**CS 724 High Performance Computing and Big Data**

**Movie Recommendation System Using ALS**

Kiran Teja Sarvamthota

UIN:01095632

**Project Abstract:**

I’ve chosen Movielens data set having user ratings for different movies. The idea is to create a recommendation system for each user based on his previous ratings. A collaborative filtering model is built to predict the ratings for the user for the movie he didn’t watch. I’ve used Normalized the ratings data and also built a scalable model.

**Recommendation Systems:** Unlike offline stores online stores have no sales people. Online stores have huge number of products:

* Netflix has thousands of movies
* Amazon has millions of books
* ITunes has tens of thousands of songs

Users on the other hand has limited time and patience and not sure what they are looking for. Recommendations helps users to:

* Navigate the maze of online stores
* Find what they are looking for
* Find things that they might look but didn’t know

Recommendation systems uses one or more of these techniques:

* Content based
* Collaborative
* Association rules

**Project Idea and Implementation:**

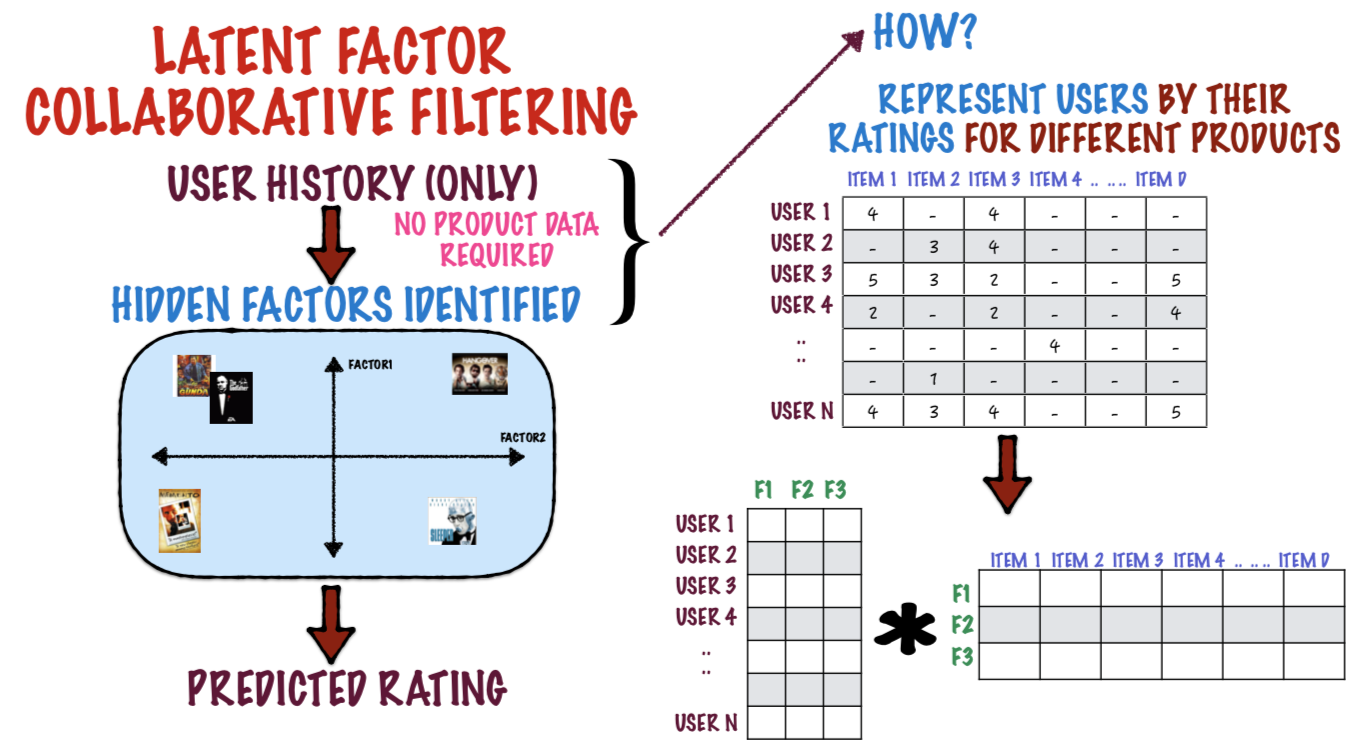
**Movie recommendation system:** The idea is to build a recommendation system which recommends movies to a user, based on other user’s previous ratings. Implementation to find the movies which user might be interested will be done by using collaborative filtering using Alternate least Squares Algorithm.

**Collaborative Filtering:** Unlike content-based filtering collaborative filtering doesn’t require any Movie description data at all. The basic premise is that if two users have same opinion about a bunch of products , they are likely to have the same opinion about other products too. The Algorithm predicts user ratings for the movies they have not rated. There are different algorithms to perform collaborative filtering they are Nearest neighbor based method and Latent factor-based methods.

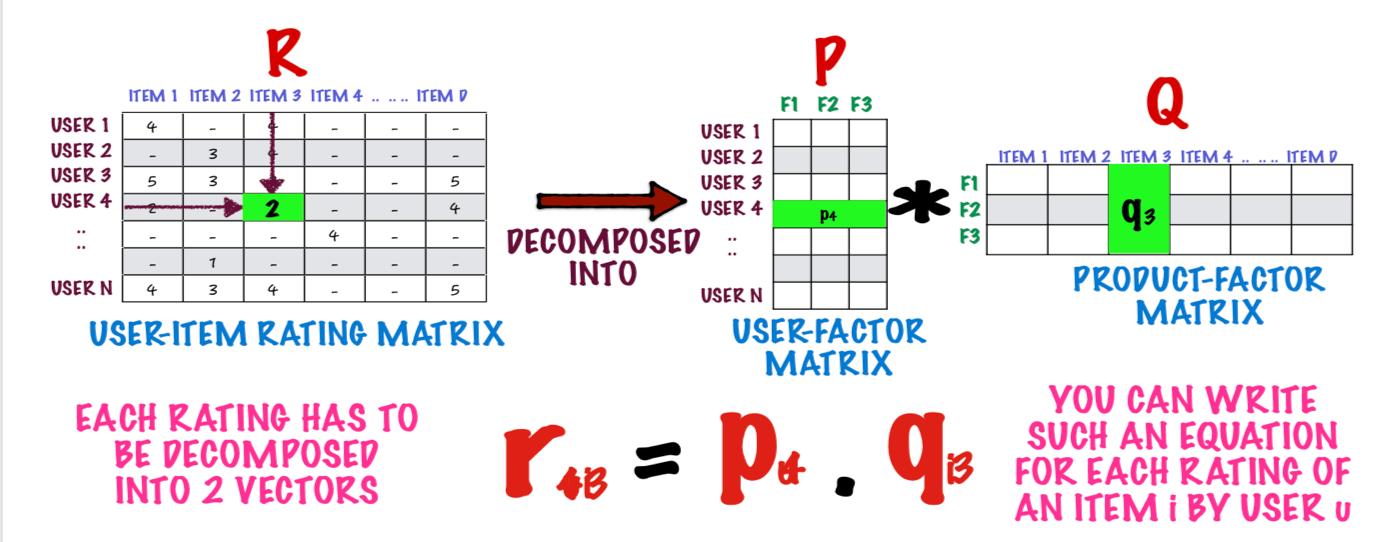
**Latent Factor based methods:** Latent Factor based methods takes user history only( no product data required). Then the hidden factors are identified and the ratings are predicted. The hidden factors are identified by representing users by their ratings for different movies.

**Matrix Factorization**: All the ratings of all the users is taken as, sparse, R matrix. Now it is considered as the product of two smaller matrices. By fixing the matrices, we get closer to the actual rating and thus how less the error value, that accurate the predictions. We can consider them a user feature matrix and movie feature matrix. This approximation is also going to smooth out the zeros and in the process give us our predicted ratings.

Matrix factorization can be done using many algorithms. In this project Alternate least Squares is used to estimate the latent factors.



**Alternate Least Squares Algorithm**: The matrix R is considered to be the product of two matrices P and Q. So the matrix factorization is done alternatively i.e. by fixing P, Q value is estimated and similarly by fixing Q, P value is estimated. After enough number of iterations, we are aiming to reach a convergence point where either the matrices P and Q are no longer changing or the change is quite small and then the product of the two matrices gives the predicted ratings.



**Dataset**: Movielens data set is used. The files, ratings and movies are used.

The ratings.csv file contains ratings given by 610 users for 9724 movies.

The movies.csv file contains the names and movie ids of the movies.

**Matrix factorization approach -ALS Implementation:**

We start with randomly chosen User matrix,Movie matrix values and adjust the values of User matrix,Movie matrix to make RMSE smaller.

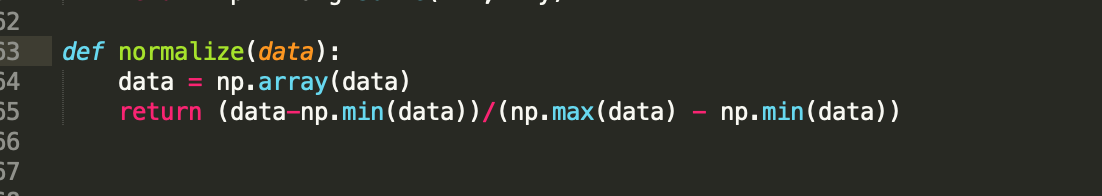
Step 1: Fix User matrix and update entries for movie matrix.

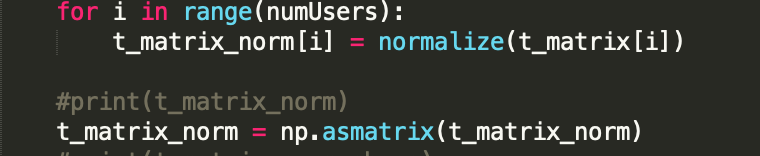
Step 2: Fix movie matrix and update entries for user matrix.

Step 3:Repeat step and step 3 until we reach a least saturation of RMSE.

**Normalization**: Here I’ve normalized the data as there are some users who gives ratings generously i.e. (if they like the movie they would rate the movie as 5 if they don’t like the movie they would rate 3). On the other hand there could be some users who could rate rate 3 if they like the movie and 1 if they don’t like the movie. By keeping all this into consideration to built a more accurate recommendation model, I’ve normalized the ratings for each user by using the below formula.

https://www.statisticshowto.datasciencecentral.com/wp-content/uploads/2015/11/normalize-data.png

