Christopher Tran
29MAR2020
Hw7

1. Recommender systems are a hot topic in data science companies. Recommender systems aim to predict the rating that a user will give for an item (e.g., a restaurant, a movie, a product, a Point of Interest). Surprise (http://surpriselib.com) is a Python package for developing recommender systems. To install Surprise, the easiest way is to use pip. Open your console: $ pip install numpy
   $ pip install scikit-surprise

2. Download an experimental dataset “restaurant_ratings.txt” from the Canvas: Files/Data/ restaurant_ratings.txt

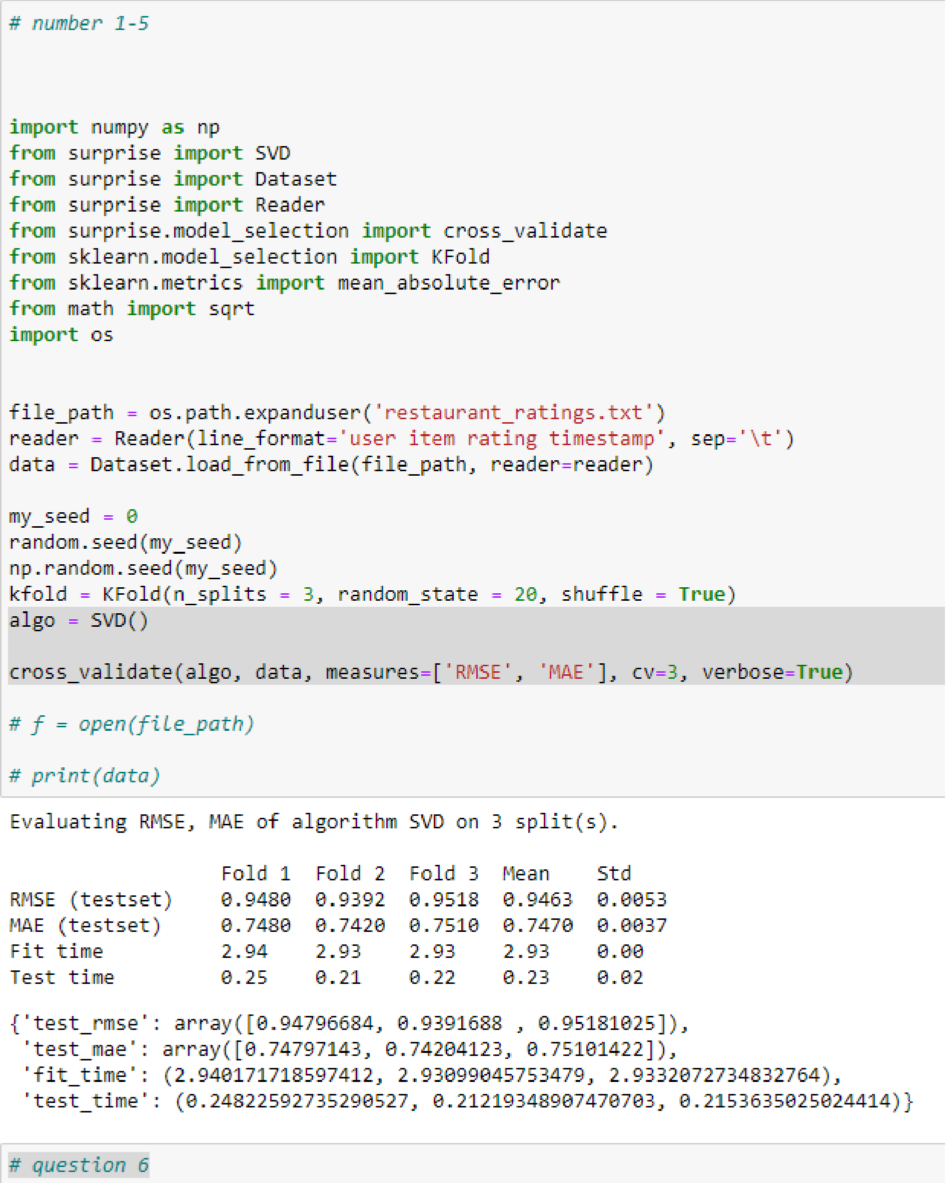
3. Load data from “restaurant_ratings.txt” with line format: 'user item rating timestamp'. The introduction of the Surprise data module can be found via http://surprise.readthedocs.io/en/v1.0.2/dataset.html. The sample python codes are as follows:

