Jitendra Patel
Renu Singh
Ajinkya Rane

ASSIGNMENT#4 COLLABORATIVE FILTERING

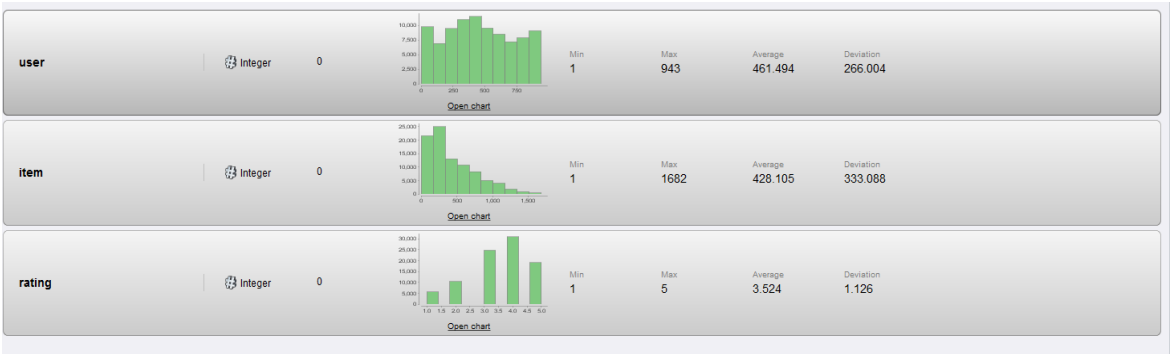
Collaborative Filtering is perhaps the most successful and popular method for providing predictions over user preferences, or recommending items. Intuitively, a recommendation system builds up a user profile based on past records, and compares it with some reference characteristics, and based on that predict the 'rating' that a user would give to an item they had not yet evaluated. The motivation behind the project is to gain a deeper understanding of how a recommender system can be applied.

Question 1

Data Exploration:-

The Given data file shows:-

	Items	Mean	SD
No. of movies that has been rated	1682	428.105	338.088
No of Users who rated the movies	943	461.494	266.044
Ratings Given by users	1 to 5	3.524	1.126



Distributions of Movies, Ratings & Users with SD and Means

A) What is the overall distribution of ratings?

The average rating across all users is 3.524 with the standard deviation of 1.126.

Rating 1- 5500 Approx. (6.2%)
Rating 2- 10300 Approx. (11.6%)
Rating 3-23700 Approx. (26.8%)
Rating 4- 29900 Approx. (33.8%)
Rating 5-19000 Approx. (21.5%)

The Distribution shows that most of the movies are rated in the range of 3, 4 & 5. Highest number of movies are rated with Rating 4.

B) On average, how do users rate movies; what ratings do movies have on average? (You may want to plot the distribution of average ratings for users, movie. Can you show this on a single plot?)

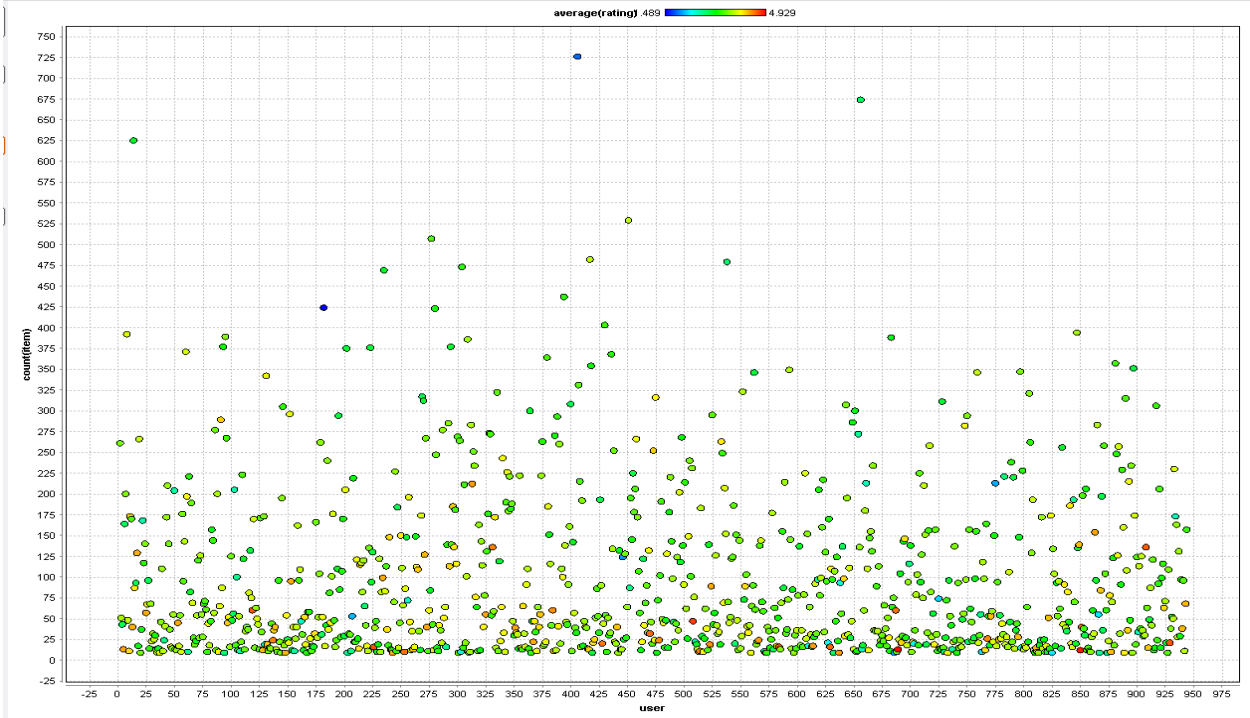


Fig 1.1 Relationship between users, items and ratings.

As it is clear that users have rated most of the movies (80%) as 3, 4 or 5. And from answers above we know that average rating of the whole data is 3.52.

From the above figure we can see that the most of the users have rated approximately 50 items (movies) with average ratings ranging from 3 to 5 (i.e. density of Yellow, Green, Orange & red dots on the lower side of the graph.)

Most of the red & orange dots (i.e. average rating above 3.5) lies on the lower part of the graph. This can be because of users avoid rating the bad movies or movies they didn't like.

Users with the movie count more than 50 has the average rating of 3.5 (i.e. green dots). These group of users can be considered as an active group and the average ratings of 3.5 shows that these people are positive raters. Their ratings will be 3 or 4 most of the times.

C) - how many movies do users rate, and how many ratings do movies get? (Consider the distribution of rating counts)

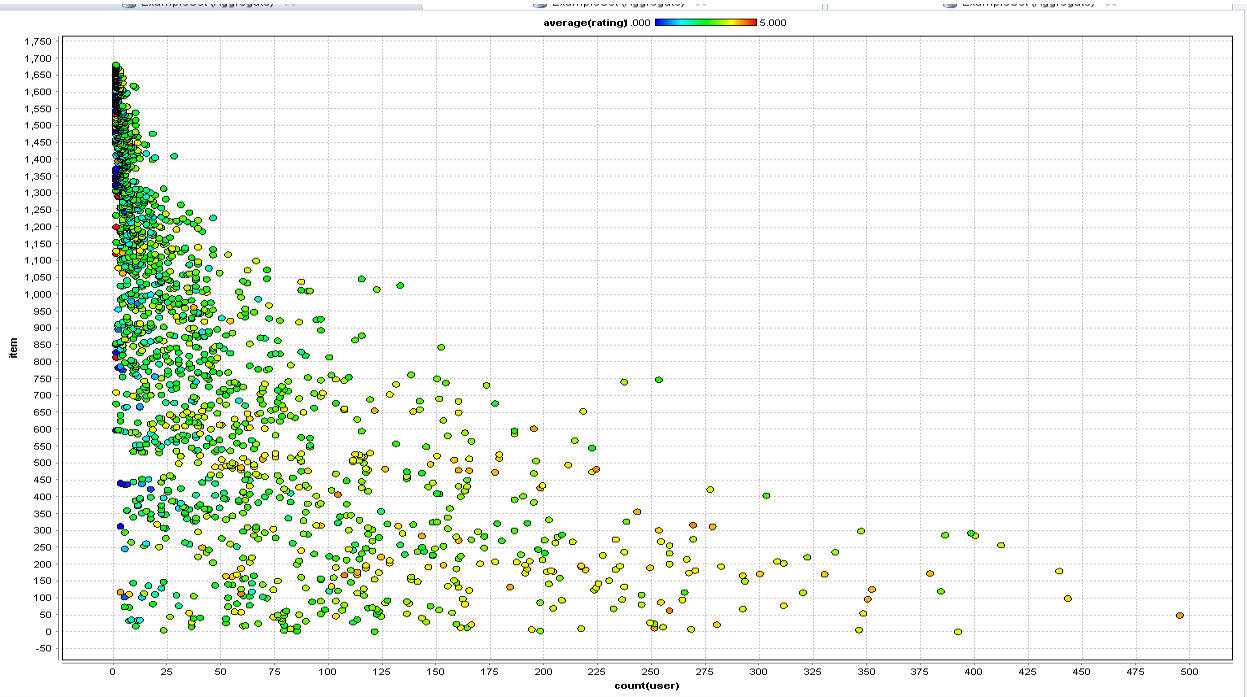


Fig 1.2 Ratings counts of Movies

From data exploration we get to know that a particular user rates approximately 428 movies (average).But Fig 1.1 shows that most of the users rates 50 movies or so. This difference indicates that the frequency of the user counts are not normally distributed.

Refer APPENDIX to explore the following data:-
Users who have rated the movies 0-50= 430
Users who have rated the movies 50-100= 140
Users who have rated the movies 100-150= 105
Users who have rated the movies 150-200= 80
Users who have rated the movies 200 & above= 188

Figure 1.2 shows that a particular movie with the user counts less than 50 have the average ratings (i.e. green dots). And the density of those green dots indicates the major chunk of the users have rated movies between 0-50. Also, the more the number of users who rated the movie Higher is the ratings for the particular movie. We can observe that with fig 1.2 showing that the dots on right side of the graphs are either yellow, orange or Red. (i.e. Higher Ratings.)

One more thing to be observed is as we go up in the graph it converges. Indicating that movies numbered 800 or more are rated by less than 75 users.
The graph also indicates that the movies lower down the graph (i.e. 0-800) are doing well overall. The density of yellow, orange, green & red supports the claim.

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

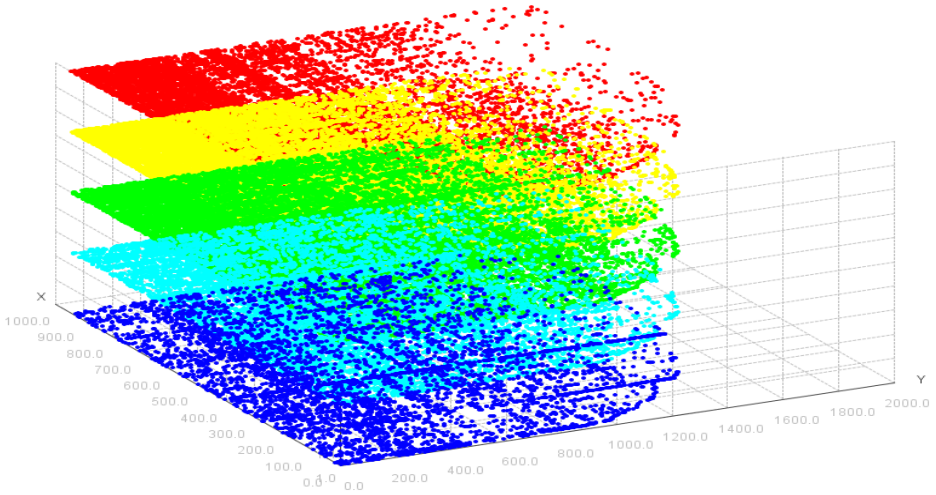


Fig 1.3 3D scattered plot for ratings vs user vs items.

Not many people have low (1, 2) ratings and high ratings (5). Most of the people have average ratings (3, 4). The Figure 1.3 shows the distribution of different rating levels.

Figure 1.3 shows that most of the people have rated the movies on a higher side. The ratings level is dense towards the lower part which indicates the frequently rated movies. The movies on the upper part are being rated by less number of users.

Question 2

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

1) To evaluate each method, we will calculate the error rate using

- **Mean Absolute Error Rate (MAE)**
- **Normalized Mean Absolute Error rate (NMAE)**
  We will be experimenting with data sets having different numbers of rating values we adopt a normalized mean absolute error, which enables comparison across data sets.
- **Root Mean Squared Error (RMSE).**
  Our goal will be to minimize the RMSE (root mean square error) when predicting the ratings on a test set. The quality of results is generally measured by RMSE, RMSE puts more emphasis on large errors compared with the alternative of Mean Absolute Error.

2) We can use Accuracy as a performance measure as well. Preparing Confusion matrix as a performance measure. We will have to provide a proper cut-off rating value and assess number of predicted vs original 0's & 1's.

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

Accuracy & Error rate:-

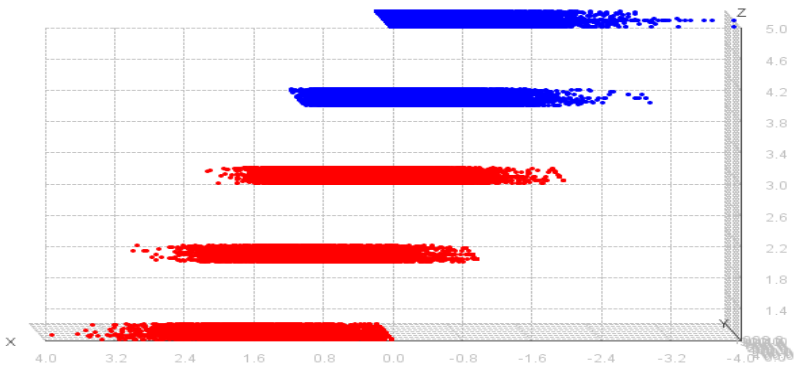
We will use Error rate as the performance measure.

As Accuracy measure will give the biased decisions towards the ratings close to the cut off value. If suppose the cut off value is set at 4 then the one movie with the predicted rating 3.9 or 3.8. And this will be the biased decision as there are chances that movie with 3.9 rating will prove as a good recommendation.

While working with error rates, we will provide a confidence interval based on the overall distribution of the error rates over the given prediction and their original ratings. This approach will give the better performance and can be useful for better recommendations.

We will choose the model will the least Root mean square error. And based on the RSME we will decide about the Confidence limits for a predicted rating for future recommendations.

For further classification and selection of the cut off rating will create confusion matrix and asses the class recalls and class precision to select proper cut off.



Error distribution at different level of ratings.

This graph indicates that lower ratings are being overly estimated as they lie in the positive quadrant of the graph above. & the higher ratings are undermined.
The major chunk of negative error is limited till "-2" approximately. While the positive error limit seems to be "3" approximately

This indicates that either:-

- There is insufficient data available for the lower ratings range
- There is not much related ratings between the users and movies that has been rated for the ratings 1 & 2. As their error span is more than the higher ratings.

To see the error distribution see the graph in appendix.

Error Distribution across users.

For most of the data the error seems to be leaning towards positive side. Predicted 1's have the less dispersion for the error rate .Where 0's have more dispersion. From the graph Errors seems to equally distribute for the positive cases (i.e. response 1) & not equally distributed for 0's.

Error Distribution across items.

Errors in this graphs are not equally distributed for any of the response (0's & 1's). The ratings based on items will be less useful as compared to the comparison based on users.

C) (1) 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:-

When we apply Global Average method and User Item Baseline methods we observe that User Item Baseline method give better accuracy. Also, changing the parameters does improve the accuracy a little notch but no significant difference was observed. For example, with default parameters following performance was observed on testing dataset

RMSE:
0.964
MAE: 0.766
NMAE:
0.191

With some changes in the number of iteration and regularization the accuracy was improved to as shown below on the testing dataset

RMSE:
0.961
MAE: 0.762
NMAE:
0.190

Also the pattern of accuracy between the training data and the testing data is nearly similar. This signifies that the distribution of ratings in both training dataset and testing dataset is similar.

User Item Baseline methods:

RMSE: 0.963
MAE: 0.765
NMAE:
0.191

With some changes in the number of iteration and regularization the accuracy was improved to as shown below on the testing dataset.

The important things observed is that,

- Change in number of iterations doesn't affect the performance keeping the regularization parameters same.
- The same value of user and item regularization parameters give the best results.
- Following results are obtaining at absolute value "4.0" as user & item regularization parameter.

RMSE: 0.958
MAE: 0.757
NMAE:
0.189

The user-item baseline provides a better performance than Global Average. RMSE, MAE and NMAE in Item baseline method is less when compared with Global Average method. RMSE is higher in Global Average, and thus the prediction of ratings & recommendation will be efficient using the User baseline method compared to global average.

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

Matrix Factorization:- When learning a matrix factorization model, the aim is to estimate User and Item in such a way that the model predictions minimize a loss on the training set. However, optimizing the model will typically yield poor predictive performance due to over fitting. With default settings Matrix factorization operator gives following performance:

RMSE:0.950
MAE: 0.759
NMAE:0.190

After trying different parameters, the best performance, we could get is

RMSE:0.943
MAE: 0.742
NMAE:0.186

While comparing different parameters we will try to minimize error rate of the model and at the same time focus on reducing the difference between performance of training and testing data. Regularization parameter is an important factor since it will help in reducing the problem of over fit. To achieve our goal, we tested different values for number of factors, learning rates, regularizations, and number of iterations. Increase in number of iteration reduces the error rate in training data amounting to the problem of over fit leading to poor performance in predicting the ratings on unseen data. We also reduced the number of factors to overcome the problem of over fit we increase the value of regularization and the difference between the error rates of training and testing data also decreases. Also reducing the number of factors to 5 decreased the difference in error rate and also improved the performance of the model.

Changing the learning rate does make a difference in the overall accuracy of the model. However, it is difficult to say if the changes in the performance is due to changes in learning rate. Since we observed changes in performance (error rate) even when there are no changes made in the parameters, i.e. if we run the model multiple times with consistent parameters we get new result every time.

(3) ***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:-

With default values:

RMSE:0.957
MAE: 0.754
NMAE:0.189

After changing parameters like k, correlation mode & regularization, the best performance we get is,

With k=30, Pearson correlation mode, user r=4, item r=4

RMSE:0.948
MAE: 0.742
NMAE:0.186

Item k-nn:-

With default values:

RMSE:0.945
MAE: 0.743
NMAE:0.186

After changing parameters like k, correlation mode & regularization, the best performance we get is,

With k=30, Pearson correlation mode, user r=4, item r=4

RMSE:0.937
MAE: 0.735
NMAE:0.184

With the increase in k the performance decreases and same goes with the regularization values. Change in regularization parameters tends to reduce the performance. Like the user baseline operator, here also the same regularization values tends to give the higher performance. The most important thing is that these operators are flexible for parameter changes.

RSME decreases when we change correlation mode from cosine similarity to Pearson. Pearson correlation mode gives the better performance in both User k-nn and item k-nn both.

The number of nearest neighbor which gives best performance across both the operators are same. k=30 for both the operators gives the best performance i.e. less RSME.

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

Operator	**RSME**	**NMAE**	**MAE**
Global average	0.961	0.190	0.762
User Baseline	0.958	0.189	0.757
Matrix Factorization	0.943	0.186	0.742
User k-nn	0.948	0.186	0.742
Item k-nn	0.937	0.184	0.735

By comparing the best performance of all the operators used above,

We will select the "item k-nn" operator as the RSME is less compared to other operators and it offers the flexibility as performance doesn't change drastically with the little change in parameter values.

Question 3

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

At cut off = 3

	Recall	precision
Global factorization	100	55.10
User baseline	94.27	64.44
Matrix factorization	97.24	64.89
User k-nn	95.54	68.28
Item k-nn	98.01	68.83

At cut off = 3.5

	Recall	precision
Global factorization	100	55.10
User baseline	75.11	74.62
Matrix factorization	80.21	77.55
User k-nn	82	80.83
Item k-nn	85.51	82.54

At cut off = 4

	Recall	Precision
Global factorization	100	55.10
User baseline	54.51	47.91
Matrix factorization	58.77	55.92
User k-nn	74.85	52.78
Item k-nn	78.34	55.61

Based on the tables above with considering the balance of recall and precision we will choose Item k-nn operator.

Selection Criterion:-

- Recall: This value shows that amongst actual 1's how many we have predicted as 1.
- Precision: This value indicates that amongst predicted 1's how many are of them are actual 1's.

From the tables above we can see that out of three observed cut off values, cut off value '3.5' gives the best results for all the operators which we have used. So 3.5 is the best cut off value as far as performance criterions are concerned.

Best performing operator results (item k-nn at cut off 3.5):

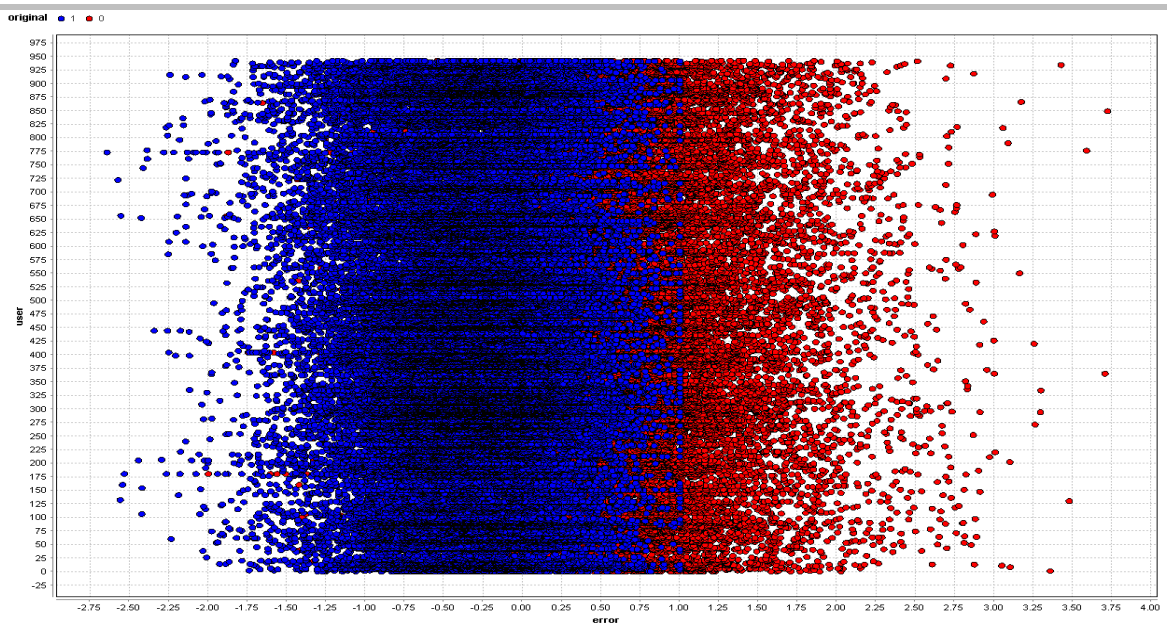
accuracy: 82.05%

	true 1	true 0	class precision
pred. 1	42676	9029	82.54%
pred. 0	7230	31635	81.40%
class recall	85.51%	77.80%	

We can see from the image above that misclassified prediction rate is low amongst the other results shown in appendix

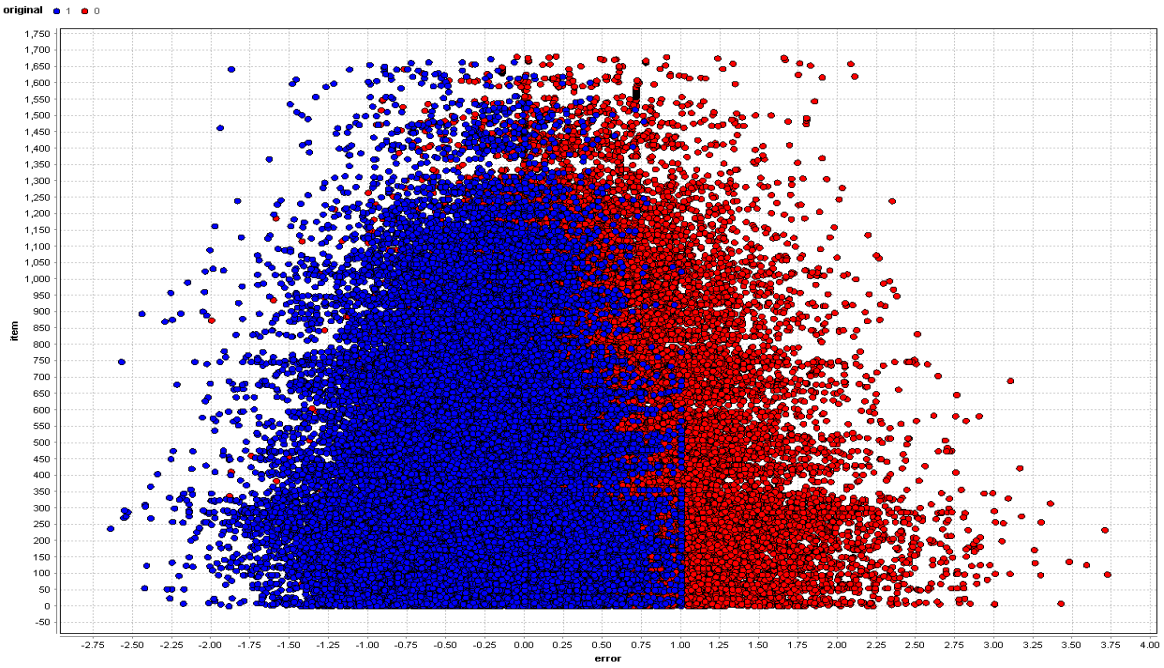
b. ***What value of 'cutoff' will you use? Are errors distributed equally across movies and across users?***

We will use 3.5 as cut off value to get the best recommendation system in order to efficiently recommend the users the movies that they would like to see.



Error distribution amongst users.

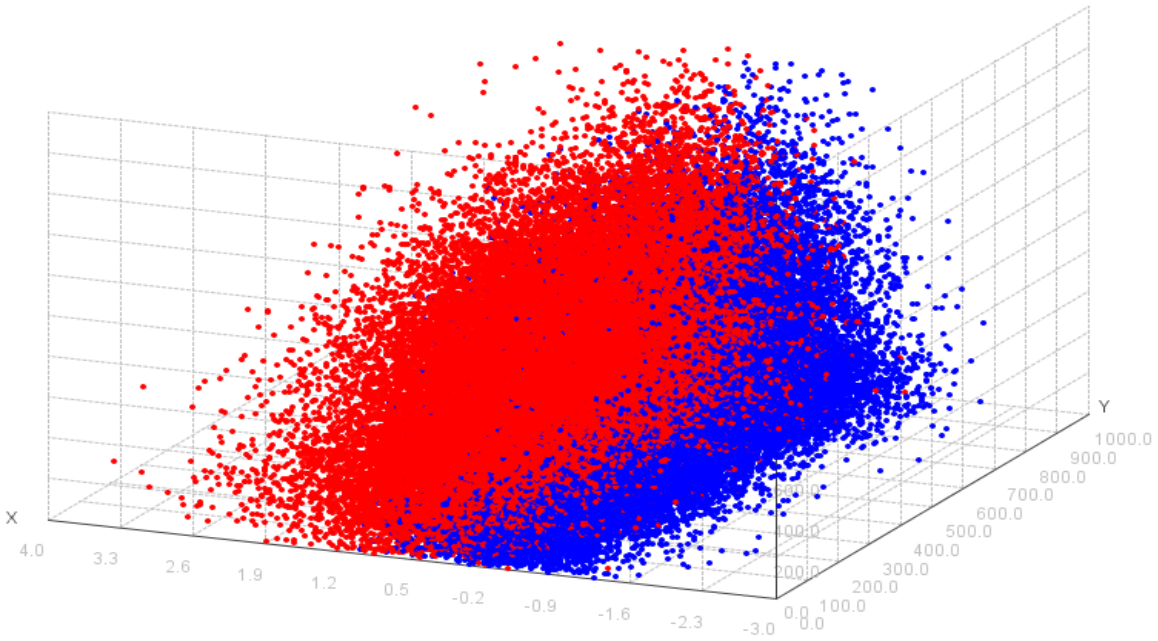
As seen from figure above it seems that the error distribution amongst users are not equally distributed. For the positive responses it seems that their error distribution are almost equally distributed & negative response have more sparse distribution.



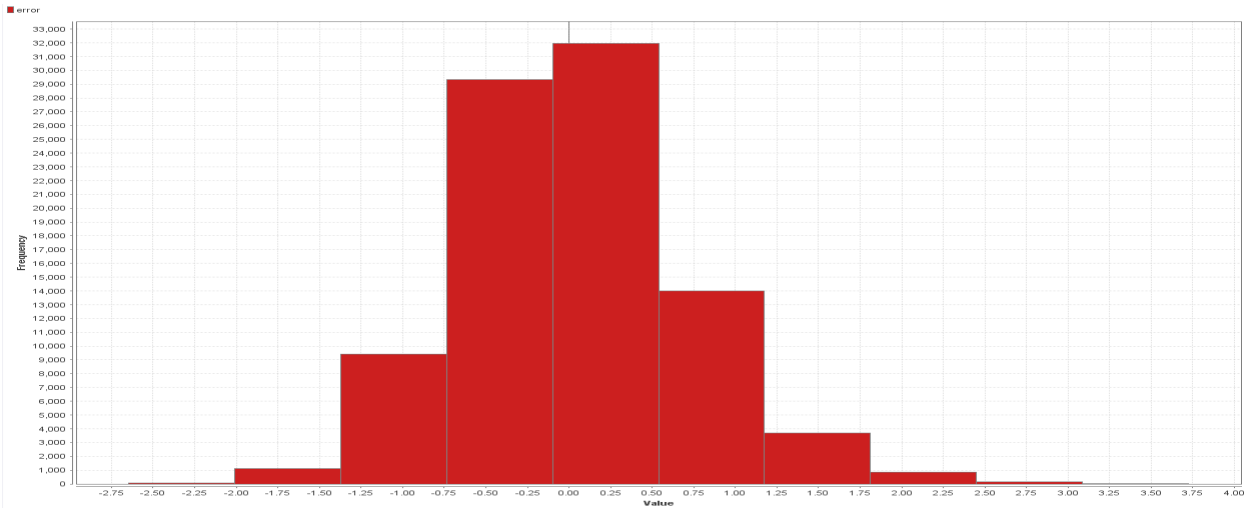
Error distribution amongst users.

As you go up in the graph the error rate interval is low compared to the movies at the bottom.Overall error distribution for positive & negative both the cases are not equal.The errors are scattered over the various ranges for the different items (movies).The pattern looks equally distributed towards the lower part of the graph.

Appendix.



Error distribution amongst the users and items.



Error distribution

Performance of the matrix factorization operator with different parameters

Matrix Factorization Table

	Training	Testing	Difference	iteration	Factors	regularization	Learn rate
RMSE	0.819	0.966	0.147	Random values			
MAE	0.644	0.763	0.119				
NMAE	0.161	0.191	0.03				
RMSE	0.797	0.952	0.155	30	6	0.015	0.01
MAE	0.625	0.751	0.126				
NMAE	0.156	0.188	0.032				
RMSE	0.744	0.964	0.22	30	10	0.015	0.01
MAE	0.584	0.757	0.173				
NMAE	0.146	0.189	0.043				
RMSE	0.81	0.948	0.138	30	5	0.015	0.01
MAE	0.635	0.746	0.111				
NMAE	0.159	0.186	0.027				
RMSE	0.813	0.954	0.141	30	5	0.1	0.01
MAE	0.637	0.751	0.114				
NMAE	0.159	0.188	0.029				
RMSE	0.814	0.962	0.148	30	5	0.015	0.01
MAE	0.637	0.756	0.119				
NMAE	0.159	0.189	0.03				
RMSE	0.786	0.978	0.192	30	5	0.1	0.015
MAE	0.613	0.76	0.147				
NMAE	0.153	0.19	0.037				
RMSE	0.847	0.95	0.103	50	5	0.1	0.02
MAE	0.676	0.756	0.08				
NMAE	0.169	0.189	0.02				
RMSE	0.854	0.958	0.104	50	5	0.1	0.05
MAE	0.68	0.763	0.083				
NMAE	0.17	0.191	0.021				
RMSE	0.849	0.943	0.094	50	5	0.1	0.02
MAE	0.675	0.751	0.076				
NMAE	0.169	0.188	0.019				

Performance of User-Item Baseline operator

User-Item Baseline Method

Changes in number of iteration

	RMSE		MAE		NMAE	
ITERATION	TRAINING	TESTING	TRAINING	TESTING	TRAINING	TESTING
10	0.92	0.963	0.729	0.765	0.146	0.153
15	0.92	0.963	0.729	0.765	0.146	0.153
50	0.92	0.963	0.729	0.765	0.146	0.153
75	0.92	0.963	0.729	0.765	0.146	0.153

Changes in number of regularization i

	RMSE		MAE		NMAE	
Regularization i	TRAINING	TESTING	TRAINING	TESTING	TRAINING	TESTING
30	0.927	0.979	0.74	0.767	0.148	0.153
10	0.924	0.97	0.731	0.763	0.146	0.153
5	0.919	0.961	0.724	0.761	0.145	0.152

Changes in number of regularization u

	RMSE		MAE		NMAE	
Regularization u	TRAINING	TESTING	TRAINING	TESTING	TRAINING	TESTING
30	0.932	0.965	0.737	0.78	0.147	0.156
10	0.922	0.961	0.733	0.772	0.146	0.155
5	0.915	0.960	0.727	0.762	0.146	0.152

Performance of different Models

Performance results for KNN

accuracy: 79.37%			
	true 1	true 0	class precision
pred. 1	40925	9704	80.83%
pred. 0	8981	30960	77.51%
class recall	82.00%	76.14%	

Performance results for Matrix Factorization: -

accuracy: 76.30%			
	true 1	true 0	class precision
pred. 1	40029	11589	77.55%
pred. 0	9877	29075	74.64%
class recall	80.21%	71.50%	

Performance results for User-Item Baseline: -

accuracy: 72.21%			
	true 1	true 0	class precision
pred. 1	37486	12751	74.62%
pred. 0	12420	27913	69.21%
class recall	75.11%	68.64%	

Performance results for Global Average: -

accuracy: 55.10%			
	true 1	true 0	class precision
pred. 1	49906	40664	55.10%
pred. 0	0	0	0.00%
class recall	100.00%	0.00%	