Suggestaria

Team *- Ayush Srivastava(220272)*
Mohd Adil(220658)
Abhishek Kumar(220045)

Dataset Used:

The dataset consists of the following four csv files downloaded from kaggle:

- Links: to provide a mapping or linkage between a different platform Id and TmdbId('Id' in the csv file).
- Tmdb_5000_movies & Tmdb_5000_credits:include various attributes such as budget, genres, keywords, cast, crew, and more.
- Ratings:It provides data of userId,rating provided by him/her,movieId

The key steps in data preprocessing include:

- Checking for missing values using pd.isnull(movies).sum().
- Merging the movies and credits datasets on the "title" column to create a unified dataset (movies_new).
- Converting data types where necessary to facilitate further analysis

Further we used only those movies which are rated by 250+ users and users who have rated more than 100 movies for better accuracy.

Algorithms Used:

This report presents an implementation of a movie recommendation system using **movie-based collaborative filtering**. The system predicts movie ratings based on similarity scores and evaluates model performance using the **Mean Absolute Error (MAE)** metric. The core functions include:

- recommend(movie_name): Recommends similar movies based on a given movie.

- predict_rating(user, movie_name): Predicts a user's rating for a specific movie.

- evaluate_model_with_sampling(sample_size): Evaluates the model by calculating the MAE using a random sample of user-movie pairs.

Movie-Based Collaborative Filtering

Movie-based collaborative filtering is a technique that recommends movies based on similarities between them. The similarity between two movies is typically calculated using metrics such as **cosine similarity** or **Pearson correlation coefficient**.

recommend(movie_name) Function

This function finds and returns the top 5 movies similar to a given movie using precomputed similarity scores.

Steps:

1. Locate the index of the given movie in the dataset.
2. Retrieve similarity scores for the given movie from similarity_score.
3. Sort the movies based on their similarity scores in descending order.
4. Extract and print the top 5 most similar movies (excluding the given movie itself).

Predicting Movie Ratings

The function predict_rating(user, movie_name) estimates how much a user would rate a movie based on ratings of similar movies.

predict_rating(user, movie_name) Function

Steps:

1. Retrieve the average rating of the given movie (neu_i).
2. Identify the index of the movie in the dataset.
3. Find the top k(=5 here) similar movies using similarity_score.
4. Compute a weighted sum of ratings from the similar movies.
5. Normalize the prediction by dividing by the total similarity score.
6. If there are no contributing ratings (sum2 == 0), return the average rating of the movie.

Model Evaluation:

The function evaluate_model_with_sampling(sample_size) calculates the **Mean Absolute Error (MAE)** over a sampled dataset.

evaluate_model_with_sampling(sample_size) Function

Steps:

1. Collect all non-zero (movie, user) rating pairs from the dataset.
2. Randomly select a subset of sample_size pairs.
3. Compute the actual and predicted ratings for these sampled pairs using predict_rating().
4. Calculate MAE, which measures the average absolute difference between actual and predicted ratings.

Model Performance and Interpretation:

After running the evaluation function on a **random sample of 10,000** user-movie pairs, the obtained **Mean Absolute Error (MAE) Score is 0.8926**.

Understanding the MAE Score:

- **MAE = 0.8926** means that, on average, the predicted ratings differ from the actual ratings by **0.89** points.
- The rating scale is **1 to 5**, an MAE of **0.89** is moderate, as it constitutes an **18% error margin**.

Possible Improvements:

- **Increase Similarity Accuracy**: Try different similarity measures like Pearson correlation or matrix factorization (SVD, ALS).
- **Tuning k in Collaborative Filtering**: Experiment with different values of k (number of similar movies considered for prediction).
- **Hybrid Approaches**: Combine user-based and item-based collaborative filtering to improve predictions.

- **Feature Engineering**: Incorporate additional metadata like movie genres, release years, and user demographics.

- **Regularization**: Use weight penalties to reduce overfitting to certain user behaviors.

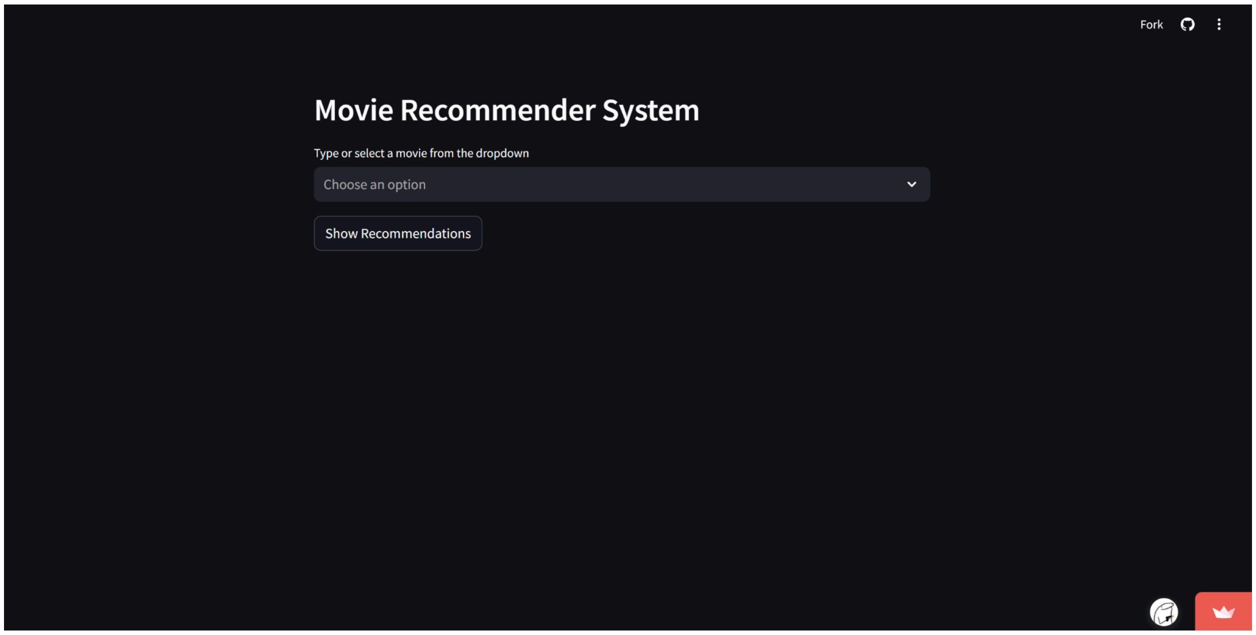
Challenges Faced:

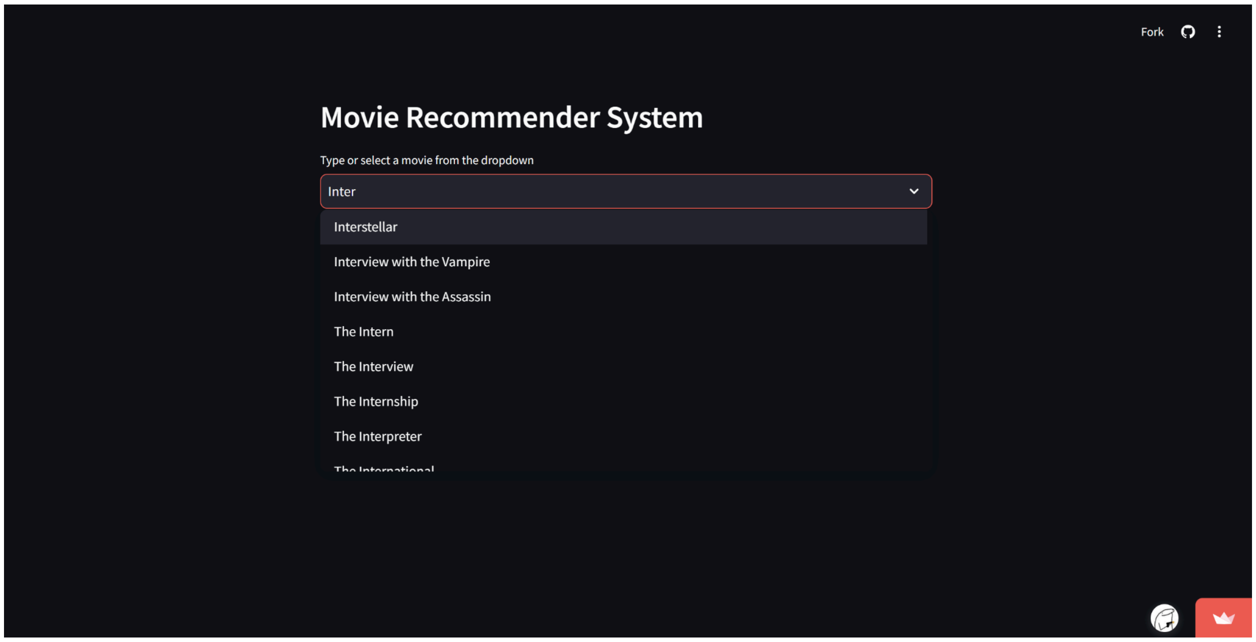
- We faced issues deploying on **Heroku** due to credit card verification failures, so we opted for **Git deployment** instead.
- For model evaluation, since we hadn't predicted ratings for any user-movie pair, we used the **KNN algorithm** to predict ratings based on **5 nearest neighbors** and calculated **MAE** on actual vs. predicted ratings.
- Additionally, the **large movie-user pivot matrix** caused slow evaluation. To address this, we implemented **random sampling (10,000 pairs out of 4000×777 total ratings)** for faster evaluation.

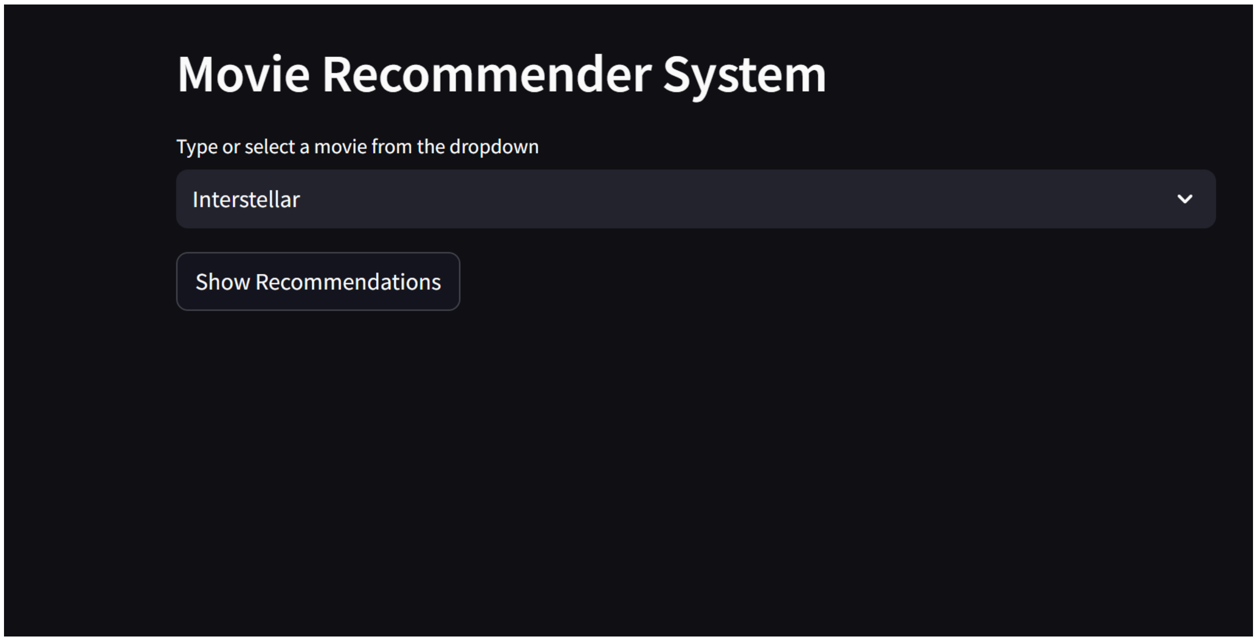
Conclusion:

This report details the implementation and evaluation of a **movie-based collaborative filtering recommendation system**. The system uses similarity scores to recommend movies and predict ratings. The evaluation using **MAE** suggests that the model provides reasonable predictions but has room for improvement.

Screenshots:


Fork
Movie Recommender System
Type or select a movie from the dropdown
Choose an option
Show Recommendations

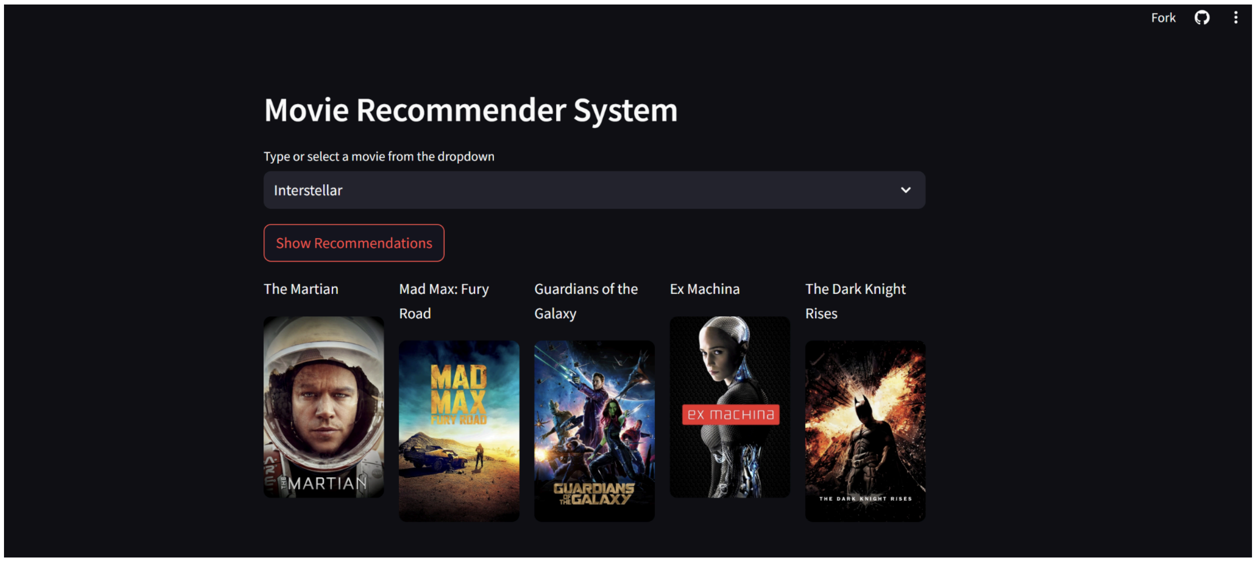

Fork
Movie Recommender System
Type or select a movie from the dropdown
Inter
Interstellar
Interview with the Vampire
Interview with the Assassin
The Intern
The Interview
The Internship
The Interpreter



Movie Recommender System

Type or select a movie from the dropdown

Interstellar



Movie Recommender System

Type or select a movie from the dropdown

Interstellar

The Martian

Mad Max: Fury Road

Guardians of the Galaxy

Ex Machina

The Dark Knight Rises