   ```
   from surprise import Dataset from
   surprise import Reader import os

   #load data from a file
   file_path = os.path.expanduser('restaurant_ratings.txt') reader =
   Reader(line_format='user item rating timestamp', sep='\t') data =
   Dataset.load_from_file(file_path, reader=reader)
   ```

4. MAE and RMSE are two famous metrics for evaluating the performances of a recommender system. The definition of MAE can be found via: https://en.wikipedia.org/wiki/Mean_absolute_error. The definition of RMSE can be found via: https://en.wikipedia.org/wiki/Root-mean-square_deviation.

5. Split the data for 3-folds cross-validation, and compute the MAE and RMSE of the SVD (Singular Value Decomposition) algorithm. The introduction of SVD algorithm can be found via http://surprise.readthedocs.io/en/v1.0.2/matrix_factorization.html.

Christopher Tran
29MAR2020
Hw7

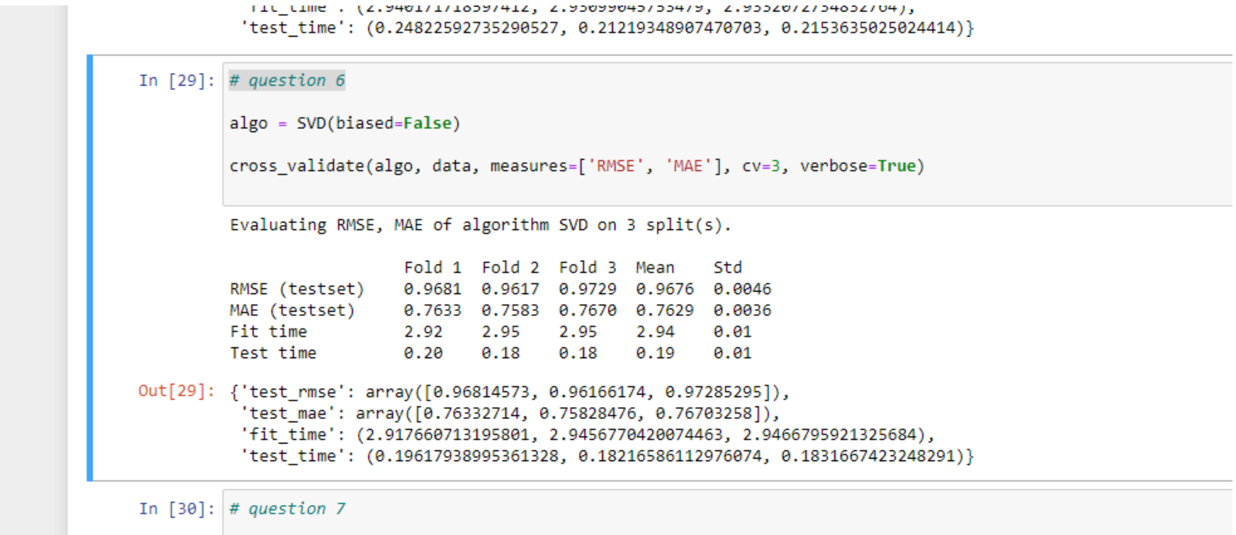


```
# number 1-5



import numpy as np
from surprise import SVD
from surprise import Dataset
from surprise import Reader
from surprise.model_selection import cross_validate
from sklearn.model_selection import KFold
from sklearn.metrics import mean_absolute_error
from math import sqrt
import os


file_path = os.path.expanduser('restaurant_ratings.txt')
reader = Reader(line_format='user item rating timestamp', sep='\t')
data = Dataset.load_from_file(file_path, reader=reader)

my_seed = 0
random.seed(my_seed)
np.random.seed(my_seed)
kfold = KFold(n_splits = 3, random_state = 20, shuffle = True)
algo = SVD()

cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)

# f = open(file_path)

# print(data)
```

```
Evaluating RMSE, MAE of algorithm SVD on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.9480  0.9392  0.9518  0.9463  0.0053
MAE (testset)     0.7480  0.7420  0.7510  0.7470  0.0037
Fit time          2.94    2.93    2.93    2.93    0.00
Test time         0.25    0.21    0.22    0.23    0.02

{'test_rmse': array([0.94796684, 0.9391688 , 0.95181025]),
 'test_mae': array([0.74797143, 0.74204123, 0.75101422]),
 'fit_time': (2.940171718597412, 2.93099045753479, 2.9332072734832764),
 'test_time': (0.24822592735290527, 0.21219348907470703, 0.2153635025024414)}
```

```
# question 6
```

