Music Recommender System
Group 5: Fantastic Five
DSBA 6156: Applied Machine Learning

Arvind Suriakanth, Chieh Wu, Pragyna Veeramallu, Sai Kumar Kasireddy, Sina Saba

Spring 2022, University of North Carolina, Charlotte

Guide: Prof. Dr. Minwoo Jake Lee
May 04, 2022

GitHub Project: https://github.com/Pragyna05/Applied_ML_FantastivFive

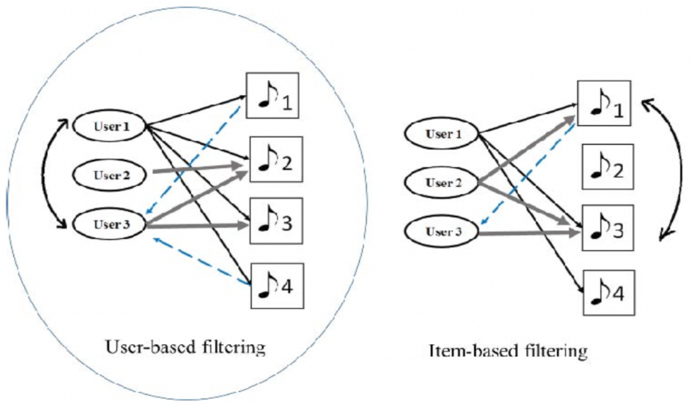
PRIMARY PAPER

A survey of music recommendation systems and future perspectives

Conference: The 9th International Symposium on Computer Music Modeling and Retrieval (CMMR) at: London, UK 2012.

Authors:

1. **Yading Song**
2. **Marcus Pearce**
3. **Simon Dixon**



Abstract

In this project, we have designed, implemented, and analyzed a song recommendation system. We used Million Song Dataset [1] provided by Kaggle to find correlations between users and songs and to learn from the previous listening history of users to provide recommendations for songs which users would prefer to listen most. In this paper, we will discuss the problems we faced, methods we have implemented, results and their analysis. We have got best results for memory based collaborative filtering algorithm. We believe that content-based model would have worked better if we would have enough memory and computational power to use the whole available metadata and training dataset.

Keywords- recommendation systems, music, Million Song Dataset, collaborative filtering, content-based.

1. Introduction

Rapid development of mobile devices and internet has made possible for us to access different music resources freely. The number of songs available exceeds the listening capacity of single individual. People sometimes feel difficult to choose from millions of songs. Moreover, music service providers need an efficient way to manage songs and help their costumers to discover music by giving quality recommendation. Thus, there is a strong need of a good recommendation system.
Currently, there are many music streaming services, like Pandora, Spotify, etc. which are working on building high-precision commercial music recommendation systems. These companies generate revenue by helping their customers discover relevant music and charging them for the quality of their recommendation service. Thus, there is a strong thriving market for good music recommendation systems.

Music recommender system is a system which learns from the users past listening history and recommends them songs which they would probably like to hear in future. We have implemented various algorithms to try to build an effective recommender system. We firstly implemented popularity-based model which was quite simple and intuitive. Collaborative filtering algorithms which predict (filtering) taste of a user by collecting preferences and tastes from many other users (collaborating) is also implemented. We have also done experiments on content-based models, based on latent factors and metadata.

1.1 Problem Statement

Rapid development of mobile devices and internet has made possible for us to access different music resources freely. The number of songs accessible far outnumbers anyone's ability to listen to them all. People find it tough to pick from millions of songs at times. Furthermore, music service providers want an effective method of managing songs and assisting their customers in discovering music through quality recommendations. As a result, an effective recommendation system is critical.

The music database is too large while life is short!!! You need someone to show you how to manage and make intelligent recommendations based on your preferences! Music service providers need a more effective system to attract customers! Music Recommender is to help users filter and discover songs according to their tastes. A good music recommender system should be able to automatically detect preferences and generate playlists accordingly.

1.2 Motivation

With the rise of digital content distribution, people now have access to music collections on an unprecedented scale. Commercial music libraries easily exceed 15 million songs, which vastly exceeds the listening capability of any single person. With millions of songs to choose from, people sometimes feel overwhelmed. Thus, an efficient music recommender system is necessary in the interest of both music service providers and customers. Users will have no more pain to make decisions on what to listen while music companies can maintain their user group and attract new users by improving users' satisfaction.

1.3 Challenges

Research in music recommender systems (MRS's) has recently experienced a substantial gain in interest both in academia and in industry. In the following, we identify and detail a selection of the grand challenges, which we believe the research field of music recommender systems is currently facing, i.e., overcoming the cold start problem, automatic playlist continuation, and properly evaluating music recommender systems.

Challenge 1: Cold start problem

One of the major problems of recommender systems in general, and music recommender systems in particular is the cold start problem, i.e., when a new user registers to the system or a new item is added to the catalog and the system does not have sufficient data associated with these items/users. In such a case, the system cannot properly recommend existing items to a new user (new user problem) or recommend a new item to the existing users (new item problem).

Challenge 2: Automatic playlist continuation

In its most generic definition, a playlist is simply a sequence of tracks intended to be listened to together. The task of automatic playlist generation (APG) then refers to the automated creation of these sequences of tracks. In this context, the ordering of songs in a playlist to generate is often highlighted as a characteristic of APG, which is a highly complex endeavor. Some authors have therefore proposed approaches based on Markov chains to model the transitions between songs in playlists. While these approaches have been shown to outperform approaches agnostic of the song order in terms of log-likelihood, recent research has found little evidence that the exact order of songs matters to users, while the ensemble of songs in a playlist and direct song-to-song transitions do matter.

Challenge 3: Evaluating music recommender systems

Having its roots in machine learning (cf. rating prediction) and information retrieval (cf. "retrieving" items based on implicit "queries" given by user preferences), the field of recommender systems originally adopted evaluation metrics from these neighboring fields. In fact, accuracy, and related quantitative measures, such as precision, recall, or error measures (between predicted and true ratings), are still the most employed criteria to judge the recommendation quality of a recommender system. In addition, novel measures that are tailored to the recommendation problem have emerged in recent years. These so-called beyond-accuracy measures address the particularities of recommender systems and gauge, for instance, the utility, novelty, or serendipity of an item. However, a major problem with these kinds of measures is that they integrate factors that are hard to describe mathematically, for instance, the aspect of surprise in case of serendipity measures. For this reason, there sometimes exist a variety of different definitions to quantify the same beyond-accuracy aspect.

1.4 Approach

To create an effective recommendation system, we developed four distinct approaches. Popularity-based models, same-artist greatest hits models, collaborative filtering models, and content-based models are among the approaches. Overall, we got the best results from collaborative filtering model with an efficiency of 8.2117. All the models are explained briefly in the below sections.

2. Backgrounds/ Related Work

2.1 A Survey of Music Recommendation Systems and Future Perspectives

From a paper titled "A Survey of Music Recommendation Systems and Future Perspectives" written by Yading Song and Marcus Pearce, we see that they explained basic metadata-based model and used two popular music recommender approaches: collaborative filtering and content-based model.

They used Collaborative filtering to recommend the items via choices of other similar users. They further divided collaborative filtering into three sub-categories: Memory-based, Model-based and Hybrid collaborative filtering.

They used content-based modeling to make predictions by analysing the song tracks.

Though they have achieved great success, their drawbacks such as popularity-bias and human efforts are obvious. Moreover, the use of hybrid model would outperform a single model since it incorporates the advantages of both methods. However, their project failed to provide a user-centric music recommendation.

Below are the cons from their project:

- Because of the subjectivity in music, the assumption that users with similar behaviours may have same tastes has been ignored.
- Though collaborative filtering recommender works well, the key problems such as cold start, popularity bias are the filters they haven't considered.
- The authors haven't fully investigated similarity-based method in terms of listener's preferences.

