A

PROJECT SCHOOL REPORT ON

GE: GNN GATED EDGE-AUGMENTED GRAPH NEURAL NETWORK

Submitted By

CHIGURAMBOTTLA SAI PRADEEP GUPTHA	**23P81A0514**
MADDIKUNTLA PAVAN KUMAR	**23P81A0534**
MUNNANURU SAI YASASVI	**23P81A0543**
NANNAPURAJU VARSHA	**23P81A0546**
SURKANTI KAVYA REDDY	**23P81A0557**

Under the guidance of

Dr.A.Shilpa Gupta

Assistant Professor, CSE (AIML), KMCE

KESHAV MEMORIAL COLLEGE OF ENGINEERING

Chinthapallyguda village, Ibrahimpatnam, Hyderabad, Telangana 500058

NOVEMBER 2025

KESHAV MEMORIAL COLLEGE OF ENGINEERING

A Unit of Keshav Memorial Technical Education (KMTES)

Approved by AICTE, New Delhi & Affiliated to Jawaharlal Nehru Technological University, Hyderabad

CERTIFICATE

This is to certify that the project work entitled **"GE:GNN GATED EDGE-AUGMENTED GRAPH NEURAL NETWORK"** *is a bonafide work carried out by* **"Chigurambottla Sai Pradeep Guptha, Maddikuntla Pavan Kumar, Munnanuru Sai Yasasvi, Nannapuraju Varsha, Surkanti Kavya Reddy,"** of III-year I semester **Bachelor of Technology** *in* **CSE** *during the academic year* **2025-2026 and** *is a record of bonafide work carried out by them.*

Project Mentor

Dr.A.Shilpa Gupta

Assistant Professor, CSE (AIML)

KMCE

ABSTRACT

Graph Neural Networks (GNNs) are important and have been widely used in fraud detection tasks. They show notable improvements in detection performance compared to traditional methods. However, within the complex structure of fraud graphs, fraudsters often hide among many benign entities. A good way to tackle the camouflage problem is by including detailed and rich edge information. Unfortunately, current GNN-based methods often fail to incorporate this vital information into the message-passing process, which limits their effectiveness. To resolve these issues, this study proposes a new Gated Edge-augmented Graph Neural Network (GE-GNN). Our approach starts with an edge-based feature enhancement mechanism that uses both node and edge features within a single relation. Next, we apply the enhanced representation to the message-passing process to update the node embeddings. Additionally, we create a gate logistic to manage how the augmented information is expressed. Finally, we combine node features across different relations to obtain a complete representation. Extensive experiments on two real-world datasets show that our method outperforms several leading techniques.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Online platforms such as e-commerce sites, financial services, and social networks face a growing threat from fraudulent users who post fake reviews, perform false transactions, or create spam accounts. Detecting these fraudsters is difficult because they often hide among genuine users by forming deceptive connections that make their behavior appear normal.

Traditional machine learning models depend on manually crafted features and usually fail to capture such complex relationships between users. To overcome these limitations, researchers have turned to **Graph Neural Networks (GNNs)**, which can learn from both user features and their connections within a network.

However, most GNN-based methods focus mainly on node information and ignore valuable edge details — such as how strongly or frequently users interact — which can reveal hidden fraud patterns. To address this gap, this research proposes the **Gated Edge-Augmented Graph Neural Network (GE-GNN)**, a model that combines node and edge features for better fraud detection. The proposed model enhances message passing using edge-based feature augmentation and a gating mechanism, achieving improved accuracy and robustness on real-world datasets like **YelpChi** and **Amazon**.

1.1 Overview of Fraud Detection in Online Networks

Fraud detection has become a major concern for modern online platforms such as e-commerce sites, digital payment systems, and social networks. As these platforms grow, so do fraudulent activities — including fake reviews, false transactions, and spam accounts. Fraudsters often hide their actions by forming deceptive links with genuine users, which makes them hard to identify.

Traditional detection systems focused mainly on static patterns or suspicious behaviors of individual users. But fraud in real networks rarely happens alone — it spreads through **connections and interactions**. Understanding how users relate to one another offers deeper clues about potential fraud. This shift from analyzing isolated data points to studying **network relationships** has made graph-based methods an increasingly effective solution.

1.2 Limitations of Traditional Fraud Detection Methods

Earlier fraud detection models mostly relied on algorithms like decision trees, logistic regression, and support vector machines. These methods could spot simple or well-known fraud patterns but often failed when relationships between users became complex or dynamic. They treated every user as a separate case, ignoring how those users were linked in a network.

To handle this, researchers started using **Graph Neural Networks (GNNs)**, which learn from both the attributes of users (nodes) and their relationships (edges). Models such as the **Graph Convolutional**

Network (GCN), **Graph Attention Network (GAT)**, and **GraphSAGE** improved performance in graph learning tasks. Still, these models have their own limits when applied to fraud detection.

Most standard GNNs focus on **node-level aggregation**, paying little attention to **edge-level information** — for example, how often two users interact or the type of transaction between them. Such details are often strong indicators of fraud. Ignoring them allows **camouflaged fraudsters** to blend in with normal users. These drawbacks highlight the need for an enhanced model that can jointly use node and edge information to detect hidden fraudulent behavior.

1.3 Role of Graph Neural Networks (GNNs) in Fraud Detection

Graph Neural Networks have become one of the most promising approaches for fraud detection because they can capture both individual characteristics and relational structures in a network. By using a **message-passing mechanism**, GNNs allow each node to learn from its neighbors, helping the model understand how entities are connected. This makes it easier to identify suspicious users who share unusual connection patterns.

For instance, a fraudulent account might frequently interact with other suspicious accounts, forming dense clusters that traditional methods would overlook. GNNs can uncover these patterns by learning from the structure of the entire graph.

However, most current GNNs still rely heavily on node features and lack awareness of **edge semantics** — information like the type or strength of a connection. To bridge this gap, the proposed **Gated Edge-Augmented Graph Neural Network (GE-GNN)** enhances the standard GNN framework by including edge features and a gating mechanism that controls how information flows between nodes. This design allows GE-GNN to capture more subtle and complex fraud patterns, improving both accuracy and robustness in real-world detection tasks.

CHAPTER 2

LITERATURE SURVEY

This section provides a detailed review of existing research related to fraud detection in online networks. It traces the evolution from traditional machine learning methods to more advanced graph-based approaches. Earlier studies mostly relied on feature-based detection techniques, whereas recent research has shifted toward using **Graph Neural Networks (GNNs)** to better capture the structural and relational patterns among users and their interactions.

The following subsections discuss various graph-based methods applied to fraud detection, explain the **camouflage problem** commonly observed in fraud graphs, and highlight the key challenges that the proposed **GE-GNN model** aims to address.

2.1 Existing Graph-Based Approaches for Fraud Detection

Before the introduction of graph-based learning, fraud detection mainly relied on **traditional machine learning models** such as **Decision Trees**, **Logistic Regression**, and **Support Vector Machines (SVMs)**. These methods analyze individual user or transaction features — like review frequency, rating patterns, or transaction amount — to identify suspicious activity. Although these models performed reasonably well on small or structured datasets, they struggled with **complex and dynamic online environments**. They treat each user as an independent entity and fail to capture the **relationships or connections** between users, which often hold crucial information about fraudulent behavior. This limitation motivated the shift toward **graph-based approaches**, where both users and their interactions can be analyzed together.

After the limitations of traditional machine learning techniques, researchers began exploring **graph-based models** for fraud detection. These methods represent users and their interactions as graphs, allowing the model to capture both individual behaviors and relational patterns that traditional approaches often miss.

Graph Based Methods:

1. Graph Convolutional Network (GCN)

The **Graph Convolutional Network (GCN)** was one of the first models to bring deep learning techniques to graph-structured data. It learns how to represent each node by aggregating information from its neighboring nodes through convolution operations. This helps the model understand local graph structures and relationships effectively. However, GCN treats all neighboring nodes with equal importance, which can be limiting in fraud detection. In reality, not every connection carries the same level of significance — some interactions are far more indicative of fraudulent activity than others.

2. Graph Attention Network (GAT)

The **Graph Attention Network (GAT)** improved upon GCN by introducing an **attention mechanism** that allows the model to assign different weights to different neighbors. This helps GAT focus more on important or influential relationships while reducing the effect of less relevant ones. It performs particularly well in heterogeneous or noisy networks. Still, GAT mainly concentrates on node information and struggles to capture detailed **edge features** — such as the type, frequency, or strength of an interaction — which can provide valuable hints in fraud detection.

3. GraphSAGE (Graph Sample and Aggregate)

GraphSAGE was designed to handle **large-scale graph data** more efficiently. Instead of using all connected neighbors for each node, it samples a small subset and aggregates their features to generate node embeddings. This approach improves scalability and allows the model to make predictions on previously unseen nodes, a process known as **inductive learning**. However, since GraphSAGE relies on random sampling, it may lose important relational information, especially when key neighbors are not included in the sample.

