**Recommendation System Application Project Report**

**Team Member:** LI HUANG

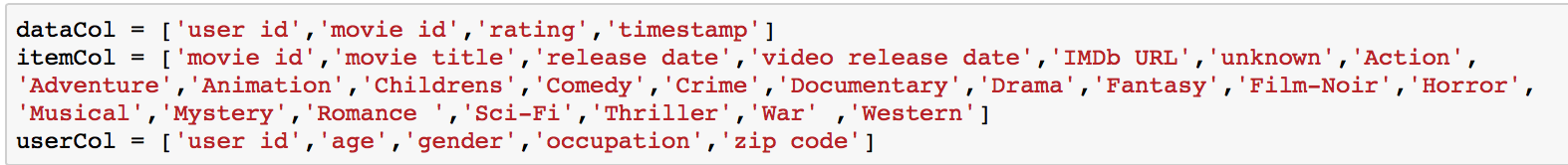
(Checking list: As required in the Project Report & Appendices, this Report include project structure, sample code, sample runs, sample output, system description, process documentation and evaluation, finally a readme part about how to use the application. The code will be in a separate python Jupiter file; The other related files will also be included in the submitted package.)

**Part 1. Data description**

<https://grouplens.org/datasets/movielens/>

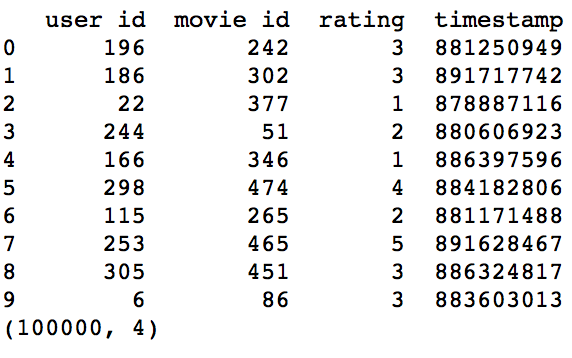
with the dataset with size 100k. I am using this dataset as it is very detailed both in user and item info for analysis. There are three main dataset that are used in this project from this movie lens data collection, ‘u.data’,’u.item’,’u.user’,’ua.test’. The size (rows, cols) of u.data, u.item and u.user datasets are (100000, 4), (1682, 24), (943, 5) respectively. The ua.test is the test dataset that set side for accuracy calculating in Part3.

The variables in the three datasets are:

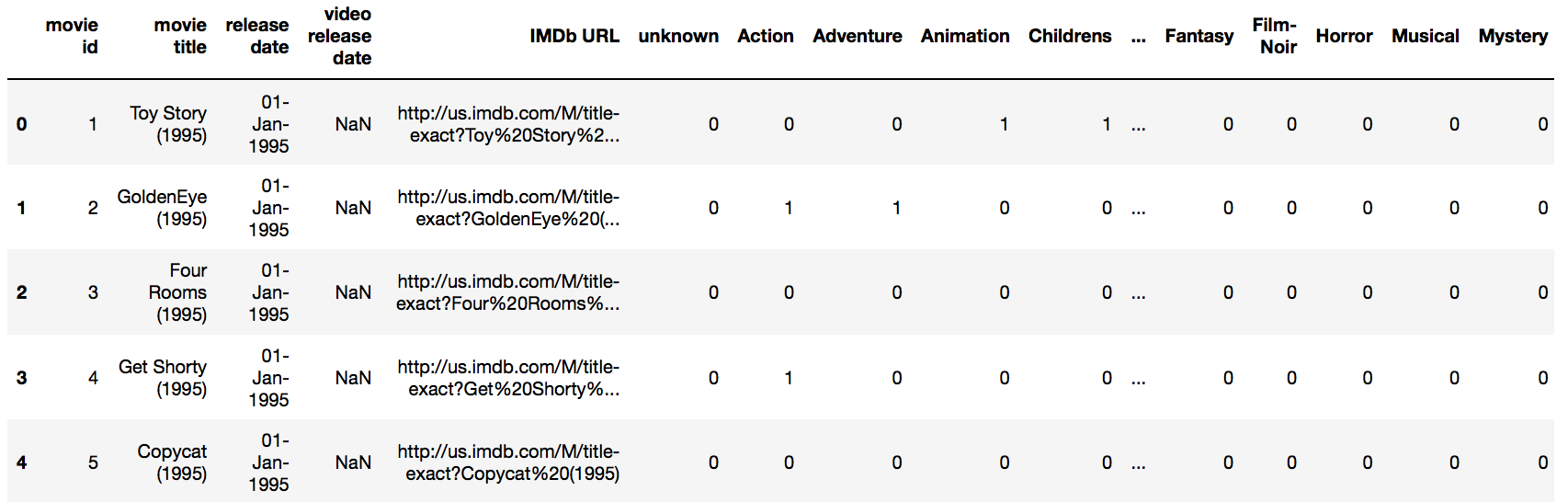


And here are how the three datasets look like:

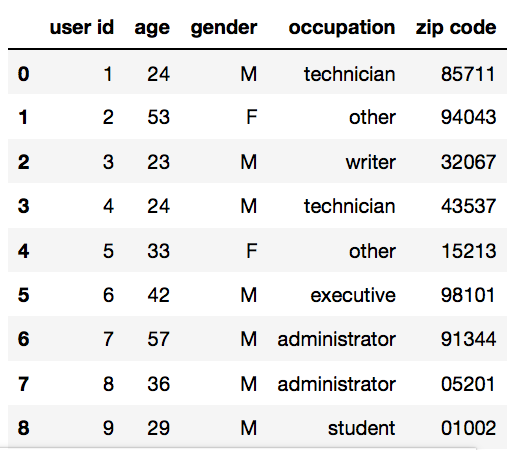
u.data



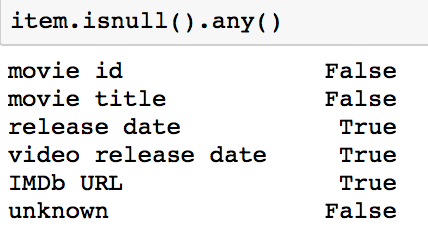
u.item



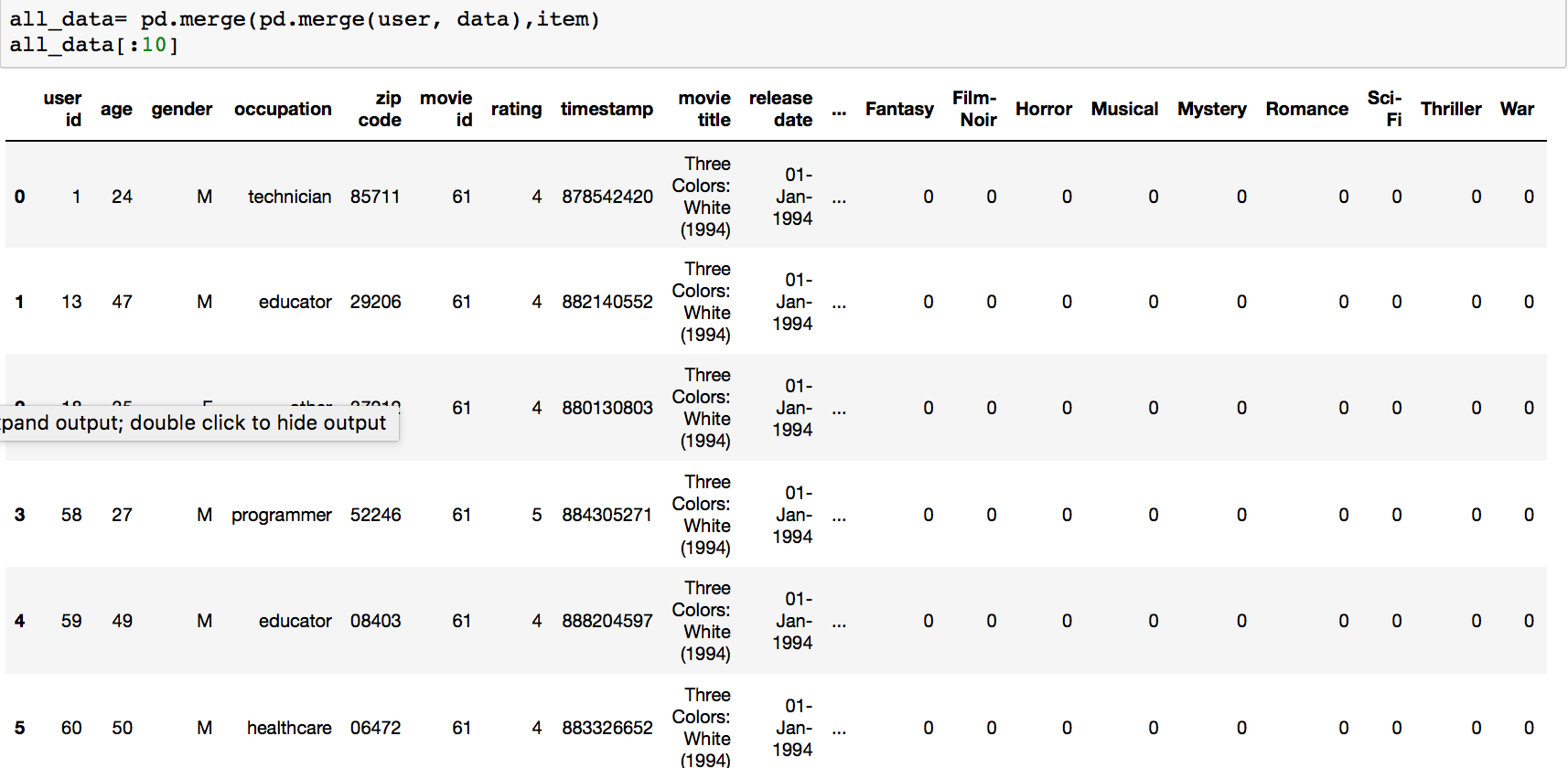
u.user



There is no missing values in the u.data and u.user dataset, and u.item have some missing values in three variables as below, but they are reference info and will not affect the implementation of our project, so we do not need to fulfill the missing values for this project.



If we merge all the data together, we get a full dataset with rating, item info and user info as following:

****

**Part 2. Methodology and Implementation Process Exploration**

In this project, I have processed the dataset with analysis and visualization tools from python library numpy, panda, matplotlib, scikit learn; performed classification, to be specific, K nearest neighbor classification; implemented user-based collaborative filtering recommender, item-based collaborative filtering recommender and matrix factorization; and evaluation the results.

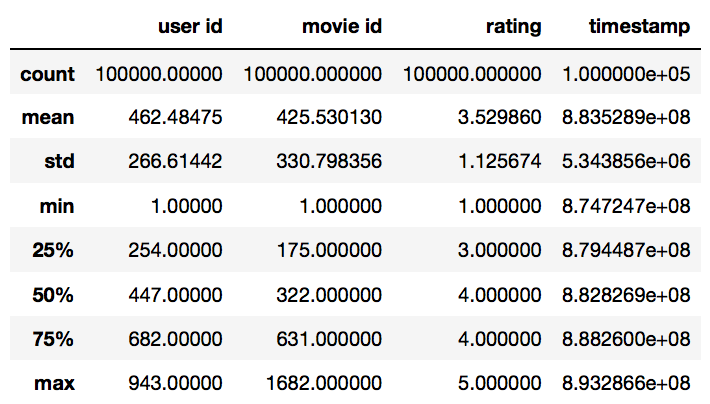
Following is the flow chart of methodology and structure of this project:

* **Data exploration and rating pivot**

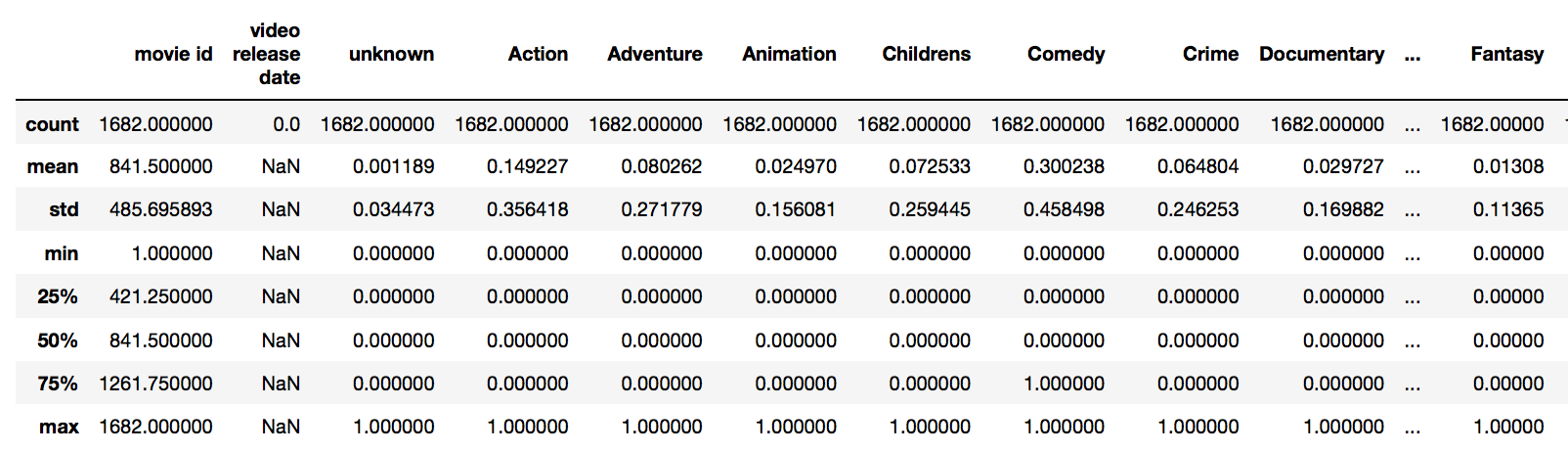
data descriptions for the three datasets:

dataset: data

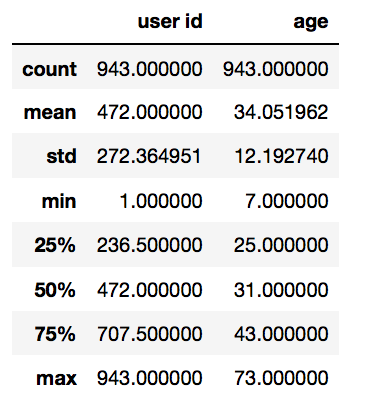
From the summary, we can see the min rating is 1, and all the 0 values in the pivot means not rated/missing.



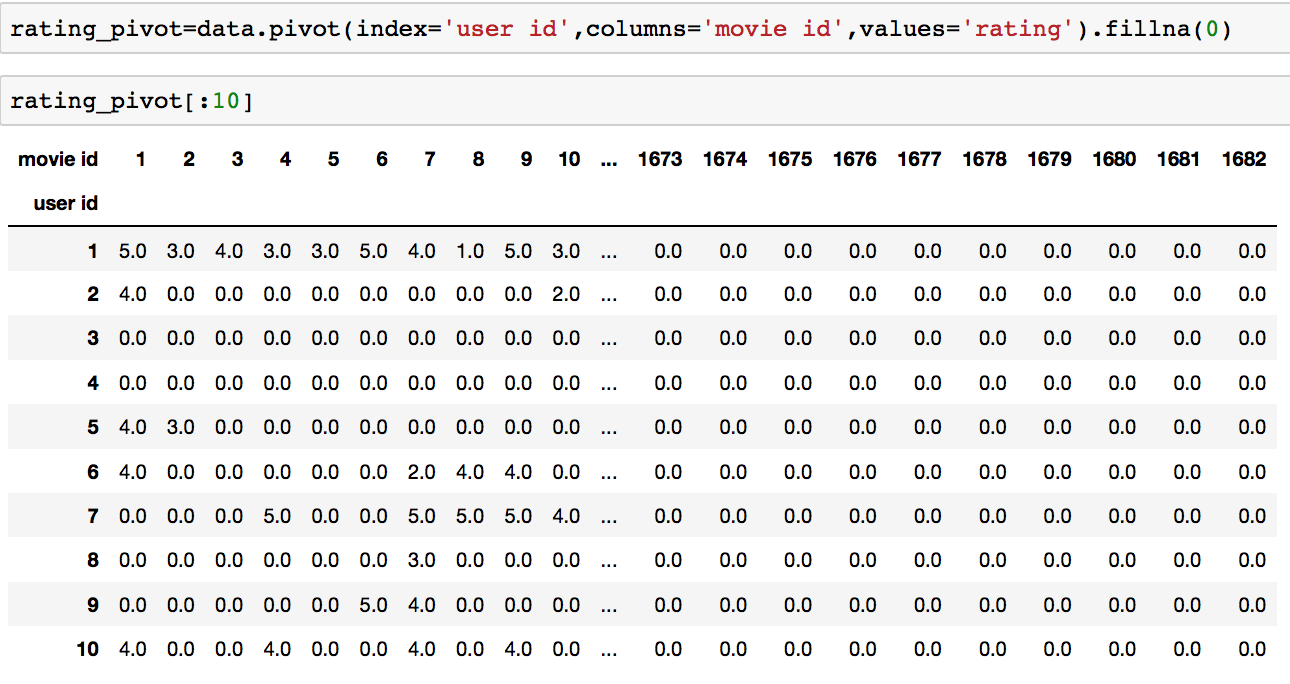
dataset: item



dataset: user



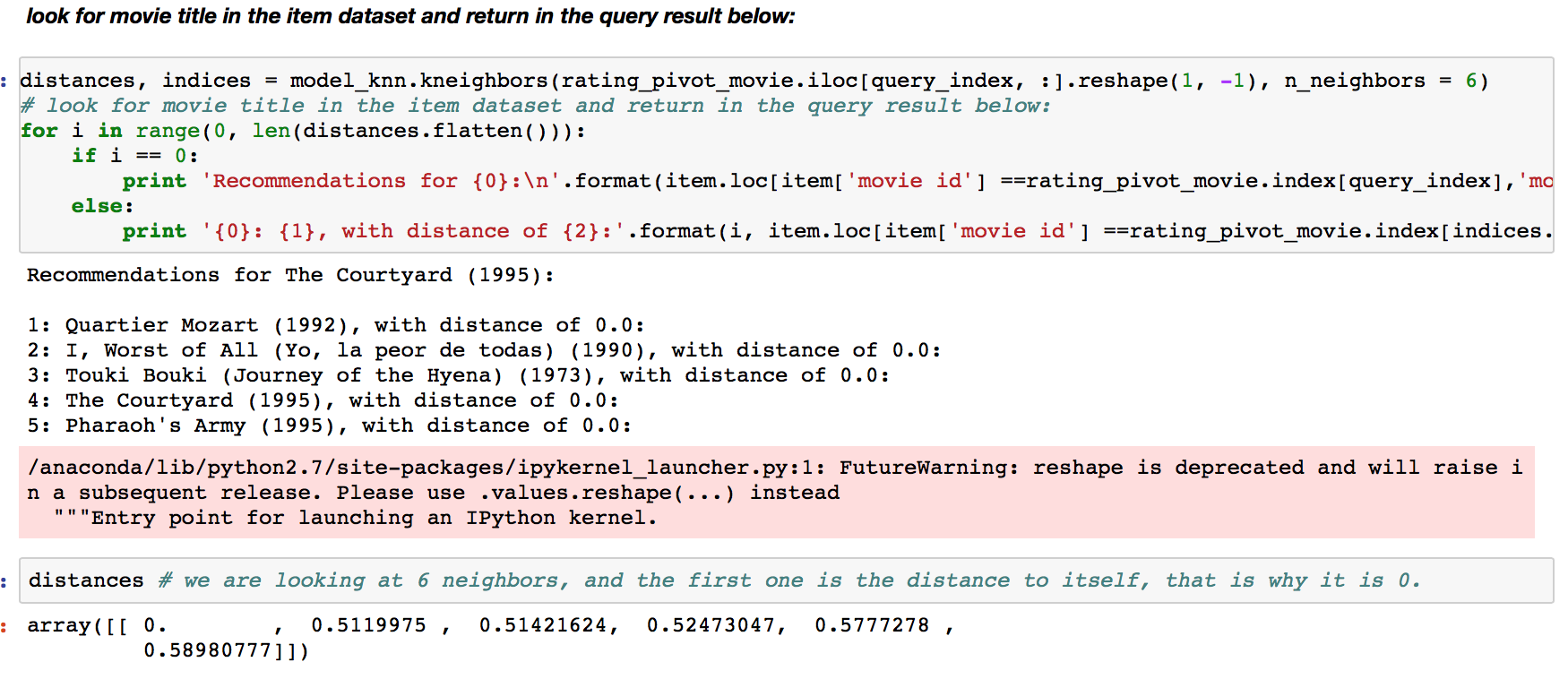
rating pivot:



* **recommender models exploration**

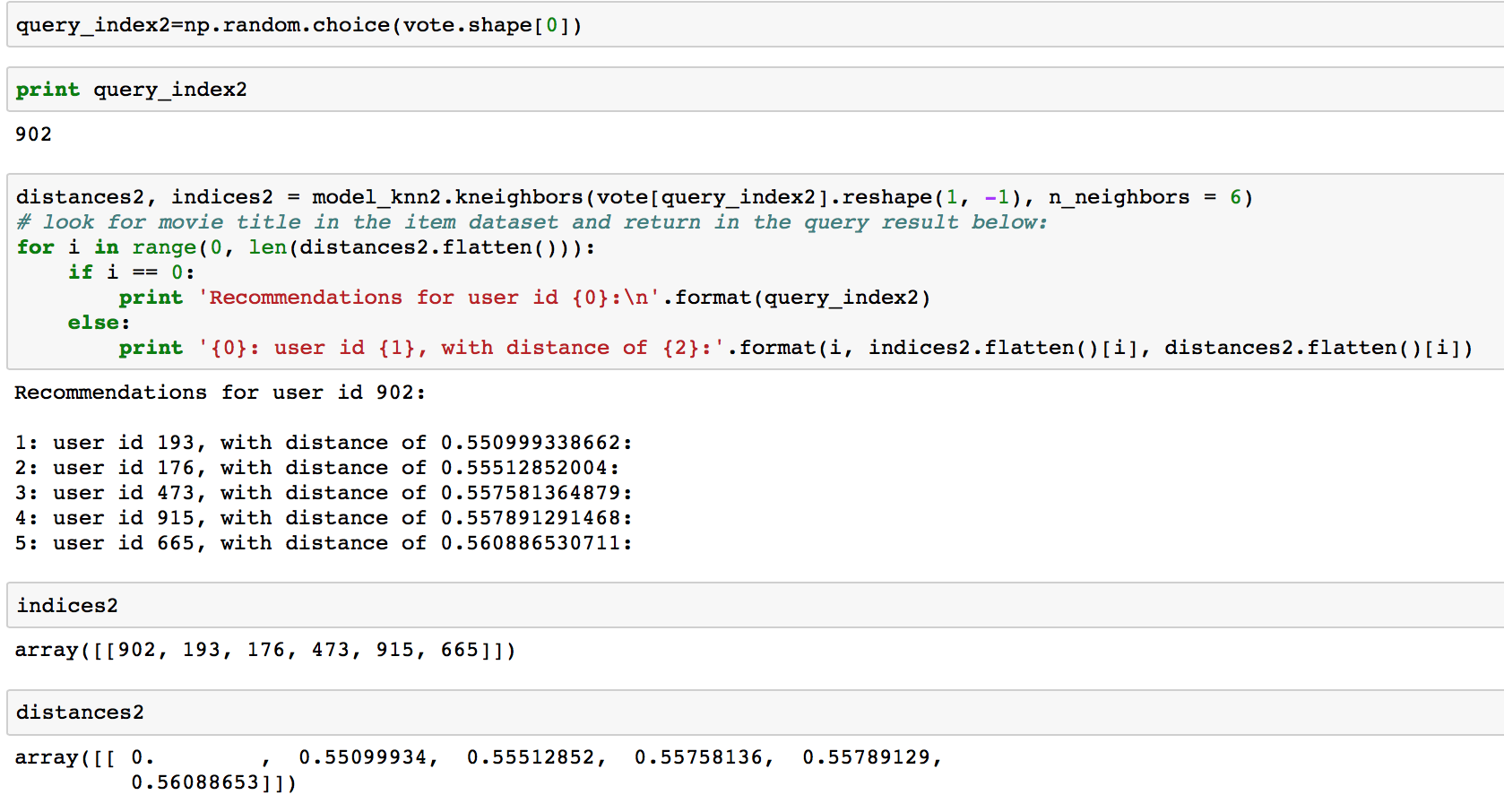
**Direction 1: given a movie query with id 1547(randomly generated), recommend the movie title and the titles of the most similar 5 movies.**

we are going to use item based knn to explore this direction. First, find the transpose of the rating pivot, and find k=6 nearest neighbors of the query item, and then return the neighbors, excluding itself. We are using k=6 here as the results also includes the movie itself and other 5 movies as required.



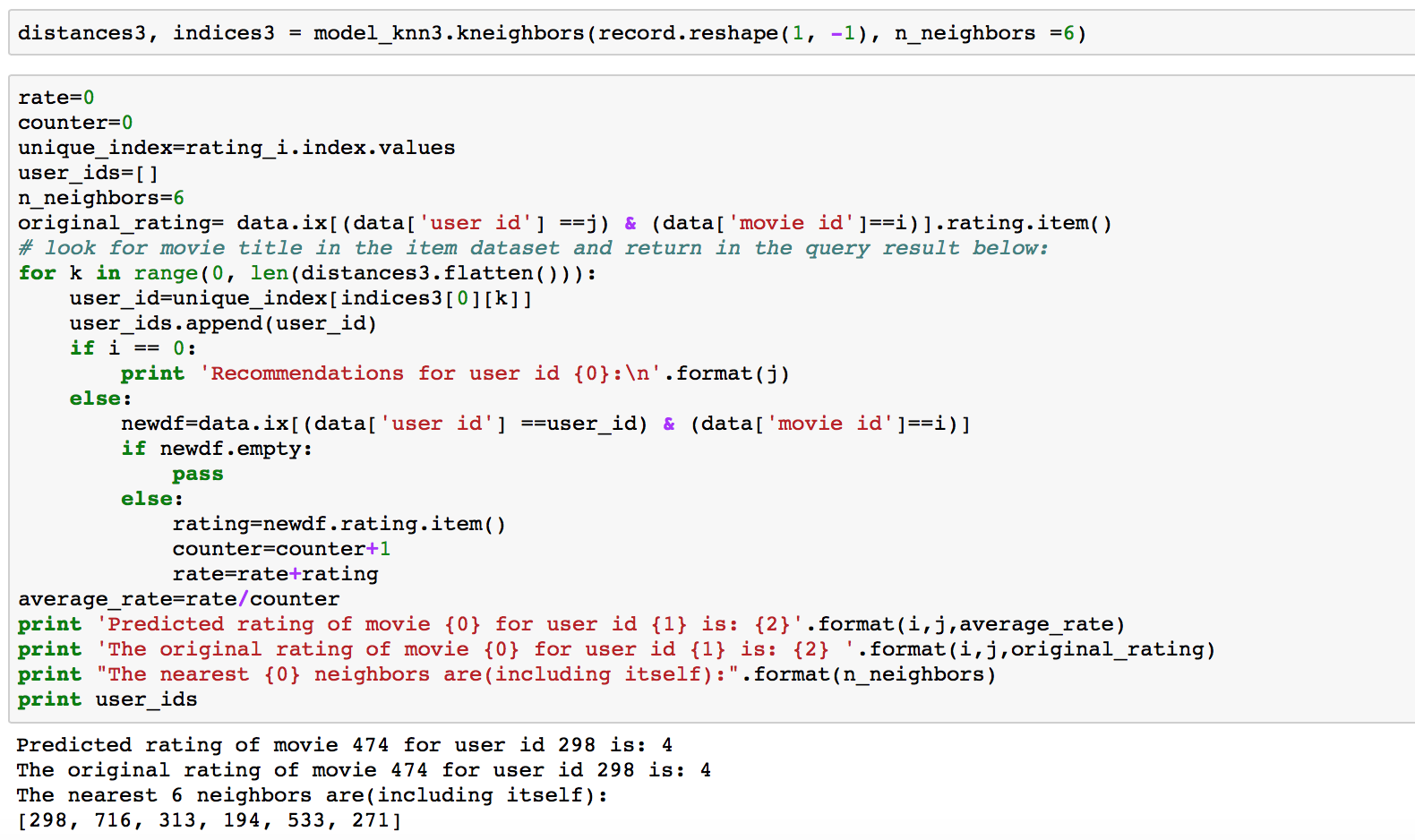
**Direction 2: given a user query id 902(randomly generated), return 5 users with most similar behavior.**

we are going to use user based knn to explore this direction with k=6, and when we get the 6 results, exclude the most similar one as it would be the user self id, the results would be 5 other user’s id as below:



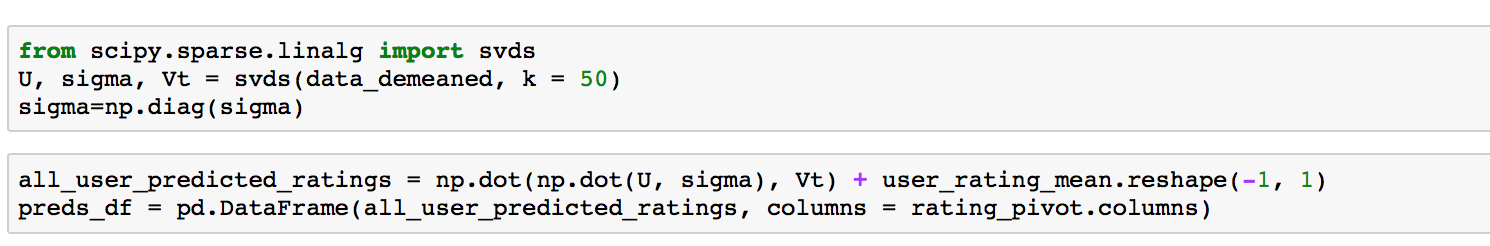
**Direction 3: given a user id 298 and movie query id 474(randomly generated), return the rating for this movie from the user.**

User based method would be: calculate nearest k users using pivot with all users who have rated this movie and without column 474(as we assume this info is unknown for user 298); as column movie 474 would be saved as target and average the nearest k neighbor's rating on this movie

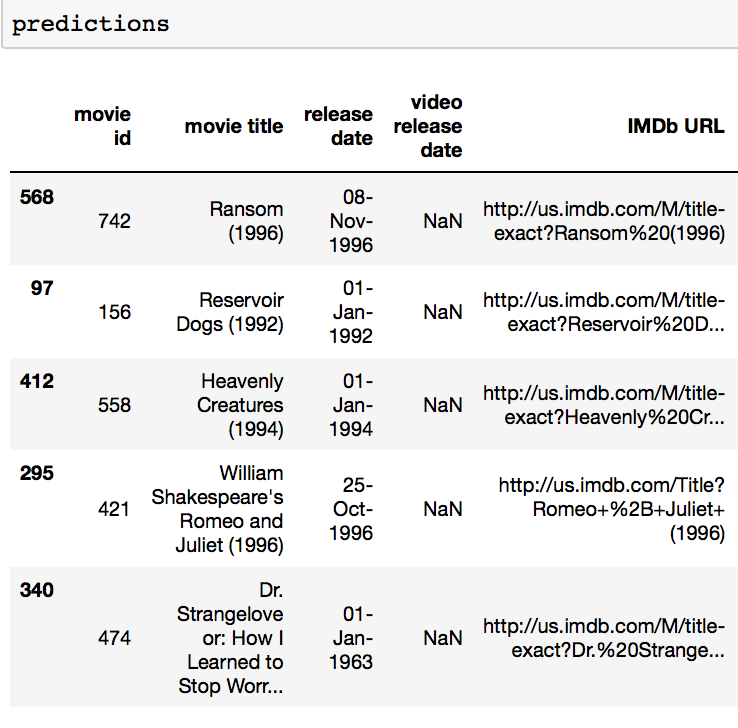


**Direction 4: given a user id, recommend some movies the user I have watched before.**

User We have tried to predict the rating by user and by movie in the last two models, now we could try to recommend new movies to user to watch using matrix factorization. Here is part of the algorithm:

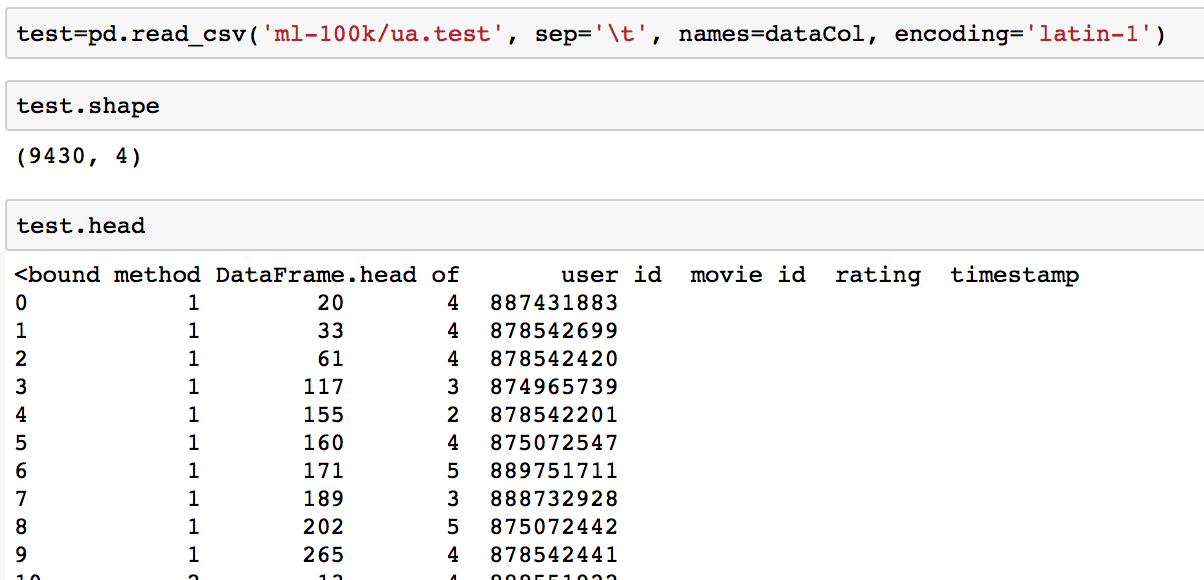


The recommended 5 movies:



* **implementation on three models**

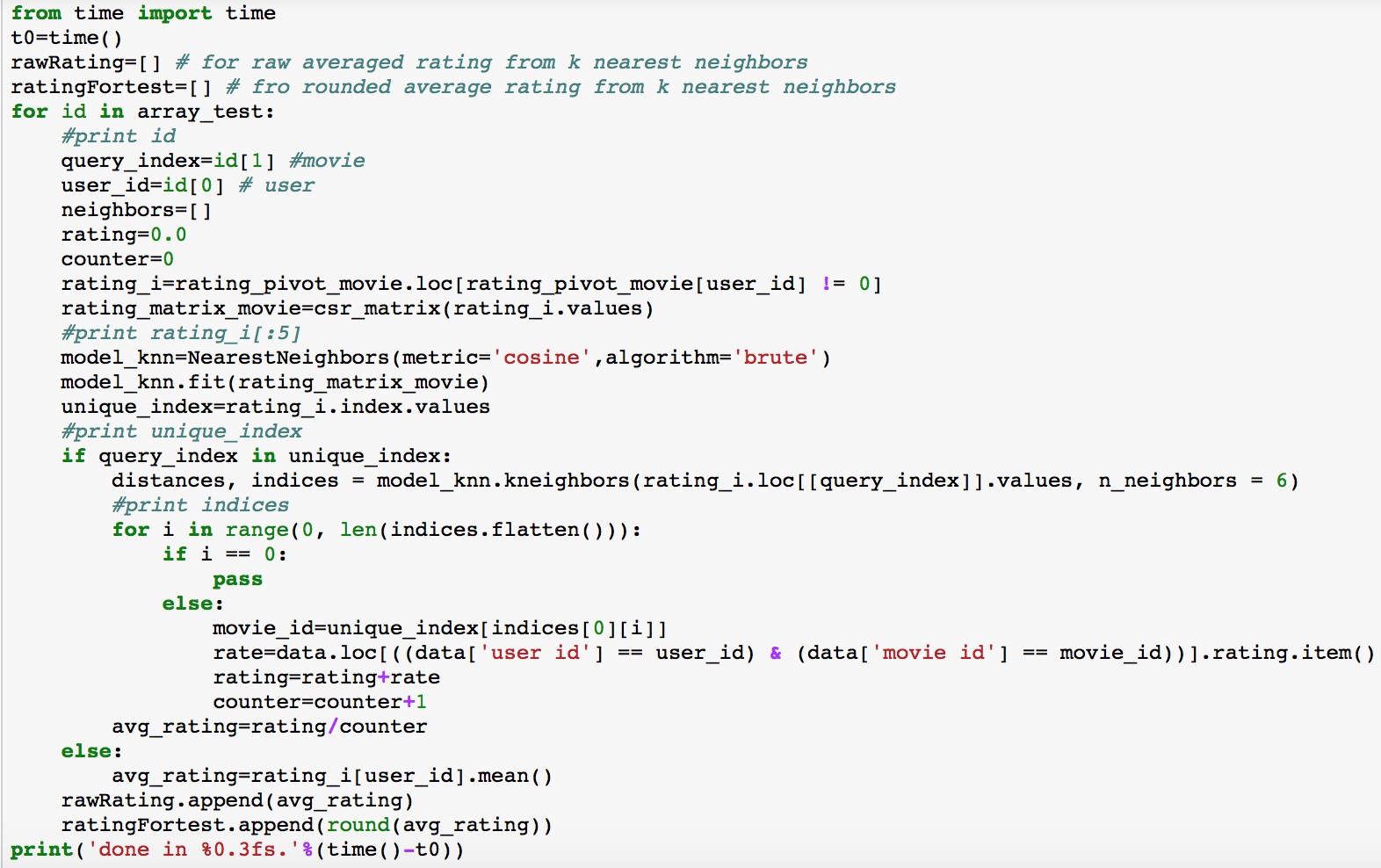
Firstly, load the test data from the m-lens data source: The data set ua.test is a test set from splitting the u data into a training set and a test set, with exactly 10 ratings per user in the test set. We are going to use the rating pivot as training dataset and the test set for testing in the following algorithms

