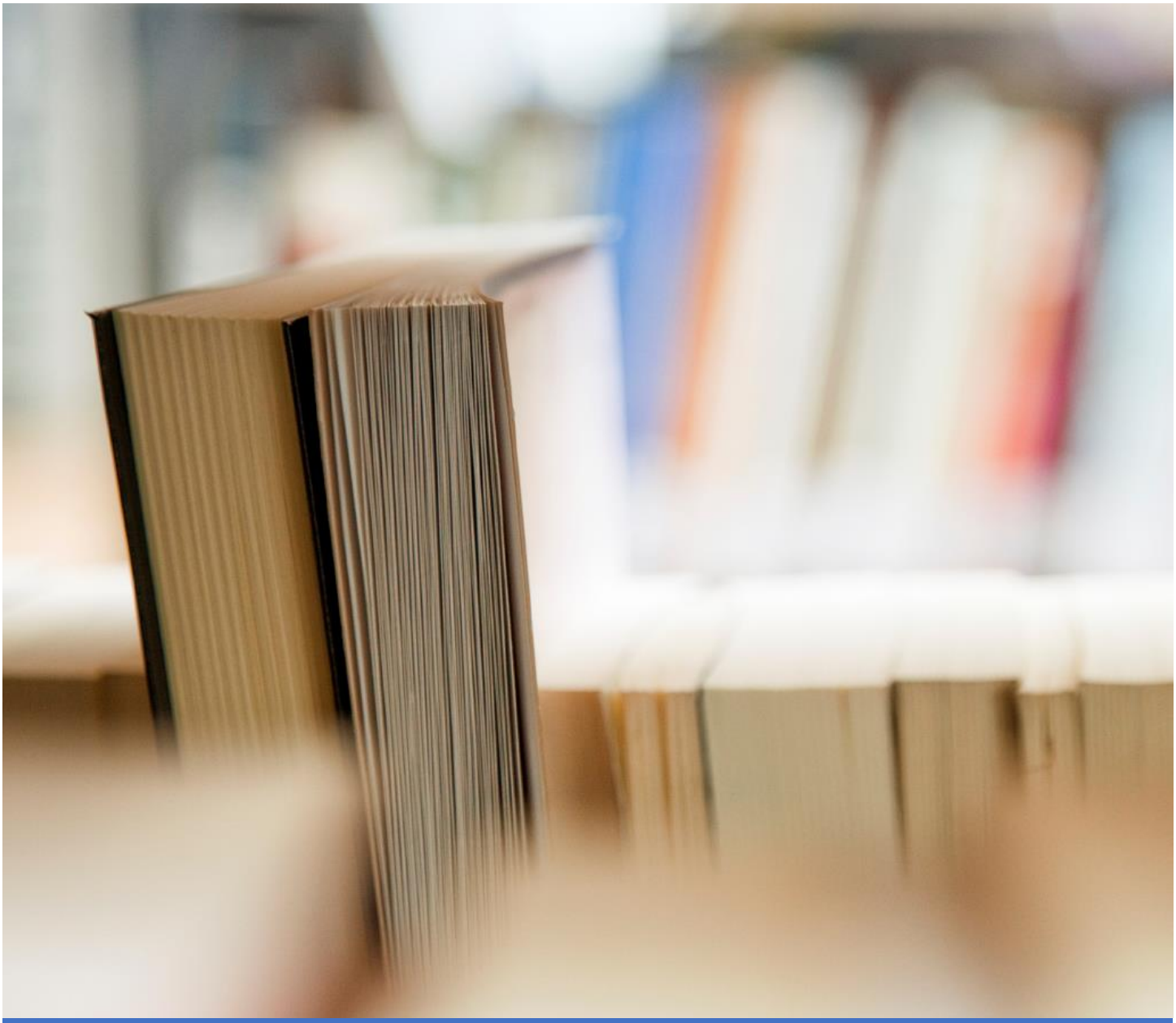


Recommendation Systems

Assignment-07

bhuvanesh Jeevarathinam 1/24/13 CAP5610

Bhuvanesh Jeevarathinam

Assignment -7

Recommendation System

5. MAE and RMSE of SVD for 3-FOLD Cross Validation:

```
from surprise import Dataset
from surprise.model_selection import cross_validate
from surprise import KNNBasic
from surprise.model_selection import KFold
from surprise import accuracy
```

Question 5 SVD

```
# Use the famous SVD algorithm.
algo = SVD()

# Run 5-fold cross-validation and print results.
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)    0.9483  0.9421  0.9458  0.9454  0.0025
MAE (testset)     0.7482  0.7424  0.7484  0.7463  0.0028
Fit time          4.47    5.35    3.75    4.52    0.65
Test time         0.33    0.21    0.21    0.25    0.06
{'test_rmse': array([0.94827469, 0.94212394, 0.94581167]),
 'test_mae': array([0.74816356, 0.74239496, 0.74840896]),
 'fit_time': (4.4664223194122314, 5.347970962524414, 3.7533814907073975),
 'test_time': (0.32790565490722656, 0.20703411102294922, 0.20955801010131836)}
```

6. MAE and RMSE of PMF for 3-FOLD Cross Validation:

Question - 6 - PMF (3 FOLD)

```
# Use the famous SVD algorithm.
algo = SVD(biased=False)#PMF

# Run 5-fold cross-validation and print results.
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)    0.9627  0.9694  0.9673  0.9665  0.0028
MAE (testset)     0.7613  0.7645  0.7613  0.7624  0.0015
Fit time          3.80    5.88    3.65    4.45    1.02
Test time         0.18    0.23    0.17    0.19    0.03
{'test_rmse': array([0.96271589, 0.96937961, 0.96728218]),
 'test_mae': array([0.76130527, 0.76447695, 0.76133344]),
 'fit_time': (3.8022966384887695, 5.88344407081604, 3.6520488262176514),
 'test_time': (0.17534875869750977, 0.22742009162902832, 0.1710805892944336)}
```

7. MAE and RMSE of NMF for 3-FOLD Cross Validation:

Question - 7 - NMF (3 FOLD)

```
# Use the famous SVD algorithm.
algo = NMF()

# Run 5-fold cross-validation and print results.
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)    0.9801  0.9698  0.9784  0.9761  0.0045
MAE (testset)     0.7715  0.7614  0.7697  0.7675  0.0044
Fit time          4.16    6.44    4.36    4.99    1.03
Test time         0.23    0.32    0.18    0.24    0.06
{'test_rmse': array([0.98006976, 0.96977295, 0.97842986]),
 'test_mae': array([0.77145743, 0.76136016, 0.76974538]),
 'fit_time': (4.15772008895874, 6.436312437057495, 4.364515066146851),
 'test_time': (0.22632122039794922, 0.32296276092529297, 0.18401646614074707)}
```

8. MAE and RMSE of User Based Collaborative Filtering for 3-FOLD Cross Validation:

Question - 8 - USER BASED COLLABORATIVE FILTERING ALGORITHM (3 FOLD)

```
# Use the famous SVD algorithm.
algo = KNNBasic(sim_options={
    'user_based':True
})

# Run 5-fold cross-validation and print results.
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)    0.9927  0.9876  0.9818  0.9874  0.0045
MAE (testset)     0.7834  0.7827  0.7768  0.7810  0.0030
Fit time          0.28    0.30    0.35    0.31    0.03
Test time         3.98    4.40    5.41    4.60    0.60
{'test_rmse': array([0.99273299, 0.98758317, 0.9817913 ]),
 'test_mae': array([0.78338307, 0.78270442, 0.77679327]),
 'fit_time': (0.28415513038635254, 0.2957644462585449, 0.349001407623291),
 'test_time': (3.9823553562164307, 4.398673057556152, 5.405828952789307)}
```

9. MAE and RMSE of Item Based Collaborative Filtering for 3-FOLD Cross Validation:

Question - 9 - ITEM BASED COLLABORATIVE FILTERING ALGORITHM (3 FOLD) ¶

```
# Use the famous SVD algorithm.
algo = KNNBasic(sim_options={
    'user_based':False
})

# Run 5-fold cross-validation and print results.
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)    0.9909  0.9783  0.9888  0.9860  0.0055
MAE (testset)     0.7851  0.7747  0.7828  0.7809  0.0044
Fit time          0.56    0.67    0.43    0.56    0.09
Test time         5.34    6.63    4.73    5.57    0.79
{'test_rmse': array([0.99089182, 0.97825996, 0.98883684]),
 'test_mae': array([0.78508949, 0.77474098, 0.78282787]),
 'fit_time': (0.563493013381958, 0.6671493053436279, 0.4348940849304199),
 'test_time': (5.338097333908081, 6.629499673843384, 4.732677221298218)}
```

Question - 10,11,12 -Performance of USER/ITEM,PMF,SVD,NMF on Each Fold Respectively

```
kf = KFold(n_splits=3)


i=0
for trainset, testset in kf.split(data):
    i+=1
    if(i==1):
        print("Fold - 1")
        print("*"*50)
        print("SVD")
        algo_svd = SVD()
        # train and test algorithm.
        algo_svd.fit(trainset)
        predictions_svd = algo_svd.test(testset)
        # Compute and print Root Mean Squared Error
        accuracy.rmse(predictions_svd, verbose=True)
        accuracy.mae(predictions_svd, verbose=True)
        print("*"*50)
        print("PMF")
        algo_PMF = SVD(biased=False)
        # train and test algorithm.
        algo_PMF.fit(trainset)
        predictions_PMF = algo_PMF.test(testset)
        # Compute and print Root Mean Squared Error
        accuracy.rmse(predictions_PMF, verbose=True)
        accuracy.mae(predictions_PMF, verbose=True)
        print("*"*50)
        print("NMF")
        algo_NMF = NMF(biased=False)
        # train and test algorithm.
        algo_NMF.fit(trainset)
        predictions_NMF = algo_NMF.test(testset)
        # Compute and print Root Mean Squared Error
        accuracy.rmse(predictions_NMF, verbose=True)
        accuracy.mae(predictions_NMF, verbose=True)
        print("*"*50)
        print("User Based Collaborative Filtering")
        algo_UC =  KNNBasic(sim_options={
        'user_based':True
        })
        # train and test algorithm.
        algo_UC.fit(trainset)
        predictions_UC = algo_UC.test(testset)
        # Compute and print Root Mean Squared Error
        accuracy.rmse(predictions_UC, verbose=True)
        accuracy.mae(predictions_UC, verbose=True)
        print("*"*50)
        print("Item Based Collaborative Filtering")
        algo_IC =  KNNBasic(sim_options={
        'user_based':False
        })
        # train and test algorithm.
        algo_IC.fit(trainset)
        predictions_IC = algo_IC.test(testset)
        # Compute and print Root Mean Squared Error
        accuracy.rmse(predictions_IC, verbose=True)
        accuracy.mae(predictions_IC, verbose=True)
        print("*"*50)
    print("#"*50)
    if(i==2):
```

```
    print("#"*50)
    if(i==2):
        print("Fold - 2")
        print("*"*50)
        print("SVD")
        algo_svd = SVD()
        # train and test algorithm.
        algo_svd.fit(trainset)
        predictions_svd = algo_svd.test(testset)
        # Compute and print Root Mean Squared Error
        accuracy.rmse(predictions_svd, verbose=True)
        accuracy.mae(predictions_svd, verbose=True)
        print("*"*50)
        print("PMF")
        algo_PMF = SVD(biased=False)
        # train and test algorithm.
        algo_PMF.fit(trainset)
        predictions_PMF = algo_PMF.test(testset)
        # Compute and print Root Mean Squared Error
        accuracy.rmse(predictions_PMF, verbose=True)
        accuracy.mae(predictions_PMF, verbose=True)
        print("*"*50)
        print("NMF")
        algo_NMF = NMF(biased=False)
        # train and test algorithm.
        algo_NMF.fit(trainset)
        predictions_NMF = algo_NMF.test(testset)
        # Compute and print Root Mean Squared Error
        accuracy.rmse(predictions_NMF, verbose=True)
        accuracy.mae(predictions_NMF, verbose=True)
        print("*"*50)
        print("User Based Collaborative Filtering")
        algo_UC =  KNNBasic(sim_options={
        'user_based':True
        })
        # train and test algorithm.
        algo_UC.fit(trainset)
        predictions_UC = algo_UC.test(testset)
        # Compute and print Root Mean Squared Error
        accuracy.rmse(predictions_UC, verbose=True)
        accuracy.mae(predictions_UC, verbose=True)
        print("*"*50)
        print("Item Based Collaborative Filtering")
        algo_IC =  KNNBasic(sim_options={
        'user_based':False
        })
        # train and test algorithm.
        algo_IC.fit(trainset)
        predictions_IC = algo_IC.test(testset)
        # Compute and print Root Mean Squared Error
        accuracy.rmse(predictions_IC, verbose=True)
        accuracy.mae(predictions_IC, verbose=True)
        print("*"*50)
    print("#"*50)
    if(i==3):
        print("Fold - 3")
        print("*"*50)
        print("SVD")
        algo_svd = SVD()
        # train and test algorithm.
        algo_svd.fit(trainset)
        predictions_svd = algo_svd.test(testset)
        # Compute and print Root Mean Squared Error
        accuracy.rmse(predictions_svd, verbose=True)
        accuracy.mae(predictions_svd, verbose=True)
```

```
    print("#"*50)
    if(i==3):
        print("Fold - 3")
        print("*"*50)
        print("SVD")
        algo_svd = SVD()
        # train and test algorithm.
        algo_svd.fit(trainset)
        predictions_svd = algo_svd.test(testset)
        # Compute and print Root Mean Squared Error
        accuracy.rmse(predictions_svd, verbose=True)
        accuracy.mae(predictions_svd, verbose=True)
        print("*"*50)
        print("PMF")
        algo_PMF = SVD(biased=False)
        # train and test algorithm.
        algo_PMF.fit(trainset)
        predictions_PMF = algo_PMF.test(testset)
        # Compute and print Root Mean Squared Error
        accuracy.rmse(predictions_PMF, verbose=True)
        accuracy.mae(predictions_PMF, verbose=True)
        print("*"*50)
        print("NMF")
        algo_NMF = NMF(biased=False)
        # train and test algorithm.
        algo_NMF.fit(trainset)
        predictions_NMF = algo_NMF.test(testset)
        # Compute and print Root Mean Squared Error
        accuracy.rmse(predictions_NMF, verbose=True)
        accuracy.mae(predictions_NMF, verbose=True)
        print("*"*50)
        print("User Based Collaborative Filtering")
        algo_UC =  KNNBasic(sim_options={
        'user_based':True
        })
        # train and test algorithm.
        algo_UC.fit(trainset)
        predictions_UC = algo_UC.test(testset)
        # Compute and print Root Mean Squared Error
        accuracy.rmse(predictions_UC, verbose=True)
        accuracy.mae(predictions_UC, verbose=True)
        print("*"*50)
        print("Item Based Collaborative Filtering")
        algo_IC =  KNNBasic(sim_options={
        'user_based':False
        })
        # train and test algorithm.
        algo_IC.fit(trainset)
        predictions_IC = algo_IC.test(testset)
        # Compute and print Root Mean Squared Error
        accuracy.rmse(predictions_IC, verbose=True)
        accuracy.mae(predictions_IC, verbose=True)
        print("*"*50)
```

```
Fold - 1
**************************************************
SVD
RMSE: 0.9484
MAE:  0.7474
**************************************************
PMF
RMSE: 0.9680
MAE:  0.7626
**************************************************
NMF
RMSE: 0.9785
MAE:  0.7662
**************************************************
User Based Collaborative Filtering
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9922
MAE:  0.7822
**************************************************
Item Based Collaborative Filtering
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9881
MAE:  0.7820
**************************************************
##################################################
##################################################
##################################################
Fold - 2
**************************************************
SVD
RMSE: 0.9365
MAE:  0.7407
**************************************************
PMF
RMSE: 0.9585
MAE:  0.7566
**************************************************
NMF
RMSE: 0.9677
MAE:  0.7618
**************************************************
User Based Collaborative Filtering
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9790
MAE:  0.7759
**************************************************
Item Based Collaborative Filtering
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9811
MAE:  0.7785
**************************************************
##################################################
--------------------------------------------------
##################################################
##################################################
##################################################
Fold - 3
**************************************************
SVD
RMSE: 0.9471
MAE:  0.7469
**************************************************
PMF
RMSE: 0.9656
MAE:  0.7610
**************************************************
NMF
RMSE: 0.9745
MAE:  0.7645
**************************************************
User Based Collaborative Filtering
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9915
MAE:  0.7829
**************************************************
Item Based Collaborative Filtering
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9858
MAE:  0.7802
**************************************************
```

