IDS 572 Assignment 4

Aditi Vishwasrao – 674849041

Mounica Sirineni - 652518967

Nishanth Reddy Konkala - 658824939

1. Explore the data to obtain an understanding of users, movies and how users have rated movies.

a) What is the overall distribution of ratings?

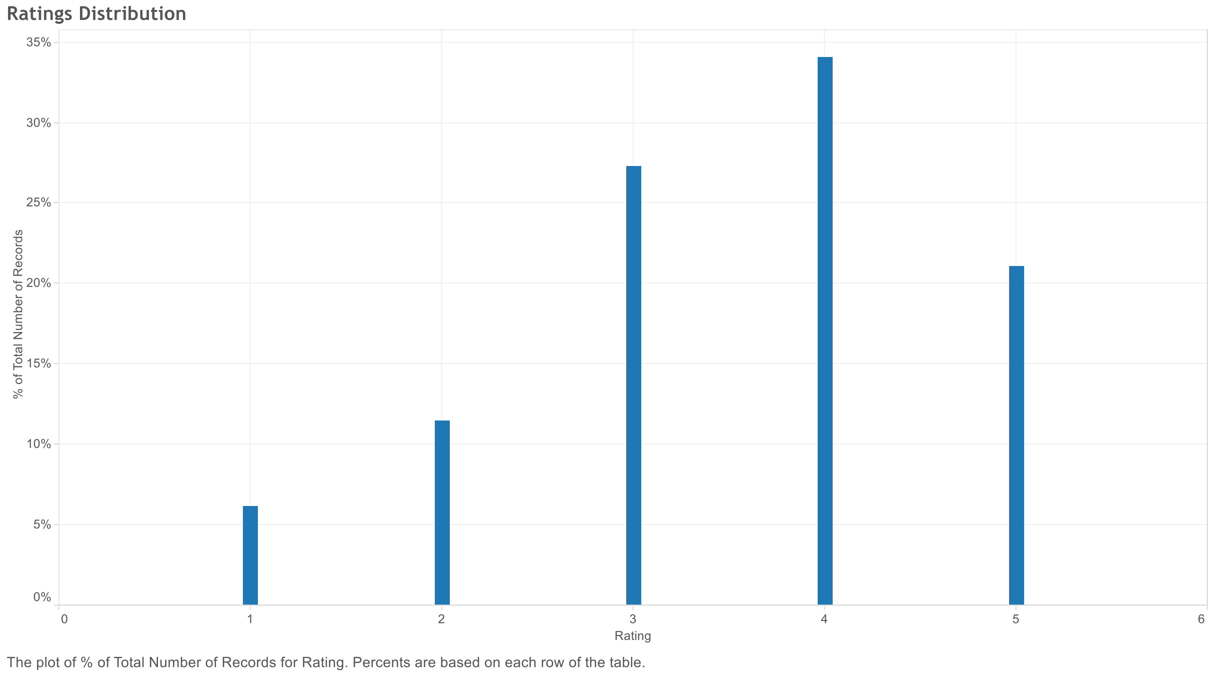


Figure 1.1 Overall Distribution of ratings

Movies have been rated from values 1 to 5 by users where 1 stands for a low rating and 5 stands for a high rating.

From the above figure 1.1, we can see that the distribution of ratings is slightly skewed to the left. This means that most movies have been rated high by users. Nearly 35% of the movies have a rating 4 and more than 25% have a rating of 3.

The average movie rating is 3.524.

b) On average, how do users rate movies; what ratings do movies have on average?

