# Collaborative filtering recommender system for predicting the movie ratings
# COMP9417 Machine Learning Project

## Introduction

Recommendation is a normal thing in this world. In some special areas, recommendation need to be implemented by machine, so a smart recommendation algorithm to achieve this is necessary. Take movie recommendation algorithm as an instance, system needs to recommend movies to users according to the information they leave on the system. In order to make the recommendation accurate, the first step is to obtain enough movies to analyze, then the second step is to have plenty of user behavior data. By analyzing and filtering the data, find a list of movies which user may like but haven't seen and send this list to the user. The filtering algorithm which is used in this project is collaborative filtering recommender algorithm.

## Implementation

This project is finished by Python 3.6. The dataset used in this project is collected by 'GroupLens Research Group'. The dataset contains 943 users, 1682 movies and 100000 ratings.

This project make recommendation based on the similarity of user preference by using Collaborative Filtering algorithm.

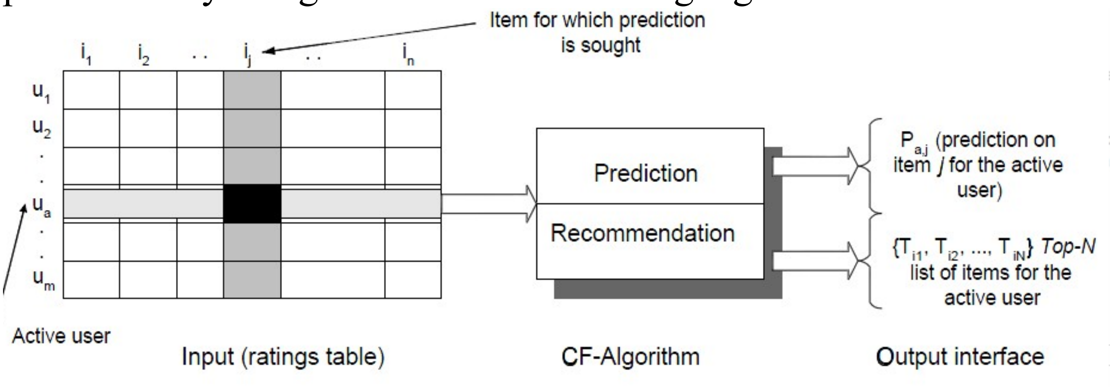



**Figure 1: The Collaborative Filtering Process.**

As shown in Figure 1, in collaborative Filtering Process, a user's preference for an item is represented by an m×n matrix. The row in the figure shows a user, the column shows an item, and Uij shows the score user i gives for item j. Collaborative Filtering Process is divided into two processes, prediction process and recommendation process. The prediction process is to predict the user's possible scores for items that have not been used. The recommendation

process is to recommend one or top-N items that the user is most likely to like based on the results of the prediction process.
Because the dataset in the project is not very big, the first important point is to make sure a good value of min_periods parameter. This parameter represents the minimum amount of movie ratings that a valid user need to give. This part will be talked specifically in Result section.
Next step is to calculate the correlation coefficient between each user. In this project, based on the relevance similarity calculation, using the Pearson correlation coefficient. The formula is as follow:

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R_i})(R_{u,j} - \bar{R_j})}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R_i})^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R_j})^2}}.$$

$R_{u,i}$ indicates the score user u gives to movie i and $\bar{R}i$ indicates the average score of the i-th movie.

Prediction: For each movie which user u have not seen, calculate the weighted sum of all ratings given by other user whose Pearson correlation coefficient with u is more than 0.1. The weights are Pearson correlation coefficient with u. Then, make all non-null elements sort in descending order and print it out.

**Results**

**1.Data regulation**

Firstly, all rating data from dataset should be store in a new DataFrame .

| | user_id | movie_id | rating | timestamp |
|---|---|---|---|---|
| **0** | 196 | 242 | 3 | 881250949 |
| **1** | 186 | 302 | 3 | 891717742 |
| **2** | 22 | 377 | 1 | 878887116 |

In this DataFrame , the column of user_id , movie_id and rating is useful for this project . So these data should be store in a new table with the index of user_id and the columns of movie_id and value of rating.

| movie_id | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | ... | 1673 | 1674 | 1675 | 1676 | 1677 | 1678 | 1679 | 1680 | 1681 | 1682 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| user_id | | | | | | | | | | | | | | | | | | | | | |
| 1 | 5.0 | 3.0 | 4.0 | 3.0 | 3.0 | 5.0 | 4.0 | 1.0 | 5.0 | 3.0 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 2 | 4.0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 2.0 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 3 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 4 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 5 | 4.0 | 3.0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

5 rows × 1682 columns

NaN means this user did not rate this movie.

Then next is calculate the Pearson correlation coefficient for each other user. In DataFrame ,there is a convenient function named ".corr(method='pearson', min_periods=1)". This function could calculate the correlation coefficient for all columns. While the min_periods is a significant parameter to be decided.

**2.Min_periods parameter determination**

The basic method for determining such a parameter is to calculate the standard deviation of the correlation coefficient when min_periods takes different values. The value of standard deviation should be more smaller more better .Although because of the sample space is very sparse, if the min_periods is too high then the result will be too small .So only one compromise can be selected.

Here the method to determine the standard deviation of the rating system is selecting a pair of users with the most overlapping ratings in data ,and using the standard deviation of the correlation coefficients between them to make a point estimate for the overall standard deviation. Under this premise ,statistics on the correlation coefficients of the pair of users under different sample sizes were taken to observe the standard deviation changes.

Firstly , find the pair of users with the highest overlap rating .There is a new square matrix be created named foo with user and the filled in the number of overlapping scores between different users.

This step is particularly time-consuming.Because this loop will run almost 1000*1000 = 1 million times. When finished , the highest overlap rating is 405 ,and the index and column of this result is 346 and 846.

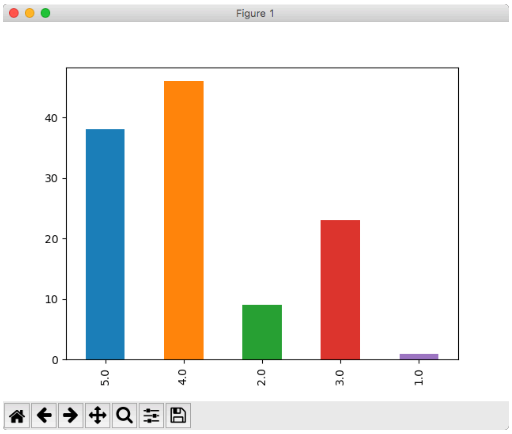
| movie_id | 2 | 4 | 11 | 12 | 22 | 29 | 31 | 33 | 50 | 53 | ... | 748 | 780 | 785 | 802 | 944 | 967 | 1018 | 1110 | 1188 | 1210 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| user_id | | | | | | | | | | | | | | | | | | | | | |
| 846 | 5.0 | 5.0 | 5.0 | 5.0 | 4.0 | 2.0 | 4.0 | 5.0 | 5.0 | 3.0 | ... | 3.0 | 4.0 | 4.0 | 2.0 | 2.0 | 3.0 | 4.0 | 3.0 | 2.0 | 2.0 |
| 346 | 5.0 | 4.0 | 4.0 | 5.0 | 5.0 | 4.0 | 4.0 | 5.0 | 5.0 | 1.0 | ... | 4.0 | 2.0 | 3.0 | 4.0 | 3.0 | 2.0 | 3.0 | 1.0 | 1.0 | 3.0 |

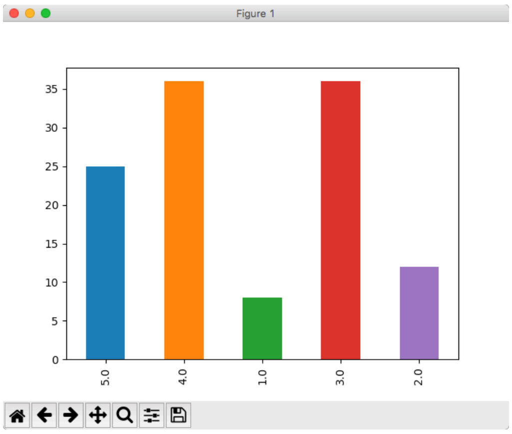
Then the Pearson correlation coefficient of this pair is calculated.

```
data.ix[346].corr(data.ix[846])
```

```
0.4348043294875953
```

The result is 0.4348 . A good result ! Then two graphs were created to show the rating distribution.The left is graph for user 846 ,and the right is graph for user 346.




For the correlation coefficient statistics of these two users, randomly selected 10, 20, 50, 100, 200, 500 sample values for min_periods, and 20 times each. Statistics:

| | 10 | 20 | 50 | 100 | 200 | 500 |
|---|---|---|---|---|---|---|
| **0** | 0.794531 | 0.494152 | 0.431123 | 0.447711 | 0.434804 | 0.434804 |
| **1** | 0.507546 | 0.515937 | 0.399440 | 0.397829 | 0.434804 | 0.434804 |
| **2** | -0.086639 | 0.621190 | 0.517636 | 0.430620 | 0.434804 | 0.434804 |
| **3** | 0.752651 | 0.331686 | 0.406148 | 0.439970 | 0.434804 | 0.434804 |
| **4** | 0.250000 | 0.623633 | 0.466578 | 0.499697 | 0.434804 | 0.434804 |