4. CARE-GNN (Camouflage-Resistant GNN)

CARE-GNN was one of the first GNN models created specifically for **fraud detection**. It tackles the **camouflage problem**, where fraudsters disguise themselves by forming connections with genuine users. CARE-GNN uses a **relation-aware neighbor sampling** approach, giving different attention to various types of relationships. This makes it more effective at identifying fraudulent nodes that try to appear normal.

Despite its strengths, CARE-GNN still struggles to fully utilize **edge semantics** such as the weight or category of connections, which can be crucial in understanding fraud behavior.

5. PC-GNN (Proximity and Context-aware GNN)

PC-GNN extends CARE-GNN by incorporating both **proximity** and **contextual information** to better distinguish between normal and fraudulent users. It captures both **local** and **global** structures of the network, improving performance on real-world datasets like **YelpChi** and **Amazon**. Although PC-GNN achieves strong results, it still underuses **edge-level features**, which often hold key information about interactions. This limitation motivated the development of the **Gated Edge-Augmented Graph Neural Network (GE-GNN)** — a model that integrates both node and edge features to enhance fraud detection accuracy.

2.2 Camouflage Problem in Fraud Graphs

One of the biggest challenges in fraud detection using graph-based models is the **camouflage problem**. Fraudsters often hide their malicious behavior by forming connections with normal or trustworthy users, making their activities appear genuine. As a result, they blend seamlessly into the network, making it difficult for traditional and even advanced GNN-based methods to distinguish them from legitimate users.

In fraud graphs, this camouflage effect weakens the ability of models to identify fraudulent nodes based solely on direct connections. For example, a fake reviewer may interact frequently with genuine users to mask their identity or create misleading patterns of trust. This behavior disrupts the natural structure of the graph, reducing the effectiveness of standard aggregation methods used in models like GCN or GraphSAGE.

To overcome this issue, newer models such as **CARE-GNN** and **PC-GNN** introduced relation-aware and context-based neighbor sampling strategies. These techniques assign different importance levels to various types of connections, helping the model focus more on suspicious relationships rather than treating all links equally. However, even with these improvements, effectively capturing **edge-level semantics** remains a challenge — motivating the development of the **Gated Edge-Augmented Graph Neural Network (GE-GNN)**, which enhances fraud detection through edge-aware learning and gated information flow.

2.3 Challenges Addressed by GE-GNN

While existing GNN-based approaches such as GCN, GAT, CARE-GNN, and PC-GNN have improved fraud detection, they still face several limitations. The proposed **Gated Edge-Augmented Graph Neural Network (GE-GNN)** addresses the following key challenges:

- **Limited Edge Utilization:** Previous models mainly focus on node features and overlook critical edge information such as interaction type and weight.
- **Camouflaged Fraudsters:** Difficulty in identifying fraudsters who form deceptive links with genuine users.
- **Over-Smoothing:** Deep GNN layers make node representations too similar, reducing discrimination power.
- **Weak Generalization:** Existing models struggle with heterogeneous graphs containing diverse node and edge types.
- **Inefficient Information Propagation:** Uniform message passing introduces noise and irrelevant signals during learning

GE-GNN effectively resolves these challenges by integrating edge-aware learning and gated information flow, resulting in improved fraud detection accuracy and robustness

CHAPTER 3 PROPOSED WORK ARCHITECTURE, TECHNOLOGY STACK, IMPLEMENTATION DETAILS

3.1 Proposed Work

The proposed system introduces a Gated Edge-Augmented Graph Neural Network (GE-GNN) designed to enhance fraud detection in online networks. Unlike conventional GNN models that focus primarily on node-level features, GE-GNN integrates both node and edge attributes within its learning process. This allows the model to effectively capture the interaction strength and relationship patterns between users, which are crucial for identifying camouflaged fraudsters.

The main idea behind GE-GNN is to employ an edge-aware message-passing framework combined with a gating mechanism that controls how information flows between connected nodes. This enables the network to filter out irrelevant or misleading connections, ensuring that only meaningful relationships contribute to the learning process.

Through this design, GE-GNN aims to:

- Utilize both node and edge information for improved representation learning.
- Overcome the camouflage problem in fraud graphs.
- Reduce over-smoothing and noise during message passing.

- Achieve higher detection accuracy across heterogeneous datasets like YelpChi and Amazon.

Overview of Proposed Work

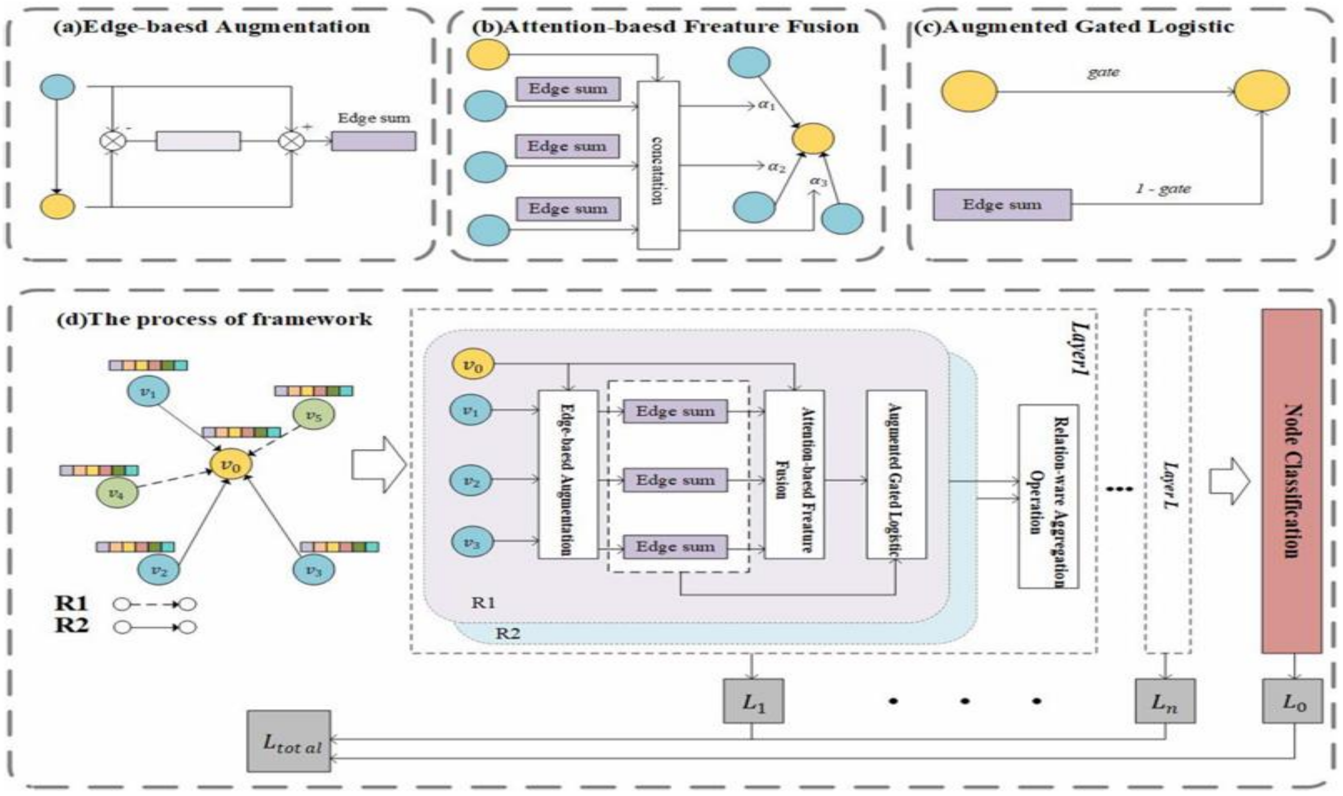

(a)Edge-baesd Augmentation
Edge sum
(b)Attention-baesd Freature Fusion
Edge sum
concatation
(c)Augmented Gated Logistic
gate
1 - gate
(d)The process of framework
R1
R2
Layer1
Edge-based Augmentation
Attention-based Freature Fusion
Augmented Gated Logistic
Relation-ware Aggregation Operation
Layer L
Node Classification
L_1
L_n
L_o
L_{total}

Figure 1. The Overall Framework of GE-GNN

The architecture of the proposed GE-GNN model is depicted in **Figure 3.1**. It comprises four main modules:
(a) Edge-based Augmentation,
(b) Attention-based Feature Fusion,
(c) Augmented Gated Logistic, and
(d) Relation-aware Aggregation Process.
Each component plays a unique role in enhancing feature extraction, fusion, and classification.

(a) Edge-Based Feature Augmentation

The **Edge-Based Augmentation** module is designed to capture hidden information between an anchor node and its neighbors.
It integrates node features from multiple sources to form a rich representation for each edge.
This module helps the model effectively propagate both **node-level** and **edge-level** information.