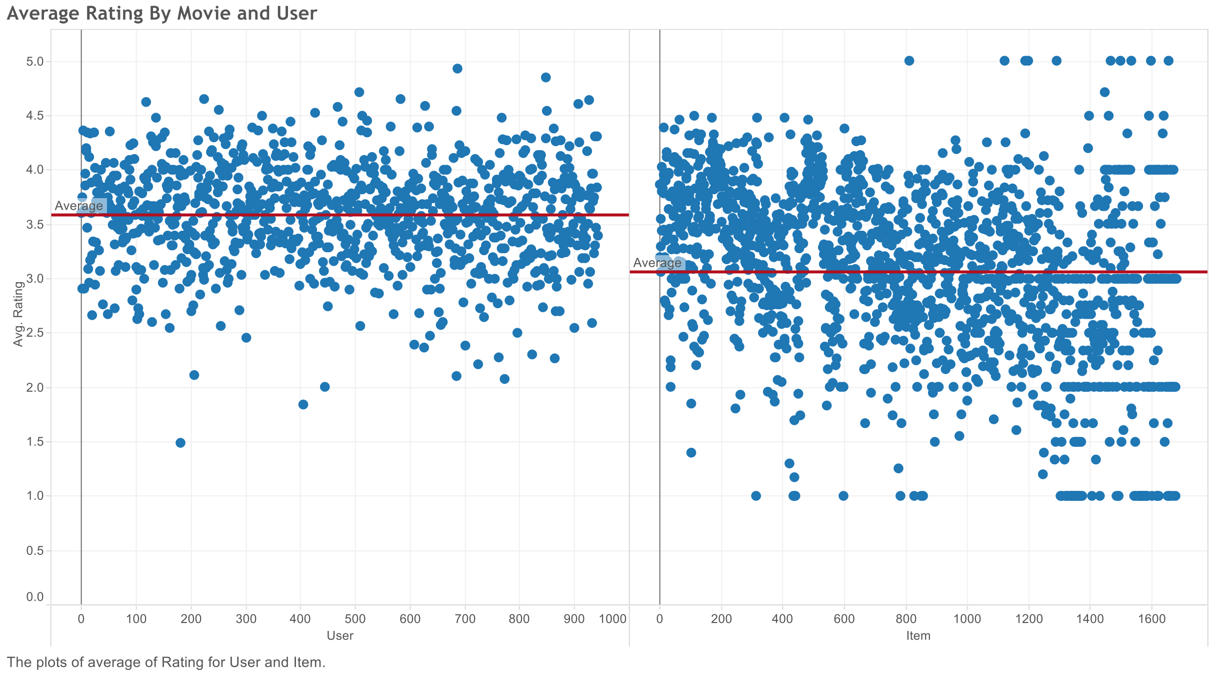
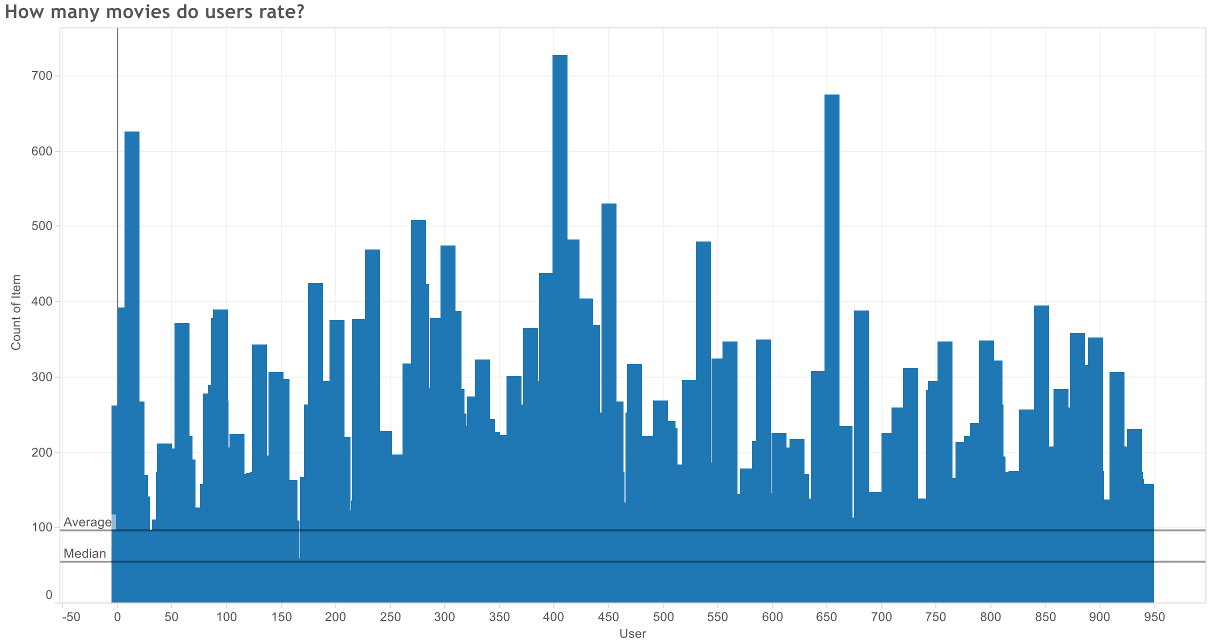


Figure 1.2 Average movie ratings and user ratings

We can see from the above plot that a user has given an average rating of 3.6 to movies. However, a movie has received an average rating of only about 3.1. Most movies have lower average ratings. However, there is no particular trend visible in the average ratings given by users.

c) How many movies do users rate, and how many ratings do movies get?

How many movies do users rate?

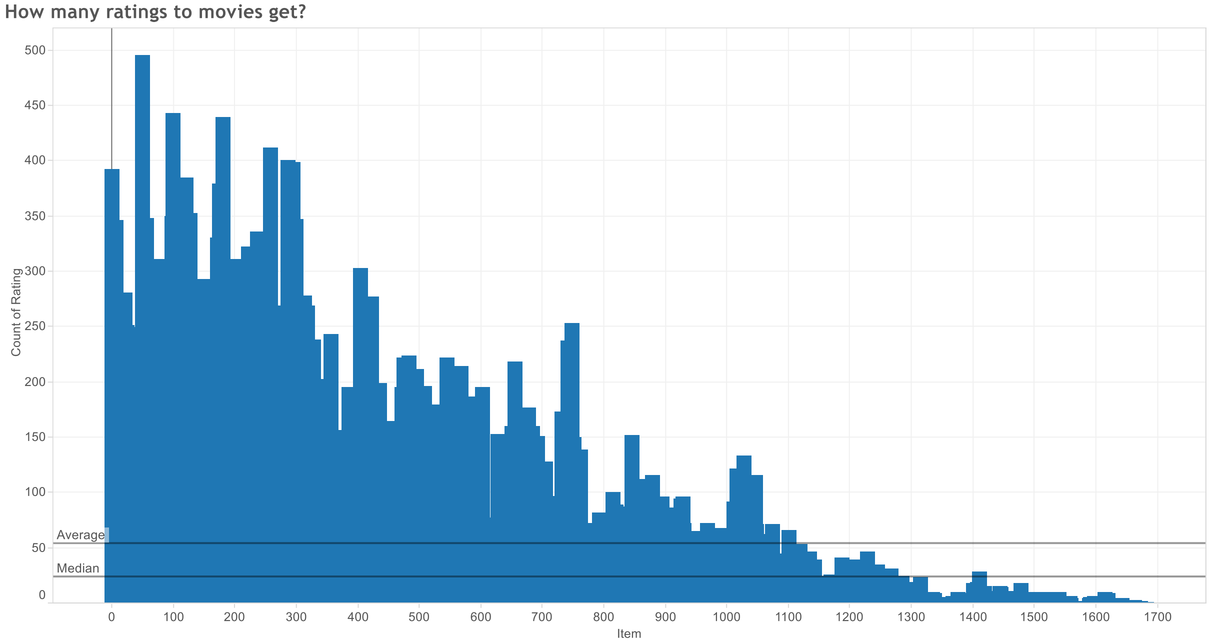


The plot of count of Item for User.

