Gene Livshin, Harish Visweswaraiya

MovieLens recommender system

11/08/2015

Contents

[Question 1 – Data Exploration 2](#_Toc434780340)

[Overall Distribution of Ratings: 2](#_Toc434780341)

[Summary 2](#_Toc434780342)

[Average ratings 3](#_Toc434780343)

[Summary 3](#_Toc434780344)

[Summary 4](#_Toc434780345)

[Number of movies rated by Users 5](#_Toc434780346)

[Summary 5](#_Toc434780347)

[Number of Ratings received by Movies 5](#_Toc434780348)

[Summary 6](#_Toc434780349)

[Distribution of rating levels 6](#_Toc434780350)

[Summary 6](#_Toc434780351)

[Summary 7](#_Toc434780352)

[Question 2 – Modeling Predicted Ratings 7](#_Toc434780353)

[Global Average Method 7](#_Toc434780354)

[User-Item Baseline Method 8](#_Toc434780355)

[Summary of Global Average and Baseline estimates method 8](#_Toc434780356)

[Matrix Factorization 9](#_Toc434780357)

[Summary 11](#_Toc434780358)

[User k-NN & Item k-NN 12](#_Toc434780359)

[Question 3 – Recommender System 16](#_Toc434780360)

[User-Item Baseline 17](#_Toc434780361)

[Matrix Factorization 18](#_Toc434780362)

[Item-knn 19](#_Toc434780363)

[User-knn 19](#_Toc434780364)

[Summary 20](#_Toc434780365)

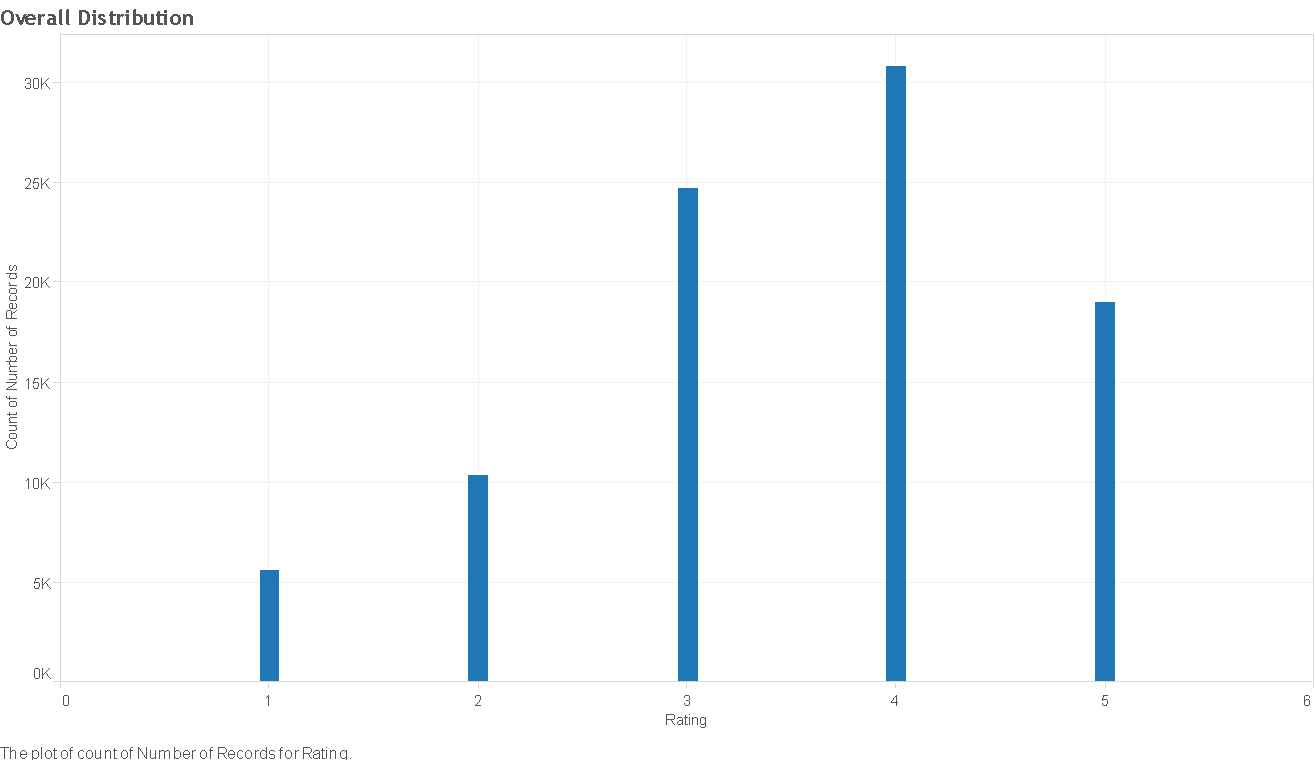
[Appendix 21](#_Toc434780366)

