Detecting Bank Account Opening Fraud Using Machine Learning

Applied Research Project submitted in partial fulfilment of the requirements for the degree of

MSc. Data Analytics

at Dublin Business School

Chukwuebuka Uwaoma

Student ID:10621864

Supervisor: Hamidreza Khaleghzadeh

January 2024

Declaration

I declare that this Applied Research Project that I have submitted to Dublin Business School for the award of MSc Data Analytics is the result of my own investigations, except where otherwise stated, where it is clearly acknowledged by references. Furthermore, this work has not been submitted for any other degree.

Signed: CHUKWUEBUKA UWAOMA

Student Number: 10621864

Date: 08/01/2024

Acknowlegement

I will like to express my appreciation to my supervisor Hamidreza Khaleghzadeh for his guidance throughout the completion of this research project. And also, thank my parents and siblings for their immense support throughout my program, without which none of this was achievable.

Abstract

This research explores machine learning techniques for detecting fraudulent bank account openings using a recently published large-scale benchmark dataset. Multiple classification algorithms are evaluated, including Logistic Regression, Decision Trees, Random Forests, and LightGBM. Following standard data preprocessing and feature engineering, these models are trained on an imbalanced dataset of one million account applications over eight months. Adaptive synthetic oversampling is utilized to mitigate the extreme rarity of positive fraud cases.

Various performance metrics assess model accuracy, real-world feasibility constraints, and fairness across protected demographic groups. Initial results indicate LightGBM achieved the best overall recall of 62%, capturing most fraudulent instances. However, enforcing a 5% false positive rate threshold is necessary for practical usage but severely impacts recall. Predictive equality analysis also exposes some algorithms that inadvertently introduce bias against seniors. These findings indicate that combining sampling techniques with gradient boosting methods has the potential to balance performance, operational constraints, and ethical considerations in identifying criminal account openings. More hyperparameter tuning and model ensembling could enhance performance. This study establishes a rigorous methodology and a benchmark for future research into machine learning for fraud detection in the banking sector.

Keywords: Bank account fraud, Fraud detection, Machine learning, Model fairness, Gradient Boosting

Contents

List of Figures

List of Tables

1 Introduction

Bank Account Opening Fraud has been an existential risk to the financial sector, undermining trust and resulting in major financial losses for both institutions and customers. Fraudsters' evolving tactics therefore require novel approaches for timely detection and prevention. The employing of algorithms that utilize machine learning appears to be a promising solution. This research proposal analyses the efficacy of various machine learning algorithms in detecting Bank Account Opening Fraud, taking into account various evaluation metrics, handling class imbalance issues, and evaluating fairness metrics.

1.1 Background of the Problem

The act of setting up fraudulent bank accounts by means of using unauthorised attempts to take advantage of the shortcomings in financial institutions' protocols is known as bank account opening fraud. These activities include a variety of strategies, such as document fabrication and identity theft. The ability of traditional rule-based systems to adapt to the constantly shifting fraud landscape has proven to be limited. As a result, data-driven techniques are becoming more and more necessary to continuously learn and adjust to new fraudulent patterns.

1.2 Statement of the Problem

This study tackles the main problem by employing machine learning techniques in identifying bank account opening instances of fraud. This amount of data continues to increase, and sophisticated fraud techniques have made traditional methods lacking in providing accurate and timely detection. The purpose of the study is to explore the potential of machine learning algorithms in detecting bank account opening fraud.

1.3 Purpose of the Study

The main objective of this study is to assess the effectiveness of various machine learning algorithms in detecting Bank Account Opening Fraud. This study aims to provide insights into the most effective techniques for improving fraud detection accuracy by thoroughly analyzing multiple algorithms and their strengths and limitations.

1.4 Research Questions

The following research questions are being addressed by this study:

- How effective are different machine learning algorithms at detecting bank account opening fraud?
- What are the difficulties associated with class imbalanced data sets in bank account fraud, and how does this affect model performance?
- How well do these models perform in terms of fairness metrics?

1.5 Significance of the Study

This study is highly important to a wide range of stakeholders. Financial institutions have the potential to benefit from the improved fraud detection system and help safeguard their financial resources and reputation. Additionally, effective fraud detection also contributes to the financial security and trust of customers in the banking system. The research community will gain insight into the effectiveness of different machine learning in real-world fraud scenarios, which will assist in helping to develop more robust detection methods. Ultimately, this research outlines an extensive look into detecting Bank Account Opening Fraud through the use of machine learning algorithms. Given the prevalence of fraud and the continuous evolution of fraudulent tactics,

the potential benefits of accurate and efficient fraud detection are important. The subsequent sections will delve deeper into the existing literature, research methodology, and expected work plan to achieve the objectives of this study.

2 Literature Review

The evolution of technology and it's incorporation into financial systems has opened up new opportunities for fraudulent individuals, necessitating innovative strategies for identifying and preventing financial crimes. Machine learning (ML) algorithms have come to light as a potential counter-measure against bank account opening fraud (BAF). This literature review presents an overview of existing research in the field, highlighting the different ML techniques implemented, their effectiveness, obstacles, and the importance of handling class imbalance and fairness metrics in fraud detection.

2.1 Review of Research

The modern banking sector massively benefits from analytics, leveraging vast daily data to inform decisions and enhance services. Banking analytics and complex data analysis techniques facilitate strategic decision-making, boost customer satisfaction, and optimise profitability (Nugroho and Hamsal 2021).

Banking analytics have revolutionised customer relationship management, risk management, and operational efficiency. Customer relationship management helps banks segment their customer base and anticipate their needs based on their transaction histories and other factors, leading to more effective marketing strategies (Domingos, Ojeme, and Daramola 2021). Banking Analytics supports proactive risk mitigation strategies, timely anomaly detection, and potential threat prediction in risk management. Regarding operational efficiency, it identifies bottlenecks to streamline workflows and automate routine tasks, ultimately enhancing productivity (The Economist Intelligence Unit, 2018). Hence, through analytics, banks can actualise decision-making, heighten customer service, mitigate risks, and streamline operations.

2.2 Bank Account Fraud

Bank account fraud has become widespread and a concerning issue affecting both the financial institutions and their customers (*Total Identity Fraud Losses Soar to $56 Billion in 2020* 2023). It iinvolves unauthorised access and misuse of bank accounts or related information for nefarious purposes. Consequences include significant financial losses, damage to credit history for individuals, and reputational harm for the institutions (Yu and He 2021; Kemp and Pérez 2023).

Although traditional prevention methods exist, they fall short of advanced fraud techniques. Machine learning presents a potentially effective tool in combating bank account fraud by identifying patterns and anomalies associated with fraudulent activities. However, data quality, model interpretability, and evolving fraud methods pose specific challenges (Žigienė, Rybakovas, and Alzbutas 2019; Amin et al. 2023). In other to ensure the integrity of the banking system, understanding the forms of bank account fraud and overcoming challenges in implementing machine learning for detection is crucial.

2.3 Significance of Bank Account Opening Fraud

The risk of fraudulent activities has increased as financial institutions increasingly rely on online and mobile platforms to provide their services, and bank account opening fraud has severe implications for both customers and financial institutions (Guo et al. 2018). Firstly, fraud directly impacts individuals' financial security and victims may suffer significant financial losses and emotional distress and face the burden of recovering their stolen identity and assets (Spathis 2002).

BAF can affect customers' trust in the financial institution when individuals become victims of such fraudulent activities, their trust in the bank's security measures, as well as the

overall integrity of the banking system can be severely undermined (Gates and Jacob 2009). Consequently, this erosion of trust can have widespread ramifications as customers might opt to switch to alternative financial institutions or exhibit hesitancy when engaging in digital banking transactions, impeding the growth and innovation within the financial industry.

Furthermore, bank account opening fraud does not only impact individual customers but also imposes significant losses upon financial institutions themselves. The resulting reputation harm and substantial monetary damages burden these establishments considerably. Investigating, mitigating, and resolving cases related to fraudulent incidents incur exorbitant costs that strain affected banks' resources while diverting attention from core business operations. Additionally, regulatory authorities impose stringent penalties on institutions that fail to implement adequate measures to prevent and detect fraudulent activities, further amplifying the significance of addressing bank account opening fraud (Consumer Financial Bureau, 2023).

In conclusion, we cannot underestimate the significance of bank account opening fraud because it threatens the financial security of individuals, erodes trust in the banking system and imposes substantial financial burdens on both customers and financial institutions. To effectively address this issue, robust fraud detection mechanisms must be developed and implemented, with a focus on proactively identifying and preventing fraudulent bank account openings using machine learning technologies.

2.4 Challenges with Fraud Detection in Machine-Learning

Machine learning (ML) application in fraud detection presents various challenges due to factors such as the dynamic nature of fraud patterns, data quality and variability, model interpretability, and dataset balance.

2.4.1 Adapting to the Dynamic Nature of Fraud

Machine learning (ML) algorithms can learn from large datasets and adapt to changing fraud patterns. However, the constantly evolving tactics of fraudsters aimed at exploiting system vulnerabilities make it difficult for traditional-based systems models to keep pace.

2.4.2 Data Quality and Availability

The performance of ML models dramatically depends on the availability of varied and high-quality training data. Collecting, validating, and maintaining a representative dataset encompassing various fraudulent activities remains a vital hurdle in fraud detection(Dal Pozzolo, Caelen, et al. 2014).

2.4.3 Interpretability and Explainability of Models

Despite their high predictive accuracy, ML models, particularly those based on deep learning, often suffer from a 'black box' problem where their decision-making process is opaque. This lack of transparency can prevent stakeholders from trusting these models, a significant problem especially in regulated sectors such as banking where justification for decisions is often mandatory.

2.4.4 Handling Imbalanced Datasets

Fraud datasets are typically skewed, composed chiefly of legitimate transactions and only a small percentage of fraudulent ones. This imbalance can lead to a bias in ML models toward the majority class (Abdallah, Maarof, and Zainal 2016).

2.5 Guaranteeing Fairness in Models

Aside from the above challenges, ensuring fairness in fraud detection models is imperative (Gianfrancesco et al. 2018). Models should not perpetuate existing biases or favour specific demographics or groups, as this could lead to disproportionate effects from false positives or negatives(Caton and Haas 2020).

2.6 Recent Research

While research on fraud detection has made significant strides, a noticeable disparity exists between applied studies in credit card fraud detection and bank account opening fraud. Much of the literature has focused on credit card fraud detection, showcasing many methods and findings that can potentially be transferred to Bank Account Opening fraud detection.

2.6.1 Methodologies

The research landscape in recent years has witnessed extensive investigation into credit card fraud detection. Notably, several studies have employed machine learning algorithms to address the unique challenges of credit card fraud. The research methodologies found in recent research have leveraged supervised and unsupervised machine learning algorithms. For supervised learning, techniques like Random Forests, Support Vector Machine (SVM), Neural Networks, LSTM (Long Short-Term Memory), Logistic Regression, Naive Bayes, Decision trees, and ensemble methods have been commonly applied (Randhawa et al. 2018; Awoyemi, Adetunmbi, and Oluwadare 2017; Dal Pozzolo, Boracchi, et al. 2018). Both classical Machine learning algorithms and deep learning models like CNN, LSTM, and GRU have also been evaluated (Varmedja et al. 2019; Roy et al. 2018).

Other researchers have utilised unsupervised learning techniques like autoencoders, self-

organising maps, isolation forests and clustering in credit card fraud detection. These methods can be beneficial for handling unlabeled data. Some studies have explored hybrid semi-supervised approaches that combine labelled and unlabeled data to improve the accuracy of their models (Zamini and Montazer 2018; Sisodia, Reddy, and Bhandari 2017; Melo-Acosta, Duitama-Muñoz, and Arias-Londoño 2017).

A common approach most research employs involves training machine learning models using historical transaction data. These models extract transaction features such as time, location, amounts, and merchant information. The trained models then classify new transactions as fraudulent or legitimate. To address class imbalance issues that often occur in this type of data, resampling techniques like SMOTE have been employed (Awoyemi, Adetunmbi, and Oluwadare 2017; Sailusha et al. 2020)

Generally, model performance was evaluated using metrics like accuracy, AUC, precision, recall, and F1-score across fraudulent and legitimate classes. However, Bahnsen et al. (2013) also considered the cost-benefit tradeoffs of their models.

2.6.2 Strengths and Weaknesses

The research methodologies employed in several studies demonstrate notable strengths. One strength is the utilisation of authentic transaction datasets from real-world scenarios, as highlighted by Randhawa et al. (2018) and Dal Pozzolo, Boracchi, et al. (2018). Additionally, researchers such as Esenogho et al. (2022) have successfully enhanced performance by combining multiple algorithms. Another commendable aspect is observed in Jiang et al. (2018)'s study, where they address concept drift by retraining models on new data.

However, reliance on single datasets, lack of model interpretability analysis, and limited demographic comparisons are weaknesses seen in studies by Awoyemi, Adetunmbi, and Oluwadare (2017), Roy et al. (2018), and Sisodia, Reddy, and Bhandari (2017). Failure to detect fraud in real-time is another limitation acknowledged by Xuan et al. (2018). More research on inte-

grating unlabeled data and comparing different neural architectures is noted by Melo-Acosta, Duitama-Muñoz, and Arias-Londoño (2017) and Roy et al. (2018).

2.6.3 Results

Through rigorous experimentation, various methods, including Random Forests, LSTM, GRU, CNN, and ensemble methods, were employed by multiple researchers, and these advanced methodologies appear to outperform traditional approaches in numerous instances consistently Randhawa et al. 2018; Roy et al. 2018; Esenogho et al. 2022. Hybrid approaches were also shown to achieve higher accuracy, AUC, precision, recall, and F1 scores than individual models Randhawa et al. 2018; Taha and Malebary 2020. Awoyemi, Adetunmbi, and Oluwadare (2017) and Ileberi, Sun, and Wang (2021) showed that addressing class imbalance via resampling improves outcomes. Sisodia, Reddy, and Bhandari (2017) demonstrated that model performance varies across datasets and types of fraud. Deep learning models require large datasets and have been shown to be computationally more costly. (Varmedja et al. 2019). However, no single approach has been shown to be universally optimal.

2.6.4 Datasets

Most research leverages public datasets from sources like Kaggle and private real-world datasets from banks and financial institutions (Randhawa et al. 2018; Dal Pozzolo, Boracchi, et al. 2018). Real-world data provides the most robust insights but is limited in availability due to confidentiality issues (Randhawa et al. 2018). Synthetic data may fail to capture fraud patterns (Aleskerov, Freisleben, and Rao 1997). Class imbalance is a consistent challenge, with fraudulent transactions heavily outnumbered by legitimate ones (Dal Pozzolo, Boracchi, et al. 2018; Sisodia, Reddy, and Bhandari 2017). Concept drift also occurs as fraud evolves (Jiang et al. 2018).

2.6.5 Resampling Techniques

Oversampling minority fraudulent classes using SMOTE and undersampling majority legitimate classes are standard techniques to address imbalance (Sisodia, Reddy, and Bhandari 2017; Ileberi, Sun, and Wang 2021). However, oversampling can increase overfitting, while undersampling loses information. In contrast, advanced forms like SMOTE ENN combine oversampling and undersampling to improved the model results (Sisodia, Reddy, and Bhandari 2017). Ensemble techniques like boosting also help against imbalance by focusing on difficult instances (Esenogho et al. 2022). Cost-sensitive learning considers misclassification costs (Bahnsen et al. 2013). No single resampling technique is ideal across all scenarios. Selecting an appropriate method based on the dataset is imperative (Sisodia, Reddy, and Bhandari 2017).

2.6.6 Fairness

There have been few studies that assess model fairness across demographics or data subgroups. Understanding how fraud detection systems affect different groups is an area that has received little attention. The evaluation of model fairness across demographics or data subgroups is a neglected area in the field, as noted by Roy et al. (2018). There is a lack of understanding regarding the impact of fraud detection systems on different groups. Techniques like adversarial debiasing and causal modelling could help improve fairness but remain unexplored for fraud detection (Chen et al. 2020). Therefore, more research is needed on fairness-aware fraud analytics.

2.6.7 Limitations and Research Gaps

(Dal Pozzolo, Boracchi, et al. 2018; Varmedja et al. 2019; Bahnsen et al. 2013) idenitifed some key limitations in the research and these include class imbalance, concept drift, lack of comprehensive real-world datasets, verification latency issues, model overfitting, and lack of model explanations.

Significant research gaps include integrating unlabeled data, hyperparameter optimisation for deep learning models, testing across diverse datasets, mobile/online fraud detection, model interpretability, real-time detection systems, and fairness evaluations across demographics (Melo-Acosta, Duitama-Muñoz, and Arias-Londoño 2017; Roy et al. 2018; Sisodia, Reddy, and Bhandari 2017). Additionally, applying fraud detection models to emerging transaction channels and other fraud types beyond credit cards requires further research (Gyamfi and Abdulai 2018).

While credit card fraud detection has benefited from extensive research, the same level of applied study is comparatively limited in the context of bank account opening fraud. However, the methodologies and insights from credit card fraud detection can serve as a valuable foundation for addressing bank account opening fraud. We can directly apply the insights gained from feature engineering, model optimisation, and addressing class imbalance to the challenges posed by bank account opening fraud, facilitating the development of effective detection strategies.

In conclusion, the recent surge in research on credit card fraud detection has yielded valuable methodologies and insights that hold the potential for detecting bank account opening fraud. By adapting and transferring these methods, bank account opening fraud detection can benefit from the progress achieved in credit card fraud detection. This knowledge transfer can contribute to developing more accurate, efficient, and adaptive fraud detection strategies tailored to the intricacies of Bank Account Opening fraud scenarios.

3 Research Methodology

3.1 Methodologies

Knowledge Discovery in Databases (KDD) is the process of extracting valuable knowledge from large amounts of data. It involves multiple steps, including data preprocessing, mining, evaluation, and interpretation.

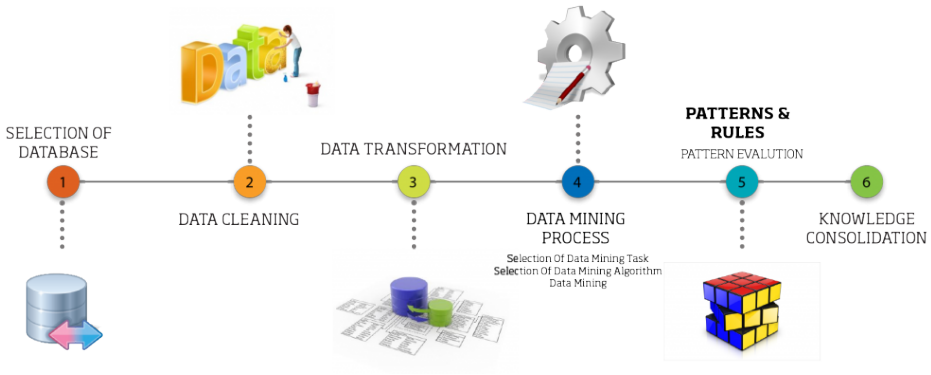

Figure 3.1: Outline of the Steps of the KDD Process

Image source:

https://www.cognixia.com/blog/data-mining-integral-part-knowledge-discovery-process/

Data preprocessing is an essential first step in the KDD process. It prepares the raw data for mining by cleaning, integrating, transforming, and reducing it. Typical tasks include handling missing values, removing noise and outliers, combining multiple data sources, normalisation, discretisation, feature selection, and dimensionality reduction. Proper preprocessing dramatically improves the quality of patterns found in later stages. Popular techniques include data cleaning, integration, transformation, removal, and feature selection (Storti et al. 2018).

Once the data is preprocessed, data mining techniques can be applied to discover patterns and build models. Standard methods include classification, clustering, regression, association

rule learning, and anomaly detection (Jui et al. 2021). Classification predicts categorical labels like 'benign' or 'malignant'. Algorithms used include decision trees, SVM, kNN, and Naive Bayes. Regression builds models predicting continuous values. Association rule learning finds relationships between attributes. Anomaly detection identifies outliers that deviate from expected patterns. The appropriate mining technique is determined by the goal and data type.

New models and patterns must be evaluated and interpreted before being used. Evaluation metrics like accuracy, precision, recall, and F1-score quantify model performance on test data. Visualisations and data summaries help interpret and explain patterns while domain knowledge is needed to validate the usefulness of discoveries. Iterating between mining, evaluation, and interpretation can lead to actionable knowledge.

KDD has diverse applications like fraud detection, customer segmentation, healthcare, education, and manufacturing (Waseem and Abidin 2023). In healthcare, it is used for diagnosis, treatment optimisation, resource management, and public health. It provides insights into student performance, dropout risks, resource usage, and personalised learning. Retailers use it to profile customers, run marketing campaigns, and manage inventory. With KDD, financial institutions can detect fraud, model risk, and identify growth opportunities. Based on its versatile methodology, KDD can be applied across multiple domains.

On the other hand, the CRISP-DM (Cross-Industry Standard Process for Data Mining) is a commonly used and robust process model for data mining projects. It breaks down the life cycle of a data mining project into six key phases: business understanding, data understanding, data preparation, modelling, evaluation, and deployment (Abasova, Tanuska, and Rydzi 2021).

Business Understanding: The first phase focuses on understanding the project objectives and requirements from a business perspective. This involves discussions with stakeholders to determine the problem definition, project plan, and data mining goals (Wowczko 2015). Critical tasks include determining the business objectives, assessing the situation, determining the data

mining goals, and producing a project plan.

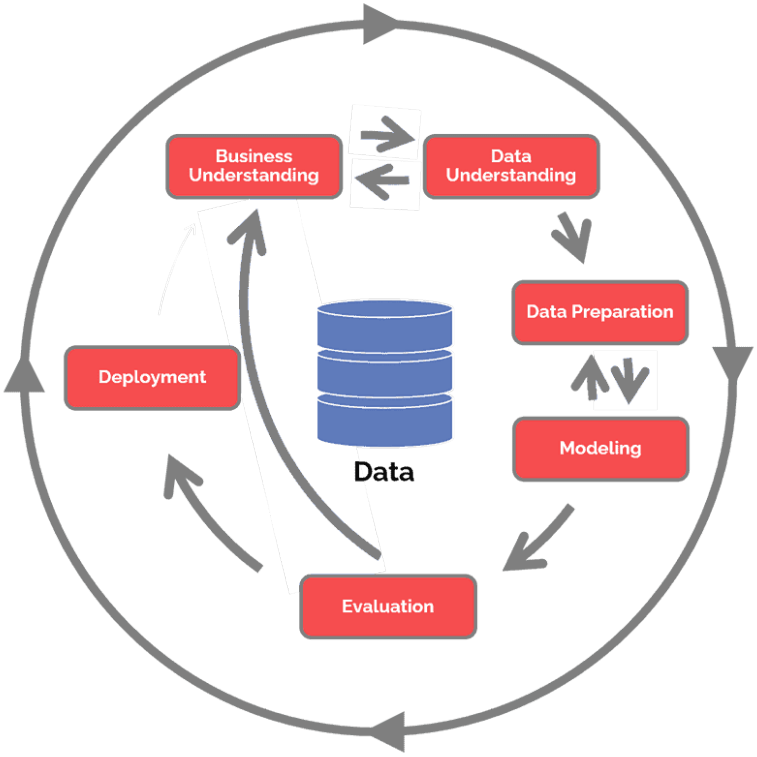

Figure 3.2: Relationship between the different phases of CRISP-DM

Image source: https://www.datascience-pm.com/crisp-dm-2/

Data Understanding: In this phase, the focus is on initial data collection, data description, data exploration, and data quality verification. The raw data is collected, described, explored, and verified for quality issues. Tasks involve collecting initial data, describing, exploring, and verifying data quality (Abasova, Tanuska, and Rydzi 2021).

Data Preparation: This crucial phase covers all activities needed to construct the final dataset for modelling. Tasks include data selection, cleaning, construction, integration, and formatting (Wowczko 2015). Activities involve table, record, attribute selection, handling missing values, noise removal, aggregation, and data transformation.

Modelling: Various modelling techniques are selected and applied to the prepared dataset

to achieve the data mining goals in this stage. Tasks involve selecting modelling techniques, generating test designs, building models, and assessing models(Abasova, Tanuska, and Rydzi 2021). Different techniques like classification, regression, and clustering are chosen based on the project goals.

Evaluation: The model is thoroughly evaluated and reviewed in this phase to ensure it achieves the business objectives. Based on the test data, the model is assessed on metrics like accuracy, precision, and recall. Tasks include evaluating results, reviewing processes, and determining the next steps (Wowczko 2015).

Deployment: The knowledge gained is organized and presented for deployment in the business. Tasks include planning deployment, monitoring and maintenance, producing final reports, and reviewing projects (Abasova, Tanuska, and Rydzi 2021). This last phase focuses on applying the data mining results.

CRISP-DM provides a structured approach to execute data mining projects through its six phases. It is a robust methodology that can help produce valuable insights from data and its business-focused outlook and emphasis on deployment ensure that the data mining outcomes are impactful and valuable.

3.2 CRISP-DM

In this research, we adopted the CRISP-DM methodology as it provides a clear framework for our data mining project on detecting bank account opening fraud. The structured approach will help develop a practical predictive model meeting project objectives.

3.2.1 Business Understanding

The business understanding phase focuses on understanding the project objectives and requirements from a business perspective and formulating a data mining problem definition (Chapman

et al. 2000). In the context of detecting bank account opening fraud, the objectives would include identifying potentially fraudulent applications and reducing the number of false positives, thereby improving the efficiency of the account opening process and mitigating financial losses due to fraudulent activity.

The main goal of this project was to create a machine-learning model that can detect fraudulent bank account opening applications. Bank account fraud leads to significant financial losses, so detecting suspicious or fraudulent applications before accounts are opened is extremely important. This was achieved with the employment of machine learning techniques and historical data to develop a predictive model that can distinguish between fraudulent and non-fraudulent applications.

Some specific business requirements for the model include:

- High recall - the model must identify as many of the fraudulent account applications as possible, even if some false positives occur
- Interpretability - the model should provide some visibility into why an application is classified as fraudulent.
- Operational efficiency - the model needs to enable quick screening of applications with minimal manual review
- Fairness - the model should treat all applications fairly and avoid unintended bias towards demographics or groups.

3.2.2 Data Understanding

The data understanding phase covers initial data collection and exploring the data to become familiar with it (Chapman et al. 2000). The key goals of this phase are to become familiar with the data, identify data quality issues, discover first insights, and detect interesting subsets or patterns in the data (Azevedo and Santos 2008).

This goal was achieved by collecting the initial data and then reviewing it at a high level to understand the format, types of variables, size of the dataset, etc. An initial exploratory data analysis of summaries, visualizations, and distributions of key variables can reveal potential data quality issues and provide a baseline understanding. Deeper analysis, such as looking at correlations, relationships, and segments in the data, can unveil more detailed insights. Understanding the business context and objectives will also inform the analysis and any required data transformations. Overall, the data understanding phase lays the groundwork for the rest of the project by enabling us to better understand the potential, limitations, and challenges of the dataset. Bearing this in mind, we effectively planned the remaining data preparation, modeling, evaluation, and deployment phases while aligning business goals.

3.2.3 Data Preparation

The data preparation phase plays an essential role for transforming the dataset that was used into the final dataset for modeling. Based on the understanding gained in the previous stage, appropriate steps were taken to clean, construct, integrate, and format the data correctly.

For this project, the following tasks were performed: handling missing values, handling class imbalance, detecting and removing outliers, transforming variables for normalization, encoding categorical variables, feature engineering to create new variables, filtering or sampling the dataset if needed, and splitting the data into training and test sets. The goal is to improve data quality and fix any issues that could negatively impact the modeling phase.

Furthermore, domain knowledge and the intended modeling techniques were used as guides in the data preparation decisions such as discretization or feature scaling. The prepared dataset contained all of the relevant variables required to meet the business objectives and was optimally structured for use with the modeling algorithms. In the subsequent phases of our project, the data was properly prepared for the fundamental to extract meaningful, valid insights and built highly accurate predictive models. The data preparation phase refined the initial dataset, which

had an impact on the data mining analysis.

3.2.4 Modelling

The modeling phase was where various data mining and machine learning techniques were selected and applied to the prepared dataset from the previous step. The goal was to explore different models and algorithms to uncover patterns, relationships, and insights aligned to the business objectives (Kotsiantis, Zaharakis, and Pintelas 2006).

Based on the goal of our project, the following algorithms were evaluated:

- Logistic Regression
- Decision Tree
- Random Forest
- Light Gradient Boosting Machine

Within each technique, multiple algorithms and parameters on training and test datasets were compared to identify the optimal models. Hyperparameter tuning was performed on the validation set for each algorithm to optimize model performance. The models were evaluated and compared on the test set using metrics like accuracy, precision, recall, F1 score, and fairness.

The key deliverable of the modeling phase was used to optimized data mining models that extracted valuable, actionable insights from the data with the rigorous modeling process completed. In the next phase, the models were thoroughly evaluated and interpreted to turn the information into business knowledge that drives accurate results.

3.2.5 Evaluation

In the evaluation stage, the data mining models generated were thoroughly accessed in the previous phase for reliability, accuracy, and business value. There are two crucial aspects of

model evaluation. First, we evaluated the quality metrics like accuracy, precision, recall, f1-score, confusion matrices, ROC curves, and fairness on test datasets to validate predictive capability. Second, was the business review of model findings and insights by domain experts to ensure alignment with objectives and real-world applicability.

Based on these data quality and business relevance assessments, we selected the best-performing models, and the others were discarded. Further iterations improved models that do not yet meet the evaluation criteria. Having an understanding of the key drivers and variables underlying the predictions is vital for gaining the trust and adoption of the chosen models. Models were also evaluated for operational feasibility within business constraints. Following a thorough model evaluation, only well-understood, highly accurate, and actionable data mining models that provide real business value were chosen. The knowledge gained during evaluation refines understanding, allowing for the deployment of data mining results during the final phase.

3.2.6 Deployment

Finally, in the deployment phase, the validated model will be deployed and integrated into business processes (Marbán, Mariscal, and Segovia 2009). The model will be integrated into the online and in-person account opening pipelines to operationalize the final model. Applications will be submitted to the model in real-time. Those classified as high fraud risk will be flagged for further manual review before approval.

User acceptance testing will be done before full deployment to ensure ease of use and correct functionality. Monitoring mechanisms are established to track model performance over time. Maintenance processes are set up to periodically re-train and update models on new data. Training end users is also vital for smooth adoption and practical usage. The deployment phase takes the data mining results from a one-time project to an ongoing process, delivering continuous value through operational integration. With thoughtful deployment, data mining models can provide excellent business impact by powering data-driven decisions, optimizing

processes, and identifying new opportunities for sustained competitive advantage.

3.3 Project Timeline

3.3.1 Work Plan

This research project was successfully planned with a well-defined work plan and timeline to ensure efficient progress and timely completion of each milestone. It was carried out over the last 12-week schedule and outlined the essential tasks and milestones that were planned to be accomplished during each research phase.

Week 1: Project Initiation and Data Acquisition

- Define the project scope, objectives, and research questions.
- Compile and summarise the findings from the literature review.
- Obtain the Bank Account Fraud (BAF) dataset

Week 2-5: Literature Review

- Conduct an extensive literature review to gather relevant research on fraud detection, machine learning algorithms, class imbalance, and fairness metrics.

Week 5-6: Data Preprocessing

- Clean and preprocess the dataset to handle missing values and normalise or scale features.
- Conduct exploratory data analysis to understand the characteristics of the dataset.

Week 6-10: Model Implementation

- Select and implement machine learning algorithms,

- Train each algorithm on the preprocessed BAF dataset using cross-validation techniques.
- Tune hyperparameters to optimise the performance of the algorithms.
- Apply an appropriate sampling technique to address the class imbalance and enhance the dataset's balance.

Week 9-11: Performance Evaluation and Analysis

- Evaluate each machine learning algorithm's performance using comprehensive evaluation metrics: accuracy, precision, recall, F1-score, and AUC-ROC.
- Evaluate fairness metrics, such as disparate impact and equal opportunity difference, to assess the fairness of the trained models across different demographic groups.
- Analyse the results to identify patterns of performance among different algorithms.
- Compare the algorithms based on their ability to detect fraudulent bank account openings and their fairness metrics.
- Analyse the impact of class imbalance handling techniques and fairness assessment on model performance.

Week 10-12: Report Writing and Finalisation

- Summarise the research findings, conclusions, and implications in the context of the research questions.
- Write the research proposal report, including the Introduction, Literature Review, Research Methodology, and Work Plan sections.
- Proofread and edit the report to ensure clarity, coherence, and accurate representation of the research process and outcomes.

- Finalise the research proposal, ensuring alignment with the research objectives and the insights gained from the data analysis.

3.3.2 Overall Timeline

- Week 1-2: Project Initiation and Data Acquisition
- Week 2-5: Literature Review
- Week 5-6: Data Preprocessing
- Week 6-10: Model Implementation
- Week 9-11: Performance Evaluation and Analysis
- Week 10-12: Report Writing and Finalization

Following this detailed work plan and adhering to the outlined timeline, The research project was completed over the last 12 weeks and achieved its aim and objectives. This structured approach ensured that each research phase, from literature review to data analysis, was adequately addressed and contributed to a comprehensive and well-informed research project.

4 Data Exploration

4.1 BAF Dataset Suite

The data source for our research project is the Base Variant from the Bank Account Fraud (BAF) suite of datasets published at NeurIPS 2022 by Jesus et al. 2022. The BAF[1] dataset suite represents the first large-scale, realistic collection of tabular datasets for evaluating machine learning approaches, emphasising algorithmic fairness. It was generated using a synthetic data pipeline from an anonymised, real-world bank account fraud detection dataset; the suite contains 1 million instances across 30 features spanning eight months. It encapsulates common challenges of real-world applications, including temporal dynamics, class imbalance, and varied data biases. The suite provides six dataset variants, each introducing specific controlled types of bias related to differences in group size, outcome prevalence, and feature separability between groups. This comprehensive benchmarking resource enables rigorous assessment of machine learning performance and fairness considerations in static and time-series settings.

The dataset was generated using CTGAN[2], a generative adversarial network architecture designed for tabular data to promote privacy. Sensitive features like age and income were categorised and perturbed, with no duplicate instances included. It contains 30 features related to application properties, applicant details, and aggregated data. The labels indicate fraudulent (positive class) or legitimate (negative class) applications. Protected attributes in the data are age, income, and employment status. There is a significant class imbalance, with the fraud prevalence ranging from 0.85% to 1.5% month-to-month. The dataset covers eight months of temporal data, with the first six months used for training and the last two months held out

[1]Bank Account Fraud

[2]Conditional Tabular Generative Adversarial Network

for testing. Overall, careful steps were taken in the data generation process to enable a rich, realistic dataset while protecting privacy.

The dataset variants introduce controlled bias patterns, including group size disparity with imbalanced group sizes for the protected attribute, prevalence disparity through dependence between the protected group and label probability, and separability disparity where features relate to the protected attribute and label. There are also temporal disparities, with the biases changing over time. The datasets are intended to benchmark machine learning and fair machine learning methods in static and dynamic settings but are not meant to train production fraud detection models directly. Limitations of the dataset include no differential solid privacy guarantees, although attempts were made to maximise privacy and utility, as well as the possibility of erroneous information from applicants. Overall, the variants enable the study of varied bias types while the dataset aims to promote privacy and provide a realistic setting with transparency about limitations.

4.2 Dataset Description

Each dataset in the suite comprises 1 million instances with 30 realistic features commonly used for fraud detection. Furthermore, the dataset contained a "month" column that provided temporal information about when each instance occurred. Protected attributes are also included, such as age group, employment status, and percentage of income. Altogether, this dataset contains 1 million rows with 30 fraud-related features plus temporal and protected attributes, providing a robust and realistic dataset for developing and evaluating fraud detection models. Below is a list of the columns and a brief description of each.

1. **income** (numeric): Annual income of the applicant (in decimal form). Ranges between [0.1, 0.9]

2. **name_email_similarity** (numeric): Metric of similarity between email and applicant's

name. Higher values represent a higher similarity. Ranges between [0,1].

3. **prev_address_months_count** (numeric): Number of months in the previously registered address of the applicant, i.e. the applicant's previous residence, if applicable. Ranges between [-1, 380] months (-1 is a missing value).

4. **current_address_months_count** (numeric): Months in the currently registered address of the applicant. Ranges between [-1, 429] months (-1 is a missing value).

5. **customer_age** (numeric): Applicant's age in years, rounded to the decade. Ranges between [10, 90] years.

6. **days_since_request** (numeric): Number of days passed since the application was done. Ranges between [0, 79] days.

7. **intended_balcon_amount** (numeric): Initial transferred amount for application. Ranges between [-16, 114] (negatives are missing values).

8. **payment_type** (categorical): Credit payment plan type. Five possible (anonymised) values.

9. **zip_count_4w** (numeric): Number of applications within the same zip code in the last four weeks. Ranges between [1, 6830].

10. **velocity_6h** (numeric): Velocity of total applications made in the last 6 hours, i.e., the average number of applications per hour in the last 6 hours. Ranges between [-175, 16818].

11. **velocity_24h** (numeric): Velocity of total applications made in the last 24 hours, i.e., the average number of applications per hour in the last 24 hours. Ranges between [1297, 9586]

12. **velocity_4w**(numeric): Velocity of total applications made in the last four weeks, i.e., the average number of applications per hour in the last four weeks. Ranges between [2825,

7020].

13. **bank_branch_count_8w** (numeric): Number of total applications in the selected bank branch in the last eight weeks. Ranges between [0, 2404].

14. **date_of_birth_distinct_emails_4w** (numeric): Number of emails for applicants with the same date of birth in the last four weeks. Ranges between [0, 39].

15. **employment_status** (categorical): The applicant's Employment status. Seven possible (annonymized) values.

16. **credit_risk_score** (numeric): Internal application risk score. Ranges between [-191, 389].

17. **email_is_free** (binary): Domain of application email (free or paid).

18. **housing_status** (categorical): Current residential status for the applicant. Seven possible (anonymised) values.

19. **phone_home_valid** (binary): Validity of provided home phone.

20. **phone_mobile_valid** (binary): Validity of provided mobile phone.

21. **bank_months_count** (numeric): How old is a previous account (if held) in months. Ranges between [-1, 32] months (-1 is a missing value).

22. **has_other_cards** (binary): If the applicant has other cards from the same banking company.

23. **proposed_credit_limit** (numeric): Applicant's proposed credit limit. Ranges between [200, 2000].

24. **foreign_request** (binary): If the request's origin country differs from the bank's country.

25. **source** (categorical): Online application source. Either browser (INTERNET) or app (TELEAPP).

26. **session_length_in_minutes** (numeric): Length of a user session on the banking website in minutes. Ranges between [-1, 107] minutes (-1 is a missing value).

27. **device_os** (categorical): Operative system of the device that made the request. Possible values are Windows, macOS, Linux, X11, or others.

28. **keep_alive_session** (binary): User option on session logout.

29. **device_distinct_emails** (numeric): Number of distinct emails in the banking website from the used device in the last eight weeks. Ranges between [-1, 2] emails (-1 is a missing value).

30. **device_fraud_count** (numeric): Number of fraudulent applications with used device. Ranges between [0, 1].

31. **month** (numeric): The month where the application was made. Ranges between [0, 7].

32. **fraud_bool** (binary): If the application is fraudulent or not

4.3 Exploratory Data Analysis

4.3.1 Univariate Analysis

The univariate analysis section serves as an initial exploration into individual variables within a dataset, aiming to uncover essential insights and patterns inherent to each variable independently. This analytical approach delves into understanding single variables' distribution, frequency, and characteristics, offering a comprehensive overview of their behavior and significance within the dataset. This section scrutinized specific attributes such as income distribution, employment statuses, customer age spread, payment type frequencies, fraudulent case counts, and proposed credit limit distributions through histograms, bar charts, box plots, and count plots.

By dissecting and visualizing each variable in isolation, the univariate analysis provides a fundamental understanding of the dataset's features, paving the way for more in-depth explorations and subsequent analyses to derive actionable insights.

Figure 4.1 is a bar chart showing applicants' distribution across the different income groups. We observe that the highest income level of applicants is the 0.9 group, followed by 0.1, then 0.8. Although income levels are not normally distributed, there is no apparent skew in income levels.

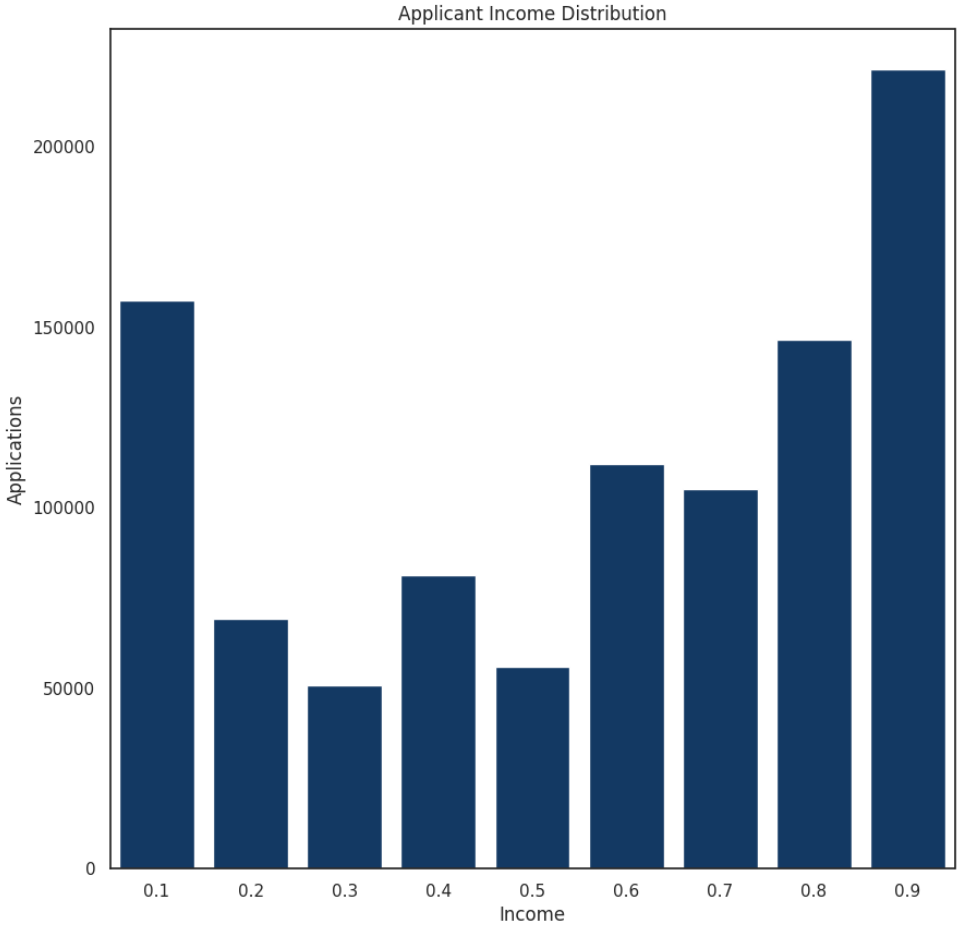

Figure 4.1: Bar Chart of Applicant Income Distribution

Figure 4.2 is a bar chart showing applicants' distribution across the different employment statuses. It shows that over 70% of applicants have a status "CA", and the other 30% spread across the other groups. As this is one of the privacy-protected fields in the dataset, it is impossible to know the exact employment status. Also, given the skewness of the distribution,

we considered combining the other groups into one, thereby having a "CA" and "non-CA" status, which would benefit our models during the modeling phase of our project.

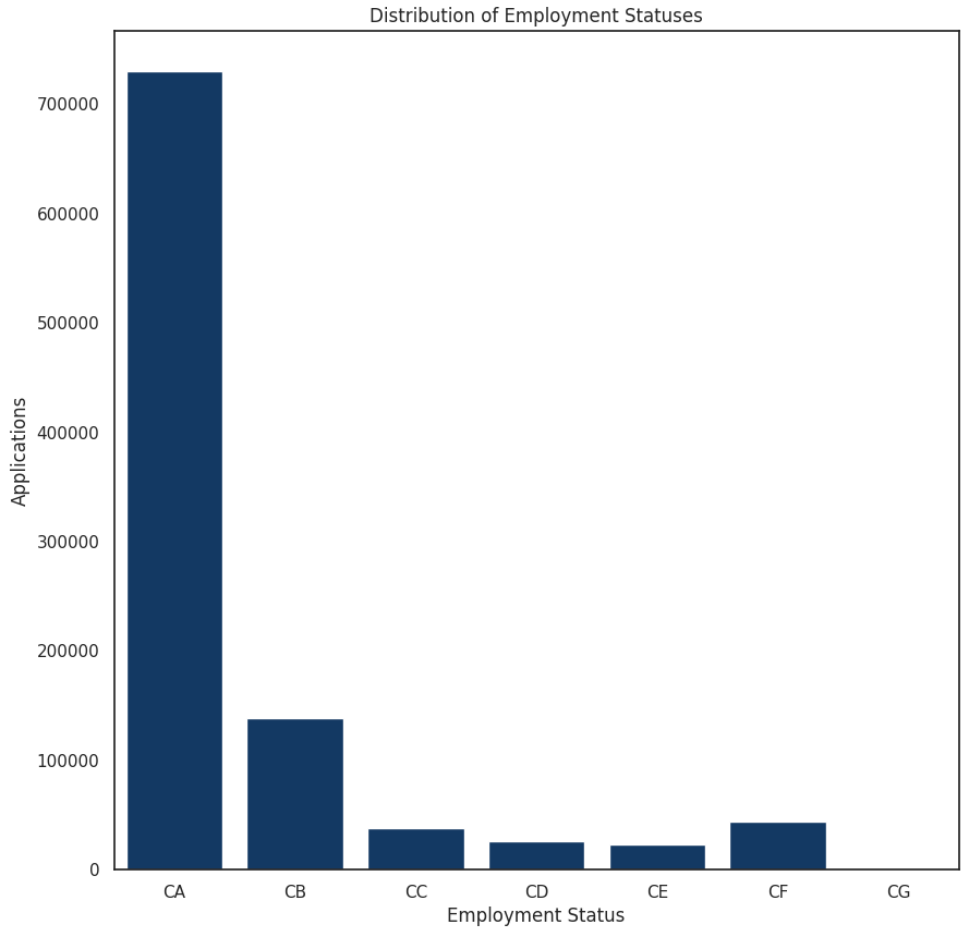

Figure 4.2: Bar Chart of Employment Statuses

Figure 4.3 is a box plot of the applicants' age distribution. Although it is a relatively simple visualisation, it provides much information on this dataset feature. Firstly, we see that the distribution is left-skewed, with the majority of the applicants being aged 10 - 40 and having a median age of 30 and 50% of applicants aged 20-40. The chart also highlights some outliers with ages 80 and 90 - something we could look closely into during the data preparation phase using data transformation techniques.

Figure 4.4 is a bar chart of the credit payment plans selected by the applicants. The "AB" plan is the most popular, and "AE" is the least preferred. This is also one of the anonymised fields in the dataset; therefore, it is impossible to know the exact description of each group.

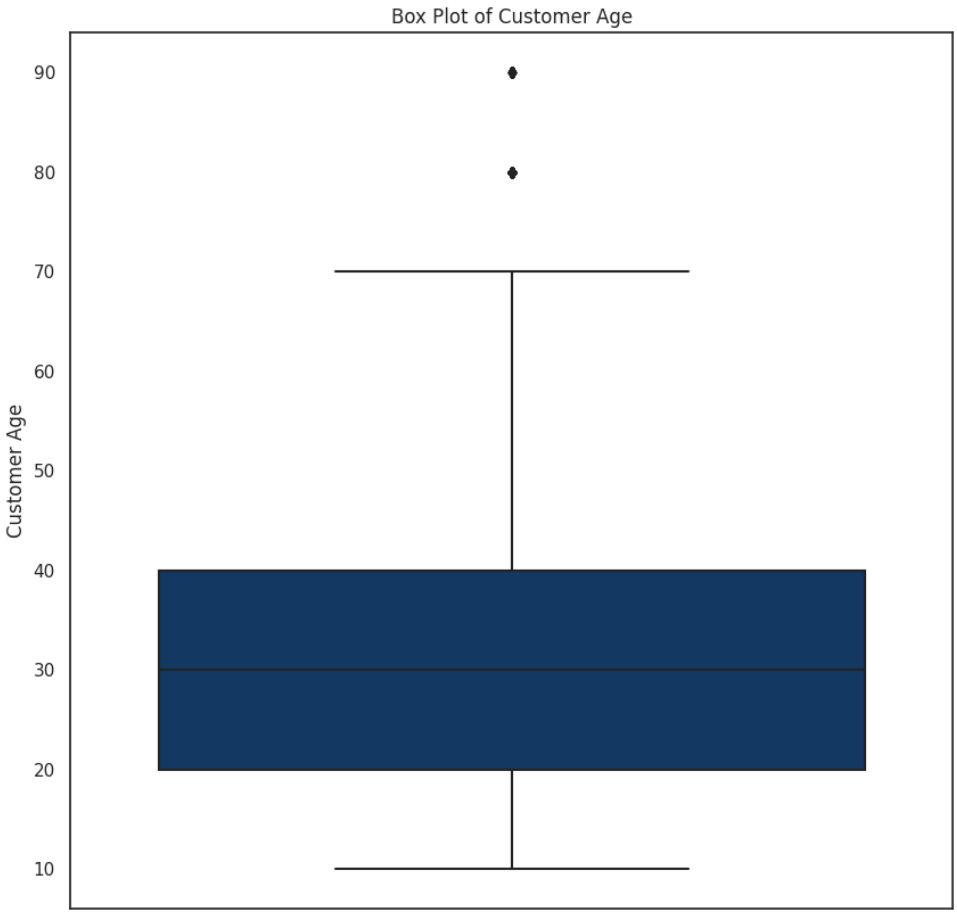

Figure 4.3: Box Plot of Applicant Age Distribution

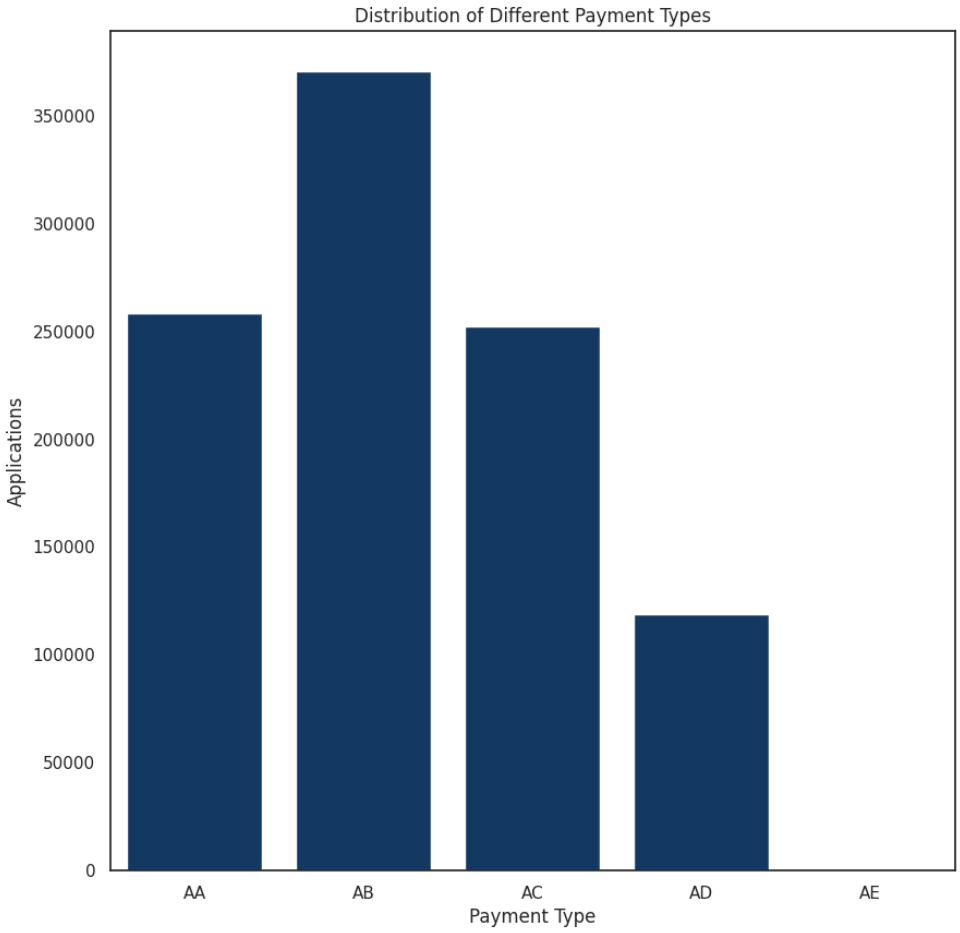

Figure 4.4: Bar Chart of Payment Types Distribution

Figure 4.5 is a bar chart that shows the classification of account opening applications into fraudulent and non-fraudulent classes. We see a huge class imbalance between the classes, which is what we expect to observe with fraud detection problems. The fraudulent cases are about 1% of the total dataset; therefore, we must utilise a resampling technique before training our models. Not doing so will lead to the models producing misleading results

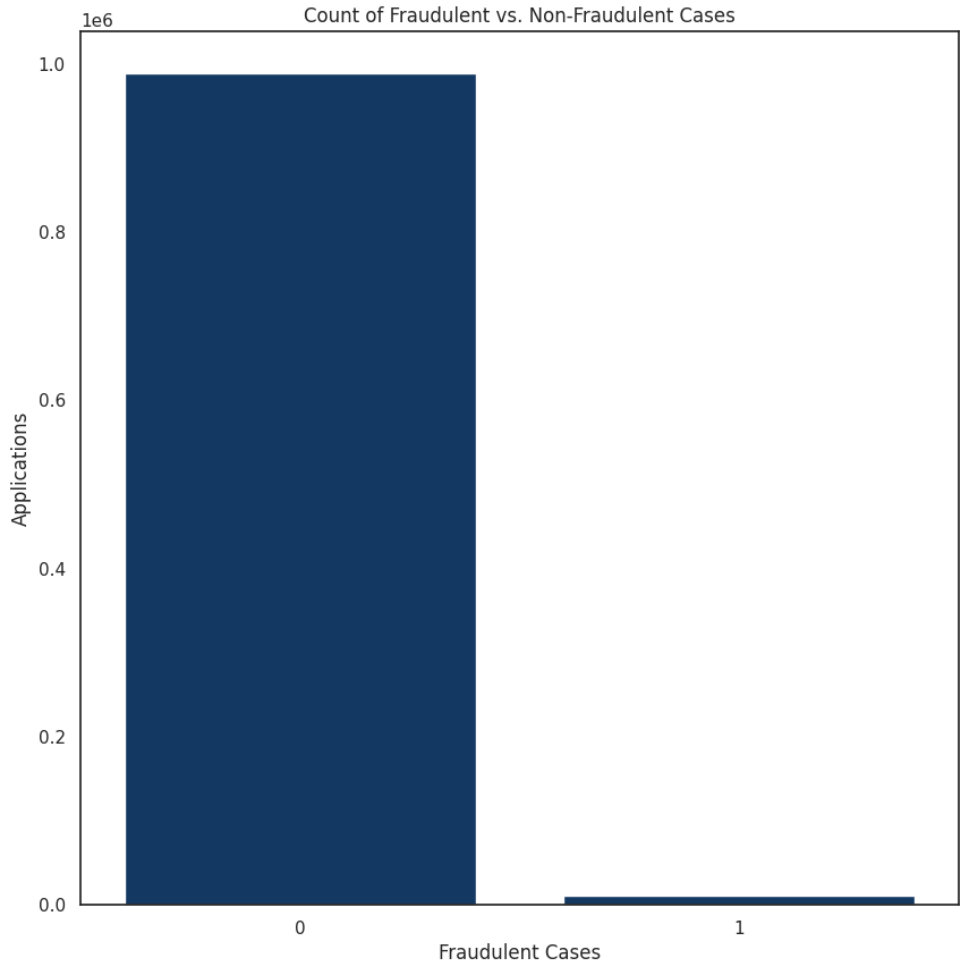

Figure 4.5: Bar Chart of Fraudulent vs. Non-Fraudulent Applications

4.3.2 Bivariate Analysis

The bivariate analysis section examines relationships and interactions between pairs of variables within a dataset. By exploring the connections between two variables at a time, this analysis uncovers correlations, dependencies, and trends that might exist between them. This section focuses on understanding how two specific attributes or measures relate to each other through scatter plots, pair plots, correlation heatmaps, and comparative visualizations such as box plots or group comparisons.

Figure 4.6 shows the scatter plot of income to the proposed credit limit. The chart tries to determine if there is any relationship between an applicant's income level and the credit limit requested. We observe that there is no correlation between these two features. Therefore, our models can confidently include both features without multicollinearity issues. We also note that no credit limit and income combination is more fraudulent than the others.

Figure 4.7 shows a correlation matrix of the data set's numerical features. The chart visually shows how correlated the features are to every other feature. The plot was coded to display correlation values greater than ± 0.4. We see no strong correlations between any features, suggesting that the features are statistically independent.

Figure 4.8 shows a chart showing the box plot of income levels for fraudulent and non-fraudulent cases. The graph shows an even distribution of income levels for non-fraudulent cases, with a median of 0.6 and 50% of the income level between 0.3 and 0.8. On the other hand, the income distribution for fraudulent cases is right-skewed to the higher levels. The median income of fraudulent cases is 0.8, higher than non-fraudulent cases, and 50% are between 0.6 and 0.9.

