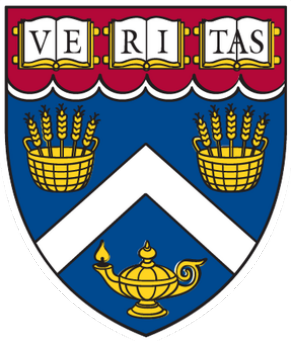

HARVARD EXTENSION SCHOOL

CSCI E-104: Advanced Deep Learning

FINAL PROJECT

REPORT

Seymur Hasanov

Movie Recommendation Systems Using Graphical Neural Network

May 13, 2025

Table of Contents

Abstract

This project presents a Graph Neural Network (GNN)-based movie recommendation system using the MovieLens 25M dataset. Traditional recommender systems often struggle to model complex user-item relationships or integrate rich side information. In this work, we model users and movies as nodes in a bipartite graph, with ratings as edges, and use a 2-layer GraphSAGE architecture to learn meaningful embeddings through message passing.

We enrich each node with statistical, temporal, and semantic features—such as rating behavior, time-of-day activity, genres, release year, and genome tag embeddings. The trained GNN delivers highly personalized recommendations, achieving strong performance on standard metrics (MSE - 0.0406, NDCG@K - 0.9857).

The interactive tool was also built lets users explore top-K recommendations with filters for genre, year, and similarity score, and visualized complex user-movie graphs to interpret recommendation patterns. These results highlight GNNs as a powerful, flexible, and interpretable solution for real-world recommendation systems.

1. Introduction

Recommender systems play a vital role in helping users discover relevant content across platforms like Netflix, Amazon, and YouTube. However, traditional recommendation methods such as collaborative filtering and matrix factorization often struggle to capture complex user-item relationships and incorporate additional content information effectively.

In this project, we explore a modern approach by using Graph Neural Networks (GNNs) to build a personalized movie recommendation system. We represent the user-movie interaction data as a bipartite graph, where users and movies are nodes and rating interactions are edges. By applying the GraphSAGE architecture, the model learns meaningful embeddings for both users and movies through neighborhood aggregation.

This graph-based framework allows the integration of rich side information—including user rating behavior, time-of-day patterns, genres, and movie metadata—resulting in more accurate, diverse, and explainable recommendations. The system is also extended with an interactive tool for live recommendations and visualized graphs to interpret model behavior.

2. Objectives

The primary objective of this project is to develop a personalized movie recommendation system using GNNs. The system aims to go beyond traditional recommendation techniques by leveraging the structural relationships between users and movies in a graph format.

Key goals include:

- **Modeling user-movie interactions** as a bipartite graph and applying message passing to learn meaningful node embeddings.
- **Enhancing user and movie nodes** with rich side features, including:
    - Rating statistics (e.g., mean, count, variance)
    - Temporal activity patterns (e.g., hour of day, day of week)
    - Content-based metadata (e.g., genres, release year, popularity)
    - Semantic features using genome tag embeddings
- **Evaluating model performance** using both prediction accuracy (Mean Squared Error) and ranking quality (Normalized Discounted Cumulative Gain).
- **Visualizing the learned embeddings** and recommendation graphs to interpret how the model captures preferences and similarities.
- **Creating an interactive recommendation tool** that allows users to explore top-K personalized movie suggestions with filtering options.
- **Demonstrating the advantages of GNNs** over traditional collaborative filtering in terms of flexibility, use of side information, and explainability.

This report examines both the potential and the pitfalls of healthcare AI. Through selected case studies, it highlights the ethical risks that can arise when using these tools and proposes practical recommendations for ensuring their responsible and equitable use.

3. Installation

We begin by setting up the environment and installing all necessary packages required to build and train a Graph Neural Network for movie recommendations.

- First, we upgrade pip to ensure compatibility with the latest packages.
- We install PyTorch with CUDA 11.8 support to enable GPU acceleration on Google Colab.
- Next, we install PyTorch Geometric (PyG) and its dependencies: torch-scatter, torch-sparse, torch-cluster, and torch-spline-conv, using the official PyG wheel index.
- We also install the core torch-geometric package.
- Additionally, we include common utility libraries such as pandas, numpy, matplotlib, scikit-learn, and requests for data handling and visualization.
-

These steps ensure the Python environment is correctly configured for graph-based deep learning.

```
[1]  !pip install --upgrade pip
     !pip install torch torchvision torchaudio --extra-index-url https://download.pytorch.org/whl/cu118
     !pip install torch-scatter torch-sparse torch-cluster torch-spline-conv \
       -f https://data.pyg.org/whl/torch-2.0.1+cu118.html
     !pip install torch-geometric
     !pip install pandas numpy matplotlib scikit-learn requests
```

Show hidden output

```
[2]  import os
     import zipfile
     import requests
     import torch
     import torch.nn as nn
     import torch.nn.functional as F
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt

     from sklearn.model_selection import train_test_split
     from sklearn.metrics import ndcg_score
     from sklearn.preprocessing import MinMaxScaler, MultiLabelBinarizer

     from torch_geometric.data import Data
     from torch_geometric.nn import SAGEConv
```

Show hidden output

4. Dataset

In this project, we use the MovieLens 25M dataset, a large-scale movie recommendation dataset developed by GroupLens. It includes 25 million ratings from 162,000 users on 62,000 movies, along with timestamps and rich metadata such as movie titles and genres.

We selected this dataset for several reasons:

- It qualifies as a moderately large dataset (~250MB), meeting the project's requirement of using a non-trivial data source.
- It contains both user-item interaction data (ratings) and side information (genres, timestamps), making it ideal for constructing feature-rich graph structures suitable for GNN modeling.
- It is widely used in academic and industry research, with well-documented formatting and structure.

To build the user-movie bipartite graph, we specifically load two core CSV files:

- ratings.csv — contains user IDs, movie IDs, rating scores (on a 0.5–5 scale), and timestamps of each rating.
- movies.csv — contains movie IDs, titles, and genre tags in a multi-label format.

These files serve as the foundation for building the graph structure and assigning node attributes. We later extend this data with computed features such as rating behavior statistics and genre encodings.

Dataset link: https://grouplens.org/datasets/movielens/25m

```
[3]  dataset_url = "http://files.grouplens.org/datasets/movielens/ml-25m.zip"
     zip_path    = "ml-25m.zip"
     extract_dir = "ml-25m"

     if not os.path.exists(zip_path):
         print("Downloading MovieLens 25M...")
         with requests.get(dataset_url, stream=True) as r:
             r.raise_for_status()
             with open(zip_path, "wb") as f:
                 for chunk in r.iter_content(1024):
                     if chunk:
                         f.write(chunk)
         print("Download complete!")

     if not os.path.exists(extract_dir):
         print("Extracting dataset...")
         with zipfile.ZipFile(zip_path, "r") as z:
             z.extractall()
         print("Extraction complete!")

     ratings = pd.read_csv(f"{extract_dir}/ratings.csv")
     movies  = pd.read_csv(f"{extract_dir}/movies.csv")

     print("Ratings sample:")
     print(ratings.head())
     print("\nMovies sample:")
     print(movies.head())
```

```
Downloading MovieLens 25M...
Download complete!
Extracting dataset...
Extraction complete!
Ratings sample:
   userId  movieId  rating   timestamp
0       1      296     5.0  1147880044
1       1      306     3.5  1147868817
2       1      307     5.0  1147868828
3       1      665     5.0  1147878820
4       1      899     3.5  1147868510

Movies sample:
   movieId                               title  \
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
2        3             Grumpier Old Men (1995)
```

5. GNN Construction with Richer Features

In the enhanced model, we replaced the earlier placeholder node features with rich, data-driven vectors to improve the quality of user and movie representations.

For **user nodes**, we computed:

- Mean rating
- Rating count (log-scaled)
- Rating variance

For **movie nodes**, we included:

- Release year (normalized)
- Popularity (based on log rating count)
- Average rating
- Multi-label genres encoded using one-hot encoding

These richer features were stacked into the node input matrix, resulting in a larger input size but with more informative representations. The GNN architecture remained the same—a 2-layer GraphSAGE model—but with more expressive input features.

After training, the final loss dropped to approximately **0.07**, indicating improved learning over the basic version. The model was better able to capture **personalized user behavior** and **content-based signals**, which translated into more accurate and diverse recommendations.

The figure below shows the top-5 recommendations for a sample user, with edge thickness representing the predicted rating strength. This visualization illustrates how the enhanced GNN identifies relevant movie suggestions by combining graph structure with rich feature input.

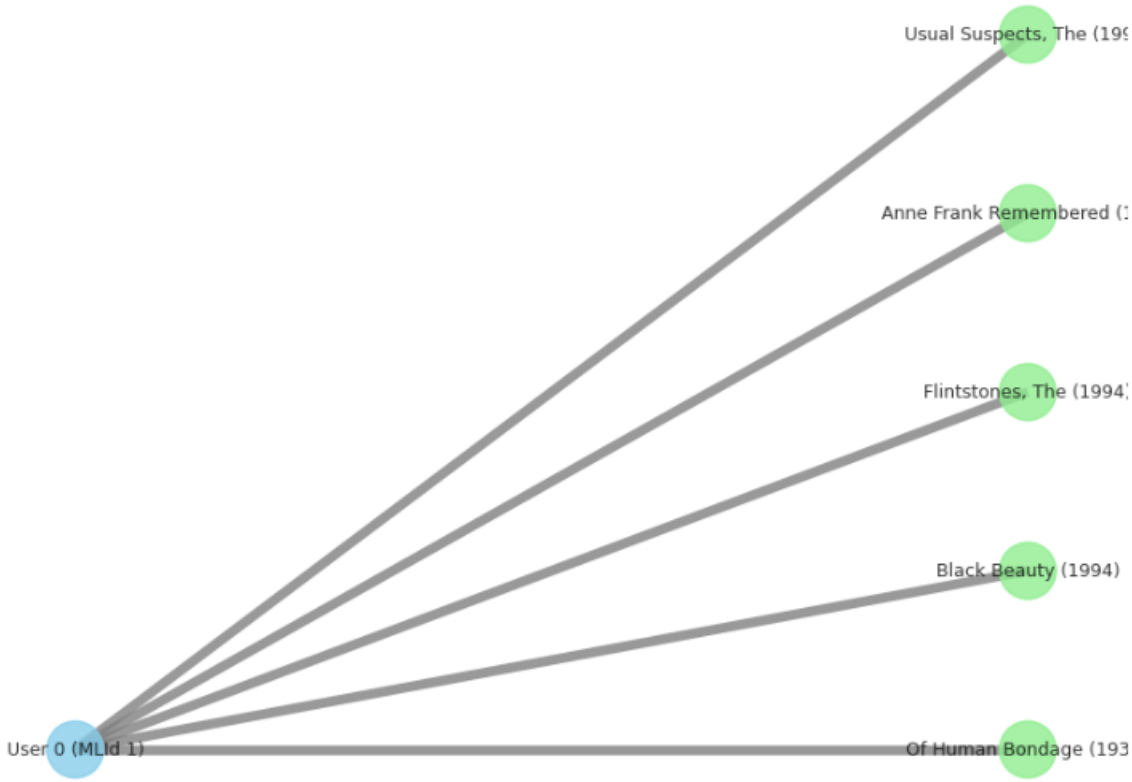

Figure 1. The top-5 movie recommendations for User 0

This graph visualizes the top-5 movie recommendations for User 0 using the trained model with rich features. Each edge represents a strong similarity between the user and the recommended movie, with edge thickness indicating predicted rating strength. This visualization confirms that the model can learn meaningful relationships and provide interpretable, personalized suggestions.

6. Enhanced Graph with Tag Embeddings (Subgraph Training) and Visualizing with UMAP

To further improve recommendation quality, we trained the model on a subgraph consisting of 10,000 users and their rated movies. This approach allowed us to experiment more efficiently while introducing even richer feature representations.

Key Enhancements:

- We incorporated **tag-based features** from the genome-scores.csv file. These tags describe detailed attributes of movies (e.g., "gritty," "space travel") and were transformed using **Truncated SVD** to capture their latent structure in a compact embedding.
- These tag embeddings were combined with previously used movie features—**year**, **popularity**, **average rating**, and **genres**—as well as user rating behavior statistics.
- By training on a smaller subgraph, we significantly reduced computational cost while retaining a representative portion of the full dataset.

Expected Improvements:

- Tag embeddings provide more nuanced content signals than genre labels alone.
- Combining various feature types allows the GNN to learn more informative embeddings, leading to better rating predictions.

