Unmasking Fraud in Transaction Networks: Harnessing Heterogeneous Graph Neural Networks for Enhanced Detection

Kanishk Goel

Feb 2025

Abstract

Credit card fraud pose a significant threat to global financial systems. They lose billions of dollars annually. Traditional machine learning techniques such as Random Forests and XGBoost have demonstrated effectiveness in fraud detection. However, they often fail to capture the complex interdependencies among entities involved in transactions. This paper explores the efficacy of Graph Neural Networks motivated by recent advances in graph-based learning. We focus on Heterogeneous Graph Convolutional Networks for enhanced credit card fraud detection.

We conduct a comparative study between classical ML models and GNNs using a synthetically generated credit card transaction dataset. Over one million records simulate interactions between customers, merchants, and transactions. The graph-based structure these entities as distinct node types connected by meaningful edges. We implement the GNN framework using PyTorch Geometric and evaluate the models using the F1-score to account for class imbalance.

Experimental results demonstrate that GNNs outperform traditional ML classifiers by leveraging relational information and neighbourhood context. They detect fraudulent patterns, especially when we have more embeddings and fewer meaningful features. This work reinforces the potential of graph-based learning as a powerful approach for high-volume and relationally rich transaction networks.

Contents

1 Introduction

In the digital economy, the rapid adoption of online transactions and plastic money has increased the prevalence of credit card fraud affecting consumers, businesses, and financial institutions [1]. The nature of credit card fraud is constantly evolving, such as phishing, counterfeit cards, identity theft, and unauthorized online transactions [2]. Fraudulent activities significantly erode user trust, increase operational costs, and introduce regulatory challenges despite forming a small proportion of all transactions [3].

Traditionally, fraud detection relied on rule-based systems and manual investigations, which are now rendered inadequate by the scale and velocity of modern digital transactions. Recent years have seen the deployment of machine learning (ML) methods such as Logistic Regression, Decision Trees, and ensemble models like Random Forests and XGBoost for fraud detection [4, 2]. These models offer promising accuracy but treat transactions as isolated events and fail to exploit relational information between entities involved in transactions.

Our work investigates Graph Neural Networks to address this limitation. GNNs model transactions as interactions within a network of entities such as users, merchants, and transactions. Such capabilities make them well-suited for detecting subtle anomalies in transaction behaviour indicative of fraud.

In this study, we propose a comparative evaluation of traditional ML models—Random Forests and XGBoost—against a GNN-based architecture implemented using PyTorch Geometric. We utilize a large-scale, synthetically [20] generated dataset [6] simulating over 1.3 million transactions and perform evaluations using F1-score to address the challenge of class imbalance. The objectives of this research are:

- To design a heterogeneous graph-based model capturing relationships between transactions, users, and merchants.
- To benchmark the performance of GNNs against state-of-the-art ML classifiers on a realistic fraud detection task.
- To assess the potential of GNNs in modelling contextual information that improves fraud detection accuracy and interpretability.

Our research contributes toward building more robust and scalable fraud detection systems capable of early detection and mitigation of financial risks by integrating transaction context through graph structures

2 Graph Neural Networks

2.1 Introduction to Graph Techniques

Graph Neural Networks (GNNs) have become increasingly popular for modeling data with an intrinsic graph structure. Unlike traditional machine learning

models that treat data points as independent entities, GNNs leverage the relationships between nodes, enabling information to flow across connected entities.

Unlike conventional models that depend only on tabular features, GNNs harness interactions between data records by iteratively aggregating information from neighbouring nodes. This message-passing mechanism allows GNNs to learn rich representations that capture both local and global patterns within the graph structure.

For instance, in financial fraud detection, GNNs can identify suspicious transactions by aggregating information from a transaction's connected entities, such as the customer and merchant. This enables the model to detect fraudulent patterns based on both direct interactions and broader transactional networks.

Several key variants of GNNs exist, including Graph Convolutional Networks (GCNs), HeteroGCNs, Graph Attention Networks (GATs), and GraphSAGE, each utilizing different approaches for message passing and aggregation. These models have been effectively applied across various domains demonstrating their adaptability and efficiency in analyzing structured financial data.

2.2 Types of Graphs

Graphs are mainly categorized among:

- **Homogeneous Graphs:** A homogeneous graph consists of only one type of node and one type of edge. Simpler to train but does not capture different entity roles in the graph. For example. Social Network Analysis: Nodes represent users, edges represent friendships.
- **Heterogeneous Graphs:** A heterogeneous graph consists of multiple types of nodes and edges, capturing more complex relationships. It is complex but captures richer relationships.

