GraphSAGE for Node-Level Classification – Alternative Report

Name: Yehia Tarek
ID: 2205062

1. Overview

This experiment aimed to explore how **GraphSAGE**, a popular Graph Neural Network framework, performs when classifying nodes in a simple, hand-crafted graph. By working with a tiny dataset, the intention was to clearly observe how the model uses both node features and network structure to make decisions about whether a user is benign or malicious.

2. How GraphSAGE Works

GraphSAGE is designed to generate node embeddings by **aggregating information from a node's neighbourhood**. Instead of learning individual embeddings for every node, it learns a function that can be applied to any node—even in unseen parts of a graph. This behaviour makes GraphSAGE scalable and ideal for large, constantly changing networks.

A node's final embedding depends on:

- The node's **own feature vector**
- The **features of its neighbouring nodes**
- The **aggregation function** used by the model

This combination naturally highlights irregular patterns, which is valuable in security contexts where malicious accounts often deviate from normal behaviour.

3. Description of the Graph

The graph used for this task contained **six nodes**, divided into two groups:

- **Benign nodes**: 0, 1, 2 → feature vector **[1, 0]**
- **Malicious nodes**: 3, 4, 5 → feature vector **[0, 1]**

Each group formed a **fully connected mini-cluster**, and only one link connected the two clusters:
Node 2 ↔ Node 3

The labels assigned were binary:
0 = benign, **1 = malicious**

This small but structured arrangement makes it easy to observe how information flows across the network.

4. Model Details

The model was built using **two GraphSAGE layers**:

- The **first layer** gathers neighbour information and applies a ReLU activation
- The **second layer** outputs scores for the two possible classes

Since the graph is small, neighbour sampling was effectively unnecessary—each node's full neighbourhood was used during aggregation.

5. Training Summary

The model was trained for **50 epochs**. During each epoch:

1. Predictions were generated
2. The **negative log-likelihood loss** was computed
3. Parameters were updated using the **Adam optimiser**

Because the dataset was cleanly separated into two clusters, training was straightforward and convergence happened quickly.

6. Results

The model ultimately produced the predictions:

[0, 0, 0, 1, 1, 1]

Every node was classified correctly.
The learned embeddings placed benign nodes close together in one part of the vector space and malicious nodes in another. The single bridge between the two groups did not significantly confuse the model, as each node was still mostly connected to others of the same type.

7. Relevance to Security Applications

Even though the example is small, it clearly demonstrates how GraphSAGE can support security-oriented tasks. In real platforms, harmful accounts often appear in unusual neighbourhoods or connect to others in suspicious ways. Aggregation-based GNNs capture these patterns naturally.

Such an approach can be applied to:

- Detecting fake or automated accounts
- Identifying coordinated malicious behaviour
- Flagging nodes with abnormal connectivity