Experiment No. 4
Implement User-based Nearest neighbor recommendation
Date of Performance:
Date of Submission:
Marks:
Sign:

Aim: Implement User-based Nearest neighbor recommendation

Objective: Ablility to perform User based nearest neighbor recommendation.

Theory: User-Based Collaborative Filtering is a technique used to predict the items that a user might like on the basis of ratings given to that item by other users who have similar taste with that of the target user. Many websites use collaborative filtering for building their recommendation system. Steps for User-Based Collaborative Filtering:

Step 1: Finding the similarity of users to the target user U. Similarity for any two users 'a' and 'b' can be calculated from the given

$$Sim(a,b) = \frac{\sum_p (r_{ap} - \bar{r}_a)(r_{ab} - \bar{r}_b)}{\sqrt{\sum (r_{ap} - \bar{r}_a)^2}\sqrt{\sum (r_{bp} - \bar{r}_b)^2}}$$

r_{up} : *rating of user u against item p*

p : *items*

Step 2: Prediction of missing rating of an item Now, the target user might be very similar to some users and may not be much similar to others. Hence, the ratings given to a particular item by the more similar users should be given more weightage than those given by less similar users and so on. This problem can be solved by using a weighted average approach. In this approach, you multiply the rating of each user with a similarity factor calculated using the above mention formula. The missing rating can be calculated as

$$r_{up} = \bar{r}_u + \frac{\sum_{i \in users} sim(u,i) * r_{ip}}{\sum_{i \in users} |sim(u,i)|}$$

Implementation:

```
import pandas as pd
import numpy as np
movies_df = pd.read_csv('movies.csv',usecols=['movieId','title'],\
                  dtype={'movieId': 'int32', 'title': 'str'})
rating_df=pd.read_csv('ratings.csv',usecols=['userId', 'movieId', 'rating'],
   dtype={'userId': 'int32', 'movieId': 'int32', 'rating': 'float32'})
movies_df.head()
rating_df.head()
df = pd.merge(rating_df,movies_df,on='movieId')
df.head()
combine_movie_rating = df.dropna(axis = 0, subset = ['title'])
movie_ratingCount = (combine_movie_rating.
```

```
    groupby(by = ['title'])['rating'].
    count().
    reset_index().
    rename(columns = {'rating': 'totalRatingCount'})
    [['title', 'totalRatingCount']]
   )
movie_ratingCount.head()
rating_with_totalRatingCount = combine_movie_rating.merge(movie_ratingCount,\
                                                         left_on = 'title',\
                                                         right_on = 'title', how = 'left')
rating_with_totalRatingCount.head()
pd.set_option('display.float_format', lambda x: '%.3f' % x)
print(movie_ratingCount['totalRatingCount'].describe())
popularity_threshold = 50
rating_popular_movie= rating_with_totalRatingCount.query('totalRatingCount >=
@popularity_threshold')
rating_popular_movie.head()
rating_popular_movie.shape
movie_features_df=rating_popular_movie.pivot_table(index='title',\
                                                  columns='userId',values='rating').fillna(0)
movie_features_df.head()
from scipy.sparse import csr_matrix
movie_features_df_matrix = csr_matrix(movie_features_df.values)
movie_features_df_matrix
from sklearn.neighbors import NearestNeighbors
model_knn = NearestNeighbors(metric = 'cosine', algorithm = 'brute')
model_knn.fit(movie_features_df_matrix)
movie_features_df.shape
query_index = np.random.choice(movie_features_df.shape[0])
print(query_index)
movie_features_df.iloc[query_index,:]
distances, indices =
model_knn.kneighbors(movie_features_df.iloc[query_index,:].values.reshape(1, -1),\
                                         n_neighbors = 6)
distances
indices
movie_features_df.head()
distances.flatten().shape
indices.flatten().shape
for i in range(0, len(distances.flatten())):
    if i == 0:
        print('Recommendations for {0}:\n'.format(movie_features_df.index[query_index]))
    else:
        print('{0}: {1}, with distance of {2}:'.format(i,
movie_features_df.index[indices.flatten()[i]],\
                                                       distances.flatten()[i]))
```

Output:

```
Recommendations for Star Trek (2009):

1: District 9 (2009), with distance of 0.3868381977081299:
2: Iron Man (2008), with distance of 0.4175134301185608:
3: Dark Knight, The (2008), with distance of 0.4531437158584595:
4: Zombieland (2009), with distance of 0.46887803077697754:
5: Guardians of the Galaxy (2014), with distance of 0.475075900554657:
```

Conclusion:

In conclusion, the user-based nearest neighbor recommendation system leverages similarities between users to offer personalized suggestions. By calculating similarity metrics like cosine similarity or Pearson correlation, it identifies neighbors with similar preferences. This collaborative filtering approach effectively captures user tastes and preferences, delivering tailored recommendations. Its reliance on user behavior fosters engagement and enhances the overall user experience in recommendation systems.