UNIVERSITY OF HERTFORDSHIRE
School of Physics, Engineering and Computer Science

MSc Cybersecurity with Advanced Research
7COM1070, Cybersecurity Masters Project
9th December, 2024

PROJECT TITLE: ENHANCING FINANCIAL FRAUD DETECTION USING DEEP LEARNING TECHNIQUES.

Name: Odigie Osolase Oses
Student ID:21071168
Supervisor: Dr. Imran Khan

Acknowledgment

Without the assistance of my supervisor, Dr. Imran Khan, who gave me clear criticism and direction at every stage, this project would not have been completed, and I will always be grateful for that.

I also want to express my gratitude to everyone of the UH personnel, including the academic and administrative staff, but especially to Christopher Treglohan, who was always willing to listen to me and provided me with the academic assistance and guidance I required for my future career in cybersecurity.

In addition, I would want to express my gratitude to my parents and siblings, my mentor Kingsley Oseghale, and classmates Dolapo Oni, Isaac Nzekwe, Omokogie Ogiator and Sai Teja Gujjari. Their tremendous encouragement and support served as the catalyst for my perseverance when it seemed impossible.

Dedication

I would like to specially dedicate this work to my dear friend Mr Odudu Okon, who sadly passed away on the 2nd of May 2023. He was one of my greatest inspirations to pursue this degree and constantly encouraged me in the early days of this journey when I faced a lot of pressure with the rigors of academic work as well as settling down in a new country.

My heartfelt dedication also goes to my loving and supportive wife Goodness Odigie, who has stood by me through the highs and lows of this journey. I do not know how I could have done this without you.

Finally, I am dedicating this to God who has given me life, the strength and courage to pursue this milestone over the last two years. All my efforts, dedication and hard work would amount to nothing without Him.

Declaration

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Cyber Security at the University of Hertfordshire (UH).
It is my own work except where indicated in the report.

I did not use human participants in my MSc Project.
I hereby give permission for the report to be made available on the university website provided the source is acknowledged.

Odigie Osolase Oses

Abstract

Financial transactions have experienced growing complexity due to fraudulent activities. This has dictated the need to advance solutions safeguarding sensitive data and prevent economic losses. This work explores the potential of deep learning models, particularly Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN), in financial fraud detection. Taking advantage of automatic feature extraction and optimization techniques, including SMOTE for class balancing and Adam for gradient optimization, the research highlights the superior accuracy and efficiency of deep learning over traditional methods. While ANN demonstrated powerful performance, CNN excelled in scalability and generalization due to its feature learning capabilities. The study also addresses challenges such as computational costs and model interpretability, proposing future directions to enhance usability, scalability and trust. Therefore, developing optimized models and conducting comparative analyses, this work provides insights for improving fraud detection systems, contributing to the security of financial ecosystems. The findings stress the critical role of deep learning in combating fraud patterns.

Table of Contents

List of Figures

List of Tables

Chapter I: Introduction

1.1 Background to the Study

The development of online banking and e-commerce has increased the volume and complexity of financial transactions, creating greater opportunities for fraud (Zhu et al., 2021). Traditional fraud detection methods, such as rule-based algorithms and statistical models, have struggled to adapt to the evolving strategies of modern fraudsters, needing more advanced solutions (Bolton et al., 2002; Abdallah, Maarof, and Zainal, 2016; Ismaeil, 2024).
Deep learning, a subset of artificial intelligence, offers capabilities in financial fraud detection by applying its ability to process large, complex datasets and identify difficult patterns. It has already proven effective in domains like anomaly detection and has the adaptability to address dynamic fraud trends, making it an effective framework for enhancing detection systems (KampanartHuanbutta et al., 2024; Yang et al., 2024).
The prevalence of cyber-financial crimes, including credit card fraud, identity theft, and account takeovers, stresses the critical need for advanced fraud detection systems. These systems must identify fraudulent activities in real time while minimizing false positives to avoid disrupting legitimate transactions (Arroyabe et al., 2024). With fraud costing businesses an estimated 5% of annual revenue, the economic imperative to develop effective solutions is clear (Hamilton, 2024).

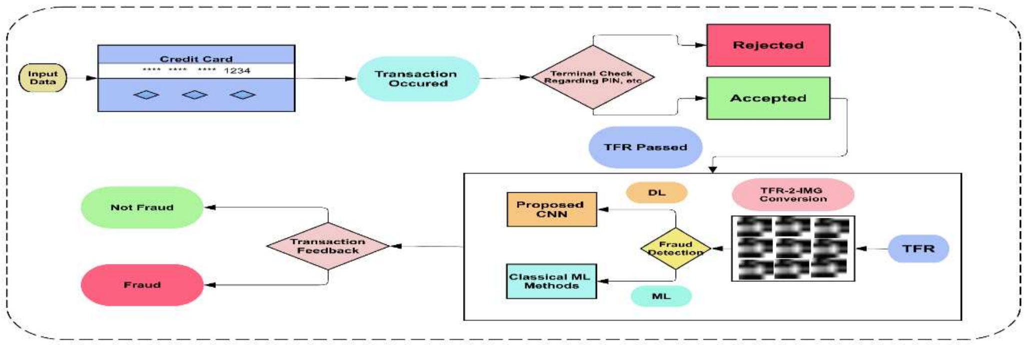


Figure 1: Overview of Enhancing Financial Fraud Detection Using Deep Learning Techniques (Alharbi et al., 2022).

The hybrid approach, illustrated in **Figure 1**, integrates deep learning with traditional machine learning techniques. This system transforms transaction data into structured feature representations or image formats, enabling advanced analysis by deep learning components like Convolutional Neural Networks (CNNs) while incorporating insights from traditional methods. The outputs are combined to classify transactions as fraudulent or legitimate, providing real-time feedback and leveraging the complementary strengths of both approaches to enhance fraud detection accuracy and flexibility (Alharbi et al., 2022).

1.2 Research Problem

Traditional fraud detection systems, relying on rule-based algorithms and statistical models, are ill-equipped to handle the sophisticated and evolving tactics of modern fraudsters, particularly in dynamic environments (Pan, 2024).

Deep learning offers a promising alternative, with its ability to detect anomalies and recognize complex patterns. However, challenges such as data privacy concerns, imbalanced datasets, and high computational demands limit its broader adoption (Hilal, Gadsden, and Yawney, 2021).

This study focuses on developing deep learning models tailored for financial fraud detection. The aim is to create scalable, efficient tools capable of real-time detection while minimizing false positives and adapting to emerging fraud patterns. The ultimate goal is to enhance the accuracy and reliability of fraud detection systems, reducing financial losses and improving trust in financial institutions.

1.3 Research Aim and Objectives

This research aims to improve financial fraud detection systems by developing and assessing adaptive deep learning models, with a focus on enhancing their accuracy, efficiency, and real-time performance.

The objectives of the study are to:

- Develop and optimize deep learning models capable of accurately detecting fraudulent transactions while minimizing false positives.

- Implement models that automatically extract features from raw transaction data, reducing the need for manual feature engineering.
- Assess the performance of the developed models using quantitative metrics such as accuracy, precision, recall, F1-score, and the Receiver Operating Characteristic Area Under the Curve (ROC-AUC) to determine their effectiveness in identifying fraudulent transactions.

1.4 Research Questions

The following research questions will be addressed in optimizing deep learning models for financial fraud detection:

- How can deep learning models be optimized to improve detection accuracy and efficiency, including through automatic feature extraction?
- How do deep learning models compare to traditional methods in fraud detection, based on key evaluation metrics?

1.5 Significance of the Study

This study holds significance for the financial sector, where fraud detection remains a major concern, as well as for academia, regulatory bodies, and the technology industry. The research applies deep learning techniques to improve the accuracy, efficiency, and adaptability of fraud detection systems. Advanced models can help reduce false positives, ensuring fewer disruptions for legitimate customers and better resource management. Additionally, deep learning's adaptability helps financial institutions stay ahead of emerging fraud patterns, strengthening their ability to prevent fraud proactively (Khalid et al., 2024).

The study provides a thorough analysis of deep learning-based fraud detection systems, evaluating their capabilities and limitations while exploring practical applications. Recommendations are made to assist financial institutions in implementing systems that meet regulatory standards, avoid fines, and build trust with investors. This research also touches on the ethical and privacy challenges associated with AI-driven fraud detection. It emphasizes the need for maintaining data security while pushing for technological advancements. The aim is to contribute to the creation of responsible, equitable AI solutions that change how financial institutions approach fraud detection.

1.6 Limitations of the Study

One of the main challenges is the use of imbalanced datasets in fraud detection, which could impact model performance (Cherif et al., 2022;Manda et al., 2024). Deep learning models are known to struggle with this issue, often resulting in biased predictions. Additionally, the study's scope is limited to certain types of financial fraud, such as credit card fraud, which means that the findings may not generalize to all forms of financial crimes. Furthermore, the complexity of deep learning models can make them computationally expensive, requiring significant resources for training and deployment (Ibomoiye Domor Mienye and Nobert Jere, 2024). Finally, while the study aims to improve detection accuracy, issues related to model interpretability remain a concern, especially when explaining decisions to non-technical stakeholders.

1.7 Definitions of Key Terms

Table 1 defines key concepts central to understanding financial fraud detection systems and the role of deep learning in improving these systems. These terms are crucial for grasping the challenges and techniques discussed in the study.

Table 1: Key terms and definitions related to financial fraud detection.

Term	Definition
Financial Fraud	Any intentional act of deception or misrepresentation carried out to gain an unfair or unlawful financial advantage. This includes credit card fraud, identity theft, phishing, and account takeovers (Bello et al., 2022).
Deep Learning	A subset of machine learning that uses neural networks with many layers (known as deep networks) to analyse large datasets, identifying patterns and making predictions without explicit programming for each task (Md Kamrul Hasan Chy, 2024).
Anomaly Detection	The process of identifying unusual patterns in data that do not conform to expected behaviour. In the context of fraud detection, this involves flagging

	transactions that deviate from typical spending patterns (Hilal, Gadsden and Yawney, 2021).
False Positive	A result where a legitimate transaction is incorrectly identified as fraudulent, leading to unnecessary disruptions for the customer and a waste of resources (Bartsiotas and Achamkulangare, 2016).
Overfitting	A common issue in machine learning where a model becomes too complex and closely fits the training data, making it less effective at generalizing to new, unseen data (Halima Oluwabunmi Bello, Adebimpe Bolatito Ige and Maxwell Nana Ameyaw, 2024).

1.8 Structure of the Thesis

The thesis is structured to guide the reader through the study step by step:

- **Chapter 2**: Reviews existing literature on traditional fraud detection methods and deep learning techniques, focusing on neural networks and anomaly detection. It highlights the shortcomings of current systems and sets the stage for the new approaches proposed in this study.
- **Chapter 3**: Details the methodology used in this research. This chapter covers data collection, the selection of deep learning algorithms, and the feature engineering process. It also discusses the experimental setup and any necessary ethical considerations.
- **Chapter 4**: Presents the results from the experiments, including an analysis of the datasets, and the evaluation metrics used. This chapter includes a comparison of model performance, both with traditional methods and new deep learning techniques.
- **Chapter 5**: Concludes the study by summarizing key findings and offering recommendations for future research. This chapter discusses how the results contribute to improving fraud detection and outlines areas for further investigation.

Chapter II: Literature Review

2.1 Introduction

This thesis examines financial fraud detection using deep learning, focusing on challenges like high false positives and dependence on manual feature engineering. It reviews fraud types, current trends, and the strengths and weaknesses of traditional and deep learning approaches.
The analysis highlights deep learning's accuracy, scalability, and ability to detect new fraud patterns while acknowledging issues such as computational demands and large dataset requirements. It identifies research gaps, including the need for real-time, adaptable models, and outlines directions for improving fraud detection systems in financial institutions.

2.2 Traditional Fraud Detection Methods

Traditional fraud detection methods, including statistical techniques and rule-based systems, have historically played a critical role in safeguarding transactions (Bello et al., 2023). Rule-based systems rely on predefined parameters set by domain experts, flagging or blocking transactions that deviate from established norms or violate regulatory requirements (Gilian Schrijver, Sarmah, and El-hajj, 2024). Similarly, statistical methods, such as regression and correlation analysis, utilize historical data to detect anomalies and calculate the likelihood of fraud. While these approaches perform well in structured and predictable scenarios, they struggle to adapt to the dynamic and sophisticated tactics of modern fraudsters, resulting in reduced accuracy and scalability (Pinto and Sobreiro, 2022).
Despite advancements like feature engineering and ensemble learning, traditional methods often face high false positive rates, limited real-time applicability, and challenges with imbalanced datasets (Ahsan, Luna, and Siddique, 2022). While hybrid models that incorporate machine learning techniques such as decision trees, logistic regression, and support vector machines offer incremental improvements, these systems are increasingly outpaced by the complexity of evolving fraud patterns.

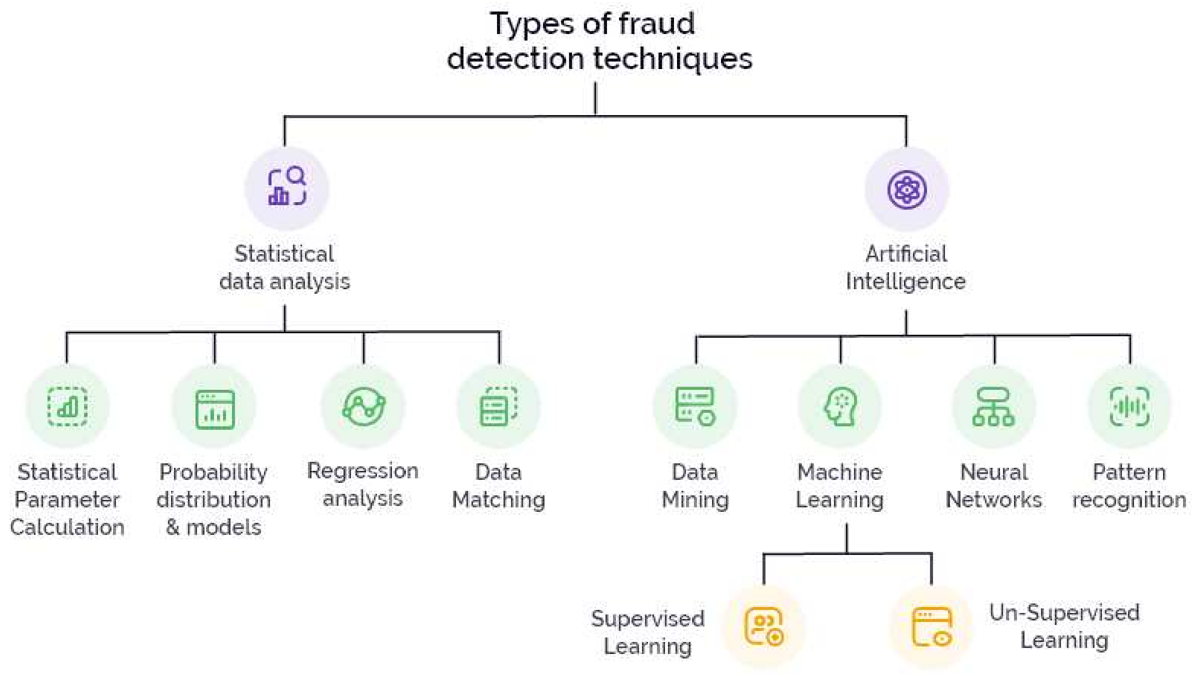


Figure 2 : An Overview of Traditional Fraud Detection Methods (Labs, 2024)

To address these limitations, emerging approaches now integrate traditional methods with advanced techniques, such as deep learning and AI. Tools like SMOTE and ADASYN generate synthetic data to enhance model training, enabling more accurate and efficient fraud detection (Xu et al., 2020; Lee et al., 2024; Sulaiman, Ibraheem Nadher, and Hameed, 2024). As **Figure 2** illustrates, while traditional methods remain valuable, the shift towards deep learning provides a more flexible and adaptive framework for combating financial fraud (Labs, 2024).

Table 2 : Summarises the key limitations of these traditional approaches.

Method	**Key Limitations**
Rule-Based Systems	Rigid, high maintenance, high false positives, predictable to fraudsters
Statistical Methods	Limited to known patterns, assumes normality, hard to scale for real-time, sensitive to data quality

Machine Learning	Requires high-quality, labelled data; struggles with imbalanced data; resource-intensive retraining; prone to overfitting
General Issues	Poor at detecting emerging fraud, limited real-time capabilities, complexity in handling large-scale data, lack of interpretability

2.3 Deep Learning Techniques in Fraud Detection

Deep learning, through its ability to analyse high-dimensional datasets and autonomously learn patterns, has become a cornerstone in modern fraud detection systems. In eliminating the need for extensive manual feature engineering, these models offer improved accuracy and adaptability compared to traditional methods (Shams Forruque Ahmed et al., 2023). Their capacity to recognize subtle, evolving fraud strategies positions them as effective tools in combating financial crime (Catelli, 2020).

Artificial Neural Networks (ANNs) excel in identifying anomalies by analysing transaction patterns over time. Similarly, Convolutional Neural Networks (CNNs), originally designed for image processing, are adapted to detect structured irregularities in financial datasets, such as recurring patterns in transaction grids (El Kafhali, Tayebi, and Sulimani, 2024). These architectures provide an efficient framework for developing dynamic and scalable fraud detection solutions.

A summary of commonly used AI, ML, and DL algorithms is presented in **Figure 3**.

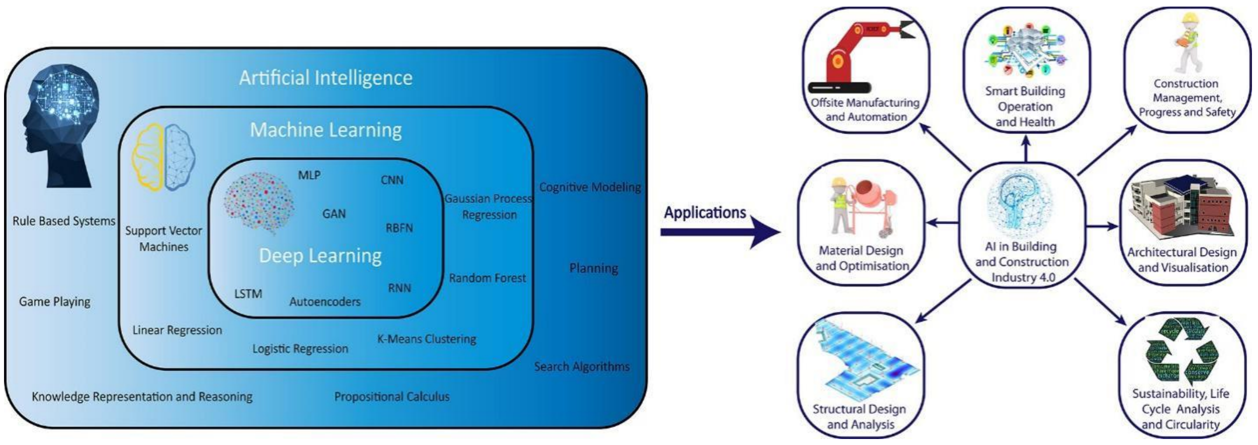

Artificial Intelligence
Machine Learning
Deep Learning
MLP
CNN
GAN
RBFN
RNN
LSTM
Autoencoders
Gaussian Process Regression
Random Forest
Support Vector Machines
Linear Regression
Logistic Regression
K-Means Clustering
Rule Based Systems
Game Playing
Cognitive Modeling
Planning
Search Algorithms
Knowledge Representation and Reasoning
Propositional Calculus
Applications
Offsite Manufacturing and Automation
Smart Building Operation and Health
Construction Management, Progress and Safety
Material Design and Optimisation
AI in Building and Construction Industry 4.0
Architectural Design and Visualisation
Structural Design and Analysis
Sustainability, Life Cycle Analysis and Circularity

Figure 3: Domains of AI, ML, DL, and Generally Used Algorithms (Baduge et al., 2022)

Autoencoders and Generative Adversarial Networks (GANs) are unsupervised learning models used in fraud detection. Autoencoders identify outliers and anomalies by reconstructing input data, flagging deviations from expected patterns. GANs generate synthetic fraudulent data to train models, enhancing their ability to detect fraud (Kamal Berahmand et al., 2024). Deep Belief Networks (DBNs) are also used to model complex data distributions. These deep learning techniques offer efficiency, adaptability, scalability, and high accuracy making them well-suited for detecting fraud in complex financial systems (Bhowmik et al., 2022).

2.4 Neural Networks for Fraud Detection

Neural networks are advanced machine learning models designed to identify patterns and relationships in data by iteratively adjusting network weights to minimize prediction errors. Their flexibility makes them effective for various tasks, including fraud detection, where they analyse complex transactional data to uncover anomalies and suspicious activities.

Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are two widely used architectures in fraud detection. CNNs, originally developed for image processing, utilize layers such as convolutional, pooling, and fully connected layers to detect spatial hierarchies and structured irregularities in transactional datasets (Sarker, 2021). Meanwhile, RNNs, designed to process sequential data, applies recurrent connections in their hidden layers to capture temporal dependencies, making them particularly adept at analysing transaction histories and identifying patterns of fraudulent behaviour (Shams Forruque Ahmed et al., 2023; Kim, Jung, and Kim, 2022).

Real-world applications highlight the effectiveness of these architectures. For instance, a CNN-based model implemented by a bank demonstrated superior accuracy in detecting fraudulent credit card transactions compared to traditional rule-based systems. Similarly, an online payment platform employed RNNs to analyse user transaction sequences in real-time, successfully identifying and mitigating fraud as it occurred (Zhang et al., 2018; Femila Roseline et al., 2022). Autoencoders, another neural network variant, have also been used to identify atypical transaction patterns, further reducing fraud risks through unsupervised anomaly detection (Marazqah Btoush et al., 2023).

Table 3 provides a comparative analysis of recent studies employing neural network models in this domain.

Table 3: An Analysis of Some Recent Studies Using Neural Network Models

Author/Year	**DL Model**	**Performance Matrix**	**Dataset**	**Limitation**
Ibomoiye Domor Mienye and Nobert Jere(2024)	Convolutional Neural Networks (CNN)	Accuracy: 93.8%, Precision: 92.0%, Recall: 90.5%	Credit Card Fraud Dataset (Kaggle)	Limited feature interpretability, challenges in handling sequential data
Oluwatoyin and Akinola (2024)	Recurrent Neural Networks (RNN)	Precision: 90.4%, Recall: 88.9%, F1-Score: 89.7%	Bank Transaction Dataset (Private)	High computational demand, poor scalability for very large datasets

Mienye, Swart and Obaido(2024)	Long Short-Term Memory (LSTM)	Accuracy: 95.2%, AUC: 0.92	Credit Card Fraud Detection (UCI)	Model complexity, requires substantial training data
vardhanetal.(2024)	Autoencoders (for Anomaly Detection)	F1-Score: 87.5%, Precision: 85%	Synthetic Fraudulent Transactions	Dependence on quality of synthetic data, limited by training on normal data
Mienye and Swart (2024)	Generative Adversarial Networks (GAN)	Accuracy: 89.8%, Recall: 88.5%	Financial Transaction Dataset (Private)	GANs are prone to instability during training, difficulty in handling complex fraud types
Kasasbeh, Aldabaybah and Ahmad2022)	Multi-Layer Perceptron (MLP)	Accuracy: 90.6%, AUC: 0.91	Real-Time Financial Transactions (Private)	Overfitting on small datasets, limited adaptability to evolving fraud patterns

2.5 Anomaly Detection in Fraud Prevention

Anomaly detection is a valuable tool in fraud prevention, identifying unusual data patterns that could indicate fraudulent activity (Bello and Olufemi, 2024). Unlike traditional methods that rely

on known patterns, anomaly detection can spot new fraud types. Combining it with rule-based and supervised learning enhances fraud detection (Sarker et al., 2024).
Unsupervised methods, like autoencoders and GANs, are effective for anomaly detection. Autoencoders reduce reconstruction errors to identify outliers in transaction data (Du et al., 2024), while GANs generate synthetic fraudulent data to assess transaction legitimacy. For example, a financial institution used autoencoders to monitor credit card transactions, improving fraud detection rates (Femila Roseline et al., 2022).
An online payment platform used GANs to generate fake transactions, enhancing its fraud detection capabilities (Chew, Yang and Lee, 2023). A retail bank also used GANs to identify fraud across multiple transaction channels, boosting client confidence and reducing losses. These methods offer early, precise fraud detection and have proven successful in real-world applications (Abdullahi, 2024).

2.6 Integration of Deep Learning Techniques

Integrating deep learning techniques with anomaly detection methods enhances fraud detection by combining their strengths (Detthamrong et al., 2024). Deep learning models, especially neural networks, can uncover hidden fraud patterns that traditional methods may miss. Autoencoders are effective for identifying transaction patterns and outliers, while RNNs capture the sequential nature of transactions over time (Nama and Obaid, 2024). GANs can generate synthetic fraudulent data to expand CNN training datasets. Combining these models into an ensemble approach provides a more comprehensive view of the data.
Integrating deep learning with streaming data enables real-time transaction analysis, reducing the risk of losses (Ishfaq Hussain Rather and Kumar, 2023). Deploying deep learning models as APIs ensures smooth integration with existing fraud detection systems, enabling seamless communication between system components. To optimize model performance, preprocessing of data, including handling missing values and feature extraction, is essential (Basheer et al., 2024). Additionally, scalable data storage systems are necessary for real-time and batch processing. Regular updates and retraining help maintain the model's effectiveness against emerging fraud trends. Overall, combining deep learning with anomaly detection improves fraud detection accuracy, offering real-time analysis, predictive capabilities, and seamless integration with existing systems (Mumuni and Mumuni, 2024).

2.7 Challenges and Solutions

Table 4 highlights challenges in deep learning-based fraud detection, alongside solutions and research gaps that need attention for improved model performance.

Table 4: Existing Challenges, Solutions and Gaps

Challenge	**Solution**	**Research Gap**
Data Quality (Elouataoui, El Mendili and Gahi, 2023)	Managing missing values, eliminating noise, and standardizing data formats through data cleaning and normalization. Detecting and removing anomalies using statistical methods and ML techniques.	Identifying advanced anomaly detection techniques tailored to financial data for more effective noise and outlier handling.
Limited Historical Fraud Data (Charitou, Dragicevic and D'avila Garcez, 2024)	Using GANs and other methods to generate synthetic data that mimics actual fraudulent transactions.	Exploring more realistic data generation techniques to accurately represent diverse fraud scenarios.
Data Privacy Constraints (Oyewole, Oguejiofor and Bakare, 2024)	Ensuring compliance with privacy regulations through secure data-sharing agreements with banks and using anonymized data for training.	Developing methods that enable more flexible data-sharing frameworks while ensuring compliance with evolving privacy laws.
Computational Demands (Liu, 2024)	Reducing model size with techniques like knowledge distillation, model pruning, and quantization. Using distributed computing and cloud platforms to manage computational needs.	Investigating cost-efficient methods for model optimization to enable resource-constrained institutions to implement deep learning models.

Model Interpretability (Md. Mahmudul Hasan, 2024)	Employing SHAP and LIME for model interpretability, developing visualization tools to showcase decision paths, and combining deep learning with interpretable models.	Enhancing interpretability techniques specific to fraud detection models to meet regulatory transparency requirements.
Evolving Fraud Tactics (Liang et al., 2024)	Using MLOps for automated training, deployment, and monitoring, CI/CD pipelines for seamless updates, online learning for gradual adaptation, and monitoring to trigger retraining when necessary.	Investigating adaptive learning mechanisms that allow models to learn and update in real-time without frequent manual intervention.

Solutions include data cleaning, normalization, anomaly detection techniques, synthetic data generation, secure data-sharing agreements, and flexible data-sharing frameworks. However, research gaps exist in identifying advanced anomaly detection techniques, developing realistic data generation techniques, reducing computational demands, improving model interpretability, and addressing evolving fraud tactics.

2.8 Benefits of Deep Learning in Fraud Detection

Deep learning models enhance fraud detection by reducing false positives and negatives, improving the accuracy of identifying fraudulent transactions (Marazqah Btoush et al., 2023b). These models can detect outliers and complex patterns in large datasets, boosting recall rates and enabling real-time fraud detection. This automation helps banks process large volumes of transactions quickly, minimizing human review while preventing fraud and financial losses. Deep learning models also adapt to new fraud patterns as data evolves, ensuring continuous improvement in detection (Faisal et al., 2024). For financial institutions handling millions of transactions daily, these models manage vast amounts of data efficiently. Integrating deep learning strengthens fraud detection systems without disrupting existing infrastructure, enhancing security and customer trust (Bello et al., 2022).

2.9 Financial Fraud: Types and Trend

Financial fraud encompasses several types, including credit card fraud, money laundering, insider trading, and securities fraud (Afjal, Salamzadeh and Dana, 2023). Credit card fraud involves unauthorized transactions using personal information, while money laundering is the process of moving illicit funds through legitimate channels. Identity theft is another form, where personal data is stolen for fraudulent purposes. Insider trading damages market integrity by trading stocks based on non-public information, and investment fraud misleads investors into backing non-existent businesses (Cornell Law School, 2019). Other examples include insurance fraud, pyramid and Ponzi schemes, and mortgage fraud. Understanding these fraud types and trends is essential for effective detection and prevention (Ľubomír Čunderlík, 2021). **Figure 4** illustrates the financial fraud framework.

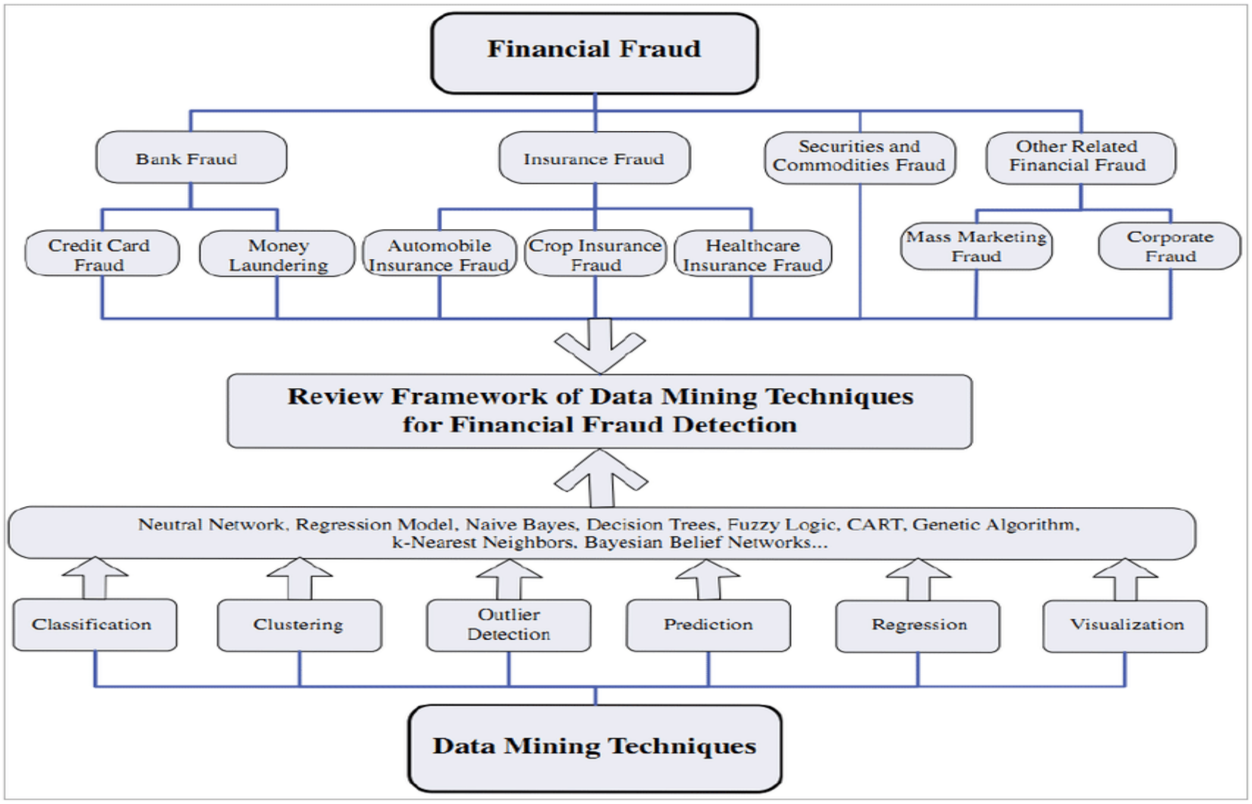

Financial Fraud
Bank Fraud
Insurance Fraud
Securities and Commodities Fraud
Other Related Financial Fraud
Credit Card Fraud
Money Laundering
Automobile Insurance Fraud
Crop Insurance Fraud
Healthcare Insurance Fraud
Mass Marketing Fraud
Corporate Fraud
Review Framework of Data Mining Techniques for Financial Fraud Detection
Neutral Network, Regression Model, Naive Bayes, Decision Trees, Fuzzy Logic, CART, Genetic Algorithm, k-Nearest Neighbors, Bayesian Belief Networks...
Classification
Clustering
Outlier Detection
Prediction
Regression
Visualization
Data Mining Techniques

Figure 4: The Financial Fraud Framework (Rizvi, 2021)

2.10 Research Gap

Deep learning models offer strong potential for financial fraud detection, but several research gaps remain. Key challenges include data quality issues such as noise, missing values, and inconsistencies in transaction data. The lack of historical fraud data also hinders model training, with Generative Adversarial Networks (GANs) proposed to generate synthetic data. Privacy concerns limit access to comprehensive financial datasets, and computational demands make it difficult for smaller institutions to implement these models effectively. Model interpretability remains an issue, as existing tools like SHAP and LIME are not specifically tailored for fraud detection. Additionally, adaptive learning mechanisms are needed to allow fraud detection models to update in real-time without manual intervention. Addressing these gaps is crucial for improving the accuracy, efficiency, and scalability of fraud detection systems.

Table 5 summarizes key research gaps in financial fraud detection using deep learning, current solutions being implemented, and the future research needs to address these challenges. It highlights the areas where improvements can be made to enhance the effectiveness and efficiency of fraud detection systems.

Table 5: Research Gaps in Financial Fraud Detection Using Deep Learning

Research Gap	**Current Solutions**	**References**	**Future Research Needs**
Data Quality	Data cleaning, normalization, and basic anomaly detection	(Camacho et al., 2023)	Advanced anomaly detection techniques tailored to financial data
Limited Historical Fraud Data	Use of GANs for synthetic data generation	(Charitou, Dragicevic and Garcez, 2021)	Realistic data generation techniques that capture the diversity of fraud scenarios
Data Privacy Constraints	Secure data-sharing frameworks, anonymization	(Oh et al., 2024)	Development of privacy-preserving techniques for

			regulatory-compliant data sharing
Computational Demands	Model compression techniques like pruning and quantization	Islam (2021); Chen & Wang (2023)	Cost-efficient optimization methods to make models feasible for smaller institutions
Model Interpretability	Use of interpretability tools like SHAP and LIME	(Md. Mahmudul Hasan, 2024)	Fraud-specific interpretability frameworks to meet regulatory transparency requirements
Evolving Fraud Tactics	MLOps and automated retraining pipelines	(Johnson, 2024)	Development of adaptive learning mechanisms for real-time model updates without manual intervention

2.11 Chapter Summary

This chapter highlights the current achievements and limitations in using deep learning to detect financial fraud. Challenges include poor data quality, insufficient historical fraud records, privacy restrictions, high computational requirements, limited model interpretability, and difficulties in adapting to changing fraud tactics.

Existing solutions, such as data cleaning, synthetic data generation with GANs, secure data-sharing practices, and tools like SHAP and LIME, have addressed some of these issues but lack specificity for financial fraud scenarios. This limits their effectiveness and applicability.

Future research should focus on advanced anomaly detection, realistic data generation for diverse fraud patterns, affordable model optimization for smaller organizations, and interpretability tailored to regulatory requirements. Mechanisms for real-time updates without manual intervention are also essential to keep systems effective against emerging threats.

Addressing these gaps will strengthen fraud detection systems, ensuring better security and reliability in financial transactions.

.

Chapter III: Methodology

3.1 Introduction

This chapter outlines the research methodology adopted for developing and evaluating deep learning models for financial fraud detection. The methodology provides a systematic approach to achieve the study's objectives, including data acquisition, model development, evaluation, and validation processes. The chapter also discusses the tools, techniques, and frameworks used in the study.

3.2 Research Design

The research adopts **an experimental design** to develop and test deep learning models for detecting financial fraud using labelled transaction data. The process includes data collection and preprocessing to ensure the quality of input, followed by model development and training to enhance performance (Georgios Zioviris, Kostas Kolomvatsos and Stamoulis, 2024). The models are then evaluated and tested using quantitative metrics to measure their accuracy and efficiency. The final step involves analysing results to assess the models' capability in identifying fraudulent transactions and meeting the research objectives.

3.3 Justification for Algorithm Selection and Optimization Approach

The choice of Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN) as primary algorithms for this research was driven by their complementary capabilities in fraud detection. ANN was selected for its versatility in handling structured tabular data and its ability to model complex non-linear relationships between features, which is critical in identifying fraudulent patterns in transactional data. CNN, traditionally employed in image and sequential data analysis, was included due to its effectiveness in capturing localized patterns and temporal relationships, which are useful in time-series features often associated with transaction data (Nisa Aulia Saputra et al., 2024).

The optimization approach utilized the Adam optimizer, a rich and widely adopted optimization algorithm known for its adaptive learning rate and momentum features. Adam effectively handles sparse gradients and noisy data, making it suitable for both ANN and CNN in this context (Shams Forruque Ahmed et al., 2023b). Its computational efficiency and ability to converge quickly support iterative experiments, ensuring models achieve high performance within practical training durations.

3.4 Justification for Study Comparison, Metrics, and Experimental Approach

To validate the research outputs, the study incorporates findings from *Dhawan and Gitanjali (2024),* which benchmarks machine learning techniques for credit card fraud detection. This comparative analysis establishes the relevance of ANN and CNN by situating their performance against industry-standard models. The inclusion of this study allows for a meaningful evaluation of whether deep learning models outperform traditional machine learning approaches in terms of accuracy and recall, two pivotal metrics in fraud detection.

The experimental approach was designed to test both models on the same dataset under controlled conditions, ensuring fairness in comparison. The use of oversampling techniques (SMOTE) addressed class imbalance, a common challenge in fraud detection datasets, thereby ensuring meaningful evaluation across minority (fraudulent) and majority (non-fraudulent) classes. The inclusion of training-validation splits and multiple epochs further strengthened the model's reliability, allowing for consistent insights across ANN, CNN, and external benchmarks (Shams Forruque Ahmed et al., 2023c).

3.5 Research Framework

The methodology follows a process aimed at achieving accurate and reliable results in financial fraud detection. The first stage, **data collection**, involves acquiring and understanding the dataset to ensure it is suitable for analysis. Next, **data preprocessing** focuses on cleaning and preparing the data, addressing issues like missing values or inconsistencies to improve its quality. The third stage, **model development**, entails designing and training deep learning models tailored to the problem, ensuring they are well-suited for detecting fraudulent transactions.

Evaluation follows, where the models are assessed using metrics such as accuracy, precision, recall, and F1-score to measure their effectiveness. Finally, the **validation** stage ensures that the models can generalize effectively to unseen data and remain reliable under different conditions. These sequential stages, as depicted in **Figure 5**, ensure a systematic approach to building a solution for financial fraud detection.

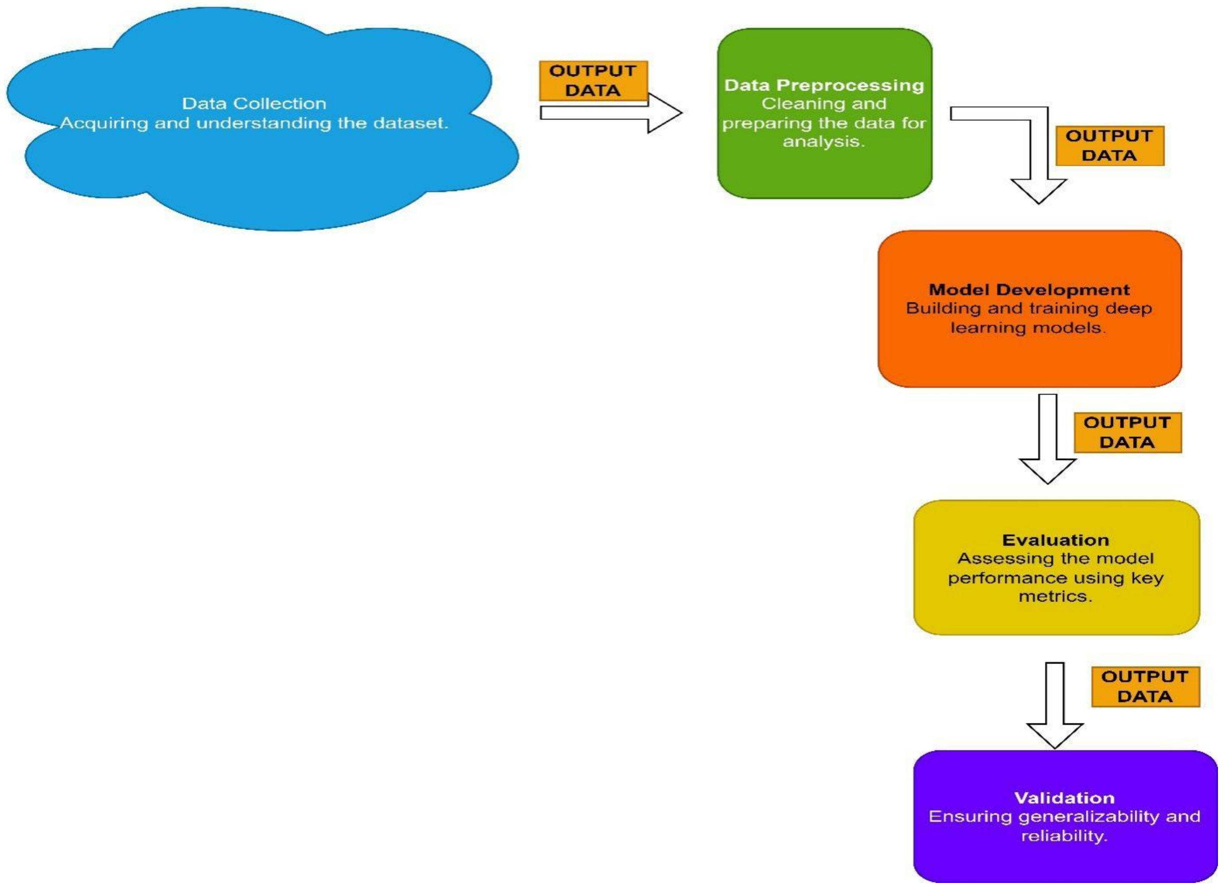

Data Collection
Acquiring and understanding the dataset.
OUTPUT DATA
Data Preprocessing
Cleaning and preparing the data for analysis.
OUTPUT DATA
Model Development
Building and training deep learning models.
OUTPUT DATA
Evaluation
Assessing the model performance using key metrics.
OUTPUT DATA
Validation
Ensuring generalizability and reliability.

Figure 5: Research Methodology Framework

3.6 Data Collection

This study utilizes a Kaggle dataset of credit card transactions labelled as fraudulent or legitimate, containing 275,929 records collected over two days in September 2013. Among these, only 484 transactions are fraudulent, representing a mere 0.2% of the total. This class imbalance mirrors the rarity of fraud in real-world scenarios and poses significant challenges for classification.

The dataset includes 31 attributes, all numerical. Twenty-eight variables, labelled V1 to V28, were generated using Principal Component Analysis (PCA) to ensure confidentiality while retaining essential patterns. Two additional features, *Time* and *Amount*, are preserved in their original form. *Time* captures the elapsed seconds since the first recorded transaction, offering insights into temporal patterns, while *Amount* represents the transaction's monetary value, a factor often linked to fraud detection strategies. The target variable, *Class*, is a binary indicator marking transactions as fraudulent (1) or legitimate (0).

The numerical format of the dataset facilitates compatibility with deep learning algorithms, while its skewed class distribution underscores the need for evaluation metrics like the Area Under the Precision-Recall Curve (AUPRC) to assess model performance effectively on imbalanced data.

Table 6: Key Features in the Dataset

Feature Name	**Description**
Time	Seconds since the first transaction in the dataset.
V1 to V28	Principal components obtained from PCA transformation.
Amount	Monetary value of the transaction.
Class	Target variable: 1 (fraudulent), 0 (non-fraudulent).

3.7 Data Preprocessing

Preprocessing the *df* dataframe derived from the *creditcard.csv* dataset involved a series of steps to optimize its suitability for deep learning tasks. Early preprocessing reveals the dataset contained primarily numerical features. Features like V1-V28, Amount, and Class were found to be numerical upon executing code *df.info()* and there were no missing values requiring no further efforts at data cleaning or imputation.

Table 7: Summarizing the key statistics and insights drawn from the dataset

Feature	Key Statistics	Insights
Time	Min: 0, Max: 166,798, Mean: 92,408.87, Std: 46,282.16	Transactions are evenly distributed over time, with no strong clustering.
V1 to V28	Mean ~ 0, Std ~ 1 (for most components), Large outliers in several features (e.g., V3, V8, V9)	Principal components are centred and scaled. Outliers in certain features could influence model performance and may require additional preprocessing.
Amount	Min: 0, Max: 25,691.16, Mean: 89.06, Std: 251.46, Median: 22.39	Most transactions are of small amounts, as indicated by the median and interquartile range (5.80 to 78.00). A few high-value outliers are present, which could serve as potential fraud indicators.
Class	Min: 0, Max: 1, Mean: 0.001754	Fraudulent transactions (Class 1) are rare, confirming the dataset's imbalanced nature. Special attention will be required to handle this imbalance effectively during model development.

Table 7 provides a detailed summary of the statistical properties of the dataset, focusing on key metrics for features such as *Time*, principal components (V1 to V28), *Amount*, and *Class*. The Time feature spans a range from 0 to approximately 166,798, with a mean of 92,408 and a standard deviation of 46,282, indicating a consistent distribution of transactions over time without apparent temporal clustering. The principal components (V1 to V28) derived from PCA, show mean values near zero, reflecting effective centring during preprocessing. Most components have standard deviations close to 1, suggesting appropriate scaling. However, certain features (e.g., V3, V8, V9) display extreme outliers, as evidenced by maximum and minimum values that deviate significantly from their interquartile ranges.

The Amount feature exhibits a wide range of transaction values, from 0 to 25,691.16, with a mean of 89.06 and a standard deviation of 251.46. Despite this broad range, the median (22.39) and interquartile range (5.80 to 78.00) highlight that most transactions involve relatively smaller amounts, with a few high-value outliers. The Class feature representing fraudulent (1) and non-fraudulent (0) transactions, shows a mean of 0.001754, indicating the dataset's significant imbalance, with fraudulent transactions constituting a small fraction. These findings emphasize the necessity of handling outliers, implementing scaling, and addressing class imbalance effectively to enhance the performance of any predictive models developed using this dataset.

The Time and Amount columns exhibited differences in scale compared to the PCA-transformed features (V1 through V28). To ensure uniform scaling, these columns were normalized using *StandardScaler.* This transformation standardised the values to have a mean of zero and unit variance, which is essential for algorithms sensitive to feature magnitude.

The plot in **Figure 6** illustrates the distribution of transaction amounts after applying a logarithmic transformation *(log1p*). The x-axis represents the log-transformed transaction amounts, while the y-axis shows the count of transactions.

Before transformation, the "Amount" feature exhibited significant skewness, with a concentration of lower transaction amounts and a long tail of higher values. Such skewed distributions can negatively impact deep learning models, as they may lead to biased weight adjustments and hinder the model's ability to generalize.

By applying the logarithmic transformation, the skewness is significantly reduced, resulting in a more symmetric distribution. This transformation not only improves the visualization of the data but also makes the feature more suitable for modelling by reducing the influence of extreme values. The smoother, bell-shaped curve observed in the figure demonstrates the effectiveness of this preprocessing step in preparing the data for better model performance.

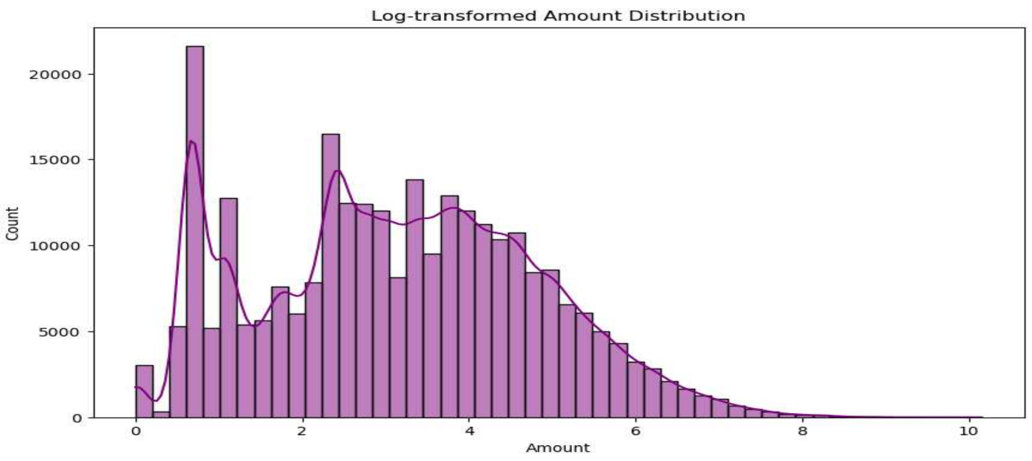


Figure 6: Distribution of Log-transformed Amount

The dataset revealed an imbalance between the two classes: non-fraudulent (Class = 0) and fraudulent transactions (Class = 1) with the minority class being 0.2% and the majority class 99.8% (**See Figure 7**). To address this issue, SMOTE (Synthetic Minority Over-sampling Technique) was used. This method generated synthetic samples for the minority class, resulting in a balanced dataset and improving the model's ability to detect fraudulent transactions.

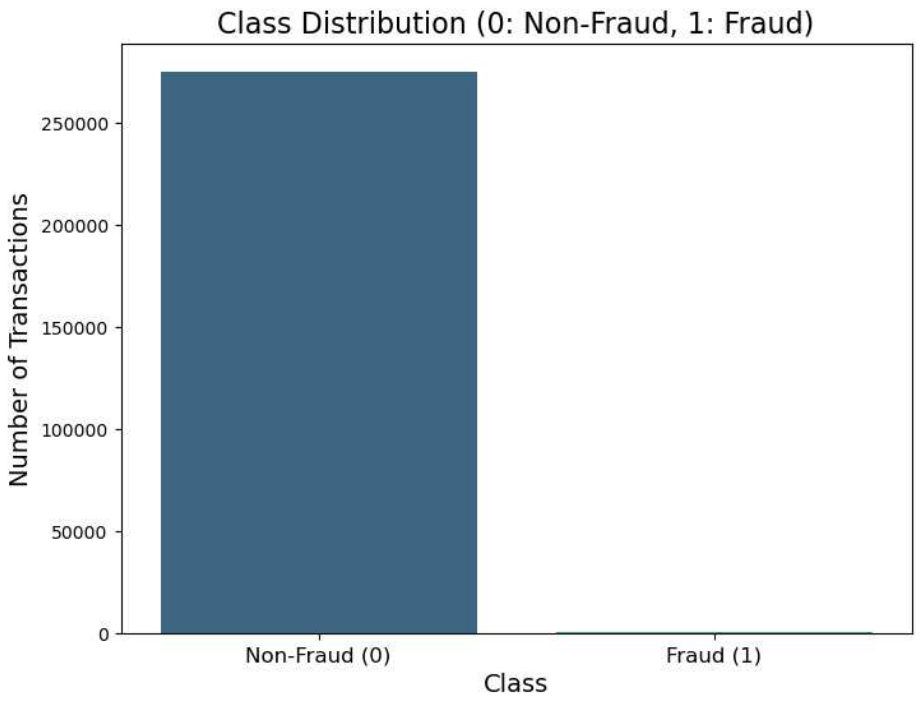


Figure 7: Class distribution.

The time distribution plot displayed in **Figure 8** compares the count of fraudulent and non-fraudulent transactions over time. The x-axis represents the time of transactions, while the y-axis shows the count of transactions. Non-fraudulent transactions are depicted with a blue histogram, overlaid with a kernel density estimate (KDE) curve, while fraudulent transactions are represented by the red bars.

From the figure, it is evident that non-fraudulent transactions follow a periodic distribution over time, with peaks and troughs suggesting temporal patterns, possibly influenced by user behaviour or system activity. Conversely, fraudulent transactions appear as relatively rare events, with counts significantly lower than non-fraudulent ones, making them nearly imperceptible in the plot.

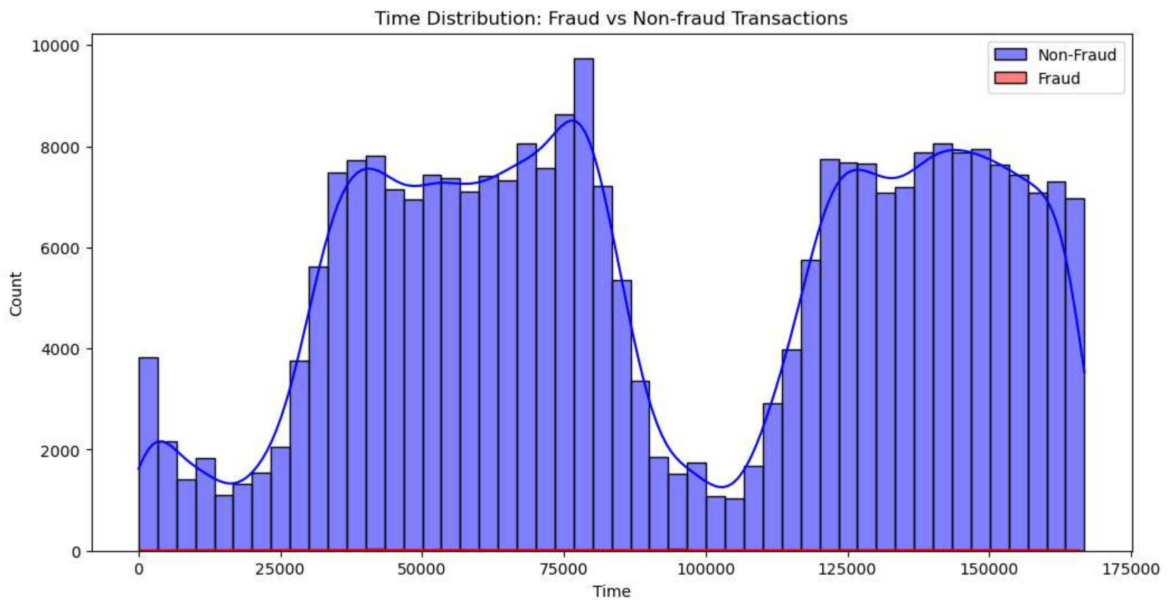


Figure 8: Time Distribution of Transactions (for Fraud and Non-fraud)

Figure 9 displays a correlation heatmap of all numerical features, including *Time*, *Amount*, and the PCA-transformed features (*V1* through *V28*). The heatmap uses a colour gradient to represent correlation values, ranging from -1 (perfect negative correlation, shown in blue) to +1 (perfect positive correlation, shown in red).

The diagonal line of red blocks represents the self-correlation of each feature with itself, which is always 1. The rest of the matrix helps in identifying the strength and direction of relationships between features. For example:

- Features with a high positive correlation (close to +1) share similar patterns, indicating redundancy or dependency.
- Features with strong negative correlation (close to -1) exhibit opposing patterns.
- Near-zero correlations suggest weak or no linear relationship between features.

This heatmap aids in visualizing the relationships between *Time*, *Amount*, and the PCA-generated features, which are crucial for understanding their impact on fraud detection. The principal components (V1 to V28) are uncorrelated with one another, as expected, but each component exhibits perfect correlation with itself (e.g., V1 with V1, V2 with V2, and so on), confirming the orthogonality property of PCA. However, the correlation between *Amount* and some other features might provide insights into feature selection or preprocessing steps.

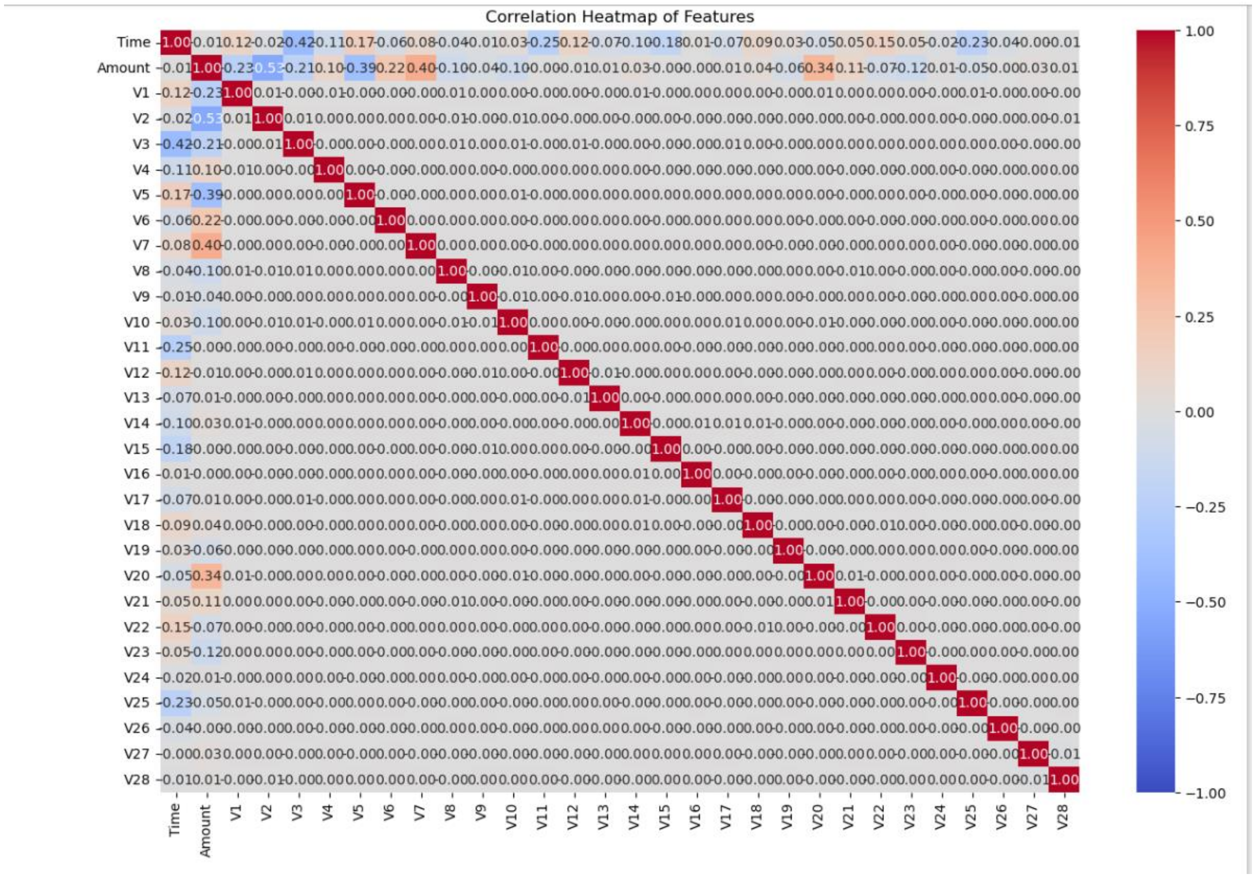


Figure 9: Correlation Heatmap (For PCA features and Amount)

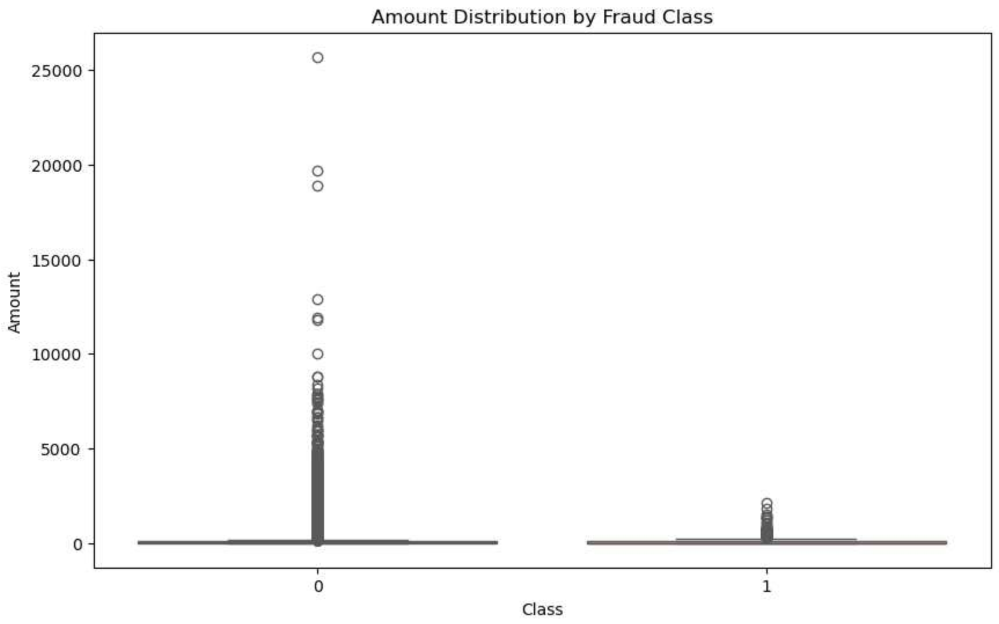


Figure 10: Boxplot for Amount Distribution by Class (Fraud vs. Non-fraud)

Figure 10 illustrates the distribution of transaction amounts for non-fraudulent (Class 0) and fraudulent (Class 1) transactions using a boxplot. Non-fraudulent transactions exhibit a wider range, with numerous outliers exceeding 10,000 units and some surpassing 25,000 units, while the majority are concentrated at lower amounts. In contrast, fraudulent transactions are generally smaller in value, with fewer outliers and amounts tightly clustered around lower values. This pattern suggests that fraudsters may favour smaller transactions to avoid detection, whereas non-fraudulent transactions include a broader spectrum, including high-value amounts. The distinct distribution between classes highlights transaction amount as a key feature for fraud detection models. This can help identify if there is a difference in the Amount distribution between fraudulent and non-fraudulent transactions.

After preprocessing, the resampled dataset was inspected for consistency, ensuring all transformations were correctly applied. The class distribution was checked to confirm the effectiveness of SMOTE. The Class distribution after SMOTE reveals: Counter ({0: 275445, 1: 275445})

Through these steps, *df* was prepared for deep learning, ensuring scaled, balanced, and features for effective training and evaluation.

3.8 Model Development

To detect financial fraud, this study implemented deep learning models, specifically Artificial Neural Networks (ANNs) and Convolutional Neural Networks (CNNs), tailored for the refined dataset. The focus was on creating scalable, generalizable, and industry-relevant architectures.

3.8.1 Artificial Neural Networks (ANNs)

An ANN was designed to uncover complex relationships within the dataset. Preprocessing steps included scaling the features with *StandardScaler* and addressing class imbalance using *SMOTE* to ensure a balanced training set. The ANN architecture comprised an input layer (See **Figure 11**), two hidden layers with 64 and 32 neurons respectively (using *ReLU* activation for non-linear pattern detection), and a sigmoid-activated output layer for binary classification.
The model was trained for *10 epochs* with a batch size of 32 using the Adam optimizer and *binary cross-entropy loss*. Validation was conducted during training on a separate test set. Performance evaluation metrics included accuracy, precision, recall, F1-score, ROC-AUC, and a confusion matrix.

