Summary Report of Capstone Project: Movie Recommendation System

Jinbo Li

Chapter 1: Introduction

1.1 Overview

With the increasing availability of movies on various platforms, the challenge of selecting the most preferred movies for users has become a critical issue in the entertainment industry. With so many options to choose from, it can be overwhelming to decide what to watch next. Therefore, the field of recommendation systems has been the subject of important and extensive research in recent years, as more and more companies, such as Netflix and Amazon Prime, developed their own recommendation systems to provide users with a more customized and personalized viewing experience. And this project aims to build a movie recommendation system to produce a ranked list of movies to provide customized and personalized recommendations based on users' preferences, rating history, and other relevant movies' features.

In this project, 3 different kinds of recommendation mechanisms are implemented. First one is content-based filtering, and second one is collaborative filtering. Then I combined content-based filtering and collaborative filtering to create a hybrid filtering as the third one that takes advantage of the strengths of both methods. By leveraging the power of collaborative filtering and content-based filtering, this system can provide accurate and relevant recommendations that help users discover new movies they might not have otherwise considered. Apart from this, an interactive web page was built. This web page is user-friendly and intuitive, with clear instructions and a simple interface, which allows users to easily access and provides users with an enhanced experience.

1.2 Prior work

Although there has been some prior work on movie recommendation system, a common disadvantage is that they mainly rely on a single filtering technique and often suffer from the cold start problem and limited accuracy. This project addresses this issue through combining matrix factorization and content-based filtering to improve recommendation accuracy and mitigate the cold start problem. The advantage of it is that it can leverage both user historical behavior and movie features, making the movie recommendation system more accurate and effective for a wide range of users. Meanwhile, the interactive web page

also provides users with a better recommendation experience.

Chapter 2: Description of Datasets

2.1 About Datasets

The datasets in the project are extracted from the official website of IMDB and MovieLens, consisting of over 100k movie rating records from 671 users and including over 45k movies, with a total of 26 columns containing detailed movie information, such as movie description, movie casts, movie keywords, movie overviews and so on etc.

There are total 6 csv files. They are movies, ratings, keywords, crew, movies_m and links.

The file ‘movies’ includes some basic movie information and attributes, consisting of 24 columns.

- **Adult**: A Boolean value indicating whether the movie is intended for adult audiences. (False or True)
- **Belongs_to_collection**: Information about the collection or franchise that the movie belongs to.
- **Budget**: The budget of the movie
- **Genres:** The genre of the movie, such as action, comedy, drama, horror, or romance.
- **Homepage**: The URL of the official website for the movie or production company.
- **Id**: A unique identifier for each movie in the dataset.
- **Imdb_id**: A unique identifier for each movie in the IMDB.
- **Original_language**: The original language of the movie, represented as a two-letter code (such as "en" for English, "fr" for French, "es" for Spanish, etc.).
- **original_title**: The original title of the movie.
- **Overview**: A short description or summary of the movie.
- **Popularity**: A numeric value representing the popularity of the movie.
- **Poster_path**: The URL of the movie poster image.
- **Production_companies**: The production companies involved in making the movie.
- **Production_countries**: The countries where the movie was produced.
- **release_date**: The date the movie was released in theaters.
- **Revenue**: The total amount of revenue generated by the movie.
- **Runtime**: The duration of the movie in minutes.
- **Spoken_languages**: The languages spoken in the movie.
- **Status**: The current status of the movie.

- **Tagline**: A short phrase or slogan used to promote or describe the movie.
- **Title**: The official title of the movie.
- **Video**: Indicates whether the movie has a video trailer available. (False or True)
- **Vote_average**: The average rating of the movie as voted on by users.
- **Vote_count**: The total number of votes or ratings that the movie has received from users.

The file ‘ratings’ includes the rating history of movies from 671 users, consisting of 4 columns.

- **userId**: The unique identifier for each user who has provided ratings on the movies.
- **movieId**: A unique identifier for each movie in the dataset, it is correlated with ‘id’ in the ‘movies’.
- **Ratings**: The rating provided by users for a movie, its scale from 0.5 to 5, with increments of 0.5.
- **Timestamps**: The date and time at which a user provided a rating for a movie.

The file ‘keywords’ includes the keywords information of movies, consisting of 2 columns.

- **Id**: A unique identifier for each movie in the dataset, it is correlated with ‘id’ in the ‘movies’.
- **Keywords**: A list of keywords associated with each movie in the dataset.

The file ‘crew’ includes the information of director, cast, and crew in the movies, consisting of 3 columns.

- **Cast**: A list of actors and actresses who appear in each movie in the dataset.
- **Crew**: A list of crew members who worked on each movie in the dataset, including directors, writers, producers, and other production staff.
- **Id**: A unique identifier for each movie in the dataset, it is correlated with ‘id’ in the ‘movies’.

The file ‘movies_m’ includes the very basic information of movies, consisting of 3 columns. It is used to build the collaborative filtering model by avoiding excessive data computation.

- **MovieId**: A unique identifier for each movie in the dataset, it is correlated with ‘id’ in the ‘movies’.
- **Title**: The official title of the movie.
- **Genres**: The genre of the movie, such as action, comedy, drama, horror, or romance.

The file ‘link’ links the information from IMDB and MovieLens, consisting of 3

columns.

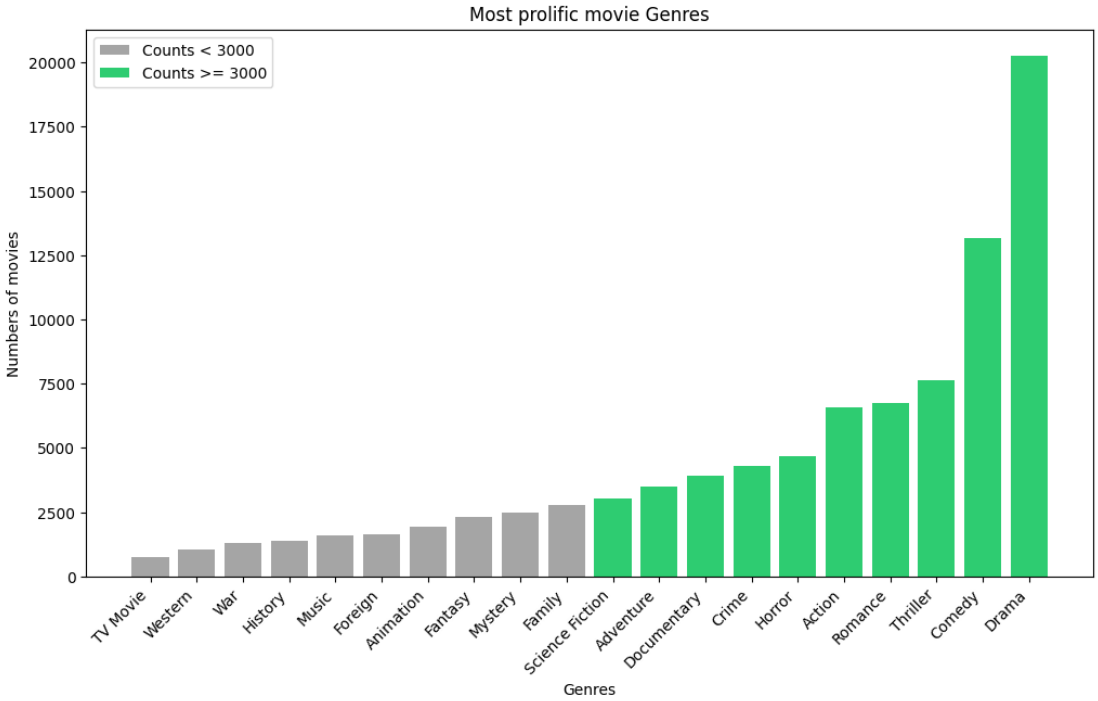
- **MovieId**: The movie id in the ‘link’ dataset.
- **ImdbId**: The movie id in the IMBD.
- **TmdbId**: The movie id in the MovieLens.

