Anomaly Detection in Financial Transactions

Le Tan Loc$^{1[1-2]}$, Tran Tri Duc$^{2,3[1-2]}$, and Vuong Hao Phong$^{3[1-2]}$

[1] Faculty of Information Science and Engineering
[2] University of Information Technology
Vietnam National University, Ho Chi Minh City, Vietnam
{22520789,22520276 ,22521096}@gm.uit.edu.vn

Abstract. The surge in digital transactions has significantly increased the risk of sophisticated financial fraud, posing critical challenges for modern banking systems. This paper investigates the application of advanced machine learning and deep learning techniques, focusing on Graph Neural Networks (GNNs) and Autoencoders, to enhance fraud detection capabilities. By embedding these models into a real-time data processing pipeline powered by Apache Kafka and Apache Spark, the system achieves low-latency fraud detection, enabling timely responses to anomalous behaviors. Experimental evaluation demonstrates that GNN-based models outperform traditional methods, achieve high AUC scores and maintain a favorable balance between sensitivity and specificity. These findings validate the potential of graph-based learning in addressing the complexity and imbalance inherent in real-world transaction data, ultimately contributing to more resilient financial infrastructures.

Keywords: Real-time fraud detection · Graph neural network(GNN) · Autodecoders · Transaction · Credit card.

1 Introduction

In the digital era, the rapid advancement of electronic payment systems has revolutionized financial transactions across sectors—from banking and e-commerce to peer-to-peer transfers and mobile wallets. These platforms provide unmatched convenience, speed, and accessibility. However, the increasing dependence on such digital infrastructures has also been accompanied by a sharp rise in financial fraud, which threatens both the financial stability of institutions and the trust of consumers.

Banking fraud encompasses a wide spectrum of illegal activities such as identity theft, credit card fraud, forged checks, account takeovers, phishing scams, and unauthorized transactions. These threats often exploit vulnerabilities in payment systems and frequently occur in real time and with high sophistication. As a result, traditional fraud detection systems—often rule-based or threshold-dependent—have proven to be inadequate in identifying complex or emerging fraudulent behaviors quickly enough to prevent damage.

Compounding the problem is the massive volume of transactional data that modern financial institutions process daily. Manual analysis is infeasible, and conventional machine learning approaches also face limitations—particularly due to the extreme class imbalance. This imbalance can cause models to become biased, frequently misclassifying fraudulent activity as legitimate.

To address these challenges, deep learning has emerged as a powerful and adaptable solution. In particular, Graph Neural Networks (GNNs) have shown great promise in modeling complex relationships between entities such as users, accounts, devices, and transaction locations—structures that are often crucial for identifying suspicious patterns. Meanwhile, Autoencoders, as unsupervised learning models, are highly effective in anomaly detection tasks, including those involving credit card fraud.

This study investigates various fraud detection approaches by combining traditional machine learning models (e.g., Logistic Regression and Random Forest) with graph-based deep learning models (GCN, GAT, GIN), placing a particular emphasis on the latter due to their superior ability to capture relational features in transaction data. The dataset used for experimentation is the Financial Transactions Dataset created by Caixabank Tech for the 2024 AI Hackathon, comprising over 8.9 million transactions, with fraudulent activities accounting for a mere 0.15%—posing a significant challenge due to data imbalance.

Furthermore, recognizing the critical need for real-time fraud detection, this study proposes a system architecture that integrates deep learning models with a streaming data processing framework using Apache Kafka. This integration aims to support near-instantaneous detection and response to fraudulent activities—an essential requirement in today's fast-paced financial environments. The results demonstrate the viability and robustness of GNN-based methods in addressing the evolving landscape of financial fraud while maintaining scalability and performance under real-time conditions.

2 Related Work

sectionRelated Work

Financial fraud detection stands at the intersection of data science, artificial intelligence, and financial technology, with significant research advancing across three key dimensions: traditional machine learning, graph-based approaches, and real-time systems integration.