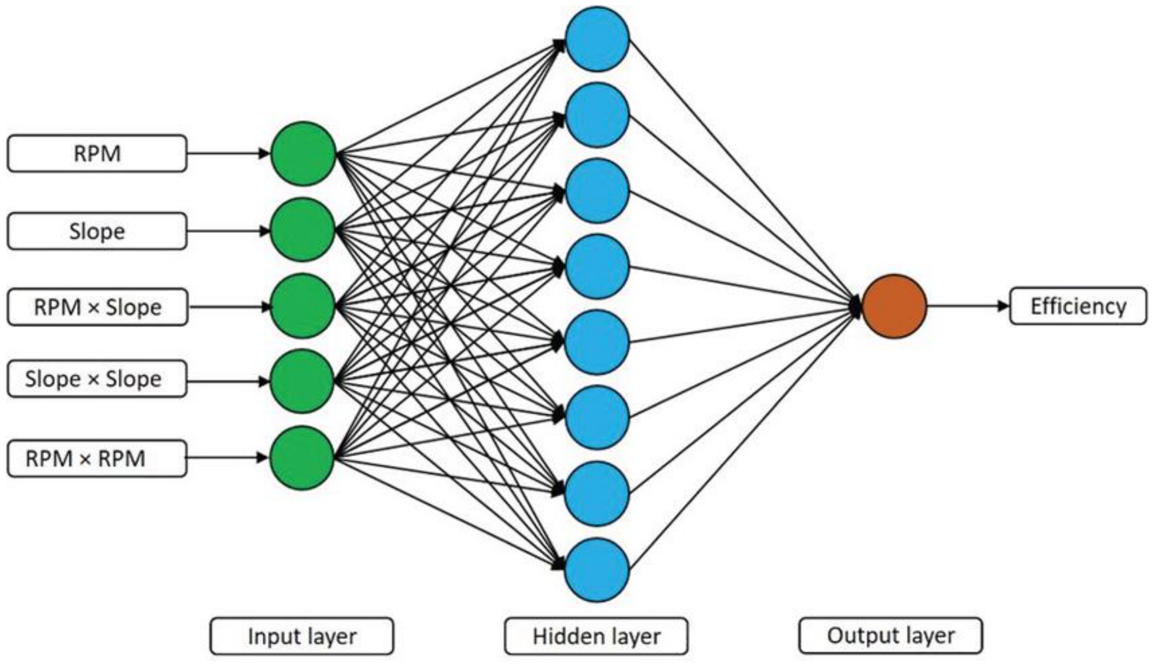


Figure 11: Schematic diagram of artificial neural network (ANN) architecture ((Tito et al., 2021)

3.8.2 Convolutional Neural Networks (CNNs)

A CNN was implemented to apply its ability to detect spatial patterns within transactional data. The normalized and balanced dataset (via *SMOTE*) was reshaped into a 1D format suitable for convolutional operations. In **Figure 12** an architecture detailing CNN approach is showcased. The architecture included convolutional layers with 64 and 128 filters, using *ReLU* activation, followed by max-pooling to reduce dimensionality and retain critical features. The convolutional layers' output was flattened and passed through a dense network, culminating in a sigmoid-activated output neuron for binary classification. This design enabled the CNN to automatically learn feature hierarchies and detect intricate patterns indicative of fraudulent activity.

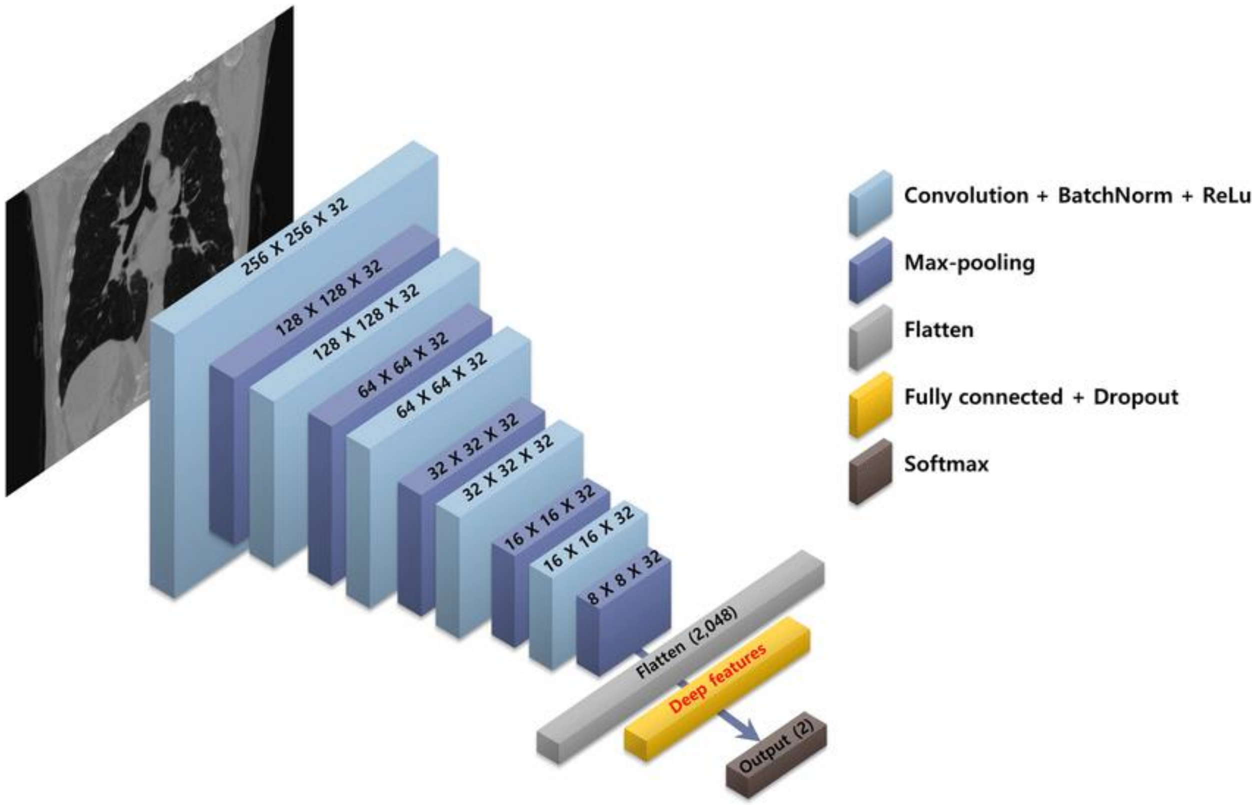


Figure 12: Convolutional neural network (CNN) model architecture. (Yun et al., 2021)

3.9 Model Evaluation

The predictive performance of the trained models was assessed using standard metrics that are well-established in fraud detection. These included:

- Accuracy: Evaluates the overall correctness of predictions.
- Precision: Focuses on the reliability of positive predictions by measuring the proportion of correctly identified fraudulent transactions out of all transactions flagged as fraud.
- Recall: Highlights the ability to detect all actual fraud cases, minimizing the risk of false negatives.
- F1-Score: Combines precision and recall into a single metric, offering a balanced view of the model's ability to handle imbalanced datasets.
- ROC-AUC: Measures the model's capacity to distinguish between fraudulent and legitimate transactions across varying decision thresholds.

Table 8 provides a side-by-side comparison of these metrics for the ANN and CNN models, illustrating their respective performance strengths.
This analysis reveals that while both models performed strongly, the CNN demonstrated a slight edge in recall and ROC-AUC, making it particularly effective at minimizing false positives—critical for fraud detection. These findings underscore the importance of tailoring model selection to specific fraud detection priorities, such as minimizing missed fraud cases or ensuring high overall reliability (Jolaosho Ahmed Oluwatoyin and Akinola, 2024a).

Table 8: Comparative Performance of ANN and CNN in Credit Card Fraud Detection

Metric	**Artificial Neural Network (ANN)**	**Convolutional Neural Network (CNN)**
Accuracy	High overall classification accuracy.	Mirrors similar performance.
Precision	Effectively reduces false positives.	Reduces false positives far more than ANN.
Recall	Efficient at capturing fraudulent cases.	Efficient at minimizing missed fraudulent cases.

F1-Score	Balances precision and recall effectively.	Achieves a stronger balance of precision and recall.
ROC-AUC	Reliable in distinguishing between fraud and legitimate cases.	Excels at identifying complex patterns for nuanced discrimination.

3.10 Tools and Software

The implementation of the models and analysis relied on several tools and software to ensure efficient development and accurate results. Python served as the primary programming language due to its versatility and extensive ecosystem of libraries tailored for Deep Learning and data analysis. Key libraries like Keras, which facilitated the development and training of deep learning models, while Scikit-learn supported preprocessing tasks, model evaluation, and metric calculations. Pandas was used for data manipulation and cleaning, providing a seamless interface for handling the dataset, and Matplotlib allowed for visualizing patterns, results, and performance metrics (Mahalaxmi, Donald and Srinivas, 2023).

The development environment utilized was Jupyter Notebook, chosen for its interactive nature and ability to combine code execution with real-time documentation. This setup streamlined the workflow by enabling iterative testing and debugging while maintaining a clear record of the process.

3.11 Ethical Considerations

This research adheres to ethical standards, including the **University of Hertfordshire's** ethical guidelines, to ensure compliance and integrity throughout the study. The Kaggle dataset utilized for this research is publicly available, eliminating the need for ethical consent as no direct data collection or interaction with individuals was involved.

Key ethical measures taken include:

1. The dataset was sourced from a public repository and is intended for research purposes. Proper attribution to the original owners ensures compliance with usage terms.

2. The dataset is pre-anonymized, containing no personally identifiable information (PII), safeguarding privacy and confidentiality.
3. Procedures followed align with institutional guidelines, confirming that no additional approvals or consents were required.

These steps affirm the ethical integrity of the research while ensuring respect for data privacy and ownership(Gold et al., 2024).

3.12 Chapter Summary

This chapter has outlined the methodology used to develop and evaluate deep learning models for financial fraud detection. The research design uses an experimental approach, supported by Kaggle's transaction dataset. Data preprocessing steps, model development, evaluation metrics, and tools have been detailed to ensure reproducibility and reliability.

Advanced architectures, including Artificial Neural Networks (ANNs) and Convolutional Neural Networks (CNNs) models, were designed to address the complex nature of fraudulent transactions. Each model was trained and evaluated using metrics such as accuracy, precision, recall, F1 score, and ROC-AUC.

A key focus was on addressing data imbalance through SMOTE, ensuring that fraudulent transactions were adequately represented during training. Additionally, data normalization and scaling were critical in preparing the dataset for effective processing by the models.

The chapter also emphasised the importance of adhering to ethical considerations, such as data privacy and attribution to the dataset owners, ensuring compliance with the University of Hertfordshire's guidelines. Tools like Python, TensorFlow, and Scikit-learn, as well as visualization libraries such as Matplotlib and Seaborn, facilitated the implementation of the methodology.

The following chapters will delve into the practical implementation of the described models and analyse their results to determine the most effective approach for fraud detection in financial transactions.

Chapter IV: Results and Discussion

4.1 Introduction

Chapter 4 provides an analysis of the results obtained from the experiments performed to develop and evaluate various models for financial fraud detection. The chapter emphasizes the evaluation of deep learning models, Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN).

The experiments were conducted on a preprocessed and resampled Kaggle credit card transaction dataset to address issues like class imbalance and ensure reliable performance assessment. Key performance metrics—accuracy, precision, recall, F1-score, and ROC-AUC—were used to evaluate and compare model effectiveness. Additionally, the models' strength in detecting fraudulent transactions, while maintaining low false positive rates, was critically analysed.

This chapter also explores the implications of the findings in the context of real-world financial fraud detection systems. The results are interpreted with reference to the stated research objectives, shedding light on the trade-offs between complexity, performance, and interpretability of the models.

4.1 Overview of Experiments

This section provides an outline of the experimental phases undertaken to evaluate and compare the effectiveness of the proposed deep learning models in detecting financial fraud. The experiments were designed to assess the performance of Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN). The results from this phase sheds light on the inherent strengths of the deep learning algorithms in detecting fraud but also revealed areas for improvement which will be detailed in the challenges and limitation sections subsequently.

The models were fine-tuned using various preprocessing and balancing techniques. Data preprocessing included normalization to enhance model input quality as discussed in **Section 3.5.** The Technique oversampling (SMOTE) was applied to address the significant class imbalance typically observed in fraud detection datasets. Additionally, this phase explored the trade-offs in computational efficiency and interpretability between traditional and deep learning approaches.

Table 9: Experimental Phases Overview

Phase	**Description**	**Techniques/Models**	**Objective**
Optimized Models	Testing ANN and CNN with data preprocessing (e.g., normalization, scaling) and balancing techniques (SMOTE).	ANN, CNN	Evaluate the effect of preprocessing and data balancing on model accuracy and robustness.

Table 9 summarizes the optimisation effort, outlining the key activity, models evaluated, and objective. This structured overview provides a roadmap for understanding the progression of experiments and their contribution to addressing the research objectives.

4.2 Performance Analysis

This section analyses the performance of Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN) trained on the optimized dataset. The focus is on their ability to detect fraudulent transactions while emphasizing key distinctions and shared strengths.

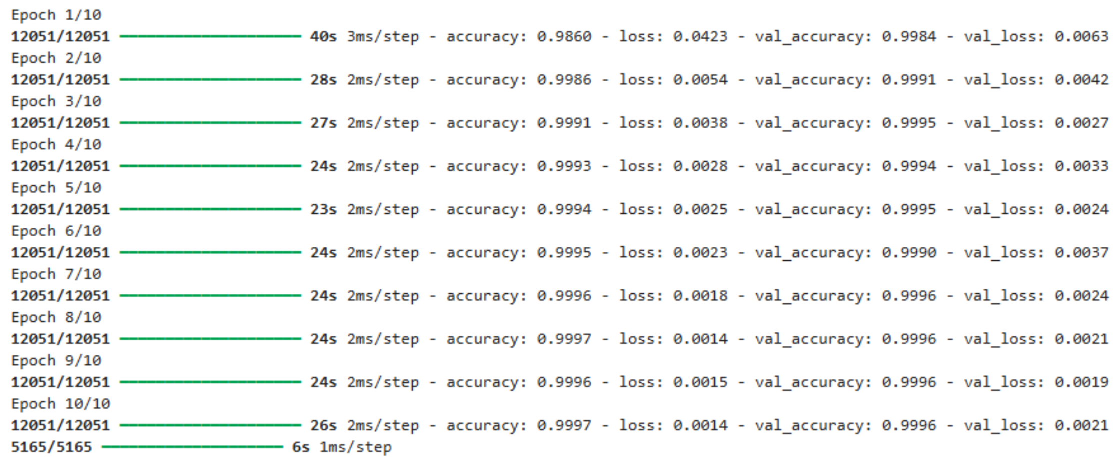
4.2.1 Model Training Performance

Both ANN and CNN exhibited steady improvements in accuracy and loss across 10 epochs, with minimal overfitting. Validation metrics closely mirrored training results, indicating effective generalization (**see Figures 13 and 14**).

Table 10: Training and Validation Metrics Overview

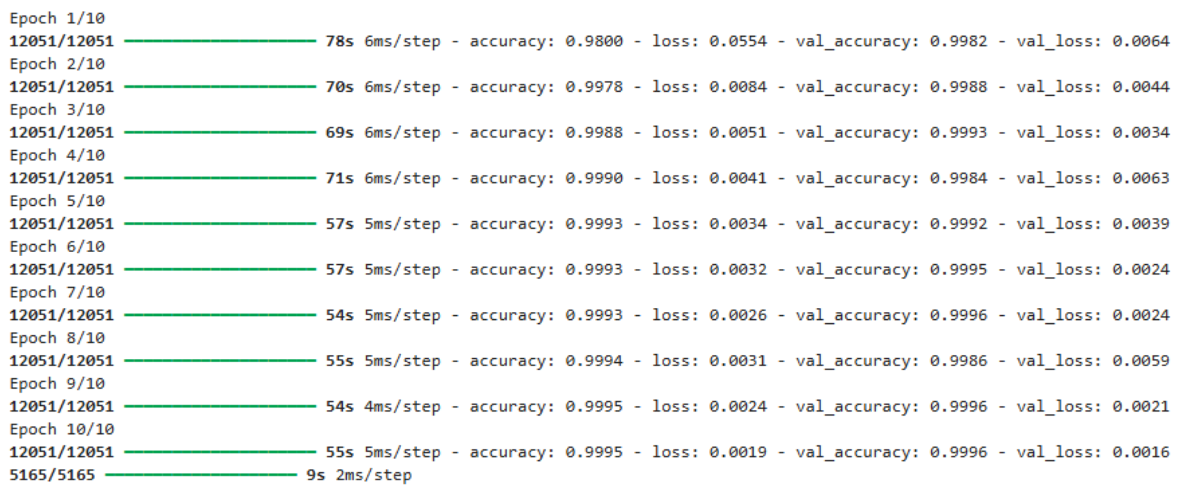
Model	**Epoch**	**Accuracy (Train)**	**Loss (Train)**	**Accuracy (Validation)**	**Loss (Validation)**
ANN	1	0.9860	0.0423	0.9984	0.0063
ANN	10	0.9997	0.0014	0.9996	0.0021
CNN	1	0.9800	0.0554	0.9982	0.0064
CNN	10	0.9995	0.0019	0.9996	0.0016

The ANN model demonstrated faster training, completing all 10 epochs in 264 seconds (approximately 4 minutes). In contrast, the CNN model required 620 seconds (approximately 10 minutes) to complete the same number of epochs. This difference amplifies the higher computational complexity of the CNN architecture, which involves resource-intensive operations like convolutional and pooling layers.



```
Epoch 1/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 40s 3ms/step - accuracy: 0.9860 - loss: 0.0423 - val_accuracy: 0.9984 - val_loss: 0.0063
Epoch 2/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 28s 2ms/step - accuracy: 0.9986 - loss: 0.0054 - val_accuracy: 0.9991 - val_loss: 0.0042
Epoch 3/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 27s 2ms/step - accuracy: 0.9991 - loss: 0.0038 - val_accuracy: 0.9995 - val_loss: 0.0027
Epoch 4/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 24s 2ms/step - accuracy: 0.9993 - loss: 0.0028 - val_accuracy: 0.9994 - val_loss: 0.0033
Epoch 5/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 23s 2ms/step - accuracy: 0.9994 - loss: 0.0025 - val_accuracy: 0.9995 - val_loss: 0.0024
Epoch 6/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 24s 2ms/step - accuracy: 0.9995 - loss: 0.0023 - val_accuracy: 0.9990 - val_loss: 0.0037
Epoch 7/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 24s 2ms/step - accuracy: 0.9996 - loss: 0.0018 - val_accuracy: 0.9996 - val_loss: 0.0024
Epoch 8/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 24s 2ms/step - accuracy: 0.9997 - loss: 0.0014 - val_accuracy: 0.9996 - val_loss: 0.0021
Epoch 9/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 24s 2ms/step - accuracy: 0.9996 - loss: 0.0015 - val_accuracy: 0.9996 - val_loss: 0.0019
Epoch 10/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 26s 2ms/step - accuracy: 0.9997 - loss: 0.0014 - val_accuracy: 0.9996 - val_loss: 0.0021
5165/5165 ━━━━━━━━━━━━━━━━━━━━ 6s 1ms/step
```

Figure 13: ANN's Epoch



```
Epoch 1/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 78s 6ms/step - accuracy: 0.9800 - loss: 0.0554 - val_accuracy: 0.9982 - val_loss: 0.0064
Epoch 2/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 70s 6ms/step - accuracy: 0.9978 - loss: 0.0084 - val_accuracy: 0.9988 - val_loss: 0.0044
Epoch 3/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 69s 6ms/step - accuracy: 0.9988 - loss: 0.0051 - val_accuracy: 0.9993 - val_loss: 0.0034
Epoch 4/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 71s 6ms/step - accuracy: 0.9990 - loss: 0.0041 - val_accuracy: 0.9984 - val_loss: 0.0063
Epoch 5/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 57s 5ms/step - accuracy: 0.9993 - loss: 0.0034 - val_accuracy: 0.9992 - val_loss: 0.0039
Epoch 6/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 57s 5ms/step - accuracy: 0.9993 - loss: 0.0032 - val_accuracy: 0.9995 - val_loss: 0.0024
Epoch 7/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 54s 5ms/step - accuracy: 0.9993 - loss: 0.0026 - val_accuracy: 0.9996 - val_loss: 0.0024
Epoch 8/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 55s 5ms/step - accuracy: 0.9994 - loss: 0.0031 - val_accuracy: 0.9986 - val_loss: 0.0059
Epoch 9/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 54s 4ms/step - accuracy: 0.9995 - loss: 0.0024 - val_accuracy: 0.9996 - val_loss: 0.0021
Epoch 10/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 55s 5ms/step - accuracy: 0.9995 - loss: 0.0019 - val_accuracy: 0.9996 - val_loss: 0.0016
5165/5165 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step
```

Figure 14: CNN's Epoch

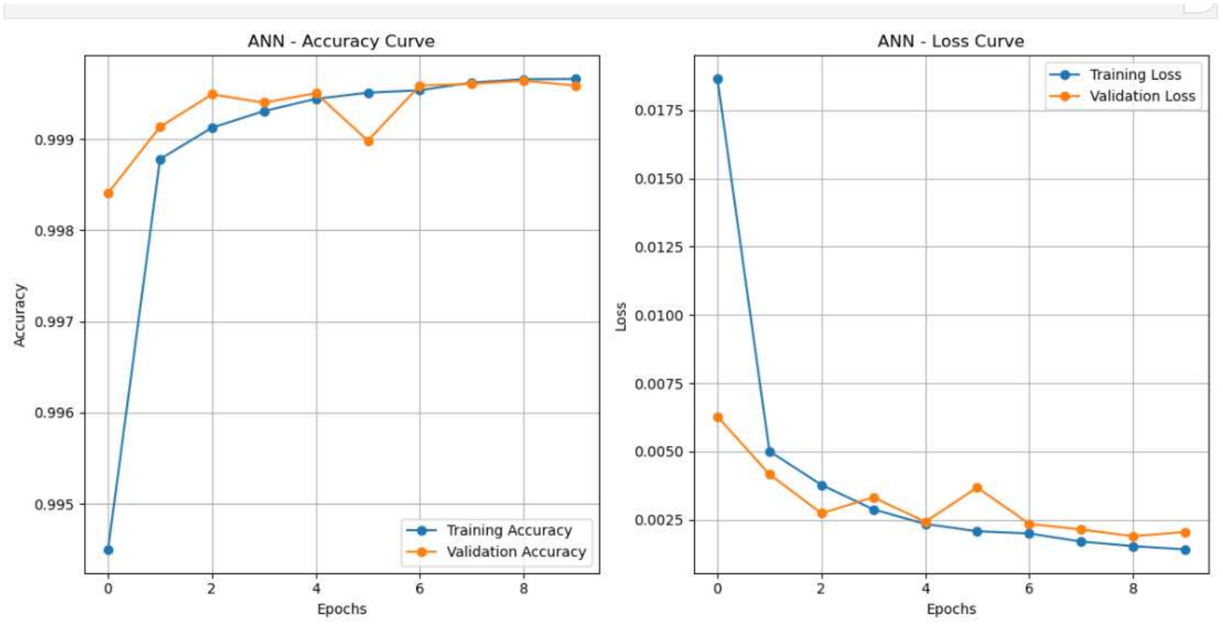


Figure 15: Training and Validation Accuracy over Epochs for ANN

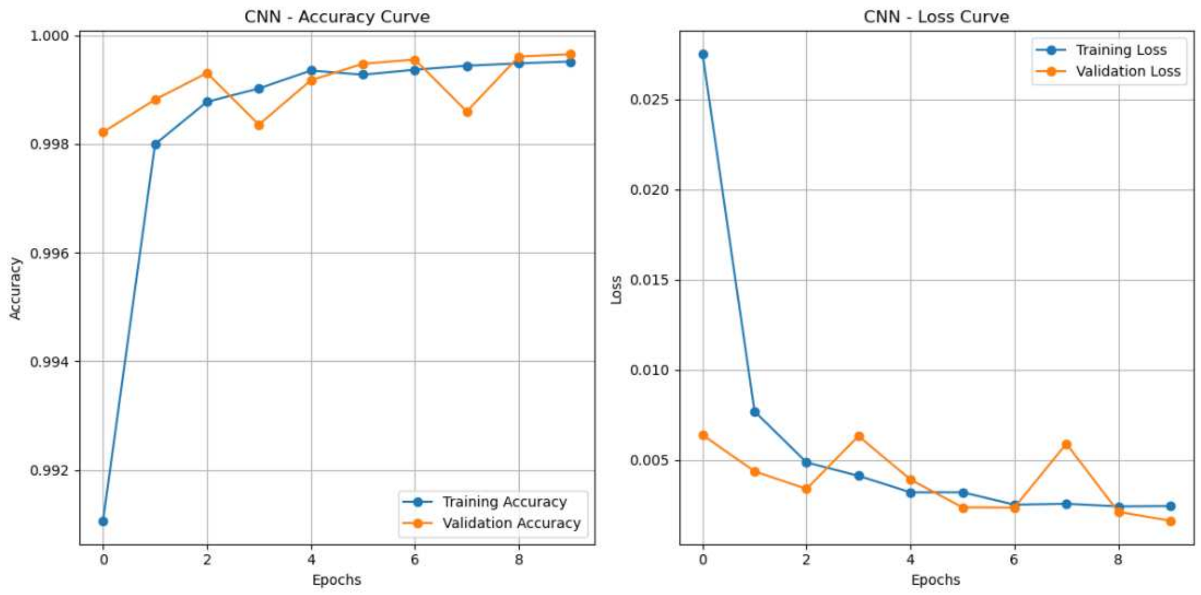


Figure 16: Training and Validation Loss over Epochs for CNN

4.2.2 Final Evaluation Metrics

When tested on the evaluation dataset, ANN and CNN achieved nearly identical performance metrics, as shown in **Table 11**.

Table 11: Evaluation Metrics for ANN and CNN

Metric	**ANN**	**CNN**
Accuracy	99.96%	99.96%
Precision	99.92%	99.93%
Recall	100.00%	100.00%
F1-Score	99.96%	99.96%
ROC-AUC	99.99%	100.00%

Both models demonstrated exceptional performance, with perfect recall ensuring that all fraudulent transactions were accurately detected. Precision and F1-scores were also near-perfect, reflecting a low rate of false positives and overall detection capabilities. While the metrics were similar, CNN's architecture lends itself to capturing complex patterns in transactional data, justifying its slightly higher computational cost.

While both ANN and CNN models performed exceptionally well, achieving high accuracy, recall, and precision, their computational demands differed. CNNs offer a slight edge in learning intricate patterns, but ANNs are computationally more efficient, making them a suitable choice in resource-constrained scenarios. Both models validate their applicability in addressing the challenges of credit card fraud detection effectively.

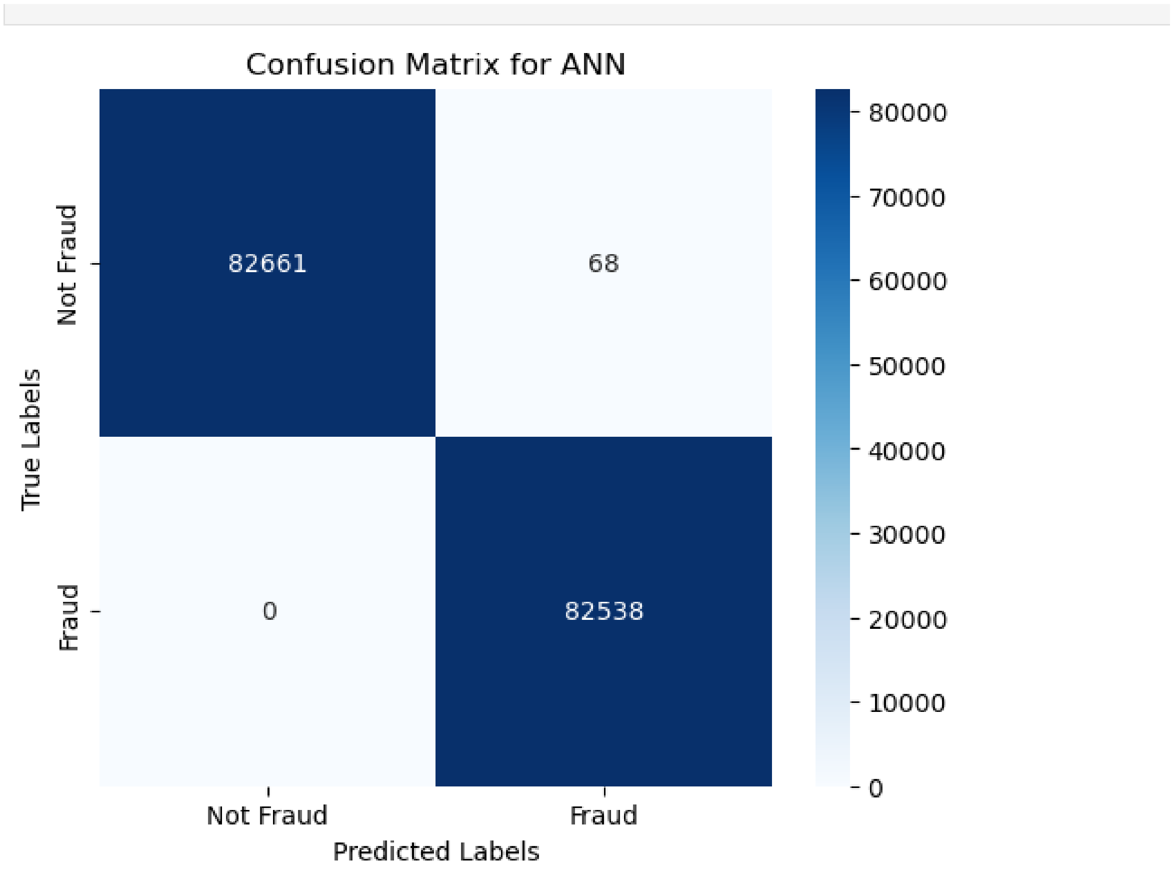


Figure 17: Confusion Matrix for ANN

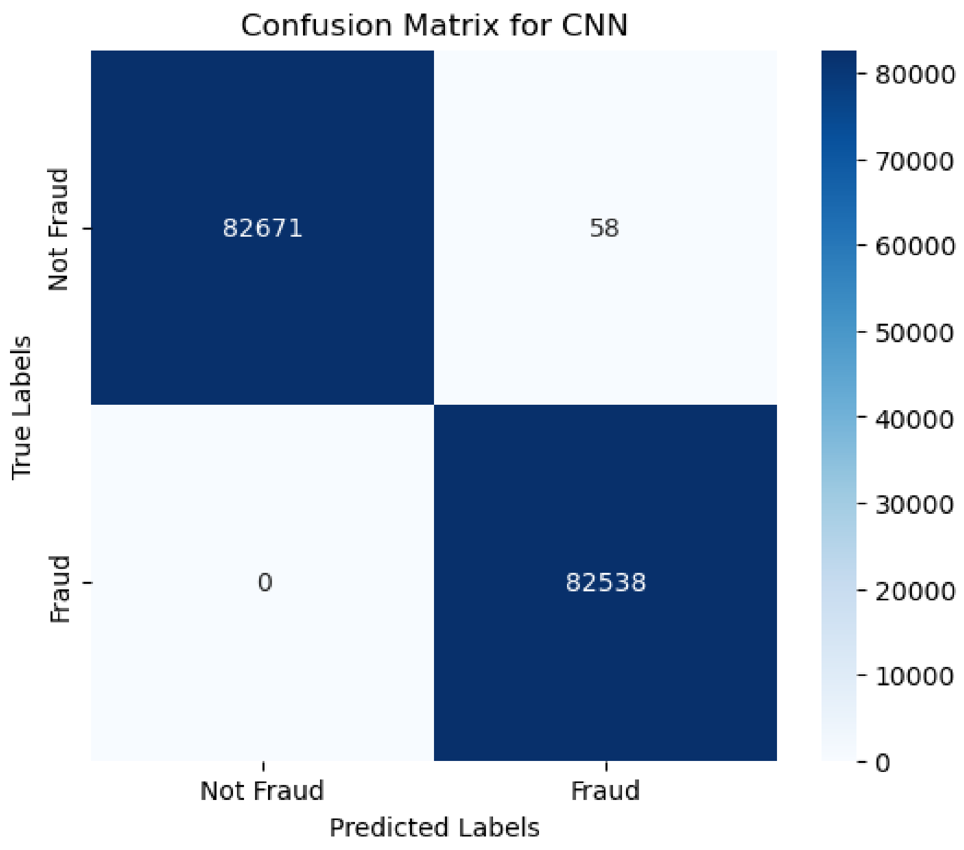


Figure 18: Confusion Matrix for CNN

4.2.3 Confusion Matrix Analysis

The confusion matrix offers a detailed breakdown of the classification results, comparing the performance of ANN and CNN in detecting fraudulent transactions. **Table 12** summarizes the results for both models.