2.2 EDA

Some EDA plots are drawn to provide an intuitive understanding of the dataset, which can then be used to guide further analysis and modeling.

2.2.1 Most Prolific Movie Genres

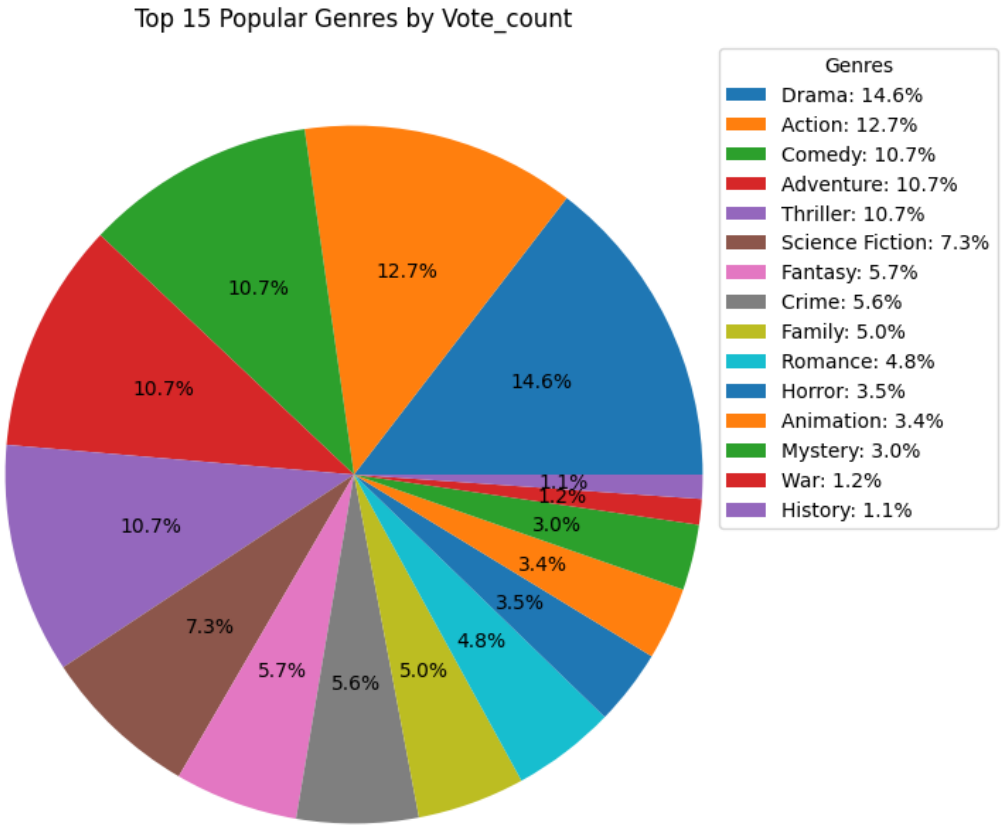
This is a histogram of the most prolific movie genres.



Based on the plot, we can observe that drama, comedy, and thriller are the most prolific movie genres, meaning they have the highest number of movies produced in those genres. This information can be useful in understanding the trends in movie production.

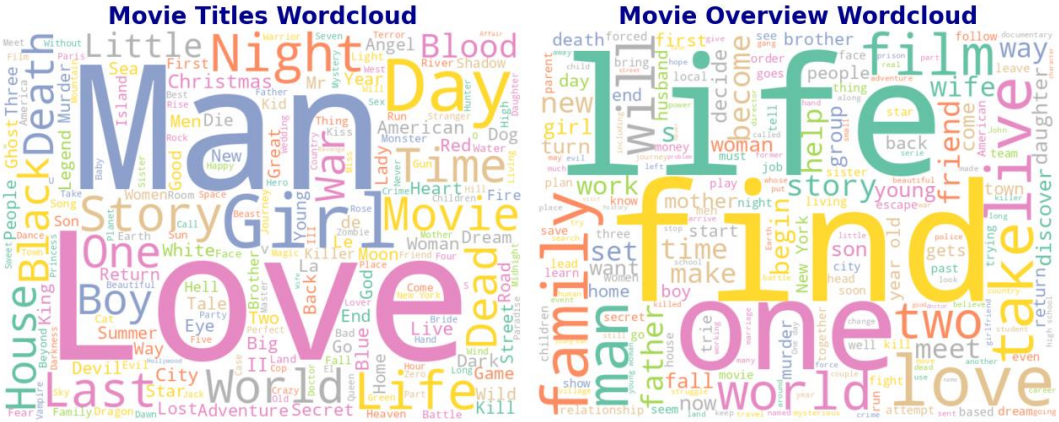
2.2.2 Top 15 Popular Genres by Vote Count

This is a pie chart shows the top 15 popular movie genres by using vote count number as the index.



Based on the plot, we can observe that drama, action, and comedy are the most popular movie genres, as they have the highest number of vote counts. This information may be useful in understanding audience preferences for movie genres.

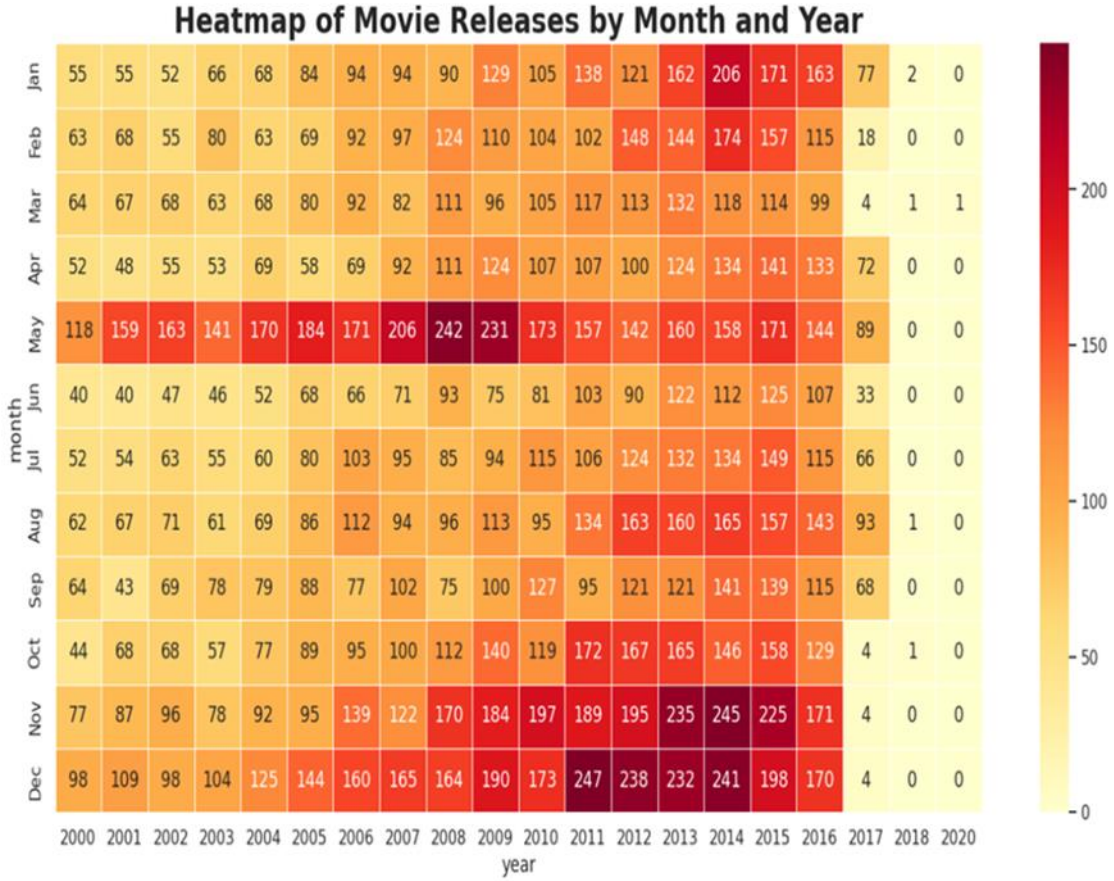
2.2.3 Movie Titles Word Cloud and Movie Overview Word Cloud



