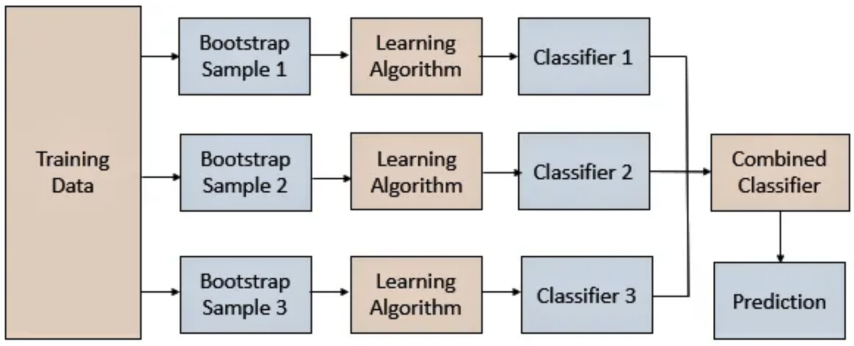

Figure 5.2: Bagging Process in Model Training [24]

Justification

Bagging was chosen for credit score prediction due to its ability to improve model stability and reduce overfitting, particularly with high-variance models like Decision Trees. By training multiple models on different bootstrap samples and aggregating their outputs, bagging enhances the accuracy and robustness of predictions. This is especially important for noisy or imbalanced credit score datasets. In our study, bagging performed exceptionally well, especially when paired with ROS, making it an effective approach for accurate credit score prediction.

5.3.2 Extra Trees

Extra Trees, also called Extremely Randomized Trees, is a group learning technique that builds upon the concept of decision trees[25]. Decision trees operate by repeatedly dividing the dataset into smaller groups. These divisions are based on feature values that provide the most useful information or lead to the least mixed results. Each internal node represents a decision based on a feature, and the leaves represent the final prediction or class. It is also similar to the Random Forest algorithm, but they do have some differences, making it more random. Extra Trees introduces two main randomizations:

- **Random Feature Subset:** Like Random Forests, Extra Trees randomly select a subset of features at each split.
- **Random Splitting Points:** Extra Trees further randomize the splitting by choosing the splitting thresholds randomly from the feature values, instead of trying to find the optimal threshold as in traditional decision trees or Random Forests.

Figure 5.3 emphasizes the ensemble nature of Extra Trees, with multiple trees working together, making individual predictions, and finally producing a robust aggregate prediction. To predict credit scores, this model will take input data such as payment behaviour, credit mix, number of loans, etc., and predict a credit score class: good, standard, or bad credit. The Extra Trees Classifier generates 100 decision trees, each using distinct subsets of the dataset for training and producing independent predictions. The ultimate prediction is determined by combining the outputs from all these trees. This method resulted in improved prediction accuracy.

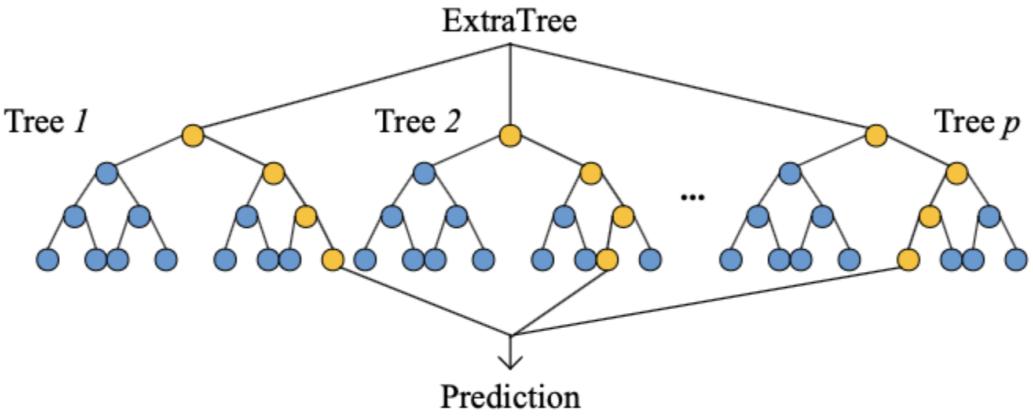

Figure 5.3: Extra Trees for Ensemble Prediction[25]

Justification

Extra Trees was selected for this study due to its impressive accuracy in credit score classification. The algorithm's additional randomness generates more diverse decision trees, reducing overfitting and enhancing the model's capacity to generalize the unseen data. The research by [26] showed Extra Trees outperformed other ensemble models in similar tasks, further supporting our choice. These characteristics make Extra Trees particularly appropriate for the complex and dynamic nature of credit score prediction, where capturing subtle patterns in financial data is essential.

5.4 Artificial Neural Network (ANN)

The ANN is a key concept in ML and AI. It works like biological neural networks, ANNs are computational models used to recognize patterns, classify data, and solve complex tasks such as image recognition, Text analytics, and control of autonomous systems. In this study, a MLP, a type of ANN, is implemented to classify credit scores.

5.4.1 Multilayer Perceptron (MLP)

A MLP is a form of feedforward ANN. Its structure includes at least three layers: an input layer, one, two or more hidden layers, and an output layer. MLPs employ backpropagation, a type of supervised learning technique, for network training. In this architecture, neurons in each layer are fully linked to those in neighboring layers. The key strength of MLPs lies in

their capacity to model intricate functions by learning weights and biases from training data [27].

Figure 5.4 illustrates the process of MLP:

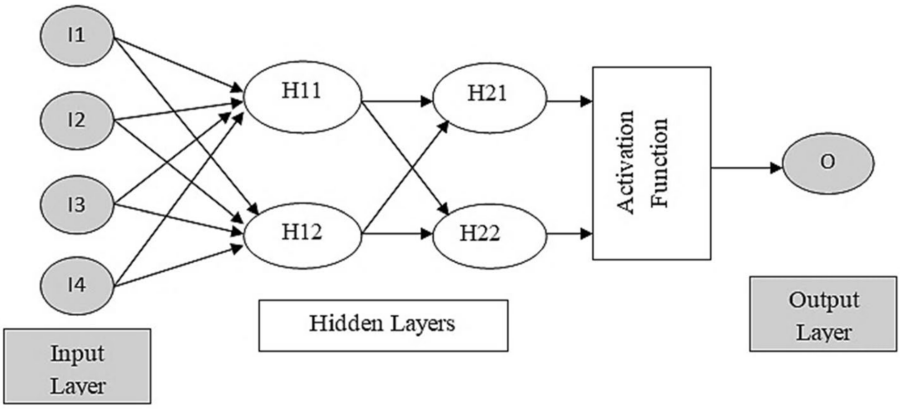

Figure 5.4: Structure of a MLP Model [27]

The MLP process can be broken down into the following components:

- **Input Layer:** The process begins at the input layer (I1-I4) as shown in figure 5.4, where each node or neuron represents a feature of the input data.
- **Hidden Layers:** In a MLP, the hidden layers work to analyze the input information by identifying patterns and connections that aren't readily visible in the original data. Each hidden layer applies transformations (through weighted connections and activation functions) to detect complex features, helping the network understand and model non-linear patterns in the data. Figure 5.4 shows two hidden layers, whose outputs are computed using the following equations [28]:

  The output of the first hidden layer(H11, H12) is computed as:

$$\mathbf{z}_1 = f(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \tag{5.4.1}$$

  The vector $\mathbf{z}_1$ represents the output from the initial hidden layer, which contains neurons H11 and H12. $\mathbf{W}_1$ denotes the matrix of weights for this first hidden layer. The input data is represented by the vector $\mathbf{x}$. For the first hidden layer, $\mathbf{b}_1$ is the vector of biases. Finally, f signifies the activation function used in the network.

  The output of the second hidden layer(H21, H22) is computed as:

$$\mathbf{z}_2 = f(\mathbf{W}_2\mathbf{z}_1 + \mathbf{b}_2)$$

Where the vector $\mathbf{z}_2$ represents the output from the second hidden layer, which contains neurons H21 and H22. $\mathbf{W}_2$ denotes the matrix of weights for this second hidden layer. The input to this layer, $\mathbf{z}_1$, is the output from the preceding hidden layer. For the second hidden layer, $\mathbf{b}_2$ is the vector of biases.

- **Activation Function:** This applies a non-linear transformation to the output of the neurons in each layer. The sigmoid function is used as activation function, and is calculated as [29]:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5.4.2}$$

where $\sigma(x)$ is the output of the sigmoid function and e^{-x} is the exponential function applied to the negative of the input x.

- **Output Layer:** This is the last layer that generates the prediction or output after applying the activation function. The final output is computed as:

$$\mathbf{y} = f(\mathbf{W}_3\mathbf{z}_2 + \mathbf{b}_3) \tag{5.4.3}$$

Where the vector $\mathbf{y}$ represents the final output (O) of the network. $\mathbf{W}_3$ denotes the matrix of weights for the output layer. The input to this layer, $\mathbf{z}_2$, is the output from the second hidden layer. For the output layer, $\mathbf{b}_3$ is the bias. The activation function is represented by f, which may differ from the activation functions used in the hidden layers.

The MLP's learning process involves fine-tuning the weights and biases using backpropagation. This method aims to reduce the discrepancy between the network's predictions and the actual outputs in the training dataset. For predicting credit scores using the MLP process described above, implemented a specific configuration of the MLPClassifier from the scikit-learn library. Our model architecture includes one hidden layer with 100 neurons, which provides a balance between model complexity and computational efficiency. The number of iterations for training was set to 1000, allowing sufficient opportunity for the model to converge while preventing overfitting.