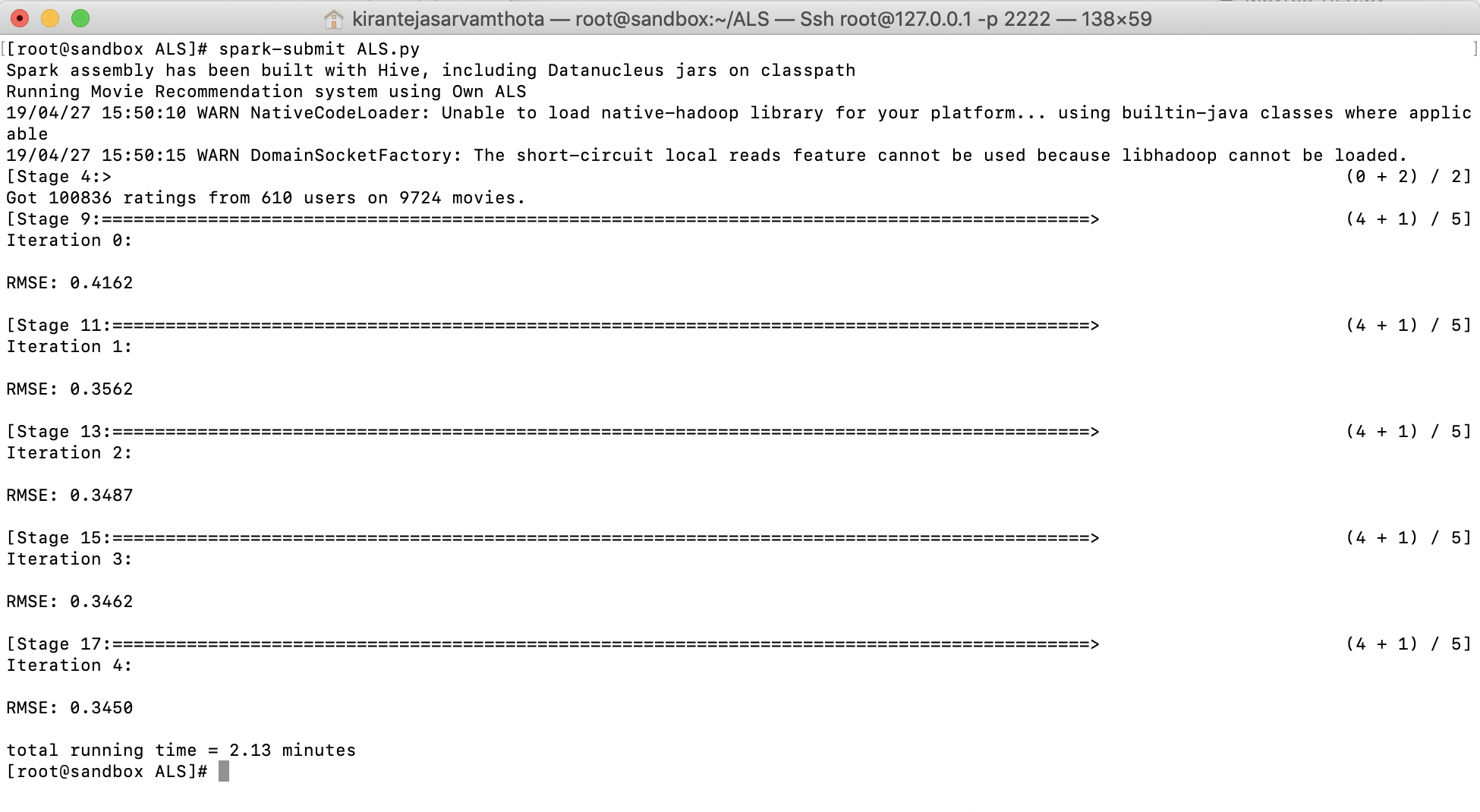




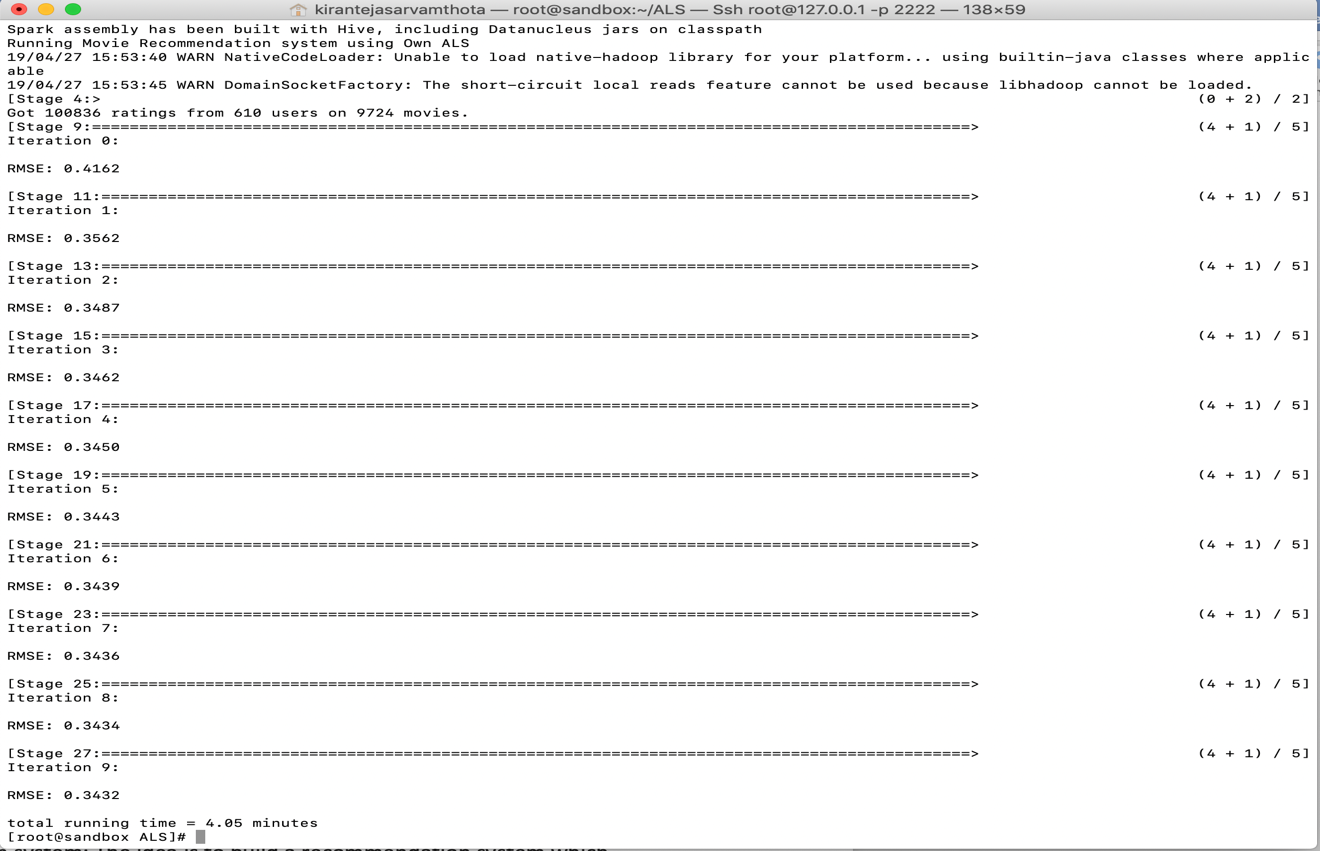
**Results:**

The predicted ratings for every user are written to recommendationsoutput.txt file. The output is in the format: Movie title, Movie ID and predicted rating for every user.