****

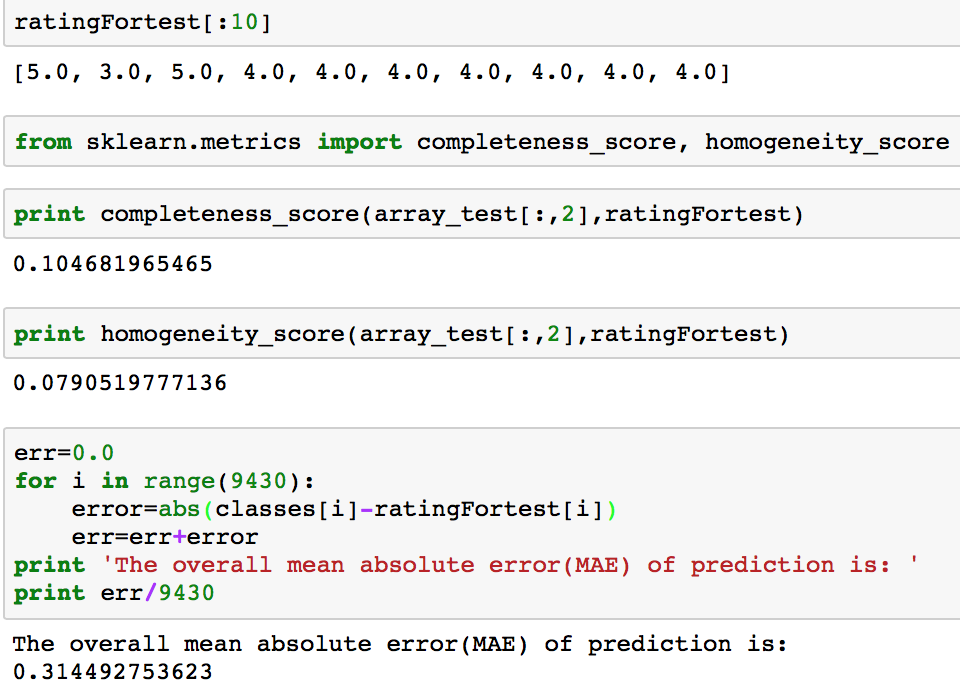
1. **item-based collaborative recommender**

For each instance (user\_id, movie\_id, rating) in the test, find all the movies the user have rated, selecte the closest k movies with the movie\_id in the instance; the k movies would include the moive\_id itself as the closest one, we would need to exclude it when calculating rating, then get a average rating for this moive\_id from this k-1 movies as the rating. save the rounded calculated rating into a list named labels; save the original rating from test as classes, so we could calculate the accuracy later.

here is the main function:



And calculate completeness score and homogeneity score, mean absolute error.



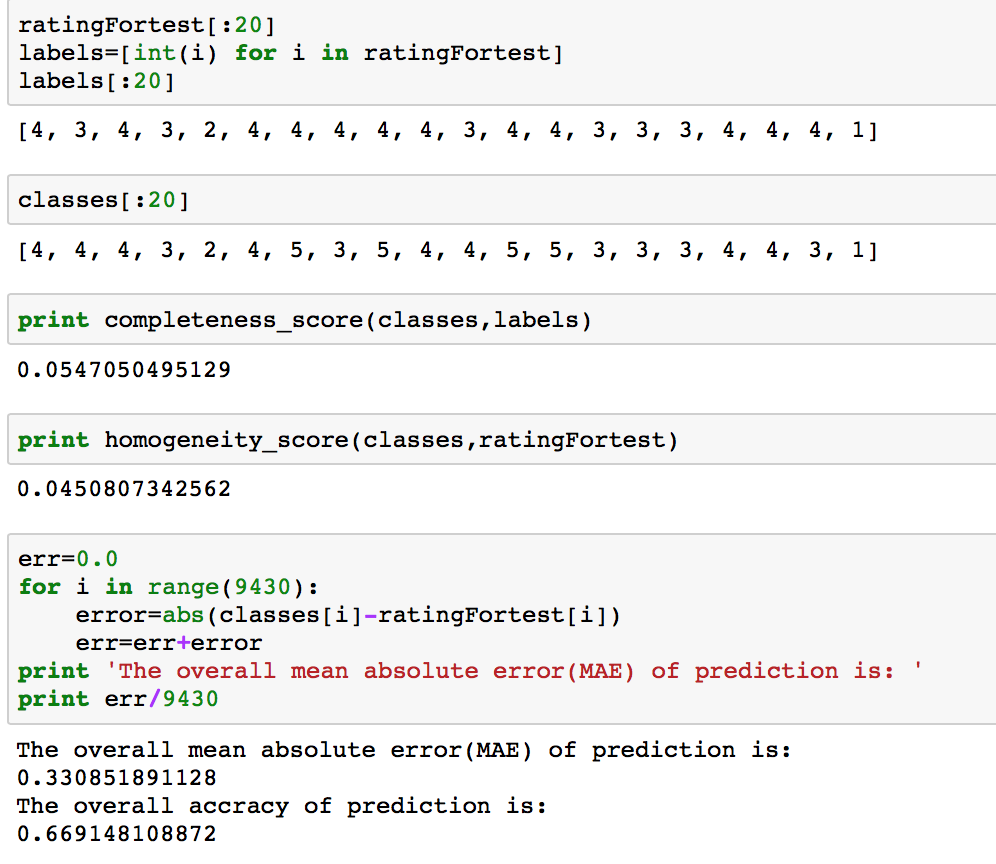
1. **user based collaborative recommender**

For each instance(user\_id, movie\_id, rating) in the test, find the subset of ratings of all the ratings for this movie, find the k nearest neighbors of user\_id, exclude itself, and calculate the average rating as the rating calculated.

main function:

****

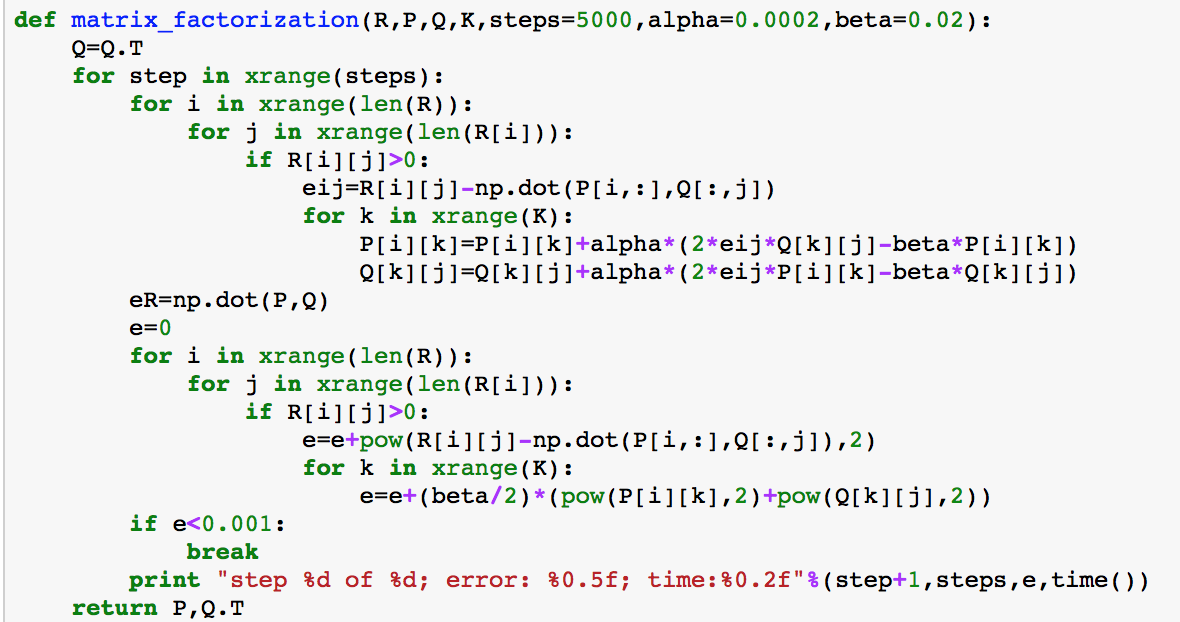
And calculate completeness score and homogeneity score, mean absolute error, as in item-based recommender.