They are the plots of movie titles word cloud and movie overview word cloud. On the left side, we have a word cloud of movie titles which displays the most commonly occurring words in the titles. On the right side, we have a word cloud of movie overviews which displays the most commonly occurring words in the

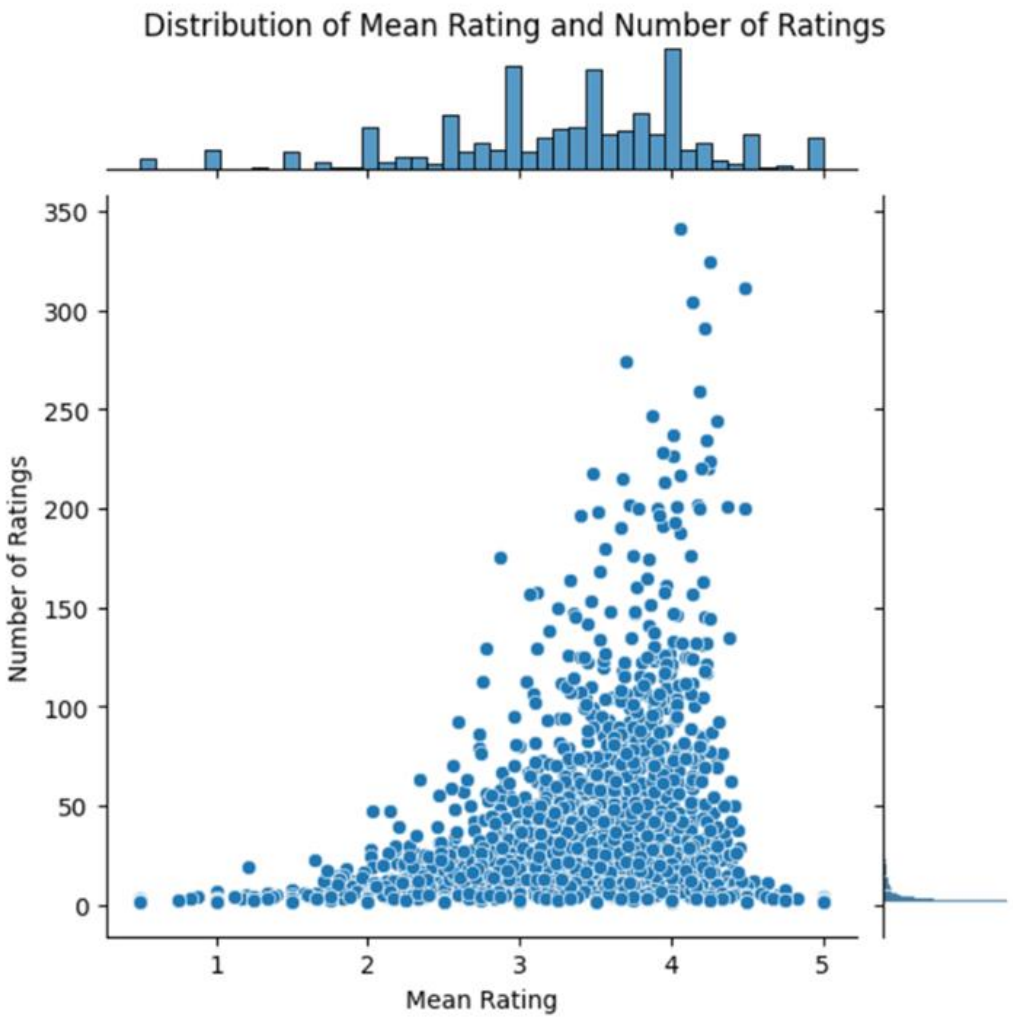
movie summaries. The word cloud of movie titles reveals that 'Love' is the most frequently used word in movie titles. On the other hand, the word cloud of movie overviews indicates that the words 'Life' and 'find' are the most commonly used words in movie summaries. This information can provide insights into the themes and topics that are most commonly depicted in movies.

2.2.4 Heatmap of Movie Releases by Month and Year
This a heatmap plot shows the movie releases by month and year.



The heatmap displays the distribution of movie releases by month and year for all movies released in the 21st century. This visualization provides valuable insights into the seasonal trends of movie releases and can be used to identify patterns in the movie industry over the years.

2.2.5 Distribution of Mean Rating and Number of Ratings
This is a scatter plot shows the distribution of mean rating and number of ratings.



The scatter plot displays the relationship between the average rating and the number of ratings for the movies in the dataset. It can be observed that the majority of movies have an average rating of around 4. Furthermore, the distribution of the number of ratings indicates that most movies have less than 150 ratings. This information can be used to identify popular movies and to evaluate the reliability of the rating scores.

Chapter 3: Building Recommenders

3.1 Top 50 movies

I applied a weighted rating formula to select the best 50 movies. The purpose of this step was to provide users with a diverse range of options. When users are unsure of what movie to watch, they can refer to the top 50 movie charts for inspiration, as all the movies in the top 50 charts are high-quality films.

This is the weighted rating formular: $w = R\left(\frac{v}{v+m}\right) + C\left(\frac{m}{v+m}\right)$

W stands for weighted rating. R stands for average rating of the movie. V stands for number of votes for the movie. M stands for minimum votes required for a movie to be listed in the Top 50. Here I used 90% as the cutoff to remove the movies of low number of votes. C stands for mean vote average.

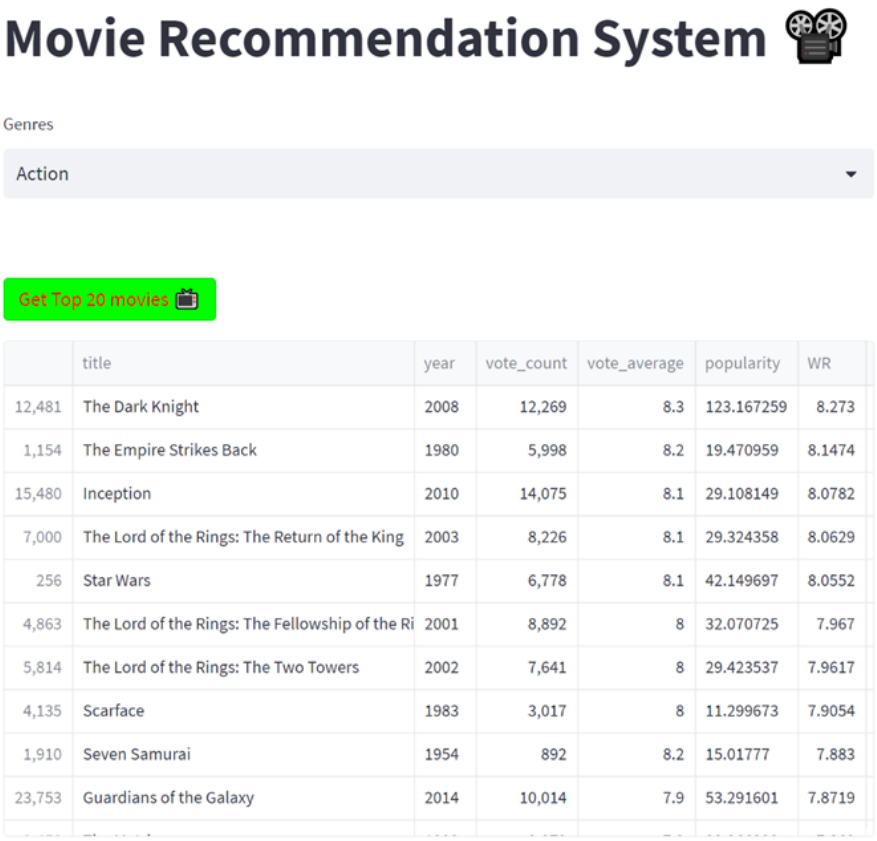
The purpose of using this formula is to provide a more comprehensive ranking of the movies based on both the average rating and the number of votes. It ensures that the movies with a higher number of votes and a higher average rating are ranked higher. The minimum votes required parameter ensures that only movies with a significant number of votes are considered, thereby avoiding bias towards newer or less popular movies. The mean vote average parameter provides a normalization factor, ensuring that movies with a higher rating but a lower number of votes are not ranked higher than movies with a lower rating but a higher number of votes.