Figure 1.3 Number of movies rated by users

On an average users rate about 95 movies. The median of this distribution is 55. This means that most users rate a lower number of movies compared to the average. The range of the number of movies that users rate, however, is very large. The minimum number of movies rated by a user is 10 and the maximum is 727.

How many ratings to movies get?



The plot of count of Rating for Item.

Figure 1.4 Number of ratings for movies

The distribution of the number of ratings that movies get is highly right skewed. This means that most movies get a lower count of ratings. A movie get around 50 ratings on an average. However, the median of the distribution is much less at approximately 25.

The range of the number of ratings a movie gets ranges from 1 to 495.

d) How are rating levels distributed, do many people have high/low ratings?

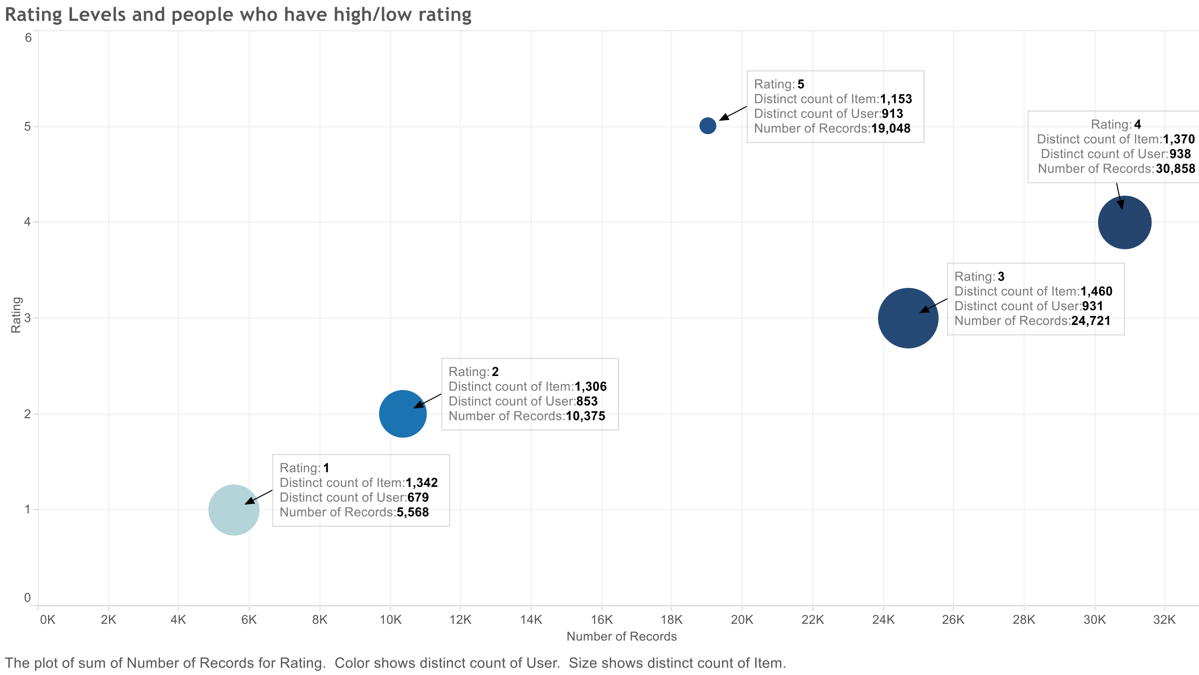


Figure 1.5 Distribution of rating level and trend in ratings

From the above graph we can infer the following:

- Less number of users have chosen to give lower rating to a movie.
- Most users have chosen to give an average or high rating to a movie.
- Most movies have received a rating of 3. A slightly lower number of movies have a rating of 2 and 4.
- The number of movies being rated low is much higher than the number of movies being rated high.
- Also, as is evident from the above figure and also from figure 1.1, the number of low ratings is lesser than high ratings.

2. Consider collaborative filtering based rating prediction. We will evaluate performance of different approaches for predicting ratings.

What measures will you use for assessing performance (why)?

The performance of the model can be evaluated based on the following 3 parameters:

1) RMSE- Root Mean Square error
2) MAE-Mean Absolut Error
3) NMAE-Normalized Mean Absolut error.

We chose the above predictive accuracy metrics instead of classification accuracy metrics because we want to recommend movies to the users with least error. At different levels of ratings, we examine the error to assess the performance of the model.

And what relationships will you examine -- for example, error (or accuracy) at different levels of ratings; are errors distributed equally across movies, users? etc.

