Jaypee Institute of Information Technology, Noida

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING AND INFORMATION TECHNOLOGY

PROJECT TITLE: FRAUD GUARD

Enrol No.	Name of Students
9921103003	Astha Mishra
9921103011	Yashi Ratnakar
9921103043	Divya Darshana

Course Name: Minor Project -2
Course Code: 15B19C1591
Program: B. Tech. CS&E
3rd Year 6th Sem

2023-2024

ACKNOWLEDGEMENT

I would like to place on record my deep sense of gratitude to Dr Devpriya Soni, Jaypee Institute of Information Technology, India for her generous guidance, help, and useful suggestions.

I express my sincere gratitude to the Dept. of CS&E and IT, JIIT Noida, for their stimulating guidance, continuous encouragement, and supervision throughout the course of the present work.

I also wish to extend my thanks to my other classmates for their insightful comments and constructive suggestions to improve the quality of this project work.

Name(s) of Students

Astha Mishra (9921103003)
Yashi Ratnakar (9921103011)
Divya Darshana (9921103043)

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and beliefs, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma from a university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: Jaypee Institute of Information Technology
Date: 27/4/2024

Name: Astha Mishra
Enrolment No.: 9921103003

Name: Yashi Ratnakar
Enrolment No.:9921103011

Name: Divya Darshana
Enrolment No.:9921103043

CERTIFICATE

This is to certify that the work titled “Fraud Guard” submitted by Astha Mishra(9921103003), Yashi Ratnakar(9921103011), and Divya Darshana(9921103043) of B.Tech of Jaypee Institute of Information Technology, Noida have been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of any other degree or diploma.

Signature of Supervisor

Dr Devpriya Soni

(Department of CS&E and IT, JIIT)

Date : 27/04/2024

ABSTRACT

In today's interconnected digital landscape, fraudsters are continually evolving their tactics to exploit vulnerabilities within systems. Traditional methods of fraud detection often fall short in detecting sophisticated fraudulent activities. However, with the advent of advanced technologies like machine learning and graph theory, there's a promising avenue for detecting and preventing fraud more effectively.

Fraud detection using graph nodes is a novel approach that capitalizes on the inherent relationships and connections between entities within a network. By representing these relationships as nodes and edges in a graph, analysts can gain valuable insights into the patterns and anomalies indicative of fraudulent behaviour. Leveraging machine learning models in conjunction with graph analysis further enhances the accuracy and efficiency of fraud detection systems.

Key Points:

- **Graph Representation:** In a fraud detection scenario, entities such as users, transactions, accounts, and devices can be represented as nodes in a graph, while the relationships between them (e.g., transactions between users, shared devices, etc.) are denoted by edges. This graphical representation provides a comprehensive view of the complex interactions within a network, enabling analysts to identify suspicious patterns that may indicate fraudulent activities.
- **Anomaly Detection:** Machine learning algorithms play a crucial role in identifying anomalies within the graph structure. By training models on historical data, these algorithms learn to distinguish between normal and abnormal behaviour, thereby flagging suspicious nodes or edges that deviate from expected patterns. Techniques such as clustering, classification, and anomaly scoring are commonly employed to detect fraudulent activities based on graph features.
- **Feature Engineering:** Effective feature engineering is essential for constructing informative input representations for machine learning models. Features derived from graph properties such as node centrality, community structure, and connectivity measures offer valuable insights into the network topology and help in distinguishing between legitimate and fraudulent entities. Additionally, auxiliary features such as transaction frequency, geographical location, and behavioural patterns further enrich the feature space and improve the model's predictive performance.
- **Graph-based Algorithms:** Graph-based algorithms, including graph neural networks (GNNs) and random walk-based methods, are specifically tailored to exploit the structural information encoded in graphs. GNNs, in particular, excel at capturing complex relationships and dependencies within the graph, making them well-suited for fraud detection tasks. By aggregating information from neighbouring nodes and iteratively refining node representations, GNNs can effectively uncover subtle patterns indicative of fraudulent behaviour.
- **Real-time Monitoring:** In dynamic environments where fraudsters constantly adapt their strategies, real-time monitoring are crucial for maintaining the effectiveness of fraud detection systems. By integrating real-time monitoring, systems pre-emptively thwart fraudulent activities, safeguarding against financial losses.