Mathematical Representation

Let $G = (V, E, X)$be a relation heterogeneous directed graph, where:

- $V = \{v_1, v_2, \ldots, v_N\}$is the set of nodes
- $X = \{x_1, x_2, \ldots, x_N\}$are node features

- $E_r$represents the set of edges under relation type r

Each node feature $x_i^0$is first transformed into a common latent space for a specific relation E_r:

$$x_i^{E_r} = \sigma(W^{E_r} \cdot x_i^0 + b^{E_r})$$

where $W^{E_r}$and $b^{E_r}$are learnable parameters and $\sigma(\cdot)$is an activation function (ReLU).

Next, for a neighbor node $v_j$of anchor node v_i, the **edge feature** is computed as:

$$e_{ji}^{E_r} = x_j^{E_r} - x_i^{E_r}$$

The **augmented edge representation** is then obtained by combining both node and edge information:

$$\tilde{e}_{ji}^{E_r} = \tanh(x_j^{E_r} + x_i^{E_r} + e_{ji}^{E_r})$$

Here, $\tilde{e}_{ji}^{E_r}$represents the enriched feature capturing the relationship between node $v_i$and its neighbor $v_j$under relation E_r.

(b) Attention-Based Feature Fusion

Once edge-based features are generated, the **Attention-Based Feature Fusion** module is employed to aggregate information from neighboring nodes.
This mechanism ensures that nodes with higher importance (in terms of relationships) receive greater attention during the message-passing phase.

Mathematical Representation

Let $N_{E_r}(v_i)$denote the set of neighbors of node $v_i$under relation E_r.
The **importance score** $s_{ji}^{E_r}$between nodes $v_i$and $v_j$is calculated as:

$$s_{ji}^{E_r} = a_r^{\top}[W_r x_j^{E_r} \;||\; W_r x_i^{E_r} \;||\; W_r \tilde{e}_{ji}^{E_r}]$$

where $a_r \in \mathbb{R}^{1\times 3d}$is a learnable attention vector, $W_r \in \mathbb{R}^{d\times d}$is a transformation matrix, and $||$denotes concatenation.

The **attention coefficient** $\alpha_{ji}^{E_r}$is obtained by normalizing scores using the softmax function:

$$\alpha_{ji}^{E_r} = \frac{\exp(\mathrm{LeakyReLU}(s_{ji}^{E_r}))}{\sum_{k\in N_{E_r}(v_i)} \exp(\mathrm{LeakyReLU}(s_{ki}^{E_r}))}$$

Finally, the aggregated representation of node $v_i$is computed as:

$$z_i^{E_r} = \sigma(\sum_{j\in N_{E_r}(v_i)} \alpha_{ji}^{E_r} x_j^{E_r})$$

For better expressiveness, **multi-head attention** is applied with Kheads:

$$z_i^{E_r} = \|_{k=1}^{K} \sigma(\sum_{j\in N_{E_r}(v_i)} \alpha_{ji}^{E_r,k} x_j^{E_r,k})$$

In the final layer, the average of all attention heads is used instead of concatenation:

$$z_i^{E_r} = \sigma(\frac{1}{K}\sum_{k=1}^{K} \sum_{j\in N_{E_r}(v_i)} \alpha_{ji}^{E_r,k} x_j^{E_r,k})$$

This ensures stable learning and helps the network focus on the most influential neighboring nodes.

(c) Augmented Gated Logistic Mechanism

This component introduces a **gate mechanism** to balance the influence between aggregated features and augmented edge representations.
The gate dynamically controls how much information from each should flow into the final node embedding.

Mathematical Representation

First, the **global augmented representation** of node $v_i$under relation $E_r$is obtained by averaging all augmented edge embeddings:

$$e_i^{E_r} = \frac{1}{| N_{E_r}(v_i) |} \sum_{j \in N_{E_r}(v_i)} \tilde{e}_{ji}^{E_r}$$

Next, a **gating score** is computed to determine the balance between aggregated and augmented information:

$$\text{gate}_i^{E_r} = \sigma(\beta_i^{E_r}[e_i^{E_r} \ || \ z_i^{E_r}])$$

The final node representation $h_i^{E_r}$is derived as a weighted combination:

$$h_i^{E_r} = \text{gate}_i^{E_r} \cdot z_i^{E_r} + (1 - \text{gate}_i^{E_r}) \cdot e_i^{E_r}$$

This helps the model adaptively decide how much of each type of information contributes to the final node representation.

(d) Relation-Aware Aggregation and Framework Process

In a heterogeneous graph, nodes are connected by multiple types of relations.
After computing the final representation $h_i^{E_r}$for each relation E_r, the **Relation-Aware Aggregation** module fuses all relation-specific embeddings into a unified representation.

Mathematical Representation

All relation-specific embeddings are concatenated:

$$h_i^{\text{whole}} = \|_{r=1}^{R} h_i^{E_r}$$

The final low-dimensional embedding is then obtained using a transformation matrix:

$$h_i = W_d \cdot h_i^{\text{whole}}$$

where $W_d$is a learnable parameter matrix.
The resulting node embeddings $h_i$are then used for **node classification** tasks (e.g., fraud vs. non-fraud).

Model Training

The model is trained in a semi-supervised setting using **cross-entropy loss**.
Each layer contributes an individual layer-specific loss, and the total loss is computed as:

$$L_{\text{total}} = \frac{1}{L} \sum_{l=1}^{L} [- \sum_{v \in V_t} (y_v \log p_v^l + (1 - y_v) \log(1 - p_v^l))]$$

where $p_v^l = \text{softmax}(h_i^l)$denotes the predicted probability of node $v$in layer l.

3.2 Technology Stack

The proposed system, GE-GNN (Gated Edge-Augmented Graph Neural Network), brings together web technologies, database management, and machine learning techniques to ensure smooth model execution and effortless communication between different components of the application.

Frontend Development:

- **Framework:** React.js
- It is Used to design an interactive and user-friendly interface that allows users to upload data files and view fraud detection results. React ensures smooth rendering, fast performance, and responsive UI behavior.

Backend Development:

- **Language:** Python
- **Framework:** Flask
- Flask serves as the backend framework that connects the frontend with the machine learning model. It handles incoming requests, communicates with the model hosted on Google Colab, and returns prediction results to the frontend.

Model Environment:

- **Platform:** Google Colab
- **Libraries and Frameworks:** PyTorch, PyTorch Geometric, NumPy, Pandas, Scikit-learn.
- The GE-GNN model is implemented and trained in Google Colab using Python. The model performs edge-based augmentation, gated message passing, and attention-based feature fusion for accurate fraud detection.

Database: MongoDB

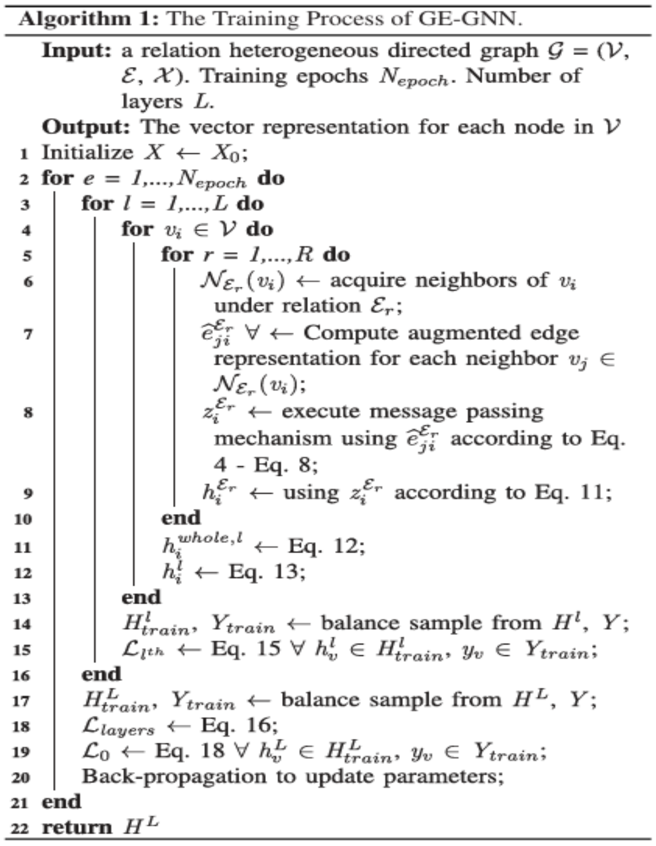
- MongoDB is used to store user-uploaded data, prediction outputs, and system logs. Its NoSQL structure supports flexibility and scalability for handling large volumes of heterogeneous data.

Deployment Environment:

- **Platform:** Render
- The Flask backend will be deployed on **Render**, allowing real-time interaction between the frontend, backend, and database. Render provides a reliable cloud environment for hosting web services with continuous availability.

