Banking Fraud Detection Using GNN

Thesis to be submitted in partial fulfillment of the requirements for the degree

of

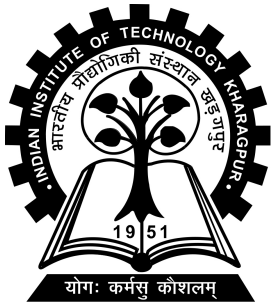
Master of Technology

by

Priyank Mundra
20EC39024

Under the guidance of

Prof. Pabitra Mitra and Prof. Arijit De



ELECTRONICS AND ELECTRICAL COMMUNICATION ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Department of Electronics and Electrical
Communication Engineering
Indian Institute of Technology, Kharagpur
India - 721302

CERTIFICATE

This is to certify that we have examined the thesis entitled **Banking Fraud Detection Using GNN**, submitted by **Priyank Mundra** (Roll Number: *20EC39024*) an undergraduate student of **Department of Electronics and Electrical Communication Engineering** in partial fulfillment for the award of degree of Master of Technology. We hereby approve it as a study carried out and presented in a manner required for its acceptance in partial fulfillment of the postgraduate degree for which it has been submitted. The thesis has met all the requirements according to the Institute regulations and reached the required submission standard.

Co-Supervisor
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Supervisor
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Date: 28th November 2024
Place: Kharagpur

ACKNOWLEDGEMENTS

I express my profound gratitude to all those who have contributed to the completion of this thesis. First and foremost, I would like to extend my heartfelt thanks to my supervisor, Prof. Pabitra Mitra, and my co-supervisor Prof. Arijit De for their unwavering support, insightful guidance, and invaluable expertise throughout this research journey. Their dedication and encouragement played a pivotal role in shaping the direction of this study.

I am deeply grateful to the faculty members of the Indian Institute of Technology, Kharagpur, whose knowledge and feedback enriched my understanding and inspired me to delve deeper into the intricate domain of Deep Learning.

My sincere appreciation goes to my colleagues and friends who provided endless encouragement, engaging discussions, and moral support. Your camaraderie made this academic endeavor a truly enriching experience. I am indebted to my family for their unconditional love, understanding, and encouragement. Their unwavering belief in my abilities kept me motivated during challenging times.

Lastly, I extend my gratitude to the academic community, researchers, and authors whose work laid the foundation for this study. Your contributions have been invaluable in shaping my perspective and improving the quality of this research.
This thesis stands as a testament to the collective support and inspiration I have received, and I am profoundly grateful for each individual who has been part of this academic endeavor.

Priyank Mundra
Indian Institute of Technology, Kharagpur
Date: 28th November 2024

ABSTRACT

Financial fraud detection has become increasingly critical as transaction networks become more complex. This study explores the application of **Graph Neural Networks** (GNNs), specifically **Graph Attention Networks** (GATs), to identify fraudulent activities in banking transactions. Traditional fraud detection methods often fail to capture intricate relationships within transactional data, leading to suboptimal performance. In this project, financial transactions are represented as a graph, where accounts serve as nodes and transactions as edges, allowing the model to utilize relational information for improved fraud detection.

To address challenges such as data imbalance and feature heterogeneity, extensive preprocessing and feature engineering were conducted, ensuring effective representation of account and transaction attributes. The GAT model dynamically assigns attention weights to neighboring nodes, enabling the identification of suspicious patterns in complex networks. The results demonstrate the model's ability to achieve a True Positive Rate of 69% and a precision of 93%, indicating its robustness in detecting fraudulent transactions while minimizing false positives.

The findings highlight the potential of graph-based learning techniques in financial fraud detection, offering a scalable and accurate solution for real-world applications. Future work aims to enhance model sensitivity, classify transaction trails, and reconstruct incomplete transaction paths to uncover complex fraud schemes.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview of Financial Fraud and Money Laundering

Financial fraud and money laundering represent significant threats to the global economy, costing billions annually and undermining the integrity of financial institutions. Financial fraud refers to any deliberate act intended to deceive for financial gain, while money laundering involves concealing the origins of illicit funds to make them appear legitimate. In recent years, the rapid digitization of financial services has increased the complexity and volume of transactions, creating new avenues for fraudulent activities. Transactions now happen at high volumes and speeds, often across borders and currencies, making it challenging for traditional monitoring systems to keep up.
Due to the prevalence of these crimes, financial institutions, governments, and regulatory bodies have prioritized fraud detection and prevention strategies. Despite substantial investment in anti-fraud technologies, conventional methods, such as rule-based systems, often fall short. Fraudsters continuously adapt their methods, and static rules are ill-suited to capture sophisticated, evolving fraud tactics. This gap highlights the need for advanced data-driven techniques that can dynamically detect fraud within complex transaction patterns.

1.2 Challenges in Fraud Detection within Financial Transactions

Detecting fraudulent behavior in financial transactions presents multiple challenges.

1. First, fraud detection models must operate in highly imbalanced datasets, as the proportion of fraudulent transactions is typically minute compared to legitimate ones. This imbalance often causes models to favor majority classes, leading to a higher rate of false negatives, where fraudulent transactions go undetected.

2. Second, financial transactions are frequently interconnected, with money flows and relationships linking various accounts. These connections can form intricate patterns, which can be

challenging to analyze and interpret using traditional machine learning techniques that only consider isolated data points.

3. Another core challenge is adaptability. Fraudsters continuously modify their tactics to circumvent detection systems, often exploiting weak points in the detection process. As a result, any effective detection system must be able to learn from evolving patterns and adapt to new fraud techniques.

Additionally, the need for real-time or near-real-time detection adds a layer of complexity, as large transaction datasets must be processed efficiently to identify anomalies without disrupting the flow of legitimate transactions.

1.3 Fraud Techniques Using Multiple Accounts

1. **Multiple Accounts for Smurfing (Money Laundering)**
   Smurfing, also known as "structuring," is a money laundering technique fraudsters use to avoid detection when transferring large sums of illicit money. Rather than moving a large amount in a single transaction, which may raise red flags in anti-money laundering (AML) systems, fraudsters break the transaction into smaller, more manageable amounts. This helps in evading attention from financial institutions and law enforcement agencies that monitor large or unusual transfers. Fraudsters set up multiple accounts either under their name or using fake identities. They then deposit smaller amounts of money into each of these accounts, carefully keeping the amounts below the threshold that would trigger AML systems or suspicion. After distributing the illicit funds across several accounts, the fraudster begins moving the money between these accounts, and eventually, they may transfer the funds to other individuals or offshore accounts. The complexity of the transactions, involving several accounts and intermediaries, makes it harder to trace the origin of the funds. The primary goal of smurfing is to launder illicit funds without attracting attention. By spreading the money across multiple accounts, the fraudster ensures that no single transaction is large enough to trigger a suspicious activity report. This helps them "clean" the money, eventually making it appear legitimate and difficult to trace back to its criminal origin.

