A PROJECT REPORT

On

"Comparative Study of Different Collaborative Filter based Recommendation Engines"

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirements for the Award of

BACHELOR'S DEGREE IN

COMPUTER SCIENCE AND COMMUNICATION ENGINEERING

BY

RIDDHIMAN KUMAR (1729052)

DEBASRITO LAHIRI (1729122)

HRITAM JAISWAL (1729130)

SATVIK TIWARI (1729155)

UNDER THE GUIDANCE OF

PROF. RAJDEEP CHATTERJEE

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAR, ODISHA – 751024

December 2020

KALINGA INSTITUTE
OF INDUSTRIAL TECHNOLOGY
Deemed to be University U/S 3 of the UGC Act, 1956

CERTIFICATE

This is to certify that the project entitled

"Comparative Study of Different Collaborative Filter based Recommendation Engines"

Submitted by:

RIDDHIMAN KUMAR (1729052)

DEBASRITO LAHIRI (1729122)

HRITAM JAISWAL (1729130)

SATVIK TIWARI (1729155)

In partial fulfilment of the requirements for the award of the **Degree of Bachelor of Technology** in **Computer Science and Communication Engineering** is a bonafide record of the work carried out under my guidance and supervision at School of Computer Engineering, Kalinga Institute of Industrial Technology, Deemed to be University.

PROF. RAJDEEP CHATTERJEE

………..

The project was evaluated by us on:

EXAMINER'S NAME:

SIGNATURE:

ACKNOWLEDGEMENT

We take this opportunity to express our deep gratitude to all those helping hands without whom this project would have not been what it is. We take immense pleasure to express our thankfulness to our mentor, Prof. Rajdeep Chatterjee for his constant motivation and timely suggestions which helped us sail smooth through the odds we faced in our project. I would like to thank our Dean, Prof. Amulya Ratna Swain for giving us this opportunity to enhance our skills by giving a chance for this project and for motivating us throughout the B. Tech. course and also thank the complete faculty of Computer Science and Engineering who have provided us the knowledge to successfully complete the project. Last but not the least we would like to thank our friends who provided their helping hands in our project as and when required and by providing us with valuable feedback from time to time.

RIDDHIMAN KUMAR (1729052)

DEBASRITO LAHIRI (1729122)

HRITAM JAISWAL (1729130)

SATVIK TIWARI (1729155)

ABSTRACT

Recommendation Engines have been in use since a long time for predicting content for users. Common uses include YouTube, Amazon, Netflix etc. Based on past user browsing history and reviews of users, new content is recommended to the user based on their personal taste and overall review of the contents.

It is of various types, the most common being:

a. Content based filtering
b. Collaborative filter based

Here, we will compare different techniques of collaborative filtering based recommendation engines. The types of collaborative filtering methods we compare here are:

a. Memory based
b. Matrix Factorization

We will construct the different models and train them on the dataset. Then we will try to predict the user ratings for the remaining data in the dataset.

We will be using K-nearest neighbours for the memory based recommendation engine and singular value decomposition for the matrix factorization method.

The dataset we use is ml-100k provided by GroupLens.

Finally, we will check the RMSE, MSE and MAE for the prediction model.

CONTENT

1. **Introduction…………………………………………………………………....6-7**
    1.1. Organization of study……………………………………………………..6
    1.2. Problem Statement……………………………………………………...…6
    1.3. Objectives……………………………………………………………….….7
    1.4. Proposed project overview……………………………………………...7
2. **Basic Concepts/Libraries Used……………………………………………...7-11**
    2.1. Recommendation enines……………………………………………..……7
    2.2. Collaborative filtering……………………………………………….…...8-9
    2.3. Numpy……………………………………………………………………....…9
    2.4. Pandas..…………………………………………………………………......9
    2.5. Surprise……………………………………………………………………...9
    2.6. Cosine Similarity matrix……………………………………………….…9-10
    2.7. K-nearest neighbours……………………………………………….….…10
    2.8. SVD………………………………………………………………………...….10
    2.9. Memory based collaborative filtering…………………………….…..11
    2.10. Model based collaborative filtering………………………………..…12
    2.11. GridSearchCV……………………………………………………………..12
    2.12. Performance metrics……………………………………………………….12-13
3. **Project breakdown…………………………………………………………......13**
4. **Environment Setup………………………………………………………….….14**
5. **Dataset…………………………………………………………………………….14**
6. **Implementation and related code……………………………………………14-17**
7. **Test results of the project……………………………………………………...17**
8. **Conclusion……………………………………………………………………….18**
9. **References………………………………………………………………………..19**

1. Introduction

1.1. Organization of study:

1. Chapter 1 introduces us to the need of the project, it’s uses and intended users. It also gives a brief overview of the project.
2. Chapter 2 gives ideas of the libraries, concepts and various methods used to do our project.
3. Chapter 3 gives a step by step breakdown of the project implementation.
4. Chapter 4 shows the environment setups needed.
5. Chapter 5 gives some details about the dataset used for the project.
6. Chapter 6 shows the implementation of the mentioned concepts to build the proposed models and the related codebase.
7. Chapter 7 shows a test run of the model and it’s output and comparison between the models.
8. Chapter 8 gives a final note on our project work.
9. Chapter 9 gives a reference of all the materials that we used during the creation of this project.

1.2. Problem Statement:

Recommendation engines are widely in use now in different services provided by different companies. There are different types of recommendation engines which are applicable for different scenarios. Collaborative filtering based are one of the most common types. Such recommendation engines can be implemented using various algorithms with varying accuracy.

Here, we will compare the performance of Memory based and matrix factorization based collaborative filters for their performances.

1.3. Objectives:

The main objective of this project is to train a memory based and a matrix factorization based collaborative filter on the same dataset and compare their performances in giving correct recommendations.

1.4. Proposed project overview:

Our project will include training 2 different recommendation engine models on a part of the same dataset and predicting user ratings on the other part of it and comparing the predicted and actual ratings. The 2 recommendation engines are memory based and matrix factorization based collaborative filters.

