RECOMMENDATION SYSTEM

PREPARED BY
Ravi Shankar

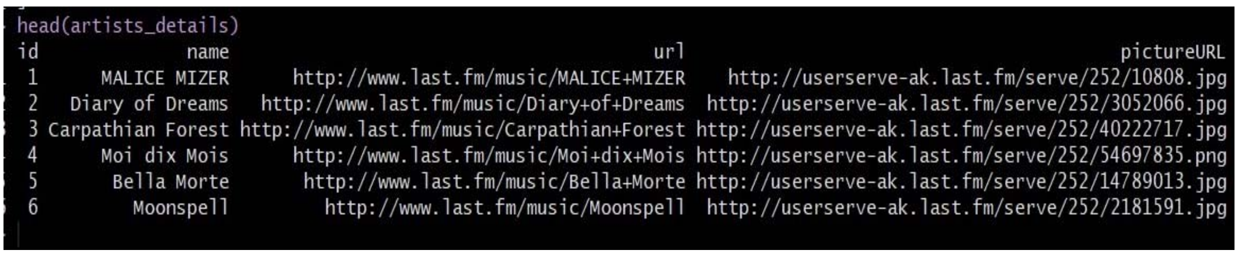
INTRODUCTION

Data: Music streaming service LastFM open source data in creating and evaluating different content based and collaborative filtering recommendations systems. Four different data files are provided for the purpose.

READING DATASETS AND MANIPULATIONS

1. **Artists.dat**

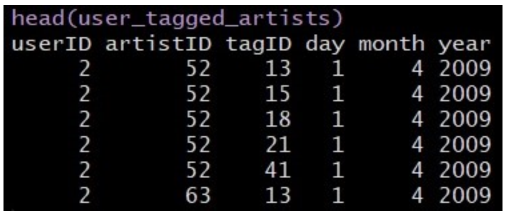
This data file contains details information about various music artist. It has four columns. There is a total of 14036 unique artists. Input into dataset name artist_details in R. Table has details about 14036 artist and details Column is 4.



```
head(artists_details)
 id              name                                      url                                            pictureURL
  1      MALICE MIZER      http://www.last.fm/music/MALICE+MIZER    http://userserve-ak.last.fm/serve/252/10808.jpg
  2   Diary of Dreams   http://www.last.fm/music/Diary+of+Dreams  http://userserve-ak.last.fm/serve/252/3052066.jpg
  3 Carpathian Forest http://www.last.fm/music/Carpathian+Forest http://userserve-ak.last.fm/serve/252/40222717.jpg
  4      Moi dix Mois      http://www.last.fm/music/Moi+dix+Mois http://userserve-ak.last.fm/serve/252/54697835.png
  5       Bella Morte       http://www.last.fm/music/Bella+Morte http://userserve-ak.last.fm/serve/252/14789013.jpg
  6         Moonspell         http://www.last.fm/music/Moonspell  http://userserve-ak.last.fm/serve/252/2181591.jpg
```

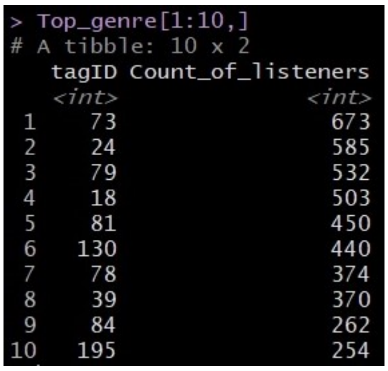
2. **User_tagged_artists.dat**

This data file contains the tag assignments of artists provided by user /listeners. They also contain the timestamps when the tag assignment was done. It has 6 columns. There is a total of 186479 observation of the 1892 users tagging artist and the timestamp of when it was done. Only 12523 unique artists have been tagged. A total of 9749 music genre was tagged. TagId shows genre type of the artist given by user. Artist has multiple tagId given by various users.



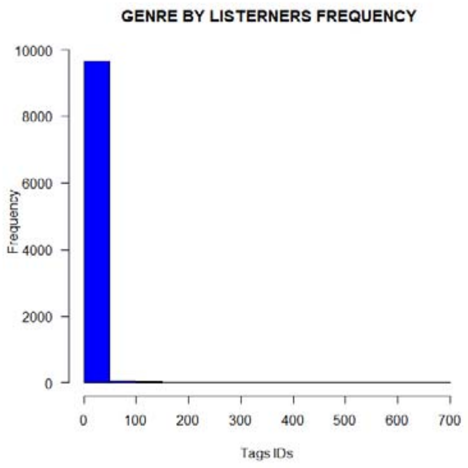
```
head(user_tagged_artists)
userID artistID tagID day month year
     2       52    13   1     4 2009
     2       52    15   1     4 2009
     2       52    18   1     4 2009
     2       52    21   1     4 2009
     2       52    41   1     4 2009
     2       63    13   1     4 2009
```

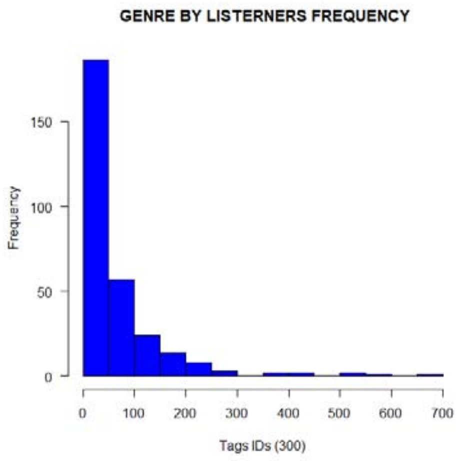
Extracting details about most used tag by grouping tagId and summarising into count of listeners. This set is termed as Top_genre.



```
> Top_genre[1:10,]
# A tibble: 10 x 2
   tagID Count_of_listeners
   <int>              <int>
 1    73                673
 2    24                585
 3    79                532
 4    18                503
 5    81                450
 6   130                440
 7    78                374
 8    39                370
 9    84                262
10   195                254
```

Top genre is highly skewed. Only few genre are listened by users a lot and others are heard very rarely. We have to remove some skewness by selecting top 300 tagID (genre) from 9749 tagID.

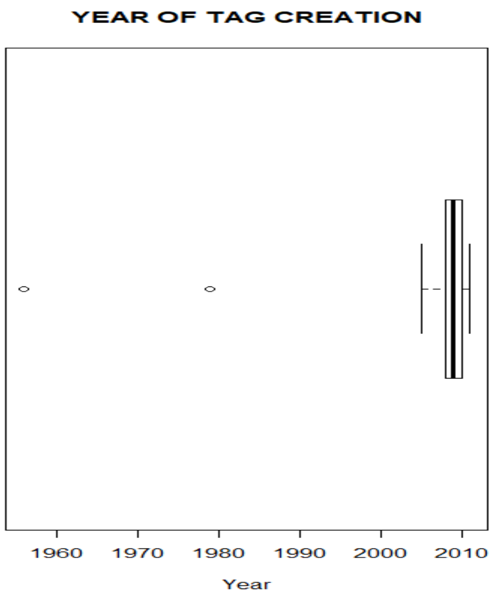




Before: Frequency of listening genre data

After: Selecting top 300 Genre from the data

Checking for outliers by year of tagging by the users. few observations are found as outliers. So, we have to delete these observations from the dataset. we have to select tagging year more than year 2000.

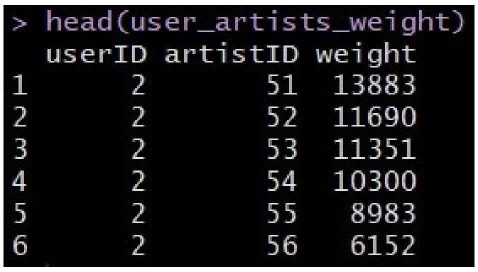


After deleting outliers and selecting top 300 genre, we have to subset main data user_tagged_artists. Final user_tagged_artrist dataset has dimension of

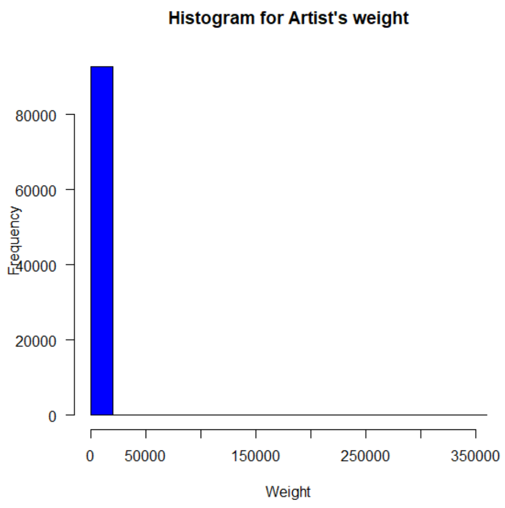
	Number of Rows	Number of Columns
Original dataset	186479	6
Manipulated final dataset	139362	6

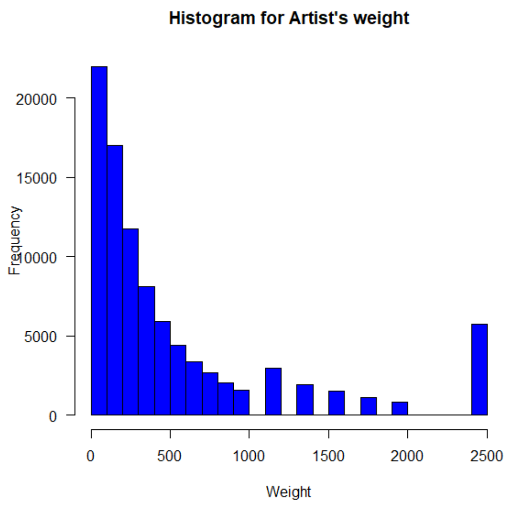
3. User_artists.dat

This data file contains the artists listened by each user. It also provides listening count for each artist by users as a pair. It has 3 columns. There is a total of 92834 observations from 1892 different users, ranking 17632 different artists. The weight variable is the listening count for a user for an artist.



```
> head(user_artists_weight)
  userID artistID weight
1      2       51  13883
2      2       52  11690
3      2       53  11351
4      2       54  10300
5      2       55   8983
6      2       56   6152
```





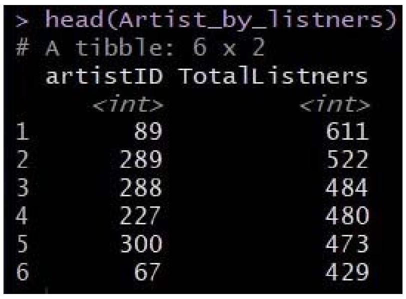
Before: Weight of artists in distributed format **After:** Weight of Artists categorised in Bands

If we analyse weight provided by users, we will find that's its highly distributed and skewed. So, to decrease the skewness, I will distribute weight in various weight bands. After converting into weight bands. The skewness if weight got highly reduced.

Weight Value Range	Weight Band
1 - 50	50
51 - 100	100
101 - 150	150
151 - 200	200
201 – 250	250
251 – 300	300
300 – 350	350
351 – 400	400
401 – 450	450
451 – 500	500
501 – 600	600
600 – 700	700
701 – 800	800
801 – 900	900
901 – 1000	1000

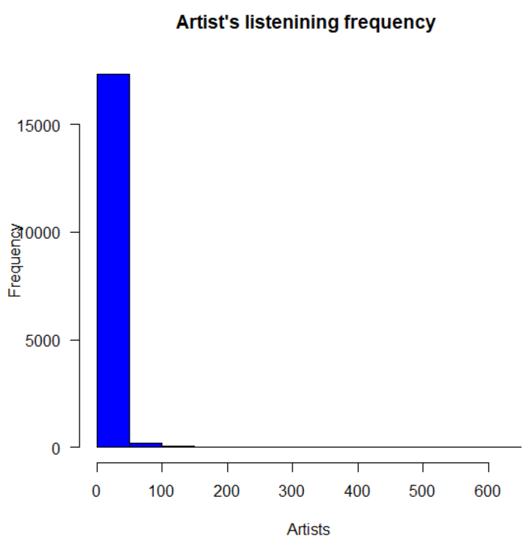
1001 – 1200	1200
1201 – 1400	1400
1401 – 1600	1600
1601 – 1800	1800
1801 – 2000	2000
2000 -	2500

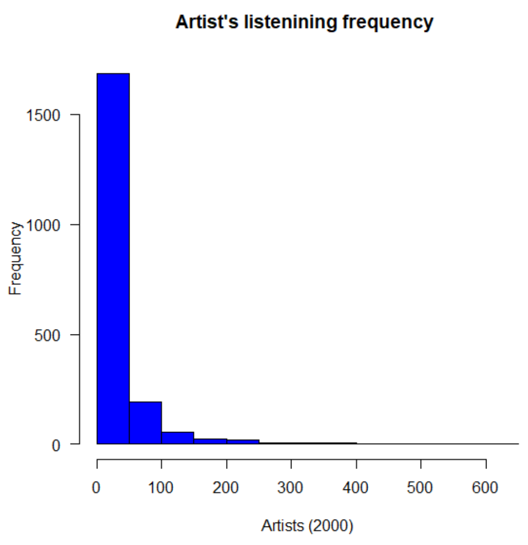
We can also visualize the artists data in respect of how many users listen this artist. We can make table by taking count of users grouped by artist. Which will give top artist popular or listen by most of the users.



```
> head(Artist_by_listners)
# A tibble: 6 x 2
  artistID TotalListners
     <int>         <int>
1       89           611
2      289           522
3      288           484
4      227           480
5      300           473
6       67           429
```

Data of artists by number of user / listeners is highly skewed. That means most of the users listen to only certain groups of artists. So, to decrease the skewness we have to decrease the number of artists. We will select top 3000 artists out of 17632 artists.





Before: Its quite skewed. Only few artists are heard by most of the user

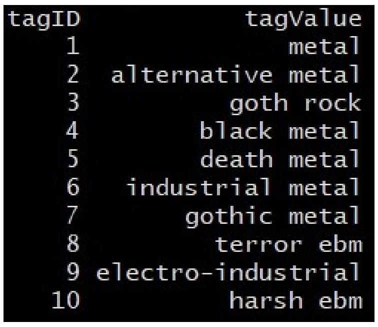
After: Skewness got little reduced after selecting top 2000 artists from data.

After selecting 2000 top listen artist from the data decreases some skewness.Now we have to subset parent data set user_artist_weight dataset according to the top 2000 artists. Final dimension of user_artist_weight dataset

	Number of Rows	Number of Columns
Original dataset	92834	3
Manipulated final dataset	66784	3

Tags.dat

This data file contains a set of tags available in the dataset. Each tag represents a music genre.it has 2 columns. There is a total of 11946 unique music genre tags available.



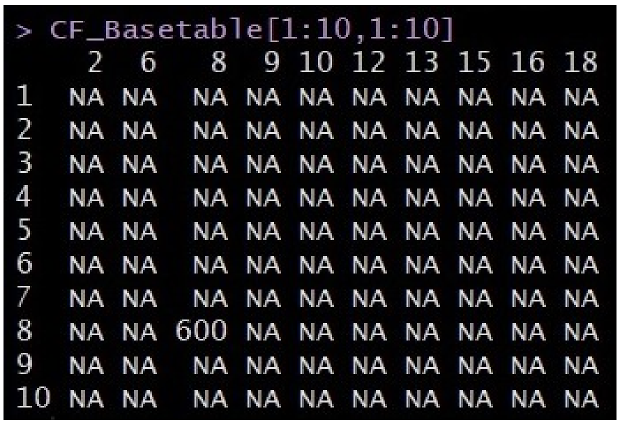
```
tagID          tagValue
    1             metal
    2  alternative metal
    3         goth rock
    4       black metal
    5       death metal
    6   industrial metal
    7      gothic metal
    8        terror ebm
    9 electro-industrial
   10         harsh ebm
```

CREATION OF BASETABLE / MATRIXES

Collaborative Filtering Basetable / Matrixes

Spread user_artists_weight datset and create user per row data table. Also delete column (artist) which has not given weight by at least 50 users. Subset with user_tagged_artists data sets such that that both tables has same set of artists and give final name of basetable as CF_Basetable. Final Dimension of CF_Basetable

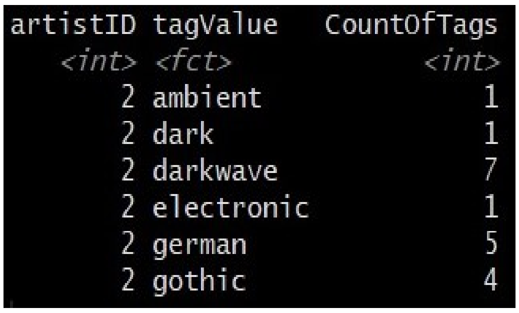
	Number of Rows	Number of Columns
CF_Basetable	1877	1665



```
> CF_Basetable[1:10,1:10]
    2  6   8  9 10 12 13 15 16 18
1  NA NA  NA NA NA NA NA NA NA NA
2  NA NA  NA NA NA NA NA NA NA NA
3  NA NA  NA NA NA NA NA NA NA NA
4  NA NA  NA NA NA NA NA NA NA NA
5  NA NA  NA NA NA NA NA NA NA NA
6  NA NA  NA NA NA NA NA NA NA NA
7  NA NA  NA NA NA NA NA NA NA NA
8  NA NA 600 NA NA NA NA NA NA NA
9  NA NA  NA NA NA NA NA NA NA NA
10 NA NA  NA NA NA NA NA NA NA NA
```

Content based Basetable / Matrixes

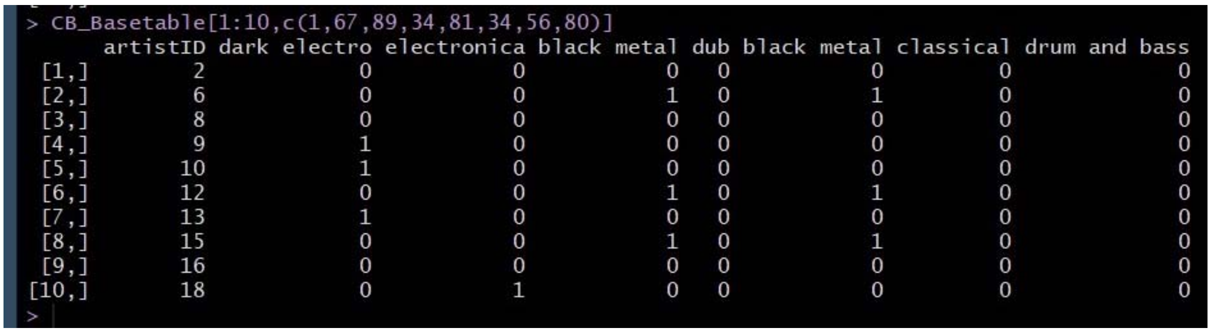
After subseting user_tagged_artist dataset such that CF_Basetable and user_tagged_artists have same set of artists. Merge user_tagged_artist with tags and call it tagged_merged so that we also get tagId and tag value in same table. Chose only artist and tag value from this tagged_merged table. Now group by artistId then Tagvalue and create summarise variable name CountOfTags.



```
artistID tagValue   CountOfTags
   <int> <fct>            <int>
       2 ambient              1
       2 dark                 1
       2 darkwave             7
       2 electronic           1
       2 german               5
       2 gothic               4
```

Now spread this table. It will give artistID per row, tagValue as variable and CountOfTag as value .Covert it into matrix and name it as CB_BAaetable. Final dimension of Content based basetable matrix are mentioned in table.

	Number of Rows	Number of Columns
CB_Basetable	1665	298



```
> CB_Basetable[1:10,c(1,67,89,34,81,34,56,80)]
      artistID dark electro electronica black metal dub black metal classical drum and bass
 [1,]        2    0           0           0   0           0         0             0
 [2,]        6    0           0           1   0           1         0             0
 [3,]        8    0           0           0   0           0         0             0
 [4,]        9    1           0           0   0           0         0             0
 [5,]       10    1           0           0   0           0         0             0
 [6,]       12    0           0           1   0           1         0             0
 [7,]       13    1           0           0   0           0         0             0
 [8,]       15    0           0           1   0           1         0             0
 [9,]       16    0           0           0   0           0         0             0
[10,]       18    0           1           0   0           0         0             0
>
```

FILTERING FUNCTIONS CREATION

User-Based & Item-Based collaborative filtering function

In this section deals with hand written code for user-based filtering function and Item-Based filtering function. User-based and Item-based Collaborative Filtering. Both are similar in structure.

Similarity Computation

The basic idea of similarity computation is co-rating between the pairs. For user-based collaborative filtering, similarity between user *i* and user *j* are computed using the items pairs which have been rated by both users. Formula for user-based, Pearson correlation between user *u* and user *v* is written below:

$$w_{u,v} = \frac{\sum_{i\in I}(r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i\in I}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{i\in I}(r_{v,i} - \bar{r}_v)^2}}$$

Where *I* is the set of items rated by both user *I* and user *j* and user *j*, $r_{u,i}$ is the rating of user *u* on item *I* and $\bar{r}_i$ is the average rating of user *i.*

For item-based collaborative filtering, similarity between item *I* and item *j* are computed by working on the users who have rated both items. Formula for item-based, Pearson correlation between user *i* and user j is written below:

$$w_{i,j} = \frac{\sum_{u\in U}(r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u\in U}(r_{u,i} - \bar{r}_i)^2}\sqrt{\sum_{u\in U}(r_{u,j} - \bar{r}_j)^2}}$$

Where U is the set of users who have rated both item *I* and item j, $r_{u,i}$ is the rating of user u on item *I* and $\bar{r}_i$ is the average rating of item *i.*

Neighbourhood selection

After the similarity computation, Function has to select the most similar user for the similarity table of user. This function selects the top k nearest-neighbours who have rated the given item. This strategy in selecting top k nearest users It might exclude high similar neighbours just because they have not rated the given items.

Generating Recommendations

In user-based collaborative filtering predictions are generated based in a weighted aggregate of their ratings. To make the prediction for the selected user a on an item *I,* weighted sum is computed using all the ratings of the selected neighbours on that item by formula:

$$P_{a,i} = \bar{r}_a + \frac{\sum_{u\in U}(r_{u,i} - \bar{r}_u)w_{a,u}}{\sum_{u\in U}||w_{a,u}||}$$

Where $\bar{r}_a$ and $\bar{r}_u$ are the average rating of user a and user *u,* $w_{a,u}$ is the weight between user a and user u. The summations are over all the user $u \in U$ who have rated item *i.*

Cluster-Based Collaborative Filtering

The cluster-based collaborative filtering function is based on the K-means algorithm. This algorithm tries to cluster the different observations into k clusters in which each observation belongs to the cluster with the nearest arithmetic mean.

Content Based Collaborative Filter

Content based collaborative filtering is based on recommending items which come from mixed factorization of User-Item matrix and Item-content matrix. Item-content matrix contains genre tagging done by users of the respective item. Similarity computation is done on Cosine method.

Item-based and User-based Hybrid filter

Item-based and user-based hybrid recommendation filter combines results of parent filters by mean method and utilize it for recommendation.

Content-based and Item-based Hybrid Filter

Content-based and Item-based hybrid recommendation filter combines results of parent filters by mean method and utilize it for recommendation.

EVALUATION FUNCTIONS CREATION

MAE (Prediction measure)

MAE is prediction measure for calculating the deviation of recommendations from their true user-specified values. For each ratings-prediction pair $< p_i, q_i >$ metric takes the absolute error between them, i.e., $|p_i - q_i|$. The MAE is computed by first summing these absolute errors of the N corresponding ratings-prediction pairs and them computing the average.

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N}$$

The lower the MAE, the more accurate the recommendation predicts.

F1 (Classification measure)
The F1 is the harmonic mean of precision and recall and are calculated as follows:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Function first calculate the true positive, false positive and false negative based on a selected threshold (topN). Then used to calculate the recall and precision values

$$Recall = \frac{true\ positive}{true\ positive + false\ positive}$$

$$Precision = \frac{true\ positive}{true\ positive + false\ positive}$$

MODELS & RESULTS

In this section compares results from all used techniques and their performance and discuss the advantages and disadvantages of the model for LastFM to use.

Split Train and Test

Train data percentage	Test Data Percentage
80	20

Models Executed
These are the model and recommendation filters are executed : User Based Collaborative Filtering with correlation similarity, Item Based Collaborative Filtering with correlation similarity, Content Based Model with Cosine Similarity, Cluster Based Model – By user defined function, Item-based & User-based Hybrid and Model – Item-based & Content-based Hybrid

Results

	User-based	**Item-based**	**Content-based**	**Item-based & User based Hybrid**	**Item-based & Content-based**
MAE	4.306621	4.185333	3.888825	4.210998	4.057173
Recall	0.5653986	0.4454048	0.4591985	0.6034574	0.6036338
Precision	1	1	1	1	1
F1	0.7223701	0.6163046	0.6293846	0.7526953	0.7528325
RMSE	67.748	68.44601	65.19019	67.59526	66.85149
Run-time	long runtime but not more than Item-based	Very long time	Not that Longer	Longer:depends on other models	Longer:depends on other models

Pros, corns and solutions

Content based filter recommendation system is best because this outperforms other models in many provided tests.

Pros: Comparison between items possible, no meta data engineering is required & it is adaptive.

Drawback: over specialised and need to have content details and genre tagging for the item.

Solution: Encourage users to tag items by awarding them with perks and privileges.