COLLABORATIVE FILTERING USING THE NETFLIX DATA

University Of New Haven

Nihal Singh

05-14-2021

Abstract:

The goal of this project is to build a machine learning model using collaborative filtering which accurately predict the rating of the given test data set. In this project, we will also comparing the prediction model built using user based collaborative filtering and items based collaborative filtering. There are various types of recommender system out there in market but in this project we will just be focusing ourselves on collaborative filtering recommendation system.

Table of Content:

Introduction:

Recommender system are widely used in the present world. Almost everyone in everyday life are on the receiving end of this technology advancement. Almost all the tech company in the current world is using a recommender system. Google searches, Netflix movies recommendation, uber eats restaurant suggestions, youtube recommended videos for you, customized add on your web browsers, etc. and many more company are using recommender system. One can hide from the most infectious disease in the world but I cannot claim that they have recommender system in their daily life.

Recommender system in technical words means that, it is a subclass of information filtering system which aims to predict the user preference for a given item. There are mainly six types of recommendation system which are listed below.

- Collaborative Recommender System
- Content Based Recommender System
- Demographic Based Recommender System
- Utility Based Recommender System
- Knowledge Based Recommender System
- Hybrid Recommender System

Collaborative filtering addresses some of the limitation of content-based filtering. It uses similarities between users and items provide recommendation. This filtering systems apply the so-called similarity index-based technique. Number of users are selected based on their similarity to the active user. Inference for the active user is made by calculating a weighted average of the ratings of the selected users.

There are two types of Collaborative filtering. They are

- User-based
- Item-based

Similarity between target users and other users is measured in user-based collaborative filtering.

Similarity between the items that target users rate is measured.

Problem Definition:

The intention of this project is to build a machine learning model using collaborative filtering which accurately predict the rating of the given test data set.

About Data:

The data set is about the ratings given by users to movies. This was first published by Netflix in 2009. There approximately 3.2 million samples present in training data set. Also, there are approximately 100,000 samples present in testing data set. There are more than 17,000 movie titles in movie_titles.txt.

Implementations:

Pyspark Setup:

The first step in this project is setting up pyspark jupyter notebook.

Steps:

Part 1

1. Go to aws and click on emr
2. Click on create cluster
3. During system configuration choose SPARK.
4. Select m4x.large as instance
5. Select the ec2 key pair for access.

Part 2

1. Once the cluster is ready, open putty in your laptop.
2. Meanwhile add port number 8888 and 22 to your security group.
3. Connect to your emr using putty and ec2 key pair.

Once connected using putty, follow below instruction for setting up spark notebook. Type the following commands in the terminal.

Part 3

1. sudo su
2. python3 -m pip install pyyaml ipython jupyter ipyparallel pandas boto –U
3. exit
4. export PYSPARK_DRIVER_PYTHON=/usr/local/bin/jupyter
5. export PYSPARK_DRIVER_PYTHON_OPTS="notebook --no-browser --ip=0.0.0.0 --port=8888"
6. source .bashrc
7. pyspark

A link will appear after you type pyspark. Make edit to the link using DNS of master ec2 instance. It will look something like this:

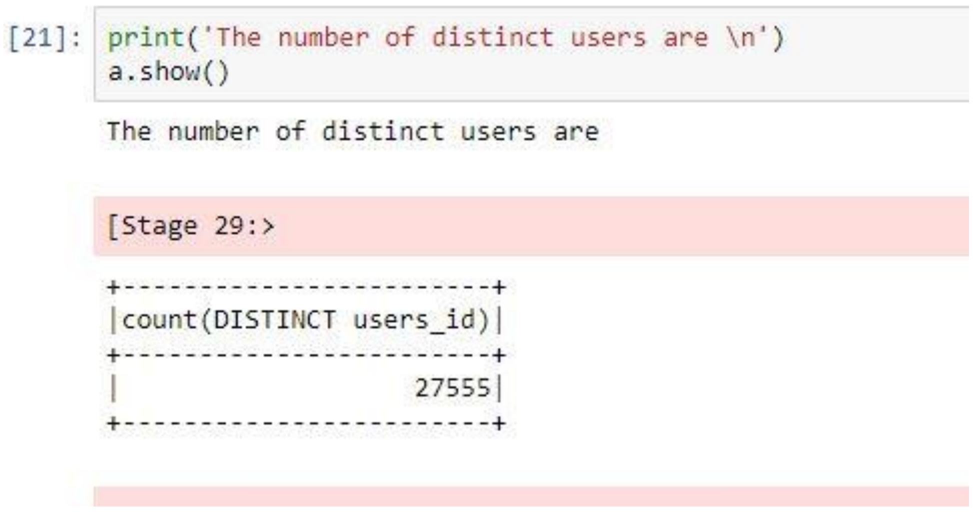
http://ec2-3-239-100-189.compute-1.amazonaws.com:8889/notebooks/EMR_PYspark_DSCI6007_Final%20(2)%20(2).ipynb#

