Samuel Atkins, 1002951754
February 27, 2021
MIE1628: Big Data Science

PySpark Recommender System

1. Project Description

The applications in this project helped me gain practical experience with big-data tools, specifically, PySpark. Through the large-scale collaborative-filtering project and mini-applications implemented in this repo, I was able to gain a solid understanding of data handling and model formulation using PySpark.

In this repository, I implement simple map-reduce programs for counting odd/even integers, counting words in a text document, and calculating simple properties present in a supplied large data-frame.

I also implement a large-scale collaborative-filtering recommender system. This system takes in user rating data pertaining to various movie IDs and models the relationship between the users by utilizing the collaborative filtering approach with alternating least squares (ALS).

2. Setup

2.1 Organization

The scripts for each of the PySpark applications are present in the "scripts" folder. The results for each of the implementations are in the "results" folder. These results have also been appended to the end of this report. The outputs from each script have been included in the "outputs" folder.

2.2 Conda Environment Setup

Create environment from environment.yml:
From base directory:
conda env create -f ./environment.yml

Update environment from environment.yml:
From base directory and after activating existing environment:
conda env update --file ./environment.yml

2.3 Execution

After creating a conda environment using the supplied environment.yml file, inspect the files within the "scripts" folder to understand the purpose of each PySpark application. Then, simply run each of the scripts and observe the output.

NOTE: the recommender_sys.py file has a question flag that enables different branches of the script to run. Be sure to set this flag if you wish to control the flow of the script

3. Insights

3.1 RMSE vs. MSE vs. MAE

According to Piazza, there was a mistake in the assignment, and we were meant to compare RMSE with MAE. This makes more sense as it is rare that model developers need to discern between RMSE and MSE. Nonetheless, we will still compare all three metrics to be thorough. Mean absolute error (MAE) measures the average absolute difference between the predicted and true data. It does not punish large prediction errors. Mean squared error (MSE) measures the squared average distance between the predicted and true data. Large errors are heavily penalized. The problem with using the MSE is that the error units are squared which is not justified. Root mean squared error (RMSE) is the square root of the mean squared error. It penalizes large errors while also rectifying the squared error unit problem. The main problem with the MSE and RMSE is they are very sensitive to outliers. MAE, on the other hand, penalizes all errors in the same way.

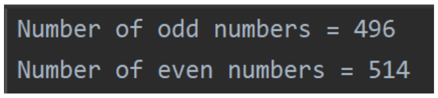
In the context of our recommender system, our dataset is quite small. Most movies are sparsely characterized. This likely implies that the standard deviation of the ratings distribution across the various movies and users is large. Therefore, outliers are prevalent. As noted above, the RMSE is quite sensitive to outliers. Perhaps the MAE would be a good choice in this situation as the MAE does not discriminate between small and large errors. Furthermore, if our recommender system is dynamic, i.e., if our recommender system needs to compute real-time prediction error, then perhaps the timing benefits of these metrics would become relevant. RMSE and MSE require an additional multiplication as they calculate the squared error. Therefore, MAE would once again be a better metric.

3.2 ALS Parameter Tuning

The parameters that I focused on for tuning were rank, regParam, and alpha. Rank is the total number of latent factors used by the model. The default value for rank is 10. After reading online that rank can vary from 5-200, I chose to test the following rank values: [5, 10, 20, 40, 80]. The regParam is the regularization parameter used by the ALS model. I decided to vary the regularization parameter using the following values: [0.1, 0.01, 0.001]. These are typical values used for regularization in other applications. Furthermore, after reading a little bit about other PySpark recommender systems, these values seemed to be commonplace. I also decided to vary the alpha parameter. This parameter controls the implicit feedback variant of ALS. It determines the model's baseline confidence with respect to its observations. I decided to vary this model using these values: [2, 3]. Using the built-in PySpark libraries, grid-search was conducted, and an optimal model was extracted from these parameters for train/test splits of 75/25 and 80/20.

4. Results

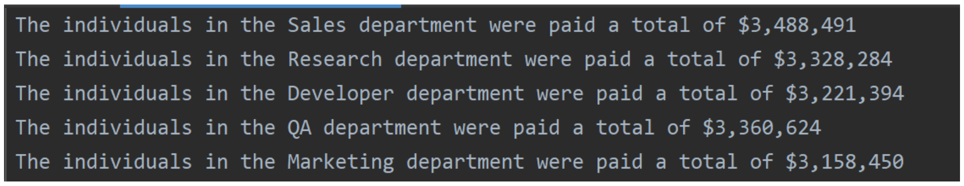
4.1 Odds/Evens



```
Number of odd numbers = 496
Number of even numbers = 514
```

Figure 1: Odds/evens program output

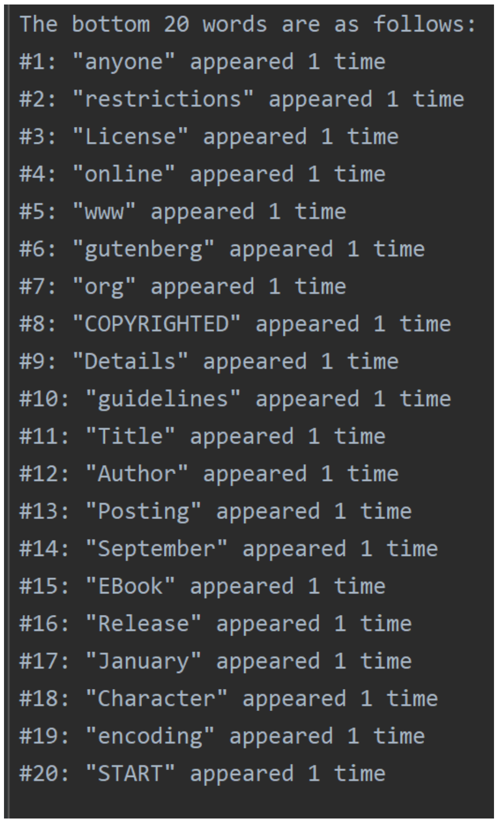
4.2 Department Salary



```
The individuals in the Sales department were paid a total of $3,488,491
The individuals in the Research department were paid a total of $3,328,284
The individuals in the Developer department were paid a total of $3,221,394
The individuals in the QA department were paid a total of $3,360,624
The individuals in the Marketing department were paid a total of $3,158,450
```

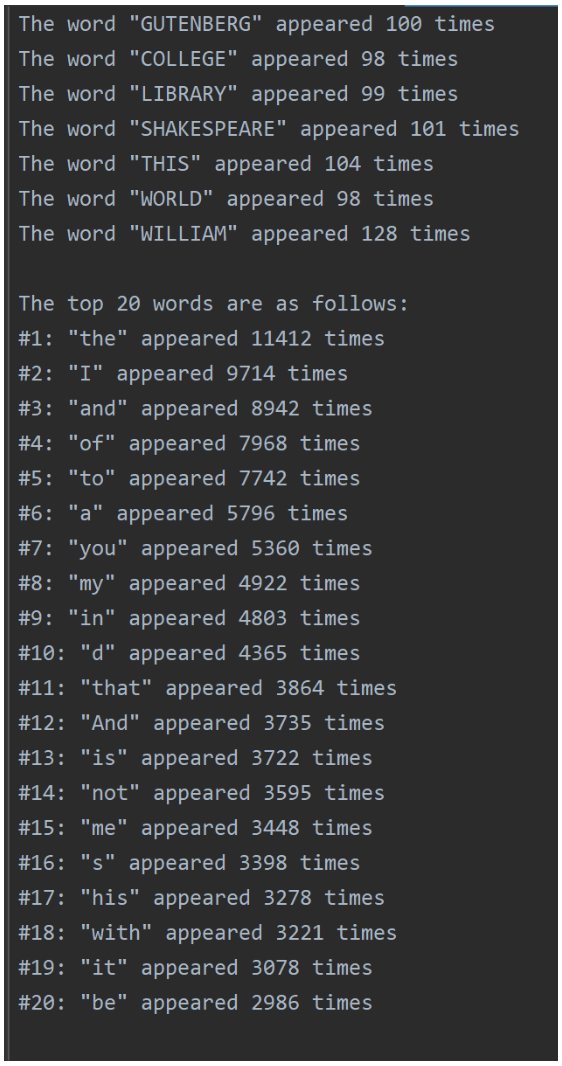
Figure 2: Department salary program output