Results:

- **Final training loss:** 0.0599
- **Subgraph Test MSE:** 0.0463
- **NDCG@K:** 0.9807

These results confirm that enriching the graph with **semantic tag-based features** leads to better performance, both in prediction accuracy and ranking quality.

```
[17] import umap
     import matplotlib.pyplot as plt

     model_sub.eval()
     with torch.no_grad():
         emb_sub = model_sub(data_sub.x.to(device), data_sub.edge_index.to(device)).cpu().numpy()

     movie_embs = emb_sub[num_sub_users:]

     reducer = umap.UMAP(n_components=2, random_state=42)
     umap_out = reducer.fit_transform(movie_embs)

     genres_list = [g.split('|')[0] for g in meta_sub['genres']]
     unique_genres = list(set(genres_list))
     genre_to_idx = {g:i for i,g in enumerate(unique_genres)}
     colors = [genre_to_idx[g] for g in genres_list]

     plt.figure(figsize=(8,6))
     plt.scatter(umap_out[:,0], umap_out[:,1], c=colors, cmap='tab20', s=5, alpha=0.7)
     plt.title("UMAP of Movie Embeddings (colored by dominant genre)")
     plt.axis('off')
     plt.show()
```

Figure 2. UMAP of Movie Embeddings color by dominant genre

The UMAP plot shows strong clustering based on genre:

- **Clear Genre Clusters**: Distinct, solid-color regions indicate that movies from the same genre are grouped close together in the embedding space.
- **Genre Overlaps and Bridges**: Areas where colors blend show multi-genre films (e.g., Action-Comedy) that connect different content types.
- **Outliers and Niche Groups**: Some isolated clusters represent rare or niche genres like experimental or documentary films.
- **Feature Fusion Validation**: Since embeddings combine year, rating stats, genres, and tag features, the quality of this clustering suggests the GNN effectively fused these into meaningful vectors.

This plot provides both **interpretability** and **validation** of the model's learned structure.

To evaluate the effectiveness of the enhanced GNN model, we trained the system for **200 epochs** using a subgraph of 10,000 users. The model showed a **steady and stable decrease in training loss** throughout the training process.
The **training loss curve** (shown below) demonstrates smooth convergence, indicating that the model learned effectively without overfitting or sudden fluctuations.
Final performance metrics:

- **Final Training Loss:** 0.0409
- **Subgraph Test MSE:** 0.0406
- **Subgraph Test NDCG@K:** 0.9857

These results reflect strong predictive accuracy and effective ranking performance. Notably, the inclusion of **time-based user features** (e.g., hour-of-day and day-of-week activity patterns) led

to small but consistent improvements in performance. These features helped the model better capture **nuanced viewing behaviors** that go beyond simple rating history.
Overall, the model demonstrates both **quantitative strength** and **qualitative interpretability**, making it a solid foundation for a real-world recommendation engine.

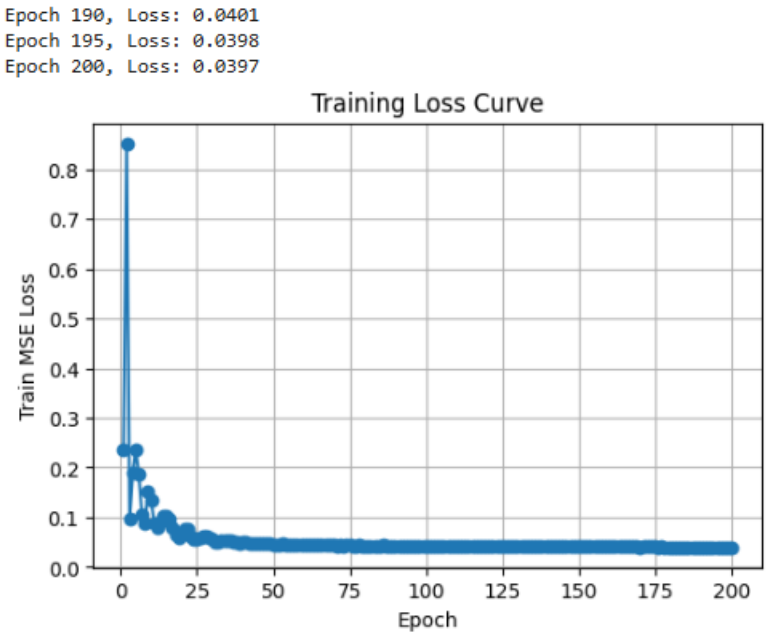

Figure 3. Training results for 200 epoch

7. Recommendation Graph Demo

To demonstrate the model's ability to generate personalized recommendations, we visualize the top-5 recommended movies for a selection of users using the trained GraphSAGE model.