Coding in Pyspark Jupyter Notebook

1. Upload the file to s3 bucket.
2. Imported the libraries with were going to be used.
3. Read the movie_titles.txt,TestingRating.txt and TrainingRatings.txt file from s3 bucket
4. Calculated distinct user and distinct movies in testing bucket
5. Calculated the correlation matrix
6. Imported libraries for ASL
7. Built model using ASL (Alternative Least Square) method for both user based and item based collaborative filtering.
   (Apache Spark ML implements alternating least squares (ALS) for collaborative filtering, a very popular algorithm for making recommendations.)
8. Optimized the model by trying various hyper-parameters.
9. Predicted the rating of testing data.
10. Calculated the root mean squared error and mean squared error
11. Created my_rating data set and merged it with training data set.
12. Predicted the rating for myself using the same above model
13. Calculated the root mean squared error and mean squared error after adding my data.

Distinct Users in TestingRatings.txt

There were 27,555 distinct users in TestinRatings.txt

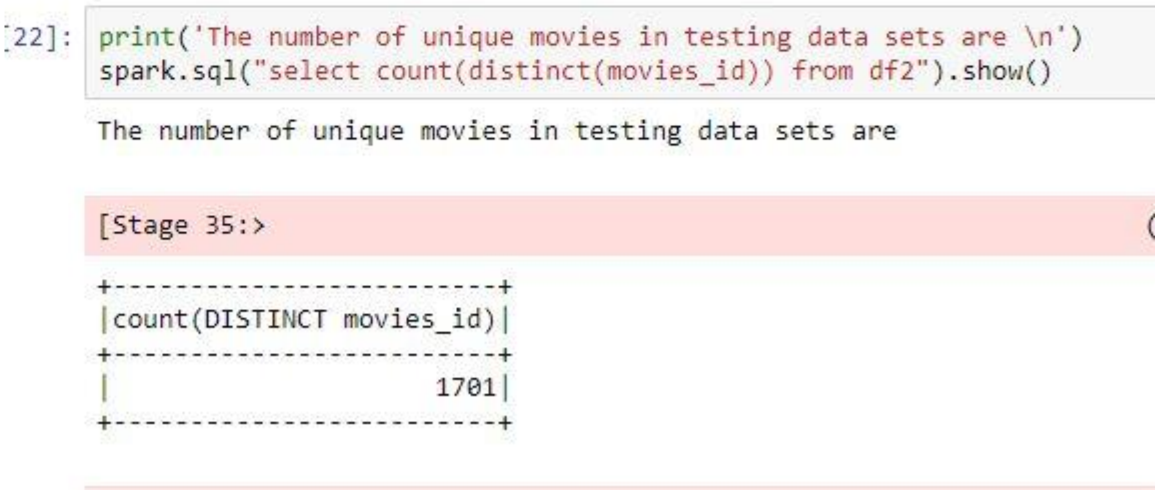


```
[21]: print('The number of distinct users are \n')
      a.show()
```

```
The number of distinct users are 

[Stage 29:>

+-----------------------+
|count(DISTINCT users_id)|
+-----------------------+
|                  27555|
+-----------------------+
```

Distinct Items in TestingRatings.txt

There were 1,701 distinct items in TestingRatings.txt



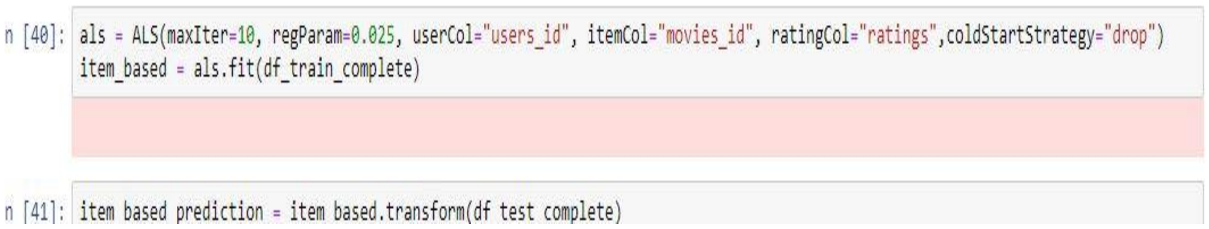
```
[22]: print('The number of unique movies in testing data sets are \n')
      spark.sql("select count(distinct(movies_id)) from df2").show()

      The number of unique movies in testing data sets are

      [Stage 35:>                                                          (

      +-------------------------+
      |count(DISTINCT movies_id)|
      +-------------------------+
      |                     1701|
      +-------------------------+
```

ALS Model Building:

We can build ALS model by importing the following libraries.

- from pyspark.ml.evaluation import RegressionEvaluator
- from pyspark.ml.recommendation import ALS
- from pyspark.sql import Row

ALS parameters are maxIter =15, regParam= 0.005



```
n [40]: als = ALS(maxIter=10, regParam=0.025, userCol="users_id", itemCol="movies_id", ratingCol="ratings",coldStartStrategy="drop")
        item_based = als.fit(df_train_complete)

n [41]: item based prediction = item based.transform(df test complete)
```

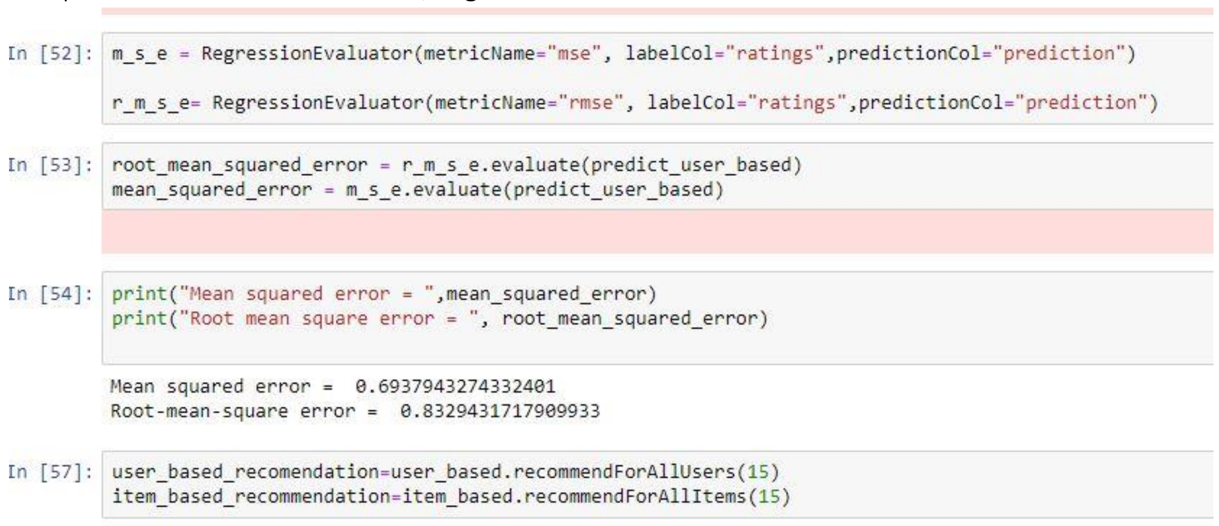
Results:

Since, in this project we are predicting the rating of the user, the best metrics here for measuring the accuracy of the model will be root mean squared error and mean squared error.

The error of different situation is calculated and comparison is made.

Error for user based collaborative filter

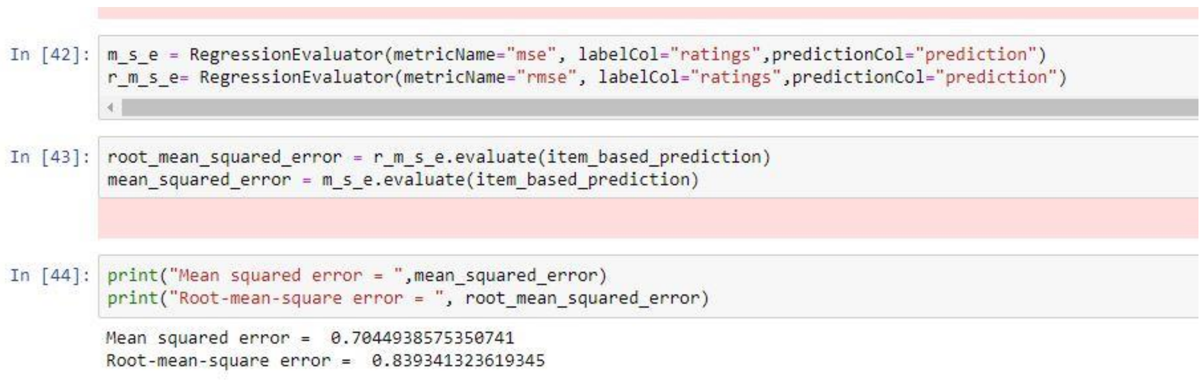
ALS parameters are maxIter =10, regParam= 0.025



```
In [52]: m_s_e = RegressionEvaluator(metricName="mse", labelCol="ratings",predictionCol="prediction")

         r_m_s_e= RegressionEvaluator(metricName="rmse", labelCol="ratings",predictionCol="prediction")

In [53]: root_mean_squared_error = r_m_s_e.evaluate(predict_user_based)
         mean_squared_error = m_s_e.evaluate(predict_user_based)

In [54]: print("Mean squared error = ",mean_squared_error)
         print("Root mean square error = ", root_mean_squared_error)

         Mean squared error =  0.6937943274332401
         Root-mean-square error =  0.8329431717909933

In [57]: user_based_recomendation=user_based.recommendForAllUsers(15)
         item_based_recommendation=item_based.recommendForAllItems(15)
```

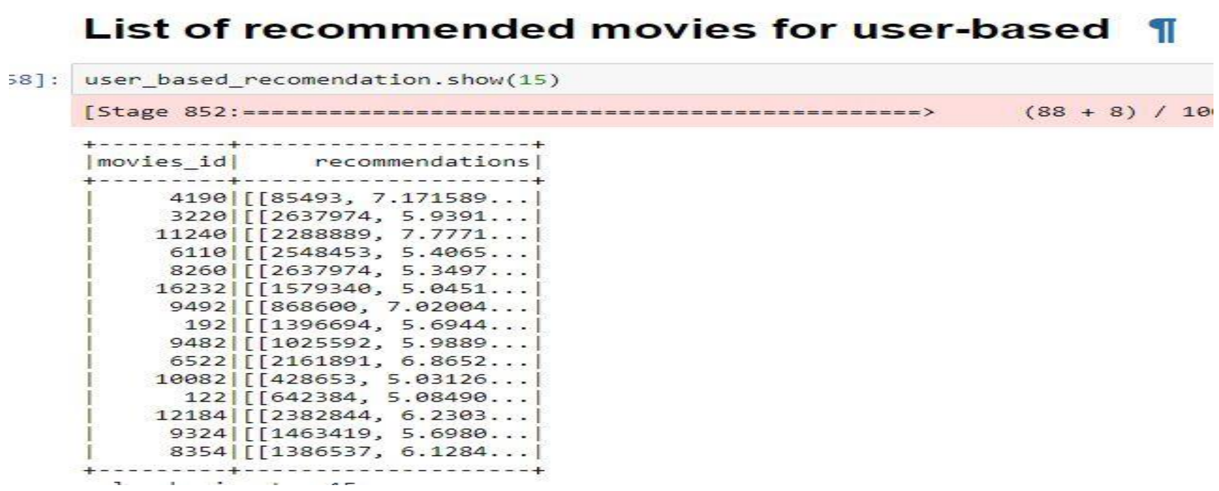
Error for item based collaborative filter

ALS parameters are maxIter =10, regParam= 0.025



```
In [42]: m_s_e = RegressionEvaluator(metricName="mse", labelCol="ratings",predictionCol="prediction")
         r_m_s_e= RegressionEvaluator(metricName="rmse", labelCol="ratings",predictionCol="prediction")

In [43]: root_mean_squared_error = r_m_s_e.evaluate(item_based_prediction)
         mean_squared_error = m_s_e.evaluate(item_based_prediction)

In [44]: print("Mean squared error = ",mean_squared_error)
         print("Root-mean-square error = ", root_mean_squared_error)

         Mean squared error =  0.7044938575350741
         Root-mean-square error =  0.839341323619345
```

