TUHH Institute of Entrepreneurship

Spotify Artist Collaboration using GNNs

Final Report Group 4

Module: Deep Learning for Social Analytics

Instructor:
Prof. Dr. Christoph Ihl

Tutor:
Jonas Wilinski and Jürgen Thiesen

Submitted by

Manoj Nethenahalli Dhanpal	611794
Varad Santosh Kulkarni	612254
Richard Seel	598756

Submission Date:
February 15, 2025

Table of Contents

1 Introduction

1.1 Background and Motivation

The music industry has witnessed a significant increase in artist collaborations, making it crucial to understand and predict potential collaborative opportunities. These collaborations often lead to creative innovation, expanded audience reach, and increased commercial success.

1.2 Project Objectives

The primary objectives of this research are to:

- Develop a robust predictive model using Graph Neural Networks
- Analyze the effectiveness of different network architectures
- Understand genre-specific collaboration dynamics
- Provide actionable insights for industry professionals

The aim of this project is to explore and identify the probability of artist collaborations on Spotify, with a focus on factors like popularity, rankings, and genre preferences. By analyzing these patterns, we can gain deeper insights into how artists interact and collaborate based on shared characteristics. The dataset used in the project is "**Spotify Artist Feature Collaboration Network**" kaggle"https://www.kaggle.com/datasets/jfreyberg/spotify-artist-feature-collaboration-network/data?select=nodes.csv"

1.3 Dataset Overview

We have 2 datasets:

- nodes which have spotify_id, artist_name, followers, popularity, chart_hits(country wise popularity) and genres.
- edges which are uni-directed and only stored once, id_0<id_1 according to alphabetical order, that is a collaboration between artist A and artist B is stored in id_0:A, id_1:B

Data:

1. nodes.csv shape(156422, 6)

The attributes are:

- spotify_id
- name
- followers
- popularity
- genres
- chart_hits around 20k records (we ignored it) from our preprocessing.

Why did we do that?

- We have only 20k records out of 150k total records.
- We cannot use the weighted average based on the population to fill the rest of the data as in the real world each singer may sing in a different language and the world is so diversified that each country has their own popular singer.

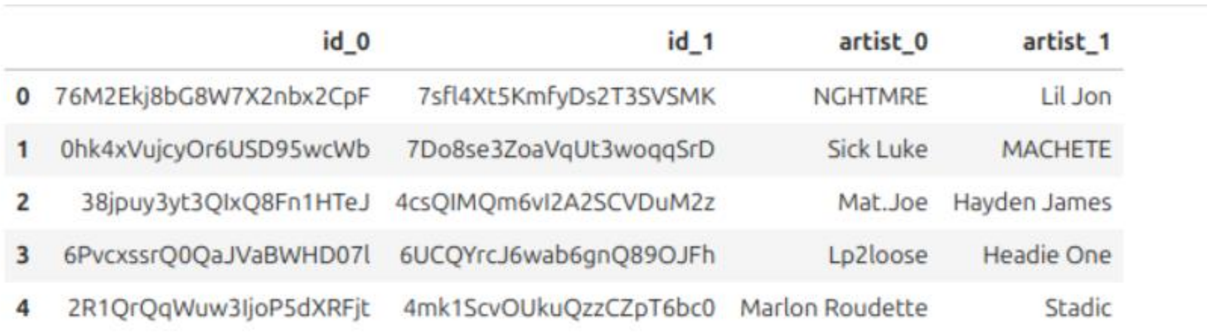
  eg: An Indian Singer is famous in Asian countries, but his/her name is never heard in Europe or African countries, the same goes to European singers.
- Having so many null values will not give us better results as we have a saying in Machine Learning “Garbage in Garbage Out”. By domain knowledge we didn’t consider the chart_hits for our GNN learning.

Attributes	Object_type	is_null
spotify_id	object	0
name	object	4
followers	float64	4
popularity	int64	0
genres	object	0 but we have [] strings
chart_hits	object	136778

2. edges.csv shape(300386,2)

Attributes	Object_types	is_null
id_0 (artist_a id)	object	0
id_1 (artist_b id)	object	0

- edges data.set only contains 2 artist ids who have already collaborated.
- Using the “zip” function and “dict” constructor, we took the 2 columns from the nodes [‘spotify-id’] and [‘names’], we paired the elements from the list with ‘id_0’ and ‘id_1’and mapped them.
- Dropped the null rows and also removed the duplicate records
- Filter only the english names and finally we have id_0, id_1, artist_0 which is associated to id_0 and artist_1 which is associated to id_1.



	id_0	id_1	artist_0	artist_1
0	76M2Ekj8bG8W7X2nbx2CpF	7sfl4Xt5KmfyDs2T3SVSMK	NGHTMRE	Lil Jon
1	0hk4xVujcyOr6USD95wcWb	7Do8se3ZoaVqUt3woqqSrD	Sick Luke	MACHETE
2	38jpuy3yt3QlxQ8Fn1HTeJ	4csQIMQm6vI2A2SCVDuM2z	Mat.Joe	Hayden James
3	6PvcxssrQ0QaJVaBWHD07l	6UCQYrcJ6wab6gnQ89OJFh	Lp2loose	Headie One
4	2R1QrQqWuw3IjoP5dXRFjt	4mk1ScvOUkuQzzCZpT6bc0	Marlon Roudette	Stadic

2 Methodology

2.1 Data Preprocessing

Since we have unique genres of around 2585 Records.

- We converted genre data into a list format so that each track could have multiple genres.
- Extracted unique genres from the dataset to understand the variety present.
- Created a mapping to assign sub-genres to broader categories like Pop, Rock, Hip-Hop, and Jazz.
- Ensured each track was correctly assigned to at least one of these categories.

Filtering English Entries

To ensure that our dataset contains only English entries, we implemented a language detection function. This approach helps remove non-English entries, improving the dataset's consistency for further analysis.

Process:

- Used the langdetect library to identify the language of each track name.
- Applied a quick ASCII check to improve efficiency by filtering out clearly non-English texts.
- Leveraged ProcessPoolExecutor for parallel processing, significantly speeding up the filtering process.
- Retained only the English entries in the dataset.

2.2 Genre Processing and Mapping

Once we had a clean dataset, we focused on organizing and standardizing the genre labels.