2. Basic concepts/libraries used:

2.1. : Recommendation Engines

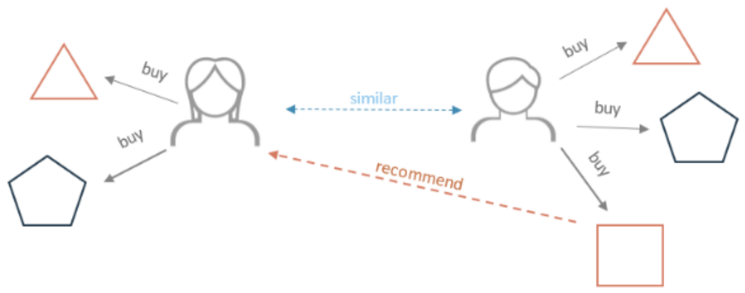
Recommendation Systems are systems that try to predict the rating or preference an user would give to a certain item. It considers previous choices of the user to recommend new choices the user can make. It also may consider the collective choices of multiple users to give suggestions to an user. Based on the method of functioning it is mainly of two types:

i. Content based
ii. Collaborative filter based
iii. Multi-Criteria Recommender System
iv. Hybrid Recommender Systems

Here we will work with collaborative filter based type.

2.2. Collaborative filtering:

Collaborative Filtering is a recommendation system technique that collaborates past history of many users to give new recommendations to a specific user. The underlying assumption here is that if a person *A* has the same opinion as a person *B* on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person.



Collaborative filtering suffers from 7 main problems which are:

i. Cold Start: When a new user or item does not have enough data for accurate recommendations.
ii. Scalability: Generally in places where such recommendation engines are used, the datasets are very large.
iii. Sparsity: The number of items is often very large and so users normally would have rated only a small subset of all the items. So, even the most popular item may get ignored during recommendation due to lack of ratings.
iv. Synonyms: Similar items may be named differently and thus treated as different entities by the engine.
v. Gray Sheep: These are users whose preference do not consistently agree or disagree with any set of users and thus do not benefit from collaborative filtering.
vi. Shilling attacks: When the ratings given by users are biased due to competition between 2 different items. For example, when users give a positive rating for their own items and a negative rating for their competitor's items.
vii. Diversity: Recommendation engines are supposed to increase the diversity of product recommendations. However, sometimes such engines recommend different users the same

set of items based on positive feedback all the while ignoring another set of items which might have a slight negative feedback. This creates a bias in the recommendations as the positive recommendations keep getting more positive ratings while the negative ones get ignored.

2.3. Numpy:

Numpy is a python library that is used for numerical computations in python. It stands for Numerical python. It allows us to use large multidimensional arrays and do complex mathematical operations.

2.4. Pandas:

Pandas is also a python library that has various data structures and related operations for operating on numerical data and time series.

2.5. Surprise:

It is a python scikit library used for creating recommendation systems and work on explicit rating data provided by users. It is unable to perform content based recommendations. It also contains many inbuilt datasets and various validation tools.

2.6. Cosine Similarity Matrix:

Cosine similarity is a mathematical method to find the similarity between 2 items. It is the angle in between the spatial distribution vectors of the items and is given by:

$$\cos\theta = \frac{\vec{A}.\vec{B}}{\|\vec{A}\|\|\vec{B}\|}$$

Where,

$\vec{A}\ and\ \vec{B}\ =$Vectors representing the 2 items and are a collection of their numerical features

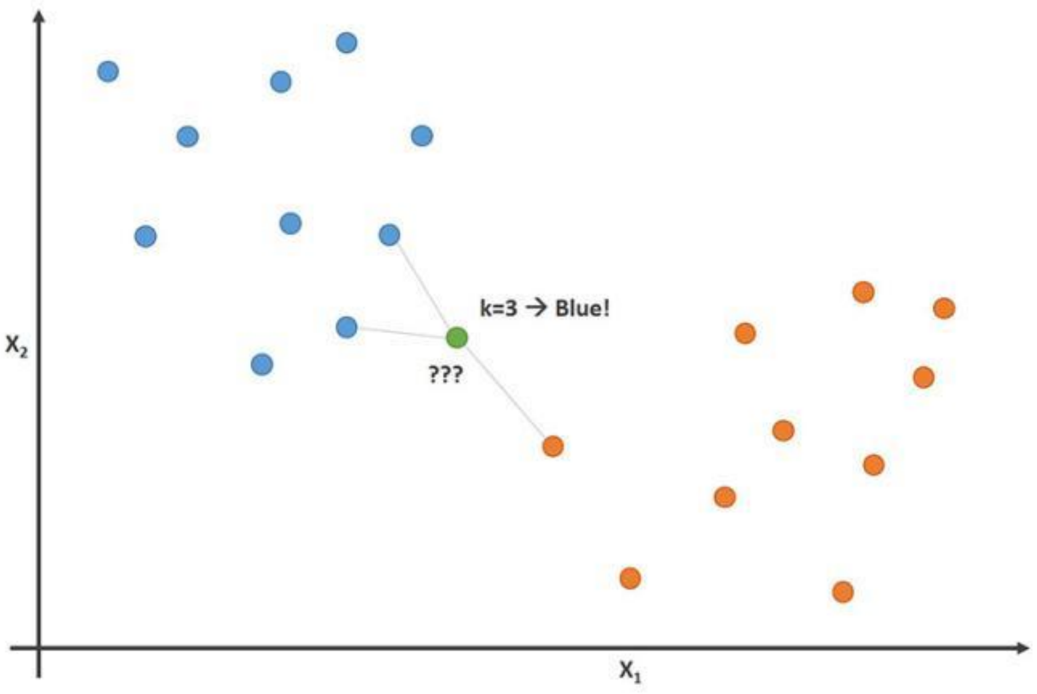
$\theta =$ The angle between the vectors.

The less the value of the angle the more similar the items are.

A matrix representing all such cosine similarities between all the items with respect to each other is called a cosine similarity matrix.

2.7. K-Nearest Neighbours:

It is a supervised machine learning algorithm used for classification and regression problems. It plots the data points on a graph and classifies it into labels. Then, when a new data point is inserted it finds the nearest K number of data points to it and based on the label the majority of the points have, the new data is classified into a label.