4.3 Map-Reduce Word Count



```
The bottom 20 words are as follows:
#1: "anyone" appeared 1 time
#2: "restrictions" appeared 1 time
#3: "License" appeared 1 time
#4: "online" appeared 1 time
#5: "www" appeared 1 time
#6: "gutenberg" appeared 1 time
#7: "org" appeared 1 time
#8: "COPYRIGHTED" appeared 1 time
#9: "Details" appeared 1 time
#10: "guidelines" appeared 1 time
#11: "Title" appeared 1 time
#12: "Author" appeared 1 time
#13: "Posting" appeared 1 time
#14: "September" appeared 1 time
#15: "EBook" appeared 1 time
#16: "Release" appeared 1 time
#17: "January" appeared 1 time
#18: "Character" appeared 1 time
#19: "encoding" appeared 1 time
#20: "START" appeared 1 time
```

Figure 3: Map-reduce bottom 20 words

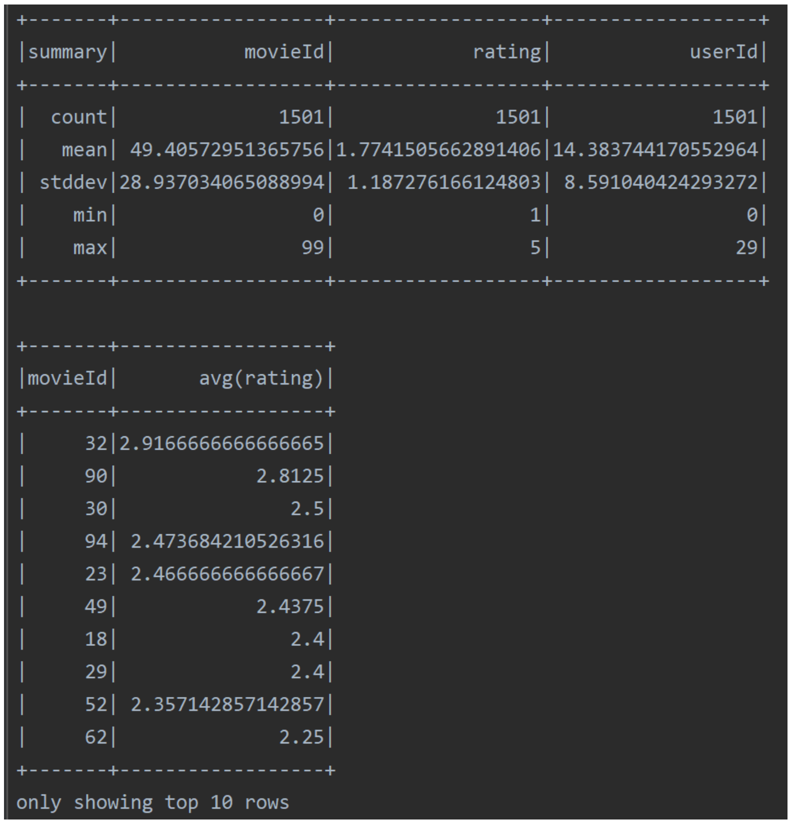


```
The word "GUTENBERG" appeared 100 times
The word "COLLEGE" appeared 98 times
The word "LIBRARY" appeared 99 times
The word "SHAKESPEARE" appeared 101 times
The word "THIS" appeared 104 times
The word "WORLD" appeared 98 times
The word "WILLIAM" appeared 128 times

The top 20 words are as follows:
#1: "the" appeared 11412 times
#2: "I" appeared 9714 times
#3: "and" appeared 8942 times
#4: "of" appeared 7968 times
#5: "to" appeared 7742 times
#6: "a" appeared 5796 times
#7: "you" appeared 5360 times
#8: "my" appeared 4922 times
#9: "in" appeared 4803 times
#10: "d" appeared 4365 times
#11: "that" appeared 3864 times
#12: "And" appeared 3735 times
#13: "is" appeared 3722 times
#14: "not" appeared 3595 times
#15: "me" appeared 3448 times
#16: "s" appeared 3398 times
#17: "his" appeared 3278 times
#18: "with" appeared 3221 times
#19: "it" appeared 3078 times
#20: "be" appeared 2986 times
```

