Griffith UNIVERSITY

3803ICT
Big Data Analysis

Lab 05 – Predictive Data Analysis

Trimester 1 - 2019

Table of Contents

1. Basics of Recommendation Algorithm

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.

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

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
- ❖ Compute the ranking quality of the above recommendations

2. Movie Recommendation

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 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 Uj (if user rate for a movie, he/she has the related history with the movies'genres).

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

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

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

```
[[0.23570226 0.         0.57735027 ... 0.57735027 0.40824829 0.57735027]
 [0.23570226 0.         0.57735027 ... 0.57735027 0.40824829 0.57735027]
 [0.23570226 0.         0.57735027 ... 0.57735027 0.40824829 0.57735027]
 ...
 [0.23570226 0.         0.57735027 ... 0.57735027 0.40824829 0.57735027]
 [0.23570226 0.         0.57735027 ... 0.57735027 0.40824829 0.57735027]
 [0.23570226 0.         0.57735027 ... 0.57735027 0.40824829 0.57735027]]
```

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.

```
user_id   1    2    3    4    5    6    7    8    9    10   ...  91   92   \
movie_id                                                     ...
1         3.0  3.0  3.0  NaN  2.0  5.0  NaN  NaN  4.0  NaN  ...  1.0  NaN
2         NaN  NaN  1.0  NaN  4.0  NaN  NaN  NaN  NaN  NaN  ...  NaN  5.0
3         5.0  NaN  4.0  NaN  4.0  NaN  4.0  3.0  NaN  NaN  ...  NaN  5.0
4         NaN  NaN  NaN  NaN  4.0  NaN  NaN  NaN  5.0  4.0  ...  NaN  NaN
5         NaN  5.0  3.0  NaN  NaN  NaN  NaN  4.0  NaN  4.0  ...  1.0  5.0
6         3.0  NaN  5.0  NaN  NaN  3.0  3.0  NaN  5.0  4.0  ...  NaN  4.0
7         NaN  NaN  3.0  3.0  4.0  4.0  NaN  NaN  2.0  4.0  ...  NaN  NaN
8         NaN  NaN  NaN  3.0  NaN  NaN  NaN  NaN  2.0  NaN  ...  4.0  NaN
9         3.0  2.0  3.0  NaN  4.0  5.0  3.0  1.0  NaN  2.0  ...  4.0  NaN
10        2.0  4.0  NaN  5.0  NaN  3.0  NaN  4.0  NaN  NaN  ...  5.0  NaN
11        4.0  NaN  NaN  3.0  NaN  1.0  NaN  NaN  NaN  3.0  ...  4.0  NaN
12        4.0  NaN  NaN  NaN  NaN  3.0  NaN  NaN  NaN  NaN  ...  NaN  NaN
13        1.0  NaN  NaN  NaN  3.0  NaN  NaN  3.0  NaN  NaN  ...  NaN  NaN
14        NaN  2.0  NaN  NaN  3.0  3.0  NaN  NaN  NaN  NaN  ...  NaN  3.0
15        5.0  NaN  NaN  NaN  3.0  NaN  NaN  5.0  NaN  2.0  ...  NaN  NaN
16        NaN  NaN  4.0  2.0  4.0  NaN  5.0  NaN  2.0  NaN  ...  NaN  NaN
17        NaN  NaN  NaN  NaN  4.0  NaN  NaN  5.0  4.0  NaN  ...  4.0  NaN
18        4.0  4.0  NaN  2.0  NaN  2.0  2.0  NaN  4.0  5.0       NaN  NaN
```

- ❖ 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.

```python
# RMSE on the test data
print('User-based CF RMSE: ' + str(rmse(user_prediction, test_data_matrix.values)))
```

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.