



## *Final Review Report*

# **Music Recommendation System Using machine learning algorithms**

**Submitted By-**

T.Guna Sekhar - 18BCI0002

G.B.S.Sathvik - 18BCI0087

## **ABSTRACT**

On regular basis, individuals dependably require a few advices when settling on the choices. Regardless of whether it is which motion picture to watch at Friday night or are there any fascinating new items accessible on the Amazon. Under this unique situation, we fabricate a framework that can consequently prescribe new melodies to customers dependent on their listening history before. Lots of businesses are nowadays using recommender systems for their benefit like Amazon and flip kart for selling products (e-commerce), wynk music and ganna.com for music streaming, for selling books, for movies, YouTube for videos recommendations. This helps both businesses and users as businesses are getting monetary benefit by attracting customers and users are getting benefitted by getting better services. Everybody is using recommender systems nowadays in various forms and day to day these are getting improved because researchers are researching on making them better and better each day due to high competition in the market for giving better services and attracting customers.

This project mainly focuses on only suggestion music for music lovers to help them listen to the songs those they may like .This framework enables clients to find new collections or tunes making the melodic list accessible for tuning in. Along these lines the framework can evaluate what artist or gathering would coordinate client inclinations to the client. For music enthusiastic people music is life and music is a larger part of everybody's lives because everything in this universe can be correlated to frequency and vibrations. Music helps us tuning ourselves with the universe and best thing about music is that nothing can relax you more than a pleasing melody. Due to all the good things about music and high demand of recommender systems in the market we have chosen to do this project

## 1. INTRODUCTION

Nowadays lots of music industries like amazon music, wink music, gaana.com are using recommender systems and the old fashioned way of selling music has changed to a totally different cloud based .Now all the music resources are present in their cloud and users can listen to the songs directly from the cloud. But the issue is there are lot of songs present in the cloud system. so we need to classify all the songs based on different genres ,artists locations , age groups, languages and the main goal is to classify these set of songs in accordance to the taste of the user. Because user expects valuable return after the investment of time as well as money thereby we can attract a lot of customers by providing various valuable services of their interests For this project we are using various machine learning algorithms as well as data mining techniques. We have implemented various algorithms and compared the results with one another to find the effective algorithm that suits our model.

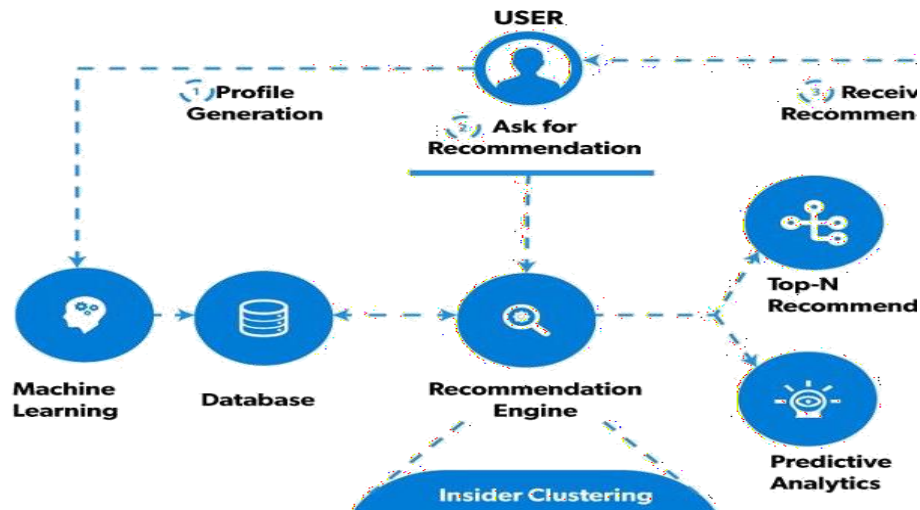
The most common approaches people have used implementing various recommender systems are collaborative filtering and content based models. These algorithms aim to find similarity between various users various songs and artists. Other than these algorithms we have also used random forest algorithm and decision tree algorithm. Both of these algorithms aims to predict a decision based on different attributes. For improving the efficiency and accuracy of the model we have also used cross validation technique in which we have recursively changed the training and testing data set to get optimum results. We have faced a lot of problems which making this project like the data we have chosen was too big (1.2 GB) so we had to create a subset of that data set for optimum usage. There were lot of outliers present in that data set which required complex data pre-processing.



## **1.1 Recommender System**

With the promotion of the Internet and the advancement of E-trade, the Ecommerce destinations offer a great many items available for purchasing. Picking among such a large number of choices is challenging for buyers. So clients typically lose all sense of direction in the huge space of ware data and can't discover the products they truly need. Recommender frameworks have risen in light of this issue. A recommender framework for an E-trade site prescribes items that are probably going to meet client's requirements.

Suggestion frameworks have rapidly changed the way in which the life less sites can now interact and speak with their clients and users. As opposed to provide a constant involvement in which clients look for and conceivably purchase items, recommender frameworks increment cooperation to give a more extravagant or deal. Recommender architecture is used by the E-business objective to propose and suggest items and services which are similar to their clients. There are many constraints and parameters on which an internet service provider recommends a user certain choices and options depending upon some restrained set of parameters. These parameters can include language , age , nationality, history, likes, ratings, purchase and many more. The items can be prescribed and suggested dependent on the best generally speaking and interacting vender on a particular site, in the presence of the social and economic constraints of the client, or dependent on an examination of the past purchasing conduct of the client as an expectation for future purchasing conduct. These parameters enable the service providers to analyse and determine a set of choices and preferences. Moreover, these strategies have been used extensively and are a piece of personalization on a site, since this enable the site to adjust to every client in accordance to the client or user .Moreover these famous and successful online service providers use personalization as a key component while recommendations because generalization cannot be more accurate under defined parameters. Recommender frameworks mechanize personal services and platforms on the internet, which facilitates and empowers singular personalization for every client.



(Recommender Architecture)

## 1.2 Why Recommender Systems?

As the web moved from a proprietor model to an open publicly supporting model and enabled individuals to contribute unreservedly, it saw an exponential ascent in the measure of substance accessible, which was something to be thankful for. Be that as it may, this prompted two noteworthy issues:

- i. Aggregation:** The measure of data turned out to be large to the point that it inspired extreme to oversee it while as yet having the capacity to run a web benefit that was reachable to all parts of the world. This issue was tackled by building overall substance conveyance and dissemination systems, helped by the ascent of NoSQL Database frameworks and diminishing stockpiling costs.
- ii. Searching:** The second significant issue was the means by which to guarantee that the data is inside the scope of the client and that the client does not become mixed up in the immense information dumps accessible. This turned out to be a significantly more concerning issue than accumulation since the information troves are tremendous and every client carries alongside him/her a remarkable point of view and consequently a one of a kind pursuit design. We are as yet attempting to take care of this issue today and are a long way from accomplishing an ideal answer for it. This is the place recommender frameworks become possibly the most important factor.

More or less, a recommender framework is a framework that predicts client reaction to an assortment of choices. Anticipating what the client may get next is the basic point of a recommender framework. There is a broad class of web applications that include foreseeing the client's reaction to choices. Such an office is known as a recommender framework.



### **Generalized Recommender**

In today's services like Netflix, Spotify, youtube offers their customers with plenty of choices. A person using such a service is known as client or user and to assist user, services can use information filtering to recommend items to users. Items can be several things such as books, music, movies, news etc.

Recommender systems need information for functioning, data about a particular user. This particular data can be fetched directly or indirectly. Directly collecting data means that user of a particular service gives feedback and review of the item. Indirectly means that system will analyze the users interaction with the particular service consisting of history and present services.

### **1.3 Problem statement**

The main goal is suggesting best set of options to the user. For a specific user we had their song history frequency list liked songs. From all this information we had to predict what songs user might like then the question comes: how can we use all this information to achieve our goal. As it

not a straight forward task to find the relevance between various songs it might be possible that one song which looks similar to other may be completely different and users may dislike that song or may be that song is not of users taste .there are lots of user around the world and lots of songs so making a relevance between songs and users is a tedious task.

## **1.4 Objective**

### **1.4.1 Learning Objective**

Learning objective of doing this project was to first to learn about machine learning and its key concepts and various data mining techniques and algorithms .other goal was to learn a lot of machine learning algorithm and how to use them .Just learning algorithms doesn't makes you an engineer the real task is to learn which is the right algorithm to apply for a specific project .

### **1.4.2 Outcome objective**

The main object in terms of outcome was to create a framework for users which can help them suggesting the right songs for them .this project aims to find the correlation and similarity between different music lovers their tastes and various songs so that if a user's taste is similar to the other one we can recommend the songs of one to another on the basis of similar taste. Or if a song is similar to the other we can suggest that song to the user that listen the first one. One of the object of this project is to reduce the time that user generally wastes on looking for the right song

## **1.5 Methodology**

### **1.5.1Functional Requirement**

The functional requirement specification of the project are mainly categorized as user requirements, security requirements, and device requirement each of which are explained in detail below:

**i. User Requirement:** User ought to have account on framework and client must have somewhere around one song listened to investigate the identity for the music suggestion.

### **1.5.2Non-functional Requirement**

- i. Performance:** The framework will have a speedy, exact and dependable outcomes.
- ii. Capacity and Scalability:** The framework will have the capacity to store identity registered by the framework into the database.
- iii. Availability:** The framework will be accessible to client whenever at whatever point there is an Internet association.
- iv. Recovery:** if there should arise an occurrence of breaking down or inaccessibility of server, the framework ought to have the capacity to recuperate and keep any information misfortune or excess.
- v. Flexibility and Portability:** System will be available whenever from any areas



## 2. LITERATURE SURVEY

The proposition issue in the music region has additional challenges as person & music perception depends upon various parameters and constraints. In a research it was found that songs acumen is impacted by the setting of the customer. They found that music preference mainly differs on the basis of age differences, locations and languages. These parameters further can be classified into sub age groups, countries, states, regional languages and many more. It was reported that artists of similar sounds does not necessarily have the similar music and taste of listeners may differ. Music can be near or undeniable to the extent in every way that really matters any property that can be used to depict music, for instance, sort, melody, beat, arrive starting and instrumentation, which makes it possible to answer the subject of similarity between two skilled workers from different perspectives. In a research it was found that most of the music listeners are in the age between 16 to 45 years of age and that was further divided into sub - groups:

- i. Broad taste:** People whose melodic learning are exceptionally broad. They contributed 7 percent of total division.
- ii. Enthusiasts:** There are lot of people in this world who believes that music is life and they are crazy for music. Indeed, music is the most relaxing thing in this world. They include 21 percent of this division.
- iii. Casual music listeners:** People who casually listen to music in their free time include 32 percent of this division.
- iv. Indifferent:** They have different mindset about music and including 40% of this age group.

As per a research every person requires unique set of suggestions. Academics is exceptionally urgent and are along these lines the most troublesome audience members to give suggestions to. They require unsafe and shrewd proposals rather than famous ones. Casuals and indifferent, who speak to 72% of the populace, don't require confused proposals and famous standard music that they can without much of a stretch relate to would accommodate their melodic needs. In this way, it is critical for a recommender framework to have the capacity to recognize the kind of client and act as needs be.

The objective was to enhance suggestion precision by including more sound information from numerous melodies. For this purpose, songs from similar collection and similar artists were analysed to find the correlation and was named as “collection effect”.

As of late, be that as it may, inquire about on recommenders utilizing communitarian separating has picked up a greater prominence in the music space. The main music recommender framework utilizing community oriented. It utilised a compelled person connection for computing similarity effect which corresponds to total like content.

Then again, slithered client related to an enormous account of robotized tunes that drove web looks of blueprints to somebody's gifted specialists. They utilized system sifting techniques on the information to diagram proposals. Then they utilized substance based and synergistic sifting approaches unreservedly to support music subject to music and client social gatherings. The music packs contained melodies the client was beginning late enchanted by and client bunches set clients with comparative interests. They accomplished higher precision with the substance-based framework, yet the synergistic secluding system gave all the moreover dumbfounding suggestions. Sanchez-Moreno et al. (2016) proposed a total separating method that utilized listening coefficients as an approach to manage area the decrease sheep issue of synergistic sifting.

In order to distinguish between clients, the listening clients conduct with respect to specialists they tune into which is utilized to describe the clients dependent on the exceptionalness of their inclinations. The proposed strategy fundamentally surpassed more customary community oriented sifting technique.

### 3. SYSTEM DEVELOPMENT

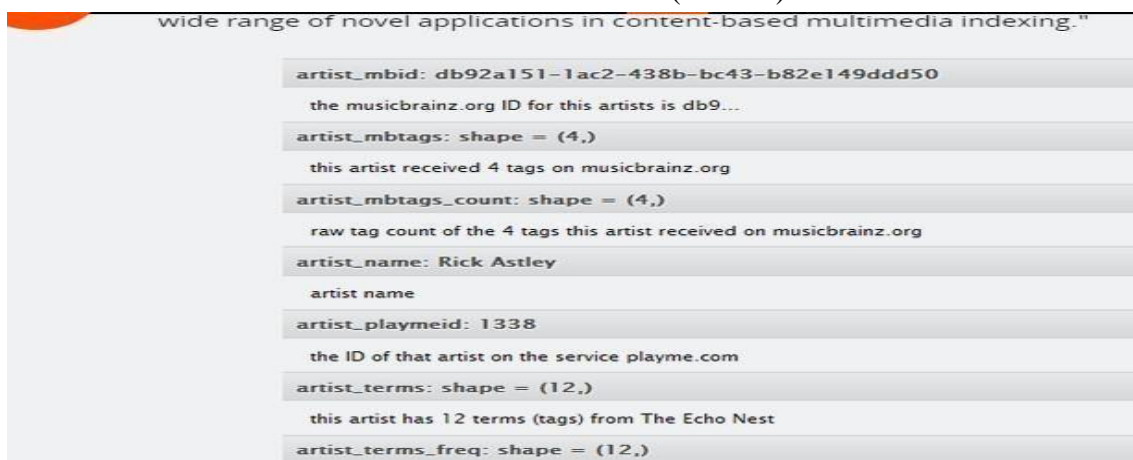
#### 3.1 Approach

- i. Gathering the data (choosing dataset)
- ii. Applying data mining techniques on dataset (data pre-processing and data cleaning)
- iii. Making dataset ready to be used in various algorithms
- iv. Choosing the right algorithms for the project
- v. Applying various machine learning algorithms
- vi. Comparing the results and then choosing the right model

While preparing a machine learning model the very first thing to do is to gather the data. We have lots of open datasets available for music recommendation and we have used the most common and reliable dataset.

#### 3.2 Dataset

For this project we have chosen million song dataset which was taken from Kaggle and was prepared by Columbia University for research of audio systems. This dataset included the song listening history of around million people but because the dataset was too big (1.2GB) so we had to take the subset of the dataset for our project. The dataset had various attributes like user id, song counts, language, artist, genre, and year. The dataset was further divided into two parts namely train set and test set and were stored in individual train.csv and test.csv files to analyse and to train the model on the basis of train set and then test with test set (test.csv) file.



The screenshot shows a Jupyter Notebook interface with a title bar that reads "wide range of novel applications in content-based multimedia indexing." The notebook contains a series of code cells and their corresponding outputs. The code cells use the `artist_mbid` to look up information for Rick Astley on MusicBrainz and Playme.com. The outputs show the artist's name, their Playme.com ID, and the number of tags and terms associated with them.

Code Cell	Output
<code>artist_mbid: db92a151-1ac2-438b-bc43-b82e149ddd50</code>	the musicbrainz.org ID for this artists is db9...
<code>artist_mbtags: shape = (4,)</code>	this artist received 4 tags on musicbrainz.org
<code>artist_mbtags_count: shape = (4,)</code>	raw tag count of the 4 tags this artist received on musicbrainz.org
<code>artist_name: Rick Astley</code>	artist name
<code>artist_playmeid: 1338</code>	the ID of that artist on the service playme.com
<code>artist_terms: shape = (12,)</code>	this artist has 12 terms (tags) from The Echo Nest
<code>artist_terms_freq: shape = (12,)</code>	

Figure 3.1(Sample Data set)

### 3.3 Data mining

These days, information is surrounding us, individuals everywhere throughout the globe are overpowered with information. The idea of information mining has begun to grow starting with one scholarly field then onto the next one, for example, business, restorative exercises, insights, sociological and numerous others. In simple terms data mining means firstly extracting the useful data and then making that data suitable to be used in the algorithms by cleaning the data and transforming it.

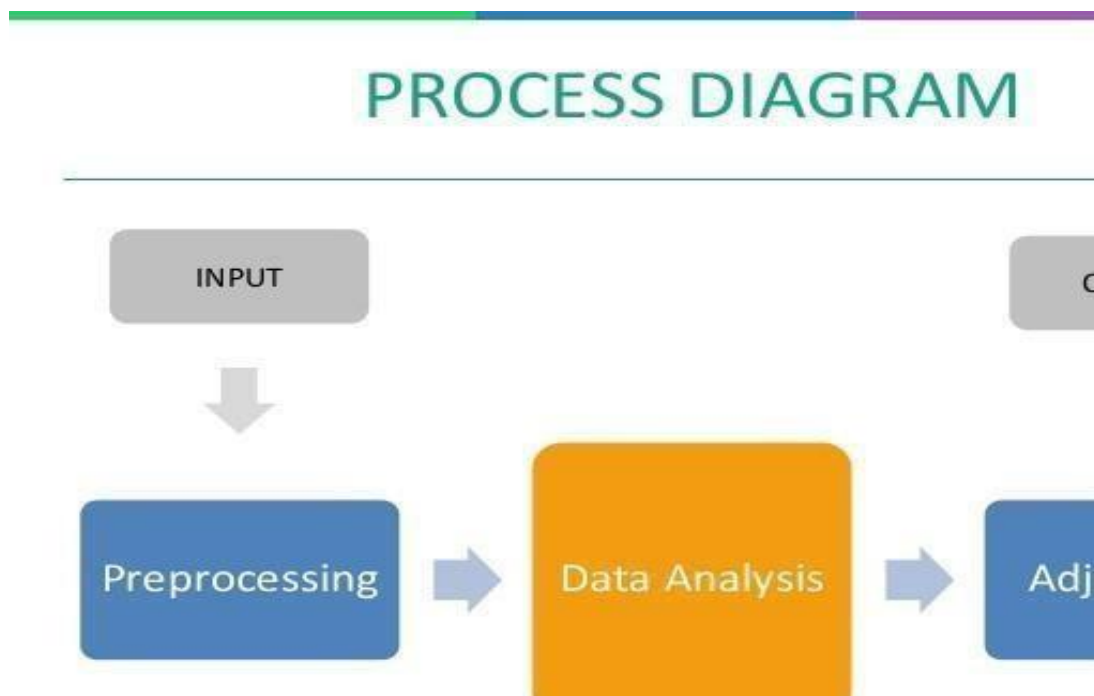


Figure 3.2(Data Mining)

#### 3.3.1 Pre- processing

**i. Expulsion of Stop Words:** There isn't one distinct rundown of stop words which all devices utilize and such a channel isn't constantly utilized. Any gathering of words taxi be picked as the stop words for a given reason. By expelling the stop words amid information pre-

processing we lessen the computational intricacy of the program and thus the task can keep running in a compelling way.

**ii. Convert each character to lowercase:** This progression is done with the end goal to expel the refinement between similar words written in upper and lower case, so demonstrate doesn't treat them in an unexpected way.

**iii. Sentence level handling/tokenization:** sentences are tokenized into the words and some all the more pre-processing are connected after that.

### 3.4 Feature Extraction

**i. Bag of Words:** The pack of words show is a rearranging portrayal utilized in characteristic dialect preparing and data retrieval (IR). In this model, a content, (for example, sentence or a report) is spoken to as the pack dismissing language and even word arrange yet keeping assortment. The sack of words demonstrate is generally utilized in techniques for record characterization where the recurrence or event of each word is utilized as a component for preparing a classifier. It is equivalent to the skip gram model of the unigram in the dialect display.

**ii. Feature Vector Creation:** Feature Vector Creation is the procedure of change of pack of words model of into the vector shape where by every word are spoken to with the frequencies. For highlight vector creation at first a vocabulary is manufactured utilizing the majority of the corpus accessible in dataset which makes a vector space display for words and highlight vector are gotten from the every corpus in like manner.

### 3.5 Classification for recommender frameworks

**i. User profile:** a user profile is required to know about user interest and taste from their listening history.

**ii. History based:** listening history of a particular user and frequency(polycounts) can help predict the popular and personalised recommendations

**iii. Vector space:** likelihood of songs is computed by frequencies using vector space

**iv. Location:** location of a person has a great impact on his choice preferences. Languages also differs with the location

**v. User-thing evaluations framework:** Some community oriented separating frameworks keep up a client thing appraisals lattice as a piece of the client profile. The client thing evaluations grid contains verifiable client appraisals on things. A large portion of these frameworks don't utilize a profile learning strategy.

### 3.6 Profile generation

**i. New user:** when the person is new to the interface. We don't have anything to predict with.

**ii. Manually created:** when person creates profile and starts listening to.

**iii. Related info:** when we have all the data about age group, gender, location, language.

**iv. Training set:** giving the clients a few from them can select

**v. Profile learning procedure:** time to time after user had listened lot of song the profile automatically changes.

**vi. Not required:** sometimes profiles is not required .We only recommend most popular songs.

**vii. Clustering:** toward gathering data objects with respect to some regular highlights acquired to its data setting. Client profiles are frequently bunched with the end goal to bunch as indicated by some standard. To survey which clients share basic interests. Recommenders like Last.fm<sup>3</sup> or iRate<sup>4</sup> play out this strategy

**vii. Classifiers:** classifying songs on the basis of genre, language, locations etc by using various machine learning classification techniques and then classifying them into groups.

In this project we will learn how to build a music recommendation system using real data. Our million songs data set contains two files: Triplet \_ file and metadata \_ file. The triplet file contains user \_ id, song \_ id and listen time. The meta data file consists of song \_ id, title, release \_ by and artist \_ name. Our first job is to integrate our data set which is essential as we want to construct a data processing pipeline. To integrate both triplet and metadata files we will use the popular python library pandas.

We firstly define the two files we will work on:

```
Triplets _ file = 'https://static.turi.com/datasets/millionsong/10000.txt'
```

```
songs_ metadata_ file = 'https://static.turi.com/datasets/millionsong/song_data.csv'
```

We will now read the table of triplet using pandas and then define the three columns as user \_ id, song \_ id, listen \_ count.

```
song_df_1 = pandas. read _table(triplets _file ,header =None)  
song_df_1.columns = ['user _id', 'song _id', 'listen _count']
```

Now we will read the metadata file and then combine the metadata file with the triplet file. Whenever we will combine two or more datasets then there will be duplicate columns. So we will drop the duplicates using song\_id.

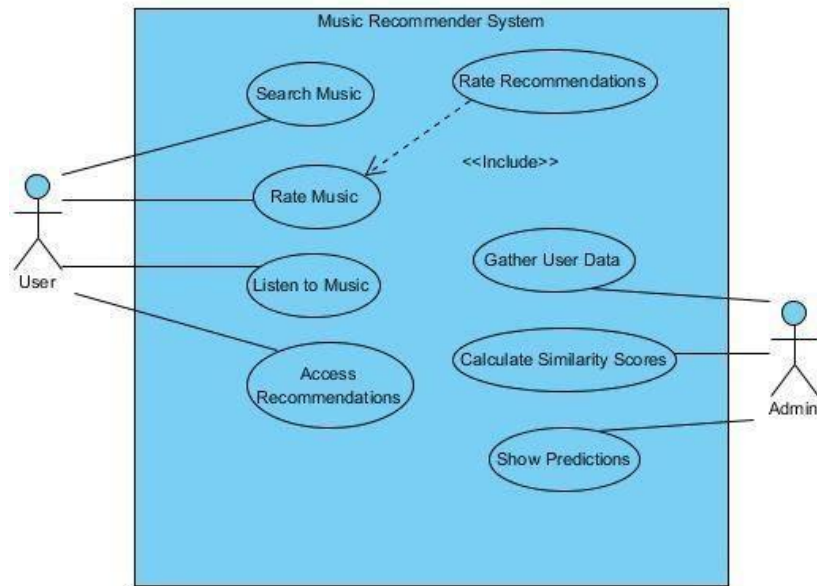
```
song_df_2 = pandas .read _ csv(songs _metadata _file)  
song _ df = pandas. merge(song_df_1, song_df_2.drop_duplicates(['song _id']),  
on="song _id", how="left")
```

### **UML Implementation of the System:**

Before going for the designing the structure and behavior of the system, in software engineering, requirements are gathered and analysed. Here the requirement analysing involves the context and scenarios where the system is going to be used. The user scenarios are used for constructing the Use case[7] diagrams. They basically depict the specific functionalities provided by the system to its users.

### **Use case diagram:**

Here the actors are the user and the admin. Each use case comes with the description, actors, pre/post conditions and the flow of control. Here the admin is the one that maintains the application.



**Fig1: Use case diagram for Music Recommender System**

Actors	Use cases	Description
User	Search Music	Querying the system
	Rate Music, Recommendations	Convey the liking of the music
	Listen to Music	Play/ Stop music
	Access Recommendations	Get the recommended songs/artists
Admin	Gather User Data	Log user activities, Listen count etc.
	Calculate Similarity Scores	Apply the pearson correlation
	Show Predictions	Recommendfrom prediction scores

Coming to the conditions on the use cases, the implicit one is that the user must be logged into the system to access recommendations. Flow of control would be the inherent activities of each use case like the processing of the request as soon as the user enters a query for searching the music, the compiling of the list generated from the scores and showing those with high values etc.

### Activity Diagram:

Main activities for each of the service provided are described by an activity diagram. This activity diagrams deal with the workflows of the recommender system. The work flows are depicted in sequence and often have conditions specified on the control-flow lines. The diagram is accompanied by the description, the initiator of the activity and the workflow. The initiator is generally a function module that gets called when an activity starts. Some of the functions in this music recommender system implementation are:

- `initMusicRecSys(...)` – create new instance of the system
- `loadUserProfile(...)` – authenticates and loads a user profile
- `processRequest(...)` – analyses the query entered by the user
- `dispResults()` – get the results and display them
- `logUserActivity(...)` – playcount, timestamp etc are all logged for further calculation
- `getRecommendation(...)` – show the recommendations after calculating similarity and prediction scores



- logExplicitRating(...) – user rating to specific song is noted for further calculations

Note that these are all parameterized methods and will be known during the implementation stage and are being omitted here.

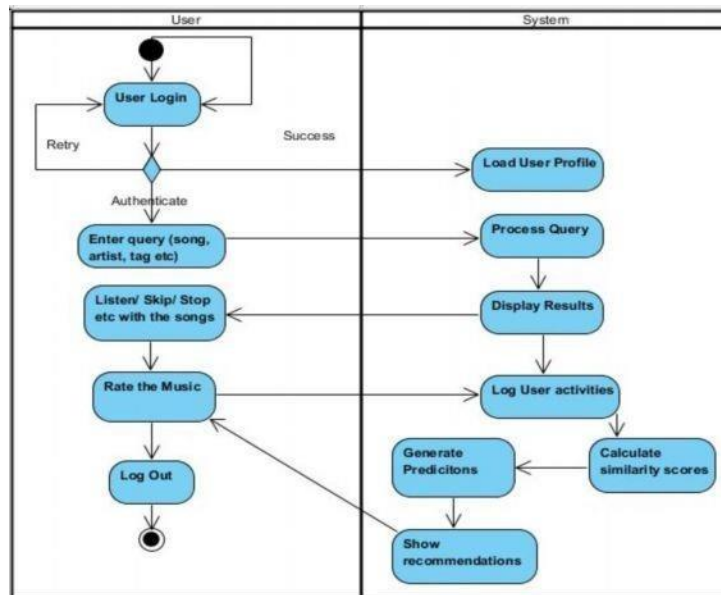


Fig2: Activity diagram for Music Recommender System

### Class Diagram:

The class diagrams describe the structure of the system being modelled. They are the building blocks so to speak for object oriented modelling as with them comes all the object oriented concepts that exist among various individual components of the system. The three compartment figure of classes holds the name, the list of attributes and the operations.

In the current music recommender system being modelled, there are three main classes: the User, the Artist and the Song. The additional two classes Similarity and Recs are for calculating similarity scores and getting recommendations of songs and artists. These classes are just depicting the business logic part of the system and not the user interface. The Fig3 depicts the multiplicity also which tells the number of instances of one class referenced by the other in an association relationship:

- Each user is entitled to 0 or more recommendations of songs or artists.
- Each recommendation is based on the rating of the song by user and calculating similarity score for other with this song.

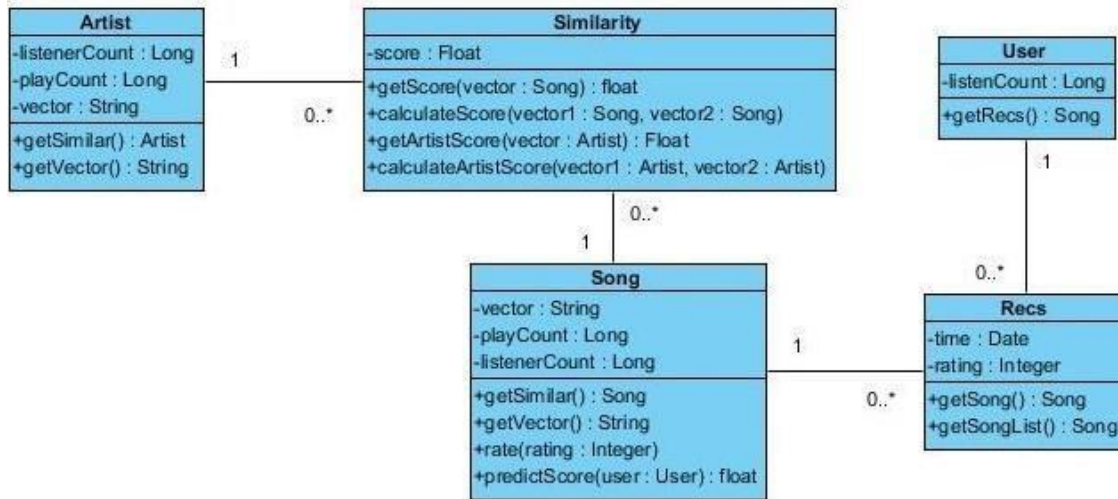


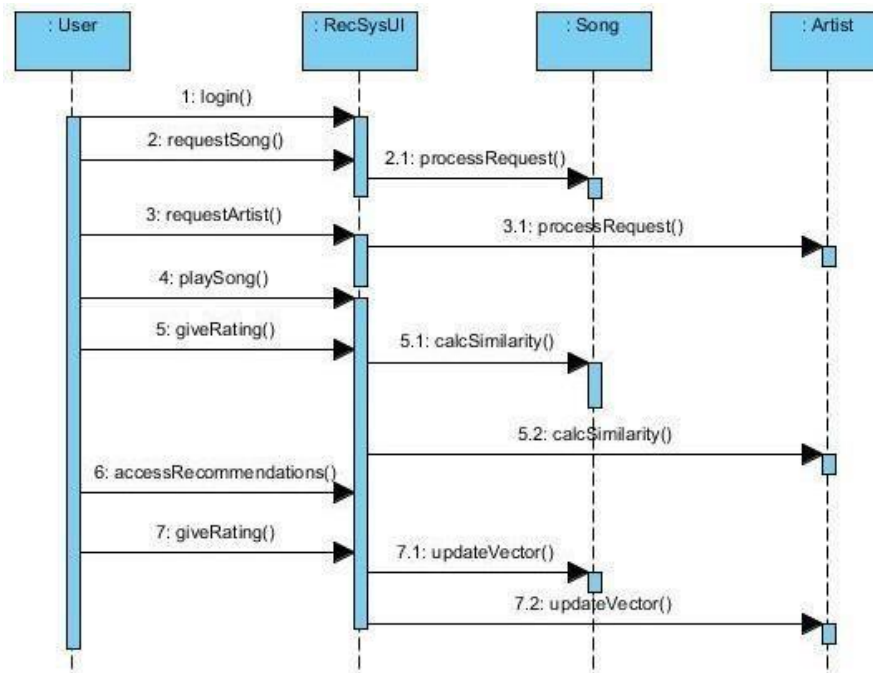
Fig3: Class diagram for Music Recommender System

- For Each song there may be 0 or more number of ratings.
- Also, for each song there will be 0 or more number of similar songs.
- For each artist there will be 0 or more similar artists.

### Sequence Diagram:

The sequence diagrams are one of the interaction diagrams that depict the communication between the objects. The collaboration of objects is modelled based on a time sequence. The objects involved in the scenario pass messages between themselves. Here the return messages are not shown but are to be understood as implicit. The diagram below is basically asynchronous way of communication as in the recommender system the similarity scores are to be computed asynchronously without the user having to wait for long.

Fig4: Sequence diagram for Music Recommender System



The RecSysUI object comes into picture here as it is the one with which the user communicates. The RecSysUI class has many visual elements that contribute to the data source in terms of user activity. The search button, the rating elements, the recommendation display section, the song player and play/stop buttons etc., all contribute to the logging of user activity. Since only behavioral nature of the recommender system is being discussed in the sequence diagrams, the user interface elements are abstracted out into a single class.

### 3.7 Combined Data Set

Using the command `song _ df .head()` we can visualize the combined data set

	user_id	song_id	listen_count	title
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1	The Cove
1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBBMDR12A8C13253B	2	Entre Dos Aguas
2	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBXHDL12A81C204C0	1	Stronger
3	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBYHAJ12A6701BF1D	1	Constellatic

Table3.1 (Combined data set)

Here we have the song index, user\_ id, song\_ id, listen \_count, title, release and artist \_ name. The second step is data transformation as we will select a subset of this data set. We will then merge the song and artist\_ name in one column.

### 3.8 Data Set after Transformation

The first line in the code below will group the song \_ df by number of listen \_ count. The second line will calculate the group \_sum by adding the listen \_count of each song. The third line will add a new column called percentage by dividing the listen \_count by the sum of listen \_count of all songs. Then multiply by 100. The last line will list song according to popularity in ascending order.

```
Song _grouped = song _df. Group by(['song']).agg({'listen _count': 'count'}).reset _index()
grouped _sum = song _grouped['listen _count'].sum()
song _grouped['percentage'] = song _grouped['listen _count'].div(grouped _sum)*100 song
_grouped .sort _values(['listen _count', 'song'], ascending = [0,1])
```

Table 3.2(Data set after Transformation)

	<b>song</b>	<b>listen_co</b>
<b>3660</b>	Sehr kosmisch - Harmonia	45
<b>4678</b>	Undo - Björk	32
<b>5105</b>	You're The One - Dwight Yoakam	32
<b>1071</b>	Dog Days Are Over (Radio Edit) - Florence + Th...	28
<b>3655</b>	Secrets - OneRepublic	28
<b>4378</b>	The Scientist - Coldplay	27

## 4. PERFORMANCE ANALYSIS

After the completing all the data pre-processing and making the dataset suitable for creating the model it is time to think about the algorithms to be used for this.

The most common algorithm are:

4.1 Content Based Filtering

4.2 Collaborative Filtering

4.3 Clustering And Classification Algorithm

4.4 Hybrid Approach

**4.1 Content based Filtering:** content based filtering is the most popular algorithm when it comes to recommender systems. It is common sense that if a user listens to a song and another song is similar to that one then user may also like that also. Basic idea behind this is to predict on the basis of similarity

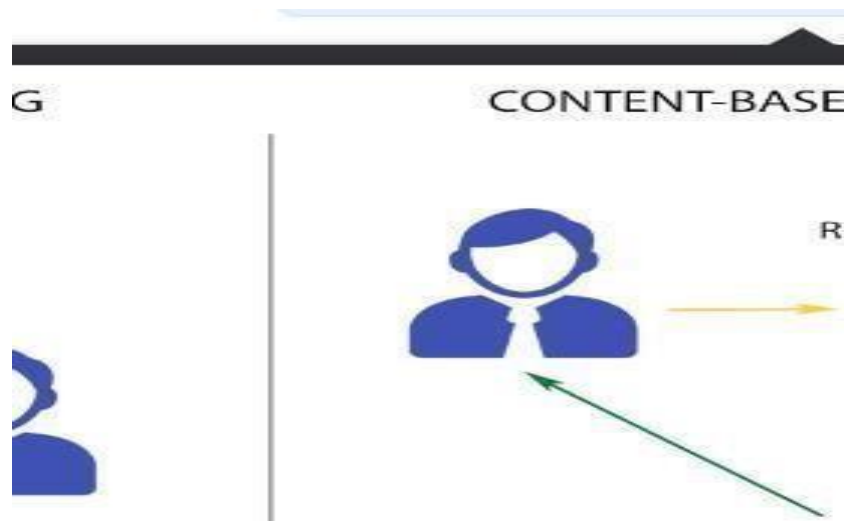


Figure 4.1(Content Based Filtering)

**i. Item Profile Creation:** Here initially, item profile is created in order with the help of it's feature.

In case of movie, music meta data available can be used for item profile creation.

**ii. User Profile Creation:** User profile is created , based on their interaction with the item for example with the help of their rating on the items. Hence user profile is created with the help of the item profile either by taking the average of item-profile or weighted average of item-profile.

Content based filtering can outperform the collaborative, whenever the ratio of item to user is very high.

Content-based filtering is also commonly known by the name of cognitive filtering because it recommends and suggests items which are based on comparisons between the materials of an item or several items and the user profile or multiple user profiles. The content of each item is shown and depicted as a set of terms and parameters predefined , specifically the words and statements that occur in the particular document. The user profile is also clearly shows by similar terms and conditions and also by studying the contents of items that are viewed and rated by a particular user or customer.

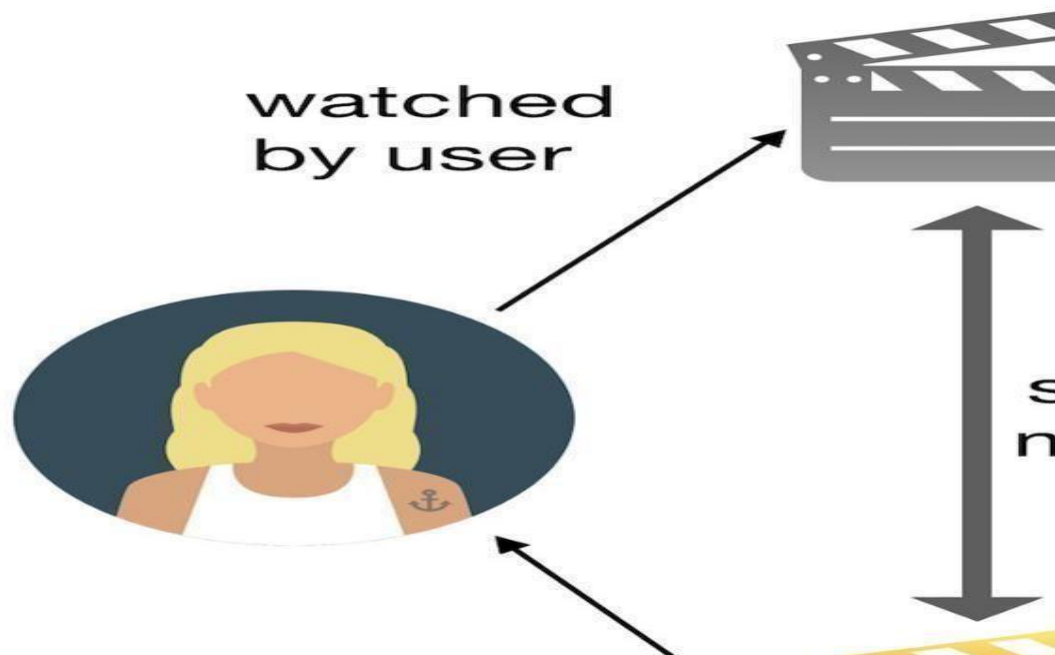


Figure 4.2(Content based filtering)

Several constraints have been taken into account while implementing content –based filtering method. The term can be assigned manually or automatically. The information sources that the content based filtering system use are mostly text documents.

The learning strategies applied to content based filtering basically find the most important and relevant documents depending upon a user and the behavior of a user.

A recommender system is capable and efficient enough to pickup and decide between two kinds of information or multiple types of information in the form of documents while delivering or providing a user with certain recommendations and more precisely before recommending it to a user or client:

- i. **Exploitation** – In this the recommender system will chooses the material in the form of documents which are more relevant and important to the material or content that the user has already given a preference or even a recommendation.
- ii. **Exploration** – On the other hand in this the recommender system will choose the material provided in the form of documents in case the user does not provide any prediction or suggestion to access the users choice.

#### **4.1.1 Advantages of content based filtering**

- i. Results are highly relevant
- ii. Recommendations are transparent
- iii. Users can get started quickly
- iv. New items can be recommended immediately
- v. Technically easier to implement

#### **4.1.2 Disadvantages of content based filtering**

- i. Limited analysis of content and data
- ii. Over-Specialization providing limited degree of novelty
- iii. Insufficient information of user for profile creation



## 4.2 Collaborative-filtering

Collaborative-filtering is commonly known as a social filtering process in which it generally filters the content and information by the use of recommendations of other users in similar domain. The basic idea behind this technique is that users who agree and are interested in the evaluation of a specific item in previous history or purchase in the form of use, access and purchase are more likely to agree or match again in future. A user willing to listen particular music for example can ask or search for recommendation from friends and people having similar tastes and interests.

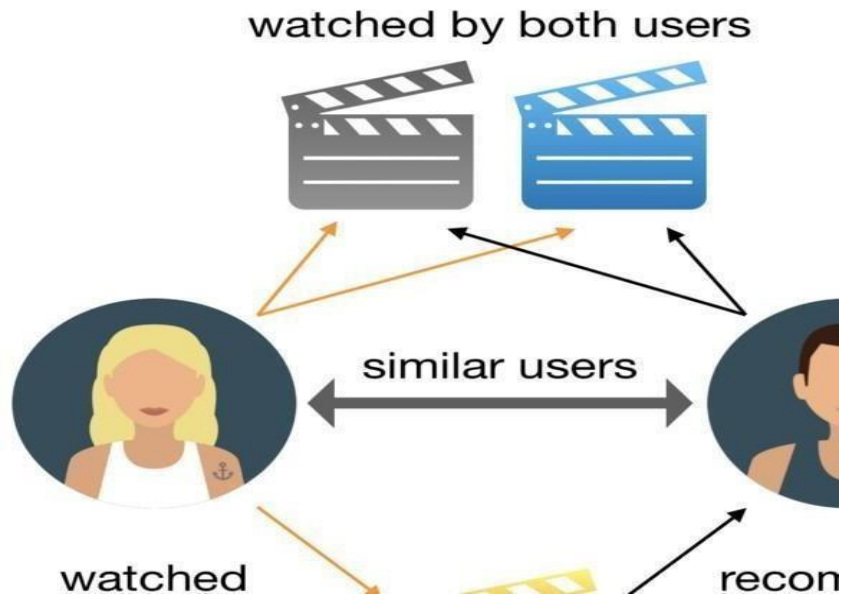


Figure 4.3(Collaborative filtering)

**4.2.1 User based collaborative:** If two users like similar songs then we can recommend the liked songs of one to another. In this project we can predict the collection of one user to another based on their tastes.

Consider that we want to recommend a song to our friend. We can assume that similar people can have similar tastes and interests. Suppose that I and my friend have seen the similar songs and we rated those movies almost equally. If I liked a particular song it is most likely that he

will too like that particular song. User based – collaborative filtering uses this logic and recommends songs by finding similar users and then suggests similar songs.

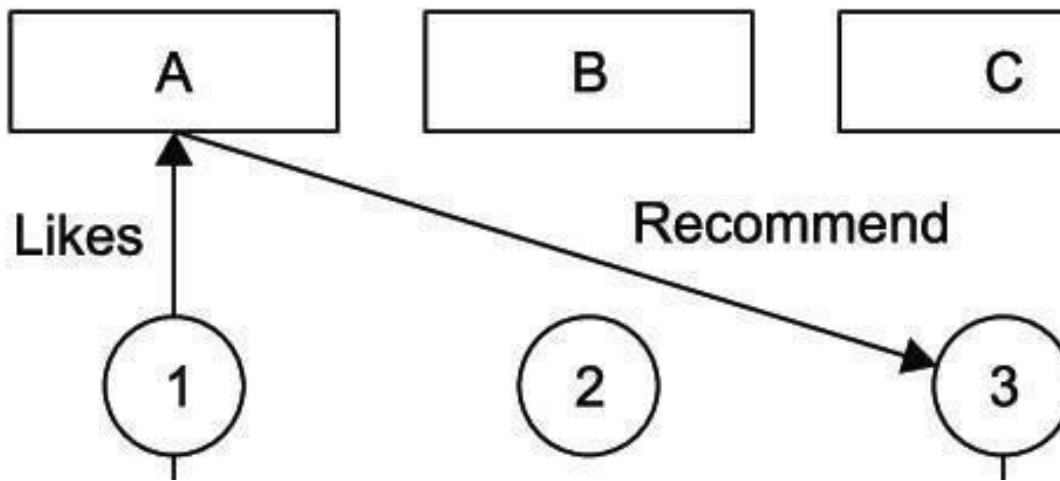


Figure 4.4(User based collaborative filtering)

With a user based approach in prediction analysis, the system can calculate similarity between pairs of items of users by using the cosine function. Usually such approach of calculation take longer time frame and also require to be computed than those in item based approach. That is because:

You have a lot more users than you have items

You will expect items to change less frequently than the users

With more number of users less change in the items , you can use many more attributes than purchasing history while calculating user similarity

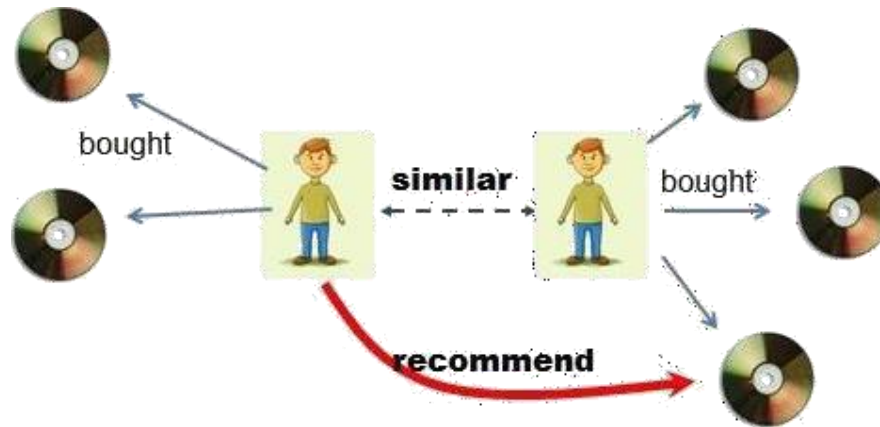


Figure 4.5(User based collaborative filtering

#### 4.2.1.1 Advantages Of User Based Collaborative Filtering

- i. Context is independent
- ii. Compared to other methods, such as content-based, it is more accurate.
- iii. Easy to implement

#### 4.2.1.2 Disadvantages Of User Based Collaborative Filtering

- i. Sparsity - Percentage of users who rate items is really low
- ii. Scalability -The more users in system, the greater the cost of finding nearest neighbour

**4.2.2 Item based collaborative:** If user like some songs and there is another song similar to others then we can suggest that song to user. In item based collaborative filtering we generally search for the items which have similar or identical domains , features and attributes to the article or items that a user has already used in past and has rated and recommended most similar items . So what does this similarity mean in this particular case? In this particular case we do not mean that two items are the same by attributes as in sedan car and hatchback car are similar because both of them are cars. Both the sedan and the hatchback are categories of cars but cannot be considered similar on the basis of this common domain.

This methodology is more effective as well as stable when particularly compared to the user-based collaborative filtering because the mean items generally have a lot more ratings and

reviews than the average user has because of more access of those items widely. So in this case the user rating will not have an impact because of the unpopularity or less number of rating.

To calculate and analyse similar attributes between two items or more than two items , we will look into the given set of items that the target user or client has suggested and rated and then compute how much similarity there is between the target and then select k most similar items. Here the target item and target client can be defined as the item particularly focused on which recommendation has to be done. Whereas, the target client is a service user who has to be provided with optimum suggests and preferences .Similar attributes between two or more items is calculated by taking the ratings into considerations of the users who have previously rated two or more items and then use the cosine function of similarity.

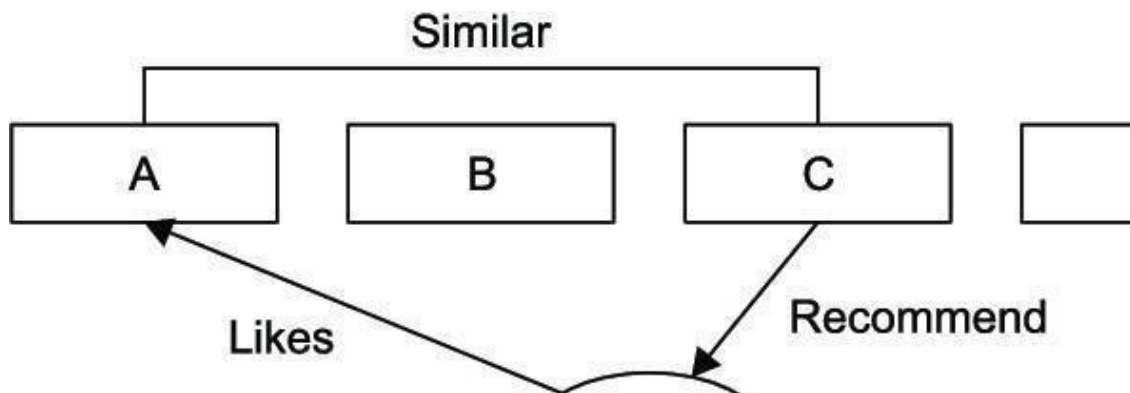


Figure 4.6(Item based collaborative filtering)

Considering the similar example of our friend, instead of focusing on our friend , we can focus on the items from various options which are more similar to what our friend likes. This new shift of focus is known as Item-based collaborative filtering.

This methodology is more stable as well as effective in itself in comparison to user- based collaborative filtering as the mean or average items have higher ratings than the average user. This enables it to effectively perform in recommending any particular set of itmes to a specific user or client.

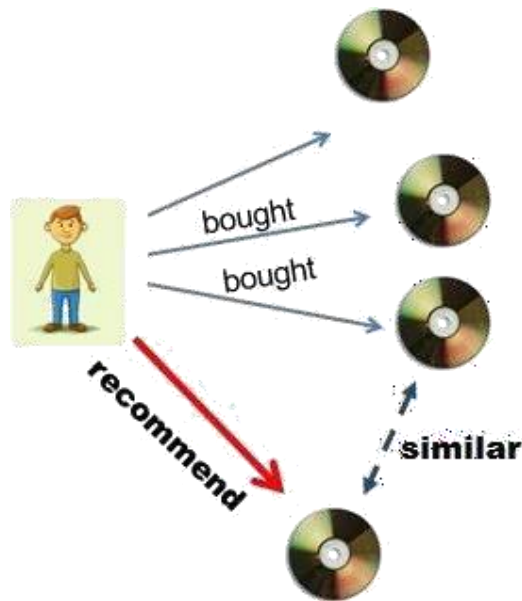


Figure 4.7(Item based collaborative filtering)

#### 4.3.1 K means clustering

The k means is a generic and easy to implement clustering-algorithm with minimal complications and complexities. In clustering algorithms the data is accepted and taken as input data as input and then it will uses the mathematical precision of calculations and methods to search particular groups which are of similar items as predefined as input or users as using that particular data as predefined in input. For example supposing we are provided a set of 6 people having age group as follows: 4,7,17,27,46,48. This information defines number of persons and the ages of those particular persons .If we are asked to split that group into sets, generally one will split the given data into minors (5,7,17) and non-minors(44 and 68).This type of separation or classification is based on a factor which is age .But we cannot call this clustering but this what exactly is being done: clustering like with similar based on the data set given. Clustering-algorithms such as k-means algorithm generally perform the same thing as defined earlier, but with huge amount of data at a large amount of scale. It is a good and easy method to arrange and augment.

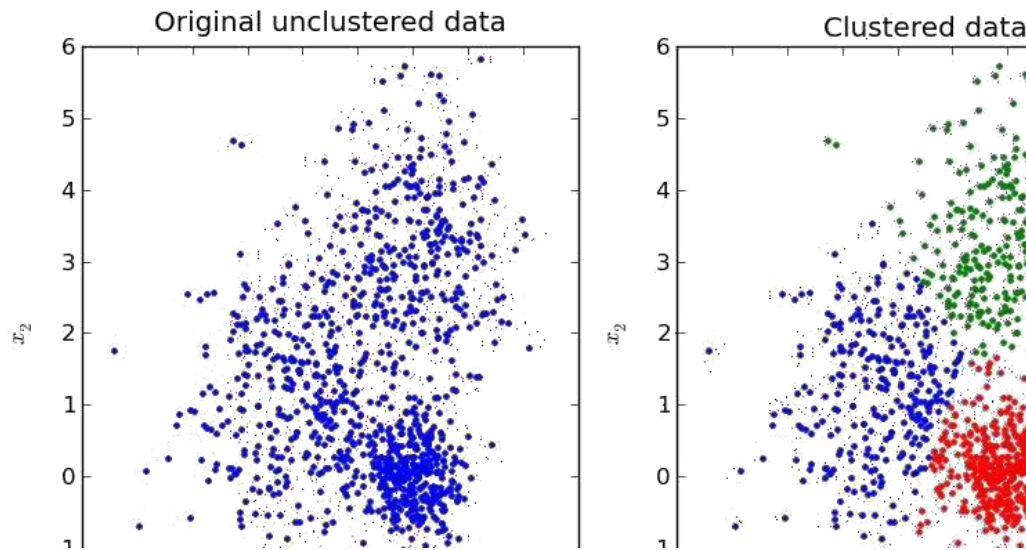


Figure 4.8(Clustering data)

This algorithm aims to group together various songs or various users based on various attributes .this first finds the similarity between objects and then group them together .These groups are named as clusters. One thing about this algorithm is that previously unknown clusters or groups can be discovered using this. Nearest cluster for each object is calculated and then they are assigned the nearest cluster.

In k means clustering we are provided with a set of data items relevant or irrelevant to the domain, with specific attributes. Our primary goal is to arrange and categorize these set of items into groups based on given constraints and parameters. For us to attain this, we will have to use the k-Means algorithm to obtain required and optimum results .

The algorithm will make category of the items into k groups of similarity.

This algorithm will works as in the given order:

Firstly initializing of k points will be done, which is done uncertainly or randomly. Further we will the categorize all items to the closest mean and then update the mean's coordinates, which are the averages of items categorized in that mean so far.

We the repeat the process for any given number of iterations and then at end, we have our clusters.

#### **4.3.1.1 Process**

We are describing the process to show how it works. We take care of all with only a single line of code. In order to keep the procedure straight and simple, we will now consider 30 users in a given Training set or given set of Class, each user will have 2 or more than 2 scores. We now will create separate non identical clusters based on the scores of the first tests performed.

Now this is how the k-means algorithm will work with this particular setup:

It will firstly randomly it will take 2 or more than 2 different points that will serve as initial cluster.

Then it will calculate the distance between each of the given 30 data points from the given each centre.

Now it will allot each of the data points to the cluster with the closest centre.

Now recalculate each of given centres of clusters using just those given data points that were assigned to cluster.

Recalculate distance of each of 20 data points from each of new cluster centre.

It will then reassign each of the data points to the cluster and then with closest centre.

Further it will continue repeating process till no such data points need to be assigned again even after the calculation of the new cluster centres has been done.

#### **4.3.1.2 Does k- means produces optimum result?**

No, it will not but normally it works optimum.

The primary stage is randomly choosing starting cluster centre locations. This means that the received output can sometimes be affected and changed to some degree level by that particular starting point. In some rare cases, the outcome can be substandard to other clustering outputs. This means all clustering outputs will not necessarily be same or identical. In practice, we should firstly set up a random input and then reproduce the result. Moreover, in formation of this post, very commonly we receive undesirable results. The method to obtain desired results was to set up random seeds or input.

### 4.3.1.3 Algorithm

number defined at the beginning of the algorithm. The centroids of the clusters are updated and the process is repeated in order to get a better result of the data points which they belong <sup>[27]</sup>. The algorithm pseudocode is shortly described below.

1. Choose a value of  $k$ .
2. Select  $k$  objects in an arbitrary fashion. Use them as initial centroids.

### 4.3.1.4 Advantages Of K-Means Clustering

Quick, robust and easy to analyse for effective implementation. Relatively it is more effective and optimum.

Gives more desirable results when the data set is different or well separated from one another.

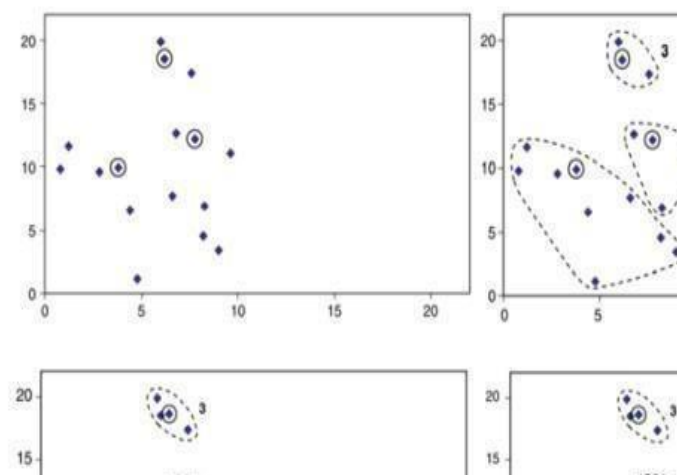
### 4.3.1.5 Disadvantages Of K-Means Clustering

Requires appropriate specifications of the number of cluster centres.

In case there are two or more highly overlapping sets of data, then the k-means will not be able to resolve it.

It is applicable only when the mean is known.

and closer to the center of the cluster from one iteration to another.



4.9 ( K means clustering diagram)



### 4.3.2 Logistic Regression

Given a training set having one or more independent (input) variables where each input set belongs to one of predefined classes (categories), what logistic regression model tries to do is come up with a probability function that gives the probability for the given input set to belong to one of those classes

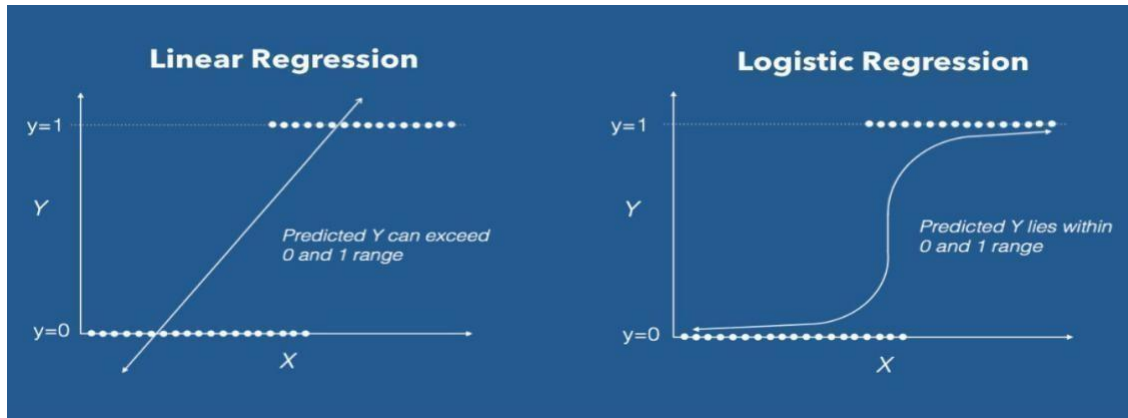


Figure 4.10(Logistic Regression)

The basic logistic regression model is a binary classifier (having only 2 classes), i.e., it gives the probability of an input set to belong to one class instead of the other. If the probability is less than 0.5, we can predict the inputs set to belong to the latter class. But logistic regression can be hacked to work for multi-class classification problem as well by using concepts like “one vs. rest”. What we basically do is create a classifier for each class that predicts the probability of an input set to belong to that particular class instead of all other classes. It is popular because it is a relatively simple algorithm that performs very well on wide range of problem domains

The logistic function (also known as sigmoid function) is defined as

The name “*logistic*” comes from the probability fu  
function (also known as *sigmoid* function) is defir

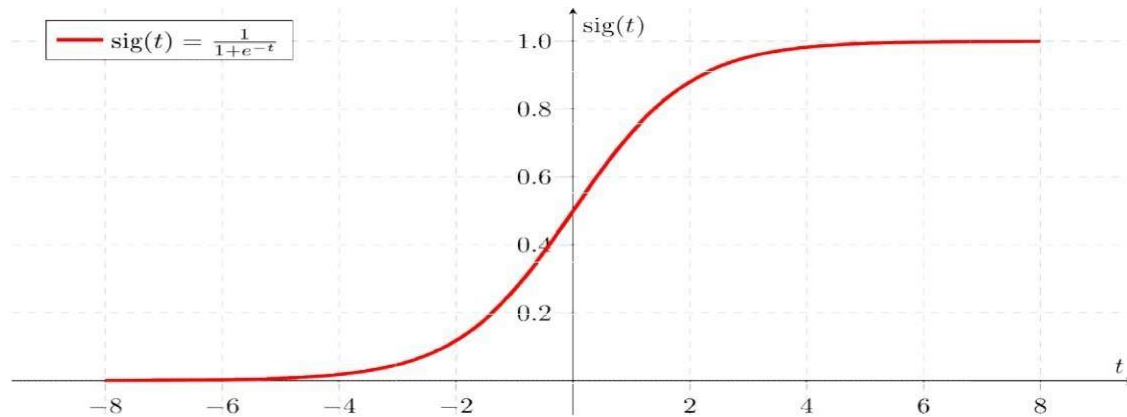


Figure 4.11(Sigmoid function)

Let us consider that we have  $N$  users,  $M$  songs, and  $K$  features per song. For each user, we can define a feature vector with  $M \times K$  entries. When we want to predict or recommend interaction for any pair  $(u, m)$  of user  $u$  and song  $m$ , one feature with indices in  $[mK, (m+1)K]$  will be on. The other  $M(K-1)$  features will be set to zero. This will enable us to pack  $M$  separate logistic regression models into one single logistic regression model.

In our case, for each movie  $m$ , we will use the interaction with the  $M-1$  other songs as the features. Thus  $K=M-1$  and our vector will have  $M(M-1)$  entries. This should immediately raise concern about memory usage – the number of features will scale quadratic with the number of movies. In fact, this is one of the reasons why we should not train separate model for each song in the first place.

This is where feature hashing comes into play. The interactions are very sparse. A small percentage of the  $M-1$  features for each song are likely to be on. Instead of having a vector of  $M(M-1)$  entries for each user-song pair  $(u, m)$ , we can create a small vector of  $2^B$  entries.

#### 4.3.3 Random Forests

Random forest is a non rigid and easily used and implement machine –learning type of algorithm.

In this algorithm we will not even use the hyper parameter tuning, still we will obtain a good result and obtained most of the time. It is also one of the most frequently used algorithms because it is simple and also it uses both classification and regression tasks. The random forest is a controlled type of learning algorithm as it creates a forest and makes it random. This random forest creation enables multiple options and decisions. In simple

definition random forest builds multiple decision trees and then merges them to obtain a more optimum and accurate as well as stable prediction.

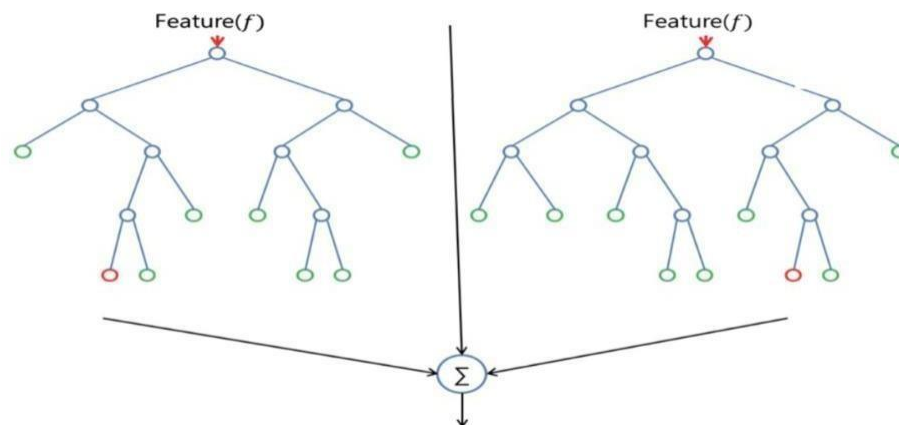


Figure 4.12(Random Forest)

Random forest is a decision making algorithm which makes prediction on the basis of various attributes function at nodes is computed and then best node is predicted. The random forest is a compatible and commonly used machine learning type of algorithm that gives good results most of the time. It creates a forest and then makes it random. In simple words random forests creates multiple decision trees and merges them together to obtain more accurate and optimum predictions.

One of the major benefit and or strongpoint of this particular technique are that it can use both the classification as well as the regression. This generally forms the majority of the machine learning systems. Random forests have nearly similar hyper-parameters as a decision tree. This means it has certain level of similarity with the decision tree. Moreover it instead of searching for the most important feature or attribute while splitting of the node, it searches for the most important feature among a random subset of features or attributes. This means that it focuses on the best and optimum attribute and not the most vital. Random inputs

are categorized and divided into subsets. This results in a wide diversity that generally results in a better model. Therefore in the random forest algorithm, only the random subsets are taken into consideration by the algorithm for splitting of the data .

### Real life analogy

Consider a user or person for understanding that wants to confirm and decide which places that person should travel in various parts of the world or a country or even a state during fixed period of time .He will firstly ask people who know the person for their suggestion and advice. Firstly the person goes to his friend and asks about the travel history and experience

.This is a typical decision tree algorithm .In such case the person's friend created rules to guide and direct his decisions and choices about what the particular person should recommend. Similarly after asking several friends the person derives recommendations. Then the person chooses places to travel the places that were suggested by most people, which is the typical random forest approach.

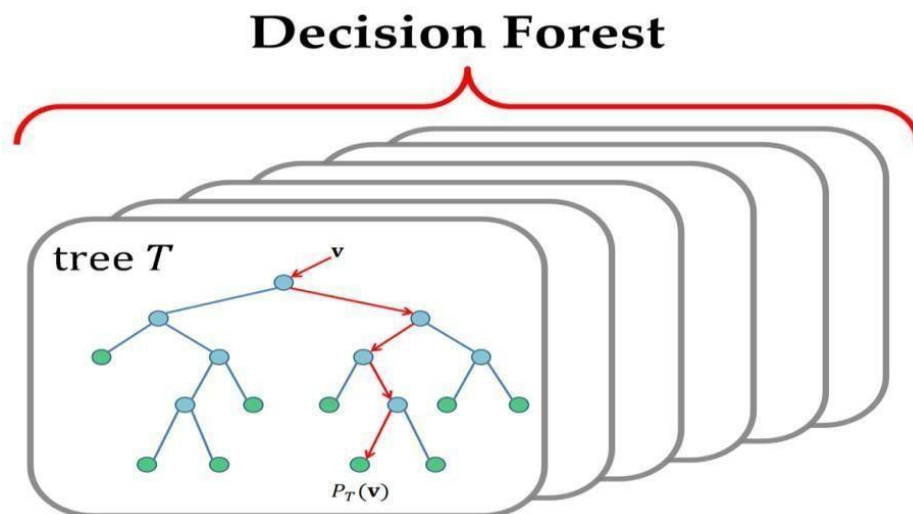


Figure 4.13(Decision Forest)

## Advantages Of Random Forest

The predictive performance can compete with the best supervised learning algorithms  
Provide a reliable feature importance estimate

Offer efficient estimates of the test error without incurring the cost of repeating model training  
Associate with cross-validation

## Disadvantages Of Random Forest

An ensemble model is inherently less interpretable than an individual decision tree

Training a large number of deep trees can have high computational costs and use a lot of memory

Predictions are very slow which may create challenges for applications

## Algorithm for random forest:

---

### Algorithm 1 Random Forest

---

**Precondition:** A training set  $S := (x_1, y_1), \dots, (x_n, y_n)$ ,  
of trees in forest  $B$ .

```
1 function RANDOMFOREST( $S, F$ )  
2    $H \leftarrow \emptyset$   
3   for  $i \in 1, \dots, B$  do  
4      $S^{(i)} \leftarrow$  A bootstrap sample from  $S$   
5      $h_i \leftarrow$  RANDOMIZEDTREELEARN( $S^{(i)}, F$ )  
6      $H \leftarrow H \cup \{h_i\}$   
7   end for  
8   return  $H$   
9 end function  
10 function RANDOMIZEDTREELEARN( $S, F$ )
```

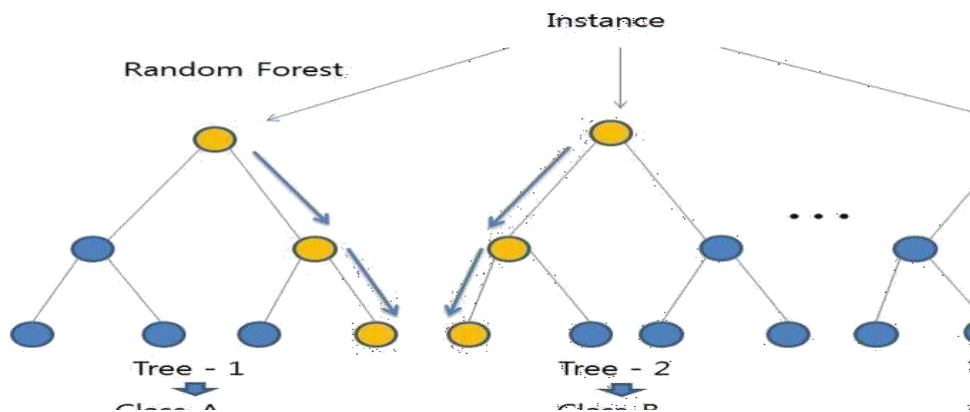


Figure 4.14(Random Forest)

#### 4.3.4 Decision Tree

Decision tree is commonly and easy to implement algorithm used all along. The Decision trees can be used both for the classification as well as regression of the problems. The Decision trees frequently copies and then follows the human standard of thinking so that it is simple to understand and make good interpretations of the given set of data . A decision tree effectively and efficiently makes us understand the concept behind for the data so as to interpret it and develop and produce optimum output. This means that a decision tree is defined as a tree in which each node will represent an attribute or a feature, also each of the links will depict a decision and each leaf will represent an outcome.

This is somewhat similar to random forest .It predicts decisions on the basis of gain and split ration and then best node is calculated. This acts as a tree further node selection takes place till the last node by same method.

Gain ratio is calculated as follows.  
Gain(A)= gain

$$gain = info(T) - \sum_{i=1}^s \frac{|T_i|}{|T|} info(T_i)$$

SplitInfo(A)= Split(T)

$$Split(T) = - \sum_{i=1}^s \left( \frac{|T_i|}{|T|} \right) \log_2 \left( \frac{|T_i|}{|T|} \right)$$

Figure 4.15(Gain Ratio)

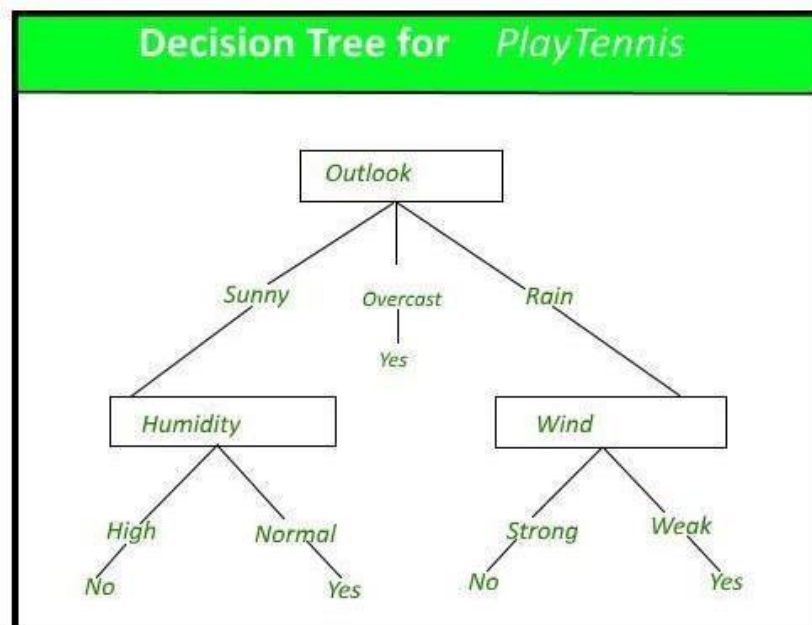


Figure 4.16(Decision Tree)

### **Advantages of Decision Tree**

- Generate rules which can be understood
- Perform the classification without the requiring much computation Handle both continuous and categorical variables
- Give clear indication of which fields are most vital for prediction

### **Disadvantages of Decision Tree**

- Less appropriate for the assessment of tasks where focus is predicting value of continued attributes
- Prone to errors in the classification of problems in case of many class and in case of small number of training
- Computationally expensive to perform



## Test plan

### 1. Data exploration

We have explored our data with the help of software and computed lot of things

Table 4.1

Field	count	mean
analysis_sample_rate	10000	2205
artist_7digitalid	10000	10954
artist_familiarity	9996	0.
artist_hottnesss	10000	0.
artist_latitude	3742	3
artist_longitude	3742	-63.
artist_playmeid	10000	2554
danceability	10000	0.
duration	10000	23
end_of_fade_in	10000	0.
energy	10000	0.
key	10000	5.
key_confidence	10000	0.
loudness	10000	-10.
mode	10000	0.
mode_confidence	10000	0

Table 4.2

year	10000	933
------	-------	-----

Table 1: Basic statistics on the 10

Field	count	mean	std
artist_familiarity	999815	0.557203	0.13
artist_hottnesss	999988	0.379813	0.1
artist_latitude	357492	38.999425	15.19
artist_longitude	357492	-58.370804	54.95
danceability	1000000	0	
duration	1000000	249.500755	126.22
energy	1000000	0	
key	1000000	5.321964	3.60

## Histograms for various attributes of the dataset

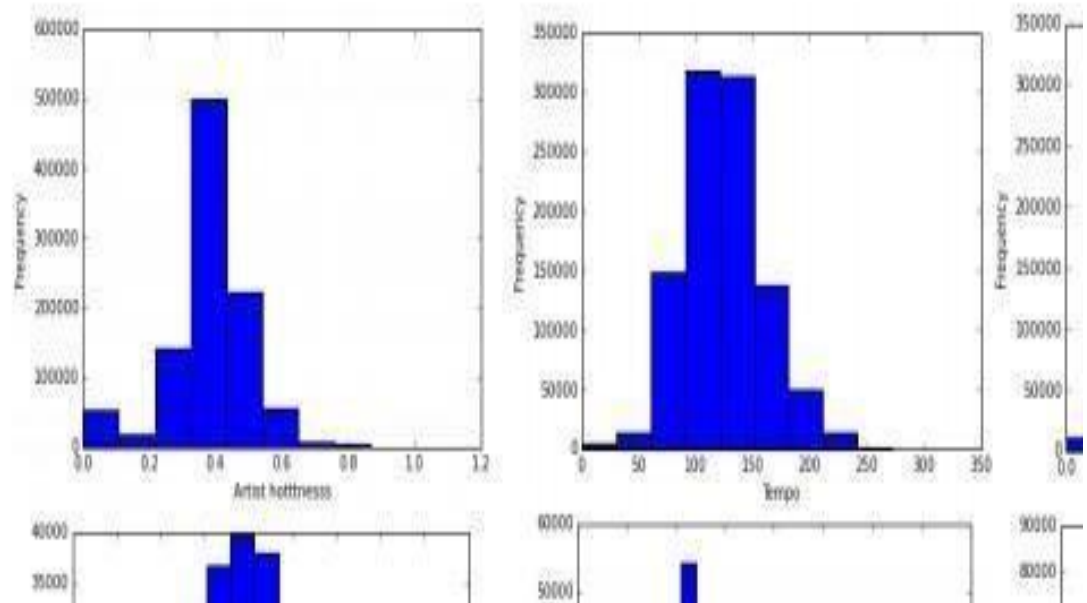


Figure 4.17(Histograms)

## Scatter plots:

We also plotted scatter plots to see examine the correlation between and our target class.

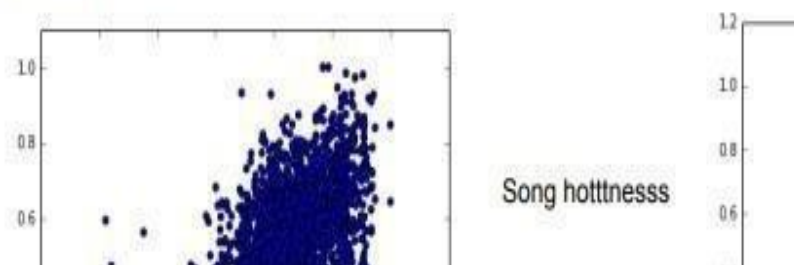


Figure 4.18(Scatter Plot)

### 4.3.5 Cross-validation

It is a method of model or design validation and to analyze and study how the results of a statistical data model will be generalized into an independent data set. This means it simply converts the data of statistical domain into generalized domain. It is most widely used in conditions where the primary goal is prediction, and one wants to see how accurately a predictive model will perform.

The primary objective of cross-validation is classification a data set so that the model can be tested in the training set. This will definitely reduce and limit problems such as under fitting, over fitting and so that we can get to know about how the model will generalize to independent data set. There are basically two types of data sets. One is the training set of data and other is the test data set. Cross validation is very useful for assessing the performance of machine learning model .When we are provided with a machine learning problem, we will be given two types of data sets –known data which is training data set and unknown data which is test data set.

In one round we will separate the original training data set into two parts:

Cross validating the training set

Cross validating testing set and validation set

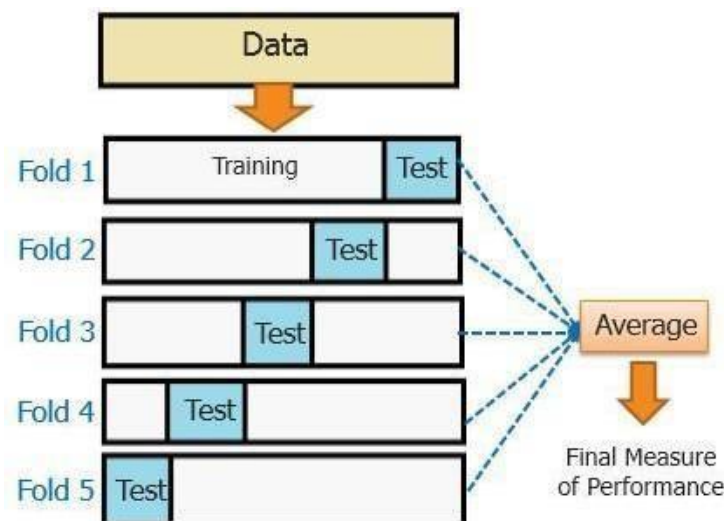


Figure 4.19(Cross Validation)

### What is over-fitting and under-fitting?

Under-fitting refers to not capturing enough patterns in the data. The model performs poorly both in the training set and the test set.

Over-fitting means capturing noise and capturing patterns which do not generalize well to unseen data. The model performs well in the training set but poorly in the test set.

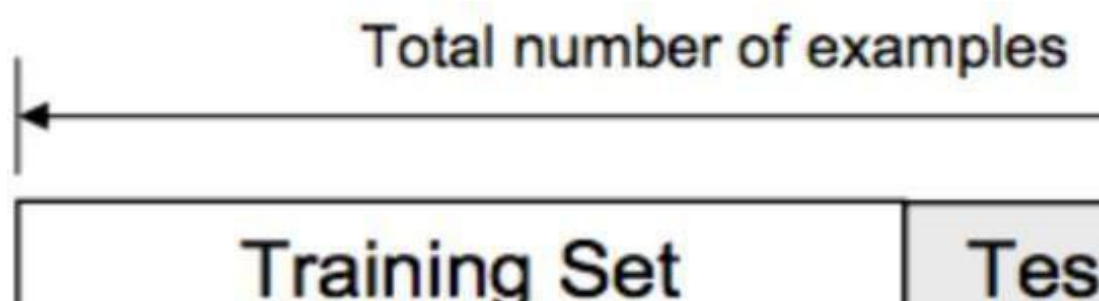


Figure 4.20(Training and Testing Set)

Cross validation is a technique which is used to improve the efficiency of a model by recursively interchanging the training dataset and testing dataset to train the model. In the first step if we have taken first half as train and second as test then we interchange them by making second half as train and first as test set to validate the model and train to get effective results .

### 4.3 K - Fold Cross-Validation

K-fold cross validation is a famous method of validating that is very often used in machine learning algorithms. One of the most famous and widely used cross validation technique is k fold cross validation . in this algorithm there's just a single attribute which is k which asks for number of break to be done on the dataset like if the value of n is 4 then this algorithm creates four sets of the dataset for cross validation.

K-fold cross validation is implemented as in the following steps:

- We will firstly Partition the given training data set into  $n$  number of equal subsets. Each of the subset is named a fold. we will name the folds as  $f-1, f-2, f-3, \dots, f-k$ .
- For  $j=1$  to  $j=k$
- We will keep the fold  $f-i$  as validation set and keep all remaining  $k-1$  folds in cross validation training set.
- Train machine learning model using cross validation training set and calculate training set and accuracy of model by validating predicted results against validation.
- Estimate accuracy of machine learning model by averaging by the accuracies derived in all  $k$  cases of cross validation.

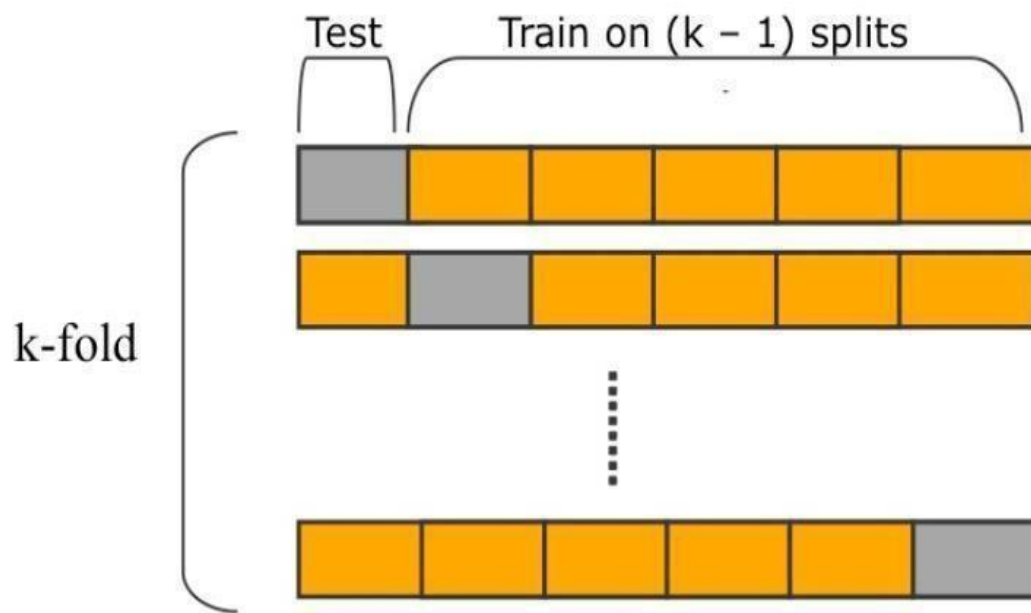


Figure 4.21(K-Fold Cross Validation)

**Let us see the algorithm to better understand**

```
1 Define sets of model parameter values to evaluate
2 for each parameter set do
3   for each resampling iteration do
4     Hold-out specific samples
5     [Optional] Pre-process the data
6     Fit the model on the remainder
7     Predict the hold-out samples
8   end
9   Calculate the average performance across hold-out predictions
10 end
11 Determine the optimal parameter set
12 Fit the final model to all the training data using the optimal parameter set
```

## Code Snippet

First we need to import pandas and numpy libraries

```
import numpy as np
import pandas
```

### sourcecode for popularity recommendation:

This model is used to recommend you songs which are popular or say, trending in your region. Basically this model works based by the songs which are popular among your region or listened by almost every user in the system

```
class popularity_recommender():
    def __init__(self):
        self.t_data = None
        self.u_id = None                    #ID of the
user                                       #ID of Song
        self.i_id = None                    #ID of Song
the user is listening to
        self.pop_recommendations = None    #getting
popularity recommendations according to that

#Create the system model
    def create_p(self, t_data, u_id, i_id):
        self.t_data = t_data
        self.u_id = u_id
        self.i_id = i_id
#Get the no. of times each song has been listened as
recommendation score
        t_data_grouped =
t_data.groupby([self.i_id]).agg({self.u_id:
'count'}).reset_index()
t_data_grouped.rename(columns = {'user_id':
'score'},inplace=True)

#Sort the songs based upon recommendation score
        t_data_sort = t_data_grouped.sort_values(['score',
self.i_id], ascending = [0,1])

#Generate a recommendation rank based upon score
        t_data_sort['Rank'] =
t_data_sort['score'].rank(ascending=0, method='first')

#Get the top 10 recommendations
        self.pop_recommendations = t_data_sort.head(10)
```

```

#Use the system model to give recommendations
def recommend_p(self, u_id):
    u_recommendations = self.pop_recommendations

    #Add user_id column for which the recommended songs are
    generated
    u_recommendations['user_id'] = u_id

    #Bring user_id column to the front

    cols = u_recommendations.columns.tolist()
    cols = cols[-1:] + cols[:-1]
    u_recommendations = u_recommendations[cols]

    return u_recommendations

```

#### **sourcecode for similarity recommendation:**

This model works according to the songs you listen to everyday.

```

#Class for Item similarity based Recommender System model
class similarity_recommender():
    def __init__(self):
        self.t_data = None
        self.u_id = None
        self.i_id = None
        self.co_matrix = None
        self.songs_dic = None
        self.rev_songs_dic = None
        self.i_similarity_recommendations = None

    #Get unique songs corresponding to a given user
    def get_u_items(self, u):
        u_data = self.t_data[self.t_data[self.u_id] == u]
        u_items = list(u_data[self.i_id].unique())

        return u_items

    #Get unique users for a given song
    def get_i_users(self, i):
        i_data = self.t_data[self.t_data[self.i_id] == i]
        i_users = set(i_data[self.u_id].unique())

```



```

        return i_users

#Get unique songs in the training data
def get_all_items_t_data(self):
    all_items = list(self.t_data[self.i_id].unique())

    return all_items

#Construct cooccurrence matrix
def construct_co_matrix(self, u_songs, a_songs):

    #Get users for all songs in user_songs.
    u_songs_users = []
    for i in range(0, len(u_songs)):
        u_songs_users.append(self.get_i_users(u_songs[i]))

    #Initialize the item cooccurrence matrix of size
len(user_songs) X len(songs)
    co_matrix = np.matrix(np.zeros(shape=(len(u_songs),
len(a_songs))), float)

    #Calculate similarity between songs listened by the user
and all unique songs in the training data
    for i in range(0, len(a_songs)):
        #Calculate unique listeners (users) of song (item) i
        songs_i_data = self.t_data[self.t_data[self.i_id] ==
a_songs[i]]
        users_i = set(songs_i_data[self.u_id].unique())

        for j in range(0, len(u_songs)):

            #Get unique listeners (users) of song (item) j
            users_j = u_songs_users[j]

            #Calculate the songs which are in common
listened by users i & j
            users_intersection =
users_i.intersection(users_j)

            #Calculate cooccurrence_matrix[i,j] as Jaccard
Index
            if len(users_intersection) != 0:
                #Calculate all the songs listened by i & j
                users_union = users_i.union(users_j)

                co_matrix[j,i] =
float(len(users_intersection))/float(len(users_union))

```

```

        else:
            co_matrix[j,i] = 0

    return co_matrix

    #Use the cooccurrence matrix to make top recommendations
    def generate_top_r(self, user, cooccurrence_matrix, a_songs,
u_songs):
        print("Non zero values in cooccurrence_matrix :%d" %
np.count_nonzero(cooccurrence_matrix))

        #Calculate the average of the scores in the cooccurrence
matrix for all songs listened by the user.
        user_sim_scores =
cooccurrence_matrix.sum(axis=0)/float(cooccurrence_matrix.shape[0]
)
        user_sim_scores = np.array(user_sim_scores)[0].tolist()

        #Sort the indices of user_sim_scores based upon their
value also maintain the corresponding score
        s_index = sorted(((e,i) for i,e in
enumerate(list(user_sim_scores))), reverse=True)

        #Create a dataframe from the following
columns = ['user_id', 'song', 'score', 'rank']
        #index = np.arange(1) # array of numbers for the number
of samples
        df1 = pandas.DataFrame(columns=columns)

        #Fill the dataframe with top 10 songs
        rank = 1
        for i in range(0,len(s_index)):
            if ~np.isnan(s_index[i][0]) and
a_songs[s_index[i][1]] not in u_songs and rank <= 10:

df1.loc[len(df1)]=[user,a_songs[s_index[i][1]],s_index[i][0],ran
k]

            rank = rank+1

        #Handle the case where there are no recommendations
        if df1.shape[0] == 0:
            print("The current user don't have any song for
similarity based recommendation model.")
            return -1
        else:
            return df1

```

```

#Create the system model
def create_s(self, t_data, u_id, i_id):
    self.t_data = t_data
    self.u_id = u_id
    self.i_id = i_id
#Use the model to make recommendations
def recommend_s(self, u):

    #A. Get all unique songs for this user
    u_songs = self.get_u_items(u)

    print("No. of songs for the user: %d" % len(u_songs))

    #B. Get all the songs in the data
    a_songs = self.get_all_items_t_data()

    print("No. of songs in the list: %d" % len(a_songs))

    #C. Make the cooccurrence matrix of size len(user_songs)
X len(songs)
    co_matrix = self.construct_co_matrix(u_songs, a_songs)

    #D. Use the matrix to make recommended songs
    df_r = self.generate_top_r(u, co_matrix, a_songs,
u_songs)
    return df_r

#Create a function to get similar songs
def similar_items(self, i_list):

    u_songs = i_list

    #A. Get all the songs from the data
    a_songs = self.get_all_items_t_data()

    print("no. of unique songs in the set: %d" %
len(a_songs))

    #B. Make the cooccurrence matrix of size len(user_songs)
X len(songs)
    co_matrix = self.construct_co_matrix(u_songs, a_songs)

    #C. Use the matrix to make recommendations
    u = ""
    df_r = self.generate_top_r(u, co_matrix, a_songs,
u_songs)

```

```
return df_r
```

**Now using the Recommendation package along with relevant python libraries, we import them in a new file:**

```
import pandas
from sklearn.model_selection import train_test_split
import numpy as np
import time
import Recommenders as Recommenders
```

**After that, we'll load the data from the .csv file & retrieve the no. of times a user listens to a song in rows of five:**

```
#Read user_id, song_id, listen_count
#This step might take time to download data from external
sources
triplets =
'https://static.turi.com/datasets/millionsong/10000.txt'
songs_metadata =
'https://static.turi.com/datasets/millionsong/song_data.csv'

song_df_a = pandas.read_table(triplets,header=None)
song_df_a.columns = ['user_id', 'song_id', 'listen_count']

#Read song metadata
song_df_b = pandas.read_csv(songs_metadata)

#Merge the two dataframes above to create input dataframe for
recommender systems
song_df1 = pandas.merge(song_df_a,
song_df_b.drop_duplicates(['song_id']), on="song_id",
how="left")
song_df1.head()
```

**Now create the popularity based music recommendation using the recommendation package (source code) of popularity recommendation :**

```
pm = Recommenders.popularity_recommender()
#create an instance of the class
pm.create(train, 'user_id', 'song')
```

```
user_id1 = u[5]
#Recommended songs list for a user
pm.recommend(user_id1)
```

**Now create the similarity based music recommendation using the recommendation package (source code) of similarity recommendation :**

```
is_model = Recommenders.similarity_recommender()
is_model.create(train, 'user_id', 'song')
```

## Result Analysis and Discussion

Using the Recommendation package along with relevant python libraries, we import them in a new file:

```
1. import pandas
2. from sklearn.model_selection import train_test_split
3. import numpy as np
4. import time
5. import Recommenders as Recommenders
```

After that, we'll load the data from a given .csv file & retrieve the no. of times a user listens to a song in rows of five:

```
1. #Read user_id, song_id, listen_count
2. #This step might take time to download data from external sources
3. triplets = 'https://static.turi.com/datasets/millionsong/10000.txt'
4. songs_metadata = 'https://static.turi.com/datasets/millionsong/song_data.csv'
5.
6. song_df_a = pandas.read_table(triplets,header=None)
7. song_df_a.columns = ['user_id', 'song_id', 'listen_count']
8.
9. #Read song metadata
10. song_df_b = pandas.read_csv(songs_metadata)
11.
12. #Merge the two dataframes above to create input dataframe for recommender systems
13. song_df1 = pandas.merge(song_df_a, song_df_b.drop_duplicates(['song_id']),
    on="song_id", how="left")
14. song_df1.head()
```

## Output

	user_id	song_id	listen_count	title	release	artist_name	year
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1	The Cove	Thicker Than Water	Jack Johnson	0
1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOB8MDR12A8C13253B	2	Entre Dos Aguas	Flamenco Para Niños	Paco De Lucia	1976
2	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBXHDL12A81C204C0	1	Stronger	Graduation	Kanye West	2007
3	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBYHAJ12A6701BF1D	1	Constellations	In Between Dreams	Jack Johnson	2005
4	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SODACBL12A8C13C273	1	Learn To Fly	There Is Nothing Left To Lose	Foo Fighters	1999

Now create the popularity based music recommendation using the recommendation package (source code) of popularity recommendation :

```
1. pm = Recommenders.popularity_recommender() #create an instance of the class
2. pm.create(train, 'user_id', 'song')
3.
4. user_id1 = u[5] #Recommended songs list for a user
5. pm.recommend(user_id1)
```

**Output:**

:	user_id	song	score	Rank
3194	9bb911319fbc04f01755814cb5edb21df3d1a336	Sehr kosmisch - Harmonia	37	1.0
4083	9bb911319fbc04f01755814cb5edb21df3d1a336	Undo - Björk	27	2.0
931	9bb911319fbc04f01755814cb5edb21df3d1a336	Dog Days Are Over (Radio Edit) - Florence + Th...	24	3.0
4443	9bb911319fbc04f01755814cb5edb21df3d1a336	You're The One - Dwight Yoakam	24	4.0
3034	9bb911319fbc04f01755814cb5edb21df3d1a336	Revelry - Kings Of Leon	21	5.0
3189	9bb911319fbc04f01755814cb5edb21df3d1a336	Secrets - OneRepublic	21	6.0
4112	9bb911319fbc04f01755814cb5edb21df3d1a336	Use Somebody - Kings Of Leon	21	7.0
1207	9bb911319fbc04f01755814cb5edb21df3d1a336	Fireflies - Charttraxx Karaoke	20	8.0
1577	9bb911319fbc04f01755814cb5edb21df3d1a336	Hey_ Soul Sister - Train	19	9.0
1626	9bb911319fbc04f01755814cb5edb21df3d1a336	Horn Concerto No. 4 in E flat K495: II. Romanc...	19	10.0

In the above code snippet, user\_id1 represents the list of popular songs recommended to the user. We will include the same for user\_id2 being the list for another user.

```
1. user_id2 = u[8]
2. pm.recommend(user_id2)
```

## Output

	user_id	song	score	Rank
3194	9bb911319fbc04f01755814cb5edb21df3d1a336	Sehr kosmisch - Harmonia	37	1.0
4083	9bb911319fbc04f01755814cb5edb21df3d1a336	Undo - Björk	27	2.0
931	9bb911319fbc04f01755814cb5edb21df3d1a336	Dog Days Are Over (Radio Edit) - Florence + Th...	24	3.0
4443	9bb911319fbc04f01755814cb5edb21df3d1a336	You're The One - Dwight Yoakam	24	4.0
3034	9bb911319fbc04f01755814cb5edb21df3d1a336	Revelry - Kings Of Leon	21	5.0
3189	9bb911319fbc04f01755814cb5edb21df3d1a336	Secrets - OneRepublic	21	6.0
4112	9bb911319fbc04f01755814cb5edb21df3d1a336	Use Somebody - Kings Of Leon	21	7.0
1207	9bb911319fbc04f01755814cb5edb21df3d1a336	Fireflies - Charttraxx Karaoke	20	8.0
1577	9bb911319fbc04f01755814cb5edb21df3d1a336	Hey_ Soul Sister - Train	19	9.0
1626	9bb911319fbc04f01755814cb5edb21df3d1a336	Horn Concerto No. 4 in E flat K495: II. Romanc...	19	10.0

Now create the similarity based music recommendation using the recommendation package (source code) of similarity recommendation :

As we built the system for popularity recommendation, we will do the same according to the songs listened by the users.

users user\_id1 & user\_id2 using similarity\_recommender class from the Recommendation package. First, we create an instance of the package, after that we proceed for making the list:

1. `is_model = Recommenders.similarity_recommender()`
2. `is_model.create(train, 'user_id', 'song')`

### a) for first user (user\_id1)

1. `#Print the songs for the user`
2. `user_id1 = u[5]`
3. `user_items1 = is_model.get_user_items(user_id1)`
4. `print(".....")`
5. `print("Songs played by first user %s:" % user_id1)`
6. `print(".....")`
- 7.



```

8. for user_item in user_items1:
9.     print(user_item)
10.
11.     print(".....")
12.     print("Similar songs recommended for the first user:")
13.     print(".....")
14.
15. #Recommend songs for the user using personalized model
16. is_model.recommend(user_id1)

```

## Output

```

-----
Songs played by first user 4bd88bfb25263a75bbdd467e74018f4ae570e5df:
-----

```

```

Just Lose It - Eminem
Without Me - Eminem
16 Candles - The Crests
Speechless - Lady GaGa
Push It - Salt-N-Pepa
Ghosts 'n' Stuff (Original Instrumental Mix) - Deadmau5
Say My Name - Destiny's Child
My Dad's Gone Crazy - Eminem / Hailie Jade
The Real Slim Shady - Eminem
Somebody To Love - Justin Bieber
Forgive Me - Leona Lewis
Missing You - John Waite
Ya Nada Queda - Kudai

```

```

-----
Similar songs recommended for the first user:
-----

```

```

No. of songs for the user: 13
No. of songs in the list: 4483
Non zero values in cooccurrence_matrix :2097

```

	user_id	song	score	rank
0	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Superman - Eminem / Dina Rae	0.088692	1
1	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Mockingbird - Eminem	0.067663	2
2	4bd88bfb25263a75bbdd467e74018f4ae570e5df	I'm Back - Eminem	0.065385	3
3	4bd88bfb25263a75bbdd467e74018f4ae570e5df	U Smile - Justin Bieber	0.064525	4
4	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Here Without You - 3 Doors Down	0.062293	5
5	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Hellbound - J-Black & Masta Ace	0.055769	6
6	4bd88bfb25263a75bbdd467e74018f4ae570e5df	The Seed (2.0) - The Roots / Cody Chestnutt	0.052564	7
7	4bd88bfb25263a75bbdd467e74018f4ae570e5df	I'm The One Who Understands (Edit Version) - War	0.052564	8
8	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Falling - Iration	0.052564	9
9	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Armed And Ready (2009 Digital Remaster) - The ...	0.052564	10

## b) for second user (user\_id2)

```
1. user_id2 = u[7]
2. #Fill in the code here
3. user_items2 = is_model.get_user_items(user_id2)
4. print(".....")
5. print("Songs played by second user %s:" % user_id2)
6. print(".....")
7.
8. for user_item in user_items2:
9. print(user_item)
10.
11. print(".....")
12. print("Similar songs recommended for the second user:")
13. print(".....")
14.
15. #Recommend songs for the user using personalized model
16. is_model.recommend(user_id2)
```

## Output

```
-----
Songs played by second user 9d6f0ead607ac2a6c2460e4d14fb439a146b7dec:
-----
Swallowed In The Sea - Coldplay
Life In Technicolor ii - Coldplay
Life In Technicolor - Coldplay
The Scientist - Coldplay
Trouble - Coldplay
Strawberry Swing - Coldplay
Lost! - Coldplay
Clocks - Coldplay
-----

Similar songs recommended for the second user:
-----

No. of songs for the user: 8
No. of songs in the list: 4483
Non zero values in cooccurence_matrix :3429
```

	user_id	song	score	rank
0	9d6f0ead607ac2a6c2460e4d14fb439a146b7dec	She Just Likes To Fight - Four Tet	0.281579	1
1	9d6f0ead607ac2a6c2460e4d14fb439a146b7dec	Warning Sign - Coldplay	0.281579	2
2	9d6f0ead607ac2a6c2460e4d14fb439a146b7dec	We Never Change - Coldplay	0.281579	3
3	9d6f0ead607ac2a6c2460e4d14fb439a146b7dec	Puppetmad - Puppetmastaz	0.281579	4
4	9d6f0ead607ac2a6c2460e4d14fb439a146b7dec	God Put A Smile Upon Your Face - Coldplay	0.281579	5
5	9d6f0ead607ac2a6c2460e4d14fb439a146b7dec	Susie Q - Creedence Clearwater Revival	0.281579	6
6	9d6f0ead607ac2a6c2460e4d14fb439a146b7dec	The Joker - Fatboy Slim	0.281579	7
7	9d6f0ead607ac2a6c2460e4d14fb439a146b7dec	Korg Rhythm Afro - Holy Fuck	0.281579	8
8	9d6f0ead607ac2a6c2460e4d14fb439a146b7dec	This Unfolds - Four Tet	0.281579	9
9	9d6f0ead607ac2a6c2460e4d14fb439a146b7dec	high fives - Four Tet	0.281579	10

## **Observations**

The lists of both the users in popularity based recommendation is the same but different in case of similarity-based recommendation. This is because the former recommends the list which is popular among a region or worldwide but the latter recommends the list similar to the choices of the user.

## **5. CONCLUSION**

### **5.1 Conclusions**

We had a great learning experience doing this project. we have learn about data mining and data cleaning .this is the very first task of a machine learning model to remove all the problem creating objects from the dataset. data cleaning and data exploration were very useful to making dataset algorithm ready .we have learnt to create machine learning model ,train the model and then doing testing on it .most common algorithms used before by others were content and collaborative but the hybrid of the two gave extraordinary results. Best suited algorithm for this project was random forest algorithm.

### **5.2 Future work**

Due to lack of time we couldn't make a model using singular value decomposition and support vector machine .Popularity based models are also good at recommendations so we'll try to implement this also to predict top-N songs to the users which are most popular at a given time.

There are of the recommender system is vast and covers various parameters. It is developing and emerging in modern day generation of e services and commerce. But simultaneously there is a need to develop and optimize the working and output of recommender system. Several service providers facilitate the users with a list of items. But this is not sufficient because customers have different preferences and choices which may mainly depend upon various factors and constraints. Also in many cases it may not be possible to recommend specific items to particular users. Therefore there is a scope for incorporating the concept of multiple dimensions in music recommender system particularly.

Most of the products and services provided by the various e commerce sites are expensive and therefore less used by customers. This leads to the inability to rate an item or set of items correctly and specifically. Thus traditional recommender system techniques are not optimum. This lays a way towards further work and development i building an optimized recommender system which also takes into considerations the feedback and all other constraints related to recommendation.

## 6. 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.
- [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] (2013, March) Google Patents. [Online].  
<http://www.google.com/patents/US7003515?dq=7,003,515>
- [8] Harrison Weber. (2014, March) TNW. [Online].  
<http://thenextweb.com/insider/2012/12/06/spotify-announces/#!A9fSz>
- [9] Spotify. (2013, March) Spotify Music for everyone. [Online].  
<https://www.spotify.com/us/>
- [10] Vikas Sindhwani Prem Melville, "Recommender Systems," in Encyclopedia of Machine Learning. Yorktown Heights, USA: IBM T. J. Watson Research Center, 2010, ch. 00338.
- [11] Brusilovsky Peter, The Adaptive Web, Alfred Kobsa, Wolfgang Nejdl Peter Brusilovsky, Ed. Pittsburgh PA, USA, 2007.
- [12] ÒscarCelma, Music Recommendation and Discovery, Springer, Ed., 2010. [15] Phil Simon, Too Big to Ignore: The Business Case for Big Data.: Wiley, 2013.
- [13] Afshin Rostamizadeh and Ameet Talwalkar MehryarMohri, Foundations of Machine Learning., 2012.
- [14] EthemAlpaydin, Introduction to Machine Learning, second edition ed.: MIT Press, 2009.