DEVELOP RECOMMENDATIONS USING PRODUCT RATINGS DATA

RECOMMENDATION ENGINE FOR AMAZON BABY PRODUCTS

Submitted by :
Tripti Singh
M S in Data Analytics [2016 – 2018]

Submitted to :
Dr. Hong Lin
Professor
College of Science & Technology
University of Houston - Downtown

Data

Two data files were available in json format: Ratings Data and Metadata

Ratings Data: 915,446 observations on 9 variables –

X	- serial number of the observation
reviewer ID	- 531,890 unique reviewers
asin	- 64,426 unique baby products
helpful	- number of votes on Yes, No; [Yes, No]
reviewText	- review text
overall	- product ratings, 5 levels – 1/2/3/4/5
summary	- review summary text
unixReviewTime	- date and time of review

Data

Metadata: 71,317 observations on 8 variables –

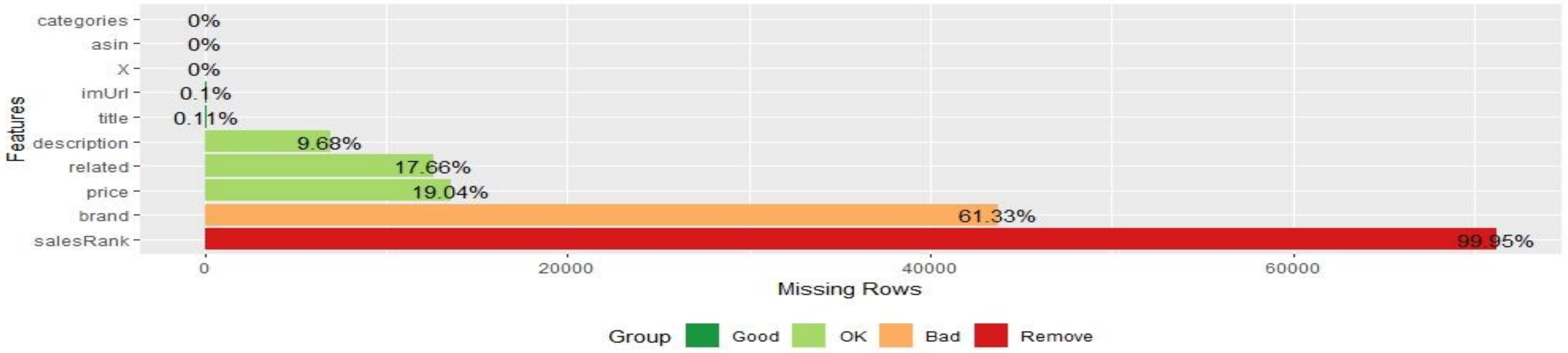
X	-	serial number of the observation
asin	-	71,317 unique baby products
title	-	name of the product
price	-	price in USD (at the time of crawl)
imURL	-	url of the product image
related	-	related products (also bought/ also viewed / bought together /buy after viewing)
salesRank	-	sales rank information
brand	-	brand name
categories	-	list of categories the product belongs to

Data Preparation

Lot of time was spent in converting data from JSON to .csv

Had to clean some encoded characters such as '->' for R to read .csv. Removed Reviewer Name column as that was ridden with such encoded characters and was redundant in our analysis

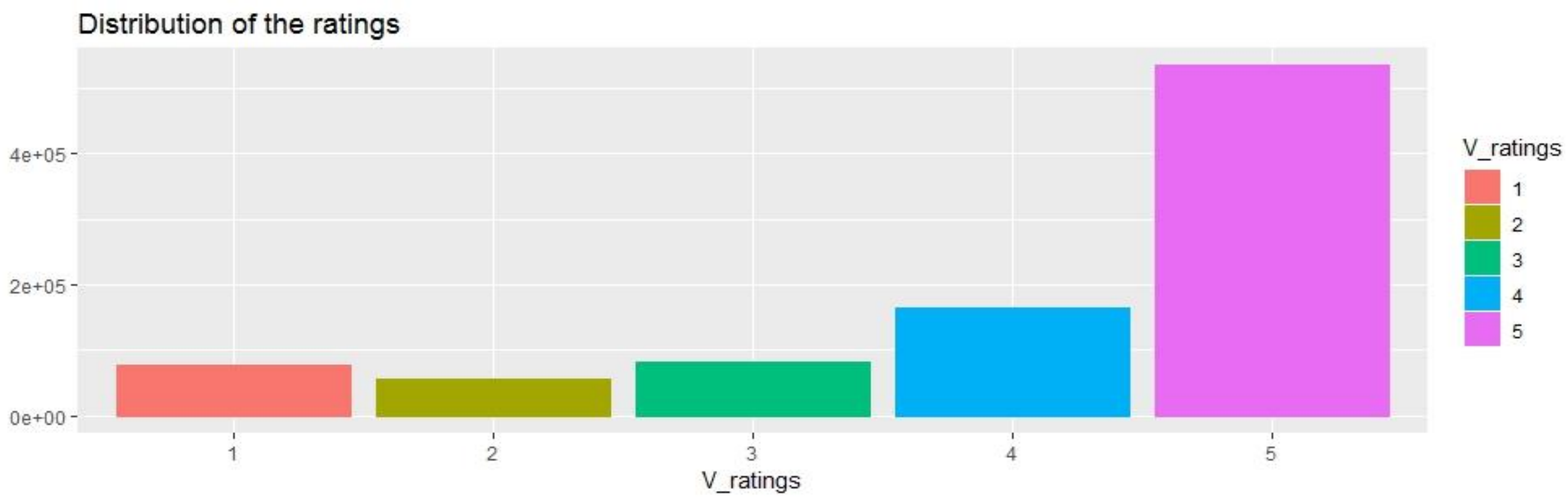
There were lot of empty cells in data so my objective to create a perfect data-set by merging Ratings Data and Metadata was not realized



Exploratory analysis

Majority of users rated items 4 or 5 (76%)

