Movie Recommender System

Introduction

“What movie should I watch this evening?”

Have you ever had to answer this question at least once when you came home from work? As for me—yes, and more than once. From Netflix to Hulu, the need to build robust movie recommendation systems is extremely important given the huge demand for personalized content of modern consumers. Our project aims to build a movie recommendation system using the MovieLens dataset. We will implement collaborative filtering, a popular technique that recommends movies based on user ratings. By analyzing patterns in how users rate different movies, we can suggest the top 5 movies that a user is most likely to enjoy.

This recommendation system will help users discover new content they love while increasing engagement for streaming platforms.

Business Problem

The modern film enthusiast faces an overwhelming decision - a wealth of cinematic options, yet a struggle to find films that align with their preferences. The challenge lies in the initial selection as well as finding movies within the same niche or genre. Users often find themselves lost in the vast sea of content, seeking a solution that not only recommends the first movie but also facilitates a smooth journey through related titles.

Business Objective

The business objectives for us are:

- To create a Collaborative Filtering based Movie Recommendation System.
- Predict the rating that a user would give to a movie that he has not yet rated.
- Minimize the difference between predicted and actual rating (RMSE and MAPE)

Data Understanding

This dataset (ml-20m) utilizes information from IMDb and TMDb and describes 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service. It contains 100836 ratings and 3683 tag applications across 9742 movies.

- Movielens

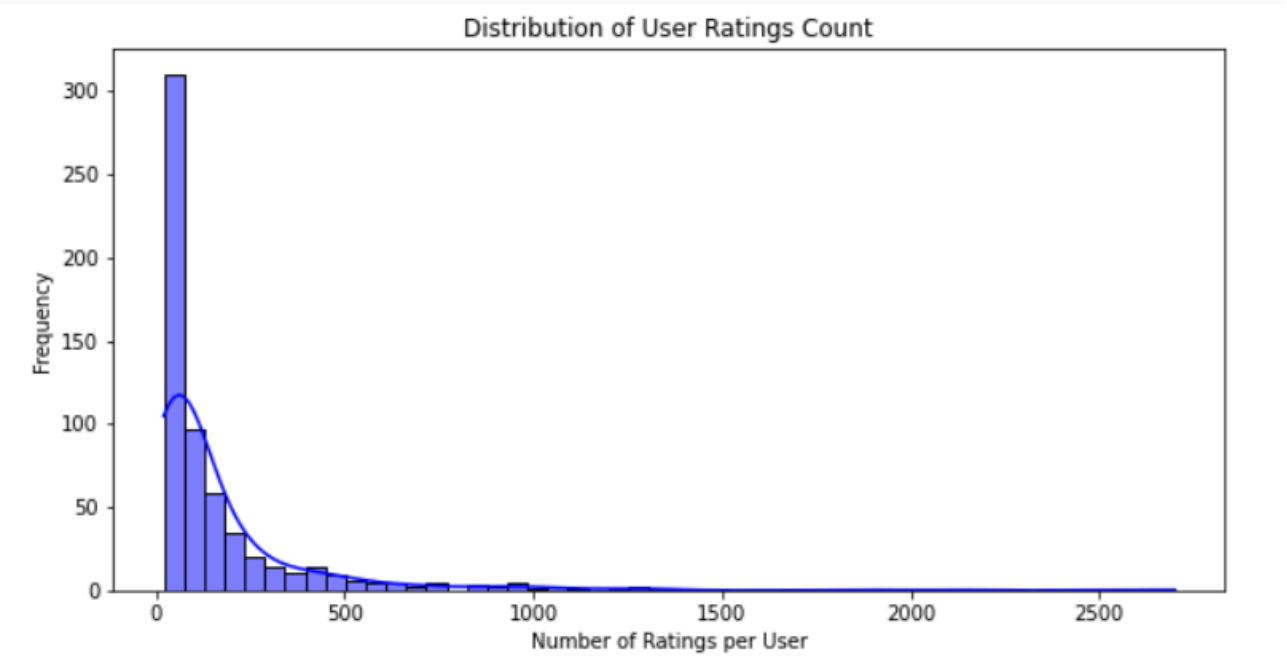
We start by importing the necessary libraries:

- Pandas and Numpy for data handling and numerical operations.
- Matplotlib and Seaborn for visualization.
- Surprise for CF models such as SVD.
- Sklearn for preprocessing and evaluation metrics.