In the graph:

- **Orange nodes** represent users.
- **Blue nodes** represent recommended movies.
- **Edges** show the predicted similarity score between each user and movie, derived from the **cosine similarity** of their learned embeddings. A higher score indicates a stronger match.

Steps used to generate this graph:

1. Extract embeddings for users and movies from the trained GNN.
2. Compute similarity between each sampled user and all movies.
3. Select the **top-5** movies with the highest similarity scores per user.
4. Visualize the user-movie relationships using **NetworkX**, with edge labels showing predicted scores.

This visual makes it easy to interpret how the GNN model connects users to different movies based on patterns it has learned from the graph structure and node features. The spread of movie

nodes demonstrates that the system does not recommend the same content to everyone—instead, it generates **diverse and tailored suggestions** for each user.

```
nx.draw_networkx_nodes(G, pos, nodelist=[n for n in G if n.startswith('U')],
                       node_color='orange', node_size=900, label='Users')
nx.draw_networkx_nodes(G, pos, nodelist=[n for n in G if not n.startswith('U')],
                       node_color='skyblue', node_size=900, label='Movies')

nx.draw_networkx_edges(G, pos, width=1.5, alpha=0.6)
nx.draw_networkx_labels(G, pos, font_size=8)
edge_labels = {(u, m): f"{s:.2f}" for u, m, s in edges}
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_size=7)
plt.axis('off')
plt.tight_layout()
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
   movieId                               title  \
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
2        3             Grumpier Old Men (1995)
3        4            Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                   Adventure|Children|Fantasy
2                               Comedy|Romance
3                         Comedy|Drama|Romance
4                                       Comedy
```

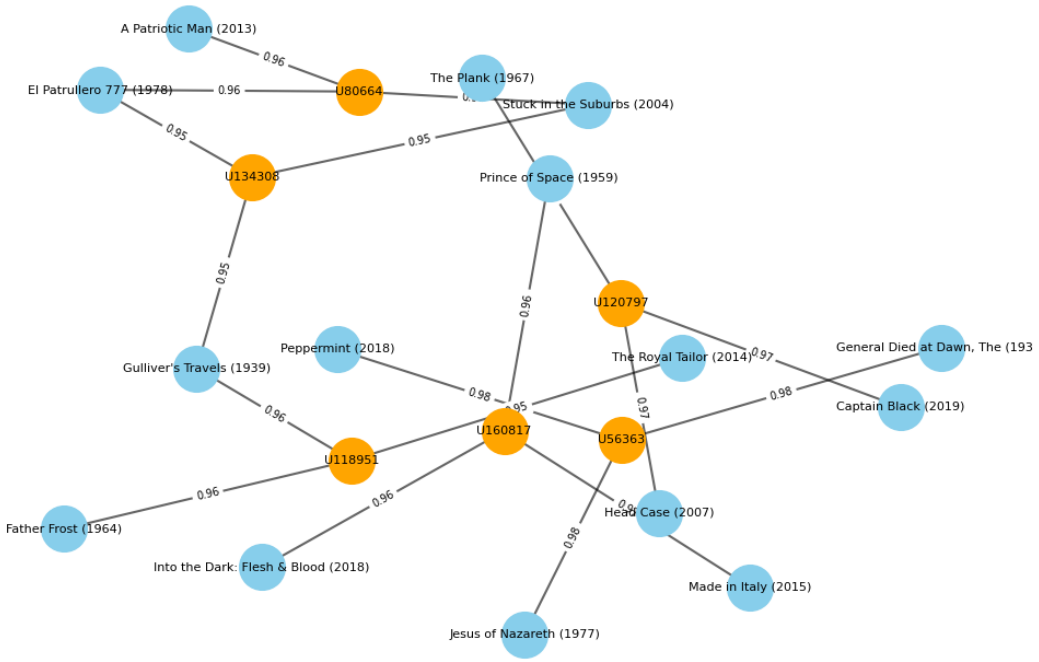

Figure 4. Complex Network of Recommended System

To further evaluate the model's output, we generated a recommendation table showing the **top-5 movies** for a random sample of users. Each row corresponds to a user, and each column lists their most highly recommended movies in ranked order.

In the sample, two titles—**"Shopping (1994)"** and **"Run-ning-maen (2013)"**—appeared as the top recommendation for more than one user. However, most recommendations were **unique**, highlighting the model's ability to personalize suggestions.
These results suggest two key insights:

- The model has learned to identify **general appeal movies** that are popular or broadly relatable. These likely sit near the **center of the embedding space**, where many users' preferences intersect.
- At the same time, the model also makes **user-specific recommendations**, taking into account **temporal activity patterns** and **rating behavior**.

This diversity is made possible by the integration of rich features:

- **User features:** 34-dimensional vectors combining rating stats and time-of-day behavior
- **Movie features:** 73-dimensional vectors including genre, release year, popularity, and tag-based semantic attributes

Together, these features enable the GNN to align **individual user profiles** with **movie content**, producing recommendations that are both **accurate and varied** across users. This balance of generalization and personalization is essential for a real-world recommender system.

To make the recommendation system more accessible and user-friendly, we developed an **interactive tool** within the notebook. This tool allows users to explore top movie recommendations in real-time by adjusting several filters.
Demo Features:

- **User selection:** Choose any user from the dataset.
- **Year range slider:** Filter movies by release year.
- **Genre filter:** Select one or more genres to narrow down suggestions.
- **Minimum similarity score:** Only include movies above a chosen similarity threshold.
- **Top-K selector:** Control how many top recommendations are shown.

When the "Get Recommendations" button is clicked, the system uses the trained **GraphSAGE model** to:

1. Generate movie embeddings for all items.
2. Compute cosine similarity between the selected user and each movie.
3. Filter and sort movies based on the selected criteria.

The tool then displays two ranked lists:

- **Top Recommended Movies** — personalized suggestions based on the user's embedding and selected filters.
- **Least Relevant Movies** — items with the lowest similarity, offering insight into what the model considers poor matches.

This interactive demo highlights the model's ability to deliver **customized, diverse, and content-aware recommendations**. It also allows users to better understand how different filters affect the outcome, making the recommendation process more transparent.

