Credit Card Fraud Detection with Deep Multilayer Perceptrons

A Project Report

Submitted to the Faculty of Graduate Studies and Research

In Partial Fulfilment of the Requirements

For the Degree of

Masters of Science

In

Computer Science

University of Regina

By

Sadia Rahman

Regina, Saskatchewan

December 09, 2024

ABSTRACT

It is imperative to address credit card fraud to protect consumer trust and enhance financial security. The primary goal of this paper is to design a reliable system which is capable of detecting credit card fraud or non-fraud with the utmost accuracy. The deep multilayer perceptrons, is used to facilitate the precise prediction of complex patterns and the accurate identification of the classes of a dataset. The proposed model consists of 12 consecutive layers, to acquire intricate patterns from transactional data with a high number of dimensions. The utilization of key approaches like feature scaling, class weighting, and dropout regularization is implemented to improve the performance and reduce the occurrence of overfitting. The performance of the model measures using a dataset of credit card transactions by using different split ratios and a 10-fold cross-validation process, with considerable imbalance between fraudulent and non-fraudulent activities. The goal of this model is to help the improvement in the detection of fraudulent activities, as evidenced by measures such as accuracy, precision, recall, and F1-score.

ACKNOWLEDGEMENT

My first debt of gratitude goes to my supervisor Dr. JingTao Yao for all his support, encouragement, and guidance throughout my graduate studies. I have started my Masters' program as a project-based student. My Supervisor Dr. JingTao Yao introduced me to this exciting research topic which motivated me to work on my topic. Without his continuous guidance and support, I wouldn't be able to complete my project.

I am thankful to the Department of Computer Science for the support in the form of teaching assistantships. I would also like to acknowledge the Faculty of Graduate Studies and Research and the University of Regina for their support during my graduate studies.

I would also like to thank my project report reviewer, Dr. Nashid Shahriar, for reviewing my paper.

Last but not least, I am grateful for the mental and financial support from my parents and my little sister throughout this journey. I would also like to dedicate my master's degree to my beloved husband for his immense support and sacrifices so that I can cherish my dream. Without their help, I would be unable to keep on track and finish the research work.

Contents

Chapter 1

INTRODUCTION

Credit card is a financial instrument that banks or credit institutions offer to enable individuals to borrow money for a range of objectives, such as making purchases, paying bills, or withdrawing cash. The financial institution establishes the credit limit, a line of credit determined by the cardholder's creditworthiness. This creditworthiness is frequently evaluated using indicators such as the cardholder's credit score, income, and payment history. Credit cards differ from debit cards because they allow cardholders to borrow money from the issuer to cover expenses, rather than relying on the user's funds. It is anticipated that the borrowed amount will be repaid in full by a designated due date to avoid interest charges, or over time with interest accruing on the outstanding balance. Credit cards are uniquely designed to provide the user with the flexibility to make purchases within their credit limit and repay the amount as required, allowing them to access a revolving line of credit. In the modern financial landscape, credit cards have evolved into a critical component of the ecosystem, providing much more than a mere financing mechanism [4]. A diverse array of features and incentives are frequently included, such as reward programs, Perks as extended warranties on purchases, purchase protection, and travel insurance

are also included in many credit cards, providing the cardholder with additional value and protection. Moreover, credit cards are broadly accepted worldwide, which allows users to purchase products and services in a variety of currencies without the need to carry a substantial amount of cash.

Credit cards provide significant advantages, particularly in the areas of financial management and security. By incorporating fraud detection mechanisms and zero-liability protection, credit cards offer a safer alternative to currency by protecting users from unauthorized transactions. Additional security measures, such as two-factor authentication and real-time transaction alerts, guarantee that suspicious activities are promptly resolved. Additionally, credit cards are essential for the establishment and preservation of a robust credit history, as well as for providing security. Responsible use and expeditious payments can enhance one's credit score, which is crucial for obtaining loans and mortgages at favorable rates. Conversely, your credit score may be adversely affected by late payments or misuse. Globally, credit cards are indispensable financial instruments for millions due to their adaptability and versatility, which enables them to facilitate both routine expenditures and substantial financial obligations.

Credit card fraud is a criminal offense that involves the illicit use of another person's credit card or credit card details to complete transactions, make purchases, or withdraw money. This may occur through a variety of methods, such as card-not-present fraud, which necessitates only the card details for online or phone transactions, or counterfeit card fraud, in which fraudsters utilize skimming devices to extract card data from magnetic stripes and develop counterfeit cards. Additional methods include the use of personal information obtained from phishing or data breaches to gain access to accounts, the fraudulent use of lost or stolen cards, and the implementation of large-scale data breaches that target banks or retailers. There are substantial fi-

nancial losses for both consumers and financial institutions because of each of these fraud categories. Further complicating the financial consequences of credit card fraud is friendly fraud, which arises when cardholders dispute legitimate charges to obtain refunds.

Fraudulent activities flourish despite technological advancements, frequently employing the same tools that are intended to improve security. Sophisticated techniques, including malware, phishing, and social engineering, are employed by fraudsters to deceive consumers into contributing sensitive information. Using email or text messages that appear to be legitimate, fraudsters can impersonate trusted businesses or government agencies, for instance. These messages could inspire users to click on links leading to fake websites, and they are tricked into disclosing personal information, including credit card numbers, social security numbers, or banking details. Upon obtaining access to this information, fraudsters can conduct unauthorized transactions, deplete bank accounts, or even sell the data on the dark web.

Fraudsters continue to evolve by discovering new vulnerabilities to exploit, despite the presence of advanced defenses such as encryption, two-factor authentication, and biometric verification. Voice cloning and artificial intelligence (AI) technologies are also being exploited to facilitate the more convincing impersonation of individuals in online or phone interactions by fraudsters. Additionally, consumers who are careless in their data protection efforts—such as disclosing personal information on unsecured websites or responding to suspicious emails—unknowingly become susceptible to these fraudulent schemes. In some instances, fraudsters persuade victims to share login credentials or transfer money, resulting in an imminent financial loss. Thus, the human element—trust, distraction, or a lack of awareness—continues to be a critical vulnerability that fraudsters exploit, despite the existence of technological safeguards.

In 2023, consumers reported losing over $10 billion to fraud, the first time losses had reached this benchmark. This represents a 14% increase from 2022. Investment scams led the way with more than $4.6 billion in losses, a 21% rise from the previous year, followed by nearly $2.7 billion lost to impostor scams. Bank transfers and cryptocurrency were the most used methods for fraud. The Federal Trade Commission (FTC) received 2.6 million fraud reports, with impostor scams being the most reported category, followed by online shopping issues, prize scams, and job opportunity scams [14]. For the first time, email became the most common method scammers used to reach consumers, surpassing text messages and phone calls. The FTC, in response to rising fraud trends, has taken comprehensive action, including a nationwide crackdown on illegal telemarketing, proposing a ban on impersonation scams, and confronting emerging frauds such as AI-enabled voice cloning. The FTC's Consumer Sentinel Network, which gathers consumer complaints, received 5.4 million reports in 2023, including over 1 million identity theft complaints [14].

Fraud detection is essential for protecting consumers, financial institutions, and businesses from reputations injury and financial losses. The sophistication of fraud is increasing, and it can lead to severe economic consequences, identity theft, and the erosion of consumer trust, because of the proliferation of digital transactions. Effective fraud detection mitigates these risks by identifying and preventing unauthorized activities before they accumulate substantial financial losses. It is also essential for safeguarding sensitive data from misuse, ensuring compliance with regulations, and sustaining the integrity of financial systems.

Traditional fraud detection methods frequently depend on rule-based systems that identify suspicious transactions by comparing them to predetermined thresholds or conditions. For instance, a transaction might be identified as potentially fraudulent if it exceeds a specific amount or takes place in a location that is not typical. These

systems also implement manual evaluations, which involve the examination of flagged transactions by human analysts to ascertain whether fraud has occurred. Even though rule-based systems have been somewhat effective, they are restricted in their ability to address more sophisticated fraud tactics, such as account takeovers or new fraud patterns, due to their reliance on static rules that may not evolve rapidly enough to address emergent threats.

Machine learning (ML) and AI are employed in new fraud detection methods to address the constraints of conventional systems. By enabling the real-time analysis of vast datasets, these methods are more capable of adapting to changing fraud patterns than rule-based systems. By analyzing transactional behaviors across a variety of factors, such as the timing of transactions, location, purchasing patterns, and even device-specific details like IP addresses, ML algorithms can identify anomalies. The accuracy of these systems is enhanced over time due to their dynamic nature, which allows them to learn from prior fraudulent activities and legitimate transactions. Furthermore, new methodologies incorporate sophisticated technologies such as pattern recognition and behavioral biometrics, which improve security by identifying aberrant user behaviors, such as navigation patterns or typing speed, during online transactions.

Traditional fraud detection methods are surpassed by ML techniques, which employ data-driven models that learn and adapt in real time. In contrast to rule-based systems, which are based on predetermined conditions, ML models can identify intricate fraud patterns and subtle abnormalities that may not be readily visible to human analyzers. For instance, unsupervised learning techniques are particularly advantageous for the identification of novel forms of fraud, as they can identify outliers in transactional data without the need for labeled datasets. On the other hand, supervised learning employs historical fraud data to develop models that can accurately

predict and classify future fraudulent activities. This methodology dramatically improves detection rates and mitigates false positives, which are frequently encountered in conventional methodologies. Furthermore, ML-based systems are capable of processing substantial quantities of data from a variety of sources, such as transaction history, customer behavior, and external data, which facilitates a more comprehensive examination of potential fraud cases. They also facilitate scalability, which enables financial institutions to simultaneously monitor and analyze thousands of transactions, a task that is difficult to accomplish through manual review processes. Artificial neural networks (ANNs) have demonstrated their efficacy in identifying non-linear relationships in transactional data, thereby providing a greater degree of accuracy in the identification of fraud than conventional methods [16].

My research objective is to enhance the accuracy of detection and to confront the obstacles presented by complex and imbalanced transaction data by utilizing deep learning models, with a particular emphasis on ANNs. Implementing an advanced fraud detection system using deep learning models, particularly the ANN technique, has proven to be a powerful approach to identifying intricate patterns and behaviors from vast amounts of transactional data [53]. One of the key architectures used in this domain is the multilayer perceptron (MLP), a type of deep neural network model. MLP excels at learning complex relationships between inputs and outputs through multiple layers of neurons, each applying nonlinear transformations to capture subtle variations within the data [11] [28] [40]. This architecture is particularly effective for fraud detection as it enables the system to learn from highly detailed, high-dimensional datasets, which are characteristic of credit card transactions. The model can differentiate between legitimate and fraudulent behaviors by analyzing a variety of features, such as transaction amounts, location, timing, and user behavior patterns over time.

The deep MLP model is further improved by employing advanced techniques such as feature engineering, which involves the selection and transformation of data attributes to better inform the model, and class weighting, which aims to address the inherent imbalance in fraud detection datasets, where fraudulent transactions are significantly rarer than legitimate ones. Traditional models frequently encounter this imbalance; however, the implementation of class weighting guarantees that fraudulent transactions are prioritized during the training stage, thereby enhancing the fraud detection capabilities of the model. Additionally, implementing sophisticated optimization techniques and training algorithms, including early stopping, prevents overfitting by interrupting training when the model's performance on validation data ceases to improve. Enabling the model to perform well on unseen data, early stopping is essential in deep learning models, as it balances model complexity with generalization.

This research is particularly noteworthy due to its comprehensive examination of the numerous critical deficiencies of conventional fraud detection systems, which frequently encounter the high-dimensional and imbalanced nature of credit card transaction data. The nuanced, evolving character of fraudulent activity may be overlooked by these conventional systems, particularly when dealing with large, complex datasets. In contrast, the deep multilayer perceptron is particularly well-suited to identifying both apparent and subtle fraud patterns, as it is capable of processing complex interactions between a variety of input features. The model's detection accuracy is considerably enhanced by its ability to adapt and improve through advanced algorithms, rendering it a critical tool in the ongoing battle against credit card fraud. To optimize predictive performance and remain adaptable to new and evolving fraud patterns, the MLP-based fraud detection system employs techniques such as class weighting and early stopping to prevent overfitting. This ensures that the model is not biased toward the more frequent legitimate transactions.

The following parts of the report are structured as follows. A comprehensive literature review is presented in Chapter 2, which examines key research studies and previous works related to credit card fraud detection. The focus is on the use of deep learning models and ANN. Chapter 3 explores the foundational knowledge that is essential for comprehending the methodologies employed in this project and contributions of this research are delineated with pseudo-code and the explanation of the benefits are given. Chapter 4 provides a comprehensive explanation of the methods employed to develop the model, such as feature engineering, class weighting, and the architecture of the deep MLP model. This chapter also emphasizes how these methods address the intricacies of fraud detection in high-dimensional data. The experiments and results that were conducted during this endeavor are the focus of Chapter 5. Various preprocessing techniques, including data missing and normalization, are discussed in this section, which begins with the process of data preparation using a credit card transaction dataset. The initial phase of the experiments entails the implementation of numerous train-test segments to assess the model's performance under a variety of circumstances. Subsequently, stratified k-fold cross-validation is implemented to guarantee reliable and robust outcomes, thereby reducing the likelihood of overfitting by distributing class imbalances uniformly across each fold. In summary, Chapter 6 concludes the project by highlighting the research's overall contribution and summarizing the main findings. Furthermore, it suggests experiments that could be conducted on a larger scale of datasets for higher accuracy or utilizing supplementary datasets and techniques.

Chapter 2

LITERATURE REVIEW

This chapter begins by introducing the progress of ANN, highlighting its application in multiple domains. It then discusses deep q-networks (DQN) and deep neural networks (DNN), followed by deep learning (DL) applications in stock analysis and fraud detection. The chapter compares ANN with other techniques, delves into MLP, particularly deep MLP, noting a key limitation, and concludes with similar MLP models that leverage different features or tuning approaches relevant to the study.

The application of ANNs across various domains highlights their potential to outperform traditional forecasting, decision-making, and predictive analysis models. Each study demonstrates how neural networks when properly optimized, can improve accuracy, profitability, and efficiency in financial forecasting, retail strategy, and marketing decisions.

Yao and Tan's research demonstrates ANN applications in forecasting foreign exchange rates, focusing on the American dollar and key currencies including the Japanese yen, British pound, and Australian dollar [67]. The study found the neural network to outperform traditional ARIMA models, particularly for the Australian

Dollar, Swiss Franc, and British Pound, though less effective for the Japanese Yen due to the efficient nature of its market. They recommend using a mix of NMSE, gradient, and profit metrics along with periodic retraining for optimal performance. Additionally, they argue that integrating neural network predictions with actual trading strategies, such as a 'buy-and-hold' approach, is beneficial [67]. Yao et al. further applied ANNs to stock index prediction on the Kuala Lumpur Stock Exchange, using a backpropagation network [68]. The results showed ANNs' predictive accuracy and profitability, especially compared to ARIMA models, while also underscoring the importance of model tuning to handle challenges like recency effects and time frame selection [68].