Figure 4: Map-reduce desired word counts and top 20 words

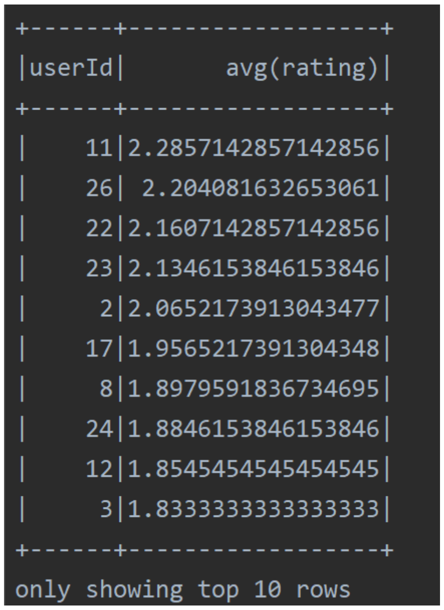
4.4 Recommender System Data Exploration



```
+-------+-----------------+-----------------+-----------------+
|summary|          movieId|           rating|           userId|
+-------+-----------------+-----------------+-----------------+
|  count|             1501|             1501|             1501|
|   mean| 49.40572951365756|1.774150566291406|14.383744170552964|
| stddev|28.937034065088994| 1.187276166124803| 8.591040424293272|
|    min|                0|                1|                0|
|    max|               99|                5|               29|
+-------+-----------------+-----------------+-----------------+


+-------+-----------------+
|movieId|      avg(rating)|
+-------+-----------------+
|     32|2.9166666666666665|
|     90|           2.8125|
|     30|              2.5|
|     94| 2.473684210526316|
|     23| 2.466666666666667|
|     49|           2.4375|
|     18|              2.4|
|     29|              2.4|
|     52| 2.357142857142857|
|     62|             2.25|
+-------+-----------------+
only showing top 10 rows
```