Exploratory Data Analysis

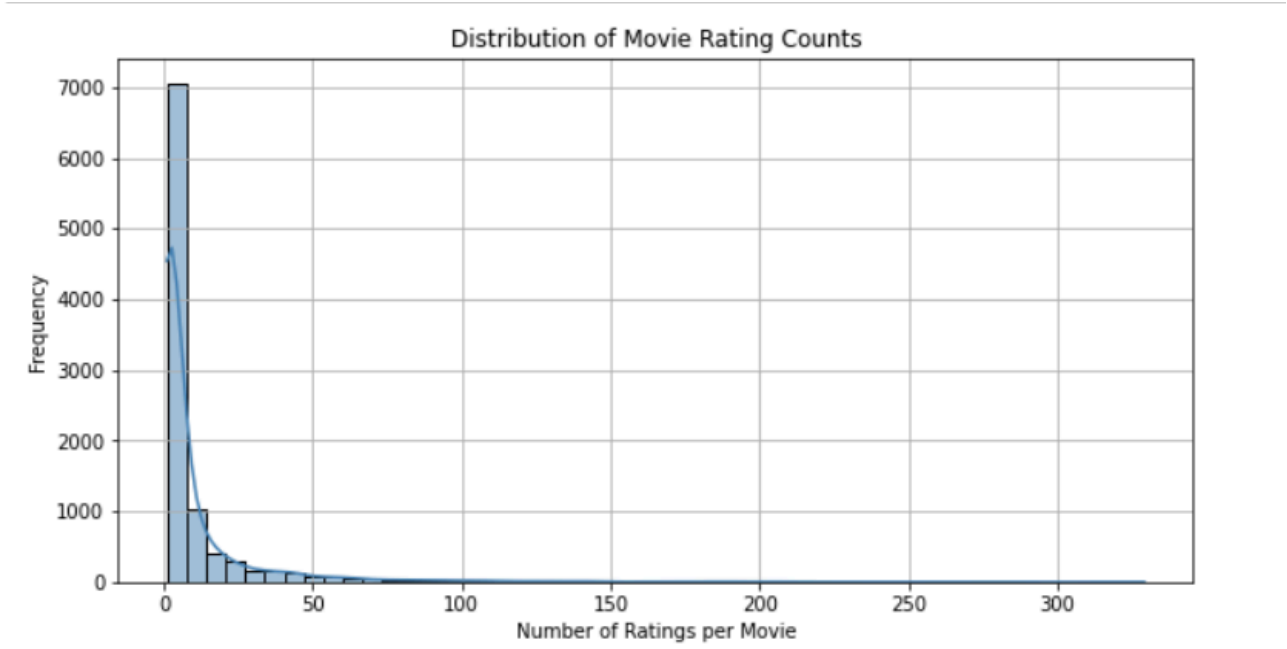
In this section, we explore the datasets inorder to get a guide for model development. We'll look at the distribution of ratings, the number of movies and users, and conduct a genre analysis.