In retail, Jintanasonti et al. developed an ANN model to support manufacturers in optimizing purchasing strategies by predicting retailer behaviors [27]. Their model, integrated into excel visual basic for applications, achieved 95% accuracy in forecasting order decisions using inputs like holding costs and expected demand, suggesting that strategic discount offers could boost profits [27]. In a marketing context, Yao et al. utilized ANNs to create a decision support system for predicting the sales performance of color TVs in Singapore, reducing input variables from eighteen to five through sensitivity analysis [69]. Key factors like average price, screen size, and seasonal effects emerged, showcasing ANNs' potential to enhance marketing strategies. These studies collectively highlight ANNs' versatility and effectiveness across diverse sectors, suggesting that with rigorous model tuning and integration with expert systems, neural networks can offer powerful insights and decision-making tools [69].

Asha and Kumar, leverage ANNs alongside k-nearest neighbor (k-NN) and support vector machine (SVM) algorithms to enhance credit card fraud detection accuracy [54]. Using a dataset with 31 attributes related to transactions and consumer details, the study demonstrates that ANNs, with hidden layers and relu activation,

outperform other models in identifying fraudulent activity. Through pre-processing techniques such as normalization and under-sampling, the model addresses data imbalance, achieving high accuracy in classifying legitimate and fraudulent transactions. This study highlights the potential of ANNs in evaluating complicated financial data and shows their usefulness in enhancing fraud detection solutions. [54].

Recent research in credit card fraud detection has increasingly focused on utilizing DL to enhance real-time adaptability and accuracy in prediction. Qayoom et al. propose a real-time fraud detection system using DQN that autonomously adapts to changing transaction patterns by learning from recent transaction histories through computed z-scores [50]. Trained on a widely used Kaggle dataset, the DQN achieved a validation accuracy of 97.10%, demonstrating its effectiveness. The system, built with Python, C++, and C#, ensures scalability and easy integration within financial institutions. Additionally, it leverages Apache Kafka for data streaming to support efficient real-time processing, making it highly suitable for diverse financial environments [50].

Similarly, Habibpour et al. focus on enhancing credit card fraud detection with DNNs, addressing prediction accuracy and model reliability through uncertainty quantification techniques [18]. Recognizing that fraud tactics evolve, the study introduces the Monte Carlo dropout, ensemble. It shows the process of assembling Monte Carlo dropout methods to quantify uncertainty and improve decision-making. Their experiments, conducted on a high-dimensional dataset with 385 categorical and numerical features, reveal that the ensemble method most effectively captures uncertainty, thus boosting both prediction accuracy and model confidence. Together, these studies highlight the importance of adaptable, reliable DNN-based systems for fraud detection, suggesting that integrating uncertainty quantification techniques and real-time adaptation can considerably increase the robustness of financial security systems [18].

Deep learning has significantly advanced financial technology, as shown by Du and

Samuseva [8]. Their review highlights the effectiveness of models like deep MLPs, and convolutional neural networks (CNNs). Furthermore, they showed need the for recurrent neural networks and Long short-term memory networks. It restricted boltzmann machines, and auto-encoders in tasks such as stock trading, forecasting, and fraud detection. While more traditional models remain widely used, emerging techniques like deep reinforcement learning, generative adversarial networks, and capsule networks present new opportunities. The study suggests that combining deep learning with tools like natural language processing could further drive innovation in finance [8].

Recent studies have extensively compared various machine learning and deep learning approaches to enhance credit card fraud detection. El-Hlouli et al. investigate the performance of MLP and extreme learning machine classifiers, showing that while MLP achieved a higher accuracy of 97.84%, it offered faster prediction times, making it suitable for real-time applications [11]. Unogwu et al. further explore methods for handling data limitations and class imbalance by using synthetic minority oversampling technique (SMOTE) and principal component analysis (PCA) for feature selection, demonstrating that MLP outperformed other machine learning algorithms, including naive bayes and random forest, on imbalanced, high-dimensional data [65].

In another study, Putrada et al. leverage an autoencoder with a novel feature selection metric called mean decrease in the impurity of absolute and squared error to enhance MLP performance. The study evaluates the performance of four classification methods—decision tree, KNN, logistic regression, and MLP—before and after applying the auto-encoder. This approach significantly improved the area under the curve for MLP, highlighting the effectiveness of auto-encoders in refining feature selection and reducing false positives [49]. John and Murali compared XGBoost, SVM, and MLP for credit approval. The study focuses on building and comparing machine learning models, particularly XGBoost, with SVM and MLP. They show the pro-

cess of enhancing accuracy and efficiency in credit card processing. By utilizing the credit card dataset, the research highlights the strengths of XGBoost, an advanced gradient-boosting technique known for its ability to handle missing data, parallel processing, and regularization to prevent overfitting. The findings reveal that XGBoost outperforms both SVM and MLP across key metrics, achieving the highest accuracy at 90.07%, the best recall, and the lowest false positive rate at 6.06%. While SVM lagged in performance, MLP demonstrated competitive results but still fell short compared to XGBoost [28]. Similarly, El-Naby et al. propose the oversampling with the CNN model, which achieved superior accuracy in comparison to both MLP alone and MLP combined with SMOTE, reaching 98% accuracy, underscoring the potential of combining CNN with oversampling techniques for fraud detection [12].

Negi et al. also compare MLP, XGBoost, and logistic regression on a credit card fraud detection dataset, with MLP showing the best performance across most metrics, particularly recall, which is crucial for effective fraud detection [43]. Lastly, Raju et al. present a DL ensemble model combining long short-term memory and gated recurrent unit networks, with MLP as a meta-learner in a stacking framework. The study applies a synthetic minority oversampling process and edited nearest neighbor (SMOTE-ENN) to balance the dataset, and the results indicate that this ensemble approach outperforms traditional ML models. Additionally, the system is deployed in real-time using a flask framework with SQLite, enabling a practical application for fraud prevention [53].

Advancements in DL techniques have shown promise in tackling the challenges of credit card fraud detection, particularly in addressing class imbalance. Mienye et al. propose a DL-based stacking ensemble framework that combines long short-term memory and gated recurrent unit neural networks as base learners, with a MLP as the meta-learner [38]. To manage data imbalance, the hybrid SMOTE-ENN resam-

pling method was applied. The ensemble model achieved near-perfect performance metrics, including a sensitivity of 1.000, specificity of 0.997, and an area under the curve of 1.000 on the European credit card dataset, underscoring the effectiveness of combining ensemble DL models with data resampling techniques for high-accuracy fraud detection [38].

Conversely, Strelcenia et al. address data imbalance limitations in DL models by introducing K-CGAN, a modified generative adversarial network (GAN) framework that generates high-quality synthetic data for fraud detection [62]. This approach counters the limitations observed in traditional classifiers like random forest, logistic regression, and MLP, which struggle with imbalanced datasets. K-class conditional generative adversarial network (K-CGAN), incorporating divergence in the generator loss function, was compared against conventional oversampling methods like SMOTE, borderline-SMOTE, adaptive synthetic sampling, and other generative adversarial network-based methods. Results indicated that models trained on K-CGAN-generated synthetic data outperformed those trained on original imbalanced datasets, particularly in f1 score metrics, resulting in a highly effective strategy for fraud detection [62].

In recent years, studies have increasingly focused on refining the MLP algorithm and comparing it with other methods to enhance credit card fraud detection. Swathiga et al. highlight the effectiveness of MLP in detecting fraudulent transactions, leveraging Python and Jupyter for data preprocessing and classification [63]. The model demonstrated excellent specificity and sensitivity, surpassing traditional methods. Kasasbeh et al. further optimize MLP for fraud detection, experimenting with various hidden layer configurations and evaluating performance through precision, sensitivity, and f-measure [31]. Their best configuration achieved an F-measure of 84.75%, indicating the importance of precise model tuning [31].

Odeniyi et al. compare MLP with a 1D-convolutional neural network model, demonstrating that both models are effective for real-time fraud detection, particularly on large datasets [45]. They employ information gain as a feature selection method for MLP. Show its impact on high prediction accuracy and underscoring the role of big data in fraud detection [45]. Pillai et al. explore the impact of tuning MLP parameters, including activation functions such as logistic, hyperbolic tangent, and relu [48]. Their results reveal that logistic and hyperbolic tangent functions yield high sensitivity in models with multiple hidden layers, emphasizing that parameter tuning significantly influences fraud detection performance [48].

Similarly, Sadgali et al. compare MLP with CNN and demonstrate that MLP achieves higher accuracy when configured with 100 neurons in hidden layers and a batch size of 20 [56]. The study concludes that MLP is particularly well-suited for fraud detection due to its adaptability to complex transaction data. These studies collectively illustrate that MLP, whether used independently or in combination with other architectures like CNNs, remains a robust choice for detecting fraud in financial transactions. Additionally, the research underscores the importance of tuning and configuration to optimize the accuracy and sensitivity of fraud detection models in modern financial applications [56].

This research addresses critical gaps in credit card fraud detection by introducing a deep MLP model designed to handle imbalanced datasets with enhanced precision and accuracy. Building on existing methods improves the model's capability to detect fraud more reliably. The integration of advanced preprocessing techniques, an optimized number of hidden layers, and refined training strategies further distinguishes this work, contributing to a more robust approach than previous models.

Chapter 3

BACKGROUND KNOWLEDGE

Over the last few years, lightning-fast developments in DL and ML have transformed various industries, including finance and healthcare. Artificial neural networks that can make intelligent predictions based on vast data are at the core of these technological advancements. We begin our examination with an overview of the history of credit card importance and usage, ML, ANN, and DL.

3.1 Invention of Credit Card

The concept of credit has its roots in ancient times; however, the modern credit card system began to develop in the 20th century. In the early 1900s, individual stores began issuing cards or tokens to loyal customers to enable them to purchase products on credit. Oil companies and department stores introduced 'charge cards' for their customers to pay for gas or merchandise later The world's first bank-issued credit card, called 'Charg-It', was introduced by John Biggins, a Brooklyn banker, in 1946. This card permitted consumers to make local purchases. The sales slips were deposited by merchants at the bank; however, this card was only functional in local

areas and required that the consumer maintain an account with the issuing bank.

In 1950, Frank McNamara established Diners Club, the first independent credit card corporation, which was the next significant development. McNamara had the idea after forgetting his wallet during a meal, according to popular stories. The Diners Club card was valid for use at a variety of restaurants in New York city, and cardholders were obligated to settle their entire balance at the end of each month. In 1958, Bank of America introduced the BankAmericard in Fresno, California. This was the first general-purpose credit card that permitted consumers to carry a balance from month to month, thereby introducing the concept of revolving credit. This innovation was instrumental in the widespread adoption of credit cards, and BankAmericard ultimately transformed into Visa The Interbank Card Association was established in 1966 by a group of banks. It is a significant milestone in the expansion of credit card networks beyond individual banks, as it was subsequently re-branded as Master Charge and subsequently became MasterCard [13].

During the 1970s, regulatory reforms were implemented to regulate the credit card industry. Consumers were safeguarded by the Fair Credit Reporting Act (1970) and the Fair Credit Invoice Act (1974), which established protocols for fraud prevention and billing practices Advancements in computer technology were the driving force behind the integration of credit cards into global commerce during the 1980s and 1990s. For improved security, magnetic stripes were implemented, and the chip-and-PIN system was introduced [13]. Credit card dominance in online purchasing was also a consequence of the Internet's proliferation. Contactless payments, mobile wallets, and enhanced security measures such as tokenization have all contributed to the ongoing transformation of the credit card landscape in recent years.

In today's economy, credit cards provide numerous benefits, offering both convenience and valuable financial advantages to consumers. One of the primary advantages of

credit cards is their convenience, as they enable users to make purchases online or in-store without the need to carry cash. Additionally, they provide simple access to funds during emergencies or while traveling. Proper utilization of credit cards can enhance one's credit score, which is essential for securing loans, mortgages, or other financial products. Furthermore, numerous credit cards provide incentives, including cashback, points, or miles, that may be redeemed for travel, purchase, or other services. Another substantial advantage is fraud protection; credit card companies guard against unauthorized transactions, thereby reducing the cardholder's liability for fraudulent activity. In addition, credit cards frequently provide an interest-free credit period, which typically ranges from 30 to 45 days. This feature enables users to postpone payments without incurring interest. In addition, credit cards frequently include extended warranties and purchase protection for the items they purchase, which enhances their security. Finally, credit cards are a dependable payment method for travelers worldwide, as they are extensively accepted on a global scale.

In addition, there are credit cards that are specifically designed for specific demographics and purposes. Student credit cards are particularly intended for college students and frequently feature low credit limits, as well as benefits such as no annual fees and rewards for responsible use. In addition to rewards for business-related spending on items such as office supplies and travel, business credit cards provide features that are specifically designed for small and large enterprises, such as expense tracking, higher credit limits, and employee cards [10]. Users are required to settle the entire balance each month with charge cards. Finally, prepaid credit cards are distinct from traditional credit cards in that they require pre-loading funds before use and are frequently employed by individuals who lack access to traditional banking services.

3.2 Machine Learning

Machine learning is a specialization of AI that concentrates on the creation of systems that can learn and process decisions based on data without the need for explicit programming [15] [58]. ML models predict outcomes by utilizing patterns in data and continuously improving through experience. These models are constructed using algorithms that can adapt to new data, identify patterns, and make informed decisions [34]. The capability of ML to manage complex and extensive datasets is what renders it applicable to a diverse range of industries [46].

Importance of Using ML: The significance of ML lies in its ability to automate intricate tasks and the formulation of data-driven decisions, which in turn results in more efficient processes in a variety of sectors, including finance, healthcare, and marketing. Companies can utilize vast quantities of data to enhance consumer experiences, optimize supply chains, and identify anomalies. ML enables the continuous adjustment of predictions and the processing of real-time data, thereby facilitating quicker, more accurate, and scalable decision-making [46].

ML in the Financial Industry: ML is essential for the improvement of fraud detection, credit assessment, and risk management. In real-time, ML models can identify fraudulent transactions by analyzing historical transaction data and user behavior, thereby reducing financial losses. In a similar vein, ML is employed to automate loan approval processes by evaluating the creditworthiness of applicants using historical and behavioral data. ML also significantly improves personalized financial services, algorithmic trading, and risk management.The general objective of ML models is to minimize a loss function $L(\hat{y}, y)$, where:

- $\hat{y}$ is the predicted output,
- y is the true label or actual value,
- θ represents the model parameters (weights).

The optimization goal is to find the set of parameters θ^* that minimizes the loss function:

$$\theta^* = \arg\min_{\theta} L(\hat{y}, y) \tag{3.1}$$

ML Algorithm:

1. Initialize: Start with random model parameters θ.
2. Training:
   - For each data point (x_i, y_i):
     - Predict the output: $\hat{y}_i = f(x_i; \theta)$,
     - Compute the loss: $L(\hat{y}_i, y_i)$,
     - Update the parameters:

$$\theta \leftarrow \theta - \eta \frac{\partial L}{\partial \theta}$$

       where η is the learning rate.
3. Repeat until convergence or a stopping criterion is met.

3.3 Deep Learning

Deep learning is a subfield of ML that prioritizes the use of ANNs with numerous layers (hence the term deep) to simulate intricate patterns and relationships in data. Images, audio, and text are among the high-dimensional data types that these models, frequently referred to as DNNs, are particularly adaptable at managing [32]. By automatically learning hierarchical features from raw data, DL models can attain state-of-the-art performance in tasks such as image classification, natural language processing, and autonomous driving [19].

