Data Mining To Detect Illegal Transactions In The Banking System

1st Tuan Nguyen Duc, 2nd Tran Xuan Tuan Anh, and ThuyDuong Thi Vu*
FPT University, Ho Chi Minh Campus, Vietnam
tuanndse173318@fpt.edu.vn, tuananhvn0203@gmail.com, duongvtt9@fe.edu.vn

Abstract

Financial fraud and illicit transactions pose significant challenges to the banking sector, necessitating the development of robust anomaly detection techniques. This study explores the application of data mining and machine learning methodologies to identify unusual transactions within largescale financial datasets. Our approach leverages Graph Neural Networks (GNNs) and anomaly detection algorithms to analyze transaction patterns, detect deviations from normal behavior, and flag potentially fraudulent activities. By constructing a transaction graph, we capture the relationships between entities and exploit network structures to enhance the detection accuracy. Additionally, we employ feature engineering and unsupervised learning techniques to improve model robustness in identifying rare yet critical anomalies. The proposed approach is evaluated on realworld financial datasets, demonstrating its effectiveness in reducing false positives while maintaining high detection rates. This research contributes to the advancement of intelligent financial monitoring systems, providing a scalable and efficient solution for detecting illicit transactions in dynamic banking environments.

Index Terms

Graph Neural Networks, Machine Learning, Financial Transactions, Unsupervised Learning, Supervised Learning.

I. Introduction

This paper presents a comprehensive approach to detect fraud in banking transactions by combining network analysis and advanced machine learning techniques. In the context of the rapidly developing digital economy, electronic banking transactions have become an integral part of everyday life. However, this convenience also comes with significant security challenges, particularly the increasing sophistication of financial fraud. Fraudulent activities not only cause financial losses to organizations and individuals, but also erode trust in the banking system. Therefore, the development of effective methods to detect and prevent transaction fraud is crucial. Traditional fraud detection methods often rely on manual rules or simple statistical models, which may be insufficient to handle large volumes of data and complex fraud patterns. In recent years, machine learning and network analysis have emerged as powerful tools to address this problem. Machine learning algorithms have the ability to learn from historical data and detect anomalous patterns, while network analysis helps uncover hidden relationships between accounts and transactions. In this paper, we propose a comprehensive approach to detect fraud in banking transactions, combining the strengths of machine learning and network analysis. We utilize big data processing algorithms such as Apache Spark to efficiently preprocess large volumes of transaction data. Subsequently, we apply the Kmeans algorithm and combine supervised learning with unsupervised learning to identify anomalous behaviors in transactions. Furthermore, we analyze the transaction network using the PageRank algorithm and Graph Neural Networks (GNNs) to identify intermediary accounts and suspicious account groups. Experimental results demonstrate that our proposed method achieves high reliability in detecting fraudulent transactions, showcasing its potential applicability in realworld financial fraud detection systems. The main contributions of this paper include: (1) an efficient data preprocessing pipeline for large volumes of transaction data, (2) a combined approach of machine learning and network analysis for fraud detection, and (3) a comprehensive experimental evaluation of the proposed method's performance on a realworld banking transaction dataset.

II. RELATED WORK

In recent years, the field of bank transaction fraud detection has attracted significant attention from both academia and industry. Various methods and techniques have been proposed and applied to address the increasing sophisti-

cation of fraudulent activities. This section will provide an overview of existing research, focusing on the use of machine learning and network analysis in the context of financial fraud detection.

A. Traditional Fraud Detection Methods

Traditional fraud detection methods often rely on predefined business rules or simple statistical models. Rulebased systems typically require experts to identify suspicious transaction patterns based on experience and knowledge of known fraudulent behaviors. While easy to understand and implement, these methods can be ineffective in detecting new and complex forms of fraud [1]. Statistical models, such as logistic regression or time series analysis, can help identify abnormal transactions based on historical data [2]. However, they often struggle to handle nonlinear data and complex relationships between transactions.

B. Application of Machine Learning in Fraud Detection

The development of machine learning has provided powerful tools for detecting transaction fraud. Machine learning algorithms have the ability to learn from large amounts of data and automatically detect abnormal patterns that traditional methods may miss [3]. Supervised learning techniques, such as decision trees, support vector machines (SVM), and neural networks, have been widely used to build classification models capable of distinguishing between legitimate and fraudulent transactions based on labeled historical data [5]. However, collecting and labeling a large amount of fraudulent transaction data can be costly and difficult.