Some of the pros from their project were as follows:

- By using Hybrid collaborative filtering, they proved that by combining different collaborative filtering models, it is easy to make predictions and have proved that Hybrid collaborative filtering model outperforms any individual.
- Since music is a self-expression tool, it always performs with expression. The authors used Emotion based modeling and proved that music emotion has become the main trend for music discovery and recommendation using a fundamental emotional model (2D valence-arousal).
- They have also used context-based information retrieval model that uses public opinion to discover and recommend music.

How do we relate their planned approach to our approach?

In addition to their algorithms used i.e., Collaborative filtering and content-based filtering, we developed the recommendation system using two more algorithms which are Popularity based and the greatest hits of the same artist.

Similar to their approach, we have used K-Nearest Neighbor algorithm for each user whose past ratings have the strongest correlation.

We also used some similarity measures to compare between two songs or between two users.

Furthermore, we used SVD (Singular Value Decomposition) too which decomposes music into a latent feature space that relates users to songs.

2.2 Deep learning in Music Recommendation Systems

Deep Learning (DL) is increasingly used in Music Recommendation Systems (MRS), as it is in many other fields of research. Deep neural networks are particularly useful in this sector for extracting latent characteristics of music items from audio signals or metadata, as well as learning sequential patterns of music items (tracks or artists) from playlists or listening sessions.

Author Michael A. Riegler outlines the peculiarities of the music domain in Recommendation System research in this article. It provides an overview of the state of the art in music recommendation using deep learning. The discussion is divided into four categories: neural network type, input data, recommendation approach (content-based filtering, collaborative filtering, or both), and task (standard or sequential music recommendation).

Below are the cons from their project:

- Because of the recurring problems in existing algorithms, there was a lack of real-time updates and multiple variable inputs.
- There was also a lack of established multimodal datasets and the large variety of evaluation metrics used to compare results between approaches.
- The author has also failed to account for changing user preferences, such as the fact that while the user may have one intention today when listening to a song, the same user may have a different intention the next day.

Some of the pros from their project were as follows:

- By using content based and hybrid approaches, the authors used the output of the last hidden layer as latent content representation of songs and assume that this representation captures music semantics.
- By using sequence-aware music representation approach, the authors report several performance measures at 20 recommended items. The proposed model thereby outperforms several traditional and state-of-the-art approaches used as baseline.
- Notwithstanding the importance of a system-centric perspective, they even had a challenge to achieve a balance between a purely system-centric view and a holistic user-centric view which was achieved.

How do we relate their planned approach to our approach?

In addition to their approaches used, we will be using other approaches taking their ideas from their project.

We have also used precision metric as an evaluation measure since precision is much more important than recall because false positives can lead to a poor user experience.

After generating user- and item-based lists, we combined them using stochastic aggregation. This is accomplished by selecting one of them at random from their probability distribution and then recommending the highest-scoring items from it.

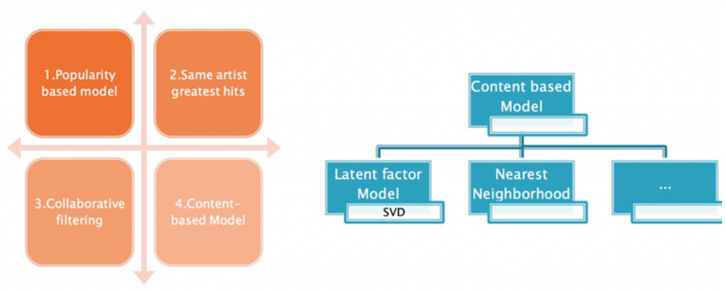
3. Dataset

We used data provided by Million Song Data Challenge hosted by Kaggle. It was released by Columbia University Laboratory for the Recognition and Organization of Speech and Audio. The data is open; meta-data, audio content analysis, etc. are available for all the songs. It is also very large and contains around 48 million (userid, songid, play count) triplets collected from histories of over one million users and metadata (280 GB) of millions of songs [7]. But the users are anonymous here and thus information about their demography and timestamps of listening events is not available. The feedback is implicit as play-count is given instead of explicit ratings. The contest was to predict one half of the listening histories of 11,000 users by training their other half and full listening history of other one million users.