DL is crucial for tackling complex challenges that traditional ML methods struggle to address. It excels in automatically extracting meaningful features from unstructured data, significantly reducing the need for extensive manual feature engineering. This capability has propelled advancements in various fields, including natural language processing, speech recognition, and computer vision [59]. DL models have demonstrated the ability to identify objects, translate languages, and even generate new content, showcasing their versatility [15] [39] [60]. Applications of DL span numerous industries, such as image recognition, medical diagnostics, and autonomous vehicles [7] [22]. Moreover, it is integral to speech recognition in virtual assistants like Siri and Alexa, as well as recommendation systems used by e-commerce platforms like Amazon and Netflix, optimizing user experiences and business outcomes [7].

DL in the Financial Industry: DL is implemented in the financial sector to facilitate algorithmic trading, fraud detection, and risk modeling [22]. Through the acquisition of knowledge from an extensive volume of transactional data, DL models can identify subtle patterns that may suggest fraudulent activities. In addition, it is advantageous for risk modeling in financial markets, as it can analyze extensive historical data to forecast market trends [1] [10] [67] [68]. DL algorithms are also used in algorithmic

trading to generate high-frequency trading decisions that are informed by real-time data. The models aim to minimize the loss function similar to general ML, but they operate on multiple layers [7]. The details of the equation of DL are provided below [15] [52].

The loss function is defined as:

$$L(\hat{y}, y) = \frac{1}{N} \sum_{i=1}^{N} L(f(x_i; \theta), y_i) \tag{3.2}$$

where N is the number of data points, $f(x_i; \theta)$ represents the neural network's prediction with parameters θ, and L is the loss function.

DL Algorithm:

1. Initialize: Randomly initialize the weights θ of the network.

2. Training:

   - For each input x_i, compute the output $\hat{y}_i = f(x_i; \theta)$,
   - Calculate the loss $L(\hat{y}_i, y_i)$,
   - Compute the gradient of the loss with respect to the weights: $\nabla_\theta L$,
   - Update weights:

     $$\theta \leftarrow \theta - \eta \nabla_\theta L$$

     where η is the learning rate.

3. Repeat until convergence or a stopping criterion is met.

3.4 Artificial Neural Networks

The neural networks that are formed in the human brain. It serves as the inspiration for computational models known as artificial neural networks [17]. An ANN comprises one or more hidden layers, an input layer, and an output layer, which are interconnected neurons or nodes. Every neuron in a layer receives input, which then applies a weighted sum and an activation function before returning the output to the subsequent layer. Classification, regression, and function approximation are all tasks that are highly effective when ANNs are employed to model complex relationships between inputs and outputs [35].

Importance of Using ANNs: ANNs are crucial due to their ability to acquire intricate mappings between inputs and outputs, which makes them highly adaptable in in a diverse array applications, including pattern recognition and decision-making [33]. In fields such as autonomous control systems, speech recognition, and image processing, they are particularly adept at tasks that involve a non-linear and intricate relationship between the desired output and the input data [37].

ANNs in the Financial Industry: In finance, ANNs are employed in the prediction of stock prices, fraud detection, consumer behavior analysis, and credit scoring. To generate precise predictions regarding future trends or anomalies, they can learn from historical financial data. To assist banks in making more informed lending decisions, ANNs can be employed to analyze historical credit behavior and predict the probability of default [24] [44]. The equation of an ANN is given below:

The output of a neuron in an ANN-

$$\hat{y} = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{3.3}$$

where:

- x_i are the input features,
- w_i are the weights,
- b is the bias,
- σ is the activation function (e.g., sigmoid, relu).

ANNs Algorithm:

1. Initialize: Randomly initialize the weights w_i and biases b.
2. Forward Pass: For each input x_i, compute the output:

$$\hat{y} = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

3. Loss Calculation: Compute the loss $L(\hat{y}, y)$.
4. Backpropagation: Compute the gradients of the loss for the weights and biases.
5. Weight Update: Update weights and biases using gradient descent:

$$w_i \leftarrow w_i - \eta \frac{\partial L}{\partial w_i}$$

6. Repeat until convergence or stopping criteria.

3.5 Multilayer Perceptrons

A multilayer perceptron is a form of ANN. It is composed of a minimum of three layers of nodes: an input layer, one or more hidden layers, and an output layer.

Complex relationships between the input and output data are modeled by each node in the network. Except for the input layer,it use a non-linear activation function. It is widely recognized that MLPs are capable of learning non-linear decision boundaries and are frequently employed in supervised learning tasks, including regression and classification activities [12] [43].

Importance of Using MLP: MLPs are especially crucial when basic models, such as linear regression, are insufficient to capture the complexity of the data. In fields such as pattern recognition, image classification, and time-series prediction, they have been effectively implemented in the modeling of non-linear relationships [12]. The flexibility of MLPs enables them to address issues with multiple outputs and varying data dimensionality.

MLP in the Financial Industry: MLPs are employed in the financial sector for algorithmic trading, credit risk assessment, and fraud detection [53]. MLPs can identify suspicious activity that may suggest deception by learning from an extensive amount of transaction data. The complex relationships between various financial indicators are also utilized to forecast market trends and price financial instruments. The general equation of an MLP is stated below [64]:

$$a^{(l+1)} = \sigma(W^{(l)}a^{(l)} + b^{(l)}) \tag{3.4}$$

where:

- $a^{(l)}$ is the activation at layer l,
- $W^{(l)}$ is the weight matrix for layer l,
- $b^{(l)}$ is the bias vector for layer l,

- σ is the activation function.

MLPs Algorithm:

1. Initialize: Randomly initialize the weights and biases.

2. Forward Pass: Compute the activations for each layer:

$$a^{(l+1)} = \sigma(W^{(l)}a^{(l)} + b^{(l)})$$

3. Compute Loss: Calculate the loss $L(\hat{y}, y)$.

4. Backpropagation: Compute the gradients of the loss with respect to weights and biases.

5. Update Parameters: Use gradient descent to update weights and biases.

6. Repeat until the loss converges or a stopping condition is reached.

Chapter 4

FRAMEWORK OF DEEP MULTILAYER PERCEPTRONS FOR CREDIT CARD FRAUD DETECTION

This chapter presents the proposed solution to the problem outlined in Chapter 1. This section will discuss the concept of deep MLPs, activation functions, class weights, and specific network layers like dense, and dropout which are foundational elements of the model. The proposed model's step-by-step algorithm details are given to detect credit card fraud. This contains a full explanation of the model training, data analysis, and prediction stages, with a focus on the key computational strategies and approaches that ensure the model's efficacy. The presentation will also go into the optimization and assessment procedures used to improve and analyze the model's performance and purpose.

4.1 Foundational Components

4.1.1 Deep Multilayer Perceptron

A deep multilayer perceptron is a type of DNN. It has a fully connected structure, which means each neuron in one layer is fully connected to every neuron in the succeeding hidden layers. It follows a forward manner. It effectively models non-linear relationships, detecting patterns and anomalies for tasks [32]. In more detail, a deep MLP is characterized by additional concealed layers, which render it deep. The model can learn more intricate and abstract representations from the data due to these deep architectures. The additional layers are particularly beneficial for tasks that involve unstructured data, such as images, audio, and text, as they aid in the capture of higher-order relationships between input features [21] [42] [66].

Importance of Using Deep MLP: Deep MLPs are essential for addressing issues that necessitate the extraction of complex patterns from data. The model's ability to represent complex functions increases as the number of layers increases, enabling it to handle tasks that simple, shallow networks are unable to do. Deep MLPs are a potent instrument in regions where the relationship between input and output is highly non-linear, as they can approximate any function [6]. Some examples of it to effectively model non-linear relationships are medical diagnosis [47], classify user behavior on social networks [6], and detection of patterns and anomalies for tasks like fraud detection or behavior prediction [32].

The forward pass in a deep MLP is like an MLP but with multiple hidden layers. For each layer, the input to a neuron is a weighted sum of the outputs from the previous layer $a^{(l-1)}$ (or the input data for the first hidden layer) plus a bias term. The equation and general form of the algorithm are described below [23]:

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)} \tag{4.1}$$

where:

- $W^{(l)}$ is the weight matrix for layer l,
- $a^{(l-1)}$ is the activation (output) of the previous layer,
- $b^{(l)}$ is the bias vector for layer l,
- $z^{(l)}$ is the net input to the neurons in layer l.

4.1.2 Dense Layer

An essential element of neural networks, particularly deep neural networks, is called dense layers. It is also referred to as a fully connected layer. In a dense layer, each neuron is connected to every neuron in both the preceding and succeeding layers. This structure enables the layer to learn from all of the features of the incoming data, making it extremely effective at capturing complex relationships [26].

Dense layers in DNNs are often used in the network's latter stages to perform tasks such as classification or regression [29]. These layers take the output of preceding layers (for example, convolutional or recurrent layers) and combine the learned characteristics to make final predictions. Dense layers in DL models, such as MLP, connect input and output, allowing for more accurate and balanced predictions across all classes, including minority ones

4.1.3 Dropout Layer

The dropout layer is a regularization technique used to prevent overfitting in artificial neural networks, particularly in DL models [57] [61]. During training, it randomly drops out (sets to zero) a subset of the neurons in the layer. It forces the model to become more robust by not depending too heavily on certain neurons. This strategy promotes network redundancy, allowing it to more accurately generalize to previously unseen data.

In DNNs, dropout is often used after dense layers to enhance generalization. It is especially beneficial in minimizing overfitting when the network has a high number of parameters as it inhibits co-adaptation among neurons [25]. Dropout is disabled during testing (inference) to ensure that all neurons contribute to the model's predictions. It briefly deactivates a random collection of neurons during training to minimize overfitting and ensures that the model does not rely too heavily on any neurons. Dropout layers do not result in data loss or hinder future layers from learning because:

- Dropout is random and temporary, so all neurons participate in training across different iterations.
- It is not applied during inference, meaning all neurons are active when making predictions.
- Output scaling compensates for the dropped neurons, ensuring the remaining neurons contribute properly.

4.1.4 Principal Component Analysis

Principal component analysis is a widely used statistical technique in machine learning for dimensionality reduction. It transforms large datasets with many features into a

smaller set of uncorrelated variables called principal components. The goal of PCA is to simplify complex datasets by minimizing the number of features while retaining as much variance (information) as possible. This method is highly effective for analyzing relationships among variables and representing them in a more compact form, with minimal loss of information, making it ideal for machine learning tasks involving large datasets [30]. The key steps are given below:

1. **Data Matrix:** Let X be the $n \times p$ matrix, where n is the number of observations and p is the number of features.

2. **Mean Centering:** Subtract the mean from each feature to center the data:

$$\tilde{X} = X - \mu \tag{4.2}$$

   where μ is the mean of each feature.

3. **Covariance Matrix:** Compute the covariance matrix:

$$C = \frac{1}{n-1} \tilde{X}^T \tilde{X} \tag{4.3}$$

4. **Eigenvalue Decomposition:** Perform eigenvalue decomposition on the covariance matrix to obtain eigenvectors (principal components) and eigenvalues:

$$Cv = \lambda v \tag{4.4}$$

   where v are the eigenvectors and λ are the eigenvalues.

5. **Principal Components:** Project the data onto the top k eigenvectors:

$$Z = XV_k \tag{4.5}$$

where Z is the transformed data in the reduced space.

These steps summarize PCA in a compact form, showing how it reduces dimensions while preserving important information in the data. By transforming data into a lower-dimensional space, PCA minimizes the complexity of ML models, often enhancing their efficiency and interpretability. This dimensionality reduction technique is beneficial in preprocessing stages, where it can help mitigate overfitting by eliminating redundant features.

4.2 Operational Components and Techniques

4.2.1 Activation Functions

In ANNs, an activation function is a mathematical function that introduces non-linearity to the model. It enables the model to learn intricate data patterns. The capacity of neural networks to address complex issues would be significantly diminished if activation functions were absent. After that, they would behave similarly to linear models. It decides whether a neuron should be activated or not by turning its input signal into an output that can be sent to the next layer. In DNNs, the activation function can significantly impact the model's performance and training efficiency [5] [9]. These are the most popular and frequently used non-linearity layers. These functions include the logistic sigmoid, tanh, rectified linear unit (ReLU), and exponential linear unit). Below are the details ReLU and Sigmoid are provided as these two have been used in this project:

ReLU Activation Function

The rectified linear unit is a popular activation function, especially in feedforward, CNN, and other DL models. Its significance stems from its potential to alleviate the disappearing gradient problem, which is typical in deep networks. The disappearing gradient problem can make model training difficult by delaying or stopping learning. Relu addresses this by enabling gradients to flow efficiently, particularly in deep architectures, simplifying complex pattern learning and speeding up convergence during training [2].

The relu activation function is defined as:

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{4.6}$$

This means that the function outputs zero when the input is negative and returns the input itself when the input is positive.

ReLU is extremely efficient because of its simplicity, which decreases computational complexity. However, it is prone to the 'dying relu' problem, which occurs when neurons become dormant and regularly output zero. Despite its limitations, it is still one of the most successful activation functions for DNNs, particularly in hidden layers.

Sigmoid Activation Function

The sigmoid function is another activation function commonly employed in the output layer. It is used particularly for binary classification tasks. This function transforms the input into a number between 0 and 1, making it perfect for generating proba-

bilities. This feature makes it appropriate for applications requiring the output to indicate probabilities or make binary judgments.

The sigmoid activation function is defined as [41]:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.7}$$

In this equation, x represents the input value, which can be any real number. The term e^{-x} refers to the exponential decay, where e (approximately 2.718) is raised to the power of $-x$, resulting in a rapid decrease as x becomes larger. The denominator, $1 + e^{-x}$, shifts and scales the exponential output, ensuring that the function maps values into the range $[0, 1]$. Finally, the normalized output, $\frac{1}{1+e^{-x}}$, provides a smooth transition between 0 and 1, making it particularly useful for interpreting outputs as probabilities.

4.2.2 Class Weights

Class weights are a crucial technique for handling imbalanced datasets in which some classes are underrepresented in comparison to others [3]. This is typical in real-world classification tasks such as fraud detection or medical diagnosis, where the minority Class 0 (fraud) is frequently more difficult to predict effectively. Without class weights, models tend to favor the dominant one, resulting in poor performance for the minority. Class weights increase prediction accuracy by altering the loss function. It distributes more weight to minority classes and less weight to majority classes.

Class weights are a crucial tool for increasing a model's overall performance.By effectively treating imbalanced data, measurements such as precision, recall, and f1-score

can be enhanced for the minority class. Furthermore, class weighting aids in the prevention of bias, which can occur when a model gets skewed towards the majority class, resulting in deceptive accuracy rankings. Class weights keep the model balanced, allowing it to produce more accurate and fair predictions across all classes, especially minority ones. The equations for using this is briefly described below with an example:

Weighted Loss Function

In a binary classification problem where Class 0 is the majority and Class 1 is the minority, the weighted version of the binary cross-entropy loss function with class weights is designed to account for class imbalance.

- y_i represents the actual values (0 or 1),
- $\hat{y}_i$ is the predicted probability by the neural network.