  For example. Financial Fraud Detection: Nodes represent Customers, Merchants, Transactions.

2.3 Types of classifications

- **Node Classification**
- **Link Prediction**
- **Graph Prediction**
- **Graph Embedding**

2.4 Graph Structure

To overcome the limitations of traditional graph structures—such as homogeneous graphs where only transactions act as nodes, or models where transactions are merely edges between customer (or card) and merchant nodes, leading to

static attribute reliance and feature loss— I adopted a heterogeneous graph representation with learnable parameters. In this design, transactions are the central nodes, enriched with detailed transaction-specific features. Customers and merchants are also represented as nodes, but unlike transactions, they do not carry inherent features and are instead modeled as learnable embeddings. This structure allows the model to learn rich representations of interactions through different edge types: transaction-to-customer, transaction-to-merchant, and even transaction-to-transaction links. The primary task is to classify transaction nodes, framing the problem as a node classification task. This heterogeneous framework captures complex relational patterns within the data and provides a more expressive and adaptable approach for detecting fraudulent behavior.

2.4.1 Node Representation

In a graph, each node represents an entity (e.g., a user or account in a financial system) and is associated with a feature vector capturing its attributes. Formally, each node i is represented by a feature vector $\mathbf{x}_i \in \mathbb{R}^d$, where d is the dimensionality of the feature space. These vectors may include explicit features (such as account type or transaction count) or be learned embeddings. Each node may also be assigned a label y_i, which could be discrete (e.g., fraud or non-fraud) or continuous.

2.4.2 Edge Representation

Edges in a graph encode the relationships between nodes and can carry additional information. In financial systems, these may represent transactions or shared accounts. Each edge between nodes i and j can have a feature vector $\mathbf{a}_{ij} \in \mathbb{R}^{d'}$, where d' is the dimension of the edge features. These could include transaction amount, type, or frequency.

2.5 Graph Architecture: Core Mechanisms

Graph Neural Networks (GNNs) are powerful architectures designed to learn from graph-structured data by leveraging the relationships and interactions between entities (nodes) and their connections (edges). A typical GNN layer consists of three key operations: **Message Passing**, **Aggregation**, and **Update**. Innovations in graph deep learning often stem from modifications to one or more of these components.

2.5.1 Message Passing

The message passing step propagates information from a node's neighbors. Let $\mathcal{N}_i$ denote the neighborhood of node i. For each neighbor $j \in \mathcal{N}_i$, a message is computed using a transformation function F:

$$\mathbf{m}_{j \to i} = F(\mathbf{h}_j) = \mathbf{W}_j \cdot \mathbf{h}_j$$

These messages encode the neighbor information to be passed to node i.

2.5.2 Aggregation

The incoming messages are then aggregated using a permutation-invariant function G. Common choices include:

$$\text{Sum:} \quad \bar{\mathbf{m}}_i = \sum_{j \in \mathcal{N}_i} \mathbf{m}_{j \to i}$$

$$\text{Mean:} \quad \bar{\mathbf{m}}_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{m}_{j \to i}$$

$$\text{Max:} \quad \bar{\mathbf{m}}_i = \max_{j \in \mathcal{N}_i} \mathbf{m}_{j \to i}$$

This aggregated message $\bar{\mathbf{m}}_i$ summarizes information from the neighborhood of node i.

2.5.3 Update

The node's representation is then updated by combining its current embedding with the aggregated message. Two common strategies are:

Addition-based update:

$$\mathbf{h}_i = \sigma\left(K\left(H(\mathbf{h}_i) + \bar{\mathbf{m}}_i\right)\right)$$

3 Methodology

3.1 Classical Models

For the comparative analysis, I started with several machine learning models - Decision Trees, Random Forest, XGBoost, etc, to serve as baselines for fraud detection. As the data is in tabular format, classical modeling makes sense. Traditionally, these models do a very good job in the majority of cases with tabular records. These models are trained on structured tabular features extracted from transactions, users, and merchants.

Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ be a dataset of N transactions, where:

- $\mathbf{x}_i \in \mathbb{R}^d$ is the feature vector for the i-th transaction
- $y_i \in \{0, 1\}$ is the corresponding binary label indicating whether the transaction is fraudulent ($y_i = 1$) or not ($y_i = 0$)

Features include transaction metadata (amount, city_pop), user (age, lat, long, gender), and merchant (merch_lat, merch_long). All categorical variables are one-hot encoded, and time-based features are encoded using sinusoidal transformations (e.g., $\sin(\cdot)$, $\cos(\cdot)$ for hours/days).