2. **Complex Transfers**
   Complex transfers involve moving money through a series of transactions across multiple accounts to confuse investigators and obscure the source and destination of illicit funds. Fraudsters use this method to hide the flow of money, often leveraging multiple financial institutions, payment systems, and even international accounts. The complexity of the transfers creates a difficult-to-follow trail, making it harder for banks and law enforcement to identify fraudulent activity. A fraudster typically begins by transferring funds into an account they control.

From there, they move the money through several intermediary accounts, which may be set up under fake identities or stolen credentials. These intermediary accounts can be spread across various banks or payment systems, both domestic and international. The fraudster might also use remittance services or cryptocurrencies to further obfuscate the transactions. By routing the funds through multiple layers, each transaction appears smaller or unrelated, complicating the investigation and making it more difficult to trace the money back to the fraudster. The objective behind using complex transfers is to create a confusing financial web that makes it difficult for investigators to trace the origin of illicit funds. This method also allows fraudsters to avoid detection by anti-fraud systems that monitor for large or suspicious transactions. By making the transactions appear normal and unconnected, the fraudster can move money freely while reducing the risk of being caught.

3. **Identity Theft and Account Takeovers**
   Fraudsters often use identity theft or account takeovers to gain control of multiple bank accounts, which allows them to conduct fraudulent activities without directly implicating themselves. By stealing personal information or hacking into a victim's account, fraudsters can open accounts in the victim's name or take control of their existing accounts to make unauthorized transactions. In the case of identity theft, fraudsters steal personal information such as Social Security numbers, credit card details, or bank account information. Using this data, they can open multiple accounts in the victim's name or use the information to conduct fraudulent transactions. In account takeovers, fraudsters hack into existing bank accounts, often through phishing or social engineering, to change account credentials. Once in control, they can transfer funds, make purchases, or withdraw money from these accounts. Fraudsters may also use multiple accounts for further transactions, spreading the illicit money across various places to avoid detection. The goal of identity theft and account takeovers is to exploit stolen personal information or hacked accounts to conduct fraudulent activities. By using multiple accounts, fraudsters can cover their tracks, making it more difficult for investigators to trace the funds back to the original source. This technique enables fraudsters to siphon off funds without directly connecting the crime to their identity.

4. **Phishing Scams**
   Phishing scams involve fraudsters tricking individuals into revealing sensitive personal information such as usernames, passwords, and banking details. Through these scams, fraudsters gain access to victims' bank accounts and other financial services. Using this stolen information, they can then access multiple accounts to conduct fraudulent transactions, including transferring money or making unauthorized purchases. Fraudsters carry out phishing attacks by sending emails, text messages, or creating fake websites that mimic legitimate institutions like banks or online payment platforms. These communications often contain a sense of urgency, prompting victims to click on links or download attachments that lead to fraudulent

websites. Once a victim enters their login credentials or personal information, fraudsters gain access to their bank account. The fraudster can then use the account to make unauthorized transactions or transfer funds to other accounts they control. Multiple accounts may be used to transfer the stolen funds, making it harder to trace the transactions. The goal of phishing scams is to deceive individuals into revealing their personal and banking information, which fraudsters can then use to gain unauthorized access to their accounts. By moving money between multiple accounts, fraudsters aim to obfuscate the trail and make it difficult for authorities to trace the fraudulent transactions back to the original source. This technique allows fraudsters to steal funds without being directly connected to the crime.

1.4 Motivation and Objectives of the Project

This project is motivated by the following challenges and seeks to utilize graph-based deep learning methods to improve fraud detection capabilities. By modeling financial transactions as a graph, where nodes represent accounts and edges signify transactional links, the project aims to capture relationships within the transaction network to reveal hidden patterns and suspicious behavior.

This project primarily explores the use of Graph Attention Networks (GATs) to detect fraudulent nodes (accounts) within a banking network. GATs offer several advantages over traditional Graph Convolutional Networks (GCNs) as they can dynamically assign different importance weights to different neighbors, allowing for more nuanced information aggregation. This attention mechanism is particularly valuable in fraud detection, where some transaction relationships may be more indicative of fraudulent behavior than others. GATs are especially effective in this context because they can:

1. Learn to focus on the most relevant neighboring nodes
2. Capture asymmetric relationships in the transaction network
3. Adapt to varying node neighborhoods without being constrained by fixed weights
4. Handle nodes with varying degrees of connectivity more effectively

1.5 Project Contribution and Research Questions

This project contributes a novel approach to fraud detection by integrating graph-based techniques with deep learning, specifically Graph Attention Networks (GATs), to leverage relationships within transaction data. The research addresses several key questions:

- **How can the structural properties of transaction networks be used to improve fraud detection?** By constructing a graph where nodes represent accounts and edges represent transactions, this project explores how the network topology and account interactions reveal fraud patterns.
- **To what extent can GATs improve the accuracy and robustness of fraud detection compared to traditional methods?** The ability of GATs to dynamically weight and aggregate features from neighboring nodes enables them to capture complex dependencies and selectively focus on relevant connections, potentially providing better fraud detection performance than methods that treat all relationships equally.
- **What features and relationships are most indicative of fraudulent activity in a transaction network?** By examining the attention weights assigned to different node attributes (for example, location, transaction count) and edge characteristics (e.g., amount, transaction type), this project investigates which relationships are most relevant for identifying fraud and how their importance varies across different contexts.

1.6 Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs)[4] are a class of neural networks specifically designed for processing graph-structured data, where entities and their relationships can be naturally represented as nodes and edges. Traditional neural networks, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), excel in domains like image and sequence data but struggle with the non-Euclidean structure of graphs, which requires the capacity to handle irregular node connections and varying neighborhood sizes. GNNs address these limitations by leveraging the relational structure in graphs, making them effective for tasks like node classification, link prediction, and graph classification.

1.7 Introduction to Graph Neural Networks (GNNs)

In a GNN[6], each node's feature representation is iteratively updated by aggregating and transforming information from its neighboring nodes. This neighborhood aggregation, or "message-passing" process, enables GNNs to capture local and global structural patterns in the graph, making them well-suited for tasks where connectivity patterns and relationships hold significant meaning, such as fraud detection, recommendation systems, and social network analysis.

A GNN typically consists of multiple layers, where each layer refines the node representations by aggregating information from increasingly distant nodes. After a certain number of layers, each node has an embedding that encodes information from its multi-hop neighborhood, allowing the network to learn contextually enriched representations.

1.7.1 Theoretical Foundation of GNNs

Graph Neural Networks (GNNs) rely on two primary components for their operations: message passing (aggregation) and updating node states. Mathematically, this can be described as follows:

Message Passing (Aggregation Step)

Each node v aggregates information from its neighbors $N(v)$ at each layer l of the GNN. This process is generally represented as:

$$m_v^{(l)} = \text{AGGREGATE}^{(l)}\left(\{h_u^{(l-1)} : u \in N(v)\}\right)$$

Here, $h_u^{(l-1)}$ is the feature vector of a neighboring node u at the previous layer, and $m_v^{(l)}$ represents the aggregated message for node v. The aggregation function can be a simple sum, mean, or a more sophisticated attention-based operation.

Node Update (Transformation Step)

Once the node aggregates information from its neighbors, the aggregated message is combined with the node's previous state to update its feature representation:

$$h_v^{(l)} = \text{UPDATE}^{(l)}\left(h_v^{(l-1)}, m_v^{(l)}\right)$$

This step allows each node to retain its previous information while incorporating new information from its neighbors. The UPDATE function can involve a neural network layer, such as a fully connected layer or non-linear transformation (e.g., a ReLU activation), to improve expressiveness.

After several iterations (or layers), each node's feature vector encodes both its attributes and the structural information of its neighborhood, capturing dependencies and relationships across the graph.

1.7.2 Types of Graph Neural Networks

Several types of GNNs[8] exist, each with different aggregation and update mechanisms tailored to various tasks and graph structures. Below are some prominent GNN architectures:

- **Graph Convolutional Networks (GCNs)**: GCNs apply convolution-like operations on graph data. At each layer, they aggregate and transform features from neighboring nodes, incorporating adjacency information to form node representations. The aggregation in GCNs is generally normalized by the degree of nodes, making it a popular choice for semi-supervised classification tasks on graphs. GCNs are particularly effective in capturing first-order and second-order neighborhood relationships and serve as a core method for banking fraud detection by identifying patterns in transaction networks.

- **Graph Attention Networks (GATs)**: GATs extend GCNs by incorporating an attention mechanism that assigns different importance levels to neighboring nodes during aggregation. This allows the model to prioritize important neighbors, which is beneficial for graphs where not all connections are equally informative. Attention mechanisms help GATs capture more refined relationships, making them suitable for detecting fraud based on varying transaction importance.

- **GraphSAGE**: Unlike GCNs, which use the entire neighborhood for aggregation, GraphSAGE samples a fixed number of neighbors and aggregates features based on the sampled subset. This is computationally efficient for large graphs, making GraphSAGE advantageous in real-time applications and scenarios with resource constraints. It also allows inductive learning, enabling the model to generalize to unseen nodes.

1.8 Graph Attention Networks (GATs) in Banking Fraud Detection

For this project, Graph Attention Networks (GATs) are used as the primary GNN architecture. GATs excel at tasks like node classification and link prediction, which are directly applicable to fraud detection in banking transaction networks. Unlike traditional GCNs, GATs introduce an attention mechanism that allows the model to assign different importance weights to different neighboring nodes, making them particularly effective at identifying suspicious patterns in transaction networks where some connections are more indicative of fraud than others.

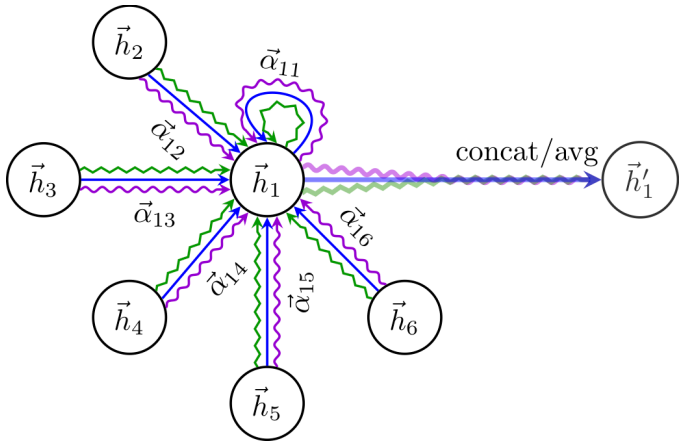


Figure 1.1: Graph Attention Networks: An illustration of multi-head attention (with K = 3 heads)[7]

The architecture of GATs involves multiple graph attention layers, where each layer computes attention coefficients to weigh the importance of neighboring nodes during feature aggregation. This attention-based approach allows the model to:

- Dynamically focus on the most relevant transaction relationships

- Capture asymmetric importance in node relationships
- Learn complex patterns without being constrained by fixed neighborhood weightings

This multi-layer architecture allows the model to capture both immediate and multi-hop relationships among accounts, with the attention mechanism helping to identify the most suspicious trails indicative of laundering or fraudulent activity. The resulting node embeddings, enriched with attention-weighted information from the neighborhood, are used to classify nodes (accounts) as either fraudulent or legitimate.

Chapter 2

Methodology And Result Analysis

The methodology of this project involves several key steps, from data preprocessing to model training and evaluation. This section outlines the process used to convert transactional data into a graph, engineer features, and apply Graph Attention Networks (GATs) to detect fraudulent activity.

2.1 Dataset Description

The dataset utilized in this project serves as the foundation for constructing a graph representation of financial transactions, with each record detailing account interactions in a transactional network. This dataset includes multiple key features that provide insights into transactional behavior, enabling the application of Graph Convolutional Networks (GCNs) to detect fraudulent accounts. The following sections outline the dataset's features, pre-processing steps, and challenges associated with real-world transaction data.

2.1.1 Description of Dataset Features

The dataset contains the following columns:

1. **Time**: Records the transaction timestamp, capturing the exact time of day when the transaction occurred. This is particularly useful for establishing a trial's temporal sequence of transactions.

2. **Date**: Specifies the transaction date, later decomposed into multiple columns for effective feature extraction. The date enables chronological ordering in the transaction network.

3. **Sender_account**: The unique identifier for the account initiating the transaction. This feature is crucial in defining the source node in the graph representation.

4. **Receiver_account**: The unique identifier for the account receiving the transaction. This feature defines the target node in the graph and, combined with the sender account, establishes the directed edges.

5. **Amount**: The monetary value of the transaction. As an edge feature in the graph, the amount indicates the volume of funds transferred between nodes, which can signal suspiciously large or irregular transactions.