****

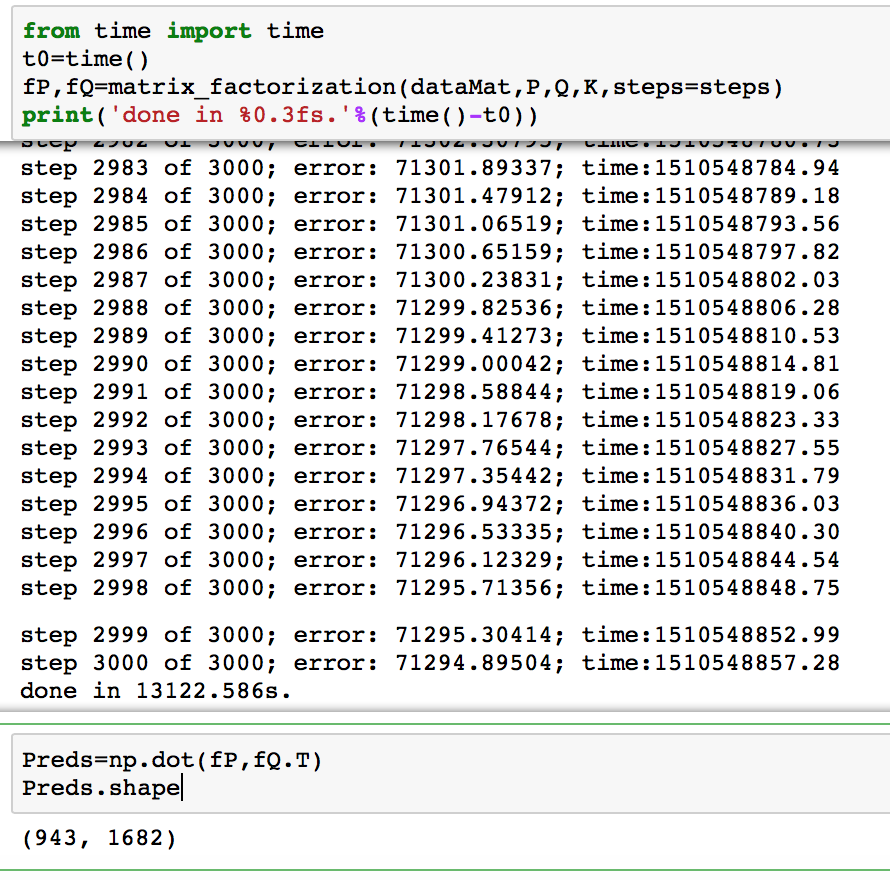
1. **matrix factorization based recommender**

For each instance(user\_id, movie\_id, rating) in the test, find the rating of all the this user for this movie from the predictions of the matrix factorization results.

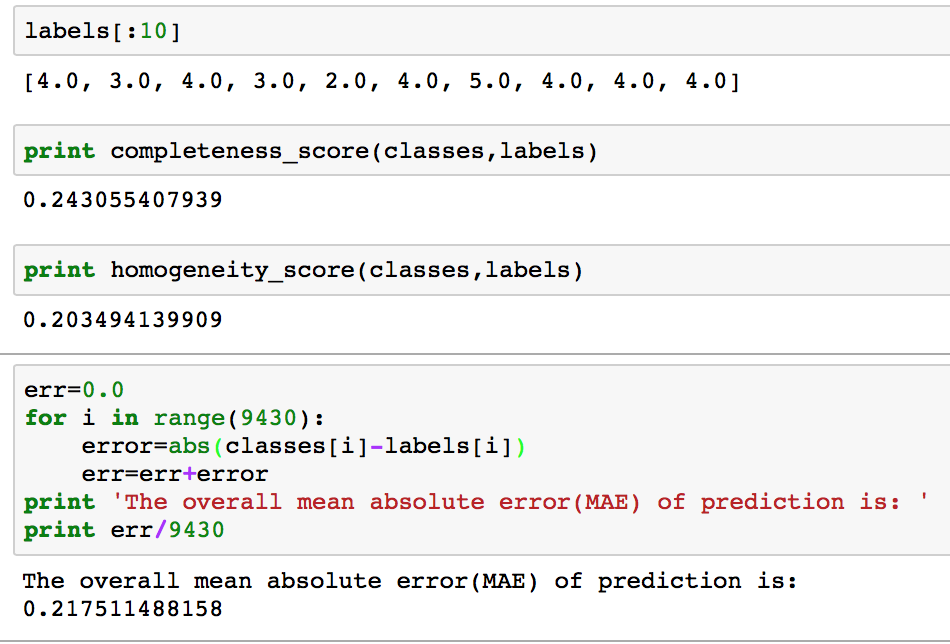
main function:

****

**result:**

****

And calculate completeness score and homogeneity score, mean absolute error:

****

**Part 3. Performance evaluation**

Result:



From the comparison of the result, we can see the Matrix\_factorization based recommender is more accurate than the other two recommenders, but it also takes a much longer time to generate the prediction result. User\_based recommender takes a little bit longer than the item\_based recommender, and the Mean absolute error is also a little higher than item\_based recommender.

**Part 4. Future direction**

Due to the capacity of my computer, I am using a 100,000 rating dataset, which include some one rating only movie, and make it hard to predict its rating. For future research reference, if possible, using a larger dataset to test on these three algorithms would generate better results.

**Readme file：**

The data used in this application is movie-lens 100k rating data, with mainly four features used, movie\_id, user\_id, rating and movie\_title from 3 dataset u.data, u.item, u.user; and a set aside testing dataset ua.test, including mainly three features used for each instance, movie\_id, user\_id and rating.

From the models explored, this application could answer below queries:

1. Give a user\_id, return k nearest neighbors with the most similar rating behavior;
2. Given a movie\_id, return k nearest neighbors with the most similar rating;
3. Given a user\_id, recommend k movies the user have not watched before;
4. Given a user\_id, movie\_id, return the predicted rating from three algorithms: user\_based recommender, item\_based recommender and matrix\_factorization\_based recommender.
5. Gvien a list of user\_id, movie\_id, return the predicted rating and MAE for the three algorithms: user\_based recommender, item\_based recommender and matrix\_factorization\_based recommender.