| | 10 | 20 | 50 | 100 | 200 | 500 |
|---|---|---|---|---|---|---|
| **count** | 20.000000 | 20.000000 | 20.000000 | 20.000000 | 2.000000e+01 | 2.000000e+01 |
| **mean** | 0.370003 | 0.396016 | 0.448627 | 0.427873 | 4.348043e-01 | 4.348043e-01 |
| **std** | 0.306590 | 0.210776 | 0.086908 | 0.027774 | 1.665335e-16 | 2.138592e-16 |
| **min** | -0.315854 | -0.019061 | 0.231795 | 0.370375 | 4.348043e-01 | 4.348043e-01 |
| **25%** | 0.187354 | 0.300564 | 0.404471 | 0.417707 | 4.348043e-01 | 4.348043e-01 |
| **50%** | 0.334545 | 0.473156 | 0.455713 | 0.430984 | 4.348043e-01 | 4.348043e-01 |
| **75%** | 0.613678 | 0.568353 | 0.508711 | 0.440223 | 4.348043e-01 | 4.348043e-01 |
| **max** | 0.799868 | 0.674171 | 0.599712 | 0.499697 | 4.348043e-01 | 4.348043e-01 |

From the line of std , the best value of min_periods should be 100.

**3.Algorithm test**

In order to confirm the degree of reliability of the recommended algorithm under min_periods=100, it is better to do a test first. The specific method is to randomly select 250 users among users whose evaluation number is greater than 100, each person randomly extracts one evaluation and saves it in an array, and deletes this evaluation in the data table. Then, based on the castrated data table, the expected 1000-rated scores are extracted. Finally, the correlations are compared with the real evaluation arrays to see what the results are.

The next graph is the selected true rating .

```
5   168    3.0
6   506    4.0
7   78     3.0
11  603    4.0
13  478    4.0
dtype: float64
```

Next calculate the rating expectations for the 250 user-movie pairs. The calculation method is: weighted average of other user ratings with a user correlation coefficient greater than 0.1, the weight value is the correlation coefficient:

The next graph is the rating calculated.

```
5   168    4.592187
6   506    4.104080
7   78     2.239712
11  603    4.410184
13  478    4.113704
dtype: float64
```

The next graphs are the result of correlation coefficient.

```
result.corr(check_ser.reindex(result.index))
```

```
0.3813086765643036
```

```
count    193.000000
mean       0.841249
std        0.629747
min        0.000000
25%        0.327459
50%        0.810975
75%        1.144610
max        3.000000
dtype: float64
```

From the graph ,the sample size of 193 can reach a correlation 0.3813(some of them were evaluated because the rating number is less than 100). It should be said that the result is not bad. If not filter out users whose evaluation number is less than 100 at the beginning, it will obviously cost longer when calculating , and the sample size in result will be small. However, because of the smaller sample size, the correlation coefficient can be increased to 0.5~0.6.

In addition, from the statistics of the absolute value of the difference between the expected and actual evaluation, the data is also ideal.

**4.Realize recommendations**

We randomly select a user to make a recommendation list for him:
random user_id=524

| movie_id | | | | | |
|---|---|---|---|---|---|
| 1368 | 5.000000 | 262 | 4.670026 | 599 | 1.000000 |
| 1643 | 5.000000 | 1240 | 4.597609 | 1509 | 1.000000 |
| 989 | 5.000000 | 1512 | 4.593395 | 981 | 1.000000 |
| 1367 | 5.000000 | 1398 | 4.589620 | 247 | 1.000000 |
| 1599 | 5.000000 | 169 | 4.543403 | 669 | 1.000000 |
| 1592 | 5.000000 | 114 | 4.541364 | 84 | 1.000000 |
| 1122 | 5.000000 | 316 | 4.526592 | 1508 | 1.000000 |
| 1558 | 5.000000 | 408 | 4.515048 | 681 | 1.000000 |
| 851 | 5.000000 | 1131 | 4.503310 | 1376 | 1.000000 |
| 1536 | 5.000000 | 357 | 4.487768 | 687 | 1.000000 |
| 119 | 5.000000 | 592 | 4.485684 | 688 | 1.000000 |
| 814 | 5.000000 | | ... | 1304 | 1.000000 |
| 868 | 5.000000 | 314 | 1.000000 | 1087 | 1.000000 |
| 1656 | 5.000000 | 1621 | 1.000000 | 1250 | 1.000000 |
| 1450 | 5.000000 | 1025 | 1.000000 | 1355 | 1.000000 |
| 626 | 5.000000 | 1408 | 1.000000 | 908 | 1.000000 |
| 1201 | 5.000000 | 1034 | 1.000000 | 907 | 1.000000 |
| 1449 | 4.745681 | 1392 | 1.000000 | 901 | 1.000000 |
| 1269 | 4.726562 | 352 | 1.000000 | 897 | 1.000000 |