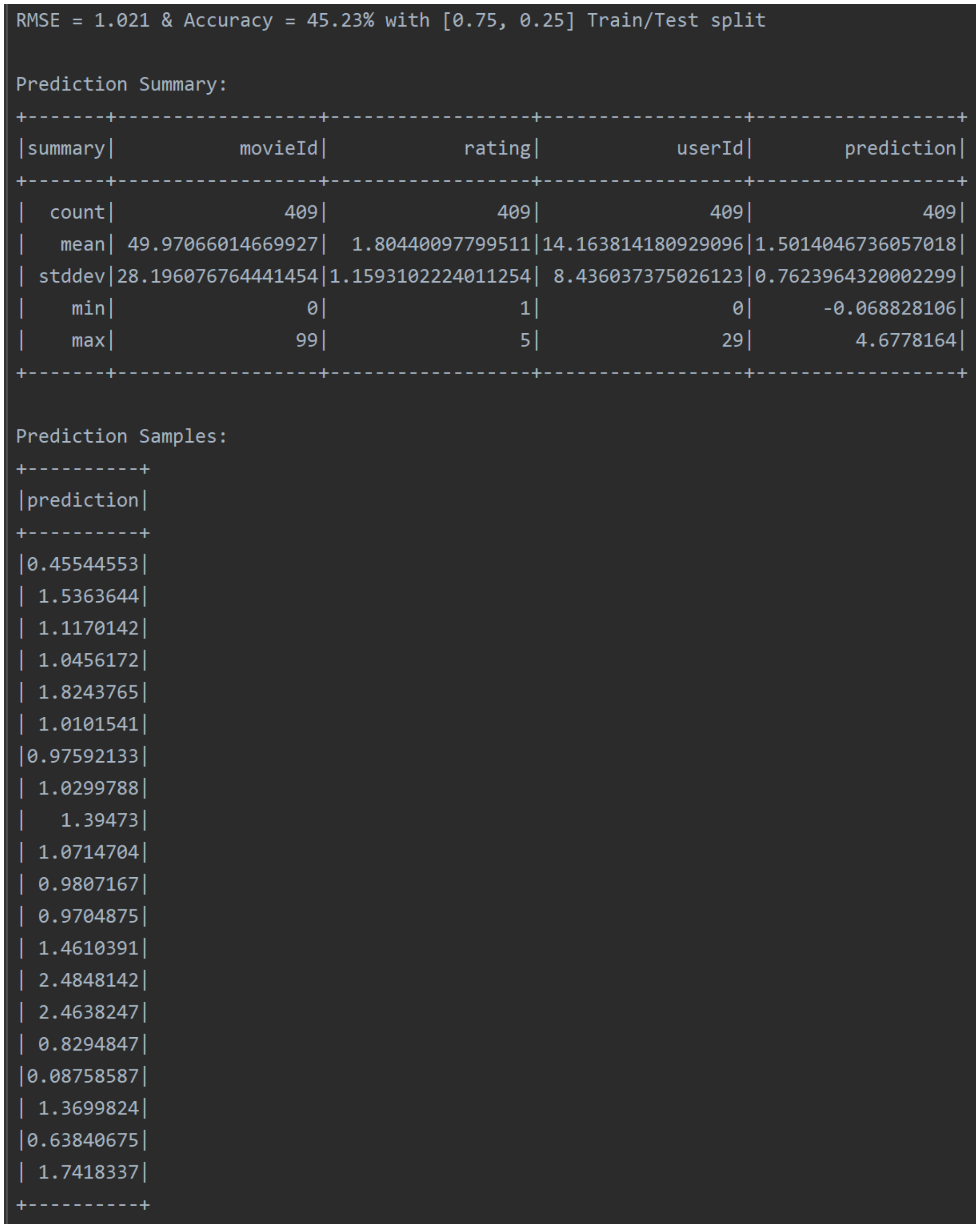
Figure 5: Described dataset and top 10 movies



```
+------+------------------+
|userId|       avg(rating)|
+------+------------------+
|    11|2.2857142857142856|
|    26| 2.204081632653061|
|    22|2.1607142857142856|
|    23|2.1346153846153846|
|     2|2.0652173913043477|
|    17|1.9565217391304348|
|     8|1.8979591836734695|
|    24|1.8846153846153846|
|    12|1.8545454545454545|
|     3|1.8333333333333333|
+------+------------------+
only showing top 10 rows
```