# Question 1 – Data Exploration

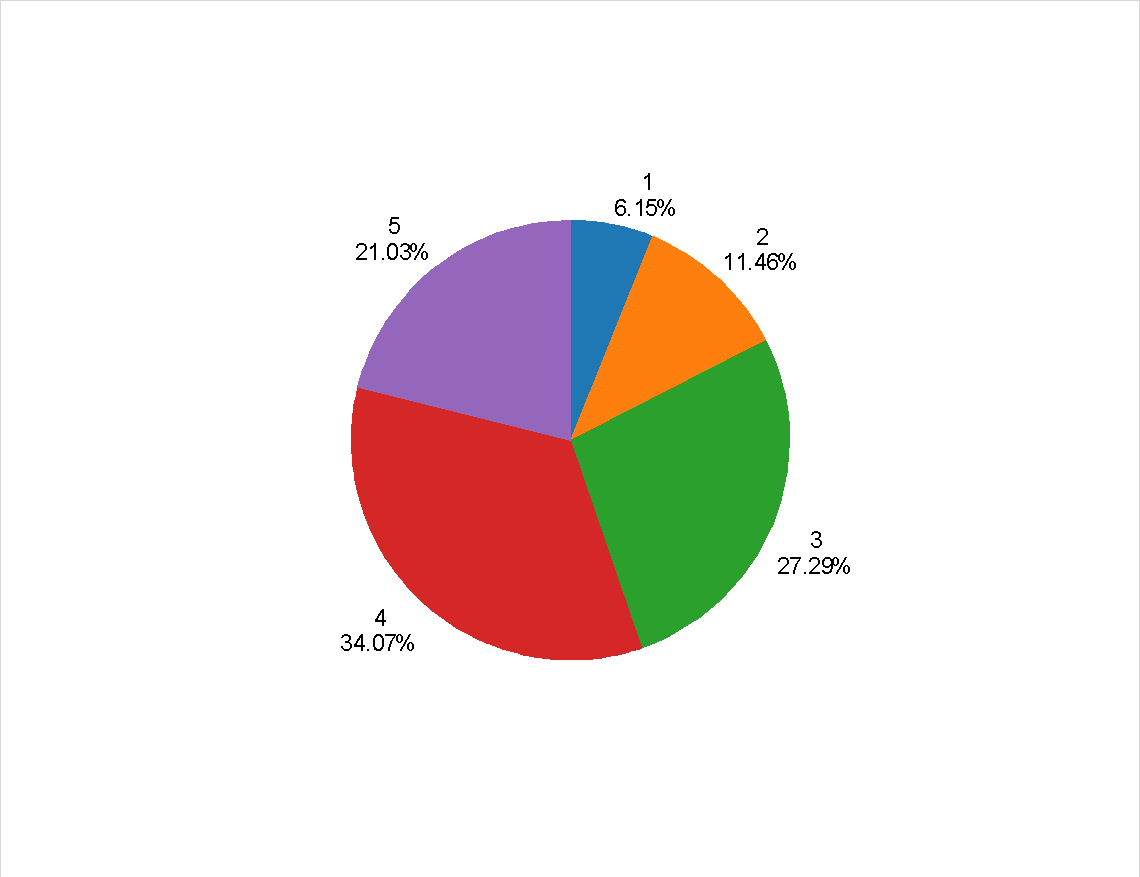
We explored the MovieLens data to find the distribution of ratings across movies and users.

## Overall Distribution of Ratings:

Below plot shows the respective rating frequency in the data set.



Following pie-chart explains the % distribution for respective ratings

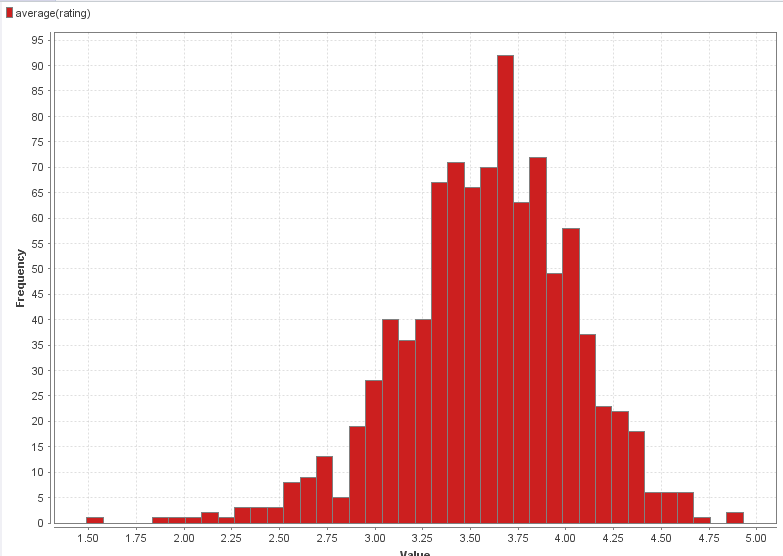


### Summary

1. Movie ratings are in the range 1-5 ( Low to High)
2. Most common rating is 4 (34.07%) for the given data set
3. More than 82% of the records are rated 3 or above.

## Average ratings

**Group by users**

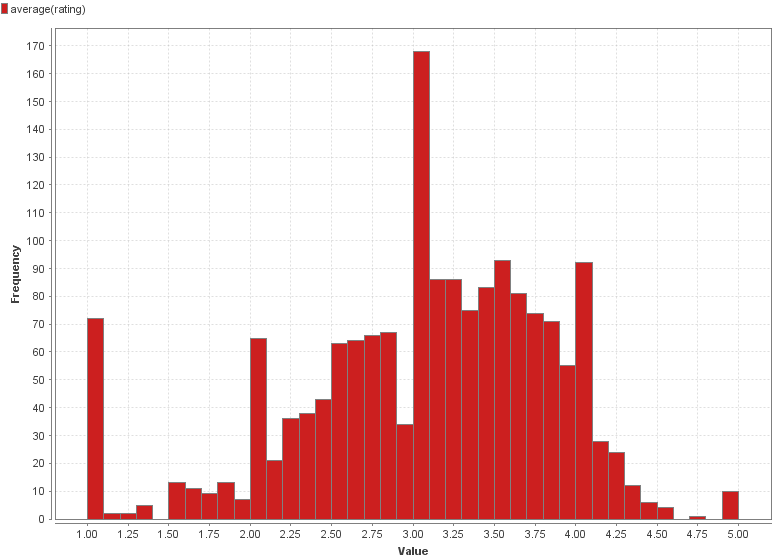
The below chart is plotted by Average rating given by users against the number of users. 

### Summary

1. Majority of average ratings given by users falls between 3 & 4.25. We could interpret that out of 943 users, most of them have rated the movies when they liked it
2. Number of users who give low average ratings (< 2) are quite low
3. Few users always give high ratings (Avg rating > 4.25)

**Group by items**

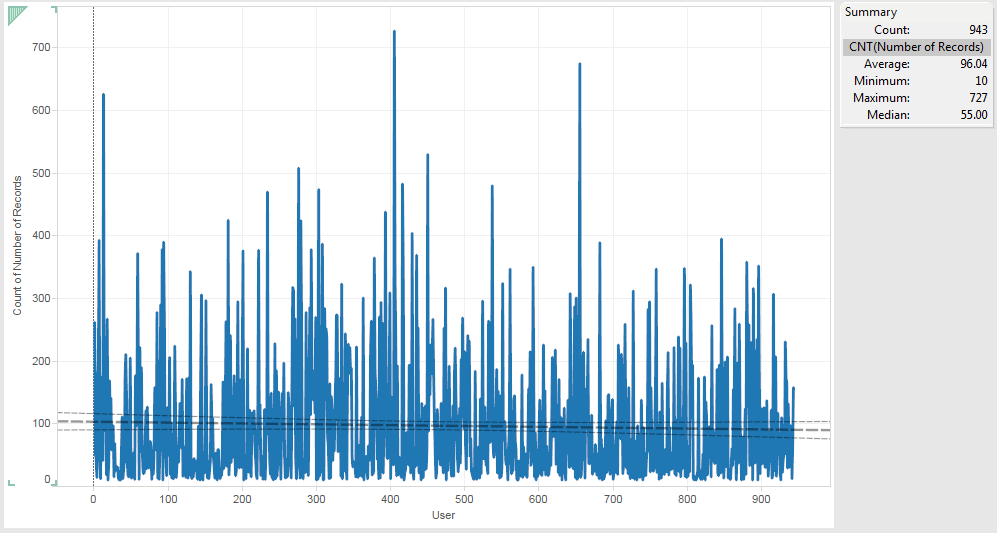
The below chart is plotted by Average rating given by users against the number of users. We identified