2.8. SVD:

SVD stands for single value decomposition. It is a decomposition of a rectangular matrix A of dimensions mxn where,

$$A = USV^T$$

Here,

A= The original mxn matrix

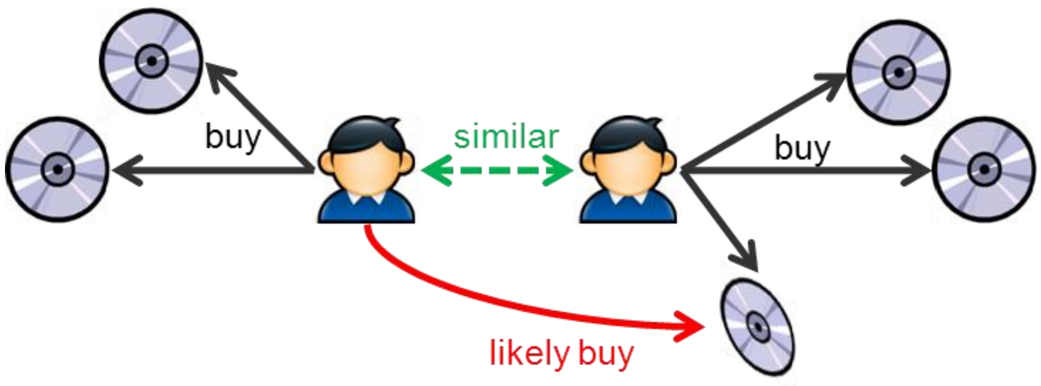
U= an mxm orthogonal matrix with eigenvectors of AA^T

S= an mxn diagonal matrix with eigenvalues of both $A^T A \ and \ AA^T$

V=an nxn orthogonal matrix with eigenvectors of $A^T A$

2.9. Memory Based collaborative filtering:

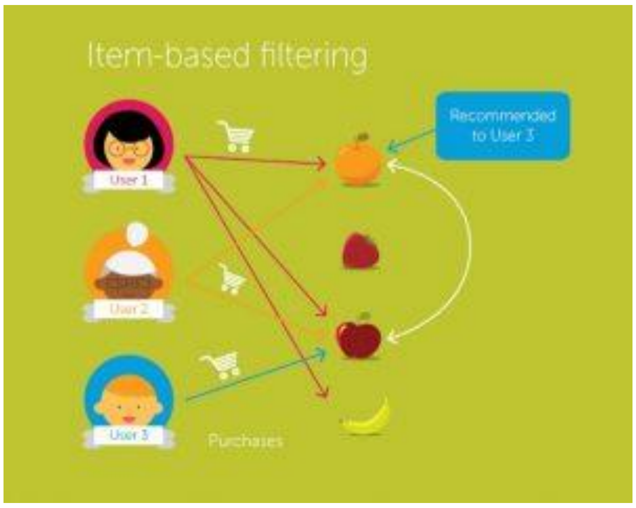
This method uses a rating system to predict the preferences of one user by taking into consideration the preferences of a similar user, or the 'neighbor'.



It is of 2 types:

i. User based: Here, the existing ratings of an user A is matched with other users with similar ratings for the same items. Then, for the items which have not been rated yet by A, the ratings of these other set of users for these items are considered to predict the likely rating to be given by A for these.

ii. Item based: In Item-Based Collaborative Filtering, we compare two items and assume them to be similar when one user gives the two items similar ratings. We then predict that user's rating for an item by calculating the weighted average of ratings on most X similar items from this user.



2.10. Model Based Collaborative filtering:

In this method of collaborative filtering recommender systems, different data mining and machine learning algorithms are used to develop a model to predict a user's rating of an unrated item. Some examples of these models are Bayesian networks, clustering models, singular value decomposition, probabilistic latent semantic analysis, multiple multiplicative factor, latent Dirichlet allocation and Markov decision process-based models. Such models find trends in the ratings being given by users to predict the rating an user will give to an unrated item.

2.11. GridSearchCV:

It is a function available in Scikit-learn's model_selection package and also in surprise's model_selection package that automates cross validation and hyperparameter tuning of the ML models. We will use it in combination with RMSE and MAE to do the tuning.

2.12. Performance metrics:

Here we will use 4 performance metrics. Those are:

i. MSE: It is the mean of the squares of the difference between the actual values and predicted values. It is given by:

$$\boldsymbol{MSE = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}$$

Here,

n=number of predictions

$\boldsymbol{y_i}$ =actual value of the ith item

$\boldsymbol{\hat{y}_i}$ =predicted value of the ith item

ii. RMSE: It is the square root of the MSE. It is given by:

$$\boldsymbol{RMSE = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}}$$

Here,

n=number of predictions

$\boldsymbol{y_i}$ =actual value of the ith item

$\boldsymbol{\hat{y}_i}$ =predicted value of the ith item

iii. MAE: It is the mean of the squares of the difference between the actual values and predicted values. It is given by:

$$\boldsymbol{MAE = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)}$$

Here,

n=number of predictions

$\boldsymbol{y_i}$ =actual value of the ith item

$\boldsymbol{\hat{y}_i}$ =predicted value of the ith item

3. Project breakdown:

Our project consists of the following steps:

i. Importing of libraries
ii. Importing of dataset and extracting the ratings data from it
iii. Dividing the dataset into 2 parts
iv. Setting the hyperparameters for GridSearchCV using KNN algorithm
v. Fitting the smaller dataset part in the model
vi. Finding the best score and set of parameters for the model
vii. Building the training dataset using the larger part of the dataset and the test dataset using the smaller part.
viii. Training the model using the training dataset
ix. Do predictions using the trained model and the test dataset
x. Find RMSE, MSE and MAE for Memory based
xi. Setting the hyperparameters for GridSearchCV using SVD
xii. Fitting the larger dataset part in the model
xiii. Finding the best score and set of parameters for the model
xiv. Building the full training dataset and the test dataset using the smaller part.
xv. Training the model using the training dataset
xvi. Do predictions using the trained model and the test dataset
xvii. Find RMSE, MSE and MAE for Matrix factorization based

4. Environment Setup:

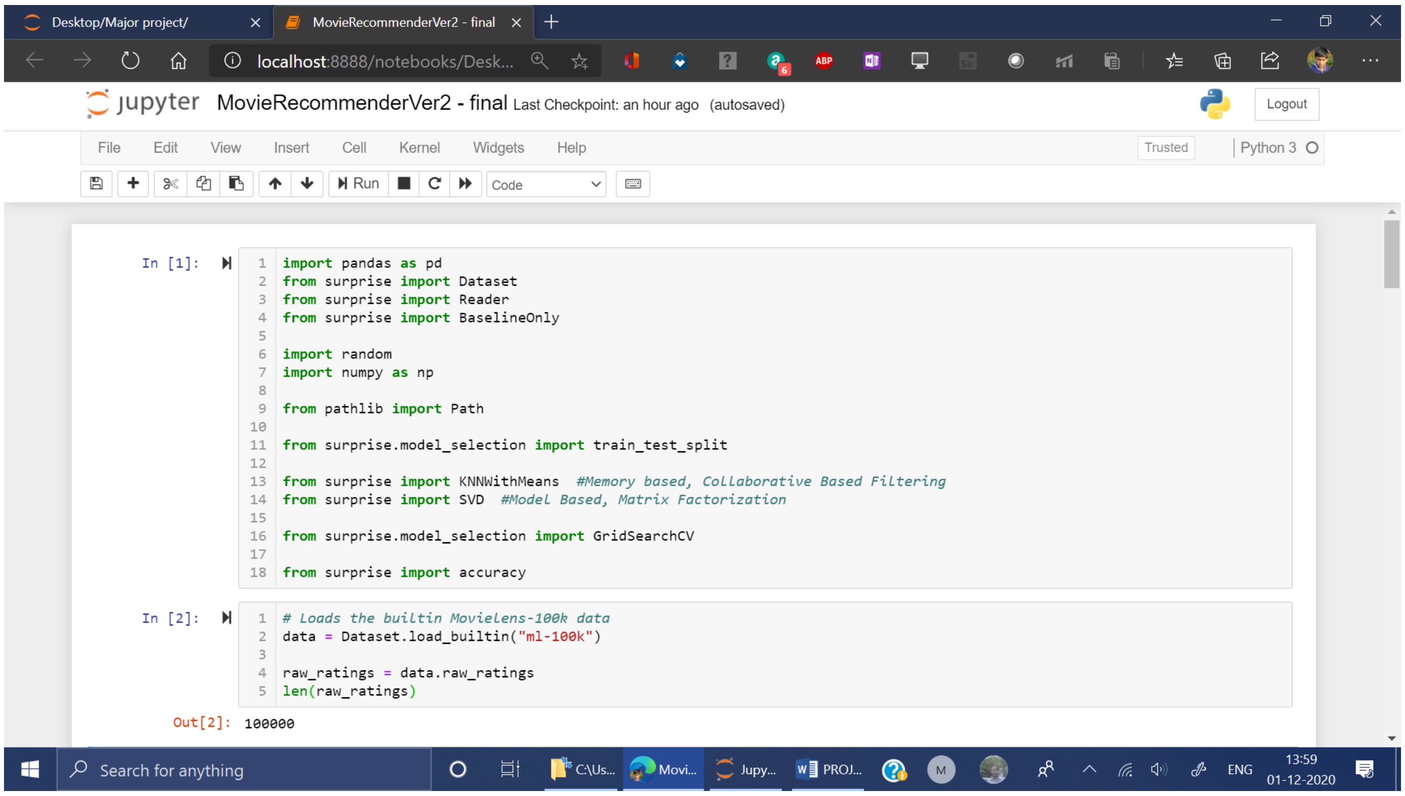
i. Install anaconda 4.9.2 with python 3.7
ii. Open anaconda powershell
iii. Run the command “pip install pandas” to install pandas
iv. Run the command “pip install numpy” to install numpy
v. Run the command “conda install -c conda-forge scikit-surprise” to install surprise package

5. Dataset:

Here, we are using the dataset “ml-100k” provided by GroupLens. It contains 100000 ratings given by different users for different movies and each user has rated atleast 20 movies. It is inbuilt in the surprise package.