The loss function is expressed as:

$$\text{Loss} = -\sum_{j=1}^{k} w_j \cdot (y_j \cdot \log(\hat{y}_j)) \tag{4.8}$$

where k is the number of classes, and w_j are the class weights.

For the cost function:

$$\text{Cost} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij} \cdot \log(\hat{h}_{ij}) \tag{4.9}$$

where:

- n is the total number of samples,
- k represents the classes (binary in this case).

4.2.3 Adam Optimizer

Adam optimizer, also called adaptive moment estimation. It is an optimization algorithm employed in DL models to train ANNs. It integrates the benefits of two other widely used optimizers: adaptive gradient algorithm, which is proficient in managing non-stationary objectives, and root mean square propagation, which is effective with sparse gradients. Adam's learning rates are dynamically adjusted for each parameter. It helps to compute the first and second moments (mean and variance) of the gradients. It also ensures that it is highly adaptable to a variety of data and models.

Adam's distinguishing qualities are noteworthy. The initial benefit is that its adaptive learning rates enable it to modify by gradient moments, resulting in more precise updates. Furthermore, adam is computationally efficient and necessitates less memory relative to other optimizers. The final component is bias correction, which mitigates the potential bias that may develop during the initial phases of training when the number of data points is restricted. Through this bias correction, the first and second-moment estimates are rendered impartial, thereby enhancing the precision and dependability of the training results.

The adam optimizer updates the parameters of a ANN by using estimates of the first and second moments of the gradients, along with bias corrections. Here's a breakdown of the key equations for the adam optimizer:

1. Update the first moment estimate (mean of the gradients):

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{4.10}$$

   - m_t is the exponentially decaying average of past gradients (first moment estimate).
   - g_t is the gradient at time step t.
   - β_1 is the decay rate for the first moment estimate, typically set to 0.9.

2. Update the second moment estimate (uncentered variance of the gradients):

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{4.11}$$

   - v_t is the exponentially decaying average of squared gradients (second moment estimate).
   - β_2 is the decay rate for the second moment estimate, typically set to 0.999.

3. Bias correction for both the first and second moments:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{4.12}$$

   - $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected estimates of the first and second moments.

4. Update the parameters (weights) using the following rule:

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{4.13}$$

   - θ_t represents the model parameters (weights) at time step t.

- η is the learning rate.
- ϵ is a small constant (usually 10^{-8}) to avoid division by zero.

4.2.4 Binary Cross-Entropy

Binary cross-entropy is a loss function commonly used in binary classification problems. It is also called logarithmic loss. It evaluates the effectiveness of a classification model where the result is a probability value between 0 and 1 [36] [55]. By penalizing incorrect predictions more heavily, it encourages the model to output probabilities closer to the true class labels, enhancing classification accuracy.

Binary cross-entropy is defined as:

- y is the true label (0 or 1),
- $\hat{y}$ is the predicted probability (between 0 and 1).

The Binary cross-entropy loss for a single example is given by:

$$\text{Loss} = -\left(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})\right) \tag{4.14}$$

Furthermore,

- If $y = 1$ (the true label is positive):
  - The loss is $-\log(\hat{y})$, which penalizes the model if the predicted probability $\hat{y}$ is far from 1.
- If $y = 0$ (the true label is negative):
  - The loss is $-\log(1 - \hat{y})$, for predicted probability $\hat{y}$ is far from 0.

4.3 Evaluation Matrix

An evaluation matrix is a structured table or instrument employed to evaluate and compare the performance of a model, typically in the context of data analysis or ML [54]. It encompasses critical performance metrics, including precision, recall, f1-score, and accuracy, which are determined by the predictions of the model [20].

The confusion matrix, which serves as the basis for the computation of numerous evaluation metrics, is frequently referred to as the evaluation matrix in the context of classification tasks. The confusion matrix is a grid that deconstructs predictions into:

- True Positives (TP): Correctly predicted positive cases.
- True Negatives (TN): Correctly predicted negative cases.
- False Positives (FP): Incorrectly predicted positive cases (false alarms).
- False Negatives (FN): Incorrectly predicted negative cases (missed instances).

Precision:

Precision measures the proportion of correctly identified fraud cases. It is calculated from all cases that the model predicted as fraud.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{4.15}$$

This metric is important in cases where false positives (false alarms) need to be minimized, such as in fraud detection.

Recall:

Recall measures the proportion of actual fraud cases that were correctly identified by the model.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{4.16}$$

Recall is crucial when the goal is to minimize false negatives (missed fraud cases), ensuring the model catches as many frauds as possible.

F1-Score:

The f1-score represents the harmonic mean of precision and recall. It balances the two measures by taking into account both false positives and false negatives. It is especially useful for circumstances when we need to avoid significant compromises between precision and recall. The formula for the f1-score is:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.17}$$

The F1 score is most useful when the dataset is imbalanced (e.g., fraud detection) because it provides a single metric that gives equal weight to both precision and recall. In fraud detection, it helps ensure the model is not only catching fraud cases (high recall) but also minimizing false alarms (high precision).

Support:

Support refers to the number of actual occurrences of each class in the dataset. It indicates how many transactions in the dataset were fraudulent and how many were legitimate. Support helps to evaluate precision, recall, and f1-score, as a small support value for a Class 1 (like fraud) can explain why certain metrics may be lower or higher.

Accuracy:

Accuracy gives the overall correctness of the model by measuring the proportion of true predictions (both fraud and non-fraud) out of the total number of predictions.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total Number of Predictions}} \tag{4.18}$$

While accuracy can give a general idea of the model's performance, it is often misleading in imbalanced datasets like fraud detection, where the majority Class 0 (non-fraud) dominates the dataset. In such cases, metrics like precision and recall provide better insights into the model's effectiveness.

4.4 Proposed Architecture

The proposed deep MLP model combines the basic architecture and learning methods of a regular feedforward technique of an MLP—such as input, hidden, and output layers, with the enhanced depth of multiple hidden layers shown in figure below [51]. The depth allows for hierarchical feature learning, capturing complex patterns, and enhancing generalization, making deep MLPs ideal for handling unseen data.

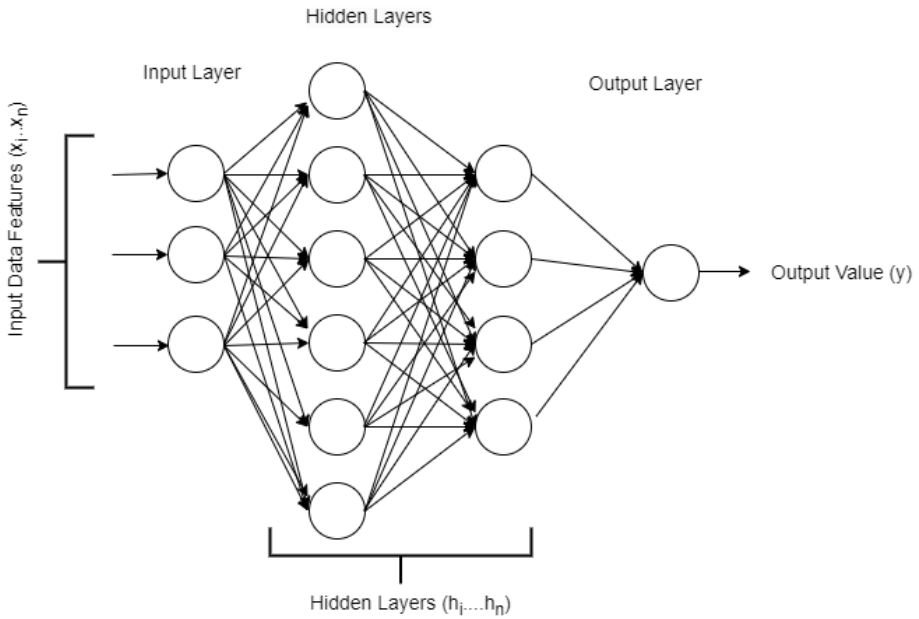

Figure 4.1: Model architecture

Figure 4.1 represents a deep MLP architecture with several hidden layers, which is ideal for a task like credit card fraud detection. The detailed architecture is described below:

Input Layer: The architecture begins with an input layer that extracts features from the dataset. Credit card fraud detection features include transaction attributes such as transactional amount, time, client ID, and other relevant parameters. The input layer has the same number of nodes as there are features in the dataset. Each input node receives a certain feature from the transaction data and forwards it to the network.

Hidden Layer: This deep MLP model has several hidden layers, as shown in figure 4.1. The hidden layers are the primary computational layers where the actual learning occurs. Each layer is fully linked, meaning each node (or neuron) in one layer communicates with every node in the next. Neurons use activation functions such as the relu to bring nonlinearity to learn more complex patterns in the data.

Each layer processes the input from the previous layer, applies the activation function, and then transmits the output to the next layer. This is an example of a deep learning model, as it has numerous hidden layers. The network's depth (the number of layers) enables it to detect hierarchical patterns in data, which is particularly beneficial for distinguishing between fraudulent and non-fraudulent transactions.

Activation Functions and Non-linearity: Each hidden layer has an activation function such as relu, enabling the model to learn nonlinear relationships in the data. relu's nonlinearity assures that the model can generalize beyond basic linear translations of input and output. This nonlinearity is critical for fraud detection tasks, as the patterns that distinguish fraudulent from genuine transactions are frequently complicated and not linearly separable.

Output Layer: The output layer is where the final prediction is generated. In the case of binary classification (fraud vs. non-fraud), the output layer consists of a single neuron with a Sigmoid activation function. The Sigmoid function returns a probability between 0 and 1, which can be read as the possibility that a transaction is fraudulent. The threshold (usually 0.5) is then used to determine whether the transaction was fraudulent (1) or non-fraudulent (0).

Feedforward Process and Optimization: The model is developed using feedforward neural networks, the information flows in a single direction: from the input layer to the hidden layers, and finally to the output. Each layer processes the previous layer's input and forwards the output to the next layer. During this forward trip through the network, the model produces predictions based on the current weight values.

During training, the model's prediction is compared to the actual result (fraudulent or non-fraudulent transaction). The error (or loss) is derived by subtracting the

expected and actual numbers. This error is then utilized to update the weights in the network using an optimization technique like Adam, which minimizes the loss function (binary crossentropy). This method is iterated until the model's predictions improve and the error decreases.

The feedforward model passes input through the network layers, weight optimization depends solely on the direct relationship between input features and the target output. This configuration ensures that the network learns to accurately categorize transactions without relying on backward error propagation. Hierarchical data patterns are particularly beneficial for discriminating between fraudulent and non-fraudulent transactions.

Regularization(Dropout): Regularization techniques like dropout can be used on the hidden layers to prevent overfitting, which is a prevalent problem in deep networks. Dropout a subset of neurons at random throughout each training cycle, driving the model to acquire more resilient and broader patterns. This allows the model to perform better on previously unknown data, which is important in fraud detection when real-world test data may differ from the training set propagation. Dropout helps prevent overfitting and improves the model's ability to generalize to real-world data.

In a nutshell, the deep MLP architecture for credit card fraud detection is made up of an input layer that accepts transaction data, numerous hidden layers that learn complex patterns using non-linear activation functions, and an output layer that classifies transactions as fraudulent or non-fraudulent. The network's depth allows it to capture detailed patterns in the data, while regularization techniques such as dropout maintain the model's robustness and generalization to fresh data. The model is trained by feedforward and optimized to reduce classification errors, making it an effective tool for identifying fraud in credit card transactions.

4.5 Algorithm: Credit Card Fraud Detection with Deep Multilayer Perceptrons

The following algorithm outlines a combined approach for the feedforward training of a deep MLP model, to detect credit card fraud. This approach uses both train-test split ratio analysis and 10-fold old cross-validation. This algorithm states both validation techniques and guarantees accurate data preprocessing, model compilation, training, evaluation, and result storage.

The algorithm mentioned below provides a flexible framework for implementing both k-fold cross-validation and train-test split ratio analysis. It ensures the ability and accuracy of reproduction in detecting credit card fraud using a deep MLP model. This approach allows for robust model evaluation by testing different data splits, thus enhancing the model's generalization to new or unseen transactions.

Algorithm 1 Credit Card Fraud Detection with Deep Multilayer Perceptrons

Load dataset from `'creditcard.csv'`
Handle missing values using forward fill
Normalize the 'Amount' column using StandardScaler
Drop 'Time' and 'Amount' columns
Split dataset into features X and labels y
Create directories for 10-fold only and store in `'output'`
Set random seed for NumPy
Set random seed for TensorFlow
if K-fold cross-validation **then**
    Initialize K-fold with 10 splits and shuffle=True
else
    Split the dataset into training and testing sets using the following ratios: 90:10, 80:20, 75:25, 70:30, and 60:40.
end if
Calculate class weights based on class distribution
Define function to build a neural network with hidden layers and dropout layers
if 10-fold cross-validation **then**
    for each fold **do**
        Split data into training and testing for the fold
    end for
else
    Use $X\_train, X\_test, y\_train, y\_test$ directly
end if
Compile model using Adam optimizer and binary crossentropy loss
Set up early stopping with a patience of 10 epochs
if 10-fold cross-validation **then**
    for each fold **do**
        Apply class weights to handle imbalance
    end for
else
    Train model using training data and validate on test data
    Apply class weights to handle imbalance
end if
Measure training time (train-test split only)
if 10-fold cross-validation **then**
    for each fold **do**
        Make predictions on the validation set
        Evaluate model using accuracy score and classification report
    end for
else
    Make predictions on the test set
    Evaluate model using accuracy score and classification report
end if
if 10-fold cross-validation **then**
    Save the results of each fold in a text file
else
    Save the test set evaluation results in a text file
end if
Print 'Training Finished!' when completed

4.6 Improvement Strategies and Benefits

The deep MLP model contributes to improved detection accuracy, resilience to imbalanced data, and increased computational efficiency. These innovations lead to greater reliability and precision in identifying complex patterns, enhancing the model's overall performance for tasks requiring high sensitivity:

- Class imbalance handling: Class weighting is implemented to guarantee that the model prioritizes the identification of fraudulent transactions, thereby rectifying the substantial imbalance in the dataset.

- Dropout for regularization: Dropout layers are incorporated to prevent overfitting, thereby improving the model's capacity to detect fraud and generalize in unseen data.

- Lower probability threshold: The output probability threshold is set to 0.5, which increases sensitivity to fraud cases and improves detection rates at the expense of slightly more false positives and is lower than the default 0.5.

- Early stopping for efficient training: Early stopping prevents overfitting and ensures efficient model training by halting the process if the validation loss does not improve after 10 epochs.

- Optimized training parameters: The model is trained for up to 100 epochs and batch size of 256, for computational efficiency and learning effectiveness.

The model can more effectively address the distinctive challenges of credit card fraud detection because of the innovations. It leads to improved precision, recall, and generalization of data. By leveraging these advancements, the model minimizes false positives and negatives, enhancing its reliability in high-stakes financial environments.

Chapter 5

DESCRIPTION OF IMPLEMENTATION

This chapter details the step-by-step implementation of the credit card fraud detection model. It outlines the details of data preprocessing, model architecture, and training procedures, emphasizing techniques for handling class imbalance and enhancing model performance. Additionally, it discusses the tools and libraries used, providing a comprehensive view of the methods applied to build a reliable detection system.