3.3 Model Architecture:

The algorithm describes the step-by-step process followed by the **GE-GNN (Gated Edge-Augmented Graph Neural Network)** during training to learn meaningful node representations for fraud detection.



```
Algorithm 1: The Training Process of GE-GNN.
Input: a relation heterogeneous directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$. Training epochs $N_{epoch}$. Number of layers $L$.
Output: The vector representation for each node in $\mathcal{V}$
Initialize $X \leftarrow X_0$;
for $e = 1,...,N_{epoch}$ do
    for $l = 1,...,L$ do
        for $v_i \in \mathcal{V}$ do
            for $r = 1,...,R$ do
                $\mathcal{N}_{\mathcal{E}_r}(v_i) \leftarrow$ acquire neighbors of $v_i$ under relation $\mathcal{E}_r$;
                $\widehat{e}_{ji}^{\mathcal{E}_r} \forall \leftarrow$ Compute augmented edge representation for each neighbor $v_j \in \mathcal{N}_{\mathcal{E}_r}(v_i)$;
                $z_i^{\mathcal{E}_r} \leftarrow$ execute message passing mechanism using $\widehat{e}_{ji}^{\mathcal{E}_r}$ according to Eq. 4 - Eq. 8;
                $h_i^{\mathcal{E}_r} \leftarrow$ using $z_i^{\mathcal{E}_r}$ according to Eq. 11;
            end
            $h_i^{whole,l} \leftarrow$ Eq. 12;
            $h_i^l \leftarrow$ Eq. 13;
        end
        $H_{train}^l$, $Y_{train} \leftarrow$ balance sample from $H^l$, $Y$;
        $\mathcal{L}_{l^{th}} \leftarrow$ Eq. 15 $\forall h_v^l \in H_{train}^l$, $y_v \in Y_{train}$;
    end
    $H_{train}^L$, $Y_{train} \leftarrow$ balance sample from $H^L$, $Y$;
    $\mathcal{L}_{layers} \leftarrow$ Eq. 16;
    $\mathcal{L}_0 \leftarrow$ Eq. 18 $\forall h_v^L \in H_{train}^L$, $y_v \in Y_{train}$;
    Back-propagation to update parameters;
end
return $H^L$
```

Fig 2: Training process of GE-GNN

1. **Initialization:**
   The model begins by initializing the feature matrix $X$with the initial node features X_0. These represent the basic attributes of each node before any message passing.
2. **Epoch Loop:**
   The network is trained for a fixed number of epochs (N_{epoch}). In each epoch, the model updates node representations through multiple layers of computation.
3. **Layer-wise Propagation:**
   For each layer L, the model iteratively processes every node $v_i$in the graph.
4. **Neighbor Retrieval:**
   For each node, the model gathers its neighboring nodes under each relation type E_r(e.g., user-product-user, user-sentiment-user, etc.).
5. **Edge Augmentation:**
   The algorithm computes **augmented edge representations** ($\tilde{e}_{ji}^{E_r}$) by combining information from both the source and target nodes, enhancing the message-passing process with edge-level details.
6. **Message Passing:**
   Using the augmented edge features, the model performs message passing to aggregate information from neighboring nodes. This helps each node learn contextual information from its connected nodes.
7. **Node Update:**
   The node embeddings are updated using the aggregated messages to form new feature representations ($h_i^{E_r}$) for each relation type.
8. **Relation-aware Aggregation:**
   The relation-specific embeddings are then fused into a unified representation ($h_i^{whole,l}$), combining insights from all types of relations in the graph.

9. **Loss Calculation:**
   For each layer, the model balances samples from the training data and calculates a **layer-specific loss** to ensure stable learning across different layers.
10. **Final Loss and Backpropagation:**
    The total loss is computed by combining all intermediate and final layer losses. Using this, **backpropagation** is performed to update model parameters and improve performance.
11. **Output:**
    After all epochs are completed, the final node representations $H^L$are obtained, which capture both structural and relational information useful for classifying nodes as fraudulent or genuine.

3.4 Implementation details

3.4.1 Data Collection and Preprocessing

Data Collection: The dataset used in this project was sourced from the **Amazon Product Reviews**, specifically the *Sports and Outdoors* category. It contains detailed information such as reviewer ID, product ID (ASIN), rating, review text, summary, and helpfulness votes, which collectively describe user behavior and interactions. The dataset was obtained in **JSON format** and processed in **Google Colab** using **Python**, **Pandas**, and **NumPy**. To ensure class balance and improve the accuracy of fraud detection, an equal number of genuine and fraudulent reviews were selected, resulting in a balanced dataset of **20,000 records** (10,000 per class). The dataset was then stored and used for subsequent preprocessing,feature engineering, and model training stages.
🔗 **Dataset on Kaggle**: https://www.kaggle.com/datasets/naveedhn/amazon-product-review-spam-and-non-spam

Data Preprocessing: Before training the model, extensive preprocessing was carried out to clean and prepare the dataset for analysis. Missing values in fields such as reviewText, summary, and reviewerName were filled with empty strings to maintain consistency. The "helpful" column, which originally contained lists like [helpful_votes, unhelpful_votes], was separated into distinct numerical columns, and an additional feature—the **helpful-to-unhelpful ratio**—was computed. Temporal attributes were derived by converting reviewTime into a datetime format, from which the **day gap** between reviews and a **same-day indicator** were calculated to analyze reviewer activity frequency.

Further, reviewer-level statistics, such as total helpful and unhelpful votes, were aggregated to represent user reliability. Textual data from reviews and summaries were converted into numerical representations using **TF-IDF (Term Frequency–Inverse Document Frequency)** followed by **Truncated SVD** for dimensionality reduction. **Sentiment scores** were generated using the **VADER Sentiment Analyzer** from the NLTK library to capture the emotional polarity of each review. Categorical fields like reviewerID, asin, and reviewerName were numerically encoded using **Target Encoding** and **Count Encoding**, while all numerical features were standardized using **StandardScaler** to ensure smooth model convergence. Finally, the preprocessed dataset was saved as a **.mat file** and later converted into a **DGL (Deep Graph Library)** format to serve as input for the GE-GNN model training.

3.4.2 Feature Extraction and Embedding Generation

After preprocessing, the next step focused on extracting important features and generating embeddings that capture both user behavior and relationships between reviews. Numerical features such as ratings, helpfulness ratio, day gap, sentiment score, and word count were included as **node features**. Text data from the review text and summary were transformed into numerical vectors using **TF-IDF**, and dimensionality was reduced using **Truncated SVD** to obtain meaningful low-dimensional text embeddings.

Categorical attributes like reviewerID, asin, and reviewerName were converted into numeric form using **Target Encoding** and **Count Encoding**, and all features were combined into a single feature matrix representing each node. To model relationships between reviews, three types of connections were defined: **UPU (User-Product-User)** for reviews of the same product, **USU (User-Sentiment-User)** for reviews by the same user with similar ratings, and **UVU (User-View-User)** for reviews with similar word lengths.

These relations were stored as adjacency matrices and converted into a **heterogeneous DGL graph**, where nodes represented reviews and edges represented their relationships. This graph structure allowed the **GE-GNN** model to learn both node-level and edge-level patterns effectively for fraud detection.

3.4.3 Training and Validation process

The training process of the **GE-GNN (Gated Edge-Augmented Graph Neural Network)** model was carried out on the constructed DGL graph. The model was trained in a **semi-supervised** manner using node labels that indicated whether a review was fraudulent or genuine. During training, the graph data was divided into **training**, **validation**, and **testing** sets to ensure balanced learning and unbiased evaluation.

The model optimized a **cross-entropy loss function**, and an **Adam optimizer** was used to update the parameters. An **early stopping mechanism** was applied to prevent overfitting, stopping the training automatically if the validation performance did not improve for several epochs. The learning rate was dynamically adjusted using a **ReduceLROnPlateau scheduler** for stable convergence.

After each epoch, performance was evaluated on the validation set using metrics such as **AUC**, **F1-score**, and **Recall**. Once the best model was identified, it was tested on unseen data to assess generalization capability. The trained model, along with the scaler and configuration files, was then saved for future inference and deployment.

3.3.4 Evaluation metrics

To assess the performance of the GE-GNN model, several evaluation metrics were used to ensure both accuracy and reliability in fraud detection. The key metrics included **Accuracy, F1-Score, Precision, Recall, and AUC (Area Under the ROC Curve).**

- Accuracy measured the overall correctness of predictions.
- Precision evaluated how many of the detected fraudulent reviews were actually fraud.
- Recall indicated how effectively the model identified all existing fraud cases.
- F1-Score, the harmonic mean of precision and recall, provided a balanced measure of performance, especially for imbalanced datasets.
- AUC assessed the model's ability to distinguish between fraudulent and non-fraudulent reviews across all thresholds.

Together, these metrics offered a comprehensive understanding of how well the GE-GNN model performed in detecting fraud patterns and handling real-world, complex review data.