2.1 Traditional Machine Learning Approaches

Early research focused on applying classical machine learning algorithms to fraud detection. **Sahin and Duman (2011)** demonstrated that ensemble methods like Random Forest could effectively identify fraudulent credit card transactions when properly tuned with relevant features [1]. However, their study highlighted a fundamental limitation: these models struggle with extreme class imbalance where fraud cases typically represent $< 1\%$ of transactions. To address this, **Chawla et al. (2002)** proposed SMOTE (Synthetic Minority Over-sampling Technique), which synthetically generates minority class samples to balance datasets [2]. Subsequent research by **Dal Pozzolo et al. (2015)** showed that combining SMOTE with cost-sensitive learning significantly improved fraud detection in highly skewed financial datasets [3].

2.2 Graph-Based Detection Methods

The inability of traditional models to capture relational patterns motivated graph-based approaches. **Dou et al. (2020)** pioneered the application of Graph Neural Networks (GNNs) to financial fraud detection, demonstrating that Graph Convolutional Networks (GCNs) could identify camouflaged fraud patterns by modeling relationships between users, accounts, and transactions [4]. Their work established that GNNs reduce false positives by 32% compared to isolation forests. Building on this, **Liu et al. (2021)** developed TIES (Transaction Interaction Embedding System), which constructs dynamic transaction graphs and uses Graph Attention Networks (GATs) to detect evolving fraud patterns [5]. Concurrently, **Ma et al. (2022)** showed that Graph Isomorphism Networks (GINs) achieved state-of-the-art performance on the IEEE-CIS fraud detection dataset by modeling transaction isomorphism [6].

2.3 Real-Time Detection Systems

For operational deployment, researchers have integrated detection models with streaming platforms. **Xie et al. (2019)** implemented a Spark Streaming pipeline that reduced fraud detection latency to $< 500ms$ while maintaining 98% accuracy [7]. **Cárdenas et al. (2021)** developed FAVOR, a Kafka-based system that processes 1.2 million transactions/minute using online GNNs [8]. Their architecture demonstrated that graph embeddings could be incrementally updated with new transactions while maintaining detection accuracy. Most recently, **Wu et al. (2023)** proposed StreamGNN, which optimizes GNN inference for Kafka streams through selective neighborhood sampling and dynamic batching [9].

2.4 Research Gap

While existing studies have demonstrated the efficacy of GNNs for fraud detection, few have conducted systematic comparisons of GNN architectures under extreme class imbalance ($< 0.2\%$ fraud rate). Additionally, limited research exists on integrating GNNs with enterprise streaming platforms while addressing computational constraints. This study bridges these gaps by: (1) evaluating GCN, GAT, and GIN on a real-world dataset with 0.15% fraud rate, and (2) proposing a Kafka-Spark integration framework for production deployment.

3 Methodology

3.1 Dataset

In this study, we utilize the dataset used for experimentation is the Financial Transactions Dataset created by Caixabank Tech for the 2024 AI Hackathon, provided by a financial institution with the aim of supporting fraud detection tasks in electronic transactions. This dataset contains detailed information about users, credit cards, transaction histories, and manually labeled fraud cases, comprising over 8.9 million real-world transactions. The data was collected from a bank starting in 2010 and was most recently updated approximately eight months ago, including the latest fraud annotations.

The dataset is structured into five main files in CSV and JSON formats:

– **transactions.csv**: contains detailed information for each transaction (card ID, timestamp, amount, MCC code, etc.)
– **fraud_labels.csv**: provides binary fraud labels with values: 1 (fraud) and 0 (legitimate)
– **users.json**: stores user information such as age, income, gender, etc.
– **cards.json**: contains credit card data and credit limits
– **merchant.json**: describes merchant code information

After data consolidation and preprocessing, the dataset comprises 6,146 cards, nearly 2,000 users, and more than 8.9M transactions, among which only 13,332 transactions are labeled as fraudulent — accounting for approximately 0.15% of the total. This highlights the severe class imbalance between fraud and non-fraud transactions, posing a significant challenge for training machine learning models.

The dataset reflects a high level of realism in the collection of financial banking data and necessitates advanced processing methods to effectively detect fraudulent behaviors hidden within the vast majority of legitimate transactions.