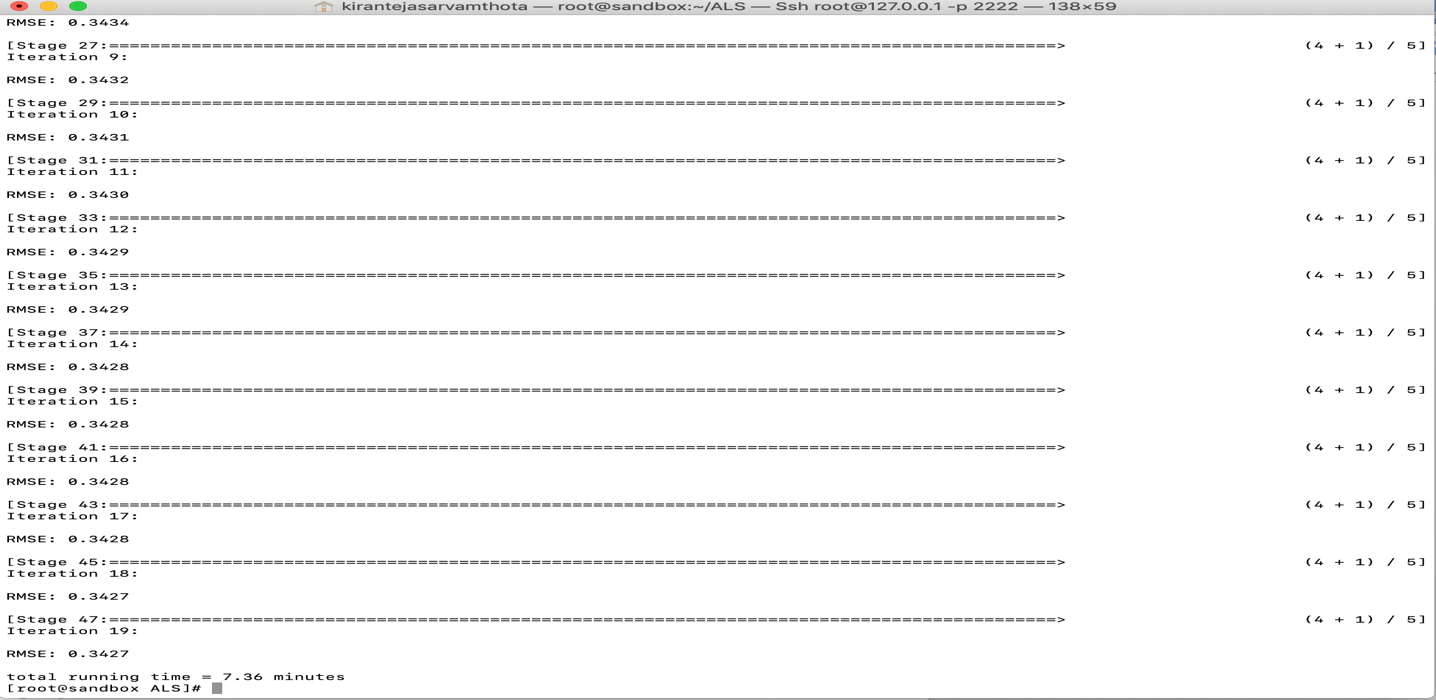
Total running time of the program is 2.13 minutes for 5 iterations for whole data.



Total running time of the program is 4.05 minutes for 10 iterations for whole data.