6. Implementation and related code:

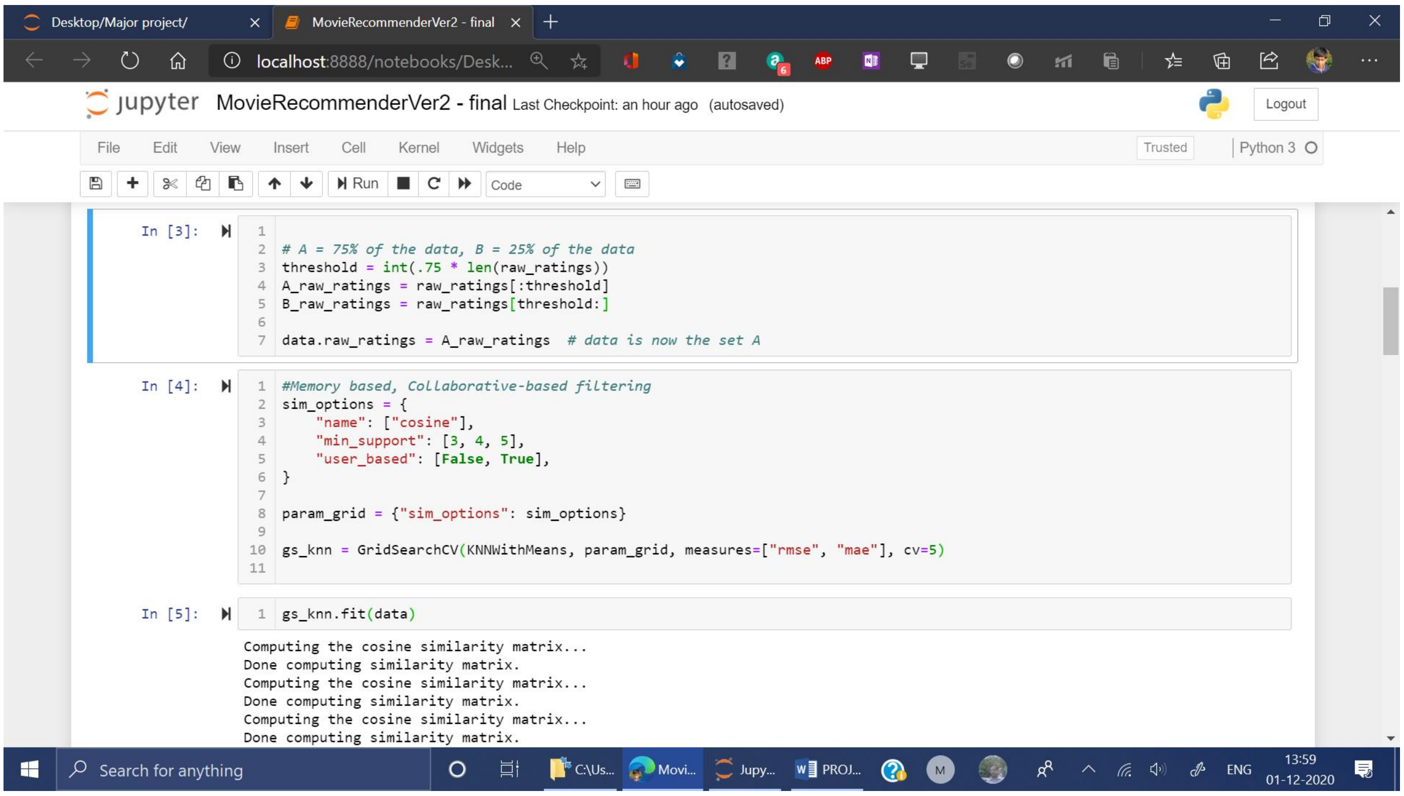
Steps 1 and 2:



```
In [1]:
import pandas as pd
from surprise import Dataset
from surprise import Reader
from surprise import BaselineOnly

import random
import numpy as np

from pathlib import Path

from surprise.model_selection import train_test_split

from surprise import KNNWithMeans  #Memory based, Collaborative Based Filtering
from surprise import SVD  #Model Based, Matrix Factorization

from surprise.model_selection import GridSearchCV

from surprise import accuracy
```

```
In [2]:
# Loads the builtin Movielens-100k data
data = Dataset.load_builtin("ml-100k")

raw_ratings = data.raw_ratings
len(raw_ratings)

Out[2]: 100000
```

Steps 3 to 5:



Desktop/Major project/
MovieRecommenderVer2 - final
localhost:8888/notebooks/Desk...
jupyter MovieRecommenderVer2 - final Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Search for anything
13:59
01-12-2020

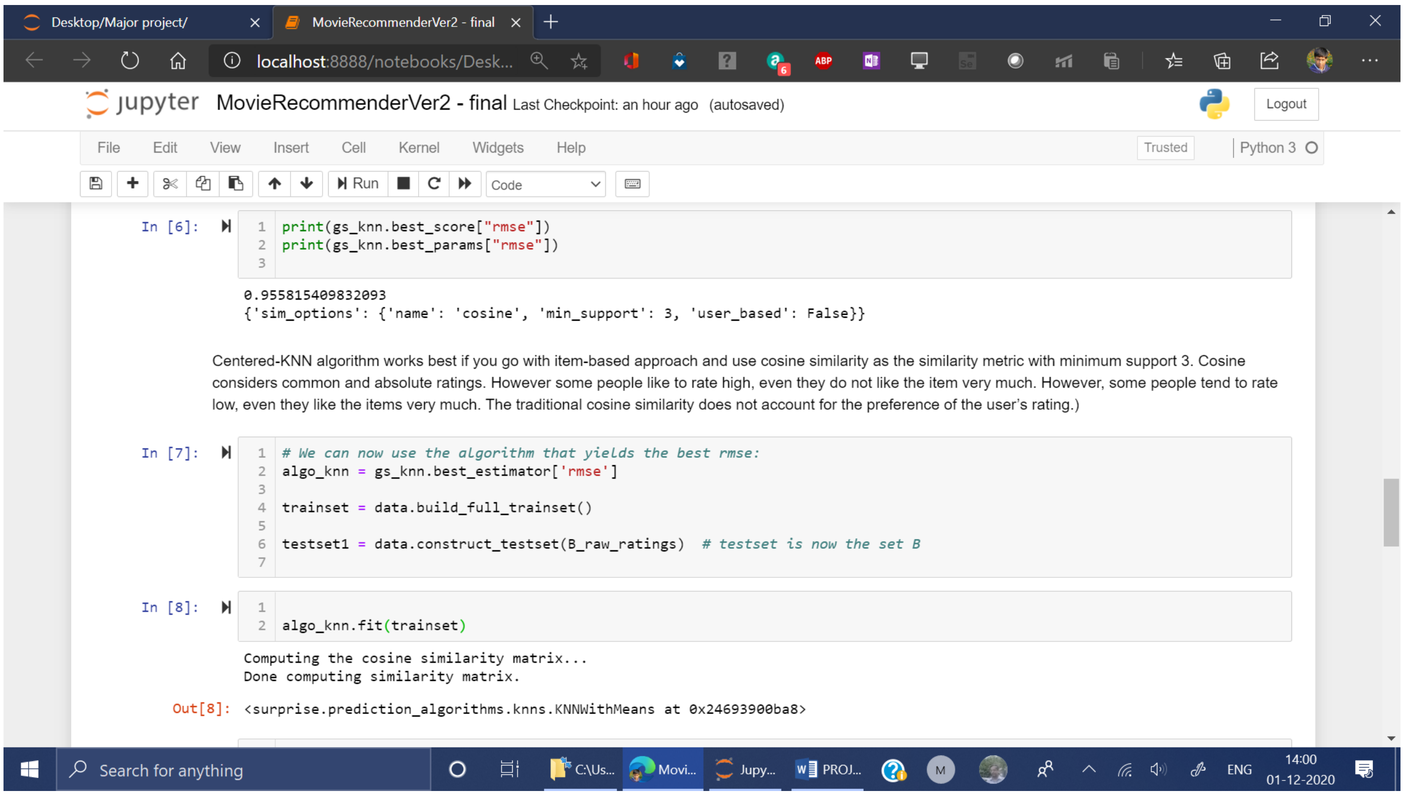
In [3]:
```
# A = 75% of the data, B = 25% of the data
threshold = int(.75 * len(raw_ratings))
A_raw_ratings = raw_ratings[:threshold]
B_raw_ratings = raw_ratings[threshold:]

data.raw_ratings = A_raw_ratings  # data is now the set A
```

In [4]:
```
#Memory based, Collaborative-based filtering
sim_options = {
    "name": ["cosine"],
    "min_support": [3, 4, 5],
    "user_based": [False, True],
}

param_grid = {"sim_options": sim_options}

gs_knn = GridSearchCV(KNNWithMeans, param_grid, measures=["rmse", "mae"], cv=5)

```

In [5]:
```
gs_knn.fit(data)
```

```
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.
```

Steps 6 to 8:



Desktop/Major project/
MovieRecommenderVer2 - final
localhost:8888/notebooks/Desk...
jupyter MovieRecommenderVer2 - final Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Search for anything
14:00
01-12-2020

In [6]:
```
print(gs_knn.best_score["rmse"])
print(gs_knn.best_params["rmse"])

```

```
0.955815409832093
{'sim_options': {'name': 'cosine', 'min_support': 3, 'user_based': False}}
```

Centered-KNN algorithm works best if you go with item-based approach and use cosine similarity as the similarity metric with minimum support 3. Cosine considers common and absolute ratings. However some people like to rate high, even they do not like the item very much. However, some people tend to rate low, even they like the items very much. The traditional cosine similarity does not account for the preference of the user's rating.)

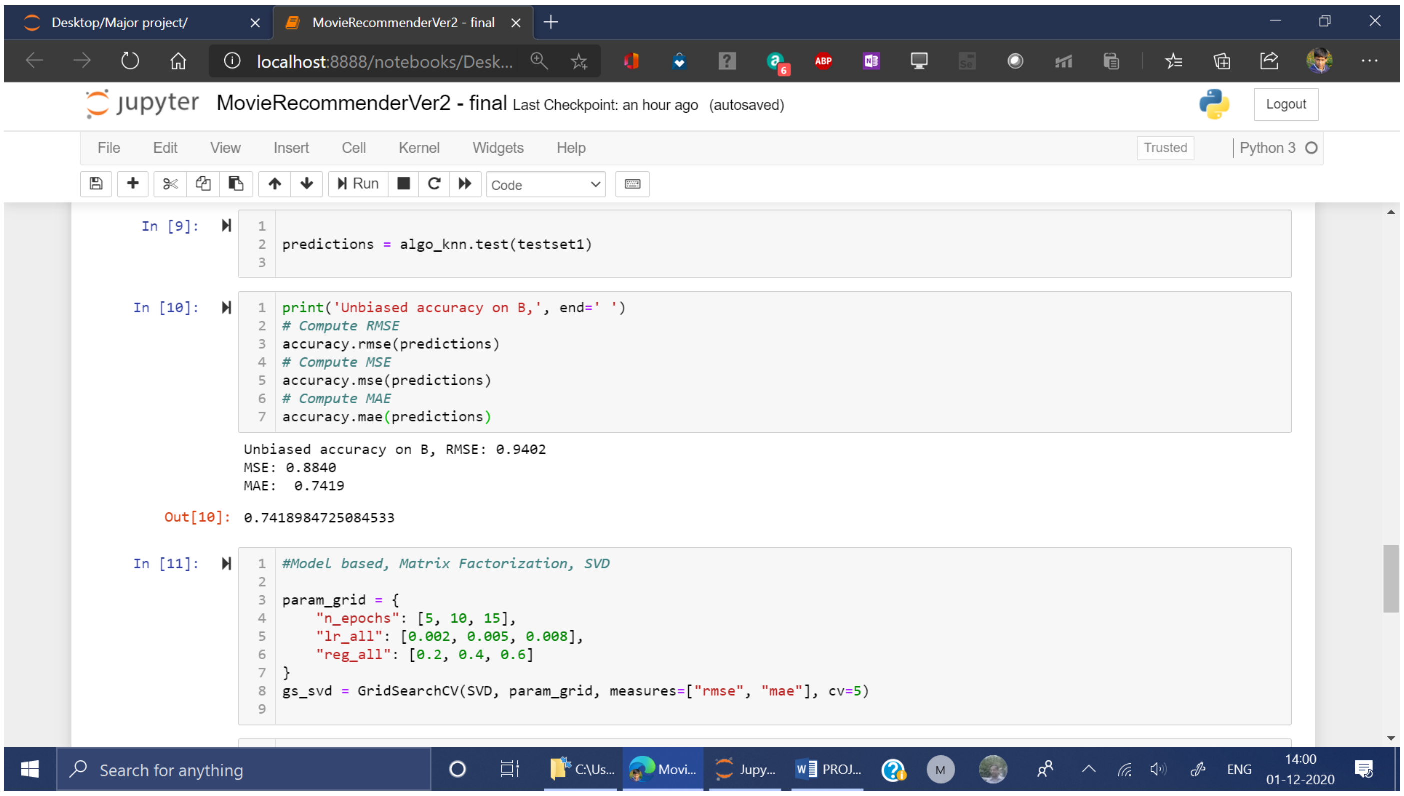
In [7]:
```
# We can now use the algorithm that yields the best rmse:
algo_knn = gs_knn.best_estimator['rmse']

trainset = data.build_full_trainset()

testset1 = data.construct_testset(B_raw_ratings)  # testset is now the set B

```

In [8]:
```

algo_knn.fit(trainset)
```

```
Computing the cosine similarity matrix...
Done computing similarity matrix.
```

Out[8]:
```
<surprise.prediction_algorithms.knns.KNNWithMeans at 0x24693900ba8>
```