1	2	3	4	5
8.40	6.15	9.05	18.03	58.34



Majority of users have reviewed very few items

99% have reviewed less than 7 items, 72% have reviewed just 1 item

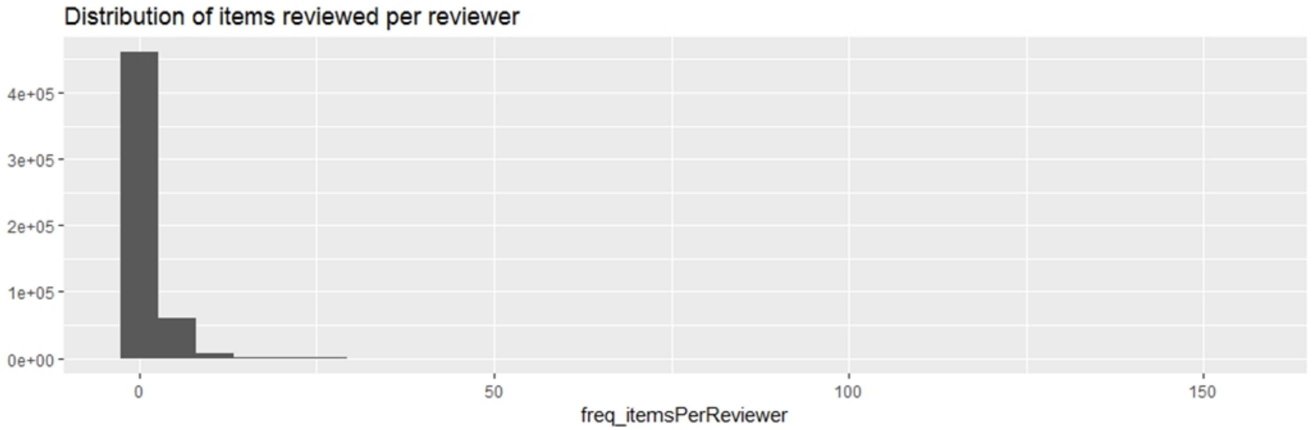
freq_itemsPerReviewer (%)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
72	14	6	3	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

53	54	55	56	57	59	60	61	64	65	66	69	71	74	76	77	78	81	82	93	95	105	122	140	155
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	ReviewerID	ItemsReviewed
498579	ARIFCL50JD5SK	155
466859	AJGU56YG8G1DQ	140
450552	AF8SREA2XE7BJ	122
466374	AJC88791BZEW7	105
169307	A276OI0NHBYORX	95
241127	A2PNW6QDW8OPY0	93
86967	A1M5ZT35YX6TIN	82
563	A100L918633LUO	81
486302	AOEUN9718KVRD	81
526652	AYNNJ0DBGL5H7	81



Majority of items are reviewed by very few users

91% items have been reviewed by less than 20 users, 52% items are reviewed by 1-2 users only

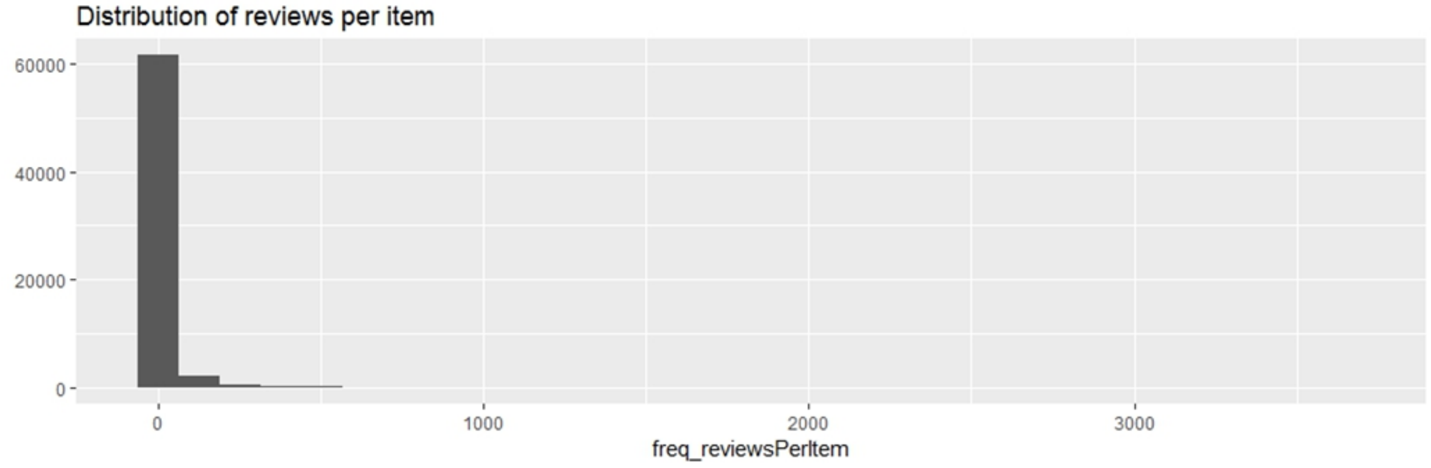
freq_reviewsPerItem (%)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
36	16	9	6	4	3	2	2	2	1	1	1	1	1	1	1	1	1	1	1	0
22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