To address this issue, unsupervised learning techniques, such as the Kmeans algorithm, DBSCAN, and Isolation Forest, have been applied to detect abnormal transactions without the need for prior data labeling [4]. These methods seek to identify transactions that differ significantly from the majority of normal transactions.

C. Network Analysis in Fraud Detection

Network analysis has emerged as an effective method for uncovering hidden relationships between entities in a transaction system, such as accounts and transactions. By modeling transactions as edges in a graph and accounts as nodes, network analysis can help identify fraudulent communities, intermediary accounts, and suspicious transaction patterns based on the network structure [7]. Algorithms such as PageRank, initially developed for ranking web pages [6], have been adapted to identify influential or central accounts in the transaction network, which may be involved in fraudulent activities. Furthermore, Graph Neural Networks (GNNs) have shown powerful capabilities in learning node and edge representations in graphs, allowing the detection of complex fraud patterns based both on node attributes and network structure [8].

D. Combining Machine Learning and Network Analysis

Recently, there has been a growing interest in combining the power of machine learning and network analysis to improve fraud detection capabilities. These combined methods leverage the ability of machine learning to learn from data and the ability of network analysis to capture complex relationships between entities [10]. For example, network characteristics, such as centrality and clustering coefficient, can be used as input for machine learning models [9]. In contrast, machine learning models can be used to label or weight nodes or edges in the network, facilitating deeper network analysis.

E. Gaps in Current Research and Motivation of the Proposed Method

Although significant research has been conducted in the field of banking transaction fraud detection, there are still several challenges and gaps to address. Many existing methods focus on detecting specific types of fraud or may struggle with processing large volumes of data [11] and complex fraud patterns. Furthermore, the effective integration of machine learning and network analysis remains a developing area of research.

The method proposed in this paper aims to address these limitations by providing a comprehensive approach that combines big data processing techniques [12], supervised and unsupervised machine learning, as well as network analysis based on the PageRank algorithm and Graph Neural Networks. Using the synergistic power of these techniques, our aim is to develop a more robust and effective fraud detection system capable of dealing with the diversity and complexity of modern financial fraud. Our method specifically focuses on efficiently processing large volumes of transaction data and exploring the hidden relationships between accounts and transactions to identify potential fraudulent activities.

III. METHODOLOGY

This section presents a detailed overview of the systematic approach we employed to detect fraudulent activities in banking transactions. By integrating network analysis with advanced machine learning techniques, our methodology aims to enhance the accuracy and efficiency of fraud detection in largescale financial datasets. The approach is structured into multiple stages, each designed to identify hidden patterns, detect anomalies, and improve the interpretability of fraudulent behaviors. Through a combination of datadriven analysis and computational techniques, our study provides a robust framework for addressing the growing challenges of financial fraud in modern banking systems.

A. Data Description

In this study, we utilize a financial transaction dataset, which, for the purpose of this research, was generated randomly to simulate realworld banking transactions. The dataset comprises approximately 20 million records, totaling 2GB in size, with each record containing key attributes such as a unique transaction identifier (UUID), sender and receiver account identifiers (randomly generated integers), transaction timestamp, transaction amount, transaction location, and transaction type (e.g., transfer, withdrawal, payment). To efficiently manage and prepare this substantial dataset, we employed big data processing frameworks, specifically and Apache Spark, for preprocessing. This phase involved data cleaning procedures to address missing values and eliminate duplicate entries. Furthermore, data transformation techniques were applied, including timestamp standardization to a uniform format and encoding of categorical variables. The transactions were also organized chronologically to facilitate temporal analysis. To enhance the capabilities of our anomaly detection models, we performed feature engineering, deriving insightful features such as the total transaction value per account, transaction frequency within defined time intervals, and the count of distinct counterpart accounts for each account.

B. Data Preprocessing for Transaction Data

1) Dataset Characteristics: For this research, we utilize a synthetically generated financial transaction dataset, created to simulate realworld banking operations while ensuring data privacy. The dataset, approximately 2GB in size and stored in CSV format, initially contains around 20 million transaction records *(as processed by the Spark pipeline described below; the initial generation aimed for approximately 20 million records, with variations possible in synthetic generation)*. Each record includes several key attributes:

- `transaction_id`: A unique identifier (UUID format) for each transaction.
- `sender_id`: An identifier representing the sending account (handled as `StringType` in processing, though potentially originating from generated numerical sequences).
- `receiver_id`: An identifier representing the receiving account (handled as `StringType`).
- `timestamp`: The transaction date and time, initially as a string (”yyyyMMdd HH:mm:ss”).
- `amount`: The transaction value, initially as a string.
  - `location`: The geographical location associated with the transaction (e.g., *”New York”*, *”Ho Chi Minh City”*, *”Hanoi”*, *”London”*).
  - `transaction_type`: The category of the transaction. Common values include: *”Payment”*, *”Deposit”*, *”Withdrawal”*, *”Transfer”*

The substantial volume of this dataset necessitated the use of a distributed computing framework, specifically Apache Spark, for efficient management and preprocessing.

2) Data Preprocessing Methodology with Apache Spark: To prepare the raw data for analysis, a rigorous preprocessing pipeline was implemented using Apache Spark (via PySpark). This pipeline focused on cleaning, validation, and standardization, ensuring scalability for the 2GB dataset. The key steps included:

1) **Schema Definition and Loading:** Data was loaded using a predefined schema (`StructType`) specifying initial `StringType` for most fields to handle potential inconsistencies, read via `spark.read.csv`.
2) **Type Conversion:** Critical fields were converted: `timestamp` strings to `TimestampType` (using `to_timestamp` with format ”yyyyMMdd HH:mm:ss”) and `amount` strings to `DoubleType` (using `cast`). Errors during conversion were handled by setting invalid entries to `NULL`.

3) **Missing/Invalid Data Removal:** Records with `NULL` values in essential columns (`transaction_id`, converted `timestamp`, converted `amount`) were discarded using `nadrop`.
4) **String Cleaning:** `location` and `transaction_type` fields were standardized using `lower(trim())` for case and whitespace consistency.
5) **Data Validation:** Transactions were validated, including filtering to ensure nonnegative `amount` values (`filter(col("amount_num") >= 0)`).
6) **Deduplication:** Duplicate records, based on `transaction_id`, were removed using `dropDuplicates`, reducing the record count from 20,000,000 to 19,600,000 in this execution.
7) **Final Selection:** Relevant, cleaned, and correctly typed columns were selected for the final preprocessed dataset, renaming converted columns (`timestamp`, `amount`).

This pipeline established a clean and consistent data foundation. Note that further transformations, such as the encoding of categorical variables (`location`, `transaction_type`) into numerical representations, might be performed in subsequent stages depending on the requirements of specific analytical models (e.g., machine learning algorithms). The conversion to `TimestampType` facilitates temporal analysis.

3) Preprocessing Results and Foundation for Feature Engineering: The described Apache Spark pipeline successfully processed the raw data, yielding a cleaned and validated dataset containing 19,600,000 unique transactions. The final schema comprises `transaction_id` (String), `sender_id` (String), `receiver_id` (String), `timestamp` (Timestamp), `amount` (Double), `location` (String, cleaned), and `transaction_type` (String, cleaned).

Illustrative analyzes of these preprocessed data included filtering highvalue transactions (`amount > 5000`) and aggregating total amounts by `transaction type` (showing sums around 2.46 x 109 for each type: deposit, payment, transfer, withdrawal). These results were saved to CSV.

Crucially, this preprocessed data set serves as the input for the subsequent **feature engineering** phase, essential for enhancing the capabilities of our anomaly detection models. As mentioned initially, this next phase involves deriving more insightful features, such as:

- Total transaction value per account.
- Transaction frequency for accounts within specific time intervals.
- The count of distinct counterpart accounts (senders/receivers) associated with each account.

These engineered features, built upon the clean data foundation described here, are detailed in further detail in the relevant methodology sections of this paper.

C. Abnormal Transaction Detection

Abnormal transaction detection in banking systems is vital for identifying potential threats such as fraud, unauthorized access, and money laundering. Since such anomalous behaviors are often rare and hidden within massive amounts of normal transaction data, our approach combines both **supervised** and **unsupervised** machine learning algorithms to detect abnormalities with high precision and reliability.

1) Supervised Learning Approach: Random Forest: Random Forest is an ensemble learning algorithm that constructs multiple decision trees and combines their predictions to improve accuracy and reduce overfitting. It is particularly wellsuited for highdimensional banking data.

Training Data: The algorithm is trained on labeled transaction datasets, with each record labeled as either "normal" or "abnormal."

a) Algorithm Pseudocode::

```
# Load and preprocess data
df = pd.read_csv('bank_transactions_data_2.csv')
X = df.drop(['TransactionDate', 'Label'], axis=1)
y = df['Label']

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
# Train Random Forest model
clf = RandomForestClassifier(n_estimators=100)
clf.fit(X_train, y_train)

# Make predictions
y_pred = clf.predict(X_test)
```

b) Feature Importance Calculation::

$$\text{Importance}(f_i) = \frac{1}{N} \sum_{t=1}^{N} \sum_{n \in \text{Nodes}_t} \Delta i(n, f_i)$$

Where:

- N: total number of trees
- $\Delta i(n, f_i)$: impurity reduction by feature f_i at node n

Evaluation metrics: Accuracy, Precision, Recall, ROCAUC, PRAUC.

2) Unsupervised Learning Approaches: In realworld scenarios, labeled data is often limited. Thus, we also utilize unsupervised learning techniques to detect anomalies without requiring labeled examples.

a) KMeans Clustering:: KMeans clusters transactions based on features like amount, time, and frequency. Outliers far from cluster centers are flagged as anomalies.

Pseudocode:

```
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
kmeans = KMeans(n_clusters=3)
kmeans.fit(X_scaled)

distances = kmeans.transform(X_scaled)
threshold = np.percentile(np.min(distances, axis=1), 95)
```

Distance Formula:

$$d(x_i, \mu_k) = \sqrt{\sum_{j=1}^{n} (x_{ij} \mu_{kj})^2}$$

b) Isolation Forest:: This algorithm isolates outliers using random binary trees. Anomalies are isolated faster due to their rarity.

Pseudocode:

```
iso = IsolationForest(n_estimators=100, contamination=0.05)
iso.fit(X)
anomaly_score = iso.decision_function(X)
y_pred = iso.predict(X)  # 1 for anomaly, 1 for normal
```

Anomaly Score Formula:

$$s(x) = 2^{\frac{E(h(x))}{c(n)}}$$

Where:

- $E(h(x))$: average path length for instance x
- $c(n)$: normalization factor, approximated by $2H(n1)\frac{2(n1)}{n}$

3) Hybrid Detection Framework: To maximize accuracy, we integrate all three methods into a hybrid detection pipeline:

1) **Stage 1 Isolation Forest:** Performs unsupervised prefiltering to identify potential anomalies.
2) **Stage 2 Random Forest:** Refines detection using a trained classifier on labeled anomalies.
3) **Stage 3 KMeans Clustering:** Groups suspicious transactions by similarity, highlighting behavioral patterns.

This layered strategy ensures robust anomaly detection by combining the generalization of unsupervised learning with the precision of supervised classification.

Key Contribution: We propose a novel integration where Random Forest is embedded within the Isolation Forest pipeline to reinforce decisionmaking with labeled knowledge, forming a hybrid intelligent anomaly detection system.

D. PageRank Calculation and GNN (Graph Neural Network) for the Transaction Network

1) PageRank Algorithm: Concept

PageRank is an algorithm originally developed to rank web pages based on their connectivity and importance within a graph structure. In the context of fraud detection, we leverage PageRank to assess the influence and centrality of accounts and merchants within the transaction network. Nodes with higher PageRank scores are typically more involved or central in the flow of transactions.

Mathematical Definition

The PageRank score of a node i is computed as:

$$PR(i) = \frac{1-d}{N} + d \sum_{j \in M(i)} \frac{PR(j)}{L(j)}$$

where:

- d is the damping factor, usually set to 0.85.
- N is the total number of nodes in the graph.
- $M(i)$ represents the set of nodes that link to node i.
- $L(j)$ denotes the number of outbound links from node j.

Analytical Insights

PageRank analysis highlights nodes with significant centrality or influence in the transaction graph. Based on the score distribution:

- Nodes with high PageRank tend to be structurally important, often acting as hubs in transaction flows. These may represent major merchants or accounts with broad transaction reach.
- Nodes with extremely low or near-zero PageRank values are often isolated or inactive. These might correspond to new, dormant, or suspicious accounts designed to avoid attention.
- Incorporating PageRank into machine learning pipelines provides valuable topological context, complementing behavioral features such as transaction frequency or amount.