Justification

As artificial neural networks work very well when it comes to image processing, this study wanted to explore particularly how ANN performs for classification tasks. MLP was chosen

for classification because it has several advantages. It can handle high-dimensional data, which is common in credit scoring scenarios. The adaptability of MLP is important as credit data tends to change often, making this model a good choice. The hidden layers in MLP can learn and represent interactions between different input features, which is valuable for understanding complex relationships in credit data. Additionally, the scalability of MLP allows it to handle large volumes of credit applications. These characteristics make MLP a promising approach for credit score classification tasks.

5.5 Hyperparameters

Hyperparameters are configuration settings for machine learning algorithms that are determined before training begins and can't be directly derived from the data. They govern various aspects of the model's learning and prediction processes. These settings include factors that shape the training process, like the number of neighbors in KNN or the architecture of an ANN. Unlike model parameters, which the algorithm learns during training, hyperparameters are set externally. They play a crucial role in determining the model's performance. Optimizing these hyperparameters is key to enhancing the model's accuracy and overall efficiency. This optimization is typically done through methods such as grid search or random search, which explore different hyperparameter combinations to identify the optimal setup for the specific task at hand.

5.5.1 KNN Hyperparameters

In KNN hyperparameter modeling, the algorithm considers nearby points (neighbors) typically set at 3, 5, 7, and 9. Using too few neighbors can lead to overfitting, while using too many can result in underfitting. To handle the fitting issue, the weights parameter is used. It can either treat all neighbors equally (uniform) or give more importance to closer neighbors (distance). This parameter controls how the neighbors contribute to the final prediction. The distance between points is measured by different metrics, such as Euclidean, Manhattan, or Minkowski, and the choice of metric can impact how the model interprets proximity between data points [30]. These hyperparameters allow for fine-tuning how the KNN algorithm finds neighbors, weights their influence, and measures their distance, which collectively impact model accuracy. By executing this approach, KNN showed improved results compared to

before hyperparameter tuning.

5.5.2 MLP Hyperparameters

To improve prediction accuracy, various hyperparameters are configured for the MLP[31]. One of the key hyperparameters is the size and structure of the hidden layers, which determines the number of neurons and layers in the network. For example, a configuration with 50 neurons represents one hidden layer, while configurations such as 100, 50 indicate two hidden layers with 100 and 50 neurons respectively. A setup of 100, 100, 50 represents three hidden layers. These configurations influence how well the network can learn complex patterns in the data.

Activation Parameters

The activation function introduces non-linearity to the model, allowing it to capture more complex relationships. Common activation functions used in MLP include identity (where no activation is applied), ReLU (Rectified Linear Unit), which outputs 0 for negative inputs and the input value itself for positive inputs, sigmoid, which maps outputs to a range between 0 and 1, and tanh, which maps outputs to a range between -1 and 1. Another important hyperparameter is the solver algorithm, which is used for optimizing the model's weights. Stochastic Gradient Descent (SGD) updates weights iteratively based on random samples of data, Adam is an adaptive learning rate optimizer that combines momentum and RMSprop, and LBFGS is a quasi-Newton method suitable for smaller datasets. These are the parameters included for activation functions as part of hyperparameter tuning.

Learning rate Parameters

The learning rate, which calculates how quickly or slowly the model updates weights during training, is another important parameter. In our analysis, we implemented a constant learning rate, which maintains a fixed value throughout the training process. This approach contrasts with alternative strategies such as 'invscaling', which gradually decreases the learning rate over time, and 'adaptive', which maintains a steady learning rate as long as the training loss continues to decrease but reduces it when improvement slows or stops. The choice of learning rate strategy notably impacts the model's convergence speed and its ability to find

optimal solutions.

Regularization parameter

The regularization parameter (alpha) is used to help prevent overfitting by constraining large weights, using values 0.0001, 0.001, 0.01, and 0.1. Additionally, early stopping, which is also utilized, the training process can be stopped if the model's performance on a validation dataset ceases to show improvement. The batch size, set to auto, 64, 128, or 256, defines how many training samples are processed at once and further affects the model's training efficiency.

Carefully tuning these hyperparameters is essential for balancing overfitting and underfitting, ensuring the model generalizes well to unseen data. These extensive parameters helped the credit score model achieve impressive accuracy.

5.6 Method Evaluation

Model evaluation refers to the practice of measuring a machine learning model's effectiveness in performing its designated task. The use of evaluation metrics is a fundamental aspect of the development and validation process for models. They are used to choose the best model, enable comparisons, and help decide which models need further refinement through hyperparameter tuning. This evaluation is carried out using different metrics [32]. The evaluation metrics employed for this research include:

Confusion Matrix

The confusion matrix is kind of table that explains the performance of a model and is widely used for classification tasks. The credit score dataset has three target classes (as shown in Figure 3.1). Usually, for binary classification problems, a 2x2 matrix is used to compute the confusion matrix. However, in our case, as the dataset has 3 classes, we implemented a 3x3 grid confusion matrix. Table 5.1 portrays the calculation for this multiclass classification problem. In the matrix, each row represents the true class based on the observed data, while each column shows the classes predicted by the model.

- **TP (True Positive)**: These are correctly identified instances where the predicted class

exactly matches the actual class. The diagonal cells running from the upper left to the lower right of the matrix contain the True Positives. These indicate the number of instances for each class that the model accurately predicted.

- **FP (False Positive)**: These are cases where the model incorrectly predicts a specific class, whereas the actual class is different. False Positives are located in the off-diagonal cells of each column. These are typically referred to as misclassifications.
- **FN (False Negative)**: These are instances where the model fails to predict the correct class for an actual instance of that class. False Negatives are located in the off-diagonal cells of each row. The FN also called misclassified entries

TN (True Negatives) aren't used in multiclass confusion matrices because each class is considered individually, with all others treated as "negative" and TN are implicitly represented in the confusion matrix's off diagonal elements.

	Predicted Class 0	**Predicted Class 1**	**Predicted Class 2**
Actual Class 0	TP	FP	FP
Actual Class 1	FN	TP	FP
Actual Class 2	FN	FN	TP

Table 5.1: Confusion Matrix Table for a Multiclass Classification Problem

Accuracy

In multiclass classification, similar to binary classification, accuracy serves as a metric to evaluate a model's performance in correctly labeling new data. It represents the ratio of accurate predictions to the total number of predictions made. This encompasses both correctly identified positive cases (TP) and correctly identified negative cases (TN) across all classes. The mathematical representation of this concept is typically presented in a formula, which you've referred to as equation 5.6.1.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{5.6.1}$$

Precision

Precision measures how accurate a model's positive predictions are for each individual class. It is calculated by determining what fraction of the observations predicted to be positive for a class are actually positive. The typical mathematical expression for precision is:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.6.2}$$

In multiclass classification, precision needs to be computed for each class individually. It can then be averaged to find the mean precision, which gives an overall indication of the model's accuracy in predicting positive classes correctly. In this study, we use Weighted-average Precision to calculate the overall precision for the dataset.

Recall

Recall, which is also referred to as sensitivity, quantifies the model's ability to identify all actual positive instances. This metric is crucial in evaluating the performance of machine learning classification models. Recall is mathematically defined as:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.6.3}$$

As our study focuses on a multiclass classification setting, recall must be considered for each class separately and then aggregated to provide a holistic measure across the entire model. For multiclass classification, we employed a Weighted-Average Recall.

F1-Score

The F1-score is a metric commonly employed in classification tasks, especially when dealing with imbalanced datasets. It's particularly valuable in scenarios where striking a balance between precision and recall is crucial. This score represents the harmonic mean of precision and recall, offering a unified measure that accounts for both false positives and false negatives. The mathematical expression for calculating the F1-score is:

$$\text{F1 Score} = 2 \times \left(\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \right) \tag{5.6.4}$$

For multiclass classification problems, the Weighted-average F1-Score is implemented to calculate the overall F1-score.

AUROC Score

The Area Under the Receiver Operating Characteristic Curve, abbreviated as AUROC, is a metric used to evaluate classification model performance across different threshold levels. This measure quantifies how well a model can distinguish between classes. The ROC curve is constructed by plotting the True Positive Rate (TPR) in relation to the False Positive Rate (FPR) at various classification thresholds. AUROC represents the total area beneath this curve, offering a comprehensive assessment of model performance irrespective of the chosen threshold. AUROC values fall between 0 and 1. A score of 0.5 indicates that the model's predictions are no better than random chance, while a score of 1.0 signifies perfect classification accuracy, with 100% sensitivity and 100% specificity. Thus, AUROC provides insight into a model's overall ability to separate classes across all possible decision thresholds. For multiclass problems, AUROC can be calculated using a one-vs-rest approach, where each class is considered against all others combined. The weighted average AUROC is utilized to calculate the metric for each class individually and then compute a weighted mean. This strategy provides a more balanced evaluation of the model's performance across all classes, especially useful in multiclass classification problems or when dealing with imbalanced datasets.

Training and Testing Time

To assess the models' efficiency, measured the duration of both the training and testing phases. For the training phase, we recorded the time from the initiation of the learning process until its conclusion. This measurement encompasses the entire period during which the model processes the training data and fine-tunes its parameters. Similarly, for the testing phase, we timed the process from the moment prediction begins on the test data until it finishes. This includes the total time needed for the model to generate predictions for all test set instances[9]. These timing measurements offer very useful information into each model's computational efficiency, enabling a thorough evaluation of their performance in both learning and prediction tasks. By examining these metrics, can better understand the practical implications of deploying each model in real-world scenarios.