Since processing of such a large dataset is highly memory and CPU-intensive, we used validation set as our main data. It consists of 10,000,000 triplets of 10000 users. We used metadata of only 10,000 songs (around 3GB). From the huge amount of song metadata, we focus only on features that seem to be most relevant in characterizing a song. We decided that information like year, duration, hotness, danceability, etc. may distinguish a song most from other songs. To increase processing speed, we converted user and song ids from strings to integer numbers.

4. Method/ Algorithms

We have implemented four different algorithms to build an efficient recommendation system. Algorithms include Popularity based model, same artist – greatest hits model, collaborative filtering model and content-based model.



4.1 Popularity based Model

It is the most basic and simple algorithm. We find the popularity of each song by looking into the training set and calculating the number of users who had listened to this song. Songs are then sorted in the descending order of their popularity. For each user, we recommend top most popular songs except those already in his profile. This method involves no personalization, and some songs may never be listened in future.

Computing Popularity

Here we only use the testing part because it is smaller, but a better system would use the full training set. In the following lines of code, we open the file, create a mapping from a song ID to the number of times this song appears, and close the file.

```
In  [1] : f = open('kaggle_visible_evaluation_triplets.txt', 'r')
In  [2] : song_to_count = dict()
In  [3] : for line in f:
     . . . :  _, song, _ = line.strip().split('\t')
     . . . :  if song in song_to_count:
     . . . :              song_to_count[song] += 1
     . . . :  else:
     . . . :              song_to_count[song] = 1
     . . . :
In  [4] : f.close()
```

For instance, the following song appears 13 times:

```
In  [1]    : song_to_count['SONZTNP12A8C1321DF']
Out  [5] : 13
```

Next, we re-order the songs by decreasing popularity:

```
In  [6] : songs_ordered = sorted(song_to_count.keys(),
                key=lambda s: song_to_count[s],
                reverse=True)
```

We will recommend the most popular songs to every user, but we must filter out songs already in the user's library. Reopening the triplets file, we will create a map from user to songs they have listened to.

```
In  [7] : f = open('kaggle_visible_evaluation_triplets.txt', 'r')
In  [8] : user_to_songs = dict()
In  [9] : for line in f:
     . . . :  user, song, _ = line.strip().split('\t')
     . . . :  if song in user_to_songs:
     . . . :            user_to_songs[user].add(song)
     . . . :  else:
     . . . :            user_to_songs[user] = set([song])
     . . . :
In  [15] : f.close()
```

For each user, we now have a list of songs ordered by popularity and listening history for each user.

4.2 Same artist – Greatest hits

We need to add more dictionaries to this model. One will hold the users as keys and the artists whom the user listened to their songs as the values. Other dictionary with keys to be artists and values to be the songs of that artist. Once we have these extra dictionaries, we can sort the songs base on each user's history.

First, for each user we selected all the songs from the same artists that the user listened to, and then order those songs by popularity. In the next step we added more popular songs if we had not enough songs base on user's history. Now we have a new order of songs for this specific user base on user's history and popularity of the songs and we then suggested new songs to this user.

We can see this algorithm is easy to implement but it is not efficient, since we need to reorder the songs base on each user's history. So, to get more efficient model we need to implement Machine Learning models and use more features from the songs.

4.3 Collaborative filtering based Model

Collaborative filtering involves collecting information from many users and then making predictions based on some similarity measures between users and between items. This can be classified into user-based and item based models.

In item-based model [Figure1], it is assumed that songs that are often listened together by some users tend to be similar and are more likely to be listened together in future also by some other user.

According to user-based similarity model [Figure2], users who have similar listening histories, i.e., have listened to the same songs in the past tend to have similar interests and will probably listen to the same songs in future too.

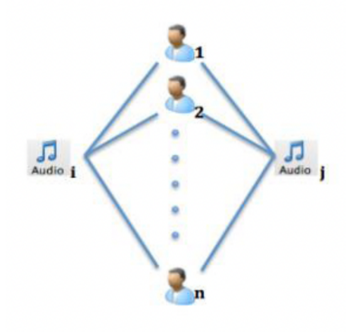


Figure 1: Item-based

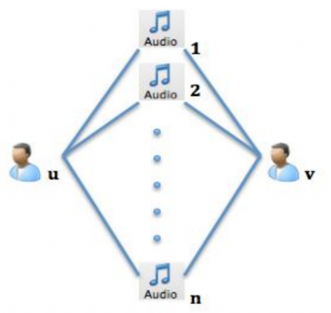


Figure 2: User-Based

Experiments

We need some similarity measure to compare between two songs or between two users. Cosine similarity weighs each of the users equally which is usually not the case. User should be weighted less if he has shown interests to many varieties of items (it shows that either she does not discern between songs based on their quality, or just likes to explore). Likewise, user is weighted more if listens to very limited set of songs. The similarity measure, $w_{ij} = P(i/j)$, also has drawbacks that some songs which are listened more by users have higher similarity values not because they are similar and listened together but because they are more popular.

We have used conditional probability-based model of similarity [2] between users and between items:
$W_{u,v} = P(v/u)^{\alpha}P(v/u)^{1-\alpha}$, $\alpha\in(0, 1)$

$$W_{uv} = \frac{|I(u) \bigcap I(v)|}{|I(u)|^{\alpha}|I(v)|^{1-\alpha}} [2]$$

Different values of α were tested to finally come with a good similarity measure.

Then, for each new user u and song i, user-based scoring function is calculated as

$h^{U}_{ui} = \sum_{v\in ui} f(w_{uv})$ [2]

Similarly, item-based scoring function is

$h^{I}_{ui} = \sum_{j\in uI} f(w_{ij})$ [2]

Locality of scoring function is also necessary to emphasize items that are more similar. We have used exponential function to determine locality.

$f(w) = w^{q}$, $q\in N$ [2]

This determines how individual scoring components affects the overall scoring between two items. The similar things are emphasized more while less similar one's contribution drop down to zero.

After computing user-based and item-based lists, we used stochastic aggregation to combine them. This is done by randomly choosing one of them according to their probability distribution and then recommending top scored items from them. When the song history of a user is too small to utilize the user-based recommendation algorithm, we can offer recommendations based on song similarity, which yields better results when number of songs is smaller than that of users.

	Item1	Item2	Item3	Item4	Item5	Item6
User1						
User2						
User3						
User4						

Figure 3: Matrix M

We got the best results for stochastic aggregation of item-based model with values of q and α as 3 and 0.15, and of user-based model with values 0.3 and 5 for α and q respectively, giving overall mAP 0.08212. (See details about mAP in the Sec 4.6)
This method does not include any personalization. Moreover, majority of songs have too few listeners, so they are least likely to be recommended. But still, we got best results from this method. Here, we have not used play count information in final result as they did not give good result because similarity model is biased to a few songs played multiple times and calculation noise was generated by a few very popular songs.

4.4 SVD Model

Listening histories are influenced by a set of factors specific to the domain (e.g., genre, artist). These factors are in general not at all obvious and we need to infer those so-called latent factors [4] from the data. Users and songs are characterized by latent factors.

Here, to handle such a large amount of data, we build a sparse matrix [6] from user-song triplets and directly operate on the matrix [Figure 3], instead of looping over millions of songs and users. We used truncated SVD for dimensionality reduction.

Experiments

We used SVD algorithm in this model as follows:

Firstly, we decompose Matrix M into latent feature space that relates user to songs.