Table 12: A Summary of Confusion Matrix Results

Model	**Predicted: Non-Fraudulent**	**Predicted: Fraudulent**
Actual: Non-Fraudulent (ANN)	82,661	68
Actual: Non-Fraudulent (CNN)	82,671	58
Actual: Fraudulent (Both Models)	0	82,538

Key Insights

- True Positives (TP): Both models successfully identified all 82,538 fraudulent transactions, achieving zero false negatives (FN = 0).
- True Negatives (TN): CNN slightly outperformed ANN with 82,671 true negatives compared to ANN's 82,661, indicating 10 fewer non-fraudulent transactions were misclassified by CNN.
- False Positives (FP): CNN exhibited fewer false positives (58) compared to ANN (68), reflecting slightly better precision.
- Recall: Both models achieved perfect recall (100%), ensuring no fraudulent cases were overlooked.
- Precision: CNN's lower false positive count contributed to marginally better precision compared to ANN.

Both ANN and CNN demonstrated exceptional performance, particularly in recall, ensuring accurate detection of all fraudulent cases. While CNN has a slight edge in reducing false positives and achieving higher precision, both models are highly effective and reliable for fraud detection.

4.2.4 Insights and Observations

- Both ANN and CNN demonstrated exceptional accuracy (99.96%) and perfect recall (100%), ensuring all fraudulent transactions were correctly identified. This highlights their effectiveness in high-stakes applications where missing fraudulent cases is unacceptable, with CNN showing a slight edge in precision due to fewer false positives.
- ANN demonstrated slightly more false positives (FP) than CNN, resulting in a marginally higher precision for CNN (99.93%) compared to ANN's 99.92%, while both models achieved the same F1-score (99.96%). This exceptional robustness minimizes unnecessary investigations and operational costs in financial systems.

- Both models converged effectively within 10 epochs, showing steady improvements in training and validation metrics without overfitting, despite the dataset's class imbalance.
- ANN's simpler architecture makes it easier to train, deploy, and scale, particularly in resource-constrained environments. This positions ANN as a highly adaptable model for real-world scenarios requiring flexibility and computational efficiency.
- While CNN excels in processing spatial patterns and offers advantages in feature extraction, ANN's comparable performance with a less complex design highlights its practicality for systems prioritizing reliability and scalability.
- ANN's ability to integrate additional features or adapt to new datasets ensures its long-term applicability in dynamic fraud detection contexts.

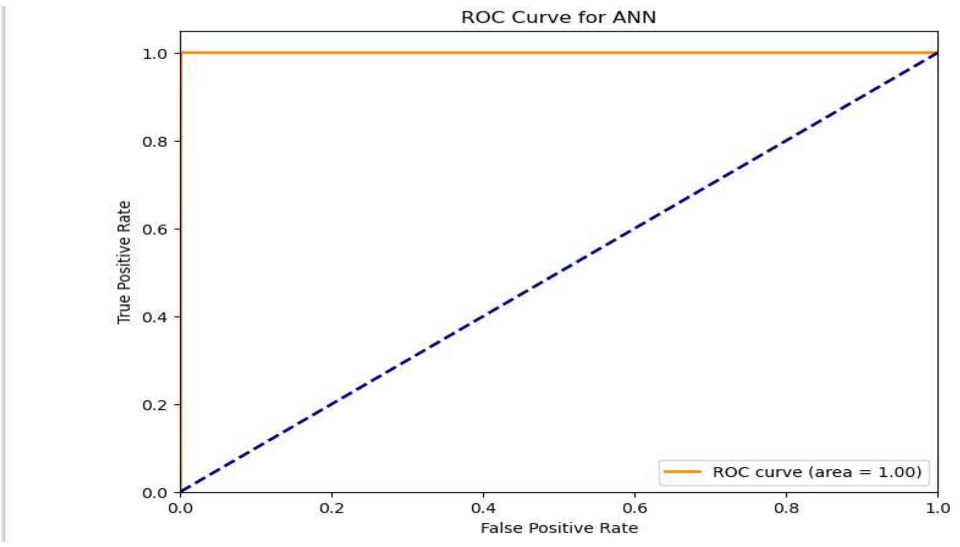


Figure 19: Comparative ROC Curves for ANN

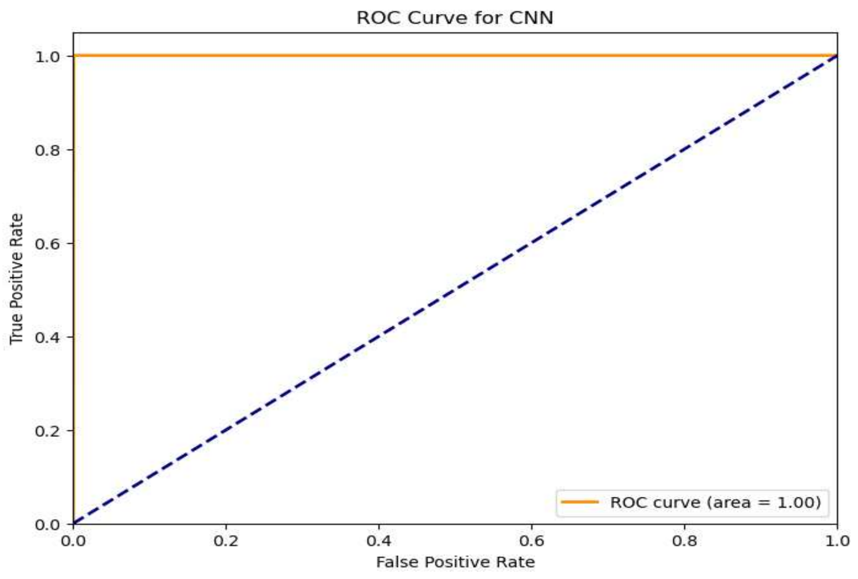


Figure 20: Comparative ROC Curves for ANN and CNN

4.2.5 Comparative ROC Curves for ANN and CNN

Figure 19 depicts the Receiver Operating Characteristic (ROC) curve for the Artificial Neural Network (ANN) model used in the fraud detection system. The ROC curve evaluates the model's classification performance by plotting the *True Positive Rate* (sensitivity) against the *False Positive Rate* at various classification thresholds.

Key observations from the figure include:

1. **Perfect Performance**: The ROC curve reaches the top-left corner (True Positive Rate = 1, False Positive Rate = 0) without any deviation. This indicates that the ANN model is classifying all instances correctly with a minimal false positive, suggesting perfect predictive performance.
2. **Area Under the Curve (AUC)**: The AUC value is 1.00, further confirming that the model achieves ideal discrimination between fraudulent and non-fraudulent transactions. A perfect AUC score indicates that the model distinguishes positive and negative classes without error.

3. **Dashed Line**: The diagonal dashed line represents a random classifier's performance (AUC = 0.5), which serves as a baseline for comparison. The ANN model's ROC curve lying entirely above this line demonstrates its superior classification ability.

This plot highlights the ANN model's exceptional effectiveness in detecting fraud, although such results warrant careful evaluation to ensure that the model's performance generalizes well to unseen data and is not due to overfitting.

Also, **Figure 20** shows the Receiver Operating Characteristic (ROC) curve for the Convolutional Neural Network (CNN) model used in the fraud detection system. Like the ANN model, this curve evaluates the classification performance of the CNN by plotting the *True Positive Rate* (sensitivity) against the *False Positive Rate* at various thresholds.

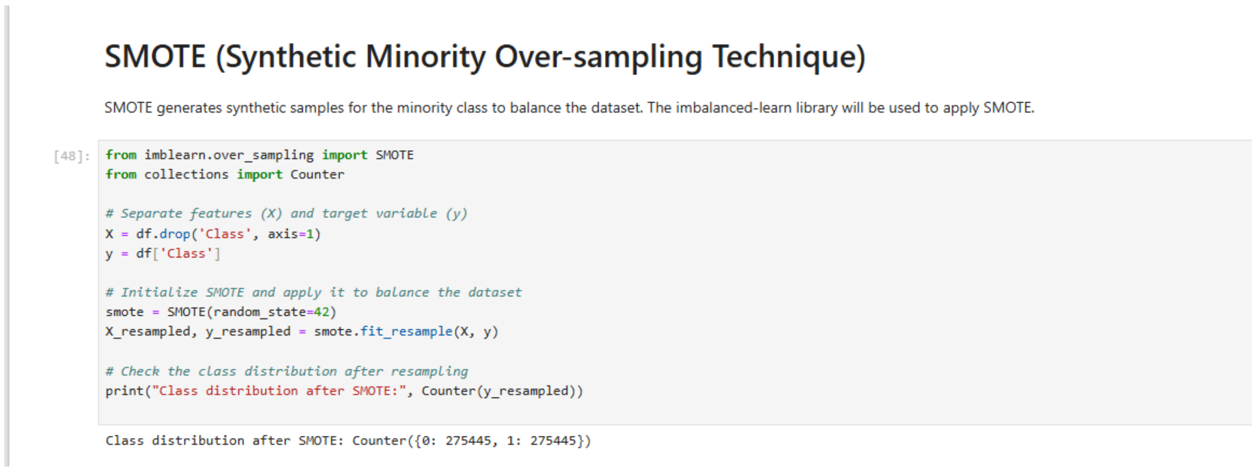
1. **Perfect Classification**:
    - The ROC curve for the CNN reaches the upper-left corner of the graph (True Positive Rate = 1, False Positive Rate = 0). This means the model achieved perfect classification performance, where it correctly identified all fraudulent transactions (True Positive Rate = 1) without incorrectly classifying any non-fraudulent transactions as fraudulent (False Positive Rate = 0). This indicates that the CNN model has an exceptional ability to distinguish between fraudulent and non-fraudulent transactions.
2. **Area Under the Curve (AUC)**:
    - The AUC for the CNN model is 1.00, which represents perfect performance. This indicates that the CNN model achieves ideal separation between the positive (fraudulent) and negative (non-fraudulent) classes.
3. **Baseline Comparison**:
    - The diagonal dashed line in the plot represents the performance of a random classifier (AUC = 0.5). The CNN's ROC curve being entirely above this line demonstrates its superiority in fraud detection.

This result, like the ANN's performance, highlights the CNN model's effectiveness. However, such flawless results may also warrant further investigation to ensure the model generalizes well and is not overfitting to the training data. Regularization techniques and evaluation on a separate validation dataset are essential to confirm robustness.

The performance analysis demonstrated that both ANN and CNN are highly effective at detecting fraudulent transactions in the given dataset. The nearly identical metrics suggest that either model could be deployed in practical applications, depending on specific requirements such as speed or interpretability.

4.3 Impact of Data Preprocessing

Data preprocessing played a role in the performance of both the ANN and CNN models. Techniques such as data normalization, Synthetic Minority Oversampling Technique (SMOTE) for class balancing improved the models' ability to classify fraud effectively (see **Figure 21**), addressing key challenges inherent in the dataset.



SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE generates synthetic samples for the minority class to balance the dataset. The imbalanced-learn library will be used to apply SMOTE.

```
[48]: from imblearn.over_sampling import SMOTE
from collections import Counter

# Separate features (X) and target variable (y)
X = df.drop('Class', axis=1)
y = df['Class']

# Initialize SMOTE and apply it to balance the dataset
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Check the class distribution after resampling
print("Class distribution after SMOTE:", Counter(y_resampled))
```

```
Class distribution after SMOTE: Counter({0: 275445, 1: 275445})
```

Figure 21: Synthetic Minority Over-sampling Technique

The use of preprocessing techniques improves the precision and recall for ANN. Specifically, SMOTE addressed the data imbalance issue, allowing the model to better identify minority fraud cases that were previously underrepresented. Data normalization ensured consistent feature scaling, reducing biases in model training and improving convergence speed.

For CNN, preprocessing enhanced its feature extraction capabilities, which are integral to its convolutional layers. The application of normalization (See **Appendices A.2.1 and A.2.2**) standardized input data, enabling the CNN to detect intricate patterns with greater accuracy. Similarly, the use of SMOTE balanced the dataset, ensuring robust model learning across all classes. This led to higher ROC-AUC scores, reflecting the model's improved ability to distinguish between fraudulent and legitimate transactions as shown in **Figure 20**.

4.4 Comparative Analysis

Drawing from Insights and observations in **section 4.2.4**, a broader comparative analysis on traditional models can be achieved using works from existing studies

The analysis emphasises the clear advantages of ANN and CNN in fraud detection tasks, particularly in complex and imbalanced datasets. Their ability to achieve high recall and precision makes them indispensable for systems where the cost of missing fraudulent cases is high. Traditional machine learning models like Random Forest, while strong performers, fall short of the comprehensive capabilities provided by deep learning models. Logistic Regression, being a simpler algorithm, is less effective for detecting complex fraud patterns but remains available option for less demanding scenarios.

Dhawan and Gitanjali (2024) emphasized the need for a comprehensive credit card fraud detection systems due to the evolving nature of fraudulent activities driven by technological advancements. Their comparative study of machine learning algorithms, including logistic regression, decision tree, random forest, and Naïve Bayes, showcased key strengths and limitations of these models based on performance metrics like accuracy, precision, recall, F1-score, and ROC-AUC. While these traditional approaches are effective, the scalability, learning capability, and generalizability of deep learning algorithms such as Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN) provide significant advantages. ANN's strength lies in its ability to model complex, non-linear patterns, making it highly adaptable to diverse datasets as shown in the results obtained from this research. CNN, on the other hand, excels in learning hierarchical features, which enhances its ability to generalize across varied data distributions. Both models are scalable due to their capacity to process vast amounts of data with appropriate computational resources, offering enhanced adaptability in dynamic environments. These attributes make ANN and CNN promising solutions for advancing fraud detection systems, especially in scenarios requiring high precision and adaptability as this research currently shows from results shown in **Tables 11 and 12**.

4.5 Discussion on Research Questions and Objectives

The findings of this study address the research questions comprehensively, highlighting the advantages of deep learning models for financial fraud detection while evaluating their optimization.

4.5.1 Optimization Techniques for Model Performance and the Role of Automatic Feature Extraction in Fraud Detection

The question of how deep learning models can be optimized for both accuracy and efficiency were central to this research. Technique such as the use of advanced optimization algorithms like Adam led to the impressive performance of both ANN and CNN models as shown in **Table 11**. The role of optimization in enhancing convergence and generalization also contributed to their performance. These results exceeded those of traditional machine learning models found in Dhawan and Gitanjali (2024) **Table 13**, which lack the architecture to adapt to complex datasets. However, the time and computational resources required for training CNN were notably higher than for ANN, raising questions about scalability and deployment in resource-constrained environments.

Automatic feature extraction, a key advantage of CNN, was critical in elevating fraud detection accuracy. Unlike ANN, which relies on preprocessed features, CNN's convolutional layers extracted intricate patterns directly from raw data. This capability translated to a high ROC-AUC score of 100.00% for CNN, demonstrating superior differentiation between fraudulent and legitimate transactions. ANN, while powerful, depended more on preprocessing techniques such as SMOTE and normalization to achieve comparable results. In contrast, Adaboost and Random Forest (**See Table 14**) from Dhawan and Gitanjali, 2024, which rely entirely on manual feature engineering, had a poor performance than the recall of 100% recorded for both ANN and CNN (**See Table 11**), indicating its limitations in detecting minority fraud cases.

Table 13: Results from Traditional Machine Learning (Dhawan and Gitanjali, 2024).

Method	**Precision**	**Recall**	**Accuracy**
LR (Logistic Regression)	58.82%	91.84%	97.46%
RF (Random Forest)	96.38%	81.63%	99.96%
NB (Naive Bayes)	16.17%	83.00%	99.00%
MLP (Multi-Layer Perceptron)	79.21%	81.63%	99.93%

Table 14: Performance Metrics of Different Classification Techniques (Dhawan and Gitanjali, 2024).

Technique	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC**
Random Forest	99.90%	95%	77%	85%	94.29%
Adaboost	99.90%	78.19%	64.19%	70.50%	96.88%

4.5.2Comparison with Traditional Models

The comparative analysis revealed the distinct superiority of deep learning models over traditional methods like Logistic Regression and Random Forest. The deep learning models, with their high recall of 100% (**See Figures 22 and 23**), demonstrated their critical value in fraud detection systems where false negatives must be minimized.

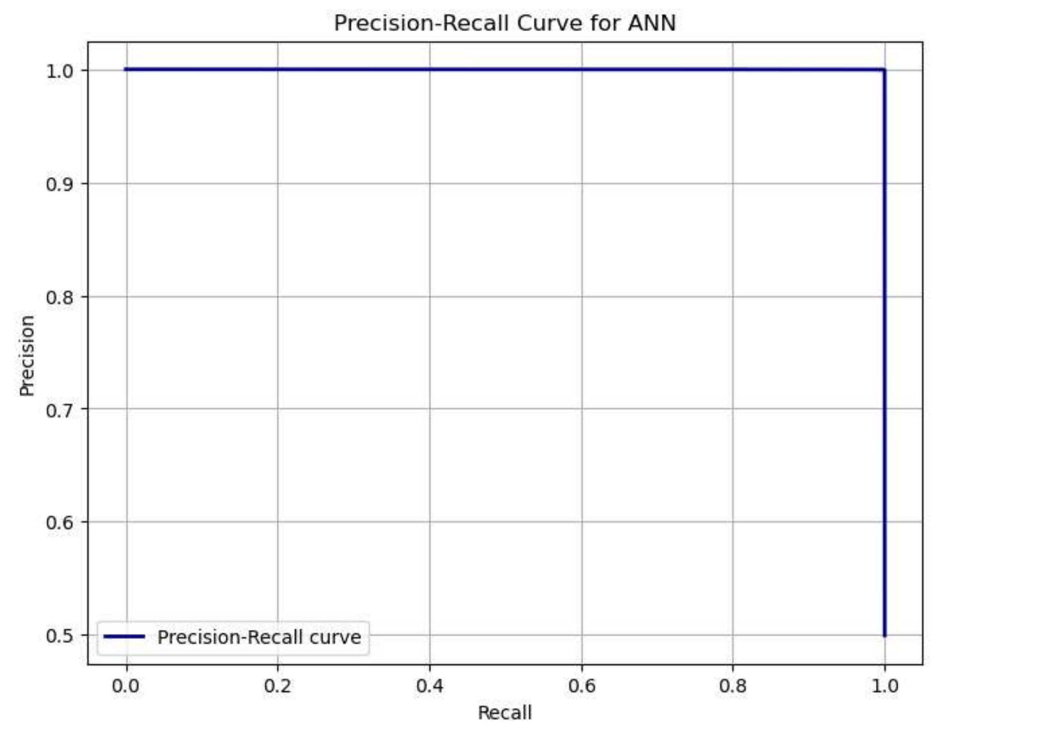


Figure 22: ANN Precision-Recall Curve

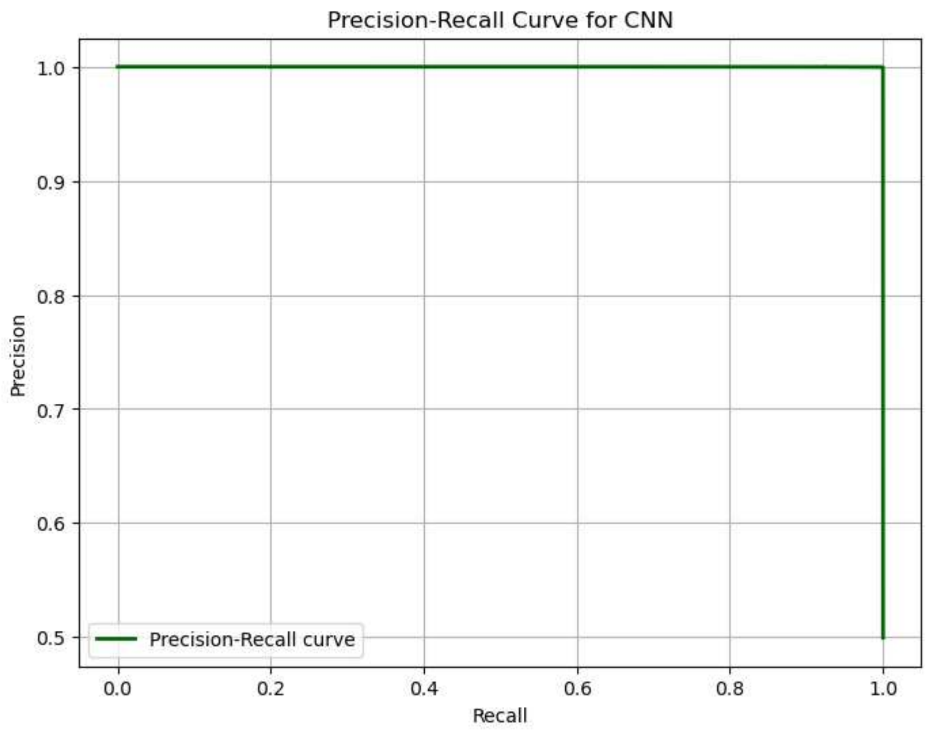


Figure 23: CNN Precision-Recall Curve

4.5.3 Critical Reflections

Despite the impressive results, limitations must be acknowledged. While deep learning models excelled in accuracy and recall, their dependence on large datasets and computational resources could hinder their adoption in low-resource settings. Additionally, the complexity of CNN architecture introduces challenges in interpretability, which is a crucial factor in regulatory and real-world financial systems. Although ANN offered slightly better interpretability, it required extensive preprocessing to achieve optimal performance.

In conclusion, this research demonstrated that deep learning models, particularly CNN, can revolutionize fraud detection by using automatic feature extraction and advanced optimization techniques. However, their adoption should be carefully weighed against practical constraints such as computational cost, scalability, and the need for interpretability, depending on the specific application requirements.

4.6 Chapter Summary

This chapter presented a comprehensive analysis of the results obtained from experiments conducted using Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN) for financial fraud detection. The preprocessing steps, which included data normalization, feature scaling, and the application of techniques like SMOTE, played a crucial role in enhancing the models' ability to identify fraudulent transactions.

The results were further compared to traditional machine learning models by integrating the findings from Dhawan and Gitanjali (2024), confirming that deep learning techniques, particularly ANN and CNN, provided superior performance. ANN demonstrated strong learning capabilities in detecting patterns within complex transaction data, while CNN exhibited an even higher potential for identifying intricate, spatial relationships within data, making it particularly adept for fraud detection tasks that require pattern recognition in multidimensional datasets. This comparative analysis aligns with the results observed in this practical study, where both ANN and CNN outperformed traditional methods but exhibited distinct trade-offs in terms of computational demand and interpretability.

Chapter V: Summary, Conclusion, and Future Work

5.1 Summary of Findings

This study investigated the application of Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN) to enhance financial fraud detection. Both models were developed, optimized, and evaluated to address the research objectives.

Key findings are the accuracy and recall achieved by both ANN and CNN in identifying fraudulent transactions and their suitability for high-stakes financial applications. ANN's simpler architecture and lower computational demands make it a choice for resource-constrained environments, while CNN's advanced feature extraction capabilities enable it to identify complex fraud patterns, offering generalization for dynamic fraud scenarios.

Optimization techniques, including the Adam optimizer for stable learning, and SMOTE for addressing class imbalance, proved instrumental in enhancing the models' performance. These methods ensured both ANN and CNN effectively handled imbalanced datasets, achieving reliable fraud detection.

Overall, this research consolidates the scalability, adaptability, and efficiency of deep learning models in developing fraud detection systems. The results validate their critical role in mitigating financial fraud and highlight the importance of selecting appropriate architectures based on specific operational requirements.

5.2 Contributions of the Study

This study makes several significant contributions, specifically through the application of deep learning techniques. The key contributions include:

- The research designed and optimized deep learning models, namely Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN), to improve fraud detection accuracy while minimizing false positives. These optimizations enhanced model reliability, addressing a critical challenge in fraud detection systems.

- Through applying CNN’s automatic feature extraction capabilities, the study demonstrated its ability to handle raw transaction data effectively. This reduces the need for manual feature engineering, making the models adaptable to complex and diverse fraud patterns.
- The research provided a comparison between deep learning models and traditional machine learning methods. The analysis highlighted the performance of ANN and CNN in dynamic fraud scenarios, pointing to their scalability, adaptability, and higher predictive accuracy.

5.3 Practical Implications and Industry Adoption

The adoption of advanced deep learning architectures like ANNs and CNNs, offers benefits for financial fraud detection. These models enhance precision in detecting fraudulent transactions while minimizing false positives. Their ability to automate feature extraction and enable real-time fraud detection streamlines operations, cuts down on manual reviews, and accelerates response times. Additionally, the cost savings from reduced fraud losses and labour demands, coupled with the capability to incorporate privacy-preserving measures to meet regulatory requirements, position these models as vital tools for improving security and customer trust.

For successful industry adoption, a phased approach is essential. Initial steps involve validating model performance through pilot studies using anonymized real-world datasets. Once validated, integration with existing systems via APIs ensures compatibility with legacy frameworks and transaction pipelines. Cloud-based deployments enhance scalability. Continuous updates through online learning mechanisms and regular model retraining are critical for adapting to evolving fraud patterns. Comprehensive training programs for fraud analysts and IT teams ensure proper maintenance and ethical application, with a focus on transparency and responsible use of AI-driven decisions.

Interpretability remains a significant challenge for stakeholder acceptance. Techniques like SHAP and LIME can provide clarity on model decisions by highlighting influential features, while Explainable AI frameworks offer insights to bridge the gap between technical outputs and human understanding. Transparent operations, supported by detailed logs and user-friendly dashboards, help build trust and meet regulatory requirements. Ethical considerations, including privacy-preserving measures and diverse model validation, are crucial for addressing stakeholder concerns, ensuring responsible and transparent deployment of these advanced fraud detection systems.

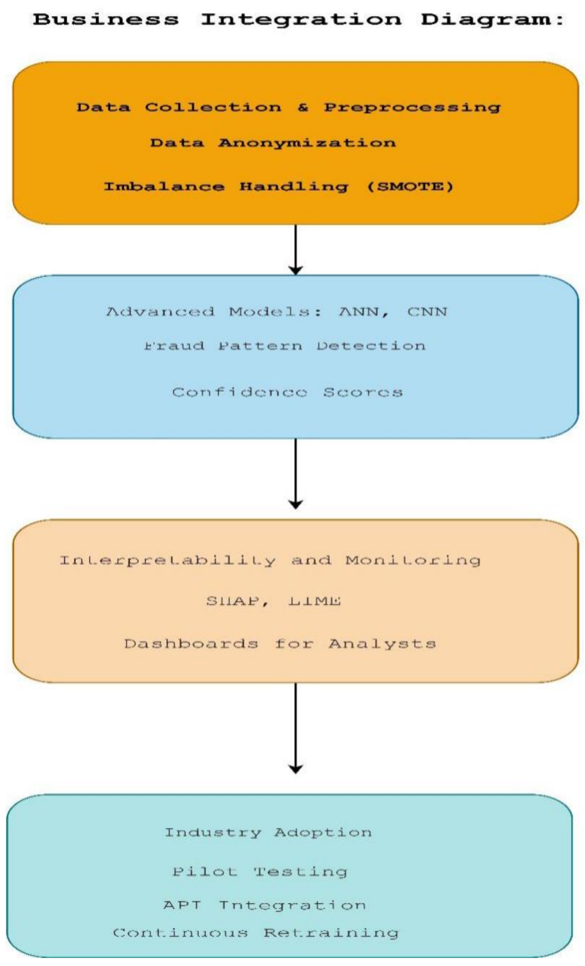


Figure 24: Business Integration Approach

5.4 Conclusion

This study details the potential of deep learning in financial fraud detection systems. Both Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN) demonstrated their ability to adapt to the complex and ever-evolving nature of fraudulent activities, addressing critical challenges such as improving accuracy, enhancing efficiency, and automating feature extraction. CNN displayed superior performance due to its advanced feature extraction capabilities, making it highly effective in detecting intricate fraud patterns directly from raw transaction data.

While the research emphasizes the strengths of these models, it also acknowledged key limitations, such as higher computational costs, especially for CNN, and challenges in model interpretability for non-technical stakeholders. Nonetheless, the findings affirm the feasibility of integrating advanced deep learning techniques into financial fraud detection systems, for adaptive solutions to bolster financial security.

This work contributes to the growing body of knowledge in AI-driven fraud detection, providing a foundation for future advancements aimed at refining model efficiency, scalability, and explainability.

5.5 Limitations of the Study

While the research successfully met its objectives, several challenges were identified, highlighting areas for improvement in deploying deep learning models for financial fraud detection:

- CNN models demonstrated superior performance but required significantly more computational resources, longer training times, and advanced hardware. This limits scalability, especially for smaller organizations with constrained resources.
- Both ANN and CNN operate as "black-box" models, making their decision-making processes difficult to interpret. This lack of explainability poses challenges for adoption in domains where transparency is critical, such as regulatory compliance and financial operations.
- ANN offered faster training and lower computational demands but lacked the feature extraction sophistication of CNN. Conversely, while CNN excelled in identifying complex patterns, its computational cost and implementation complexity remain barriers for widespread use.

Table 15 : Summary of Limitations and Trade-offs

Aspect	**Limitation**	**Trade-Off**
Computational Demand	High for CNN	Improved accuracy but requires advanced hardware
Model Interpretability	Lack of explainability for both ANN and CNN	Accurate models but limited transparency and stakeholder trust

To overcome these limitations, future studies could focus on:

- Integrating explainability techniques such as SHAP or LIME to make the models' decisions more transparent and actionable.
- Exploring innovative techniques to better handle extreme class imbalances and reduce residual biases.
- Developing lightweight yet efficient CNN architectures or using hardware acceleration to make high-performing models more accessible.

5.6 Future Work

Future work in financial fraud detection could focus on key areas:

1. Improving the representation of diverse fraud patterns through advanced synthetic data generation techniques. These methods could better address class imbalances and ensure more realistic simulation of rare fraudulent activities, further improving model performance.
2. Developing fraud-specific explainability frameworks tailored to the needs of non-technical stakeholders and regulatory bodies for clear adaptation.
3. Designing adaptive learning mechanisms that allow models to update in real time based on new fraud patterns. This would eliminate the need for manual retraining and ensure the system remains strong in dynamic environments.

4. Researching lightweight deep learning architectures, such as MobileNet or TinyML frameworks, to reduce computational requirements. These scalable solutions would make advanced fraud detection systems more accessible to smaller institutions with limited resources.

5. Investigating the integration of deep learning models with complementary technologies like blockchain, which can provide immutable transaction records, and anomaly detection systems to create a layered and comprehensive defence strategy.

Table 16: Summary of Proposed Directions

Area	**Proposed Approach**
Improved Data Handling	Advanced synthetic data generation for better representation of fraud patterns.
Model Interpretability	Fraud-specific explainability frameworks to enhance trust and compliance.
Real-time Adaptation	Adaptive learning mechanisms for continuous updates without manual retraining.
Scalable Solutions	Lightweight architectures to reduce computational costs and improve accessibility.
Integration with Systems	Combining deep learning with blockchain and other mechanisms for a holistic defence.

References

Abdallah, A., Maarof, M.A. and Zainal, A. (2016). Fraud detection system: A survey. *Journal of Network and Computer Applications*, [online] 68, pp.90–113. doi:https://doi.org/10.1016/j.jnca.2016.04.007.

Abdullahi, M.J. (2024). Leveraging Machine Learning in Classifying Fraudulent and Legitimate Transactions in Banking Sector. *Ilorin Journal of Computer Science and Information Technology*, [online] 7(1), pp.40–64. Available at: https://iljcsit.com.ng/index.php/ILJCSIT/article/view/103 [Accessed 3 Dec. 2024].

Abhaya, A. and Patra, B.Kr. (2023). An efficient method for autoencoder based outlier detection. *Expert Systems with Applications*, 213, p.118904. doi:https://doi.org/10.1016/j.eswa.2022.118904.

Afjal, M., Salamzadeh, A. and Dana, L.-P. (2023). Financial Fraud and Credit Risk: Illicit Practices and Their Impact on Banking Stability. *Journal of Risk and Financial Management*, [online] 16(9), p.386. doi:https://doi.org/10.3390/jrfm16090386.

Ahsan, M.M., Luna, S.A. and Siddique, Z. (2022). Machine-Learning-Based Disease Diagnosis: A Comprehensive Review. *Healthcare*, [online] 10(3), p.541. doi:https://doi.org/10.3390/healthcare10030541.

Alharbi, A., Alshammari, M., Okon, O.D., Alabrah, A., Rauf, H.T., Alyami, H. and Meraj, T. (2022). A Novel text2IMG Mechanism of Credit Card Fraud Detection: A Deep Learning Approach. *Electronics*, 11(5), p.756. doi:https://doi.org/10.3390/electronics11050756.

Arroyabe, M.F., Carlos F.A. Arranz, Fernandez, I. and Carlos, J. (2024). Revealing the Realities of Cybercrime in Small and Medium Enterprises: Understanding Fear and Taxonomic Perspectives. *Computers & security*, 141(103826), pp.103826–103826. doi:https://doi.org/10.1016/j.cose.2024.103826.

Baduge, S.K., Thilakarathna, S., Perera, J.S., Arashpour, M., Sharafi, P., Teodosio, B., Shringi, A. and Mendis, P. (2022). Artificial intelligence and smart vision for building and construction 4.0: Machine and deep learning methods and applications. *Automation in Construction*, [online] 141(2), p.104440. Available at: https://www.sciencedirect.com/science/article/pii/S0926580522003132.

Bartsiotas, G. and Achamkulangare, G. (2016). *FRAUD PREVENTION, DETECTION AND RESPONSE IN UNITED NATIONS SYSTEM ORGANIZATIONS*. [online] Available at: https://www.unjiu.org/sites/www.unjiu.org/files/jiu_document_files/products/en/reports-notes/JIU%20Products/JIU_REP_2016_4_English.pdf.

Basheer, N., Islam, S., Mohammed and Spyridon Papastergiou (2024). Adoption of Deep-Learning Models for Managing Threat in API Calls with Transparency Obligation Practice for Overall Resilience. *Sensors*, [online] 24(15), pp.4859–4859. doi:https://doi.org/10.3390/s24154859.

Bello, A. and Olufemi, K. (2024). Artificial intelligence in fraud prevention: Exploring techniques and applications challenges and opportunities. *Computer science & IT research journal*, 5(6), pp.1505–1520. doi:https://doi.org/10.51594/csitrj.v5i6.1252.

Bello, O., Folorunso, A., Ejiofor, O., Zainab Budale, F., Adebayo, K., Olayemi, A., Babatunde and Bello, C. (2023). Machine Learning Approaches for Enhancing Fraud Prevention in Financial Transactions. *International Journal of Management Technology*, [online] 10(1), pp.85–109. doi:https://doi.org/10.37745/ijmt.2013/vol10n185109.

Bello, O., Folorunso, A., Ogundipe, A., Kazeem, O., Zainab, F., Ejiofor, O. and Bello, O. (2022). Enhancing Cyber Financial Fraud Detection Using Deep Learning Techniques: A Study on Neural Networks and Anomaly Detection. *International Journal of Network and Communication Research*, [online] 7(1), pp.90–113. doi:https://doi.org/10.37745/ijncr.16/vol7n190113.

Bello, O., None Courage Idemudia and Iyelolu, V. (2024). Implementing machine learning algorithms to detect and prevent financial fraud in real-time. *Computer science & IT research journal*, 5(7), pp.1539–1564. doi:https://doi.org/10.51594/csitrj.v5i7.1274.

Bhowmik, A., Sannigrahi, M., Chowdhury, D., Dwivedi, A.D. and Rao Mukkamala, R. (2022). DBNex: Deep Belief Network and Explainable AI based Financial Fraud Detection. *2022 IEEE International Conference on Big Data (Big Data)*. doi:https://doi.org/10.1109/bigdata55660.2022.10020494.

Bolton, R.J., Hand, D.J., Provost, F., Breiman, L., Bolton, R.J. and Hand, D.J. (2002). Statistical Fraud Detection: A Review. *Statistical Science*, 17(3), pp.235–255. doi:https://doi.org/10.1214/ss/1042727940.

Camacho, J., Wasielewska, K., Fuentes-García, M. and Rodríguez-Gómez, R. (2023). *Quality In / Quality Out: Assessing Data quality in an Anomaly Detection Benchmark*. [online] arXiv.org. Available at: https://arxiv.org/abs/2305.19770 [Accessed 3 Dec. 2024].

Charitou, C., Dragicevic, S. and D'avila Garcez, A. (2024). *Synthetic Data Generation for Fraud Detection using GANs*. [online] Available at: https://arxiv.org/pdf/2109.12546.

Cherif, A., Badhib, A., Ammar, H., Alshehri, S., Kalkatawi, M. and Imine, A. (2022). Credit card fraud detection in the era of disruptive technologies: A systematic review. *Journal of King Saud University - Computer and Information Sciences*, [online] 35(1). doi:https://doi.org/10.1016/j.jksuci.2022.11.008.

Chew, P., Yang, Y. and Lee, B.-G. (2023). Enhancing Financial Fraud Detection through Addressing Class Imbalance Using Hybrid SMOTE-GAN Techniques. *International Journal of Financial Studies*, 11(3), pp.110–110. doi:https://doi.org/10.3390/ijfs11030110.

Cornell Law School (2019). *Credit Card Fraud*. [online] LII / Legal Information Institute. Available at: https://www.law.cornell.edu/wex/credit_card_fraud.

Detthamrong, U., Chansanam, W., Boongoen, T. and Iam-On, N. (2024). Enhancing Fraud Detection in Banking using Advanced Machine Learning Techniques. *International Journal of Economics and Financial Issues*, 14(5), pp.177–184. doi:https://doi.org/10.32479/ijefi.16613.

Dal Pozzolo, A., Caelen, O., Johnson, R.A. & Bontempi, G. (2015) 'Calibrating Probability with Undersampling for Unbalanced Classification', *Symposium on Computational Intelligence and Data Mining (CIDM)*, IEEE.

Dhawan, R. and Gitanjali, ., 2024. *A Comparative Performance Analysis of Machine Learning Techniques for Credit Card Fraud Detection*. [online] Available at: https://ssrn.com/abstract=4915083 [Accessed 2 December 2024].

Du, X., Chen, J., Yu, J., Li, S. and Tan, Q. (2024). Generative adversarial nets for unsupervised outlier detection. *Expert Systems with Applications*, 236, p.121161. doi:https://doi.org/10.1016/j.eswa.2023.121161.

El Kafhali, S., Tayebi, M. and Sulimani, H. (2024). An Optimized Deep Learning Approach for Detecting Fraudulent Transactions. *Information*, [online] 15(4), p.227. doi:https://doi.org/10.3390/info15040227.

Elouataoui, W., El Mendili, S. and Gahi, Y. (2023). An Automated Big Data Quality Anomaly Correction Framework Using Predictive Analysis. *Data*, [online] 8(12), p.182. doi:https://doi.org/10.3390/data8120182.

Faisal, N.A., Nahar, J., Sultana, N. and Mintoo, A.A. (2024). Fraud Detection In Banking Leveraging Ai To Identify And Prevent Fraudulent Activities In Real-Time. *Non human journal.*, [online] 1(01), pp.181–197. doi:https://doi.org/10.70008/jmldeds.v1i01.53.

Femila Roseline, J., Naidu, G., Samuthira Pandi, V., Alamelu alias Rajasree, S. and Mageswari, Dr.N. (2022a). Autonomous credit card fraud detection using machine learning approach☆. *Computers and Electrical Engineering*, 102, p.108132. doi:https://doi.org/10.1016/j.compeleceng.2022.108132.

Femila Roseline, J., Naidu, G., Samuthira Pandi, V., Alamelu alias Rajasree, S. and Mageswari, Dr.N. (2022b). Autonomous credit card fraud detection using machine learning approach☆. *Computers and Electrical Engineering*, 102, p.108132. doi:https://doi.org/10.1016/j.compeleceng.2022.108132.

Georgios Zioviris, Kostas Kolomvatsos and Stamoulis, G. (2024). An intelligent sequential fraud detection model based on deep learning. *˜The œJournal of supercomputing/Journal of supercomputing*. doi:https://doi.org/10.1007/s11227-024-06030-y.

Gilian Schrijver, Sarmah, D.K. and El-hajj, M. (2024). Automobile Insurance Fraud Detection Using Data Mining: A Systematic Literature Review. *Intelligent systems with applications*, pp.200340–200340. doi:https://doi.org/10.1016/j.iswa.2024.200340.

Gold, N., Udeh, A., Adaga, M., DaraOjimba, D. and Osato, N. (2024). ETHICAL CONSIDERATIONS IN DATA COLLECTION AND ANALYSIS: a REVIEW:

INVESTIGATING ETHICAL PRACTICES AND CHALLENGES IN MODERN DATA COLLECTION AND ANALYSIS. *International Journal of Applied Research in Social Sciences*, 6(1), pp.1–22. doi:https://doi.org/10.51594/ijarss.v6i1.688.

Halima Oluwabunmi Bello, Adebimpe Bolatito Ige and Maxwell Nana Ameyaw (2024). Adaptive machine learning models: Concepts for real-time financial fraud prevention in dynamic environments. *World Journal of Advanced Engineering Technology and Sciences*, [online] 12(2), pp.021–034. doi:https://doi.org/10.30574/wjaets.2024.12.2.0266.

Hamilton, K. (2024). *Fraud on the Rise: New ACFE Report*. [online] Brady Ware CPAs. Available at: https://bradyware.com/new-acfe-report-exposes-rising-fraud-costs-trends/.

Hilal, W., Gadsden, S.A. and Yawney, J. (2021). A Review of Anomaly Detection Techniques and Applications in Financial Fraud. *Expert Systems with Applications*, 193(1), p.116429. doi:https://doi.org/10.1016/j.eswa.2021.116429.

https://www.kaggle.com/code/rehamh/credit-card-fraud-detection

https://app.diagrams.net/

Ibomoiye Domor Mienye and Nobert Jere (2024). Deep Learning for Credit Card Fraud Detection: A Review of Algorithms, Challenges, and Solutions. *IEEE Access*, 12, pp.96893–96910. doi:https://doi.org/10.1109/access.2024.3426955.

Ishfaq Hussain Rather and Kumar, S. (2023). Generative adversarial network based synthetic data training model for lightweight convolutional neural networks. *Multimedia Tools and Applications*. doi:https://doi.org/10.1007/s11042-023-15747-6.

Jaiswal, Rashi; Singh, Brijendra (2014a). Financial fraud prevention with synthetic data generation using gan. *Arya Bhatta Journal of Mathematics and Informatics*, [online] 14(2), pp.161–166. Available at: https://www.indianjournals.com/ijor.aspx?target=ijor:abjmmi&volume=14&issue=2&article=004 [Accessed 3 Dec. 2024].

Jaiswal, Rashi; Singh, Brijendra (2014b). Financial fraud prevention with synthetic data generation using gan. *Arya Bhatta Journal of Mathematics and Informatics*, [online] 14(2), pp.161–166. Available at:

https://www.indianjournals.com/ijor.aspx?target=ijor:abjmmi&volume=14&issue=2&article=004 [Accessed 3 Dec. 2024].

Johnson, E. (2024). Automating Model Retraining in DevOps Pipelines with MLOps: Addressing Model Drift and Data Evolution. *Distributed Learning and Broad Applications in Scientific Research*, [online] 10, pp.317–323. Available at: https://dlabi.org/index.php/journal/article/view/157 [Accessed 3 Dec. 2024].

Kamal Berahmand, Fatemeh Daneshfar, Elaheh Sadat Salehi, Li, Y. and Xu, Y. (2024). Autoencoders and their applications in machine learning: a survey. *Artificial Intelligence Review*, 57(2). doi:https://doi.org/10.1007/s10462-023-10662-6.

Kampanart Huanbutta, Kanokporn Burapapadh, PakornKraisit, PornsakSriamornsak, ThittapornGanokratana, KittipatSuwanpitak and Tanikan Sangnim (2024). The Artificial Intelligence-Driven Pharmaceutical Industry: A Paradigm Shift in Drug Discovery, Formulation Development, Manufacturing, Quality Control, and Post-Market Surveillance. *European Journal of Pharmaceutical Sciences*, [online] pp.106938–106938. doi:https://doi.org/10.1016/j.ejps.2024.106938.

Kasasbeh, B., Aldabaybah, B. and Ahmad, H. (2022). Multilayer perceptron artificial neural networks-based model for credit card fraud detection. *Indonesian Journal of Electrical Engineering and Computer Science*, 26(1), p.362. doi:https://doi.org/10.11591/ijeecs.v26.i1.pp362-373.

Khalid, A.R., Owoh, N., Uthmani, O., Ashawa, M., Osamor, J. and Adejoh, J. (2024). Enhancing Credit Card Fraud Detection: An Ensemble Machine Learning Approach. *Big Data and Cognitive Computing*, [online] 8(1), p.6. doi:https://doi.org/10.3390/bdcc8010006.

Kim, J., Jung, H. and Kim, W. (2022). Sequential Pattern Mining Approach for Personalized Fraudulent Transaction Detection in Online Banking. *Sustainability*, 14(15), p.9791. doi:https://doi.org/10.3390/su14159791.

Labs, A. (2024). *What Is Fraud Detection?* [online] https://www.arkoselabs.com/explained/what-is-fraud-detection/. Available at: https://www.arkoselabs.com/explained/what-is-fraud-detection/.

Lee, J., Jung, D., Moon, J. and Rho, S. (2024). Advanced R-GAN: Generating anomaly data for improved detection in imbalanced datasets using regularized generative adversarial networks. *Alexandria Engineering Journal*, [online] 111, pp.491–510. doi:https://doi.org/10.1016/j.aej.2024.10.084.

Liang, P., Song, B., Zhan, X., Chen, Z. and Yuan, J. (2024). Automating the training and deployment of models in MLOps by integrating systems with machine learning. *Applied and Computational Engineering*, [online] 67(1), pp.1–7. doi:https://doi.org/10.54254/2755-2721/67/20240690.

Lindemulder, G. and Kosinski, M. (2024). *What Is Fraud Detection? | IBM*. [online] www.ibm.com. Available at: https://www.ibm.com/topics/fraud-detection.

Liu, D. (2024). *Contemporary Model Compression on Large Language Models Inference*. [online] arXiv.org. Available at: https://arxiv.org/abs/2409.01990 [Accessed 3 Dec. 2024].

Ľubomír Čunderlík (2021). Fraudulent Schemes in the Financial Market (Financial Pyramids) – Detection and Prevention. *Financial Law Review*, [online] 2021(Issue 21 (1)/ 2021), pp.16–30. Available at: https://ejournals.eu/en/journal/financial-law-review/article/fraudulent-schemes-in-the-financial-market-financial-pyramids-detection-and-prevention [Accessed 3 Dec. 2024].

Mahalaxmi, G., Donald, A.D. and Srinivas, T.A.S. (2023). A Short Review of Python Libraries and Data Science Tools. *South Asian Research Journal of Engineering and Technology*, 5(1), pp.1–5. doi:https://doi.org/10.36346/sarjet.2023.v05i01.001.

Manda, V.T., Dheeraj Kondapalli, Malla, A. sai, M, J.N. and Charan, Y. (2024). Imbalanced Data Challenges and Their Resolution to Improve Fraud Detection in Credit Card Transactions. *Research Square (Research Square)*. [online] doi:https://doi.org/10.21203/rs.3.rs-3962043/v1.

Marazqah Btoush, E.A.L., Zhou, X., Gururajan, R., Chan, K.C., Genrich, R. and Sankaran, P. (2023). A systematic review of literature on credit card cyber fraud detection using machine and deep learning. *PeerJ Computer Science*, 9, p.e1278. doi:https://doi.org/10.7717/peerj-cs.1278.

Md Kamrul Hasan Chy (2024). Proactive fraud defense: Machine learning's evolving role in protecting against online fraud. *World Journal of Advanced Research and Reviews*, 23(3), pp.1580–1589. doi:https://doi.org/10.30574/wjarr.2024.23.3.2811.

Md. Mahmudul Hasan (2024). Understanding Model Predictions: A Comparative Analysis of SHAP and LIME on Various ML Algorithms. *Journal of Scientific and Technological Research*, 5(1), pp.17–26. doi:https://doi.org/10.59738/jstr.v5i1.23(17-26).eaqr5800.

Mienye, I.D. and Swart, T.G. (2024). A Hybrid Deep Learning Approach with Generative Adversarial Network for Credit Card Fraud Detection. *Technologies*, 12(10), p.186. doi:https://doi.org/10.3390/technologies12100186.

Mienye, I.D., Swart, T.G. and Obaido, G. (2024). Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications. *Information*, [online] 15(9), pp.517–517. doi:https://doi.org/10.3390/info15090517.

Mumuni, A. and Mumuni, F. (2024). Automated data processing and feature engineering for deep learning and big data applications: a survey. *Journal of Information and Intelligence*. [online] doi:https://doi.org/10.1016/j.jiixd.2024.01.002.

Nama, F.A. and Obaid, A.J. (2024). Financial Fraud Identification Using Deep Learning Techniques. *Al-Salam Journal for Engineering and Technology*, 3(1), pp.141–147. doi:https://doi.org/10.55145/ajest.2024.03.01.012.

Nikolados, E.-M. and Oyarzún, D.A. (2023). Deep learning for optimization of protein expression. *Current Opinion in Biotechnology*, [online] 81, p.102941. doi:https://doi.org/10.1016/j.copbio.2023.102941.

Nisa Aulia Saputra, Lala Septem Riza, Setiawan, A. and Hamidah, I. (2024). Choosing the appropriate deep learning method: A systematic review. *Decision Analytics Journal*, [online] 12, pp.100489–100489. doi:https://doi.org/10.1016/j.dajour.2024.100489.

Oh, J., Son, S., Kwon, D., Kim, M., Park, Y. and Park, Y. (2024). Design of Secure and Privacy-Preserving Data Sharing Scheme Based on Key Aggregation and Private Set Intersection in Medical Information System. *Mathematics*, 12(11), pp.1717–1717. doi:https://doi.org/10.3390/math12111717.

Oluwatoyin, J.A. and Akinola, S. (2024). Real Time Credit Card Fraud Detection and Reporting System Using Machine Learning. *European Journal of Computer Science and Information Technology*, 12(4), pp.36–56. doi:https://doi.org/10.37745/ejcsit.2013/vol12n43656.

Oyewole, T., Oguejiofor, B. and Bakare, S. (2024). DATA PRIVACY LAWS AND THEIR IMPACT ON FINANCIAL TECHNOLOGY COMPANIES: A REVIEW. *Computer science & IT research journal*, 5(3), pp.628–650. doi:https://doi.org/10.51594/csitrj.v5i3.911.

Pan, E. (2024). Machine Learning in Financial Transaction Fraud Detection and Prevention. *Transactions on Economics, Business and Management Research*, [online] 5, pp.243–249. doi:https://doi.org/10.62051/16r3aa10.

Pinto, S.O. and Sobreiro, V.A. (2022). Literature review: Anomaly detection approaches on digital business financial systems. *Digital Business*, p.100038. doi:https://doi.org/10.1016/j.digbus.2022.100038.

Rizvi, S.R.Z. (2021). *ROLE OF BIG DATA IN FINANCIAL INSTITUTIONS FOR FINANCIAL FRAUD*. [online] papers.ssrn.com. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3800777.

Rojan, Z. (2024). Financial Fraud Detection Based on Machine and Deep Learning: A Review. *Indonesian Journal of Computer Science*, 13(3). doi:https://doi.org/10.33022/ijcs.v13i3.4059.

Saha, P., Shubhra Aanand, Shah, P., Ritesh Khatwani, Pradip Kumar Mitra and Sekhar, R. (2023). Comparative Analysis of ML Algorithms for Fraud Detection in Financial Transactions. *IEEE Explore*, 119(2137). doi:https://doi.org/10.1109/icaeeci58247.2023.10370930.

Sarker, I.H. (2021a). Deep Learning: a Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN Computer Science*, 2(6). doi:https://doi.org/10.1007/s42979-021-00815-1.

Sarker, I.H. (2021b). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, [online] 2(3), pp.1–21. doi:https://doi.org/10.1007/s42979-021-00592-x.

Sarker, I.H., Janicke, H., Mohamed Amine Ferrag and Alsharif Abuadbba (2024). Multi-aspect rule-based AI: Methods, taxonomy, challenges and directions toward automation, intelligence, and transparent cybersecurity modelling for critical infrastructures. *Internet of Things*, pp.101110–101110. doi:https://doi.org/10.1016/j.iot.2024.101110.

Shams Forruque Ahmed, Bin, S., Hassan, M., Mahtabin Rodela Rozbu, TaoseefIshtiak, Rafa, N., M. Mofijur, Ali and Gandomi, A.H. (2023a). Deep learning modelling techniques: current progress, applications, advantages, and challenges. *Artificial Intelligence Review*, 56(1). doi:https://doi.org/10.1007/s10462-023-10466-8.

Shams Forruque Ahmed, Bin, S., Hassan, M., Mahtabin Rodela Rozbu, TaoseefIshtiak, Rafa, N., M. Mofijur, Ali and Gandomi, A.H. (2023b). Deep learning modelling techniques: current progress, applications, advantages, and challenges. *Artificial Intelligence Review*, 56(1). doi:https://doi.org/10.1007/s10462-023-10466-8.

Sulaiman, S.S., Ibraheem Nadher and Hameed, S.M. (2024). Credit Card Fraud Detection Using Improved Deep Learning Models. *Computers, materials & continua (Print)*, 78(1), pp.1049–1069. doi:https://doi.org/10.32604/cmc.2023.046051.

Tan, K.-L., Ivy S.H. Hii, Huang, Y. and Yan, Y. (2024). Reducing dishonest disclosures during expense reimbursement: investigating the predictive power of the technology acceptance model with a corporate governance perspective. *Journal of Accounting & Organizational Change*, 213(213). doi:https://doi.org/10.1108/jaoc-01-2023-0019.

Thimonier, H., Fabrice Popineau, Rimmel, A., Doan, B.-L. and Daniel, F. (2024). Comparative Evaluation of Anomaly Detection Methods for Fraud Detection in Online Credit Card Payments. *Lecture notes in networks and systems*, pp.37–50. doi:https://doi.org/10.1007/978-981-97-4581-4_4.

Tito, A., Sugumar Anandakumar, Pare, A., Chandrasekar, V. and Natarajan Venkatachalapathy (2021). Modeling of process parameters to predict the efficiency of shallots stem cutting machine using multiple regression and artificial neural network. *Journal of Food Process Engineering*, 45(6). doi:https://doi.org/10.1111/jfpe.13944.

vardhan, Mr.A.V., P V S N M L Ankitha, M., Sri, P.D., Battula, M. and Priyanka, M.V.T.S. (2024). Anomaly Detection in Credit Card Transactions using Autoencoders. *IJARCCE*, [online] 13(3). doi:https://doi.org/10.17148/ijarcce.2024.13320.

Xu, Z., Shen, D., Nie, T. and Kou, Y. (2020). A hybrid sampling algorithm combining M-SMOTE and ENN based on Random forest for medical imbalanced data. *Journal of Biomedical Informatics*, [online] 107, p.103465. doi:https://doi.org/10.1016/j.jbi.2020.103465.

Yang, Z., Wang, Y., Shi, H. and Qiu, Q. (2024). Leveraging Mixture of Experts and Deep Learning-Based Data Rebalancing to Improve Credit Fraud Detection. *Big Data and Cognitive Computing*, 8(11), p.151. doi:https://doi.org/10.3390/bdcc8110151.

Yun, J., Young Hoon Cho, Sang Min Lee, Hwang, J., Jae Seung Lee, Oh, Y.-M., Lee, S.-D., Loh, L.-C., Ong, C.-K., Joon Beom Seo and Kim, N. (2021). Deep radiomics-based survival prediction in patients with chronic obstructive pulmonary disease. *Scientific reports*, 11(1). doi:https://doi.org/10.1038/s41598-021-94535-4.

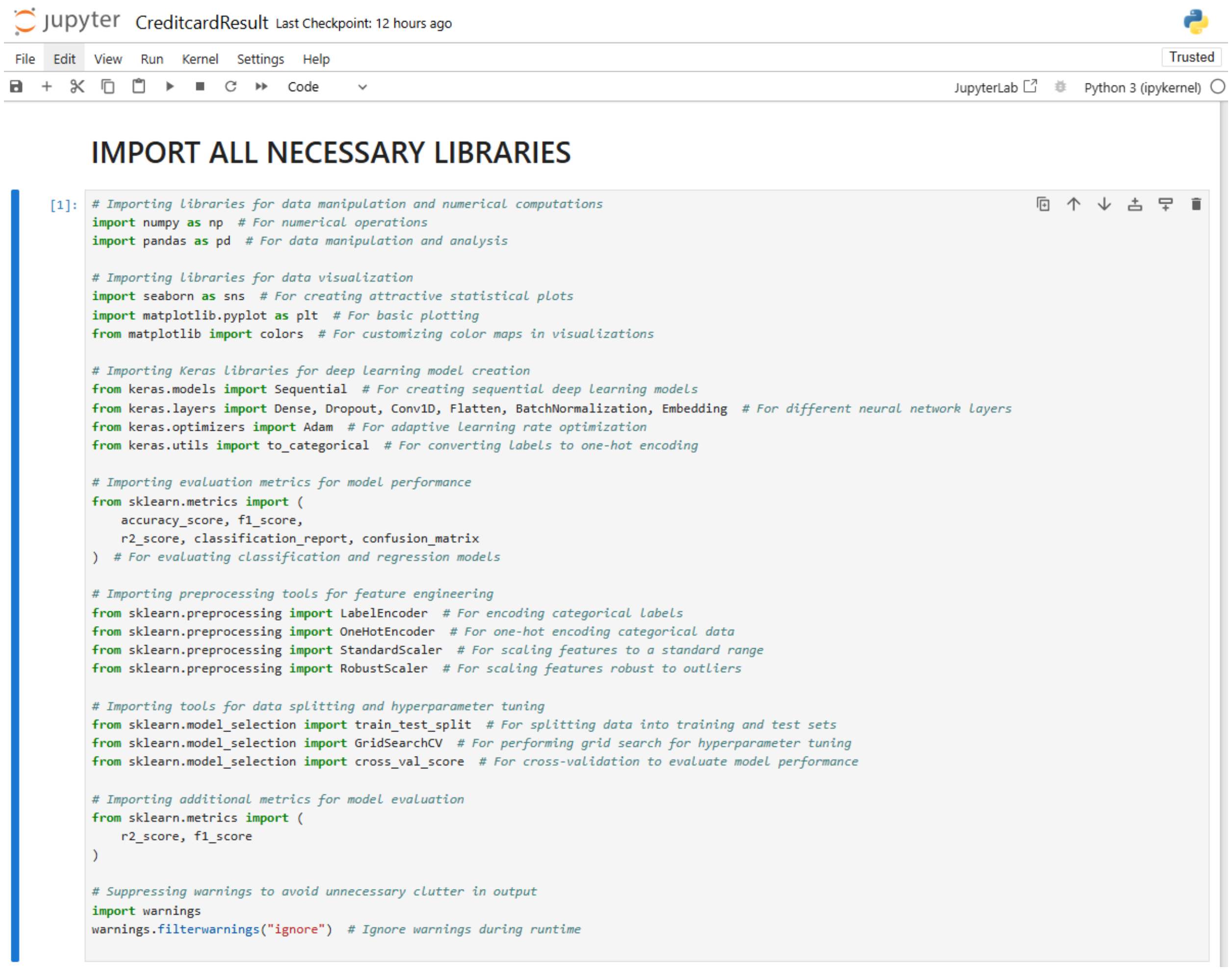
Zhang, Z., Zhou, X., Zhang, X., Wang, L. and Wang, P. (2018). A Model Based on Convolutional Neural Network for Online Transaction Fraud Detection. *Security and Communication Networks*, 2018, pp.1–9. doi:https://doi.org/10.1155/2018/5680264.

Zhu, X., Ao, X., Qin, Z., Chang, Y., Liu, Y., He, Q. and Li, J. (2021). Intelligent Financial Fraud Detection Practices in Post-Pandemic Era: A Survey. *The Innovation*, 2(4), p.100176. doi:https://doi.org/10.1016/j.xinn.2021.100176.

Appendix: Supporting Documentation

Appendix A: Code Snippets

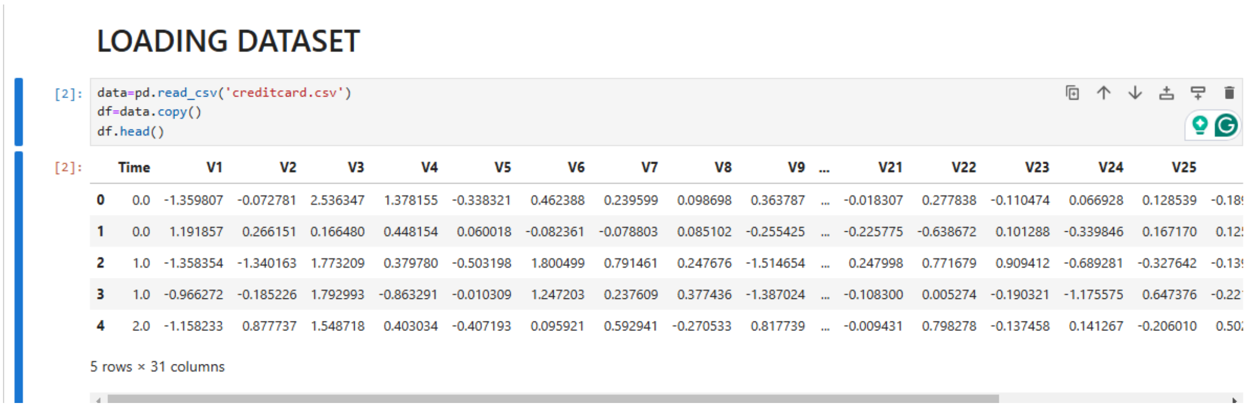
A.1 Preprocessing Code


jupyter
CreditcardResult Last Checkpoint: 12 hours ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

IMPORT ALL NECESSARY LIBRARIES

```
[1]: # Importing libraries for data manipulation and numerical computations
import numpy as np  # For numerical operations
import pandas as pd  # For data manipulation and analysis

# Importing libraries for data visualization
import seaborn as sns  # For creating attractive statistical plots
import matplotlib.pyplot as plt  # For basic plotting
from matplotlib import colors  # For customizing color maps in visualizations

# Importing Keras libraries for deep learning model creation
from keras.models import Sequential  # For creating sequential deep learning models
from keras.layers import Dense, Dropout, Conv1D, Flatten, BatchNormalization, Embedding  # For different neural network layers
from keras.optimizers import Adam  # For adaptive learning rate optimization
from keras.utils import to_categorical  # For converting labels to one-hot encoding

# Importing evaluation metrics for model performance
from sklearn.metrics import (
    accuracy_score, f1_score,
    r2_score, classification_report, confusion_matrix
)  # For evaluating classification and regression models

# Importing preprocessing tools for feature engineering
from sklearn.preprocessing import LabelEncoder  # For encoding categorical labels
from sklearn.preprocessing import OneHotEncoder  # For one-hot encoding categorical data
from sklearn.preprocessing import StandardScaler  # For scaling features to a standard range
from sklearn.preprocessing import RobustScaler  # For scaling features robust to outliers

# Importing tools for data splitting and hyperparameter tuning
from sklearn.model_selection import train_test_split  # For splitting data into training and test sets
from sklearn.model_selection import GridSearchCV  # For performing grid search for hyperparameter tuning
from sklearn.model_selection import cross_val_score  # For cross-validation to evaluate model performance

# Importing additional metrics for model evaluation
from sklearn.metrics import (
    r2_score, f1_score
)

# Suppressing warnings to avoid unnecessary clutter in output
import warnings
warnings.filterwarnings("ignore")  # Ignore warnings during runtime
```

Figure A1: Importing Necessary Libraries



LOADING DATASET

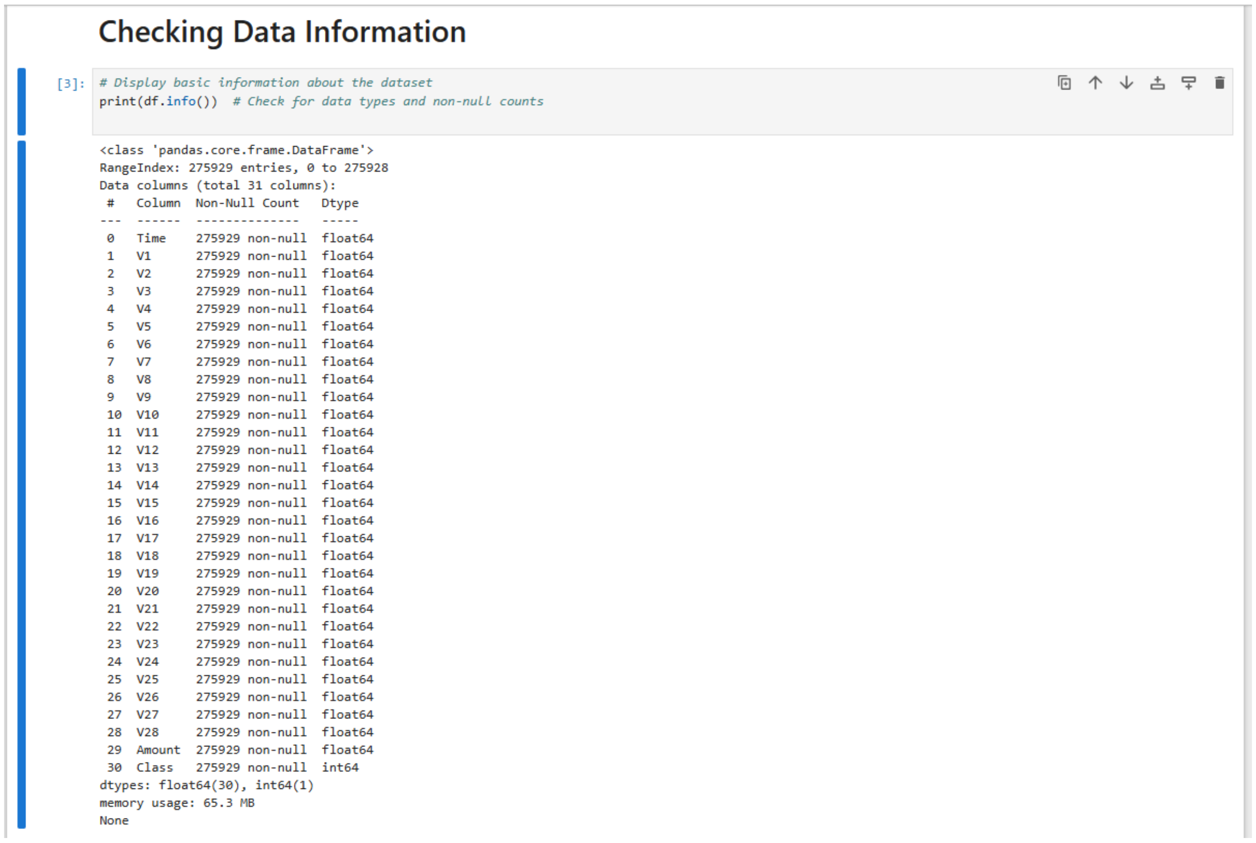
[2]:
```
data=pd.read_csv('creditcard.csv')
df=data.copy()
df.head()
```

[2]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.18
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.12
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.13
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.22
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.50

5 rows × 31 columns

Figure A2: Loading Dataset 1

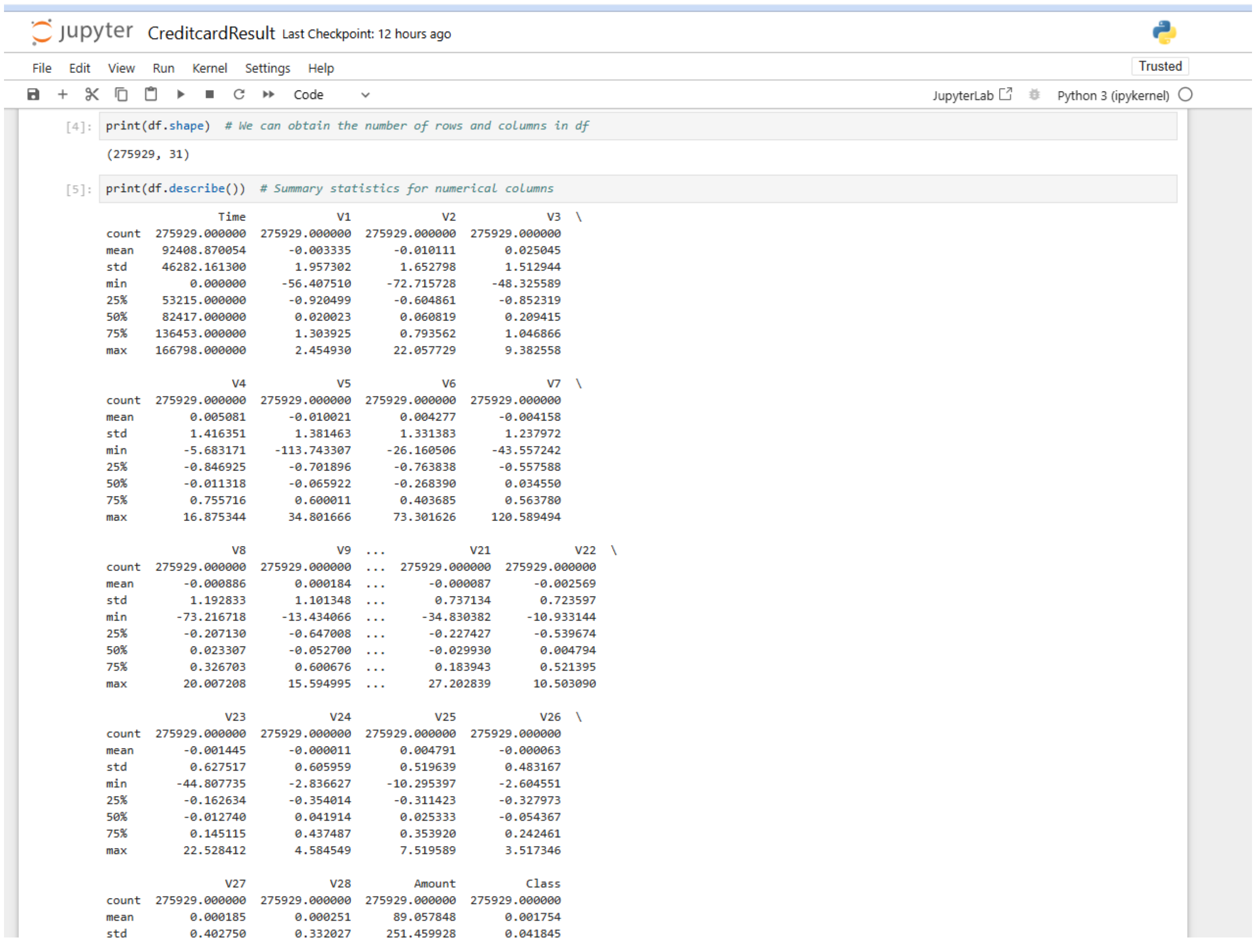


Checking Data Information

[3]:
```
# Display basic information about the dataset
print(df.info())  # Check for data types and non-null counts
```

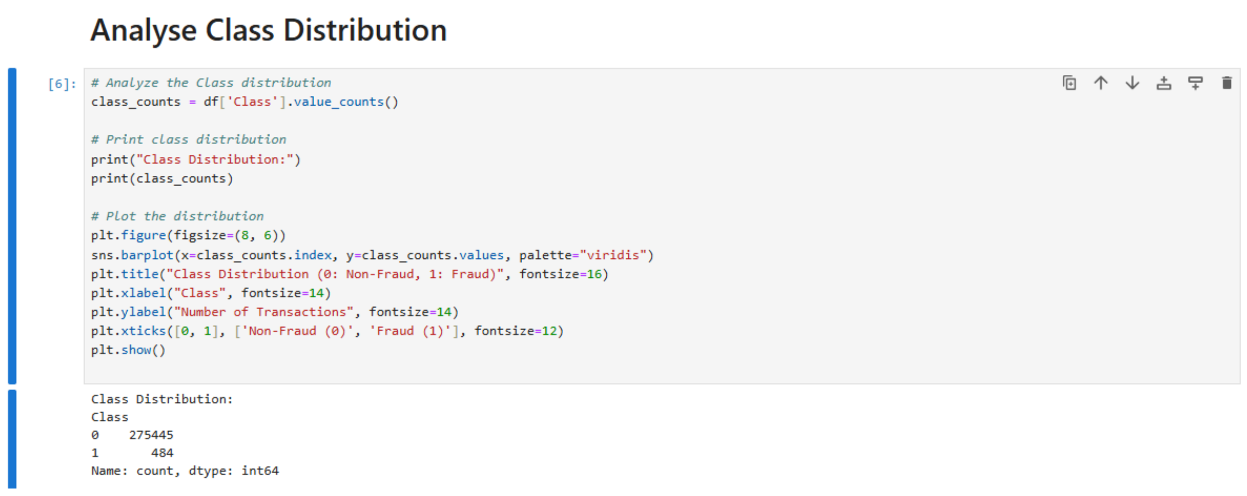
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 275929 entries, 0 to 275928
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    275929 non-null  float64
 1   V1      275929 non-null  float64
 2   V2      275929 non-null  float64
 3   V3      275929 non-null  float64
 4   V4      275929 non-null  float64
 5   V5      275929 non-null  float64
 6   V6      275929 non-null  float64
 7   V7      275929 non-null  float64
 8   V8      275929 non-null  float64
 9   V9      275929 non-null  float64
 10  V10     275929 non-null  float64
 11  V11     275929 non-null  float64
 12  V12     275929 non-null  float64
 13  V13     275929 non-null  float64
 14  V14     275929 non-null  float64
 15  V15     275929 non-null  float64
 16  V16     275929 non-null  float64
 17  V17     275929 non-null  float64
 18  V18     275929 non-null  float64
 19  V19     275929 non-null  float64
 20  V20     275929 non-null  float64
 21  V21     275929 non-null  float64
 22  V22     275929 non-null  float64
 23  V23     275929 non-null  float64
 24  V24     275929 non-null  float64
 25  V25     275929 non-null  float64
 26  V26     275929 non-null  float64
 27  V27     275929 non-null  float64
 28  V28     275929 non-null  float64
 29  Amount  275929 non-null  float64
 30  Class   275929 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 65.3 MB
None
```

Figure A3: Checking Dataset Information


jupyter CreditcardResult Last Checkpoint: 12 hours ago
Trusted
File Edit View Run Kernel Settings Help
Code
JupyterLab
Python 3 (ipykernel)

```
[4]: print(df.shape)  # We can obtain the number of rows and columns in df

     (275929, 31)

[5]: print(df.describe())  # Summary statistics for numerical columns

                    Time             V1             V2             V3  \
     count  275929.000000  275929.000000  275929.000000  275929.000000
     mean    92408.870054      -0.003335      -0.010111       0.025045
     std     46282.161300       1.957302       1.652798       1.512944
     min         0.000000     -56.407510     -72.715728     -48.325589
     25%     53215.000000      -0.920499      -0.604861      -0.852319
     50%     82417.000000       0.020023       0.060819       0.209415
     75%    136453.000000       1.303925       0.793562       1.046866
     max    166798.000000       2.454930      22.057729       9.382558

                       V4             V5             V6             V7  \
     count  275929.000000  275929.000000  275929.000000  275929.000000
     mean        0.005081      -0.010021       0.004277      -0.004158
     std         1.416351       1.381463       1.331383       1.237972
     min        -5.683171    -113.743307     -26.160506     -43.557242
     25%        -0.846925      -0.701896      -0.763838      -0.557588
     50%        -0.011318      -0.065922      -0.268390       0.034550
     75%         0.755716       0.600011       0.403685       0.563780
     max        16.875344      34.801666      73.301626     120.589494

                       V8             V9  ...            V21            V22  \
     count  275929.000000  275929.000000  ...  275929.000000  275929.000000
     mean       -0.000886       0.000184  ...      -0.000087      -0.002569
     std         1.192833       1.101348  ...       0.737134       0.723597
     min       -73.216718     -13.434066  ...     -34.830382     -10.933144
     25%        -0.207130      -0.647008  ...      -0.227427      -0.539674
     50%         0.023307      -0.052700  ...      -0.029930       0.004794
     75%         0.326703       0.600676  ...       0.183943       0.521395
     max        20.007208      15.594995  ...      27.202839      10.503090

                      V23            V24            V25            V26  \
     count  275929.000000  275929.000000  275929.000000  275929.000000
     mean       -0.001445      -0.000011       0.004791      -0.000063
     std         0.627517       0.605959       0.519639       0.483167
     min       -44.807735      -2.836627     -10.295397      -2.604551
     25%        -0.162634      -0.354014      -0.311423      -0.327973
     50%        -0.012740       0.041914       0.025333      -0.054367
     75%         0.145115       0.437487       0.353920       0.242461
     max        22.528412       4.584549       7.519589       3.517346

                      V27            V28         Amount          Class
     count  275929.000000  275929.000000  275929.000000  275929.000000
     mean        0.000185       0.000251      89.057848       0.001754
     std         0.402750       0.332027     251.459928       0.041845
```

Figure A4: Output for Descriptive Stati



Analyse Class Distribution

```
[6]: # Analyze the Class distribution
     class_counts = df['Class'].value_counts()

     # Print class distribution
     print("Class Distribution:")
     print(class_counts)

     # Plot the distribution
     plt.figure(figsize=(8, 6))
     sns.barplot(x=class_counts.index, y=class_counts.values, palette="viridis")
     plt.title("Class Distribution (0: Non-Fraud, 1: Fraud)", fontsize=16)
     plt.xlabel("Class", fontsize=14)
     plt.ylabel("Number of Transactions", fontsize=14)
     plt.xticks([0, 1], ['Non-Fraud (0)', 'Fraud (1)'], fontsize=12)
     plt.show()

     Class Distribution:
     Class
     0    275445
     1       484
     Name: count, dtype: int64
```

Figure A5: Analysing Class Distribution

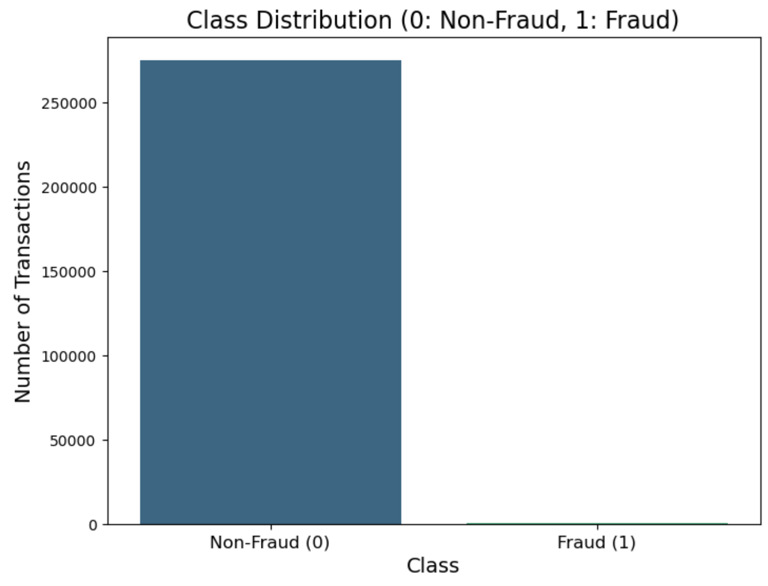
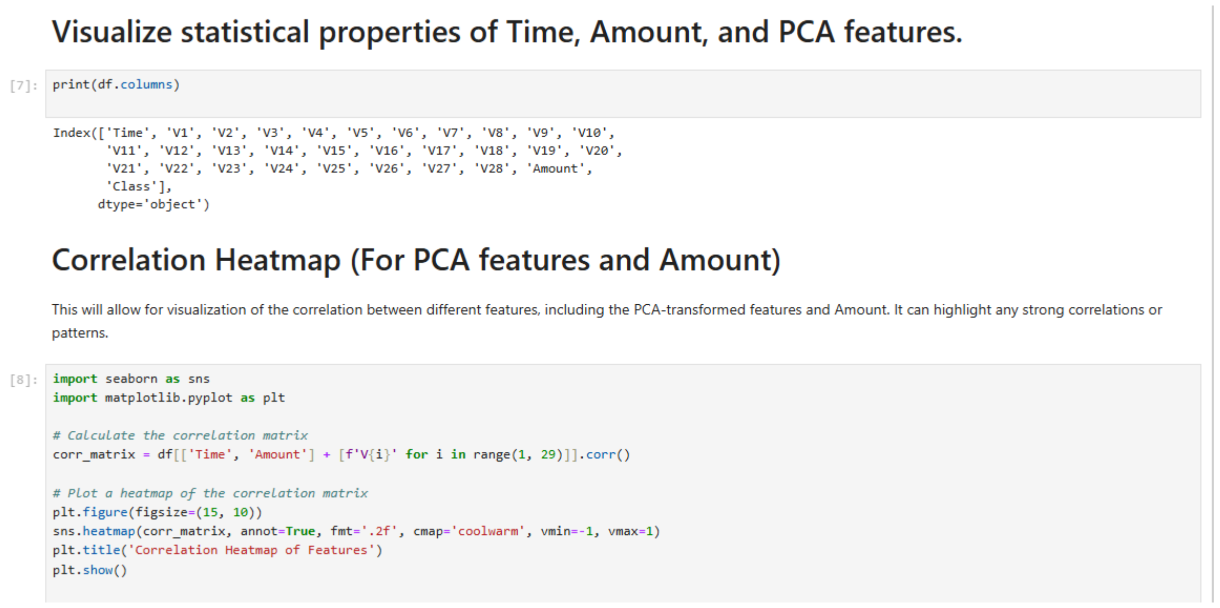


Figure A6: Class Distribution Visualisation



Visualize statistical properties of Time, Amount, and PCA features.

```
[7]: print(df.columns)
```

```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```

Correlation Heatmap (For PCA features and Amount)

This will allow for visualization of the correlation between different features, including the PCA-transformed features and Amount. It can highlight any strong correlations or patterns.

```
[8]: import seaborn as sns
     import matplotlib.pyplot as plt

     # Calculate the correlation matrix
     corr_matrix = df[['Time', 'Amount'] + [f'V{i}' for i in range(1, 29)]].corr()

     # Plot a heatmap of the correlation matrix
     plt.figure(figsize=(15, 10))
     sns.heatmap(corr_matrix, annot=True, fmt='.2f', cmap='coolwarm', vmin=-1, vmax=1)
     plt.title('Correlation Heatmap of Features')
     plt.show()
```

Figure A7: Visualise Properties of PCA and Correlation Heatmap of PCA Features, Amount and Time

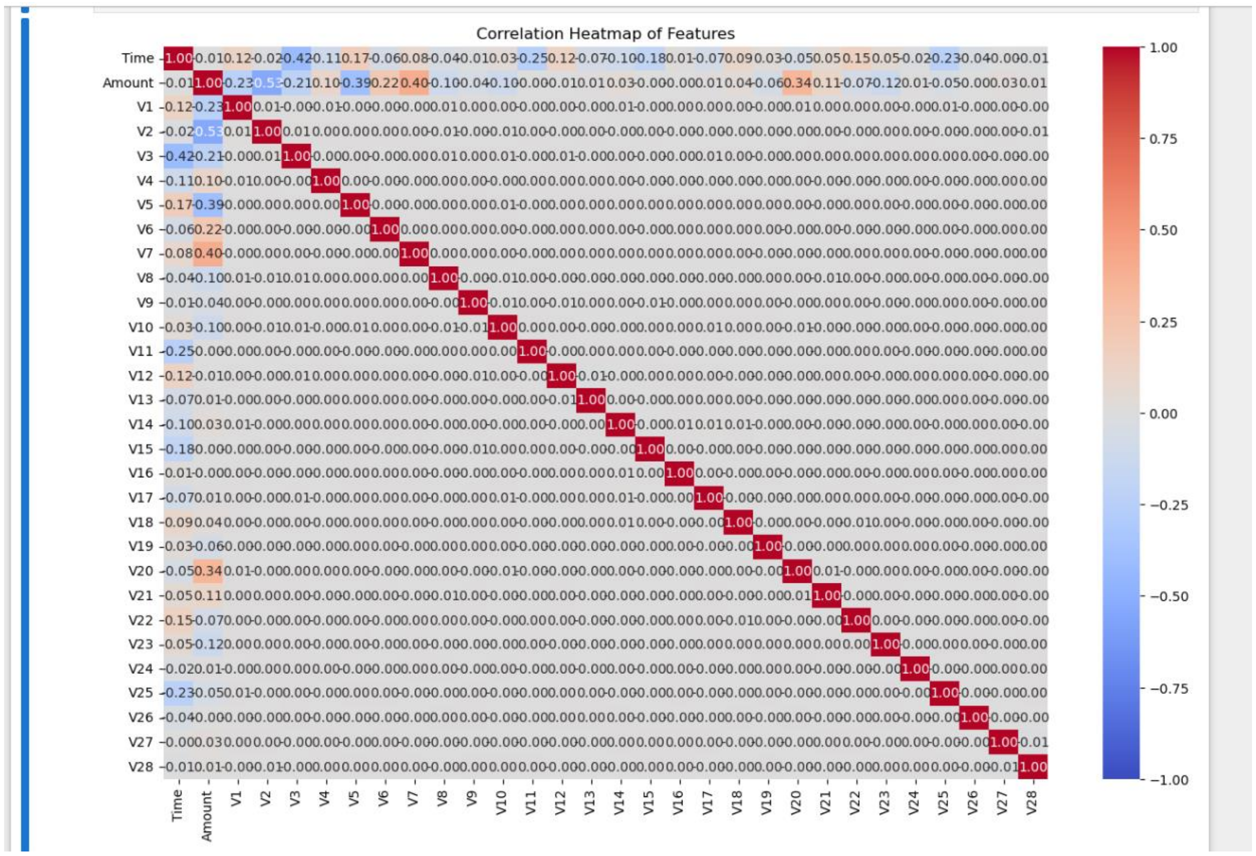
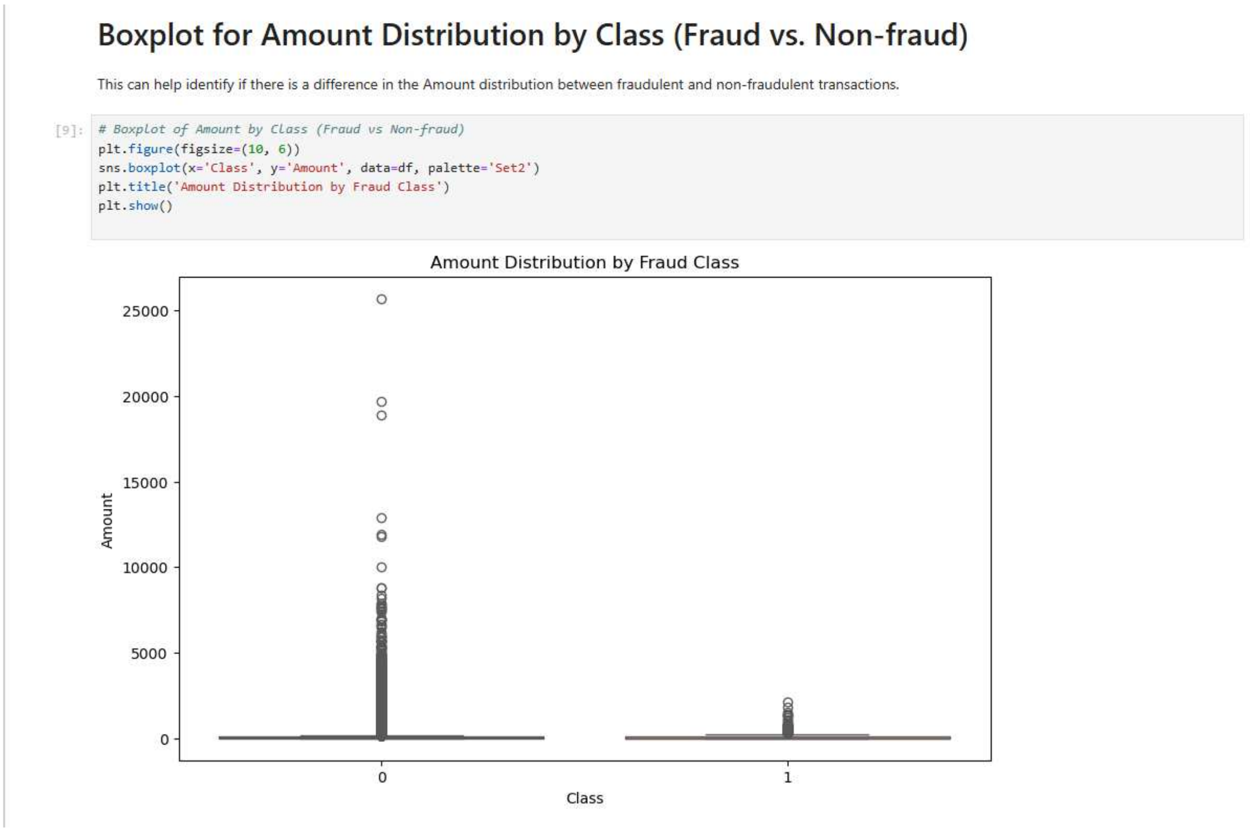


Figure A8: Output of Correlation Heatmap

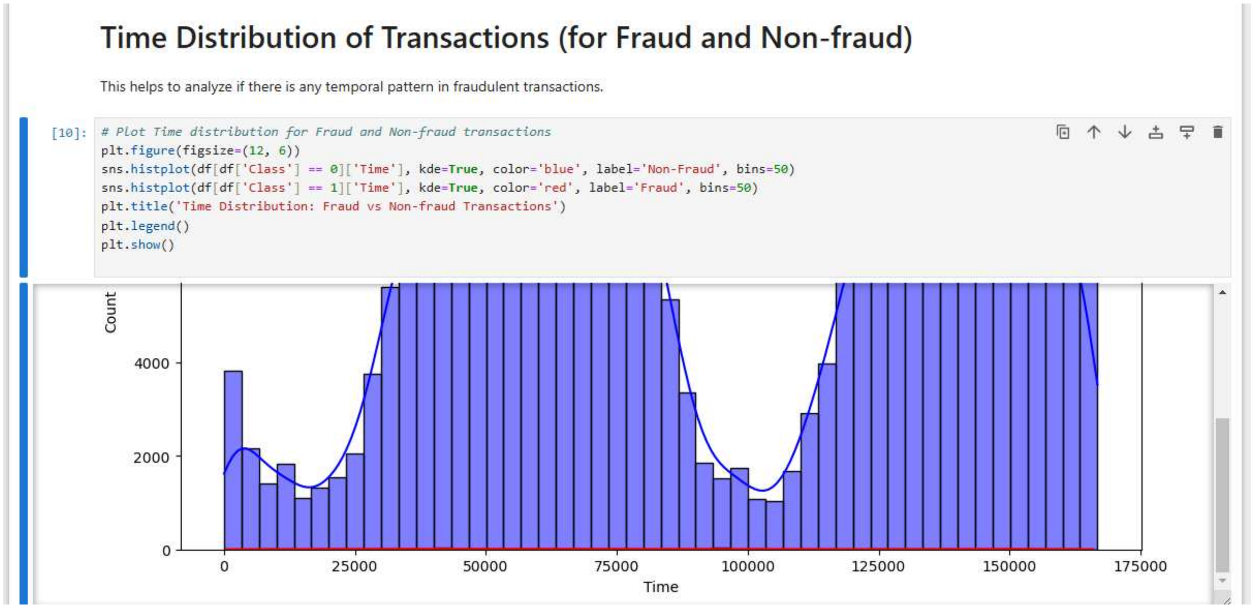


Boxplot for Amount Distribution by Class (Fraud vs. Non-fraud)

This can help identify if there is a difference in the Amount distribution between fraudulent and non-fraudulent transactions.

```
[9]: # Boxplot of Amount by Class (Fraud vs Non-fraud)
plt.figure(figsize=(10, 6))
sns.boxplot(x='Class', y='Amount', data=df, palette='Set2')
plt.title('Amount Distribution by Fraud Class')
plt.show()
```

Figure A9: Boxplot of Amount Distribution by Fraud



Time Distribution of Transactions (for Fraud and Non-fraud)

This helps to analyze if there is any temporal pattern in fraudulent transactions.

```
[10]: # Plot Time distribution for Fraud and Non-fraud transactions
plt.figure(figsize=(12, 6))
sns.histplot(df[df['Class'] == 0]['Time'], kde=True, color='blue', label='Non-Fraud', bins=50)
sns.histplot(df[df['Class'] == 1]['Time'], kde=True, color='red', label='Fraud', bins=50)
plt.title('Time Distribution: Fraud vs Non-fraud Transactions')
plt.legend()
plt.show()
```

Figure A10: Time Distribution of Transaction for Fraud and Non-Fraud

Distribution of Log-transformed Amount

For better visualization, sometimes applying a log transformation to skewed data like Amount helps in understanding its distribution.

```
[11]: # Log-transformed Amount Distribution
      plt.figure(figsize=(10, 6))
      sns.histplot(np.log1p(df['Amount']), kde=True, color='purple', bins=50)
      plt.title('Log-transformed Amount Distribution')
      plt.show()
```

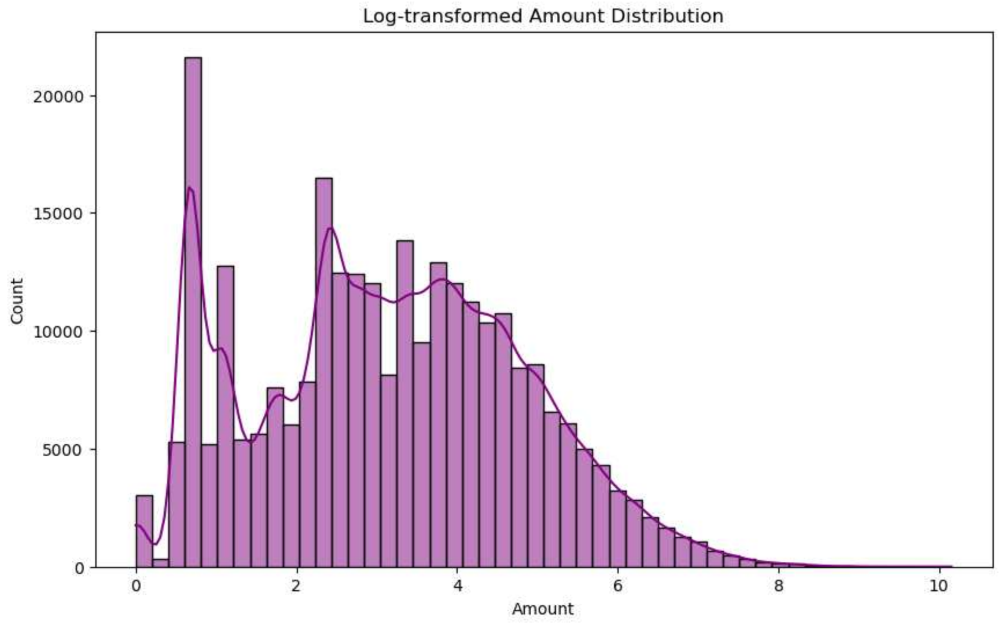


Figure A11: Distribution of Log Transformed Amount

Amount

Amount vs Time Scatter Plot

This scatter plot helps understand the relationship between the Amount and Time variables.

```
[12]: # Scatter plot of Amount vs Time
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Time', y='Amount', data=df, hue='Class', palette='coolwarm', alpha=0.6)
plt.title('Scatter Plot: Amount vs Time (Colored by Fraud Class)')
plt.show()
```

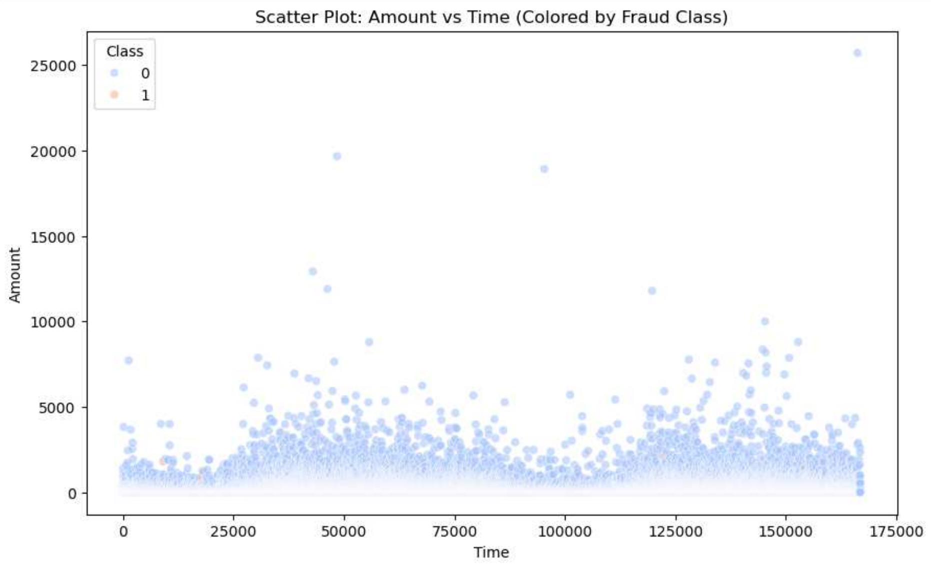


Table 12: Amount vs. Time Scatter Plot

Class Distribution over Time

This plot will help identify if fraud happens more frequently at certain times of day or certain periods.

```
[13]: # Plotting Class distribution over Time
plt.figure(figsize=(12, 6))
sns.countplot(x='Time', hue='Class', data=df, palette='coolwarm', linewidth=0.8)
plt.title('Class Distribution Over Time (Fraud vs Non-fraud)')
plt.show()
```

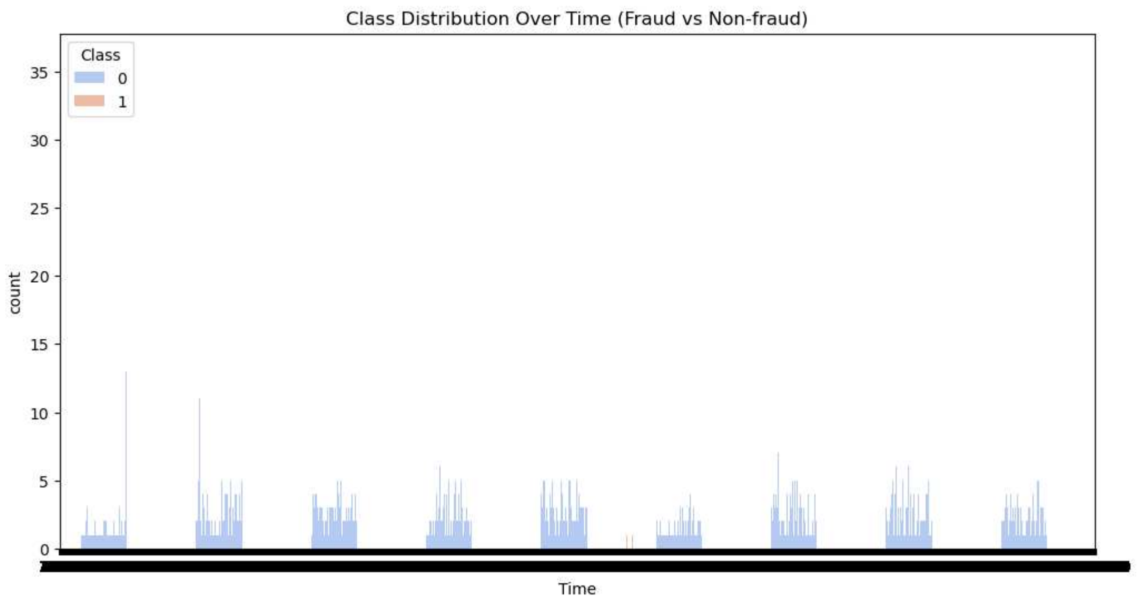
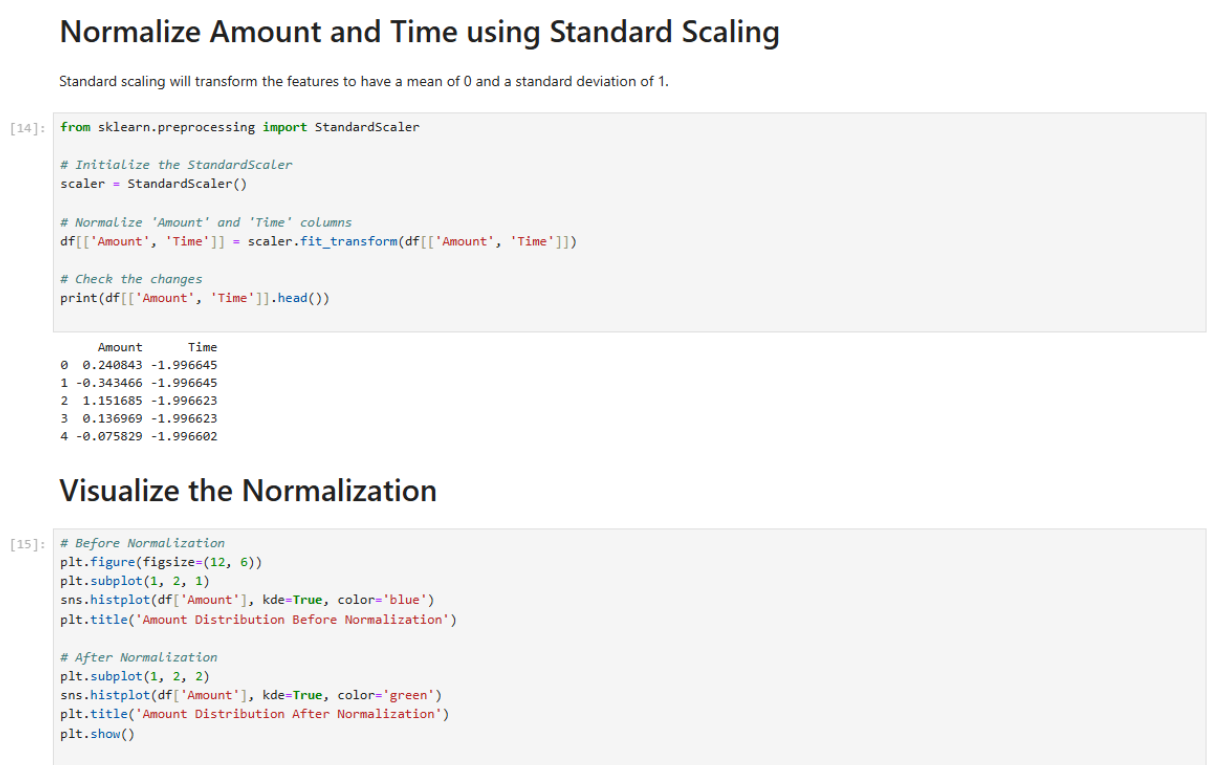


Figure A13: Class Distribution Over Time



Normalize Amount and Time using Standard Scaling

Standard scaling will transform the features to have a mean of 0 and a standard deviation of 1.

[14]:
```
from sklearn.preprocessing import StandardScaler

# Initialize the StandardScaler
scaler = StandardScaler()

# Normalize 'Amount' and 'Time' columns
df[['Amount', 'Time']] = scaler.fit_transform(df[['Amount', 'Time']])

# Check the changes
print(df[['Amount', 'Time']].head())
```

```
     Amount      Time
0  0.240843 -1.996645
1 -0.343466 -1.996645
2  1.151685 -1.996623
3  0.136969 -1.996623
4 -0.075829 -1.996602
```

Visualize the Normalization

[15]:
```
# Before Normalization
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(df['Amount'], kde=True, color='blue')
plt.title('Amount Distribution Before Normalization')

# After Normalization
plt.subplot(1, 2, 2)
sns.histplot(df['Amount'], kde=True, color='green')
plt.title('Amount Distribution After Normalization')
plt.show()
```

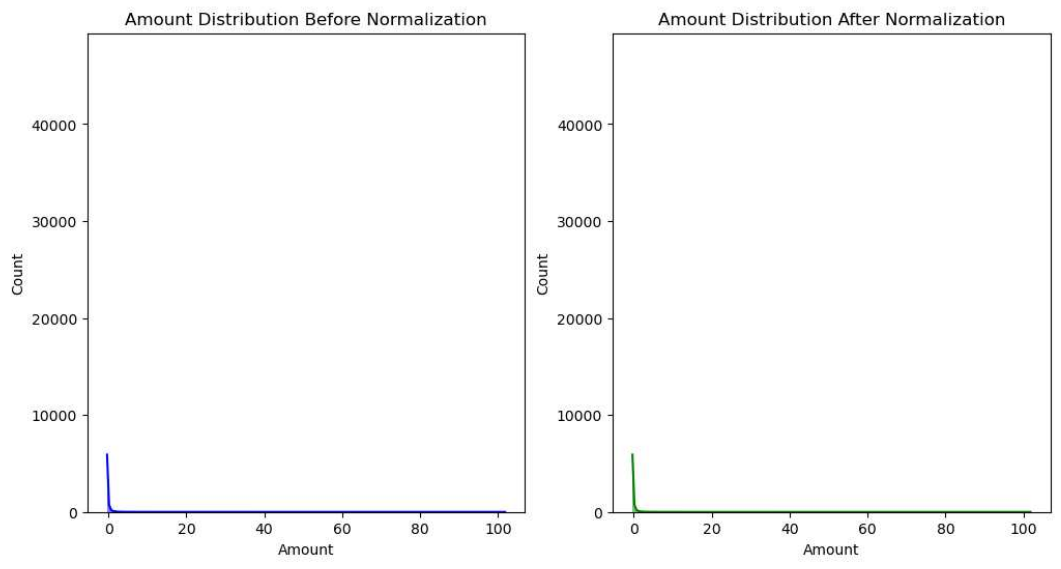
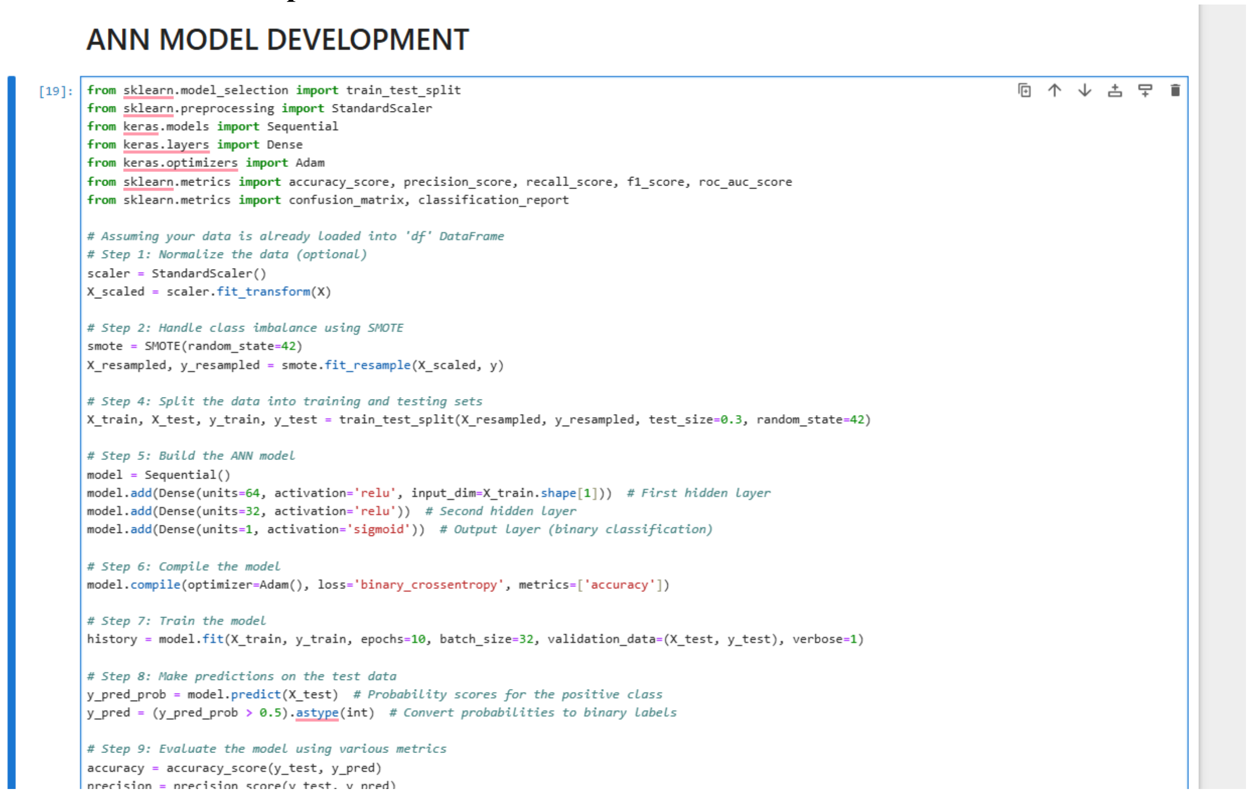


Figure A14: Normalisation of Amount

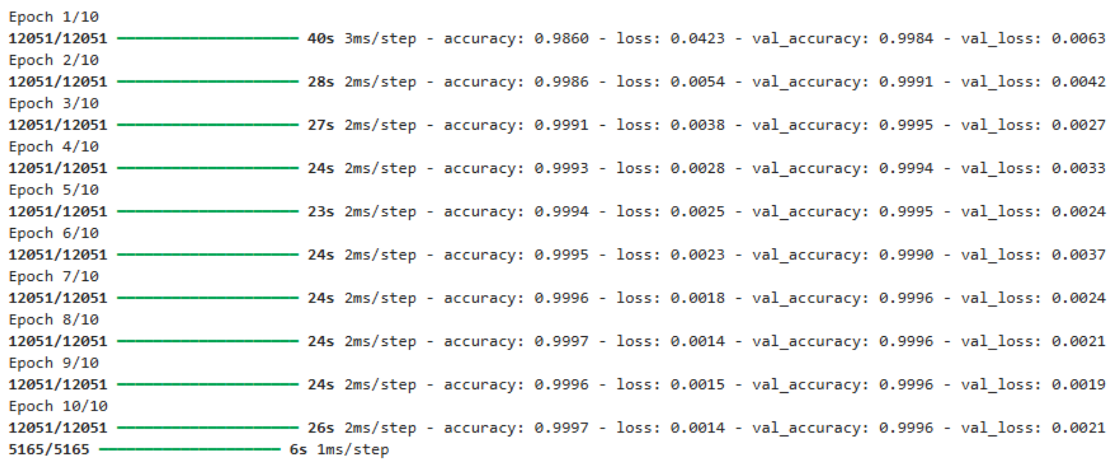
A.2 Model Development

A.2.1 ANN Model Implementation

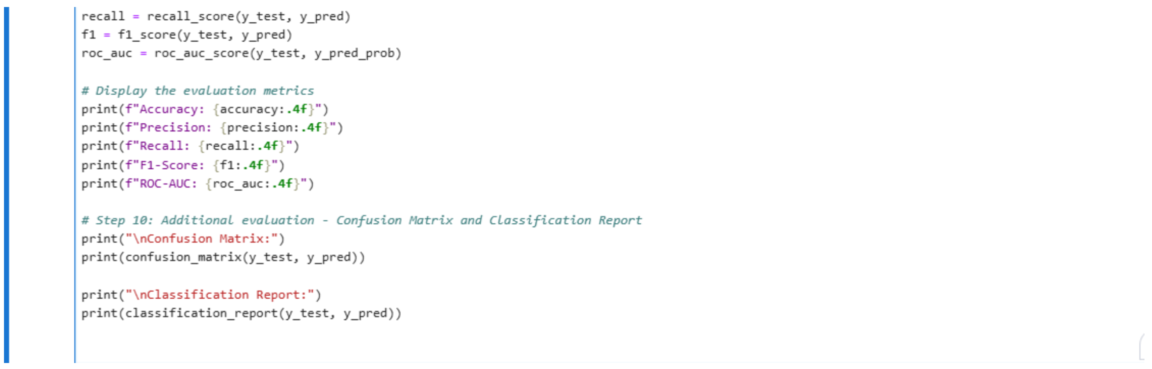


ANN MODEL DEVELOPMENT

```
[19]: from sklearn.model_selection import train_test_split
      from sklearn.preprocessing import StandardScaler
      from keras.models import Sequential
      from keras.layers import Dense
      from keras.optimizers import Adam
      from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
      from sklearn.metrics import confusion_matrix, classification_report

      # Assuming your data is already loaded into 'df' DataFrame
      # Step 1: Normalize the data (optional)
      scaler = StandardScaler()
      X_scaled = scaler.fit_transform(X)

      # Step 2: Handle class imbalance using SMOTE
      smote = SMOTE(random_state=42)
      X_resampled, y_resampled = smote.fit_resample(X_scaled, y)

      # Step 4: Split the data into training and testing sets
      X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.3, random_state=42)

      # Step 5: Build the ANN model
      model = Sequential()
      model.add(Dense(units=64, activation='relu', input_dim=X_train.shape[1]))  # First hidden layer
      model.add(Dense(units=32, activation='relu'))  # Second hidden layer
      model.add(Dense(units=1, activation='sigmoid'))  # Output layer (binary classification)

      # Step 6: Compile the model
      model.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])

      # Step 7: Train the model
      history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test), verbose=1)

      # Step 8: Make predictions on the test data
      y_pred_prob = model.predict(X_test)  # Probability scores for the positive class
      y_pred = (y_pred_prob > 0.5).astype(int)  # Convert probabilities to binary labels

      # Step 9: Evaluate the model using various metrics
      accuracy = accuracy_score(y_test, y_pred)
      precision = precision_score(y_test, y_pred)
```

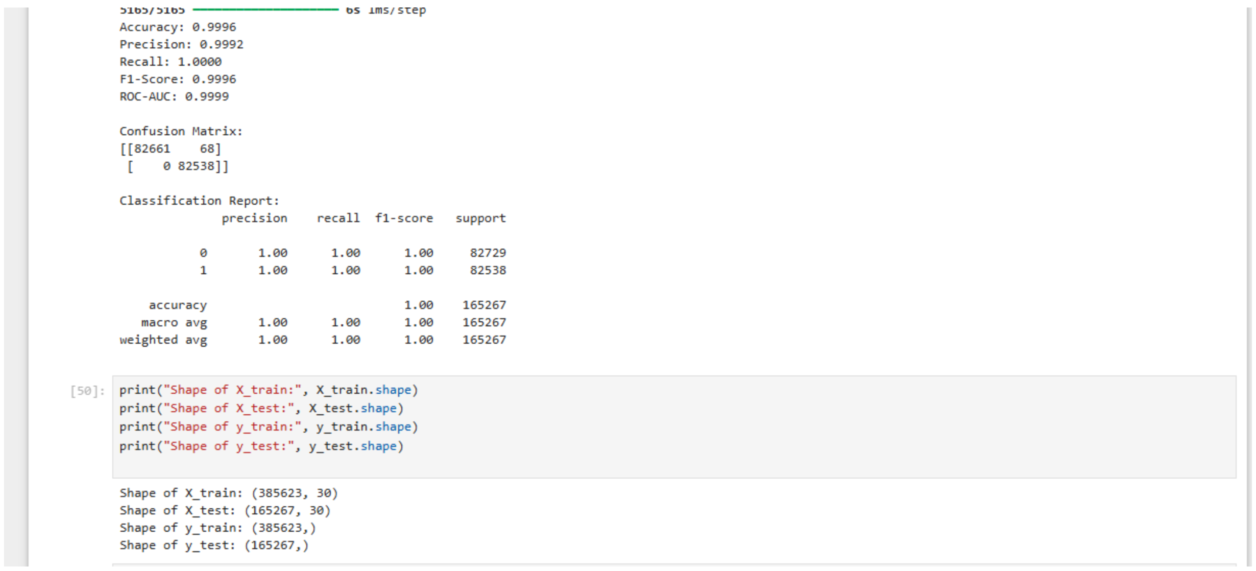


```
Epoch 1/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 40s 3ms/step - accuracy: 0.9860 - loss: 0.0423 - val_accuracy: 0.9984 - val_loss: 0.0063
Epoch 2/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 28s 2ms/step - accuracy: 0.9986 - loss: 0.0054 - val_accuracy: 0.9991 - val_loss: 0.0042
Epoch 3/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 27s 2ms/step - accuracy: 0.9991 - loss: 0.0038 - val_accuracy: 0.9995 - val_loss: 0.0027
Epoch 4/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 24s 2ms/step - accuracy: 0.9993 - loss: 0.0028 - val_accuracy: 0.9994 - val_loss: 0.0033
Epoch 5/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 23s 2ms/step - accuracy: 0.9994 - loss: 0.0025 - val_accuracy: 0.9995 - val_loss: 0.0024
Epoch 6/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 24s 2ms/step - accuracy: 0.9995 - loss: 0.0023 - val_accuracy: 0.9990 - val_loss: 0.0037
Epoch 7/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 24s 2ms/step - accuracy: 0.9996 - loss: 0.0018 - val_accuracy: 0.9996 - val_loss: 0.0024
Epoch 8/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 24s 2ms/step - accuracy: 0.9997 - loss: 0.0014 - val_accuracy: 0.9996 - val_loss: 0.0021
Epoch 9/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 24s 2ms/step - accuracy: 0.9996 - loss: 0.0015 - val_accuracy: 0.9996 - val_loss: 0.0019
Epoch 10/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 26s 2ms/step - accuracy: 0.9997 - loss: 0.0014 - val_accuracy: 0.9996 - val_loss: 0.0021
5165/5165 ━━━━━━━━━━━━━━━━━━━━ 6s 1ms/step
```



```
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_prob)

# Display the evaluation metrics
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")
print(f"ROC-AUC: {roc_auc:.4f}")

# Step 10: Additional evaluation - Confusion Matrix and Classification Report
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

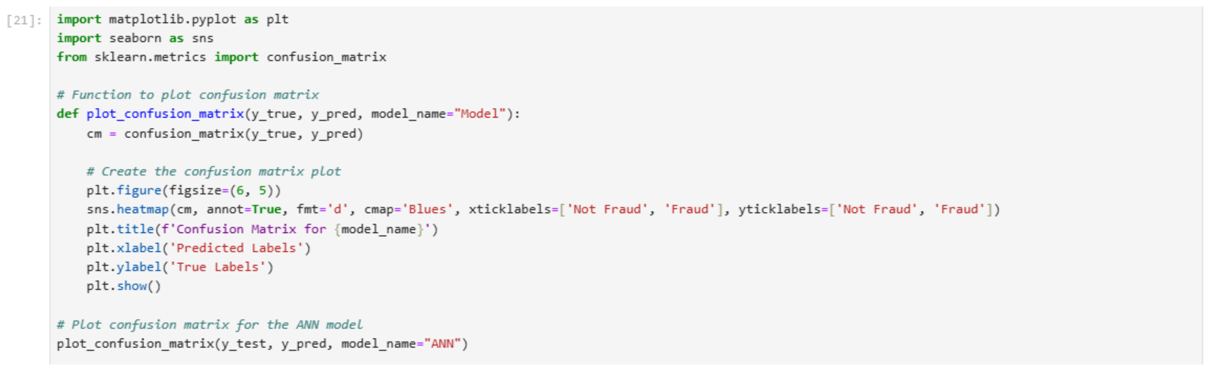
Figure A15: ANN Model Development



```
5165/5165 ━━━━━━━━━━━━━━━━━━━━ 6s 1ms/step
Accuracy: 0.9996
Precision: 0.9992
Recall: 1.0000
F1-Score: 0.9996
ROC-AUC: 0.9999

Confusion Matrix:
[[82661    68]
 [    0 82538]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     82729
           1       1.00      1.00      1.00     82538

    accuracy                           1.00    165267
   macro avg       1.00      1.00      1.00    165267
weighted avg       1.00      1.00      1.00    165267
```

```
[50]: print("Shape of X_train:", X_train.shape)
      print("Shape of X_test:", X_test.shape)
      print("Shape of y_train:", y_train.shape)
      print("Shape of y_test:", y_test.shape)
```

```
Shape of X_train: (385623, 30)
Shape of X_test: (165267, 30)
Shape of y_train: (385623,)
Shape of y_test: (165267,)
```



```
[21]: import matplotlib.pyplot as plt
      import seaborn as sns
      from sklearn.metrics import confusion_matrix

      # Function to plot confusion matrix
      def plot_confusion_matrix(y_true, y_pred, model_name="Model"):
          cm = confusion_matrix(y_true, y_pred)

          # Create the confusion matrix plot
          plt.figure(figsize=(6, 5))
          sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Fraud', 'Fraud'], yticklabels=['Not Fraud', 'Fraud'])
          plt.title(f'Confusion Matrix for {model_name}')
          plt.xlabel('Predicted Labels')
          plt.ylabel('True Labels')
          plt.show()

      # Plot confusion matrix for the ANN model
      plot_confusion_matrix(y_test, y_pred, model_name="ANN")
```

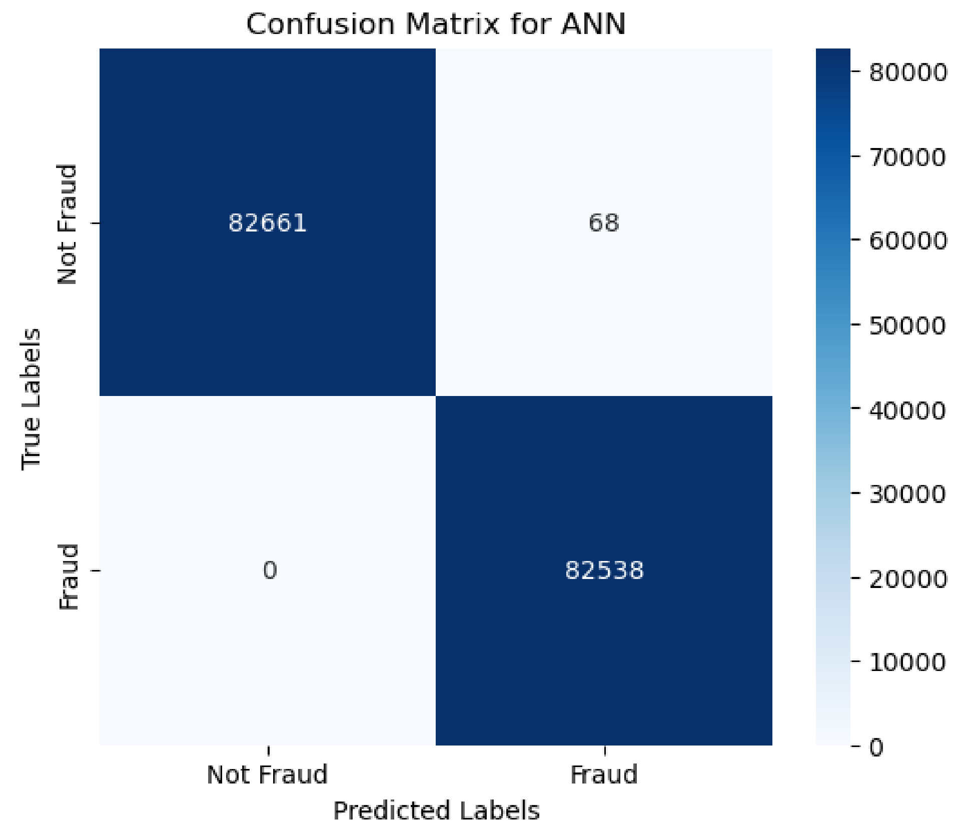
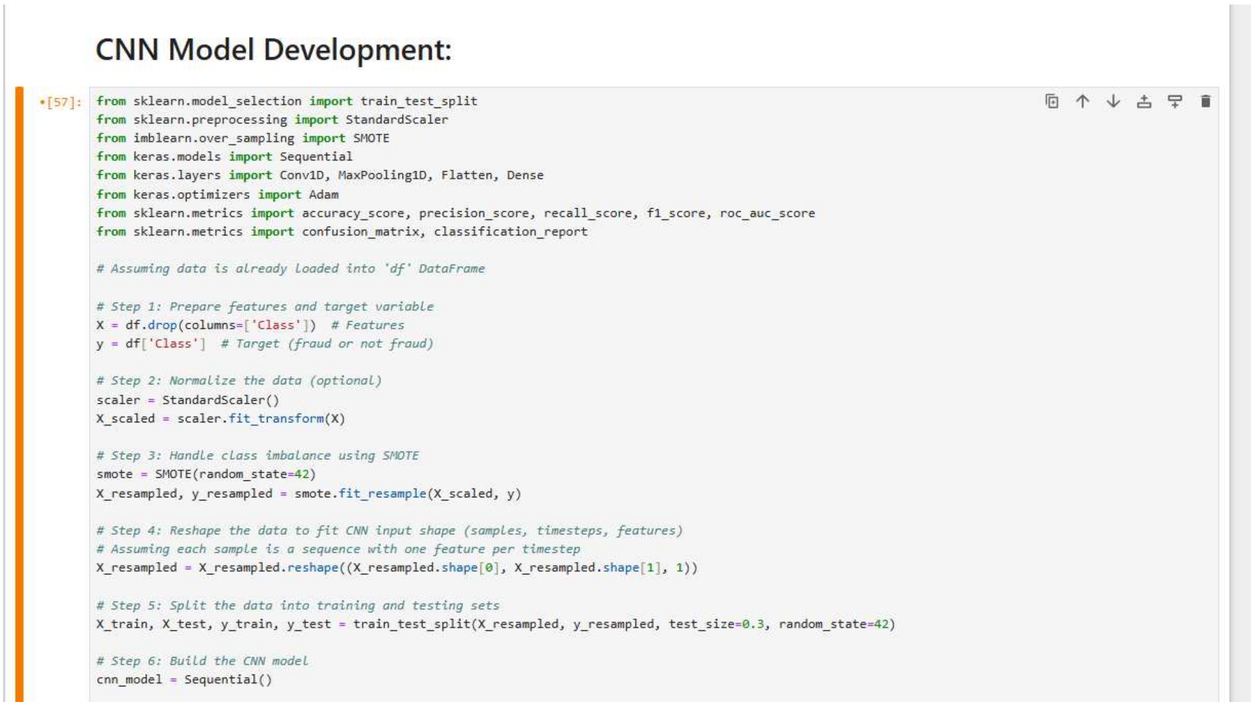


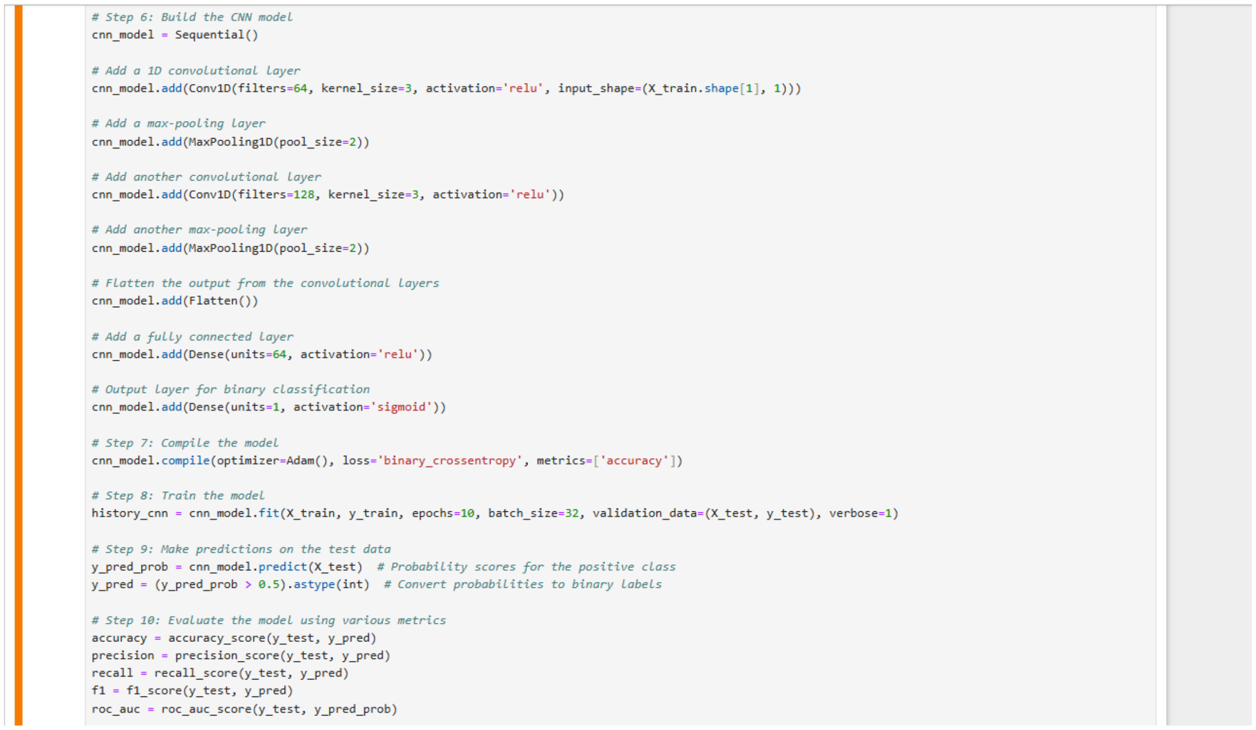
Figure A16: ANN Model Evaluation

A.2.2 CNN Model Implementation

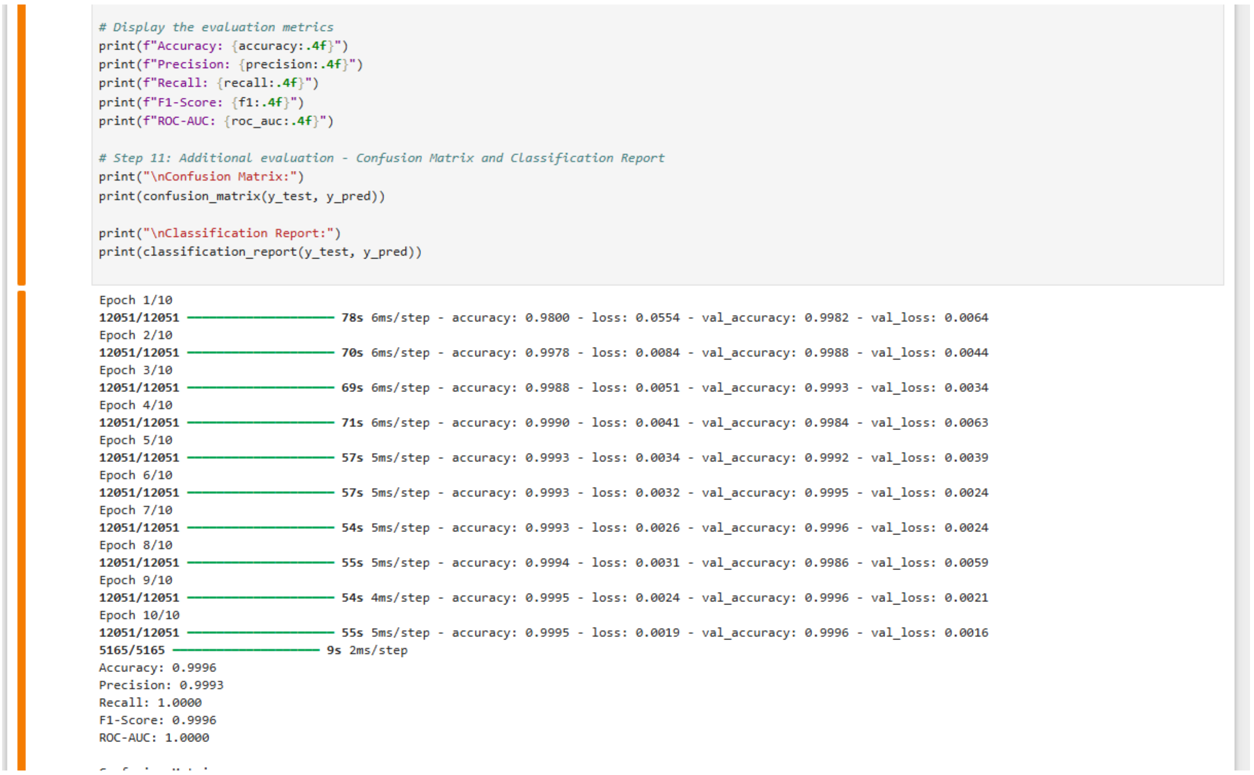


CNN Model Development:

•[57]:
```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE
from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense
from keras.optimizers import Adam
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
from sklearn.metrics import confusion_matrix, classification_report

# Assuming data is already loaded into 'df' DataFrame

# Step 1: Prepare features and target variable
X = df.drop(columns=['Class'])  # Features
y = df['Class']  # Target (fraud or not fraud)

# Step 2: Normalize the data (optional)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 3: Handle class imbalance using SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_scaled, y)

# Step 4: Reshape the data to fit CNN input shape (samples, timesteps, features)
# Assuming each sample is a sequence with one feature per timestep
X_resampled = X_resampled.reshape((X_resampled.shape[0], X_resampled.shape[1], 1))

# Step 5: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.3, random_state=42)

# Step 6: Build the CNN model
cnn_model = Sequential()
```

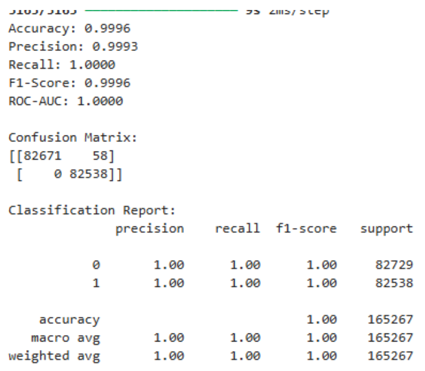


```python
# Step 6: Build the CNN model
cnn_model = Sequential()

# Add a 1D convolutional layer
cnn_model.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)))

# Add a max-pooling layer
cnn_model.add(MaxPooling1D(pool_size=2))

# Add another convolutional layer
cnn_model.add(Conv1D(filters=128, kernel_size=3, activation='relu'))

# Add another max-pooling layer
cnn_model.add(MaxPooling1D(pool_size=2))

# Flatten the output from the convolutional layers
cnn_model.add(Flatten())

# Add a fully connected layer
cnn_model.add(Dense(units=64, activation='relu'))

# Output layer for binary classification
cnn_model.add(Dense(units=1, activation='sigmoid'))

# Step 7: Compile the model
cnn_model.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])

# Step 8: Train the model
history_cnn = cnn_model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test), verbose=1)

# Step 9: Make predictions on the test data
y_pred_prob = cnn_model.predict(X_test)  # Probability scores for the positive class
y_pred = (y_pred_prob > 0.5).astype(int)  # Convert probabilities to binary labels

# Step 10: Evaluate the model using various metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_prob)
```



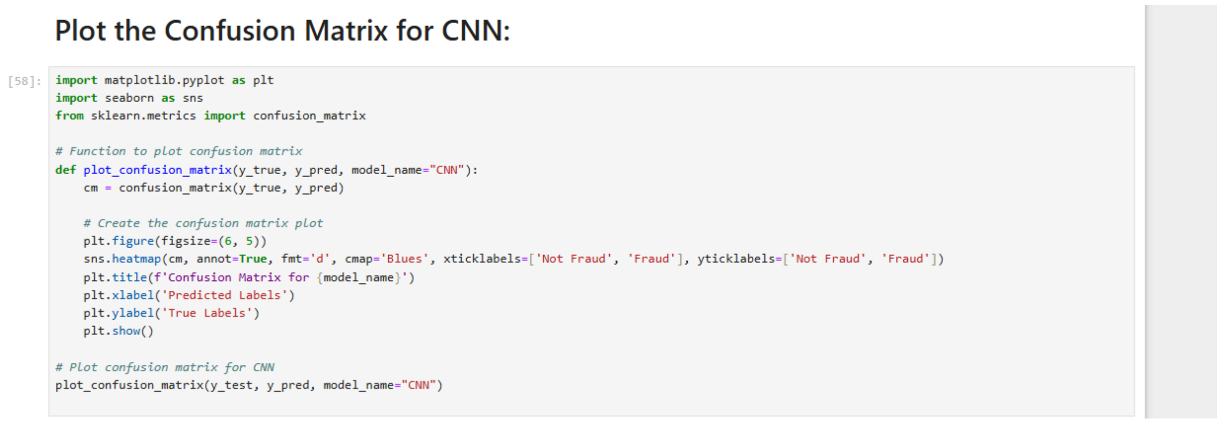
```python
# Display the evaluation metrics
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")
print(f"ROC-AUC: {roc_auc:.4f}")

# Step 11: Additional evaluation - Confusion Matrix and Classification Report
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Epoch 1/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 78s 6ms/step - accuracy: 0.9800 - loss: 0.0554 - val_accuracy: 0.9982 - val_loss: 0.0064
Epoch 2/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 70s 6ms/step - accuracy: 0.9978 - loss: 0.0084 - val_accuracy: 0.9988 - val_loss: 0.0044
Epoch 3/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 69s 6ms/step - accuracy: 0.9988 - loss: 0.0051 - val_accuracy: 0.9993 - val_loss: 0.0034
Epoch 4/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 71s 6ms/step - accuracy: 0.9990 - loss: 0.0041 - val_accuracy: 0.9984 - val_loss: 0.0063
Epoch 5/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 57s 5ms/step - accuracy: 0.9993 - loss: 0.0034 - val_accuracy: 0.9992 - val_loss: 0.0039
Epoch 6/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 57s 5ms/step - accuracy: 0.9993 - loss: 0.0032 - val_accuracy: 0.9995 - val_loss: 0.0024
Epoch 7/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 54s 5ms/step - accuracy: 0.9993 - loss: 0.0026 - val_accuracy: 0.9996 - val_loss: 0.0024
Epoch 8/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 55s 5ms/step - accuracy: 0.9994 - loss: 0.0031 - val_accuracy: 0.9986 - val_loss: 0.0059
Epoch 9/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 54s 4ms/step - accuracy: 0.9995 - loss: 0.0024 - val_accuracy: 0.9996 - val_loss: 0.0021
Epoch 10/10
12051/12051 ━━━━━━━━━━━━━━━━━━━━ 55s 5ms/step - accuracy: 0.9995 - loss: 0.0019 - val_accuracy: 0.9996 - val_loss: 0.0016
5165/5165 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step
Accuracy: 0.9996
Precision: 0.9993
Recall: 1.0000
F1-Score: 0.9996
ROC-AUC: 1.0000
```



```
Accuracy: 0.9996
Precision: 0.9993
Recall: 1.0000
F1-Score: 0.9996
ROC-AUC: 1.0000
```

```
Confusion Matrix:
[[82671    58]
 [    0 82538]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     82729
           1       1.00      1.00      1.00     82538

    accuracy                           1.00    165267
   macro avg       1.00      1.00      1.00    165267
weighted avg       1.00      1.00      1.00    165267
```



Plot the Confusion Matrix for CNN:

```python
[58]: import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Function to plot confusion matrix
def plot_confusion_matrix(y_true, y_pred, model_name="CNN"):
    cm = confusion_matrix(y_true, y_pred)

    # Create the confusion matrix plot
    plt.figure(figsize=(6, 5))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Fraud', 'Fraud'], yticklabels=['Not Fraud', 'Fraud'])
    plt.title(f'Confusion Matrix for {model_name}')
    plt.xlabel('Predicted Labels')
    plt.ylabel('True Labels')
    plt.show()

# Plot confusion matrix for CNN
plot_confusion_matrix(y_test, y_pred, model_name="CNN")
```

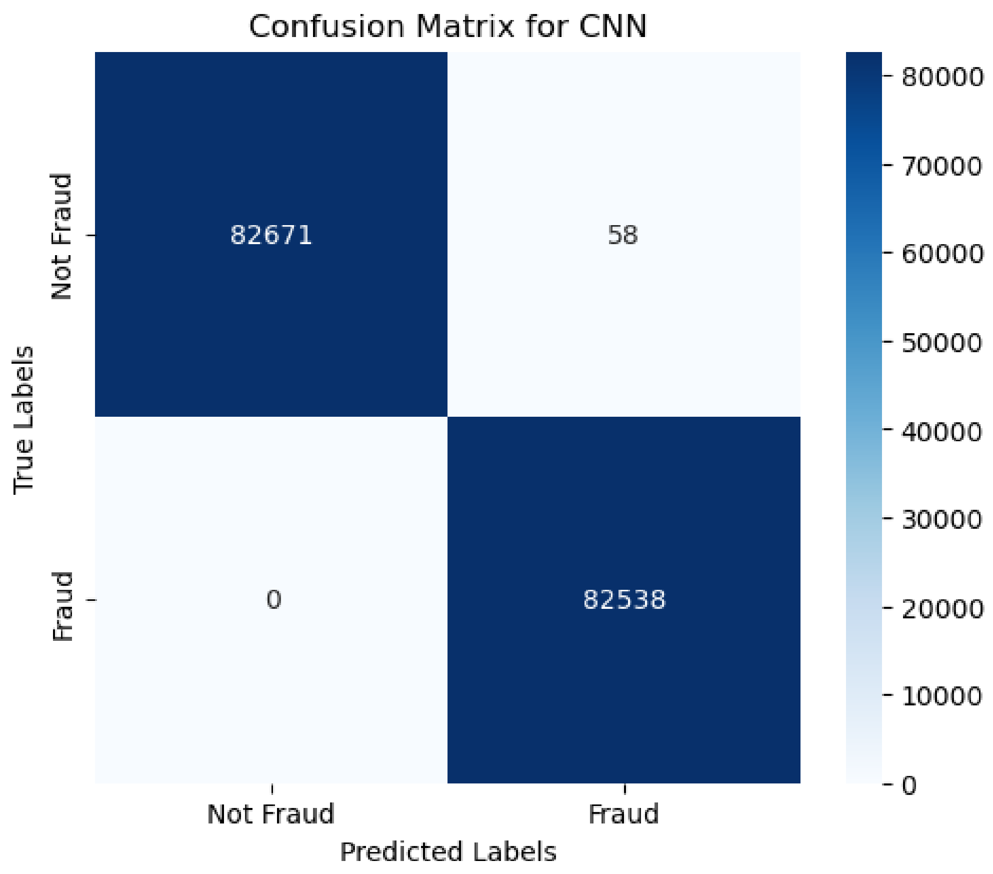
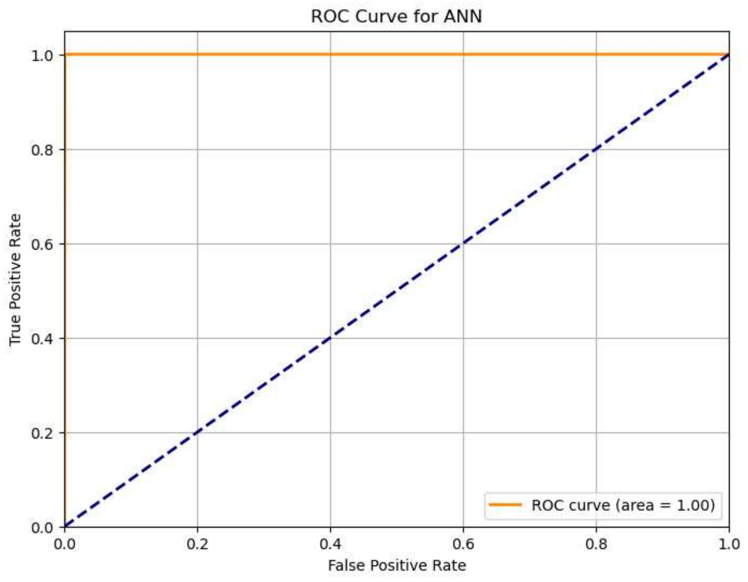


Figure A17: CNN Model Development

A.3 Evaluation and Metrics

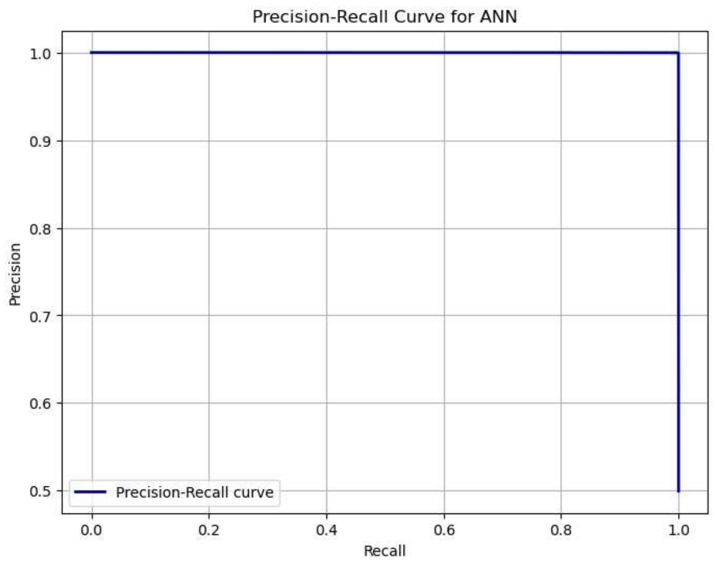
ROC CURVE ANN

[54]:
```python
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# ROC Curve function
def plot_roc_curve(y_true, y_pred_prob, model_name="ANN"):
    # Compute False Positive Rate (FPR), True Positive Rate (TPR), and thresholds
    fpr, tpr, thresholds = roc_curve(y_true, y_pred_prob)
    roc_auc = auc(fpr, tpr)

    # Plot the ROC curve
    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')  # Diagonal line
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'ROC Curve for {model_name}')
    plt.legend(loc='lower right')
    plt.grid()
    plt.show()

# Call the function to plot the ROC curve
plot_roc_curve(y_test, y_pred_prob, model_name="ANN")
```



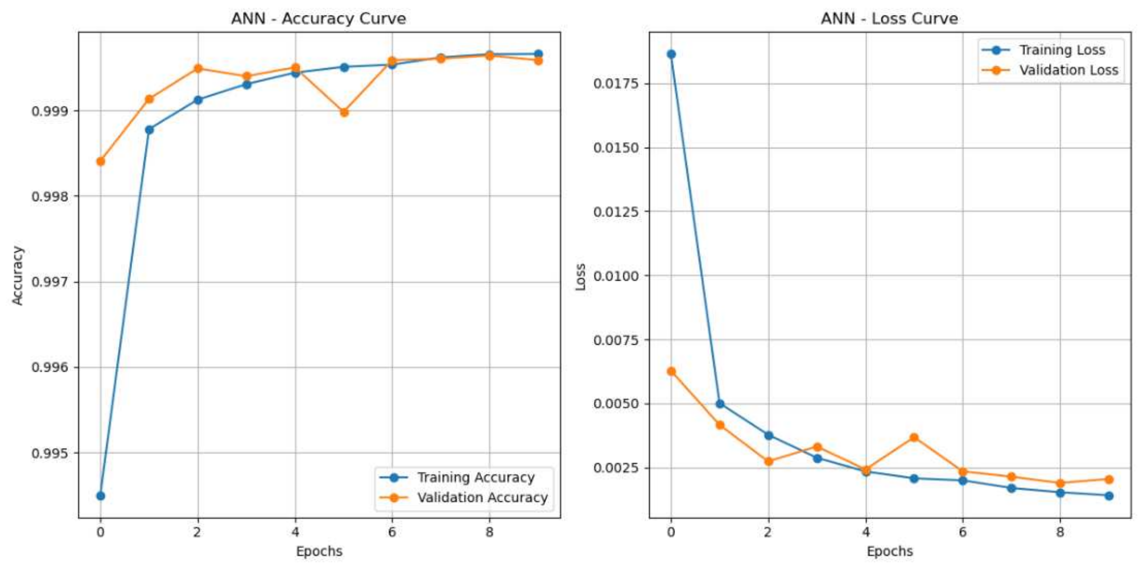
PRECISION-RECALL CURVE FOR ANN

```
[55]: from sklearn.metrics import precision_recall_curve
      import matplotlib.pyplot as plt

      # Precision-Recall Curve function
      def plot_precision_recall_curve(y_true, y_pred_prob, model_name="ANN"):
          # Compute precision, recall, and thresholds
          precision, recall, thresholds = precision_recall_curve(y_true, y_pred_prob)

          # Plot the Precision-Recall curve
          plt.figure(figsize=(8, 6))
          plt.plot(recall, precision, color='darkblue', lw=2, label=f'Precision-Recall curve')
          plt.xlabel('Recall')
          plt.ylabel('Precision')
          plt.title(f'Precision-Recall Curve for {model_name}')
          plt.legend(loc='lower left')
          plt.grid()
          plt.show()

      # Call the function to plot the Precision-Recall curve for ANN
      plot_precision_recall_curve(y_test, y_pred_prob, model_name="ANN")
```



ACCURACY AND LOSS CURVE OF ANN

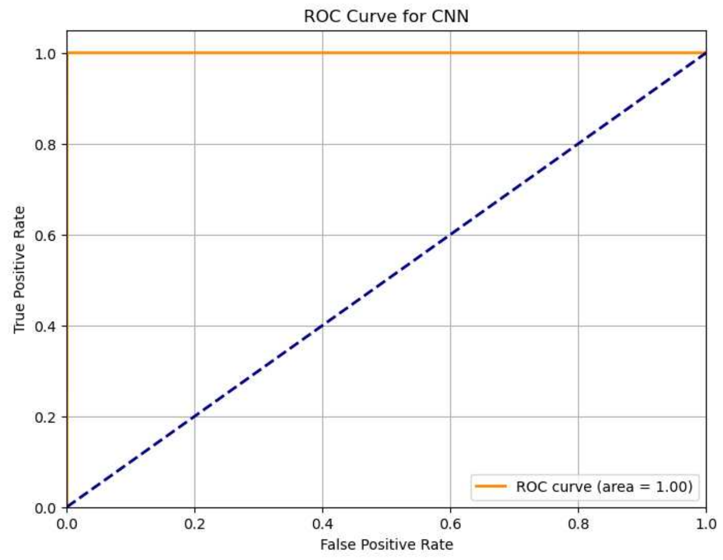
```
[63]: import matplotlib.pyplot as plt

# Function to plot Accuracy and Loss Curve
def plot_accuracy_loss_curve(history, model_name="ANN"):
    # Plot training & validation accuracy values
    plt.figure(figsize=(12, 6))

    plt.subplot(1, 2, 1)  # 1 row, 2 columns, 1st subplot
    plt.plot(history.history['accuracy'], label='Training Accuracy',marker ='o')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy',marker ='o')
    plt.title(f'{model_name} - Accuracy Curve')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend(loc='lower right')
    plt.grid()

    # Plot training & validation loss values
    plt.subplot(1, 2, 2)  # 1 row, 2 columns, 2nd subplot
    plt.plot(history.history['loss'], label='Training Loss',marker ='o')
    plt.plot(history.history['val_loss'], label='Validation Loss',marker ='o')
    plt.title(f'{model_name} - Loss Curve')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend(loc='upper right')
    plt.grid()

    plt.tight_layout()
    plt.show()

# Call the function to plot the Accuracy and Loss curve for ANN
plot_accuracy_loss_curve(history_ann, model_name="ANN")
```



ROC CURVE FOR CNN

[59]:

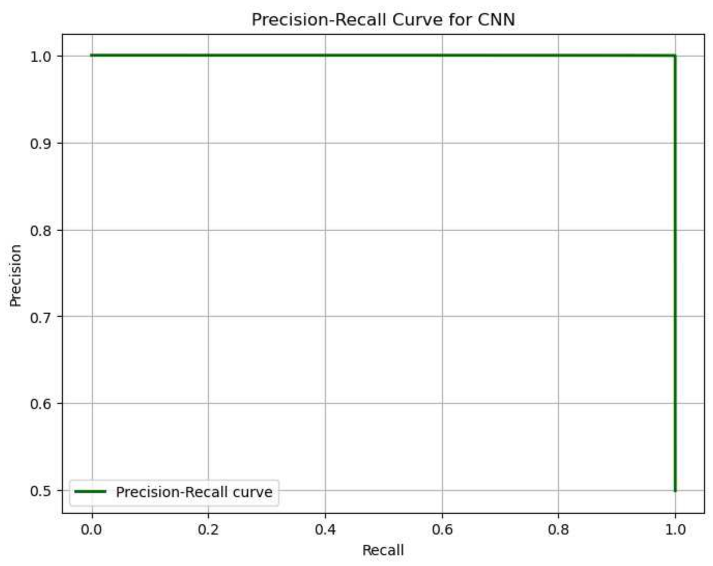
```python
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# ROC Curve function
def plot_roc_curve(y_true, y_pred_prob, model_name="CNN"):
    # Compute False Positive Rate (FPR), True Positive Rate (TPR), and thresholds
    fpr, tpr, thresholds = roc_curve(y_true, y_pred_prob)
    roc_auc = auc(fpr, tpr)

    # Plot the ROC curve
    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')  # Diagonal line
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'ROC Curve for {model_name}')
    plt.legend(loc='lower right')
    plt.grid()
    plt.show()

# Call the function to plot the ROC curve for CNN
plot_roc_curve(y_test, y_pred_prob, model_name="CNN")
```



PRECISION RECALL CURVE FOR CNN

[60]:

```python
from sklearn.metrics import precision_recall_curve
import matplotlib.pyplot as plt

# Precision-Recall Curve function
def plot_precision_recall_curve(y_true, y_pred_prob, model_name="CNN"):
    # Compute precision, recall, and thresholds
    precision, recall, thresholds = precision_recall_curve(y_true, y_pred_prob)

    # Plot the Precision-Recall curve
    plt.figure(figsize=(8, 6))
    plt.plot(recall, precision, color='darkgreen', lw=2, label=f'Precision-Recall curve')
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.title(f'Precision-Recall Curve for {model_name}')
    plt.legend(loc='lower left')
    plt.grid()
    plt.show()

# Call the function to plot the Precision-Recall curve for CNN
plot_precision_recall_curve(y_test, y_pred_prob, model_name="CNN")
```



ACCURACY AND LOSS CURVE FOR CNN

[64]:
```
import matplotlib.pyplot as plt

# Function to plot Accuracy and Loss Curve
def plot_accuracy_loss_curve_cnn(history, model_name="histoty_cnn"):
    # Plot training & validation accuracy values
    plt.figure(figsize=(12, 6))

    plt.subplot(1, 2, 1)  # 1 row, 2 columns, 1st subplot
    plt.plot(history.history['accuracy'], label='Training Accuracy',marker ='o')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy',marker = 'o')
    plt.title(f'{model_name} - Accuracy Curve')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend(loc='lower right')
    plt.grid()

    # Plot training & validation loss values
    plt.subplot(1, 2, 2)  # 1 row, 2 columns, 2nd subplot
    plt.plot(history.history['loss'], label='Training Loss',marker ='o')
    plt.plot(history.history['val_loss'], label='Validation Loss',marker='o')
    plt.title(f'{model_name} - Loss Curve')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend(loc='upper right')
    plt.grid()

    plt.tight_layout()
    plt.show()

# Call the function to plot the Accuracy and Loss curve for CNN
plot_accuracy_loss_curve_cnn(history_cnn, model_name="CNN")
```

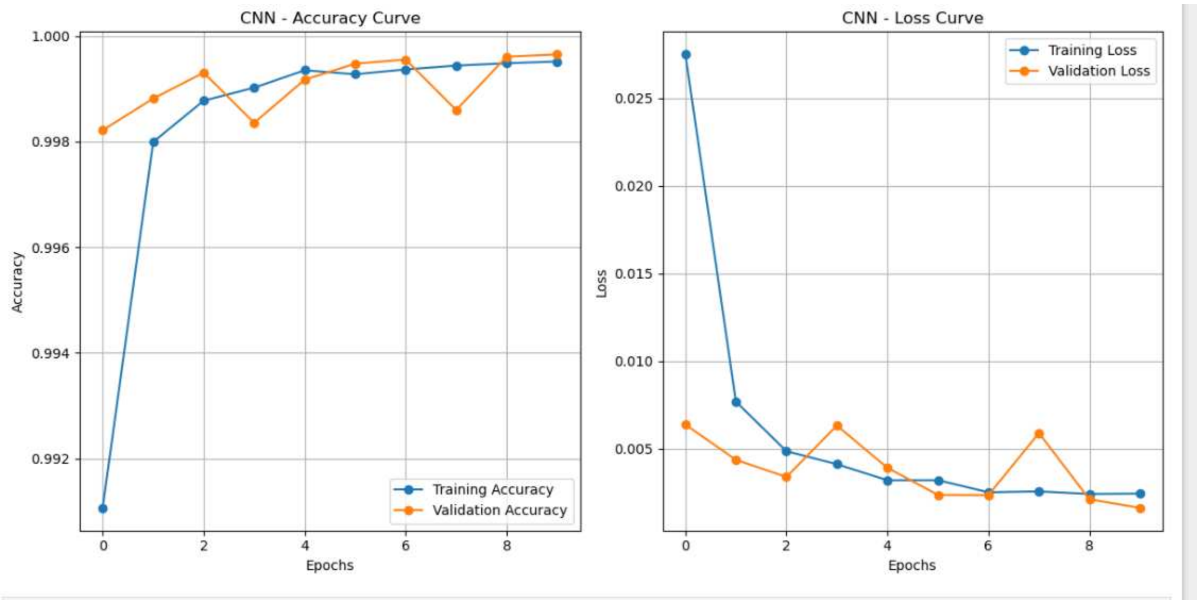
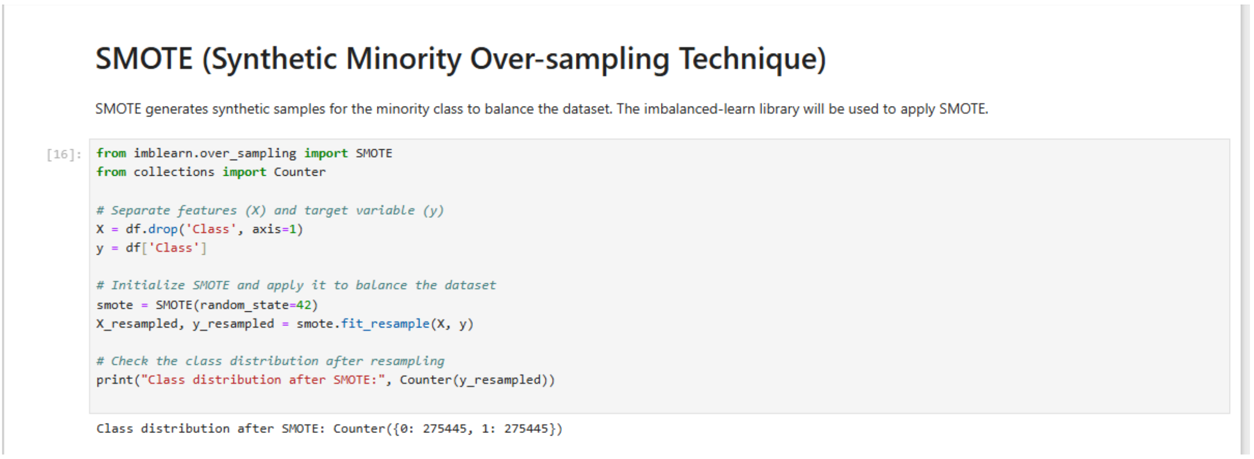


Figure A18: CNN and ANN Model Performance Visualisation

Appendix B: Data Handling

B.1 Data Balancing



SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE generates synthetic samples for the minority class to balance the dataset. The imbalanced-learn library will be used to apply SMOTE.

```
[16]: from imblearn.over_sampling import SMOTE
from collections import Counter

# Separate features (X) and target variable (y)
X = df.drop('Class', axis=1)
y = df['Class']

# Initialize SMOTE and apply it to balance the dataset
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Check the class distribution after resampling
print("Class distribution after SMOTE:", Counter(y_resampled))
```

```
Class distribution after SMOTE: Counter({0: 275445, 1: 275445})
```

Figure B1: Data Balancing