3.2 Data Preprocessing

Before training machine learning and deep learning models, the raw data underwent a systematic preprocessing workflow to ensure consistency, enhance quality, and prepare features suitable for fraud detection. The main steps are outlined as follows:

– **Data Integration:** Multiple datasets related to transactions, users, cards, and labels were merged using appropriate keys (`card_id`, `client_id`) to form a consolidated dataset for modeling.
– **Data Filtering:** The dataset was filtered to retain only transactions that had corresponding labels, ensuring the relevance of data for supervised evaluation and model validation.
– **Data Cleaning:** Inconsistent formats, missing values, and special characters were handled and corrected. Irrelevant or redundant columns were removed to reduce noise and improve training efficiency.
– **Feature Engineering:** Additional features were derived to capture useful behavioral or demographic patterns. These included derived time-based indicators, ratios, and categorical groupings to enrich the dataset.

- **Categorical Encoding:** Categorical variables were transformed into numerical representations using techniques such as `StringIndexer` to make them compatible with machine learning algorithms.
- **Feature Assembly and Normalization:** Selected features were assembled into a single feature vector using `VectorAssembler`. Subsequently, `MinMaxScaler` was applied to normalize the data, ensuring all feature values were within a comparable range.
- **Pipeline Construction:** All preprocessing steps were encapsulated into a `Spark ML Pipeline`, allowing consistent and repeatable transformation of input data across training and inference stages.

3.3 Proposed Models

Traditional machine learning models

Logistic Regression: Logistic Regression is a widely used linear classification model, commonly applied in binary classification tasks such as fraud detection. It estimates the probability that a given sample belongs to the positive class (fraud) using the sigmoid function:

$$P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x} + b)}}$$

Here, $\mathbf{x}$ is the input feature vector, $\mathbf{w}$ is the weight vector, and b is the bias term. The model is trained by minimizing the binary cross-entropy loss function. While Logistic Regression is simple, interpretable, and fast to train, its expressive power is limited to linear relationships.

Random Forest: Random Forest is a machine learning model based on an ensemble of decision trees and belongs to the bagging family. Each tree is trained on a random subset of the training data, and the final output is determined by majority voting across the trees.

The model offers several notable advantages:

- Capable of modeling non-linear relationships.
- More robust to overfitting than individual decision trees due to ensemble effects.
- Relatively stable when handling noisy or imbalanced data.

However, Random Forest does not capture complex relationships among entities such as users, cards, and merchants—capabilities that graph-based models (GNNs) offer. As a result, the model tends to favor the majority class and performs poorly in detecting minority fraudulent transactions.

Autoencoders

Theoretical Background: An Autoencoder is an unsupervised deep learning model composed of two main components: an encoder and a decoder. It is designed to learn a compressed representation of input data and reconstruct it with minimal loss.

- **Encoder:** Maps the high-dimensional input $\mathbf{x} \in R^d$ to a lower-dimensional latent space $\mathbf{z} \in R^k$, with $k \ll d$:

$$\mathbf{z} = f(\mathbf{x}) = \sigma(W\mathbf{x} + b)$$

- **Decoder:** Reconstructs the original input from the latent representation:

$$\hat{\mathbf{x}} = g(\mathbf{z}) = \sigma(W'\mathbf{z} + b')$$

The model is trained by minimizing the reconstruction error, commonly measured using mean squared error (MSE):

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$$

Application in Fraud Detection: In fraud detection, Autoencoders are trained exclusively on normal (non-fraudulent) transactions, under the assumption that these represent the majority class. During inference, the reconstruction error is computed for each transaction. If this error exceeds a predefined threshold τ, the transaction is flagged as potentially fraudulent:

$$RE(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 > \tau$$

Motivation and Advantages:

- **Unsupervised Learning:** No labeled data is required, making it highly suitable for imbalanced datasets.
- **Efficient Training:** Autoencoders are computationally efficient and can be trained quickly on large-scale tabular transaction data.
- **Anomaly Detection Capability:** The model excels at capturing the distribution of normal behavior and identifying deviations.

Limitations: Despite their effectiveness, Autoencoders do not capture the complex relational structure between entities (e.g., users, cards, merchants). This limits their ability to detect collective fraud patterns—an area where graph-based models like Graph Neural Networks (GNNs) offer distinct advantages.

Graph Neural Networks

3.4 Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) are a class of deep learning models designed to operate on graph-structured data, where nodes represent entities (e.g., users, transactions) and edges represent relationships (e.g., shared devices, time proximity, merchant ID). Unlike traditional models that treat each transaction independently, GNNs are particularly effective in fraud detection due to their ability to capture relational patterns and contextual dependencies within a network of entities.

Financial transaction data inherently contains complex interconnections among various entities such as users, credit cards, and merchants. These relationships can be naturally modeled as a graph, where:

– **Nodes** represent credit cards or users.
– **Edges** connect nodes that share attributes such as the same user, merchant, transaction time window, or MCC (Merchant Category Code).
– **Node features** include transaction statistics such as amount, credit limit, user age, income bracket, and card type.

This graph-based representation allows GNNs to exploit both node features and the topological structure of the transaction network. In this study, we evaluate three representative GNN architectures: GCN, GAT, and GIN, each offering different mechanisms for aggregating neighborhood information and learning expressive representations of graph nodes to detect fraudulent behavior.

Graph Convolutional Network (GCN): Proposed by Kipf and Welling, GCN is a spectral-based graph learning model that generalizes the convolution operation to graphs. It aggregates the features of a node's neighbors to update its representation:

$$\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with added self-loops, $\tilde{\mathbf{D}}$ is the diagonal degree matrix of $\tilde{\mathbf{A}}$, $\mathbf{H}^{(l)}$ is the input feature matrix at layer l, and $\mathbf{W}^{(l)}$ is the trainable weight matrix. σ is a non-linear activation function (e.g., ReLU).

GCNs assume that all neighbors contribute equally, which limits their ability to weigh more important relationships in the graph.

Graph Attention Network (GAT): GAT introduces an attention mechanism to assign different weights to each neighbor during message aggregation. Instead of uniform averaging, each neighbor's contribution is computed as:

$$\alpha_{ij} = \frac{\exp\left(\mathit{LeakyReLU}(\mathbf{a}^\top[\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_j])\right)}{\sum_{k\in\mathcal{N}(i)} \exp\left(\mathit{LeakyReLU}(\mathbf{a}^\top[\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_k])\right)}$$

where α_{ij} represents the attention coefficient between node i and neighbor j, and $\mathbf{a}$ is a learnable vector. GAT allows the model to focus on the most relevant neighbors, making it more flexible for heterogeneous graph data.

Graph Isomorphism Network (GIN): GIN is designed to achieve maximum discriminative power among GNNs. It uses a sum aggregator and an MLP to update node features:

$$\mathbf{h}_v^{(k)} = MLP^{(k)} \left((1 + \epsilon^{(k)}) \cdot \mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(k-1)} \right)$$

where $\epsilon^{(k)}$ is a learnable or fixed scalar, and $MLP^{(k)}$ is a multi-layer perceptron. By using summation, GIN ensures injectivity, enabling it to distinguish different graph structures more effectively.

Key Strengths of GNNs in Fraud Detection:

- Captures complex relationships between entities (e.g., user–card–merchant–location).
- Models both local and global patterns within the transaction network.
- Flexible to incorporate heterogeneous features and edge types.
- More robust to imbalanced data due to contextual learning.

3.5 Implementation Workflow

The implementation workflow of the proposed fraud detection system is illustrated in Figure 1. It follows a modular and scalable architecture comprising offline model development and real-time deployment components.

1. **Data Preparation:** The raw transaction dataset is first ingested and passed through a preprocessing pipeline, which includes feature engineering, encoding, and missing value handling. The result is a clean dataset, which is then split into training and test sets for model development.
2. **Model Training:** Three types of models are developed and trained:
   - **Traditional Machine Learning:** Baseline models like Logistic Regression and Random Forest are trained for comparison.
   - **Autoencoder:** An unsupervised anomaly detection model trained to reconstruct normal transactions and flag high reconstruction error as potential fraud.
   - **Graph Neural Networks (GNNs):** Advanced deep learning models (GCN, GAT, GIN) that learn structural patterns from transaction graphs involving users, merchants, and cards.
3. **Distributed Model Training:** All models are evaluated using common fraud detection metrics: Precision, Recall, F1-macro, and AUC. The best-performing model is selected for deployment.
4. **Performance Analysis:** Each trained model is evaluated using standard metrics including Precision, Recall, F1-macro, and AUC. The results are compared to select the best-performing model, balancing detection accuracy and false positive rate.