TABLE OF CONTENTS

List of Tables/Figures

1. Figures:

List of Abbreviations

UI	User Interface
DL	Deep Learning
UX	User Experience
API	Application Programming Interface
GCP	Google Cloud Platform
HTTPS	Hypertext Transfer Protocol Secure

INTRODUCTION

1.1 General Introduction

Mobile payment systems have become an integral part of our daily lives, offering convenience and efficiency in financial transactions. As the popularity of mobile payments continues to rise, so does the risk of fraudulent activities within these systems. Fraudsters are constantly adapting and devising new methods to exploit vulnerabilities in mobile payment platforms, making it imperative for businesses and financial institutions to implement robust fraud detection mechanisms. Mobile payment systems operate in a dynamic and interconnected environment, facilitating transactions through various channels such as mobile apps, contactless payments, and online platforms. This complexity increases the challenges associated with identifying and preventing fraudulent activities. Fraud detection in mobile payment systems involves the use of advanced technologies and algorithms to analyse patterns, detect anomalies, and safeguard users' financial transactions.

The dataset for a real-world Peer-to-Peer (P2P) platform has been anonymized, with unique 128-bit identifiers for user accounts and random UUIDs for credit cards, devices, and IP addresses in the graph schema. Each user node includes a Money Transfer Fraud indicator (1 for known fraud, 0 otherwise), determined by credit card chargeback events and manual review. Fraud is identified when a user has at least one chargeback and approximately 0.7% of user accounts are flagged as fraudulent. Chargebacks involve reversing electronic payments and triggering a dispute resolution process, typically for billing errors and unauthorized credit use.

1.2 Key Challenges in Mobile Payment Fraud

- **Rapidly Evolving Threat Landscape**: Fraudsters continuously develop sophisticated techniques, including phishing, account takeovers, and identity theft, to exploit weaknesses in mobile payment systems.
- **Increased Transaction Volume:** The sheer volume of mobile transactions makes it challenging to distinguish legitimate activities from fraudulent ones. Identifying patterns and anomalies in real time is crucial to prevent fraudulent transactions.
- **Diverse Transaction Channels:** Mobile payments can occur through various channels, such as apps, NFC technology, QR codes, and online platforms. Each channel presents unique challenges, requiring tailored fraud detection strategies.
- **User Experience Concerns:** Striking a balance between effective fraud detection and a seamless user experience is crucial. Implementing stringent security measures should not result in inconvenience or frustration for legitimate users.

1.3 Problem Statement

The rapid growth of mobile payment transactions has revolutionized the way individuals conduct financial transactions. However, this surge in mobile payments has also led to an increase in fraudulent activities, posing significant risks to users, financial institutions, and businesses alike.

Current methods are struggling to keep pace with the dynamic nature of fraud, exposing users, financial institutions, and businesses to financial losses and compromised data.. Balancing the need for enhanced security with user privacy is paramount, especially as mobile payment systems span cross-border transactions, navigating varied regulations and transaction patterns.

BACKGROUND STUDY

2.1 Overview

In this chapter, we delve into the foundational aspects relevant to our project, focusing on the evolution of detecting frauds in mobile payments and a comparative analysis of existing algorithms which can be applied to enhance the problem.

2.2 Research Paper

The research paper explores diverse algorithms and methodologies for fraud detection in payment transactions. It introduces a novel approach leveraging Neo4j graph databases, offering insights into the effectiveness of graph-based techniques in identifying fraudulent patterns. Additionally, it delves into the application of neural network models, elucidating their potential in enhancing fraud detection accuracy through deep learning techniques. These papers contribute to the advancement of fraud detection strategies, providing valuable insights for improving security measures in payment systems.

2.3 Solution Proposed

Our solution for fraud detection in mobile payment systems revolves around developing a robust microservice hosted on Google cloud platform. This service would seamlessly integrate an ensemble Deep Learning model to fortify fraud prevention. Leveraging a graph database and deep learning model based on graphical features to detect fraud in a dataset containing columns such as IP address, device, credit card, user ID, and time transactions, can solve several important fraud detection use cases.

We have designed a Payment Fraud Detection API for detecting frauds using Graph analytics emerges as a superior approach, providing a robust defense against modern fraud tactics. Further deploying this API on Cloud as a standalone microservice that can be integrated in various payment interfaces.

2.4 Roadmap

Our goal is to establish ourselves as the premier provider of secure and cost-effective mobile money solutions.

1. Dataset Analysis and Feature Engineering: Leveraging graph database capabilities, we meticulously analyze our data, incorporating graphical features to exploit the inherent advantages of network structures.

2. Model Experimentation and Selection: Through extensive experimentation, we explore a spectrum of deep learning techniques tailored to various fraud scenarios. Our aim is to select the most effective models for integration into our system.

3. Flask API Development: Upon identifying the optimal model, we develop a robust Flask API, ensuring seamless integration and flexibility for future enhancements.

4. Deployment on Google Cloud Platform: Harnessing the scalability of Google Cloud, we deploy our Flask API as a scalable microservice. Docker containerization streamlines deployment and management, ensuring efficient operation.