$M = U \sum V$, where

$M \in R^{m*n}$, U^{m*k}, P^{k*k} and V^{k*n}

Here, U represents user factors and V represents item factors. Then, for each user, a personalized recommendation is given by ranking following item for each song as follows:

$W_i = U^T_u . V_i$

Though the theory behind SVD is quite compelling, there is not enough data for the algorithm to arrive at a good prediction. The median number of songs in a user's play count history is fourteen to fifteen; this sparseness does not allow the SVD objective function to converge to a global optimum.

4.5 KNN Model

In this method, we utilize the available metadata. We create a space of songs according to their features from metadata and find out neighborhood of each song. We choose some of the available features (e.g., loudness, genre, mode, etc.) which we found most relevant to distinguish a song from others. After creating the feature space, to recommend songs to the users, we look at each user's profile and suggest songs which are neighbors to the songs present in his listening history. We have taken top 50 neighbors of each song. This model is quite personalized and uses metadata. But since, we had 280GB file of metadata which takes huge amount of time in processing, we extracted features of only 3GB (10,000 songs), which is less than 2 % of total number. Due to this, we had features of only small number of songs, which gives us very small precision.

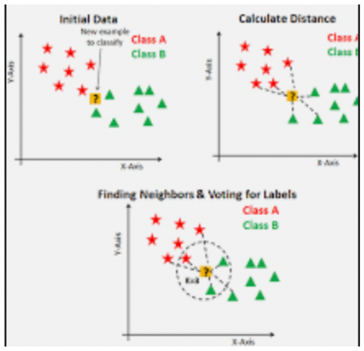


Figure 4: K-NN

4.6 Evaluation Metric

We used mean Average Precision (mAP) as our evaluation metric. The reason behind using this is that this metric was used in the Kaggle challenge which helps us to compare our results with others. Moreover, precision is much more important than recall because false positives can lead to a poor user experience. Our metric gives the average of the proportion of correct recommendations giving more weight to the top recommendations. There are three steps of computing mAP as follows:

Firstly, precision at each k is calculated. It gives the proportion of correct recommendations within the top-k of the predicted rankings.
$P_k(u, r) = \frac{1}{k} \sum_{j=1}^{k} M(u, r(j))$

Then, for each user, average precision at each k is evaluated.
$AP(u, r) = 1/ n_u \sum_{k=1}^{t} P_k(u, r).M(u, r(k))$

Finally, mean of all the users is taken.
$mAP = \frac{1}{m}\sum_{u=1}^{m} AP(u, r_u)$

5. Experiments/ Results

We got best results for memory based collaborative filtering algorithm. Our SVD based latent factor model gives better results than popularity-based model. It lags collaborative filtering algorithm because the matrix was too sparse which prevented objective functions to converge to global optimum. Our K-NN model did not work well and performs worse than even the popularity model. The reason behind this is that we have the features of only 10,000 songs, which is less than 3 % of the whole dataset so only some of these 10,000 songs could be recommended. The huge lack of information leads to the bad performance of this method.

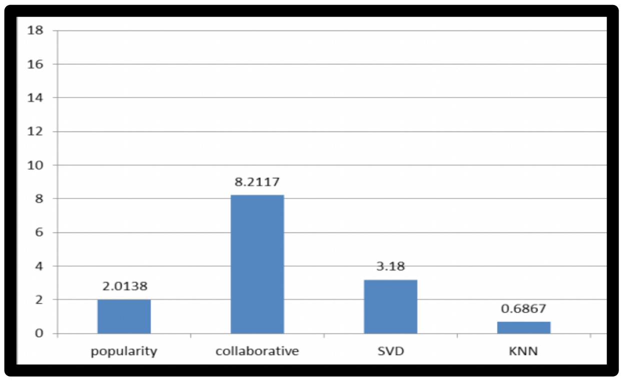


Figure 5: Results

6. Conclusion

This is a project of our Applied Machine Learning course. We find it is very good as we got a chance to practice theories that we have learnt in the course, to do some implementation and to try to get a better understanding of a real machine learning problem: Music Recommendation System.