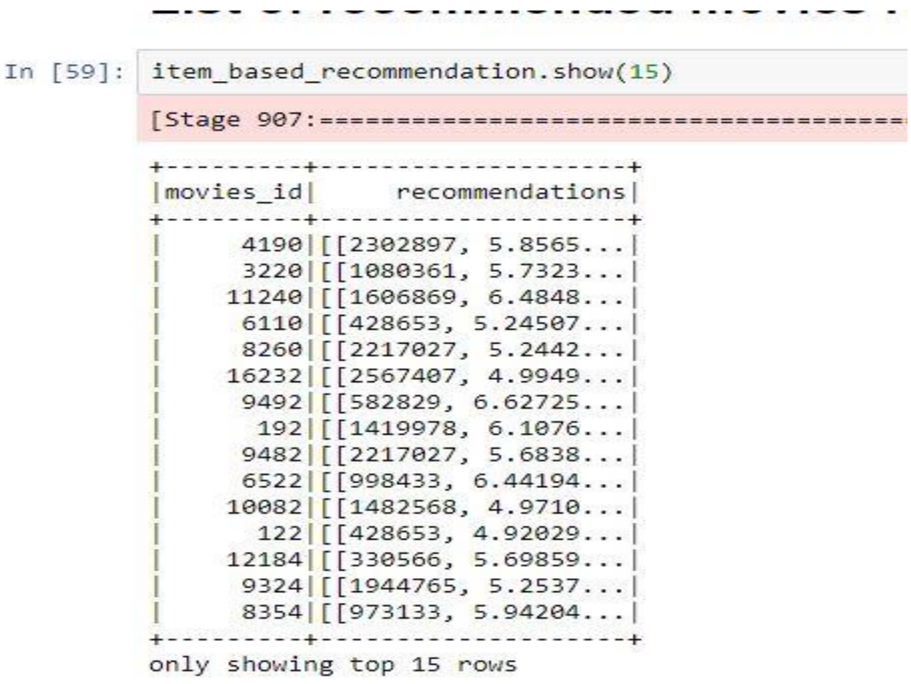
Prediction of testing data on the basis of user based model



List of recommended movies for user-based ¶

```
58]: user_based_recomendation.show(15)
     [Stage 852:================================================>      (88 + 8) / 10

     +---------+--------------------+
     |movies_id|     recommendations|
     +---------+--------------------+
     |     4190|[[85493, 7.171589...|
     |     3220|[[2637974, 5.9391...|
     |    11240|[[2288889, 7.7771...|
     |     6110|[[2548453, 5.4065...|
     |     8260|[[2637974, 5.3497...|
     |    16232|[[1579340, 5.0451...|
     |     9492|[[868600, 7.02004...|
     |      192|[[1396694, 5.6944...|
     |     9482|[[1025592, 5.9889...|
     |     6522|[[2161891, 6.8652...|
     |    10082|[[428653, 5.03126...|
     |      122|[[642384, 5.08490...|
     |    12184|[[2382844, 6.2303...|
     |     9324|[[1463419, 5.6980...|
     |     8354|[[1386537, 6.1284...|
     +---------+--------------------+
```

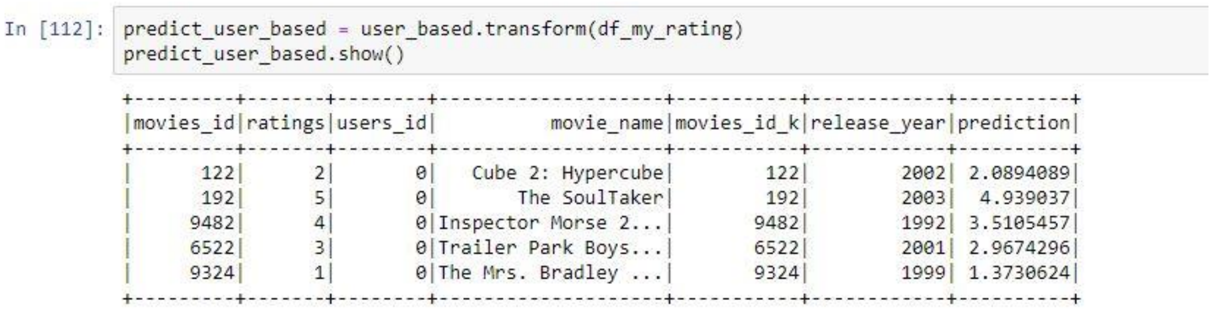
Prediction of testing data on the basis of item based model



```
In [59]:  item_based_recommendation.show(15)

           [Stage 907:========================================

           +---------+--------------------+
           |movies_id|     recommendations|
           +---------+--------------------+
           |     4190|[[2302897, 5.8565...|
           |     3220|[[1080361, 5.7323...|
           |    11240|[[1606869, 6.4848...|
           |     6110|[[428653, 5.24507...|
           |     8260|[[2217027, 5.2442...|
           |    16232|[[2567407, 4.9949...|
           |     9492|[[582829, 6.62725...|
           |      192|[[1419978, 6.1076...|
           |     9482|[[2217027, 5.6838...|
           |     6522|[[998433, 6.44194...|
           |    10082|[[1482568, 4.9710...|
           |      122|[[428653, 4.92029...|
           |    12184|[[330566, 5.69859...|
           |     9324|[[1944765, 5.2537...|
           |     8354|[[973133, 5.94204...|
           +---------+--------------------+
           only showing top 15 rows
```

