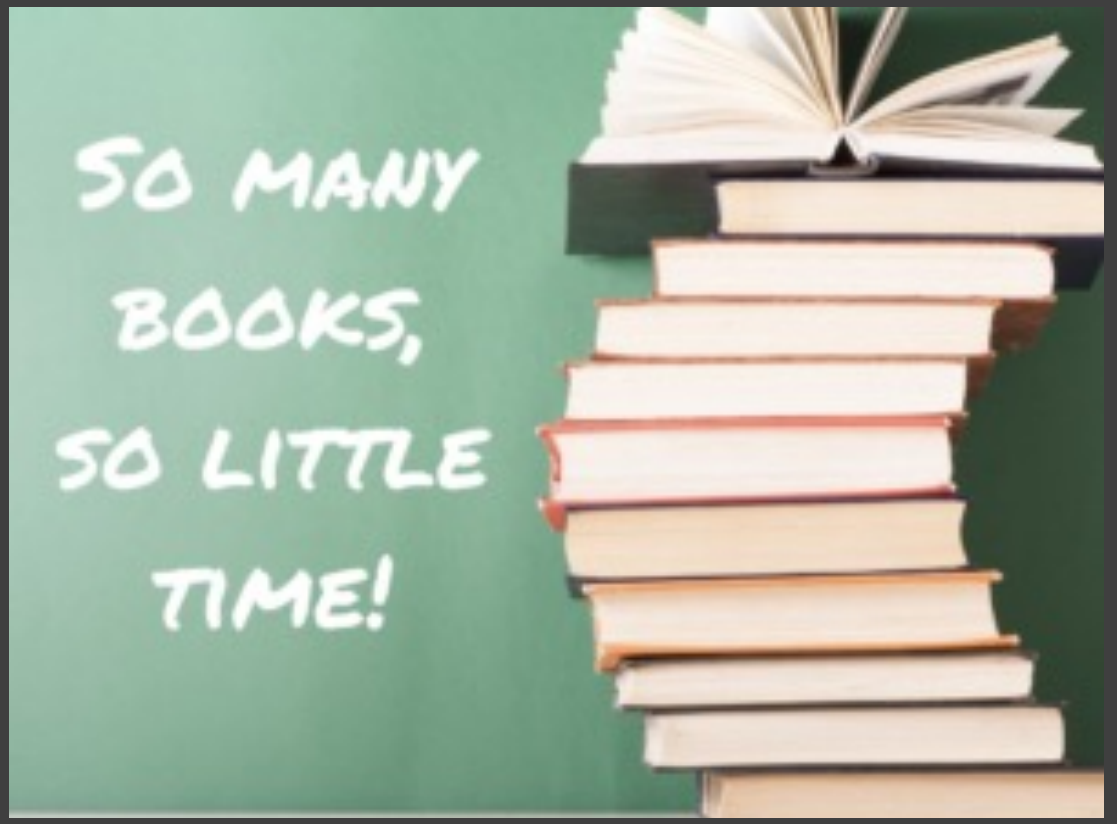


BOOK RECOMMENDATION

BY FARHANA AFROZE

What is recommender system?

- Recommender system is information filtering system that try to predict ratings or choices of a users and recommend product that are interesting to them.
- To build a recommender system there are two most popular approaches which are-
  1. Content based approach
  2. Collaborative approach

Content Based Approach: This approach requires lots of information rather than just knowing user interests.

Collaborative Approach: On the other hand, this approach doesn't need that much information. It needs user past experiences on a set of items so it can recommend product based on user historical preference.

Book Recommendation using Recommender system

The goal of this project is-

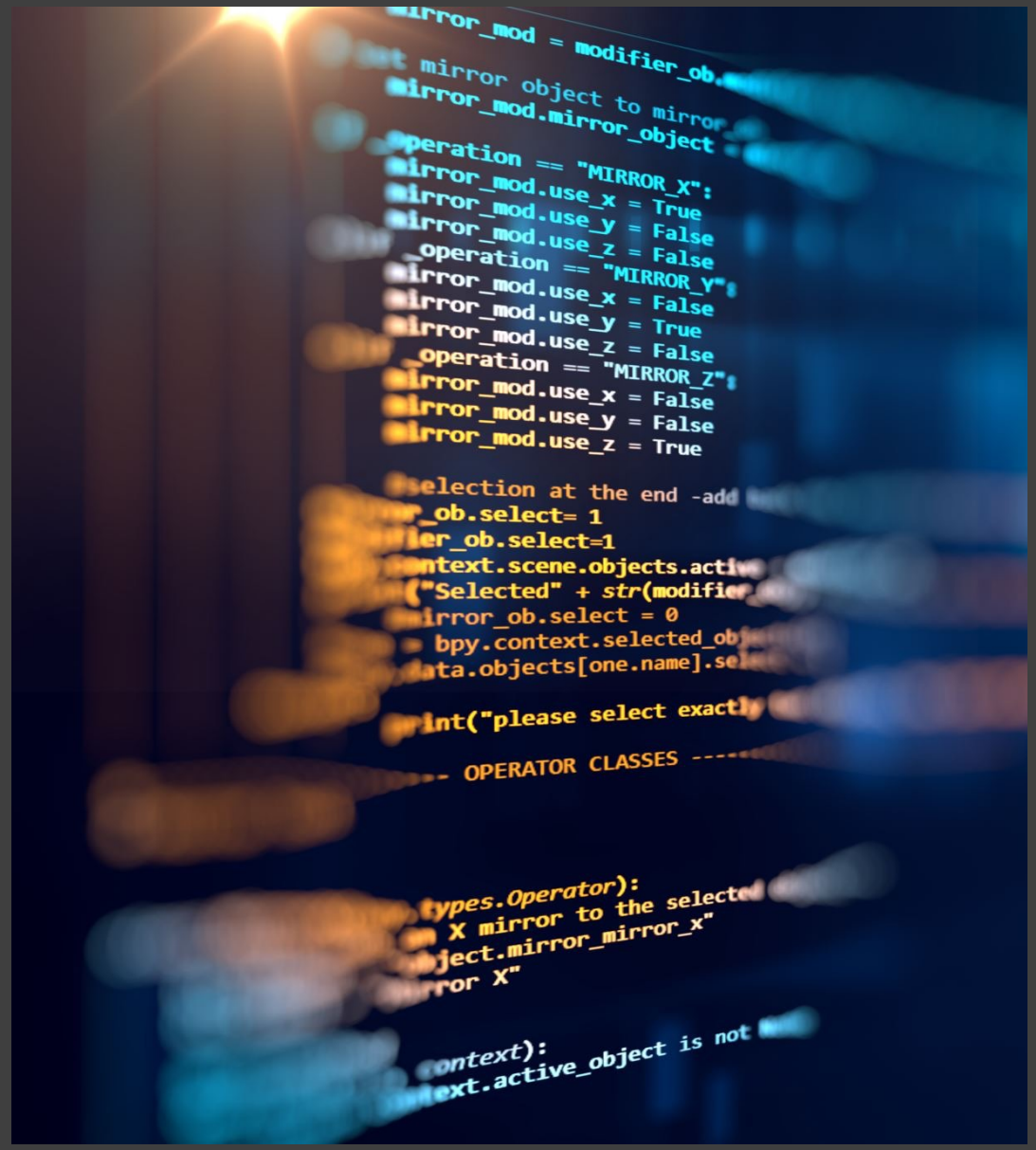
To find out similar users so we can recommend the book to new users based on the new user's similar interest of other users.

Finding out popular books.

Finding out new book that will interest the users based on their past buying of book.

Project Implementation Tools

- For project implementation I used ‘R’ programming language.
- I used recommender lab package for implementation.
- Used various algorithms from recommender lab



Data set for project

- The dataset I chose for this project is from Kaggle. Here is the link-

https://www.kaggle.com/arashnic/book-recommendation-dataset?select=Books.csv

The data set has three csv files.

1. Books.csv,
2. Ratings.csv and
3. Users.csv

Exploring the dataset

- In the Book.csv file it has 8 columns and 271,360 entries

Columns are- ISBN, Book.Title, Book.Author, Year.Of.Publication, Publisher, Image.URLS, Image.URL.M, Image.URL.L

- In the Rating.csv file it has 3 columns and 1,149,780 entries

Columns are- User.ID, ISBN, Book.Rating

- In the Users.csv file it has 3 columns and 278,858 entries

Coumns are- User.ID, Location, Age

Preprocessing the dataset

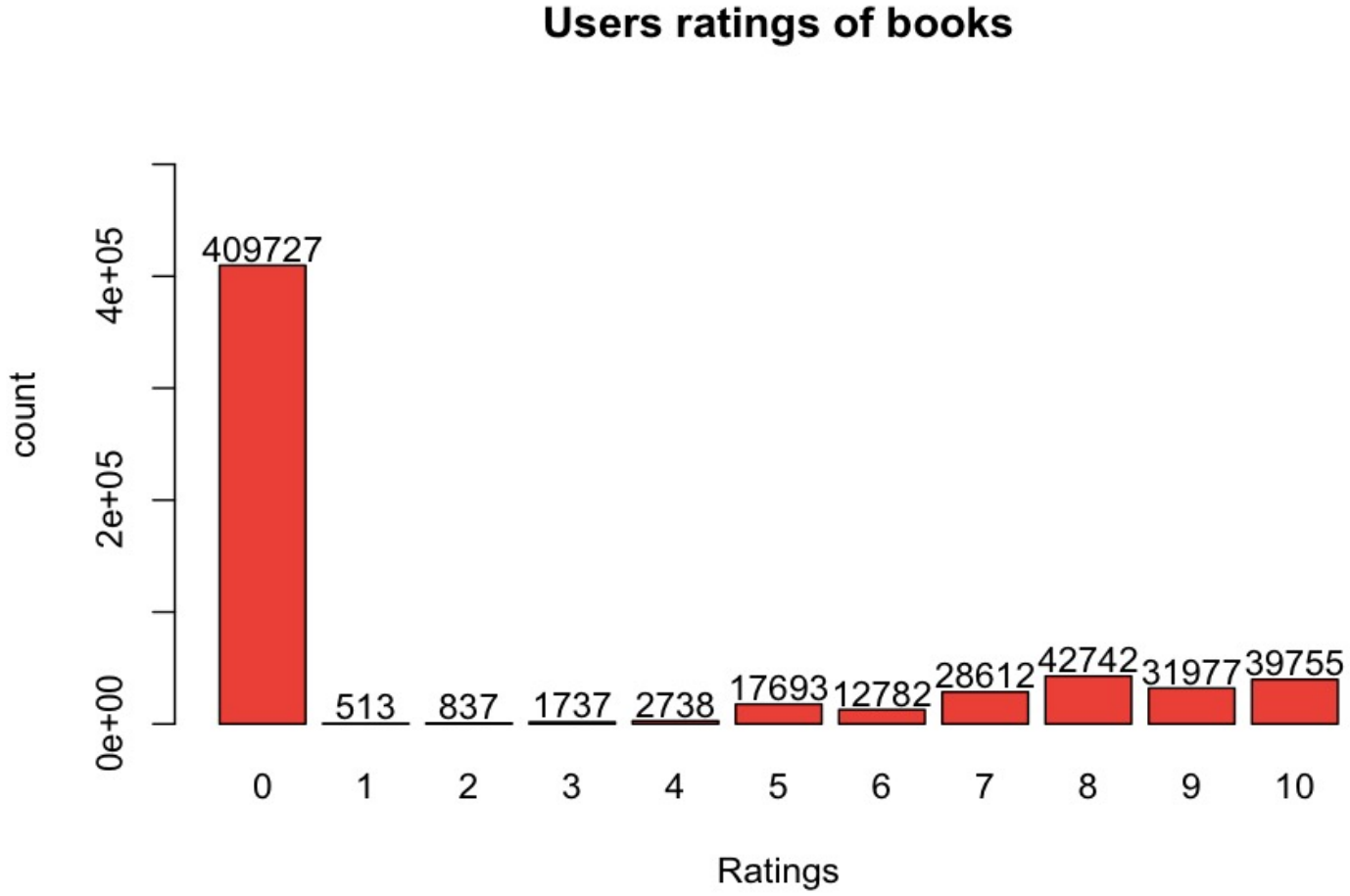
- At first, I would like to merging the book and rating dataset using the column ISBN
- Then I get the new dataset and again we would like to merge the new dataset with user dataset by using the column User.ID and the new dataset got 12 columns with 1,031,136 entries
- Next, I would like to see if there are any missing data in the dataset. There are like 277835 missing values. Then I omit missing values by deleting the rows. Now we got 73,301 entries.
- Next, I removed some columns that I don’t need it for building recommendation system.

Preprocessing the dataset

- Removed columns are-
- ISBN, Book.Author, Year.Of.Publication, Publisher, Location,Age, Image.URL.S, Image.URL.M, Image.URL.L
- Now, we have three columns which are User.ID, Book.Title and Book.Rating and has 753,301 entries.
- Next, I would remove duplicate rows where user rated same book twice. And there should be unique rows where user rated book once.
- After removing duplicate rows, we got 750095 entries in it.
- Then we would like to remove users who rated less than 25 books. After removing users we got 589,113 entries with 3 columns

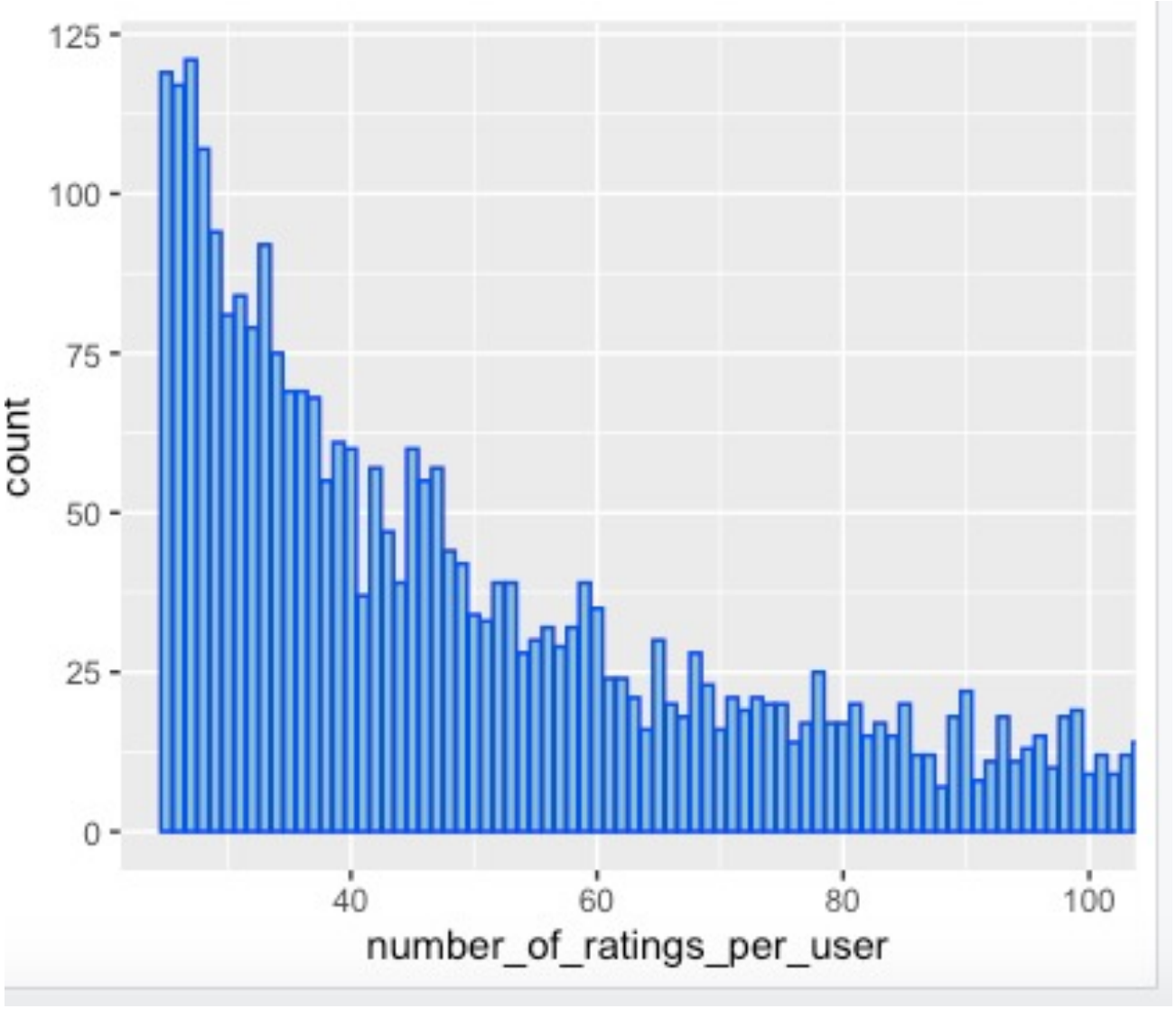
Visualization of the Dataset

First, We would like to see ratings frequency of the dataset-



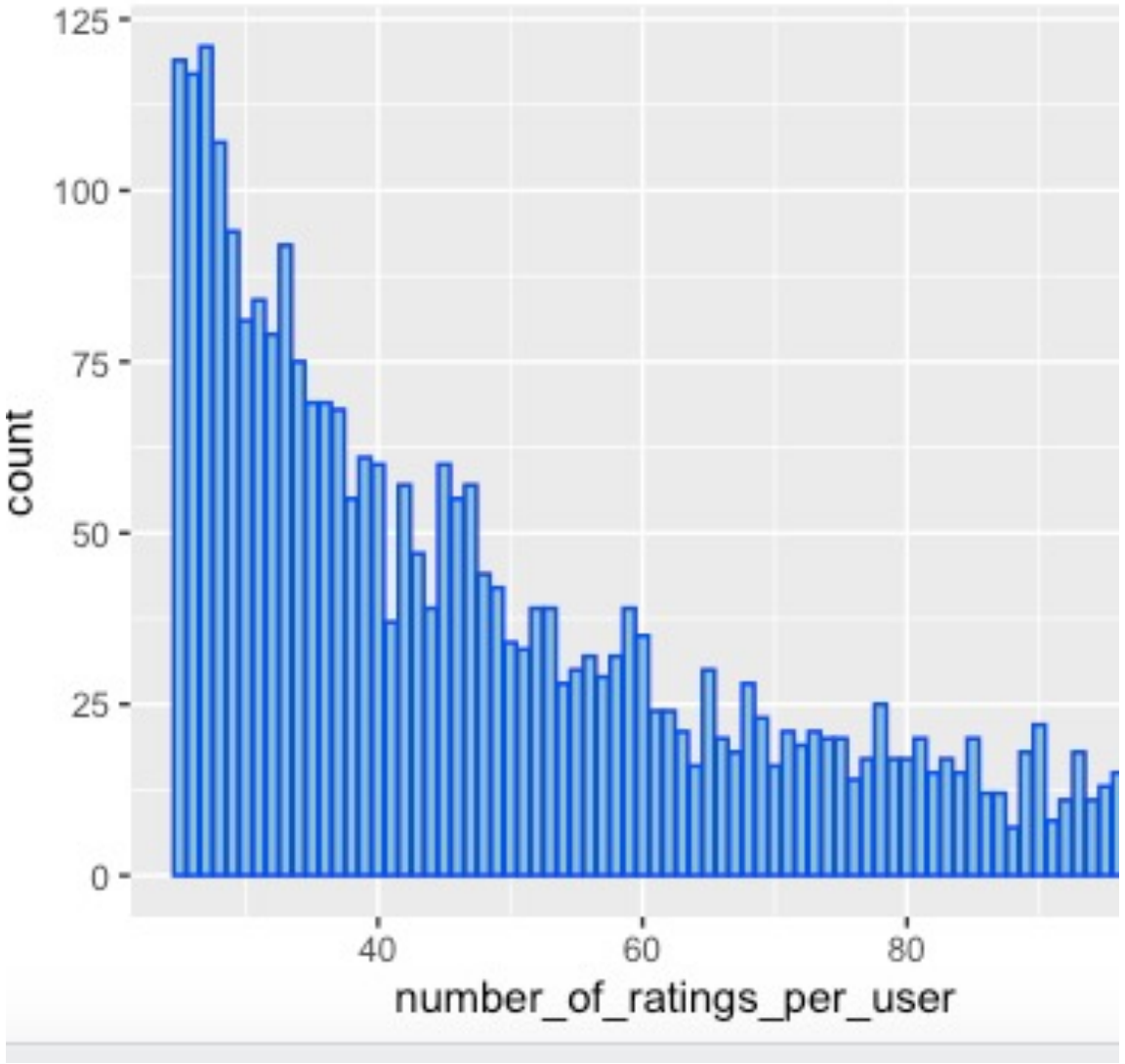
Visualization of the Dataset

- Here we would like to see number of book ratings per user-



Subset of the dataset

- In case we get memory issue while run that big dataset we would like to subset our dataset, so our recommender system doesn't halt while running.
- we chose 10% of the data using sample_frac() and then again we removed the users who rated less than 20 books. Now we have 36,803 entries
- Here is the number of ratings per user plot looks now-

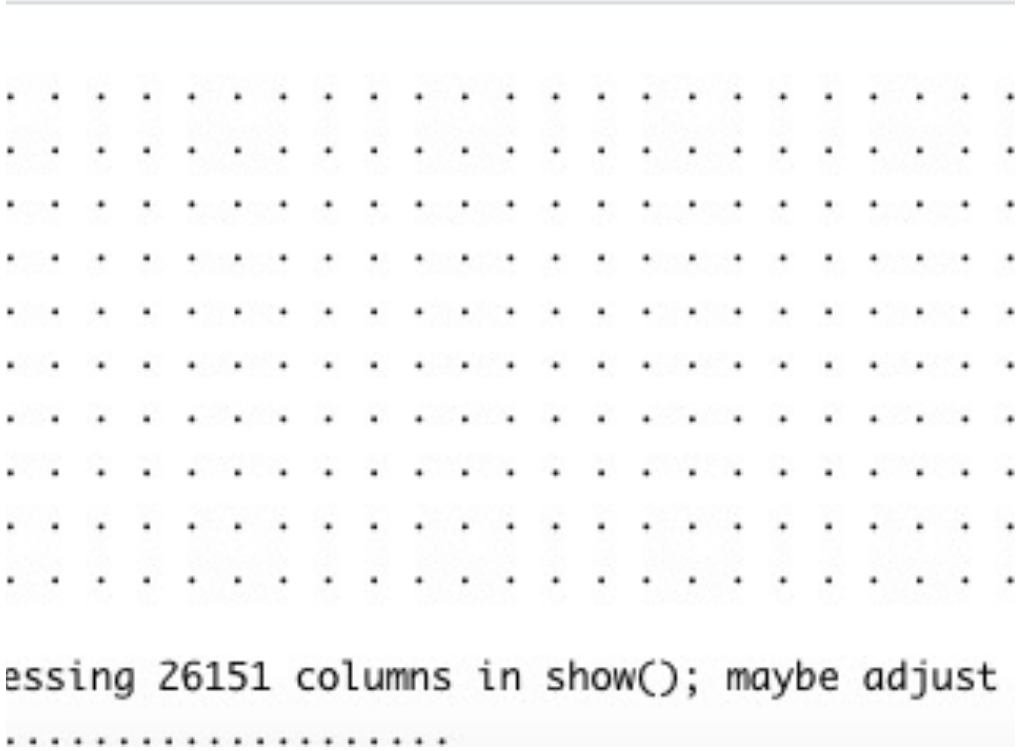


Recommender System Modeling

- For Recommender modeling we are going to use recommender lab package which will allow us building recommender model and evaluate recommendation system. For recommender lab to use we need data in matrix form. The data will convert into matrix using reaRatingMatrix.
- After converted into matrix we got 670 users with 26192 books with 36803 ratings.

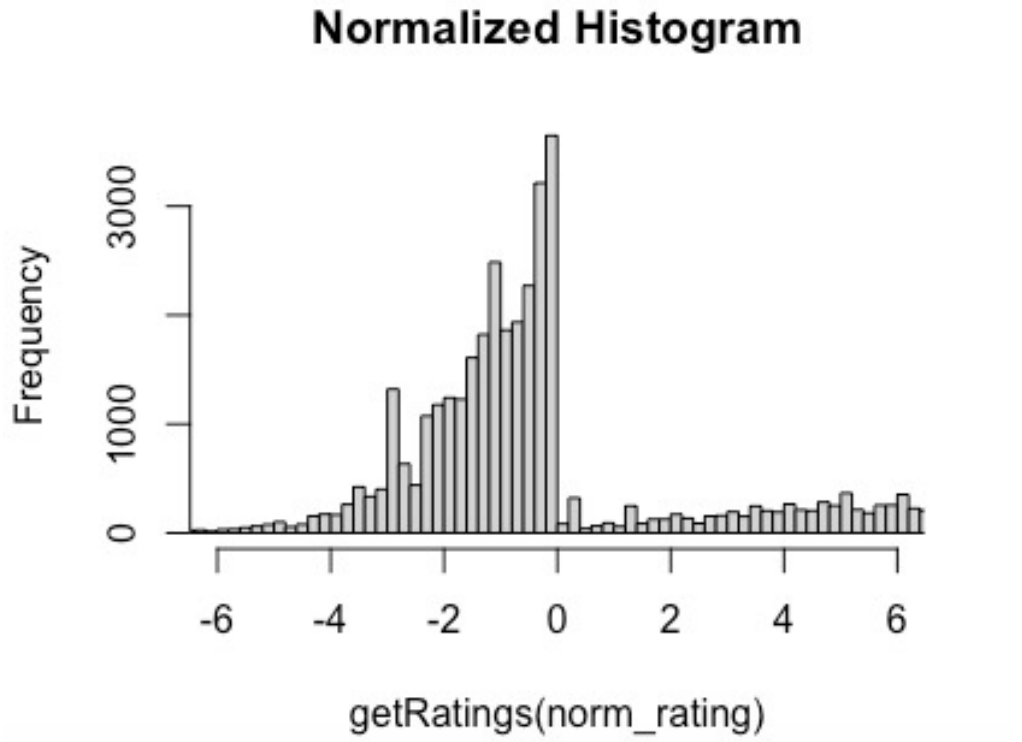
Looking through matrix and visualization

- First, we would like to see ratings of the user given.



Looking through matrix and visualization

- As we saw from previous page that matrix is very very sparse. We saw lots of users didn't rate books. And this is the reason matrix is very sparse.
- Next, we would like to normalize the matrix so if some of users gives very high ratings and some of user gives low ratings it won't effect. Let's see normalized rating graph-



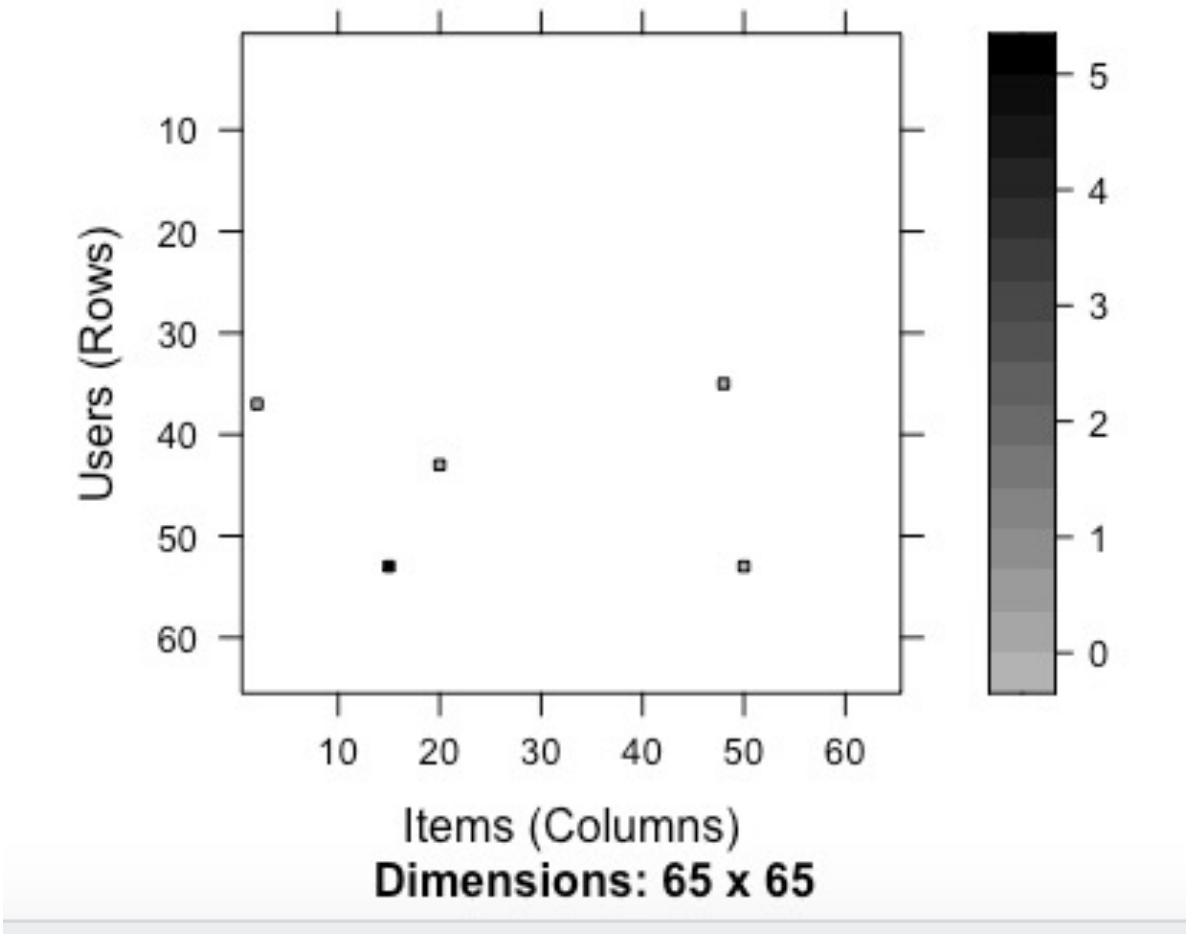
Looking through matrix and visualization

- Let’s see now how many ratings each ratings has-

```
ratingg
      0       1       2       3       4       5       6       7       8       9
17539945      26      26      59     100     786     535    1301    2046    1769
      10
    2047
```

Looking through matrix and visualization

- Here we would like to visualize overall ratings between users and books using 65 users and 65 books. And we would like to observe this through heatmap.

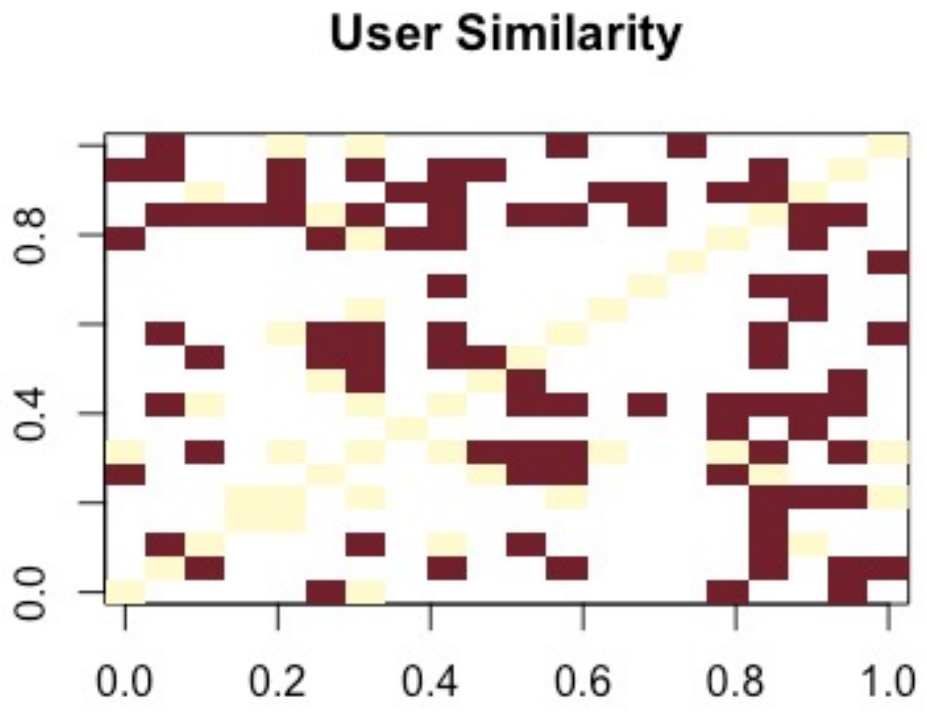


Data preparation for Recommender system

- As our data matrix is big we would like to subset the data matrix.
- We selected data where user rated books more than 7 times plus book has been rated more than 5 times.
- We got 670 users with 448 movies with 3797 ratings. And again we selected another dataset from this matrix. Where we selected users who rated more than 7 times. And the final data has 162*448 rating matrix of class ‘realRatingMatrix’ with 2026 ratings.

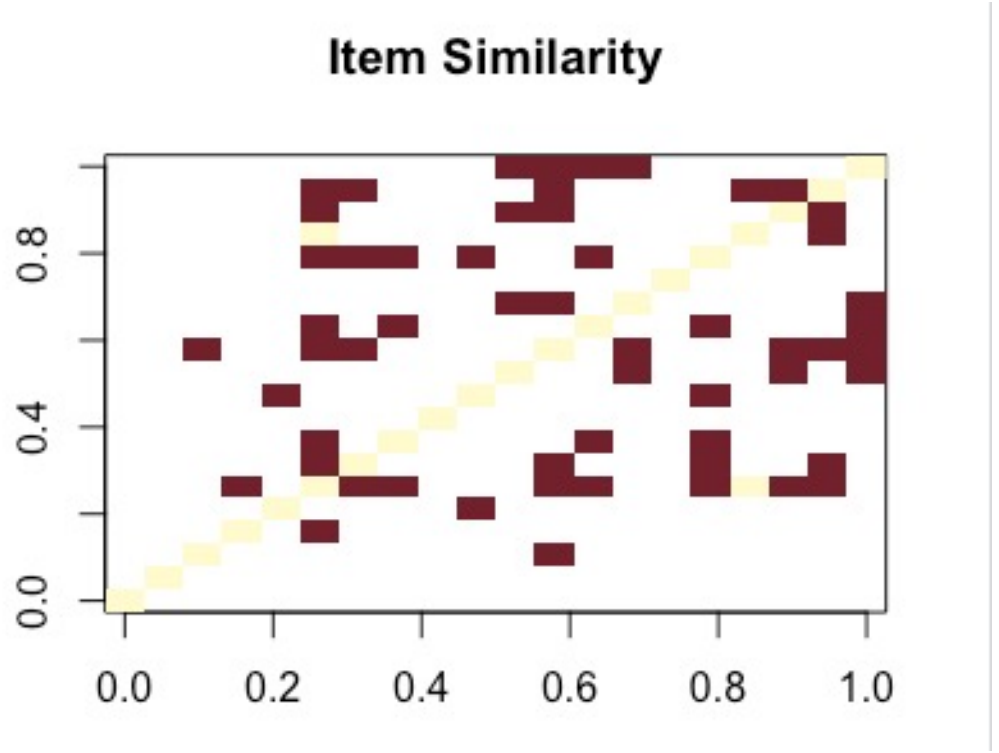
Similarity Matrix

- Here we would like to see similarity matrix between users and books
- Similarity between users calculated using cosine distance.



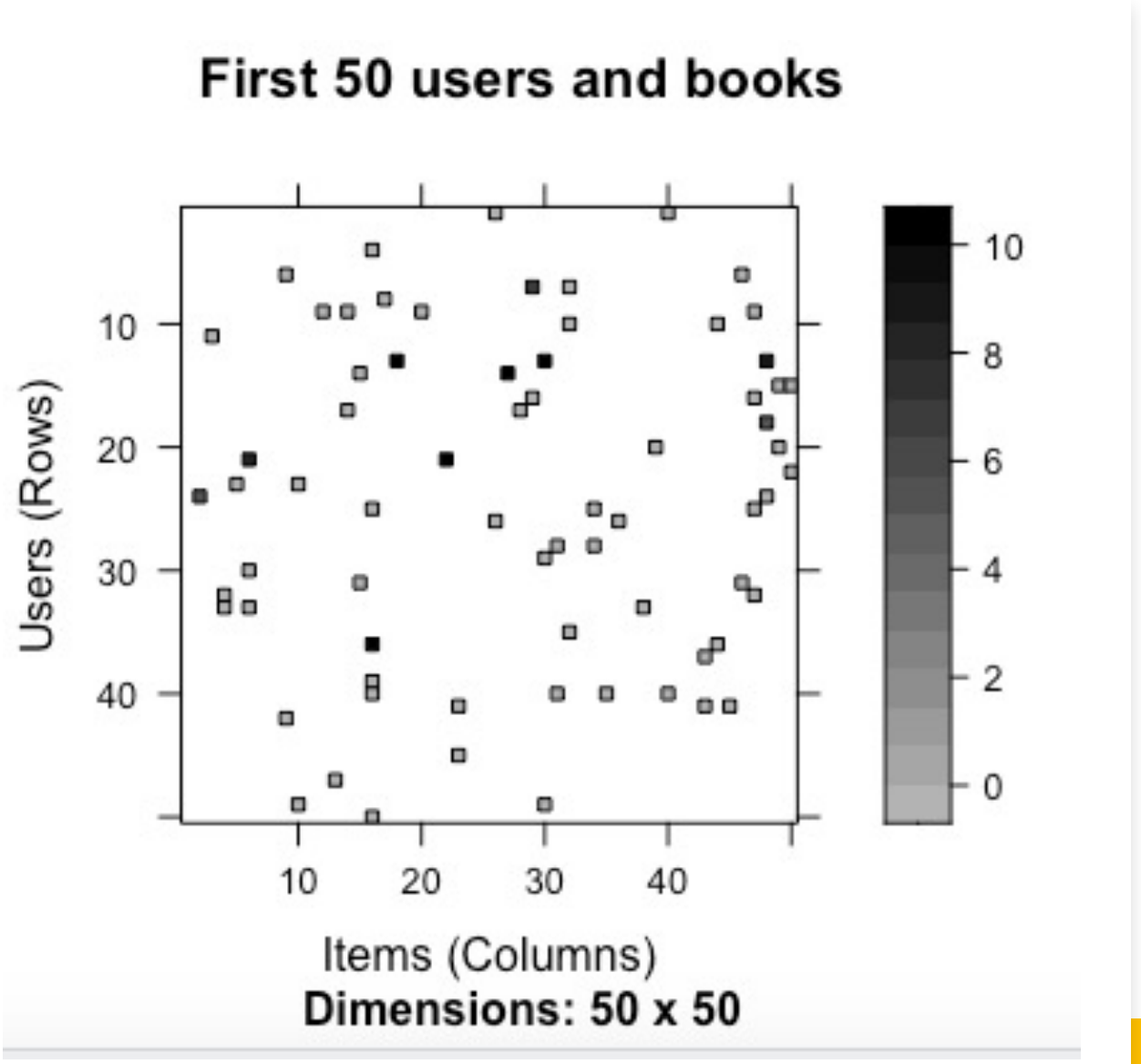
Similarity Matrix

- Similarity between items is calculated using cosine distance.



Heatmap to see overall ratings

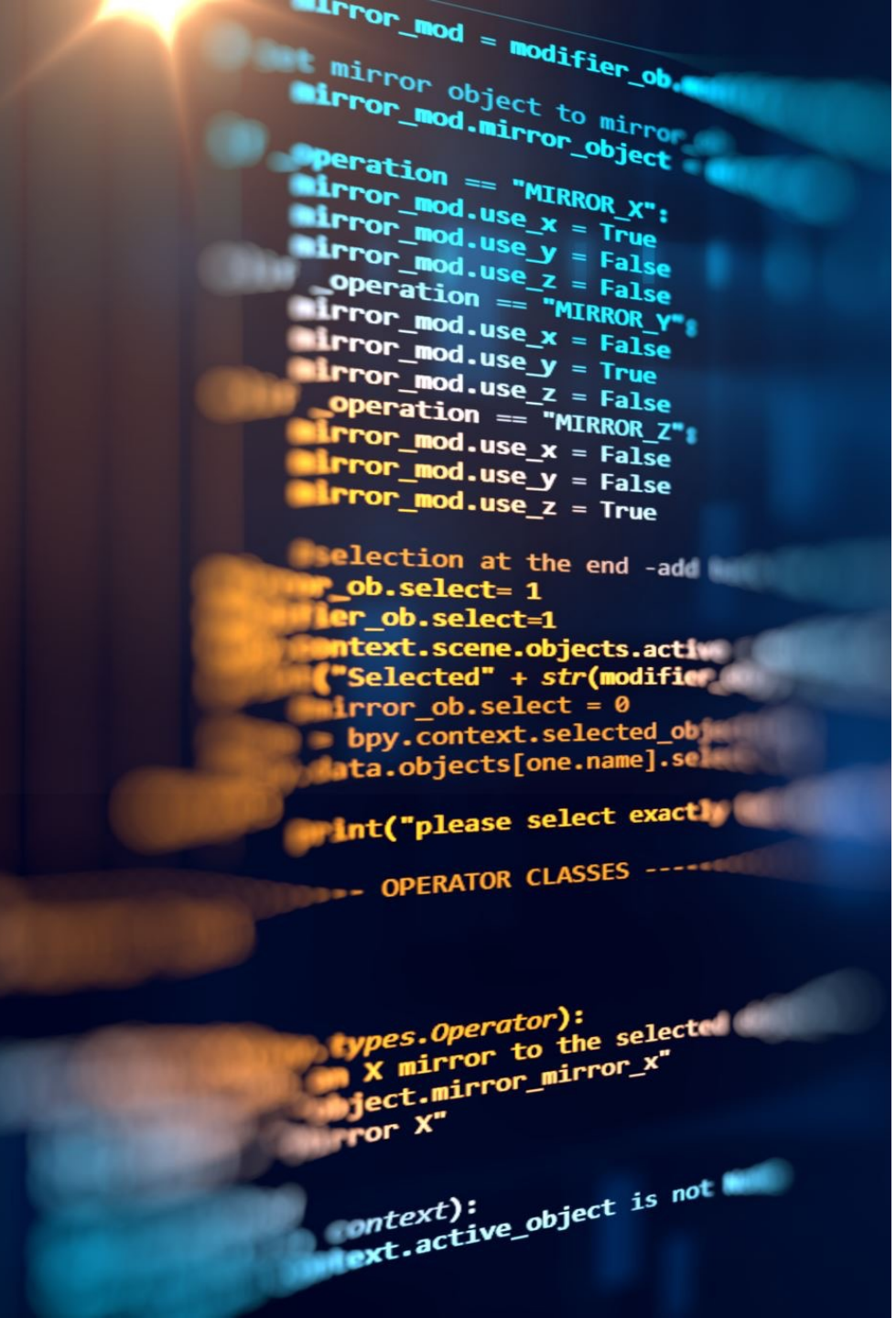
- Here we would like to see overall ratings between users and items through heatmap and it's based on the final dataset where we had 2026 ratings on 162 users and 448 books. We would like to see heatmap on first 50 users and first 50 books



Looking through some algorithm in recommender lab

- In the recommender lab there are various algorithm and supported algorithms are:

```
> names(model)
 [1] "HYBRID_realRatingMatrix"          "ALS_realRatingMatrix"
 [3] "ALS_implicit_realRatingMatrix"    "IBCF_realRatingMatrix"
 [5] "LIBMF_realRatingMatrix"           "POPULAR_realRatingMatrix"
 [7] "RANDOM_realRatingMatrix"          "RERECOMMEND_realRatingMatrix"
 [9] "SVD_realRatingMatrix"             "SVDF_realRatingMatrix"
[11] "UBCF_realRatingMatrix"
> 14
```



Algorithms of recommender lab

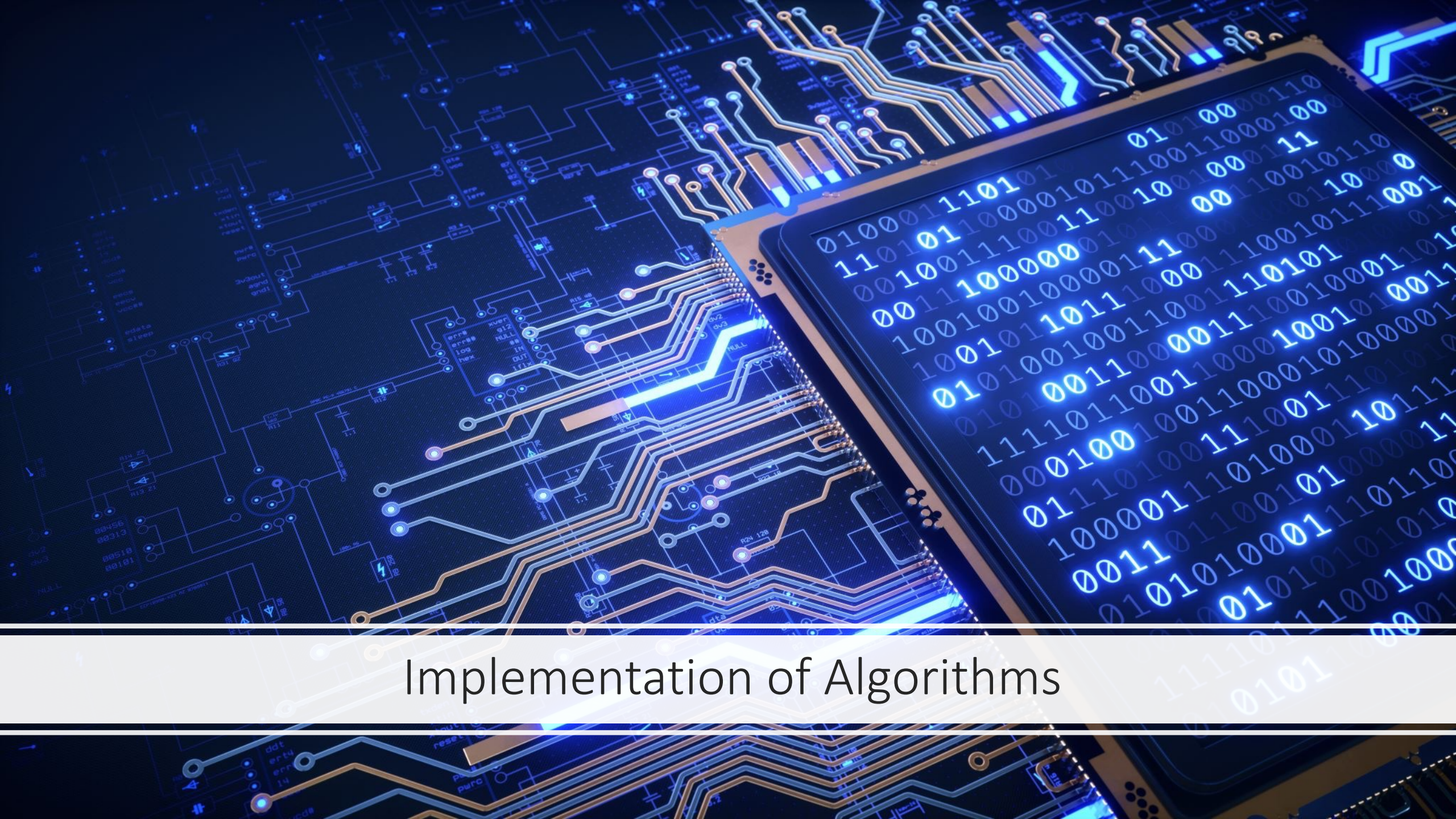
From those algorithm I implemented Three of those-

- UBCF algorithm
- Popular algorithm
- IBCF algorithm

Algorithm description

Let’s see the description of those three algorithms-

- Popular Algorithm: This algorithm is based on the item popularity. You choose top items based on the most rated items
- UBCF algorithm: UBCF algorithm recommends items by finding similar users to the user who recommends the movie. It used k-nearest neighbors to measure the distance between each pair of users.
- IBCF algorithm: In IBCF we try to find out item similar items to the user. We select the items based on the past buying history of users. IBCF uses cosine based similarity, jaccard distance and other types of distance measurement.



Implementation of Algorithms

Popular Algorithm

Here we implementing algorithm on the first 100 users.

Then we are predicting top 5 books on the 15 users. Here is the screen shot of first 4 users and their top 5 books-

```
'7072`
"Little Altars Everywhere: A Novel"
"Stupid White Men ...and Other Sorry Excuses for the State of the N
"Interview with the Vampire"
"A Time to Kill"
"The Lovely Bones: A Novel"

'8667`
"Little Altars Everywhere: A Novel"
"Stupid White Men ...and Other Sorry Excuses for the State of the N
"Interview with the Vampire"
"A Time to Kill"
"The Lovely Bones: A Novel"

:4299`
"Little Altars Everywhere: A Novel"
"Stupid White Men ...and Other Sorry Excuses for the State of the N
"Interview with the Vampire"
"The Lovely Bones: A Novel"
"The Catcher in the Rye"

:5233`
"Little Altars Everywhere: A Novel"
"Stupid White Men ...and Other Sorry Excuses for the State of the N
"Interview with the Vampire"
"A Time to Kill"
"The Lovely Bones: A Novel"
```

UBCF Algorithm

- So first we will implement the algorithm and then predict the algorithm on each user.
- We selected 5 most similar user for prediction
- So, what it will do is- UBCF will identify books in terms of having same interest by the same 5 users. And then it will predict books to a new user.

UBCF Algorithm outcome

- So, for the first user we found 40 ratings of 40 books-
- For the second user we found 0 books-
- For third user we found 34 books with 34 ratings
- Snapshot of third user books and ratings-(first couple ratings and books)

```
$`143175`
                                                                                                A Bend in the Road
                                                                                                         9.6000000
                                                                                                         Bleachers
                                                                                                         5.8333333
                                                                                                   Circle of Friends
                                                                                                         9.6000000
                                                                                  Confessions of an Ugly Stepsister : A Novel
                                                                                                         0.8333333
                                                                                                             Daddy
                                                                                                         6.8000000
                                                                                                     Dawn (Cutler)
                                                                                                         6.8000000
E Is for Evidence: A Kinsey Millhone Mystery (Kinsey Millhone Mysteries (Paperback))
                                                                                                         0.8333333
                                                                                        Heart of the Sea (Irish Trilogy)
                                                                                                         0.8333333
                                                                                                  Ladder of Years
                                                                                                         0.8333333
                                                                                  Lucy Sullivan Is Getting Married
                                                                                                         6.8000000
                                                                                                   Midnight Bayou
```

IBCF Algorithm

- Before implementing IBCF algorithm we splits our data matrix where 75% for train data and 25% for test data.
- In train data we got 112 users with 448 books with 1424 ratings
- In test data we got 50 users with 448 books with 602 ratings
- Next, we implemented IBCF algorithm. We have the parameter is k = 30 which is k-most similar items.
- Then we predicted using test data and we have parameters n = 5 which is number of books recommend to each user.
- Next page, Let's see the first 5 users with their 5 books recommended.

```
$`7346`
[1] "\\ Lamb to the Slaughter and Other Stories (Penguin 60s S.)\""
[2] "\\A\\\" is for Alibi : A Kinsey Millhone Mystery (A Kinsey Millhone Mystery)\""
[3] "\\It's Never Too Late to Have a Happy Childhood\\\": Inspirations for Adult Children\""
[4] "\\Surely You're Joking, Mr. Feynman!\\\": Adventures of a Curious Character\""
[5] "\\The Man with the Twisted Lip (Penguin 60s S.)\""

$`12538`
character(0)

$`13552`
[1] "101 Dinosaur Jokes"
[2] "12 Steps to Living Without Fear"
[3] "18Mm Blues"
[4] "A 5th Portion of Chicken Soup for the Soul : 101 Stories to Open the Heart and Rekindle the Spirit"
[5] "A Confederacy of Dunces (Evergreen Book)"

$`16634`
[1] "102 Haunted House Jokes"
[2] "A Cast of Killers"
[3] "A Deadly Little Christmas"
[4] "A Deepness in the Sky : A Novel (Zones of Thought)"
[5] "2001 Lippincott's Nursing Drug Guide (Book with Mini CD-ROM for Windows &amp;amp; Macintosh)"

$`31315`
[1] " Twenty Thousand Leagues Under the Sea (Illustrated Classics)"
[2] "101 Hopelessly Hilarious Jokes"
[3] "40 Tons Of Trouble (Women Who Dare) (Harlequin Super Romance, No 726)"
[4] "A 4th Course of Chicken Soup for the Soul: 101 More Stories to Open the Heart and Rekindle the Spirit"
[5] "A Breach of Promise (William Monk Novels (Paperback))"
```

IBCF Algorithm

Comparing different Recommender Algorithms

Comparing Different algorithms

- Here we would like see performance of IBCF, Popular and random methods and would like to compare them so we know which methods is predicting best.
- We like to evaluate these using recommender lab. We would like to use k-fold cross validation for that. The parameters for evaluation scheme are k = 3 which means we chose 3-fold cross validation, then given = 4 which is at least 4 books user rated and the goodRating = 4 which is we consider ratings good at least 4.
- Next, we will evaluate this model with all 3 algorithms with parameters n = c(1,3,5,10,15,20,25,30) which is the top recommendation list

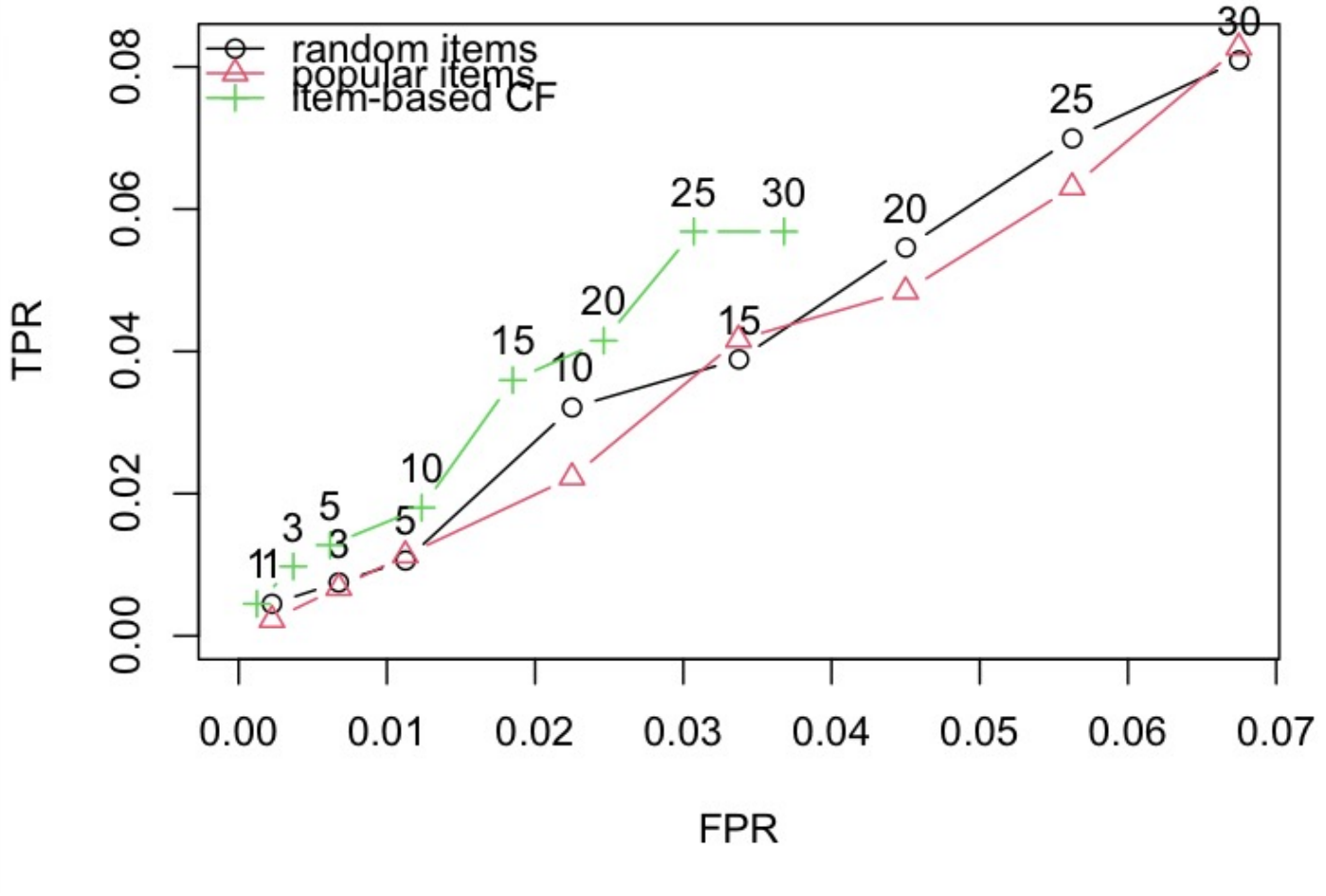
Algorithm prediction time

- Let’s see all 3 methods prediction or modeling time-

```
RANDOM run fold/sample [model time/prediction time]
         1  [0.017sec/0.097sec]
         2  [0.005sec/0.066sec]
         3  [0.002sec/0.072sec]
POPULAR run fold/sample [model time/prediction time]
         1  [0.016sec/0.271sec]
         2  [0.01sec/0.257sec]
         3  [0.011sec/0.259sec]
IBCF run fold/sample [model time/prediction time]
         1  [0.19sec/0.049sec]
         2  [0.179sec/0.037sec]
         3  [1.272sec/0.036sec]
>
```

Graph of the evaluation model

- Here we would like to see the TPR and FPR graph of three algorithms.



Graph Interpretation

- So, from last page graph what's the TPR and FPR?
- TPR(True Positive Rate): These are recommended books that have been purchased.
- FPR(False Positive Rate): These are recommended books that haven't been purchased
- From the previous graph we see IBCF has FPR is less than other methods. It's showing that this algorithm finds out more recommending items that have been purchased and also it's not misinterpreting like other two methods where other two methods have more recommended items that haven't been purchased. But also other two recommendation system is doing better in TPR. Here IBCF is doing best as it is area under the curve an has less FPR.

Confusion matrix result

Next page, let’s see confusion matrix result for each methods

IBCF confusion matrix result

TP	FP	FN	TN	N	precision	recall	TPR	FPR	n
0.0185185	0.5740741	1.851852	441.5556	444	0.0312500	0.0135135	0.0135135	0.0013001	1
0.0555556	1.7222222	1.814815	440.4074	444	0.0312500	0.0292793	0.0292793	0.0038999	3
0.0740741	2.8888889	1.796296	439.2407	444	0.0250000	0.0382883	0.0382883	0.0065419	5
0.1111111	5.8148148	1.759259	436.3148	444	0.0187500	0.0540541	0.0540541	0.0131676	10
0.1296296	8.7592593	1.740741	433.3704	444	0.0145833	0.0675676	0.0675676	0.0198355	15
0.1296296	11.7222222	1.740741	430.4074	444	0.0109375	0.0675676	0.0675676	0.0265453	20
0.1481481	14.6666667	1.722222	427.4630	444	0.0100000	0.0765766	0.0765766	0.0332132	25
0.1481481	17.6296296	1.722222	424.5000	444	0.0083333	0.0765766	0.0765766	0.0399230	30

POPULAR Confusion Matrix result

TP	FP	FN	TN	N	precision	recall	TPR	FPR	n
0.0185185	0.9814815	1.851852	441.1481	444	0.0185185	0.0067568	0.0067568	0.0022197	1
0.0370370	2.9629630	1.833333	439.1667	444	0.0123457	0.0202703	0.0202703	0.0067015	3
0.0555556	4.9444444	1.814815	437.1852	444	0.0111111	0.0270270	0.0270270	0.0111830	5
0.0925926	9.9074074	1.777778	432.2222	444	0.0092593	0.0414414	0.0414414	0.0224079	10
0.1111111	14.8888889	1.759259	427.2407	444	0.0074074	0.0684685	0.0684685	0.0336752	15
0.1111111	19.8888889	1.759259	422.2407	444	0.0055556	0.0684685	0.0684685	0.0449843	20
0.1481481	24.8518519	1.722222	417.2778	444	0.0059259	0.0887387	0.0887387	0.0562094	25
0.1666667	29.8333333	1.703704	412.2963	444	0.0055556	0.1157658	0.1157658	0.0674767	30

>

RANDOM Confusion matrix result

TP	FP	FN	TN	N	precision	recall	TPR	FPR	n
0.0185185	0.9814815	1.851852	441.1481	444	0.0185185	0.0135135	0.0135135	0.0022199	1
0.0370370	2.9629630	1.833333	439.1667	444	0.0123457	0.0225225	0.0225225	0.0067016	3
0.0370370	4.9629630	1.833333	437.1667	444	0.0074074	0.0225225	0.0225225	0.0112252	5
0.0555556	9.9444444	1.814815	432.1852	444	0.0055556	0.0315315	0.0315315	0.0224923	10
0.1111111	14.8888889	1.759259	427.2407	444	0.0074074	0.0518018	0.0518018	0.0336751	15
0.1481481	19.8518519	1.722222	422.2778	444	0.0074074	0.0878378	0.0878378	0.0449004	20
0.1666667	24.8333333	1.703704	417.2963	444	0.0066667	0.0968468	0.0968468	0.0561675	25
0.1851852	29.8148148	1.685185	412.3148	444	0.0061728	0.1103604	0.1103604	0.0674347	30

>

Evaluating Metrics parameter-

- False Negative- These are not recommended books that have been purchased
- True Negative- These are not recommended books that haven’t been purchased.
- Recall – This is the measure of how many true positives get predicted out of all the positives in the dataset.
- Precision – This is the measure of correctness of positive prediction

Comparing the models- IBCF

IBCF metrics result we got-

Precision = 0.0083 for n=30 top items

Recall = 0.076 for n = 30 top items

TPR = 0.076 for n = 30 top items

FPR = 0.0399 for n = 30 top items

Comparing the models- Popular

Popular method metrics result we got-

Precision = 0.0055 for n = 30 top items

Recall = 0.115 for n = 30 items

TPR = 0.115 for n = 30 items

FPR = 0.064 for n = 30 items

Comparing the models- Random

- Precision- 0.006 for n = 30 top books
- Recall- 0.11 for n = 30 top books
- TPR- 0.011 for n = 30 top books
- FPR- 0.067 for n = 30 top book

Conclusion

- Based on those previous models I would choose IBCF model as this model precision is higher and FPR is lower than other two models. So, I would like to have more correct positive prediction than false positive prediction. And also I want FPR is lower because I want the product that has been recommended wants them purchased.