**Q3.1.1**

*Program output:*

"GlobalAverageRating" : 3.529859999999947

* Ratings on average does not coincide with the middle of the rating scale 3.
* The ratings on average are higher than the middle of the rating scale, by about 0.530.

**Q3.1.2**

*Program output:*

"UsersAverageRating" : {

"min" : 1.4919540229885058,

"max" : 4.869565217391305,

"average" : 3.5881911071848003

},

"AllUsersCloseToGlobalAverageRating" : false,

"RatioUsersCloseToGlobalAverageRating" : 0.7465535524920467

* **Not** all users rate, on average, close to the global average.
* The min of average rating of a user is about 1.492 and the max is about 4.870.
* Yes, most users rate, on average, close to the global average.
* The ratio of users with average ratings that deviate with less than 0.5 from the global average is about 0.747, i.e. about 74.7%

Compute the average rating for each user (\_ru;\_). Do all users

rate, on average, close to the global average? Check min and max for user

average and assume a di\_erence less than 0:5 is small. Do most users rate,

on average, close to the global average? Calculate and report the ratio of

users with average ratings that deviate with less than 0:5 from the global

average.

**Q3.1.3**

*Program output:*

"ItemsAverageRating" : {

"min" : 1.0,

"max" : 5.0,

"average" : 3.076044508325173

},

"AllItemsCloseToGlobalAverageRating" : false,

"RatioItemsCloseToGlobalAverageRating" : 0.48989298454221164

* **Not** all items rated, on average, close to the global average.
* The min of item average rating is 1.0 and the max is 5.0.
* No, it is not the case that most items rated, on average, close to the global average.
* The ratio of items with average ratings that deviate with less than 0.5 from the global average is about 0.490, i.e. about 49.0%

Compute the average rating for each item (\_r\_;i). Are all items

rated, on average, close to the global average? Check min and max for

item average and assume a di\_erence less than 0:5 is small. Are most

items rated, on average, close to the global average? Calculate and report

the ratio of items with average ratings that deviate with less than 0:5 from

the global average.

**Q3.1.4**

*Program output:*

"MaeGlobalMethod" : 0.9680487749999757,

"MaePerUserMethod" : 0.8501912740150447,

"MaePerItemMethod" : 0.8275684032890022,

"MaeBaselineMethod" : 0.7788884667137885

|  |  |
| --- | --- |
| Method | MAE |
| Global Method | 0.968 |
| Per User Method | 0.850 |
| Per Item Method | 0.828 |
| Baseline Method | 0.779 |

* Global method has the largest error because for each (user, item) with unknown rating, it does not adjust the prediction to the specific user nor the specific item. Instead it blindly predicts the rating to be the global average.
* Both per user and per item method has much smaller error than the global method because per user method customizes the prediction to the specific user’s average rating and the per item method customizes the prediction to the specific item’s average rating.
* Per item method’s accuracy is a bit better than per user method. This is probably due to, according to Q3.1.3, item average ratings have relatively large deviation whereas user average rating has relatively small deviation (most are close to the average). So the difference between ratings is more due to different items than due to different users. Hence when given a (user, item) pair, use the item average yields better performance than using user average.
* Baseline method takes both user and item into account. For a (u, i) pair, it utilizes both user u’s average and the average of the normalized deviations for all users on item I, and hence it has the best accuracy.