I evaluate the following supervised classification models:

3.1.1 Logistic Regression

Logistic regression models the probability of fraud as:

$$\hat{y} = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x} + b)}} \tag{1}$$

where:

- $\mathbf{x}$ is the input feature vector
- $\mathbf{w}$ are the model weights
- b is the bias term
- $\hat{y}$ is the predicted probability of fraud

The model is trained by minimizing the binary cross-entropy loss:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)] \tag{2}$$

3.1.2 Decision Tree

A decision tree partitions the input space by learning hierarchical splits that maximize information gain (e.g., using Gini impurity or entropy). For each node:

The *Gini impurity* is a commonly used criterion and is defined as:

$$\text{Gini}(S) = 1 - \sum_{c=1}^{C} p(c)^2$$

where:

- C is the number of classes
- $p(c)$ is the proportion of class c in node S

Alternatively, the splitting criterion can be based on *entropy* and *information gain*. The entropy of a node S is given by:

$$\text{Entropy}(S) = -\sum_{c=1}^{C} p(c) \log_2 p(c)$$

The *Information Gain* from splitting on attribute A is:

$$\text{IG}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

where:

- S_v is the subset of S where attribute A has value v
- $|S_v|/|S|$ is the proportion of samples in subset S_v

The decision tree chooses the feature and split that maximizes information gain (or minimizes impurity, in the case of Gini).

3.1.3 Random Forest

Random Forest is an ensemble of T decision trees:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} h_t(\mathbf{x})$$

where each h_t is a decision tree trained on a bootstrapped subset of the data with feature sub-sampling. This helps reduce overfitting and improves generalization.

3.1.4 XGBoost

XGBoost builds an ensemble of additive trees using gradient boosting:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(\mathbf{x}_i)$$

where f_t is the t-th regression tree, and η is the learning rate. The objective minimized is:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{N} l(y_i, \hat{y}_i^{(t)}) + \sum_{t=1}^{T} \Omega(f_t)$$

with regularization:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

3.2 GNN Model

3.2.1 Graph Construction

As the problem statement is to predict fraud transactions, one node type will be transaction. We also have to consider user and merchant because they act as two entities between which transaction occurs. This is why I used Heterogeneous graphs. For each node type we have primary ids. Credit card numbers are used as user id, and we add mappings for both unique transaction and unique merchant. The heterogeneous graph is constructed using PyG's `HeteroData` object. The graph schema is defined as follows:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}_v, \mathcal{T}_e)$$

where:

- $\mathcal{V}$: set of nodes

- $\mathcal{E}$: set of edges
- $\mathcal{T}_v$: set of node types (`transaction`, `user`, `merchant`)
- $\mathcal{T}_e$: set of edge types (relations) described below

Each node type $t \in \mathcal{T}_v$ has an associated feature matrix $X^{(t)} \in \mathbb{R}^{N_t \times d_t}$, where N_t is the number of nodes of type t and d_t is the number of features.

- **Node types**:
  - `transaction` nodes: $X^{(\texttt{transaction})} \in \mathbb{R}^{N_T \times 8}$
  - `user` nodes: $X^{(\texttt{user})} \in \mathbb{R}^{N_U \times 4}$
  - `merchant` nodes: $X^{(\texttt{merchant})} \in \mathbb{R}^{N_M \times 3}$
- **Edge types (bidirectional)**:
  - (`transaction`, `transaction_to_user`, `user`)
  - (`user`, `user_to_transaction`, `transaction`)
  - (`transaction`, `transaction_to_merchant`, `merchant`)
  - (`merchant`, `merchant_to_transaction`, `transaction`)
  - (`user`, `user_to_merchant`, `merchant`)
  - (`merchant`, `merchant_to_user`, `user`

The label for fraud classification is stored on the `transaction` node type as:

$$\mathbf{y}^{(\texttt{transaction})} \in \{0,1\}^{N_{\texttt{transaction}}}$$

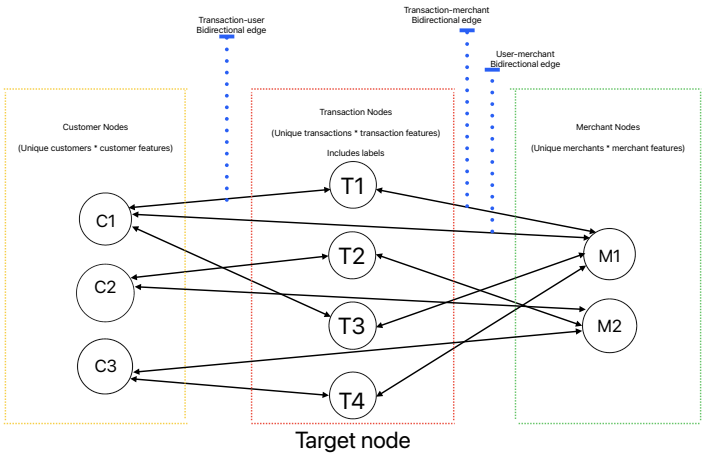


Fig. 1- Heterogeneous Graph Structure

3.2.2 PyG Model Architecture

I implemented a **Heterogeneous Graph Neural Network (HeteroGNN)** using PyTorch Geometric (PyG) for the task of fraud detection. The model performs message passing using type-specific GraphSAGE layers within a `HeteroConv` framework.

- **Input Projection**: Since feature dimensions vary by node type, input projection layers are used to project all features to a shared hidden space of dimension d:

$$h_v^{(0)} = W_{\text{proj}}^{(t)} x_v, \quad \forall v \in \mathcal{V}_t \tag{3}$$

  where:

  - $x_v \in \mathbb{R}^{d_t}$ is the original feature vector of node v of type t
  - $W_{\text{proj}}^{(t)} \in \mathbb{R}^{d_t \times d}$ is the type-specific projection matrix
  - $h_v^{(0)}$ is the initial embedding in the hidden space

- **Message Passing**: I used `HeteroConv` and `SAGEConv` for message passing step available by Pytorch Geometeric

$$h_i^{(l+1)} = \sigma\left(\sum_{r \in \mathcal{R}} \texttt{SAGEConv}_r(h_i^{(l)}, \{h_j^{(l)} : j \in \mathcal{N}_r(i)\})\right) \tag{4}$$

  where -

– `SAGEConv`$_r$: GraphSAGE layer for relation r

– $\mathcal{N}_r(i)$: neighbors of node i via edge type r

– σ: activation function (LeakyReLU)

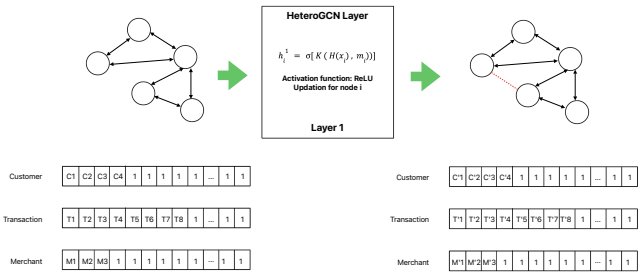
`HeteroConv` sums over messages from different relations using `SAGEConv`

- After each layer, a `LeakyReLU` is applied except the final where Linear Classifier is used to the embeddings of target transaction nodes. The model is trained using the `AdamW` optimizer with L2 weight decay regularization.

$$\hat{y}_i = \text{softmax}(W_{\text{out}} \cdot h_i^{(L)}) \tag{5}$$

- I use the `CrossEntropyLoss`, which combines softmax and log-likelihood internally:

$$\mathcal{L} = -\log\left(\frac{\exp(z_{i,y_i})}{\sum_j \exp(z_{i,j})}\right) \tag{6}$$



3.2.3 DGL Model Architecture

I also designed a Heterogeneous Relational Graph Convolutional Network (HeteroRGCN) using the Deep Graph Library (DGL).

The architecture includes:

- Node types have varying feature dimensions. To unify these, I apply type-specific linear projections to map all features into a common embedding space of hidden size. Input projection layers to align feature dimensions across node types using same equation 1.

- Multiple stacked HeteroRGCN layers that propagate relational information across the graph using mean aggregation. Each GNN layer performs relation-specific message passing. For each edge type (relation) $r \in \mathcal{R}$, I used a unique transformation matrix W_r:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_r(i)} W_r^{(l)} h_j^{(l)}\right) \tag{7}$$

where:

- $\mathcal{N}_r(i)$ denotes the neighbors of node i under relation r
- $W_r^{(l)}$ is the relation-specific transformation for layer l
- $\sigma(\cdot)$ is the activation function (LeakyReLU in our case)

- A final linear classifier to output fraud probabilities for transaction nodes. After L layers, I obtain the final embedding of transaction nodes, which are passed through a linear classifier using equation 3.

  where W_{out} is the classifier weight matrix and $h_i^{(L)}$ is the embedding of transaction node i after L layers. I use cross entropy loss which internally applies softmax.