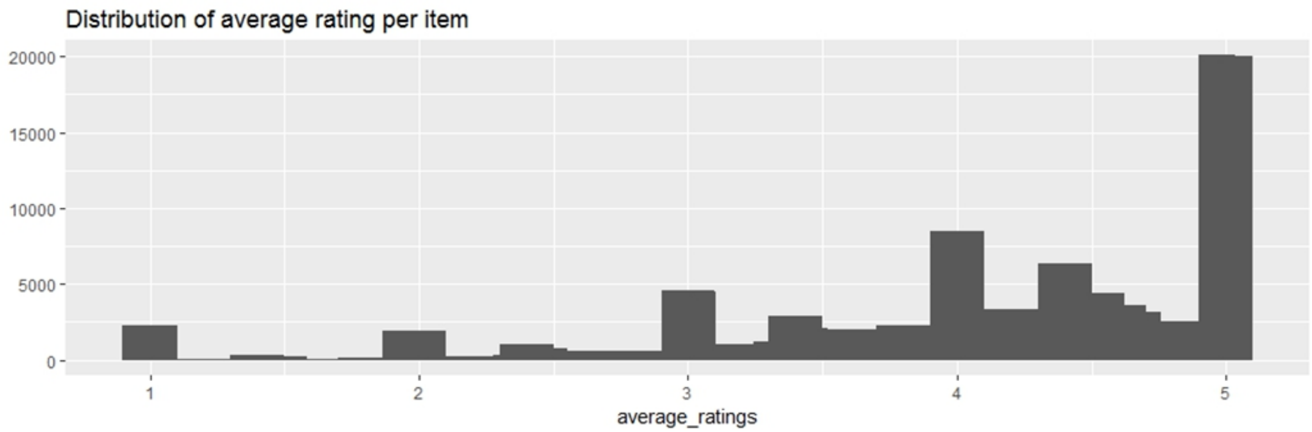
……….continued till 3648

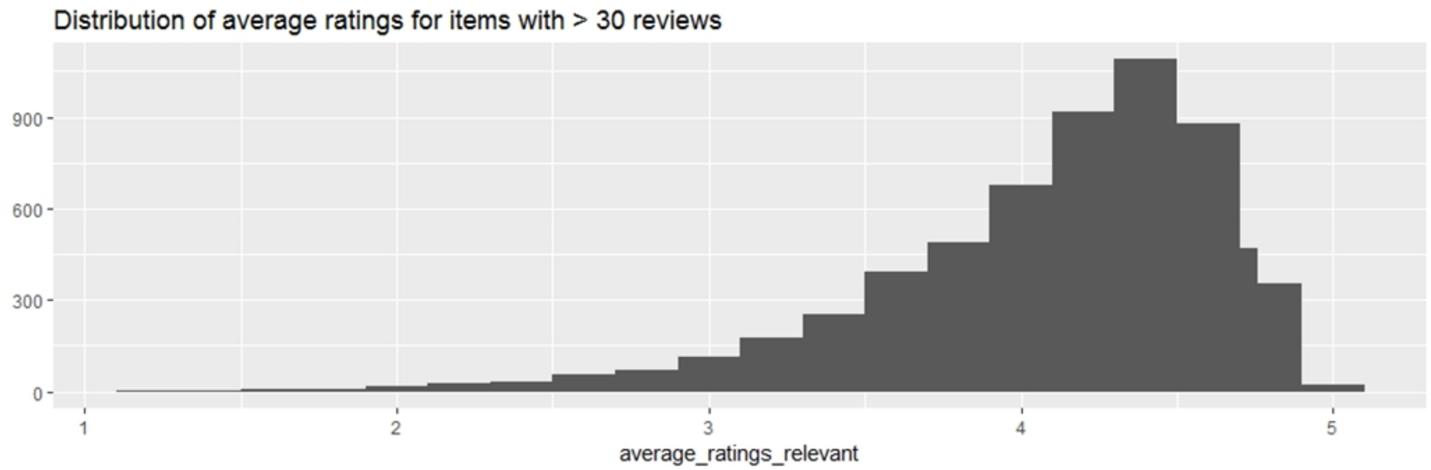
Top 10 Items and number of reviewers

	Item	reviews
5642	B000IDSLOG	3648
2997	B000BNQC58	2923
18268	B00295MQLU	2832
35871	B0052QYLUM	2830
9874	B000YDDF6O	2682
1026	B0000DEW8N	2458
6688	B000LXQVA4	2211
10535	B0011URFRE	2185
619	B000635WI	2085
15920	B001OC5UMQ	1928

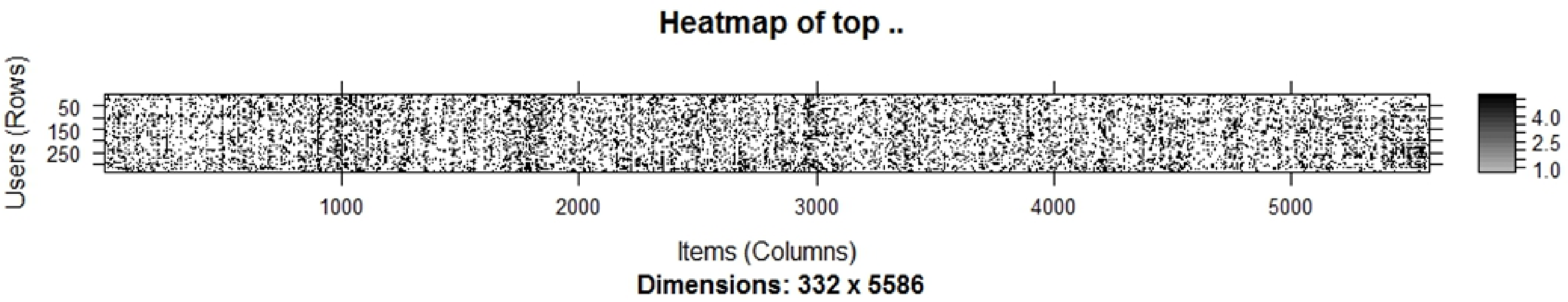


Average rating of items reviewed more than 30 times is also skewed to right, yet frequency of items with average rating 5 comes down