Figure 6: Top 10 users

4.5 Collaborative Filtering Initial Implementation

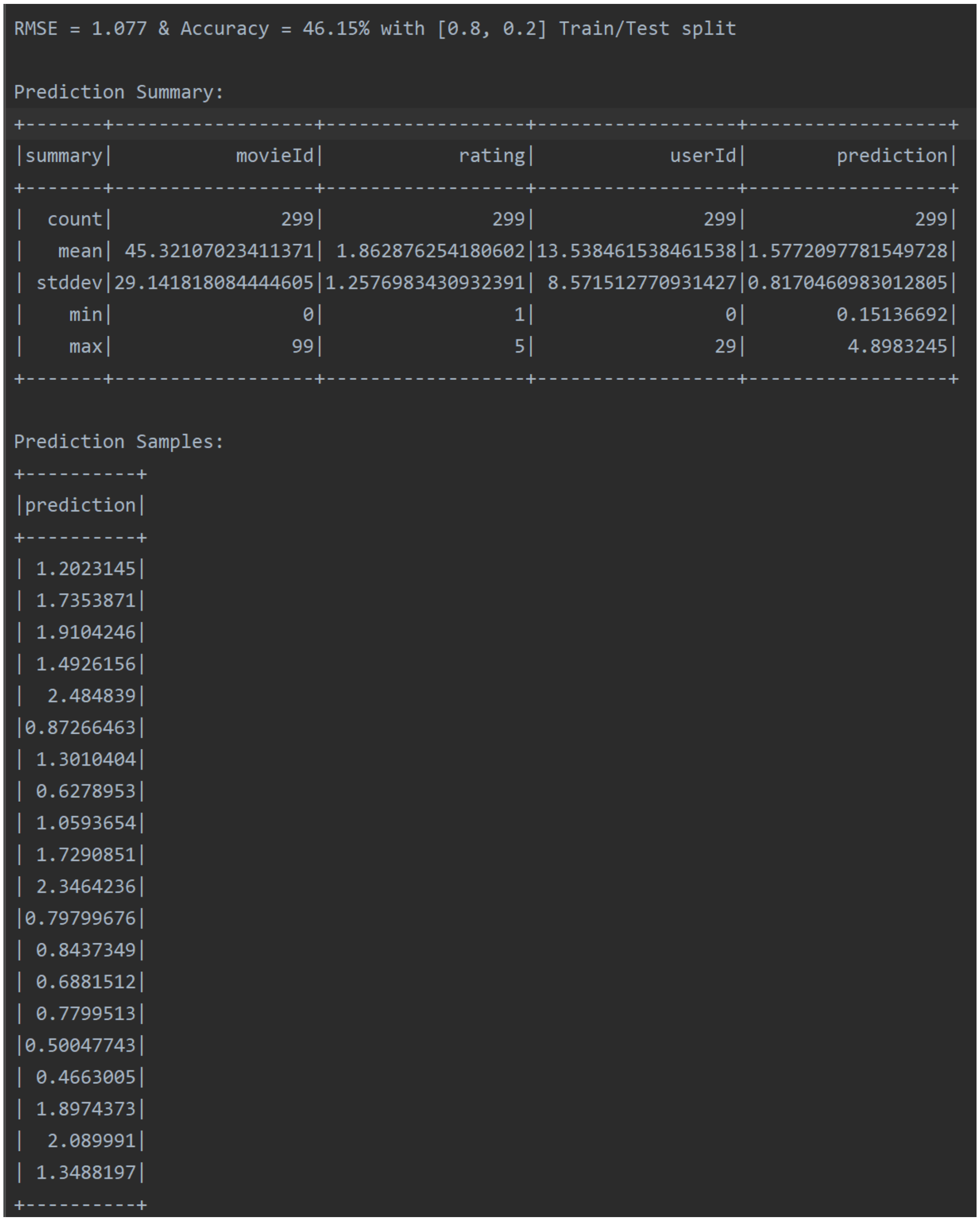


```
RMSE = 1.021 & Accuracy = 45.23% with [0.75, 0.25] Train/Test split

Prediction Summary:
+-------+------------------+------------------+------------------+------------------+
|summary|           movieId|            rating|            userId|        prediction|
+-------+------------------+------------------+------------------+------------------+
|  count|               409|               409|               409|               409|
|   mean| 49.97066014669927|  1.80440097799511|14.163814180929096|1.5014046736057018|
| stddev|28.196076764441454|1.1593102224011254| 8.436037375026123|0.7623964320002299|
|    min|                 0|                 1|                 0|      -0.068828106|
|    max|                99|                 5|                29|         4.6778164|
+-------+------------------+------------------+------------------+------------------+

Prediction Samples:
+----------+
|prediction|
+----------+
|0.45544553|
| 1.5363644|
| 1.1170142|
| 1.0456172|
| 1.8243765|
| 1.0101541|
|0.97592133|
| 1.0299788|
|   1.39473|
| 1.0714704|
| 0.9807167|
| 0.9704875|
| 1.4610391|
| 2.4848142|
| 2.4638247|
| 0.8294847|
|0.08758587|
| 1.3699824|
|0.63840675|
| 1.7418337|
+----------+
```

