Assignment

Aman Verma
22B3929

Task 1: Create a Recommendation system

- Parse movies.csv and create a content based recommendation system using TfidfVectorizer and cosine_similarity from sklearn

In the movies.csv there are three columns movieID title and genre . In my implementation I have used TfiDfvectorizer and cosine_similarity between these vectors and then recommend the user movies which are similar to the ones that has been asked ,Then I made a new column named text and combined the title and the genres and dropped the original ones then these are represented in the form of a Vector using the TfIDFVectorizer and then traverses the csv file and find the place where this name is present and then it sorts them in the order of similarity scores sorted ones and then returns the top 10 of them

- Parse ratings.csv and create a collaborative filtering based recommendation system using Surprise or Librecommender

Next, the ratings file has userID, movieID,rating and timestamp are traversed and then the datasets are divided into train and test and the Singular Value Decomposition is fit on the training dataset and then the predictions are generated and the mae and rmse is compared with the ground truth and then for a given user id the movie ratings are searched and the algorithm works for a particular user id and movie id is estimated from the list of user movies and all movies and then the predictions are made for the movies that the user has not seen and what the user would definitely be sorted and then the top 10 movies are listed

Task 2: Prediction

- Generate 10 recommendations using:
- content-based system using the user's historical movie preferences.
- collaborative filtering system using user's ratings

1. For Content-based filtering

```
# Generating 10 movies based on Content for a particular movie name
movie_name = 'Avengers'
recommendations = recommend_movies(movie_name)
print(recommendations)
```

Output is

```
  movieId                                          text
17067    89745  Avengers, The (2012) Action|Adventure|Sci-Fi|IMAX
```

```
30431   136257     Avengers Grimm (2015) Action|Adventure|Fantasy
34536   145676               3 Avengers (1964) (no genres listed)
40636   159920                    Shaolin Avengers (1994) Action
25067   122912  Avengers: Infinity War - Part I (2018) Action|...
40637   159922                The Shaolin Avengers (1976) Action
35372   147657                     Masked Avengers (1981) Action
25068   122914  Avengers: Infinity War - Part II (2019) Action...
45394   170297  Ultimate Avengers 2 (2006) Action|Animation|Sc...

54283   189217  Avengers Grimm: Time Wars (2018) Action|Advent…
```

For Collaborative Filtering part

```
# Example: Recommend movies for a specific user (e.g., userId 1)

user_id = '1'

user_movies = set(data.raw_ratings[i][1] for i in
range(len(data.raw_ratings)) if data.raw_ratings[i][0] == user_id)

all_movies = set(data.raw_ratings[i][1] for i in
range(len(data.raw_ratings)))

movies_to_predict = list(all_movies - user_movies)

user_predictions = []

for movie_id in movies_to_predict:

    user_predictions.append((movie_id, algo.predict(user_id,
movie_id).est))



# Sort the predictions by estimated rating in descending order

user_predictions.sort(key=lambda x: x[1], reverse=True)

# Print the top recommended movies

top_n = 10

for i, (movie_id, rating) in enumerate(user_predictions[:top_n], 1):
```

```
    print(f"{i}. Movie ID: {movie_id}, Estimated Rating: {rating}")
```

Recommendations for the userID=’1’;

1. Movie ID: 290, Estimated Rating: 4.694044356237093
2. Movie ID: 86377, Estimated Rating: 4.652119741344023
3. Movie ID: 2858, Estimated Rating: 4.610286024957355
4. Movie ID: 1354, Estimated Rating: 4.573060728222595
5. Movie ID: 7767, Estimated Rating: 4.550262772613847
6. Movie ID: 134853, Estimated Rating: 4.547046126183418
7. Movie ID: 116897, Estimated Rating: 4.539214489395047
8. Movie ID: 112552, Estimated Rating: 4.530725849345432
9. Movie ID: 171011, Estimated Rating: 4.530433058924414
10. Movie ID: 86345, Estimated Rating: 4.498310125656087

Task 3: Evaluation

- Provide Metrics to judge recommendation systems

The content based recommenders can be evaluated on the basis of the f1score , Recall and Precision based on what the user expected when he typed the name and what the user gets can be compared with some ground truth . For the collaborative filtering the MAE and RMSE is a good way to judge how good the model is as its more of a regression problem

- Compare and contrast the the two systems

To Actually compare the Content based vs Collaborative we need to have a common differentiating metric and a common ground truth here the content based is generating the movies that are similar to the ones that the user entered and the Collaborative one tells what can be the top 10 movies that a particular person likes So a common metric can be to know for the movies that the user rated high for a given keyword is the movie in the list given by the content based model or not

For Collaborative Model

MAE: 0.5870

RMSE: 0.7777

Mean Absolute Error (MAE): 0.587036732063796

Root Mean Squared Error (RMSE): 0.7776794685206999