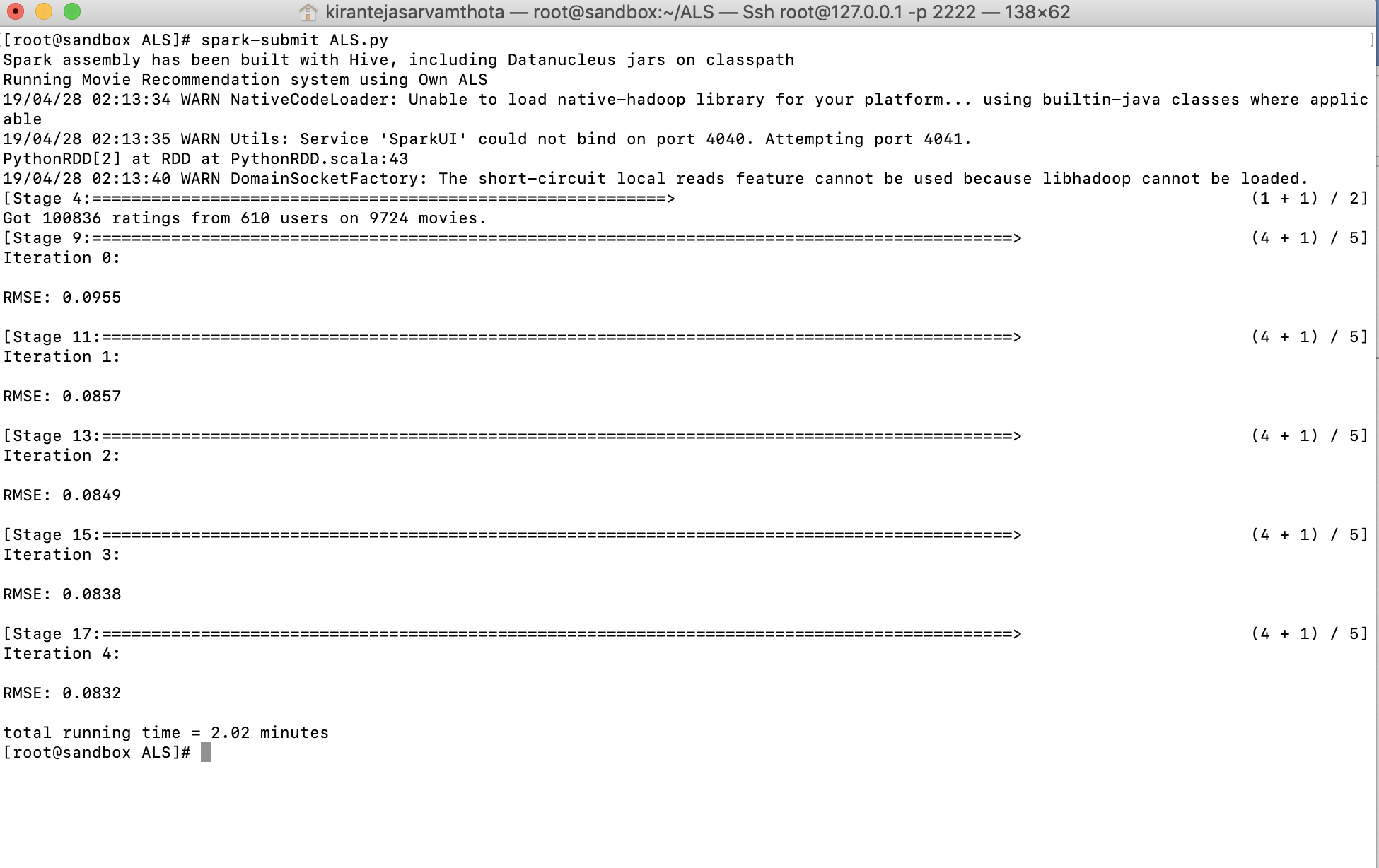


Total running time of the program is 7.36 minutes for 20 iterations for whole data.