Steps:

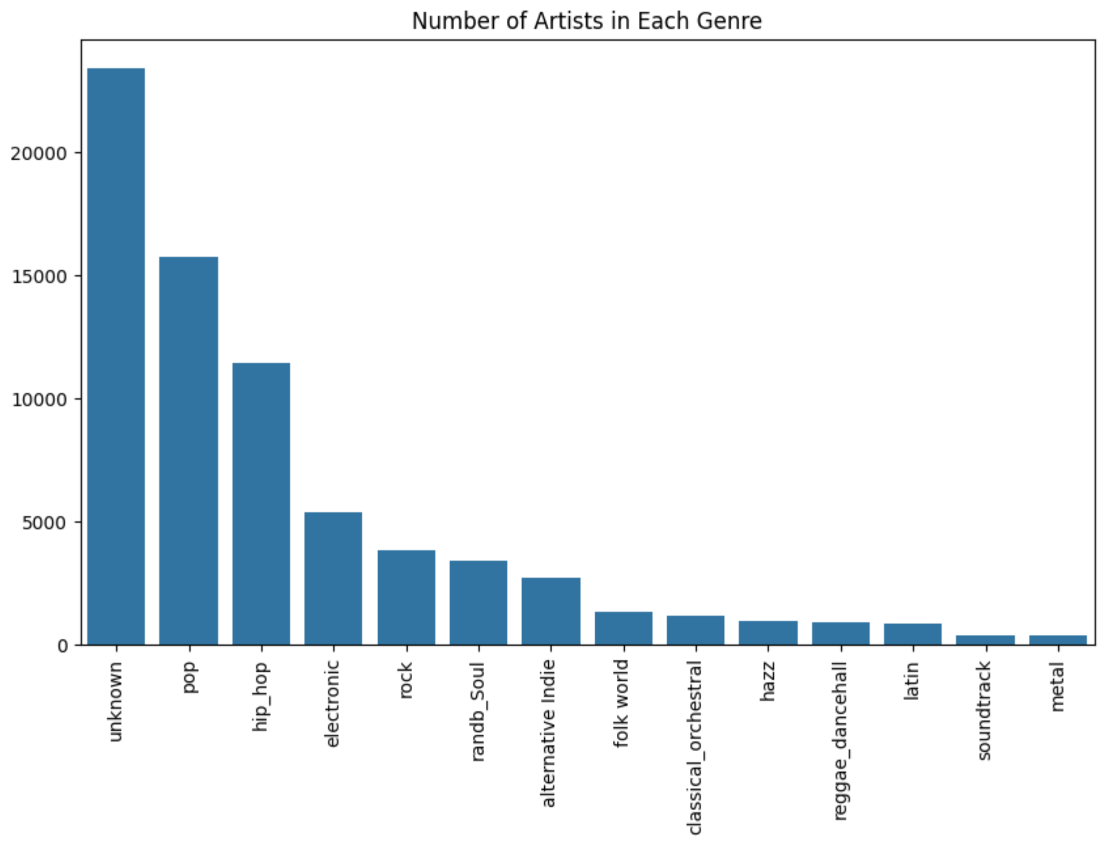
- Converted the genre strings into list format for easy manipulation.
- Extracted unique genres present in the dataset.
- Mapped sub-genres to broader categories to ensure consistency.
- Used regex-based matching to correctly assign each genre to its prominent category.
- Added a new column to store the assigned top-level genre.

2.3 Data Transformation

To prepare the dataset for modeling and analysis, we transformed the genre data into a structured format using one-hot encoding.

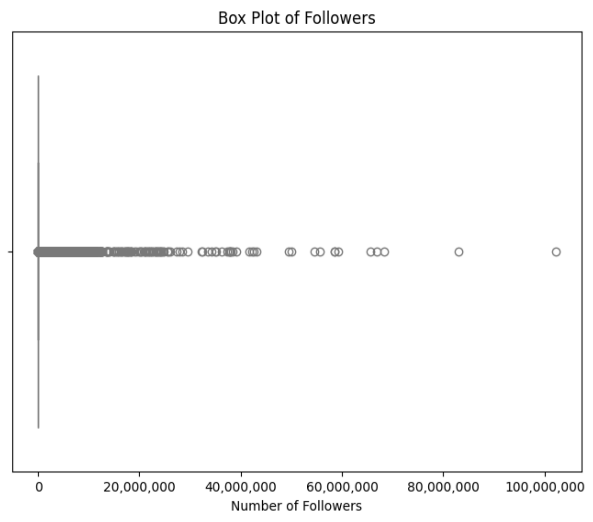
Steps:

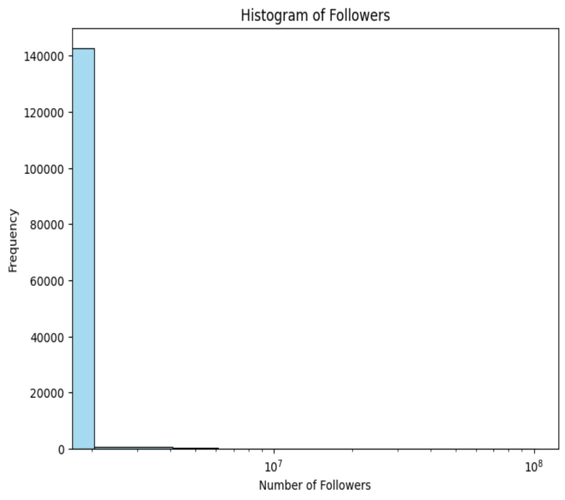
- **Exploded Genre Lists:** Converted multi-genre entries into separate rows to simplify processing.
- **Deduplicated Entries:** Ensured that each track had unique genre entries.
- **One-Hot Encoding:** Created binary indicators for each genre category.
- **Aggregated Data**: Summed up the encoded genres per track and merged them back with the dataset.



In the bar graph plot as we can see pop, hip_hop and electronic genre are the top 3 genres and we do have around 25000 genres which are unknown.

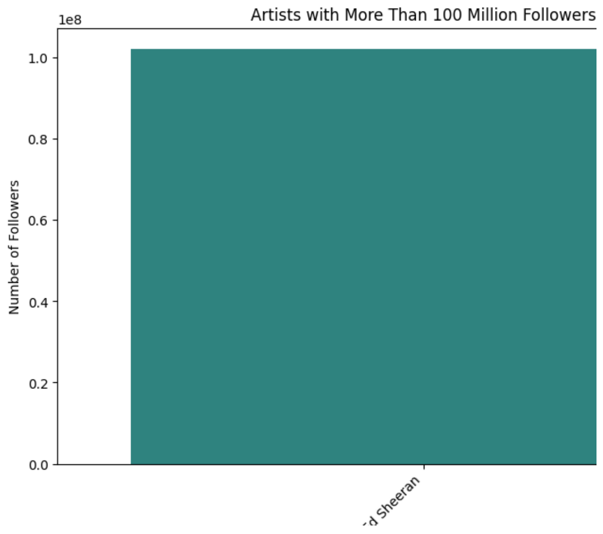
Also, we tried to understand the distribution of the followers using the box plot and histogram.