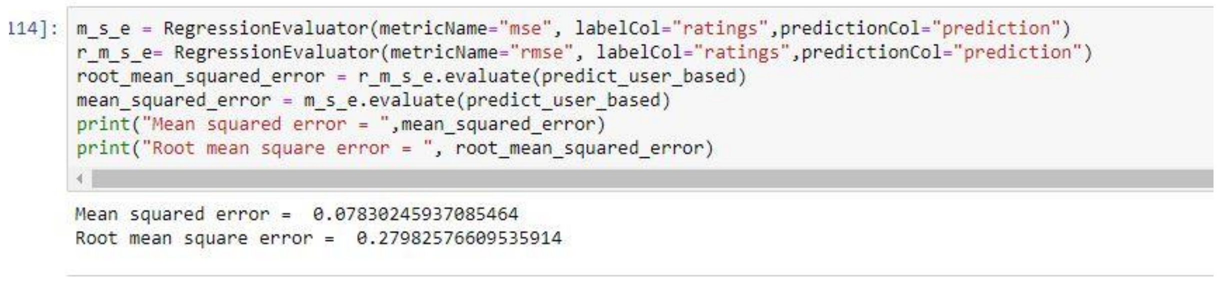
My Data:

I created my own data. I assigned myself as users_id=0 and gave rating for various movies. Then I merged this data with training data and trained the optimal model using the merged training data. After I predicted my rating using the same model which came out to very close. Also, I calculated error in my prediction when predicting my ratings.



```
In [112]:  predict_user_based = user_based.transform(df_my_rating)
           predict_user_based.show()

           +---------+-------+--------+------------------+----------+------------+----------+
           |movies_id|ratings|users_id|        movie_name|movies_id_k|release_year|prediction|
           +---------+-------+--------+------------------+----------+------------+----------+
           |      122|      2|       0|   Cube 2: Hypercube|       122|        2002| 2.0894089|
           |      192|      5|       0|       The SoulTaker|       192|        2003|  4.939037|
           |     9482|      4|       0|Inspector Morse 2...|      9482|        1992| 3.5105457|
           |     6522|      3|       0|Trailer Park Boys...|      6522|        2001| 2.9674296|
           |     9324|      1|       0|The Mrs. Bradley ...|      9324|        1999| 1.3730624|
           +---------+-------+--------+------------------+----------+------------+----------+
```

Error on my own prediction:


```
114]: m_s_e = RegressionEvaluator(metricName="mse", labelCol="ratings",predictionCol="prediction")
      r_m_s_e= RegressionEvaluator(metricName="rmse", labelCol="ratings",predictionCol="prediction")
      root_mean_squared_error = r_m_s_e.evaluate(predict_user_based)
      mean_squared_error = m_s_e.evaluate(predict_user_based)
      print("Mean squared error = ",mean_squared_error)
      print("Root mean square error = ", root_mean_squared_error)

      Mean squared error =  0.07830245937085464
      Root mean square error =  0.27982576609535914
```

Conclusion:

The user based rating in this case is giving more accurate prediction compared to item based rating. Many claim that item based is superior to user based collaborative filtering but not this case. User based collaborative filtering works well when there is a huge dataset like the one which we are using in this project while item based generally works better when the data sets are smaller. But for our project, I would prefer user based rating system. Although there is not much difference in root mean squared error of both the model but still user based would be preferred because of its accuracy.

References:

- https://www.py4u.net/discuss/1629724
- https://www.youtube.com/watch?v=3ecNC-So0r4
- https://www.youtube.com/watch?v=YLOJfN9T9NU
- https://www.youtube.com/watch?v=EfHscLaMobY
- http://recommender-systems.org/
- https://towardsdatascience.com/tagged/collaborative-filtering
- Class Notes, Lecture