### Summary

1. Majority of average ratings given for 1682 movies falls between 2 & 4.25
2. Few movies have been less than 2 and more than 4.25
3. About 70 movies have always received extremely low rating (Rating = 1) from all users

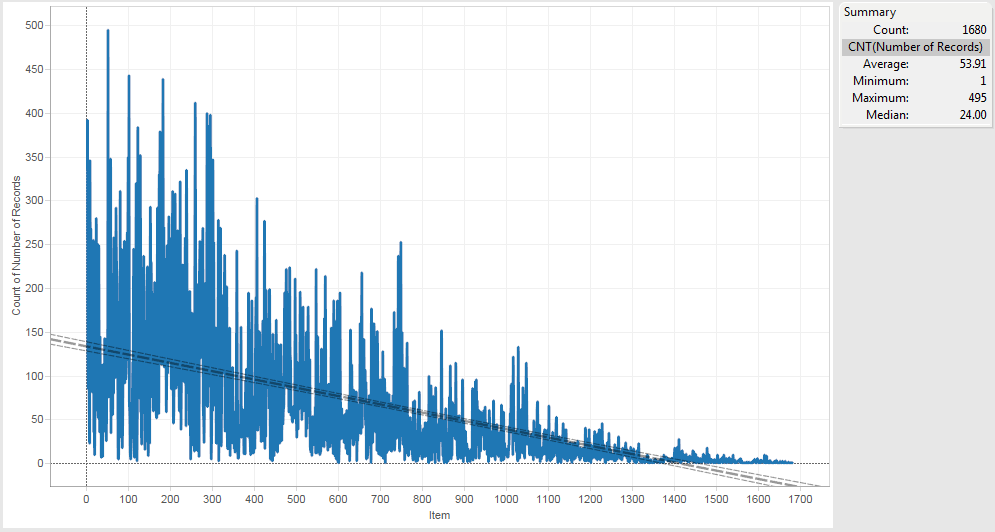
## Number of movies rated by Users



### Summary

1. We could see that median is 55. i.e most of users preferred to rate about 50+ movies
2. Average number of movies rated by 943 users is 96, which is significantly greater than the median. It denotes that some users have rated more number of movies than normal
3. Highest number of movies rated by a user is 727

## Number of Ratings received by Movies

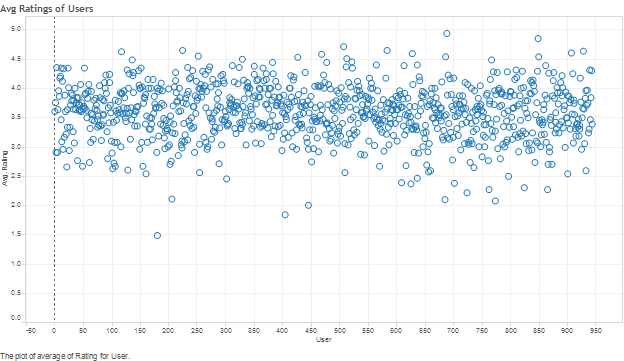


### Summary

1. Median Ratings received by movies is 24
2. Some movies have received fairly larger number of ratings as we see the average ratings received by a movie is about 54
3. Highest number of ratings received by a movie is 495

## Distribution of rating levels

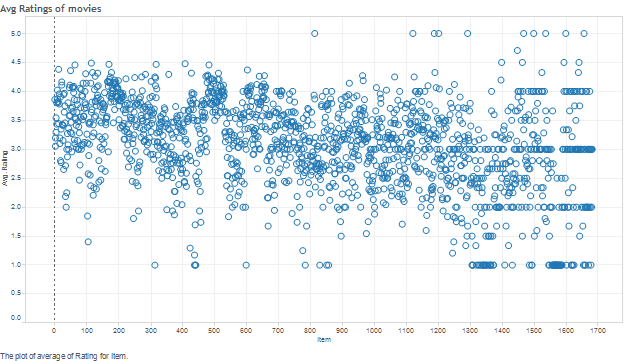
**Based on Users**



### Summary

1. From the scatter plot, we see that most of the users have given above average ratings, between 3 and 4 (Range of ratings: 1 - 5; Average - 3)
2. Few users have given high / low ratings that are either greater than 4.5 or less than 2

**Based on Movies**



### Summary

1. Again, we observe from the scatter plot that most of the movies are rated between 2 & 4
2. There are few movies which received high / low ratings that are either greater than 4.25 or less than 1.5

# Question 2 – Modeling Predicted Ratings

Performance measure used here is **‘Accuracy’** and we calculated different errors to evaluate the performance. They are as follows:

**Root mean square error (RMSE)**

* Same scale as original ratings & greater effect of higher error values because error values are squared

**Mean absolute error (MAE)**

* Same scale as original ratings but uses the absolute error values

**Normalized mean absolute error (NMAE)**

* Uses a normalized scale to calculate the error

## Global Average Method

Results noted for Global Average Method are given below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Parameters** | **Training Performance** | **Testing Performance** |
| Global Average | No parameters | RMSE: 1.126  MAE: 0.945  NMAE: 0.236 | RMSE: 1.122  MAE: 0.945  NMAE: 0.236 |