K-fold cross-validation

K-fold cross-validation is a technique used in machine learning to assess a model's effectiveness and ability to generalize. The method involves dividing the dataset into k equal parts or "folds." In this case, 10-fold cross-validation is employed, splitting the training data into 10 equal sections[33]. The process consists of 10 rounds, each depicted as a row in the referenced figure5.5. For each round, one fold (highlighted in blue) serves as the test set, while the other nine (in white) form the training set. The test fold shifts with each iteration, starting from the rightmost position and moving left until it reaches the leftmost spot in the final round. In every iteration, the model trains on the nine white folds and is evaluated on the blue fold, yielding a performance metric (E1 through E10) for each round. After completing all 10 iterations, the overall performance E is calculated by averaging the individual metrics, as shown in the formula in the figure 5.5. This approach ensures that each data point appears in the test set once and in the training set nine times. The method provides a strong evaluation of the model's performance across various data subsets, helping to mitigate overfitting and offering a more accurate estimate of how the model might handle new, unseen data.

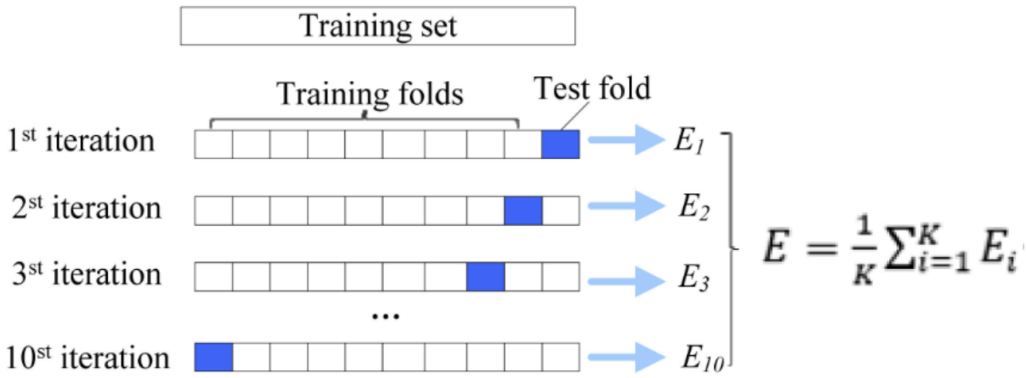

Figure 5.5: K-Fold Cross-Validation Process [34]

CHAPTER 6

Results and Discussion

This chapter presents the key findings and outcomes of the research methodology. It examines the results obtained from the experiments and analyses, providing a detailed review of the data. The central focus of this discussion is the comparison of results between imbalanced and balanced datasets. Through this comparison, the chapter explores the significance of these results, their effects for the field, and how they relate to the initial research questions and hypotheses.

6.1 Results and Comparisons of Models on Imbalanced Dataset

This section outline the results of evaluating four machine learning models—Bagging, Extra-Trees, KNN, and MLP—on an imbalanced dataset. The evaluation compares their classification performance and computational efficiency across several key metrics.

Table 6.1 summarizes the performance of various models, with each row representing a model and the columns displaying metrics such as Accuracy, Precision, Recall, F1-Score, Training Time, Testing Time, and AUROC Score. These metrics provide insights into both the predictive performance and computational efficiency of each model. Accuracy, Precision, Recall, and F1-Score assess how well the models classify the data, while the Training and Testing Times reflect their efficiency in terms of resource consumption. The AUROC Score, particularly relevant for imbalanced datasets, indicates the model's ability to differentiate between classes. By examining this table, it is possible to compare the models' strengths and

weaknesses, leading to an informed evaluation of their suitability for the task.

Model	Accuracy	Precision	Recall	F1-Score	Training Time (s)	Testing Time (s)	AUROC Score
Bagging	0.78	0.78	0.78	0.78	225.32	1.40	0.89
ExtraTrees	0.78	0.78	0.78	0.78	20.70	0.69	0.89
KNN	0.66	0.66	0.66	0.66	0.03	13.23	0.78
MLP	0.69	0.69	0.69	0.69	91.89	0.02	0.83

Table 6.1: Model Evaluation Results on Imbalanced Dataset

Performance Analysis for Imbalanced Dataset: Table 6.1 presents a comprehensive overview of model performance on the imbalanced dataset. Bagging and ExtraTrees demonstrated strong classification capabilities, both achieving identical scores of 0.78 for Accuracy, Precision, Recall, and F1-Score. These identical scores, resulting from weighted averages in multiclass classification, indicate a balance between precision and recall. However, it is essential to recognize that individual class performance may vary, as detailed in Appendix 8. In contrast, KNN and MLP displayed lower performance, scoring 0.66 and 0.69, respectively, across these metrics. Regarding computational efficiency, notable differences emerged: Bagging required the longest training time (225.32 seconds) with moderate testing time (1.40 seconds), while ExtraTrees had a more balanced profile with faster training (20.70 seconds) and efficient testing (0.69 seconds). KNN had the shortest training time (0.03 seconds) but the longest testing time (13.23 seconds), which could limit its efficiency for large datasets. MLP showed moderate training time (91.89 seconds) and the fastest testing time (0.02 seconds). The AUROC scores offer valuable insights into the models' discriminative abilities, as shown in Table 6.1, with Bagging and ExtraTrees achieving the highest scores of 0.89, MLP scoring 0.83, and KNN 0.78. Figure 6.1 visualizes these scores through ROC curves, providing an intuitive understanding of each model's performance across different classification thresholds. The figure shows curves for all models, with the x-axis representing the False Positive Rate and the y-axis the True Positive Rate. The curves for Bagging and ExtraTrees (blue and orange lines, respectively) are closest to the top-left corner, confirming their superior discrimination capabilities, even in an imbalanced dataset. MLP's curve shows moderate performance, while KNN's more gradually rising curve (green line) reflects its lower AUROC

score, suggesting less effective classification, particularly for minority classes. The dashed diagonal line represents random guessing (AUROC = 0.5), and all model curves above this line confirm their improved performance over random classification. Together, the table and figure provide a detailed view of model performance and classification abilities across various thresholds.

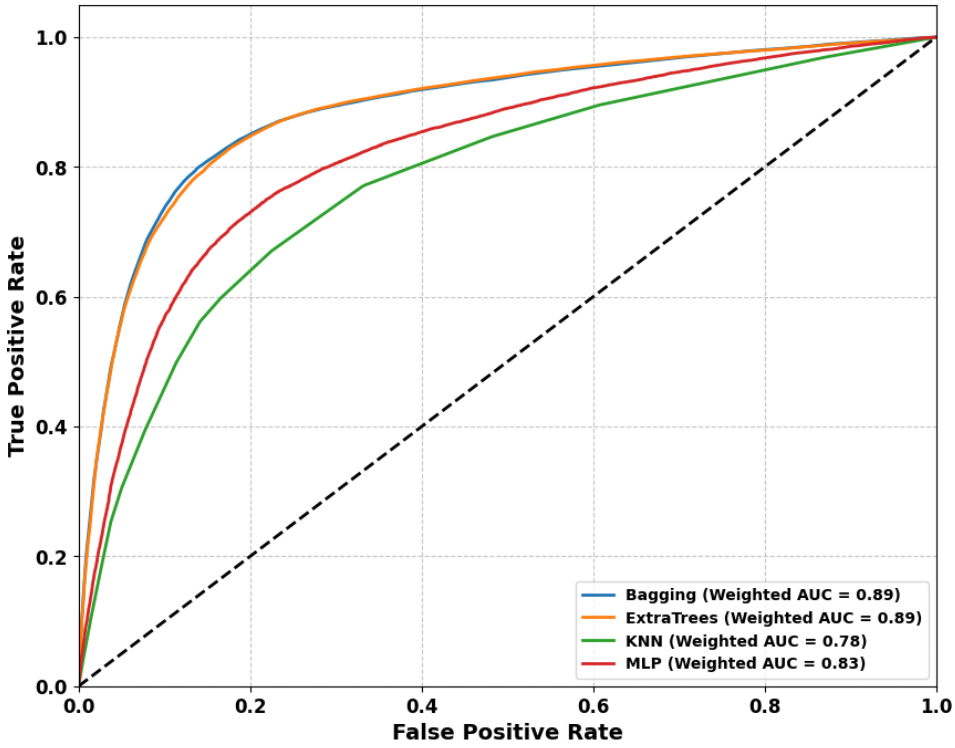

Figure 6.1: ROC Curve for Imbalanced Dataset

Performance of Extra Trees and Bagging on Imbalanced Dataset: Bagging and ExtraTrees outperformed the other models, achieving identical scores across most metrics, including an AUROC of 0.89. As mentioned in the methodology 5, their strong performance is due to their ensemble nature, where multiple decision trees are combined to make more accurate predictions. This approach allows both models to balance classifications across the majority and minority classes more effectively. Aggregating multiple trees helps reduce overfitting and makes the models more resilient to the bias introduced by the majority class. Extra Trees further improves performance by introducing additional randomness in the feature selection and tree splits, making the model more generalizable and efficient.

