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. The sample python codes are as follows:

```
from surprise import SVD
from surprise import Dataset
from surprise import evaluate, print_perf
import os
from surprise import  Reader
```

```
#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)

data.split(n_folds=3)

algo = SVD()
perf = evaluate(algo, data, measures=['RMSE', 'MAE'])
print_perf(perf)
```

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. Notice an unbiased version of SVD is exactly the PMF algorithm. The sample python codes are as follows:

```
from surprise import SVD
from surprise import Dataset
from surprise import evaluate, print_perf
import os
from surprise import  Reader

#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)

data.split(n_folds=3)
algo = SVD(biased=False) #PMF
perf = evaluate(algo, data, measures=['RMSE', 'MAE'])
print_perf(perf)
```

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. The sample python codes are as follows:

```
from surprise import NMF
from surprise import Dataset
from surprise import evaluate, print_perf d
import os
from surprise import  Reader

#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)

data.split(n_folds=3)
algo = NMF()
perf = evaluate(algo, data, measures=['RMSE', 'MAE'])
print_perf(perf)
```

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. The sample python codes are as follows:

```
from surprise import KNNBasic
from surprise import Dataset
from surprise import evaluate, print_perf
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)

data.split(n_folds=3)
algo = KNNBasic(sim_options = {
            'user_based': True
            })
perf = evaluate(algo, data, measures=['RMSE', 'MAE'])
```

```
print_perf(perf)
```

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. Notice when choosing 'user_based': false, KNNBasic is an item based collaborative filter method. The sample python codes are as follows:

```
from surprise import KNNBasic
from surprise import Dataset
from surprise import evaluate, print_perf

import os
from surprise import  Reader

#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)

data.split(n_folds=3)
algo = KNNBasic(sim_options = {
            'user_based': False
            })
perf = evaluate(algo, data, measures=['RMSE', 'MAE'])
print_perf(perf)
```

10. Compare the performances of User-based collaborative filtering, item-based collaborative filtering, SVD, PMF, NMF on fold-1 with respect to RMSE and MAE.
    Since *data.split(n_folds=3)*randomly split the data into 3 folds, please make sure you test the five algorithms on the same fold-1 so the results are comparable.

11. Compare the performances of User-based collaborative filtering, item-based collaborative filtering, SVD, PMF, NMF on fold-2 with respect to RMSE and MAE. Please make sure you test the five algorithms on the same fold-2 so the results are comparable.

12. Compare the performances of User-based collaborative filtering, item-based collaborative filtering, SVD, PMF, NMF on fold-3 with respect to RMSE and MAE. Please make sure you test the five algorithms on the same fold-3 so the results are comparable.

13. Compare the **average (mean)** performances of User-based collaborative filtering, item-based collaborative filtering, SVD, PMF, NMF with respect to RMSE and MAE. Please make sure you test the five algorithms on the same 3-fold data split plan so the results are comparable.

14. Examine how the cosine, MSD (Mean Squared Difference), and Pearson similarities impact the performances of User based Collaborative Filtering and Item based Collaborative Filtering.

    To use MSD similarity in User based Collaborative Filtering, you can:

```
algo = KNNBasic(sim_options = {
                'name':'MSD',
            'user_based': True
                        })
```

    To use cosine similarity in User based Collaborative Filtering, you can:

```
algo = KNNBasic(sim_options = {
                'name':'cosine',
            'user_based': True
                        })
```

    To use Pearson similarity in User based Collaborative Filtering, you can:

```
algo = KNNBasic(sim_options = {
                'name':'pearson',
            'user_based': True
                        })
```

    Similarity, you can change the similarity metric setting in the codes of Item based Collaborative Filtering.

    Finally, is the impact of the three metrics on User based Collaborative Filtering consistent with the impact of the three metrics on Item based Collaborative Filtering? Plot your results.

15. Examine how the number of neighbors impacts the performances of User based Collaborative Filtering or Item based Collaborative Filtering? Plot your results.

For instance, you can specify the setting of K neighbors (K=20) in User based collaborative filtering as follows:
algo = KNNBasic(k=20, sim_options = {'name':'MSD', 'user_based': True })

For instance, you can specify the setting of K neighbors (K=20) in Item based collaborative filtering as follows:
algo = KNNBasic(k=20, sim_options = {'name':'MSD', 'user_based': False })

Identify the best K for User/Item based collaborative filtering **in terms of RMSE**. Is the the best K of User based collaborative filtering the same with the best K of Item based collaborative filtering?

Please submit a report (PDF or word) that includes a link to your code, your answers/results, and your explanations or interpretations (if any).