3.5 System Architecture

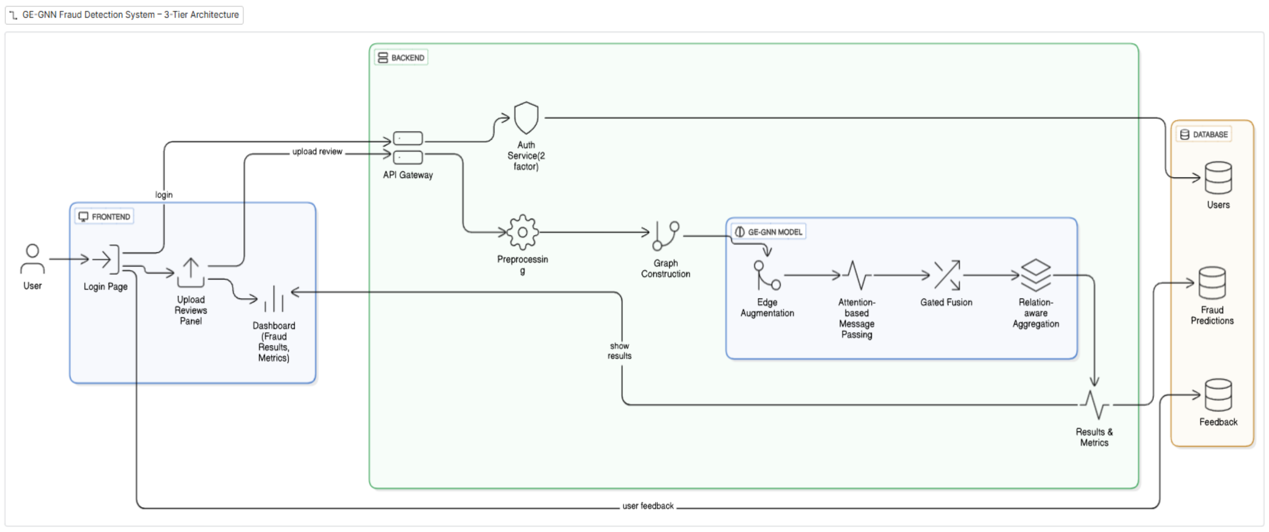


Fig:3 3-Tier Architecture of GE-GNN Fraud detection System

System Workflow

1. **User Login:**
   The user accesses the frontend built with React and logs into the system. The login request is verified through the backend authentication service.
2. **Upload Reviews:**
   After login, the user uploads a CSV file containing product reviews from the **Upload Reviews Panel** on the dashboard.
3. **Request Handling:**
   The uploaded file is sent to the **API Gateway** in the backend, which routes the request for further processing.
4. **Preprocessing:**
   The backend preprocessing module cleans the uploaded data — handling missing values, extracting features like ratings, sentiment scores, and word counts, and encoding categorical attributes.
5. **Graph Construction:**
   The preprocessed data is converted into a **heterogeneous graph** structure using the Deep Graph Library (DGL), forming different types of connections such as **UPU**, **USU**, and **UVU** between reviews.
6. **Model Inference (GE-GNN):**
   The graph is passed into the **GE-GNN model**, which processes the data through four main steps — **Edge Augmentation**, **Attention-based Message Passing**, **Gated Fusion**, and **Relation-aware Aggregation** — to detect fraudulent reviews.
7. **Results Generation:**
   The model outputs predictions showing whether each review is **fraudulent or genuine**, along with fraud probability scores.

8. **Database Storage:**
   The results, along with user and feedback data, are stored securely in **MongoDB** for record-keeping and analysis.
9. **Results Display:**
   The backend sends the prediction results back to the frontend, where they are displayed on the **Dashboard** with detailed metrics and visual summaries.
10. **User Feedback:**
    The user can review the predictions and provide feedback, which helps improve the system in future updates.

3.6 Interfaces and Communication

Here are the UI interfaces of our GE-GNN platform. They show how users interact with the system through login, access requests, and admin operations.

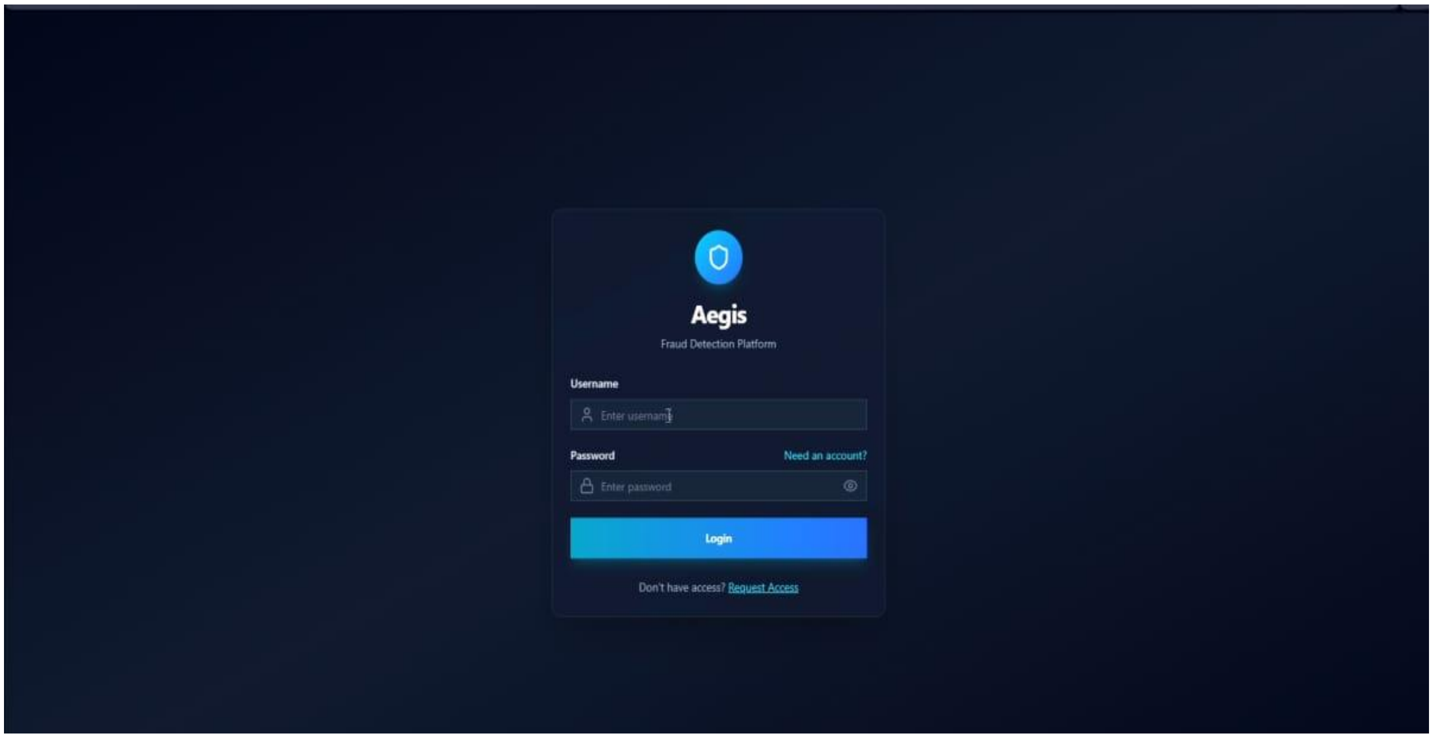


Fig-4: Login Page

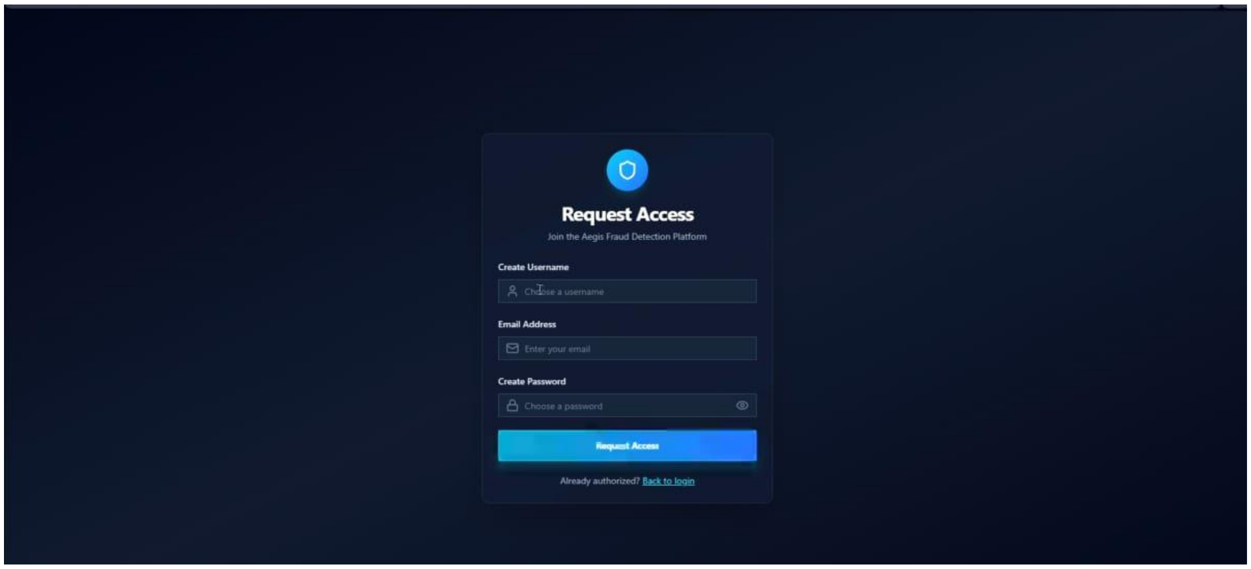
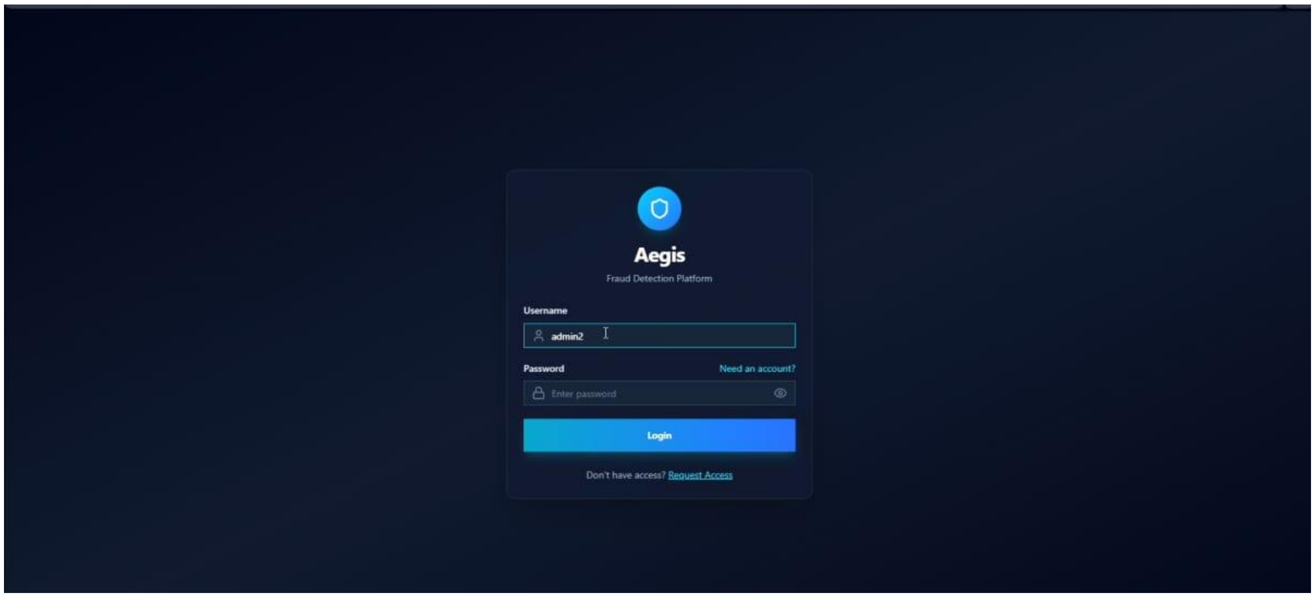
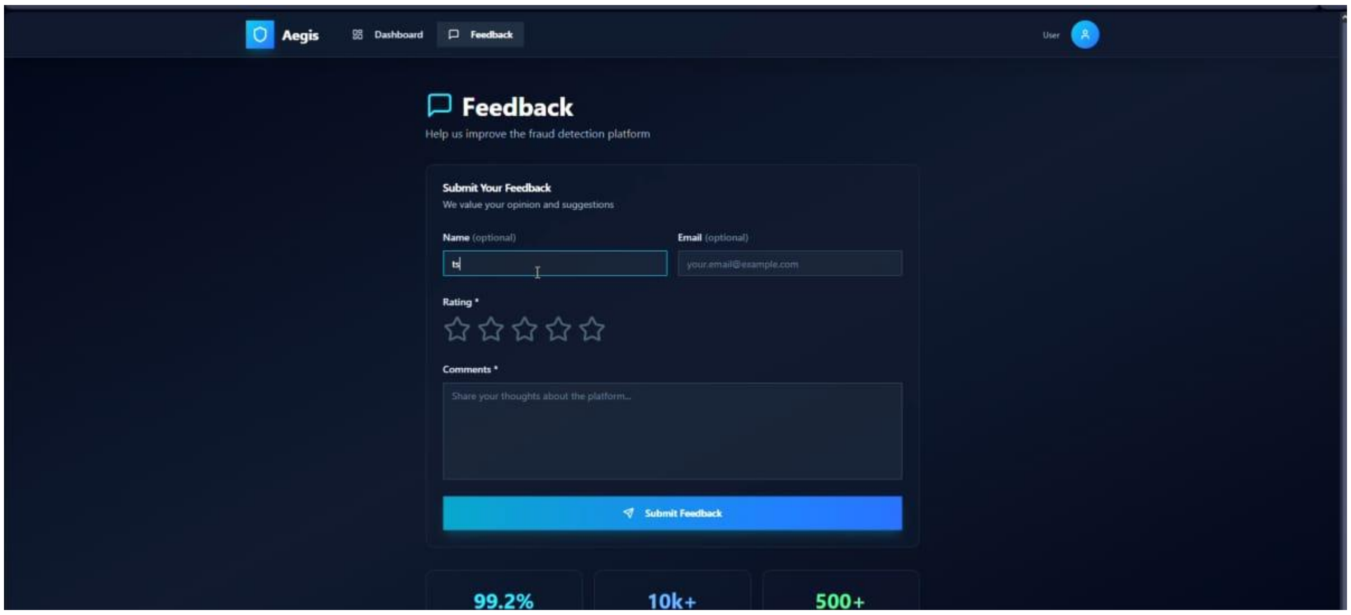


Fig 5: Request Access Page



Username: admin2

Fig 6: Admin Login Page



Feedback

Help us improve the fraud detection platform

Submit Your Feedback

We value your opinion and suggestions

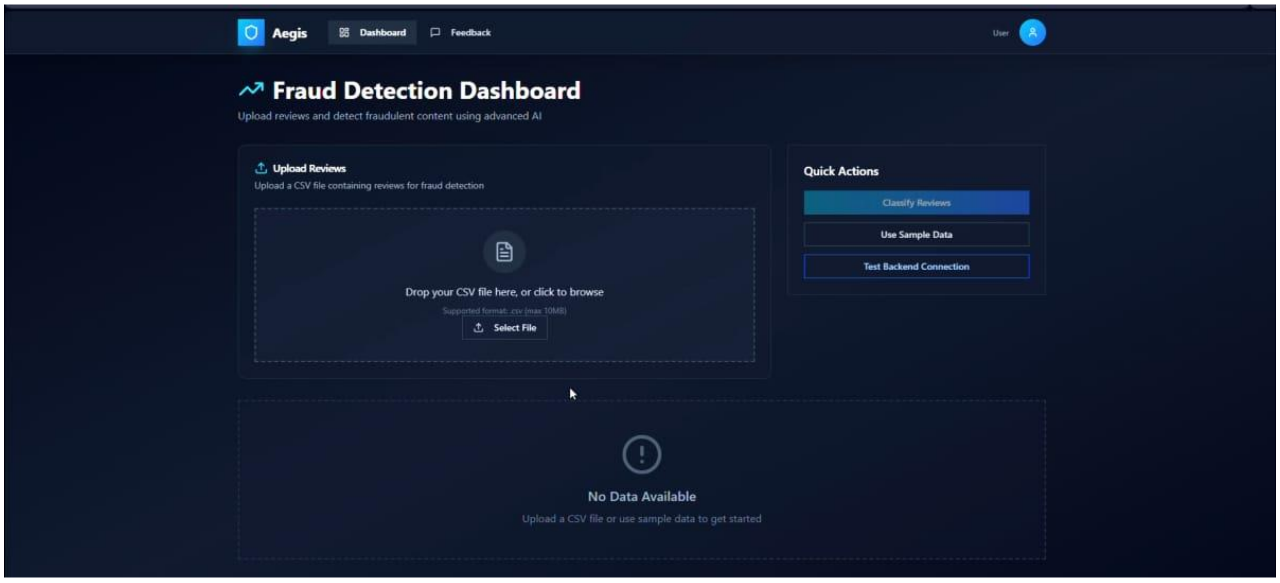
Name (optional): ts

Email (optional): your.email@example.com

Rating *

Comments *: Share your thoughts about the platform...

Fig 7: Feedback page



Fraud Detection Dashboard

Upload reviews and detect fraudulent content using advanced AI

Upload Reviews

Upload a CSV file containing reviews for fraud detection

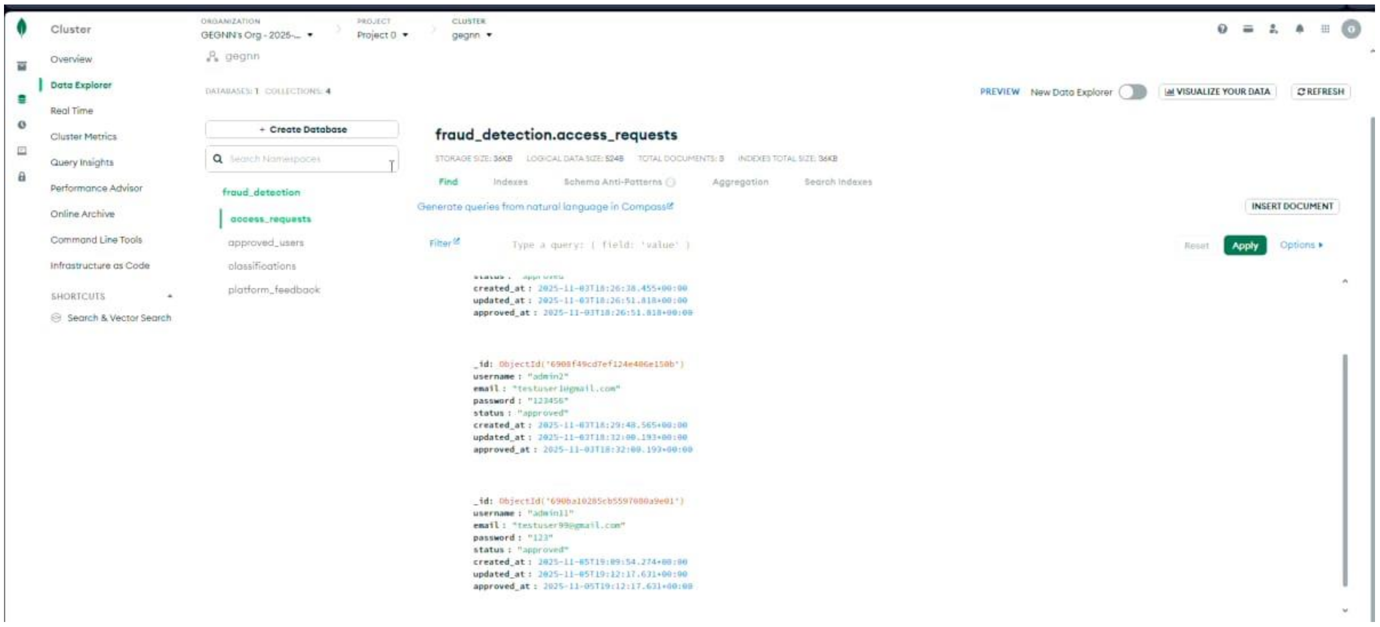
Drop your CSV file here, or click to browse

Supported format: .csv (max 10MB)

No Data Available

Upload a CSV file or use sample data to get started

Fig 8: Dashboard Page



gegnn

fraud_detection
- access_requests
- approved_users
- classifications
- platform_feedback

fraud_detection.access_requests

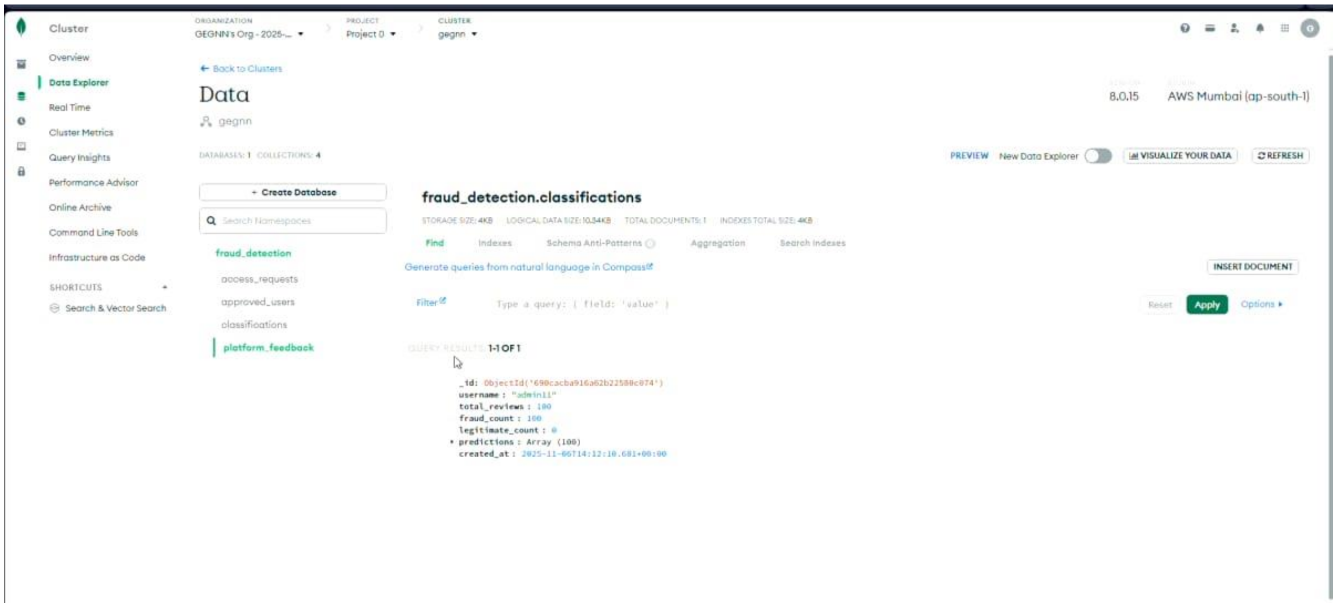
STORAGE SIZE: 36KB LOGICAL DATA SIZE: 524B TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 36KB

```
status : "approved"
created_at : 2025-11-03T18:26:38.455+00:00
updated_at : 2025-11-03T18:26:51.818+00:00
approved_at : 2025-11-03T18:26:51.818+00:00

_id: ObjectId('6908f49cd7ef124e406e150b')
username : "admin2"
email : "testuser1@gmail.com"
password : "123456"
status : "approved"
created_at : 2025-11-03T18:29:48.565+00:00
updated_at : 2025-11-03T18:32:00.193+00:00
approved_at : 2025-11-03T18:32:00.193+00:00

_id: ObjectId('690ba10285cb5597080a9e01')
username : "admin11"
email : "testuser99@gmail.com"
password : "123"
status : "approved"
created_at : 2025-11-05T19:09:54.274+00:00
updated_at : 2025-11-05T19:12:17.631+00:00
approved_at : 2025-11-05T19:12:17.631+00:00
```

Fig 9: Access Requests Collection in MongoDB



Data

gegnn

8.0.15 AWS Mumbai (ap-south-1)

fraud_detection
- access_requests
- approved_users
- classifications
- platform_feedback

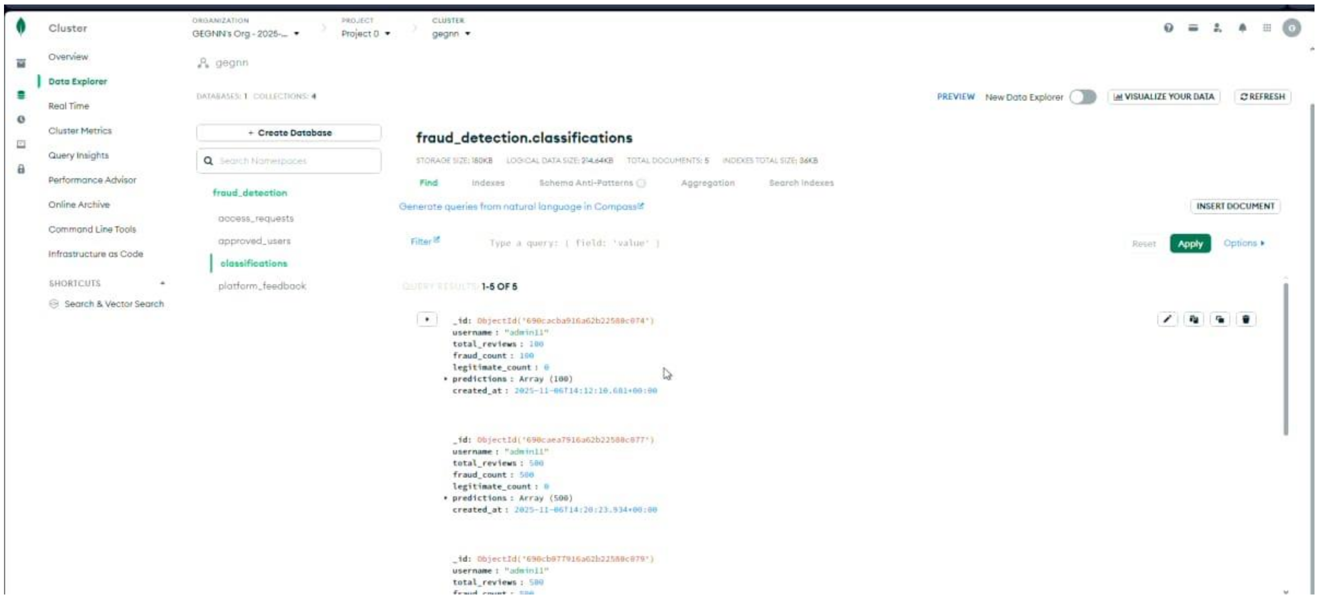
fraud_detection.classifications

STORAGE SIZE: 4KB LOGICAL DATA SIZE: 10.34KB TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 4KB