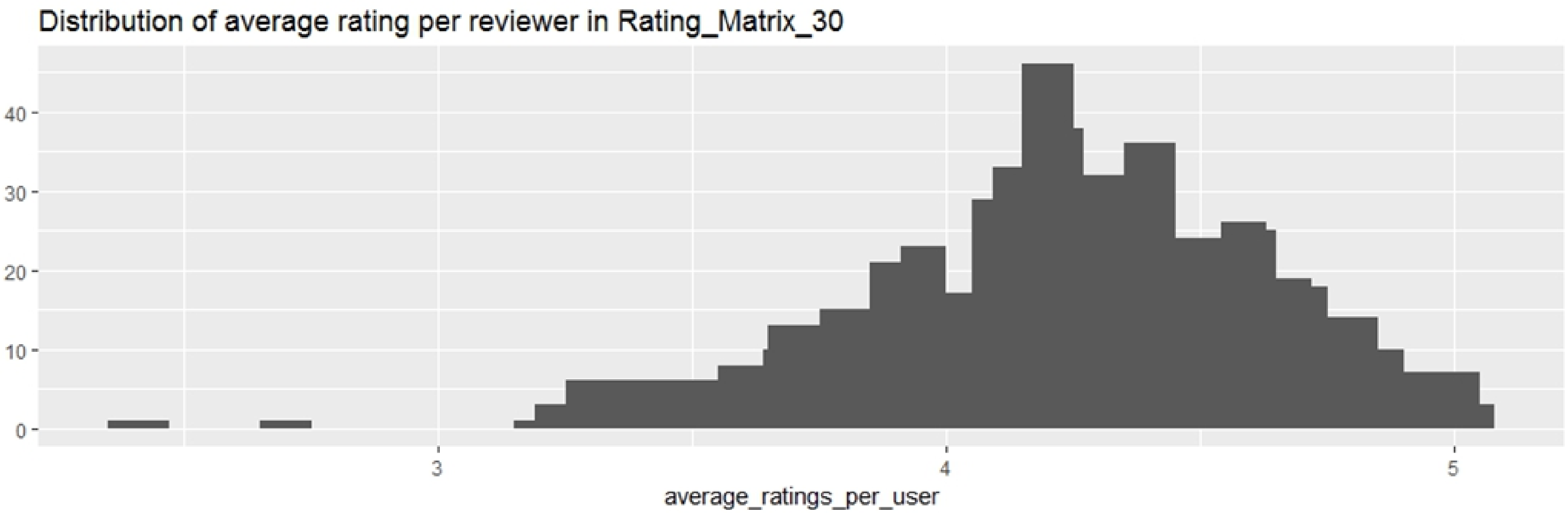




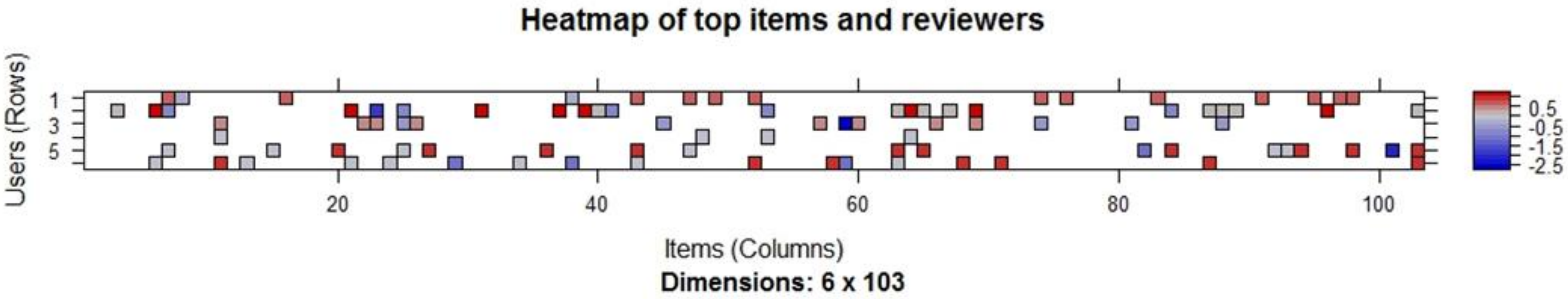
Heatmap of Top Items and Top Reviewers : not awfully sparse



Distribution of average rating for Matrix of Top Reviewers and Top Items : BIASED

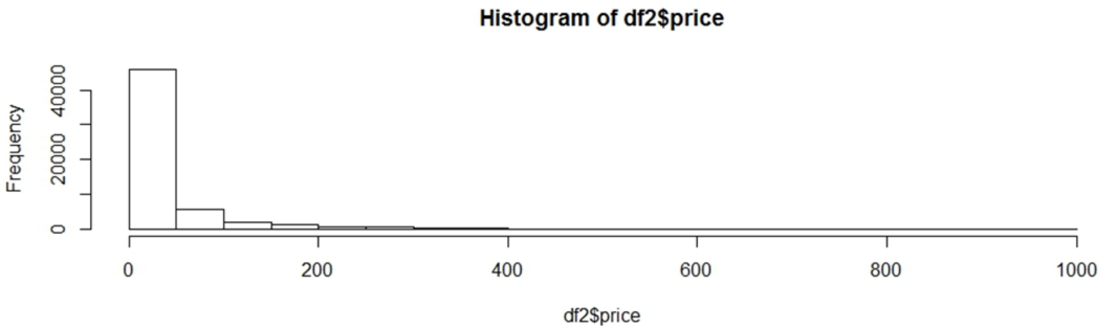


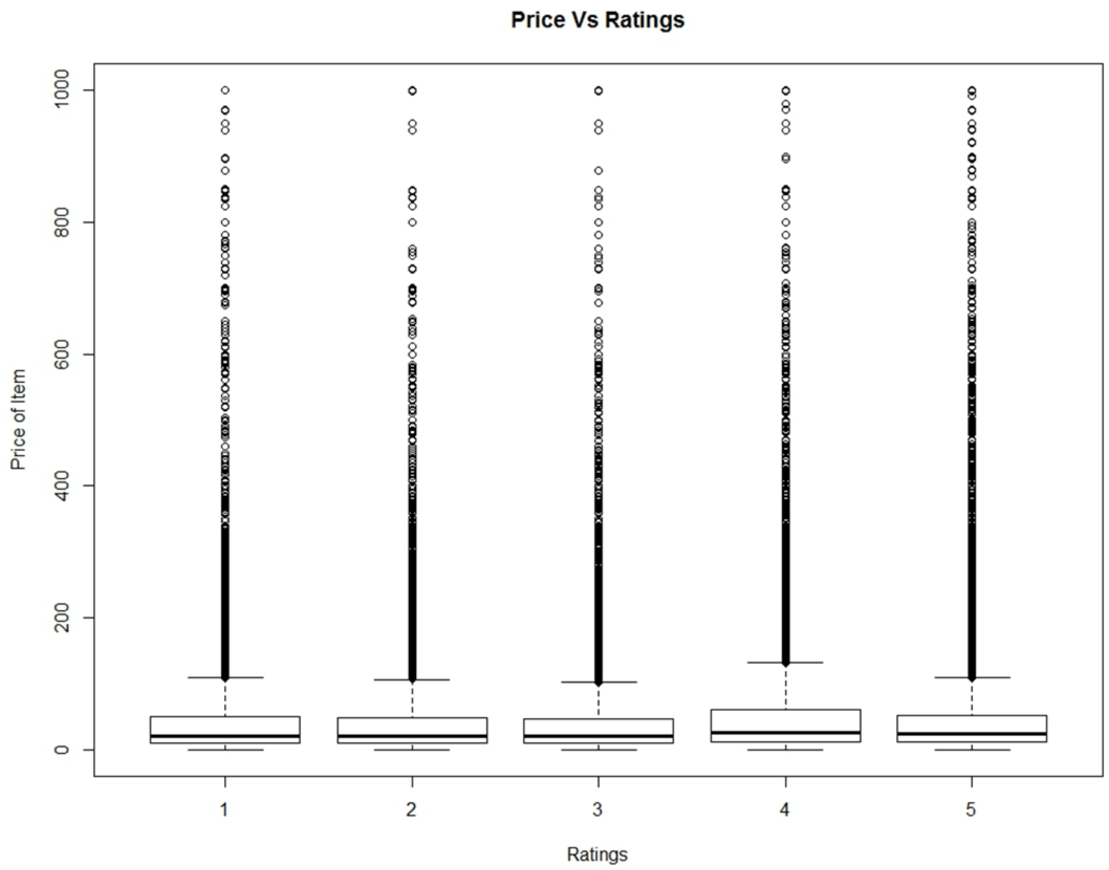
Heat map of top 2% of Normalized rating Matrix (normalization should take care of bias)