After calculating the weighted rating of each movie, I then sorted these movies in descending order by their weighted rating scores and returned the top 50 movies with the highest weighted rating score.

	title	year	vote_count	vote_average	popularity	genres	WR
314	The Shawshank Redemption	1994	8,358	8.5	51.645403	Drama Crime	8.4459
834	The Godfather	1972	6,024	8.5	41.109264	Drama Crime	8.4254
10,309	Dilwale Dulhania Le Jayenge	1995	661	9.1	34.457024	Comedy Drama Romance	8.4215
12,481	The Dark Knight	2008	12,269	8.3	123.167259	Drama Action Crime Thriller	8.2655
2,843	Fight Club	1999	9,678	8.3	63.869599	Drama	8.2564
292	Pulp Fiction	1994	8,670	8.3	140.950236	Thriller Crime	8.2514
522	Schindler's List	1993	4,436	8.3	41.725123	Drama History War	8.2066
23,673	Whiplash	2014	4,376	8.3	64.29999	Drama	8.2054
5,481	Spirited Away	2001	3,968	8.3	41.048867	Fantasy Adventure Animation Family	8.1961
2,211	Life Is Beautiful	1997	3,643	8.3	39.39497	Comedy Drama	8.1872
1,178	The Godfather: Part II	1974	3,418	8.3	36.629307	Drama Crime	8.1801
1,152	One Flew Over the Cuckoo's N	1975	3,001	8.3	35.529554	Drama	8.1643
351	Forrest Gump	1994	8,147	8.2	48.307194	Comedy Drama Romance	8.1503
1,154	The Empire Strikes Back	1980	5,998	8.2	19.470959	Adventure Action Science Fiction	8.1329
1,176	Psycho	1960	2,405	8.3	36.826309	Drama Horror Thriller	8.1327
18,465	The Intouchables	2011	5,410	8.2	16.086919	Drama Comedy	8.1258

(A part of screenshot of the top 50 movies chart)

Additionally, I created top 20 movies charts for each popular genre, which was generated by implementing movie genre as the index on the same weighted rating formular as top 50 movies. This allows users to quickly find highly rated movies within their preferred genres.



	title	year	vote_count	vote_average	popularity	WR
12,481	The Dark Knight	2008	12,269	8.3	123.167259	8.273
1,154	The Empire Strikes Back	1980	5,998	8.2	19.470959	8.1474
15,480	Inception	2010	14,075	8.1	29.108149	8.0782
7,000	The Lord of the Rings: The Return of the King	2003	8,226	8.1	29.324358	8.0629
256	Star Wars	1977	6,778	8.1	42.149697	8.0552
4,863	The Lord of the Rings: The Fellowship of the Ri	2001	8,892	8	32.070725	7.967
5,814	The Lord of the Rings: The Two Towers	2002	7,641	8	29.423537	7.9617
4,135	Scarface	1983	3,017	8	11.299673	7.9054
1,910	Seven Samurai	1954	892	8.2	15.01777	7.883
23,753	Guardians of the Galaxy	2014	10,014	7.9	53.291601	7.8719

(The screenshot of Top 20 movies in popular genres)

3.2 Content-Based Filtering

Content-based filtering is a popular recommendation technique that is to give the recommendations only based on attributes of the items, in this case, movies, to generate recommendations. Here in this project, it uses the movies' features or attributes to find the similar movies to generate recommendations.

Several movie features were used for content-based filtering, including the movie descriptions, overview, taglines, director, cast, keywords, and genres. Before using these features, some preprocessing was performed to make the data more usable. For example, leading and trailing spaces were removed from the text features, and the text was converted to lowercase to ensure consistency. Furthermore, I also used a package called snowball Stemmer to reduce words to their base or root form in the context features. This helps to group words with similar meanings together, the words 'cats' and 'cat,' which would be considered as one word after using this package.

Considering the excessive amount of cast data, I only considered the top three cast members of each movie for my model. For the director feature, since many movies only have one director, I counted the director's name three times in each movie to give it more weight relative to the other features. This way, the director's influence on the movie could be better captured in the model.

Next, I created a tf-idf vectorizer using the preprocessed movie features. The tf-idf vectorizer converts text into numerical vectors, representing words and their importance in the text. It uses a method called tf-idf (term frequency-inverse document frequency) to calculate word importance based on their frequency of occurrence in the document and use them for similarity calculation. The tf-idf method takes into account the frequency of each word in the document as well as the frequency of the word in the entire dataset. This approach ensures that words that are common across many movies (such as "the" or "and") are given less weight in the similarity calculation compared to words that are unique to a specific movie.

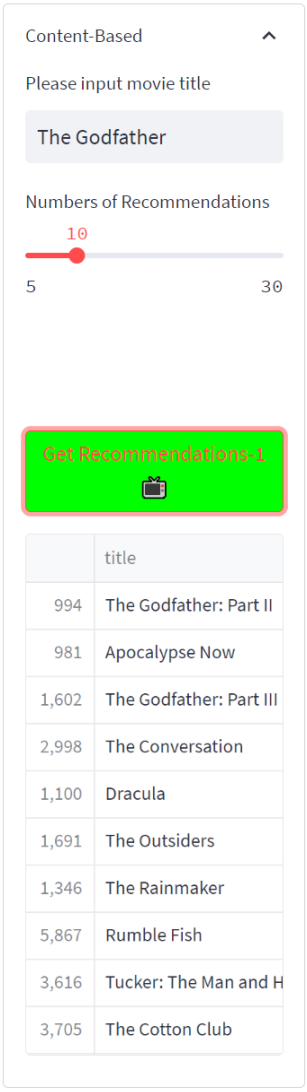
I then used this vectorizer to transform the preprocessed movie features into a tf-idf importance matrix. The tf-idf importance matrix represents each movie feature as a vector in a high-dimensional space where the length of each vector represents the importance of the feature. After creating the tf-idf importance matrix, I calculated the cosine similarity of these movie features. Cosine similarity measures the cosine of the angle between two vectors, indicating the similarity between the vectors. In this case, it indicates the similarity between two movies based on their features.

$$\mathrm{TF-IDF} = \mathrm{TF(x,y)} \times \log\left(\frac{\mathrm{N}}{\mathrm{DF(x)}}\right)$$

($TF(x,y)$= frequency of term x in document y
$DF(x)$= number of documents containing x
N= total number of documents)

Then, I used the previous weighted rating formula to remove unpopular and low-rated movies and give the final content-based recommendations. This step ensures that only high-quality and popular movies are recommended to the users.

Finally, this recommender provided the users with a list of content-based movie recommendations based on their search movies. These recommendations are generated by finding the movies with the highest cosine similarity to the search movies and sorting them in descending order of their weighted rating.



	title	vote_count	vote_average	genres	director	cast	year	keywords	WR
994	The Godfather: Part II	3,418	8.3	Drama Crime	francisfordcoppola	alpacino robertduvall dianekeaton	1974	italo-american cuba vorort melancholi prais reveng mafia lawy	8.1801
981	Apocalypse Now	2,112	8	Drama War	francisfordcoppola	martinsheen marlonbrando robertduvall	1979	guerrilla river vietnam vietcong cambodia armi insan tribe	7.8323
1,602	The Godfather: Part III	1,589	7.1	Crime Drama Thriller	francisfordcoppola	alpacino dianekeaton andygarcia	1990	itali christian newyork assassin italo-american vatican pope co	6.9644
2,998	The Conversation	377	7.5	Crime Drama Mystery	francisfordcoppola	genehackman johncazale fredericforrest	1974	sanfrancisco paranoia audiotap wiretap shadow	6.9393
1,100	Dracula	1,087	7.1	Romance Horror	francisfordcoppola	garyoldman winonaryder anthonyhopkins	1992	adulteri maze vampir bite remak roughsex wake correspond	6.9099
1,691	The Outsiders	293	6.9	Crime Drama	francisfordcoppola	mattdillon ralphmacchio c.thomashowell	1983	streetgang children'shom comingofag gang juveniledelinqu basedon	6.4473
1,346	The Rainmaker	239	6.7	Drama Crime Thriller	francisfordcoppola	matttdamon dannydevito jonvoight	1997	juror proof courtcas leukemia lawyer courtroom	6.2662
5,867	Rumble Fish	141	6.8	Action Adventure Crime Drama Romance	francisfordcoppola	mattdillon mickeyrourke dianelane	1983	streetgang billard gang	6.1718
3,616	Tucker: The Man and His Dream	71	6.5	Drama	francisfordcoppola	jeffbridges joanallen martinlandau	1988		5.8892
3,705	The Cotton Club	71	6.5	Music Drama Crime Romance	francisfordcoppola	richardgere gregoryhines dianelane	1984	jazz jazzmusician music mafia	5.8892

(Recommendations by content-based filtering
using movie ‘The Godfather’ as example)

One advantage of content-based filtering is that it does not require users' history to make recommendations. However, it has some limitations, such as the limited ability to recommend diverse movies outside the user's preferences and the inability to capture the user's taste evolution over time. Additionally, it may miss out on hidden attributes that are not easily discernible by the vectorizer. Lastly, it is difficult to have a quantitative metric to judge the recommender's performance.

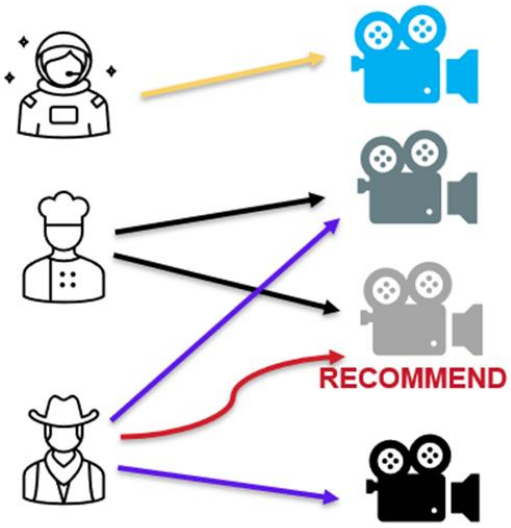
3.3 Collaborative Filtering

Considering the limitations of content-based filtering, I implemented the collaborative filtering to improve my recommendation models.

3.3.1 User-Based Collaborative Filtering

User-based collaborative filtering is a recommendation method that generates personalized recommendations based on the behavior of similar users. This method works by identifying similar users who have similar preferences and interests to the active user, and then recommending movies that these similar users have liked or rated highly.

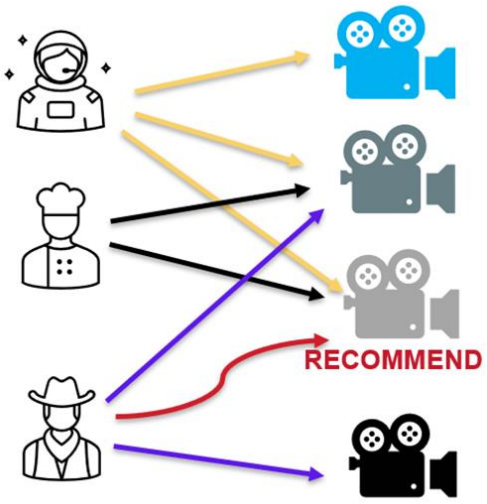
For example, let's say user 1 likes movie 1, user 2 likes movies 2 and 3, and user 3 likes movies 2 and 4. Since user 2 and user 3 both like movie 2, they are considered similar users. If user 3 has not been exposed to movie 3 yet, the system will recommend movie 3 to user 3 based on the fact that user 2, who has similar preferences, likes that movie.



3.3.2 Item-Based Collaborative Filtering

Item-based collaborative filtering is a recommendation method that focuses on identifying similarities between movies. That’s the main difference from user-based collaborative filtering. It analyzes the historical interactions between users and movies and uses this information to identify similar movies based on the patterns of user behavior.

For instance, suppose that user1 likes movie 1, movie 2, and movie 3, while user2 likes movie 2 and movie 3. User3 likes movie 2 and movie 4. In this case, movie 2 and movie 3 are both liked by user1 and user2, indicating that they are similar movies. Since user3 likes movie 2 and has not been exposed to movie 3 yet, the system will recommend movie 3 to user3 based on the similarity between these two movies.



The steps to build collaborative filtering recommender depend on the type of collaborative filtering used. For user-based collaborative filtering, the first step is to find similar users based on interactions with common movies. Then, identify the movies rated highly by these similar users but have not been seen by the target user. Next, calculate the weighted average score for these movies. Finally, rank the movies based on the score and pick the top n movies to give recommendations to the target user.

In contrast, for item-based collaborative filtering, the first step is to calculate the similarity scores between movies based on user ratings. Then, identify the top k movies that are most similar to the movies that the target user has rated. Next, calculate the weighted average score for the most similar movies by the user. Finally, rank the movies based on the score and pick the top n movies to give recommendations to the target user.

To build a user-based or item-based recommender, we need to start by constructing a rating matrix. Build a user-movie rating matrix by using user id as the index for user-based. Where the rows represent users and columns represent movies. Each cell contains the rating of a user for a movie. Conversely, build a movie-user rating matrix by using movie id as index for item-based where rows represent movies and columns represent users.

After building the rating matrix, we can calculate the cosine similarity between each pair of users or movies. For user-based, we will calculate the similarity between users, and for item-based, we will calculate the similarity between movies.

```
[[1.         0.         0.         ... 0.06291708 0.         0.01746565]
 [0.         1.         0.12429498 ... 0.02413984 0.17059464 0.1131753 ]
 [0.         0.12429498 1.         ... 0.08098382 0.13660585 0.17019275]
 ...
 [0.06291708 0.02413984 0.08098382 ... 1.         0.04260878 0.08520194]
 [0.         0.17059464 0.13660585 ... 0.04260878 1.         0.22867673]
 [0.01746565 0.1131753  0.17019275 ... 0.08520194 0.22867673 1.        ]]
```

(User-user similarity matrix)

```
[[1.         0.39451145 0.30651588 ... 0.06380429 0.05551194 0.06380429]
 [0.39451145 1.         0.21749153 ... 0.         0.06297174 0.        ]
 [0.30651588 0.21749153 1.         ... 0.         0.06094934 0.        ]
 ...
 [0.06380429 0.         0.         ... 1.         0.         0.        ]
 [0.05551194 0.06297174 0.06094934 ... 0.         1.         0.        ]
 [0.06380429 0.         0.         ... 0.         0.         1.        ]]
```

(Item-item similarity matrix)

We can then use the similarity scores to predict the rating of a user to a movie by using a formula that takes into account the ratings of similar users or movies. The predicted rating is then used to recommend movies to the user.

Pix: The predicted rating score of i on x
Ni : The set of the nearest neighbors of i
$Sim(i,j)$: The similarity between i and j

$$P_{ix} = \bar{r}_i + \frac{\sum_{j \in N_i} \text{Sim}(i,j) \times \left(r_{jx} - \bar{r}_j\right)}{\sum_{j \in N_i} \left|\text{Sim}(i,j)\right|}$$

Here, I used user id 1 as the target user to test both user-based and item-based recommender systems. The recommendation outputs were significantly different, highlighting the differences between the two approaches.

	movieId	title	genres
1057	1079	Fish Called Wanda, A (1988)	Comedy|Crime
1058	1080	Monty Python's Life of Brian (1979)	Comedy
1113	1136	Monty Python and the Holy Grail (1975)	Adventure|Comedy|Fantasy
1229	1257	Better Off Dead... (1985)	Comedy|Romance
1234	1262	Great Escape, The (1963)	Action|Adventure|Drama|War
1260	1288	This Is Spinal Tap (1984)	Comedy
3332	3421	Animal House (1978)	Comedy
3392	3481	High Fidelity (2000)	Comedy|Drama|Romance
3461	3552	Caddyshack (1980)	Comedy
8144	8827	Bill Cosby, Himself (1983)	Comedy|Documentary

(Output of user-based)

	movieId	title	genres
57687	2984	On Any Sunday (1971)	Documentary
66798	3934	Kronos (1957)	Sci-Fi
79532	6782	Leningrad Cowboys Go America (1989)	Comedy|Musical
79987	6883	Sylvia (2003)	Drama|Romance
83945	8823	Sting II, The (1983)	Comedy|Crime
88512	45969	Career Opportunities (1991)	Comedy|Romance
90767	55417	Irina Palm (2007)	Drama
91867	59834	100 Rifles (1969)	Adventure|War|Western
93548	70121	'Neath the Arizona Skies (1934)	Western
93863	71876	Amelia (2009)	Drama

(Output of item-based)

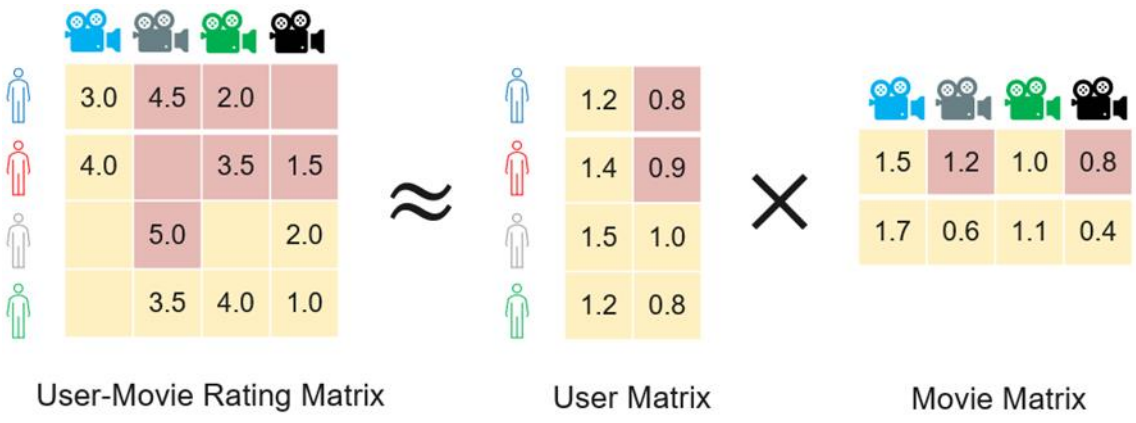
Typically, item-based collaborative filtering has some advantages over user-based collaborative filtering. For example, the movies themselves do not change, unlike users' tastes, which can be fickle and unpredictable. Additionally, there are usually fewer movies than users, making it easier to maintain and compute. Finally, users can create fake IDs to give movies fake ratings, whereas items cannot be easily manipulated in this way.

However, both user-based and item-based collaborative filtering suffer from some limitations. The cold start problem arises when there is little or no data available on a new user or a new item, making it difficult for the system to make accurate recommendations. The dataset sparsity problem arises when the dataset is very sparse, making it challenging to calculate accurate similarities between items and leading to poor recommendations.

3.3.3 Matrix Factorization

Considering the limitations of user-based and item-based collaborative filtering, I used matrix factorization here to improve my recommender.

Matrix Factorization is decomposing a matrix of user ratings into two smaller matrices that capture the underlying factors contributing to the ratings. For example, if we have a matrix of user ratings for movies, we can use matrix factorization to break it down into a user factor matrix and a movie factor matrix. By multiplying these matrices, we can obtain an approximate reconstruction of the original matrix. Then, to generate a movie recommendation for a user, we compute the dot product of the corresponding user vector and movie vector to predict the user's rating for a movie. Based on these predictions, we can recommend the top-rated movies to the user.

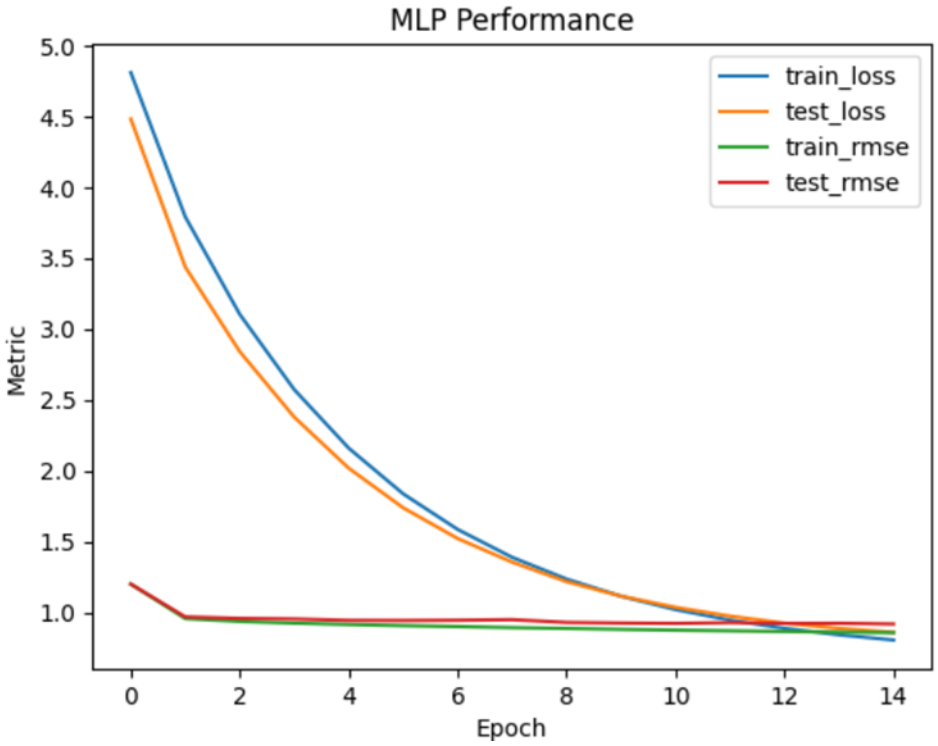


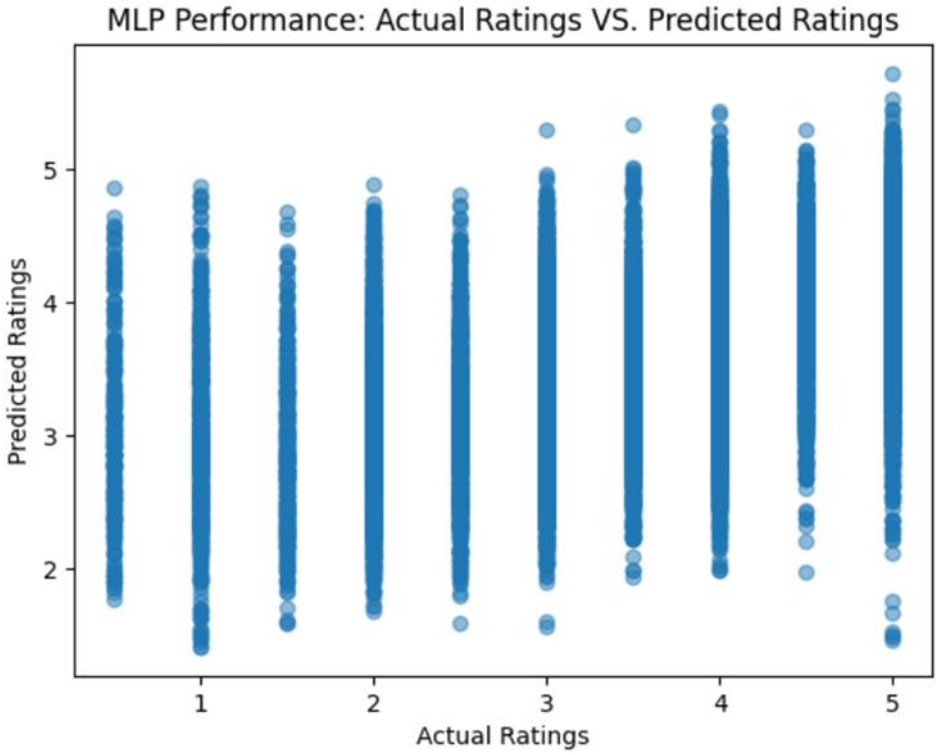
- **MLP Approach**

Here, I utilized Multi-Layer Perceptron to perform matrix factorization. This

approach involves transforming the user and item input features into embedded vectors, flattening, and concatenating them, passing them through multiple dense layers, and finally outputting a predicted rating for each user-item pair.

To evaluate the model's performance, I used two commonly used metrics - Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). The scatter plot of actual ratings and predicted ratings was also examined to visually assess the model's performance. Finally, I got the MAE score of 0.86 and RMSE score of 0.93.



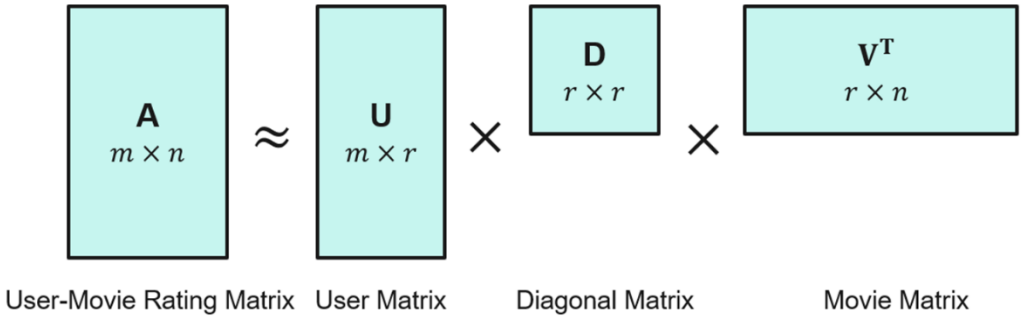


However, as we can see, the model's performance was not very satisfactory, as evident from the metrics and scatter plot. There is still room for improvement in the recommender's performance.

- **SVD (Singular Value Decomposition) Approach**

Considering the not satisfactory performance of previous MLP model, I implement another approach called SVD in matrix factorization.

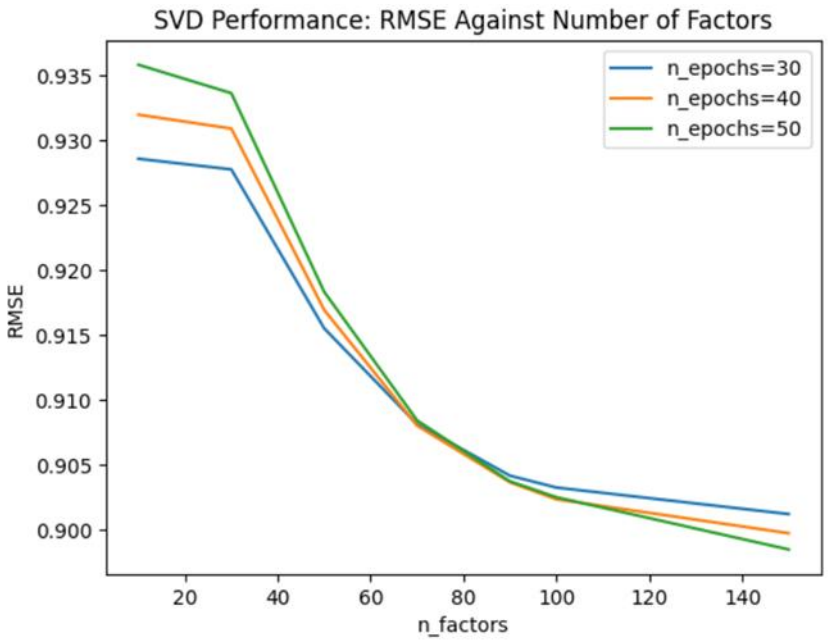
SVD works by decomposing a large matrix of user-movie ratings into three smaller matrices: a user matrix, a movie matrix, and a diagonal matrix. The diagonal matrix contains the singular values, which represent the degree of importance of each feature in the original matrix. The user matrix represents the preferences of each user, while the movie matrix represents the characteristics of each movie. The diagonal matrix captures the relative importance of each feature, such as genre or director, in predicting how much a user will enjoy a particular movie.

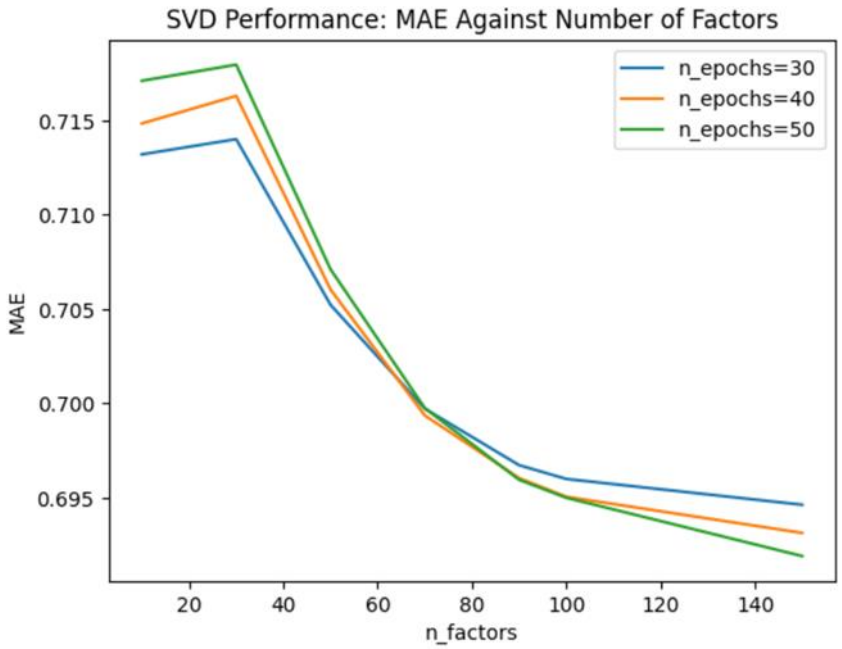


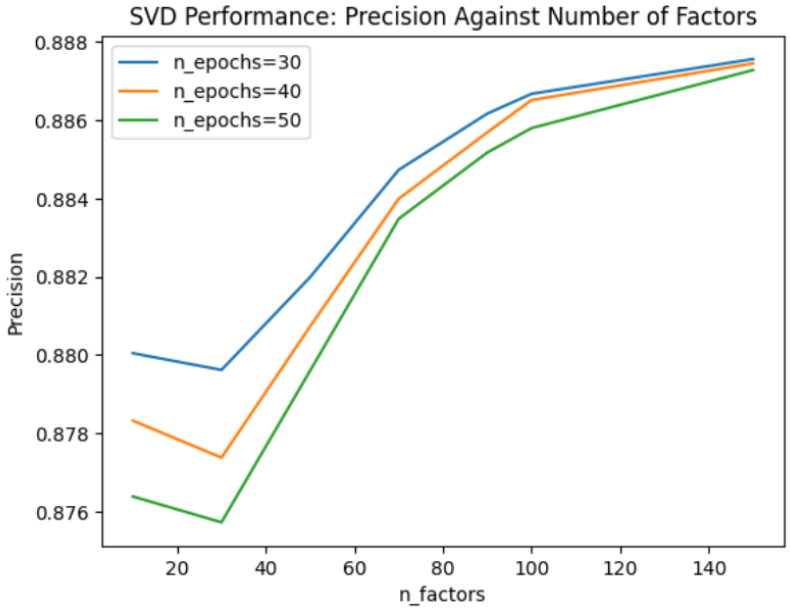
SVD is used to make personalized recommendations by filling in missing values in the user-movie rating matrix. Once the three smaller matrices are computed, the missing values can be estimated by multiplying these 3 matrices together. The resulting reconstructed matrix can then be used to make personalized recommendations for movies that a user has not yet interacted with, based on their predicted ratings.

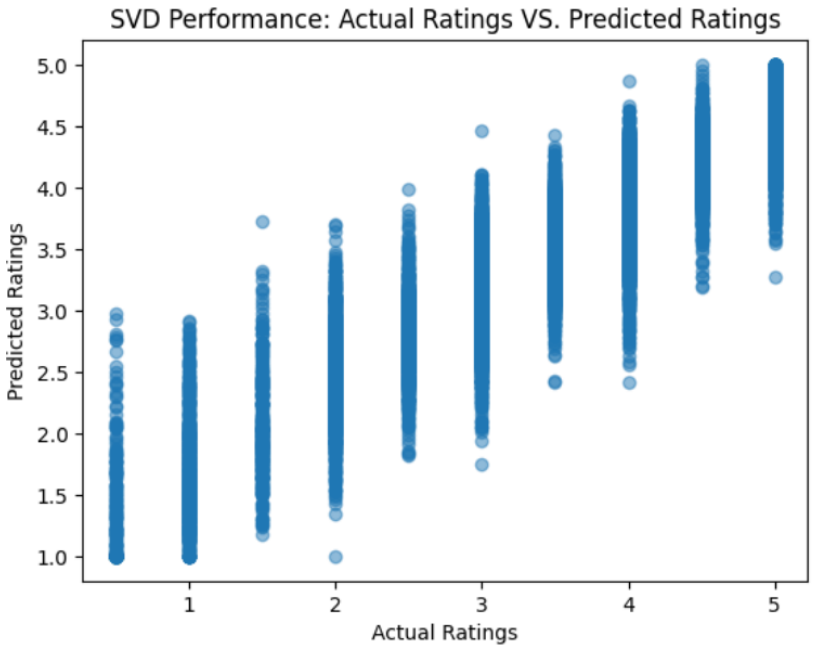
Apart from MAE and RMSE, I also used precision as the metrics to evaluate the model's performance. Here, in the SVD model, I got the MAE score of 0.67, the RMSE score of 0.87 and the precision of 0.88.

To further illustrate the model's effectiveness, I tested it on a random specific example. Specifically, I tested the rating of movie id 3671 from user id 1. The true rating for this movie by user 1 is 3, while the SVD model predicts a rating of 2.97.




SVD Performance: MAE Against Number of Factors
n_epochs=30
n_epochs=40
n_epochs=50
MAE
0.715
0.710
0.705
0.700
0.695
20
40
60
80
100
120
140
n_factors


SVD Performance: Precision Against Number of Factors
n_epochs=30
n_epochs=40
n_epochs=50
Precision
0.888
0.886
0.884
0.882
0.880
0.878
0.876
20
40
60
80
100
120
140
n_factors



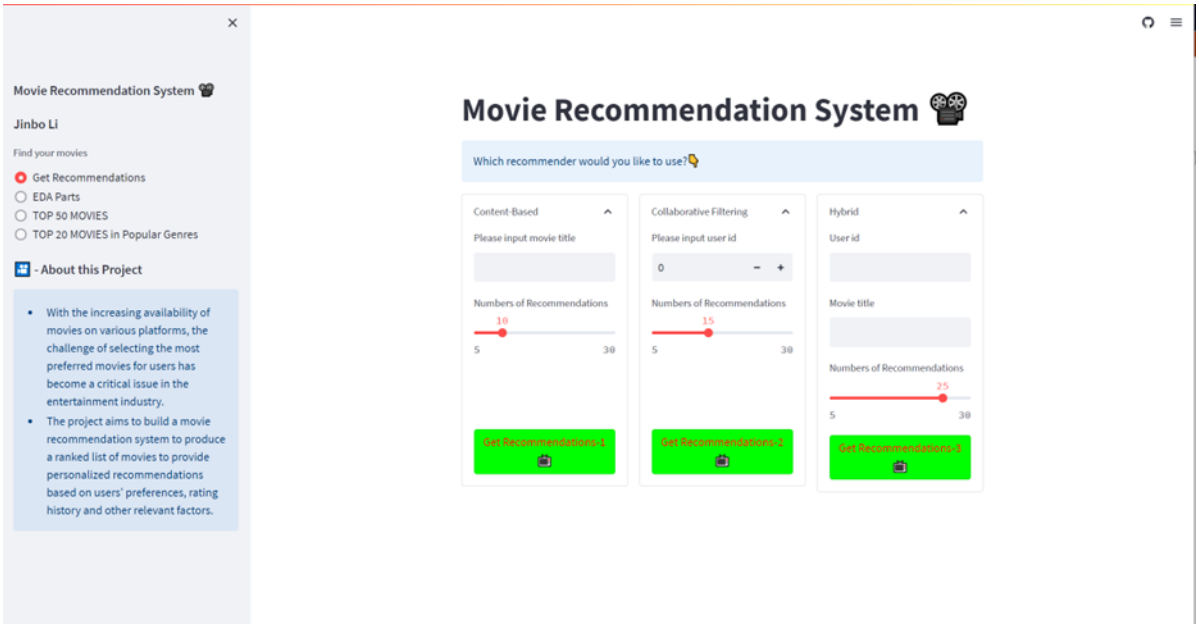

```
user: 1          item: 3671       r_ui = 3.00   est = 2.97   {'was_impossible': False}
```

As we can notice, the performance of SVD model is better than the previous MLP model. It has more accurate predictions. While various attempts have been made to improve the SVD model through parameter tuning, there is still potential for further enhancement. Although the current model shows promising results, there may be additional optimization opportunities that could lead to even better performance. Therefore, continued exploration and refinement of the model's architecture and parameters may yield further improvements. And I would like to do that works in the future study.

3.3.4 Hybrid Filtering and Interactive Web page

After developing multiple models for recommendation systems, I merged them into a hybrid model to obtain the most accurate and effective results. The hybrid model not only alleviates the cold start problem but also integrates the comprehensive movie information into the recommendations.

To further enhance the user experience, I designed and created an interactive web page for my recommendation system. (the link to the web page). The interactive web page allows users to input their preferences and receive personalized recommendations based on their individual taste, interests, and rating history. The interface is user-friendly and intuitive, making it easy for users to navigate and explore new movies.



Overall, the hybrid model and the interactive web page work together to provide users with a seamless and engaging experience. The hybrid model offers customized and personalized recommendations while the interactive web page makes the process of discovering new movies enjoyable and effortless.

Chapter 4: Conclusion

In this project, I developed three different types of recommenders using collaborative filtering and content-based filtering techniques. Furthermore, I built an interactive web page that provides personalized movie recommendations based on user preferences and feedback.

Although there has been some prior work on movie recommendation system, a common disadvantage is that they mainly rely on a single filtering technique and often suffer from the cold start problem and limited accuracy.

This project addressed these challenges by combining matrix factorization and content-based filtering, leveraging both user historical behavior and movie attributes to improve the recommendation accuracy and mitigate the cold start problem. The resulting hybrid model offered a more comprehensive and effective approach for a diverse range of users.

Additionally, the interactive web page added value to the system by providing a user-friendly and engaging interface for exploring new movie options. Users could input their preferences, receive personalized recommendations, and provide feedback, enhancing their experience and increasing the system's

accuracy over time.

Overall, this project offers a novel and practical approach to movie recommendation systems by combining multiple techniques and leveraging an interactive web page. The resulting system can provide customized and personalized recommendations to a wide range of users, finally enhancing their movie-watching experience.

Chapter 5: Reference

1. S. Agrawal and P. Jain, "An improved approach for movie recommendation system," 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2017, pp. 336-342, doi: 10.1109/I-SMAC.2017.8058367.

2. C. Birtolo, D. Ronca, R. Armenise and M. Ascione, "Personalized suggestions by means of Collaborative Filtering: A comparison of two different model-based techniques," 2011 Third World Congress on Nature and Biologically Inspired Computing, Salamanca, Spain, 2011, pp. 444-450, doi: 10.1109/NaBIC.2011.6089628.