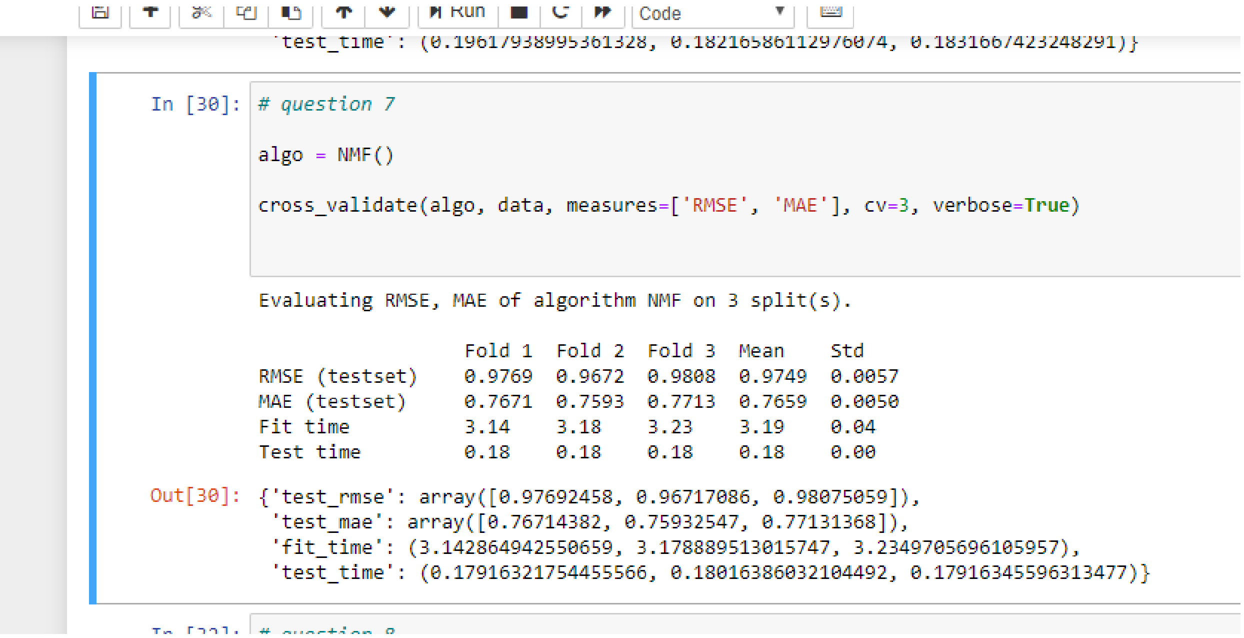
6. Split the data for 3-folds cross-validation, and compute the MAE and RMSE of the PMF (Probabilistic Matrix Factorization) algorithm. The introduction of PMF algorithm can be found via http://surprise.readthedocs.io/en/v1.0.2/matrix_factorization.html.



```
 'fit_time': (2.946171718597412, 2.95099045753479, 2.9552072754832704),
 'test_time': (0.24822592735290527, 0.21219348907470703, 0.2153635025024414)}
```