## User-Item Baseline Method

Performance measures were noted for user-item baseline method. Results are given below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Parameters** | **Training Performance** | **Testing Performance** |
| User-Baseline | num iter = 30  reg u = 15  reg i = 10 | RMSE: 0.920  MAE: 0.729  NMAE: 0.182 | RMSE: 0.963  MAE: 0.765  NMAE: 0.191 |
| User-Baseline | num iter = 30  reg u = 30  reg i = 30 | RMSE: 0.936  MAE: 0.745  NMAE: 0.186 | RMSE: 0.976  MAE: 0.778  NMAE: 0.195 |
| User-Baseline | num iter = 1000  reg u = 15  reg i = 10 | RMSE: 0.920  MAE: 0.729  NMAE: 0.182 | RMSE: 0.963  MAE: 0.765  NMAE: 0.191 |

## Summary of Global Average and Baseline estimates method

Global average had no parameters and error ratings were of the order of 1.2, which is quite high, on a scale of 1-5. We also noted that number of iterations and regularization parameters can be modified in user-item baseline operator. Obviously, it performed better than Global Average method.

|  |  |
| --- | --- |
| **Parameters of model** | **Significance of parameters** |
| num iter – Number of iterations | Default value of 10. Increasing/ Decreasing the iterations didn’t make any difference in accuracy measures |
| reg u – regularization parameter for user bias | Default value of 15. Increasing the value to 30 decreased the difference between training & testing errors (reduced over fitting) |
| reg i – regularization parameter for user bias | Default value of 10. Increasing the value to 30 decreased the difference between training & testing errors (reduced over fitting) |

## Matrix Factorization

Performance measures were noted for Matrix Factorization (MF) method. Results are given below:

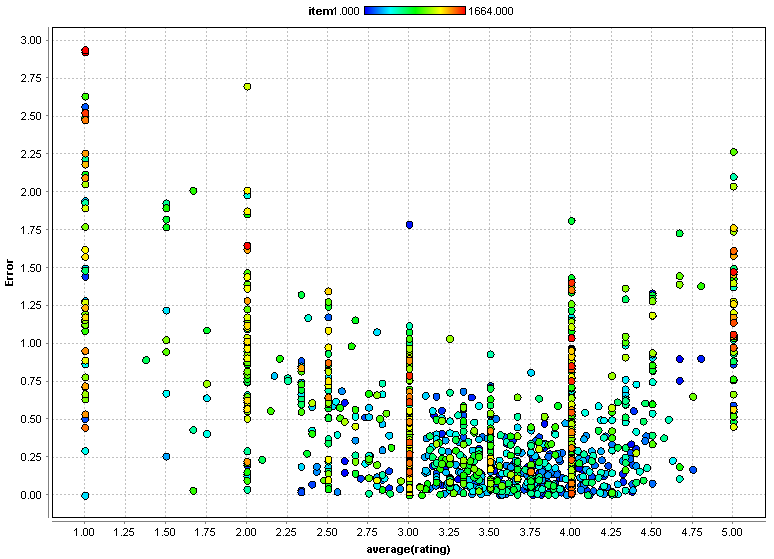
|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Parameters** | **Training Performance** | **Testing Performance** |
| MF | num factors = 10  Learn Rate = 0.01  Iteration number = 30  Regularization = 0.015  S.D = 0.1 | RMSE: 0.743  MAE: 0.582  NMAE: 0.146 | RMSE: 0.977  MAE: 0.768  NMAE: 0.192 |
| MF | num factors = 30  Learn Rate = 0.01  Iteration number = 30  Regularization = 0.015  S.D = 0.1 | RMSE: 0.554  MAE: 0.432  NMAE: 0.108 | RMSE: 1.029  MAE: 0.806  NMAE: 0.202 |
| MF | num factors = 20  Learn Rate = 0.01  Iteration number = 30  Regularization = 0.015  S.D = 0.1 | RMSE: 0.640  MAE: 0.500  NMAE: 0.125 | RMSE: 1.000  MAE: 0.787  NMAE: 0.197 |
| MF | num factors = 20  Learn Rate = 0.1  Iteration number = 30  Regularization = 0.015  S.D = 0.1 | RMSE: 0.701  MAE: 0.525  NMAE: 0.131 | RMSE: 1.158  MAE: 0.894  NMAE: 0.224 |
| MF | num factors = 20  Learn Rate = 0.05  Iteration number = 30  Regularization = 0.015  S.D = 0.1 | RMSE: 0.585  MAE: 0.441  NMAE: 0.110 | RMSE: 1.109  MAE: 0.861  NMAE: 0.215 |
| MF | num factors = 15  Learn Rate = 0.05  Iteration number = 30  Regularization = 0.05  S.D = 0.1 | RMSE: 0.683  MAE: 0.534  NMAE: 0.133 | RMSE: 1.008  MAE: 0.786  NMAE: 0.197 |
| MF | num factors = 15  Learn Rate = 0.025  Iteration number = 30  Regularization = 0.05  S.D = 0.1 | RMSE: 0.679  MAE: 0.534  NMAE: 0.133 | RMSE: 0.981  MAE: 0.769  NMAE: 0.192 |
| MF | num factors = 15  Learn Rate = 0.025  Iteration number = 50  Regularization = 0.05  S.D = 0.1 | RMSE: 0.664  MAE: 0.519  NMAE: 0.130 | RMSE: 0.992  MAE: 0.774  NMAE: 0.194 |
| MF | num factors = 15  Learn Rate = 0.025  Iteration number = 50  Regularization = 0.075  S.D = 0.1 | RMSE: 0.703  MAE: 0.555  NMAE: 0.139 | RMSE: 0.970  MAE: 0.763  NMAE: 0.191 |
| MF | num factors = 25  Learn Rate = 0.025  Iteration number = 70  Regularization = 0.075  S.D = 0.1 | RMSE: 0.634  MAE: 0.498  NMAE: 0.125 | RMSE: 0.971  MAE: 0.766  NMAE: 0.191 |
| MF | num factors = 35  Learn Rate = 0.025  Iteration number = 90  Regularization = 0.15  S.D = 0.1 | RMSE: 0.821  MAE: 0.658  NMAE: 0.165 | RMSE: 0.954  MAE: 0.764  NMAE: 0.191 |
| MF | num factors = 35  Learn Rate = 0.025  Iteration number = 90  Regularization = 0.15  S.D = 0.1 | RMSE: 0.946  MAE: 0.772  NMAE: 0.193 | RMSE: 0.993  MAE: 0.812  NMAE: 0.203 |
| MF | num factors = 15  Learn Rate = 0.01  Iteration number = 50  Regularization = 0.075  S.D = 0.1 | RMSE: 0.727  MAE: 0.577  NMAE: 0.144 | RMSE: 0.970  MAE: 0.763  NMAE: 0.191 |
| MF | num factors = 15  Learn Rate = 0.03  Iteration number = 50  Regularization = 0.075  S.D = 0.1 | RMSE: 0.702  MAE: 0.553  NMAE: 0.138 | RMSE: 0.975  MAE: 0.767  NMAE: 0.192 |
| MF | num factors = 15  Learn Rate = 0.05  Iteration number = 50  Regularization = 0.075  S.D = 0.1 | RMSE: 0.712  MAE: 0.559  NMAE: 0.140 | RMSE: 0.981  MAE: 0.772  NMAE: 0.193 |