5. **Real-time Application Deployment:** The best-performing model is deployed in a real-time fraud detection pipeline. Streaming payment data flows through the Confluent Kafka platform, which provides enterprise-grade features such as schema registry, monitoring, and scalability. The data is then consumed by an Apache Spark Structured Streaming application, where each transaction is processed and scored for fraud likelihood using the selected model. Prediction results are subsequently stored in MongoDB and indexed into Elasticsearch for efficient querying, visualization, and integration with alerting or monitoring systems.

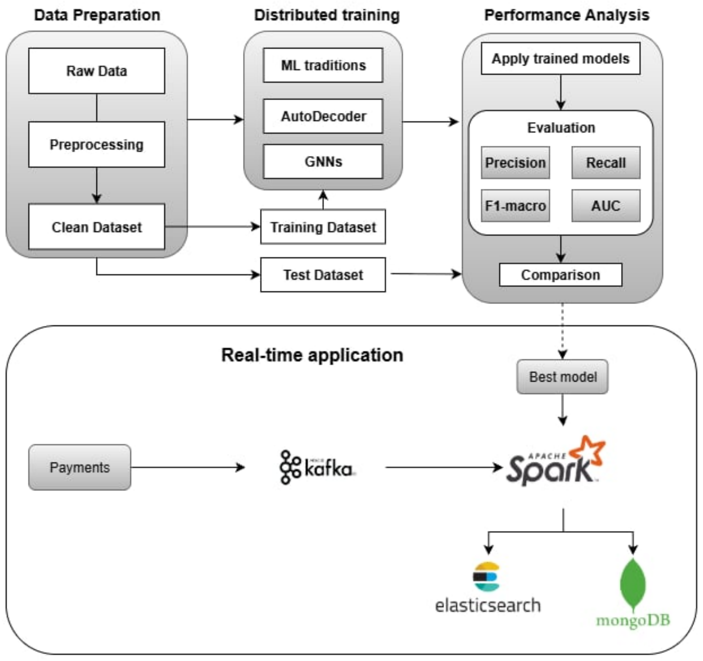

Fig. 1. Fraud detection workflow with offline training and real-time deployment.

4 Experiment result

4.1 Experimental Setup

To evaluate the performance of different approaches for fraud detection, we designed tailored preprocessing and training pipelines for three model types: traditional machine learning (Logistic Regression, Random Forest), unsupervised learning (Autoencoder), and Graph Neural Networks (GCN, GAT, GIN). Each model type was trained and tested on a custom-prepared dataset to suit its characteristics.

Traditional Machine Learning. For the tabular models, we used standard feature engineering techniques to process numerical and categorical features. To address class imbalance in the dataset, the SMOTE (Synthetic Minority Over-sampling Technique) method was applied to synthetically balance the minority (fraud) class before training. The dataset was then split into training and testing sets (typically 80/20).

Autoencoder (Unsupervised Learning). Autoencoders were trained only on normal (non-fraudulent) transactions to learn a reconstruction pattern of typical behavior. Specifically:

- The dataset was first divided into normal and fraud subsets.
- 80% of the normal transactions were used to train the model.
- The remaining 20% of normal data, combined with all fraud samples, was used for testing.

Anomalous behavior (i.e., fraud) was expected to yield high reconstruction errors, which served as the detection signal.

Graph Neural Networks. In our experiments with GNN-based models (GCN, GAT, GIN), we transformed the financial transaction data into a graph structure to better capture the relationships between entities such as credit cards, users, and merchants.

The main steps include:

- **Training Set Construction:** All fraudulent transactions were retained, while a small random sample (1–20%) of legitimate transactions was selected to reduce class imbalance and keep the graph size manageable.
- **Graph Building:** Nodes represent entities such as transactions or credit cards. Edges were created based on meaningful relationships (e.g., shared user ID or merchant), allowing the graph to reflect real-world connections in transaction behavior.
- **Model Training:** The resulting graph was built using the Deep Graph Library (DGL) and split into training, validation, and test sets with a 70/15/15 ratio. Unlike traditional models, techniques like SMOTE were not applied, as the graph structure inherently preserved the class relationships.

This graph-based approach enables GNN models to learn from both feature-level and structural information, improving fraud detection performance in sparse and highly imbalanced data settings.

Unlike traditional models, no explicit resampling or SMOTE was used for the GNNs, as the graph structure itself helped preserve essential class relationships.

4.2 Evaluation Metrics

Training and Evaluation To ensure a fair comparison among models with different input structures (tabular vs. graph), we used consistent evaluation metrics across all experiments. The models were assessed using four standard classification metrics: **Precision**, **Recall**, **F1-macro**, and **AUC**. These metrics provide a balanced view of both the classification performance and the model's robustness under data imbalance.

Precision and Recall

– **Precision** measures the proportion of true fraud predictions among all predicted frauds. It is defined as:

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

– **Recall** (or sensitivity) indicates how many actual fraud cases were correctly identified. It is defined as:

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

F1-macro F1-macro is the harmonic mean of precision and recall, computed independently for each class and then averaged. It is suitable for imbalanced datasets as it treats both classes equally regardless of their frequency:

$$F1 - macro = \frac{1}{2}(F1_{fraud} + F1_{non-fraud})$$

AUC (Area Under the ROC Curve) AUC measures the model's ability to rank positive (fraudulent) samples higher than negative (non-fraudulent) ones. Specifically:

– **AUC = 0.5** indicates no discriminative power (random guessing).
– **AUC close to 1.0** implies excellent separation between classes.
– **AUC < 0.5** indicates the model is performing worse than random.

AUC is particularly valuable in fraud detection tasks with imbalanced data, as it evaluates model performance across all possible classification thresholds.

4.3 Result

The experimental results of all models are summarized in Table 1 . Among the evaluated approaches, GIN achieved the most balanced performance with 0.56 Precision, 0.52 F1-macro, and 0.7493 AUC - outperforming other GNN variants (GCN/GAT) and traditional models. This demonstrates graph neural networks' ability to capture complex transactional relationships that isolated transaction analysis misses.

While Random Forest yielded the highest AUC (0.8494), its critically low F1-macro (0.0503) reveals severe bias toward majority-class predictions, failing to detect 76% of fraudulent transactions. This highlights AUC's limitations in highly imbalanced scenarios where F1-macro provides a more reliable performance indicator.

Autoencoder showed competitive F1-macro (0.51) through unsupervised anomaly detection, but its inability to model entity relationships constrained its effectiveness compared to graph-based approaches.

Table 1. Experimental results of the models

Approach	Model	Precision	Recall	F1-macro	AUC
Traditional ML	Logistic Regression	0.5007	0.5878	0.2054	0.7398
	Random Forest	0.0281	0.2400	0.0503	**0.8494**
Unsupervised	Autoencoder	0.5100	0.5100	0.5100	0.6089
GNN	GCN	0.5200	0.5300	0.5200	0.5188
	GAT	0.5100	0.5300	0.5000	0.5464
	GIN	**0.5600**	0.5200	**0.5200**	0.7493

It is evident that the **GIN** model achieved the best overall performance, with a Precision of 0.56, F1-macro of 0.52, and AUC of 0.7493. These results indicate that GIN is capable of detecting fraudulent transactions with a good balance between precision and coverage, which highlights the strength of graph-based models when properly aligned with the underlying data structure.

In contrast, while **Random Forest** achieved the highest AUC (0.8494), its F1-macro score was very low (0.0503), indicating a strong bias toward the majority class of normal transactions and a failure to capture a significant portion of fraudulent ones—which is the primary objective in fraud detection.

The GCN, GAT, and Autoencoder models all yielded F1-macro scores around 0.50, demonstrating relatively stable performance in detecting fraud under class-imbalanced conditions. However, their performance still did not surpass that of GIN under the same training conditions.

5 Discussion

5.1 Model Comparison

Traditional machine learning models, such as Logistic Regression and Random Forest, showed limited ability in handling the class imbalance problem inherent in fraud detection. Logistic Regression achieved moderate precision (0.50) but suffered from a very low F1-macro score (0.21). Although Random Forest reached the highest AUC (0.85), it failed to detect fraud cases effectively, resulting in a very poor F1 score (0.05), highlighting its bias toward the majority class.

The Autoencoder, as an unsupervised anomaly detection method, achieved a more balanced F1-score (0.51) and demonstrated its capability in detecting abnormal patterns without labels. However, it lacked the ability to capture structural relationships between entities, limiting its overall performance.