Price doesn't seem to be correlated with high ratings:

- There are 64426 items which are reviewed from 0-155 reviewers.
- Majority of items are priced under $50 with almost 99% under $100.
- Looking at distribution of price for unique 64426 items and ratings Vs price for all observations, we don't see any reason to explore if price is influencing ratings





amazon

MISSING DATA:

I was very keen on refining the model using content based filtering by category/sub-category/ brand name information but we don't have this information available for all items

```
> sapply(Review_Matrix, function(x) sum(is.na(x)))
      asin reviewerID reviewText    overall    summary      title      price      imUrl
         0          0        342          0          1       1493      51913       1491
```

ANALYSIS – DIRECTION OF STUDY

Recommendations should help a customer in finding new, relevant and interesting products

Three common approaches in developing recommendation systems are:

(1) Search based methods

(2) Cluster based methods

(3) Collaborative filtering methods

Collaborative filtering methods aim to predict the numerical rating that a user can give to a product. If we can do that, we will have a better understanding on the preferences and taste of this user when trying to cross-sell or up-sell

User-based collaborative filtering (UBCF):

1. Measure how similar each user is to the new one. Popular similarity measures are correlation and cosine.

$$sim(x,y) = \cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{||\vec{x}||_2 \times ||\vec{y}||_2} = \frac{\sum\limits_{s\in S_{xy}} r_{x,s} r_{y,s}}{\sqrt{\sum\limits_{s\in S_{xy}} r_{x,s}^2}\sqrt{\sum\limits_{s\in S_{xy}} r_{y,s}^2}},$$

2. Identify the most similar users. The options are:

- Take account of the top k users (k-nearest_neighbors)
- Take account of the users whose similarity is above a defined threshold

3. Rate the items purchased by the most similar users. The rating is the average rating among similar users and the approaches are:

- Average rating
- Weighted average rating, using the similarities as weights

4. Pick the top-rated items

$$\text{(a) } r_{c,s} = \frac{1}{N}\sum_{c'\in\hat{C}} r_{c',s},$$

$$\text{(b) } r_{c,s} = k\sum_{c'\in\hat{C}} sim(c,c') \times r_{c',s},$$

$$\text{(c) } r_{c,s} = \bar{r}_c + k\sum_{c'\in\hat{C}} sim(c,c') \times (r_{c',s} - \bar{r}_{c'}),$$

Item-based collaborative filtering (IBCF):

1. For a pair of two items, measure how similar they are in terms of having received similar ratings by similar users

$$sim(i,j) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{||\vec{i}||_2 * ||\vec{j}||_2}$$

2. For each item, identify the k-most similar items

3. For each user, identify the items that are most similar to the user's purchases

$$P_{u,i} = \frac{\sum_{\text{all similar items, N}} (s_{i,N} * R_{u,N})}{\sum_{\text{all similar items, N}} (|s_{i,N}|)}$$

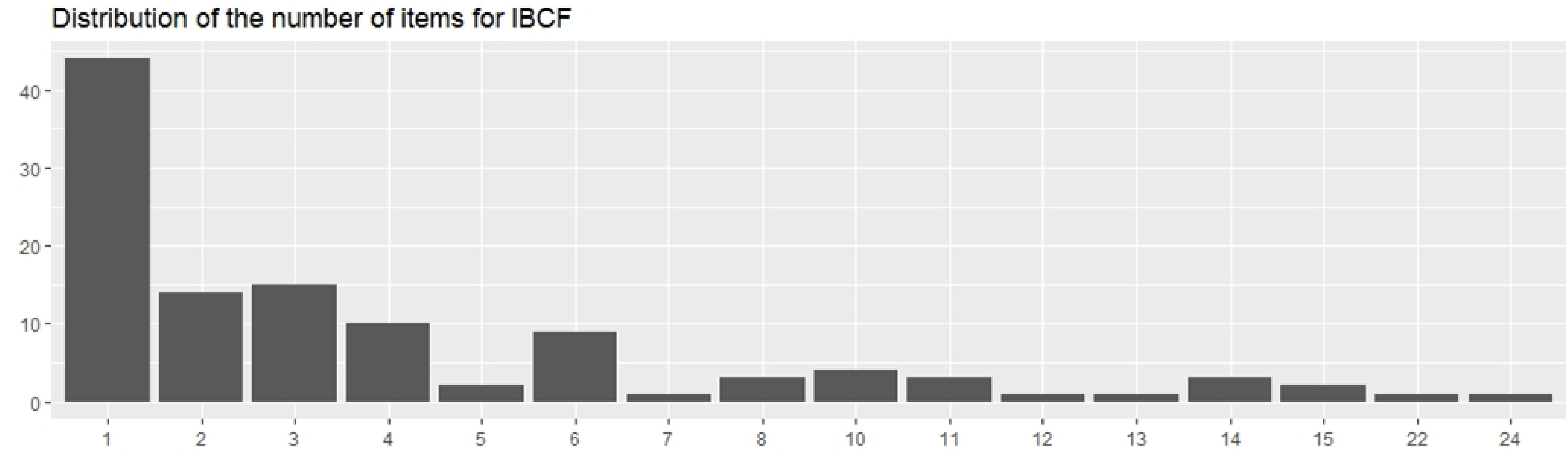
UBCF Vs IBCF

UBCF needs to access the initial data, so it is a lazy-learning model. Since it needs to keep the entire database in memory, it doesn't work well in the presence of a big rating matrix. Also, building the similarity matrix requires a lot of computing power and time.

IBCF recommends items on the basis of the similarity matrix. It's an eager-learning model, that is, once it's built, it doesn't need to access the initial data. For each item, the model stores the k-most similar, so the amount of information is small once the model is built. In addition, this algorithm is efficient and scalable, so it works well with big rating matrices.

IBCF

Distribution of most recommended items for 78 users (0 recommendations for one):



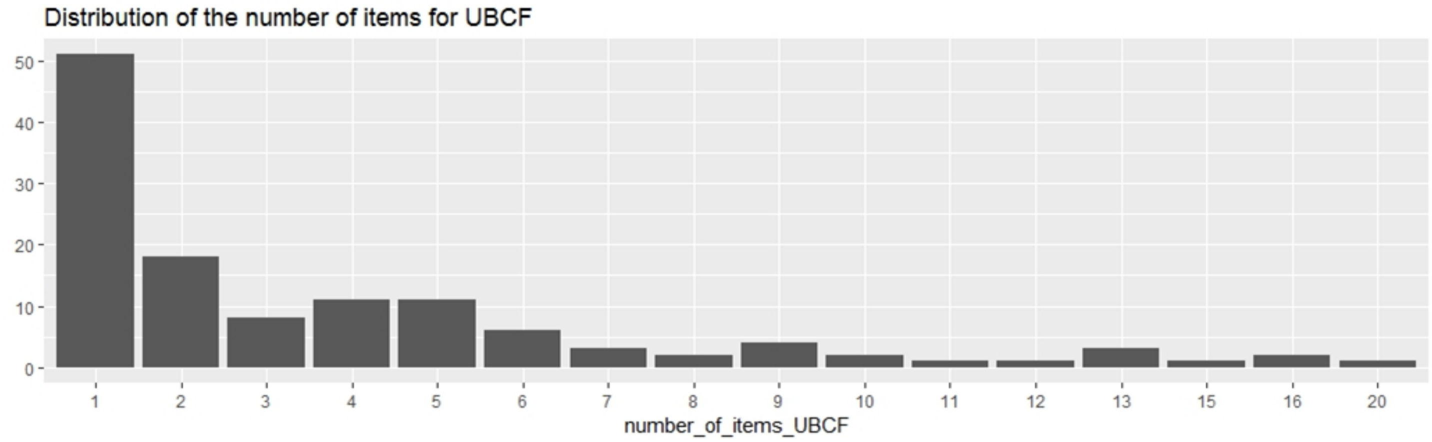
Few items recommended frequently; majority of items recommended 1-2 times

```
> table(number_of_items_sorted_IBCF)
number_of_items_sorted_IBCF
 1  2  3  4  5  6  7  8 10 11 12 13 14 15 22 24 
44 14 15 10  2  9  1  3  4  3  1  1  3  2  1  1 

ItemIDs of top 10 recommended items:
> table_top_IBCF[1:10,]
   names.number_of_items_top_IBCF. unname.number_of_items_top_IBCF.
1                       B000046S3W                               24
2                       B0000482FN                               22
3                       B00003XAKR                               15
4                       B000056C86                               15
5                       B00003XAKP                               14
6                       B00004C8S8                               14
7                       B00004W1UB                               14
8                       B000056J78                               13
9                       B000056HM5                               12
10                      9729375011                               11
```

UBCF

Distribution of most recommended items for 78 users (0 recommendations for one):

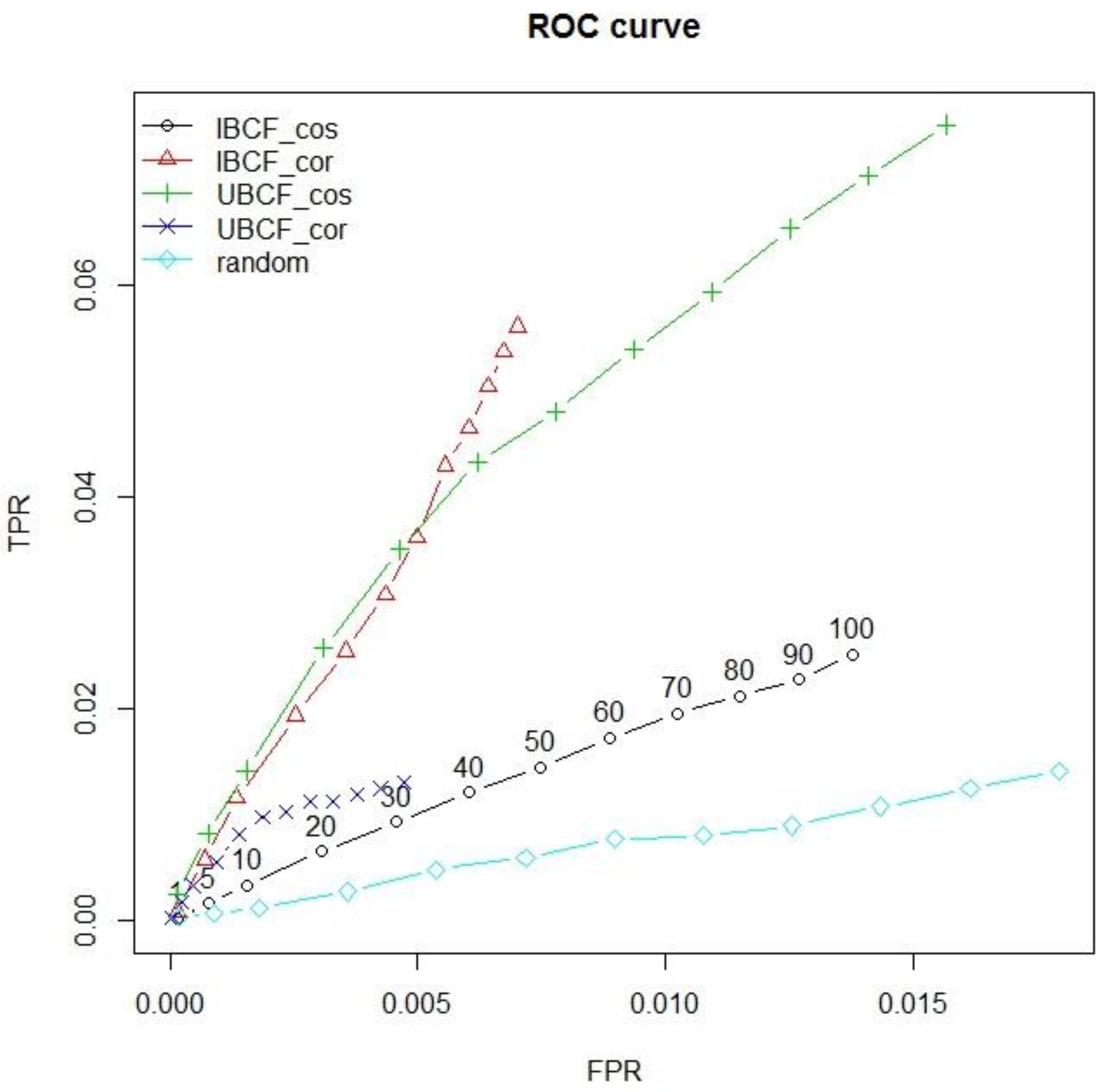



```
> table(number_of_items_UBCF)
number_of_items_UBCF
 1  2  3  4  5  6  7  8  9 10 11 12 13 15 16 20 
51 18  8 11 11  6  3  2  4  2  1  1  3  1  2  1 
```

```
> Freq_ItemsRecc_UBCF[1:10,]
   names.Top_Items_UBCF. unname.Top_Items_UBCF.
1             B000IDSLOG                     20
2             B000GJIE4E                     16
3             B000YDDF6O                     16
4             B001WAJVZM                     15
5             B00008KWO1                     13
6             B0037NXP18                     13
7             B0038JDUYI                     13
8             B000I2Q0F4                     12
9             B004DFO57M                     11
10            B001GQ2RW6                     10
```

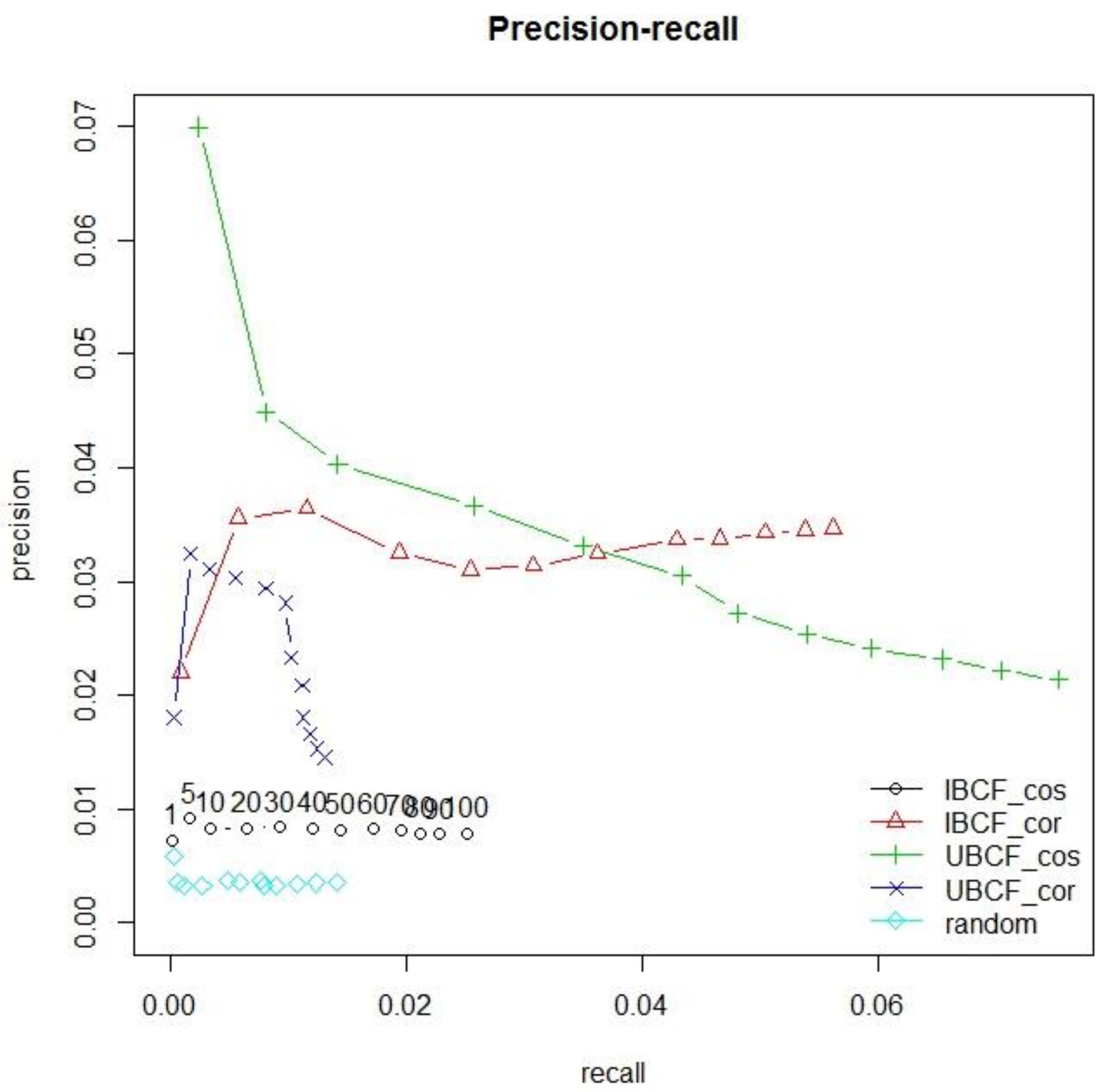
Model Selection:

ROC curve for all five models



Model Selection:

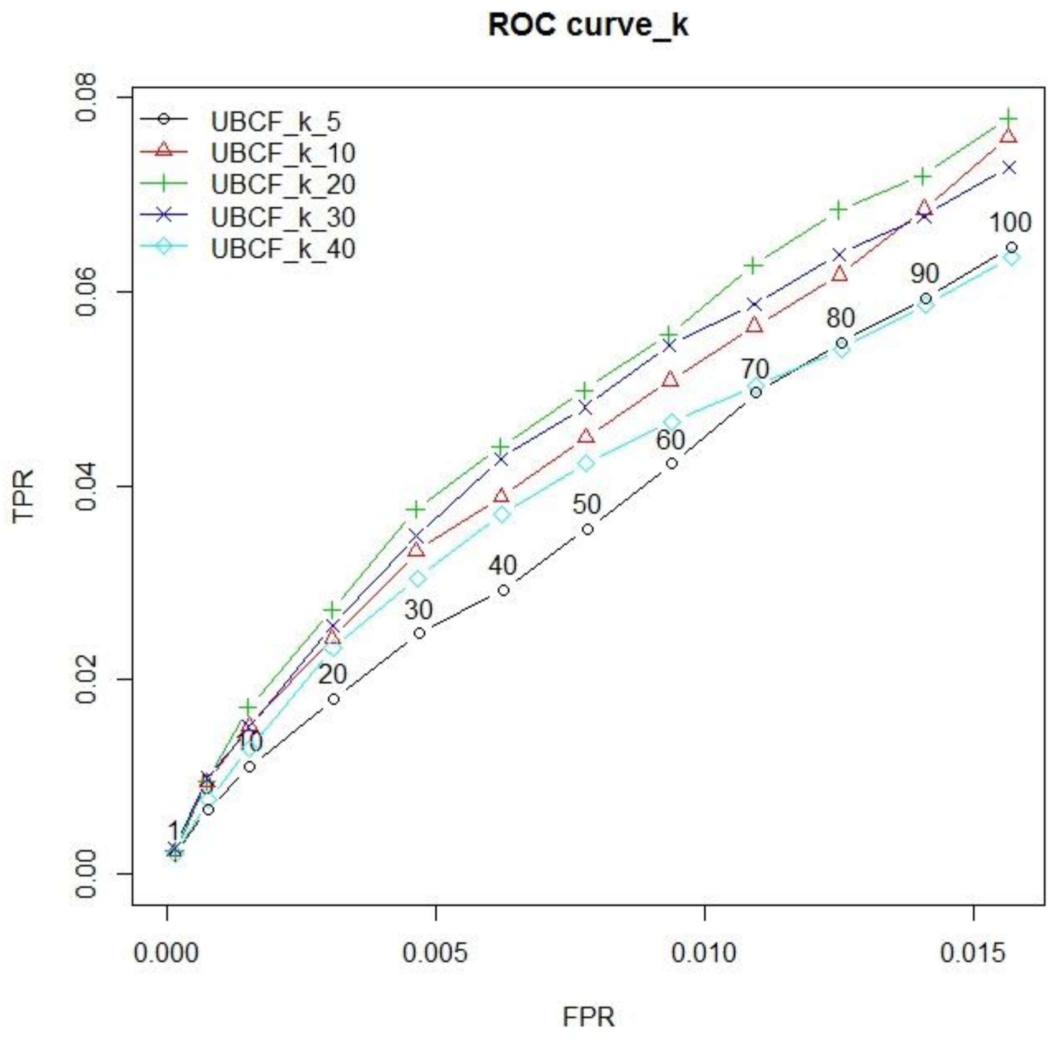
Precision- Recall curve for all five models