5.1 System and Language Specifications

This section describes the computational setup and programming tools used for the model's implementation. It covers the system environment specifications, including hardware and software configurations. Furthermore, it details the programming languages and libraries essential for developing and executing the credit card fraud detection model.

Computational Environment

The computational environment for this project was based on a Dell Inspiron 16 5630 laptop, featuring a 13th Generation Intel® Core™ i7-1360P processor with a maximum clock speed of 5.00 GHz, and 16GB of 4800MHz LPDDR5 memory. The processor's high-speed capabilities and multi-core architecture provided the necessary power for handling intensive tasks like training deep learning models. The system also included a 1TB M.2 PCIe NVMe SSD, which facilitated fast data access and reduced the time required for data loading and saving during model training. The laptop ran Windows 11 Home, offering a modern and stable environment compatible with popular machine learning libraries such as TensorFlow and Keras. Additionally, the 4-cell 64WHr battery and 65W Type-C power adapter supported long computational tasks. This setup allowed for the efficient execution of the credit card fraud detection model and ensured smooth performance when processing large datasets.

Programming Language Specification

The credit card fraud detection model was developed using Python, a widely used language ideal for machine learning and deep learning projects. Python offers extensive libraries such as NumPy and Pandas for efficient data manipulation, allowing structured handling of the credit card transaction dataset. Visualization libraries like Matplotlib were utilized to explore patterns in the data, while TensorFlow and Keras facilitated building and training the deep MLP model. These tools provided flexibility for managing model layers, applying the adam optimizer, and implementing techniques like early stopping to prevent overfitting. Python's ecosystem also supports the evaluation of metrics like precision, recall, and f1-score, which were critical for optimizing the model's performance in detecting fraudulent transactions.

5.2 Integrated Development Environment

In this project, the robust capabilities of PyCharm 2024.2.1 were utilized as an integrated development environment to enhance the project's overall efficacy and optimize the workflow while developing the credit card fraud detection model. The following capabilities of PyCharm were crucial features for its selection in this project:

Data View Enhancements: The improved data view tool is employed to visualize data with new heatmap color schemes (sequential and diverging) during the model training process. This simplified the process of interpreting the contributions of various features in the dataset to the model's predictions and facilitated the model's fine-tuning.

Debugger and Error Handling: PyCharm's integrated debugger was particularly helpful in identifying errors during model training, particularly when assessing metrics such as precision, recall, and f1-score and managing data imbalance. It enabled efficiently resolving issues, inspecting variable states, and stepping through the code.

Python 3.13 Support: The integrated development environment was able to facilitate the implementation and optimization of new functions during model development, particularly when working with user-defined narrowed functions, because of PyCharm 2024.2.1's support for Python 3.13. These features included sophisticated type inference and code completion.

5.3 Libraries and Frameworks

The libraries provide essential functionalities for data manipulation, model building, and evaluation. Their efficient implementations support faster development and enhance the overall performance of the model. Here is a brief explanation of the key

Python libraries and modules used in this project [20]:

- Pandas as pd

  Pandas is a versatile data manipulation and analysis library in Python, primarily used for working with structured data, such as CSV or excel files. It enables efficient loading, preprocessing, and organization of datasets, which is essential for preparing data for analysis. Additionally, Pandas provides powerful tools for data exploration before feeding it into an ML model.

- NumPy as np

  NumPy is a library that offers a collection of mathematical functions for manipulating arrays, with support for large, multi-dimensional arrays and matrices. It provides tools for efficient numerical computations, making it essential for data processing and analysis. This functionality is crucial for model development, allowing for faster and more optimized calculations.

- Time and OS

  - Time: Used to measure execution time for various operations, useful in tracking model training time.

  - OS: Enables interaction with the operating system, such as handling files and directories during data processing.

- Train-test split (from scikit-learn)

  This function is used to divide the dataset into training and testing sets, facilitating the evaluation of model performance on unseen data. It is essential for assessing a model's generalization capabilities. By providing a separate dataset for testing, it helps prevent overfitting.

- TensorFlow and keras

  TensorFlow is a robust open-source deep learning framework, while Keras, integrated within TensorFlow, offers a user-friendly interface for constructing neural networks. Together, they are used in this model to define and train the deep MLP, incorporating layers like dense and dropout. These layers help to enhance performance and provide regularization.

- EarlyStopping (keras callback)

  This callback function monitors a specific metric, such as validation loss, during model training. It stops the training process when the metric stops improving. By halting training at this point, earlyStopping effectively prevents overfitting.

- Classification metrics (scikit-learn) Classification report summarizes key metrics like precision, recall, and f1-score for evaluating model performance.The accuracy score provides the ratio of correctly predicted instances to total instances.

- StandardScaler (scikit-learn)

  StandardScaler normalizes features so they have a mean of zero and a unit variance. This scaling is essential for algorithms sensitive to feature magnitude differences. Standardizing feature values across different ranges improves model performance.

- Compute class weight (scikit-learn)

  This function calculates class weights to address class imbalance in datasets. It assigns higher weights to minor classes, such as fraudulent transactions, to make the model more responsive to these cases. Handling class imbalance in this way helps improve model accuracy for less frequent classes.

5.4 Data Management

In ML projects, data management is essential. The performance and reliability of the model are directly influenced by the quality, source, and structure of the dataset. This study's dataset's features, character, and the application of PCA to optimize data processing are described in the following sections.

Source of Dataset

The dataset for this study includes credit card transactions performed by European cardholders in September 2013. As part of a big data mining and fraud detection effort, Worldline collaborated with the machine learning group of the Université Libre de Bruxelles to collect and analyze the data. This dataset was sourced from Kaggle.com. The dataset contains transactions over two days and is available through public repositories.

Dataset Features

The dataset consists of 284,807 transactions, out of which 492 are identified as fraudulent, representing only 0.172% of the transactions. It is a highly imbalanced dataset, where the positive Class 1 (fraud) accounts for a very small portion of the total transactions. The dataset contains only numerical features, most of which have been transformed using PCA. There are 30 input variables in total:

- V1 to V28: These are the principal components generated by PCA from the original data due to confidentiality restrictions.
- Time: This feature represents the time elapsed in seconds between each transaction and the first transaction in the dataset.
- Amount: The amount of each transaction, that is not PCA done.

- Class: The target variable, where 1 represents fraudulent transactions and 0 represents non-fraud transactions.

	Time	V1	V2	V3	V4	V5	V6.....V22	V23	V24	V25	V26	V27	V28	Amount	Class
2	0	-1.35981	-0.07278	2.536347	1.378155	-0.33832		-0.11047	0.066928	0.128539	-0.18911	0.133558	-0.02105	149.62	0
3	0	1.191857	0.266151	0.16648	0.448154	0.060018		0.101288	-0.33985	0.16717	0.125895	-0.00898	0.014724	2.69	0
4	1	-1.35835	-1.34016	1.773209	0.37978	-0.5032		0.909412	-0.68928	-0.32764	-0.1391	-0.05535	-0.05975	378.66	0
5	1	-0.96627	-0.18523	1.792993	-0.86329	-0.01031		-0.19032	-1.17558	0.647376	-0.22193	0.062723	0.061458	123.5	0
6	2	-1.15823	0.877737	1.548718	0.403034	-0.40719		-0.13746	0.141267	-0.20601	0.502292	0.219422	0.215153	69.99	0
7	2	-0.42597	0.960523	1.141109	-0.16825	0.420987		-0.0264	-0.37143	-0.23279	0.105915	0.253844	0.08108	3.67	0
8	4	1.229658	0.141004	0.045371	1.202613	0.191881		-0.1541	-0.78006	0.750137	-0.25724	0.034507	0.005168	4.99	0
9	7	-0.64427	1.417964	1.07438	-0.4922	0.948934		0.057504	-0.64971	-0.41527	-0.05163	-1.20692	-1.08534	40.8	0
10	7	-0.89429	0.286157	-0.11319	-0.27153	2.669599		-0.20423	1.011592	0.373205	-0.38416	0.011747	0.142404	93.2	0
11	9	-0.33826	1.119593	1.044367	-0.22219	0.499361		-0.12079	-0.38505	-0.06973	0.094199	0.246219	0.083076	3.68	0
12	10	1.449044	-1.17634	0.91386	-1.37567	-1.97138		0.02774	0.500512	0.251367	-0.12948	0.04285	0.016253	7.8	0
13	10	0.384978	0.616109	-0.8743	-0.09402	2.924584		0.00913	0.99671	-0.76731	-0.49221	0.042472	-0.05434	9.99	0
14	10	1.249999	-1.22164	0.38393	-1.2349	-1.48542		0.084668	0.392831	0.161135	-0.35499	0.026416	0.042422	121.5	0
15	11	1.069374	0.287722	0.828613	2.71252	-0.1784		-0.07141	0.104744	0.548265	0.104094	0.021491	0.021293	27.5	0
16	12	-2.79185	-0.32777	1.64175	1.767473	-0.13659		1.020586	0.028317	-0.23275	-0.23556	-0.16478	-0.03015	58.8	0
17	12	-0.75242	0.345485	2.057323	-1.46864	-1.15839		-0.25657	-0.06508	-0.03912	-0.08709	-0.181	0.129394	15.99	0
18	12	1.103215	-0.0403	1.267332	1.289091	-0.736		0.013802	0.103758	0.364298	-0.38226	0.092809	0.037051	12.99	0
19	13	-0.43691	0.918966	0.924591	-0.72722	0.915679		-0.15686	-0.88839	-0.34241	-0.04903	0.079692	0.131024	0.89	0
20	14	-5.40126	-5.45015	1.186305	1.736239	3.049106		2.458589	0.042119	-0.48163	-0.62127	0.392053	0.949594	46.8	0

Figure 5.1: Features of dataset

As shown in Figure 5.1, a collected dataset containing credit card customers' information while purchasing. To handle the customer information safely, PCA has been applied for the features of V1-V28, while Time, Amount, and Class are given as they are.

Dataset Nature

The nature of the dataset is highly imbalanced, with fraud cases representing a small fraction (0.172%) of the total transactions. This imbalance poses a significant challenge for ML algorithms, as models tend to be biased towards the majority Class 0 (non-fraud), leading to poor detection of the minority Class 1 (fraud). To address this, techniques such as class weighting, oversampling of the minority Class 1, or undersampling of the majority Class 0 can be employed during model training to improve the model's ability to detect fraud. The original features of the dataset have been transformed using PCA, due to confidentiality reasons. The only variables not transformed by PCA are Time, Amount, and Class which remain in their raw form and are used to track transaction timing and monetary values, respectively.

5.5 Data Preprocessing

The dataset's structure plays a vital role in effective model development. Data preprocessing is crucial to ensure the dataset is ready for analysis. The following subsections outline the techniques applied for normalization, handling missing values, and class weighting in the credit card fraud detection model.

Normalization

In the preprocessing stage, the Amount column was normalized using the StandardScaler. This was done to standardize the values, ensuring they have a mean of zero and a standard deviation of one. This scaling is particularly useful for improving the model's convergence during training. After normalization, the original Amount and Time columns were dropped as they were no longer needed, reducing unnecessary noise in the input data.

Missing Values

Missing value is very normal in any dataset. However, addressing it individually is sometimes time-consuming, especially for large datasets like transactional data. So, in this model, the missing values in the dataset were handled by applying the forward fill method. This technique replaced missing values with the last available valid entry, ensuring data continuity without introducing significant bias, resulting in a complete dataset with no gaps.

Class Imbalance

To handle the imbalance between fraudulent and non-fraudulent transactions, the model required adjustment for this imbalance. Class weights were calculated using the compute_class_weight function from scikit-learn. This gave greater importance to

the minority Class 1 (fraudulent transactions), ensuring that the model learned to detect these rare cases more effectively during training. By applying these weights, the model could focus on identifying fraud, improving its overall sensitivity to the minority Class 1.

5.6 Data Analysis Techniques

The model's generalization ability and efficacy were evaluated using two different data analysis techniques. This section outlines the methods used to split the dataset to ensure comprehensive model assessment. Techniques like multiple data split ratios and cross-validation were implemented to achieve robust evaluation results.

Data Splitting Method

To assess the model's ability to generalize, the dataset was divided into multiple training and test sets using different ratios such as 90:10, 80:20, 70:30, 75:25, and 60:40. The model was trained on the training sets to learn the underlying patterns and relationships in the data, and then evaluated on the unseen test sets to gauge its performance. This approach allowed for a comprehensive comparison of results across different data splits, ensuring that the model could handle varying data distributions effectively.

k-fold Cross-Validation Approach

The stratified k-fold cross-validation technique, with $k = 10$, was used to validate the model's accuracy. In this approach, the dataset was divided into 10 equal parts or folds, and the model was trained and tested iteratively on each fold. For each iteration, one fold served as the test set while the remaining nine folds were used for training, ensuring that each part of the data contributed to both training and testing.

This approach provided a more robust and reliable estimate of the performance.

5.7 Model Performance Assessment

The performance of the model was evaluated using the confusion matrix, which provides a detailed breakdown of the model's classification results into four main components: true positives, true negatives, false positives, and false negatives. These values are crucial in calculating key metrics like precision, recall, f1-score, and accuracy, which measure the model's ability to identify fraudulent transactions correctly [54]. Their details are discussed below, based on the model [20]:

Using this confusion matrix, the key metrics like:

1. Precision: $\frac{\text{TP}}{\text{TP+FP}}$ – How many predicted frauds were actually fraud.
2. Recall: $\frac{\text{TP}}{\text{TP+FN}}$ – How many actual frauds were detected.
3. F1-score: $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision+Recall}}$ – A balanced metric combining precision and recall.
4. Accuracy: $\frac{\text{TP+TN}}{\text{Total}}$ – Overall correctness of the model.

In summary, after training, the model was assessed using the test data. The accuracy score, which indicates the model's overall accuracy, and the classification report, which offered precise metrics such as precision, recall, and f1-score for both fraud and non-fraud classes, were employed to evaluate predictions. These metrics were instrumental in evaluating the model's capacity to identify fraudulent transactions and its performance on an imbalanced dataset.

5.8 Flow Chart: Credit Card Fraud Detection with Deep Multilayer Perceptrons

The flowchart below illustrates the methodology of the combined 10-fold cross-validation and train-test split ratio analysis approach, which was implemented for a neural network model that was specifically designed to detect credit card fraud.