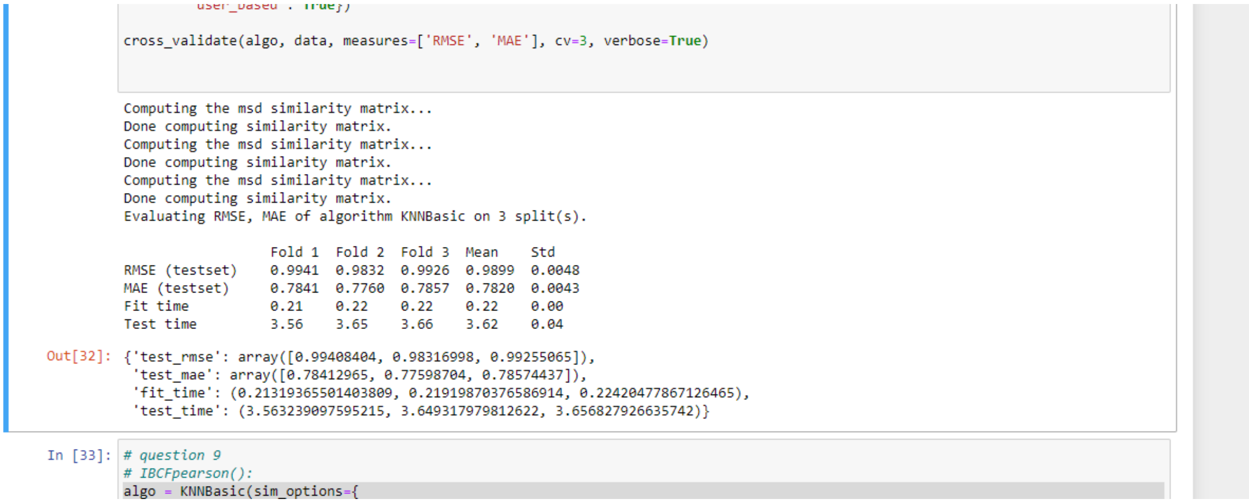
```
In [29]: # question 6

         algo = SVD(biased=False)

         cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)

         Evaluating RMSE, MAE of algorithm SVD on 3 split(s).

                           Fold 1  Fold 2  Fold 3  Mean    Std
         RMSE (testset)    0.9681  0.9617  0.9729  0.9676  0.0046
         MAE (testset)     0.7633  0.7583  0.7670  0.7629  0.0036
         Fit time          2.92    2.95    2.95    2.94    0.01
         Test time         0.20    0.18    0.18    0.19    0.01

Out[29]: {'test_rmse': array([0.96814573, 0.96166174, 0.97285295]),
          'test_mae': array([0.76332714, 0.75828476, 0.76703258]),
          'fit_time': (2.917660713195801, 2.9456770420074463, 2.9466795921325684),
          'test_time': (0.19617938995361328, 0.18216586112976074, 0.1831667423248291)}

In [30]: # question 7
```

7. 7. Split the data for 3-folds cross-validation, and compute the MAE and RMSE of the NMF (Non-negative Matrix Factorization) algorithm. The introduction of NMF algorithm can be found via http://surprise.readthedocs.io/en/v1.0.2/matrix_factorization.html.



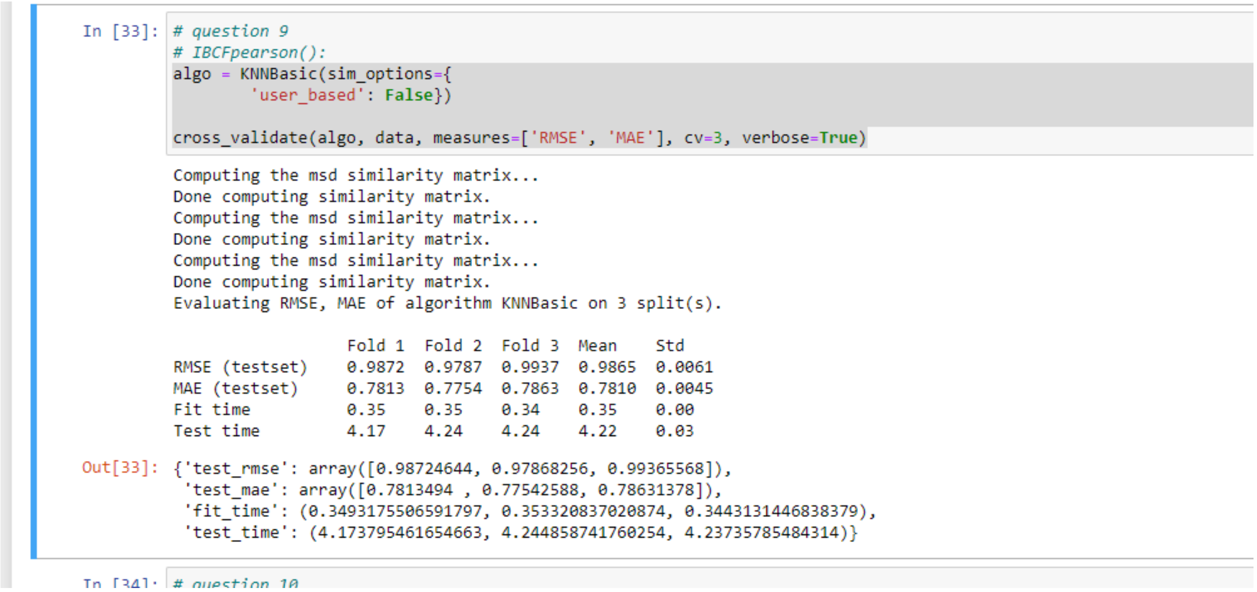
```
 'test_time': (0.19617938995361328, 0.18216586112976074, 0.1831667423248291)}

In [30]: # question 7

         algo = NMF()

         cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)

         Evaluating RMSE, MAE of algorithm NMF on 3 split(s).

                           Fold 1  Fold 2  Fold 3  Mean    Std
         RMSE (testset)    0.9769  0.9672  0.9808  0.9749  0.0057
         MAE (testset)     0.7671  0.7593  0.7713  0.7659  0.0050
         Fit time          3.14    3.18    3.23    3.19    0.04
         Test time         0.18    0.18    0.18    0.18    0.00

Out[30]: {'test_rmse': array([0.97692458, 0.96717086, 0.98075059]),
          'test_mae': array([0.76714382, 0.75932547, 0.77131368]),
          'fit_time': (3.142864942550659, 3.178889513015747, 3.2349705696105957),
          'test_time': (0.17916321754455566, 0.18016386032104492, 0.17916345596313477)}

In [31]: # question 8
```