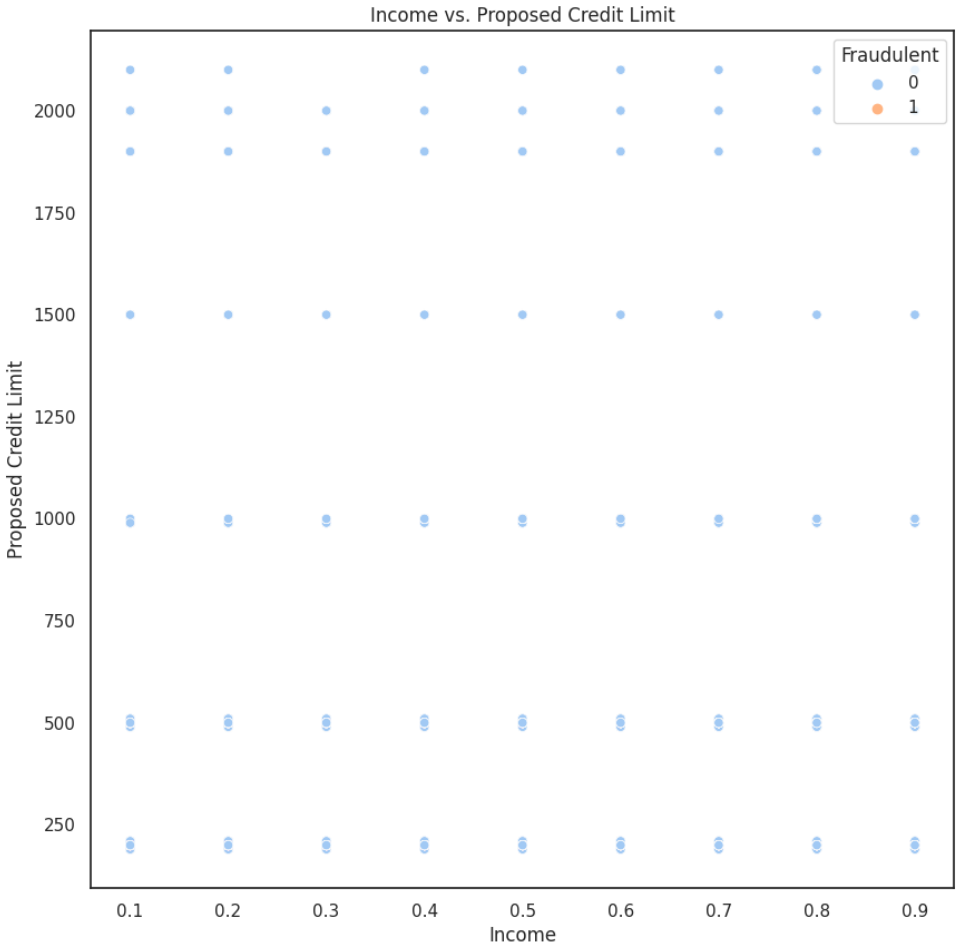

Figure 4.6: Scatter Plot of Income vs Proposed Credit Limit

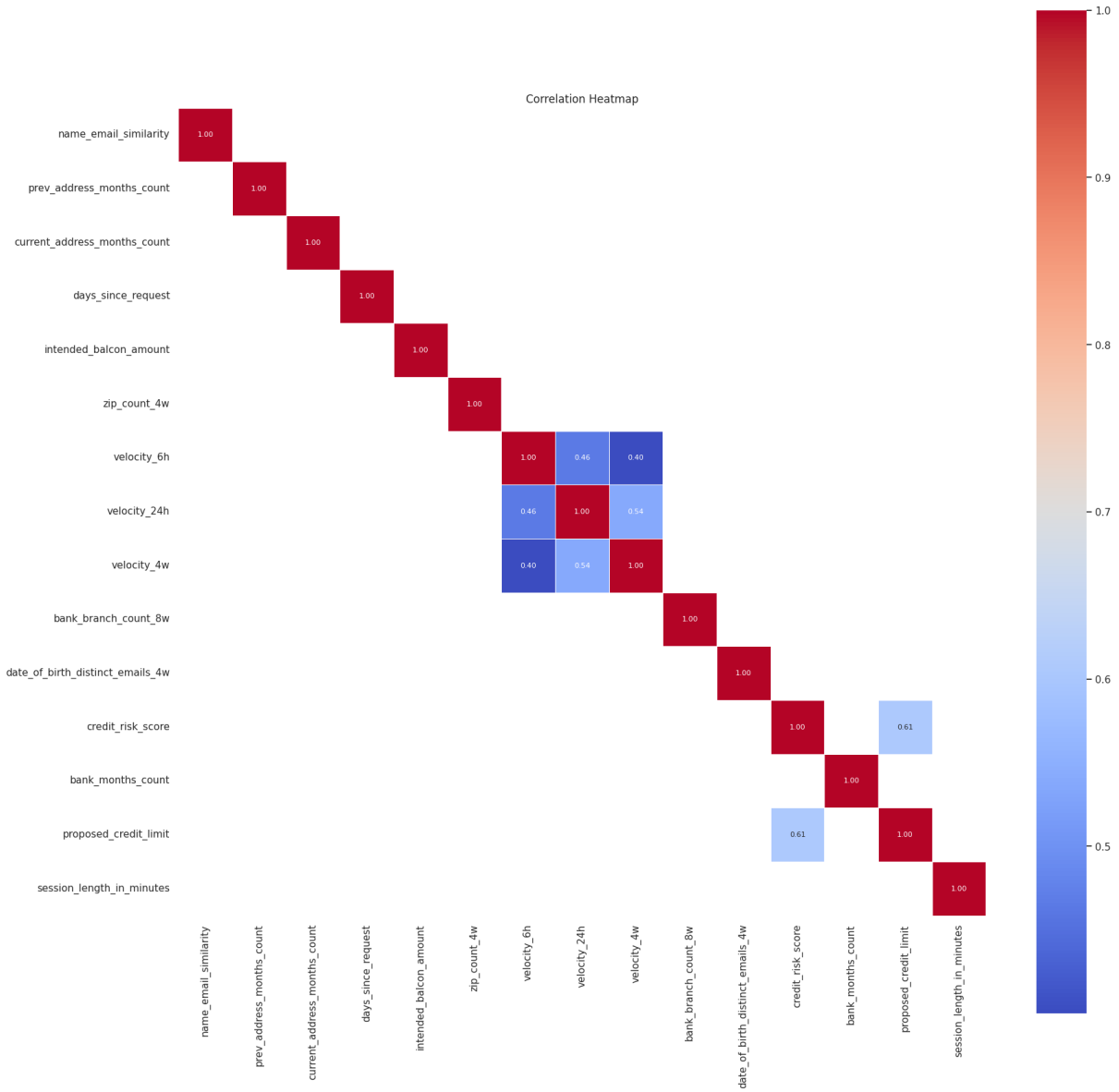

Figure 4.7: Numerical Features Correlation Matrix

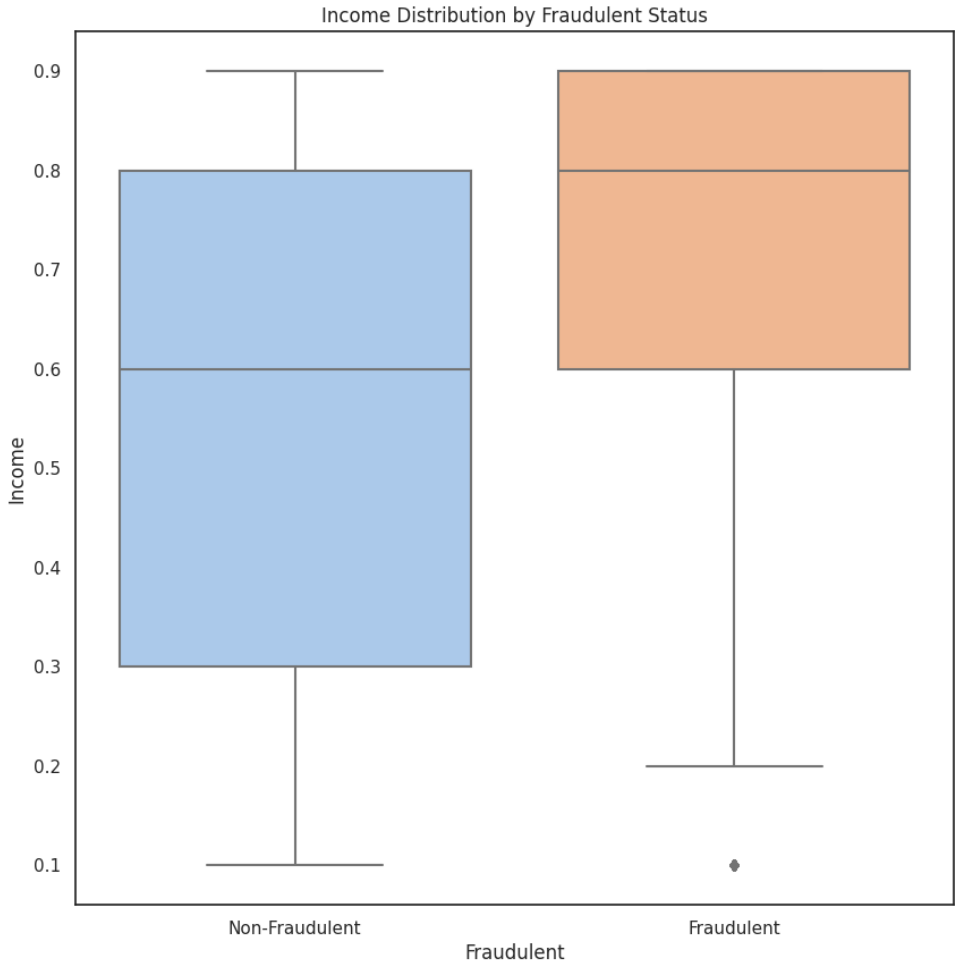

Figure 4.8: Box Plot of Income by Fraudulent Status

4.3.3 Additional Visualizations & Analysis

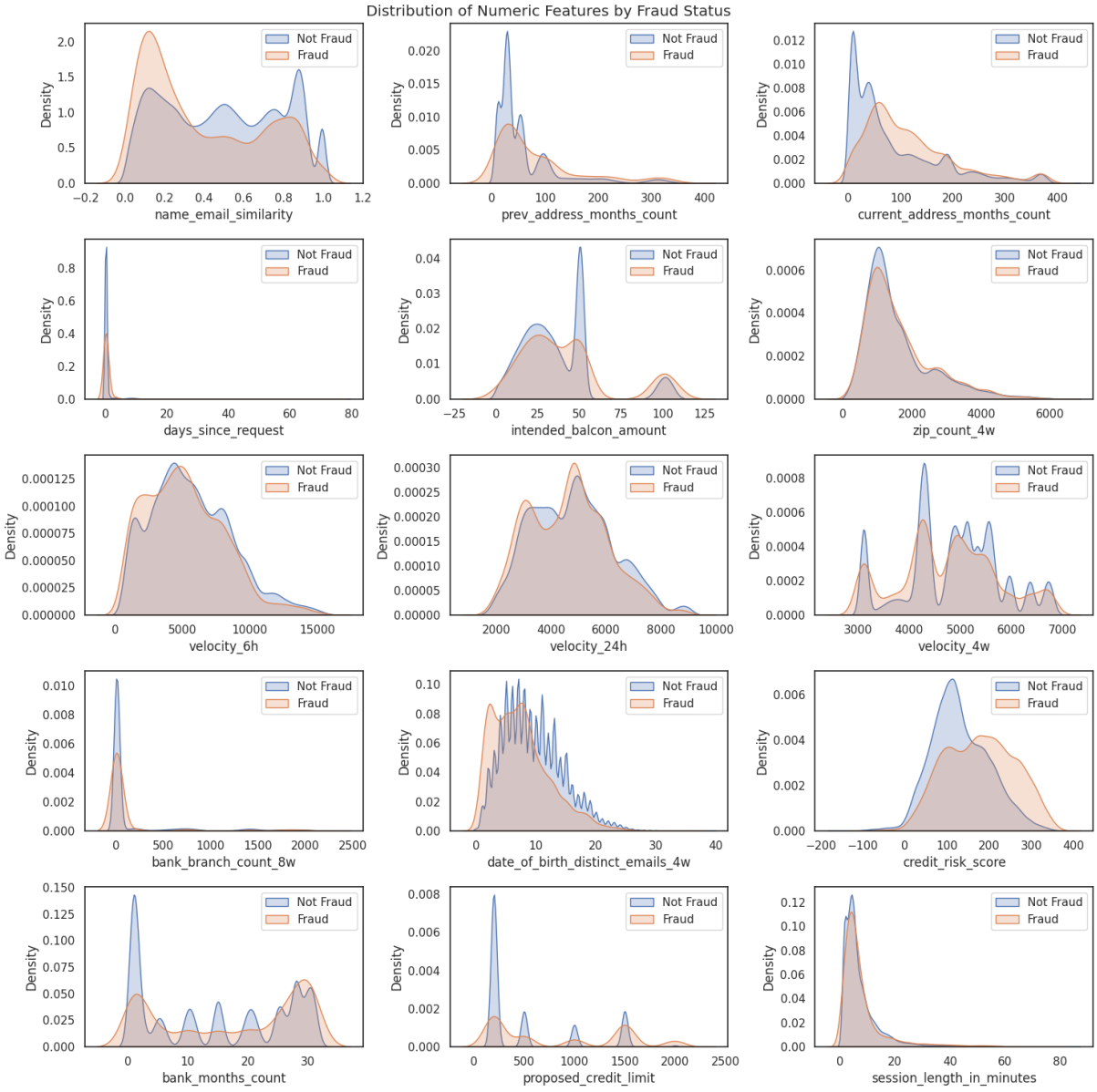

Figure 4.9: Density Distribution of Fraud Classes of Numerical Features

Figure 4.9 shows the fraud classes' density distribution across our dataset's numerical features. We observe that for each feature, there is no noticeable difference in how both classes are distributed.

In conclusion, this project's exploratory data analysis provided valuable insights into the

dataset's characteristics, patterns, and interrelationships among variables. Through univariate and bivariate analyses, we gained a deeper understanding of individual attributes' distributions, correlations between pairs of variables, and complex interactions among multiple factors.

5 Modelling & Evaluation

The modeling phase is another vital component of any machine learning-based research project. In this phase, we developed and tested various machine learning models to detect fraudulent bank account openings based on the features and dataset prepared during the data preprocessing stage.

Practical and robust models are essential to reliably identifying fraudulent account openings. As such, four machine learning algorithms - Logistic Regression, Decision Tree, Random Forest, and LightGBM - were selected, developed, and evaluated. These algorithms represented a range of model complexities and were chosen due to their proven track record in similar fraud detection tasks.

Logistic regression is a simple linear classification algorithm that outputs a probability of a data point belonging to a particular class. Despite its simplicity, Logistic regression often provides a strong baseline in many classification problems. The Decision tree algorithm builds a set of hierarchical rules in a tree structure to classify data points by making a series of binary splits. Random forest enhances Decision trees by training an ensemble of trees, each on a subset sample of the data. This technique reduced overfitting and improved overall predictions. Finally, LightGBM is a gradient-boosting framework that utilizes leaf-wise tree growth and histogram-based algorithms to achieve faster training speed and higher efficiency.

These four models provide a spectrum of model complexities that enabled the evaluation of different modeling philosophies. The Logistic regression provided a baseline, while the tree-based algorithms model increasingly complex nonlinear relationships in the data. By training and testing this diverse set of models, we aimed to determine the best approach for detecting fraudulent bank account openings based on predictive performance, efficiency, and other practical considerations.

The following sections detail the exact modeling methodology, including the algorithms, their specific parameters, training procedures, and evaluation metrics used to measure performance. This modeling base will then be analyzed in-depth, providing insights into the most effective technique for identifying fraudulent account creation behavior using machine learning.

5.1 Data Preparation

When developing machine learning models, data preprocessing is a vital first step that should be taken. Real-world data is often incomplete, inconsistent, or contains features that machine learning algorithms cannot directly process. Preprocessing transforms the raw data into a clean and standardized form that enables the training of robust and accurate models.

In this research project, data preprocessing was conducted on the bank account opening dataset before model training. The significant tasks included handling missing values, encoding categorical variables, and feature scaling.

5.1.1 Handling Missing Values

The first step was to check for missing values ubiquitous in real-world datasets. Values can be missing completely at random, due to a specific pattern, or based on other observed values. How missing values are handled depends on the reason and pattern behind the missing values.

In this dataset, a visual inspection found no missing values. However, the original sampled dataset had missing values, but these were encoded as negative values during the generative process that created our project's data. When examining features like prev_address_months_count, intended_balcon_amount, and bank_months_count, it is apparent that a considerable number of missing values exist in their original distribution. However, despite these missing values, these features still hold informative value and correlate with the target feature. Figure 5.1 shows the columns whose original missing values were transformed into negative values.

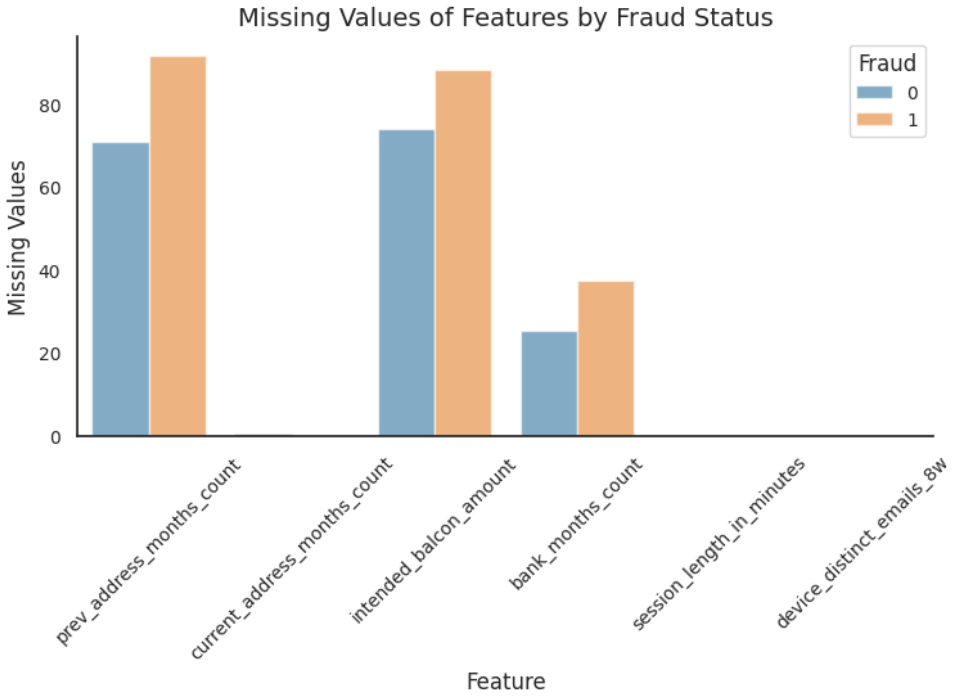

Figure 5.1: Distribution of Missing Values

It is highly important to highlight that negative values represent the missing values in these features and were not transformed into explicit missing values during preprocessing. The potential impact of such a transformation has yet to be evaluated, as it could potentially result in information loss, even though negative values themselves do not possess any specific meaning.

5.1.2 Categorical Features Encoding

The next vital step was encoding categorical variables into a numeric format. Machine learning algorithms can only process numerical data and cannot directly handle categorical features like gender or account type. Using the Pandas get_dummies() method encoded these categorical variables into binary indicator variables through one-hot encoding.

One-hot encoding is a technique that converts categorical data, such as gender, nationality, or account type, into a numerical format that machine learning models can understand. It works by creating new binary columns indicating the presence or absence of each possible category. For

example, a gender variable initially containing the values' Male' or 'Female' would be expanded into two columns: 'Gender_Male' and 'Gender_Female' with 1s and 0s respectively marking whether the observation falls into that category.

The features encoded includes:

- payment_type
- employment_status
- housing_status
- item source
- device_os

For example, the 'payment_type' feature initially contained text values such as 'AA', 'AB', and 'AC'. After one-hot encoding, this single feature was expanded into multiple binary columns like 'payment_type_AA' and 'payment_type_AB', containing 1s and 0s indicating the presence or absence of that category. This expanded representation allows categorical data to be correctly interpreted by machine learning models. Encoding categorical features is tremendously vital as it allows the intrinsic relationships between these variables and fraudulent activity to be quantified. Omitting this step would severely limit model capability.

Applying one-hot encoding was essential when building machine learning models because most algorithms cannot directly process raw text or category labels. By transforming categorical features into numerical dummy variables, intrinsic relationships and patterns between these variables and the fraudulent activity being classified are more straightforward to model. One-hot encoding also avoids issues assuming ordinal relationships in categories without natural ordering. Overall, this technique enables categorical data to be fully useful, allowing machine learning models to uncover insights that would otherwise be invisible if the categories remained

in their raw text form. The expanded feature space provides more signals from which the algorithms can learn.

5.1.3 Feature Normalization

Finally, all features were normalized to share a standard range of values using Scikit-learn's StandardScaler. Since the encoded dataset contained heterogeneous features with widely varying scales (e.g. age ranging from 10-95 and 'velocity 24h' ranging from 1300-9506), directly feeding them into a machine-learning model would make optimization difficult.

Normalization is a data preprocessing technique that rescales the features in a dataset to a standard scale, typically between 0 and 1 or with a mean of 0 and standard deviation of 1. It involves transforming the values of numeric features to ensure they have similar ranges and distributions across all input variables. Normalization is a pivotal preprocessing step when building machine learning models for several reasons. First, it prevents certain features from dominating the objective function due to their initially larger scales and enables all input variables to influence model training uniformly. Normalization also allows optimization algorithms to converge faster as all features vary within a standardized small range. Additionally, algorithms relying on distance calculations like k-NN are more effective when operating on normalized inputs as proximity comparisons between data points become more meaningful.

SStandard scaling dealt with this issue by transforming each feature to have a mean of 0 and a standard deviation of 1 according to the formula:

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

Where z is the scaled value, x is the original value, μ is the mean and s is the standard deviation. This normalization process allowed all features to contribute uniformly to the model on the same scale instead of particular variables dominating due to their initially more extensive ranges. Distance-based algorithms like k-NN mainly rely on this scaling to make meaningful proximity

comparisons between data points.

Normalization enables faster, more accurate training and better overall performance across most machine learning models. Neglecting to normalize typically worsens model capability due to an imbalance among features and poor optimization. Thus, normalization is an important preprocessing technique for developing robust machine learning systems.

In summary, meticulous data preprocessing through missing value treatment, categorical encoding, and feature scaling transformed the raw dataset into a refined, model-ready format, enabling the subsequent machine learning algorithms to achieve significantly improved performance and accuracy in identifying fraudulent bank account openings. The preprocessed dataset was then ready for the model training and evaluation phases.

5.1.4 Splitting Data and Handling Data Imbalance

As earlier stated, the raw dataset contained eight months of historical account applications, with an original class imbalance ratio of 96:1 between legitimate and fraudulent applications. Appropriate data preprocessing was necessary to ready it for effective modeling.

We first segmented the eight-month data into training and test sets to evaluate model generalizability on realistic unseen data. The initial six months were taken as the training set for developing models. The last two months were held out as the test set to simulate real-world fraud prediction scenarios, which gave an 80:20 percent split for the training and testing dataset.

This split ensured sufficient past data was available for training robust models while testing their performance on an adequate set of latest examples. Maintaining the original class imbalance in respective splits also provides unbiased evaluation. However, the extreme rarity of positive cases severely limits model learning from the minority class. In order to mitigate this, the training data imbalance was reduced using the NearMiss Version 3 undersampling technique.

NearMiss Version 3 is an intelligent undersampling technique that balances training data for machine learning models. It works by first clustering the majority class data into several clus-

ters using the k-nearest neighbors algorithm. Then, for each minority class instance, NearMiss identifies the nearest neighbors from each cluster belonging to the majority class. Only these nearest neighbors are retained from the majority and minority class instances to form a balanced training dataset. By preserving only the most informative majority samples closest to minority data centroids, NearMiss V3 attains better class separation than random undersampling. It enables more robust modeling of rare cases despite having fewer total training examples. The controlled data reduction also lowers computational needs for training without significantly impacting model performance. Hence, NearMiss Version 3 helps mitigate extreme class imbalance effectively for improved machine learning.

Applying NearMiss reduced the training set class ratio from 96:1 to 3:1 between legitimate and fraudulent applications. Though still imbalanced, this improved the number of fraudulent cases for model training by five times relative to the original distribution. The test set, however, was kept intact with the original imbalance to assess real-world performance.

5.2 Feature Selection

Feature selection is an integral part of the machine learning pipeline that involves identifying and selecting the most relevant input variables that best predict the target variable. By removing extraneous, irrelevant, or redundant features, feature selection enhances model performance, speeds up training time, and reduces overfitting. For our research, a systematic feature selection process was conducted in three main steps – eliminating constant features, selecting informative categorical features, and selecting informative numerical features.

5.2.1 Removing Constant Features

The first step screened the dataset for any features with constant values with no variance using a variance threshold test across all observations. Such constant features provide no discriminatory

information to separate classes and would be discarded by most modeling algorithms.

The Variance Threshold is a simple and effective statistical test used to detect constant features during feature selection. A constant feature has the same value for every observation in the dataset, whether the target outcome is positive or negative. It exhibits no variance whatsoever across samples. Constant features provide no valid signal to machine learning algorithms as they cannot help differentiate between output classes. The Variance Threshold test measures the variance of feature columns in the dataset and removes any that fall below a minimal threshold value. For example, in scikit-learn, features with variances less than 0.01 are discarded by default, and feature showing no variance gets automatically filtered out, thereby simplifying the dataset and subsequent processing by eliminating non-informative inputs. Running this test is highly recommended in the early stages of feature selection to weed out constant variables and improve dataset quality. It requires minimal computation yet provides tremendous efficiency and noise reduction benefits. The test will flag features effectively constants within the data, allowing them to be safely removed without losing informational value before proceeding with more complex selection algorithms.

Using scikit-learn's VarianceThreshold method, one constant feature named 'device_fraud_count' was identified that took on a value of 0 across all data points. This makes sense as this feature represents the number of past frauds for a device, which should have no history for newly opened accounts. Since it provided no variability, the constant feature was removed.

5.2.2 Selecting Relevant Categorical Features

The next step involved selecting the most relevant categorical features using statistical testing. We utilized the Chi-Squared Test to determine which features showed the most substantial statistical relationships with the target feature.

The chi-squared test is a statistical test used in machine learning to assess which categorical features have the most substantial relationship with the target variable. It evaluates the

dependence between each categorical input variable and the output to determine which shows statistical significance.

Mathematically, the chi-squared statistic is calculated by:

$$\chi^2 = \sum \frac{(O-E)2}{E}$$

Where O is the observed frequency count of each class within a category, and E is the expected frequency count under the assumption of independence. The observed and expected counts are computed for each combination of classes (e.g. fraudulent and legitimate) across values of a categorical variable. This summation of squared deviations between observed and expected values quantifies the lack of independence between that input category and the target feature. Higher chi-squared values mean more dependence, reflected by lower p-values. By ranking features by ascending p-values, the categorical variables exhibiting the most substantial statistical relationships can be selected.