6. **Payment_currency** and **Received_currency**: These columns denote the currency used for payment and receipt in the transaction, respectively. Currency information helps highlight cross-border or multi-currency transactions, which could indicate money laundering.

7. **Sender_bank_location** and **Receiver_bank_location**: These features identify the locations of the sender and receiver banks. Geographical location data helps in detecting suspicious flows of money across high-risk areas or tax havens.

8. **Payment_type**: Specifies the transaction method (e.g., Cash Deposit, Online Transfer), offering insights into transaction patterns that may vary depending on fraud typology. Different payment types could correlate with distinct fraud schemes.

9. **Is_laundering**: A binary indicator where '1' denotes a suspicious transaction linked to money laundering, and '0' represents a normal transaction. This serves as the target label for the node classification model.

10. **Laundering_type**: Describes the type of laundering activity, providing additional information for classification and aiding in model interpretability by indicating different fraud typologies.

These features enable the construction of a detailed graph representation where nodes correspond to accounts, and directed edges signify transactions. The features serve either as node attributes (e.g., account-specific attributes like location) or edge features (e.g., transaction-specific attributes like amount and date).

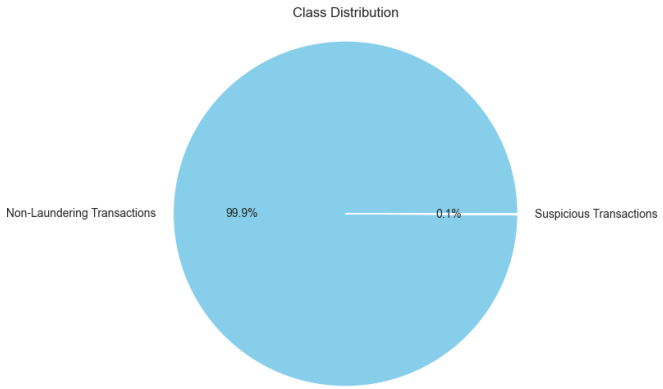


Figure 2.1: Dataset Distribution

The dataset consisted of 95,04,852 transactions with the above-mentioned 12 features. From these transactions, the following is the distribution of laundering and non-laundering transactions.

- 9494979 - Non-laundering(99.9%).
- 9873 - Laundering(0.1%).

Payment_type	Amount	Laundering_Count	Normal_Count
ACH	18272052011.854218	1159	2007648
Cash Deposit	485809045.640000	1405	223801
Cash Withdrawal	46118125.580000	1334	299143
Cheque	18328875956.816402	1087	2010332
Credit card	18308924931.183823	1136	2011773
Cross-border	9476591292.695444	2628	931303
Debit card	18372338763.838520	1124	2010979

Figure 2.2: Distribution of fraud and no-fraud transactions according to payment type

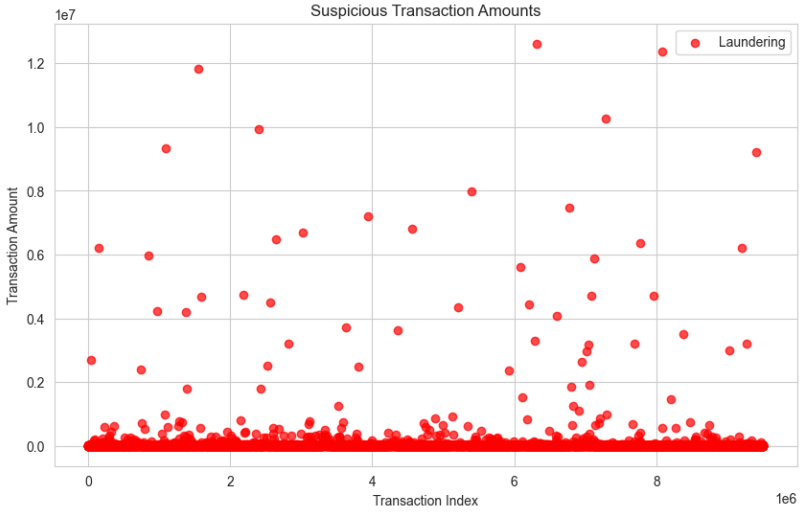


Figure 2.3: Suspicious Transaction Amounts

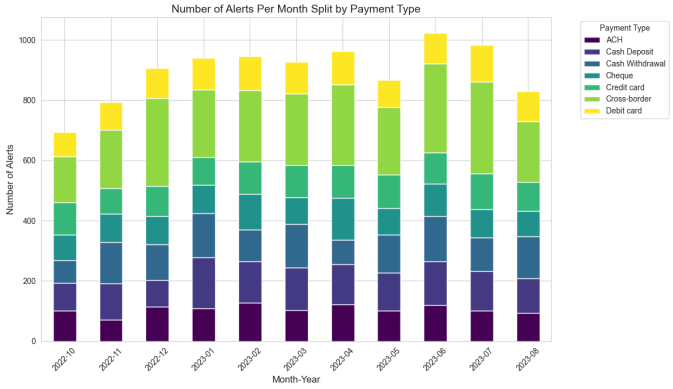


Figure 2.4: Number of Alerts Per Month Split by Payment Type

2.1.2 Notable Pre-processing Steps and Assumptions

To facilitate efficient model training and optimize feature relevance, several pre-processing steps were undertaken. Below are the key steps:

1. **Date Conversion and Expansion**:

   The Date column was converted to a date-time format to enable temporal analysis. From this, additional columns were extracted:

   - **Year**: Represents the year of the transaction.
   - **Month**: Indicates the month of the transaction, capturing seasonal patterns.
   - **Day**: Specifies the day, useful for detecting weekday versus weekend transaction patterns.

2. **Encoding Categorical Features**:

   To allow the model to process categorical features, columns like `Payment_type`, `Sender_bank_location`, and `Receiver_bank_location` were encoded using Label Encoding.

3. **Scaling of Continuous Features**: Continuous features, such as `Amount`, were scaled to bring them into a consistent range, which is essential for models like GCNs that are sensitive to feature magnitudes. Scaling helps in ensuring that high-value transactions do not disproportionately impact model predictions.

4. **Constructing Edge and Node Features**: Transaction amount and date were used as edge features, while sender and receiver location, transaction degree (in-degree and out-degree), and suspicious activity counts were defined as node features. This distinction in feature assignment is crucial for accurately modeling relationships and patterns within the transaction graph.