PageRank thus acts as a graph-based feature that captures global connectivity patterns, offering potential indicators of anomalous or fraudulent activity when combined with other account behaviors.

2) Graph Neural Networks (GNNs): Concept

Graph Neural Networks (GNNs) are deep learning architectures specifically designed to operate on graph-structured data. In the context of transaction networks, they allow the model to learn complex dependencies and relationships among entities, enhancing fraud detection by leveraging both local and global graph patterns.

Mathematical Formulation

Let the transaction network be represented as a graph $G = (V, E)$, where:

- V is the set of nodes (e.g., accounts or merchants).
- E is the set of edges (e.g., financial transactions).

In GNNs, each node iteratively aggregates information from its neighbors through a message-passing mechanism. This allows node embeddings to be updated based on their local and extended neighborhood, enabling the model to capture latent patterns in the graph structure.

Autoencoder-Based Anomaly Detection

An unsupervised graph autoencoder was employed to learn embeddings for each node and reconstruct the original graph. The model was trained by minimizing reconstruction loss. Anomalies were identified based on reconstruction errors — nodes or transactions that could not be accurately reconstructed were flagged as suspicious due to their structural divergence from normal patterns.

Analytical Insights

The reconstruction loss consistently decreased during training, demonstrating that the model effectively learned structural patterns in the transaction network. A set of transactions with high reconstruction errors were flagged as anomalies, representing potential fraudulent activity. These transactions exhibited behavior patterns significantly deviating from the majority, which the model could not reconstruct confidently.

3) Comparison with Normal Transactions: A comparative analysis of anomalous versus normal transactions showed that anomalies often differ in characteristics such as transaction volume, frequency, or connectivity patterns. This highlights the effectiveness of using GNN embeddings and reconstruction-based approaches in isolating suspicious behavior that traditional models may overlook.

4) Evaluation:

- The PageRank algorithm successfully identified structurally influential accounts or merchants, which are important targets for risk-based analysis.
- The GNN-based autoencoder enabled unsupervised learning of graph representations and revealed deviations in transactional behavior without the need for labeled data.
- The combination of both methods enhances fraud detection by merging structural influence metrics (PageRank) with learned behavioral patterns (GNN), resulting in a more holistic risk assessment framework.

Conclusion

We effectively applied PageRank to measure node centrality and GNNs to learn graph-structured behavior for fraud detection. The approach enabled the identification of multiple suspicious transactions, based solely on structural and behavioral divergence within the network. Future enhancements will explore hybrid models that combine supervised learning with PageRank and GNN-based embeddings to further improve fraud classification accuracy.

Application to Fraud Detection Accounts with abnormally high or low PageRank scores may indicate suspicious activity. Fraudulent accounts may have a high PageRank due to excessive interactions. Low PageRank may indicate newly created accounts used for money laundering.

Core Concept and Idea of PageRank

As previously described, PageRank determines an account's importance based on the importance of the accounts that send transactions to it, using an iterative voting system.

The PageRank Formula

The PageRank score PR(A) for an account A is calculated using the formula:

$$PR(A) = \frac{1d}{N} + d \sum_{T_i \in M(A)} \frac{PR(T_i)}{C(T_i)}$$

With a damping factor (d) of 0.85 and N being the total number of unique accounts. Reconstruction error distribution with anomaly threshold

Implementation and Results

The Python code provided implements the PageRank algorithm in 10 iterations. It constructs the transaction network from the preprocessed data, initializes PageRank scores equally for all accounts, and then iteratively updates these scores based on the flow of transactions and the PageRank of the sending accounts.

The output of the PageRank calculation shows the top 20 accounts with their calculated PageRank scores and their corresponding `Is_Fraud` flag. The results indicate the following for the top 20 accounts:

- **Account ID:** The account ID appearing for all top 20 results is "101206".
- **PageRank Score:** The PageRank score for this account is approximately 0.000001 for all 20 entries.
- **Is**$_{F}raudFlag : The$**Is_Fraud**$flagforthisaccountis0, indicatingthatitwasnotidentifiedaspotentiallyfraudulentbased$

The fact that the same account ID appears repeatedly in the top 20 results with an identical, very low PageRank score suggests a few possibilities. It might indicate a characteristic of the synthetic dataset where this particular account has a specific pattern of transactions that leads to this outcome. Alternatively, there might be a factor in the PageRank calculation or the result sorting process that is causing this repetition. The low PageRank score could be due to a very large number of accounts in the network, resulting in a small initial score, and the limited number of iterations might not have allowed for significant rank propagation.

The observation that the topranked account (based on this output) is not flagged as fraudulent could be an interesting finding, suggesting that high network importance (as measured by this initial PageRank run) does not necessarily correlate with the defined fraud indicators. Further investigation into the transaction patterns of account ”101206” and potentially increasing the number of PageRank iterations might provide more detailed insights.

IV. Conclusions

This paper explored two significant domains in data science: big data processing using the PageRank algorithm, and anomaly detection in transactional behavior utilizing various machine learning techniques.

In the context of big data processing, the Hadoop Distributed File System (HDFS) and MapReduce framework provide a scalable infrastructure capable of handling large-scale graph structures such as those used in PageRank. HDFS ensures reliable data storage and fault tolerance, while MapReduce facilitates parallel computation across distributed systems. Despite some challenges including latency and complexity in job coordination, the combination of HDFS and MapReduce remains a robust choice for ranking tasks that require processing web-scale data.

The PageRank algorithm itself, while powerful, faces limitations due to its reliance on static hyperlink structures. These limitations can be addressed by integrating machine learning techniques to dynamically assign link weights or combining PageRank with other ranking models to capture temporal and contextual variations of web content.

The second focus of this study was anomaly detection in financial transactions. We evaluated several machine learning models—both supervised and unsupervised—for identifying unusual patterns that may indicate fraudulent activity. The evaluation involved K-Means Clustering, Isolation Forest, Random Forest, and Graph Neural Networks (GNNs). Each method demonstrated unique advantages, as summarized below:

A. K-Means Clustering

K-Means is an unsupervised algorithm that groups transactions based on proximity to centroids. In our experiment, it identified approximately 70 anomalous transactions, many of which involved unusually high “Np tin” (deposit) amounts, such as 9157.10 and 9390.57. This indicates its utility in detecting value-based outliers.

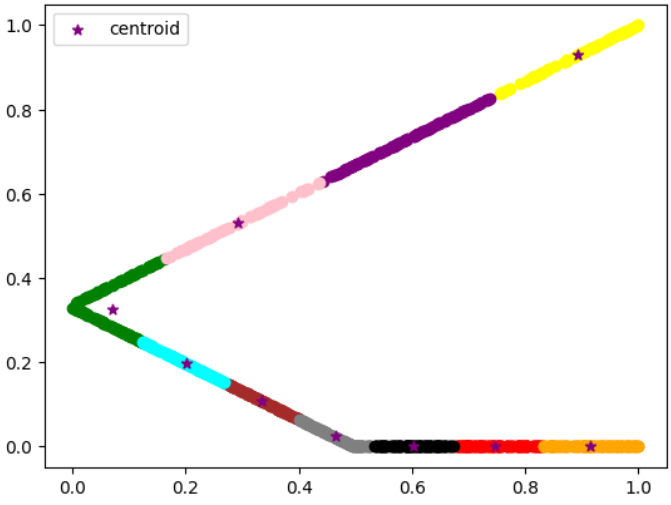

Fig. 1. K-Means Clustering Result

B. Isolation Forest and Random Forest

Isolation Forest, as an unsupervised anomaly detection model, isolates observations by randomly selecting features and splitting values. It is well-suited to detecting outliers in high-dimensional spaces without needing labeled data.

In contrast, Random Forest is a supervised model that achieved high accuracy in detecting fraudulent transactions. Its performance was evaluated using ROC AUC and Precision-Recall (PR) AUC metrics:

- **ROC AUC Score**: 0.9978
- **Precision-Recall AUC Score**: 0.7639

While the ROC AUC indicates excellent discrimination between fraud and non-fraud, the PR AUC highlights the difficulty of identifying rare fraud cases within imbalanced datasets.

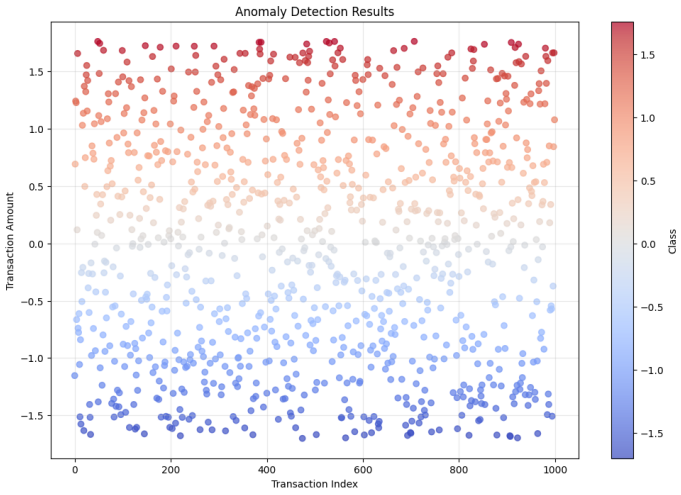

Fig. 2. Isolation Forest Results

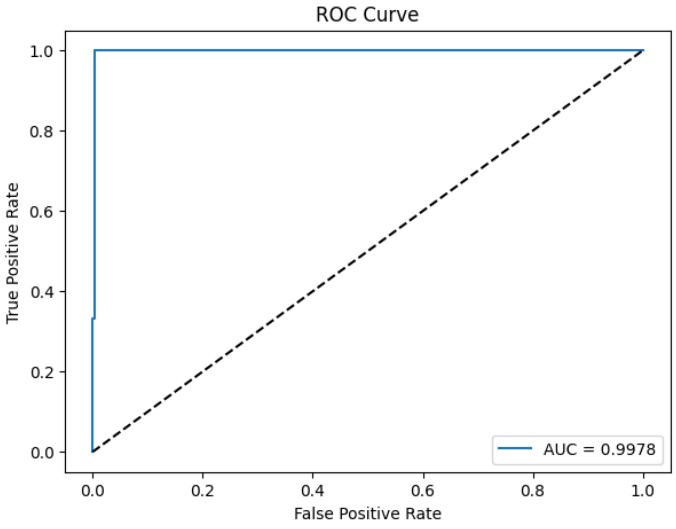

Fig. 3. Random Forest ROC Curve

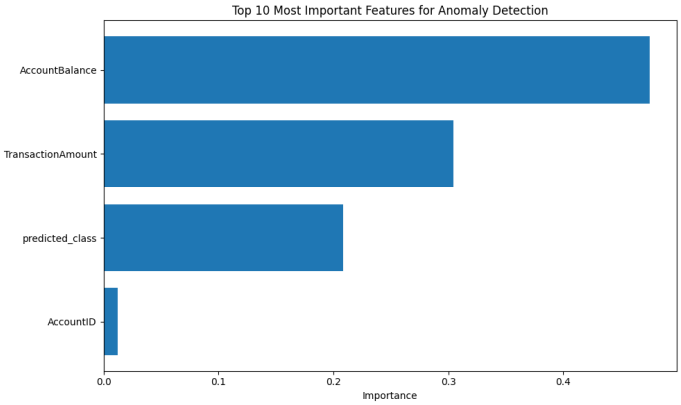

Fig. 4. Random Forest PR Curve

C. Graph Neural Networks (GNNs)

GNNs were used to model the transaction network as a graph, where nodes represent accounts and edges denote transactions. The model was trained for 100 epochs, showing consistent loss reduction, confirming its ability to learn complex transaction patterns.

TABLE I
GNN TRAINING LOSS ACROSS EPOCHS

Epoch	Loss
10	0.5745
20	0.4982
30	0.4647
40	0.4073
50	0.4028
60	0.3813
70	0.3714
80	0.3172
90	0.2795
100	0.2279

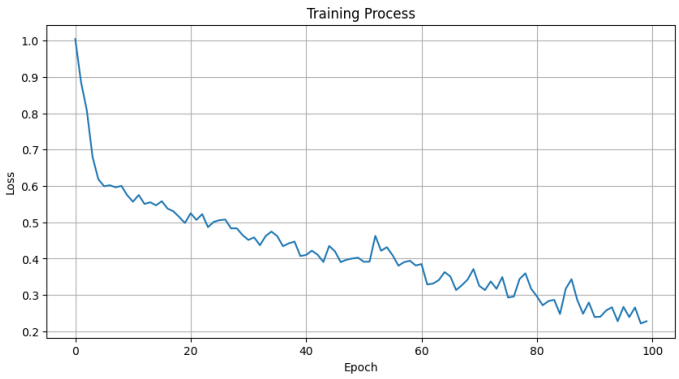

Fig. 5. GNN Loss Visualization - Epoch 1 to 100

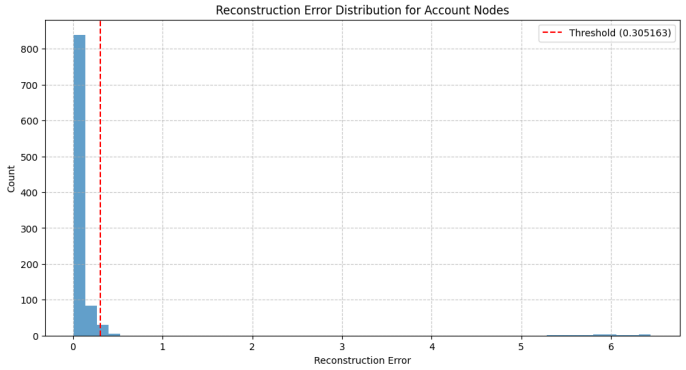

Fig. 6. GNN Embedding Space

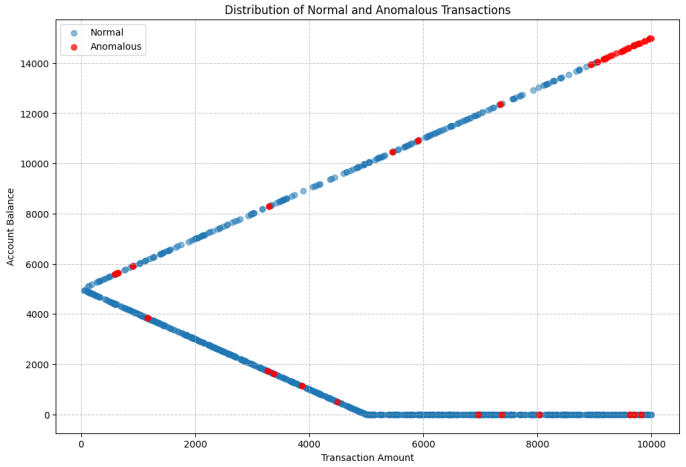

Fig. 7. Detected Fraud Nodes via GNN

As seen in Table I, the GNN model effectively minimized the loss, showing its ability to learn latent relationships in graph-structured data.

Final Remarks

In summary, this research highlights the importance of combining scalable big data infrastructure with intelligent machine learning models to build effective analytical systems. Distributed systems like HDFS and MapReduce provide the backbone for handling large-scale data, while models like GNNs and ensemble methods offer robust approaches for detecting anomalies in financial transactions. The integration of these technologies not only enhances analytical accuracy but also contributes to advancements in domains such as web search, recommendation engines, and fraud detection systems.

REFERENCES

[1] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, 2001.
[2] R. J. Bolton and D. J. Hand, "Statistical fraud detection: A review," *Statistical Science*, vol. 17, no. 3, pp. 235259, 2002.
[3] T. Fawcett and F. Provost, "Adaptive fraud detection," *Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 291316, 1997.
[4] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 158, 2009.
[5] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[6] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," *Stanford InfoLab*, 1999.
[7] L. Akoglu, H. Tong, and D. Koutra, "Graphbased anomaly detection and description: A survey," *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 603638, 2015.

[8] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 10241034.
[9] C. C. Noble and D. J. Cook, "Graphbased anomaly detection," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 631636.
[10] Y. Hu, L. Zhao, and S. Y. Philip, "Fraud detection in online transactions: A graphbased approach," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2016, pp. 572578.
[11] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107113, 2008.
[12] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, 2010, pp. 1010.
[13] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Stanford InfoLab Technical Report, 1998.
[14] S. Brin and L. Page, "Anatomy of a largescale hypertextual web search engine," Computer Networks and ISDN Systems, vol. 30, no. 17, pp. 107117, 1998.