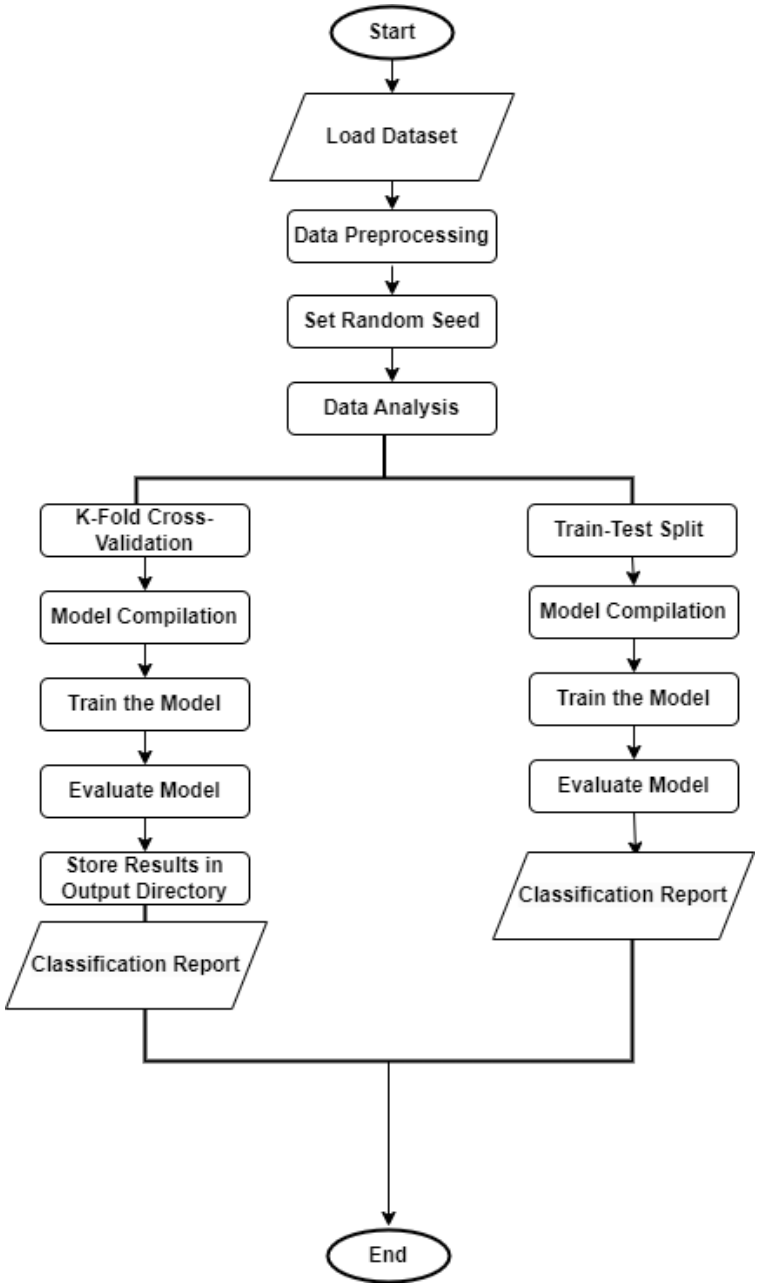

Figure 5.2: Flowchart of the model

Figure 5.2 represents the model implementation procedure in a flowchart diagram. The process commences with the loading of the dataset and is followed by critical phases such as data preprocessing, the establishment of directories for data storage, and the guarantee of reproducibility using random seeds. The workflow then diverges into two analysis processes: train-test split analysis and k-fold ($k = 10$)

cross-validation. To guarantee robustness and prevent overfitting, the model is subjected to numerous cycles of training, evaluation, and data storage for each fold in the k-fold($k = 10$) cross-validation path. The train-test split path, in contrast, partitions the dataset into training and testing sets and then conducts model training and evaluation in a single cycle. A model evaluation is the result of both paths, as classification reports offer comprehensive performance metrics.

5.9 Step-by-Step Model Operation

Below is a step-by-step analysis of the model algorithm, combining k-fold cross-validation and train-test split ratio analysis for the deep MLP model. This approach first applies k-fold cross-validation to ensure each part of the dataset contributes to both training and testing. At last varying train-test split ratios within each fold help further assess the model's performance across different data distributions.

Step 1: Load Dataset

In this initial step, the dataset is loaded into a data frame for further processing. The dataset is read from a CSV file named 'creditcard.csv' using the Pandas 'read_csv' function.

```
df = pd.read_csv('creditcard.csv')
```

Step 2: Data Preprocessing

The data is prepared through several preprocessing steps. Missing values are handled by applying a forward fill method to ensure no gaps in the data. The Amount column is normalized to have a mean of zero and unit variance using StandardScaler, which improves model performance on numerical data. Irrelevant columns, such as Time and Amount, are dropped after normalization. Finally, the features (X) and target labels (y) are separated for model training.

```
df.fillna(df.ffill(), inplace=True)
scaler = StandardScaler()
df[‘NormalizedAmount’] = scaler.fit_transform(df[‘Amount’].
values.reshape(-1, 1))
df.drop([‘Time’, ‘Amount’], axis=1, inplace=True)
X = df.drop(‘Class’, axis=1).values
y = df[‘Class’].values
```

Step 3: Directory Setup

This step involves creating directories for storing dataset splits and model output results. The split_dir is designated for storing 10-fold cross-validation splits, while output_dir is used to store model evaluation outputs. The os.makedirs function ensures that the directories are created if they do not already exist.

```
split_dir = ‘data_splits’
output_dir = ‘output’
if not os.path.exists(output_dir):
os.makedirs(output_dir)
```

Step 5: Data Splitting

This step prepares the data for training and testing through two methods: 10-fold cross-validation and train-test split ratio analysis. Initialize k-fold cross-validation with $k = 10$ splits, shuffling the data and setting a fixed random seed.

```
kf = KFold(n_splits=10, shuffle=True, random_state=42)
```

For train-test split ratio analysis the key components used in data splitting include the input features $\mathbf{X}$, which represent the independent variables, and the target variable $\mathbf{y}$, which the model will predict. The `train_test_split` function divides the data

into training and testing sets, with the parameter `test_size=0.20` indicating that 20% of the data is allocated for testing while 80% is used for training. After the split, **X_train** contains 80% of the feature data for training, and **X_test** holds the remaining 20% for evaluation. Similarly, **y_train** contains 80% of the target labels that correspond to **X_train**, and **y_test** holds the 20% of target labels that correspond to **X_test**.

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42)
```

Step 6: Compute Class Weights

This step involves calculating class weights to address data imbalance, especially between fraudulent and non-fraudulent transactions. The compute_class_weight function from scikit-learn calculates weights, assigning higher values to minority classes to help the model focus more on these instances.

```
class_weights = compute_class_weight(class_weight='balanced',
classes=np.unique(y), y=y)
class_weights = dict(enumerate(class_weights))
```

Step 7: Define Model

A function is defined to build the model using Keras' Sequential API. The model consists of multiple hidden layers and dropout layers to improve performance and prevent overfitting. The input_shape parameter specifies the number of input features.

```
def build_model(input_shape)
```

Step 8: Split Data for Training and Validation

For each iteration in the 10-fold cross-validation, the dataset is split into training and

testing sets. The kf.split(X) function yields indices for each fold, allowing the model to train on one subset and validate on another, ensuring each part of the dataset contributes to both training and testing.

```
for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

For train -test split ratio analysis, the predefined variables `X_train`, `X_test`, `y_train`, and `y_test` are used directly for training and testing, based on the different ratios defined earlier.

```
X_train, X_test, y_train, and y_test.
```

Step 9: Model Compilation

In this step, the model is built and compiled using the adam optimizer, which adjusts the learning rate during training, and the binary cross-entropy loss function, which is suitable for binary classification tasks. The accuracy metric is included to monitor the model's performance.

```
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

Step 10: Early Stopping Callback

Early stopping is configured to prevent overfitting by monitoring the validation loss. If the validation loss does not improve for a specified number of epochs (patience set to 10), training will stop automatically.

```
early_stopping = EarlyStopping(monitor='val_loss', patience=10)
```

Step 11: Train the Model

In this step, the model is trained with 10-fold cross-validation and train-test split ratio analysis. During each iteration of 10-fold cross-validation, the model is trained on one subset and validated on another, with class weights applied to address the class imbalance. For example, for the train-test split, the model is trained on 80% of the data and validated on the remaining 20%. Early stopping is utilized in both approaches to prevent overfitting by monitoring the validation loss, with training halting if no improvement is observed after 10 epochs.

```
# 10-fold Cross-Validation
model.fit(train_data, train_labels, epochs=100, batch_size=256,
verbose=2, validation_data=(test_data, test_labels),
class_weight=class_weights,callbacks=[early_stopping])


# Train-Test Split Ratio Analysis
model.fit(X_train, y_train, epochs=100, batch_size=256, verbose=2,
validation_data=(X_test, y_test), class_weight=class_weights,
callbacks=[early_stopping])
```

Step 12: Measure Training Time

This step calculates and prints the total time taken for the model training process. The `start_time` and `end_time` variables track the beginning and end of the training phase, and the difference between them gives the total training duration in seconds.

```
print(f"Training Time: {end_time - start_time} seconds")
```

Step 13: Evaluate the Model

After training, the model's performance is evaluated by making predictions on the test data. The model generates predictions on the training data during each fold of

the 10-fold cross-validation. The predictions are converted into binary classes using a threshold of 0.5 to classify transactions as either fraudulent or non-fraudulent. The `accuracy_score` function measures the model's accuracy, while the `classification_report` provides additional performance metrics, such as precision, recall, and f1-score, helping to evaluate the model's effectiveness within each fold.

```
y_pred = (model.predict(test_data) > 0.5).astype("int32")
print(accuracy_score(test_labels, y_pred))
print(classification_report(test_labels, y_pred))
```

In this code, the model makes predictions on the test set by applying a threshold of 0.5 to the predicted probabilities, converting them into binary classifications. The `accuracy_score` function then evaluates how accurately these predictions match the true labels, giving an overall performance metric. Finally, the `classification_report` provides a breakdown of the model's effectiveness on each class by calculating precision, recall, and f1-score, offering a detailed view of its performance on unseen data.

```
y_test_pred = (model.predict(X_test) > 0.5).astype("int32")
print(accuracy_score(y_test, y_test_pred))
print(classification_report(y_test, y_test_pred))
```

Step 14: Store Results Save the results of the evaluation, including the accuracy score and classification report, in a text file within the output directory.

```
with open
(os.path.join(output_dir, f"fold_{folder_count}_output.txt"),
'w') as f:
f.write(f"Test Set Accuracy:{accuracy_score(y_test, y_pred)}\n")
f.write("Classification Report:\n")
f.write(classification_report(y_test, y_pred))
```

Step 15: Completion

Once all folds or the single train-test split are completed, a message is printed to indicate the completion of the training process.