By meticulously following this roadmap, we are committed to delivering a comprehensive and adaptable solution for fraud detection, safeguarding the integrity of our users' financial transactions.

SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Purpose

- **Fraud Detection Mechanism:** Develop an effective and adaptive fraud detection system capable of identifying and preventing various forms of fraudulent activities within payment transactions.
- **User Experience Considerations:** Ensure that the implemented fraud detection measures do not compromise the user experience for legitimate transactions, striking a balance between security and convenience.
- **Real-time Detection:** Implement real-time detection capabilities to swiftly identify and respond to fraudulent transactions, minimizing potential financial losses and mitigating risks.
- **Scalability**: Design the system with scalability in mind, allowing for seamless integration with the increasing volume of mobile payment transactions as the platform grows.

3.2 Scope of the Project

The fraud guard application will include the following key features:

- Real-time monitoring of incoming transactions and user activities. Implementing automated alerting systems to notify relevant stakeholders when potential fraud is detected, enabling timely intervention.
- Creating visualizations and reports to communicate findings and insights from the graph analysis.
- Integrating the fraud detection system with existing payment platforms, financial systems, and security infrastructure. Ensuring seamless data exchange and interoperability to leverage existing data sources and workflows.
- Designing the system to handle large volumes of transaction data efficiently and scale as the volume grows. Optimizing algorithms and infrastructure to minimize computational resources and processing time while maintaining high accuracy.

3.3 Functional Requirements:

3.3.1 Data Collection and Processing:

- The system should be able to collect transaction data from mobile payment platforms, including transaction details such as timestamp, amount, user IDs, device information, location, and any other relevant metadata.
- It should preprocess the data to extract features and construct a transaction graph representing the relationships between transactions and users.

3.3.2 Graph Construction and Representation:

- The system should create a graph structure where nodes represent transactions, users, devices, or other entities, and edges represent relationships between them (e.g., shared device, shared IP address, social connections).
- It should incorporate various types of relationships and attributes into the graph, such as transaction amounts, frequencies, temporal patterns, and geographical proximity.

3.3.3 Graph Analysis and Algorithms:

- Implement graph algorithms for fraud detection, such as centrality analysis, community detection, anomaly detection, and social network analysis.
- Apply techniques like graph embedding to represent nodes in a lower-dimensional space while preserving structural relationships.
- Utilize temporal analysis to detect changes in transaction patterns over time.

3.3.4 Anomaly Detection and Alerting:

- Implement algorithms to detect anomalous patterns or behaviors within the transaction graph, indicating potential fraud.
- Define thresholds or rules for triggering alerts based on the detected anomalies.
- Generate alerts or notifications to notify administrators or users about suspicious activities in real-time.

3.3.5 Integration with Deep Learning Models:

- Integrate graph-based features and embeddings into deep learning models for fraud detection.
- Train and deploy deep learning models to identify fraudulent transactions based on graph features and other relevant data.

3.3.6 User Interface and Visualization:

- Develop a user interface to presents the probability of a user's transaction being fraudulent in three distinct scenarios.
- Administrators and analysts can compare the probability of fraud across all three cases, aiding in efficient risk assessment for each transaction.

3.4 Non-Functional Requirements:

3.4.1 Performance:

- Response Time: The system should provide real-time or near-real-time fraud detection and alerting to minimize delays in identifying suspicious activities.
- Throughput: It should be capable of processing a high volume of transactions efficiently to keep up with the pace of mobile payment transactions.
- Scalability: The system should scale horizontally to accommodate increasing data volumes and user loads without sacrificing performance.

3.4.2 Scalability:

- Horizontal Scalability: The system architecture should support horizontal scaling to handle growing transaction volumes and user bases by adding more resources or distributed components.
- Vertical Scalability: It should also support vertical scalability to accommodate increased processing power and storage capacity within individual components.

3.4.3 Security:

- Data Encryption: Transaction data, user information, and sensitive system data should be encrypted both in transit and at rest to protect against unauthorized access.
- Access Control: Role-based access control mechanisms should be implemented to restrict access to sensitive functionalities and data based on user roles and permissions.
- Audit Trails: The system should maintain detailed audit logs of user activities, system events, and data access for compliance, monitoring, and forensic purpose**.**

3.4.4 Usability:

- User Interface Design: The user interface should be intuitive, user-friendly, and visually appealing, enabling administrators and analysts to easily navigate and interact with the system.
- Customization: Provide options for customization and configuration to tailor the system's behavior, alerts, and visualizations according to specific user preferences and requirements.
- Documentation and Training: Comprehensive documentation, tutorials, and training materials should be provided to assist users in understanding the system's functionality and capabilities.