4 Experimentation and results

4.1 Dataset 1:

4.1.1 Description

This is a simulated credit card transaction dataset containing legitimate and fraud transactions from the duration 1st Jan 2019 - 31st Dec 2020. It covers credit cards of 1000 customers doing transactions with a pool of 800 merchants.

This was generated using Sparkov Data Generation — Github tool created by Brandon Harris. This simulation was run for the duration - 1 Jan 2019 to 31 Dec 2020.

This dataset has following key columns:

- `trans_date_trans_time` - Transaction DateTime
- `cc_num` - Credit Card Number of Customer
- `merchant` - Merchant Name
- `category` - Category of Merchant
- `amt` - Amount of Transaction
- `gender` - Gender of Credit Card Holder
- `city` - City of Credit Card Holder
- `state` - State of Credit Card Holder
- `zip` - Zip of Credit Card Holder

- `lat` - Latitude Location of Credit Card Holder
- `long` - Longitude Location of Credit Card Holder
- `city_pop` - Credit Card Holder's City Population
- `dob` - Date of Birth of Credit Card Holder
- `trans_num` - Transaction Number
- `merch_lat` - Latitude Location of Merchant
- `merch_long` - Longitude Location of Merchant
- `is_fraud` - Fraud Flag / Target Class

4.2 Dataset 2:

4.2.1 Description

This dataset simulates realistic financial transaction patterns and was generated using Python to support the development and evaluation of fraud detection models. It has been carefully designed to emulate a broad spectrum of transactional behaviors across diverse merchant categories, making it well-suited for studying the characteristics that differentiate fraudulent transactions from legitimate ones.

The dataset includes synthetic but lifelike financial transactions occurring in various sectors such as retail, grocery, dining, entertainment, healthcare, travel, and more. It captures a global transaction landscape with varied currencies, countries, and user behaviors, enabling robust exploratory data analysis and machine learning experimentation.

- **Comprehensive Transaction Categories:** Covers multiple domains including online and in-store retail, groceries, restaurants (from fast food to premium dining), gas stations, healthcare services, education platforms, streaming services, gaming, and travel.
- **Geographic and Demographic Diversity:** Contains transactions from various countries and cities, including metadata such as city size and currency type, to enable cross-regional fraud analysis.

This dataset has following key columns:

- `transaction_id`: Unique identifier for each transaction.
- `customer_id`: Unique identifier for each customer.
- `card_number`: Masked card number used in the transaction.
- `timestamp`: Date and time when the transaction occurred.

- `merchant_category`: Broad category of the merchant (e.g., Retail, Travel).
- `merchant_type`: Specific merchant classification within the category (e.g., "online").
- `merchant`: Name of the merchant.
- `amount`: Transaction amount in the local currency.
- `currency`: Currency of the transaction (e.g., USD, EUR).
- `country`: Country where the transaction occurred.
- `city`: City of the transaction.
- `city_size`: Size classification of the city (e.g., small, medium, large).
- `device`: Device used to perform the transaction (e.g., iOS App, Chrome Browser).
- `ip_address`: IP address logged during the transaction.
- `distance_from_home`: Binary indicator if the transaction was outside the home country.
- `high_risk_merchant`: Indicates if the transaction was with a high-risk merchant category.
- `weekend_transaction`: Boolean indicating if the transaction occurred on a weekend.
- `is_fraud`: Label indicating if the transaction is fraudulent (`True`) or legitimate (`False`).

4.3 Preprocessing

To ensure that the dataset is suitable for training machine learning and graph-based models, we apply a dedicated preprocessing pipeline. The pipeline transforms raw transactional data into a structured format with clean, informative, and numerically encoded features. The preprocessing pipeline takes dataframe as input and runs through to implement important processing steps to yield substantial features required for training.

Temporal Feature Engineering

The function first converts the timestamp column using the `pd.to_datetime()` method, from which temporal components are extracted:

- `hour`: Hour of the day
- `day`: Day of the month

To retain the cyclic nature of time (e.g., 23:00 and 00:00 are temporally close), sine and cosine transformations are applied:

$$\text{hour\_sin} = \sin\left(\frac{2\pi \cdot \text{hour}}{24}\right) \tag{8}$$

$$\text{hour\_cos} = \cos\left(\frac{2\pi \cdot \text{hour}}{24}\right) \tag{9}$$

$$\text{day\_sin} = \sin\left(\frac{2\pi \cdot \text{day}}{31}\right) \tag{10}$$

$$\text{day\_cos} = \cos\left(\frac{2\pi \cdot \text{day}}{31}\right) \tag{11}$$

These transformations enable models to interpret proximity between temporal values that wrap cyclically.

