CMPE 256 HomeWork 1

In [1]:

```
import pandas as pd
from collections import Counter
from sklearn.metrics.pairwise import cosine_similarity
from surprise import SVD
import numpy as np
import surprise
from surprise import Reader, Dataset

from scipy import sparse
```

In [2]:

```
import os
import pandas as pd
import numpy as np
from surprise import Reader
from surprise import Dataset
from surprise.model_selection import KFold
from surprise.model_selection import cross_validate
from surprise import NormalPredictor
from surprise import KNNBasic
from surprise import KNNWithMeans
from surprise import KNNWithZScore
from surprise import KNNBaseline
from surprise import SVD
from surprise import SVDpp
from surprise import NMF
from surprise import SlopeOne
from surprise import CoClustering
from surprise.accuracy import rmse
from surprise import accuracy
from surprise.model_selection import train_test_split
from surprise.model_selection import GridSearchCV
from collections import defaultdict
```

1. Loading Training Data

In [3]:

```
df=pd.read_csv('train.dat', sep='\t', header=None)
df.columns=['user', 'movie','rating','timestamp']
```

In [4]:

```
df.head()
```

Out[4]:

	user	movie	rating	timestamp
0	905	470	1	889325071
1	697	1518	5	879835275
2	855	1687	5	875638677
3	950	1447	5	877420720
4	806	1170	4	879889337

In [5]:

```
df.shape
```

Out[5]:

```
(85724, 4)
```

2. Defining the rating scale and building the dataset

In [6]:

```
reader = Reader(rating_scale=(0.5, 5.0))
data_set = Dataset.load_from_df(df[['user', 'movie', 'rating']], reader)
```

3. Trying with basline algorithms

In [7]:

```
#Reference from: https://towardsdatascience.com/movie-recommender-system-part
results = []

for alg in [SVD(), NMF(), NormalPredictor(), KNNBasic()]:

    res = cross_validate(alg, data_set, measures=['RMSE'], cv=3, verbose=Fals
    tmp = pd.DataFrame.from_dict(res).mean(axis=0)
    tmp = tmp.append(pd.Series([str(alg).split(' ')[0].split('.')[-1]],index=
    results.append(tmp)
```

```
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.
```

In [8]:

```
surprise_results = pd.DataFrame(results).set_index('Algorithm').sort_values('
surprise_results
```

Out[8]:

	test_rmse	fit_time	test_time
Algorithm			
SVD	0.950236	2.684577	0.159854
NMF	0.987491	3.169755	0.148271
KNNBasic	0.996197	0.207396	2.897233
NormalPredictor	1.522411	0.088283	0.211689

3. GridSearchCV to find best parameters in SVD

In [ ]:

```
#Reference from: https://towardsdatascience.com/movie-recommender-system-part

param_grid = {'n_factors': [25, 30, 35, 40], 'n_epochs': [15, 20, 25], 'lr_al
              'reg_all': [0.08, 0.1, 0.15]}
gs = GridSearchCV(SVD, param_grid, measures=['rmse', 'mae'], cv=3)
gs.fit(data_set)
algo = gs.best_estimator['rmse']
print(gs.best_score['rmse'])
print(gs.best_params['rmse'])
```

In [9]:

```
## Previous best scores
## Score: 0.9301
##{'n_factors': 35, 'n_epochs': 30, 'lr_all': 0.008, 'reg_all': 0.05}
```

4. Loading test Data

In [10]:

```
df_test=pd.read_csv('test.dat', sep='\t', header=None)
df_test.columns=['user', 'movie']
df_test['rating']=0
```

In [11]:

```
df_test
```

Out[11]:

	user	movie	rating
0	158	951	0
1	521	1202	0
2	98	1556	0
3	292	1583	0
4	68	1064	0
...	...	...	...
2149	537	1414	0
2150	618	1448	0
2151	154	1519	0
2152	154	1429	0
2153	826	1602	0

2154 rows × 3 columns

In [12]:

```
# Building test set
reader = Reader(rating_scale=(0.5, 5.0))
data_test = Dataset.load_from_df(df_test[['user', 'movie','rating']], reader)
```

5. Training on the whole training data

In [13]:

```
train_data = data_set.build_full_trainset()

reg = SVD(n_factors=35, n_epochs=30, lr_all=0.008, reg_all=0.05)
reg.fit(train_data)
```

Out[13]:

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x1fc54038460>
```

In [14]:

```
data_set.df.to_numpy()
```

Out[14]:

```
array([[ 905,  470,    1],
       [ 697, 1518,    5],
       [ 855, 1687,    5],
       ...,
       [ 167, 1036,    3],
       [ 508, 1528,    3],
       [  76, 1586,    3]], dtype=int64)
```

In [15]:

```
#RMSE for training data
preds_train = reg.test(data_set.df.to_numpy())
accuracy.rmse(preds_train)
```

```
RMSE: 0.6743
```

Out[15]:

```
0.6743489755060518
```

6. Pre-processing testing data

In [16]:

```
testset = [data_test.df.loc[i].to_list() for i in range(len(data_test.df))]
```

7. Making predictions on pre-processed test.dat

In [17]:

```
predictions = reg.test(testset)
```

In [18]:

```
pred=[]
for prediction in predictions:
    pred.append(prediction[3])
```

In [19]:

```
len(pred)
```

Out[19]:

```
2154
```

In [20]:

```
submission_df=pd.DataFrame(pred)
```

In [22]:

```
submission_df.to_csv('submission2.dat', index=False, header=False)
```

In [23]:

```
submission=pd.read_csv('submission2.dat')
```

In [24]:

```
submission
```

Out[24]:

```
      2.9401309514158003
   0            3.958190
   1            3.172791
   2            4.085288
   3            3.391724
   4            4.451186
 ...                 ...
2148            3.874339
2149            2.787361
2150            4.150590
2151            3.701678
2152            4.998140
```

2153 rows × 1 columns