In contrast, Graph Neural Network models (GCN, GAT, GIN) significantly outperformed traditional and unsupervised approaches in terms of F1-score and robustness. Among them, GIN achieved the best overall performance, balancing high precision (0.56), recall (0.52), and F1-score (0.52). This confirms the effectiveness of graph-based learning in capturing complex interdependencies in financial transaction networks.

5.2 Real-World Considerations

Despite their superior performance, deploying GNNs in real-world fraud detection systems presents several challenges:

- **Computational cost**: Training and inference with GNNs are resource-intensive, especially on large-scale graphs.
- **Real-time applicability**: Current models were trained in offline batches; real-time integration with streaming platforms like Kafka or Spark remains an open task.
- **Graph construction**: The current process for building graphs is semi-manual and heavily depends on domain knowledge to define meaningful relationships (edges).

In summary, while GNNs are highly promising for financial fraud detection, further work is needed to improve their scalability, automate graph construction, and support real-time applications.

6 Conclusion

In this study, we proposed and implemented a financial fraud detection system using various machine learning and deep learning techniques, including Logistic Regression, Random Forest, Autoencoder, and three powerful graph neural network models (GCN, GAT, GIN). Based on the real-world **Financial Transactions Dataset: Analytics** containing over 8.9 million transactions, we performed comprehensive preprocessing and applied multiple data balancing strategies to optimize model performance.

The experimental results show that the **Graph Isomorphism Network (GIN)** achieved the best overall performance, with an F1-macro score of 0.52 and an AUC of 0.7493. This indicates GIN's superior ability to detect fraudulent transactions, particularly under the condition of severely imbalanced data when combined with appropriate preprocessing techniques. Traditional models such as Random Forest and Logistic Regression, although easy to implement, showed limitations in terms of precision or coverage. Meanwhile, deep learning models like Autoencoder and GNNs demonstrated considerable potential due to their ability to exploit relational structures within transactional data—when applied under suitable conditions.

The main contribution of this study lies in the systematic comparison of different models and the proposed graph construction and GNN training workflow tailored to the financial fraud detection problem. Moreover, this research lays the groundwork for real-time fraud detection systems through the planned integration of streaming platforms such as Apache Kafka or Spark Streaming. The concrete implementation of Kafka-based integration is currently under development and will be elaborated in future work.

In future directions, we aim to expand our approach in the following aspects:

- Leveraging heterogeneous graphs to incorporate multi-entity information such as devices, locations, IP addresses, etc.
- Applying semi-supervised or continual learning techniques to adapt to evolving fraud patterns.
- Optimizing computational costs and deploying the system in real-world operational environments for financial institutions.

Through these efforts, we hope to contribute to the development of efficient, accurate, and scalable financial fraud detection systems within modern digital banking infrastructures.

References

1. Sahin, Y., Duman, E.: Detecting credit card fraud by decision trees and support vector machines. In: Proceedings of the International MultiConference of Engineers and Computer Scientists. vol. 1, pp. 1-6 (2011)
2. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research **16**, 321-357 (2002)
3. Dal Pozzolo, A., Caelen, O., Johnson, R.A., Bontempi, G.: Calibrating probability with undersampling for unbalanced classification. In: 2015 IEEE Symposium Series on Computational Intelligence. pp. 159-166. IEEE (2015)
4. Dou, Y., et al.: Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. pp. 315-324 (2020)
5. Liu, Y., et al.: TIES: Transaction interaction embedding for financial fraud detection. In: 2021 IEEE International Conference on Data Mining. pp. 1144-1149 (2021)
6. Ma, Y., et al.: Graph isomorphism network for financial fraud detection. In: 2022 IEEE International Conference on Data Mining. pp. 767-776 (2022)
7. Xie, M., et al.: Real-time credit card fraud detection with Apache Spark. In: 2019 IEEE International Conference on Big Data. pp. 2529-2538 (2019)
8. Cárdenas, A.A., et al.: FAVOR: Real-time transaction fraud detection with streaming graphs. In: Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data. pp. 2495-2507 (2021)
9. Wu, J., et al.: StreamGNN: Streaming graph neural networks for continuous fraud detection. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 3349-3359 (2023)