Figure 7: RMSE, accuracy, and prediction samples of initial implementation (75/25)



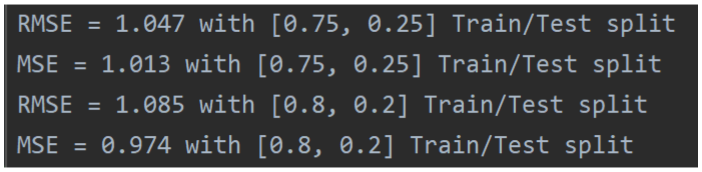
```
RMSE = 1.077 & Accuracy = 46.15% with [0.8, 0.2] Train/Test split

Prediction Summary:
+-------+------------------+------------------+------------------+------------------+
|summary|           movieId|            rating|            userId|        prediction|
+-------+------------------+------------------+------------------+------------------+
|  count|               299|               299|               299|               299|
|   mean| 45.32107023411371| 1.862876254180602|13.538461538461538|1.5772097781549728|
| stddev|29.141818084444605|1.2576983430932391| 8.571512770931427|0.8170460983012805|
|    min|                 0|                 1|                 0|        0.15136692|
|    max|                99|                 5|                29|         4.8983245|
+-------+------------------+------------------+------------------+------------------+

Prediction Samples:
+----------+
|prediction|
+----------+
| 1.2023145|
| 1.7353871|
| 1.9104246|
| 1.4926156|
|  2.484839|
|0.87266463|
| 1.3010404|
| 0.6278953|
| 1.0593654|
| 1.7290851|
| 2.3464236|
|0.79799676|
| 0.8437349|
| 0.6881512|
| 0.7799513|
|0.50047743|
| 0.4663005|
| 1.8974373|
|  2.089991|
| 1.3488197|
+----------+
```

Figure 8: RMSE, accuracy, and prediction samples of initial implementation (80/20)

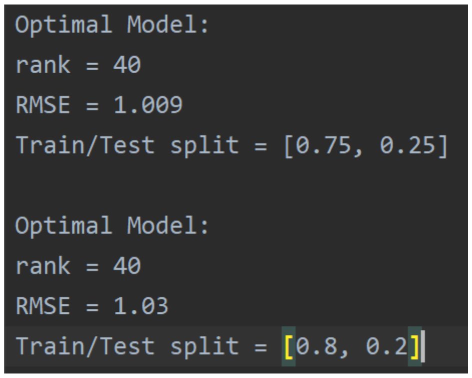
4.6 RMSE vs. MSE



```
RMSE = 1.047 with [0.75, 0.25] Train/Test split
MSE = 1.013 with [0.75, 0.25] Train/Test split
RMSE = 1.085 with [0.8, 0.2] Train/Test split
MSE = 0.974 with [0.8, 0.2] Train/Test split
```