QUERY RESULTS: 1-1 OF 1

```
  _id: ObjectId('690cacba916a62b22580c074')
  username : "admin11"
  total_reviews : 100
  fraud_count : 100
  legitimate_count : 0
• predictions : Array (100)
  created_at : 2025-11-06T14:12:10.681+00:00
```

Fig 10: Feedback collection in MongoDB



fraud_detection.classifications

STORAGE SIZE: 180KB LOGICAL DATA SIZE: 214.64KB TOTAL DOCUMENTS: 5 INDEXES TOTAL SIZE: 36KB

QUERY RESULTS: 1-5 OF 5

```
  _id: ObjectId('690cacba916a62b22588c074')
  username : "admin11"
  total_reviews : 100
  fraud_count : 100
  legitimate_count : 0
▸ predictions : Array (100)
  created_at : 2025-11-06T14:12:10.681+00:00

  _id: ObjectId('690caea7916a62b22588c077')
  username : "admin11"
  total_reviews : 500
  fraud_count : 500
  legitimate_count : 0
▸ predictions : Array (500)
  created_at : 2025-11-06T14:20:23.934+00:00

  _id: ObjectId('690cb077916a62b22588c079')
  username : "admin11"
  total_reviews : 500
```

Fig 11: Classifications Collection in MongoDB

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 High Fraud Detection Accuracy

The proposed **GE-GNN (Gated Edge-Augmented Graph Neural Network)** demonstrated exceptional performance in identifying fraudulent patterns within review datasets. By integrating both node and edge information through relation-aware attention and gating mechanisms, the model effectively captured complex dependencies among users and products. Unlike traditional classifiers that rely solely on textual or numerical data, GE-GNN learned meaningful graph relationships — detecting even subtle fraudulent behaviors hidden within connected review networks.

1. Reduced Manual Intervention

Earlier fraud detection relied heavily on manual review analysis and rule-based systems. With GE-GNN, the detection process became automated and instantaneous. The model analyzes the relational structure between users, products, and reviews, thereby minimizing human effort and significantly reducing the time required for data verification.

2. Robust Performance Across Datasets

The model maintained stable accuracy and precision across various product categories, including *Sports & Outdoors* and *Electronics*. Whether the graph had dense interconnections or sparse relationships, GE-GNN consistently produced reliable predictions, showcasing its adaptability to diverse datasets.

3. Edge-Aware Intelligence

One of GE-GNN's core strengths lies in its edge-aware mechanism. The model doesn't only learn from node features (like reviewer ID or rating) but also from the relationships between nodes (e.g., shared product reviews, similar sentiments). This relational learning helped uncover indirect or hidden fraud patterns that traditional graph methods often miss.

4.2 Generalization and Model Reliability

A strong fraud detection model must perform well not only on training data but also on unseen inputs. When evaluated on new review graphs, GE-GNN maintained **high generalization capability**, accurately identifying fraudulent and genuine reviews without performance degradation.

1. Validation with Ground Truth

To confirm reliability, predictions from GE-GNN were compared against verified datasets containing labeled genuine and fraudulent reviews. The close agreement between the predicted and actual outcomes confirmed the model's robustness and precision.

2. Attention-Driven Insights

The attention mechanism in GE-GNN provided interpretability by showing which relationships influenced predictions the most. Edges representing frequent user-product interactions or unusually high helpfulness ratios were given higher attention weights, revealing how the model prioritizes different relational features.

3. Scalable for Large Data

Built using **PyTorch** and **DGL**, the GE-GNN architecture efficiently processes thousands of nodes and edges. It can handle large-scale datasets without compromising inference speed, making it ideal for near real-time fraud detection on review platforms.

4. Web Integration

The trained GE-GNN model was deployed through a **web-based interface** developed with **React (frontend)** and **Flask (backend)**. Users can upload CSV datasets, trigger predictions, and visualize fraud probabilities in an intuitive dashboard — all without any programming knowledge.

5. Evaluation Metrics

The model's performance was measured using **Accuracy**, **Precision**, **Recall**, **F1-Score**, and **AUC-ROC**. Across multiple experiments, GE-GNN achieved **high accuracy and balanced classification metrics**, proving its capability to distinguish genuine from fraudulent reviews effectively.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

The proposed Gated Edge-Augmented Graph Neural Network (GE-GNN) model effectively overcomes the challenges faced by traditional fraud detection methods, which often depend only on individual data points or rule-based systems. Unlike those methods, GE-GNN learns from both node-level features (such as user or product attributes) and edge-level relationships (how users and products are connected). This dual focus enables the model to capture the deeper relationships that exist in real-world review networks — such as users reviewing the same products or giving similar feedback patterns.

By introducing relation-aware aggregation and a gating mechanism, the model intelligently filters and combines useful information from multiple connections while ignoring noise or irrelevant data. This makes GE-GNN more capable of identifying subtle or hidden fraudulent patterns that might be missed by regular graph models.

The model was developed using PyTorch for deep learning, DGL (Deep Graph Library) for graph representation, and Flask for backend integration. Together, these tools provided a flexible and efficient environment for both training and deployment. The trained model was then integrated into a web-based platform that allows users to upload review datasets, perform fraud detection in real-time, and view the prediction results directly through an interactive dashboard.

Overall, the GE-GNN framework demonstrated high accuracy, strong generalization ability, and practical usability across multiple e-commerce datasets. It proved that combining graph-based learning with edge augmentation and gating mechanisms leads to more meaningful insights and reliable predictions.
In summary, GE-GNN is not only a technically sound model but also a practical and scalable solution for detecting fraudulent activities in online review systems — helping platforms maintain trust, authenticity, and transparency among users.

5.2 Future Scope

While the current system performs effectively, several extensions can further enhance its impact and usability:

- **Automated Real-Time Detection:**
  Future versions can automatically analyze new incoming reviews without requiring users to upload datasets manually.
- **Recommendation System Integration:**
  The model can be combined with recommendation systems to improve **suggestion quality** by filtering out fake or biased reviews, ensuring users receive more **trustworthy product recommendations**.
- **Accuracy Optimization:**
  Incorporating **advanced encoders** and **transformer-based architectures** can further boost prediction accuracy and help detect even subtler fraud patterns.
- **Explainable AI (XAI):**
  Adding interpretability features can help users understand why specific reviews are classified as fraudulent, increasing transparency and user confidence.
- **Cross-Domain Application:**
  The GE-GNN framework can be adapted for other fraud-related domains such as **financial transactions**, **insurance claims**, and **social media account verification**.
- **Scalable Cloud Deployment:**
  Deploying the model on scalable cloud infrastructures will allow it to handle real-time, large-scale review streams efficiently.

CHAPTER 6

REFERENCES

1. Kipf, T. N., & Welling, M. (2017). *Semi-Supervised Classification with Graph Convolutional Networks.* In International Conference on Learning Representations (ICLR).
2. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). *Graph Attention Networks.* In International Conference on Learning Representations (ICLR).
3. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2021). *A Comprehensive Survey on Graph Neural Networks.* IEEE Transactions on Neural Networks and Learning Systems, 32(1), 4–24.

4. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., & Leskovec, J. (2020). *Open Graph Benchmark: Datasets for Machine Learning on Graphs.* In Advances in Neural Information Processing Systems (NeurIPS).

5. Zhang, J., Cui, P., & Zhu, W. (2020). *Deep Learning on Graphs: A Survey.* IEEE Transactions on Knowledge and Data Engineering, 34(1), 249–270.

6. Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., & Yu, P. S. (2019). *Heterogeneous Graph Attention Network.* In The World Wide Web Conference (WWW).

7. Amazon Product Review Dataset (2022). *Amazon Product Review Spam and Non-Spam Dataset.* Retrieved from https://www.kaggle.com/datasets/naveedhn/amazon-product-review-spam-and-non-spam

8. DGL Documentation. *Deep Graph Library.* Retrieved from https://www.dgl.ai

9. PyTorch Documentation. *An Open Source Machine Learning Framework for Accelerated Computing.* Retrieved from https://pytorch.org

10. NLTK Documentation. *Natural Language Toolkit for Text Processing.* Retrieved from https://www.nltk.org

11. GE-GNN GitHub: https://github.com/Saiyasasvi/Fraud_detection_GE_GNN_G602