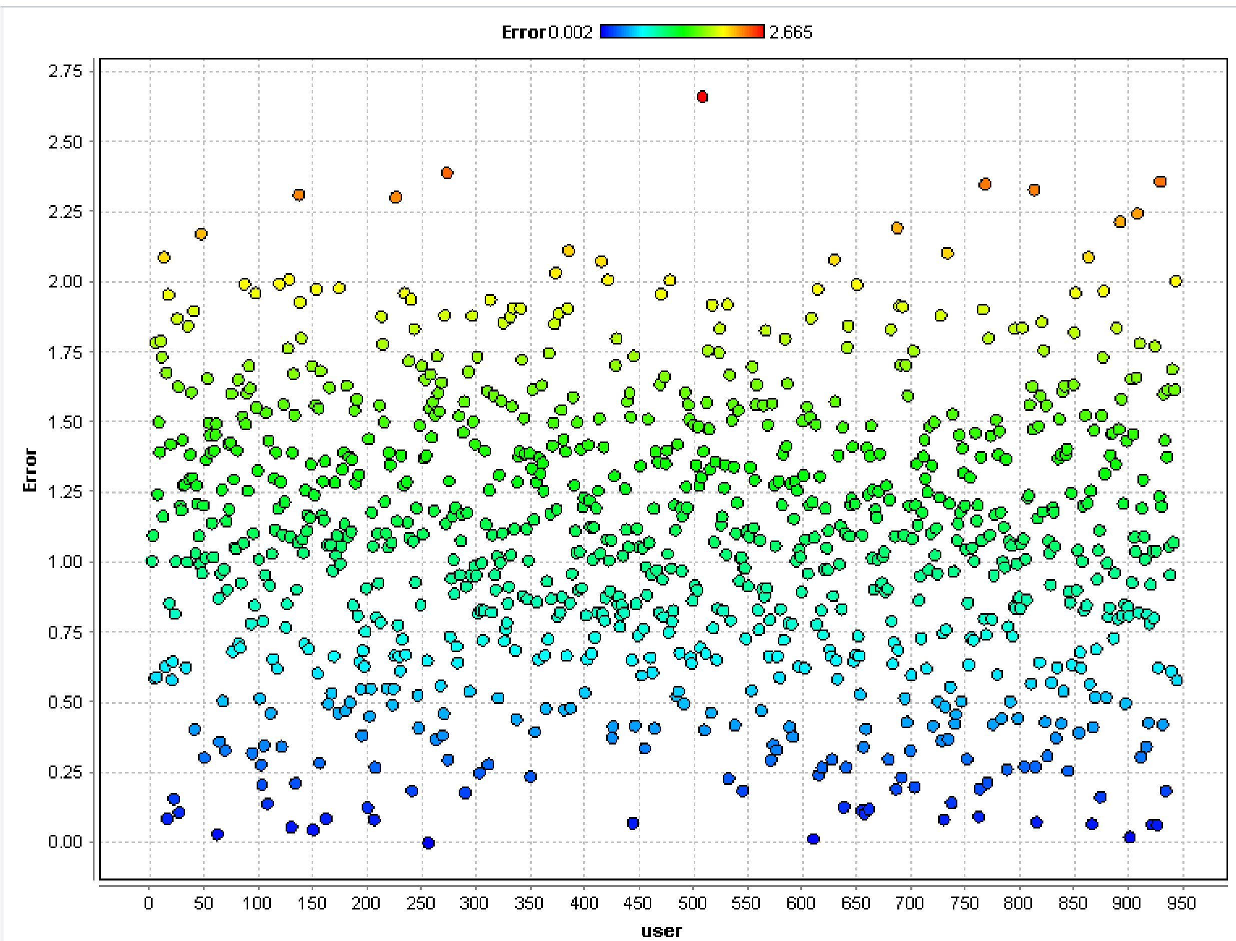


Figure 2.1 Error distribution among users

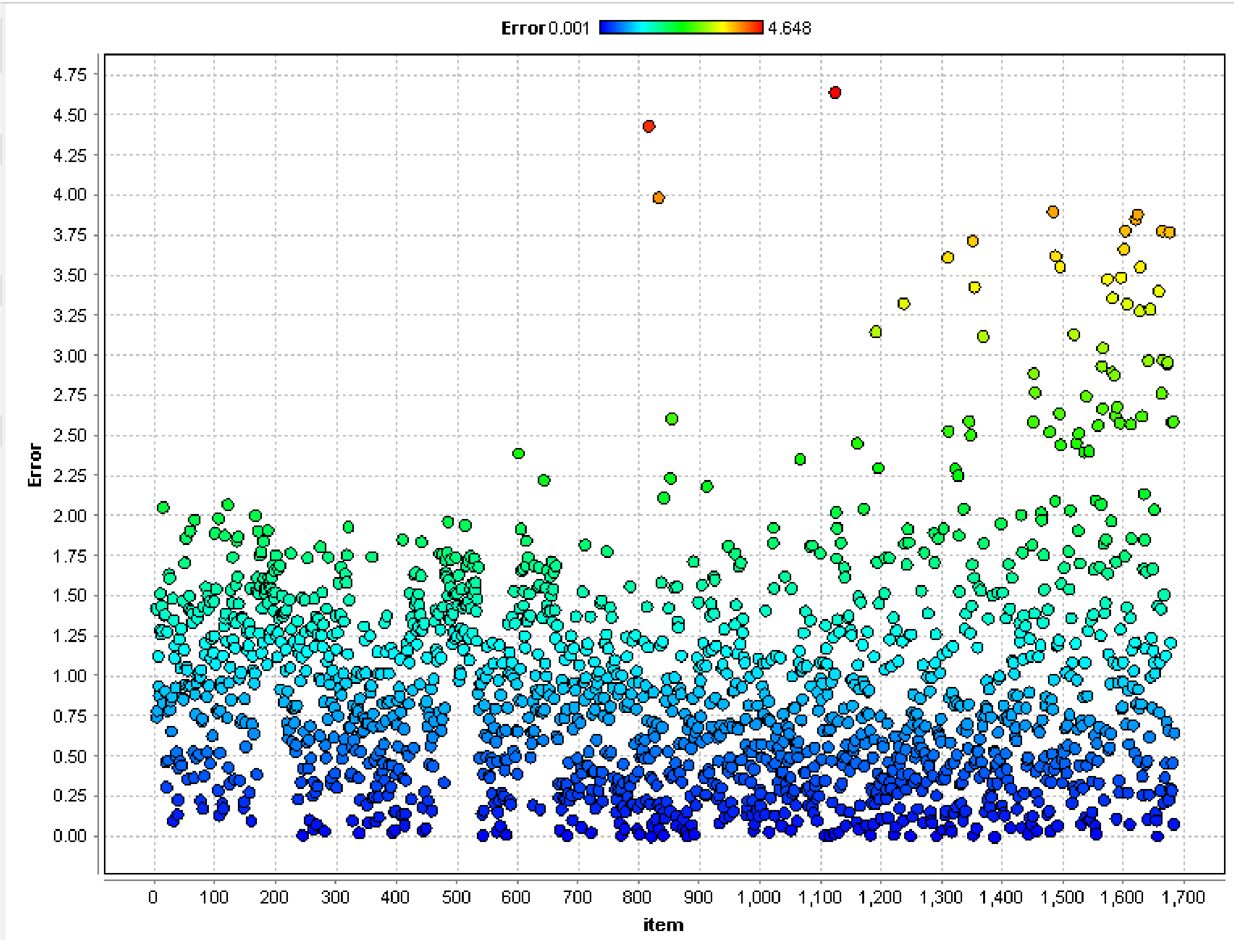


Figure 2.2 Error distribution among movies

From the above figures, we observe that there is no pattern distribution of errors among users and movies.

(a) Use the Global Average method and User-Item Baseline methods. Do you find any performance differences? Do parameter changes for the user-item baseline operator make any difference?

Global Average method

Global average method uses the average rating value over all ratings for prediction. We used minimum rating as “1” and range as “5”. The errors for this method are:

Table 2.1 Global Average method performance

Data set	RMSE	MAE	NMAE
Training data	1.126	0.945	0.236
Testing data	1.122	0.945	0.236

Click here for global average method performance vector

User-Item Baseline method

User-Item Baseline method uses the average rating values, plus a regularized user and item bias for prediction.

Parameters:

Minimum rating is 1 and Range is 5. Number of iterations did not have any impact on the performance vector. We got same values for different values of number of iterations and a fixed regularization parameter. So we used number of iterations as “10” and changed regularization parameters to know the effect and the results are tabulated below.

Table 2.2 User-Item Baseline method performance

Regularization parameter	Data set	RMSE	MAE	NMAE
Reg u = 15.0 Reg I = 10.0	Training data	0.920	0.729	0.182
	Testing data	0.963	0.765	0.191
Reg u = 20.0 Reg I = 15.0	Training data	0.925	0.734	0.183
	Testing data	0.967	0.769	0.192
Reg u = 25.0 Reg I = 20.0	Training data	0.930	0.738	0.185
	Testing data	0.971	0.773	0.193
Reg u = 30.0 Reg I = 25.0	Training data	0.934	0.743	0.186
	Testing data	0.974	0.777	0.194

From the table, we observe that errors increase with increase in regularization parameters for user (reg u) and item (reg i). We get best method at reg u = 15.0 and reg I = 10.0. Click here for User-Item Baseline method performance vector.