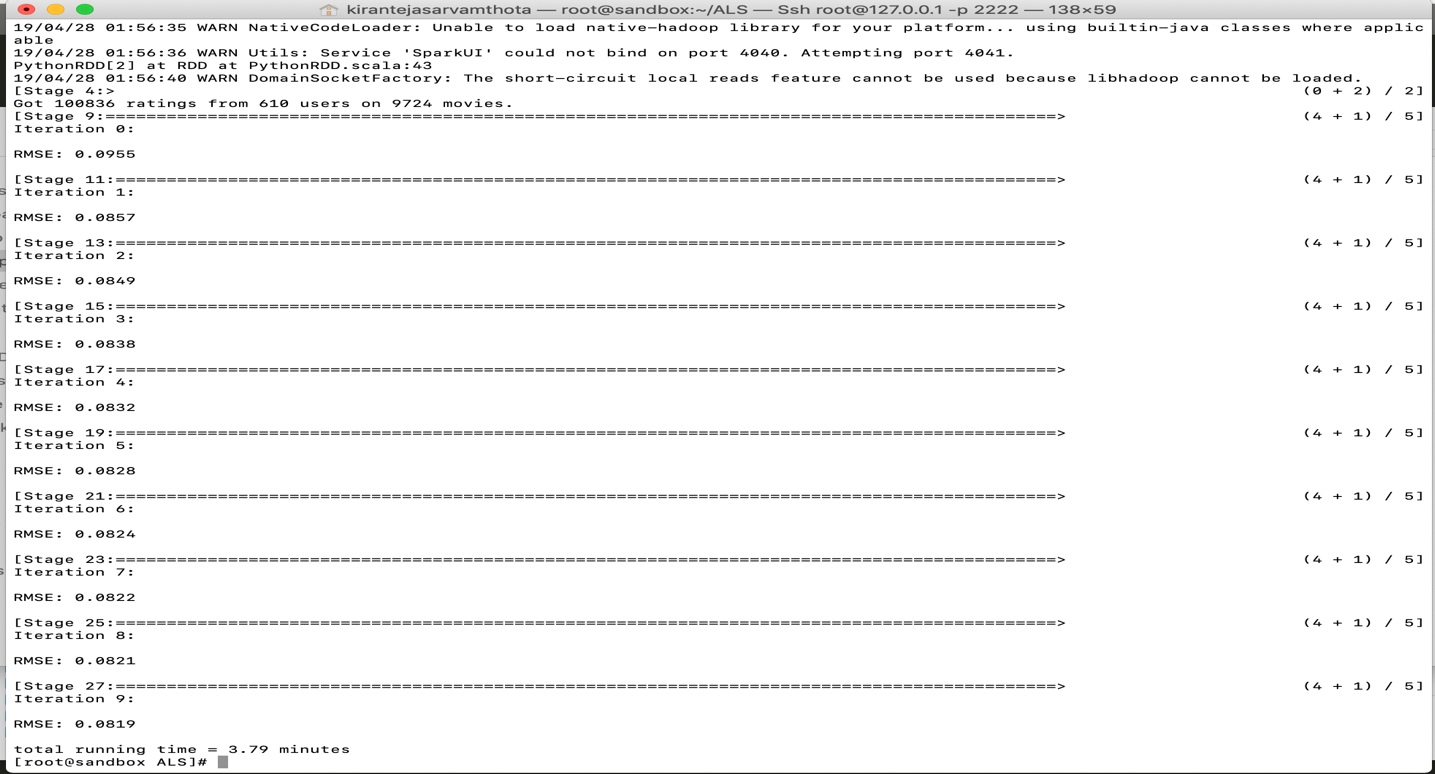


**With Normalization:** By doing Normalization the RMSE error was reduced by 76 % from (0.3724 to 0.0817) also the running time is reduced.

Total running time of the program is 3.04 minutes for 5 iterations for whole data.



Total running time of the program is 3.79 minutes for 10 iterations for whole data.

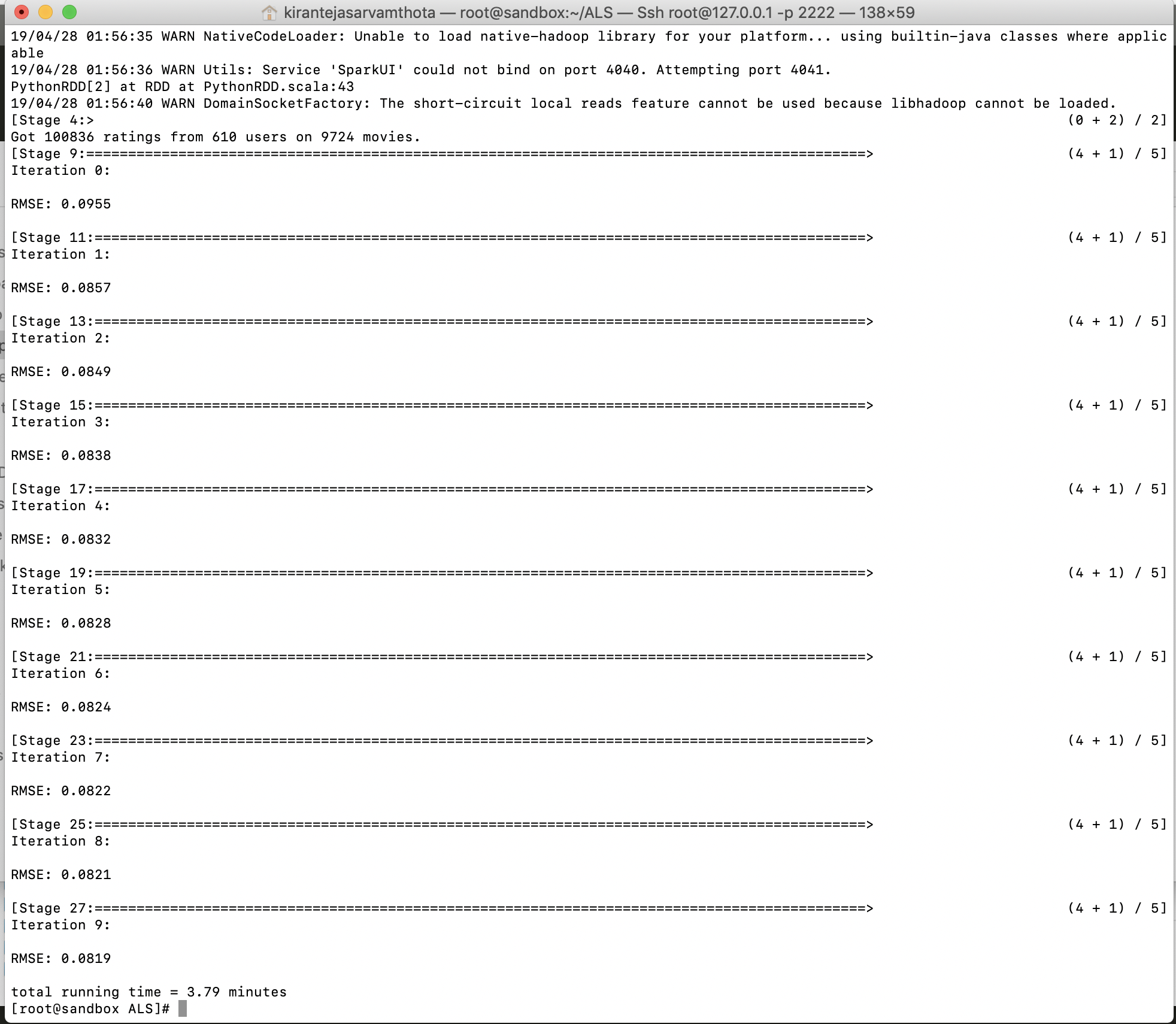


Total running time of the program is 7.01 minutes for 20 iterations for whole data.

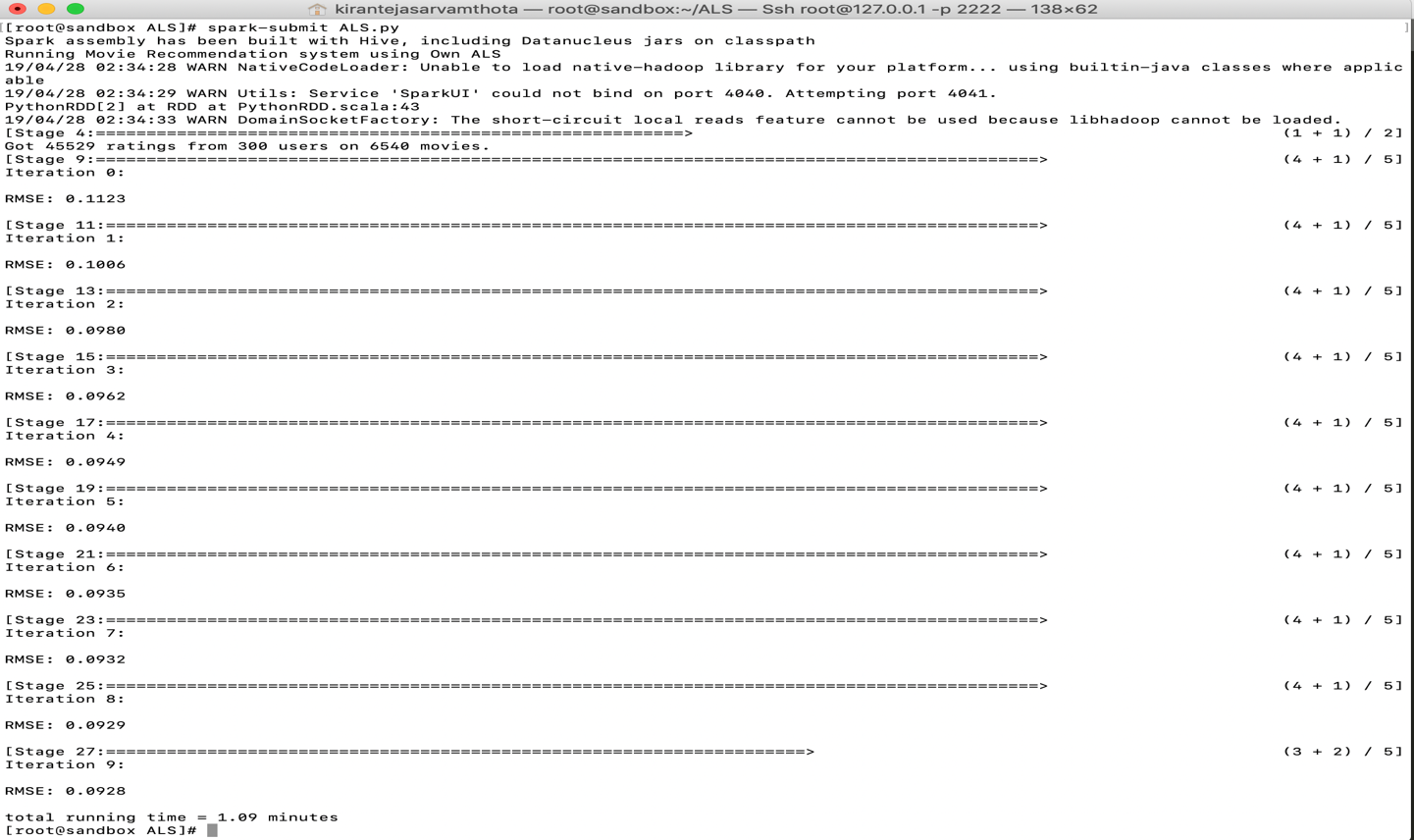


**Scalability**:

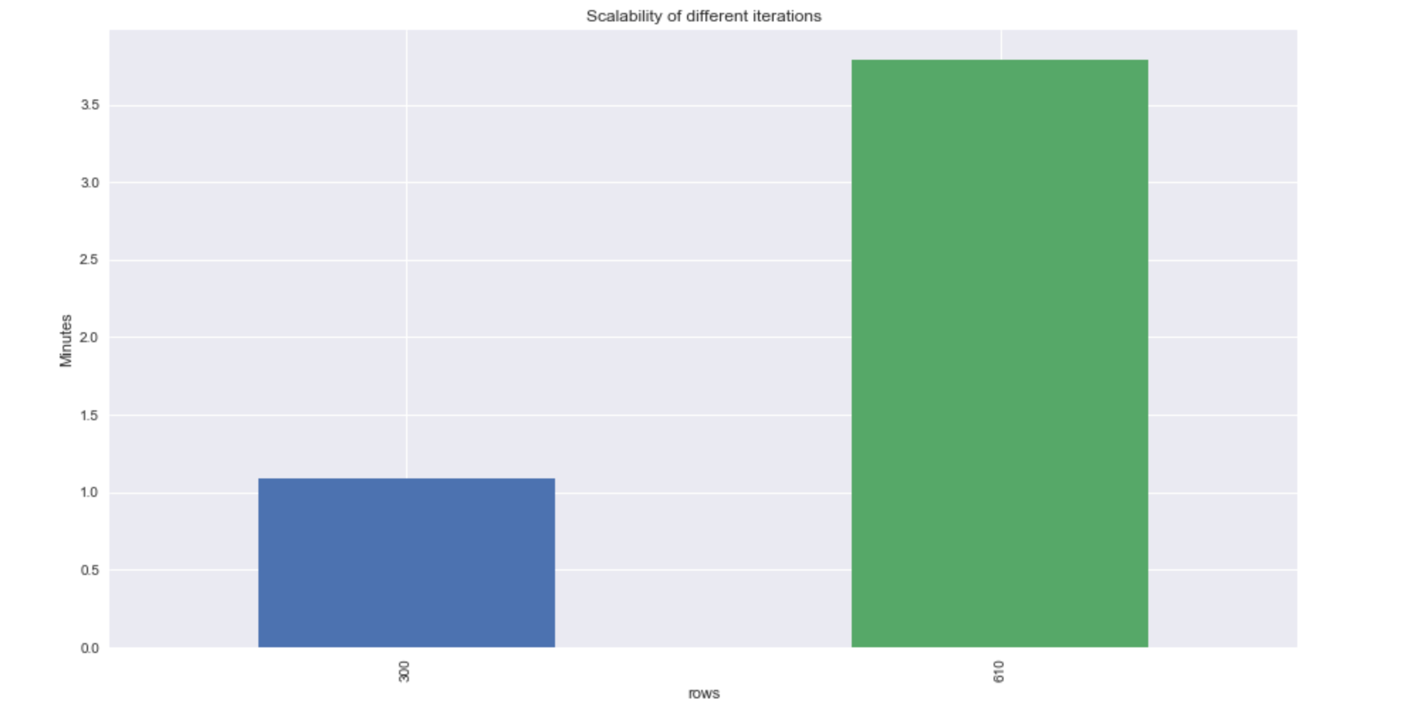
Total running time of the program is 3.79 minutes for 10 iterations on 610 users and 9724 movies.