Challenges for KNN and MLP on Imbalanced Dataset: KNN and MLP did not perform as well as Bagging and ExtraTrees, with KNN achieving an AUROC of 0.78 and MLP scoring higher at 0.83. KNN struggled primarily because it relies on nearest neighbors to classify data points, making it vulnerable to class imbalance. In an imbalanced dataset, the majority class tends to overwhelm the nearest neighbors, resulting in biased predictions. Additionally, KNN's high testing time indicates inefficiency when handling large datasets. While MLP performed better than KNN, it also faced challenges due to the complexity of tuning neural networks. Although MLP can model non-linear relationships, it requires significant fine-tuning, and its performance can degrade without sufficient data or proper regularization. MLP's moderate performance suggests it did not generalize as effectively as Bagging and ExtraTrees on this imbalanced dataset.

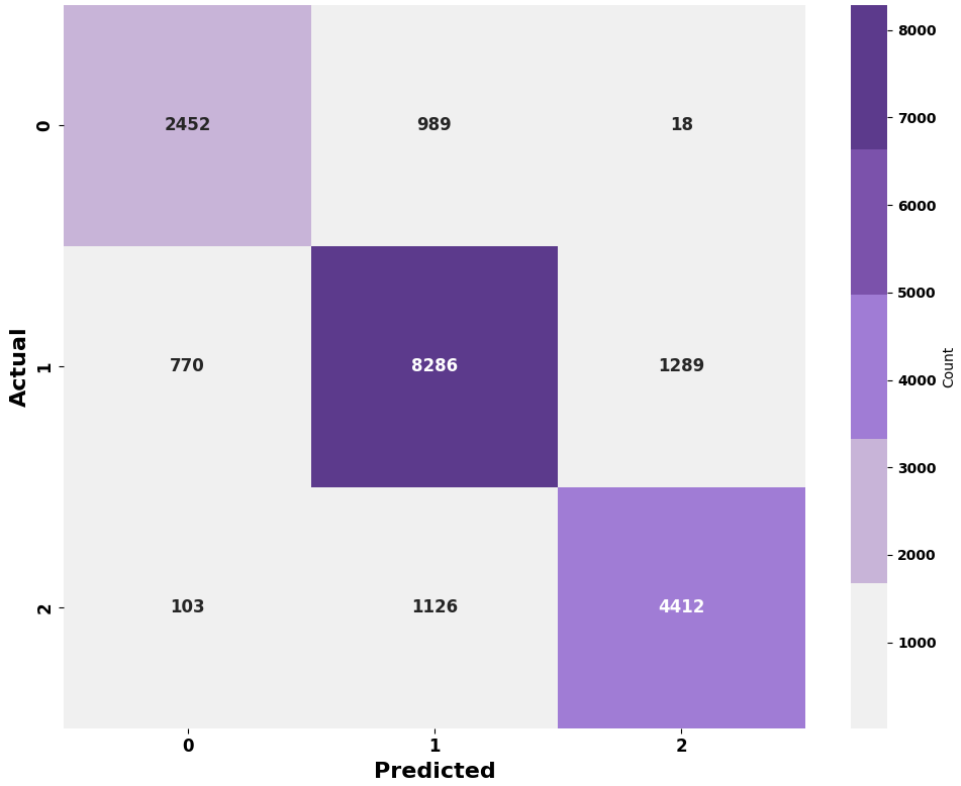

Figure 6.2: Confusion Matrix for Extra Trees on Imbalanced Dataset

Best Performing Confusion Matrix for Imbalanced Dataset: The ExtraTrees model emerged as the best-performing model on the imbalanced dataset, as illustrated by its confusion matrix in Figure 6.2. This matrix visually compares the model's classification performance, with the predicted classes on the x-axis and the actual classes on the y-axis. ExtraTrees correctly

classified 15,150 out of 19,445 credit scores, accurately identifying 2,452 poor credit cases (Class 0), 8,286 standard credit cases (Class 1), and 4,412 good credit cases (Class 2). The off-diagonal cells highlight areas for improvement, with 989 instances of Class 0 misclassified as Class 1 and 103 instances of Class 2 misclassified as Class 0. The distribution of classifications suggests that the ExtraTrees model performs best at identifying Class 1 (standard credit), followed by moderate performance for Class 2 (good credit), and the least effective performance for Class 0 (poor credit), possibly due to the imbalance in the dataset.

6.2 Results and Comparisons of Models on Balanced Dataset(ROS)

Following the assessment of the imbalanced dataset, the dataset was balanced using ROS to address the class imbalance and ensure that each class was equally represented. This balancing technique helps improve the models' ability to handle minority classes, which often leads to better overall performance, particularly in classification tasks. After balancing the dataset, the models were re-evaluated to observe how their performance changed with more equal class distribution. The outcomes of these evaluations are presented in Table 6.2 below:

Model	Accuracy	Precision	Recall	F1-Score	Training Time (s)	Testing Time (s)	AUROC Score
Bagging	0.90	0.90	0.90	0.90	272.11	2.01	0.98
ExtraTrees	0.91	0.91	0.91	0.90	28.51	1.11	0.99
KNN	0.74	0.74	0.74	0.73	0.05	37.07	0.88
MLP	0.74	0.74	0.74	0.73	337.33	0.08	0.87

Table 6.2: Model Evaluation Results on Balanced Dataset (ROS)

Performance Analysis on Balanced Dataset: The evaluation results of the models on the balanced dataset, as shown in Table 6.2, provide key insights into their effectiveness across various metrics. ExtraTrees emerged as the top performer, achieving the highest scores in accuracy, precision, and recall (0.91), with an F1-score of 0.90. Bagging followed closely, consistently scoring 0.90 across all metrics, indicating stable and consistent performance. KNN and MLP showed lower performance, with both models achieving 0.74 in overall

classification metrics and an F1-score of 0.73. The similarity between model results can be attributed to the use of weighted averages in multiclass classification, ensuring a balanced evaluation of all classes. When it comes to computational efficiency, the models exhibited diverse characteristics. MLP had the longest training time (337.33 seconds) but the shortest testing time (0.08 seconds), indicating a trade-off between training duration and quick prediction ability. Bagging, while highly accurate, also required substantial training time (272.11 seconds) and had a moderate testing time (2.01 seconds). ExtraTrees, on the other hand, combined high performance with efficiency, showing fast training (28.51 seconds) and quick testing (1.11 seconds). KNN showed the fastest training time (0.05 seconds) but had the longest testing time (37.07 seconds), reflecting its reliance on instance-based learning, which increases computation during testing.

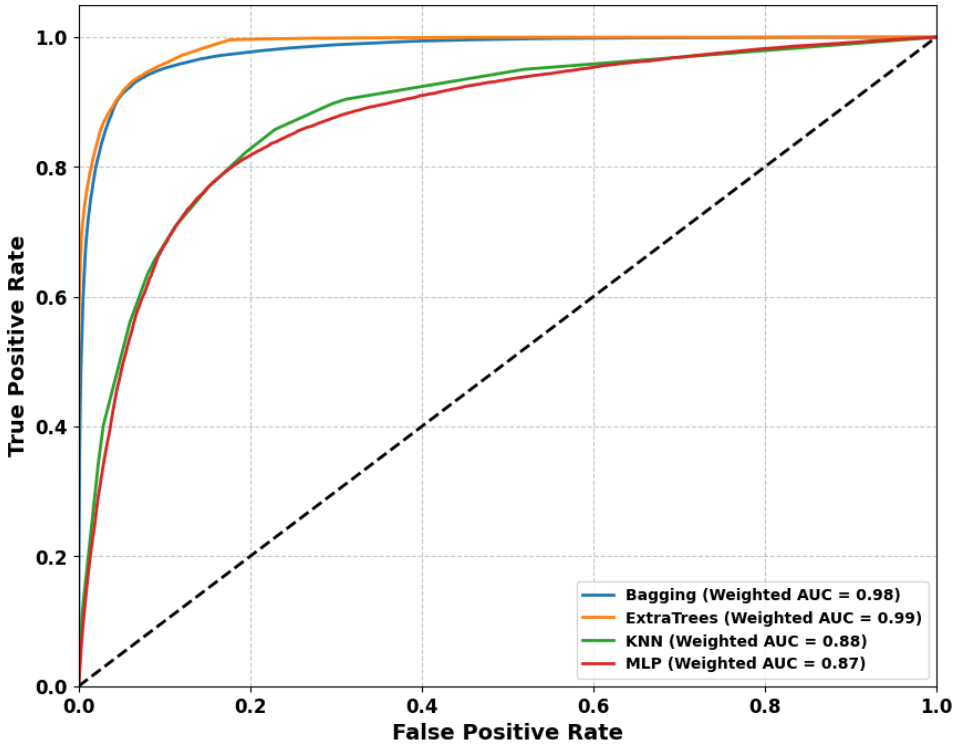

Figure 6.3: ROC Curve for Balanced Dataset(ROS)

The AUROC scores, as presented in Table 6.2 and visualized in Figure 6.3, further support these findings. ExtraTrees achieved the highest AUROC score of 0.99, with its ROC curve approaching the top-left corner, indicating excellent class discrimination. Bagging closely followed with an AUROC of 0.98. Both tree-based models maintained strong class differentiation even after dataset balancing. KNN and MLP showed lower AUROC scores, 0.88 and 0.87

respectively, with With a slow rise in the ROC curve, indicating less effective classification. All models improved compared to their performance on the imbalanced dataset, especially in handling minority classes, with the tree-based models continuing to excel in accuracy and class distinction.

Performance of Extra Trees and Bagging on Balanced Dataset: On the balanced dataset, ExtraTrees and Bagging resulted outstanding performance, with accuracy, precision, recall, and F1-scores all reaching around 0.90 or higher. The balanced distribution of classes enabled both models to leverage their ensemble structures effectively, capturing relationships between features without overfitting. As a result, ExtraTrees achieved an AUROC of 0.99, while Bagging followed closely with 0.98. Their ability to handle all classes equally, their design of aggregating distinct decision trees, allowed them to deliver powerful and consistent predictions across the board.