The number of ratings



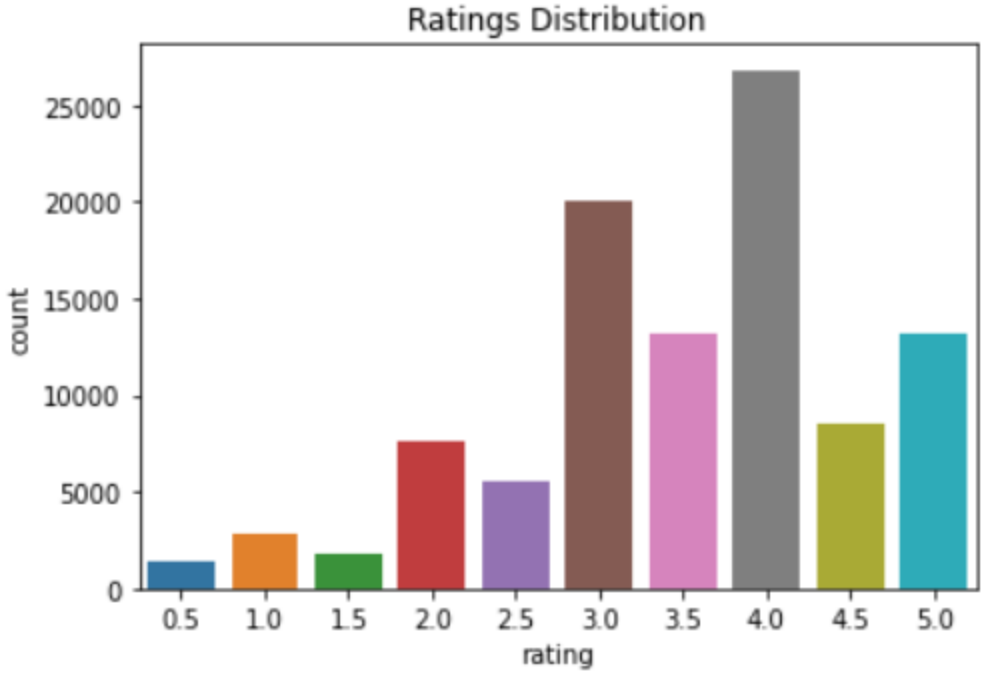
From the above plot, we can see that most users rate very few movies, and few users rate many movies. This shows that the dataset has a lot of users with little interaction

Ratings per movie



From this plot, it is evident that, very many movies have few ratings while few movies have many ratings.

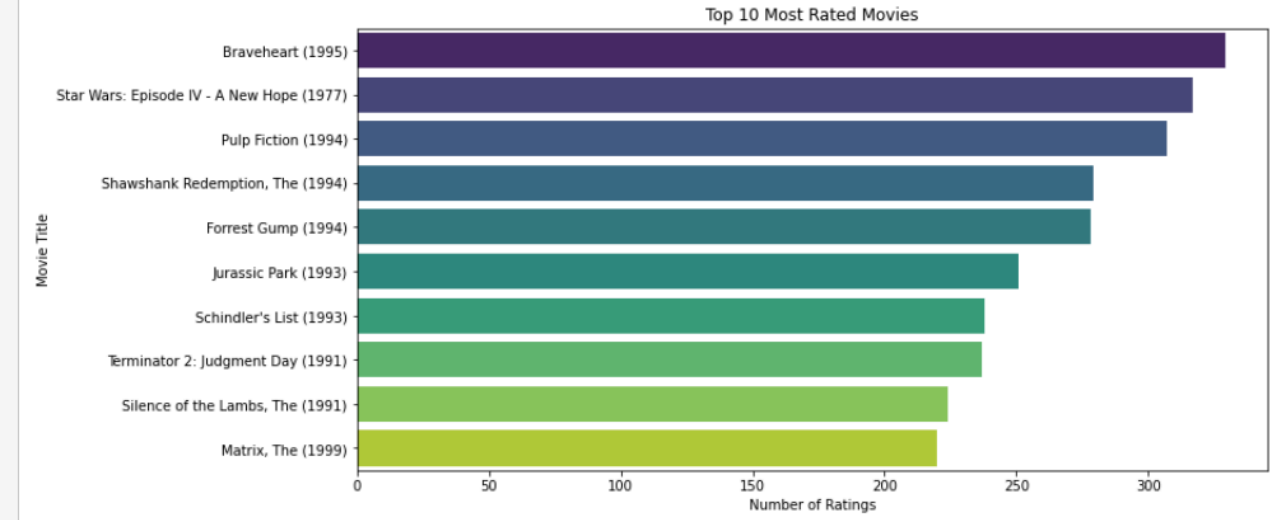
Rating distribution



Most movies recieve a rating of 4.0 followed by 3.0.

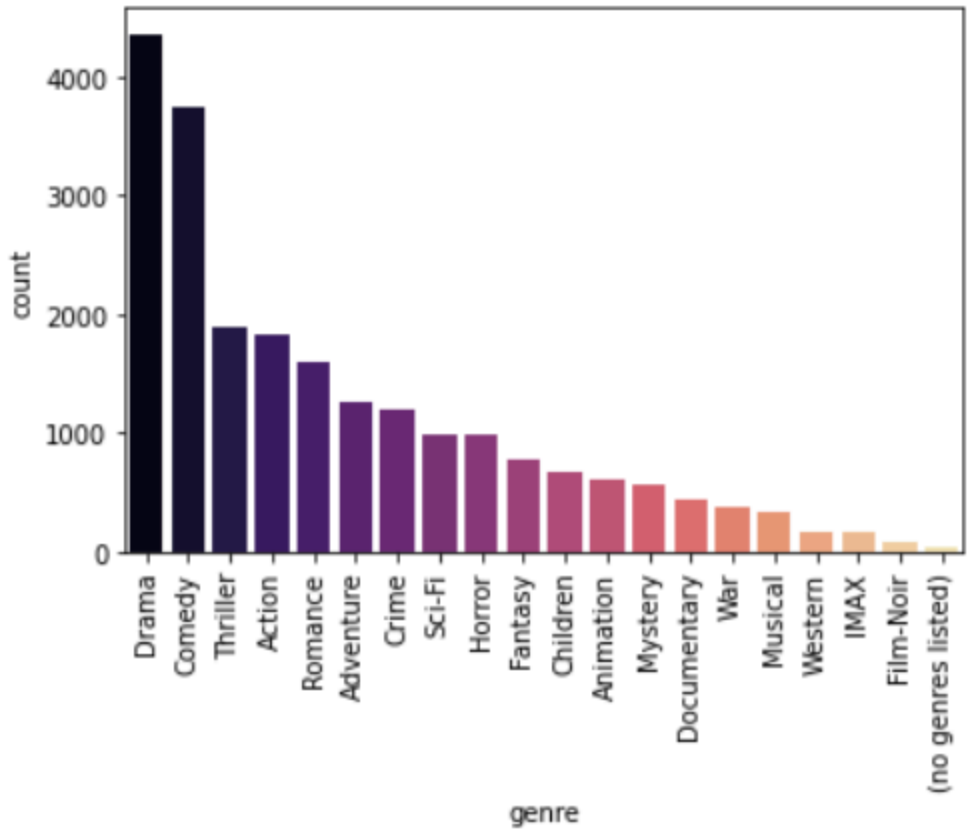
Top 10 most rated movies

Highest and Lowest Rated Movies



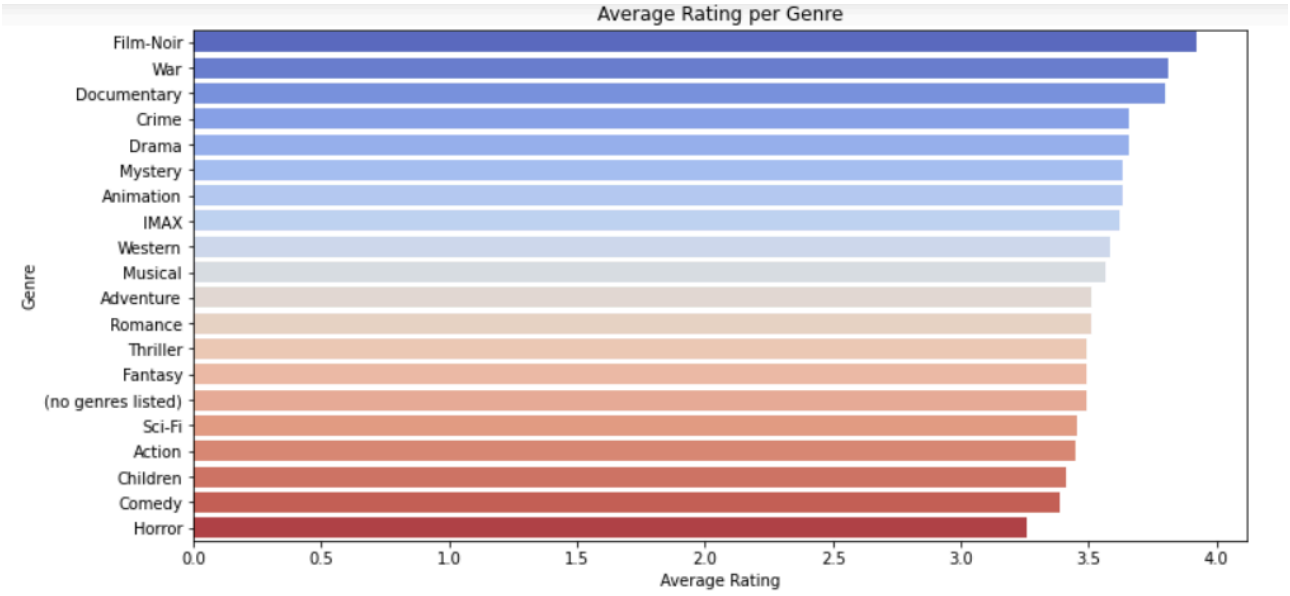
This gragh shows the top 10 most rated movies with Braveheart leading.

The frequency of each movie genre



Drama has the highest ratings followed by comedy while film-Noir has the least count of ratings

The average rating per Genre



Film-Noir, War, and Documentary have the highest average ratings although they appear least frequently in the previous graph. This implies that these genres have niche audiences, who enjoy these movies and rate them highly.

Modeling

Creating a Recommendation Model

The next step is to build a recommendation model by first implementing CF using SVD. The model is then evaluated using cross validation, measuring RMSE and MAE.

Cold Start Handling To make this work for new users, we'll use genre based filtering where instead of using ratings, the movies will be recommended based on global genre popularity. For returning users, their past ratings will be used to compute personalized genre scores.

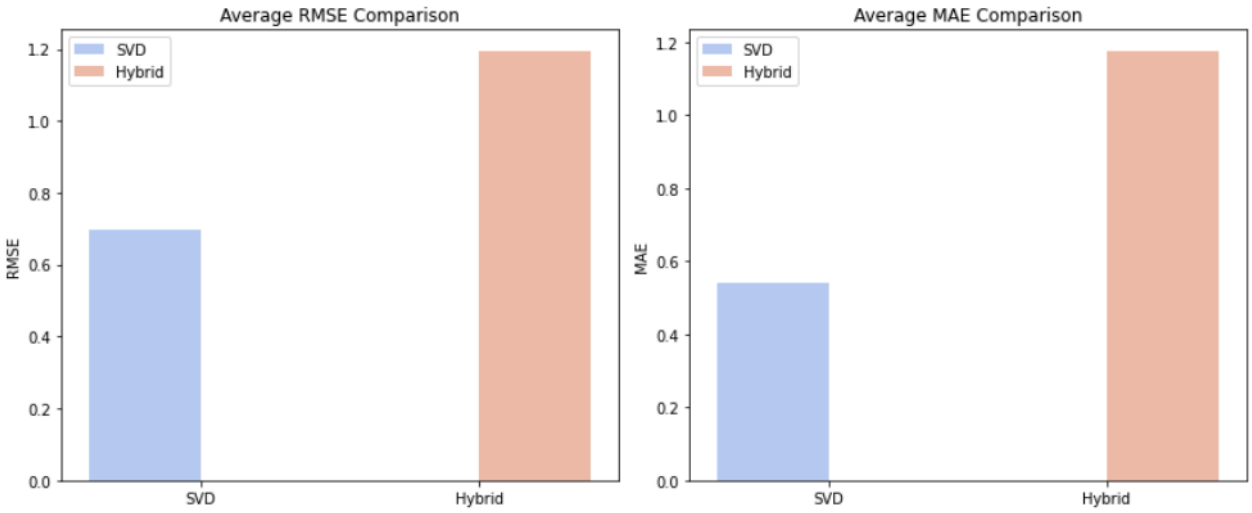
Hybrid Recommendation System

To improve recommendation quality, CF (SVD) is blended with CBF (genre similarity). This hybrid approach balances personalized predictions with genre-based similarities, helping address the cold start problem for new users.

The final score for each movie is calculated using the formula: $$ \text{final score} = \alpha \times \text{SVD score} + (1 - \alpha) \times \text{Genre score} $$ where α controls the weight of CF vs. CBF.

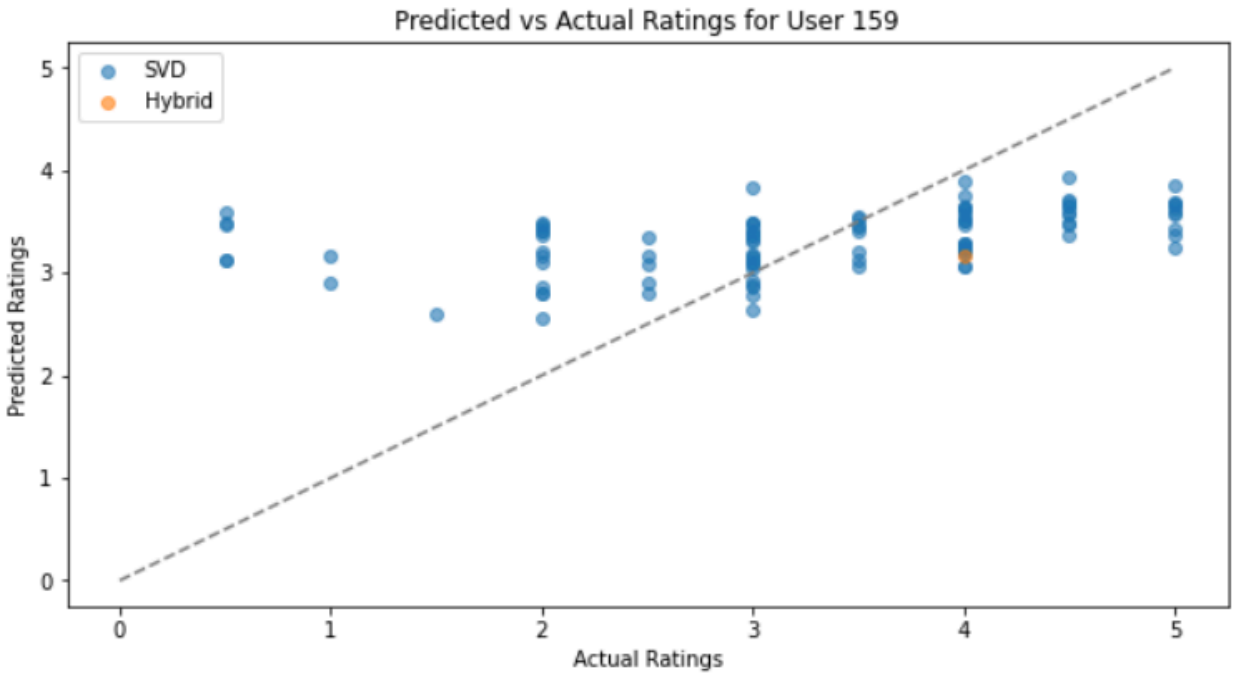
The next step to compare the performance of the model is to visualize the average RMSE and MAE for each model across randomly selected users which will help us understand the accuracy of the recommendations produced by each model in a more clear way. We'll use a barplot, scatter plot, and KDE plot.

Barplot showing RMSE and MAE



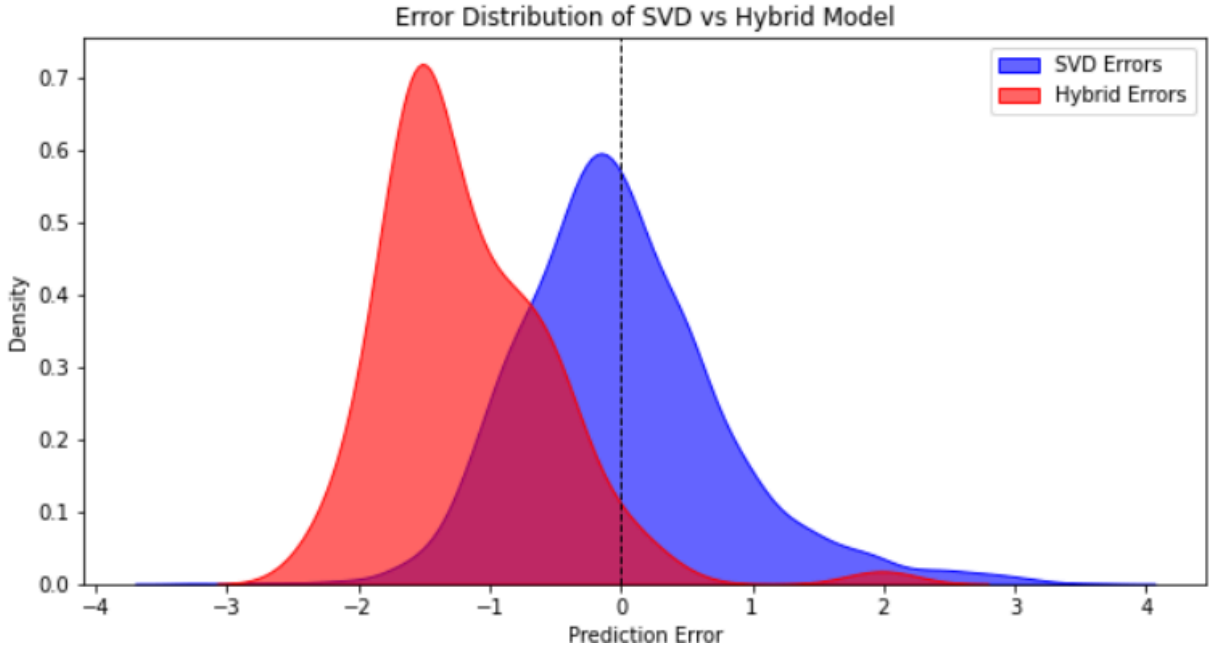
The SVD model has an average RMSE of ~0.8 and an average MAE of ~0.7 compared to the average RMSE and MAE of the Hybrid model which are ~1.3 and ~1.2 respectively. The SVD model outperforms the Hybrid model.

Scatter plot for predictions vs actual ratings



SVD predictions are close to the diagonal reference line (y=x) than hybrid predictions. This suggests that the hybrid model struggles to balance Cf and CBF contributions effectively.

KDE showing Error Distribution Plot



The Hybrid Model appears to be biased negatively but appears to be more consistent and

Recommendations

- Collaborative filtering (SVD) is the best model for the recommender system since it has a lower RMSE an MAE compared to he hybrid recommendation system.
- When a user is new ,recommend movies based on their popularity while, for a current user , use their previous information on movie ratings and genres preferred to tailor recommendation.

Conclusion

In this Analysis, we evaluated the performance of different models for predicting movie ratings; SVD and a Hybrid model. Though the hybrid model was more consistent, the goal was to have errors close to zero on average which is why the SVD model is more preferable.

Next Steps:

- **Model Tuning:** Further hyperparameter tuning for both models.
- **Hybrid model enhancements:** Advanced hybridization techniques such as weighted blending or even adding more CBF should be considered to help reduce hybrid model's bias and improve performance.
- **Cold-Start Problem:** In the analysis, we attempted to solve the problem using global genre preference. Popularity-Based Recommendatoins should also be considered to try and address the problem.