INFO6105: Mid-Term Project

Gaurav Rai
College of Engineering
Northeastern University
Toronto, ON
rai.ga@northeastern.edu

Abstract

In this report, I implemented a movie recommendation system which have become an essential part of the entertainment industry, facilitating personalized movie suggestions to users. This mini project involves the analytics and development of a movie recommendation system using the MovieLens dataset. The primary objective was to explore various recommendation techniques, analyze the results, and assess their performance. We have applied some hands-on on this dataset like Data Preprocessing and Data Visualization, Collaborative Filtering, Content Based Filtering and discovered and discussed insights.

Data Preprocessing & Data Visualization: The dataset was cleaned and analyzed to gain insights into user ratings, genres, and show trends. Data Visualizations were used to understand the distribution of movie ratings and user interactions.

Collaborative Filtering: I used this model for recommendation, based on user-item interactions and it was implemented using the Surprise library. The system made personalized movie recommendations by assessing user preferences and their behavior. We used the Surprise library, which is specifically designed for building recommendation systems and gives good results.

Content-Based Filtering: I used this recommendation system for leveraging movie genres as features. I used cosine similarity to suggest movies based on their genres features which helps recommend movies that are most like a given movie title. I also defined a function "recommend" recommending movies based on cosine similarity.

Algorithm Comparison: I used various recommendation algorithms, including collaborative filtering, content-based filtering, and hybrid approaches to compare and evaluate the algorithm based on metrics like RMSE and accuracy.

This project will conclude that collaborative filtering and content-based filtering techniques can provide valuable movie recommendations. The choice of the technique depends on the specific context and data available.

1 Datasets

1.1 Why are these interesting datasets?

I used these two datasets "movies.csv" and "ratings.csv" from MovieLens which are commonly used in recommendation system research and development purposes. The "movies.csv" dataset contains rich movie metadata, including movie titles, genres, and unique identifiers (movieId). The "ratings.csv" dataset contains user interactions with movies, specifically user ratings. These ratings reflect how users interact with movies, and we can this user feedback for collaborative filtering and other recommendation techniques.

	Data Set Characteristics	Attribute Characteristics	Associated Tasks	Number of Instances	Number of Attributes
Dataset 1	Multivariate	Real	Recommendation & Analysis	9742	3
Dataset 2	Multivariate	Real	Recommendation & Analysis	100836	4

Table 1: Basic feature of both datasets.

1.2 Data Visualization

(a): The bar plot in Figure1 visualizes the distribution of movie genres and extracts and counts the occurrences of genres and displays the top 10 genres by count.

Insights: As we can see the “drama” movie genre tops the chart which seems users like the “drama” movie genre more as compared to the other genres.

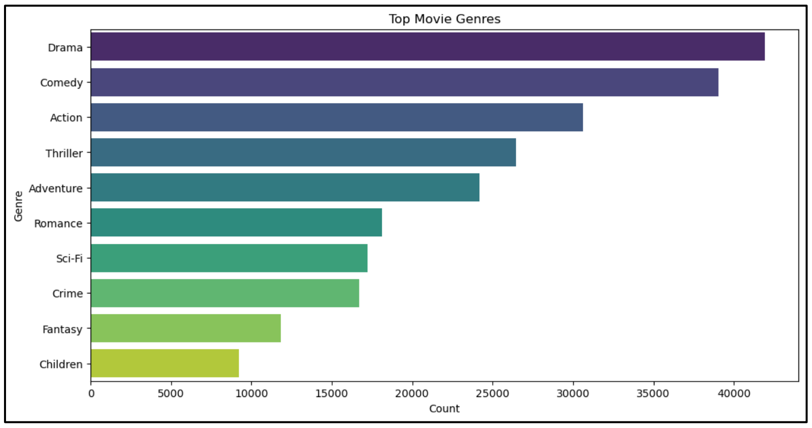


Figure 1: Distribution of Movie Genres

(b): The bar plot in Figure2 shows the distribution of user ratings and counts the number of ratings for each rating value starting from 1 to 5.

Insights: As we can see the popularity of different rating values and understand how users rate movies.

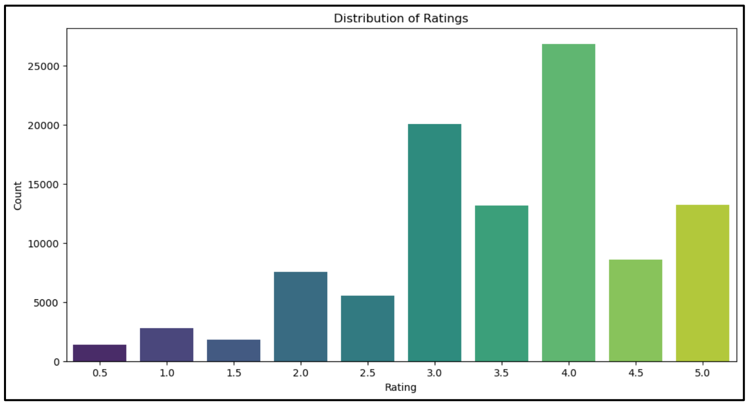


Figure 2: Distribution of User Ratings

(c): The bar plot in Figure3 displays the top-rated movies based on the average user ratings.

Insights: These are the top-rated movies based on the user ratings.

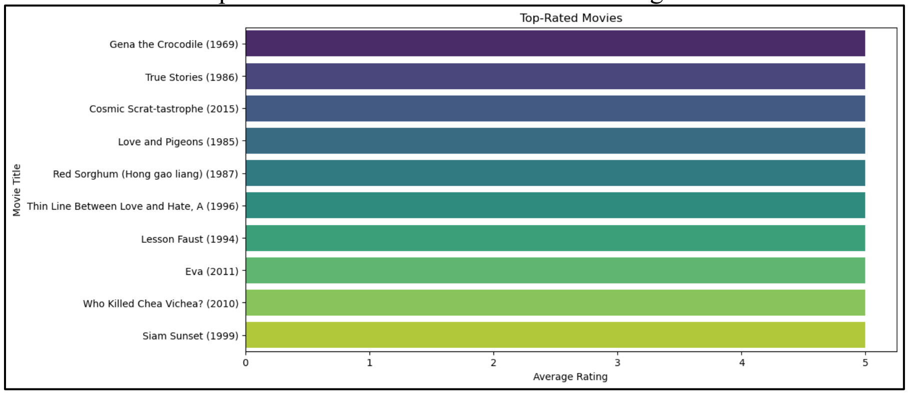


Figure 3: Top-Rated Movies

(d): The heatmap in Figure4 displays how users have rated movies across different genres.

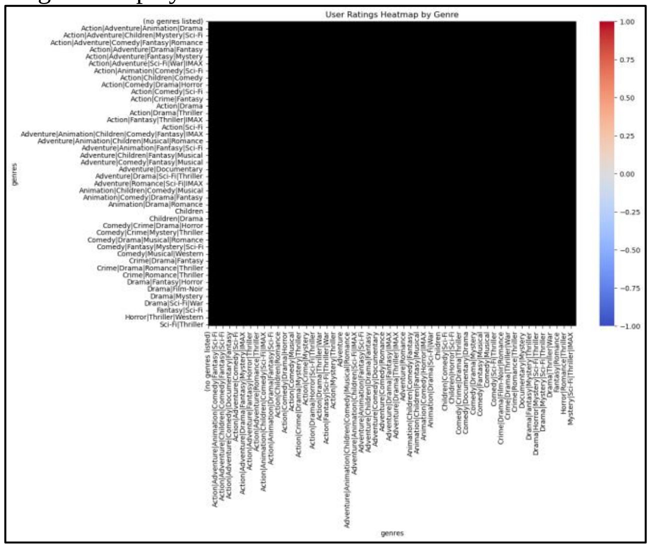
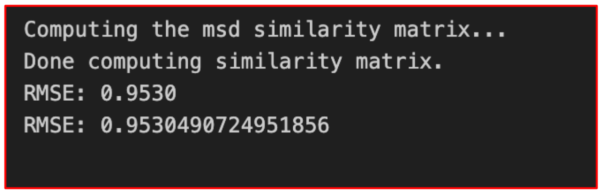


Figure 4: Ratings Heatmap

1.3 Data Processing

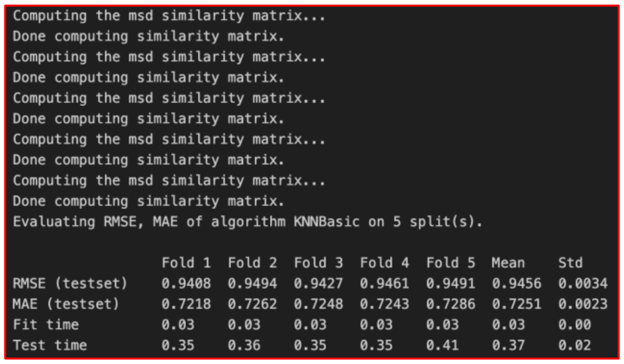
Data Splitting: I started with splitting the dataset into training and testing sets for building and evaluating the recommendation model.
Model Building: I started with building a recommendation model and use collaborative filtering models like KNN.
Model Evaluation: I started with evaluating the model's performance using metrics like RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) to make sure to test different algorithms and hyperparameters.



```
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9530
RMSE: 0.9530490724951856
```

Figure 5: Computing accuracy which is around 95%

Recommendation Generator: I generated movie recommendations for users based on the trained model and implemented user-specific recommendation strategies.
Analysis and Insights: Later I analyze the results of our recommendation system and check if it's providing accurate recommendations, if the model is overfitting, and if there are any notable trends in the recommendations.



```
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Evaluating RMSE, MAE of algorithm KNNBasic on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9408  0.9494  0.9427  0.9461  0.9491  0.9456  0.0034
MAE (testset)     0.7218  0.7262  0.7248  0.7243  0.7286  0.7251  0.0023
Fit time          0.03    0.03    0.03    0.03    0.03    0.03    0.00
Test time         0.35    0.36    0.35    0.35    0.41    0.37    0.02
```

Figure 6: Computing accuracy which measures the RMSE and MAE

1.4 Data Analysis

I started with data preparation and mentioned the data preprocessing steps, including loading and merging datasets. Later, I described how missing data and duplicates were handled.

Insights: Key insights derived from the analysis:

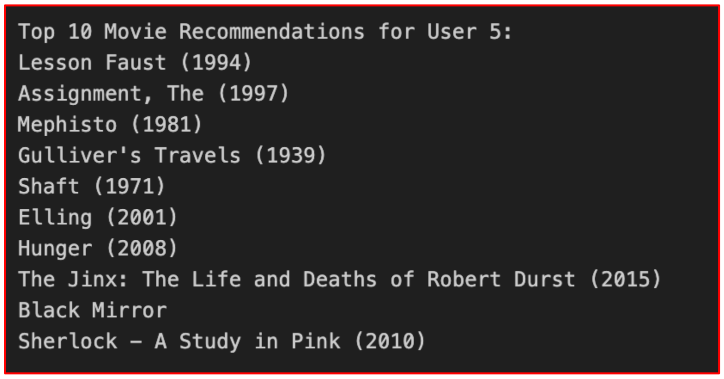
- Users tend to give higher ratings on average, with 4 being the most and 3.5- and 5-star ratings respectively being the most common.
- Drama, Comedy and Action are the top genres which are liked by the user.
- The top-rated movies often include classics and critically acclaimed films.
- Users show varying preferences for different movie genres, with some genres positively correlated and others negatively correlated in terms of user ratings.

2 Movie Recommendation Model

2.1 Collaborative Filtering Model

I chose collaborative filtering model using the Surprise library which trains it on the training data and evaluates it on the test data using RMSE(Root Mean Squared Error). I used KNN as the collaborative filtering method for making recommendations. The algorithm is initialized with parameters that determine the similarity measure. The Dataset module is used to load the movie ratings dataset into the Surprise format, making it compatible with various recommendation algorithms and the Reader module is used to specify the rating scale of the dataset, where ratings typically range from 1 to 5.

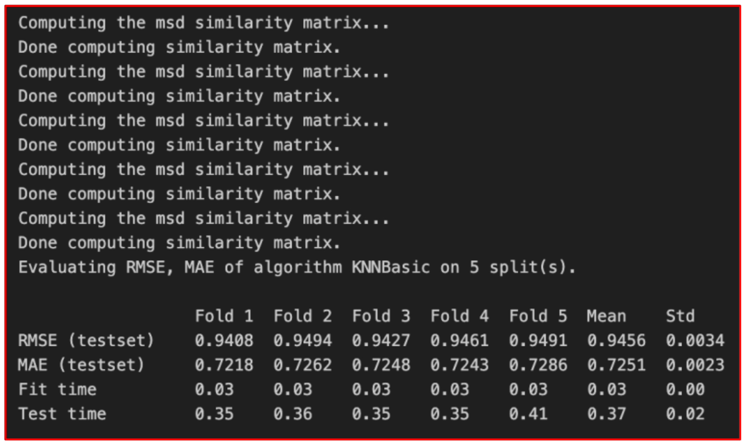
Later, I created a recommendation for specific user that will recommend movies based on the user's preferences. We can put any user id and accordingly that user will get top 10 movie recommendation list.



```
Top 10 Movie Recommendations for User 5:
Lesson Faust (1994)
Assignment, The (1997)
Mephisto (1981)
Gulliver's Travels (1939)
Shaft (1971)
Elling (2001)
Hunger (2008)
The Jinx: The Life and Deaths of Robert Durst (2015)
Black Mirror
Sherlock - A Study in Pink (2010)
```

Figure 7: Movie Recommendation List for User 5.

In addition, I performed cross-validation to evaluate the model's performance more accurately. This helps prevent overfitting and gives you a better estimate of how well the model will generalize to new users and items.



```
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Evaluating RMSE, MAE of algorithm KNNBasic on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9408  0.9494  0.9427  0.9461  0.9491  0.9456  0.0034
MAE (testset)     0.7218  0.7262  0.7248  0.7243  0.7286  0.7251  0.0023
Fit time          0.03    0.03    0.03    0.03    0.03    0.03    0.00
Test time         0.35    0.36    0.35    0.35    0.41    0.37    0.02
```

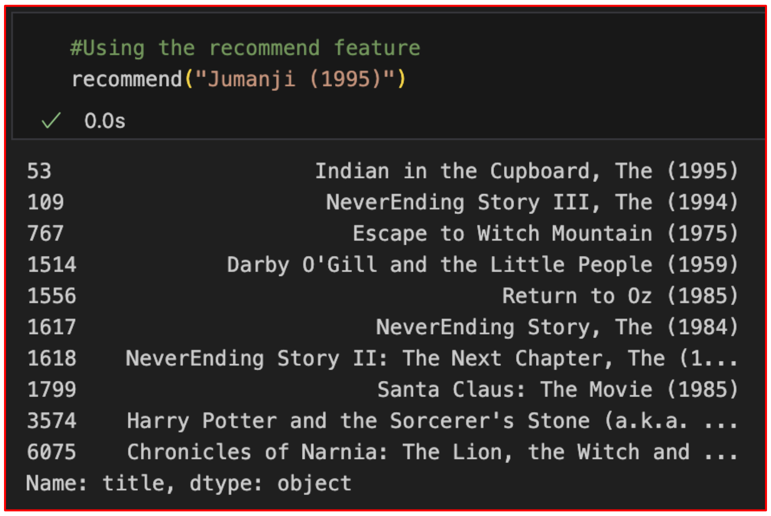
Figure 8: Model Performance.

2.2 Content Based Model

I chose to build a movie recommendation system using cosine similarity, we'll create a content-based recommendation system. In this approach, we'll use the movie's features (e.g., movie genres) to calculate the cosine similarity between movies and recommend movies based on their similarity to user preferences.

I used TFIDF Vectorizer for movie genres to represent movies as numerical vectors. These vectors capture the importance of different genres for each movie. I also used Linear Kernel to compute the cosine similarity between the TFIDF vectors representing movie genres. Movies with higher cosine similarity are considered more similar in terms of genre and recommended to users who have shown an interest in a particular movie.

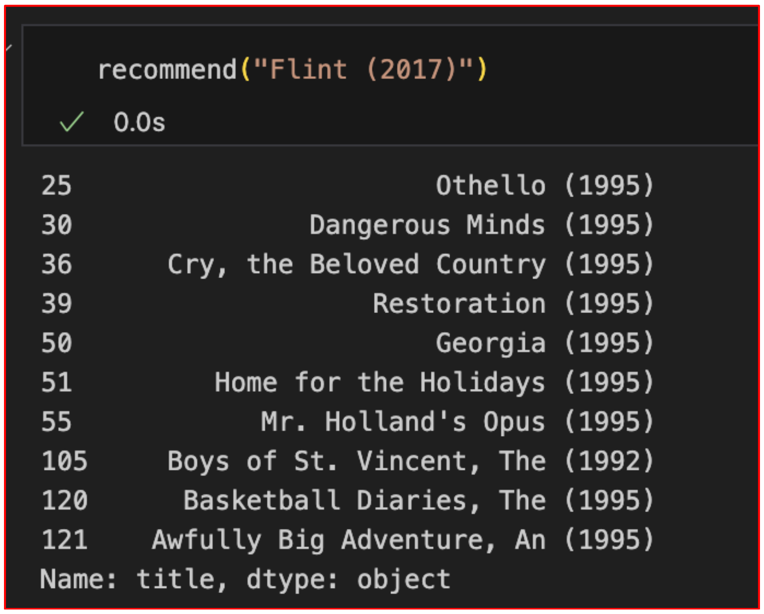
Later, I defined a function "recommend" recommending movies based on cosine similarity.



```
#Using the recommend feature
recommend("Jumanji (1995)")
✓ 0.0s

53                        Indian in the Cupboard, The (1995)
109                        NeverEnding Story III, The (1994)
767                          Escape to Witch Mountain (1975)
1514               Darby O'Gill and the Little People (1959)
1556                                     Return to Oz (1985)
1617                            NeverEnding Story, The (1984)
1618     NeverEnding Story II: The Next Chapter, The (1...
1799                             Santa Claus: The Movie (1985)
3574     Harry Potter and the Sorcerer's Stone (a.k.a. ...
6075     Chronicles of Narnia: The Lion, the Witch and ...
Name: title, dtype: object
```

Figure 9: Top 10 Movie Recommendation List for movie "Jumanji (1995)"



```
recommend("Flint (2017)")
✓ 0.0s

25                      Othello (1995)
30              Dangerous Minds (1995)
36       Cry, the Beloved Country (1995)
39                  Restoration (1995)
50                      Georgia (1995)
51         Home for the Holidays (1995)
55              Mr. Holland's Opus (1995)
105      Boys of St. Vincent, The (1992)
120       Basketball Diaries, The (1995)
121     Awfully Big Adventure, An (1995)
Name: title, dtype: object
```

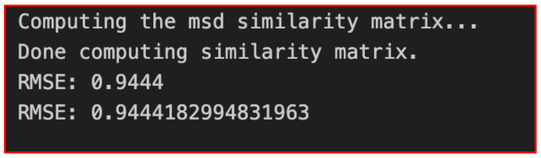
Figure 10: Top 10 Movie Recommendation List for movie "Flint (2017)"

This will calculate cosine similarity between movies based on their genre features and recommends movies that are most like a given movie title.

3 Accuracy Analysis

3.1 Porter Stemming

To build a movie recommendation system using Porter Stemming, I started with preprocessing the text data in the movie titles to apply stemming. Porter Stemming is a text normalization technique that reduces words to their root form. While this technique is more commonly used for natural language processing tasks, I was curious and wanted to try and analyze it to movie titles for a basic form of text analysis and just check the accuracy of the model.



```
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9444
RMSE: 0.9444182994831963
```

Figure 9: Model accuracy around 94%.

The key modification in this code is the use of the Porter Stemmer from the nltk library to stem the words in movie titles. This stemmer reduces words to their root form, allowing the model to generalize better for similar movie titles.

5 Conclusions

In this report, I have analyzed and implemented a movie recommendation system using the MovieLens dataset. The project's primary aim was to explore various recommendation techniques, assess their performance, and provide valuable insights into the world of recommendation systems.

Data Exploration and Preprocessing: We deep dived into the MovieLens dataset, gaining a deeper understanding of user ratings, movie genres, and temporal trends. We used visualizations to uncover the distribution of movie genres and user interactions.

Collaborative Filtering and Content-Based Filtering: We implemented both collaborative filtering and content-based filtering techniques. Collaborative filtering capitalized on user-item interactions to provide personalized movie recommendations, while content-based filtering utilized movie genres to suggest similar films.

Algorithm Comparison: We compared various recommendation algorithms, including collaborative filtering, content-based filtering, and hybrid approaches. Evaluation metrics, such as RMSE and accuracy, were employed to assess the performance.

I understood and discovered so many things and the project findings revealed that different recommendation techniques have their own strengths and limitations.

In conclusion, recommendation systems have a huge impact on user engagement, making them a pivotal aspect of the entertainment industry. This project provides a foundation for continued exploration and innovation in the realm of movie recommendations, with the potential to enhance the movie experiences of users across the globe.

Acknowledgments

The learning code is adapted from movie lens dataset: https://grouplens.org/datasets/movielens/ and references therein.

References

[1] MovieLens Dataset: https://grouplens.org/datasets/movielens/

[2] Surprise Library Documentation: https://surprise.readthedocs.io/en/stable/

[3] YouTube videos, Stack Overflow and Google