Challenges for KNN and MLP on Balanced Dataset: KNN and MLP showed moderate improvement on the balanced dataset but still lagged behind the tree-based models. KNN achieved an AUROC score of 0.88, but its dependence on proximity-based classification limited its ability to capture complex patterns in the data. While the balanced distribution reduced bias toward the majority class, KNN's nearest-neighbor approach was less effective in fully distinguishing between classes. MLP, with an AUROC score of 0.87, also saw slight gains but remained constrained by the need for extensive fine-tuning and hyperparameter optimization. Even with the more favorable class distribution, MLP's performance was hindered by its sensitivity to network architecture and regularization. In comparison to the ensemble methods, KNN and MLP struggled to generalize as effectively, even with the balanced dataset.

Best Performing Confusion Matrix for Balanced Dataset: The confusion matrix for the ExtraTrees model on the balanced dataset, shown in Figure 6.4, reflects its status as the best-performing model. The matrix reveals that the model correctly classified 28,088 out of 31,035 instances, with strong performance across all categories: 10,257 poor credit cases, 8,107 standard credit cases, and 9,724 good credit cases. Misclassifications occurred in 924 cases where standard credit cases were predicted as poor credit, and in 65 cases where good credit cases were predicted as poor credit. Additionally, 1,314 standard credit cases were

misclassified as good credit, and 556 good credit cases were misclassified as standard credit. While the model excelled at identifying both poor and good credit cases, its performance was slightly less accurate for standard credit cases. This may suggest a marginal bias toward the more extreme categories, but the model's overall effectiveness in distributing correct predictions across all classes illustrates that the class balance issue was well addressed.

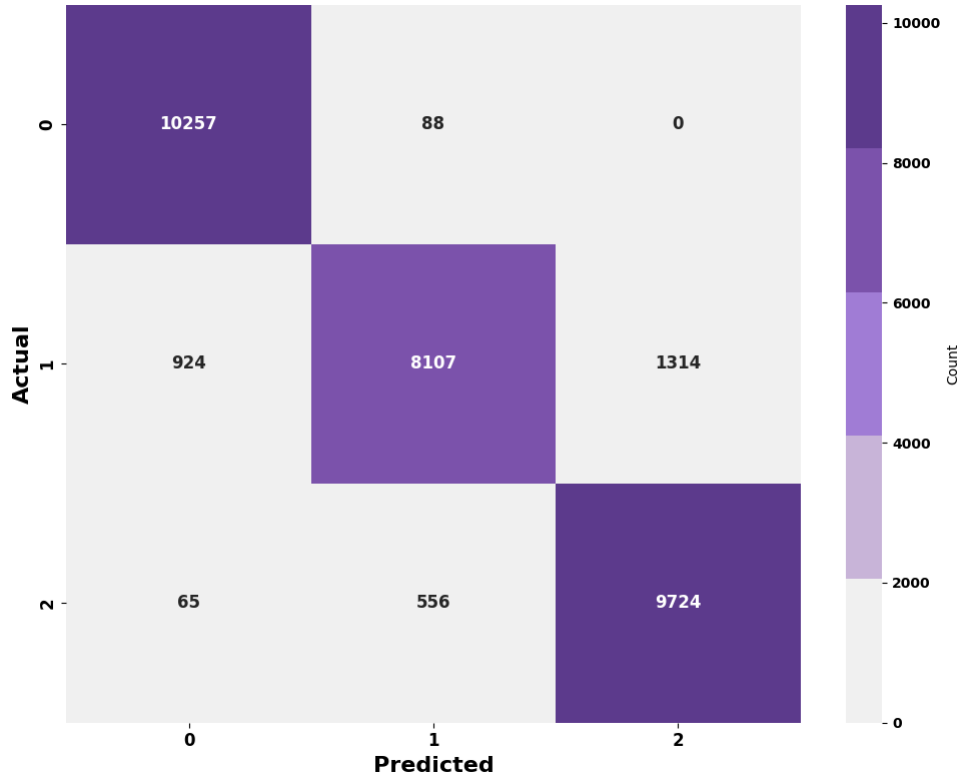

Figure 6.4: Confusion Matrix for Extra Trees on Balanced Dataset(ROS)

6.3 Hyperparameter Tuning Results

In this study, hyperparameter tuning was applied to optimize the performance of KNN and MLP models, as outlined in Section 5.5 and summarized in Table 6.3. The key hyperparameters, including the number of neighbors, distance metrics, and weighting methods for KNN, as well as the hidden layer configurations, activation functions, learning rates, and regularization parameters for MLP, were fine-tuned using grid search. By applying these hyperparameters, the models' classification performance on both balanced and imbalanced datasets was significantly enhanced.

Model	Accuracy	Precision	Recall	F1-Score	Training Time (s)	Testing Time (s)	AUROC Score
KNN	0.85	0.86	0.85	0.85	0.04	224.38	0.94
MLP	0.80	0.80	0.80	0.80	1798.33	0.25	0.90

Table 6.3: Hyperparameter Tuning Results for KNN and MLP Models

Model Performance After Hyperparameter Tuning: Results from Table 6.3 highlight the improvements in KNN and MLP performance after hyperparameter tuning. The KNN model exhibited a notable increase from its previous accuracy of 0.74 to 0.85 in overall performance metrics, with an AUROC score improving from 0.88 to 0.94. This indicates a significant enhancement in its ability to identify different classes. However, despite minimal training time (0.04 seconds), the testing time remained a drawback at 224.38 seconds. The MLP model also improved from 0.74 to 0.80 across all metrics, with an AUROC score increasing from 0.87 to 0.90. While its training time was high at 1798.33 seconds, the MLP showed faster testing performance, taking only 0.25 seconds.

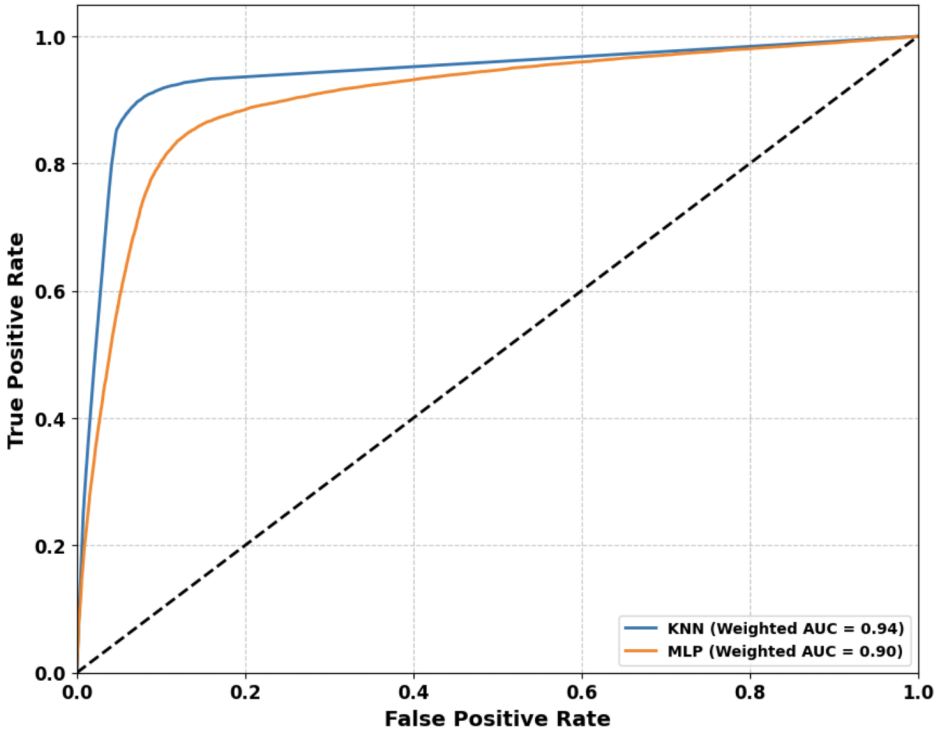

Figure 6.5: ROC Curve for Hyperparameter tunning results

Figure 6.5 shows the ROC curves for both models. KNN's sharper rise toward the upper-

left section indicates its enhanced effectiveness in separating classes, as evidenced by its higher AUROC score of 0.94, compared to MLP's 0.90. This suggests that after tuning, KNN became more proficient at identifying differences between classes in this dataset. Overall, KNN outperformed MLP post-tuning, particularly in terms of accuracy and AUROC score. This improvement can be attributed to fine-tuning parameters which enhanced its predictive power. However, its lengthy testing time underscores a computational inefficiency during evaluation. MLP, while showing progress, requires further optimization to reach similar performance levels.

6.4 Overall Comparison of Model Effectiveness

The thorough examination of model performance across imbalanced and balanced datasets, including pre- and post-hyperparameter tuning, uncovers vital findings for credit score assessment applications. ExtraTrees consistently proved to be the most effective algorithm, showing excellent performance and efficiency across both dataset types. This combination of high accuracy and rapid processing times makes ExtraTrees especially suitable for large-scale, real-time credit scoring applications, such as instant loan approvals for large populations. The model's notably enhanced ability to correctly identify poor credit cases after balancing the dataset is essential for reducing financial risks in lending decisions. While Bagging exhibited similar accuracy, its longer processing times might restrict its use in scenarios demanding quick decisions. The hyperparameter tuning results for KNN and MLP showed considerable enhancements in their class separation capabilities, although KNN's lengthy testing time could limit its application in high-volume, real-time scenarios. MLP's faster testing time after tuning could make it appropriate for complex, non-linear credit assessments requiring rapid processing. The shift in confusion matrices across datasets emphasized the importance of addressing class imbalance to improve the identification of all credit score categories, particularly aiding the detection of high-risk individuals. In practical applications, such as a bank processing thousands of loan applications daily, the ExtraTrees model would facilitate efficient and accurate categorization of applicants into risk levels, balancing the reduction of potential defaults with fair treatment of creditworthy customers. In conclusion, While ExtraTrees excels in its balanced performance across accuracy, efficiency, and adaptability, the selection of a model for credit score assessment should not be based solely on these factors.

It's crucial to consider specific operational requirements, acceptable balances between accuracy and speed, and the need for transparent decision-making processes. Each organization may have unique needs that influence their choice of model. This comprehensive evaluation highlights the importance of not only choosing the right algorithm but also ensuring appropriate data preprocessing and model tuning to develop robust and fair solutions.

6.4.1 10-Fold Cross-Validation for the best performing Model

Following the overall comparison, a 10-fold cross-validation was conducted for the best-performing ExtraTrees model to assess its stability and generalization capability. The cross-validation results, as shown in Figure 6.6, indicate a mean accuracy score of 0.8867 (88.67%) with a low standard deviation of 0.0015, highlighting the model's consistency across different subsets of the data. The individual fold scores ranged from a minimum of 0.8846 to a maximum of 0.8892, showing the model's ability to maintain strong performance across various data splits. The narrow range of scores and the low standard deviation suggest that the ExtraTrees model is not overfitting to any particular subset of the data and is likely to generalize well to unseen samples. Figure 6.7 illustrates the accuracy scores for each fold, with the x-axis representing the fold number (from 0 to 9) and the y-axis showing the corresponding accuracy score for each fold. The blue line traces the accuracy achieved by the ExtraTrees model during cross-validation, while the shaded area around the line reflects the standard deviation, indicating the range of fluctuation in accuracy across the folds. The red dashed line represents the mean cross-validation accuracy score of 0.8867 (88.67%), which serves as a reference point for evaluating the model's performance. The graph reveals some fluctuations, with a noticeable peak at the 8th fold where the accuracy reaches approximately 0.8892, its highest point. Despite these variations, the blue curve stays close to the mean accuracy score, highlighting the model's stability and consistent performance across different data subsets. The relatively narrow spread of the shaded area underscores the low variance in performance, with the standard deviation being only 0.0015. This low standard deviation suggests that the model's accuracy does not deviate significantly across the different folds, reflecting the model's reliability in achieving consistent results on unfamiliar data. While the cross-validation accuracy of 88.67% is slightly lower than the 91% accuracy achieved on the full dataset, this reduction is expected, as cross-validation provides a more realistic and unbiased evaluation by testing the model on multiple data splits. The plot overall

reinforces the effectiveness of the ExtraTrees model in handling credit scoring tasks, showing its robustness and ability to maintain strong performance without overfitting to any specific data subset.

```
ExtraTrees Cross-Validation Scores: [0.88545191 0.8881102  0.88625745 0.88851297 0.88569357 0.88649911
 0.88464637 0.88471763 0.88922903 0.88753726]

ExtraTrees Mean CV Score: 0.8867

ExtraTrees Standard Deviation of CV Score: 0.0015
```

Figure 6.6: 10-Fold Cross-Validation Results for Extra Trees Model

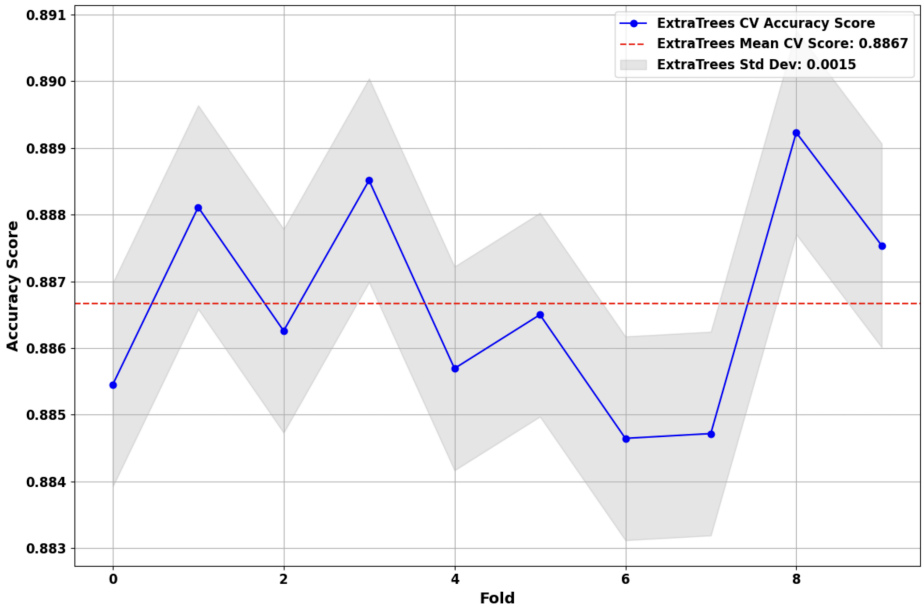

Figure 6.7: Performance Plot of Cross-Validation for Extra Trees Model

6.4.2 Feature Importance of best performing model

The feature importance analysis for the ExtraTrees model, as shown in Figure 6.8, reveals which factors play the most crucial role in the model's decision-making process. The x-axis represents various features used in the model, while the y-axis shows their corresponding importance as a percentage. At the top of the list are financial-related variables such as *Interest_Rate*, *Payment_of_Min_Amount*, *Outstanding_Debt*, and *Credit_Mix*, each contributing significantly to the model's accuracy. These top features suggest that the model relies heavily on recent financial behavior, such as how consistently an individual meets their minimum payments or manages their debt. After the top 10 to 15 features, there is a noticeable decline

in the importance scores, indicating that the model places less emphasis on subsequent factors. Notably, occupation-related features, like *Occupation_Accountant*, *Occupation_Engineer*, and *Occupation_Lawyer*, are among the least impactful, suggesting that personal characteristics or job roles contribute minimally to the prediction outcomes. Instead, the model prioritizes active financial management indicators, such as *Num_of_Delayed_Payment* and *Debt_to_Income_Ratio*, over broad categorical or historical data like occupation. This insight suggests that focusing on key financial behaviors, such as keeping interest rates low and paying off outstanding debt, could enhance predictive accuracy in credit risk assessment using the ExtraTrees model.

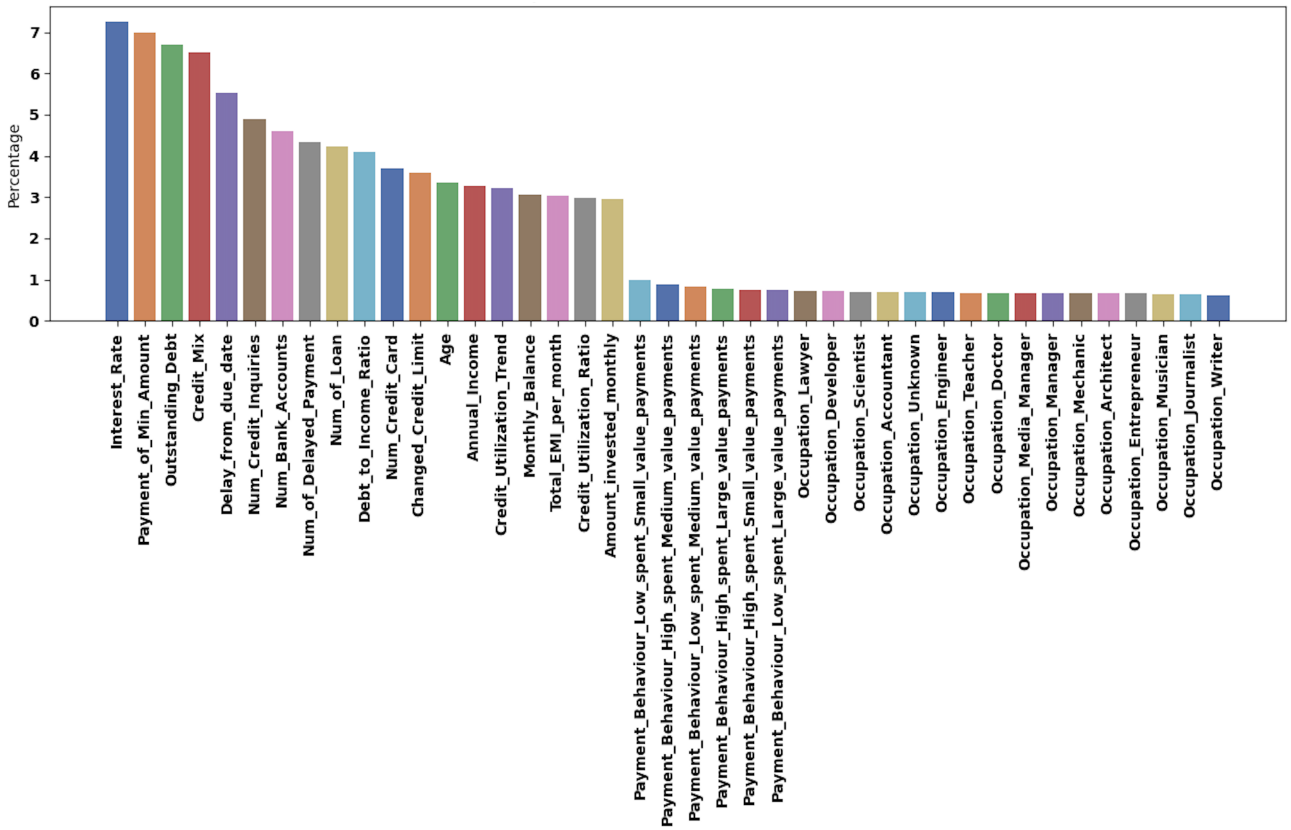

Figure 6.8: Feature Importance (Bar Chart) for Extra Trees Model

6.5 Limitations

This research encountered several limitations while experimenting with the credit score dataset, from data preprocessing to final results analysis.