```
print("Training Finished!")
```

Chapter 6

RESULTS AND EVALUATION

In this section, the experimental setup, evaluation metrics used, and the results obtained from the credit card fraud detection model are described. The experiments were conducted to validate the model's effectiveness and to assess its ability to handle the challenges posed by class imbalance, overfitting, and sensitivity in detecting fraudulent transactions. The model was trained and evaluated using train-test split and k-fold cross-validation approaches to ensure robustness and minimize the risk of overfitting.

6.1 Results

This section evaluates the model's performance on the fraud detection dataset using precision, recall, and f1-scores across various train-test split ratios. These metrics determine the model's effectiveness in distinguishing fraudulent and legitimate transactions. Additionally, the model will undergo 10-fold cross-validation, to ensure robust and unbiased results.

Train-Test Split Ratio: The performance metrics of the credit card fraud detection model were evaluated across different train-test split ratios: 90:10, 80:20, 70:30, 75:25, and 60:40. It details the precision, recall, f1-score, support for each fraud and non-fraud, and the overall accuracy for each ratio:

Table 6.1: Performance metrics for different train-test split ratio

Ratios	**Class**	**Precision**	**Recall**	**F1-score**	**Support**	**Accuracy**
90:10	0	1	0.98	0.99	28435	0.997
	1	0.08	0.91	0.15	46	
80:20	0	1	0.98	0.99	56864	0.979
	1	0.07	0.89	0.13	98	
70:30	0	1	0.98	0.99	85307	0.977
	1	0.06	0.92	0.11	136	
75:25	0	1	0.99	0.99	71089	0.988
	1	0.12	0.91	0.21	113	
60:40	0	1	1	1	113732	0.995
	1	0.28	0.89	0.42	191	

In Table 6.1, a classification model's precision, recall, f1-score, support, and accuracy for Class 0 (non-fraud) and Class 1 (fraud) are assessed. For instance, with a split ratio of 90:10, Class 0 shows good performance (precision 1.00, recall 0.98, f1-score 0.99) with 28,435 examples, whereas Class 1 is also able to detect 46 instances successfully achieving 99.7% accuracy.

For the 80:20 ratio, it achieves 97.9% accuracy. The next two ratios 70:30 and 75:25 get respectively 97.7% accuracy 98.80%. At last, for the 60:40 ratio, Class 0 has ideal metrics, and Class 1 improves (precision 0.28, recall 0.89, f1-score 0.42) with 191 occurrences and 99.5% accuracy.

10-Fold Cross-Validation: 10-fold cross-validation is performed for both Class 0 (non-fraud) and Class 1 (fraud) to ensure the results are unbiased and reliable. This approach provides further validation by testing the model's consistency across multiple data partitions.

Table 6.2: Performance metrics for 10-fold cross-validation

No-of-Fold	Class	Precision	Recall	F1-score	Support	Accuracy
Fold 1	0	1	0.99	0.99	28435	0.986
	1	0.12	0.87	0.22	46	
Fold 2	0	1	0.99	0.99	28429	0.986
	1	0.1	0.86	0.19	52	
Fold 3	0	1	0.99	1	28443	0.992
	1	0.18	0.85	0.3	38	
Fold 4	0	1	0.99	0.99	28426	0.987
	1	0.1	0.87	0.19	55	
Fold 5	0	1	0.99	1	28426	0.993
	1	0.17	0.87	0.29	55	
Fold 6	0	1	0.99	0.99	28441	0.989
	1	0.14	0.9	0.24	40	
Fold 7	0	1	1	1	28433	0.998
	1	0.41	0.82	0.55	48	
Fold 8	0	1	0.99	0.99	28427	0.989
	1	0.09	0.86	0.17	53	
Fold 9	0	1	0.99	1	28428	0.99
	1	0.12	0.87	0.21	52	
Fold 10	0	1	1	1	28427	0.995
	1	0.11	0.7	0.19	53	

In Table 6.2, k-fold cross-validation results demonstrate the model's robust performance across 10 folds for both fraud (Class 1) and non-fraud (Class 0) cases. For non-fraud transactions (Class 0), the model consistently achieves high precision and recall values of 1 across all folds, with f1 scores ranging from 0.99 to 1, indicating excellent detection of legitimate transactions. For fraud cases (Class 1), the model shows a range in precision values, with a minimum of 0.09 and a maximum of 0.41, and recall values between 0.7 and 0.9. Overall, the cross-validation confirms the model's reliability for non-fraud cases while indicating the need for further improvement in detecting fraud cases to ensure unbiased and balanced outcomes.

6.2 Evaluation

Train-Test Split Ratio: The below table summarizes the model's accuracy across various train-test split ratios, providing an overall average accuracy for performance evaluation:

Table 6.3: Accuracy across different train-test split ratios

Split Ratio	Accuracy
90:10	0.990
80:20	0.979
70:30	0.977
75:25	0.988
60:40	0.990
Average Accuracy	**0.990**

In Table 6.3, highlights the accuracy levels achieved for various train-test split ratios, with an average accuracy of 0.990, demonstrating consistent and reliable model performance. The highest accuracy is observed with the 60:40 and 90:10 split, indicating its effectiveness in optimizing the model's detection capabilities.

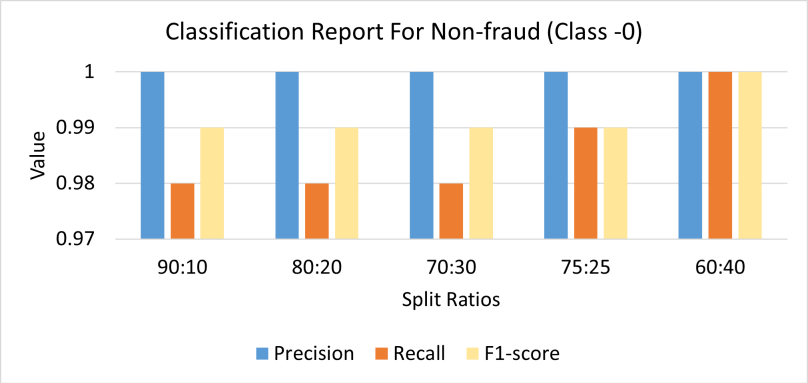

Figure 6.1: Classification report-based on split ratio for non-fraud (Class 0)

Figure 6.1, depicts precision, recall, and f1-score for non-fraud detection (Class 0) at different split ratios. The model exhibits a precision and recall of 1.0 across all splits, demonstrating its efficacy in recognizing valid transactions. The f1-score con-

sistently ranges from 0.98 to 1.0, indicating dependable performance in non-fraud categorization across various split ratios.

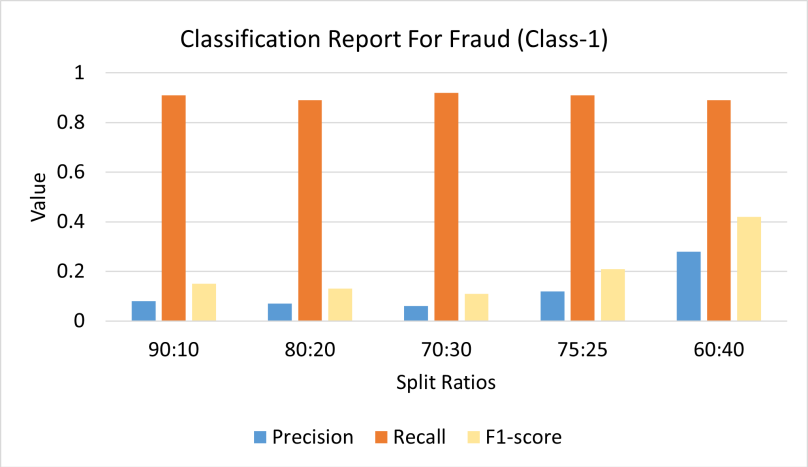

Figure 6.2: Classification report-based on split ratio for fraud (Class 1)

Figure 6.2, is a bar graph that illustrates the model's performance metrics for fraud detection (Class 1) at various split ratios. Recall remains elevated, consistently between 0.85 and 0.92, signifying the model's robust capacity to identify fraud situations. Precision varies from 0.06 to 0.28, indicating inconsistency in accurately identifying fraudulent cases. The f1-score exhibits fluctuations, ranging from 0.11 to 0.42 based on the split ratio.

Below is the comparison of precision and recall for both classes based on different train-test split ratios:

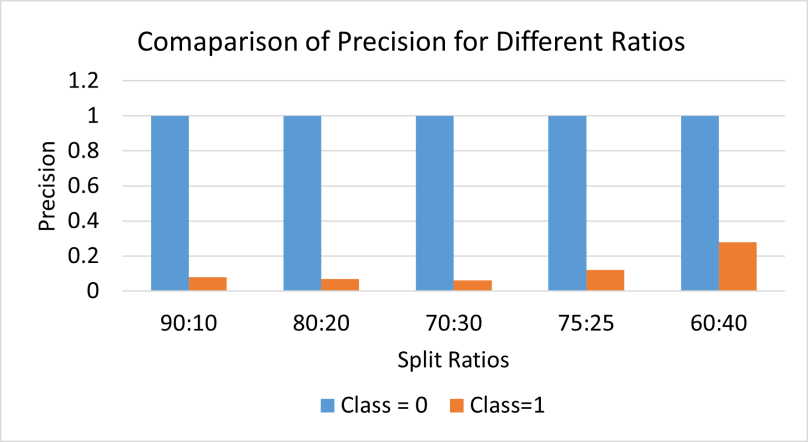

Figure 6.3: Comparison of precision for different ratios

Figure 6.3, Class 0 (non-fraud) maintains a precision of 1.0 across all split ratios, showing high accuracy. Class 1 (fraud) precision varies, from 0.06 at the 70:30 split to 0.28 at the 60:40 split.

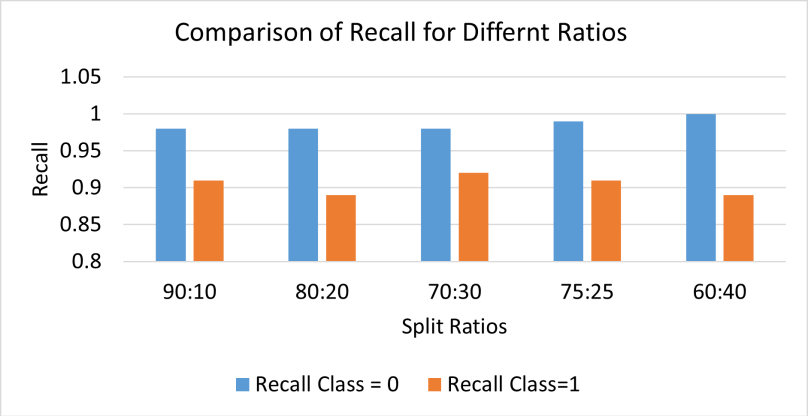

Figure 6.4: Comparison of recall for different ratios

The above-mentioned in Figure 6.4, recall values for Class 0 (non-fraud) and Class 1 (fraud) across various split ratios. Class 0 consistently achieves a high recall, ranging from 0.98 to 1.0, indicating the model's strong capability to identify legitimate transactions. In contrast, Class 1 shows recall values between 0.82 (75:25 split) and 0.92 (70:30 split).

Table 6.4: Number of epochs for different split ratios

Split Ratio	**No of Epoch**
90:10	23
80:20	23
70:30	33
75:25	31
60:40	19

This table 6.4 highlights the number of training epochs required across various train-test split ratios. It indicates that smaller training sets, such as 60:40, generally require fewer epochs.

10-Fold Cross-Validation: The table below presents the accuracy results from the 10-fold cross-validation performed on the model, along with the calculated average accuracy across all folds:

Table 6.5: Accuracy across 10-fold cross-validation

No-of-Fold	**Accuracy**
Fold 1	0.986
Fold 2	0.986
Fold 3	0.992
Fold 4	0.987
Fold 5	0.993
Fold 6	0.989
Fold 7	0.998
Fold 8	0.989
Fold 9	0.990
Fold 10	0.995
Average Accuracy	**0.991**

As shown in Table 6.5, it provides accuracy results for each fold in the 10-fold cross-validation, with an average accuracy of 0.991. This consistently high accuracy across folds indicates the model's stability and effectiveness in handling variations in data partitions.

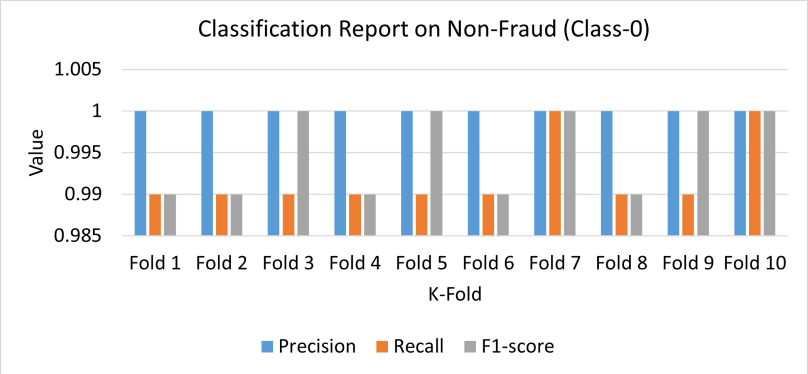

Figure 6.5: Classification report-based on K-fold for non-fraud (Class 0)

Displayed in the Figure 6.5, are precision, recall, and f1-score values for non-fraud detection (Class 0) across 10 k-folds. Precision and recall are consistently high, between 0.99 and 1.0, reflecting strong model accuracy. The f1-score remains stable and close to 1.0, confirming reliable performance for non-fraud cases.

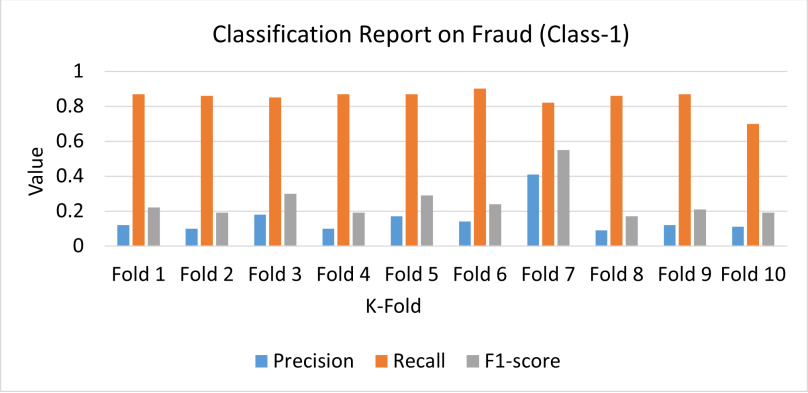

Figure 6.6: Classification report-based on K-fold for fraud (Class 1)

Figure 6.6, illustrates precision, recall, and f1-score for fraud detection (Class 1) over 10 k-folds. The recall values are high, ranging from 0.82 to 0.92, showing the model's effectiveness in identifying fraud cases. Precision varies between 0.1 and 0.41, indicating some false positives. The f1-score ranges from 0.19 to 0.55, highlighting the balance between precision and recall.

Below is the comparison of precision and recall for both classes based on the 10-fold cross-validation:

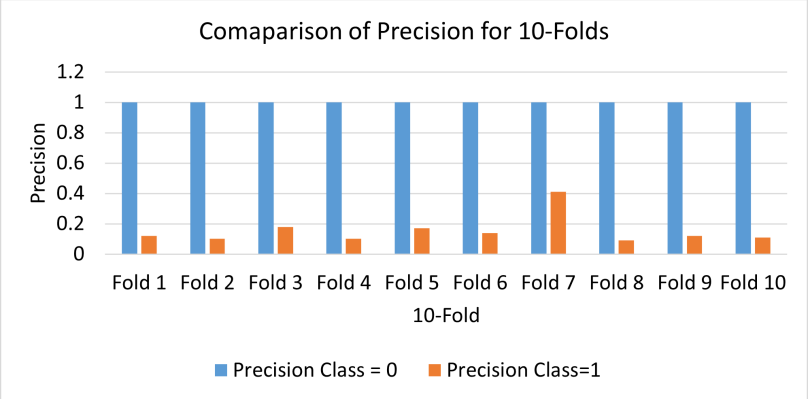

Figure 6.7: Comparison of precision for 10-fold

The given Figure 6.7, shows precision values for Class 0 (non-fraud) and Class 1 (fraud) across 10 folds. Class 0 consistently achieves a precision of 1.0 across all folds, demonstrating strong accuracy in detecting legitimate transactions. In contrast, Class 1 precision varies between 0.1 and 0.41, this range reflects the model's proactive effort to identify potential fraud cases, ensuring that most fraudulent transactions are identified.

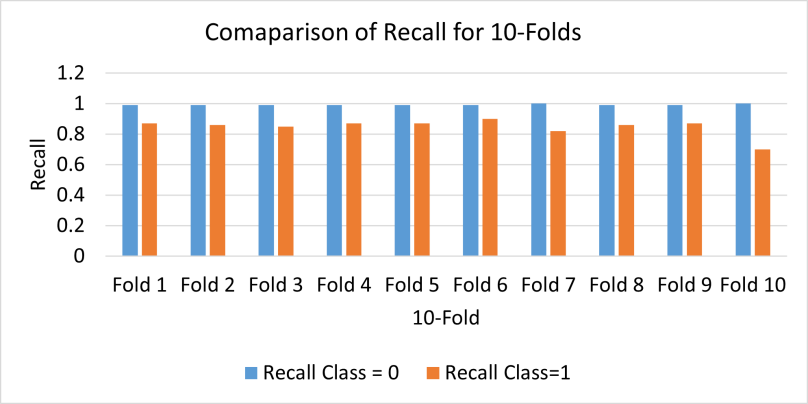

Figure 6.8: Comparison of recall for 10-fold

The Figure 6.8, of the bar diagram illustrates, the recall values for Class 0 (non-fraud) and Class 1 (fraud) throughout 10 folds. Class 0 exhibits higher recall rates, between

0.98 to 1.0, signifying efficient recognition of real transactions. The Class 1 recall ranges from 0.82 to 0.92.

The observed low precision and high recall for Class 1 (fraud) indicate that while the model is effective at identifying actual fraud cases (high recall, true positives), it tends to have more false positives, which lowers precision. As false negatives are more expensive than false positives, this trade-off is consistent in detecting as many fraudulent activities as feasible. The model's strong recall (few false negatives) reduces the probability of undetected fraud, making it useful despite its lower precision (higher false positives). Figures 4 and 5 demonstrate consistently strong recall across splits, alleviating concerns regarding performance in imbalanced datasets.

In a nutshell, precision and recall are key performance metrics in any model. Recall measures a model's ability to identify true positives. In imbalanced datasets like the one that has been used, recall often suffers, but the proposed model achieves higher recall than precision for Class 1. It is also successfully able to achieve higher accuracy as proposed.

Chapter 7

CONCLUSION

The evolution of credit cards and the corresponding increase in fraudulent activities have necessitated the development of sophisticated detection systems. This report explored the transition from traditional fraud detection techniques to advanced deep learning models, specifically ANNs. The study highlighted the historical evolution of credit cards, the methods fraudsters used, and the limitations of conventional approaches in addressing complex and emerging fraud scenarios. By leveraging the capabilities of DL techniques, this research aimed to enhance detection accuracy and improve the overall security of credit card transactions, contributing to the broader financial ecosystem's resilience against fraud.

The motivation for this study is that people are relying more and more on digital transactions and fraud strategies are getting smarter. Due to the flaws in rigid, rule-based systems, it is clear that we need real-time, flexible solutions that can identify complex fraud patterns accurately. To address these challenges, this project was initiated to build a strong model that is based on data and can learn from complicated transactional behaviors. The focus on deep MLPs technique makes it

possible to identify fraud with a higher percentage of accuracy.

An original contribution of this research is the implementation of a 12-layer deep MLP model specifically tailored to handle the complexities and high dimensionality inherent in credit card transaction data. The results from various train-test split ratios and 10-fold cross-validation approaches confirmed the robust performance of the model. The research introduced advanced techniques, such as class weighting and early stopping, to optimize the model's performance in imbalanced datasets. These techniques, along with feature engineering, ensured that the model was not only capable of learning from detailed transactional data but also robust against overfitting. By prioritizing fraudulent transactions during the training phase and dynamically adjusting training processes, the model demonstrated significant improvements in identifying fraud cases that traditional models often overlook.

The outcomes from different train-test split ratios and 10-fold cross-validation methods validated the model's strong performance. The model achieved an average accuracy of 99%, with precision and recall values for legitimate transactions consistently reaching 1.0 across all evaluations. For fraud detection, the model demonstrated recall rates between 82% and 92%, effectively capturing a substantial portion of fraudulent activities, and showcasing its sensitivity and reliability. These outcomes validate the model's ability to handle imbalanced data effectively and highlight its capacity to deliver consistent performance across different data splits. The experiments emphasize the model's scalability and suitability for real-world deployment, marking a significant step forward in developing efficient credit card fraud detection systems.

To summarize, the problem outlined is effectively tackled, demonstrating that the deep multilayer perceptron model significantly enhances the accuracy of fraud detection systems. The model's robustness and scalability were validated through various experiments, confirming its acceptability. Future work could focus on enhancing

