Team:

Sravya Sangaraju (Miner2 Username: Sravv, Mason UserID: ssangara)

Anisha Yidala (Miner2 Username: 121212, Mason UserID: ayidala)

Best public score (RMSE): 0.90, Rank: 156

HW 4: RECOMMENDER SYSTEMS

Problem:

To develop a Recommender System that uses the rating information matrix and side information about the given data. We are given with the train and test datasets which contain details about the users, movies and the rating that is given to a movie by a specific user. We are also given additional files with contains metadata regarding the movies for generating a profile for a user. The motive of this project is to predict a rating on the scale of 5 that the user gives to a specific movie.

Data Preprocessing:

For preprocessing the data, we first imported all the required libraries such as pandas, numpy, scipy, sklearn etc. and loaded the dataset using the basic file reading functions in python. Then, we created a user-movie sparse matrix from the training set where we used UserIDs as the rows, MovieIDs as the columns and rating given by the user to the movie as user_movie_matrix[x][y]. In a similar way, we created another matrix which has MovieIDs as rows and Movie Genres as the columns. For this matrix, we assigned the value as 1 if a given movie is of the genre given by the column, else we assigned it to 0.

user_movie_matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
75	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.5
78	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.5	0.0	0.0	5.0
127	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
170	3.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0
175	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
71487	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.5	0.0	0.0	0.0	0.0	0.0	0.0	0.5	4.5
71497	0.0	3.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	2.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
71509	4.0	0.0	0.0	0.0	1.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.5	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
71525	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
71529	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

2112 rows × 9936 columns

```
[15] movie_genre_matrix = MovieGenre(movie_genres_data)
     movie_genre_matrix
```

Movie_ID	Children	Western	Documentary	Film-Noir	Crime	Comedy	Mystery	Horror	Action	IMAX	Thriller	Short	Drama	Animation	Fantasy	M
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	1	0	0	0	0	0	0	0	1	1	
3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
5	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
65088	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
65091	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
65126	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	
65130	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	
65133	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	

10197 rows × 20 columns

Methodologies:

- For calculating the similarity between a given user and all the other users in the given training date, we used an approach called User Based Collaborative Filtering where we select only those users who are comparable to the subject user. Thus, we would group all the users who had rated a particular movie and then figure out the most similar user. To calculate the similarity between the users, we used measures such as Cosine Similarity, Euclidean Distance and Pearson Correlation. We used the SciPy library for calculating the Pearson Correlation from scipy.stats.pearsonr. Once the similarity is calculated, we take the average of the ratings given by the K users.

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

Where,

r = Pearson Correlation Coefficient

x_i = x variable samples $\quad$ y_i = y variable sample

$\bar{x}$ = mean of values in x variable $\quad$ $\bar{y}$ =mean of values in y variable

- For handling cold start users, i.e. the users that did not rate any movie yet, we used the movie genre corpus and converted into a TD-IDF vector with the help of

sklearn.feature_extraction. Once the genres have been converted into the TD-IDF vectors, we calculated the cosine distances between every genre and all the other genres in the given data. Again, we selected k nearest values and computed the average of these values.

- Similarly, for dealing with cold start movies, i.e. the movies that has not been rated yet by any user, we took the genre and compared with all the other genres that a particular user has rated. Then we calculated the cosine distance after converting then into TD-IDF vectors. Later, we chose K nearest neighbors and calculated the average of ratings all the movies under that genre.

- Other data given in the additional files where the tags and tag weights. We used them with the movie genres of the top 3 tag weights of the movie, but it was not helping in reducing the root mean squared error.

- In the next approach, we used the movie genres and the average rating that a user has given. We also assumed the values of the ratings as 2.5 for unseen movies in no movies in that genre has a rating.

Output :

After implementing all the above-mentioned approaches, the lowest root mean squared error that we achieved was 0.9 for the last approach.

References:

https://medium.com/sfu-cspmp/recommendation-systems-user-based-collaborative-filtering-using-n-nearest-neighbors-bf7361dc24e0
https://towardsdatascience.com/machine-learning-for-building-recommender-system-in-python-9e4922dd7e97
https://scikit-learn.org/stable/modules/classes.html
https://medium.com/yusp/the-cold-start-problem-for-recommender-systems-89a76505a7
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mstats.pearsonr.html