3.4.5 Reliability:

- Fault Tolerance: The system should be resilient to failures and errors, with built-in mechanisms for fault detection, recovery, and graceful degradation.
- High Availability: Ensure high availability of the system to minimize downtime and disruptions, with redundant components and failover mechanisms in place.

3.4.5 Compliance:

- Regulatory Compliance: The system should comply with relevant regulations and standards governing data privacy, security, and financial transactions, such as GDPR, PCI DSS and PSD2.
- Ethical Considerations: Ensure that the system's design and operation adhere to ethical guidelines and principles, respecting user privacy and rights while combating fraud effectively.

3.4.6 Performance Metrics:

- Monitoring and Reporting: Implement monitoring tools to track system performance, resource utilization, and key metrics such as fraud detection accuracy, false positive rates, and alert response times.
- Reporting: Generate regular reports and dashboards to provide insights into system performance, fraud trends, and operational metrics for stakeholders and decision-makers.

3.5. Future Enhancements:

- Implement adaptive learning for evolving fraud patterns.
- Monitor and update model in real-time for improved accuracy.
- Introduce confirmation message feature to handle cases of false positives, ensuring accurate fraud detection while minimizing disruptions for legitimate transactions.

DETAILED DESIGN

4.1 Introduction

We integrate various deep learning models and leverage graph-based feature engineering using Neo4j. Through detailed technical specifications and design elements, we aim to provide guidance for developing a robust and user-friendly solution.

4.2 Deep Learning :

4.2.1 Model Selection and Architecture

- The model architecture comprises a sequential neural network with dense layers.
- The sequential nature of the neural network allows for a straightforward stacking of layers, making it suitable for many classification tasks..
- Dense layers, also known as fully connected layers, are used to capture intricate relationships between features in the data.

4.2.2 Hyperparameter Tuning

- We conduct extensive hyperparameter tuning using techniques such as grid search and random search.
- Parameters related to learning rate, batch size, and dropout rates are optimized to enhance model performance.
- The impact of hyperparameter tuning on metrics such as precision, recall, and F1-score is carefully evaluated.

4.2.3 Data Preprocessing

- Prior to model training, we preprocess the data to handle missing values and scale features.
- Categorical variables are encoded using techniques such as one-hot encoding or label encoding.
- Data augmentation techniques such as synthetic minority oversampling technique (SMOTE) are applied to address class imbalance issues.

4.3 Database Design

4.3.1 Data Transformation & Feature Engineering Using Neo4j's Graph Data Science

- Our database architecture leverages Neo4j's graph data science capabilities.
- Graph algorithms such as PageRank and community detection are utilized for feature engineering.
- Contextual embeddings generated by Neo4j are stored alongside user interactions to enhance fraud detection accuracy..

4.4 Deployment

- Deployment of the application and trained models is facilitated through Docker containers.
- We explore cloud deployment options such as AWS or Google Cloud Platform for scalability and reliability.
- Continuous integration and deployment (CI/CD) pipelines are set up to automate the deployment process.

IMPLEMENTATION

5.1 Introduction

The implementation phase of our fraud detection system marks a crucial step in realizing our objective of enhancing security and trust in financial transactions. In this section, we outline the practical aspects of translating our conceptual framework into a functional and effective solution. By detailing the technologies employed and the methodologies adopted, we aim to provide a comprehensive overview of the implementation process..

5.2 Technologies Used

5.2.1 Backend:

- Python Framework
- Flask API

5.2.2 Database Used:

- Neo4j

5.2.3 Tools:

- GCP
- Docker

5.3 Implementation Workflow:

5.3.1. Data Processing and Feature Engineering Using Neo4j:

- Utilized Neo4j's graph database and graph algorithms to perform data processing and feature engineering.

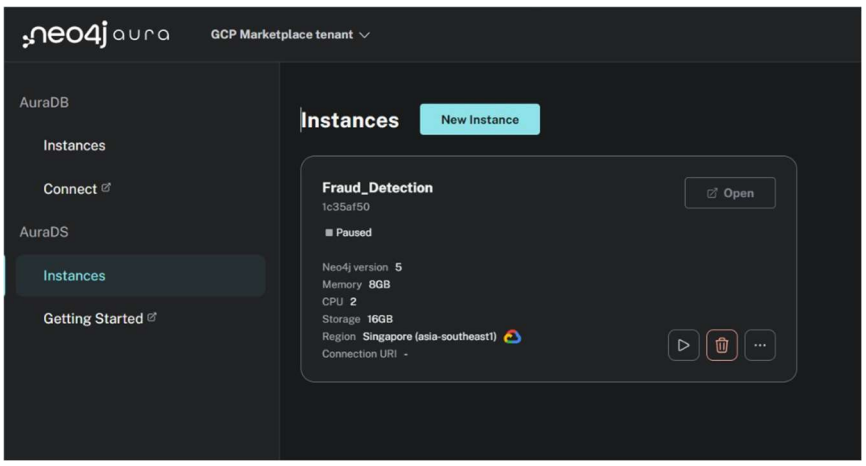

Figure 5.a. Graph database interface

	nodeLabel	nodeCount
0	User	33732
1	Device	51451
2	Card	118818
3	IP	585855

	relationshipType	relationshipCount
0	USED	55026
1	HAS_IP	1488949
2	HAS_CC	128066
3	REFERRED	1870
4	P2P	102832

Figure 5.b. Graph Node & Relationship Count

Figure 5.c. Existing Features Correlation

- None of the features correlate with the fraud risk label. As one would imagine, the number of transactions correlates with the total amount sent or received. Hence we leveraged graph algorithms to retrieve new features from the transactional data, enhancing the dataset for model training.

```
: df = gds.run_cypher('''
    MATCH(u:User)
    RETURN u.guid AS guid,
        u.wccId AS wccId,
        u.fraudRisk AS fraudRisk,
        u.fraudMoneyTransfer AS fraudMoneyTransfer,
        u.sharedIdsDegree AS sharedIdsDegree,
        u.p2pSharedCardPageRank AS p2pSharedCardPageRank,
        u.p2pSentPageRank AS p2pSentPageRank,
        u.p2pReceivedWeightedPageRank AS p2pReceivedWeightedPageRank,
        u.p2pReceivedWeightedDegree AS p2pReceivedWeightedDegree,
        u.ipDegree AS ipDegree,
        u.cardDegree AS cardDegree,
        u.deviceDegree AS deviceDegree,
        u.communitySize AS communitySize,
        u.partOfCommunity AS partOfCommunity
''')
df
```

Figure 5.d. Data frame of Extracted Features

5.3.2 Supervised and Unsupervised Model Training:

- Initially trained supervised and unsupervised models for fraud detection using the processed data.
- Evaluated model performance in terms of recall and precision, identifying limitations in achieving optimal results.

PERFORMANCE EVALUATION	Accuracy	Precision	Recall	F1 Score	AUC
Random Forest Classifier	0.85	0.03	0.89	0.06	0.95
Logistic Regression	0.73	0.02	0.93	0.045	0.90
KNN	0.99	0.66	0.04	0.083	0.63
SVM	0.86	0.03	0.64	0.05	0.86
CBLOF	0.99	0.00	0.00	0.00	0.80
Angle Based Outlier Detection	0.99	0.0	0.00	0.00	0.0
One-Class SVM	0.94	0.03	0.24	0.05	0.59
Unsupervised KNN	0.89	0.03	0.4	0.05	0.6
MInimum Co-variance Determinant	0.98	0.05	0.08	0.06	0.5
Isolation Forest	0.48	0.00	0.00	0.00	0.24
LODA	0.88	0.004	0.06	0.007	0.48
Auto-Encoder	0.98	0.00	0.00	0.00	0.29
Variational Auto-Encoder	0.94	0.03	0.24	0.05	0.59
Histogram Based Outlier Detection	0.90	0.02	0.40	0.05	

Figure 5.e. Performance evaluation of different models

5.3.3 Deep Learning Model Selection and Hyperparameter Tuning:

- Explored various deep learning architectures to improve model performance.
- Conducted hyperparameter tuning to optimize the models for different evaluation metrics, such as recall and precision.

5.3.4 Flask API Development:

- Developed a Flask API to serve the selected deep learning models.
- Integrated the models into the API to provide predictions on the probability of fraud for each user in three distinct cases.

5.3.5 Deployment on Cloud Using Docker:

- Deployed the Flask API and associated models on a cloud platform, such as Google Cloud Platform
- Containerized the application using Docker to ensure efficient deployment and scalability.

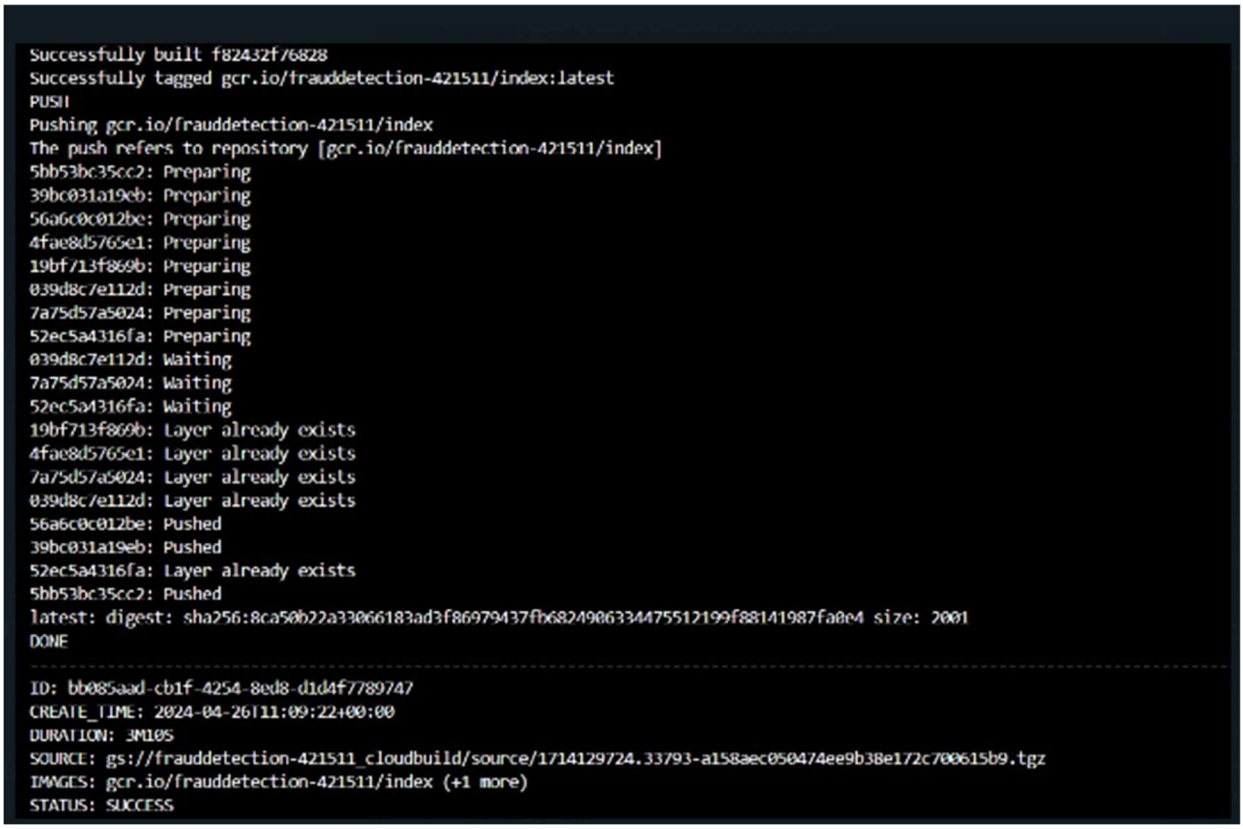

Figure 5.f. Google Cloud built submit

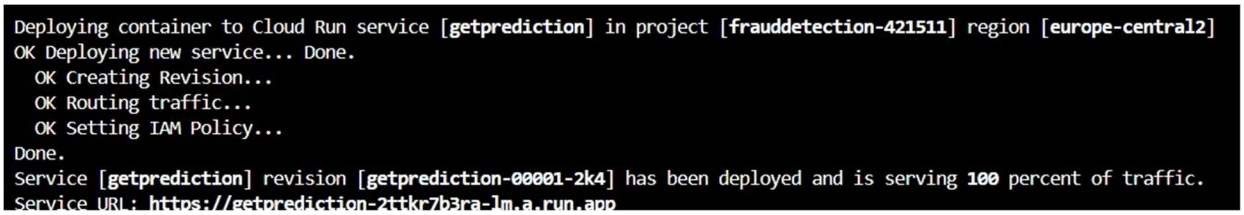

Figure 5.g. Google cloud run deploy

EXPERIMENTAL RESULTS AND ANALYSIS

6.1 Introduction:

This chapter presents a thorough examination of the experimental results obtained from the implementation of Fraud Detection application. It aims to display various functionalities, of the platform.