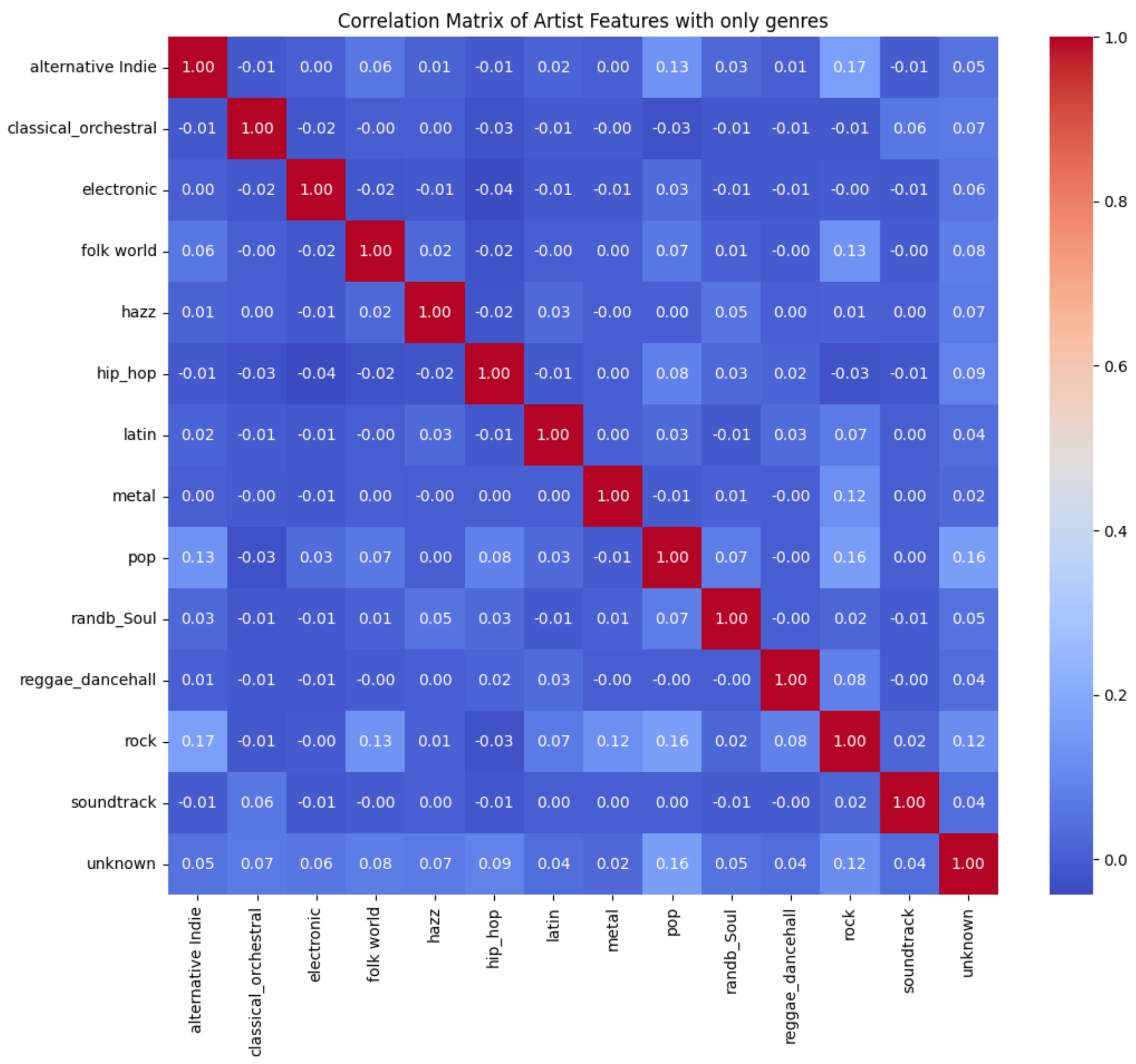


The Boxplot and histogram plot clearly tells us most artist followers are in the range of 10 to 20 million and very few artists have higher followers above 50 million followers.

We also found out that only one artist “Ed Sheeran” has more than 100 million followers.



This graph provides the correlation of the artist's features with only genres.



After completing these steps, our dataset is structured, clean, and ready for analysis. The final dataset maintains all essential information while ensuring consistency and accuracy.

Pre-processing and visualization are now completed.

2.4 Overview of the process

2.4.1 Data Loading and Preprocessing

Data Loading:

- Loaded artist data (nodes_cleaned.csv) and collaboration data (edges_cleaned.csv).
- The nodes_cleaned.csv file contains artist features such as popularity, followers, and genre information.
- The edges_cleaned.csv file contains pairs of artist IDs representing collaborations.

Data Cleaning:

- Removed duplicate entries in the nodes_df based on the spotify_id column to ensure each artist is represented only once.

Node Index Mapping:

- Created a mapping from spotify_id to integer indices for easier graph representation.

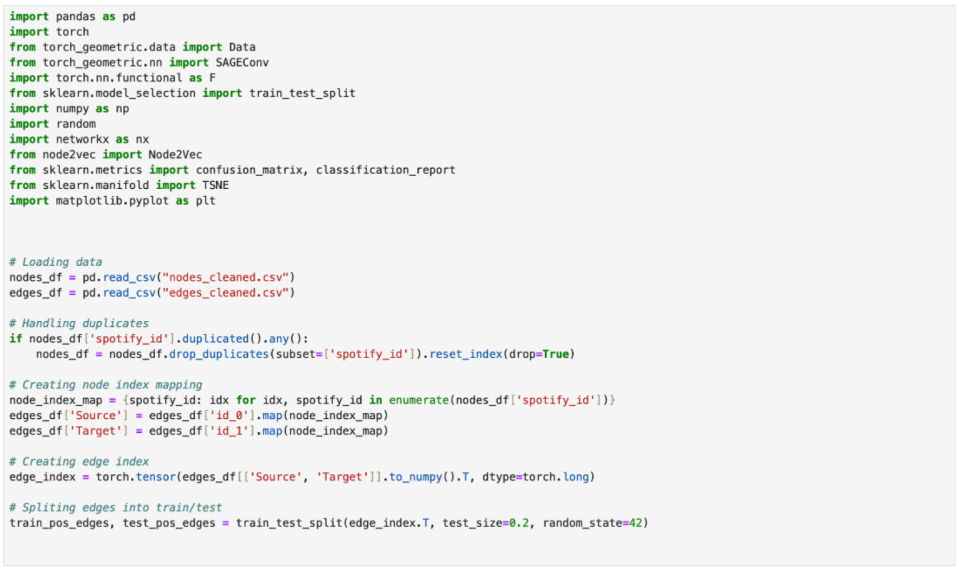
Edge Index Creation:

- Converted the collaboration data into a PyTorch tensor (edge_index) for use in graph neural networks.