Dataset Constraints: The study was based on a single credit scoring dataset, which may not capture the diverse geographic or demographic realities, restricting the relevance of the findings to other credit scoring contexts.

Balancing Techniques: Only ROS was used to address class imbalance. The impact of alternative techniques like the SMOTE, Random Under-Sampling (RUS), and other hybrid balancing techniques, which might yield different insights into model performance, was not explored.

Computational Expenses: The Bagging and MLP models required greater computational resources, with training times exceeding standard criteria, limiting the investigation of thorough hyperparameter tuning or the application of more computationally intensive models.

Initial Performance Issues: The models initially showed lower accuracy due to class imbalance, requiring the application of ROS to substantially improve performance. This indicates a limitation in the model's ability to handle imbalanced data effectively without such interventions, which may restrict its capability in diverse credit scoring environments.

Exploration of Alternative Methods: Due to time and computational constraints, the study did not analyze a wider range of machine learning algorithms, such as boosted trees or advanced ensemble methods, which could provide different advantages in handling complex datasets.

These limitations define the scope of the study's applicability and highlight areas for further research, particularly in improving the range of credit scoring models through methodological advancements and computational optimization.

6.6 Future Direction

This study has identified multiple areas for future research in credit scoring models:

Advanced Machine Learning Algorithms: Future work should explore more sophisticated algorithms such as XGBoost, LightGBM, and ensemble methods like Stacking. These

techniques may improve predictive accuracy in credit scoring tasks.

Computational Efficiency: To address the long training times observed with some models, research into parallel processing techniques and GPU acceleration is important. These strategies can considerably reduce computational costs and enable more comprehensive and efficient model tuning, allowing for the analysis of complex models and larger datasets without compromising performance.

Emerging AI Technologies: The application of recent developments in Artificial Intelligence, such as transformer models and graph neural networks, to credit scoring tasks presents a promising avenue for research. .

Feature Creation: The newly created features, debt-to-income ratio and credit utilization, in this study did not have an impact on the credit score. In future research, more features can be developed and evaluated to see if they contribute to predicting credit scores. If necessary, these features can be further refined or incorporated into the model training process.

Feature Selection: Applying feature selection techniques could lead to notable improvements in model performance. By selecting the most relevant features for credit scoring, noise can be reduced, which in turn enhances the predictive power and efficiency of the models. This approach can also simplify the model, making it more transparent and less prone to overfitting.

Dimensionality Reduction Techniques: Given the large size of the dataset, adopting techniques like Principal Component Analysis (PCA) could help reduce the dimensionality while maintaining key information. Future research could explore how well the models perform with reduced dimensionality, potentially leading to faster and more efficient model training.

Data Balancing Techniques: While this study used ROS, future research should compare this with other methods like SMOTE and RUS to determine the most effective approach for credit scoring datasets.

Real-Time Credit Scoring: Exploration of online learning models that can adapt to changing credit landscapes in real-time could lead to more responsive and accurate credit scoring systems.

By addressing these areas, future research can contribute to the development of more resilient, effective, and equitable credit scoring models. This work has the potential to greatly enhance the precision and dependability of credit assessments, benefiting both financial institutions and consumers.

CHAPTER 7

Conclusion

This research addressed the challenge of automating credit score classification for large populations by implementing machine learning models. The study aimed to develop models that could predict credit scores accurately and efficiently, surpassing the limitations of traditional manual methods. Data preprocessing, including cleaning, handling missing values, and applying the Adjusting Outliers (AO) technique, was a critical step in preparing the dataset. The AO method effectively reduced skewness and outliers, leading to more reliable feature analysis. Exploratory data analysis identified key factors influencing credit scores, with annual income standing out among numerical variables and credit mix emerging as the most impactful categorical feature. Feature engineering introduced new variables such as debt-to-income ratio and credit utilization, though these did not significantly improve model performance in this study. The research compared four machine learning models: Extra Trees, Bagging, KNN, and MLP. The Extra Trees model initially outperformed the others, achieving 78% accuracy on the imbalanced dataset. After applying ROS, the model's accuracy improved to 91%. Cross-validation confirmed that the model was not overfitting and was well-suited for this task. While KNN and MLP improved after hyperparameter tuning, they still lagged behind tree-based models. This study demonstrated the effectiveness of machine learning in classifying credit scores, with the Extra Trees model proving particularly robust. The feature importance analysis highlighted that financial behaviors, such as interest rates and outstanding debt, play a critical role in creditworthiness, while personal factors like occupation had little impact. Although this research faced limitations, such as the use of a

single dataset and limited exploration of other balancing techniques, it lays the foundation for future work. Potential areas for further investigation include exploring advanced machine learning models, enhancing computational efficiency, and developing new features for improved prediction accuracy. In summary, this research contributes to both the academic and practical understanding of automated credit scoring. The findings demonstrate that machine learning models can efficiently process large datasets and offer accurate, scalable solutions for real-world credit assessments. Addressing the identified limitations and pursuing the suggested future directions will help refine and further improve credit scoring systems, benefiting both financial institutions and consumers.

Bibliography

[1] How Is My Credit Score Calculated?
https://www.investopedia.com/ask/answers/05/creditscorecalculation.asp

[2] How do credit cards work?
https://www.mirrorreview.com/how-do-credit-cards-work/

[3] UK Credit Card Trends 2023-2024: More Spend and Delinquencies
https://www.fico.com/blogs/uk-credit-card-trends-2023-2024-more-spend-and-delinquencies/

[4] The future of credit.
https://www.fintechfutures.com/2024/05/state-of-play-the-future-of-credit/

[5] J. Galindo and P. Tamayo. Credit risk assessment using statistical and machine learning: Basic methodology and risk modeling applications. *Computational Economics*, 15:107–143, 2000. https://doi.org/10.1023/A:1008699112516.

[6] J. A. Lopez and M. R. Saidenberg. Evaluating credit risk models. *Journal of Banking & Finance*, 24(1-2):151–165, 2000. https://doi.org/10.1016/S0378-4266(99)00055-2.

[7] B. Twala. Multiple classifier application to credit risk assessment. *Expert Systems With Applications*, 37(4):3326–3336, 2010. https://doi.org/10.1016/j.eswa.2009.10.018.

[8] R. E. Turkson, E. Y. Baagyere, and G. E. Wenya. A machine learning approach for predicting bank credit worthiness. In *2016 Third International Conference on Artificial Intelligence and Pattern Recognition (AIPR)*, pages 1–7, Lodz, Poland, 2016. https://doi.org/10.1109/ICAIPR.2016.7585216.

[9] S. K. Trivedi. A study on credit scoring modeling with different feature selection and machine learning approaches. *Technology in Society*, 63:101413, 2020. https://doi.org/10.1016/j.techsoc.2020.101413.

[10] T. Mokheleli and T. Museba. Machine Learning Approach for Credit Score Predictions. *Journal of Information Systems and Informatics*, 5:497–517, 2023. https://doi.org/10.51519/journalisi.v5i2.487.

[11] Peng. Du and Hong. Shu. Exploration of Financial Market Credit Scoring and Risk Management and Prediction Using Deep Learning and Bionic Algorithm. *Journal of Global Information Management (JGIM)*, 30(9):1–29, 2023. http://doi.org/10.4018/JGIM.293286.

[12] F.M. Talaat, A. Aljadani, M. Badawy, et al. Toward interpretable credit scoring: integrating explainable artificial intelligence with deep learning for credit card default prediction. *Neural Computing & Applications*, 36:4847–4865, 2024. https://doi.org/10.1007/s00521-023-09232-2.

[13] V. Moscato, A. Picariello, and G. Sperlí. A benchmark of machine learning approaches for credit score prediction. *Expert Systems With Applications*, 165:113986, 2021. https://doi.org/10.1016/j.eswa.2020.10.048.

[14] H. Qian, P. Ma, S. Gao, and Y. Song. Soft reordering one-dimensional convolutional neural network for credit scoring. *Knowledge-Based Systems*, 266:110414, 2023. https://doi.org/10.1016/j.knosys.2023.110414. *Keywords:* Credit scoring, Soft reordering mechanism, Convolutional neural networks, Machine learning, Deep learning.

[15] Credit Score Classification Dataset, Kaggle, 2022.
https://www.kaggle.com/datasets/parisrohan/credit-score-classification
CC0: Public Domain.

[16] J. Sessa and D. Syed. Techniques to deal with missing data. In *2016 5th International Conference on Electronic Devices, Systems and Applications (ICEDSA)*, Ras Al Khaimah, United Arab Emirates, pages 1–4, 2016. https://doi.org/10.1109/ICEDSA.2016.7818486.

[17] M. Hubert and S. Van der Veeken. Outlier detection for skewed data. *Journal of Chemometrics*, 22(3-4):235–246, 2008. https://doi.org/10.1002/cem.1123.

[18] 8 Feature Engineering Techniques for Machine Learning https://www.projectpro.io/article/8-feature-engineering-techniques-for-machine-learning/423

[19] Towards Data Science. Categorical feature encoding. *Towards Data Science*, 2023. https://towardsdatascience.com/categorical-feature-encoding-547707acf4e5.

[20] Statistics How To. Standardized Values: Example. *Statistics How To*. https://www.statisticshowto.com/standardized-values-examples/.

[21] V. Moscato, A. Picariello, and G. Sperlí. A benchmark of machine learning approaches for credit score prediction. *Expert Systems With Applications*, 165:113986, 2021. https://doi.org/10.1016/j.eswa.2020.113986.

[22] K-Nearest Neighbors Algorithm. https://www.sciencedirect.com/topics/computer-science/k-nearest-neighbors-algorithm.

[23] C. Luo. A comparison analysis for credit scoring using bagging ensembles. In *Expert Systems*, 39(2):e12297, 2022. https://doi.org/10.1111/exsy.12297. *Keywords:* Credit scoring, Bagging ensembles, Machine learning, Classification, Ensemble learning, Predictive models.

[24] Corporate Finance Institute. Bagging (Bootstrap Aggregation). *Corporate Finance Institute*. https://corporatefinanceinstitute.com/resources/data-science/bagging-bootstrap-aggregation/.

[25] X. Chu, J. Yu, and A. Hamdulla. Throughput prediction based on ExtraTree for stream processing tasks. *Computer Science and Information Systems*, 18:31–31, 2020. https://doi.org/10.2298/CSIS200131031C.

[26] A. Safiya Parvin and B. Saleena. An Ensemble Classifier Model to Predict Credit Scoring - Comparative Analysis. *2020 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS)*, pages 27–30, Chennai, India, 2020. https://doi.org/10.1109/iSES50453.2020.00017.

[27] T. Ray, A. Kumar, V. Kumar, A. Kumar, R. Rai, M. Khandelwal, and T. Singh. Stability prediction of Himalayan residual soil slope using artificial neural network. *Natural Hazards*, 103, 2020. https://doi.org/10.1007/s11069-020-04141-2.

[28] Multilayer Perceptron (MLP). https://deepchecks.com/glossary/multilayer-perceptron/.

[29] GeeksforGeeks. Multi-Layer Perceptron Learning in TensorFlow. *GeeksforGeeks*. https://www.geeksforgeeks.org/multi-layer-perceptron-learning-in-tensorflow/.

[30] R. Wazirali. An improved intrusion detection system based on KNN hyperparameter tuning and cross-validation. *Arab Journal of Science and Engineering*, 45:10859–10873, 2020. https://doi.org/10.1007/s13369-020-04907-7

[31] Devskrol. Advanced Hyperparameter Tuning of a Multilayer Perceptron (MLP). *Devskrol*, December 22, 2021. https://devskrol.com/2021/12/22/advanced-hyperparameter-tuning-of-a-multilayer-perceptron-mlp/.

[32] A. Arram, M. Ayob, M. A. Albadr, A. Sulaiman, and D. Albashish. Credit card score prediction using machine learning models: A new dataset. *arXiv*, 2023. https://arxiv.org/abs/2310.02956.

[33] O. Dor and Y. Zhou. Achieving 80% ten-fold cross-validated accuracy for secondary structure prediction by large-scale training. *Proteins*, 66:838–845, 2007. https://doi.org/10.1002/prot.21298.

[34] M. Niu, Y. Li, C. Wang, and K. Han. RFAmyloid: A web server for predicting amyloid proteins. *International Journal of Molecular Sciences*, 19, 2018. https://doi.org/10.3390/ijms19072071.

CHAPTER 8

Appendix

8.1 Code Details:

1. **The GDrive link for code**: Google Drive Link
2. **The Colab link for code**: Google Colab Link

8.2 Classifcation Report

8.2.1 Imbalanced Dataset's Classifcation Report

```
Training and evaluating ExtraTrees...
Accuracy: 0.78
Precision: 0.78
Recall: 0.78
F1-Score: 0.78
Training Time: 22.70 seconds
Testing Time: 0.80 seconds

Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.71      0.72      3459
           1       0.80      0.80      0.80     10345
           2       0.77      0.78      0.78      5641

    accuracy                           0.78     19445
   macro avg       0.77      0.76      0.77     19445
weighted avg       0.78      0.78      0.78     19445
```

(a) Extra Trees report

```
Training and evaluating Bagging...
Accuracy: 0.78
Precision: 0.78
Recall: 0.78
F1-Score: 0.78
Training Time: 254.92 seconds
Testing Time: 2.48 seconds

Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.70      0.71      3459
           1       0.80      0.80      0.80     10345
           2       0.77      0.79      0.78      5641

    accuracy                           0.78     19445
   macro avg       0.76      0.76      0.76     19445
weighted avg       0.78      0.78      0.78     19445
```

(b) Bagging report

Figure 8.1: Extra Trees and Bagging classification report on Imbalanced Dataset

```
Training and evaluating KNN...
Accuracy: 0.66
Precision: 0.66
Recall: 0.66
F1-Score: 0.66
Training Time: 0.03 seconds
Testing Time: 13.66 seconds

Classification Report:
              precision    recall  f1-score   support

           0       0.51      0.55      0.53      3459
           1       0.70      0.73      0.71     10345
           2       0.68      0.59      0.63      5641

    accuracy                           0.66     19445
   macro avg       0.63      0.62      0.62     19445
weighted avg       0.66      0.66      0.66     19445
```

(a) KNN report

```
Training and evaluating MLP...
Accuracy: 0.69
Precision: 0.69
Recall: 0.69
F1-Score: 0.69
Training Time: 169.77 seconds
Testing Time: 0.07 seconds

Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.63      0.60      3459
           1       0.73      0.75      0.74     10345
           2       0.71      0.63      0.67      5641

    accuracy                           0.69     19445
   macro avg       0.67      0.67      0.67     19445
weighted avg       0.69      0.69      0.69     19445
```

(b) MLP report

Figure 8.2: KNN and MLP classification report on Imbalanced Dataset

8.2.2 Balanced Dataset's Classifcation Report

```
Training and evaluating ExtraTrees...
Accuracy: 0.91
Precision: 0.91
Recall: 0.91
F1-Score: 0.9
Training Time: 24.89 seconds
Testing Time: 1.00 seconds

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.99      0.95     10345
           1       0.93      0.78      0.85     10345
           2       0.88      0.94      0.91     10345

    accuracy                           0.91     31035
   macro avg       0.91      0.91      0.90     31035
weighted avg       0.91      0.91      0.90     31035
```

(a) Extra Trees report

```
Training and evaluating Bagging...
Accuracy: 0.9
Precision: 0.9
Recall: 0.9
F1-Score: 0.9
Training Time: 267.68 seconds
Testing Time: 2.32 seconds

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.99      0.94     10345
           1       0.94      0.76      0.84     10345
           2       0.88      0.95      0.91     10345

    accuracy                           0.90     31035
   macro avg       0.90      0.90      0.90     31035
weighted avg       0.90      0.90      0.90     31035
```

(b) Bagging report

Figure 8.3: Extra Trees and Bagging classification report on Balanced Dataset

```
Training and evaluating KNN...
Accuracy: 0.74
Precision: 0.74
Recall: 0.74
F1-Score: 0.73
Training Time: 0.05 seconds
Testing Time: 35.18 seconds

Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.91      0.80     10345
           1       0.73      0.52      0.61     10345
           2       0.78      0.78      0.78     10345

    accuracy                           0.74     31035
   macro avg       0.74      0.74      0.73     31035
weighted avg       0.74      0.74      0.73     31035
```

(a) KNN report

```
Training and evaluating MLP...
Accuracy: 0.74
Precision: 0.74
Recall: 0.74
F1-Score: 0.73
Training Time: 284.07 seconds
Testing Time: 0.06 seconds

Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.87      0.80     10345
           1       0.72      0.54      0.62     10345
           2       0.76      0.81      0.78     10345

    accuracy                           0.74     31035
   macro avg       0.74      0.74      0.73     31035
weighted avg       0.74      0.74      0.73     31035
```

(b) MLP report

Figure 8.4: KNN and MLP classification report on Balanced Dataset

8.2.3 Hyperparameters Classification Report

```
Tuning hyperparameters and evaluating KNN...
Fitting 3 folds for each of 10 candidates, totalling 30 fits
Accuracy: 0.85
Precision: 0.86
Recall: 0.85
F1-Score: 0.85
Training Time: 0.08 seconds
Testing Time: 219.85 seconds

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.99      0.91     10345
           1       0.91      0.64      0.75     10345
           2       0.84      0.93      0.88     10345

    accuracy                           0.85     31035
   macro avg       0.86      0.85      0.85     31035
weighted avg       0.86      0.85      0.85     31035
```

(a) KNN report

```
Accuracy: 0.8
Precision: 0.8
Recall: 0.8
F1-Score: 0.8
Training Time: 1724.69 seconds
Testing Time: 0.15 seconds

Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.94      0.88     10345
           1       0.77      0.63      0.69     10345
           2       0.80      0.84      0.82     10345

    accuracy                           0.80     31035
   macro avg       0.80      0.80      0.80     31035
weighted avg       0.80      0.80      0.80     31035
```

(b) MLP report

Figure 8.5: Hyperparameters Classification Report