**Q3.1.5**

*Program output:*

"DurationInMicrosecForPerItemMethod" : {

"min" : 784713.8,

"max" : 865166.9,

"average" : 832629.73,

"stddev" : 27971.31885446412

},

"RatioBetweenBaselineMethodAndGlobalMethod" : 14.93106694022548,

"DurationInMicrosecForGlobalMethod" : {

"min" : 162257.9,

"max" : 207386.1,

"average" : 180777.7,

"stddev" : 14432.506858477494

},

"DurationInMicrosecForPerUserMethod" : {

"min" : 946202.8,

"max" : 1156321.1,

"average" : 1050266.3699999999,

"stddev" : 53097.039570960056

},

"DurationInMicrosecForBaselineMethod" : {

"min" : 2515336.8,

"max" : 3093911.0,

"average" : 2699203.94,

"stddev" : 183978.10227339674

}

Measure each method each time, the stats are summarized in the following table. The unit is microsecond.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **min** | **max** | **average** | **stddev** |
| **Global Method** | **162257.9** | **207386.1** | **180777.7** | **14432.507** |
| **Per User Method** | **946202.8** | **1156321.1** | **1050266.370** | **53097.040** |
| **Per Item Method** | **784713.8** | **865166.9** | **832629.73** | **27971.319** |
| **Baseline Method** | **2515336.8** | **3093911.0** | **2699203.94** | **183978.102** |

*Technical Specification:*

Model: XPS13 9360

CPU speed: 2.50GHz

RAM: 8.00 GB

OS: Microsoft Windows 10 Home [Version 10.0.19041 Build 19041]

Language version: Scala version is 2.12.13. Spark version is 3.0.0

The baseline method is the most expensive to compute.

The baseline method takes 2518426.24 more microseconds (2.518 seconds) to compute. The baseline method’s duration is roughly 15times as long as the global method.

The ratio between the average time for computing the baseline and the average time for computing the global average is 14.931.

Measure the time required for computing predictions for all rat-

ings in the test set (ml-100k/u1.text) with all four methods by recording

the current time before and after (ex: with System.nanoTime() in Scala).

The duration is the di\_erence between the two. For all four methods,

perform ten measurements and report in a table the min, max, average,

and standard-deviation. Report also the technical speci\_cations (model,

CPU speed, RAM, OS, language version) of the machine on which you

ran the tests. Which of the four prediction methods is the most expen-

sive to compute? How much more compared to using the global average

rating (\_r\_;\_)? Calculate and report the ratio between the average time for

computing the baseline (Eq. 5) and the average time for computing the

global average.

**Q4.1.1**

[ [ 814, "Great Day in Harlem", 5.0 ], [ 1122, "They Made Me a Criminal (1939)", 5.0 ], [ 1189, "Prefontaine (1997)", 5.0 ], [ 1201, "Marlene Dietrich: Shadow and Light (1996)", 5.0 ], [ 1293, "Star Kid (1997)", 5.0 ] ]

These movies are not I have liked and I wouldn’t want to see them in the future… Investigating these movies more closely, I found that although they have 5.0 ratings, they have only rated by *3 out of 944* users. So they are those unpopular movies that only few people would watch them, but these few people really love them…

**Q4.1.2**

[ [ 50, "Star Wars (1977)", 4.242456848614165 ], [ 64, "Shawshank Redemption", 4.142979920875009 ], [ 483, "Casablanca (1942)", 4.126820416653099 ], [ 174, "Raiders of the Lost Ark (1981)", 4.036976486596832 ], [ 100, "Fargo (1996)", 3.9933004278119273 ] ]

To take popularity into account and favour more popular items, instead of predicting using *Pu,i* , we multiply *Pu,i* by a *popularity factor* to get *adjustedPu,i* and use it as the final prediction.

denotes number of ratings *item* *i* has and denotes the maximum number of ratings an item have. For example, item 50 is *Star Wars (1977)* which has 583 ratings, so . And 583 also happens to be the maximum number of ratings an item have, i.e. for our dataset.

gives the relative popularity of *item i* comparing to the most popular item in the dataset. Note that this number is between . Then we raise this ratio to the power of 1/16 to make it smoother. **When *item i* has few ratings popularity factor decreases ratings smoothly**, for example, if (item i only has 1/100 many ratings as the most rated item, i.e. roughly 6 ratings), then

;

**When *item i* has many ratings, popularity factor makes little impact**. For example if ,(i.e. item i has roughly 583\*0.75=437 ratings ) then