6.2 Model Summaries:

```
loaded_model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	1,408
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 1)	65

Total params: 9,731 (38.02 KB)

Trainable params: 9,729 (38.00 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 2 (12.00 B)

Figure 6.a. General Model Summary

```
loaded_model.summary()
```

Model: "sequential_5"

Layer (type)	Output Shape	Param #
dense_15 (Dense)	(None, 288)	3,168
dropout_10 (Dropout)	(None, 288)	0
dense_16 (Dense)	(None, 32)	9,248
dropout_11 (Dropout)	(None, 32)	0
dense_17 (Dense)	(None, 1)	33

Total params: 12,451 (48.64 KB)

Trainable params: 12,449 (48.63 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 2 (12.00 B)

Figure 6.b. Recall Focused Model Summary

```
loaded_model.summary()
```

Model: "sequential_5"

Layer (type)	Output Shape	Param #
dense_15 (Dense)	(None, 32)	352
dropout_10 (Dropout)	(None, 32)	0
dense_16 (Dense)	(None, 96)	3,168
dropout_11 (Dropout)	(None, 96)	0
dense_17 (Dense)	(None, 1)	97

Total params: 3,619 (14.14 KB)

Trainable params: 3,617 (14.13 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 2 (12.00 B)

Figure 6.c. Precision Focused Model Summary

6.3 Performance Analysis:

	Model	Precision	Recall	Accuracy	F1 Score	ROC AUC
0	General	0.783465	0.987513	0.856813	0.873734	0.912045
1	Recall	0.785672	0.992865	0.860541	0.877200	0.926377
2	Precision	0.923220	0.368218	0.667686	0.526461	0.935037

Figure 6.d. Performance Metrics of Models

6.4 Testing Flask Api on live URL: https://getprediction-2ttkr7b3ra-lm.a.run.app/

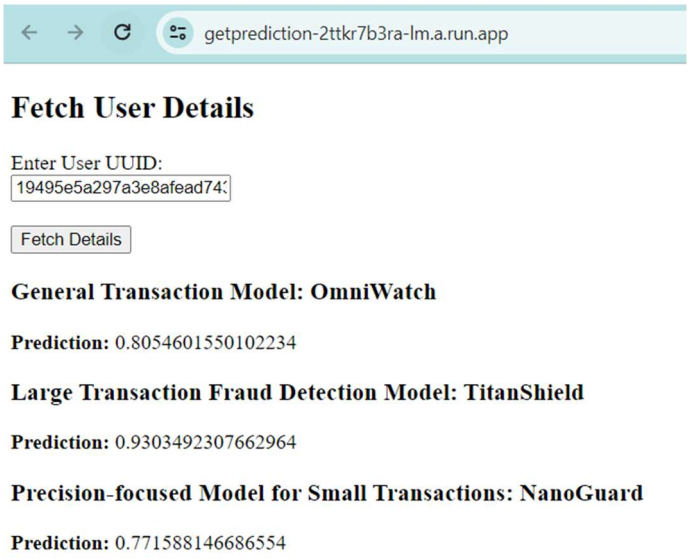

Figure 6.e. Testing of Fraud User on application programming interface

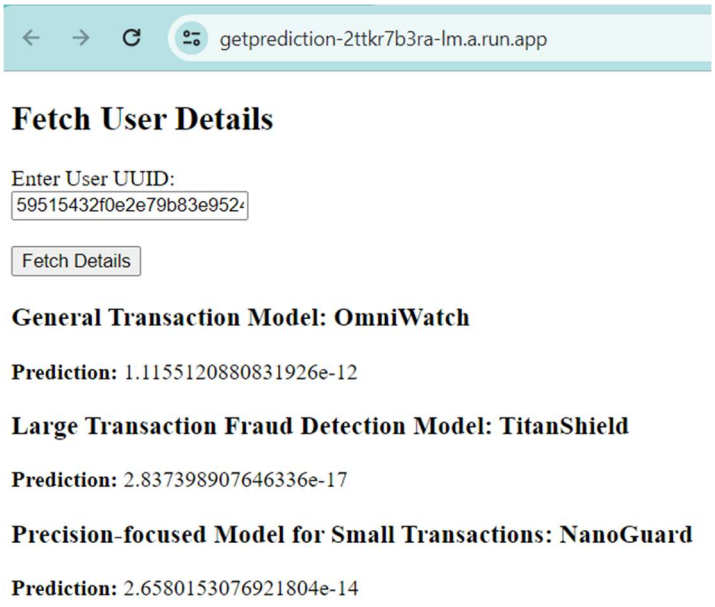

Figure 6.f. Testing of Non-Fraud User on Application Programming Interface

6.5 Experimental Analysis

Our deployed api implements three different models on transaction ID to cater to different kinds of payment transactions:

a) General Transactional Model: OmniWatch works for payment gateways handling various transaction types, from big money transfers to small daily purchases. It aims to strike a balance between accuracy and coverage, ensuring reliable performance across different transaction sizes.

b) Large Transaction Fraud Detection Model: TitanShield is focused to address those payment gateways that handle large sum transactions. Such kinds of payments need to be risk proof and hence there should be very less instances of false negatives. Thus, we implemented a recall focused hypertuned model to address the issue. Every transaction can then be analysed using neo4j data analysis techniques as discussed in the Implementation section (3.1) in case of false positive instances to resolve the error and can be used for reinforcement learning to make the model more efficient..

c) Precision-focused Model For Small Transactions: NanoGuard is focused to address those payment gateways that handle a lot of small day to day transactions. Here, priority must be given to precision, as frequent flags on small transactions can lead to loss of customer base. Also because of the sheer number of transactions, it is not possible to analyse each flagged issue. Thus, we implemented a precision focused hypertuned model to address the issue that gives priority to reducing false positives.

Depending upon the business need, one or a combination of these models can be integrated in payment gateways for safer payments.

CONCLUSION

In this project, we harnessed the potential of the Neo4j database to dissect user activity across multiple IP addresses and devices, focusing on detecting fraudulent behaviour within complex networks like rings and nets associated with peer-to-peer transactions.

Initially, we explored a range of supervised and unsupervised learning models to uncover fraudulent patterns. However, we encountered significant challenges stemming from data imbalance and the limited effectiveness of outlier detection methods in capturing diverse patterns simultaneously.

To address these challenges, we transitioned to deep learning models known for their capability to handle heterogeneous data. Through meticulous hyperparameter tuning, we developed three distinct models: a general model, optimized for overall performance; a recall-focused model, emphasizing detection of fraudulent cases to minimize false negatives; and a precision-focused model, prioritizing precise identification of fraud cases to reduce false positives.

To ensure scalability and integration into payment applications, we deployed a Flask API on Google Cloud. This deployment not only enables scalability but also enhances accessibility and usability of our fraud detection solution.

In conclusion, our technical journey underscores the importance of adaptability and innovation in addressing real-world challenges like fraud detection. By leveraging Neo4j, deep learning techniques, and cloud deployment, we've established a robust framework for combating fraudulent activities in payment systems, paving the way for a more secure digital landscape.

FUTURE SCOPE

1. **Reinforcement Learning Exploration**: Investigate the integration of reinforcement learning to adaptively enhance fraud detection strategies based on dynamic feedback loops.

2. **Continuous Integration and Deployment (CI/CD)**: Implement CI/CD pipelines to streamline the deployment of model updates, ensuring rapid and efficient delivery of enhancements.

3. **Optimization for Speed**: Explore techniques such as model quantization and pruning to optimize model inference speed, facilitating faster predictions without sacrificing accuracy.

4. **Real-Time Data Visualisation**: Integrate real-time data streams dashboards for immediate detection of fraudulent activities, enhancing the timeliness and effectiveness of our system.

5. **Enhanced Explainability**: Improve model explainability and interpretability to foster trust and transparency in our fraud detection system, enabling stakeholders to understand and validate predictions.

6. **Advanced Threat Intelligence Integration**: Incorporate advanced threat intelligence sources to enrich our fraud detection system with contextual information, enabling proactive mitigation of emerging threats.

7. **Federated Learning Exploration**: Explore federated learning techniques to collaboratively train models across distributed data sources while preserving data privacy and security.

8. **Continuous Monitoring and Evaluation**: Establish mechanisms for continuous monitoring and re-evaluation of model performance to ensure ongoing effectiveness and adaptability to evolving fraud patterns.

REFERENCES

Research Papers

[1] Ayushi Patil a, Shreya Mahajan a, Jinal Menpara a, Shivali Wagle a, Preksha Pareek a, Ketan Kotecha, **Enhancing fraud detection in banking by integration of graph databases with machine learning.**

[2] Ebenezer Esenogho; Ibomoiye Domor Mienye; Theo G. Swart; Kehinde Aruleba; George Obaido. **A Neural Network Ensemble With Feature Engineering for Improved Credit Card Fraud Detection.**

Journal Article

[3] **Using Neo4j Graph Data Science in Python to Improve Machine Learning Models** https://medium.com/neo4j/using-neo4j-graph-data-science-in-python-to-improve-machine-learning-models-c55a4e15f530

[4] **Hyperparameters in Deep Learning: Balancing the Art and Science of Model Tuning** https://medium.com/the-modern-scientist/hyperparameters-in-deep-learning-balancing-the-art-and-science-of-model-tuning-1f6281c44233

Online Resources

[5] **Flask Tutorial**, https://www.tutorialspoint.com/flask/index.htm

[6] **How To Deploy ML Models With Google Cloud Run** https://youtu.be/vieoHqt7pxo?si=2aG70fFGhv_p6Aja

[7] **Introduction to Deep Learning**, https://www.geeksforgeeks.org/introduction-deep-learning/

[8] **Cloud Run documentation**, https://cloud.google.com/run/docs

[9] **Neo4j AuraDS**, https://neo4j.com/docs/aura/