### Summary

From the above table and accuracy values recorded, we could notice that Matrix Factorization method perform better than Global Average and User-item baseline estimates.

|  |  |
| --- | --- |
| **Parameters of model** | **Significance of parameters** |
| Num Factors – Number of features/factors | Default value of 10. Increasing/ Decreasing the iterations increased the accuracy but was over fitting the model sometimes |
| Learn Rate – Learning rate of algorithm | Default value of 0.01. Increasing the value increased the accuracy (decreased the error) but started declining after a limit. A value of 0.025 was found to be optimum |
| Iteration number | Default value of 30. Increasing the value to 90 decreased the difference between training & testing errors (reduced over fitting) |
| Regularization | Default - 0.015 Increasing the value decreased the difference between training & testing errors (reduced over fitting) but the error values increased with increase in reg parameter |

**Error Distribution in Matrix Factorization**

****

**Interpretations**

Error - Absolute difference between actual and predict ratings, is high in the extremes and low in the middle. We could see that error is of the order of 1 or less for actual avg. ratings grouped by **items**, and more for average ratings less than 2 or greater than 4.

## User k-NN & Item k-NN

The following highlights the testing that was done using the User-knn and Item-knn operators.  We tried a variety of values for the number of nearest neighbors (k) for both user and item knn.  A detailed result of testing can be found in the testing performance table in the appendix.

The following are a few examples of performance for Item-knn:

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Parameters | Training Performance | Testing Performance |
| Item k-NN | K=80  Cosine | RMSE: 0.902  MAE: 0.709  NMAE: 0.177 | RMSE: 0.945  MAE: 0.743  NMAE: 0.186 |
| Item k-NN | K=40  Cosine | RMSE: 0.896  MAE: 0.703  NMAE: 0.176 | RMSE: 0.942  MAE: 0.740  NMAE: 0.185 |
| Item k-NN | K=30  Pearson | RMSE: 0.703  MAE: 0.547  NMAE: 0.137 | RMSE: 0.937  MAE: 0.735  NMAE: 0.184 |
| Item k-NN | K=15  Cosine | RMSE: 0.896  MAE: 0.700  NMAE: 0.175 | RMSE: 0.941  MAE: 0.737  NMAE: 0.184 |
| Item k-NN | K=15  Pearson | RMSE: 0.680  MAE: 0.528  NMAE: 0.132 | RMSE: 0.940  MAE: 0.737  NMAE: 0.184 |

You can see that generally values of around 30-40 for K performed well.  For large K=80 both the training and testing performed poorly whereas with small K=15 the model performed better on training but showed signs of over fit with the testing performance being worse.  The Pearson similarity measure generally performed better than Cosine as well.

The following are highlights of performance obtained with User-knn:

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Parameters | Training Performance | Testing Performance |
| User k-NN | K=40  Pearson | RMSE: 0.783  MAE: 0.609  NMAE: 0.152 | RMSE: 0.948  MAE: 0.745  NMAE: 0.186 |
| User k-NN | K=40  Cosine | RMSE: 0.922  MAE: 0.724  NMAE: 0.181 | RMSE: 0.957  MAE: 0.754  NMAE: 0.188 |
| User k-NN | K=30  Pearson | RMSE: 0.774  MAE: 0.602  NMAE: 0.150 | RMSE: 0.949  MAE: 0.746  NMAE: 0.186 |
| User k-NN | K=15  Pearson | RMSE: 0.760  MAE: 0.589  NMAE: 0.147 | RMSE: 0.955  MAE: 0.750  NMAE: 0.187 |

The performance for User-knn followed a similar pattern as Item-knn.  The Pearson similarity measure again seemed to perform better than Cosine and similar numbers of K=30-40 performed well.  Overall the performance obtained with Item-knn was better than User-knn.

Using the best performing model for Item-knn (K=30 with Pearson similarity measure) we then tried to use the Shrinkage parameter, which controls regularization, to see if we could increase performance.

You can see the increase in performance using the parameter in the table below:

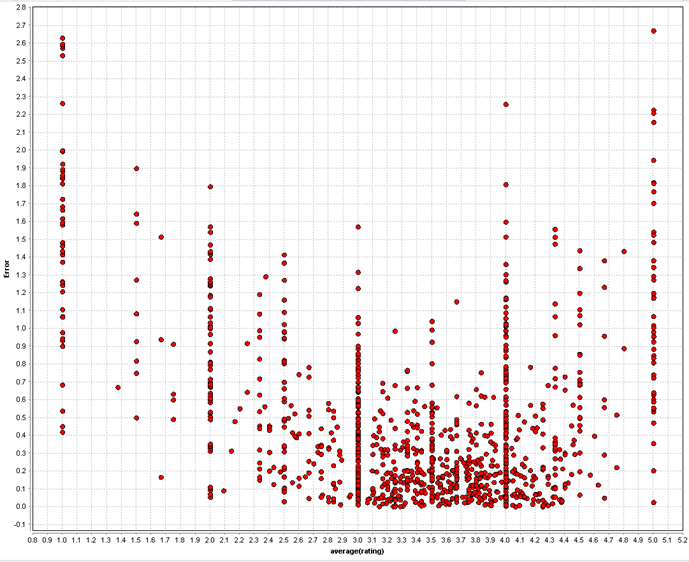
|  |  |  |  |
| --- | --- | --- | --- |
| Model | Parameters | Training Performance | Testing Performance |
| Item k-NN | K=30  Pearson | RMSE: 0.703  MAE: 0.547  NMAE: 0.137 | RMSE: 0.937  MAE: 0.735  NMAE: 0.184 |
| Item k-NN | K=30  Pearson  Shrinkage = 20.0 | RMSE: 0.717  MAE: 0.559  NMAE: 0.140 | RMSE: 0.935  MAE: 0.733  NMAE: 0.183 |
| Item k-NN | K=30  Pearson  Shrinkage = 30.0 | RMSE: 0.725  MAE: 0.566  NMAE: 0.142 | RMSE: 0.934  MAE: 0.732  NMAE: 0.183 |

With Shrinkage parameter = 30.0 we obtained the best performance for Item-knn.