Categorical Encoding

Categorical columns, such as `category`, are encoded using one-hot encoding, label encoding and target encoding.

Entity ID Mapping

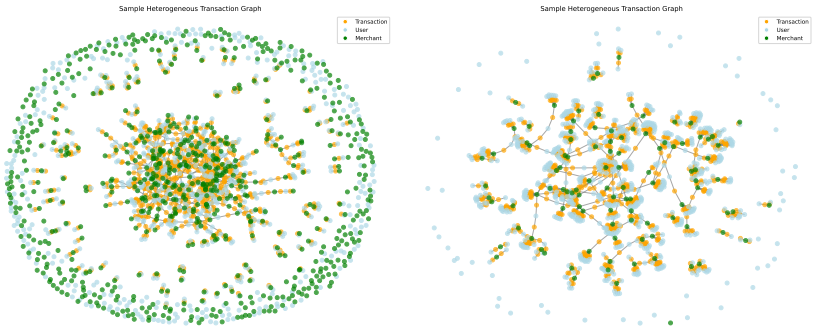
To support graph-based learning, categorical string identifiers for transactions, users and merchants are converted to unique numeric indices:

- `transaction_id` → mapped to sequential integers
- `merchant` → mapped to `merchant_id`
- `cc_num` → mapped to `user_id`

String-based or redundant columns such as `first`, `last`, `street`, `day_period` were dropped to ensure feature relevance and consistency.

4.4 Evaluation Metrics

In highly imbalanced tasks like financial fraud detection, traditional metrics such as accuracy or even ROC-AUC can be misleading. Instead, we focus on F1-score and AUC-PR (Area Under the Precision-Recall Curve), which provide more meaningful insights.



(a) Dataset 1 (b) Dataset 1

Figure 1: PyG Graph

4.4.1 F1-Score

The F1-score balances the trade-off between *precision* and *recall*, and is especially useful when both false positives and false negatives have significant consequences:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

A high F1-score indicates that the model is doing well at both detecting fraud and minimizing false alarms—crucial in financial systems where operational cost and customer experience are impacted by misclassification.

4.4.2 AUC-PR (Precision-Recall AUC)

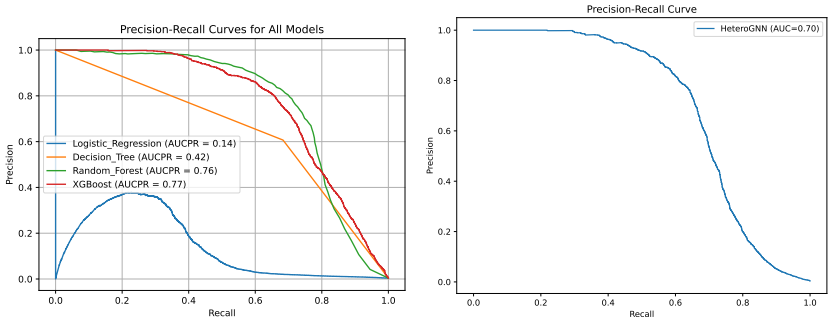
The **AUC-PR** measures the model's ability to correctly identify the **positive class (fraud)** across all thresholds. It is more informative than ROC-AUC in imbalanced datasets where the negative class dominates.

Unlike ROC-AUC, which treats false positives and false negatives equally, AUC-PR focuses on performance related to the *minority class*, which is typically more important in fraud detection.

4.4.3 Results

- **F1-score** reflects the model's balance between fraud detection and false alerts.
- **AUC-PR** captures how well the model distinguishes fraud even when class distribution is highly skewed.
- Together, they provide a reliable evaluation of fraud detection performance, more aligned with real-world cost and risk.

For dataset 1, GNN beats classical models as seen by graphs and scores in the table 1.



