Movie Recommendation System

Prepared for,

1. Dr. Jillian Tang
2. Dr. Hamid R Karimian

Prepared By,

1. Om Sai Krishna Madhav Lella (lellaom@msu.edu)
2. Raghavi Ravi (ravirag2@msu.edu)
3. Subramaniya Siva T S (tsubrama@msu.edu)
4. Aarthi Padmanabhan (padman21@msu.edu)
5. Steve Mitchell John Rajakumar (johnstev@msu.edu)

Contributions,

S. No.	Functionality	Contributed By
1.	Streamlit Web App	1. Om Sai Krishna Madhav Lella 2. Subramaniya Siva T S
2.	Basic EDA and Visualizations	1. Raghavi Ravi 2. Aarthi Padmanabhan 3. Steve Mitchell John Rajakumar
3.	Content Based Methodology	1. Aarthi Padmanabhan 2. Subramaniya Siva T S
4.	Collaborative Based Methodology	1. Raghavi Ravi 2. Steve Mitchell John Rajakumar
5.	Hybrid Methodology	1. Om Sai Krishna Madhav Lella

Introduction:
The advent of streaming platforms has led to an overwhelming volume of content, making it challenging for users to discover movies aligned with their preferences. This project aims to tackle this issue by developing a movie recommendation system using several different approaches. The dataset utilized in this project is the MovieLens dataset.

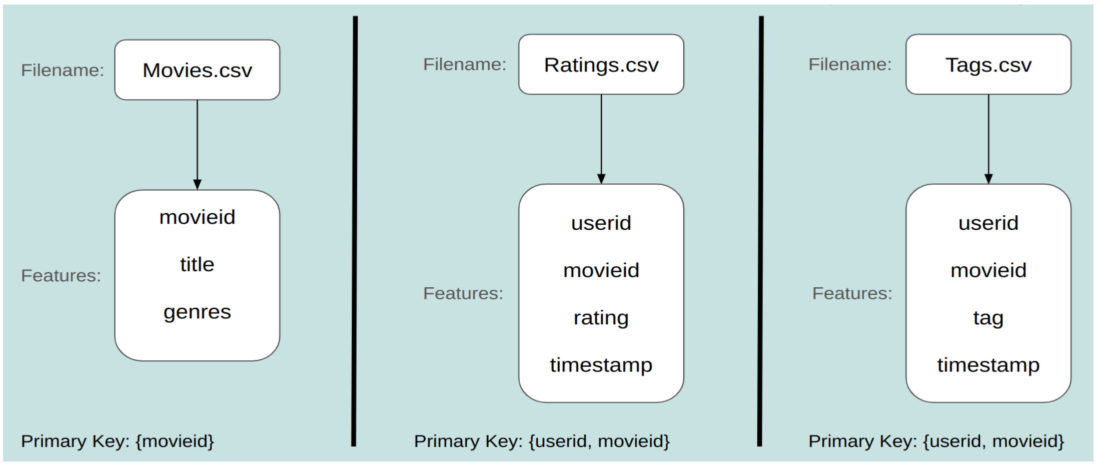
Problem Statement:
The primary goal of this project is to implement a recommendation system that accurately predicts user preferences for movies based on historical data. By employing content based methodology (NLP Based Model), collaborative based methodologies (User Based, Item Based, Matrix Factorization, Softmax Deep Neural Network), hybrid (content and collaborative based) methodologies, the system aims to enhance the accuracy and effectiveness of movie recommendations.

Data Set and Feature Selection:
Three datasets are used in this project: ratings.csv, movies.csv and tags.csv. The features of each dataset are described below:

ratings.csv:
Features = {userid, movieid, rating, timestamp}
Primary Key = {userid, movieid}

movies.csv:
Features = {movieid, title, genres}
Primary Key = {movieid}

tags.csv:
Features = {userid, movieid, tag, timestamp}
Primary Key = {userid, movieid}



The tags.csv file was used only to develop a content-based NLP recommendation system. The ratings.csv and movies.csv files were used to develop all other recommendation systems.

Methods Employed:

Content-Based filtering:

Content-based filtering is another approach used in recommendation systems, and it relies on the characteristics or features of items and users to make recommendations. Unlike collaborative filtering, content-based filtering doesn't require information about the preferences or behaviours of other users. Instead, it focuses on the properties of items and the explicit profile of the user.