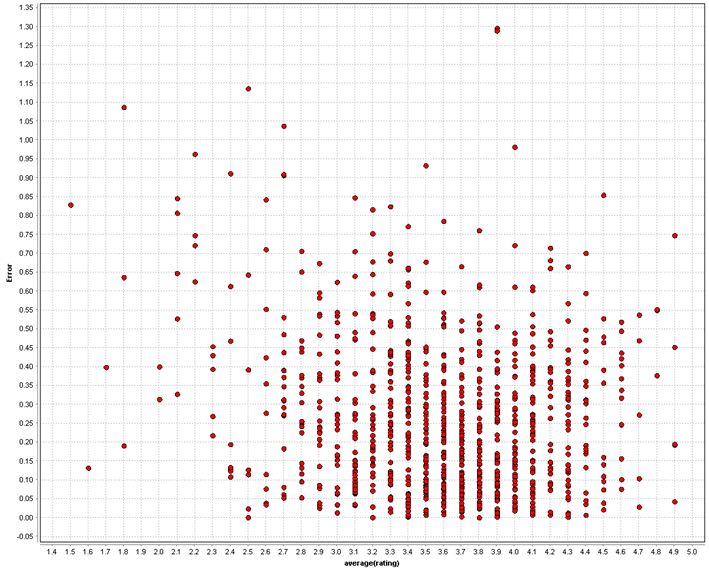
To analyze the error distribution we aggregated the data by user and movie.  This would allow us to generate an attribute which would be the difference of the Average Actual Rating for a User/Movie and the Average Predicted Rating for a User/Movie.

Below are two examples of such graphs for our best Item-knn model (K=30, Pearson, Shrinkage = 30.0):

**Grouped By Item:**



**Grouped by User:**



The first graph (grouped by Item) showed that average ratings for movies have higher error rates at low and high ratings.  This is probably because ratings of 1 and 5 are rarer.  The second graph (grouped by User) shows that overall the error amounts are smaller for grouping by users than movies.  This is most likely because some movies are not rated by a lot of users and therefore the predictions for those movies are harder to generate.  Both graphs show that average ratings are concentrated between 3 and 4.

# Question 3 – Recommender System

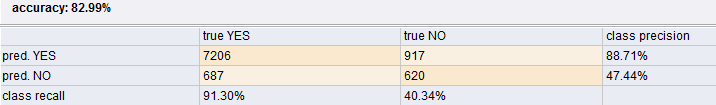
The models that were developed in Question 2 were tuned for accuracy of predicting ratings.  To operationalize the model, the objective is to use the predicted ratings to recommend movies to users.  This being the case, the importance of accuracy predicting higher ratings is more important because those would be the cases where we would recommend a movie to a user.

To accomplish this task, we used RapidMiner to generate a threshold for recommending a movie.  We tried a variety of thresholds across the models in Question 2.  For each model type we used the best performing settings obtained in Question 2. We are not considering Global Average method here because the error values are considerably higher and not suitable for this system.

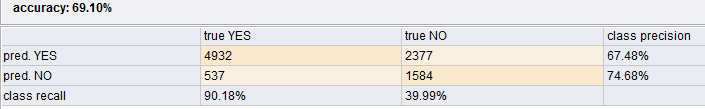
## User-Item Baseline

As stated in question 2, User-Item baseline method with default parameters gave us the best performance for this operator.

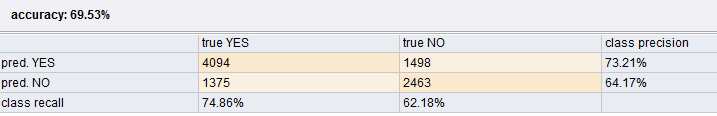
Cut off = 3



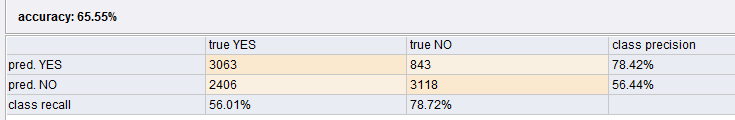
Cut off = 3.2



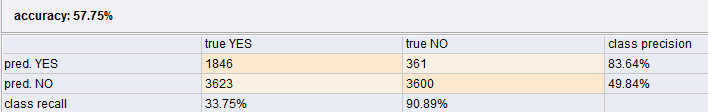
Cut off = 3.5



Cutoff = 3.75



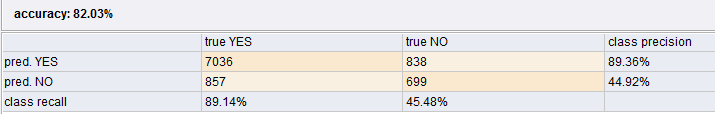
Cutoff = 4

****

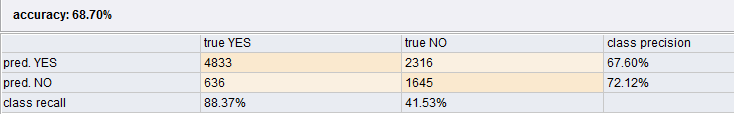
## Matrix Factorization

Set of parameters that gave us the best performing model in Matrix Factorization from Question 2 are Num Factors = 15, Learn rate = 0.025, Iteration number = 50 & Regularization = 0.075. We chose different cut off values to determine the best recommender system.

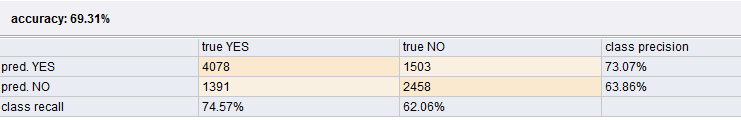
Cutoff = 3



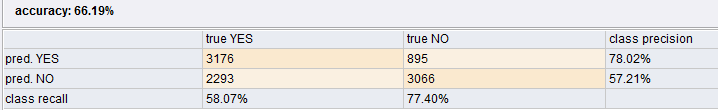
Cutoff = 3.2



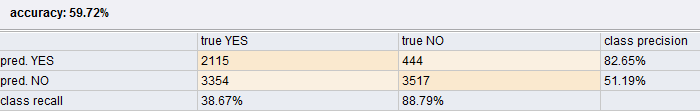
Cutoff = 3.5



Cutoff = 3.7



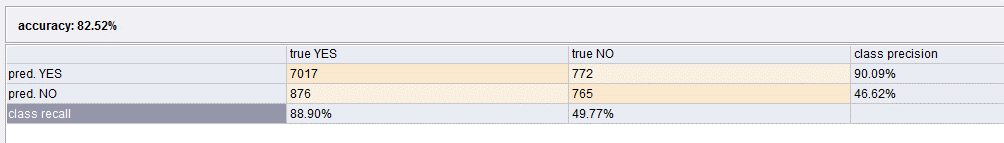
Cutoff = 4.0



## Item-knn

For Item-knn (K=30, Pearson Similarity, Shrinkage = 30.0) the following shows a sample of the confusion matrices for different thresholds:

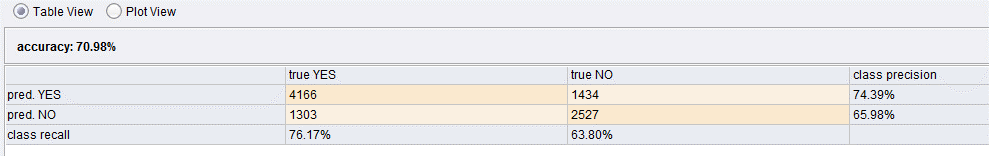
Cutoff = 3.0



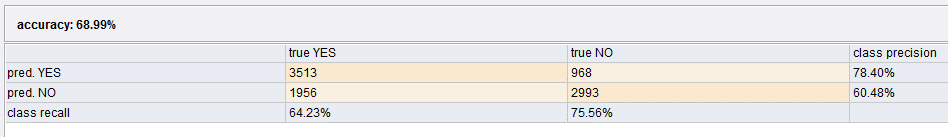
Cutoff = 4.0



Cutoff = 3.5



Cutoff = 3.7



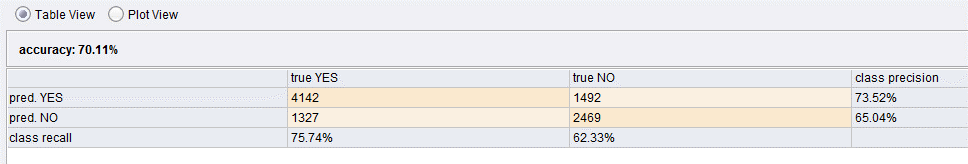
## User-knn

For User-knn (K=40, Pearson Similarity) the following shows a sample of the confusion matrices for different thresholds:

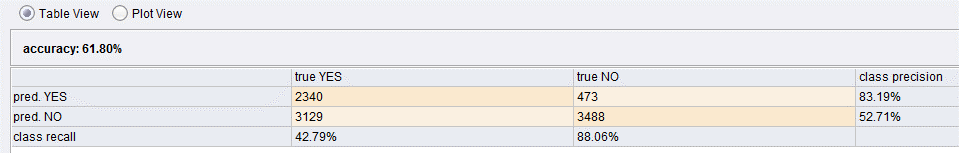
Cutoff = 3.0



Cutoff = 3.5



Cutoff = 4.0



## Summary

Focusing in on the confusion matrices for all models, you can see that the overall the cutoff value of 3 performs the best for both overall accuracy as well as Recommend Yes precision.  The Item-knn performs better than user knn & Matrix factorization, this is expected since the performance found in question 2 indicated that Item-knn had the best accuracy of predicting ratings.  However, we believe that using a cutoff value of 3 does not make enough business sense to recommend movies with such medium ratings.  Therefore, we believe that using the Item-knn model with a cutoff value of 4.0 would be the best option.  The overall accuracy of 62.53 is not great but the precision for Recommend Yes is 84.55 which is very good.  As discussed above the precision measure for Yes is more important since this is really the driver for recommending movies to users.  The Item-knn model provides the best performance for predicting ratings from Question 2 and allows us to accurately recommend movies to users.

# Appendix

Performance Testing

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Parameters | Training Performance | Testing Performance |
| Item k-NN | K=80 Cosine | RMSE: 0.902  MAE: 0.709  NMAE: 0.177 | RMSE: 0.945  MAE: 0.743  NMAE: 0.186 |
| Item k-NN | K=40 Cosine | RMSE: 0.896  MAE: 0.703  NMAE: 0.176 | RMSE: 0.942  MAE: 0.740  NMAE: 0.185 |
| Item k-NN | K=100 Cosine | RMSE: 0.904  MAE: 0.711  NMAE: 0.178 | RMSE: 0.947  MAE: 0.745  NMAE: 0.186 |
| Item k-NN | K=90 Cosine | RMSE: 0.903  MAE: 0.710  NMAE: 0.177 | RMSE: 0.946  MAE: 0.744  NMAE: 0.186 |
| Item k-NN | K=40 Pearson | RMSE: 0.715  MAE: 0.557  NMAE: 0.139 | RMSE: 0.937  MAE: 0.736  NMAE: 0.184 |
| Item k-NN | K=30 Pearson | RMSE: 0.703  MAE: 0.547  NMAE: 0.137 | RMSE: 0.937  MAE: 0.735  NMAE: 0.184 |
| Item k-NN | K=15 Cosine | RMSE: 0.896  MAE: 0.700  NMAE: 0.175 | RMSE: 0.941  MAE: 0.737  NMAE: 0.184 |
| Item k-NN | K=15 Pearson | RMSE: 0.680  MAE: 0.528  NMAE: 0.132 | RMSE: 0.940  MAE: 0.737  NMAE: 0.184 |
| User k-NN | K=40 Pearson | RMSE: 0.783  MAE: 0.609  NMAE: 0.152 | RMSE: 0.948  MAE: 0.745  NMAE: 0.186 |
| User k-NN | K=40 Cosine | RMSE: 0.922  MAE: 0.724  NMAE: 0.181 | RMSE: 0.957  MAE: 0.754  NMAE: 0.188 |
| User k-NN | K=30 Pearson | RMSE: 0.774  MAE: 0.602  NMAE: 0.150 | RMSE: 0.949  MAE: 0.746  NMAE: 0.186 |
| User k-NN | K=15 Pearson | RMSE: 0.760  MAE: 0.589  NMAE: 0.147 | RMSE: 0.955  MAE: 0.750  NMAE: 0.187 |
| Item k-NN | K=30 Pearson reg u = 15.0 reg i = 15.0 | RMSE: 0.711  MAE: 0.556  NMAE: 0.139 | RMSE: 0.939  MAE: 0.738  NMAE: 0.184 |
| Item k-NN | K=30 Pearson reg u = 5 reg i = 1 | RMSE: 0.698  MAE: 0.542  NMAE: 0.135 | RMSE: 0.940  MAE: 0.736  NMAE: 0.184 |
| Item k-NN | K=30 Pearson Shrinkage = 20.0 | RMSE: 0.717  MAE: 0.559  NMAE: 0.140 | RMSE: 0.935  MAE: 0.733  NMAE: 0.183 |
| Item k-NN | K=30 Pearson Shrinkage = 30.0 | RMSE: 0.725  MAE: 0.566  NMAE: 0.142 | RMSE: 0.934  MAE: 0.732  NMAE: 0.183 |
| MF | num factors = 10 Learn Rate = 0.01 Iteration number = 30 Regularization = 0.015 S.D = 0.1 | RMSE: 0.743  MAE: 0.582  NMAE: 0.146 | RMSE: 0.977  MAE: 0.768  NMAE: 0.192 |
| MF | num factors = 30 Learn Rate = 0.01 Iteration number = 30 Regularization = 0.015 S.D = 0.1 | RMSE: 0.554  MAE: 0.432  NMAE: 0.108 | RMSE: 1.029  MAE: 0.806  NMAE: 0.202 |
| MF | num factors = 20 Learn Rate = 0.01 Iteration number = 30 Regularization = 0.015 S.D = 0.1 | RMSE: 0.640  MAE: 0.500  NMAE: 0.125 | RMSE: 1.000  MAE: 0.787  NMAE: 0.197 |
| MF | num factors = 20 Learn Rate = 0.1 Iteration number = 30 Regularization = 0.015 S.D = 0.1 | RMSE: 0.701  MAE: 0.525  NMAE: 0.131 | RMSE: 1.158  MAE: 0.894  NMAE: 0.224 |
| MF | num factors = 20 Learn Rate = 0.05 Iteration number = 30 Regularization = 0.015 S.D = 0.1 | RMSE: 0.585  MAE: 0.441  NMAE: 0.110 | RMSE: 1.109  MAE: 0.861  NMAE: 0.215 |
| MF | num factors = 15 Learn Rate = 0.05 Iteration number = 30 Regularization = 0.05 S.D = 0.1 | RMSE: 0.683  MAE: 0.534  NMAE: 0.133 | RMSE: 1.008  MAE: 0.786  NMAE: 0.197 |
| MF | num factors = 15 Learn Rate = 0.025 Iteration number = 30 Regularization = 0.05 S.D = 0.1 | RMSE: 0.679  MAE: 0.534  NMAE: 0.133 | RMSE: 0.981  MAE: 0.769  NMAE: 0.192 |
| MF | num factors = 15 Learn Rate = 0.025 Iteration number = 50 Regularization = 0.05 S.D = 0.1 | RMSE: 0.664  MAE: 0.519  NMAE: 0.130 | RMSE: 0.992 MAE: 0.774 NMAE: 0.194 |
| MF | num factors = 15 Learn Rate = 0.025 Iteration number = 50 Regularization = 0.075 S.D = 0.1 | RMSE: 0.703  MAE: 0.555  NMAE: 0.139 | RMSE: 0.970  MAE: 0.763  NMAE: 0.191 |
| MF | num factors = 25 Learn Rate = 0.025 Iteration number = 70 Regularization = 0.075 S.D = 0.1 | RMSE: 0.634  MAE: 0.498  NMAE: 0.125 | RMSE: 0.971  MAE: 0.766  NMAE: 0.191 |
| MF | num factors = 35 Learn Rate = 0.025 Iteration number = 90 Regularization = 0.15 S.D = 0.1 | RMSE: 0.821  MAE: 0.658  NMAE: 0.165 | RMSE: 0.954  MAE: 0.764  NMAE: 0.191 |
| MF | num factors = 35 Learn Rate = 0.025 Iteration number = 90 Regularization = 0.15 S.D = 0.1 | RMSE: 0.946  MAE: 0.772  NMAE: 0.193 | RMSE: 0.993  MAE: 0.812  NMAE: 0.203 |
| MF | num factors = 15 Learn Rate = 0.01 Iteration number = 50 Regularization = 0.075 S.D = 0.1 | RMSE: 0.727  MAE: 0.577  NMAE: 0.144 | RMSE: 0.970  MAE: 0.763  NMAE: 0.191 |
| MF | num factors = 15 Learn Rate = 0.03 Iteration number = 50 Regularization = 0.075 S.D = 0.1 | RMSE: 0.702  MAE: 0.553  NMAE: 0.138 | RMSE: 0.975  MAE: 0.767  NMAE: 0.192 |
| MF | num factors = 15 Learn Rate = 0.05 Iteration number = 50 Regularization = 0.075 S.D = 0.1 | RMSE: 0.712  MAE: 0.559  NMAE: 0.140 | RMSE: 0.981  MAE: 0.772  NMAE: 0.193 |
| User-Baseline | num iter = 30 reg u = 15 reg i = 10 | RMSE: 0.920  MAE: 0.729  NMAE: 0.182 | RMSE: 0.963  MAE: 0.765  NMAE: 0.191 |
| User-Baseline | num iter = 30 reg u = 30 reg i = 30 | RMSE: 0.936  MAE: 0.745  NMAE: 0.186 | RMSE: 0.976  MAE: 0.778  NMAE: 0.195 |
| User-Baseline | num iter = 1000 reg u = 15 reg i = 10 | RMSE: 0.920  MAE: 0.729  NMAE: 0.182 | RMSE: 0.963  MAE: 0.765  NMAE: 0.191 |
| Global Average | No parameters | RMSE: 1.126  MAE: 0.945  NMAE: 0.236 | RMSE: 1.122  MAE: 0.945  NMAE: 0.236 |