There are many different approaches to this problem, and we get to know some algorithms in detail and especially the four models that we have explained in the paper. By manipulating the dataset, changing the learning set, and testing set, changing some parameters of the problem, and analyzing the result, we earn a lot of practicing skills. We have faced a lot of problem in dealing with this huge dataset, how to explore it in a better way and we also had difficulties in some programming details. However, with lot of efforts, we have overcome all of these.

The best part of this project is the teamwork. All of us come from different countries and thus have different cultures and ways of working. We took a bit of time to get to know each other, to adjust ourselves and to perform like a team. We become much more efficient by the time the team spirit is formed and we also enjoy more. We all find this project a nice experience and all the effort put is worthy. We have learnt a lot from this project.

In terms of research, we still have a lot to do to make our studies a better one. Music Recommender System is such a wide, open, and complicated subject that we can take some initiatives and do a lot more tests in future.

We also got to realize that building a recommender system is not a trivial task. The fact that its large-scale dataset makes it difficult in many aspects. Firstly, recommending 500 correct songs out of 380 million for different users is not an easy task to get a high precision. That's why we didn't get any result better than 10 %. Secondly, the metadata includes huge information and when exploring it, it is difficult to extract relevant features for song. Thirdly, technically speaking, processing such a huge dataset is memory and CPU intensive.

All these difficulties due to the data and to the system itself make it more challenging and more attractive. We hope that we will get other opportunities in the future to work in the domain of Machine Learning. We are certain that we can do a better job.

7. Future Work

- Run the algorithms on a distributed system, like Hadoop or Condor, to parallelize the computation, decrease the runtime and leverage distributed memory to run the complete MSD.
- Combine different methods and learn the weightage for each method according to the dataset
- Automatically generate relevant features

- Develop more recommendation algorithms based on different data (e.g. the how the user is feeling, social recommendation, etc)

8. Contributions

From Primary paper [8]
We too used the algorithms collaborative filtering and content-based filtering from paper [8]. We have taken the dataset from Kaggle website link. We further divided collaborative filtering into two models that is user – based model and item – based model. The project repository link has all the code.

I have built most of the code with knowledge I have gained from the course lab assignments.
I have written the code for
1. Loading the Data,
2. Pre-processing data using imputations and cleaned the Data
3. Developed four model using machine learning algorithms.
4. Calculated metrics to check the efficiency of the model.

9. Acknowledgements

We would like to acknowledge the efforts of Dr. Minwoo Lee as without his constant support and guidance this project would not have been possible.

References

[1] McFee, B., BertinMahieux,T., Ellis, D. P., Lanckriet, G. R. (2012, April). The million song dataset challenge. In Proceedings of the 21st international conference companion on World Wide Web (pp. 909916).ACM.

[2] Aiolli, F. (2012). A preliminary study on a recommender system for the million songs dataset challenge. PREFERENCE LEARNING: PROBLEMS AND APPLICATIONS IN AI

[3] Koren, Yehuda. "Recommender system utilizing collaborative filtering combining explicit and implicit feedback with both neighborhood and latent factor models."

[4] Cremonesi, Paolo, Yehuda Koren, and Roberto Turrin. "Performance of recommender algorithms on topn recommendation tasks." Proceedings of the fourth ACM conference on Recommender systems. ACM, 2010

[5] T. Bertin et al., The Million Song Dataset, Proc. of the 12th International Society for Music Information Retrieval Conference, 2011

[6] Sparse Matrices http://docs.scipy.org/doc/scipy/reference/sparse.html

[7] Mahiux Ellis 2-11 http://millionsongdataset.com/tasteprofile/

[8] A Survey of Music Recommendation Systems and Future Perspectives, Conference: The 9th International Symposium on Computer Music Modeling and Retrieval (CMMR)At: London, UK by Yading Song, Marcus Pearce, and Simon Dixon.

[9] Deep Learning in Music Recommendation Systems, Institute of Computational Perception, Johannes Kepler University, Linz, Austria by Michael A. Riegler.