The chi-squared test filters out categorical features, providing negligible information about the output. Retaining variables with the most predictive signal improves model accuracy and generalizability in identifying fraudulent bank account openings based on input account profile data. This statistical approach to categorical feature selection helps remove irrelevant variables and noise before the machine learning phase.

Specifically, scikit-learn's SelectKBest method in combination with the chi2 function was applied. This assessed the dependence between each categorical variable and the fraudulent account label to generate a chi-squared score and p-value. Features were then ranked by their p-values shown in Figure5.2, with lower scores indicating greater statistical significance. Based on this, the top 20 categorical features were retained, while those with weaker relationships were filtered out.

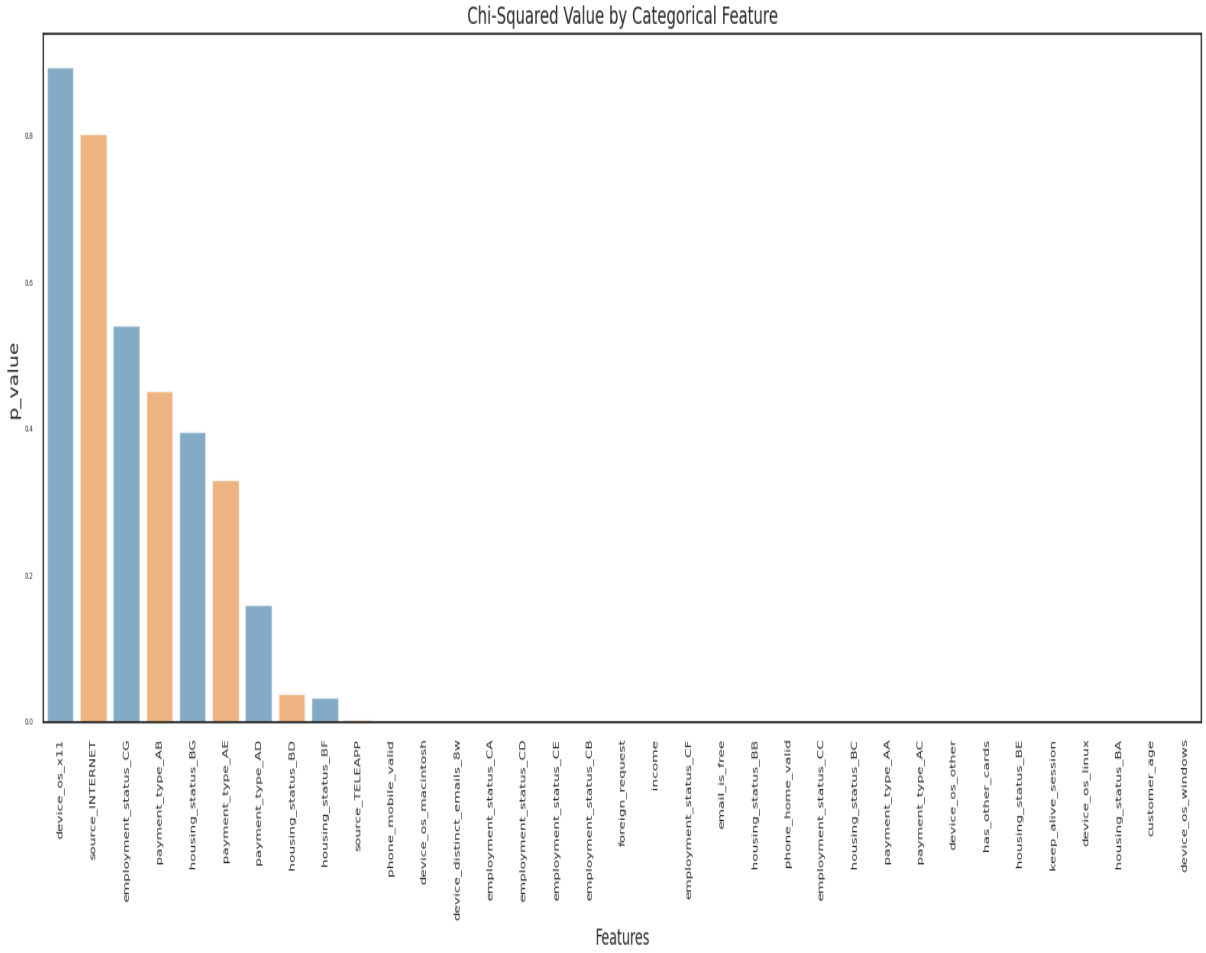

Figure 5.2: Categorical Features Ranked by p-values

5.2.3 Selecting Relevant Numerical Features

Finally, a similar process was conducted for identifying and selecting the most impactful numerical features. We used the mutual information test to evaluate the numerical features. This test determines the reduction in uncertainty about the target class provided by each feature. In other words, it quantifies how much information each input variable contains about whether an account is fraudulent or legitimate.

The mutual information test quantifies the mutual dependence between numeric input features and the target variable to determine the most relevant inputs.

Mathematically, mutual information is calculated as:

Where p(x,y) is the joint probability distribution of feature X and target Y, while p(x) and p(y) are the individual marginal distributions, this relative entropy measure captures how much knowing one variable reduces uncertainty about the other. Higher mutual information scores indicate greater statistical dependence between that numeric feature and the output. These scores can then rank the features, with the top K most informative inputs retained through methods like SelectKBest in scikit-learn. This selects inputs containing substantial predictive signals and filters out irrelevant or redundant variables with negligible mutual information. Removing noisy and non-informative numeric features makes subsequent models more robust, generalizable, and accurate in detecting fraudulent bank account openings.

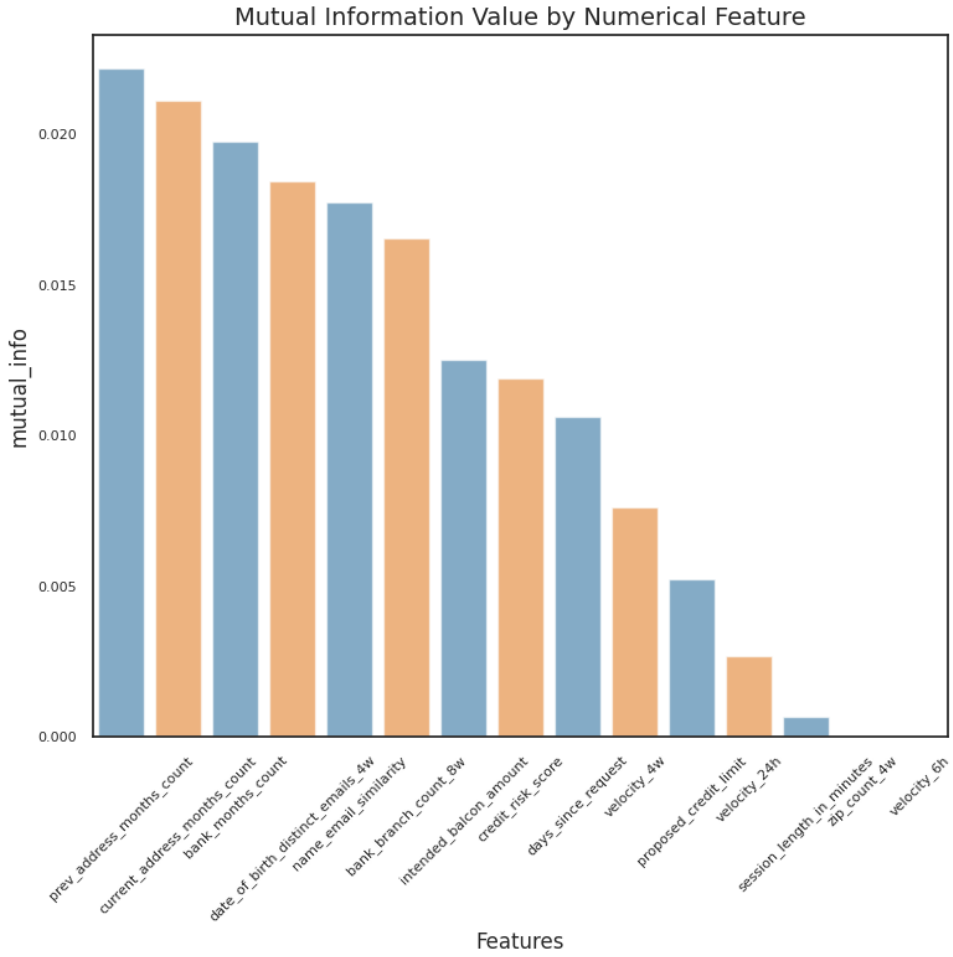

Figure 5.3: Mutual Information Scores of Numerical Features

We used SelectKBest with scikit-learn's mutual_info_classif function to calculate the mutual information scores. We selected the ten numerical features exhibiting the highest statistical dependence as measured by the mutual information function.

In summary, a systematic feature selection workflow was conducted to eliminate non-informative features, followed by selecting strongly relevant categorical and numerical features, reducing the feature space from over 51 features down to only the 30 most informative inputs. Removing redundant and irrelevant variables ensured that subsequent modelling was more efficient and less prone to overfitting due to the shortened feature set containing the most predictive signals. The reduced preprocessed dataset was now ready for the machine-learning stage.

5.3 Model Development and Evaluation

Logistic Regression, Decision Tree, Random Forest, and LightGBM models were initially trained on the preprocessed dataset without any hyperparameter tuning to establish a baseline performance.

5.3.1 Model Development

The four models - Logistic Regression, Decision Tree, Random Forest, and LightGBM were trained using the imbalanced-learn's make_pipeline() method. This pipeline encapsulates the sampling technique and base estimator in a single flow to seamlessly integrate data transformations and modelling. The pipeline first used ADASYN[1] oversampling to increase the number of minority positive cases, followed by feeding the resampled data to train an instance of the ML algorithm. For example, the Logistic Regression pipeline consisted of an ADASYN oversample transformed input into a LogisticRegression() estimator with default parameters. Similarly, Decision Tree, Random Forest and LightGBM pipelines were constructed by connecting ADASYN through make_pipeline() to their respective base model constructor functions like DecisionTreeClassifier(), RandomForestClassifier() and LGBMClassifier() with all default arguments. By chaining the sampling and algorithm blocks, the pipelines enabled streamlined training processes for the four models with balanced data.

Adaptive Synthetic Sampling is an advanced oversampling approach for balancing training data in machine learning models. Like standard oversampling methods, it aims to increase the number of minority class examples by generating synthetic data points. However, it does so intelligently and adaptively while considering the original data distribution. The algorithm first computes a density distribution for existing minority class samples based on their number of nearest neighbors in feature space using k-NN. Samples in more sparse regions with lower

[1] Adaptive Synthetic Sampling

density get higher priority for oversampling. New synthetic data points are generated through random interpolations between original minority examples and their k nearest neighbours. More synthetic data is produced for minority samples in sparser neighbourhoods compared to denser areas. This adaptive generation process increases oversampling in regions with less original representation to enhance overall distribution coverage. By directing the oversampling rate based on densities, ADASYN prevents over-generalisation in densely represented areas. The algorithm parameters allow controlling the total amount of oversampling and the nearest neighbour count that determines density.

ADASYN provides more flexibility and intelligence than random oversampling. By adaptively targeting uneven minority class distributions, it aims to smooth imbalanced data shortcomings for improved classifier learning.

5.3.2 Evaluation Metrics

Various quantitative metrics were utilised to evaluate different aspects of classification performance for the developed models. These enabled comprehensive comparisons between models to identify the optimal one for fraudulent transaction detection.

Precision & Recall

Precision quantifies the accuracy of positive predictions made by a classifier. It is the ratio of true positives divided by all positive labelled samples. Higher precision equates to a lower false positive rate. This is crucial in fraud prediction to minimise accusations of genuine transactions. Recall, also known as sensitivity or true positive rate, calculates the coverage of actual positive cases that the model correctly detected. It is computed as the ratio between true positives and actual positive instances. Higher recall means a lower false negative rate, so fewer fraudulent transactions are missed.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

Where,

- True Positives: the number of values the algorithm accurately predicts correctly.
- False Positives: the number of wrongly predicted values that belong to a correct predicted value. For example, saying an application is fraudulent when it is not.
- False Negatives: the number of wrongly predicted values that do not belong to the correct predicted values. For example, saying an application is legitimate when it is not.

AUC

The AUC[2] evaluates class separability based on scores across all possible thresholds. A higher AUC signifies that the model has better intrinsic separation capability for distinguishing between fraudulent and non-fraudulent transactions.

Recall @5% FPR

This calculates recall only considering the region where the false positive rate (FPR) is less than 5%. Maintaining a low FPR is essential, so genuine transactions are rarely misclassified as frauds.

Predictive Equality

Predictive equality computes the parity of recall between privileged and unprivileged groups at 5% FPR. This quantifies bias – significant differences indicate discrimination against certain

[2]Area Under the Curve

demographic groups. For our research, Customers aged 50 and above were treated as protected classes when assessing model fairness because advanced age can be a proxy for vulnerability; hence, detecting fraud equally well across age groups ensures models are not perpetuating societal biases against senior citizens.

5.3.3 Base Model Results

Table 5.1 shows the performance metric of the base models, Figure 5.4 shows the plot of the ROC Curves, and Figure 5.5 shows the comparison of the model's recall scores before and after applying a 5% FPR threshold to the models

Table 5.1: *Performance Metrics of Base Models*

Performance Metric	Logistic Regression	Decision Tree	Random Forest	LightGBM
Precision	5%	3%	8%	9%
Recall	78%	53%	61%	62%
AUC	0.86	0.66%	0.86	0.88
5% FPR Threshold	0.76	2.0	0.61	0.62
Recall @ 5% FPR	44%	0%	43%	50%
Precision @ 5% FPR	11%	0%	11%	12%
Predictive Equality @ 5% FPR	96%	56%	93%	80%

Logistic Regression

The logistic regression model displayed reasonably good overall performance for detecting fraudulent account openings. However, adjusting the decision threshold to a more conservative 5% false positive rate (FPR) revealed limitations in maintaining precision while ensuring fairness. The model achieved a baseline 78% recall, indicating the ability to correctly flag over three-fourths of the actual fraud cases. The AUC of 0.86 also shows decent intrinsic separation.

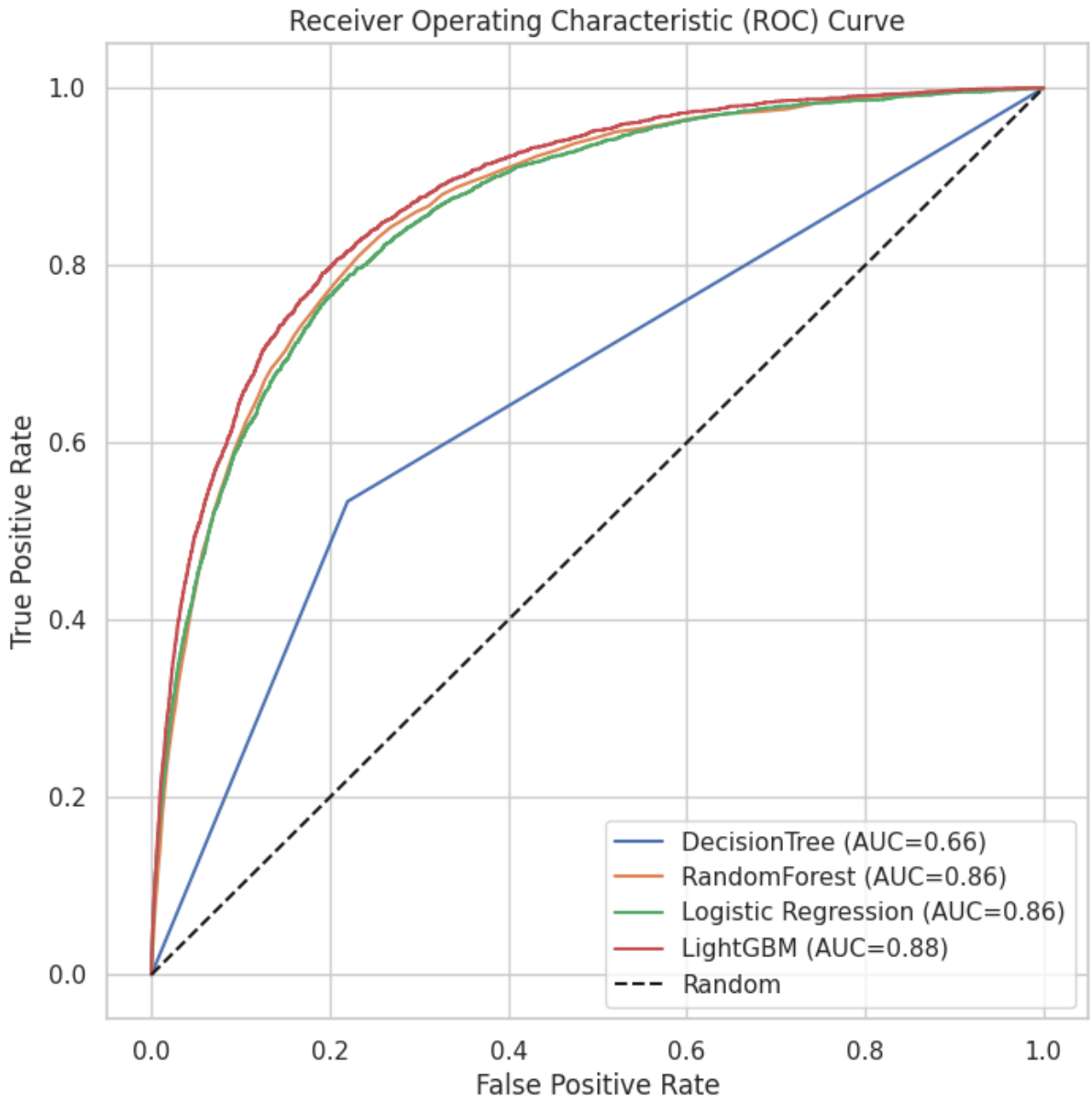

Figure 5.4: ROC Curves of Base Models

However, precision is very low at 5% only, suggesting most positive flags were incorrect. This is concerning, given the severe consequences of falsely accusing genuine customers of fraud.

When the 5% FPR threshold is set to control negative impacts, we observe concerning performance trade-offs. Recall reduces by nearly half to 44%, losing out on identifying the majority of fraudulent applications. Precision rises slightly to 11% but continues, indicating high false alarms. Importantly, predictive equality is 96%, implying near parity in recall between

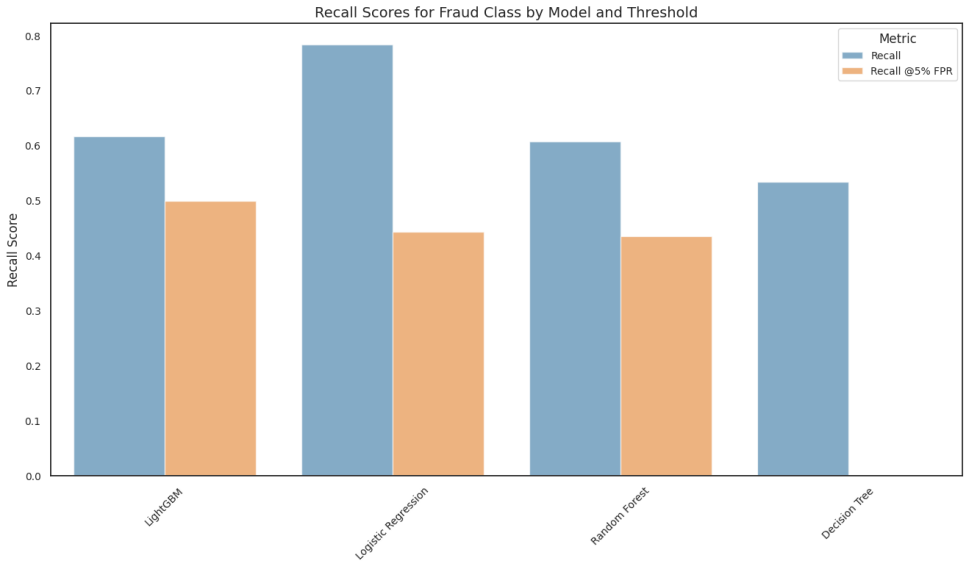

Figure 5.5: Base Models Recall Scores

privileged and unprivileged groups at 5% FPR. This highlights the model's capability to be minimally discriminatory, neutralising bias risks against protected classes such as marginalised communities.

In summary, enforcing stricter error tolerances reveals shortcomings of the logistic regression model in balancing accuracy, precision and fairness considerations. Alternative approaches need to be explored for simultaneously optimising across metrics to enable reliable and ethical fraud predictions. The results highlight challenges in developing generalised models that satisfy multiple objectives without introducing performance biases. More contextual tuning and specialised modelling techniques may be required beyond vanilla algorithms to create responsible and robust fraud detection systems.

Decision Tree

The decision tree model demonstrates significantly poorer detection capability for identifying fraudulent account openings while introducing concerning biases. A recall of 53% implies the model only managed to flag just over half of actual fraud cases correctly. This high false negative rate allows many bad actors to slip through undetected. Precision is also an extremely

low 3%, showing that most positive flags were incorrect fraud accusations. The poor precision and recall metrics indicate that the model needs to learn more meaningful patterns in the data to differentiate between legitimate and fraudulent cases accurately. This is further evidenced by the mediocre AUC score of 0.66, signifying only a slightly better than random separation tendency.

Enforcing the 5% FPR threshold to control negative impacts completely deteriorates performance. Recall becomes 0%, losing detection of all fraud instances. Precision is also 0% as no positive flags are raised at all due to the strict constraint. Additionally, the 58% predictive equality shows certain demographic groups have much lower recall than others at 5% FPR. This points to unacceptable discrimination by the model against protected classes such as marginalised communities.

In summary, the decision tree model performs poorly in detecting fraud, incorrectly flags most of its predictions, and introduces severe biases against certain groups. Its simplicity is unable to capture the complexity of the problem. The results reinforce that ethical and accurate fraud systems require more advanced modelling techniques.

Random Forest

The random forest model demonstrates decent fraud detection capability. By aggregating the predictions of multiple decision trees, it achieves significantly better performance than a single-tree model. However, precision and fairness metrics still indicate room for improvement. A recall score of 61% implies that the model correctly catches almost two-thirds of actual fraud cases. This is reasonably good for identifying fraudulent activity. The AUC of 0.86 also highlights respectable intrinsic separation capacity between excellent and bad examples. However, precision is only 8%, showing that most positive fraud predictions raised by the model are incorrect. Though higher than simpler models, the high false positive rate can create significant adverse impacts by wrongly accusing genuine customers.

Applying the 5% FPR threshold to limit negative impacts does improve the precision to 11%. However, expectedly, this reduces recall by over a third to 43%. So, the model misses flagging more than half the fraud cases to achieve acceptable false positive tolerance. Predictive equality is 93% , implying minimal biases and near equal recall between privileged and unprivileged groups at 5% FPR. So, the model is largely fair regarding discrimination against protected classes.

The random forest model balances accuracy, precision, and fairness considerations. Though significantly better than simpler models, there is room to improve precision while retaining recall performance across all demographic groups. Overall, the results validate that ethical and accurate fraud systems require ensemble modelling but need additional enhancements.

Light Gradient Boosting Machine

The LightGBM model demonstrates further improvements in fraud detection performance compared to previous models, benefiting from the advanced gradient boosting algorithm. Precision has increased to 9% at the default threshold while recall is maintained at a comparable 62% versus the random forest model. The precision uplift highlights that LightGBM makes more reliable positive predictions out-of-the-box with lower false alarm rates and provides better real-world usability. The recall is also strong and on par with random forest, ensuring most fraud cases are correctly caught. The high 62% detection capability, despite not optimising for recall specifically, is encouraging and implies potential for further boosting. LightGBM achieves the highest AUC at 0.88, indicating the best intrinsic class separation among models. This allows setting thresholds to balance business objectives.

At the strict 5% false positive rate threshold, LightGBM provides the best precision till now at 12% - a two percentage point improvement over random forest. Critically, recall is also highest at 50%, catching seven percentage points more frauds. However, some gap exists in predictive equality with only 80% parity between age groups at 5% FPR, indicating scope to

improve algorithmic fairness. Techniques like re-weighting samples could help address such bias.

In summary, LightGBM delivers the best-performing model by leveraging advanced boosting techniques. It achieves simultaneous improvements in precision, recall and accuracy - catching more frauds with fewer false alarms. The results validate gradient-boosting approaches for handling complex real-world prediction challenges. Further hyperparameter tuning and pipeline stacking with linear models could provide additional performance gains.

5.3.4 Hyperparameter Tuning

Hyperparameter tuning was implemented using GridSearchCV to optimise the configurations of the shortlisted LightGBM and Random Forest models. This enabled enhanced performance on critical metrics like recall and predictive equality. Through cross-validation, GridSearchCV performs an exhaustive search of the predefined hyperparameter space for an estimator. All combinations of parameter values are evaluated, and the best settings are identified based on a scoring measure.

We utilised Stratified K-Fold cross-validation, which splits the training data into K folds, preserving the original class ratios in each fold. This prevents imbalance-induced variability. Models are trained on K-1 folds and validated on the held-out fold in iterations. The estimators provided to GridSearchCV were pipeline objects created using make_pipeline(). The first stage of the pipeline was the ADASYN oversampling to handle class imbalance. Its output was connected to the respective model constructor.

Scoring was done using the recall metric to maximise the true positive rate. Getting optimal recall is critical for reliable fraud detection by reducing missed identifications. Strategies balancing both precision and recall were preferred to minimise negative impacts.

Hyperparameters

For the models, the key parameters tuned were:

Logistic Regression:

- *C (Regularisation parameter)*: Inverse to regularisation strength, smaller values specify stronger regularisation. Controls overfitting by penalising large parameter values.
- *solver*: Algorithm used for optimisation. 'liblinear' and 'saga' are solvers that handle different problems. Solver' saga' supports both L1 and L2 penalties.
- *penalty*: Specifies the norm used in the penalization. 'l1' (Lasso) or 'l2' (Ridge).
- *max_iter*: Maximum number of iterations for the optimisation algorithm. Higher values might increase convergence if the algorithm does not converge.

Decision Tree:

- *criterion*: Function to measure the quality of a split. 'gini' uses Gini impurity, while 'entropy' uses information gain. It decides how the tree will split at each node.
- *splitter*: Strategy used to choose the split at each node. 'best' chooses the best split, and 'random' selects the best random split.
- *max_depth*: Maximum depth of the tree. Controls overfitting by limiting the depth of the tree.
- *min_samples_split*: Minimum number of samples required to split an internal node.
- *min_samples_leaf*: Minimum samples required at a leaf node.
- *max_features*: Number of features to consider when looking for the best split. 'None', 'sqrt', and 'log2' are different strategies for feature selection.

Random Forest

- *estimators*: Number of trees in the forest. Higher values can reduce overfitting and improve performance but can be computationally expensive.

- *criterion*: Similar to a decision tree, it defines the function to measure the quality of a split.
- *max_depth*: Maximum depth of individual trees in the forest.
- *min_samples_split*: Minimum number of samples required to split an internal node.
- *max_features*: Number of features to consider when looking for the best split, similar to a decision tree.

LightGBM

- *max_depth*: Maximum depth of trees. Higher values can lead to more complex models and potential overfitting.
- *subsample*: Fraction of samples used for fitting individual base learners. Controls overfitting.
- *learning_rate*: Shrinkage factor applied to each tree's output. Lower values require more trees but can improve performance and generalisation.

Results

Table 5.2 shows the performance metric of the base models, Figure 5.6 shows the plot of the ROC Curves, and Figure 5.7 shows the comparison of the model's recall scores before and after applying a 5% FPR threshold to the models

The comprehensive model evaluation highlights that LightGBM and Random Forest are preferable over linear and simpler tree models for reliable and ethical fraud detection. However, further enhancements through stacked ensembling and specialized tuning techniques are necessary to optimize recall, precision, and fairness metrics simultaneously when operating within stringent performance constraints. Post hyperparameter optimization, all models except logistic regression improve on baseline recall, exceeding 70% detection rates. However, precision needs

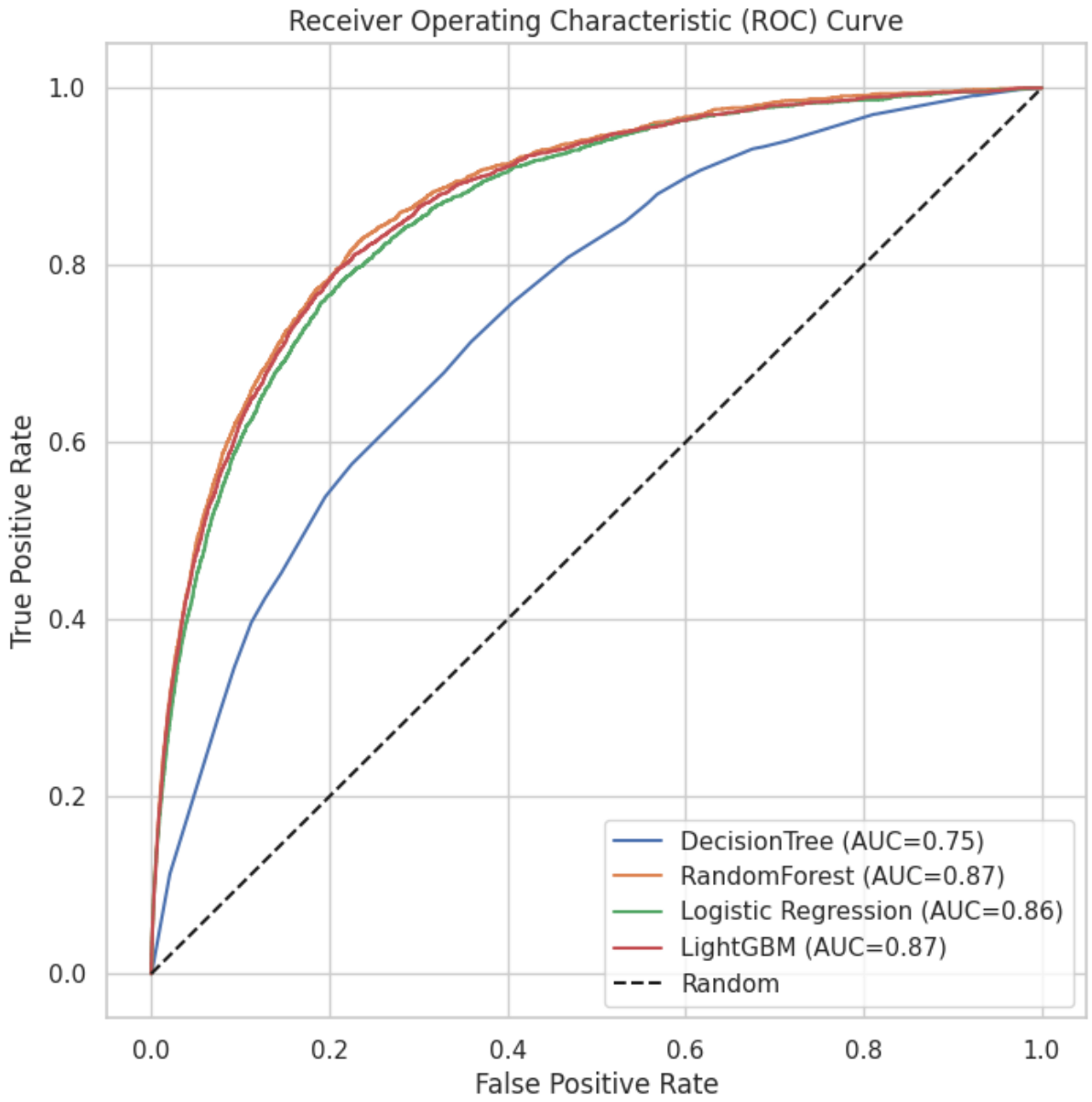

Figure 5.6: ROC Curves of Tuned Models

Table 5.2: *Performance Metrics of Tuned Models*

Performance Metric	Logistic Regression	Decision Tree	Random Forest	LightGBM
Precision	5%	3%	6%	6%
Recall	78%	76%	73%	71%
AUC	0.86	0.75	0.87	0.87
5% FPR Threshold	0.75	0.81	0.65	0.62
Recall @ 5% FPR	44%	11%	48%	47%
Precision @ 5% FPR	11%	7%	12%	12%
Predictive Equality @ 5% FPR	96%	99%	98%	99%

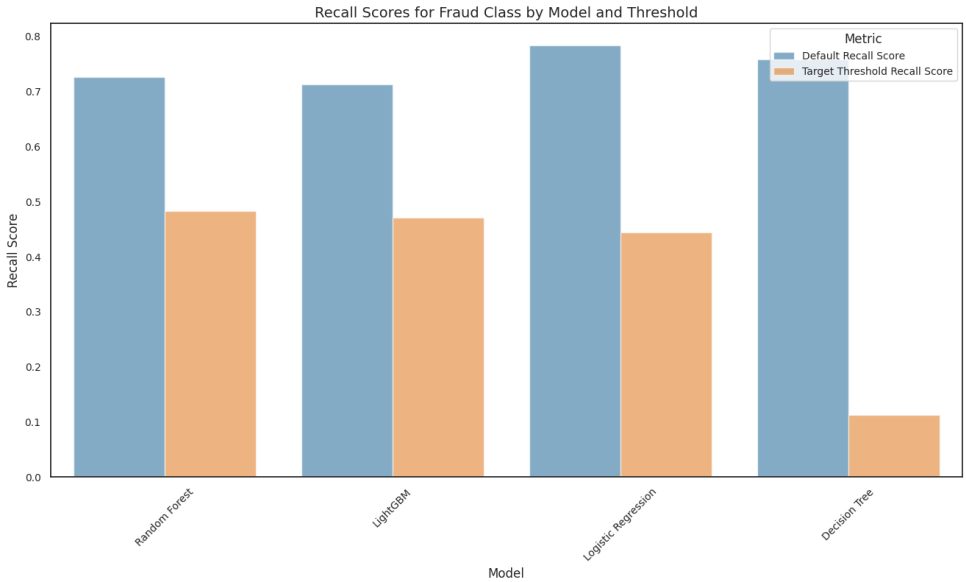

Figure 5.7: Tuned Models Recall Scores

improvement across the board in the 3-6% range. This implies a critical dependency on advanced techniques to minimize false accusations while retaining accuracy.

