Music Recommendation System

Submitted in partial fulfillment of the requirements of the degree

BACHELOR OF ENGINEERING IN COMPUTER ENGINEERING

By

Tanmay Kulkarni (Roll No.25)
Pushpak Maind (Roll No.28)
Rahul Patil (Roll No.37)
Vishal Shirke (Roll No.48)

Name of the Mentor
Prof. Naveen Vaswani

Department of Computer Engineering

Watumull Institute of Electronics Engineering and Computer Technology

Ulhasnagar-421003

University of Mumbai

(AY 2021-22)

CERTIFICATE

This is to certify that the Mini Project entitled **“Music Recommendation System”** is a bonafide work of

Tanmay Kulkarni (25)
Pushpak Maind (28)
Rahul Patil (37)
Vishal Shirke (48)

submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **“Bachelor of Engineering”** in **“Computer Engineering”.**

1.1 (Prof. Naveen Vaswani)
Mentor

1.2 (Prof. Nilesh Mehta)
Head of Department

(Prof. Sunita Sharma)
Principal

Mini Project Approval

This Mini Project entitled "Music Recommendation System" by

Tanmay Kulkarni (25)

Pushpak Maind (28)

Rahul Patil (37)

Vishal Shirke (48)

is approved for the degree of **Bachelor of Engineering** in **Computer Engineering.**

Examiners

Naveen Vaswani
(Internal Examiner Name & Sign)

(External Examiner name & Sign)

Date:

Place

ACKNOWLEDGEMENT

We would like to extend our sincere and heartfelt gratitude to our professor Mr. Naveen Vaswani who has helped us in this endeavor and has always been very cooperative and without his help, cooperation, guidance and encouragement, the project couldn’t have been what it evolved to be.

We extend our heartfelt thanks to our faculty for their guidance and constant supervision, as well as, for providing us the necessary information regarding the project. We must thanks to our classmates for their timely help& support for compilation of this project.

Last but not least, we would like to thank all those who helped us (directly or indirect) towards the completion of this project within a limited time frame.

Tanmay Kulkarni.

Pushpak Maind.

Rahul Patil.

Vishal Shirke.

(T.E. Sem V)

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

As we all know Nowadays, lot of music industries like Amazon music, gaana.com using recommender systems and the old-fashioned way of selling music has changed. But the problem is we have many songs to choose from, so it will be easy to choose a song if we classify the songs on the basis of different genres, artists locations, age groups, languages but this method doesn't work properly. Instead of this the main goal should be to classify these set of songs in accordance to the taste of the user. Also, music service providers need an efficient way to manage songs, so that their customers can discover music.

1.1 Product Scope

As the Data analyzing has had a growth in in these couple of years more and more industries are trying to use past data to recommend product to consumers because the longer, they stay the better. Recommendation's systems are improving with better, faster, and more efficient ways to recommend customers with products of their liking.

1.2 Problem Statement and Objectives

Today, Commercial music libraries easily exceeds15 million songs, which vastly exceeds the listening capability of any single person. With millions of songs to choose from, people sometimes feel overwhelmed. Also, most of the systems recommend on the same genre/artist because of which listener won't be able to access more genre/artist. Hence there is lack of variety of songs. So, the Solution to all these problems is to use large scale and personalized Music Recommender System which learns from users listening history and features of songs and predicts what users would like to listen.

2. LITERATURE SURVEY

2.1. System Requirement Specification (SRS)

Software Requirements Specification

for

Music Recommendation System

Prepared by

1. **Tanmay Kulkarni (25)**
2. **Pushpak Maind (28)**
3. **Rahul Patil (37)**
4. **Vishal Shirke (48)**

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specifications (SRS) is to fully document the specifications and requirements for the Music Recommendation System.

1.2 Document Conventions

- MRS: Music Recommendation System.
- This document will use IEEE format. For clarity, acronyms and technical jargon, deemed uncommon by the author, will be annotated and included in the glossary. The format for headings is as followed: Major headings are in bold 16pt font, and concurrent headings in bold 14 pt. font and the rest of the font in 12 pt..

1.3 Intended Audience and Reading Suggestions

The audience of this SRS will be the clients who want the software to be built and the technical professionals developing the software.
The overall SRS is organized as follows:

- Section 1(current) Introduction of project.
- Section 2 is about overall description of the project.
- Section 3 cites the External Interface Requirements.
- Section 4 about various System Features.
- Section 5 about Other Non-functional Requirements.

1.4 Product Scope

The objective of this project is to create and implement a personalized Music Recommendation System. Most of the systems recommend on the same genre/artist because of which listener won't be able to access more genre/artist. The solution for this will be a MRS which learns from users listening history and features of songs and predicts what users would like to listen.

This project has the following modules, to manage all the requirements of the MRS.

- Recommendation Module.

2. Overall Description

2.1 Product Perspective

This product is an entirely new product. It is not a component of a larger system.

The Music Recommendation System will interact with a user to recommend music on the basis of user data. The system will ask the user to select a song and get a personalized recommendation on the basis of that song.

2.2 Product Functions

2.2.1 Music Recommendation Module

The Music Recommendation function shall allow users to insert a song and get recommendation.
On the website the user will enter the song name into a search bar and after pressing the search button, he will receive a list of recommended songs on the basis of the user song.
Rationale: This provides a method for user to get music recommendation on the basis of the input song.

2.3 User Classes and Characteristics

The types of user classes are music listeners, song enthusiast, etc. depending on the user the minimum criteria changes as follows:
A user should be minimum 8 years of age. Users of the website must possess a minimal educational level and knowledge to use a website. Users of the application must know how to navigate in a website.

2.4 Operating Environment

Hardware Requirements:
- hardware i3 6th gen , 4 gb ram

Software Requirements:
- Windows 7 operating system or higher.
- Jupyter notebook or VS Code.
- Flask
- Heroku

3. External Interface Requirements

3.1 User Interfaces

The system will provide a web platform for user to get music recommendation on the basis of their song input.

- The application will be used to get a song as an input to recommend music.

4. System Feature

4.1 Music Recommendation:

4.1.1 Description and Priority

User has to enter the song name to get personalized music recommendation on the basis of user data.

4.1.2 Stimulus: Click "Search" Button: List of music recommended.

1. The system shall allow any user to access the website and get music recommendation.
2. User has to enter the name of his select song into the search bar and click the search button.
3. The system will process the user's song input and give a list of music recommendation for the user.

4.2 Functional Requirements

Client/Server System:
A client system which will be used to collect user information as the front end application, and a server system to process and run the music recommendation module i.e. A Music Recommendation system working as the back end of the application.

Data Model: The input from the user will be taken as a String and the output of recommendation will be received as a list of recommended song strings.

5 Other Nonfunctional Requirements

5.1 Software Quality Attributes

5.1.1 Availability

This system is an online system which does require internet, so as long as the user’s system supports web services like a web browser, the system should function.

5.1.2 Security

Users will be able to access only their own personal information on recommendations and not that of other users.

5.1.3 Maintainability

Any updates or defect fixes shall be able to be made on server-side Computers only without any patches required by the user.

2.2. Limitation of Existing Systems

There are trillions and trillions of songs in the world, so many that an estimate is impossible, and the potential more an infinitely greater number which have not yet been made, a world of music for us to enjoy. Keeping this general idea, one can get that the number of songs is too high for a person, even if listening to music is his or her best hobby. People sometimes feel difficult to choose from millions of songs. Moreover, music service providers need an efficient way to manage songs and help their customers to discover music by giving quality recommendation. This means it not only gives user freedom of selecting the songs he or she wants to listen but also recommends songs according to their previous listening history.

Thus, there is a strong need of a good recommendation system. In order to efficiently access, discover, and present music content to the final user, techniques for searching, retrieving, and recommending need to be appropriate for music content. There has been some work done in both academia and the industry to provide music recommendation services. Understanding patterns of music listening and consumption can help to perform accurate and satisfying music recommendations.

To meet user's demands for a recommender system, there are some music streaming websites already providing music recommendation services, for example: Spotify, Pandora, Beats music etc. The ways how they compile their recommendation lists varies between companies. Some websites make up recommendations based on users' listening records; some recommend the music that depends upon same genres/artists that user likes to listen, which means that the system assumes that they share a similar taste, and other websites recommend music based on user's mood. Although there are already lots of different ways to draw up recommendations, users are still not satisfied with the recommendation service.

2.3. Miniproject Contribution

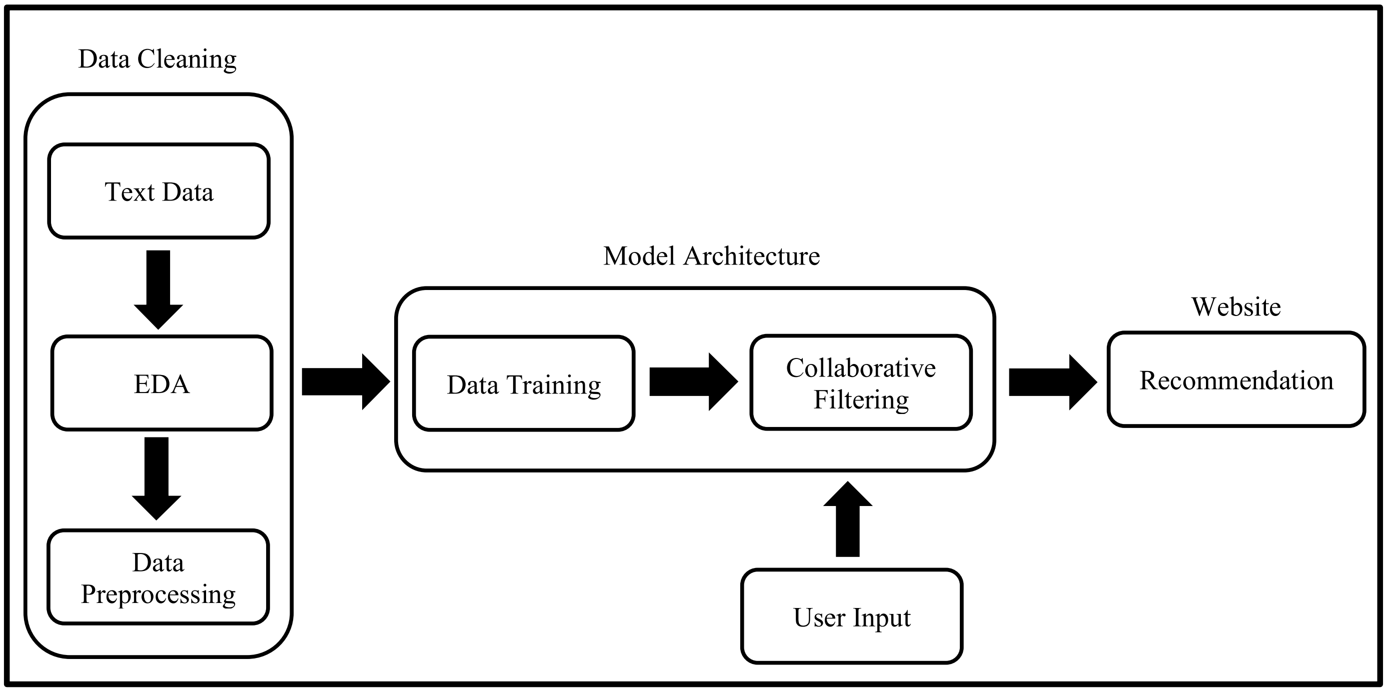
The Music recommendation system as name suggests use to recommend the songs based on user input song. There are various applications where this music recommendation system can be greatly used. It helps user to make their music feed more personalized so he/she won't have to waste hours of time on finding the related songs he already loves. This system can be majorly helpful for music & related industries to increase the customer engagement on their platforms which in turn can benefit them in long run. The system will also give idea to industries that which content is more favorable & liked by their users so they can provide more related to it in the future.

Moving to what our system uniquely contribute to the world of recommendation systems, so we've actually used User based collaborative filtering (UBCF) approach. In this, specific user will be recommended songs based on what other listeners likes to listen songs having same test to that specific user. So rather than recommending songs from user's own history or with same genre and category which is traditional & very common method, we covered some different area of songs to recommend to the user using this method. This approach is used by few technical industries to give quality recommendations to their customers but still research on how we can make more accurate & quality recommendation to consumers based on this method is going on. This is our first & little step towards the improvement in recommendation systems for music by designing of our own 'Music Recommendation system' miniproject!

3. PROPOSED SYSTEM

There is lack of variety of songs. So, we need to classify all the songs and the main goal is to classify these set of songs in accordance to the taste of the user. Moreover, music service providers need an efficient way to manage songs and help their customers to discover music by giving quality recommendation. So, the Solution to all these problems is to use large scale and personalized Music Recommender System which learns from users listening history and features of songs and predicts what users would like to listen.

3.1 Architecture/ Framework



Project Workflow

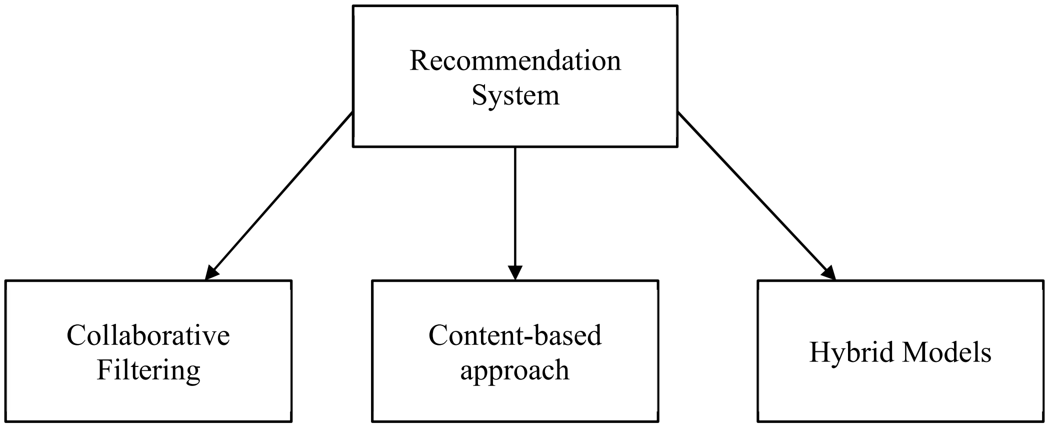
Here's the Working flow diagram of the project where in first phase the EDA techniques will be performed on raw dataset to spot anomalies, understand patterns & various correlations within the data & summarizing their main characteristics. Then in data pre-processing data gets transformed, or encoded, to bring it to such a state that now the machine can easily parse it. After the data is cleaned then it'll be trained w.r.t collaborative filtering technique & model will be tested. Based on this when end user will give any input as song, he'll get respective recommendations on exclusive webpage.

3.2 Algorithm and Process Design

Based on the problem statement, we would like to propose a music recommendation system which will be better at predicting songs and easy to use. It will consider a song as an input while giving a list of song suggestions as an output according to the user's preference. We will use machine learning techniques to predict the user's behavior and save their time.
Let's understand what a recommendation system is.

3.2.1 Recommendation System

A recommender system, or a recommendation system is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item.



Recommendation Systems

There are many ways to approach this system like a content-based system which studies the content to predict the user, a collaborative approach which studies the user rather than the item and a combination approach of these two known as hybrid model.

Here we've used collaborative filtering. So, let's see what it means actually.

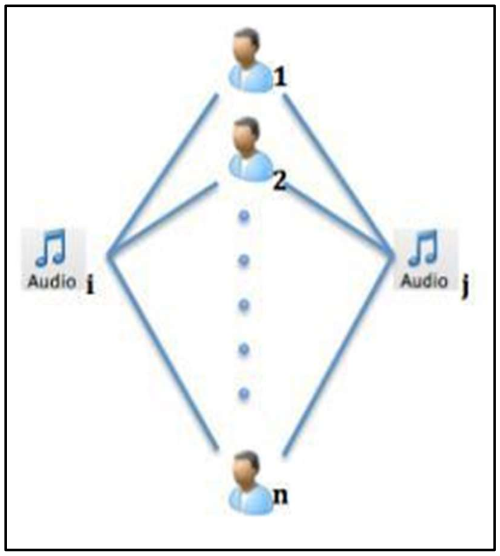
3.2.2 Collaborative Filtering

Now, talking about Collaborative filtering, it is further divided into two types:

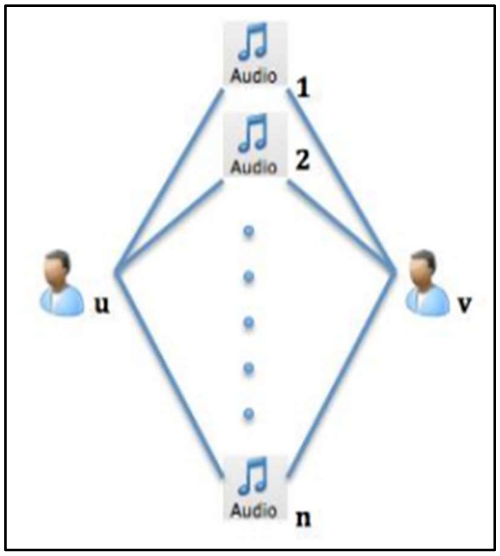
1) User Based Collaborative Filtering.
2) Item-Based Collaborative Filtering.

1. User Based Collaborative Filtering.
   User based collaborative filtering actually means a specific User will be recommended the songs on the basis of ratings given to that song by the other users which have similar interest with that of the user. The User-based collaborative filtering has greater diversity & improving ability.

2. Item-Based Collaborative Filtering.
   Especially Item based collaborative filtering which is going to be implemented in this project. In Item based collaborative filtering, here similarity between songs is check on basis of user ratings & then appropriate recommendations are done.



Item-Based



User-Based

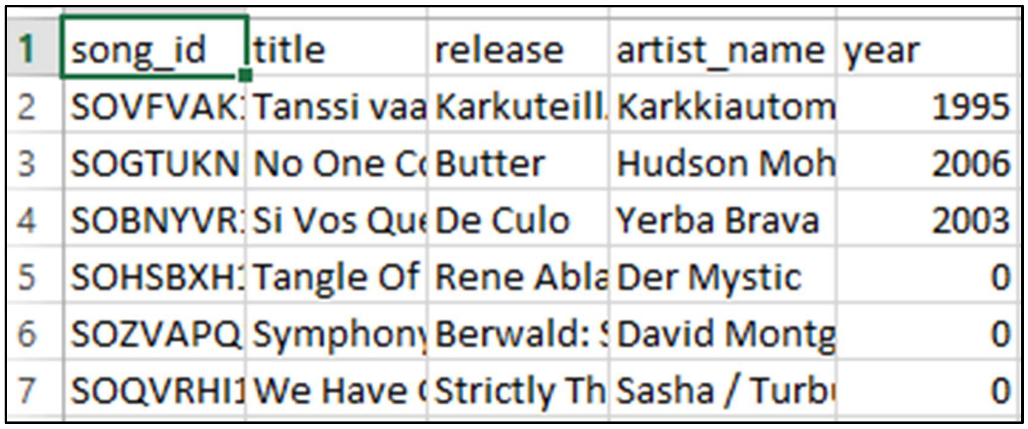
3.2.3 Dataset Information

For this project, we have used million songs dataset from Kaggle which has data of 1 million users and 1/2 million songs. This dataset contains two files.

1. Song data: which has song_id, title, release, artist name & year.

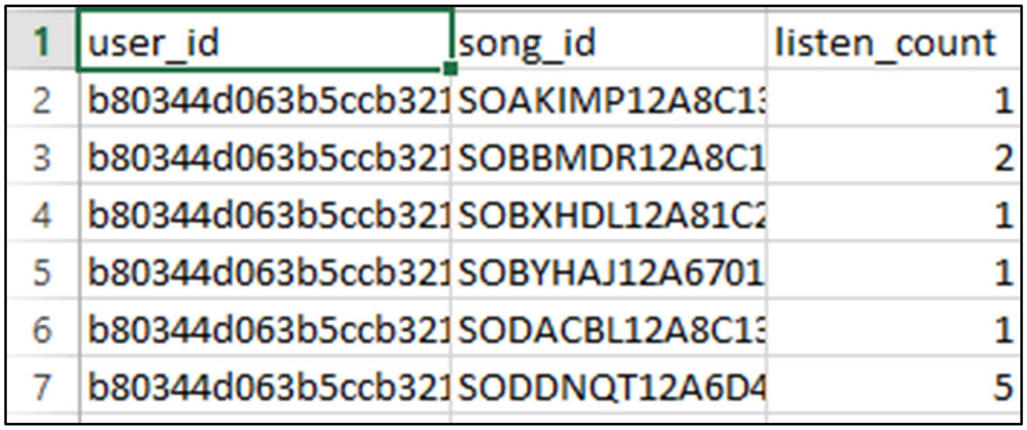
2. Triplet: which has user_id, song_id and listen count.

We have processed the data into a single file and corrected the data to be used
for the model.



1	song_id	title	release	artist_name	year
2	SOVFVAK:	Tanssi vaa	Karkuteill.	Karkkiautom	1995
3	SOGTUKN	No One C	Butter	Hudson Moh	2006
4	SOBNYVR:	Si Vos Que	De Culo	Yerba Brava	2003
5	SOHSBXH:	Tangle Of	Rene Abla	Der Mystic	0
6	SOZVAPQ	Symphony	Berwald: :	David Montg	0
7	SOQVRHI1	We Have (	Strictly Th	Sasha / Turb	0

Song_data

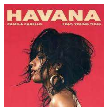


1	user_id	song_id	listen_count
2	b80344d063b5ccb321	SOAKIMP12A8C1:	1
3	b80344d063b5ccb321	SOBBMDR12A8C1	2
4	b80344d063b5ccb321	SOBXHDL12A81C:	1
5	b80344d063b5ccb321	SOBYHAJ12A6701	1
6	b80344d063b5ccb321	SODACBL12A8C1:	1
7	b80344d063b5ccb321	SODDNQT12A6D4	5

Triplet file

3.2.4 Co-Occurrence Matrix

So, now heading towards the methodology or procedure used for this recommendation system we came up with a co-occurrence matrix approach. So, what's co-occurrence matrix, it's actually a combination of terms that are likely to be used in the same context. But, why do we need it here. Actually, we're applying it here to calculate the similarity between the user songs (means the songs given by user for recommendation) & all songs means (all the unique songs in training dataset). Hence, we need to take these 2 as a parameter or say inputs to build this matrix of L (user songs) x L (all songs) initially with all zeroes.

					
	1	**5**	**6**	**3**	**2**
	2	**1**	**5**	**8**	**9**
	10	**4**	**1**	**5**	**7**
	4	**8**	**5**	**1**	**4**
	7	**9**	**3**	**2**	**1**

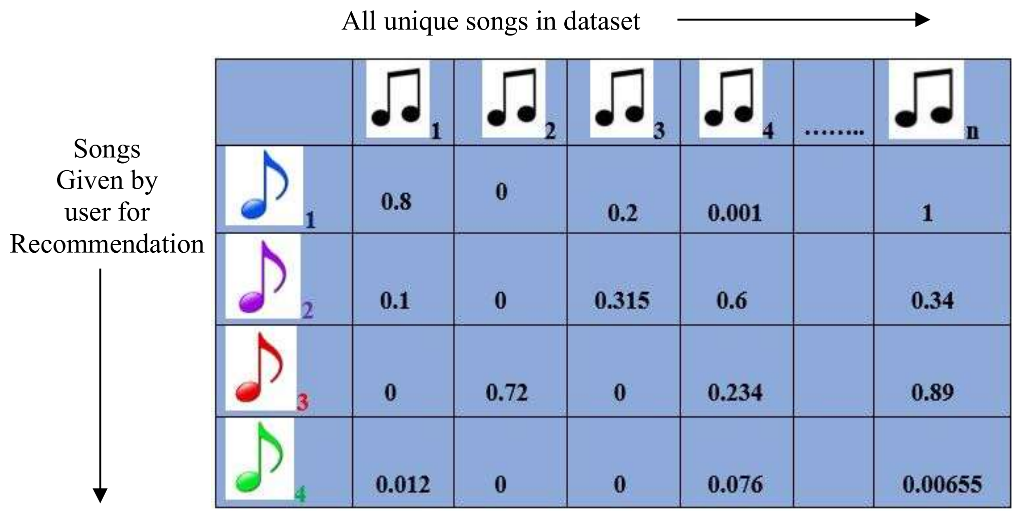
Co-Occurence Matrix

Co-occurrence matrix = L (User songs) x L (All songs)

Then user based collaborative filtering will be applied.

3.2.5 Concept of Jaccard Index

Now after just creating this matrix of said length we need to make it ready for recommendation purpose. So, the Jaccard index concept plays a crucial role here. Jaccard index is actually used to show the similarities between any 2 sample sets. Particularly speaking Jaccard index is just representing every position in co-occurrence matrix here. The two sets require here are A & B, where A is saying that the set of all unique listeners of each song, where those songs are provided by user himself for recommendation. Similarly, B says that the set of all unique listeners of each song, but those songs here are from our dataset itself. & Finally, now we're calculating Jaccard index by the formulae shown below, where we first are taking length of intersection of two sets A & B to get to know, how many common listeners are there from each set of songs. Similarly, we're calculating their union & then doing division to get actual Jaccard index value.


All unique songs in dataset
Songs Given by user for Recommendation

	1	2	3	4	…….	n
1	0.8	0	0.2	0.001		1
2	0.1	0	0.315	0.6		0.34
3	0	0.72	0	0.234		0.89
4	0.012	0	0	0.076		0.00655

Jaccard Index

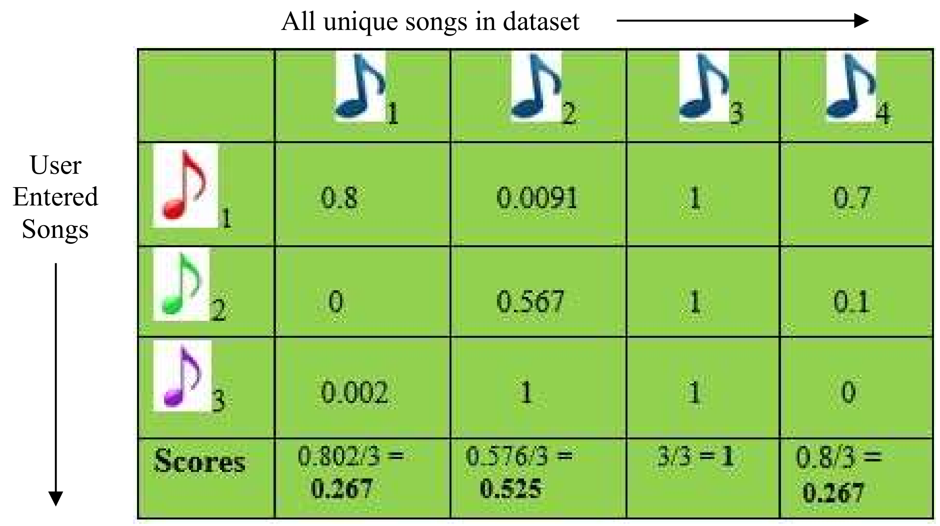
Jaccard Index Calculation:

formulae:

$$J\,(A, B) = \frac{\mathbf{len}(A \cap B)}{\mathbf{len}(AUB)} = \frac{I}{U}$$

You can see the sample figure on right which shows how matrix could look like after filling all indexes. In that, columns are representing all unique songs in dataset & rows represents songs entered by user. The more is the index value more is the similarity between songs.

3.2.6 Recommending Top Songs


All unique songs in dataset
User Entered Songs

	♪1	♪2	♪3	♪4
♪1	0.8	0.0091	1	0.7
♪2	0	0.567	1	0.1
♪3	0.002	1	1	0
Scores	0.802/3 = **0.267**	0.576/3 = **0.525**	3/3 = **1**	0.8/3 = **0.267**

Score Calculation

Scores Calculation: Weighted average method.

Scores = Sum of all rows (column wise) / Total no. of rows

At the last, from the matrix which is ready now, we're calculating scores for each song which will help us to give user proper recommendations. We've taken a weighted average of all Jaccard indexes in column wise manner because we have to recommend songs from our dataset. It's simple formulae used here as sum of all row's column wise divided by total no. of rows. Like u can see from sample figure in right, each column indexes we're adding row wise & dividing them by total no. of rows as 3 here. After calculating all scores, we're deciding ranks for each song by Highest score will have Lowest rank approach. & Hence, we can give top 10 rankings as top 10 recommended songs to the particular user. We also concluded here as more is the length of intersection in Jaccard indexes less is difference between union & intersection and hence score will be more so that particular song will be strongly recommended to the user by giving top ranking.

3.3 Details of Hardware and Software Requirements

3.3.1 Hardware Requirements

Laptop or Desktop with minimum of
intel core i5 Processor with
At least 8gb of Ram and 500gb of storage

3.3.2 Software Requirements

- Any IDLE (Visual Studio Code Recommended)
- Latest version of Python (3.9)
- Python Libraries like NumPy, Pandas, Flask, Heroku
- Jupyter Notebook

3.4 Experiment & Results for Validation & Verification

3.4.1 Experiment (Methodology)

There are 3 types of recommendation system: content-based, collaborative and Hybrid.

Our Million Songs Dataset contains of two files: triplet file and metadata file. The triplet file contains user_id, song_id and listen time. The metadata_file contains song_id, title, release_by and artist_name.

Our first job is to integrate our dataset, which is very important every time we want to build a data processing pipeline. To integrate both triplet_file and metadata_file, we are going to use a popular Python library called pandas.

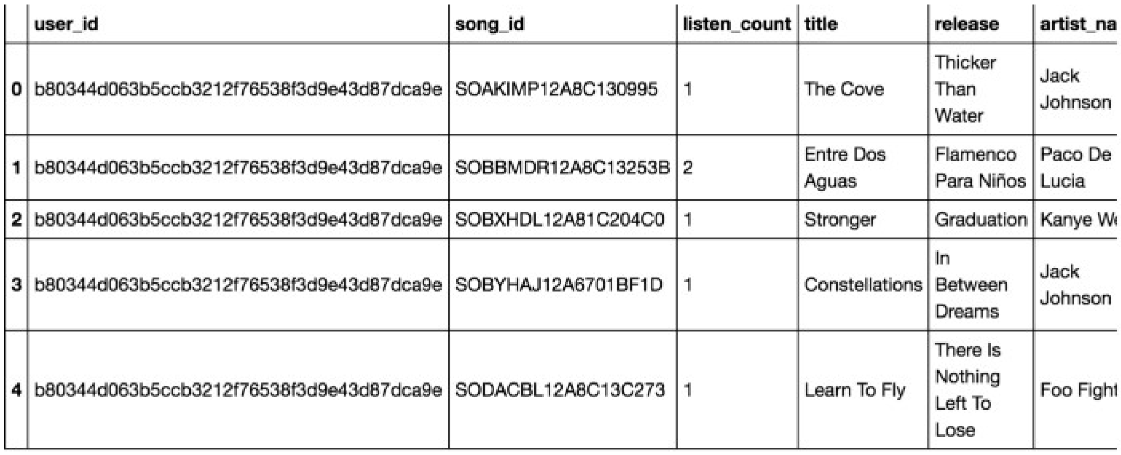
We first define the two files we are going to work with:

```
song_df_1 = pd.read_csv('triplets_file.csv')
    song_df_1.head()
    song_df_2 = pd.read_csv('song_data.csv')
    song_df_2.head()
```

We then read the table of triplet_file using pandas and define the 2 columns as user_id, song_id and listen_count (df here means data frame).
We also read the metadat_file and going to combine the metadata_file with triplets_file. Whenever you combine 2 or more datasets, there will be duplicate columns. Here we drop the duplicates between 2 datasets using song_id. We also read the metadat_file and going to combine the metadata_file with triplets_file. Whenever you combine 2 or more datasets, there will be duplicate columns. Here we drop the duplicates between 2 datasets using song_id.

```
 song_df = pd.merge(song_df_1,
song_df_2.drop_duplicates(['song_id']), on='song_id',
how='left')
```

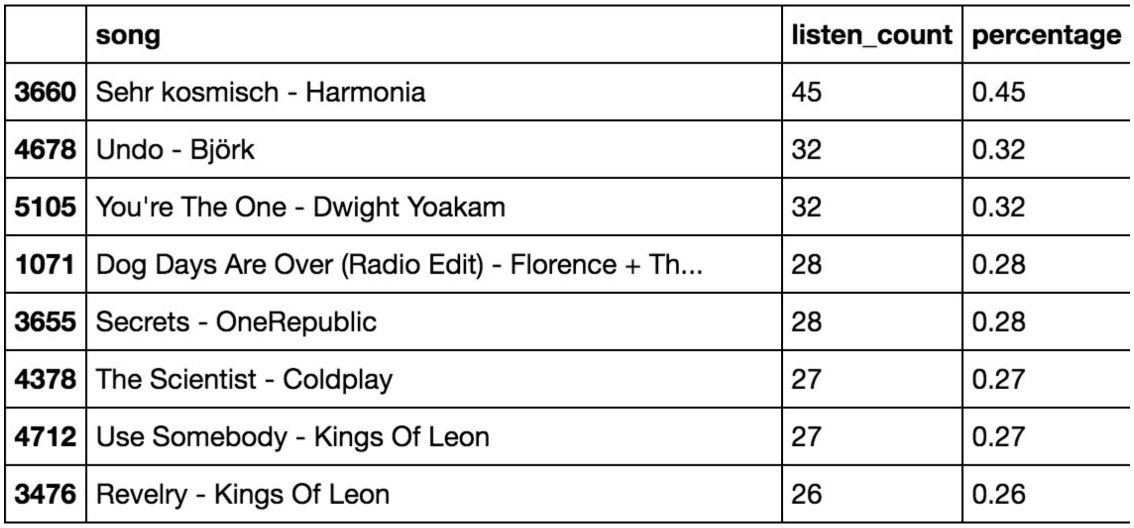
```
    song_df.head()
```



	user_id	song_id	listen_count	title	release	artist_na
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1	The Cove	Thicker Than Water	Jack Johnson
1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBBMDR12A8C13253B	2	Entre Dos Aguas	Flamenco Para Niños	Paco De Lucia
2	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBXHDL12A81C204C0	1	Stronger	Graduation	Kanye We
3	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBYHAJ12A6701BF1D	1	Constellations	In Between Dreams	Jack Johnson
4	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SODACBL12A8C13C273	1	Learn To Fly	There Is Nothing Left To Lose	Foo Fight

Here we have the song index, user_id, song_id, listen_count, title, release and artist_name. Running len(song_df) returns the total length of this dataset indexed by song are 2,000,000.
The second step of this exercise is data transformation, where we're going to select a subset of this data (the first 10,000 songs). We then merge the song and artist_name into one column, aggregated by number of times a particular song is listened too in general by all users. The first line in the code below groups the song_df by number of listen_count ascending. The second line calculate the group_sum by summing the listen_count of each song. The third line add a new column called percentage, and calculate this percentage by dividing the listen_count by the sum of listen_count of all songs and then multiply by 100.

```
song_df['song'] = song_df['title']+' -
'+song_df['artist_name']
    song_df.head()
    song_df = song_df.head(20000)
    song_grouped =
song_df.groupby(['song']).agg({'listen_count':'count'}).rese
t_index()
    song_grouped.head()
    grouped_sum = song_grouped['listen_count'].sum()
    song_grouped['percentage'] =
(song_grouped['listen_count'] / grouped_sum ) * 100
    song_grouped.sort_values(['listen_count', 'song'],
ascending=[0,1])
```



	song	listen_count	percentage
3660	Sehr kosmisch - Harmonia	45	0.45
4678	Undo - Björk	32	0.32
5105	You're The One - Dwight Yoakam	32	0.32
1071	Dog Days Are Over (Radio Edit) - Florence + Th...	28	0.28
3655	Secrets - OneRepublic	28	0.28
4378	The Scientist - Coldplay	27	0.27
4712	Use Somebody - Kings Of Leon	27	0.27
3476	Revelry - Kings Of Leon	26	0.26

The second part of this exercise is to create a ML personalized song recommender system by leveraging the *item similarity based collaborative filtering model.* Recall that recommender system is divided into 2 types: *content based* and *collaborative based*. Content based system predicts what a user like based on what that user like in the past. Collaborative based system predicts what a particular user like based on what other similar users like. Most companies like Netflix and Hulu use the hybrid approach, which provide recommendation based on the combination of what content a user like in the past as well as what other similar user like.

First, we create an instance item similarity-based recommender class and feed it with our data.

```
ir = Recommenders.item_similarity_recommender_py()
    ir.create(song_df, 'user_id', 'song')
```

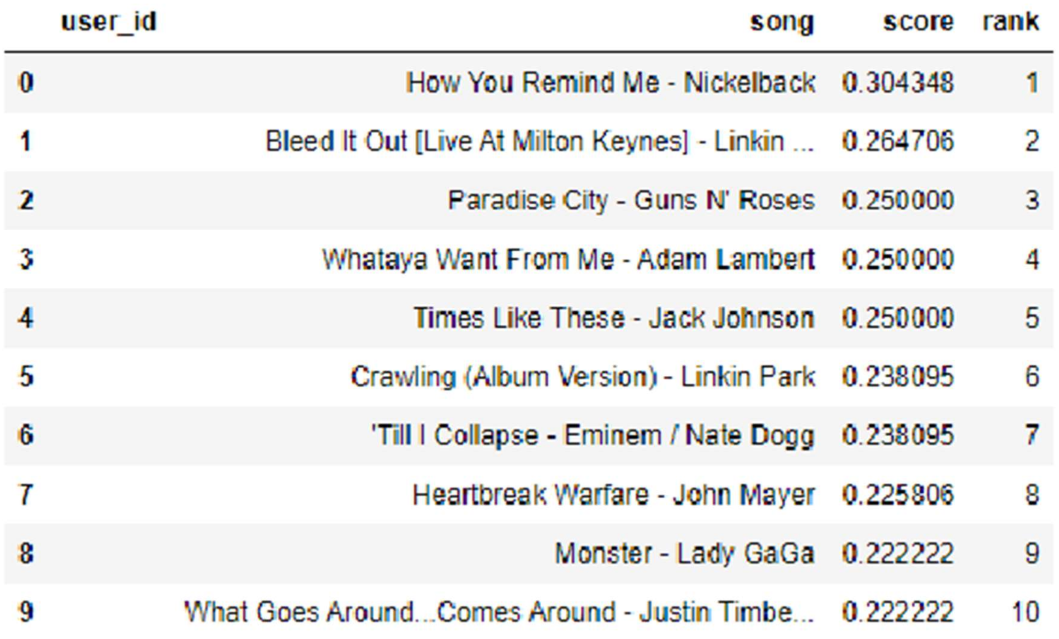
Notice that inside the recommender Class file, the generate_top_recommendations function calculated a weighted average of the scores in co-occurrence matrix for all user song. This co-occurrence matrix will tend to be sparse matrix because it's not possible to predict if a user like a particular song, whether or not he/she will like a million other song. The possibility is so vast. Using our model, we will be able to predict the list of song that a user will like.

```
df3=song_df[song_df['title'].str.contains(str(s1),
na=False)][['song']]
ir.get_similar_items([str(t1)])
```

Here we get the recommendations and we put that in a list to be displayed on the website.

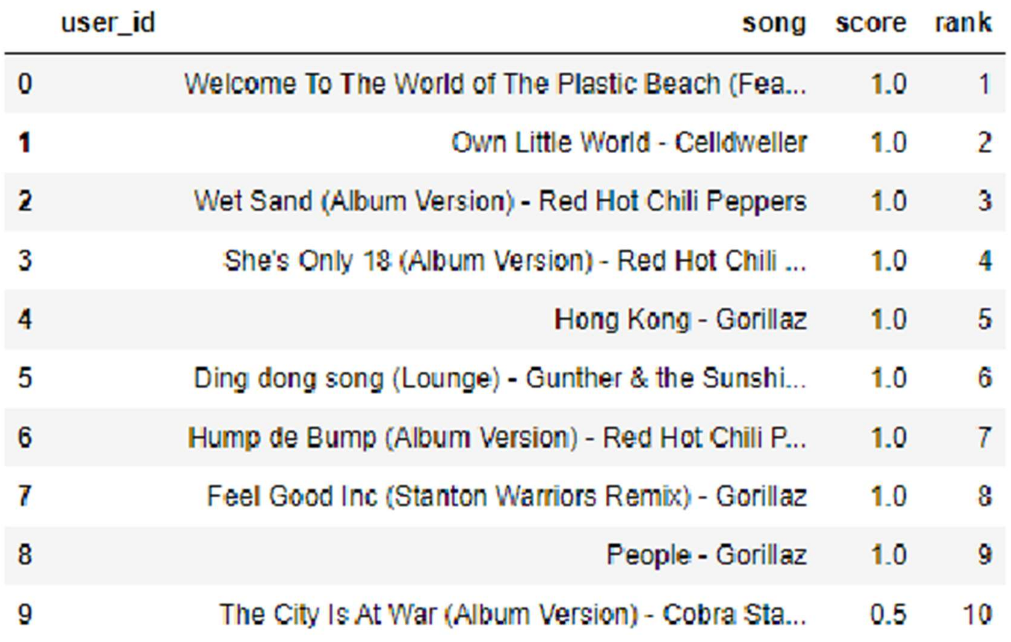
3.4.2 Results

After the song is entered by the user, we get the top ten recommendations as follows:



	user_id	song	score	rank
0		How You Remind Me - Nickelback	0.304348	1
1		Bleed It Out [Live At Milton Keynes] - Linkin ...	0.264706	2
2		Paradise City - Guns N' Roses	0.250000	3
3		Whataya Want From Me - Adam Lambert	0.250000	4
4		Times Like These - Jack Johnson	0.250000	5
5		Crawling (Album Version) - Linkin Park	0.238095	6
6		'Till I Collapse - Eminem / Nate Dogg	0.238095	7
7		Heartbreak Warfare - John Mayer	0.225806	8
8		Monster - Lady GaGa	0.222222	9
9		What Goes Around...Comes Around - Justin Timbe...	0.222222	10

result 1



	user_id	song	score	rank
0		Welcome To The World of The Plastic Beach (Fea...	1.0	1
1		Own Little World - Celldweller	1.0	2
2		Wet Sand (Album Version) - Red Hot Chili Peppers	1.0	3
3		She's Only 18 (Album Version) - Red Hot Chili ...	1.0	4
4		Hong Kong - Gorillaz	1.0	5
5		Ding dong song (Lounge) - Gunther & the Sunshi...	1.0	6
6		Hump de Bump (Album Version) - Red Hot Chili P...	1.0	7
7		Feel Good Inc (Stanton Warriors Remix) - Gorillaz	1.0	8
8		People - Gorillaz	1.0	9
9		The City Is At War (Album Version) - Cobra Sta...	0.5	10

result 2

3.4.3 Analysis (Accuracy)

Accuracy of a recommendation model is difficult to find because it’s a recommendation. So, accuracy is calculated by checking the consistency of the model. Meaning, checking the accuracy at different levels of data input and checking how many songs have actually changed.

We Calculated the accuracy by using 5 songs with five different values of data input ranging from 20,000 to 100,000 and achieved **60%** accuracy which is good for a recommendation model.

4. REFERENCES

- https://www.ijert.org/music-recommendation-system : Smt. Namitha S J 1 Assistant Professor, Department of Computer Science & Engineering, B N M Institute of Technology, Bangalore, India 1. Analysis of Music Recommendation System using Machine Learning Algorithms Prachi Singh1, Mr. Pankaj K. Singh2, Mr. Amit Ganguli3, Mr. Ajit Shrivastava4. International Research Journal of Engineering and Technology (IRJET)
- https://cse.iitk.ac.in/users/cs365/2014/_submissions/shefalig/project/: Music Recommendation System by IIT Kanpur A Survey of Music Recommendation Systems and Future Perspectives Yading Song, Simon Dixon, and Marcus Pearce.McFee, B., BertinMahieux,T., Ellis, D. P., Lanck-riet, G. R. (2012, April). The million song dataset challenge. In Proceedings of the 21st international conference companion on World Wide Web (pp. 909916).ACM
- https://code-projects.org/blood-donation-system-in-java-with-source-code/
- https://codebun.com/blood-bank-management-project-in-java-with-source-code-and-project-report/
- https://www.academia.edu/37555053/SRS_on_Blood_Bank_Management_System#:~:text=SRS%20on%20Blood%20Bank%20Management%20System%20Objective%20The%20main%20objective,of%20a%20blood%20transfusion%20and
- https://code-projects.org/blood-donation-system-in-java-with-source-code/
- https://codebun.com/blood-bank-management-project-in-java-with-source-code-and-project-report/