detection efficiency by integrating additional features and leveraging larger external datasets. Exploration of alternative ANN architectures aims to enhance accuracy and adaptability. These efforts are essential to maintain and advance the system's effectiveness as fraud techniques evolve. Additionally, incorporating real-time data streams could improve the model's usage of emerging fraud patterns. Continued refinement of preprocessing techniques and parameter optimization will further strengthen performance in diverse operational environments.

REFERENCES

[1] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," Journal of Big Data, vol. 8, article no. 53, 2021.

[2] Y. Bai, "RELU-Function and Derived Function Review," SHS Web of Conferences, vol. 144, article no. 02006, 2022.

[3] B. Bakırarar and A. Elhan, "Class Weighting Technique to Deal with Imbalanced Class Problem in Machine Learning: Methodological Research," Turkiye Klinikleri Journal of Biostatistics, vol. 15, no. 1, pp. 19–29, 2023.

[4] S. Chakravorti, "Theory of Credit Card Networks: A Survey of the Literature," Review of Network Economics, vol. 2, no. 2, pp. 50–68, 2003.

[5] B. Ding, H. Qian, and J. Zhou, "Activation functions and their characteristics in deep neural networks," Chinese Control and Decision Conference. pp. 1836–1841, 2018.

[6] X. T. Dinh and H. Van Pham, "A Proposal of Deep Learning Model for Classifying User Interests on Social Networks," Proceedings of the 4th International Conference on Machine Learning and Soft Computing, pp. 10–14, 2020.

[7] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," Computer Science Review, vol. 40, no. C, article no. 100379, 2021.

[8] B. Du and T. V. Samuseva, "The Application of Deep Learning in the Financial Field," Proceedings of the 7th International Conference on Industrial and Business Engineering, Association for Computing Machinery, pp. 59–64, 2022.

[9] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," Neurocomputing, vol. 503, no. C, pp. 92–108, 2022.

[10] W. N. E, R. Hu, and S. Peng, "Deep Learning in Finance," Digital Finance, vol. 5, no. 5, pp- 1-2, 2023.

[11] F. Z. El Hlouli, J. Riffi, M. A. Mahraz, A. El Yahyaouy, and H. Tairi, "Credit Card Fraud Detection Based on Multilayer Perceptron and Extreme Learning Machine Architectures," International Conference on Intelligent Systems and Computer Vision, pp. 1–5, 2020.

[12] A. A. El Naby, E. El-Din Hemdan, and A. El-Sayed, "Deep Learning Approach for Credit Card Fraud Detection," International Conference on Electronic Engineering, pp. 1–5, 2021.

[13] S. L. Fulford and S. D. Schuh, "Credit cards, credit utilization, and consumption," Journal of Monetary Economics, vol. 148, article no. 103619, 2024.

[14] Federal Trade Commission. "As Nationwide Fraud Losses Top $10 Billion in 2023, FTC Steps Up Efforts to Protect the Public.", 2024.

[15] R. P. França, A. C. Borges Monteiro, R. Arthur, and Y. Iano, "An overview of deep learning in big data, image, and signal processing in the modern digital

age," Trends in Deep Learning Methodologies, Hybrid Computational Intelligence for Pattern Analysis series, pp. 63–87, 2021.

[16] Ghosh and Reilly, "Credit card fraud detection with a neural-network," Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences, vol. 3, pp. 621–630, 1994.

[17] E. Grossi and M. Buscema, "Introduction to artificial neural networks," European Journal of Gastroenterology & Hepatology, vol. 19, no. 12, pp. 1046–1054, 2008.

[18] M. Habibpour, H. Gharoun, M. Mehdipour, A. R. Tajally, H. Asgharnezhad, A. Shamsi, A. Khosravi, and S. Nahavandi, "Uncertainty-aware credit card fraud detection using deep learning," Engineering Applications of Artificial Intelligence, vol. 123, no. A, article no. 106248, 2023.

[19] Z. Hao, "Deep learning review and discussion of its future development," MATEC Web Conference, vol. 277, no. C, article no. 02035, 2019.

[20] H. He and E. A. Garcia, "Learning from imbalanced data," IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, pp. 1263–1284, 2009.

[21] T. Howley, M. G. Madden, M-L. O'Connell, and A. G. Ryder, "The effect of principal component analysis on machine learning accuracy with high dimensional spectral data," Knowledge-Based Systems, vol. 19, no. 5. pp. 363-370, 2006.

[22] J. Huang, J. Chai, and S. Cho, "Deep learning in finance and banking: A literature review and classification," Frontiers of Business Research in China, vol. 14, no. 1, article no. 13, 2020.

[23] M. Hutzenthaler, A. Jentzen, T. Kruse, and T. A. Nguyen, "A proof that rec-

tified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations," SN Partial Differential Equations and Applications, vol. 1, no. 2, article no. 10, 2020.

[24] S. Inyurt and A. Sekertekin, "Modeling and predicting seasonal ionospheric variations in Turkey using artificial neural network (ANN)," Astrophysics and Space Science, vol. 364, no. 4, article no. 62, 2019.

[25] B. Jabir and F. Noureddine, "Dropout, a basic and effective regularization method for a deep learning model: A case study," Indonesian Journal of Electrical Engineering and Computer Science, vol. 24, no. 2, pp- 1009-1016, 2021.

[26] S. S. Jahromi and R. Orús, "Variational tensor neural networks for deep learning," Scientific Reports, vol. 14, no. 1, article no. 19017, 2024.

[27] P. Jintanasonti, A. Dumrongsiri, and P. Tantiwattanakul, "Retailers' order decision with setup cost using machine learning," Proceedings of the 2024 8th International Conference on Machine Learning and Soft Computing, pp. 37–44, 2024.

[28] M. P. John and G. Murali, "Performance enhancement and comparative analysis for credit approval using XGBoost, SVM and multi-layer perceptron," 2022 IEEE Global Conference on Computing, Power and Communication Technologies, pp. 1–4, 2022.

[29] V. Josephine, A. P. Nirmala, and V. Alluri, "Impact of hidden dense layers in convolutional neural network to enhance performance of classification model," IOP Conference Series: Materials Science and Engineering, vol. 1131, no. 1, article no. 012007, 2021.

[30] L. Kabari and B. Nwamae, "Principal component analysis (PCA) - An effective tool in machine learning," International Journals of Advanced Research in Com-

puter Science and Software Engineering, vol, 9, no. 5, pp. 56–59, 2019.

[31] B. Kasasbeh, B. Al-dabaybah, and H. Ahmad, "Multilayer perceptron artificial neural networks-based model for credit card fraud detection," Indonesian Journal of Electrical Engineering and Computer Science, vol. 26, no. 1, pp. 362–373, 2022.

[32] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553. pp. 436–444, 2015.

[33] M. Madhiarasan, M. Louzazni, and J. S. Mandeep, "Analysis of artificial neural network: Architecture, types, and forecasting applications," Journal of Environmental Chemical Engineering, vol. 2022, no. 12, pp. 1-23, 2022.

[34] B. Mahesh, "Machine learning algorithms - A review," International Journal of Science and Research, vol. 9,no. 1, pp. 381–386, 2019.

[35] S. B. Maind and P. Wankar, "Research Paper on Basic of Artificial Neural Network," International Journal on Recent and Innovation Trends in Computing and Communication, vol. 2, no. 1, pp. 96–100, 2014.

[36] A. Mao, M. Mohri, and Y. Zhong, "Cross-entropy loss functions: theoretical analysis and applications," Proceedings of the 40th International Conference on Machine Learning, article no. 992, pp. 23803-23828, 2023.

[37] M. S. Mhatre, F. Siddiqui, M. Dongre, and P. M. Thakur, "A review paper on artificial neural network: A prediction technique," International Journal of Scientific & Engineering Research, vol. 6, no. 12, pp. 161–163, 2015.

[38] I. D. Mienye and Y. Sun, "A deep learning ensemble with data resampling for credit card fraud detection," IEEE Access, vol. 11, pp. 30628–30638, 2023.

[39] C. Mishra and D. Gupta, "Deep machine learning and neural networks: An overview," IAES International Journal of Artificial Intelligence, vol. 6, no. 2, p. 66-73, 2017.

[40] M. K. Mishra and R. Dash, "A comparative study of Chebyshev functional link artificial neural network, multi-layer perceptron and decision tree for credit card fraud detection," Proceedings of the 2014 International Conference on Information Technology, pp. 228–233. 2014.

[41] S. Narayan, "The generalized sigmoid activation function: Competitive supervised learning," Information Science, vol. 99, no. 1–2, pp. 69–82, 1997.

[42] J. Naskath, G. Sivakamasundari, and A. A. Siddiqua Begum, "A study on different deep learning algorithms used in deep neural nets: MLP SOM and DBN," Wireless Personal Communications, vol. 128, no. 4, pp. 2913–2936, 2022.

[43] S. Negi, S. K. Das, and R. Bodh, "Credit card fraud detection using deep and machine learning," International Conference on Applied Artificial Intelligence and Computing, pp. 455–461, 2022.

[44] C. C. Nwanwe, U. I. Duru, C. Anyadiegwu, and A. I. B. Ekejuba, "An artificial neural network visible mathematical model for real-time prediction of multiphase flowing bottom-hole pressure in wellbores," Petroleum Research, vol. 8, no. 3, pp. 370–385, 2023.

[45] O. Odeniyi, O. Oyinloye, and A. Thompson, "Fraud detection using multilayer perceptron and convolutional neural network," International Journal on Advances in Security, vol. 14, no. 1–2, pp. 1–11, 2021.

[46] U. Orji and E. Ukwandu, "Machine learning for an explainable cost prediction of medical insurance," Machine Learning with Applications, vol. 15, article no.

100516, 2024.

[47] T. D. Phong, H. N. Duong, H. T. Nguyen, N. T. Trong, V. H. Nguyen, T. V. Hoa, and V. Snasel, "Brain hemorrhage diagnosis by using deep learning," Proceedings of the 2017 International Conference on Machine Learning and Soft Computing, pp. 34–39, 2017.

[48] T. R. Pillai, I. A. T. Hashem, S. N. Brohi, S. Kaur, and M. Marjani, "Credit card fraud detection using deep learning technique," 2018 Fourth International Conference on Advances in Computing, Communication & Automation, pp. 1–6, 2018.

[49] A. G. Putrada and N. G. Ramadhan, "MDIASE-autoencoder: A novel anomaly detection method for increasing the performance of credit card fraud detection models," 2023 twenty-ninth International Conference on Telecommunications, pp. 1–6, 2023.

[50] A. Qayoom, M. Khuhro, K. Kumar, M. Waqas, U. Saeed, S. Rehman, Y. Wu, and S. Wang, "A novel approach for credit card fraud transaction detection using deep reinforcement learning scheme," PeerJ Computer Science, vol. 10, article no. e1998, 2024.

[51] S. Rahman and J. T. Yao, "Credit Card Fraud Detection with Deep Multilayer Perceptrons," Proceedings of the 9th International Conference on Machine Learning and Soft Computing, to appear, 2025.

[52] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," Journal of Computational Physics, vol. 378, no. C, pp. 686–707, 2019.

[53] M. S. Raju, M. Reena, P. P. Kumar, K. S. Kiran, and B. N. Kumar, "Optimizing Credit Card Fraud Detection Using Deep Learning By Smote-Enn Technique," International Refereed Journal of Engineering and Science, vol. 13, no. 2, pp. 190–200, 2024.

[54] Asha RB. and S. Kumar KR., "Credit card fraud detection using artificial neural network," Global Transitions Proceedings, 1st International Conference on Advances in Information, Computing and Trends in Data Engineering, vol. 2, no. 1, pp. 35–41, 2021.

[55] U. Ruby and V. Yendapalli, "Binary cross entropy with deep learning technique for image classification," International Journal of Advanced Trends in Computer Science and Engineering, vol. 9, no. 4, pp. 5393–5397, 2020.

[56] I. Sadgali, N. Sael, and F. Benabbou, "Fraud detection in credit card transaction using neural networks," Proceedings of the 4th International Conference on Smart City Applications, Association for Computing Machinery, article no. 95, pp. 1-4. 2019.

[57] I. Salehin and D.-K. Kang, "A Review on Dropout Regularization Approaches for Deep Neural Networks within the Scholarly Domain," Electronics, vol. 12, no. 14, article no. 3106, 2023.

[58] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," SN Computer Science, vol. 2, no. 3, article no. 160, 2021.

[59] I. H. Sarker, "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions," SN Computer Science, vol. 2, no. 6, article no. 420, 2021.

[60] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Networks, vol. 61, no. 3, pp. 85–117, 2015.

[61] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," The Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, 2014.

[62] E. Strelcenia and S. Prakoonwit, "Generating Synthetic Data for Credit Card Fraud Detection Using GANs," International Conference on Computers and Artificial Intelligence, pp. 42–47, 2022.

[63] S. Swathiga, J. Thanushya, and A. Fatima, "Deep Learning Algorithms Using Fraudulent Detection in Banking Datasets," International Research Journal of Education and Technology, vol. 6, no. 3, pp. 339–347, 2024.

[64] R. M. Terol, A. Reina, S. Ziaei, and D. Gil, "A Machine Learning Approach to Reduce Dimensional Space in Large Datasets," IEEE Access, vol. 8, pp. 148181–148192, 2020.

[65] O. Unogwu and Y. Filali, "Fraud Detection and Identification in Credit Card Based on Machine Learning Techniques," Wasit Journal of Computer and Mathematics Science, vol. 2, no. 3, pp. 15–21, 2023.

[66] H. Wan, Z. Lu, W. Qi, and Y. Chen, "Plant Disease Classification Using Deep Learning Methods," Proceedings of the 4th International Conference on Machine Learning and Soft Computing, pp. 5–9, 2020.

[67] J. T. Yao and C. L. Tan, "A case study on using neural networks to perform technical forecasting of forex," Neurocomputing, vol. 34, no. 1-4, pp. 79–98, 2000.

[68] J. T. Yao, C. L. Tan, and H. L. Poh, "Neural networks for technical analysis: A

study on KLCI," International Journal of Theoretical and Applied Finance, vol. 2, no. 2, pp. 221–241, 1999.

[69] J. T. Yao, N. Teng, H. L. Poh, and C. L. Tan, "Forecasting and analysis of marketing data using neural networks," Journal of information science and engineering, vol. 14, no. 4, pp. 523-545, 1998.