2.2 Creation of Subset of the Original Dataset

To develop an effective dataset for fraud detection, a subset of the original data was created to address the significant class imbalance in transaction labels. The original dataset contains over **9.4 million transaction records**, classified as follows:

- **Fraud transactions**: 9,873 samples
- **Non-fraud transactions**: 9,494,979 samples

Given the goal of ensuring meaningful representation for accounts involved in fraudulent activity, the non-fraud samples were split into two distinct parts:

1. **First Part**: This subset includes non-fraud transactions where accounts (either as sender or receiver) were also associated with fraudulent transactions. By including these, we establish connections within the data conducive to graph-based analysis, as these accounts represent cases of potential indirect fraud involvement.

2. **Second Part**: This subset contains non-fraud transactions from accounts with no involvement in fraudulent transactions, thus serving as a baseline for typical non-fraud behavior.

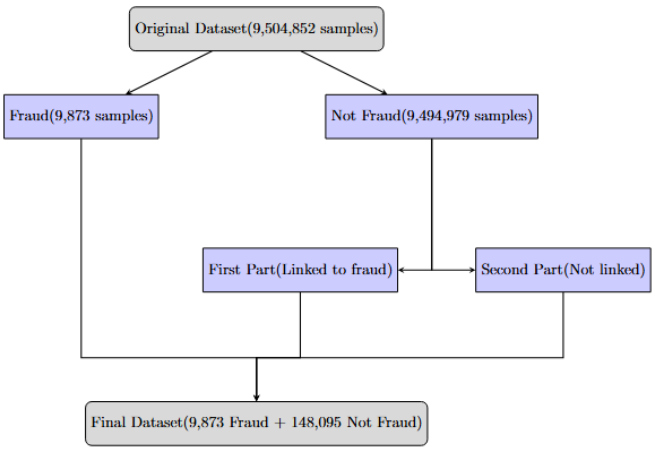


Figure 2.5: Taking subset of Original Dataset

For constructing the final dataset, a subset of samples was chosen to ensure both class representation and manageable dataset size:

- **All 9,873 fraud samples** were retained.
- **148,095 non-fraud samples** were selected, representing 15 times the number of fraud samples.

To achieve a representative non-fraud sample, a stratified selection approach was employed:

- **40%** of the non-fraud samples were sourced from the **First Part** (accounts associated with fraud),
- **60%** of the non-fraud samples were sourced from the **Second Part** (accounts not associated with fraud).

The dataset used for fraud detection was carefully constructed with a graph-based approach. The graph consists of nodes and edges, where nodes represent accounts involved in transactions, and edges represent the transactions between those accounts. The key characteristics of the graph are:

- **Number of nodes**: 138,388
  - Each node represents a unique account involved in a transaction (either as a sender or receiver).
- **Number of edges**: 157,968
  - Each edge represents a transaction between two accounts. The total number of edges matches the size of the dataset, as each sample (whether fraud or non-fraud) corresponds to a transaction between two accounts.

The construction of this graph follows the stratification approach used in the dataset creation. The nodes are connected based on transactions, with fraudulent and non-fraudulent accounts forming edges between them. The graph is used to capture the relationships between accounts involved in fraud or not, enabling more sophisticated graph-based fraud detection techniques.

2.3 Data Preprocessing

Data preprocessing is essential for ensuring that the dataset is clean, consistent, and ready for analysis. The steps taken during preprocessing include:

1. **Data Cleaning**: Missing values for key attributes, such as account location or transaction amount, were either imputed with default values or removed, depending on the relevance of the feature to fraud detection. For instance, default codes were used for missing location or currency data, ensuring that such entries could still be used in the graph.
2. **Feature Extraction**: To capture temporal information, the transaction date was converted into separate year, month, and day columns. Additionally, a continuous representation of the date was created to allow the model to interpret temporal relationships in the data.
3. **Normalization**: Continuous variables, such as transaction amounts, were normalized to improve model stability and performance. This helps the GAT model better handle variations in transaction amounts across different accounts.

2.4 Graph Construction

Constructing a graph from raw transactional data is a foundational step in applying Graph Neural Networks (GNNs) to the task of banking fraud detection. The goal of this process is to convert individual transaction records into a structured graph format, where each node represents an account, and each edge represents a transaction between accounts. This graph structure enables us to model complex relationships within the data, allowing GNNs to capture patterns in account behaviors, transaction flows, and potential fraudulent activity across networks of accounts.

2.4.1 Defining Nodes and Edges

In this project, nodes represent either the sender or receiver accounts involved in the transactions and edges represent individual transactions. This node-edge relationship is defined as follows:

- **Nodes**: Each node corresponds to a unique account involved in one or more transactions. The nodes have several attributes (or features) that capture information about the account, such as location, currency type, transaction frequency, and suspicious behavior count.
- **Edges**: Each edge represents a single transaction from a sender account to a receiver account. Edge features include transaction details like the amount, payment type, and a numeric representation of the transaction date. These edges are directional, where an edge from node A to node B indicates a transaction from account A to account B.

2.4.2 Node and Edge Features

To enable effective learning, each node and edge in the graph is associated with specific features derived from the dataset. These features help represent the unique characteristics of accounts and transactions, which are crucial for identifying suspicious patterns.

- **Node Features**:
  - **Sender Location and Receiver Location**: The geographical locations of the sender and receiver accounts are encoded as numeric values. If a location is unavailable, it is represented by a default value (e.g., 18). If there are multiple locations used by sender then the most frequently used location is taken as final sender location.
  - **Payment Currency and Received Currency**: The currencies involved in the transactions are also encoded numerically. In cases where no currency information is available, a default value (e.g., 12) is used. If multiple currencies are being used by the sender then the most frequently used currency is taken as feature.
  - **Out-degree and In-degree**: The out-degree of a node represents the number of transactions sent from the account, while the in-degree represents the number of transactions received by the account. These values are indicative of transaction frequency and can signal potential red flags if they fall outside typical ranges.
  - **Suspicious Count**: A numeric feature that counts the number of laundering transactions associated with a given account, based on previous knowledge or flagged transactions.
- **Edge Features**:

- **Transaction Amount**: The transaction amount is represented as a continuous feature, capturing the scale of each transaction.
- **Payment Type**: This categorical feature encodes the type of transaction, such as cash deposit, transfer, or online payment. This feature can help distinguish between typical and suspicious transaction types.
- **Date Representation**: The date of each transaction is converted to a continuous value representing the year, month, and day as a numeric date format:

$$\text{Year} + \frac{\text{Month}}{12} + \frac{\text{Day}}{365}$$

This format helps capture the timing of transactions in a compact and computationally efficient way, enabling the model to recognize patterns over time.