" User based collaborative filtering (Cosine similarity measure) seems to a winner so will go-ahead with parameter optimization for the same "

Parameter optimization:

ROC CURVE FOR NUMBER OF NEIGHBORS 5, 10, 20, 30, 40:



Parameter optimization:

PRECISION – RECALL CURVE FOR NUMBER OF NEIGHBORS 5, 10, 20, 30, 40:

“ UBCF Cosine for number of neighbors = 20 is best model ”

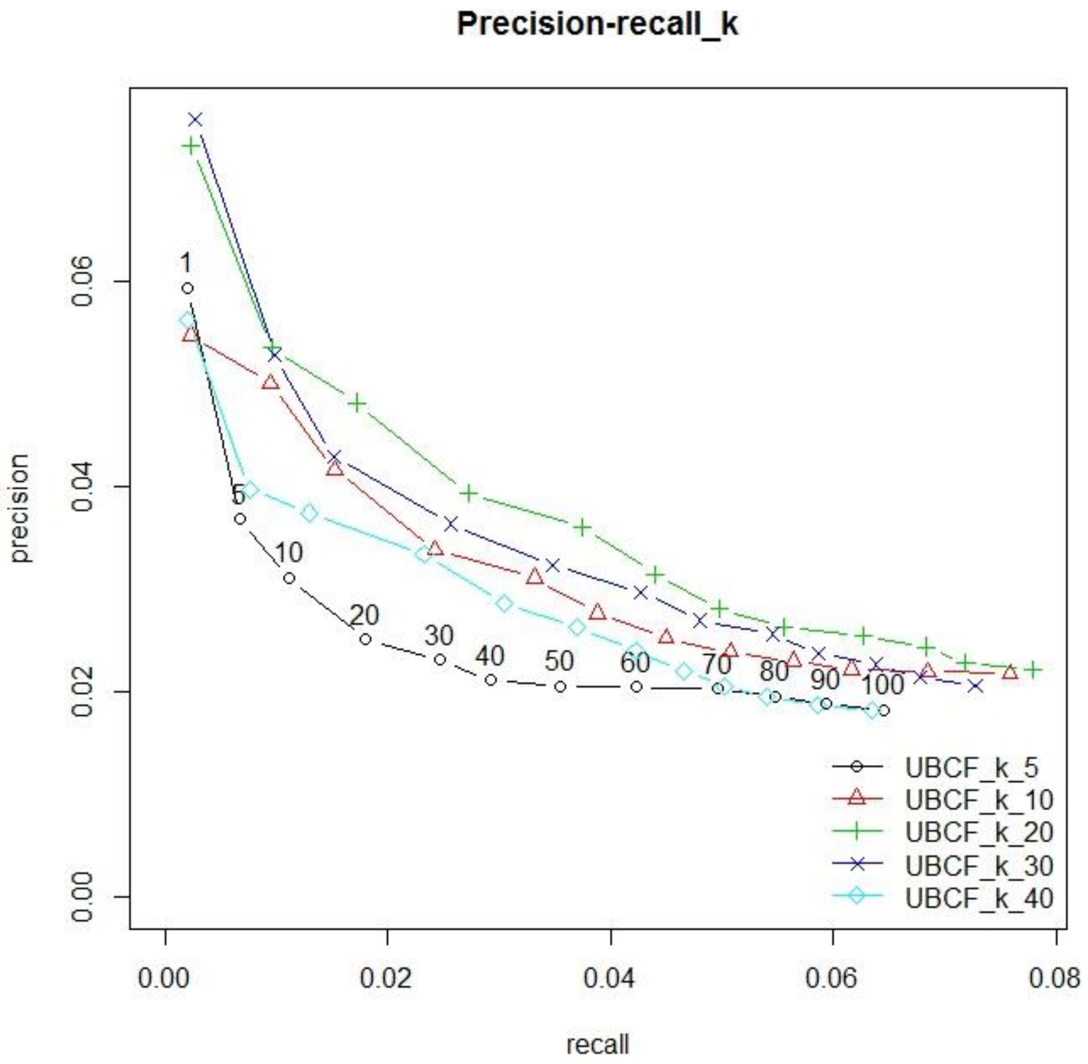


TABLEAU VISUALIZATION OF UBCF RECOMMENDATIONS

VIZ_ UBCF RECOMMENDATIONS FOR TEST SET

Conclusion

1. Both IBCF and UBCF have their limitations and can successfully cater to only frequent buyers: COLD START PROBLEM with new users and new items.

Take in to account only rating matrices – contextual information makes model more intelligent

So, can never replace search based or knowledge based models, however both develop new and engaging recommendations and have their own use cases. I am especially impressed that none of the models favor ‘Majority Rule’ so are anyway better than data aggregation methods.

Conclusion

2. There is lot of support for IBCF especially because User based filtering is not scalable and thus can not be applied to big data effectively yet logically User based Collaborative filtering makes better sense

In my application, got best results for UBCF
(because of nature of my data)

THANK YOU