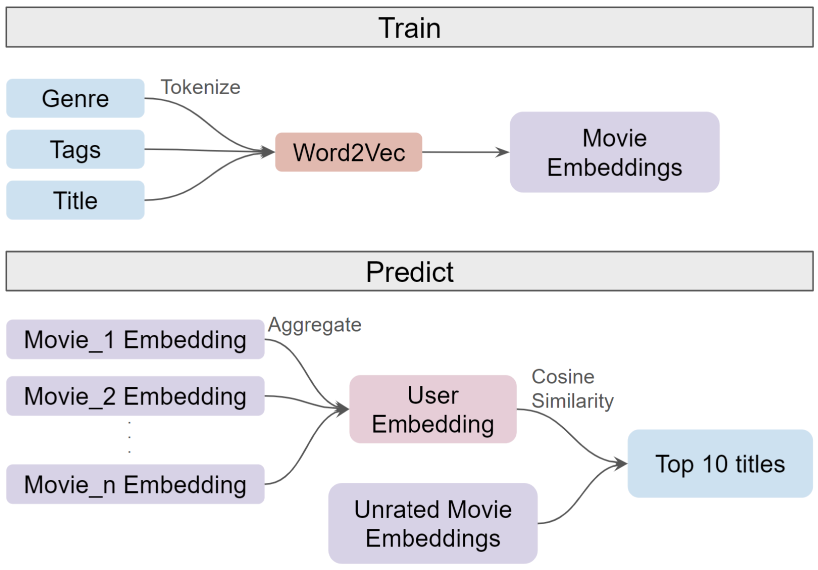
Content-Based Prediction Using NLP:

Word2Vec Training: Tokenize movie tags and train a Word2Vec model with specified parameters, such as vector size and window size. This process generates embeddings that capture semantic relationships within the textual content.

Movie Embeddings: Compute unique embeddings for each movie using the trained Word2Vec model. These embeddings represent the semantic content of the movies.

User Embeddings: Determine user embeddings by aggregating the embeddings of rated movies, reflecting individual preferences in the Word2Vec space.

Cosine Similarity for Recommendations: Utilize cosine similarity to measure the similarity between user and movie embeddings. Recommend movies with the highest cosine similarity scores, aligning with user preferences.



Collaborative Filtering:

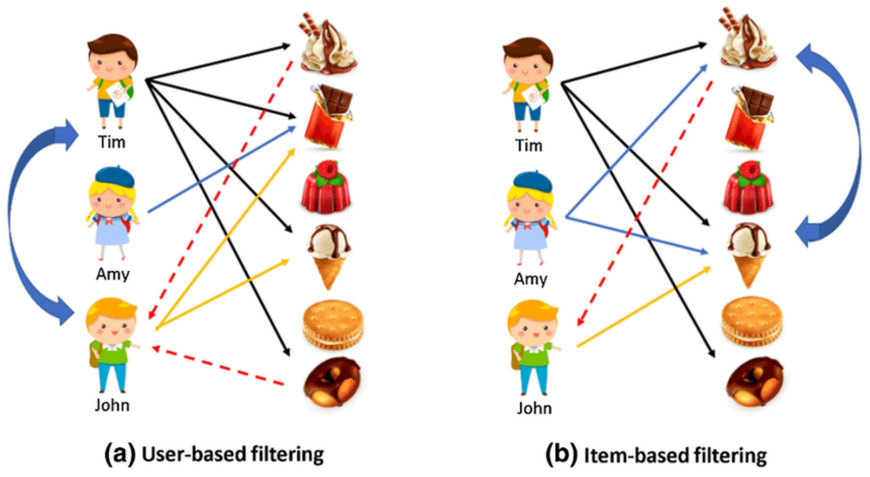
Collaborative filtering is a type of recommendation system that leverages the preferences, behaviors, or opinions of users to make predictions about their interests. The basic idea is to recommend items or content to a user based on the preferences of other users who are similar to them.

User based:

User-based collaborative filtering is a recommendation system technique that relies on the preferences and behaviors of similar users.

Item based:

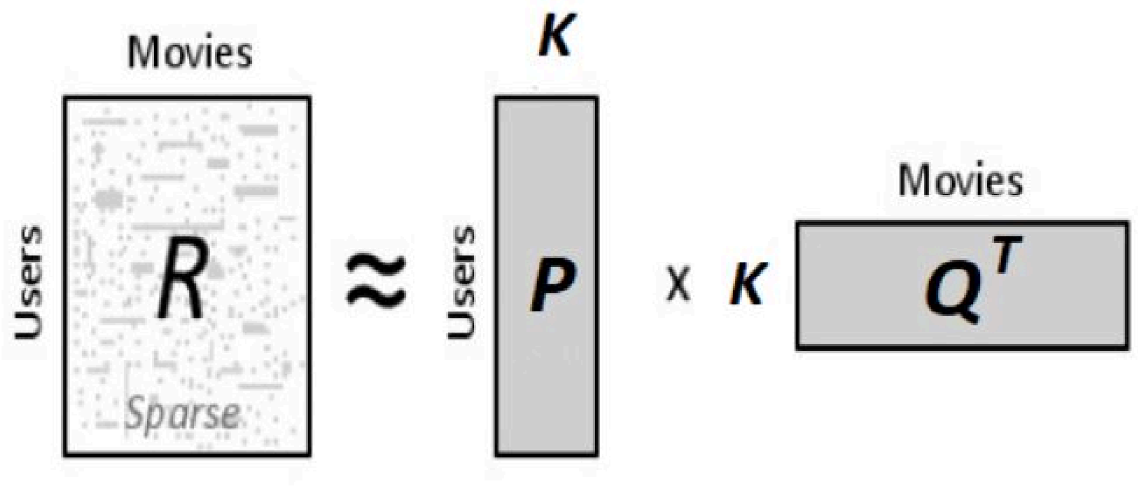
User-based collaborative filtering is a recommendation system technique where the focus is on the similarity between items rather than users.



Matrix Factorization:

Matrix factorization stands out as a widely used technique for collaborative filtering. The fundamental concept involves breaking down the user-item matrix into two matrices with lower ranks: one depicting user preferences and the other reflecting movie characteristics. The reconstruction of the original user-item matrix is accomplished by computing the dot product of these two matrices.

The user-item matrix features rows corresponding to users and columns corresponding to movies. The matrix entries denote the ratings provided by users for the respective movies. Typically, this matrix is sparse, given that each user tends to rate only a small portion of the available movies.



The goal of matrix factorization is to decompose the user-item matrix R into two lower-rank matrices, P and Q, with their product closely approximating R. Matrix P signifies user's preferences, while matrix Q signifies items' characteristics. Each row in matrix P is a vector representing a user's preferences, and each row in matrix Q is a vector representing an item's characteristics. This approach allows for a meaningful representation of user-item interactions in terms of preferences and characteristics.

This method aims to discover matrices P and Q that minimize the discrepancy between the approximated matrix and the real matrix. Typically, this disparity is gauged through the root mean squared error (RMSE) between the predicted ratings and the actual ratings.

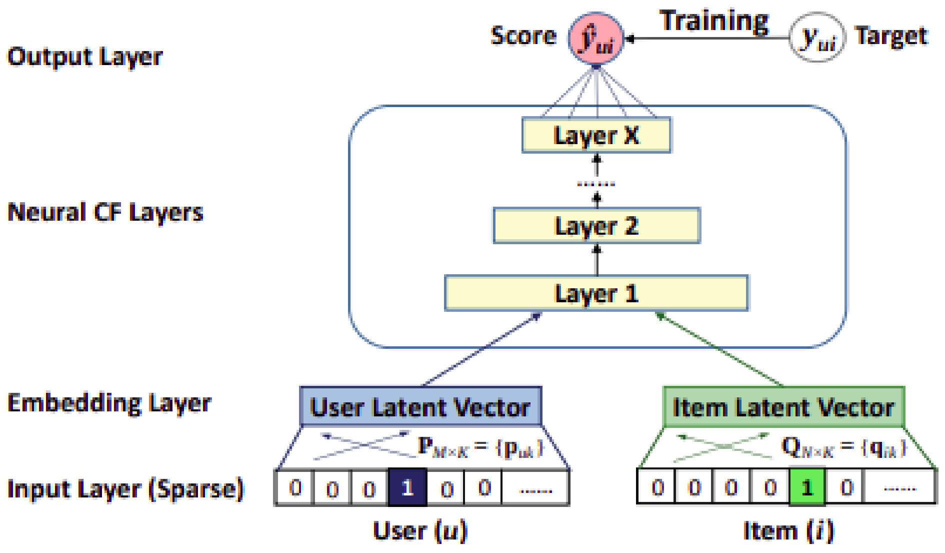
In this project, the SVD (Singular Value Decomposition) technique is incorporated for matrix factorization using the Surprise library in Python. SVD is a linear algebra technique used for matrix factorization. In the context of collaborative filtering, it helps decompose the user-item matrix into three matrices (P, Σ, Q^T), where P represents user preferences, Σ is a diagonal matrix of singular values, and Q^T represents movie characteristics.

Deep Neural Network Model Based:

Deep Neural Network (DNN) models provide a promising solution for addressing challenges like the cold start problem and improving the relevance of recommendations. The flexibility of the input layer in DNNs allows for the incorporation of user and movie features, enabling a more nuanced understanding of user preferences and enhancing the relevance of recommendations.

In this project, Deep Neural Networks are employed for movie recommendations. Users and movies undergo one-hot encoding and are then input into the Deep Neural Network as distinct inputs, with the ratings being generated as the output.

The construction of the Deep Neural Network model involved extracting the latent features of users and movies using Embedding layers. Subsequently, Dense layers with dropout mechanisms were stacked, followed by the addition of a final Dense layer comprising 9 neurons (representing each rating from 1 to 5) and incorporating a Softmax activation function. For the optimization algorithm, it was decided to use SGD and Sparse Categorical Cross Entropy for the loss function.



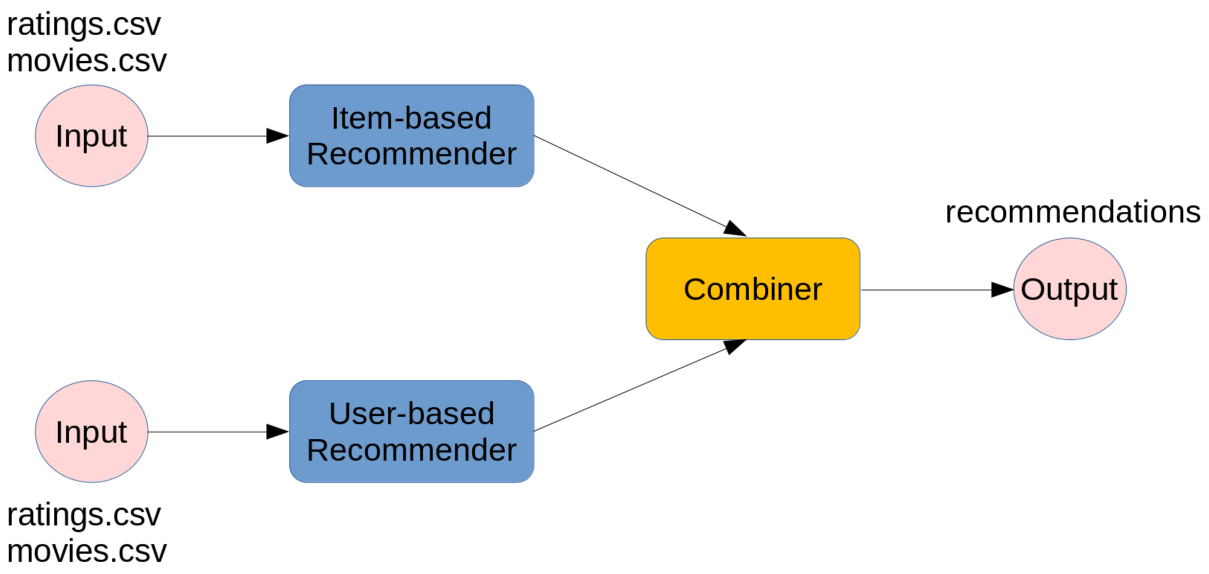
The workflow involves users inputting their ID, extracting unseen movies, and utilizing a DNN model that takes user and movie IDs to predict ratings. The model's predictions are normalized and used to identify movies likely to interest the user, without converting ratings back to the original scale. The DNN model is thus a valuable tool for predicting user preferences for unseen movies.

Hybrid Based:

Hybrid recommendation is developed as an ensemble model of item-based and user-based recommendation systems. To develop these recommendation systems, a data matrix with rows as user, columns as movies and values as ratings is constructed. In this model, both item-based and user-based similarities are used for movie recommendations. The approaches used for identifying item-based similarities and user-based similarities are described below.

For **item-based** similarity, initially, the user's most recent favourite movie is identified. Later, using the data matrix, movies which are strongly correlated with this favourite movie are identified as potential recommendations to the user.

For **user-based** similarity, initially, a list of movies rated by the user is found. Next, all the users who rated at least 60% (hyperparameter) of the previous list of movies and who have a correlation of at least 0.75 are identified. After that, this list of similar users is used to estimate the weighted average rating of the movies. Finally, top movies with the highest weighted average rating are identified as the potential recommendations to the user.

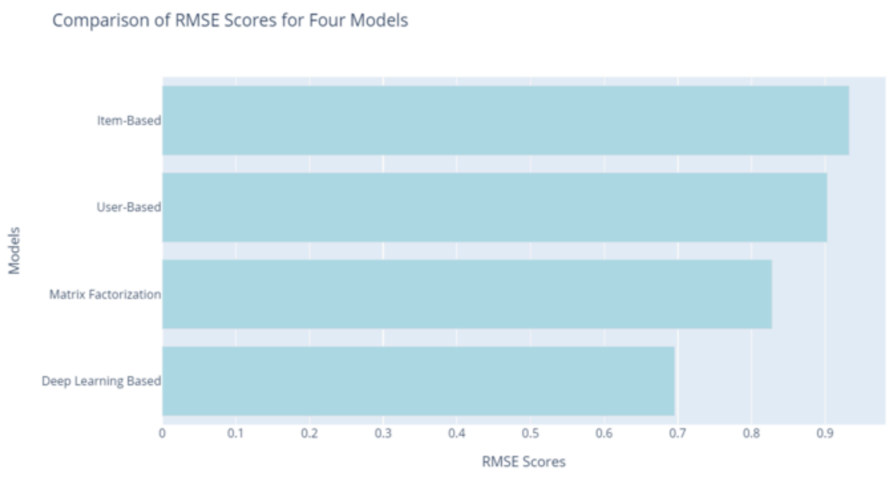


Results, Observations, and Discussion:

Performance Evaluation

The movie recommendation system was evaluated using Root Mean Squared Error (RMSE) as a metric to measure the disparity between predicted ratings and actual user ratings for the test set. The RMSE scores obtained for different models were as follows:

- Item-Based: RMSE = 0.9329
- User-Based: RMSE = 0.9031
- Matrix Factorization: RMSE = 0.8280
- Deep Learning Model: RMSE = 0.6959



From these results, it's evident that both the Matrix Factorization and Deep Learning models outperformed the traditional Item-Based and User-Based approaches, showcasing lower RMSE values. These findings signify the superior predictive capabilities of the Matrix Factorization and Deep Learning methods in capturing user preferences and providing accurate recommendations.

Model Insights

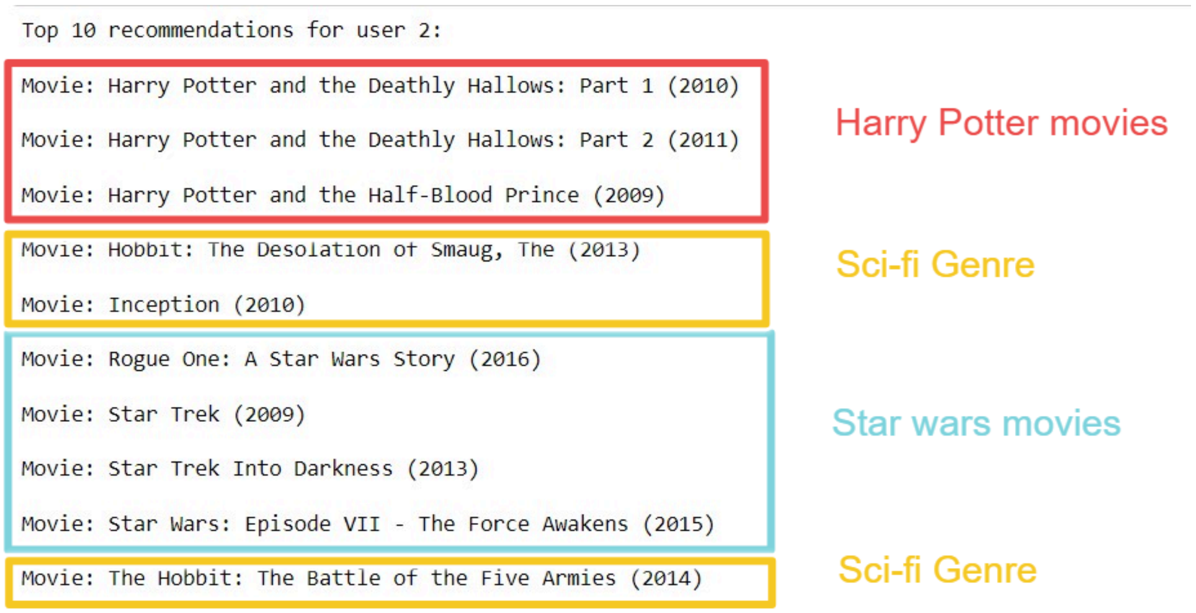
The Matrix Factorization model, in particular, demonstrated remarkable proficiency in providing personalized movie recommendations. By leveraging latent factors derived from the user-item interactions, the model effectively identified patterns and relationships within the data. Notably, the model outputted recommendations aligning with the user's previous movie preferences.

For instance, the model successfully recommended Harry Potter movies based on the user's history of watching previous releases from the franchise. Similarly, it suggested Star Wars and Star Trek films as the user had engaged with earlier instalments. Notably, the recommendation of "The Hobbit" movie, despite the user not having previously watched it, illustrates the model's ability to comprehend broader preferences. Leveraging the user's fondness for sci-fi genres, the system intelligently recommended popular and acclaimed sci-fi movies like "Inception" and "The Hobbit."

Graphical Representation

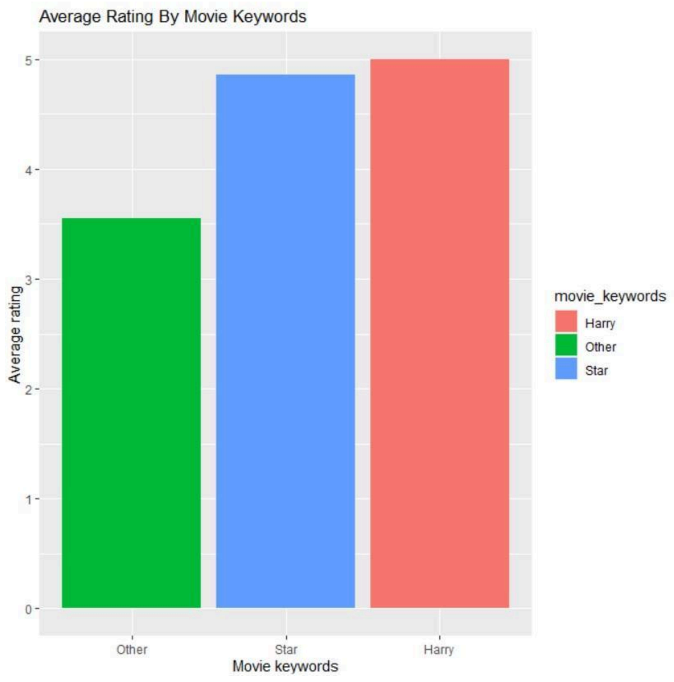
Below are the graphical representations depicting the user's movie ratings for specific franchises like Harry Potter, Star Wars, and Star Trek, along with genre-wise ratings, illustrating the user's inclination towards particular movie series and genres.

The recommendations are:

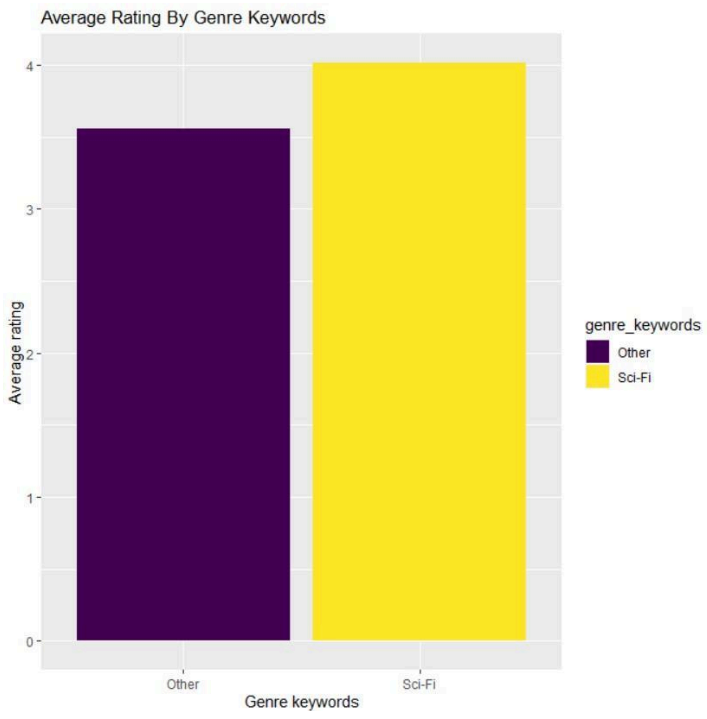


```
Top 10 recommendations for user 2:
Movie: Harry Potter and the Deathly Hallows: Part 1 (2010)
Movie: Harry Potter and the Deathly Hallows: Part 2 (2011)
Movie: Harry Potter and the Half-Blood Prince (2009)
Movie: Hobbit: The Desolation of Smaug, The (2013)
Movie: Inception (2010)
Movie: Rogue One: A Star Wars Story (2016)
Movie: Star Trek (2009)
Movie: Star Trek Into Darkness (2013)
Movie: Star Wars: Episode VII - The Force Awakens (2015)
Movie: The Hobbit: The Battle of the Five Armies (2014)
```

The inclination towards Harry Potter/ Star wars/ Star trek movies is shown below:



The inclination towards the Sci-fi genre is shown below:



System's Intelligent Recommendations

The recommendations made by the Matrix Factorization model weren't arbitrary but rather based on the user's specific movie tastes and preferences. It didn't merely propose random sci-fi movies but instead offered popular ones that aligned closely with the user's interests, demonstrating a thoughtful and nuanced understanding of the user's preferences beyond their explicit viewing history.

Conclusion

The algorithm's commendable performance on the test set indicates its proficiency in suggesting movies that align closely with human recommendations when trained on a comprehensive dataset. The successful incorporation of factors such as similar movies, genres, and popularity contributes to the system's effectiveness in generating relevant suggestions.

In summary, the Matrix Factorization model's ability to deliver precise and tailored movie recommendations based on nuanced user preferences showcases its potential for creating personalized and engaging recommendation systems. Its performance signifies a significant leap forward in enhancing user experience and satisfaction in content recommendation platforms.

Lessons Learned:

- **Algorithm Selection and Trade-offs:** Understanding the trade-offs between different recommendation algorithms, such as matrix factorization and deep learning-based methods, including computational complexity, scalability, and interpretability. Recognizing the strengths and limitations of each approach aids in making informed choices for specific use cases.
- **Challenges in User Engagement Prediction:** Acknowledging the challenges in predicting user engagement accurately. Factors like evolving user preferences, seasonal trends, and context-specific behaviours pose challenges in building models that consistently deliver high-quality recommendations. This understanding emphasizes the need for continuous model refinement and adaptation.
- **Importance of Experimentation and Evaluation:** Emphasizing the significance of robust experimentation and evaluation methodologies. Employing various evaluation metrics and experimentation setups helps in comprehensively assessing model performance. Additionally, the iterative nature of experimentation facilitates continual improvement and fine-tuning of recommendation systems.
- **Real-world Deployment Considerations:** Recognizing the complexities involved in deploying recommendation systems in real-world scenarios. Considering aspects such as computational resources, model deployment frameworks, and user privacy concerns early in the development phase is crucial for seamless integration and adoption.

Future Improvements:

- **Dynamic and Personalized Models:** Move towards dynamic and personalized recommendation models that adapt in real-time to evolving user preferences. Incorporating reinforcement learning or continual learning techniques might enable the system to continually improve its recommendations based on user feedback.
- **Scalability and Efficiency:** Focus on optimizing computational efficiency and scalability, especially for large-scale datasets, by exploring parallel computing techniques or model compression strategies. This would facilitate the deployment of the recommendation system in real-world, resource-constrained environments.
- **Incorporating temporal factors**: Considering how movie trends and audience preferences evolve over time. By accounting for temporal aspects such as changing graphics quality or narrative styles across decades, the system could further refine its recommendations, ensuring they remain relevant and appealing to users amidst evolving cinematic landscapes.

Application Link:

https://movie-recommendation-system-cse482.streamlit.app/