Steps 9 to 11:



Desktop/Major project/
MovieRecommenderVer2 - final
localhost:8888/notebooks/Desk...
jupyter MovieRecommenderVer2 - final Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Search for anything

In [9]:
```
predictions = algo_knn.test(testset1)
```

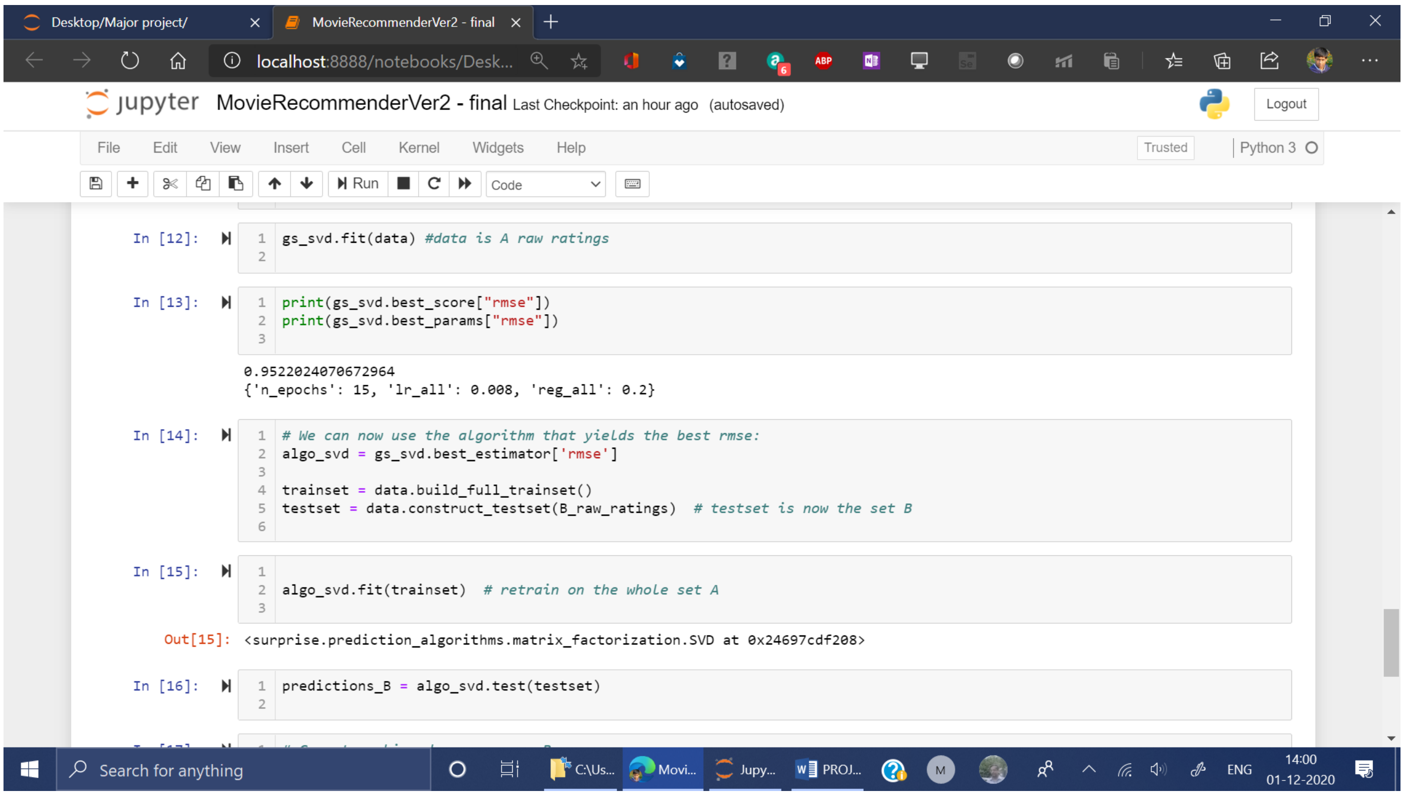
In [10]:
```
print('Unbiased accuracy on B,', end=' ')
# Compute RMSE
accuracy.rmse(predictions)
# Compute MSE
accuracy.mse(predictions)
# Compute MAE
accuracy.mae(predictions)
```

```
Unbiased accuracy on B, RMSE: 0.9402
MSE: 0.8840
MAE:  0.7419
```

Out[10]:
```
0.7418984725084533
```

In [11]:
```
#Model based, Matrix Factorization, SVD

param_grid = {
    "n_epochs": [5, 10, 15],
    "lr_all": [0.002, 0.005, 0.008],
    "reg_all": [0.2, 0.4, 0.6]
}
gs_svd = GridSearchCV(SVD, param_grid, measures=["rmse", "mae"], cv=5)
```

Steps 12 to 16:



Desktop/Major project/
MovieRecommenderVer2 - final
localhost:8888/notebooks/Desk...
jupyter MovieRecommenderVer2 - final Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Search for anything

In [12]:
```
gs_svd.fit(data) #data is A raw ratings
```

In [13]:
```
print(gs_svd.best_score["rmse"])
print(gs_svd.best_params["rmse"])
```

```
0.9522024070672964
{'n_epochs': 15, 'lr_all': 0.008, 'reg_all': 0.2}
```