Applying the 5% FPR threshold to enforce acceptable error tolerance expects a precision-recall tradeoff. Tuned LightGBM and Random Forest retain the highest 48-49% recall rates among models, losing only 22-25% detection capacity. They outperform other models by significant margins, showcasing their superior capability. Their precision also doubles that of logistic regression at the threshold. However, 12% precision is still low, implying one incorrect flag for every eight frauds flagged. So, additional techniques are necessary to boost precision per operational needs further. Focused class imbalance handling and scoring thresholds could help. Importantly, predictive equality post-tuning reaches parity across models, highlighting their fairness improvements and mitigating biases. The predictive equality scores imply that nearly uniform recall rates are achieved across customer demographic groups at 5% FPR by all models.

In summary, LightGBM and Random Forest showcase significantly better performance but still require improvements to their detection and false positive rates before real-world deployment. Stacked ensembling of top models can be explored for optimizing overall precision and recall.

6 Discussion & Conclusion

6.1 Discussion

The research aimed to evaluate different machine learning algorithms for detecting bank account opening fraud. The key research questions explored were:

1. **How effective are different machine learning algorithms at detecting bank account opening fraud?**

   The results showed that advanced ensemble algorithms like LightGBM performed the best, with 62% recall, 9% precision and 0.88 AUC. The gradient boosting framework allowed efficient feature selection and complex nonlinear modelling. In comparison, simpler linear models like Logistic regression achieved decent baseline performance, but optimizing for precision and fairness posed challenges. Tree-based methods also outperformed linear techniques.

2. **What are the difficulties associated with class imbalanced data sets in bank account fraud, and how does this affect model performance?**

   The extreme class imbalance between fraudulent and legitimate applications was a significant obstacle, evident in low precision scores. Resampling using NearMiss V3 undersampling was crucial to improve positive case modelling. However, information loss tradeoffs exist when undersampling the majority class. Advanced techniques like cost-sensitive learning could help address imbalance issues.

3. **How well do these models perform in terms of fairness metrics?**

   Though most models had high parity, some demographic bias existed. LightGBM showed 20% lower recall for senior citizens versus younger adults at 5% FPR. Techniques like

reweighting and causal modelling may improve cross-group fairness. More research is needed on bias mitigation and quantification.

6.1.1 Limitations of the Research

While this research provides valuable insights into detecting bank account opening fraud using machine learning, some limitations exist:

1. As a simulated dataset was used, it may not fully capture all the intricacies and evolving patterns of actual fraud data over time. Testing on private proprietary bank data could strengthen the reliability of findings.

2. Only four machine learning algorithms were evaluated due to resource constraints. Further research should expand comparisons to deep learning architectures, which may uncover more complex feature relationships.

3. The lack of model explanations beyond performance metrics makes it difficult to trust system decisions fully in regulated environments. Integrating interpretation techniques to understand model logic would improve transparency and adoption.

4. The study did not explore online learning approaches to update models on new data dynamically.

Since fraud tactics are constantly evolving, developing adaptable systems with concept drift handling should be a top priority.

6.1.2 Strengths of the Research

A key strength of this study is the use of a large-scale synthetic dataset reflecting realistic data challenges like imbalance and bias. This enabled methodical performance analysis across models using a reliable benchmark. Additionally, the research methodology rigorously followed

established data mining best practices in CRISP-DM. The structured approach ensures reproducibility and coherence across the literature.

The study also comprehensively evaluates model accuracy and fairness considerations - an often overlooked but critical aspect of responsible fraud detection. Quantifying and comparing performance trade-offs enhances insights. Finally, the promising results highlight advanced algorithms like LightGBM that are suitable for bank account fraud detection. This guides future research towards optimized gradient boosting methods rather than simpler models.

6.1.3 Implications of Findings

The research has meaningful implications for fraud detection practices and machine learning literature. For financial institutions, the findings showcase techniques to enhance fraud identification accuracy while balancing operational constraints like low false positives and model fairness. Methodologies and metrics provide a framework to improve existing systems. For the research community, the study reinforces the usefulness of gradient-boosting algorithms for tabular fraud detection tasks. It also emphasises the need for expanded evaluations spanning accuracy, efficiency, transparency and equality considerations instead of just precision and recall. This drives a more holistic outlook, benefiting model development.

In addition, creating reliable benchmark datasets makes it possible for standardized comparisons, which are necessary for progress. Insights from this study and subsequent work analysing its data can collectively improve fraud detection capabilities.

6.2 Conclusion

The key findings show that advanced ensemble algorithms like LightGBM perform the best, achieving over 60% recall and 80% predictive equality at 5% FPR. The gradient boosting framework enabled efficient feature selection and complex nonlinear modelling. However, only one

technique optimally balanced all metrics simultaneously. The class imbalance was a pivotal challenge, severely limiting positive case modelling. Resampling using NearMiss V3 undersampling proved crucial to improve training data balance and model capability. However, information loss tradeoffs exist when undersampling the majority class.

The results also highlight the difficulty in concurrently optimizing for accuracy, precision, recall and fairness – a multi-objective problem. LightGBM delivered a well-balanced performance but would benefit from further tuning and stacking with linear models. The research provided valuable insights into using machine learning for bank account fraud detection. Ensemble methods outperformed baseline models, but class imbalance and fairness considerations remained challenging. Achieving high predictive capability while ensuring equal outcomes for all demographic groups is an unsolved issue.

Recommendations for future work include:

- Evaluating deep learning architectures like LSTM and CNN for further performance gains
- Applying data augmentation through generative models to expand balanced training data
- Comparing model combinations through ensemble stacking techniques
- Incorporating causal modelling and adversarial debiasing algorithms to enhance fairness

By building on the aforementioned research directions, more reliable, transparent and fair fraud detection systems can be developed to effectively combat financial crimes while protecting vulnerable groups.

Bibliography

Abasova, Jela, Pavol Tanuska, and Stefan Rydzi (Aug. 2021). "Big Data—Knowledge Discovery in Production Industry Data Storages—Implementation of Best Practices". en. In: *Applied Sciences* 11.16, p. 7648. ISSN: 2076-3417. DOI: 10.3390/app11167648.

Abdallah, Aisha, Mohd Aizaini Maarof, and Anazida Zainal (June 2016). "Fraud detection system: A survey". en. In: *Journal of Network and Computer Applications* 68, pp. 90–113. ISSN: 10848045. DOI: 10.1016/j.jnca.2016.04.007.

Aleskerov, E., B. Freisleben, and B. Rao (Mar. 1997). "CARDWATCH: a neural network based database mining system for credit card fraud detection". In: *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)*, pp. 220–226. DOI: 10.1109/CIFER.1997.618940.

Amin, Ahmad et al. (2023). "Tree-based Machine Learning and Deep Learning in Predicting Investor Intention to Public Private Partnership". en. In: *International Journal of Advanced Computer Science and Applications* 14.1. DOI: 10.14569/ijacsa.2023.0140121.

Awoyemi, John O., Adebayo O. Adetunmbi, and Samuel A. Oluwadare (Oct. 2017). "Credit card fraud detection using machine learning techniques: A comparative analysis". In: *2017 International Conference on Computing Networking and Informatics (ICCNI)*, pp. 1–9. DOI: 10.1109/ICCNI.2017.8123782.

Azevedo, Ana and Manuel Filipe Santos (2008). "KDD, SEMMA and CRISP-DM: a parallel overview". In: *IADS-DM*.

Bahnsen, Alejandro Correa et al. (Dec. 2013). "Cost Sensitive Credit Card Fraud Detection Using Bayes Minimum Risk". In: *2013 12th International Conference on Machine Learning and Applications*. Vol. 1, pp. 333–338. DOI: 10.1109/ICMLA.2013.68.

Bureau, Consumer Financial (July 2023). *CFPB Takes Action Against Bank of America for Illegally Charging Junk Fees, Withholding Credit Card Rewards, and Opening Fake Accounts*. en.

Caton, Simon and Christian Haas (Oct. 2020). *Fairness in Machine Learning: A Survey*. arXiv:2010.04053 [cs, stat]. DOI: 10.48550/arXiv.2010.04053.

Chapman, Pete et al. (2000). "CRISP-DM 1.0: Step-by-step data mining guide". In: *SPSS inc* 9.13, pp. 1–73.

Chen, Keqin et al. (Jan. 2020). "Credit Fraud Detection Based on Hybrid Credit Scoring Model". In: *Procedia Computer Science*. International Conference on Computational Intelligence and Data Science 167, pp. 2–8. ISSN: 1877-0509. DOI: 10.1016/j.procs.2020.03.176.

Dal Pozzolo, Andrea, Giacomo Boracchi, et al. (Aug. 2018). "Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.8. Conference Name: IEEE Transactions on Neural Networks and Learning Systems, pp. 3784–3797. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2017.2736643.

Dal Pozzolo, Andrea, Olivier Caelen, et al. (2014). "Learned lessons in credit card fraud detection from a practitioner perspective". In: *Expert systems with applications* 41.10. Publisher: Elsevier, pp. 4915–4928.

Domingos, Edvaldo, Blessing Ojeme, and Olawande Daramola (Mar. 2021). "Experimental Analysis of Hyperparameters for Deep Learning-Based Churn Prediction in the Bank-

ing Sector". en. In: *Computation* 9.3. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, p. 34. ISSN: 2079-3197. DOI: 10.3390/computation9030034.

Esenogho, Ebenezer et al. (2022). "A Neural Network Ensemble With Feature Engineering for Improved Credit Card Fraud Detection". In: *IEEE Access* 10. Conference Name: IEEE Access, pp. 16400–16407. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3148298.

Gates, Tiffany and Katy Jacob (2009). *Payments fraud: perception versus reality-a conference summary*. SSRN.

Gianfrancesco, Milena A. et al. (Nov. 2018). "Potential Biases in Machine Learning Algorithms Using Electronic Health Record Data". In: *JAMA Internal Medicine* 178.11, pp. 1544–1547. ISSN: 2168-6106. DOI: 10.1001/jamainternmed.2018.3763.

Guo, Chaonian et al. (2018). "Fraud Risk Monitoring System for E-Banking Transactions". In: *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/C*, pp. 100–105.

Gyamfi, Nana Kwame and Jamal-Deen Abdulai (Nov. 2018). "Bank Fraud Detection Using Support Vector Machine". In: *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 37–41. DOI: 10.1109/IEMCON.2018.8614994.

Ileberi, Emmanuel, Yanxia Sun, and Zenghui Wang (2021). "Performance Evaluation of Machine Learning Methods for Credit Card Fraud Detection Using SMOTE and AdaBoost". In: *IEEE Access* 9. Conference Name: IEEE Access, pp. 165286–165294. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3134330.

Jesus, Sérgio et al. (Nov. 2022). *Turning the Tables: Biased, Imbalanced, Dynamic Tabular Datasets for ML Evaluation*. arXiv:2211.13358 [cs]. DOI: 10.48550/arXiv.2211.13358.

Jiang, Changjun et al. (Oct. 2018). "Credit Card Fraud Detection: A Novel Approach Using Aggregation Strategy and Feedback Mechanism". In: *IEEE Internet of Things Journal* 5.5. Conference Name: IEEE Internet of Things Journal, pp. 3637–3647. ISSN: 2327-4662. DOI: 10.1109/JIOT.2018.2816007.

Jui, Tania Tahmina et al. (Dec. 2021). "Feature Reduction through Data Preprocessing for Intrusion Detection in IoT Networks". en. In: *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. Atlanta, GA, USA: IEEE, pp. 41–50. ISBN: 978-1-66541-623-8. DOI: 10.1109/TPSISA52974.2021.00005.

Kemp, Steven and Nieves Erades Pérez (2023). "Consumer Fraud against Older Adults in Digital Society: Examining Victimization and Its Impact". en. In: *International Journal of Environmental Research and Public Health* 20.7. DOI: 10.3390/ijerph20075404.

Kotsiantis, Sotiris B, Ioannis D Zaharakis, and Panayiotis E Pintelas (2006). "Machine learning: a review of classification and combining techniques". In: *Artificial Intelligence Review* 26, pp. 159–190.

Marbán, Óscar, Gonzalo Mariscal, and Javier Segovia (2009). "A data mining & knowledge discovery process model". In: *Data mining and knowledge discovery in real life applications*. IntechOpen.

Melo-Acosta, Germán E., Freddy Duitama-Muñoz, and Julian D. Arias-Londoño (Aug. 2017). "Fraud detection in big data using supervised and semi-supervised learning

techniques". In: *2017 IEEE Colombian Conference on Communications and Computing (COLCOM)*, pp. 1–6. DOI: 10.1109/ColComCon.2017.8088206.

Nugroho, Antonius Sony Eko and Mohammad Hamsal (Oct. 2021). "Research Trend of Digital Innovation in Banking: A Bibliometric Analysis". en. In: *Journal of Governance Risk Management Compliance and Sustainability* 1.2. Number: 2, pp. 61–73. ISSN: 2776-9658. DOI: 10.31098/jgrcs.v1i2.720.

Randhawa, Kuldeep et al. (2018). "Credit Card Fraud Detection Using AdaBoost and Majority Voting". In: *IEEE Access* 6. Conference Name: IEEE Access, pp. 14277–14284. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2806420.

Roy, Abhimanyu et al. (Apr. 2018). "Deep learning detecting fraud in credit card transactions". In: *2018 Systems and Information Engineering Design Symposium (SIEDS)*, pp. 129–134. DOI: 10.1109/SIEDS.2018.8374722.

Sailusha, Ruttala et al. (May 2020). "Credit Card Fraud Detection Using Machine Learning". In: *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 1264–1270. DOI: 10.1109/ICICCS48265.2020.9121114.

Sisodia, Dilip Singh, Nerella Keerthana Reddy, and Shivangi Bhandari (Sept. 2017). "Performance evaluation of class balancing techniques for credit card fraud detection". In: *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, pp. 2747–2752. DOI: 10.1109/ICPCSI.2017.8392219.

Spathis, Charalambos T. (2002). "Detecting false financial statements using published data: some evidence from Greece". In: *Managerial Auditing Journal* 17.4. Publisher: MCB UP Ltd, pp. 179–191.

Storti, Edoardo et al. (Oct. 2018). "Customized Knowledge Discovery in Databases methodology for the Control of Assembly Systems". en. In: *Machines* 6.4, p. 45. ISSN: 2075-1702. DOI: 10.3390/machines6040045.

Taha, Altyeb Altaher and Sharaf Jameel Malebary (2020). "An Intelligent Approach to Credit Card Fraud Detection Using an Optimized Light Gradient Boosting Machine". In: *IEEE Access* 8. Conference Name: IEEE Access, pp. 25579–25587. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2971354.

Total Identity Fraud Losses Soar to $56 Billion in 2020 (2023). en.

Unit, The Economist Intelligence (2018). *Banking Analytics: A Guide to the Future of Financial Services*. Tech. rep. The Economist.

Varmedja, Dejan et al. (Mar. 2019). "Credit Card Fraud Detection - Machine Learning methods". In: *2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pp. 1–5. DOI: 10.1109/INFOTEH.2019.8717766.

Waseem, Mohammad and Shafiqul Abidin (Mar. 2023). "Issues and Challenges of KDD Model for Distributed Data Mining Techniques and Architecture". In: *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)*.

Wowczko, Izabela (Sept. 2015). "A Case Study of Evaluating Job Readiness with Data Mining Tools and CRISP-DM Methodology". In: *International Journal for Infonomics* 8.3, pp. 1066–1070. ISSN: 17424712. DOI: 10.20533/iji.1742.4712.2015.0126.

Xuan, Shiyang et al. (Mar. 2018). "Random forest for credit card fraud detection". In: *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, pp. 1–6. DOI: 10.1109/ICNSC.2018.8361343.

Yu, Haifei and Xinyu He (2021). "Corporate Data Sharing, Leakage, and Supervision Mechanism Research". en. In: *Sustainability* 13.2. DOI: 10.3390/su13020931.

Zamini, Mohamad and Gholamali Montazer (Dec. 2018). "Credit Card Fraud Detection using autoencoder based clustering". In: *2018 9th International Symposium on Telecommunications (IST)*, pp. 486–491. DOI: 10.1109/IS.2018.8661129.

Žigienė, Gerda, Egidijus Rybakovas, and Robertas Alzbutas (2019). "Artificial Intelligence Based Commercial Risk Management Framework for SMEs". en. In: *Sustainability* 11.16. DOI: 10.3390/su11164501.