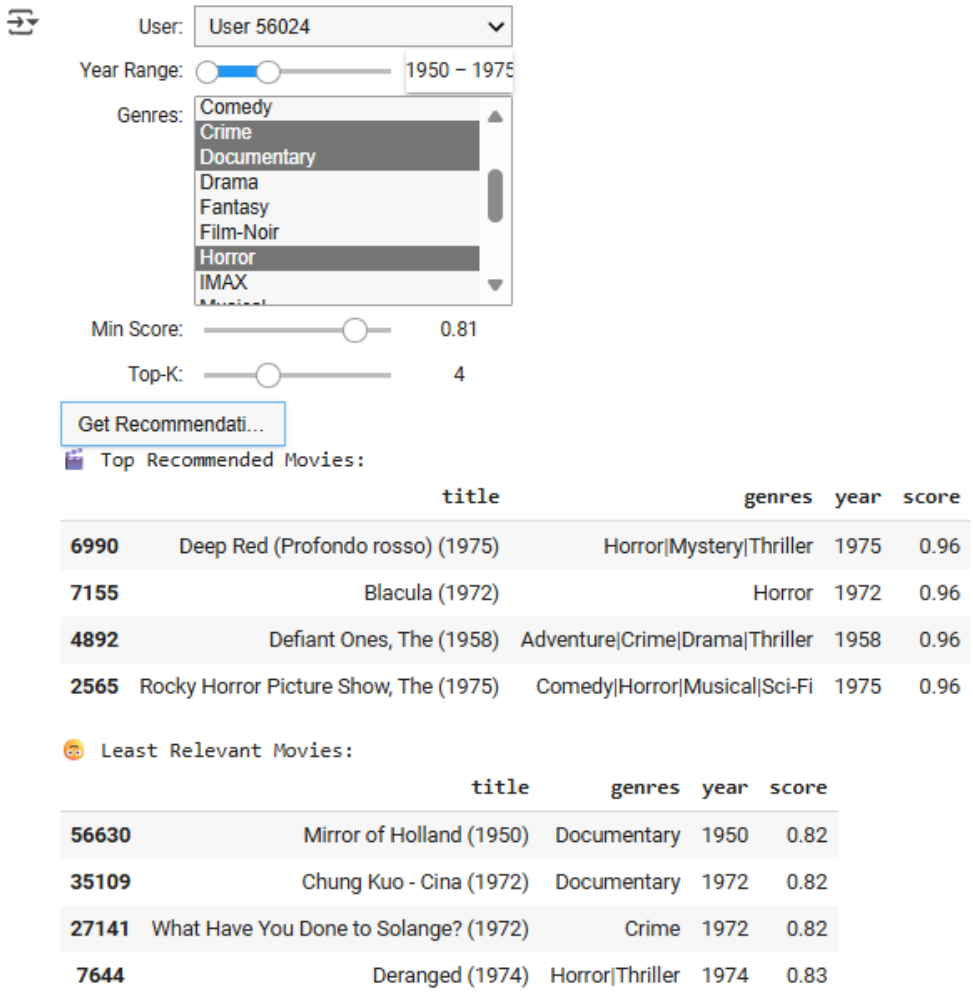

Figure 5. Interactive demo built in Jupyter notebook

8. Conclusion

This project showed how Graph Neural Networks (GNNs) can effectively power a personalized and explainable movie recommendation system.

We modeled the MovieLens 25M dataset as a user-movie interaction graph, where users and movies are nodes and ratings are edges. Rich node features were used—including statistical behavior, temporal activity, content metadata (genres, year), and semantic info via genome tags (SVD).

A 2-layer GraphSAGE model learned meaningful user and movie embeddings through message passing and neighborhood aggregation.

The system achieved strong performance:

- **Quantitative**: Low MSE (0.0406), high NDCG@K (0.9857)
- **Qualitative**: Clear genre-based clusters and personalized recommendations

We also developed an interactive tool that:

- Lets users choose a user ID, genre(s), year range, and minimum score
- Outputs the top-K recommended movies dynamically

In addition, we visualized a complex user-movie graph showing how users connect to their top predicted movies, highlighting the model's ability to capture diverse and structured preferences. Overall, this project confirms that GNNs offer a powerful and flexible approach to recommendation tasks—combining accuracy, personalization, and interpretability.

8. References

GNN Theory and Architecture

1. Bronstein, M.M., Bruna, J., Cohen, T., & Velickovic, P. (2021). *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. arXiv:2104.13478
2. Hamilton, W. (2020). *Graph Representation Learning*. Springer. https://www.cs.mcgill.ca/~wlh/grl_book/
3. Battaglia, P., et al. (2018). *Relational Inductive Biases, Deep Learning, and Graph Networks*. arXiv:1806.01261
4. Grover, A., & Leskovec, J. (2016). *node2vec: Scalable Feature Learning for Networks*. arXiv:1607.00653
5. Ward, I.R., et al. (2021). *A Practical Tutorial on Graph Neural Networks*. arXiv:2010.05234

Tools and Libraries

6. Fey, M., & Lenssen, J.E. (2019). *Fast Graph Representation Learning with PyTorch Geometric*. arXiv:1903.02428
   PyG Library: https://pytorch-geometric.readthedocs.io
7. NetworkX Documentation. https://networkx.org/documentation/stable/
8. MovieLens 25M Dataset. GroupLens Research. https://grouplens.org/datasets/movielens/25m/

Course Materials

9. Djordjević, B. (2025). *Lecture 07: Review of Graph Neural Networks*, CSCI E-104 Advanced Deep Learning, Harvard Extension School.
10. Stanford CS224W: *Machine Learning with Graphs* by Jure Leskovec. http://cs224w.stanford.edu

Datasets and Evaluation

11. Harper, F.M., & Konstan, J.A. (2015). *The MovieLens Datasets: History and Context*. ACM Transactions on Interactive Intelligent Systems (TiiS). DOI:10.1145/2827872
12. sklearn.metrics.ndcg_score. Scikit-learn documentation. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ndcg_score.html

Appendix

YouTube Demo Links:

- 2-Minute Summary: https://youtu.be/dPr9RzEOj6g
- 15-Minute Presentation: https://youtu.be/siK5MfFOr-w