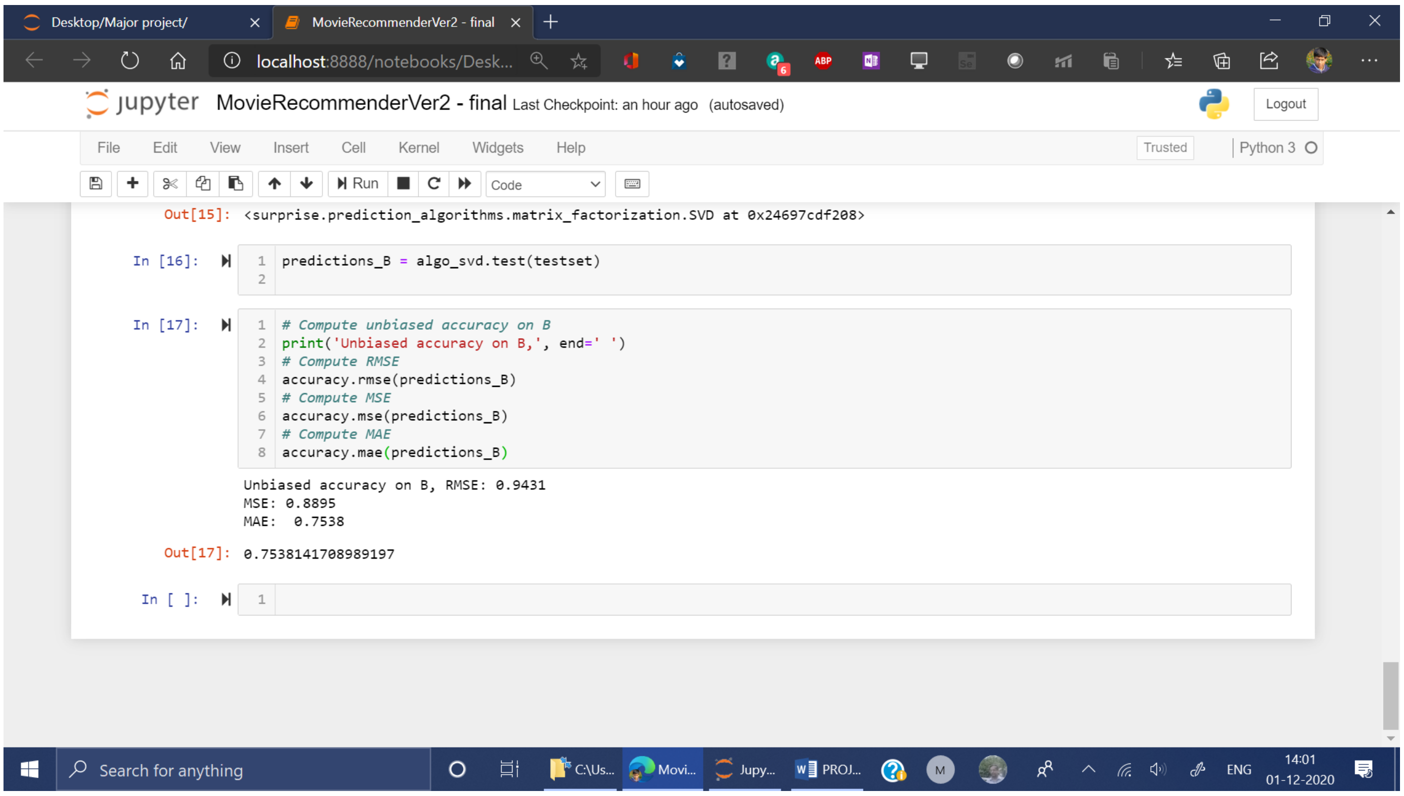
In [14]:
```
# We can now use the algorithm that yields the best rmse:
algo_svd = gs_svd.best_estimator['rmse']

trainset = data.build_full_trainset()
testset = data.construct_testset(B_raw_ratings)  # testset is now the set B
```

In [15]:
```
algo_svd.fit(trainset)  # retrain on the whole set A
```

Out[15]:
```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x24697cdf208>
```

In [16]:
```
predictions_B = algo_svd.test(testset)
```

Step 17:


Desktop/Major project/
MovieRecommenderVer2 - final
localhost:8888/notebooks/Desk...
jupyter MovieRecommenderVer2 - final Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Search for anything
ENG
14:01
01-12-2020

```
Out[15]: <surprise.prediction_algorithms.matrix_factorization.SVD at 0x24697cdf208>

In [16]: predictions_B = algo_svd.test(testset)


In [17]: # Compute unbiased accuracy on B
         print('Unbiased accuracy on B,', end=' ')
         # Compute RMSE
         accuracy.rmse(predictions_B)
         # Compute MSE
         accuracy.mse(predictions_B)
         # Compute MAE
         accuracy.mae(predictions_B)

         Unbiased accuracy on B, RMSE: 0.9431
         MSE: 0.8895
         MAE:  0.7538

Out[17]: 0.7538141708989197

In [ ]:
```

7. Test results of the project:

Memory Based Collaborative Filter:

```
           Unbiased accuracy on B, RMSE: 0.9402
           MSE: 0.8840
           MAE:  0.7419

Out[10]:   0.7418984725084533
```

Model Based collaborative Filter:

```
           Unbiased accuracy on B, RMSE: 0.9431
           MSE: 0.8895
           MAE:  0.7538

Out[17]:   0.7538141708989197
```

8. Conclusion:

The following metrics were obtained in the different filters:

Metric	Memory Based	Model Based
MAE	0.7419	0.7538
MSE	0.8840	0.8895
RMSE	0.9402	0.9431

As we can see, for every metric both the models give almost similar amounts of errors. The memory based filter only performs slightly better which is almost insignificant. This shows that both the filters are equally efficient in predicting the user ratings.

Also, it should be noted that the error is always less than 1. This means that the models can predict the user ratings with a maximum error which is less than 1. But considering that the ratings are between 1 and 5 and are only integer values, the model should effectively predict the correct rating without any error in most cases. The only time there will be an error is when the number of users to predict for is very high and the average rating of all those users are considered for evaluation.

This shows that such recommendation engines are fairly accurate and effective in performing recommendation for users provided the input data given to the engines are cleaned and processed properly.

The value of these errors can be reduced significantly further by using neural networks for our recommendation engines.

9. References:

[1] https://en.wikipedia.org/wiki/Recommender_system

[2] https://en.wikipedia.org/wiki/Collaborative_filtering

[3] http://mlwiki.org/index.php/Singular_Value_Decomposition

[4] https://medium.com/acing-ai/what-is-cosine-similarity-matrix-f0819e674ad1

[5] https://en.wikipedia.org/wiki/Cosine_similarity

[6] https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm

[7] https://www.mygreatlearning.com/blog/gridsearchcv/

[8] http://surpriselib.com/