2.4.3 Constructing Transaction Trails

Fraudulent behavior is often identified not from individual transactions but from patterns across multiple transactions. To capture this, transaction trails are created within the graph, where a sequence of transactions (edges) connects a series of accounts (nodes). For example, if Account A sends money to Account B, which in turn sends money to Account C, a transaction trail is formed as:

$$A \rightarrow B \rightarrow C \rightarrow ...$$

This enables the model to learn dependencies along transaction paths, helping identify money-laundering chains and other fraudulent trails. Longer transaction trails are particularly useful for detecting complex schemes, as they reveal intricate transaction relationships and account dependencies.

2.5 Feature Engineering

Feature engineering focused on enhancing both node and edge features to maximize the model's ability to detect fraud:

- **Node Features**: Key features included account location, transaction count (indegree and outdegree), currency type, and a suspicious count that indicates the number of laundering-related transactions associated with an account.
- **Edge Features**: Edge features were designed to capture transaction-specific details, such as transaction amount, payment type, and a continuous date representation.

These engineered features enrich the graph with critical information that helps the GAT capture both account-level and transaction-level patterns.

2.6 Model Selection: Graph Attention Networks (GATs)

Graph Attention Networks (GATs) were chosen for this project because of their effectiveness in learning from graph-structured data and their ability to assign different importance to different nodes in the neighborhood. Unlike traditional GCNs, which treat all neighbors equally, GATs apply attention mechanisms to weigh the contributions of different neighbors dynamically. Key components of the GAT architecture include:

- **Attention-based Message Passing**: Each node computes attention coefficients with its neighbors and uses these to weigh the importance of different nodes during feature aggregation. This enables the model to focus on the most relevant connections and identify suspicious patterns more effectively.
- **Multi-head Attention**: Multiple attention mechanisms run in parallel, allowing the model to capture different aspects of node relationships simultaneously and stabilize the learning process.
- **Node Update**: Nodes update their representations based on the attention-weighted aggregated information, allowing each node's representation to evolve through the network layers while preserving the most relevant neighborhood information.
- **Self-attention Mechanism**: Each node can attend to its own features, helping preserve important local information while incorporating neighborhood context.

2.7 Model Architecture

2.7.1 Graph Attention Network Model

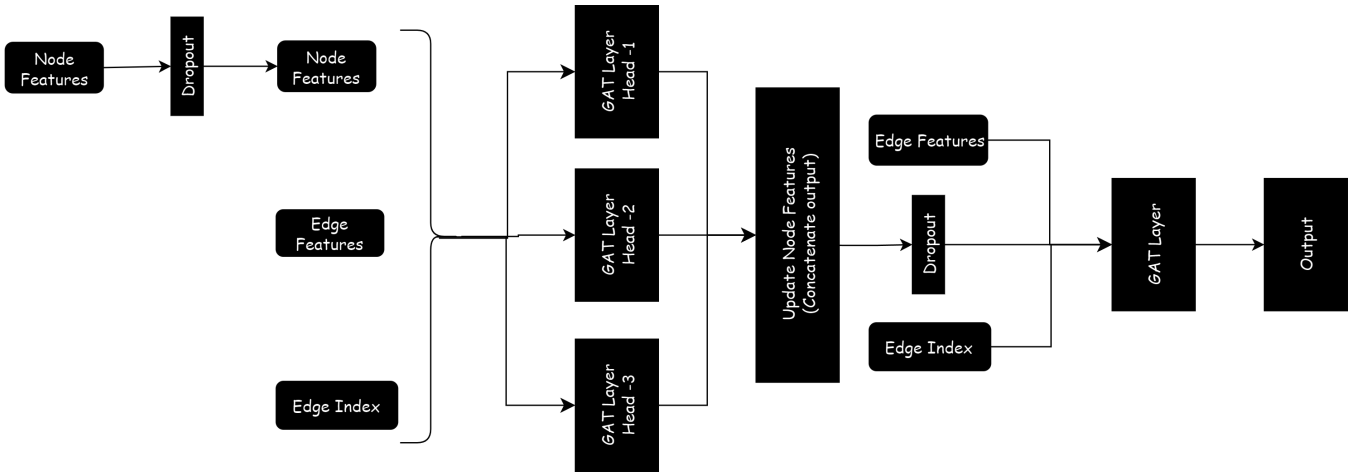


Figure 2.6: Graph Attention Network without MLP layer for edge feature transaformation

2.7.2 Edge Enhanced Graph Attention Network Model

EdgeEnhanced GAT Model: Theory

The EdgeEnhanced GAT (Graph Attention Network) model extends traditional GAT architectures by incorporating **edge feature transformations**, enabling it to utilize both node and edge features for enhanced graph representation learning. While standard GAT models focus on node features for attention computation, EdgeEnhanced GAT processes edge features through a **multi-layer perceptron (MLP)**.

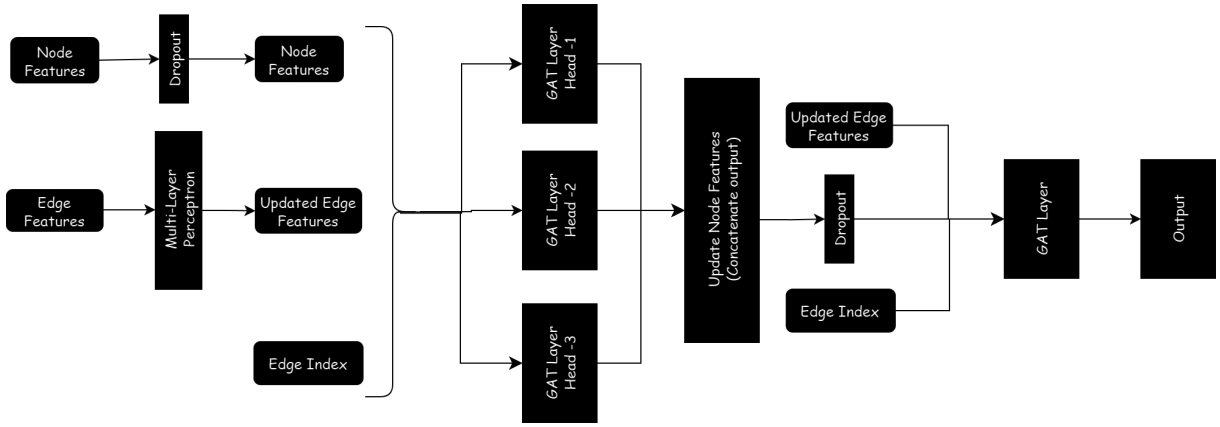


Figure 2.7: Edge Enhanced Graph Attention Network[3]

The model operates on three inputs: **node features**, **edge indices**, and **edge features**. Edge features are first transformed by the MLP and then integrated into the GAT layers, which compute dynamic attention scores based on both node and edge information. It employs two GAT layers: the first uses **multi-head attention** to capture diverse relational patterns, while the second uses **single-head attention** for refined feature aggregation.