Conclusion: When we compare errors in global average method and user-item baseline methods, we observe that errors are less in user-item baseline method. The regularization parameter in user-item baseline method helps to reduce overfit. Also, it supports several iterations of alternating optimization, instead of just one. Therefore, best method is **User-Item Baseline method**.

(b) Use the Matrix factorization operator. Explore performance with varying number of factors. Does learning rate make a difference to performance?

Matrix factorization operator factorizes the observed rating values using a factor matrix for users and one for items.

Parameters:

Min rating = 1

Range = 4

Table 2.3 Matrix factorization performance

Parameters	Data set	RMSE	MAE	NMAE
Num Factors = 10	Training data	0.744	0.584	0.146
Learn rate = 0.01	Testing data	0.976	0.764	0.191
Num Factors = 25	Training data	0.596	0.465	0.116
Learn rate = 0.01	Testing data	1.021	0.799	0.200
Num Factors = 50	Training data	0.436	0.338	0.084
Learn rate = 0.01	Testing data	1.030	0.811	0.203
Num Factors = 10	Training data	1.106	0.925	0.231
Learn rate = 0.1	Testing data	1.120	0.941	0.235
Num Factors = 10	Training data	0.795	0.613	0.153
Learn rate = 0.001	Testing data	1.099	0.849	0.212

Conclusion: From the table, we observe that as the number of factors increase, errors in training data decrease but increase in testing data. In such a case, when number of factors is "10", the difference between training and testing data errors is minimum. As we increase learning rate, we observe that training and testing data errors decrease and then increase. As a trade-off, the best parameters are number of factors = 10 and learning rate = 0.01. Click here for matrix factorization performance vector.

(c) Use the User-knn and Item-knn operators. Explore performance with varying the number of nearest neighbors k? Also do you notice any differences between using the cosine similarity measure and the Pearson measure? Are the neighborhood sizes, k, that give good performance, comparable across the two operators (why?)?

User-knn

Table 2.4 User-KNN performance

		K	Training	Testing
		60	0.922	0.957
RMSE	Cosine	70	0.923	0.957
		80	0.923	0.957
		90	0.923	0.957
RMSE	Pearson	60	0.795	0.949
		70	0.800	0.949
		80	0.804	0.949
		90	0.807	0.949
MAE	Cosine	60	0.724	0.754
		70	0.725	0.754
		80	0.725	0.754
		90	0.726	0.755

MAE	Pearson	60	0.619	0.746
		70	0.623	0.746
		80	0.627	0.746
		90	0.629	0.746
NMAE	Cosine	60	0.181	0.189
		70	0.181	0.189
		80	0.181	0.189
		90	0.181	0.189
NMAE	Pearson	60	0.155	0.186
		70	0.156	0.186
		80	0.157	0.186
		90	0.157	0.187

Conclusion: In most of the cases, there was no change in error when the 'K' value changed. In the cases where there is change in the error, it is observed that the error value is least for small value of K. The error value is less when Pearson coefficient is used. The neighborhood size, which is giving the best results, doesn't give the same error in both the operators. Therefore, the best model in our case is with **Pearson coefficient, K=60.** Click here to view the performance vector User KNN.

Item-knn

Table 2.5 Item-KNN performance

		K	**Training**	**Testing**
		60	0.899	0.944
RMSE	Cosine	70	0.901	0.945
		80	0.902	0.945
		90	0.903	0.946
RMSE	Pearson	60	0.733	0.938
		70	0.740	0.938
		80	0.745	0.938
		90	0.750	0.938
MAE	Cosine	60	0.706	0.742
		70	0.707	0.743
		80	0.709	0.743
		90	0.710	0.744
MAE	Pearson	60	0.572	0.736
		70	0.578	0.736
		80	0.583	0.736
		90	0.587	0.737

NMAE	Cosine	60	0.177	0.186
		70	0.177	0.186
		80	0.177	0.186
		90	0.177	0.186
NMAE	Pearson	60	0.143	0.184
		70	0.145	0.184
		80	0.146	0.184
		90	0.147	0.184

Conclusion: In most of the cases, there was no change in error when the 'K' value changed. In the cases where there is change in the error, it is observed that the error value is least for small value of K. The error value is less when Pearson coefficient is used. The neighborhood size, which is giving the best results, doesn't give the same error in both the operators. For the same values of 'K', the errors in USER KNN operator are slightly higher than the item KNN. Therefore, the best model in our case is **Item –KNN, Pearson coefficient, K=60.** Click here to view the performance vector for Item KNN

Comparing performance across the different operators, which would you prefer to use (why)?

When we compare errors across all operators used in question2, Item KNN has least. So, the best method is Item KNN.

3. Consider the decision support objective of recommending movies to users. Movies predicted to receive high ratings will be recommended for a user. We then need to determine a cutoff rating for 'high' (for example, any rating >=4 is 'high'). To access performance for this, we can consider a confusion matrix and related measures like precision, sensitivity etc (or, how many predicted highs correspond to actual high, etc.). Using the predicted ratings for the test data, determine such decision support performance using the operators in Question 2. Comparing performance across the different operators, which would you prefer to use (why)? What value of 'cutoff' will you use?

Table 3.1 Comparison of different cutoff values

Cutoff	Overall Accuracy	Class Recall (true Y)	Class Recall (true N)
3.5	70.67%	75.63%	63.82%
3.7	68.77%	63.58%	75.94%
4.0	61.87%	41.85%	89.50%

From the above table, we observe that we get maximum accuracy and recall for cut-off = 3.5.

Table 3.2 Comparison of different methods with cut-off = 3.5

Method	Overall Accuracy	Class Recall (true Y)	Class Recall (true N)
Global Average	58%	100%	0%

User-Item Baseline	69.53%	74.86%	62.18%
Matrix Factorization	68.44%	73.72%	61.15%
User KNN	70.18%	75.88%	62.31%
Item KNN	70.67%	75.63%	63.82%

From the above table, we observe that among all methods, we get maximum accuracy and recall for Item KNN. Click here for the performance vector of Item KNN.

Therefore, when we consider errors in question 2 and accuracy in question 3, we observe that **Item KNN with Pearson coefficient and K = 60** is the best method.

Are errors distributed equally across movies and across users?

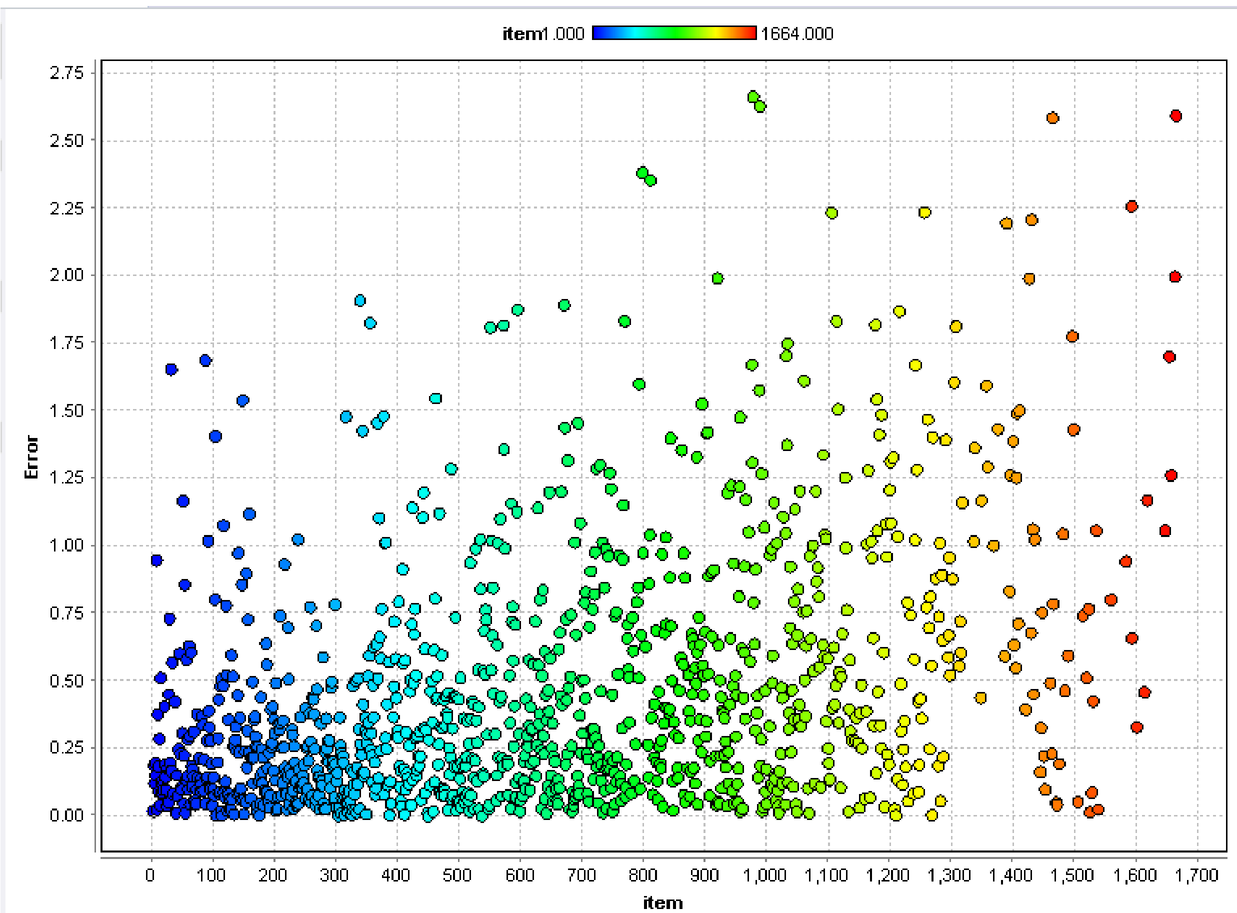


Figure 3.1 Error distribution among movies

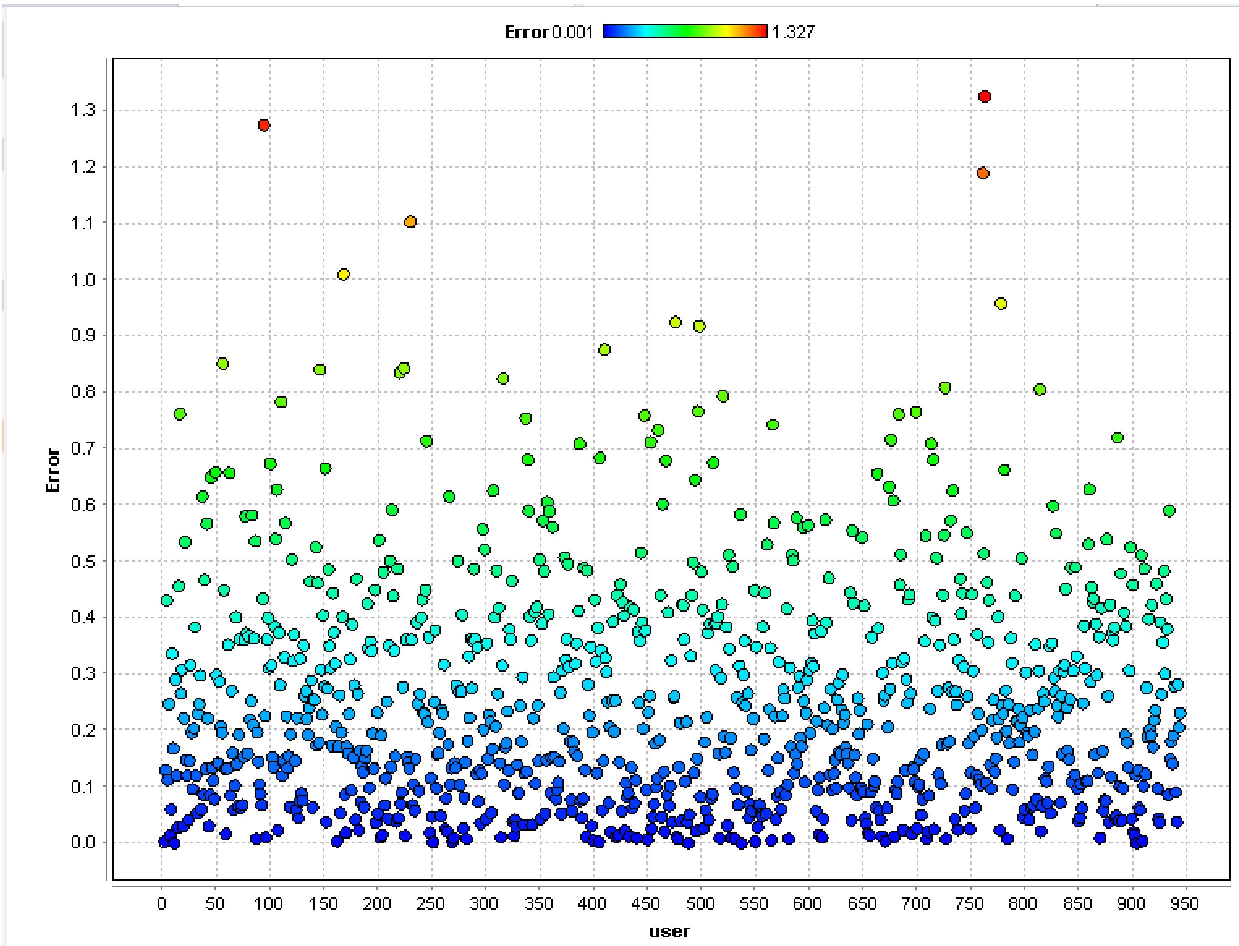


Figure 3.2 Error distribution among users

From figure 3.1, we observe that maximum movies have error in the range of 0.0 and 0.5. From figure 3.2, maximum users have error in the range of 0.0 and 0.3.

4. Appendix

4.1 Global Average method

[Back to the top]

4.1.1 Training data performance vector

PerformanceVector

```
PerformanceVector:
RMSE: 1.126
MAE: 0.945
NMAE: 0.236
```

4.1.2 Testing data performance vector

PerformanceVector

```
PerformanceVector:
RMSE: 1.122
MAE: 0.945
NMAE: 0.236
```

4.2 User-Item Baseline method

[Back to the top]

4.2.1 Training data performance vector

PerformanceVector

```
PerformanceVector:
RMSE: 0.920
MAE: 0.729
NMAE: 0.182
```

4.2.2 Testing data performance vector

PerformanceVector

```
PerformanceVector:
RMSE: 0.963
MAE: 0.765
NMAE: 0.191
```

4.3 Matrix Factorization

[Back to the top]

4.3.1 Training data performance vector

PerformanceVector

```
PerformanceVector:
RMSE: 0.744
MAE: 0.584
NMAE: 0.146
```

4.3.2 Testing data performance vector

