Recommendation system

Approach:

Step 1: Read data from the train.dat file.
Step 2: Create a utility matrix with rows as the User id and columns as the Movie id.
Step 3: Find similarity between the users based on their ratings for the movies that they have watched.
Step 4: Read data from the test data file.
Step 5: Call the get_ratings() function and pass the data from the test data file. It returns the predicted ratings for each user movie pair.
Step 6: Call the get_Nan() function. It returns the user movie pair for which the rating could not be predicted.
Step 7: Read any additional file from the additional data files to predict the rating of the non-predicted pairs based on the user's preference/choice of movies.
Step 8: Call the get_preference_ratings() function. It predicts the remaining user movie pair ratings.

Methodology of choosing the approach:

The entire process is divided into two parts-

Firstly, we implement the collaborative filtering method and then on the remaining test records, we implement the content-based recommendation method. We do this because some movies are not watched by any of the nearest users and hence, we cannot predict the rating using collaborative method.

1. Using collaborative filtering based on user-user similarity:
   For collaborative filtering, the utility matrix is created with the rows as users and columns as movies.
   We find the similarity between the users using this utility matrix
   by calculating the cosine similarity matrix.
   It gives the similarity value between each user and the remaining users based on the ratings given by them for the movies that they have watched.
   Using this similarity measure, we predict the rating that a user might give for a movie.

2. Using content-based recommendation based on user preference:
   For this part, we predict the rating by a user for a movie using the additional information given in the files such as the movie_actors, movie_directors, movie_genres etc.
   To predict the rating for a movie, we find the movies that the user has watched having common set of actors/ common genres/ common directors etc.
   We predict rating for the desired movie based on the ratings given by the user to the most similar movies.

Content features you used if any and what worked, what did not:

Solutions tried:

1. Calculating the predictions based on the mean of all the ratings given to the movie.
   The RSME error of the predicted ratings is greater as the ratings are calculated based on the ratings of other users irrespective to their similarity with the desired user. This is a naïve approach to predict the ratings.

2. Calculating the predictions based on mean of ratings of the users that have rated movies that are in common with the movies rated by the desired user.
   The RSME error of the predicted ratings is comparatively lower. We take into consideration only the ratings of the users who have rated common to the desired user, hence having more similar choices of movies as the user.

3. Calculating the predictions based on weighted mean of ratings of the users that have rated movies that are in common with the movies rated by the desired user.
   This solution uses the same logic as the 2nd solution. The only difference is that it calculates the weighted average of the ratings.
   So, the formula is weighted average = Summation of predicted rating for each of the k nearest users [(rating*similarity)/sum of the similarities of k users].
   Thus, more the similarity measure with the user, greater weight of the rating.

4. Calculating the predictions based on mean of ratings of the desired user given to movies having common actors.
   We find out the movie actors of the movie for which the rating is to be predicted. Then, we find out all the movies that the user has watched. After vectorizing all the data, we use the cosine similarity to find which movies have the common actors as the movie to be rated.
   Then we calculate the mean of the ratings of those set of movies with common actors.

5. Calculating the predictions based on mean of ratings of the desired user given to movies having common directors.
   We find out the director of the movie for which the rating is to be predicted. Then, we find out all the movies that the user has watched. After vectorizing all the data, we use the cosine similarity to find which movies have the common director as the movie to be rated.
   Then we calculate the mean of the ratings of those set of movies with common director.

6. Calculating the predictions based on mean of ratings of the desired user given to movies having common genres.
   We find out the genre of the movie for which the rating is to be predicted. Then, we find out all the movies that the user has watched. After vectorizing all the data, we use the cosine similarity to find which movies have the common genre as the movie to be rated.
   Then we calculate the mean of the ratings of those set of movies with common genre.

   Note: A generic code snippet has been programmed to use different features of the movies such as actors, directors, genre etc to predict the rating.
   The only change that needs to be made is the filename that is being used and the name of the columns that the file has

Libraries used (Main libraries):

from sklearn.metrics.pairwise import cosine_similarity:

This library is used to find the similarity matrix between the users.
The highest value is 1, along the diagonal of the matrix, which means the similarity is calculated of the user with itself.
The values range from 0 to 1. Greater the value, more the similarity between the users.

Therefore, we find the k highest values of users which gives us the k nearest users.

from sklearn.feature_extraction.text import CountVectorizer/TfidfVectorizer:

This library is used to vectorize the movie feature data such as actor's data for example.
It converts the data of actors in the test movie and the data of actors in the movies that the user has rated into vectors using the CountVectorizer/TfidfVectorizer. It finds the common set of words (actors names) and converts into vectors.
We use cosine similarity to find the nearest vectors/records for which there are common set of actors.

Algorithm of the recommendation system: (Hybrid method)

1. Read the user data from the train file which consist of User_id (u), Movie_id (m) and Rating.
2. Using the pivot function, construct a matrix with User_id as the rows and Movie_id as the columns. The intersection cell has the value of the rating for the jth Movie_id by the ith User_id.
3. Movie_id's not rated by the Users have Nan values in the cells.
4. Replace the Nan values by 0.0
5. Construct a similarity matrix DataFrame with rows and columns as the User_id and the intersection cells have the similarity values.
6. Read the test data file for the User-Movie pairs for which the rating must be predicted.
7. For each pair, find the k similar users to 'u' from the similarity matrix.
8. Create a subset of the k users who have rated the movie 'm'.
9. Calculate the mean of the ratings.
10. Update the rating value for each user-movie pair.
11. Issue – The movies which have not been rated by any of the k similar user have the rating Nan in the predictions list.
12. Find the records having Nan as the rating.
13. Read any movie feature file using which you would want to make the predictions.
14. For every record having Nan as the rating, find the movie feature value.
15. Find the movies that the user 'u' has rated and has similar movie feature values as 'm'.
16. Vectorize both the data record/records in line 14 and 15.
17. Find the k1 (for prediction we have used k1=5) nearest vectors using the cosine similarity function and calculate the mean of the ratings of k1 records.
18. Assign the rating to the specific user-movie pair.
19. Repeat the procedure from 14-18 for every record having Nan as the rating value.
20. Thus, all the rating predictions are made.

Parameter choices:

- For collaborative filtering, we have used k = 200 i.e. 200 users with similar ratings and movies watched.
- For content-based recommendation, we have used k = 5 i.e. mean of 5 movie ratings which have the same feature values as the movie for which the prediction is to be made.
- We have used the TfidfVectorizer for vectorizing the feature data.

Runtime:

The collaborative part takes about 7-8 minutes for predicting the ratings.
Further, the number of records for which we need content-based method depends on the value of k we choose in the collaborative method.
Higher the value of k, lesser the number of records with Nan ratings.
The running time of content-based method depends highly on the selected values of k.
The running time is high for a greater number of records with Nan ratings. It takes about 15 mins for 556 records. (k=200)

Time complexity for collaborative method: O(n) (single for loop)
Time complexity for content-based method: O(n^2) (nested for loop)

Observation/Conclusion/Analysis:

1. All the solutions for different movie features after implementation, yield the same RSME score.
2. The RSME score for ratings calculated using unrelated user ratings is greater than the ratings calculated using the similar users.
3. If the value of k is too small, there are many movie ratings which cannot be predicted as the number of related users who have rated the movie is 0.
4. Also, due to small value of k, the chances that the related users have watched the movie decreases.
5. The content-based recommendation method takes more time as compared to the collaborative model as the prediction is made by taking into consideration the user's choice of movie and tailoring the features accordingly.
6. The weighted average gives more RSME than the mean for some combination of features and value of k.