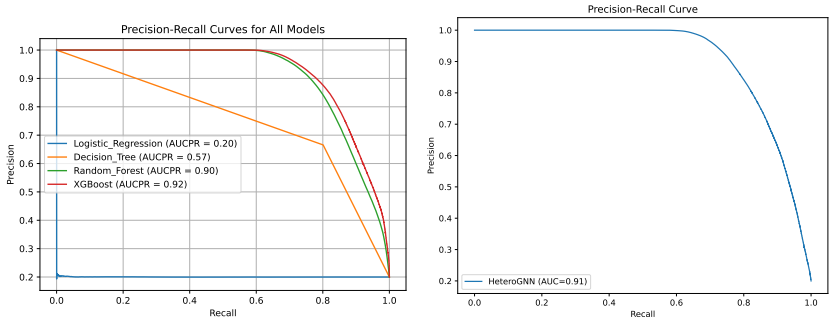
(a) Traditional ML (b) GNN Model

Figure 2: Comparison of AUCPR Curves

Methods	**Precision**	**Recall**	**F1-Score**	**AUCPR**
XGBoost	0.89	0.52	0.66	0.76
Random Forest	0.93	0.52	0.67	0.76
Decision Tree	0.60	0.68	0.64	0.41
Logistic Regression	0.00	0.00	0.00	0.13
GNN	0.73	0.65	**0.68**	0.47

Table 1: Performance Metrics of Different Methods for dataset 1

For dataset 2, GNN beats classical models as seen by graphs and scores in the table 2.



(a) Traditional ML (b) GNN Model

Figure 3: Comparison of AUCPR Curves

Methods	**Precision**	**Recall**	**F1-Score**	**AUCPR**
XGBoost	0.92	0.75	**0.83**	0.92
Random Forest	0.94	0.72	0.81	0.90
Decision Tree	0.66	0.80	0.72	0.57
GNN	0.90	0.75	**0.82**	0.73

Table 2: Performance Metrics of Different Methods for dataset 2

5 Conclusion

In this study, we performed a comprehensive comparative analysis between classical machine learning models and Graph Neural Networks (GNNs) for the task of credit card fraud detection. Our approach leveraged both tabular and graph-based representations of financial transaction data to evaluate the effectiveness of these paradigms across two large datasets.

The results from Dataset 1 demonstrate that the GNN-based model outperforms classical ML models, particularly in settings where explicit features are limited and relational patterns dominate. The ability of graph architecture to exploit structural and contextual cues, through heterogeneous message passing between users, merchants, and transactions, resulted in superior fraud classification performance.

While the GNN still showed strong performance on Dataset 2, classical models like XGBoost and Random Forest slightly outperformed it. A key distinction in this dataset was the availability of numerous high-quality, logically meaningful features that classical models could leverage effectively without relying on inter-node relationships. This contrast highlights a critical insight: GNNs are

most advantageous in environments where feature engineering is constrained but relational data is rich, whereas classical models retain their strength when abundant, well-engineered features are present.

Overall, our findings affirm that GNNs are a powerful and scalable alternative to traditional models, particularly suited for fraud detection systems embedded in relationally complex environments.

References

[1] K. Chaudhary, J. Yadav, and B. Mallick, Dept. of Computer Science, GCET, Greater Noida, India. "A review of fraud detection techniques: credit card," 2012. cloudfront.net/...libre.pdf

[2] Xuan, Shiyang and Liu, Guanjun and Li, Zhenchuan and Zheng, Lutao and Wang, Shuo and Jiang, Changjun. "Random forest for credit card fraud detection," 2018.
https://ieeexplore.ieee.org/abstract/document/8361343

[3] Sailusha, Ruttala and Gnaneswar, V. and Ramesh, R. and Rao, G. Ramakoteswara. "Credit card fraud detection using machine learning," 2020.
https://ieeexplore.ieee.org/abstract/document/9121114

[4] Awoyemi, John O. and Adetunmbi, Adebayo O. and Oluwadare, Samuel A. "Credit card fraud detection using machine learning techniques: A comparative analysis," 2017.
https://ieeexplore.ieee.org/abstract/document/8123782/...

[5] Ismat Samadov, Soonhyeong Kwon, Aadya Singh. "Transactions," 2020.
https://www.kaggle.com/datasets/ismetsemedov/transactions/data

[6] Kartik Shenoy, Brandon Harris. "Credit card transactions fraud detection dataset," 2020.
https://www.kaggle.com/datasets/kartik2112/fraud-detection/data

[7] Yufeng Kou and Chang-Tien Lu and Sirwongwattana, S. and Yo-Ping Huang. "Survey of fraud detection techniques," 2004.
https://ieeexplore.ieee.org/abstract/document/1297040

[8] Abdulghani, Ahmed Qasim and UCAN, Osman Nuri and Alheeti, Khattab M. Ali. "Credit card fraud detection using XGBoost algorithm," 2021.
https://ieeexplore.ieee.org/abstract/document/9719580

[9] Tan, Runnan and Tan, Qingfeng and Zhang, Peng and Li, Zhao. "Graph neural network for ethereum fraud detection," 2021.
https://ieeexplore.ieee.org/abstract/document/9667674

[10] Kim, Hwan and Lee, Byung Suk and Shin, Won-Yong and Lim, Sungsu. "Graph anomaly detection with graph neural networks: current status and challenges," 2022.
https://ieeexplore.ieee.org/document/9906987

[11] C. Leo, "Deep dive into graph neural networks: break down the math, explore message passing mechanisms, and build a GCN from scratch in Python," 2024. [Online]. Available: https://medium.com/@cristianleo120/...

[12] Amazon Web Services AI Shanghai Lablet et al., "Deep graph library package," 2024. [Online]. Available: https://www.dgl.ai/

[13] PyTorch Geometric, "Blogs and tutorials," 2022. [Online]. Available: https://pyg.org/blogs-and-tutorials

[14] Z. Wu et al., "A comprehensive survey on graph neural networks," 2019. [Online]. Available: https://arxiv.org/pdf/1901.00596

[15] R. Anand, "Math behind graph neural networks," 2022. [Online]. Available: https://rish-16.github.io/posts/gnn-math/

[16] Sing Kwan NG and Anthony TAING as part of the Stanford CS224W course project. "Fraud detection with graph attention networks," 2022. https://medium.com/stanford-cs224w/...

[17] Omar Hussein, AI Engineer. "Graph neural networks series," 2022. https://medium.com/the-modern-scientist/...

[18] Amazon. "Build a GNN-based real-time fraud detection solution using Amazon SageMaker, Amazon Neptune, and the deep graph library," 2023. https://aws.amazon.com/blogs/machine-learning/...

[19] Dawei Cheng, Yao Zou, Sheng Xiang, Changjun Jiang; Department of Computer Science and Technology, Tongji University, Shanghai, China; Shanghai Artificial Intelligence Laboratory, Shanghai, China; National Collaborative Innovation Center for Internet Financial Security, Shanghai, China; AAII, University of Technology Sydney, Australia. "Graph neural networks for financial fraud detection: a review," 2024. https://arxiv.org/pdf/2411.05815

[20] Ismat-Samadov. "Synthetic data generator," 2024. https://github.com/Ismat-Samadov/fraud$_d$*etection*/*blob*/*main*/*data*$_g$*en.py*

[21] Cheng, D., Zou, Y., Xiang, S. et al. "Graph neural networks for financial fraud detection: a review," 2025. https://link.springer.com/article/10.1007/s11704-024-40474-yAbs1

[22] Khaled Alarfaj, Fawaz and Shahzadi, Shabnam. "Enhancing fraud detection in banking with deep learning: graph neural networks and autoencoders for real-time credit card fraud prevention," 2025. https://ieeexplore.ieee.org/abstract/document/10689393

[23] Jianian Wang, Sheng Zhang, Yanghua Xiao, and Rui Song, Department of Statistics, North Carolina State University; School of Computer Science, Fudan University. "A review on graph neural network methods in financial applications," 2022. https://arxiv.org/pdf/2111.15367

[24] Dawei Cheng, Yao Zou, Sheng Xiang, Changjun Jiang; Department of Computer Science and Technology, Tongji University, Shanghai, China; Shanghai Artificial Intelligence Laboratory, Shanghai, China; National Collaborative Innovation Center for Internet Financial Security, Shanghai, China; AAII, University of Technology Sydney, Australia. "Graph neural networks for financial fraud detection: A review," 2024. https://arxiv.org/pdf/2411.05815

[25] Doug Steen, Data Analyst, Geologist, Data Science & ML Enthusiast. *Precision-Recall curves*. (2020). `https://medium.com/@douglaspsteen/precision-recall-curves-d32e5b290248`

[26] Liu, GuanJun and Tang, Jing and Tian, Yue and Wang, Jiacun. "Graph neural network for credit card fraud detection," 2021. https://ieeexplore.ieee.org/abstract/document/9736204

[27] Liu, GuanJun and Tang, Jing and Tian, Yue and Wang, Jiacun. "Graph neural network for credit card fraud detection," 2021. https://ieeexplore.ieee.org/abstract/document/9736204