Question - 13- average (mean) performances of User-based collaborative filtering, itembased collaborative filtering, SVD, PMF, NMF with respect to RMSE and MAE

```
algo = SVD()

# Run 5-fold cross-validation and print results.
svd=cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)

# Use the famous SVD algorithm.
algo = SVD(biased=False)#PMF

# Run 5-fold cross-validation and print results.
pmf=cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)
# Use the famous SVD algorithm.
algo = NMF()

# Run 5-fold cross-validation and print results.
nmf=cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)

# Use the famous SVD algorithm.
algo = KNNBasic(sim_options={
    'user_based':True
})

# Run 5-fold cross-validation and print results.
ub=cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)

# Use the famous SVD algorithm.
algo = KNNBasic(sim_options={
   'user_based':False
})

# Run 5-fold cross-validation and print results.
ib=cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)
print("Average mean Performance of SVD")
print("MAE"+str(svd['test_mae'].mean()))
print("RMSE"+str(svd['test_rmse'].mean()))
print("*"*25)
print("Average mean Performance of PMF")
print("MAE"+str(pmf['test_mae'].mean()))
print("RMSE"+str(pmf['test_rmse'].mean()))
print("*"*25)
print("Average mean Performance of NMF")
print("MAE"+str(nmf['test_mae'].mean()))
print("RMSE"+str(nmf['test_rmse'].mean()))
print("*"*25)
print("Average mean Performance of USER BASED")
print("MAE"+str(ub['test_mae'].mean()))
print("RMSE"+str(ub['test_rmse'].mean()))
print("*"*25)
print("Average mean Performance of ITEM BASED")
print("MAE"+str(ib['test_mae'].mean()))
print("RMSE"+str(ib['test_rmse'].mean()))
print("*"*25)
```

```
                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.9491  0.9406  0.9476  0.9457  0.0037
MAE (testset)     0.7462  0.7437  0.7478  0.7459  0.0017
Fit time          7.47    9.08    8.49    8.35    0.67
Test time         0.57    0.57    0.55    0.57    0.01
Evaluating RMSE, MAE of algorithm SVD on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.9657  0.9639  0.9660  0.9652  0.0009
MAE (testset)     0.7605  0.7609  0.7611  0.7608  0.0003
Fit time          7.13    8.24    6.96    7.44    0.57
Test time         0.48    0.40    0.50    0.46    0.04
Evaluating RMSE, MAE of algorithm NMF on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.9761  0.9777  0.9750  0.9763  0.0011
MAE (testset)     0.7683  0.7661  0.7653  0.7666  0.0013
Fit time          8.52    7.74    9.49    8.59    0.72
Test time         0.39    0.35    0.66    0.47    0.14
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)    0.9869  0.9847  0.9954  0.9890  0.0046
MAE (testset)     0.7773  0.7804  0.7881  0.7819  0.0045
Fit time          0.63    0.74    0.31    0.56    0.18
Test time         8.09    8.22    7.20    7.84    0.45
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)    0.9875  0.9865  0.9826  0.9856  0.0021
MAE (testset)     0.7832  0.7807  0.7769  0.7803  0.0026
Fit time          0.95    1.01    1.03    1.00    0.03
Test time         11.13   11.00   11.09   11.07   0.06
Average mean Performance of SVD
MAE0.7458800683319114
RMSE0.945745632164695
*************************
Average mean Performance of PMF
MAE0.7608405444448696
RMSE0.9651873450816737
*************************
Average mean Performance of NMF
MAE0.766571164155646
RMSE0.9762763748103896
*************************
Average mean Performance of USER BASED
MAE0.7819358489865925
RMSE0.9890312816788134
*************************
Average mean Performance of ITEM BASED
MAE0.7802742147996753
RMSE0.985551921114856
*************************
```

Question - 14- MSD,COSINE,PEARSON impact on User and ITEM Based Collaborative Filtering

User Based

```
print("User based Colloborative Filtering")
print("*"*25)
print("MSD")
print("*"*25)
algo = KNNBasic(sim_options = {
        'name': 'MSD',
        'user_based': True
        })
User_based_value_MSD=cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)

#User-based collaborative filtering. Cosine

print("*"*25)
print("Cosine")
print("*"*25)
algo = KNNBasic(sim_options = {
        'name': 'cosine',
        'user_based': True
        })
User_based_value_Cosine=cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)
#User-based collaborative filtering. Pearson.

print("*"*25)
print("Pearson")
print("*"*25)
algo = KNNBasic(sim_options = {
        'name': 'pearson',
        'user_based': True
        })
User_based_values_Pearson=cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)
```

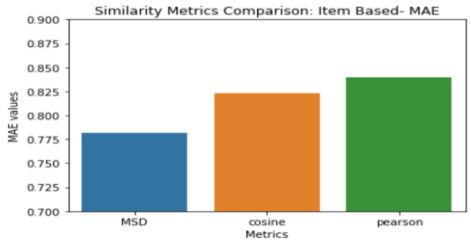
```
User based Colloborative Filtering
*************************
MSD
*************************
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Evaluating RMSE, MAE of algorithm KNNBasic on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.9888  0.9880  0.9879  0.9883  0.0004
MAE (testset)     0.7809  0.7801  0.7819  0.7810  0.0007
Fit time          0.29    0.30    0.34    0.31    0.02
Test time         3.81    4.34    4.54    4.23    0.31
*************************
Cosine
*************************
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine 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)    1.0264  1.0171  1.0208  1.0214  0.0038
MAE (testset)     0.8138  0.8051  0.8067  0.8085  0.0038
Fit time          1.07    1.07    1.13    1.09    0.03
Test time         4.05    4.17    4.83    4.35    0.35
*************************
Pearson
*************************
Computing the pearson similarity matrix...
Done computing similarity matrix.
Computing the pearson similarity matrix...
Done computing similarity matrix.
Computing the pearson 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)    1.0182  1.0225  1.0214  1.0207  0.0018
MAE (testset)     0.8082  0.8108  0.8098  0.8096  0.0011
Fit time          1.60    1.63    2.22    1.82    0.29
Test time         3.93    6.11    4.76    4.93    0.90
```

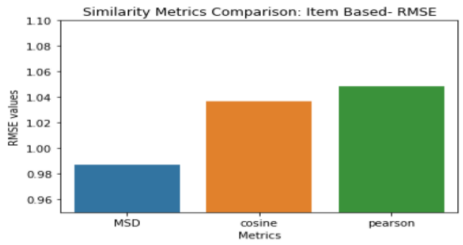
Item Based

```
print("Item based Colloborative Filtering")
print("*"*25)
print("MSD")
print("*"*25)
algo = KNNBasic(sim_options = {
        'name': 'MSD',
        'user_based': False
        })
item_based_values_MSD=cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)

#User-based collaborative filtering. Cosine

print("*"*25)
print("Cosine")
print("*"*25)
algo = KNNBasic(sim_options = {
        'name': 'cosine',
        'user_based': False
        })
item_based_values_cosine=cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)#User-based collaborative filtering. Pearson.

print("*"*25)
print("Pearson")
print("*"*25)
algo = KNNBasic(sim_options = {
        'name': 'pearson',
        'user_based': False
        })
item_based_values_pearson=cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)
```

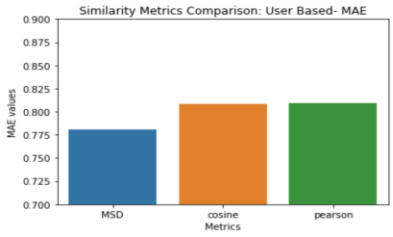
```
Item based Colloborative Filtering
*************************
MSD
*************************
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Evaluating RMSE, MAE of algorithm KNNBasic on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.9846  0.9851  0.9910  0.9869  0.0029
MAE (testset)     0.7777  0.7816  0.7844  0.7812  0.0027
Fit time          0.59    0.48    0.44    0.50    0.06
Test time         5.59    6.89    5.56    6.01    0.62
*************************
Cosine
*************************
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine 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)    1.0400  1.0382  1.0321  1.0368  0.0034
MAE (testset)     0.8275  0.8208  0.8212  0.8232  0.0031
Fit time          2.73    1.88    3.55    2.72    0.68
Test time         7.23    5.05    6.31    6.20    0.89
*************************
Pearson
*************************
Computing the pearson similarity matrix...
Done computing similarity matrix.
Computing the pearson similarity matrix...
Done computing similarity matrix.
Computing the pearson 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)    1.0508  1.0462  1.0490  1.0487  0.0019
MAE (testset)     0.8404  0.8398  0.8380  0.8394  0.0010
Fit time          2.50    2.83    2.28    2.53    0.22
Test time         6.70    4.54    6.79    6.01    1.04
```

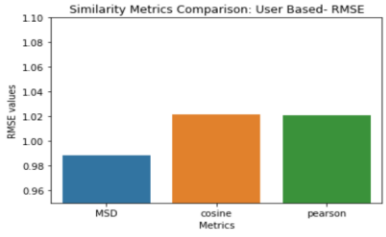
```
metrics = np.array(['MSD', 'cosine', 'pearson'])
values=[item_based_values_MSD['test_mae'].mean(), item_based_values_cosine['test_mae'].mean(), item_based_values_pearson['test_mae'].mean()]
series = pd.Series(name='MAE', data=values)
ax = sns.barplot(metrics, series.values)
ax.set_title("Similarity Metrics Comparison: Item Based- MAE")
ax.set_ylabel('MAE values')
ax.set_xlabel('Metrics')
ax.set_ylim(.7, 0.9)
plt.show()
metrics = np.array(['MSD', 'cosine', 'pearson'])
values=[item_based_values_MSD['test_rmse'].mean(), item_based_values_cosine['test_rmse'].mean(), item_based_values_pearson['test_rmse'].mean()]
series = pd.Series(name='RMSE', data=values)
ax = sns.barplot(metrics, series.values)
ax.set_title("Similarity Metrics Comparison: Item Based- RMSE")
ax.set_ylabel('RMSE values')
ax.set_xlabel('Metrics')
ax.set_ylim(.95, 1.1)
plt.show()
```






```
metrics = np.array(['MSD', 'cosine', 'pearson'])
values=[User_based_value_MSD['test_mae'].mean(), User_based_value_Cosine['test_mae'].mean(), User_based_values_Pearson['test_mae'].mean()]
series = pd.Series(name='MAE', data=values)
ax = sns.barplot(metrics, series.values)
ax.set_title("Similarity Metrics Comparison: User Based- MAE")
ax.set_ylabel('MAE values')
ax.set_xlabel('Metrics')
ax.set_ylim(.7, 0.9)
plt.show()
metrics = np.array(['MSD', 'cosine', 'pearson'])
values=[User_based_value_MSD['test_rmse'].mean(), User_based_value_Cosine['test_rmse'].mean(), User_based_values_Pearson['test_rmse'].mean()]
series = pd.Series(name='RMSE', data=values)
ax = sns.barplot(metrics, series.values)
ax.set_title("Similarity Metrics Comparison: User Based- RMSE")
ax.set_ylabel('RMSE values')
ax.set_xlabel('Metrics')
ax.set_ylim(.95, 1.1)
plt.show()
```



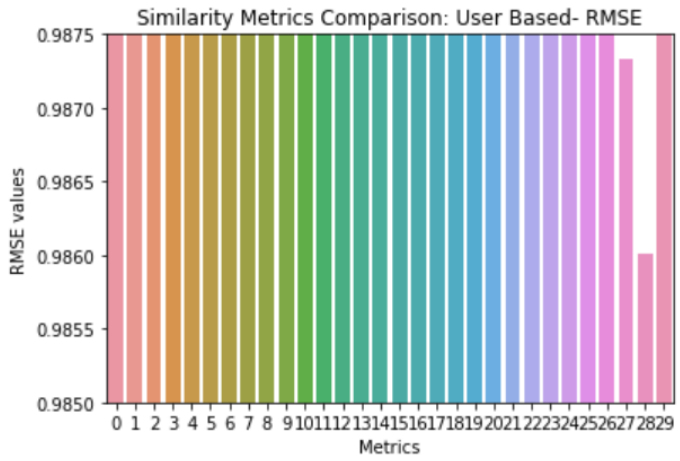


15. Impact of neighbors on Performance of User based and Item Based:

USER BASED:

```
print("User based Colloborative Filtering")
print("*"*25)
mean_rmse=[]
for i in range(0,30):
    print("*"*25)
    print("K value is "+str(i))
    print("*"*25)
    algo = KNNBasic( k=i,sim_options = {
            'name': 'MSD',
            'user_based': True
            })
    User_based_value_k=cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)
    mean_rmse.append(User_based_value_k['test_rmse'].mean())
```

```
K = [i for i in range(30)]
series = pd.Series(name='RMSE', data=mean_rmse)
ax = sns.barplot(K, series.values)
ax.set_title("Similarity Metrics Comparison: User Based- RMSE")
ax.set_ylabel('RMSE values')
ax.set_xlabel('Metrics')
ax.set_ylim(.985, 0.9875)
plt.show()
```



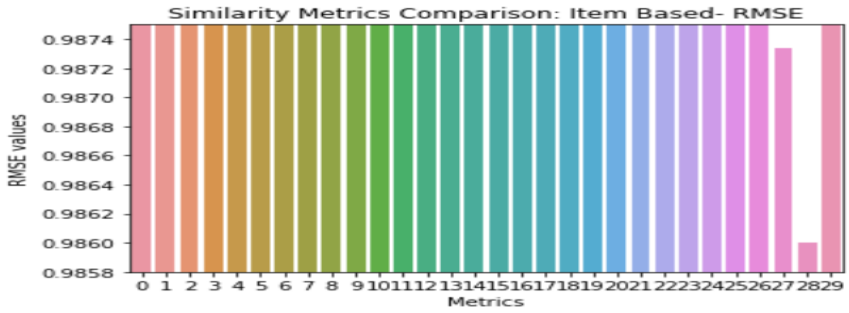
RMSE for Different K values from 1-30 is found. It is found that k=28 has least Error.

K=28 is best for User Based Filtering.

ITEM Based:

```
print("Item based Colloborative Filtering")
print("*"*25)
mean_rmse=[]
for i in range(0,30):
    print("*"*25)
    print("K value is "+str(i))
    print("*"*25)
    algo = KNNBasic( k=i,sim_options = {
            'name': 'MSD',
            'user_based': False
            })
    Item_based_value_k=cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True)
    mean_rmse.append(Item_based_value_k['test_rmse'].mean())
```

```
K = [i for i in range(30)]
series = pd.Series(name='RMSE', data=mean_rmse)
ax = sns.barplot(K, series.values)
ax.set_title("Similarity Metrics Comparison: Item Based- RMSE")
ax.set_ylabel('RMSE values')
ax.set_xlabel('Metrics')
ax.set_ylim(.9858, 0.9875)
plt.show()
```



- RMSE for Different K values from 1-30 is found. It is found that k=28 has least Error.
- K=28 is best for Item Based Filtering.
- Hence the best K =28 of User based collaborative filtering the same with the best K=28 of Item based collaborative filtering.

Code Link:

https://github.com/bhuvaneshkj/CAP5610-MachineLearning-Assignment