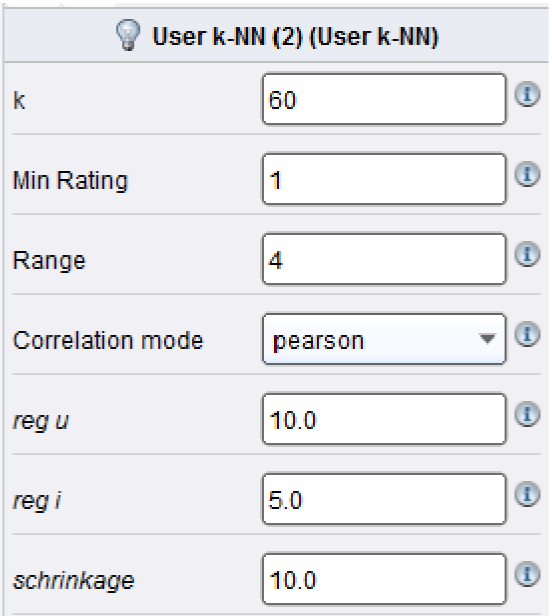
PerformanceVector

```
PerformanceVector:
RMSE: 0.976
MAE: 0.764
NMAE: 0.191
```

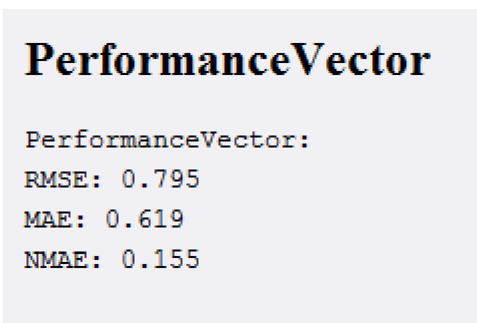
4.4 User K-NN

[Back to the top]

4.4.1 Parameters



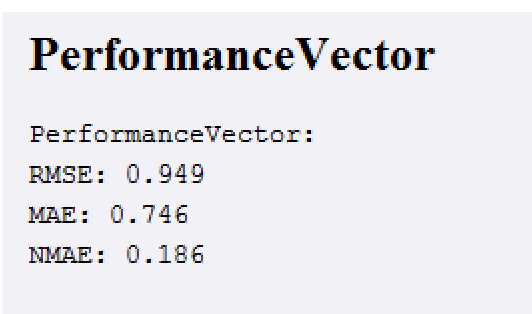
4.4.2 Training data performance vector



PerformanceVector

PerformanceVector:
RMSE: 0.795
MAE: 0.619
NMAE: 0.155

4.4.3 Testing data performance vector



PerformanceVector

PerformanceVector:
RMSE: 0.949
MAE: 0.746
NMAE: 0.186

4.5 Item K-NN

[Back to the top]

4.5.1 Parameters

Item k-NN

k	60
Min Rating	1
Range	4
reg u	10.0
reg i	5.0
schrinkage	10.0
Correlation mode	pearson

4.5.2 Training data performance vector

ExampleSet (1 example, 0 special attributes, 3 regular attributes)

Row No.	RMSE	MAE	NMAE
1	0.733	0.572	0.143

4.5.3 Testing data performance vector

ExampleSet (1 example, 0 special attributes, 3 regular attributes)

Row No.	RMSE	MAE	NMAE
1	0.938	0.736	0.184

4.6 Item KNN performance vector

[Back to the top]

Table View   Plot View

accuracy: 70.67%

	true Y	true N	class precision
pred. Y	4136	1433	74.27%
pred. N	1333	2528	65.48%
class recall	75.63%	63.82%	