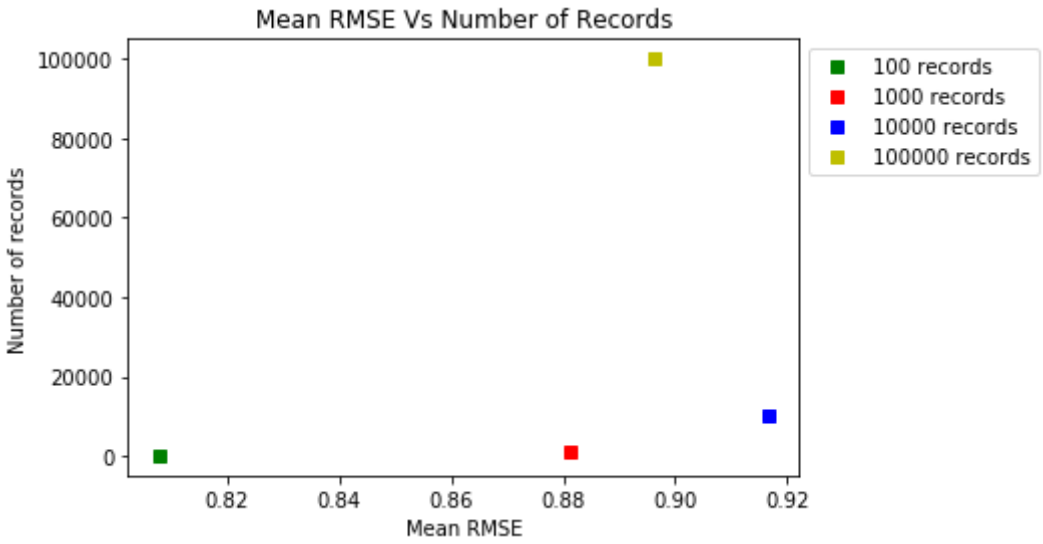
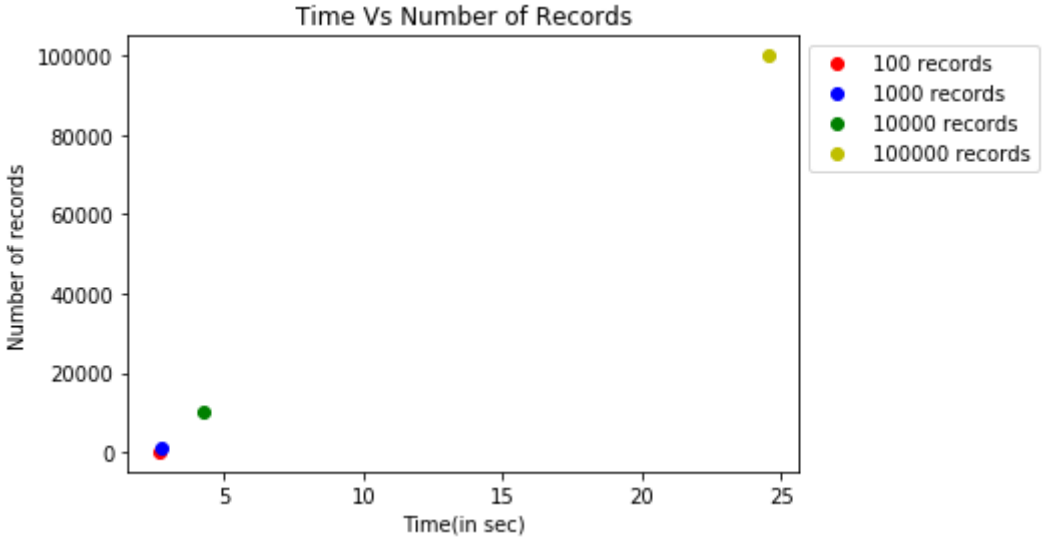
```
Python 3.6.1 |Anaconda custom (64-bit)| (default, May 11 2017, 13:25:24) [MSC v.1900 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 5.3.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

Restarting kernel...
```

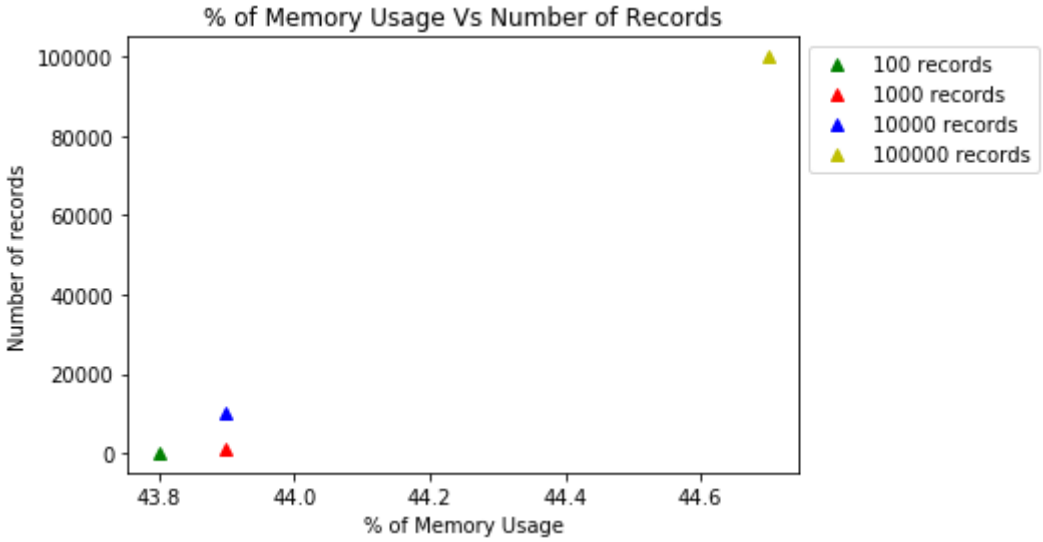
```
In [1]: import numpy as np
   ...: import pandas as pd
   ...: from surprise import SVD
   ...: from surprise import Dataset
   ...: from surprise.model_selection import cross_validate
   ...: from surprise import Reader
   ...: import time
   ...: import psutil
   ...: import matplotlib.pyplot as plt
   ...:
   ...:
   ...:
   ...: x=[]
   ...: timex=[]
   ...: mem=[]
   ...: m1=psutil.virtual_memory().percent
   ...: #print(m1)
   ...:
   ...: #For 100 record dataset
   ...: start = time.time()
   ...: df1 = pd.read_csv('C:/Users/dell pc/Desktop/Project/ratings_1million1.csv', dtype=
{'rating': float})
   ...: reader = Reader(rating_scale=(1, 5))
   ...: data = Dataset.load_from_df(df1[['user_id','book_id','rating']], reader)
   ...: algo = SVD()
   ...: result1=cross_validate(algo, data, measures=['RMSE'], cv=5, verbose=True)
   ...: #print(result1)
   ...: x.append(np.mean(result1['test_rmse']))
   ...: end = time.time()
   ...: #print("Time1",end - start)
   ...: timex.append(end-start)
   ...: #process=psutil.Process(os.getpid())
   ...: m2=psutil.virtual_memory().percent
   ...: #print(m2)
   ...: mem.append(m2)
   ...:
   ...:
   ...: #For 1000 record dataset
   ...: start = time.time()
   ...: df2 = pd.read_csv('C:/Users/dell pc/Desktop/Project/ratings_1million2.csv', dtype=
{'rating': float})
   ...: data = Dataset.load_from_df(df2[['user_id','book_id','rating']], reader)
   ...: result2=cross_validate(algo, data, measures=['RMSE'], cv=5, verbose=True)
   ...: #print(result2)
   ...: x.append(np.mean(result2['test_rmse']))
   ...: end = time.time()
   ...: #print("Time2",end - start)
   ...: timex.append(end-start)
   ...: #process=psutil.Process(os.getpid())
   ...: m3=psutil.virtual_memory().percent
   ...: #print(m3)
   ...: mem.append(m3)
   ...:
   ...:
```

```
   ...: #For 10000 record dataset
   ...: start = time.time()
   ...: df3 = pd.read_csv('C:/Users/dell pc/Desktop/Project/ratings_1million3.csv', dtype=
{'rating': float})
   ...: data = Dataset.load_from_df(df3[['user_id','book_id','rating']], reader)
   ...: result3=cross_validate(algo, data, measures=['RMSE'], cv=5, verbose=True)
   ...: #print(result3)
   ...: x.append(np.mean(result3['test_rmse']))
   ...: end = time.time()
   ...: #print("Time3",end - start)
   ...: timex.append(end-start)
   ...: #process=psutil.Process(os.getpid())
   ...: m4=psutil.virtual_memory().percent
   ...: #print(m4)
   ...: mem.append(m4)
   ...:
   ...: #For 100000 record dataset
   ...: start = time.time()
   ...: df4 = pd.read_csv('C:/Users/dell pc/Desktop/Project/ratings_1million4.csv', dtype=
{'rating': float})
   ...: data = Dataset.load_from_df(df4[['user_id','book_id','rating']], reader)
   ...: result4=cross_validate(algo, data, measures=['RMSE'], cv=5, verbose=True)
   ...: #print(result4)
   ...: x.append(np.mean(result4['test_rmse']))
   ...: end = time.time()
   ...: #print("Time4",end - start)
   ...: timex.append(end-start)
   ...: #process=psutil.Process(os.getpid())
   ...: m5=psutil.virtual_memory().percent
   ...: #print(m5)
   ...: mem.append(m5)
   ...:
   ...: #Plotting the Mean RMSE Vs Number of Records
   ...: y = [len(df1),len(df2),len(df3),len(df4)]
   ...: plt.plot( x[0],y[0],'gs',label='100 records')
   ...: plt.plot( x[1],y[1],'rs',label='1000 records')
   ...: plt.plot( x[2],y[2],'bs',label='10000 records')
   ...: plt.plot( x[3],y[3],'ys',label='100000 records')
   ...: legend = plt.legend(loc='upper left',bbox_to_anchor=(1, 1))
   ...: frame = legend.get_frame()
   ...: plt.xlabel('Mean RMSE')
   ...: plt.ylabel('Number of records')
   ...: plt.title('Mean RMSE Vs Number of Records')
   ...: plt.show()
   ...:
   ...:
   ...: #Plotting the Time Vs Number of Records
   ...: y = [len(df1),len(df2),len(df3),len(df4)]
   ...: plt.plot( timex[0],y[0],'ro',label='100 records')
   ...: plt.plot( timex[1], y[1],'bo',label='1000 records')
   ...: plt.plot( timex[2],y[2],'go',label='10000 records')
   ...: plt.plot( timex[3], y[3],'yo',label='100000 records')
   ...: legend = plt.legend(loc='upper left',bbox_to_anchor=(1, 1))
   ...: frame = legend.get_frame()
   ...: plt.xlabel('Time(in sec)')
   ...: plt.ylabel('Number of records')
   ...: plt.title('Time Vs Number of Records')
   ...: plt.show()
   ...:
   ...: #Plotting the % of Memory Usage Vs Number of Records
   ...: y = [len(df1),len(df2),len(df3),len(df4)]
   ...: plt.plot( mem[0],y[0],'g^',label='100 records')
   ...: plt.plot( mem[1],y[1],'r^',label='1000 records')
   ...: plt.plot( mem[2],y[2],'b^',label='10000 records')
   ...: plt.plot( mem[3],y[3],'y^',label='100000 records')
   ...: legend = plt.legend(loc='upper left',bbox_to_anchor=(1, 1))
   ...: frame = legend.get_frame()
   ...: plt.xlabel('% of Memory Usage')
   ...: plt.ylabel('Number of records')
```

```
   ...: plt.title('% of Memory Usage Vs Number of Records')
   ...: plt.show()
Evaluating RMSE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.7749  0.6715  0.8874  0.8856  0.8195  0.8078  0.0802
Fit time          0.01    0.01    0.01    0.01    0.01    0.01    0.00
Test time         0.00    0.00    0.00    0.00    0.00    0.00    0.00
Evaluating RMSE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8039  0.9177  0.8928  0.8993  0.8934  0.8814  0.0398
Fit time          0.10    0.08    0.09    0.09    0.08    0.09    0.01
Test time         0.00    0.00    0.00    0.00    0.00    0.00    0.00
Evaluating RMSE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9212  0.9123  0.9216  0.9081  0.9203  0.9167  0.0055
Fit time          1.05    1.08    1.05    0.93    0.93    1.01    0.07
Test time         0.05    0.04    0.04    0.03    0.03    0.04    0.01
Evaluating RMSE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8908  0.8982  0.8991  0.8924  0.9005  0.8962  0.0039
Fit time          13.23   13.35   13.32   13.07   8.25    12.25   2.00
Test time         0.51    0.57    0.51    0.34    0.29    0.45    0.11
```






% of Memory Usage Vs Number of Records
100000
80000
60000
40000
20000
0
Number of records
43.8
44.0
44.2
44.4
44.6
% of Memory Usage
100 records
1000 records
10000 records
100000 records

In [2]: