ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Professor Peng Wang for giving me an opportunity to work on this project. His valuable guidance and constant support have been of immense help to me towards the completion of this project. I am very grateful to him for his profound advices and suggestions at each step throughout the course. I look forward to applying my learnings from the capstone project in my career.

ABSTRACT

Streaming music have become one of the top sources of entertainment for millennials. Because of globalization people all around the world are now able to access different kinds of music. The global recorded music industry is worth $15.7 billion and is growing at 6% as per 2016. Digital music is responsible for driving 50% of those sales. There are 112 million paid subscribers for the streaming business and roughly a total of 250 million users, if we include those who don't pay. Thus, it becomes very important for streaming service providers like Youtube, Spotify and Pandora to continuously improve their service to the users.

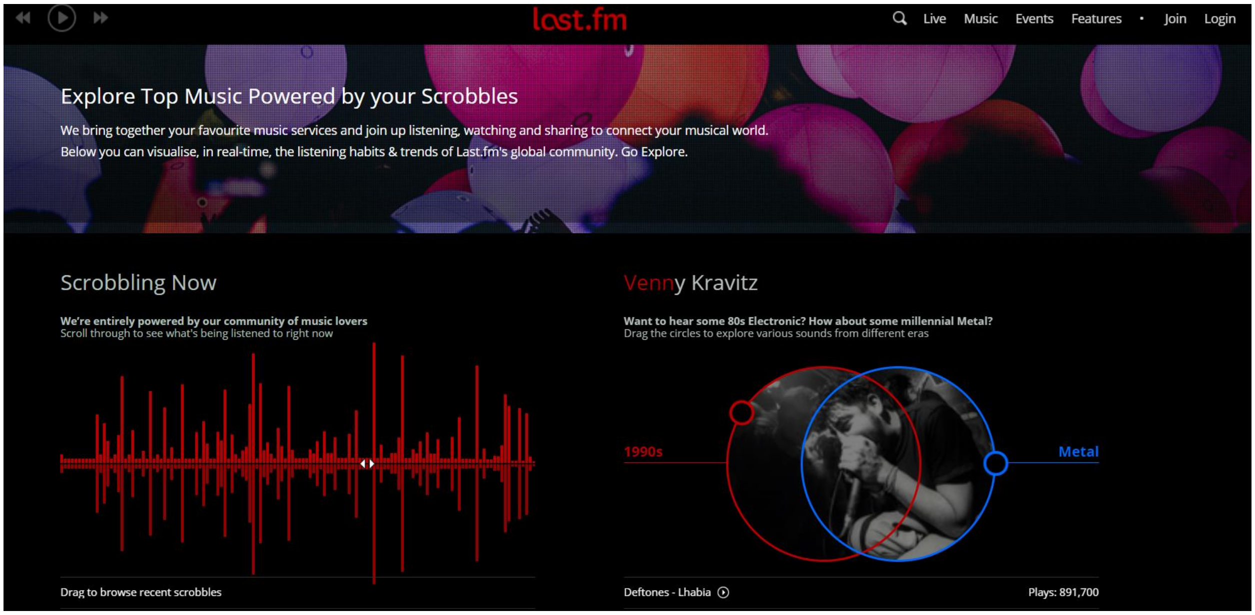
Recommendation Systems are one such information retrieval technique to predict the ratings or popularity a user would give/have for an item. In this project I would be exploring bunch of methods to predict ratings of users for different artists using GroupLen's Last.FM dataset.

CONTENT

TOPICS **PAGE**

ABOUT DATA

The data for this project uses records of play counts of an artist by various users from the Last.Fm website. Below is a screenshot of the website.



Explore Top Music Powered by your Scrobbles

We bring together your favourite music services and join up listening, watching and sharing to connect your musical world.
Below you can visualise, in real-time, the listening habits & trends of Last.fm's global community. Go Explore.

Scrobbling Now

We're entirely powered by our community of music lovers
Scroll through to see what's being listened to right now

Drag to browse recent scrobbles

Venny Kravitz

Want to hear some 80s Electronic? How about some millennial Metal?
Drag the circles to explore various sounds from different eras

Deftones - Lhabia

Plays: 891,700

The data for the project is available free to use for non-commercial purposes here by GroupLens. GroupLens is a research lab at the University of Minnesota which publish research articles in conferences and journals primarily in the field of computer science, but also in other fields including psychology, sociology, and medicine. The owners constantly update the dataset in the page. At the time of writing this report this dataset contained social networking, tagging, and music artist listening information from a set of 2K users from Last.FM online music streaming site. The zip file contains namely the following files:

artists.dat	Contains information about music artists listened and tagged by the users
tags.dat	Contains the set of tags available in the dataset
users_artists.dat	Contains information on listening counts of an artist by a user
user_friends.dat	Contains information on bi-directional user-friend relation
user_taggedartists.dat	Contains information on tag assignments of an artists provided by each particular user
user_taggedartists-timestamp.dat	Contains above information which the timestamp

Some detailed information about the data:

1. Information about 17632 artists tagged by 1892 users
2. An artist can be tagged differently by a different user
3. Weights in user_artists dataset is the listening count of an artist by a user

All the datasets would be used for making a recommendation system except user_taggedartists-timestamp.dat because it doesn't provide any new useful information. The number of time a user listened an artist will be used as a rating (or proxy for rating). Below is a screenshot of the Last.FM website.

LITERATURE SURVEY

According to various blogs read, it is found that we can implement recommendation system in many ways: each with a different algorithm running them. One of the most popular technique is Collaborative Filtering. It can be done performed in two ways:

a. ***User Based Filtering***: Where similar users are first found using their similar decisions in the past and then their items are recommended to each other. The rating prediction is done based on calculating the weighted average of user ratings and their similarity to the user in question. For calculating the similarity, the most popular choice is: centered-cosine-distance which easily differentiates a tough and an easy user.

b. ***Content Based Filtering****:* Where if a user decides on an item, he/she is recommended the next item based on the item which is like the item he just decided upon. Here the predicted rating is calculated in a similar fashion as in user-based filtering.

Other important and popular technique is a model-based algorithm called Matrix Factorization. There are several variants of it. Singular Value Decomposition or SVD is one of them.

In general, it has been reported that for many cases Content Based Filtering (also known as item-item) outperforms User Based Filtering (or user-user). And the reason is very interesting. Items are "simpler" than users in the sense that items can have only certain genres, but a user's taste can vary largely. For example, an artist can belong to metal rock but very unlikely that he/she would also belong to classical opera music. Whereas users can like metal rock and can listen to classical opera music occasionally. But this again highly depends on the data, and for the Last.FM data, it was found that the User-Based Collaborative Filtering outperformed all other methods.

Other possible methods were also explored like Popularity-Based Collaborative Filtering but were found to be not very useful in this case along with SVD in this case.

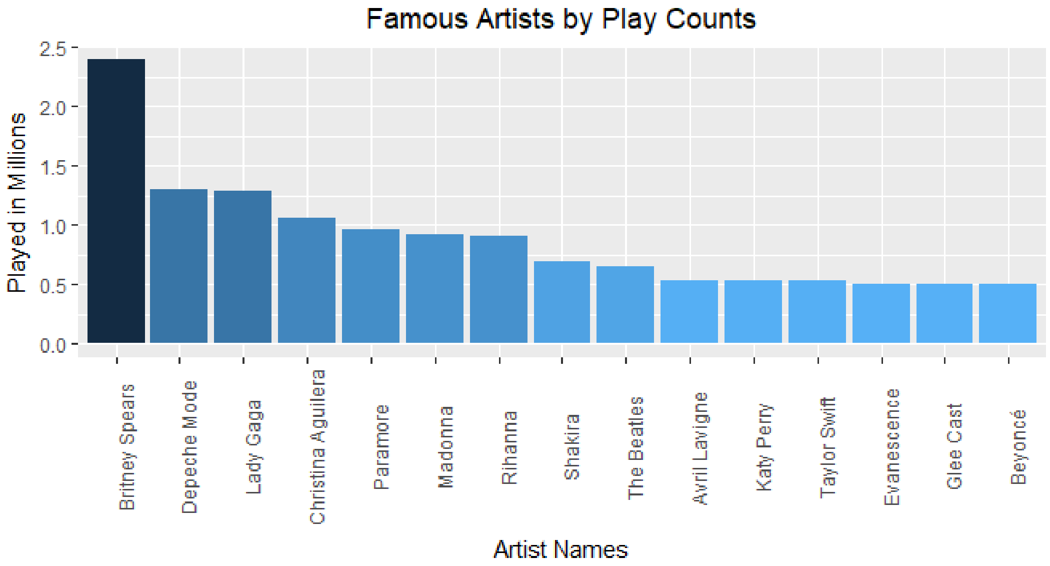
DATA CLEANING

Out of 6 datasets, the dataset with information on artists was found to be a little messy. There were bunch of 636 artist names which were not in plain English. Also, most of their play counts were less. Few names used foreign language characters and hence were obscure. So, to keep the future results understandable those artist names were removed from further analysis.

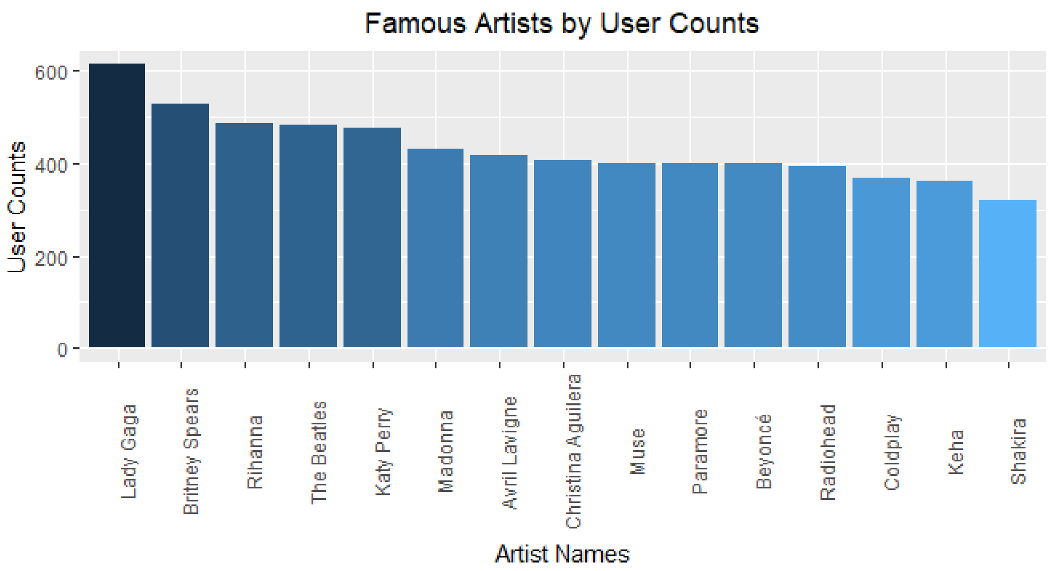
DATA EXPLORATION

Next, the data was explored to identify most played artists and artists who were played by most number of unique users in the last.fm. Following were observations:

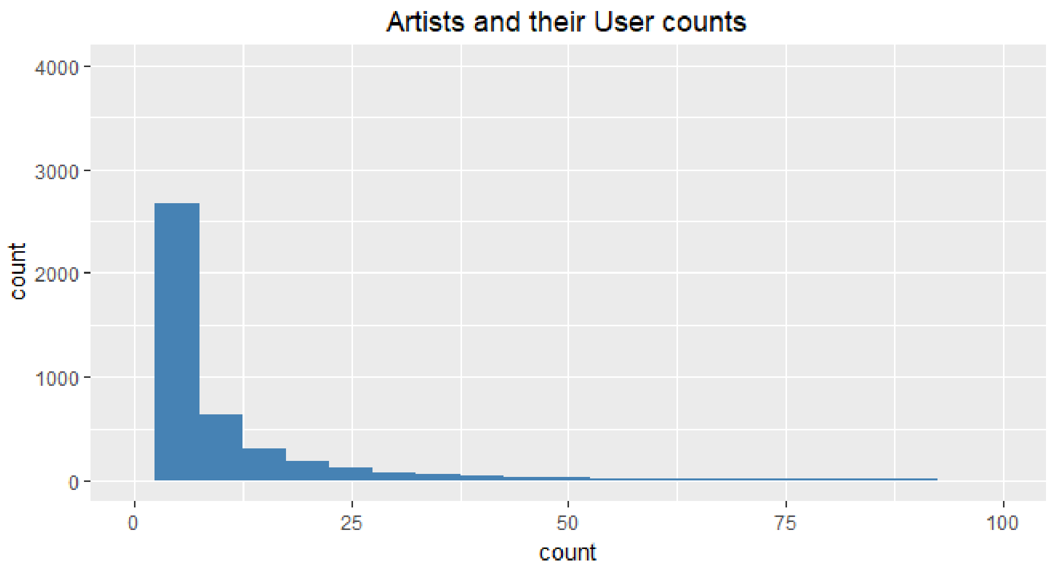
1. Britney Spears, Depeche Mode and Lady Gaga were the most played artists by all users



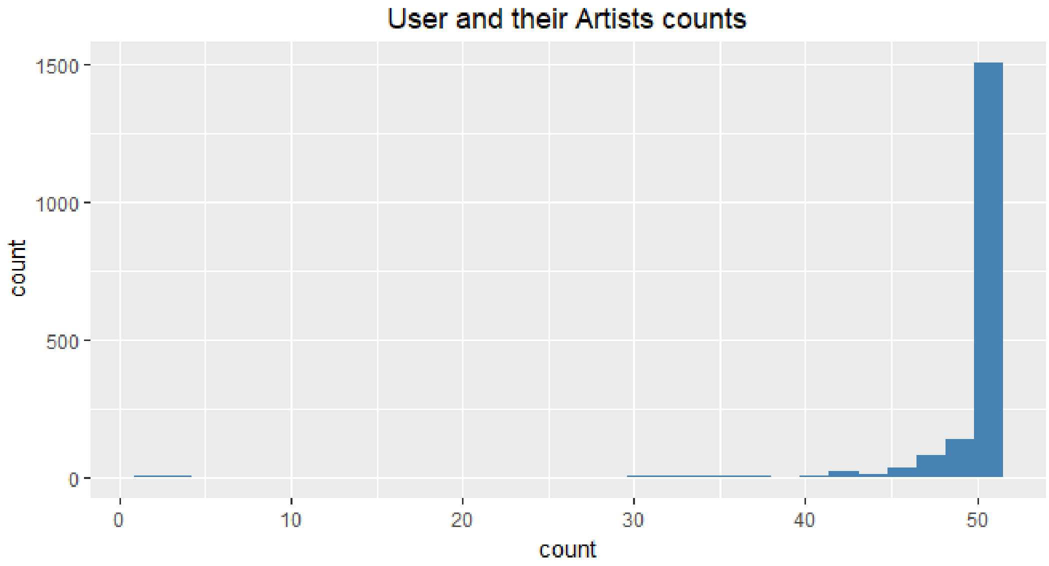
2. While most of the artists who were played the most, were also played by the most number of users/ But surprisingly Depeche Mode even after being listened to more than a million times, they were not played by many users. This indicates that: *Some users tend to play a lot of certain artists which others might not be playing. Hence, this can be a room to recommend artists in the future.*



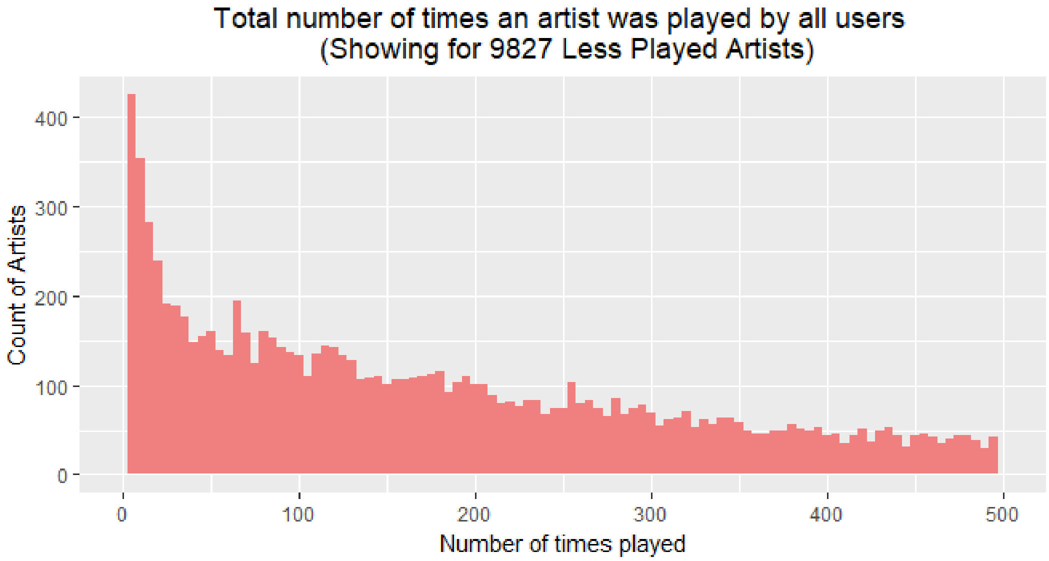
3. Out of 16996 available artists, a whopping 15605 (=91.8%) artists were listened only by 10 or even less users. Hence in the dataset the users were mostly listening to very selected artists.

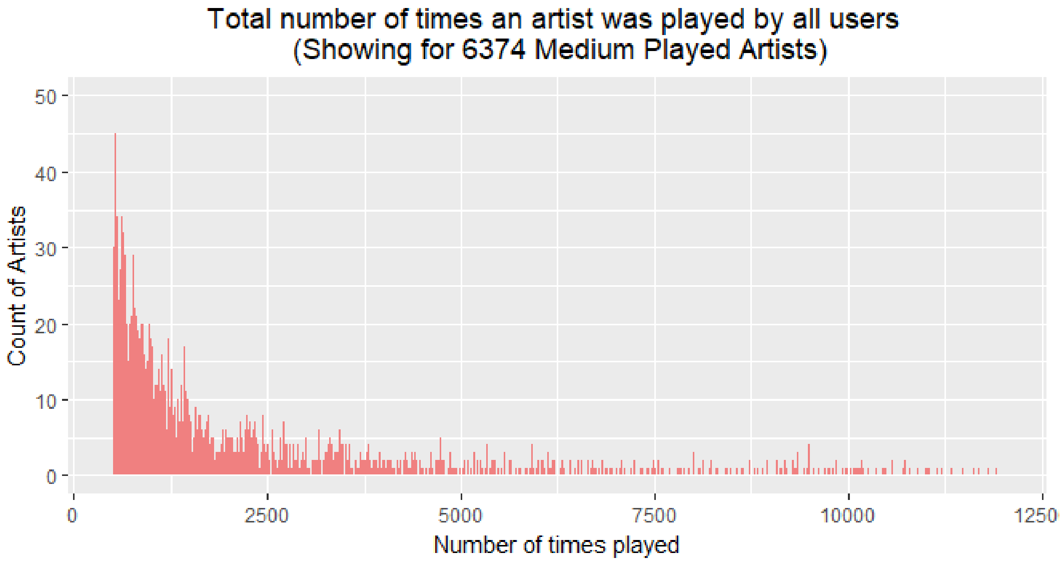


4. Out of 1891 users, 1506 (=79.6%) users have listened to 50 artists. But these set of 50 artists for one user can be entirely different from a set of 50 artists of another user, since the total available artists are 16996.



5. Below histograms show that out of 16996 artists, 16201 (= 9827 + 6374) artists were played less than 2500 times (very small compared to some artists like Britney Spear who was played 2.5 million times). Thus, it clearly shows that only a handful of artists are popular and dominating in the dataset.





DATA PROCESSING

It was found that some artists which were present in artists table were not present in the user – artist data. Same case was with few tags. Hence, a master dataset was prepared using only the userID, artistsID, and tagID which were common across all the tables.

It reduced the data size from that of 1891 users and 16996 artists to 1870 users and 6640 artists. Now again, in this reduced dataset there were some users who had listened only to one, two or basically a very handful of artists. For collaborative filtering, the recommendations are made based on similarity between users or items. Similarity can be measure in different ways: Euclidean, Cosine, Pearson, Jaccard etc. In all these measures, the similarity between two users or two items is calculated using vector of ratings. If the vector size is not big enough, then the recommendations will not be accurate. Hence, we needed to make sure that the rating vector for each user should be sufficiently larger. So, in the next step, those users were excluded who had not played enough artists.

Now if 1 single artist is listened million times but only by 1 single user, it doesn’t give enough information that this artist would be liked by other users as well and thus this artist can’t be recommended to any new user. So, it was assumed that any listening count which was below 30%ile of all the counts was not helpful. *Hence, some more artists were excluded to retain only those artists who were played by at least 6 unique users and with more than 211 times (40 %lie).* This further reduced the final data to 446 users and 262 artists. The sparse density at this stage was 4.5%

FITTING RECOMMENDERS

Next step was to fit recommenders to the above dataset. This step requires the data to be in the matrix form. Hence, the above user – artist data was transformed into a larger and sparse user – artist matrix. This is how the user-artists matrix looked like. Every row represents each user and every column represents each artist. The numbers in the cell are the number of times that artist was played by that user.

userID	A1001	A1013	A1044	A1045	A1047	A1048
U1003	261					
U1007						
U1009						84
U101					4218	
U1010		3				
U102						
U1020			530			

Next, we considered making two recommender systems because in some cases the absolute ratings produce weaker ratings while converting them to binary produce much better recommendations. For each of these systems, a different data matrix was needed.

Non-Binary version: The existing range of listening counts was very huge (as seen in exploratory data analysis) and the recommenderlab package in R (which was used to fit the recommenders) works best with a rating scale of 1-5. Hence, the absolute listening counts were converted to a scale of 1-5 based on the quantile the listening count belonged to. Below were the new ratings:

Quantile of Listening Count	**New Rating (1-5 Scale)**
0-20	1
20-40	2
40-60	3
60-80	4
80-100	5

Binary Version: One based on whether a user has listened to an artist. For this we coded 1 for any play count or user-artists interaction which was greater than the 30% ile of all the counts. (1 = Yes and 0 = No). Below is a table showing the quantile of the counts.

0%	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**
1	73.6	139	211	297	411	570	821	1238.8	2361

Below is the final table with binary coding which uses above table's quantiles.

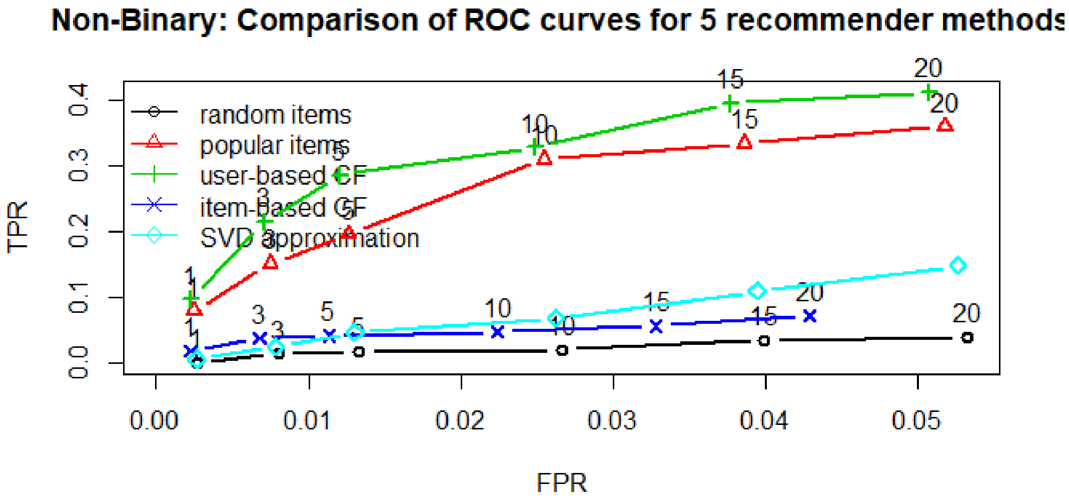
userID	A1001	A1013	A1044	A1045	A1047	A1048
U1003	1	0	0	0	0	0
U1007	0	0	0	0	0	0
U1009	0	0	0	0	0	1
U101	0	0	0	0	1	0
U1010	0	1	0	0	0	0
U102	0	0	0	0	0	0
U1020	0	0	1	0	0	0

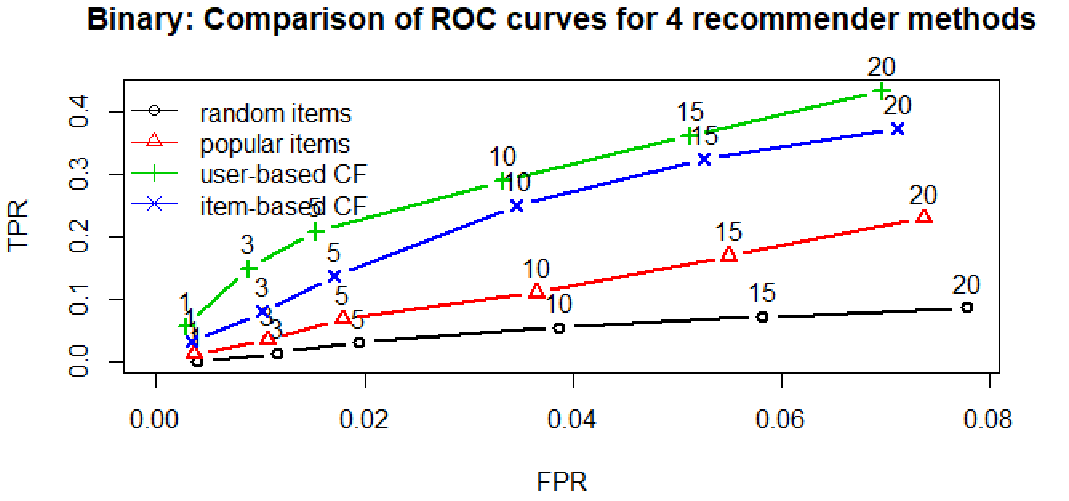
Next in both the versions following recommenders were fit and evaluated.

Non-Binary	**Binary**
User-Based	User-Based
Item-Based	Item-Based
Popularity-based	Popularity-based
Random	Random
Single Value Decomposition	*SVD can't be applied in case of Binary

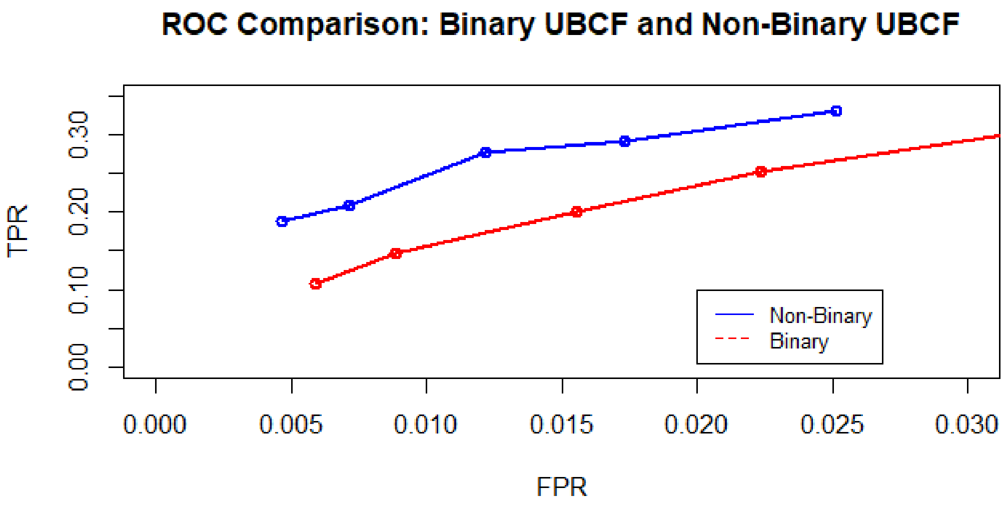
RECOMMENDER EVALUATION

After preparing the data matrix as "**realRatingMatrix**" object, we started search for the best recommender in each version. It was found that the User-Based recommender was consistent and outperformed other algorithms in both the versions.


Non-Binary: Comparison of ROC curves for 5 recommender methods

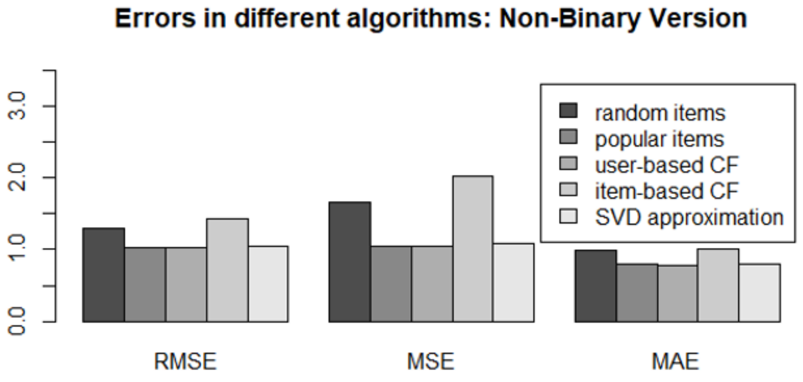

Binary: Comparison of ROC curves for 4 recommender methods

After that a comparison was made between the Binary and the Non-Binary version of the same user-Based algorithm. It was found that the Non-Binary version had a higher performance than a binary version. Below is the ROC comparison:



Next, we dig deeper in user-based CF for Non-binary version. Below table and barplot shows that the RMSE and other errors were lowest for the user-based recommender. Next plots are of ROC curves showing that user-based CF performs better than others. It should be noted here the ROC plots were done for top – 2,3,5,7 and 10 recommendations.

Method / Error	**RMSE**	**MSE**	**MAE**
UBCF	1.02595	1.05258	0.78212
POPULAR	1.02601	1.05269	0.79576
SVD	1.03972	1.08101	0.79256
IBCF	1.42426	2.02851	1.03236
RANDOM	1.28214	1.64389	0.95135



Hence, it can be concluded that for this dataset, user-based collaborative filtering with Non-Binary setting can be used to recommend new artists to users.

SAMPLE RECOMMENDATIONS

Next, we made recommendations for unseen new users who were not used to train the final recommender. From the recommendations we can see that artists with similar genres and who were more popular were recommended and thus in line with the expectations.

UserID	Type of Artist	Artists Names	Genres
u10	Previous	Arcade Fire, The National, Beirut	Indie, Alternative Rock
	Recommended	MGMT	Indie Pop, Psychedelic, Alterative
u1003	Previous	Duran, Madonna, Queen	80s, 1992, Pop
	Recommended	Lady Gaga, Depeche Mode	Techno, Synthpop, Female Vocalist, Electronic

Conclusion from the recommendations:

# U10: First user was more into indie, alternative rock music and other popular indie and alternative artists were recommended.

# U1003: Second user was more into pop and electronic music and artists like Lady Gaga and Depeche Mode were recommended.

BIBLIOGRAPHY

- Last.fm website: https://www.last.fm/
- Music industry (Abstract): https://www.forbes.com/sites/hughmcintyre/2017/04/25/the-global-music-industry-grew-by-6-in-2016-signalling-brighter-days-ahead/#33e56d6163e3
- Data Source: Grouplens.org http://files.grouplens.org/datasets/hetrec2011/hetrec2011-lastfm-2k.zip
- Role of Matrix Factorization Model in Collaborative Filtering Algorithm: A Survey https://arxiv.org/ftp/arxiv/papers/1503/1503.07475.pdf
- Matrix Factorization: http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/