A Report on

Fraud Detection

A Graph Based Approach

Submitted by :
Radhika Raghuwanshi - G32395729
Submitted to:
Dr. Armin Mehrabian
Dr. Sardar Hamidian

Course Code: CSCI6365
Course Name: Advanced Machine Learning

Objective:

The report focuses on leveraging Graph Neural Networks (GNNs), particularly GraphSAGE and Graph Attention Networks (GAT), for detecting fraudulent activities in transactional data. It highlights the critical role of graph-based machine learning models in identifying complex patterns and relationships that traditional methods often overlook.

Dataset Overview:

Dataset:

- **Source**: IBM Credit Card Transactions Dataset.
- **Data Size**:
    - **Total Records**: 24 million transactions.
    - **Entities**:
        - **Merchants**: 6,000 unique merchants.
        - **Cardholders**: 100,000 unique cardholders.
    - **Selected Sample**: Due to computational limitations, a subset of **100,000 transactions** is extracted for this study.

Graph Representation:

- **Nodes**: Represent entities, such as cardholders and merchants.
- **Edges**: Represent transactions between nodes with attributes like amount, timestamp, and transaction type.
- **Features**:
    - Node features include merchant type, cardholder profile data, and transaction history.
    - Edge features include transaction amount, timestamp, and location.

Data Preprocessing:

a. Data Cleaning

- Missing Values:
    - Checked for null or missing values in key attributes like transaction amount, timestamp, and merchant information.
    - Imputed or removed records with incomplete data based on their importance.
- Duplicate Records:
    - Identified and removed duplicate transactions to ensure data integrity.

b. Feature Engineering

- Node Features:
    - Cardholder profiles: Aggregated features such as average transaction amount, frequency of transactions, and types of merchants interacted with.

    - Merchant profiles: Calculated features like average transaction value and frequency of transactions across different cardholders.
- Edge Features:
    - Incorporated transaction-specific attributes, such as amount, timestamp (converted to a numerical format), and geolocation, as edge features.

c. Graph Construction

- Converted the dataset into a graph structure with:
    - Nodes: Cardholders and merchants.
    - Edges: Transactions between cardholders and merchants.
    - Attributes:
        - Node features: Merchant category, transaction history, and cardholder type.
        - Edge features: Transaction amount, timestamp, and transaction type.

d. Encoding Categorical Data

- Categorical features like merchant category and transaction type were encoded using one-hot encoding to make them compatible with the model.

e. Normalization

- Numerical features, such as transaction amount and frequency, were normalized to a range of 0 to 1 for uniformity and to prevent model bias toward larger numerical values.

f. Sampling for Computational Efficiency

- Due to hardware constraints, 100,000 transactions were sampled from the original dataset while ensuring a balanced representation of fraudulent and non-fraudulent transactions.

Baseline Model: Graph Neural Network (GNN)

In this project, we began by implementing a **Graph Neural Network (GNN)** model as our baseline to assess the potential of graph-based models in fraud detection. The GNN approach served as a simple reference model for graph-based learning before exploring more sophisticated models like **Graph Attention Network (GAT)** and **GraphSAGE**.

Model Implementation:

1. Graph Neural Network (GNN)

The GNN was implemented as the baseline model to aggregate information from neighboring nodes and learn node representations.

Implementation Steps:

- **Input Representation:**
    - Nodes: Cardholders and merchants.
    - Edges: Transactions with attributes like amount and timestamp.
- **Graph Layers:**

    - Used a Graph Convolutional Network (GCN) layer to compute node embeddings by aggregating information from neighboring nodes.
    - Updated node features iteratively to incorporate local neighborhood structure.
- **Activation Function:**
    - Applied ReLU to introduce non-linearity after each layer.
- **Loss Function:**
    - Binary cross-entropy loss to predict whether a transaction is fraudulent.
- **Optimizer:**
    - Adam optimizer for efficient gradient updates.

Model Training:

- The model was trained over multiple epochs with a batch size of transactions to minimize the loss function. Early stopping was used to prevent overfitting

2. Graph Attention Network (GAT)

GAT introduced attention mechanisms to assign importance scores to neighbors, allowing the model to focus on significant interactions.

Implementation Steps:

- **Attention Mechanism:**
    - Used multi-head self-attention to compute importance scores for each node's neighbors.
    - Attention coefficients were normalized using a softmax function.
- **Layer Structure:**
    - A GATConv layer was implemented to aggregate weighted information from neighbors.
- **Node Embeddings:**
    - Combined information from neighbors based on their attention scores.
- **Activation and Dropout:**
    - Applied LeakyReLU for activation and dropout for regularization to prevent overfitting.
- **Loss and Optimization:**
    - Same as in the GNN, with binary cross-entropy loss and Adam optimizer.

Key Advantage:

- GAT allowed the model to prioritize specific relationships, such as frequent or high-value transactions, while minimizing the influence of less relevant connections.

3. GraphSAGE

GraphSAGE was used to address scalability issues by sampling fixed-size neighborhoods for aggregation.

Implementation Steps:

- **Neighborhood Sampling:**
    - Instead of using the full neighborhood, sampled a fixed number of neighbors for each node during aggregation.
- **Aggregation Functions:**
    - Tested mean, max, and LSTM-based aggregation methods to learn embeddings from sampled neighbors.
- **Layer Stacking:**
    - Stacked multiple GraphSAGE layers to capture higher-order neighborhood information.
- **Node Embeddings:**
    - Generated node embeddings by combining the node's features with its aggregated neighbors' features.
- **Loss and Optimization:**
    - Same as previous models, focusing on detecting fraudulent transactions.

Model Evaluation:

In this section, we will evaluate the performance of the models used for fraud detection: **Graph Neural Network (GNN)**, **Graph Attention Network (GAT)**, and **GraphSAGE**. We will use various metrics such as precision, recall, F1 score, and accuracy to assess the performance of each model on the fraud detection task. It's important to note that the numbers presented may not seem high, but they are competitive within the context of fraud detection tasks on similar datasets. In fact, the highest precision achieved by other researchers on this dataset has been around **0.21**. Thus, the models in this study show competitive performance for fraud detection, though there remains significant room for improvement, especially in terms of precision and overall model optimization.

5.1 GNN Model Evaluation

- **Precision**: 0.1071
- **Recall**: 0.1790
- **F1 Score**: 0.1340
- **Correct Fraud Predictions**: 179/1000

The **GNN** model achieved a precision of **0.1071**, which indicates that only a small portion of the transactions predicted as fraud were actually fraudulent. The recall was **0.1790**, meaning the model identified only about **18%** of all fraud cases. While the F1 score was **0.1340**, the overall performance of the model could be improved significantly, as evidenced by the relatively low precision and recall.

5.2 GAT Model Evaluation

- **Precision**: 0.0993
- **Recall**: 0.4637
- **F1 Score**: 0.1636
- **Correct Fraud Predictions**: 474/1000

The **GAT** model performed better in terms of **recall** (**0.4637**) than the **GNN** model, which indicates it was better at detecting fraudulent transactions. The **precision** of **0.0993** is still low, suggesting a large number of non-fraudulent transactions were incorrectly classified as fraud. The **F1 score** was **0.1636**, indicating a balance between precision and recall, though the model still has room for improvement.

5.3 GraphSAGE Model Evaluation

- **Precision**: 0.1051
- **Recall**: 0.3467
- **F1 Score**: 0.1613
- **Correct Fraud Predictions**: 698/2013

The **GraphSAGE** model achieved a **recall** of **0.3467**, which is better than the **GNN** model but lower than the **GAT** model. Despite the lower recall, **GraphSAGE** correctly identified **698 fraud cases** out of **2013** total fraud cases in the dataset, indicating it detected a larger number of fraud cases overall. The model's **precision** was **0.1051**, which is similar to the GNN model but lower than GAT's precision.

Model	Precision	Recall	F1 Score	Correct Fraud Predictions
GNN	0.1071	0.1790	0.1340	179/1000
GAT	0.0993	0.4637	0.1636	474/1000
GraphSAGE	0.1051	0.3467	0.1613	698/2013

Challenges:

1. Large Dataset

The IBM Credit Card Transactions dataset contains **24 million transactions**, which poses significant computational challenges. Due to limited system resources, processing such a large dataset was not feasible. As a result, we had to work with a sample of **100,000 records** instead of the full dataset.

2. Imbalanced Classes

One of the most significant challenges in fraud detection is the **class imbalance** between fraudulent and non-fraudulent transactions. In the dataset, fraudulent transactions are much less frequent than non-fraudulent transactions, making it difficult for the models to detect fraud effectively. The **imbalanced class distribution** can lead to models that are biased toward predicting the majority class (non-fraudulent transactions), resulting in high false negative rates (i.e., failing to detect fraudulent transactions).

To address this issue, techniques such as **oversampling** the minority class (fraudulent transactions), **undersampling** the majority class (non-fraudulent transactions) and using **class weights** in the model were explored.

Conclusion:

In this study, we evaluated three different Graph Neural Network (GNN)-based models—GNN, GAT, and GraphSAGE—for fraud detection using the IBM Credit Card Transactions dataset. Although the performance of these models may not seem particularly high in terms of precision, recall, and F1 score, the results are competitive when compared to the state-of-the-art in similar tasks, where the highest precision achieved on this dataset has been around 0.21.

GAT (Graph Attention Network) demonstrated the best performance in identifying fraudulent transactions, with the highest recall (0.4637) among the models. This can be attributed to the attention mechanism used in GAT, which likely allowed the model to focus more on the fraud class during training. The attention mechanism in GAT enables the model to assign varying importance to different neighbors in the graph, which may have helped it to better capture the underlying structure of fraudulent transactions and enhance its ability to detect fraud. Although GAT had a slightly lower precision (0.0993) compared to GraphSAGE, its ability to focus on the fraud cases may have made it more sensitive to the imbalanced nature of the dataset, where fraudulent transactions are much less frequent than non-fraudulent ones.

GraphSAGE, on the other hand, showed a reasonable trade-off between precision and recall. With a recall of 0.3467 and a precision of 0.1051, GraphSAGE detected a larger number of fraud cases overall (698 fraud cases out of 2013) compared to GAT's 474 fraud cases. The key strength of GraphSAGE lies in its ability to aggregate features from a node's local neighborhood, which helps the model learn better representations of nodes in the graph, including the fraud cases. While GraphSAGE did not outperform GAT in recall, its performance showed that neighborhood aggregation can also be a powerful approach for capturing fraud-related patterns in transaction data.

GNN (Graph Neural Network) performed the weakest among the three models, with the lowest recall (0.1790) and a precision of 0.1071. While GNN can still capture some structural information from the graph, it may lack the specialized focus on fraud detection that GAT's attention mechanism provides. This likely resulted in GNN's lower ability to detect fraud compared to GAT and GraphSAGE.