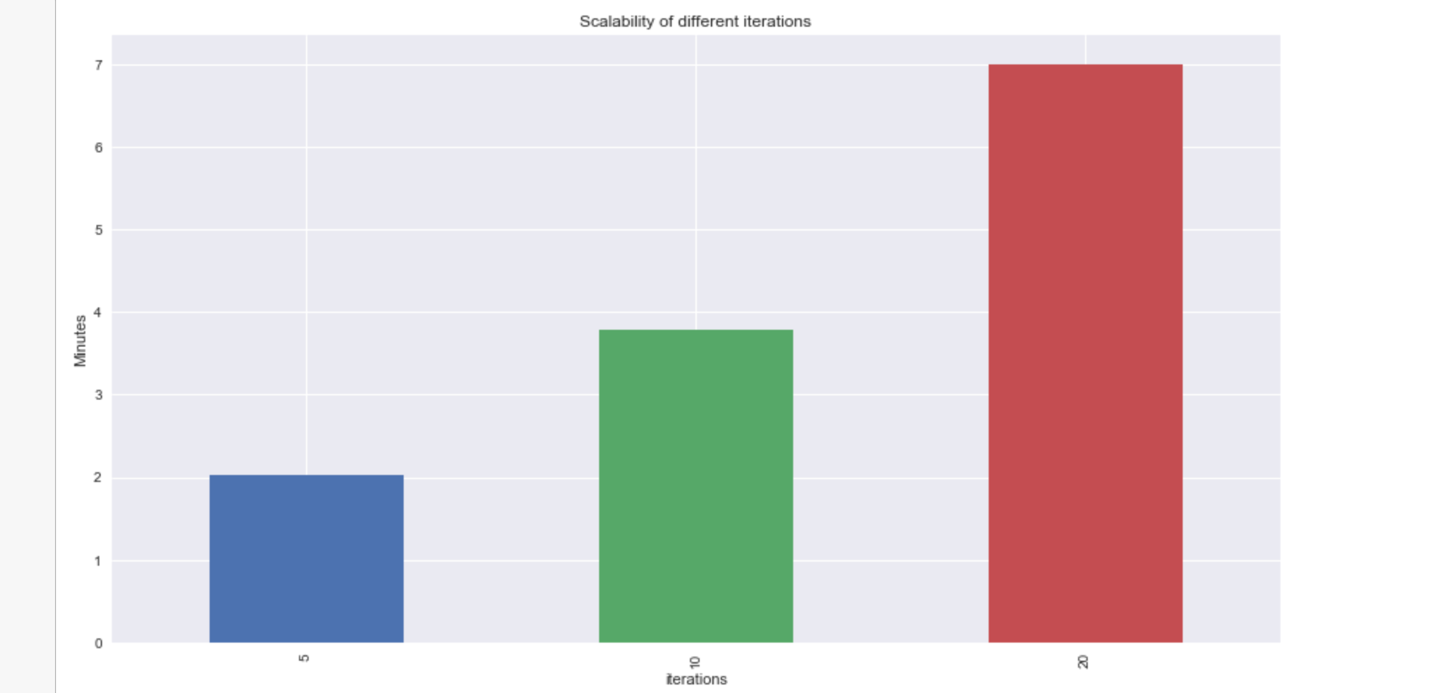
Total running time of the program is 3.79 minutes for 10 iterations on 300 users and 6540 movies.



Scalability for different rows of data:



Scalability of different iterations for whole data:



**Conclusion:** As part of this project I’ve implemented Movie Recommendation systems using collaborative filtering. I’ve solved the problem using matrix factorization method implemented Alternating least squares algorithm to achieve high performance in calculations in distributed environment (Apache Spark).By doing Normalization the RMSE error was reduced by 76 % from (0.3724 to 0.0817) also the running time is reduced. I’ve also scaled the model for different data. The Results I’ve observed is satisfactory and believe there is a lot of scope of improving the prediction model to bring accurate results.

**References:**

1. <https://bugra.github.io/work/notes/2014-04-19/alternating-least-squares-method-for-collaborative-filtering/>
2. <https://blog.insightdatascience.com/explicit-matrix-factorization-als-sgd-and-all-that-jazz-b00e4d9b21ea>
3. <https://blog.insightdatascience.com/explicit-matrix-factorization-als-sgd-and-all-that-jazz-b00e4d9b21ea>