8. Split the data for 3-folds cross-validation, and compute the MAE and RMSE of the User based Collaborative Filtering algorithm. The introduction of User based Collaborative Filtering algorithm can be found via http://surprise.readthedocs.io/en/v1.0.2/knn_inspired.html.



```
        'user_based': True})

cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)

Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Evaluating RMSE, MAE of algorithm KNNBasic on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.9941  0.9832  0.9926  0.9899  0.0048
MAE (testset)     0.7841  0.7760  0.7857  0.7820  0.0043
Fit time          0.21    0.22    0.22    0.22    0.00
Test time         3.56    3.65    3.66    3.62    0.04

Out[32]: {'test_rmse': array([0.99408404, 0.98316998, 0.99255065]),
          'test_mae': array([0.78412965, 0.77598704, 0.78574437]),
          'fit_time': (0.21319365501403809, 0.21919870376586914, 0.22420477867126465),
          'test_time': (3.563239097595215, 3.649317979812622, 3.656827926635742)}

In [33]: # question 9
         # IBCFpearson():
         algo = KNNBasic(sim_options={
```

9. Split the data for 3-folds cross-validation, and compute the MAE and RMSE of the Item based Collaborative Filtering algorithm. The introduction of Item based Collaborative Filtering algorithm can be found via http://surprise.readthedocs.io/en/v1.0.2/knn_inspired.html.



```
In [33]: # question 9
         # IBCFpearson():
         algo = KNNBasic(sim_options={
                 'user_based': False})

         cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)

         Computing the msd similarity matrix...
         Done computing similarity matrix.
         Computing the msd similarity matrix...
         Done computing similarity matrix.
         Computing the msd similarity matrix...
         Done computing similarity matrix.
         Evaluating RMSE, MAE of algorithm KNNBasic on 3 split(s).

                           Fold 1  Fold 2  Fold 3  Mean    Std
         RMSE (testset)    0.9872  0.9787  0.9937  0.9865  0.0061
         MAE (testset)     0.7813  0.7754  0.7863  0.7810  0.0045
         Fit time          0.35    0.35    0.34    0.35    0.00
         Test time         4.17    4.24    4.24    4.22    0.03

Out[33]: {'test_rmse': array([0.98724644, 0.97868256, 0.99365568]),
          'test_mae': array([0.7813494 , 0.77542588, 0.78631378]),
          'fit_time': (0.3493175506591797, 0.353320837020874, 0.3443131446838379),
          'test_time': (4.173795461654663, 4.244858741760254, 4.23735785484314)}

In [34]: # question 10
```

Questions 10-15 can be found at the following link due to length of code and output: