Personalized Data Analytics

Table of Contents

I. Recommendation Systems

You are one of the organizers a festival on a university campus with plenty of food and drinks. You are put in charge of ordering beers for the event, and you want to use a recommender system to make sure that you can better model the preferences of the students in different sections. For such reason, you meet a few students in different sections and ask them to rate the 4 beers for which you gathered information (in a scale from 1 to 5). Unfortunately, not all of them know the beers in question, therefore the rating table is incomplete. (Complete the TODOs in recommendation.ipynb)

Student from:	Desperados	Guinness	chimay triple	Leffe
ICT	4	3	2	3
Medicine	1	2	3	1
Business	?	2	1	?
Environment	4	3	?	?

- Use cosine similarity to compute the missing rating in this table using user-based collaborative filtering (CF).

```
def user_cf(M, metric='cosine'):
    pred = np.copy(M)
    n_users, n_items = M.shape
    avg_ratings = np.nanmean(M, axis=1)
    sim_users = sim_matrix(M, 'user', metric)
    for i in range(n_users):
        for j in range(n_items):
            if np.isnan(M[i,j]):
                pred[i,j] = avg_ratings[i] + np.nansum(sim_users[i] * (M[:,j] - avg_ratings)) / sum(sim_users[i])
    return pred
```

- Similarly, computing the missing rating using item-based CF.

```
def item_cf(M, metric='cosine'):
    pred = np.copy(M)
    n_users, n_items = M.shape
    avg_ratings = np.nanmean(M, axis=0)
    sim_items = sim_matrix(M, 'item', metric)
    for i in range(n_users):
        for j in range(n_items):
            if np.isnan(M[i,j]):
                pred[i,j] = avg_ratings[j] + np.nansum(sim_items[j] * (M[i,:] - avg_ratings)) / sum(sim_items[j])
    return pred
```

This is the rating ground truth for the above data:

Student from:	Desperados	Guinness	Chimay triple	Leffe
ICT	4	3	2	3
Medicine	1	2	3	1
Business	1	2	1	2
Environment	4	3	2	4

- Compute the predictive accuracy of the above recommendations

```
evaluateRS(M, M_result, 'user_cf', 'cosine')
evaluateRS(M, M_result, 'user_cf', 'correlation')
evaluateRS(M, M_result, 'item_cf', 'cosine')
evaluateRS(M, M_result, 'item_cf', 'correlation')
```

- Compute the ranking quality of the above recommendations

```
results = []
for method in ['user_cf', 'item_cf']:
    for metric in ['cosine', 'correlation']:
        rank_acc = evaluate_rank(M, M_result, method, metric)
        results += ["Rank accuracy of {0} with {1} metric: {2}".format(method[1], metric, rank_acc)]
print("\n".join(results))
```

II. Exercises (OPTIONAL)

Recommendation Systems (Advanced)

You are provided 3 csv files: movies.csv, users.csv and ratings.csv. Please use those datasets and complete the following challenges.

a. Content-Based Recommendation Model

- Find list of used genres which is used to category the movies.

```
print(listGen)
```

```
['Animation', "Children's", 'Comedy', 'Adventure', 'Fantasy', 'Romance', 'Drama', 'Action', 'Crime', 'Thriller', 'Horror',
'Sci-Fi', 'Documentary', 'War', 'Musical']
```

- Vectorize the relationship between movies and genres and put them into Ij.

```
print(Ij[:4])
```

```
[[1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]]
```

- Vectorize the relationship between users and genres and put them into Uj (if user rate for a movie, he/she has the related history with the movies'genres).

```
print(Uj[:4])
```

```
[[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0], [1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1]]
```

- Compute the cosine_similarity between movies and users. Hint: you can use sklearn.metrics.pairwise and cosine_similarity for quick calculation.

```
[[0.46291005 0.46291005 0.37796447 ... 0.37796447 0.26726124 0.37796447]
 [0.46291005 0.46291005 0.37796447 ... 0.37796447 0.26726124 0.37796447]
 [0.4472136  0.4472136  0.36514837 ... 0.36514837 0.25819889 0.36514837]
 ...
 [0.46291005 0.46291005 0.37796447 ... 0.37796447 0.26726124 0.37796447]
 [0.4472136  0.4472136  0.36514837 ... 0.36514837 0.25819889 0.36514837]
 [0.4472136  0.4472136  0.36514837 ... 0.36514837 0.25819889 0.36514837]]
```

b. Collaborative Filtering Recommendation Model by Users

- Use train_test_split to split above dataset with the ratio 50/50. The test dataset will be used as groundtruth to evaluate the rating calculated by using the train dataset
- Create matrix for users, movies and ratings in both training and testing datasets. Hint:

  train_data_matrix = train_data.pivot_table(index='user_id', columns='movie_id', values='rating').astype('float64')

  test_data_matrix = test_data.pivot_table(index='user_id', columns='movie_id', values='rating').astype('float64')

- Calculate the user correlation. Hint: you can reference help_function.txt for some necessary functions, but you can write the function by yourself. The similarity between item and itself should be 0 to not affect the result.

```
[[ 0.         -0.01578146 -0.20121784 ...  0.08171063 -0.29064092
   0.05356102]
 [-0.01578146  0.          0.0073552  ... -0.04626997  0.09664223
  -0.07852209]
 [-0.20121784  0.0073552   0.         ... -0.01127893  0.00718984
   0.2729944 ]
 ...
 [ 0.08171063 -0.04626997 -0.01127893 ...  0.         -0.26604897
   0.05947466]
 [-0.29064092  0.09664223  0.00718984 ... -0.26604897  0.
  -0.08159598]
 [ 0.05356102 -0.07852209  0.2729944  ...  0.05947466 -0.08159598
   0.        ]]
```

- Implement a predict based on user correlation coefficient.
- Predict on train dataset and compare the RMSE with the test dataset.

c. Collaborative Filtering Recommendation Model by Items.

- Calculate the item correlation
- Implement function to predict ratings based on Item Similarity.

```
[[ 0.         -0.17105086  0.04233412 ...  0.36847422  0.08410575
   0.00899673]
 [-0.17105086  0.         -0.31577814 ... -0.06670856 -0.45442053
   0.34242022]
 [ 0.04233412 -0.31577814  0.         ...  0.04466245 -0.07067555
  -0.57321736]
 ...
 [ 0.36847422 -0.06670856  0.04466245 ...  0.         -0.1191302
   0.34675131]
 [ 0.08410575 -0.45442053 -0.07067555 ... -0.1191302   0.
  -0.4095297 ]
 [ 0.00899673  0.34242022 -0.57321736 ...  0.34675131 -0.4095297
   0.        ]]
```

- Predict on train dataset and compare the RMSE with the test dataset.
- Compare the results between User-based and Item-based. Make conclusion.