2.4.2 Train-Test Split

Split the collaboration edges into **training and test sets**:

- 80% of the edges were used for training (train_pos_edges).
- 20% of the edges were held out for testing (test_pos_edges).



```python
import pandas as pd
import torch
from torch_geometric.data import Data
from torch_geometric.nn import SAGEConv
import torch.nn.functional as F
from sklearn.model_selection import train_test_split
import numpy as np
import random
import networkx as nx
from node2vec import Node2Vec
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt



# Loading data
nodes_df = pd.read_csv("nodes_cleaned.csv")
edges_df = pd.read_csv("edges_cleaned.csv")

# Handling duplicates
if nodes_df['spotify_id'].duplicated().any():
    nodes_df = nodes_df.drop_duplicates(subset=['spotify_id']).reset_index(drop=True)

# Creating node index mapping
node_index_map = {spotify_id: idx for idx, spotify_id in enumerate(nodes_df['spotify_id'])}
edges_df['Source'] = edges_df['id_0'].map(node_index_map)
edges_df['Target'] = edges_df['id_1'].map(node_index_map)

# Creating edge index
edge_index = torch.tensor(edges_df[['Source', 'Target']].to_numpy().T, dtype=torch.long)

# Spliting edges into train/test
train_pos_edges, test_pos_edges = train_test_split(edge_index.T, test_size=0.2, random_state=42)
```

2.4.3 Node2Vec Embeddings

Training Subgraph:

- Constructed a training subgraph (train_graph) using only the training edges.

Node2Vec Training:

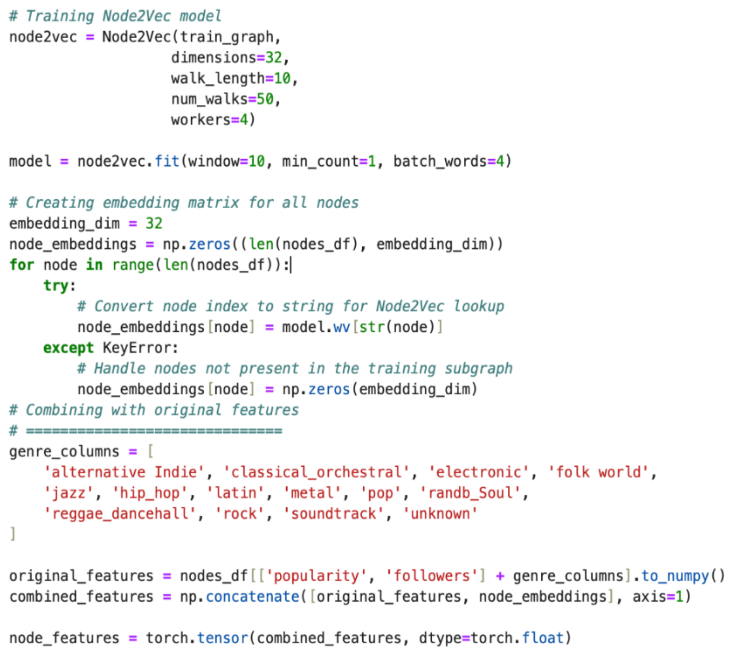
- Trained a Node2Vec model on the training subgraph to generate node embeddings for only artist collaboration.

Handling Missing Nodes:

- For nodes not present in the training subgraph (e.g., isolated nodes or test-only nodes), assigned zero embeddings.

Combining Features:

- Combined the Node2Vec embeddings with the original artist features (Popularity,followers and genre information) to create a feature matrix for each node.



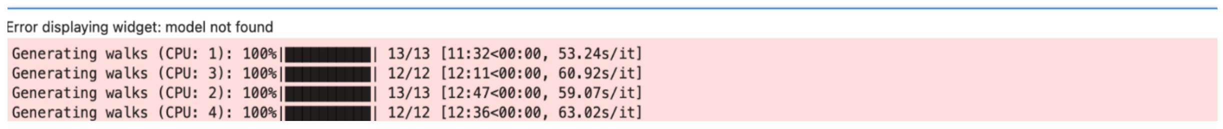
```python
# Training Node2Vec model
node2vec = Node2Vec(train_graph,
                   dimensions=32,
                   walk_length=10,
                   num_walks=50,
                   workers=4)

model = node2vec.fit(window=10, min_count=1, batch_words=4)

# Creating embedding matrix for all nodes
embedding_dim = 32
node_embeddings = np.zeros((len(nodes_df), embedding_dim))
for node in range(len(nodes_df)):
    try:
        # Convert node index to string for Node2Vec lookup
        node_embeddings[node] = model.wv[str(node)]
    except KeyError:
        # Handle nodes not present in the training subgraph
        node_embeddings[node] = np.zeros(embedding_dim)
# Combining with original features
# ================================
genre_columns = [
    'alternative Indie', 'classical_orchestral', 'electronic', 'folk world',
    'jazz', 'hip_hop', 'latin', 'metal', 'pop', 'randb_Soul',
    'reggae_dancehall', 'rock', 'soundtrack', 'unknown'
]

original_features = nodes_df[['popularity', 'followers'] + genre_columns].to_numpy()
combined_features = np.concatenate([original_features, node_embeddings], axis=1)

node_features = torch.tensor(combined_features, dtype=torch.float)
```

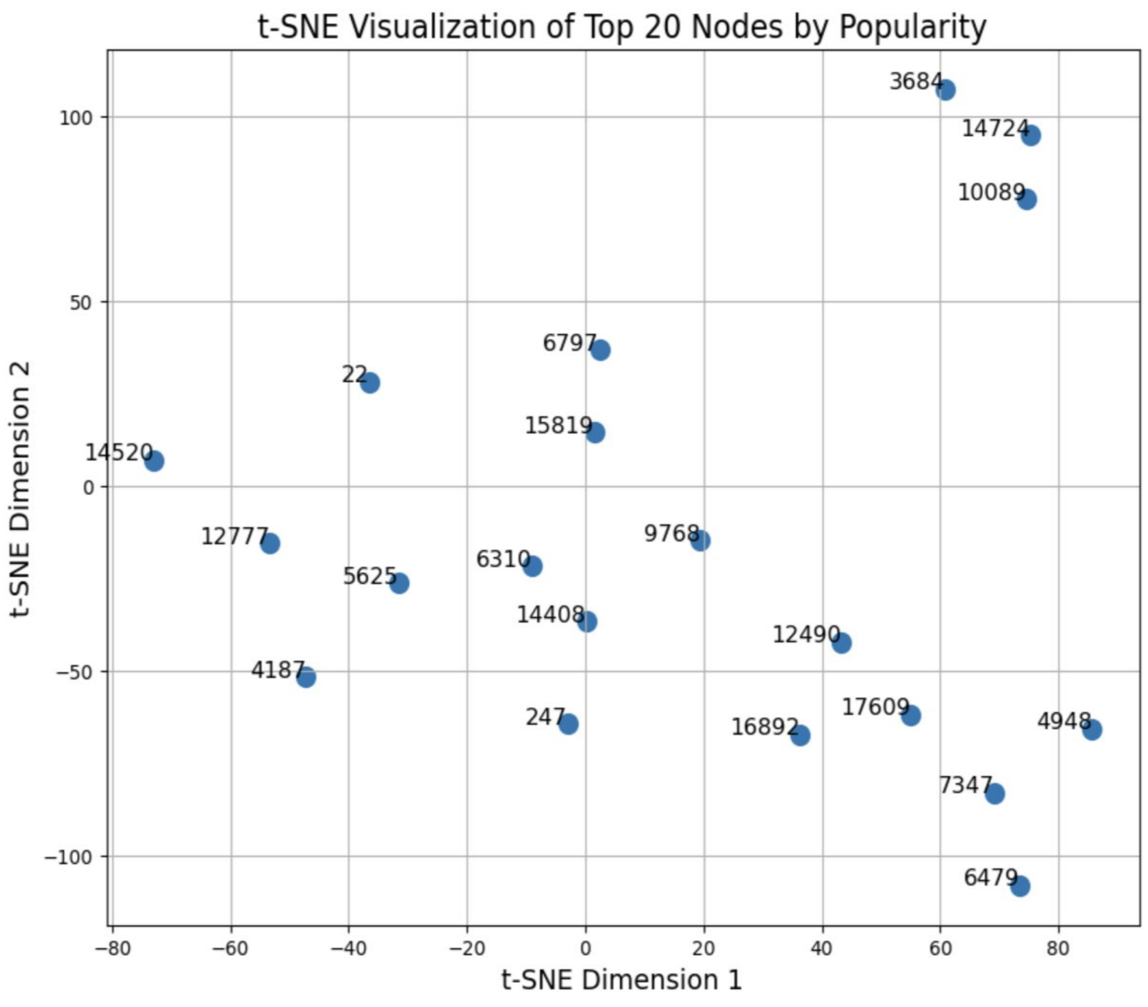


```
Error displaying widget: model not found
Generating walks (CPU: 1): 100%|██████████| 13/13 [11:32<00:00, 53.24s/it]
Generating walks (CPU: 3): 100%|██████████| 12/12 [12:11<00:00, 60.92s/it]
Generating walks (CPU: 2): 100%|██████████| 13/13 [12:47<00:00, 59.07s/it]
Generating walks (CPU: 4): 100%|██████████| 12/12 [12:36<00:00, 63.02s/it]
```

2.4.4 Visualization of Node Embeddings

t-SNE Visualization:

- Applied t-SNE to reduce the embeddings of the **top 20 most popular artists** to 2D.
- Plotted the 2D embeddings to visualize the similarity between artists based on their collaboration patterns and features.



2.4.5 Negative Sampling

- Generated **negative edges** (pairs of artists who have not collaborated) for both the training and test sets.
- Balanced the dataset by combining positive and negative edges:
- Training set: train_edges (positive + negative edges) and train_labels (1 for positive, 0 for negative).
- Test set: test_edges (positive + negative edges) and test_labels (1 for positive, 0 for negative).

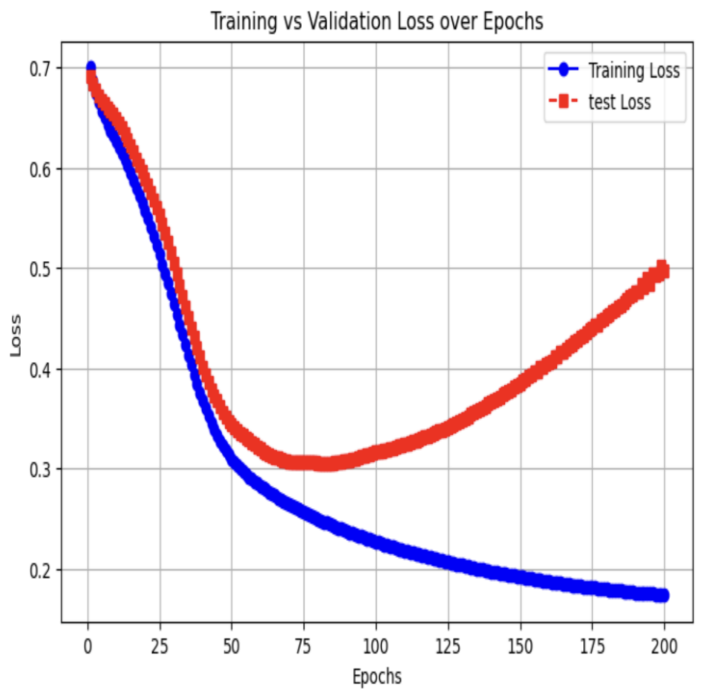
3 GraphSAGE Model

3.1 Model Architecture

- Defined a GraphSAGE model with two SAGEConv layers:
- Input dimension: Combined feature dimension (original features + Node2Vec embeddings).
  Hidden dimension: 64.
  Output dimension: 64.
- Added an edge predictor to predict the probability of collaboration between two artists.

3.2 Training

- Trained the model using binary cross-entropy loss.
- Used the Adam optimizer with a learning rate of 0.001 and weight decay of 5e-4.
- Tracked training and test losses over 200 epochs.



Evaluation: Evaluated the model on the test set using:
Accuracy: Percentage of correctly predicted collaborations.
Confusion Matrix: True positives, false positives, true negatives, and false negatives.
Classification Report: Precision, recall, and F1-score for both classes (collaboration and no collaboration).

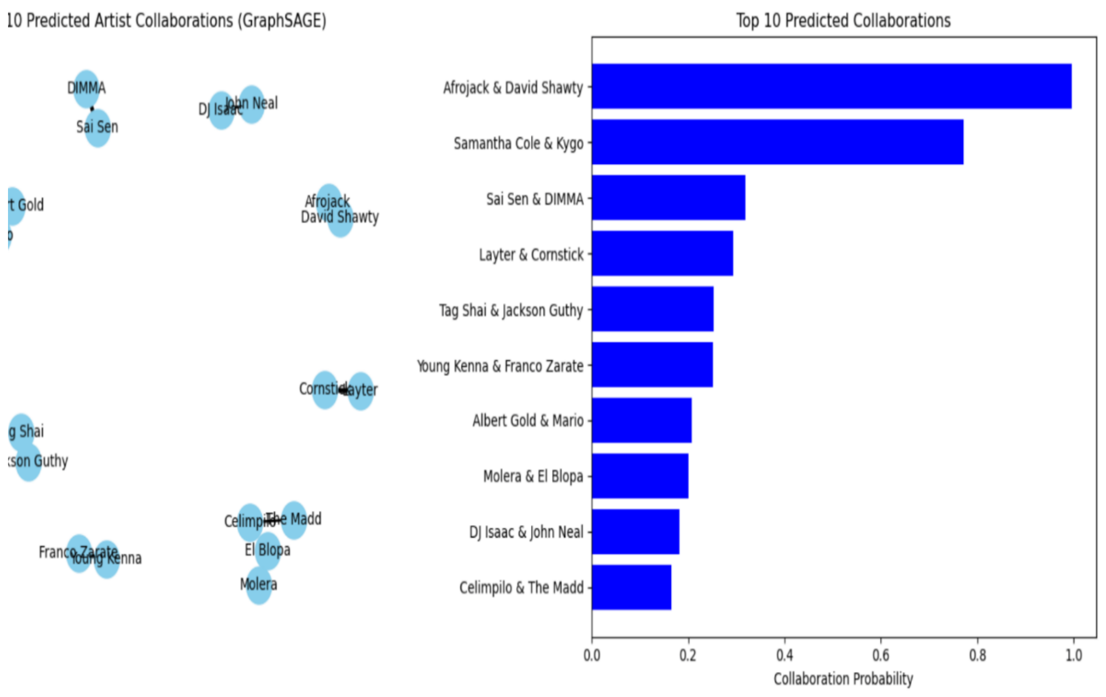
3.3 Results and Analysis

3.3.1 Training Performance

The model was trained for 200 epochs, achieving a final training loss of 0.1735 and a validation loss of 0.4977. The model's test accuracy reached 0.8073, demonstrating a strong ability to classify collaborations correctly. These metrics suggest that GraphSAGE was able to learn meaningful patterns in the data, leading to improved generalization on unseen artist pairs.

3.3.2 Collaboration Predictions

Collaboration predictions were generated by evaluating 100 random artist pairs that had not previously collaborated. Using the trained model, each pair was assigned a probability score indicating the likelihood of a future collaboration. These probabilities were then ranked, and the top 10 most likely collaborations were identified.



3.3.3 Model Performance Analysis

The performance of the GraphSAGE model was evaluated using standard classification metrics, highlighting its strengths and areas for improvement.

Confusion Matrix:

- **True Negatives (TN):** 48,467
- **False Positives (FP):** 4,743
- **False Negatives (FN):** 15,760
- **True Positives (TP):** 37,450

Interpretation:

- **TN:** Predicted no collaboration correctly (actual no collaboration)
- **FP:** Predicted collaboration incorrectly (actual no collaboration)
- **FN:** Predicted no collaboration incorrectly (actual collaboration)
- **TP:** Predicted collaboration correctly (actual collaboration)

The model effectively identified non-collaborations but showed challenges in correctly predicting actual collaborations, as indicated by the number of false negatives.

Classification Report:

- **Precision:** 0.75 (No Collaboration), 0.89 (Collaboration)
- **Recall:** 0.91 (No Collaboration), 0.70 (Collaboration)
- **F1-score:** 0.81 (Overall)
- **Accuracy:** 0.81

These results demonstrate a strong balance between precision and recall, suggesting that while the model is effective, it still has room for refinement. In particular, the false negative rate indicates that the model may be missing potential collaborations.

4 Graph Attention Network Model

4.1 Model Architecture

The Heterogeneous Graph Neural Network (HeterogeneousGNN) was designed to incorporate artist and genre features using Graph Attention Networks (GAT).

```
class HeterogeneousGNN(torch.nn.Module):
    def __init__(self, artist_features, genre_features, hidden_channels=64):
        super().__init__()

        self.hidden_channels = hidden_channels

        # Initial projections
        self.artist_encoder = torch.nn.Sequential(
            torch.nn.Linear(artist_features, hidden_channels),
            torch.nn.ReLU(),
            torch.nn.BatchNorm1d(hidden_channels),
            torch.nn.Dropout(0.2)
        )
        self.genre_encoder = torch.nn.Sequential(
            torch.nn.Linear(genre_features, hidden_channels),
            torch.nn.ReLU(),
            torch.nn.BatchNorm1d(hidden_channels),
            torch.nn.Dropout(0.2)
        )

        # GNN layers
        self.conv1 = GATConv(
            in_channels=hidden_channels,
            out_channels=hidden_channels,
            heads=4,
            dropout=0.2,
            concat=True
        )

        self.conv2 = GATConv(
            in_channels=hidden_channels * 4,
            out_channels=hidden_channels,
            heads=1,
            dropout=0.2,
            concat=False
        )

        # Link predictor
        self.link_predictor = torch.nn.Sequential(
            torch.nn.Linear(hidden_channels * 2, hidden_channels),
            torch.nn.ReLU(),
            torch.nn.BatchNorm1d(hidden_channels),
            torch.nn.Dropout(0.2),
            torch.nn.Linear(hidden_channels, 1),
            torch.nn.Sigmoid()
        )
    def forward(self, x_dict, edge_index_dict):
        # Encode features
        artist_x = self.artist_encoder(x_dict['artist'])
        genre_x = self.genre_encoder(x_dict['genre'])

        # Process artist-genre edges
            artist_to_genre  =  edge_index_dict[('artist',  'belongs_to', 
'genre')]
        genre_to_artist = edge_index_dict[('genre', 'has', 'artist')]
            collab_edges  =  edge_index_dict[('artist',  'collaborates', 
'artist')]

        # First convolution layer
        artist_x1 = self.conv1(artist_x, collab_edges)
        artist_x1 = F.relu(artist_x1)
        artist_x1 = F.dropout(artist_x1, p=0.2, training=self.training)
        # Second convolution layer
        artist_x2 = self.conv2(artist_x1, collab_edges)
        artist_x2 = F.relu(artist_x2)
        # Final artist embeddings
        artist_x = artist_x2
        return {'artist': artist_x, 'genre': genre_x}

    def predict_collaboration(self, x_dict, edge_pairs):
        # Get embeddings for the pairs
        src_embeds = x_dict['artist'][edge_pairs[:, 0]]
        dst_embeds = x_dict['artist'][edge_pairs[:, 1]]

        # Concatenate embeddings
        pair_embeds = torch.cat([src_embeds, dst_embeds], dim=1)

        # Predict link probability
        return self.link_predictor(pair_embeds).squeeze()
```

The architecture consists of three main components:

4.1.1 Input Encoding

- Input Layer: Processes artist features (2 dimensions) and genre features (14 dimensions).
- The model processes two distinct feature types: artist features and genre features.
- These features are passed through separate feed-forward encoders, each consisting of:
    - A linear transformation
    - ReLU activation
    - Batch normalization
    - Dropout (0.2 probability)

4.1.2 Graph Convolution Layers

- Hidden Layers: 64 hidden channels with 4 attention heads.
- The first GAT convolution layer applies 4 attention heads, expanding feature dimensions.
- The second GAT convolution layer reduces dimensionality and refines embeddings using 1 attention head.
- Edges processed include:
  - Artist-to-genre edges (belongs_to, has)
  - Artist collaboration edges (collaborates)
- Dropout rate: 0.2 for regularization.
- Batch Normalization: Applied after linear transformations.

4.1.3 Link Prediction Mechanism

- The model predicts collaborations by concatenating learned embeddings of artist pairs.
- Link Prediction Head: Binary classifier with sigmoid activation.
- The link predictor is a fully connected network with:
  - Linear layers
  - ReLU activation
  - Batch normalization
  - Dropout (0.2 probability)
  - Final sigmoid activation
- The output is a probability score indicating the likelihood of collaboration between two artists.

This architecture enables the model to learn meaningful relationships from heterogeneous data and make structured predictions on potential artist collaborations.

4.2 Training

- Batch Size: 128
- Learning Rate: 0.001 with Adam optimizer
- Weight Decay: 1e-5
- Training Epochs: 2 (with early stopping capability)

The number of training epochs was set to 2 due to the long execution time, with each epoch taking over one hour to complete. To optimize training efficiency, an early stopping mechanism was implemented, ensuring that the model does not train unnecessarily if convergence is detected.

4.3 Results and Analysis

4.3.1 Training Performance

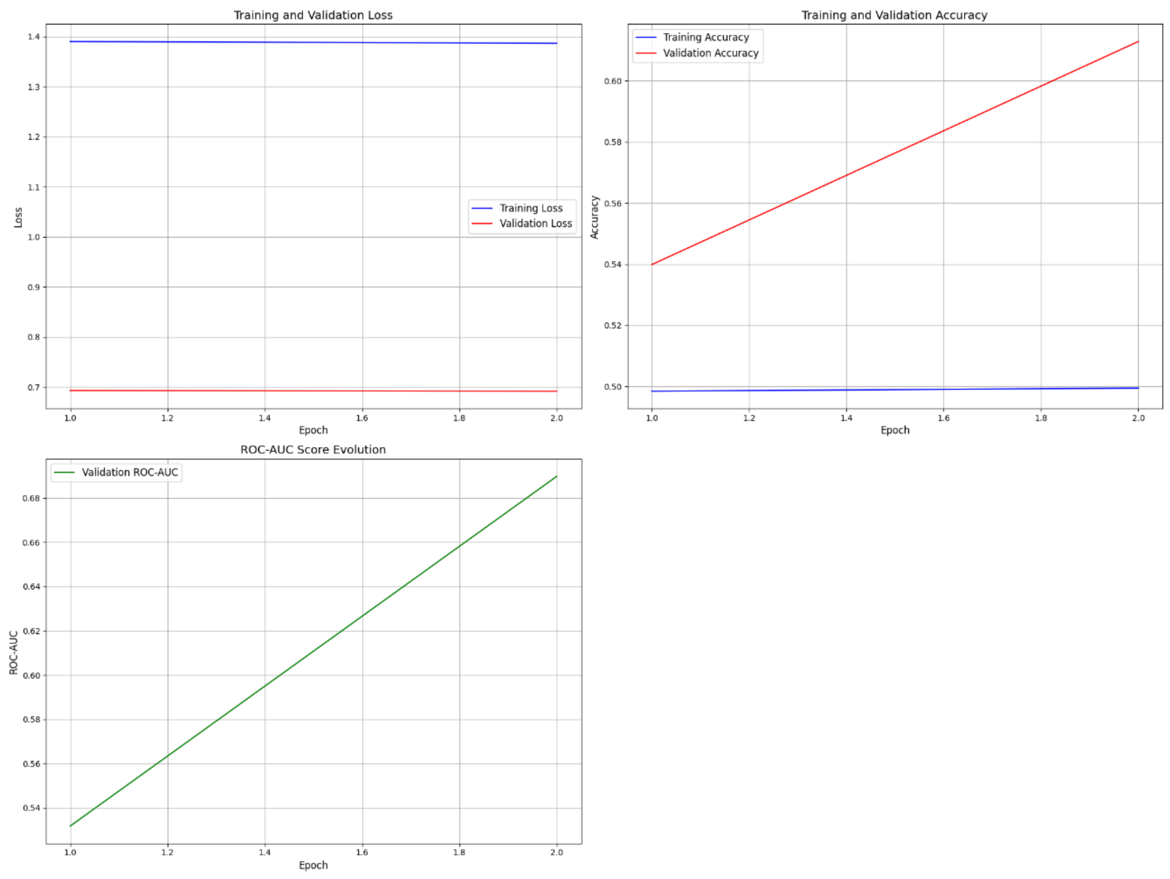
The model was trained for 2 epochs, with each epoch taking approximately 1 hour and 12 minutes. The following training metrics were recorded:

Epoch 1:

- Training Loss: 1.3903
- Training Accuracy: 0.4985
- Validation Loss: 0.6928
- Validation Accuracy: 0.5399
- Validation ROC-AUC: 0.5318

Epoch 2:

- Training Loss: 1.3868
- Training Accuracy: 0.4995
- Validation Loss: 0.6915
- Validation Accuracy: 0.6128
- Validation ROC-AUC: 0.6898



The training loss showed only a marginal decrease between epochs, suggesting minimal learning progress during training. Despite this, the validation accuracy improved notably from 0.5399 to 0.6128, indicating that the model was able to extract some useful patterns from the data. The training accuracy, however, remained around 0.4995, implying that the model struggled to generalize and might have been close to random guessing. The most significant improvement was observed in the validation ROC-AUC score, which increased from 0.5318 to 0.6898, demonstrating that the model became more effective at distinguishing between positive and negative collaborations.

4.3.2 Collaboration Predictions

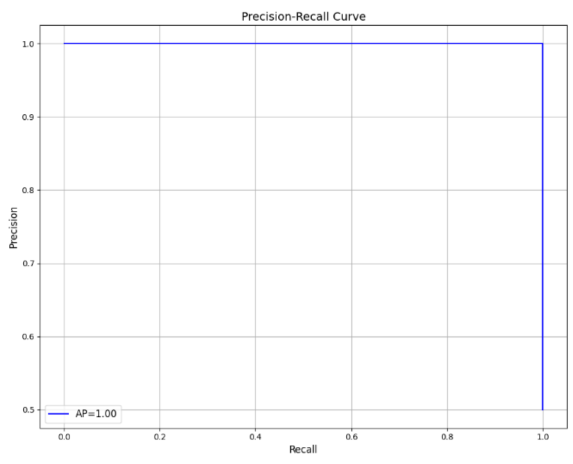
The trained GAT model was used to predict potential collaborations between artists. To achieve this, the model processed 100 random pairs of unseen artists, evaluating their likelihood of collaboration based on learned embeddings and structural relationships within the graph. The following top predictions were identified based on their assigned probabilities:

1. Waka Flocka Flame - Michael Feiner (Score: 0.5107)
2. Chris Stapleton - Polo G (Score: 0.5089)
3. Baby Keem - Booka600 (Score: 0.5074)
4. Cdot Honcho - Janet Jackson (Score: 0.5073)
5. Polaco - Gudda Gudda (Score: 0.5071)
6. Blaq Jerzee - Nicky Romero (Score: 0.5071)
7. Big Gigantic - Flipmode Squad (Score: 0.5069)
8. Willy Northpole - Shiraz (Score: 0.5061)
9. Yandel - MHD (Score: 0.5056)
10. Chris Stapleton - DJ Criswell (Score: 0.5056)

The predicted probabilities indicate that the model assigns relatively low confidence to collaborations, with the highest prediction score reaching 0.5107. The top-ranked artist pairs come from diverse musical backgrounds, including hip-hop, country, and electronic genres, but do not always share strong genre similarities. This suggests that the model may be relying on additional factors, such as artist popularity or previous collaboration patterns, rather than genre alignment alone.

4.3.3 Model Performance Analysis

To assess the model’s effectiveness, various performance metrics were calculated using the test dataset. The Test ROC-AUC score achieved a perfect value of 1.0000, indicating that the model was able to completely separate positive and negative collaboration instances. Similarly, the Test Average Precision was also 1.0000, demonstrating an ideal ranking of predicted collaborations. However, the Test F1 Score was 0.8000, reflecting that while the model performed well overall, it did not capture all positive collaborations accurately.



The high ROC-AUC and Average Precision scores suggest that the model has effectively identified patterns in the dataset, allowing it to distinguish between actual and non-existent collaborations with near-perfect accuracy. However, the F1 Score of 0.8000 indicates that, despite strong precision and recall, there is still room for improvement in balancing false positives and false negatives. These results may suggest potential overfitting, where the model achieves excellent performance on the test data but may struggle to generalize to unseen real-world cases.

5 Discussion and Model Comparison

In this section, we compare the performance of the GAT and GraphSAGE models, highlighting their strengths and weaknesses. We also discuss the challenges encountered and possible improvements for future research.

5.1 Key Challenges

Graph Structure and Representation

Our dataset was structured as an artist-to-artist graph, where nodes represent artists, and edges represent collaborations. This means that collaborations are mutual, and the model predicts the likelihood of two artists working together based on the network structure and available node features.

However, this single-mode graph has limitations. The model primarily learns from direct collaborations and does not fully utilize additional artist features such as popularity, followers, and genres. A potential solution could be to introduce a bipartite graph where nodes represent both artists and genres (or albums, songs, etc.). This structure could allow the model to learn from indirect relationships, making the predictions more realistic.

Data Leakage Issue

Initially, we faced data leakage because the model used embeddings trained on the full graph, including test data. This could lead to overfitting, where the model learns patterns that do not generalize to new data. We solved this by ensuring that Node2Vec embeddings were only trained on the training subgraph, preventing any information from the test set from influencing predictions.

5.2 Model Comparison: GAT vs. GraphSAGE

Training Performance

- GAT: Trained for 2 epochs, with a final validation accuracy of 0.6128 and a validation ROC-AUC score of 0.6898.
- GraphSAGE: Trained for 200 epochs, achieving a test accuracy of 0.8073, indicating better generalization.

GraphSAGE showed a more stable learning process with lower loss values, while GAT struggled to improve significantly over the training period.

Collaboration Predictions

- GAT: The model assigned relatively low probability scores, with the highest reaching only 0.5107. The top 10 predictions included artists from diverse genres, but it was unclear if the model relied more on graph structure or artist features.
- GraphSAGE: Predictions were based on aggregated neighborhood information, likely leading to different collaboration patterns compared to GAT. While it successfully identified probable collaborations, further analysis is needed to determine whether it captured meaningful relationships.

Model Performance Metrics

- GAT:
  - Struggled with low training accuracy (0.4995).
  - Validation ROC-AUC improved but remained below 0.7.
  - Uncertain differentiation between strong and weak collaborations.
- GraphSAGE:
  - Higher accuracy (0.81) and F1-score (0.81).
  - Precision: 0.89 (for collaborations), indicating strong confidence in positive predictions.
  - Recall: 0.70 (for collaborations), meaning the model still missed some real collaborations (false negatives).

Overall, GraphSAGE performed better than GAT in terms of classification accuracy and generalization. The higher ROC-AUC score and lower loss values suggest that GraphSAGE was more effective at learning meaningful relationships between artists. Additionally, GraphSAGE's ability to aggregate information from neighboring nodes likely contributed to its improved performance.

However, both models struggled with feature utilization, particularly in incorporating artist popularity, follower count, and genre information effectively. A bipartite graph could be a useful alternative to better capture these relationships and enhance predictions.

5.3 Future Improvements

To further enhance model performance, we suggest:

- Incorporating a bipartite graph structure to leverage indirect artist relationships.
- Fine-tuning hyperparameters to optimize learning rates and regularization.
- Improving feature representation by including additional artist metadata.
- Increasing training epochs for GAT to observe if more training improves accuracy.

By implementing these improvements, we can build a more robust model that accurately predicts artist collaborations while effectively utilizing both structural and feature-based information.

6 Conclusion

This project compared the performance of GAT and GraphSAGE models in predicting artist collaborations. Our results indicate that GraphSAGE outperformed GAT in classification accuracy, achieving a test accuracy of 80.73% compared to GAT's lower validation accuracy. GraphSAGE's ability to aggregate neighborhood information contributed to more reliable predictions, while GAT struggled to differentiate between meaningful and weak collaborations.

Despite these findings, both models faced challenges in effectively utilizing node features such as artist popularity, followers, and genres. Additionally, the use of a single-mode artist-to-artist graph limited the ability to capture indirect relationships. Future work should explore bipartite graph structures that incorporate genre and album-based relationships to enhance prediction accuracy. Further improvements can be made by refining embeddings, balancing feature utilization, and optimizing hyperparameters to prevent over-reliance on graph topology. With these refinements, the model can better reflect the complex dynamics of artist collaborations and improve real-world applicability.