2.8 Training and Evaluation

Once the graph was constructed and features were engineered, the GCN model was trained to classify nodes (accounts) as either "fraudulent" or "non-fraudulent." The training process included:

1. **Train-Test Split**: The dataset was divided into training and test sets, with careful consideration to ensure the split maintains meaningful relationships within the graph.

2. **Loss Function and Optimization**: A cross-entropy loss function was used for the binary classification task and gradient descent was applied for optimization.

3. **Evaluation Metrics**: The model's performance was evaluated using precision and TPR (True Positive Rate) to assess the model's ability to identify fraudulent accounts.

Chapter 3

Result Analysis

3.1 Evaluation Metrics

- **Precision**: The proportion of true positive fraud predictions among all predicted fraud cases, indicating the model's effectiveness in minimizing false positives.
- **True Positive Rate (TPR)**: Also known as sensitivity or recall, TPR is the ratio of correctly identified fraudulent transactions (true positives) to the total actual fraud cases (true positives plus false negatives). This metric indicates the model's effectiveness in capturing fraud cases and is critical in fraud detection scenarios, where missing even a small percentage of fraud cases can have significant financial implications.
- **Confusion Matrix:** The confusion matrix visually represents the model's classification results by breaking down correctly and incorrectly classified samples for fraud and non-fraud transactions. In this matrix:
  - **True Positives (TP)**: Fraud transactions that are correctly identified as fraud.
  - **True Negatives (TN)**: Non-fraud transactions that are correctly identified as non-fraud.
  - **False Positives (FP)**: Non-fraud transactions that are incorrectly identified as fraud.
  - **False Negatives (FN)**: Fraud transactions that are incorrectly identified as non-fraud.

  This breakdown provides insight into the model's effectiveness in detecting fraud while minimizing false positives and false negatives.

3.2 Model's Performance on Subset of Dataset

The analysis of the model's performance on the created dataset 2.2, including key metrics such as precision, recall, and true positive rate (TPR).

3.3 Model Configuration and Class Weights

To address the class imbalance inherent in fraud detection tasks, specific class weights were applied to prioritize accurately classifying minority classes. The class weights used in this model are:

- **Non-fraud (Class 0)**: 0.0521
- **Fraud (Class 1)**: 0.96

Model Parameters

The model architecture is configured with the following parameters:

- **Input Channels**: 7
- **Hidden Channels**: 16
- **Output Channels**: 2 (binary classification: fraud vs. non-fraud)
- **Attention Heads**: 3

These parameters were selected to optimize feature representation while maintaining an efficient and interpretable structure suitable for fraud detection tasks.

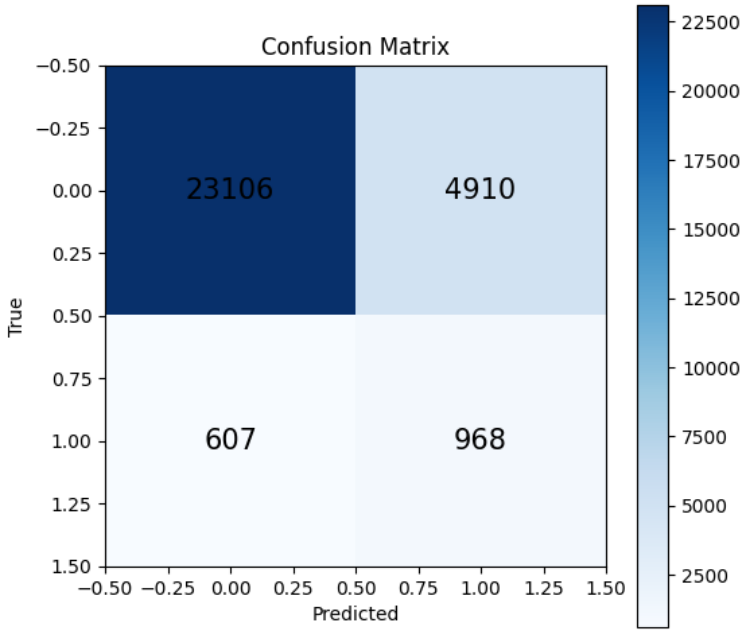


Figure 3.1: Confusion Matrix of GAT Model

The confusion matrix, shown in Fig3.1, summarizes the model's predictions:

- **True Negatives (TN)**: 23,106 – Non-fraud transactions correctly identified as non-fraud.
- **False Positives (FP)**: 4,910 – Non-fraud transactions incorrectly classified as fraud.
- **False Negatives (FN)**: 607 – Fraud transactions incorrectly classified as non-fraud.
- **True Positives (TP)**: 968 – Fraud transactions correctly classified as fraud.
- **True Positive Rate (TPR)**

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{968}{968 + 607} \approx 0.61$$

  This means that the model successfully identifies approximately 61% of the fraudulent transactions in the dataset.

- **Precision**: 0.93

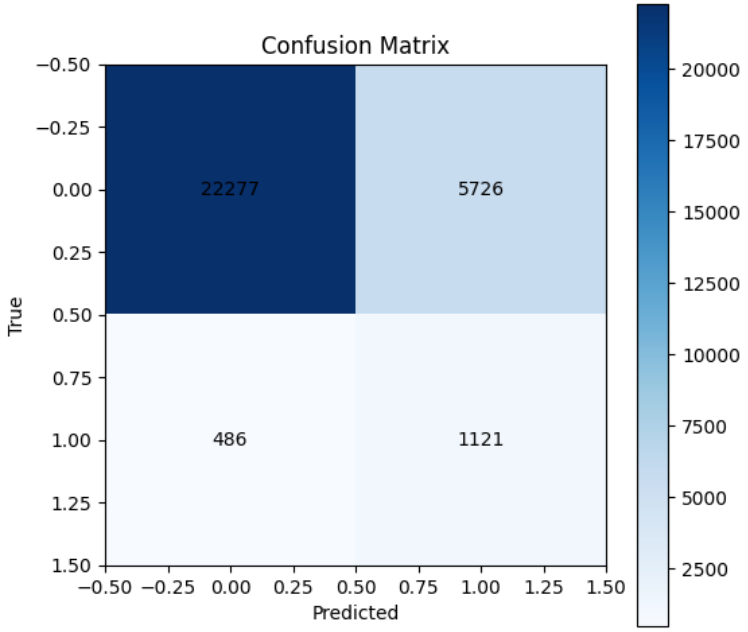


Figure 3.2: Confusion Matrix of Edge Enhanced GAT Model

The confusion matrix, shown in Fig3.2, summarizes the model's predictions:

- **True Negatives (TN)**: 22,277 – Non-fraud transactions correctly identified as non-fraud.

- **False Positives (FP)**: 5,726 – Non-fraud transactions incorrectly classified as fraud.
- **False Negatives (FN)**: 486 – Fraud transactions incorrectly classified as non-fraud.
- **True Positives (TP)**: 1,121 – Fraud transactions correctly classified as fraud.
- **True Positive Rate (TPR)**

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{1121}{1121 + 486} \approx 0.69$$

  This means that the model successfully identifies approximately 69% of the fraudulent transactions in the dataset.

- **Precision**: 0.93

3.4 GAT vs. EdgeEnhanced GAT

In the GAT model, edge features were incorporated directly into the attention mechanism, but without the use of a multi-layer perceptron (MLP) to process them. This model achieved a True Positive Rate (TPR) of 61% for detecting fraudulent transactions. However, the EdgeEnhanced GAT model, which also utilizes edge features but includes an MLP to transform them before passing them into the attention mechanism, led to a notable improvement in TPR, increasing it to 69%. While precision remained unchanged, the accuracy of the EdgeEnhanced GAT model slightly decreased, suggesting that the introduction of the MLP layer enabled the model to better capture intricate patterns in the data, improving its fraud detection capabilities but at the cost of overall classification performance.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

This project successfully tackled the challenge of detecting fraud in financial transaction networks by leveraging the power of Graph Neural Networks (GNNs), specifically Graph Attention Networks (GATs). Due to the complexity of financial transaction networks and the highly imbalanced nature of fraud datasets, a comprehensive approach was required to model transactions as graphs, where nodes represent accounts and edges represent transactional links.

During the project, significant efforts were made to preprocess the data, engineer relevant features, and construct a robust graph representation of the financial transactions. The GAT model was able to dynamically assign different attention weights to neighboring nodes, allowing it to capture intricate relationships in the transaction network and better distinguish between legitimate and fraudulent transactions.

However, to further improve model performance, additional measures were implemented. Due to the computational constraints, a new, more manageable dataset was created. Additionally, class weights were introduced to address the class imbalance, which led to an improvement in the model's ability to detect fraudulent transactions, resulting in an initial True Positive Rate (TPR) of approximately 60%.

To enhance the model's performance further, a normal neural network was incorporated to strengthen the edge transformations used in the GAT. This improvement significantly boosted the model's attention mechanism, enabling it to better prioritize important transaction relationships and detect fraud more effectively. As a result, the TPR increased to 69%, demonstrating a marked improvement in identifying fraudulent transactions while maintaining high precision (93%), thus reducing the number of false positives.

In conclusion, the integration of Graph Attention Networks, along with the use of advanced feature engineering, class balancing techniques, and neural network enhancements, has shown promising results for financial fraud detection. The model effectively balances the need for identifying fraudulent activities while minimizing false alerts, making it suitable for real-world deployment in detecting fraud in financial systems.

4.2 Future Work

Despite the promising results achieved in this project, several avenues for improvement and expansion remain. The current model demonstrates strong performance in detecting individual fraudulent transactions but can be further refined to address specific challenges and expand its scope.

- **Improving True Positive Rate:** The current model's True Positive Rate (TPR) indicates room for improvement in capturing even more fraudulent transactions. Future work will explore different GNN architectures, such as Graph Convolutional Networks (GCNs), GraphSAGE Network,s and Graph Isomorphism Networks (GINs), which might offer different strengths in capturing complex graph structures and improving classification performance. Experimenting with advanced attention mechanisms, such as hierarchical or multi-level attention, could further enhance the model's sensitivity to subtle fraud patterns.

- **Classifying Transaction Trails (Subgraphs):** A significant extension will involve predicting transaction trails (subgraphs) as fraudulent or non-fraudulent. This approach considers the broader context of transactions, identifying chains of transactions or subgraphs that exhibit suspicious patterns over time. The system can better capture complex money-laundering schemes and other fraudulent activities that span multiple transactions and accounts by developing models that can classify entire subgraphs as fraud or not fraud.

- **Generating Transaction Trails:** An important area for future exploration is the development of a model capable of reconstructing transaction trails or completing incomplete trails within the dataset to better determine whether they indicate fraudulent activity. In real-world scenarios, financial transaction data is often incomplete due to missing records, fragmented information, or limitations in data collection processes. These gaps can obscure critical patterns and relationships, particularly in complex fraud schemes that span multiple accounts and transactions over time. Addressing this challenge by generating and completing transaction trails would enhance both data quality and model performance. The proposed approach involves leveraging advanced graph-based modeling techniques, to infer missing links within transaction trails. By analyzing the structural context of existing data, the model can generate plausible connections between accounts, reconstructing previously incomplete trails.

Bibliography

[1] Abdul Joseph Fofanah, David Chen, L. W. S. Z. [2023], 'Addressing imbalance in graph datasets: : Introducing gate-gnn with graph ensemble weight attention and transfer learning for enhanced node classification', **255**.

[2] Jie Zhou, Ganqu Cui, S. H. Z. Z. C. Y. Z. L. L. W. C. L. M. S. [2019], 'Graph neural networks: A review of methods and applications', *AI open* **5**.

[3] Jun Chen, H. C. [2021], 'Edge-featured graph attention network', *cs.LG* **1**.

[4] Khemani, B., Patil, S., Kotecha, K. and Tanwar, S. [2024], 'A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions', *Journal of Big Data* **11**.

[5] Liyu Gong, Q. C. [2019], 'Exploiting edge features in graph neural networks : This explores edge feature enhancements in gnns, including applications in attention mechanisms', *cs.LG* **2**.

[6] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. and Monfardini, G. [2009], 'The graph neural network model', *IEEE Transactions on Neural Networks* **20**(1), 61–80.

[7] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. and Bengio, Y. [2018], 'Graph attention networks', *ICLR* **1**.

[8] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C. and Sun, M. [2020], 'Graph neural networks: A review of methods and applications', *AI Open* **1**, 57–81.

[9] Ziniu Hu, Yuxiao Dong, K. W. Y. S. [2020], 'Heterogeneous graph transformer : This paper introduces extensions to gat for heterogeneous graphs, including edge-specific enhancements.', *cs.LG* **1**.