Figure 9: RMSE vs. MSE for each split

4.7 Collaborative Filtering Cross-Validation

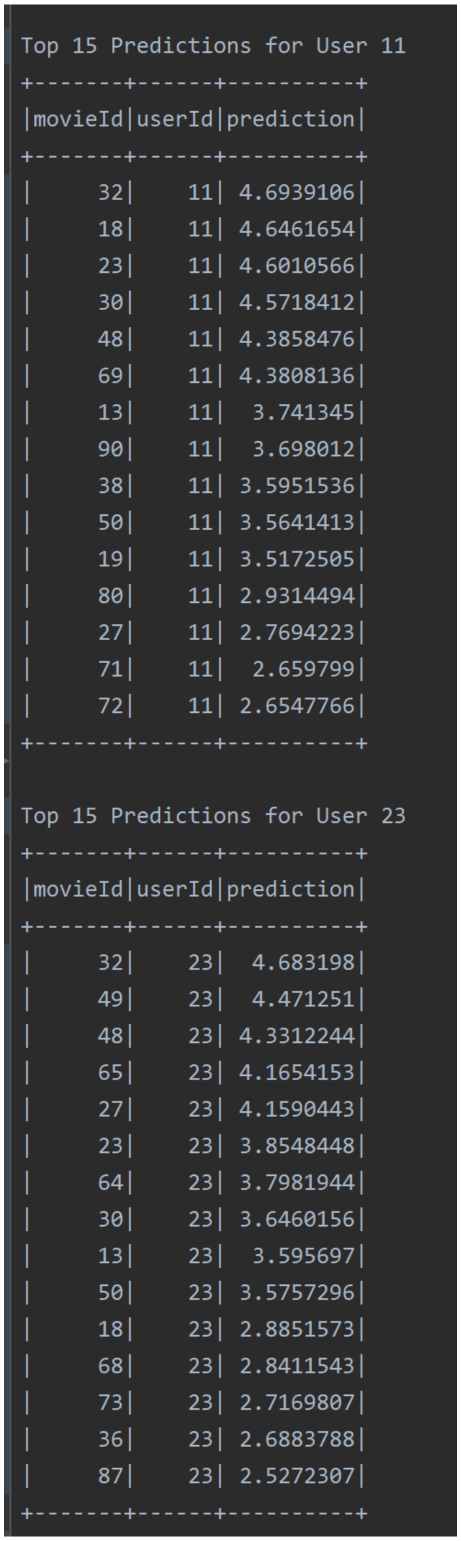


```
Optimal Model:
rank = 40
RMSE = 1.009
Train/Test split = [0.75, 0.25]

Optimal Model:
rank = 40
RMSE = 1.03
Train/Test split = [0.8, 0.2]
```

Figure 10: Collaborative filtering CV for each model

4.8 Top 15 Movies for User 11 and User 23



```
Top 15 Predictions for User 11
+-------+------+----------+
|movieId|userId|prediction|
+-------+------+----------+
|     32|    11| 4.6939106|
|     18|    11| 4.6461654|
|     23|    11| 4.6010566|
|     30|    11| 4.5718412|
|     48|    11| 4.3858476|
|     69|    11| 4.3808136|
|     13|    11|  3.741345|
|     90|    11|  3.698012|
|     38|    11| 3.5951536|
|     50|    11| 3.5641413|
|     19|    11| 3.5172505|
|     80|    11| 2.9314494|
|     27|    11| 2.7694223|
|     71|    11|  2.659799|
|     72|    11| 2.6547766|
+-------+------+----------+

Top 15 Predictions for User 23
+-------+------+----------+
|movieId|userId|prediction|
+-------+------+----------+
|     32|    23|  4.683198|
|     49|    23|  4.471251|
|     48|    23| 4.3312244|
|     65|    23| 4.1654153|
|     27|    23| 4.1590443|
|     23|    23| 3.8548448|
|     64|    23| 3.7981944|
|     30|    23| 3.6460156|
|     13|    23|  3.595697|
|     50|    23| 3.5757296|
|     18|    23| 2.8851573|
|     68|    23| 2.8411543|
|     73|    23| 2.7169807|
|     36|    23| 2.6883788|
|     87|    23| 2.5272307|
+-------+------+----------+
```

Figure 11: Top 15 predictions for User 11 and User 23