**CS735- DATA MINING TECHNIQUES**

A REPORT ON THE PROJECT ENTITLED

**“MUSIC RECOMMENDATION SYSTEM”**

****

SUBMITTED BY:

**Abhirup Sengupta Debashish Bose**

232IS001 232IS009

I SEMESTER M-TECH CSE (CSE-IS) I SEMESTER M-TECH CSE (CSE-IS)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA**

**SURATHKAL**

**2023 – 2024**

Table of Contents

[ABSTRACT 3](#_Toc151386278)

[Chapter 01: INTRODUCTION 4](#_Toc151386279)

[a) Context 4](#_Toc151386280)

[b) Types of recommendation systems 5](#_Toc151386281)

[1) Collaborative approach 5](#_Toc151386282)

[2) Content Based 7](#_Toc151386283)

[3) Context-based approach 8](#_Toc151386284)

[4) Hybrid approach 9](#_Toc151386285)

[Chapter 2 : Problem Statement and Methodology 10](#_Toc151386286)

[Problem Statement 10](#_Toc151386287)

[Methodology 11](#_Toc151386288)

[Chapter 04: Implementation 12](#_Toc151386289)

[4.1 Date Set Used in the Major Project 12](#_Toc151386290)

[Spotify Million Song Dataset 12](#_Toc151386291)

[4.2 Date Set Features 12](#_Toc151386292)

[4.2.1 Types of Data Set 12](#_Toc151386293)

[4.2.2 Number of Attributes, fields, description of the data set 12](#_Toc151386294)

[4.3 Design of Problem Statement 13](#_Toc151386295)

[4.4 Algorithm / Pseudo code of the Project Problem 14](#_Toc151386296)

[TF-IDF 17](#_Toc151386297)

[4.5 Screenshots of the various stages of the Project 19](#_Toc151386298)

[Chapter 05: CONCLUSION 23](#_Toc151386299)

[5.1 Application of the Project 23](#_Toc151386300)

[5.2 Limitation 24](#_Toc151386301)

[5.3 Future Work 25](#_Toc151386302)

[REFERENCES 26](#_Toc151386303)

# ABSTRACT

Recommendation systems have emerged as a result of the large amount of data available on the Internet. Many firms, such as Amazon and Flipkart for e-commerce, wynk music and ganna.com for music streaming, are now employing recommender systems to their advantage. We provide a framework in this particular situation that can then recommend new melodies to clients based on their preferences. This initiative primarily focuses on providing music recommendations to music fans in order to assist them in listening to tracks that they may enjoy. Clients can use this framework to identify new collections of tunes, making the melodic list available for tuning in. Music is life for music fans, and it has become a larger part of everyone's lives. Music helps us tune in to the cosmos, and the best part about music is that nothing can soothe you like a soothing melody. We chose to do this project because of all the positive aspects of music and the increasing demand for recommender systems on the market. The report comprises a topic description, and a full summary of the work completed thus far. The paper includes thorough explanations of the work completed, including snapshots of implementations, various techniques, and tools used thus far. The project schedule and deliverables are also included in the report. The major goal of music recommendation in this study is to provide strong human-computer interaction and deliver good recommendations to users. It is fluid and can be changed by variables other than the listening history of users or songs

# Chapter 01: INTRODUCTION

## a) Context

The first suggestion system was created in 1979. Elaine Rich defined her Grundy library system as follows: it is used to offer books to users after a brief interview in which the user is requested to fill in his first and last name, and then Grundy asks them to define themselves in a few key terms in order to discover their preferences and classify them as a "stereotype." Grundy provides an initial suggestion by providing a summary of the book after the data has been recorded. If the user is unhappy with the option, Grundy asks questions to figure out which part of the book it made a mistake on and then proposes a fresh one.

Recommendation systems, which first appeared in the 1990s, have advanced significantly in recent years, particularly with the introduction of Machine Learning and networks. On the one hand, the expanding use of today's digital world, which is characterized by a wealth of data, has enabled us to collect massive user databases. On the other hand, when computing power increased, it became possible to handle these data, particularly using Machine Learning, when human skills were no longer capable of conducting a thorough examination of such a large amount of data.

Unlike search engines, which get queries with specific information about what the user wants, a recommendation system does not receive a direct request from the user, but instead must provide them fresh options based on their past behaviors. E-commerce sites that want to sell as many commodities or services as possible to customers (travel, books, etc.) must swiftly recommend appropriate commodities. The purpose of services that provide streaming music and movies is to keep people on their platform for as long as possible. The recurring theme is that appropriate recommendations are required. Recent advancements in this industry have been significant, and these tips are advantageous to both businesses looking to maximize earnings and customers who are no longer overwhelmed by the quantity of options available. Making decisions is therefore made simple, and a good tip saves a lot of time.

The Recommender System is a software application and algorithm that provides suggestions for items that a user is most interested in. Recommendations are used in a variety of real-world situations, such as deciding what products to buy, listening to music, or reading the latest news. On the other side, there has been a shift in recorded commodity music, particularly after Apple acquired Beats Music in 2014. The music 7 industry's economic model has recently shifted from commodity sales to subscriptions and streaming. In comparison to prior eras, the availability of digital music is now abundant due to the new business model in the music industry. As a result, the importance of a music recommender system for music suppliers cannot be overstated. It is foreseeing.Collaborative filtering makes suggestions based on the collaborative power of the available evaluation by users. It is assumed that if people rate music things similarly or behave similarly, they would rate other music items similarly as well. The sparse evaluation matrix is the major issue in collaborative filtering methods since most users only see a tiny portion of all music libraries, hence most assessments are not decided. Content-based filtering, on the other hand, makes suggestions based on the characteristics of the music pieces.

We will see if we can get better recommendations by using real-time data, such as a user's heart rate and the time of day, when making recommendations in this project. The recommendations will be made by a system that employs several machine learning techniques and is accessible via a mobile application. The system uses a smart watch to recognise the user’s heart rate in order to give recommendations of songs according to what kind of music is usually associated with that heart rate and time of day for that specific user. For instance, if a user is out running, the user’s heart rate is probably higher than normal.

Many music firms, such as Amazon Music, Wynk Music, and Gaana.com, now use recommender algorithms, and the old technique of selling music has shifted to a cloudbased one. All of their music resources are now available in the cloud, and customers may listen to tracks directly from there. However, the problem is that the cloud system has a large amount of music. As a result, we must categorize all of the songs based on various genres, artists' regions, age groups, and languages, with the primary purpose of categorizing these songs according to the user's preferences. Because users demand a good return on their time and money, we can attract a large number of clients by offering a variety of valuable services that they are interested in. We're using a variety of machine learning methods as well as data mining techniques for this project. We tested a number of algorithms and compared the results to determine the most effective algorithm for our model.

## b) Types of recommendation systems

Collaborative filtering, content-based information retrieval techniques, and contextbased recommendation are the three basic recommendation systems that allow users to 8 construct personalized music playlists. Collaborative filtering makes suggestions based on the collaborative power of the available evaluation by users. The purpose of services that provide streaming music and movies is to keep people on their platform for as long as possible. The recurring theme is that appropriate recommendations are required. Recent advancements in this industry have been significant, and these tips are advantageous to both businesses looking to maximize earnings and customers who are no longer overwhelmed by the quantity of options available. Making decisions is therefore made simple, and a good tip saves a lot of time.

It is assumed that if people rate music things similarly or behave similarly, they would rate other music items similarly as well. The sparse evaluation matrix is the major issue in collaborative filtering methods since most users only see a tiny portion of all music libraries, hence most assessments are not decided. Content-based filtering, on the other hand, makes suggestions based on the characteristics of the music pieces. It is possible to combine the preceding strategies, which is referred to as hybrid.

### 1) Collaborative approach

This kind of recommendation is based on an examination of both the behavior of the listeners and the behavior of all other platform users. The basic premise is that other users' opinions may be utilized to make a credible prediction about another user's preferences for an item that they have not yet rated: a user is given recommendations based on other users who share their tastes. Indeed, for years, we've asked our friends, family, and coworkers for recommendations when it came to music, restaurants, movies, and other entertainment. This mechanism is the one that is being attempted to be replicated here. This strategy (based on stars offered by other users) was pioneered by Netflix, but it is now widely used, notably for Spotify's Discover Weekly.

Collaborative filtering makes suggestions based on the collaborative power of the available evaluation by users.

It is assumed that if various users rate music things similarly or have similar behaviour, they would rate other music items similarly. The sparse evaluation matrix is the major issue in collaborative filtering methods since most users only see a tiny portion of all music libraries and hence most assessments are not decided.

When a new user is added to the system it will not initially have enough ratings for the system to find sufficiently similar users and thus the accuracy of the predictions will be limited. Another example of cold start is when a new item is added to the system. The 9 item will not have any ratings so it won’t be recommended to any users. There are many ways so solve this problem, asking new user for initial ratings or recommending the most popular items while gathering more information are two approaches to overcome the new user cold start problem. An approach to solve the new item problem is to implement content-based filtering in a hybrid approach which will be discussed in the coming section.

For example, if user X and user Y have rated a large number of products similarly, their relationship will be strong, and when user X offers a fresh high rating on an item that user Y has not yet reviewed, the system will suggest that item to user Y. A neighbourhood is a collection of people who share similar tastes, and predictions and suggestions may be generated by looking at the neighbor's rankings and user history. Pearson's correlation coefficient might be used to determine how similar two users are.

The fundamental assumption here is that the opinions of other users can be used to provide a reasonable prediction of another user’s preferences for an item that they have not yet rated: a user is given recommendations based on users with whom they share the same tastes with. Indeed, during years, in order to choose music, restaurants, movies, etc.. We have been asking our friends, family, and colleagues to recommend something they liked. And it is this mechanism that is attempted to be reproduced here. Netflix was a pioneer of this method (based on stars given by other users) but it is now widely used, including for Spotify’s Discover Weekly. Collaborative filtering makes suggestions based on the collaborative power of the available evaluation by users.

Implementing content-based filtering in a hybrid method to handle the new item problem is one solution, which will be described in the next section. The cold start problem for new users also applies to content-based filtering, because a new user will have no songs to filter and recommend.

As stated in the scope, our system does not rely on collaborative filtering because we do not have the necessary user base to support it. Deep neural networks are the only implementation that uses collaborative filtering, and even then, its use is limited. We had planned to use collaborative filtering in conjunction with content-based filtering and real-time data, but this was not feasible. It is assumed that if people rate music things similarly or act in similar ways, they would rate other music items similarly. Because most users only encounter a tiny portion of all music libraries, the sparse evaluation matrix is the fundamental issue in collaborative filtering approaches. As a result, most assessments are not decided.

### 2) Content Based

The investigation of the content of the items candidates for suggestion is what contentbased recommendation is all about. This method attempts to deduce the user's tastes in order to suggest goods that are similar in content to those they have previously enjoyed. This method does not require listener feedback; it is only based on sound similarity, which is calculated using information taken from previously heard songs.

The characteristics of each item are used in content-based filtering to locate items that are comparable. We may propose an item based on how similar it is to all other things in the dataset by assigning a score to how similar each item is.

We utilize the properties (loudness, pace, etc.) of each song in a Spotify playlist to calculate the average score of the entire playlist. Then we suggest a song that has a comparable score to the playlist but isn't on it.

Collaboration filtering has concerns with new items, while content-based filtering does not, as will be addressed in the following section. However, one disadvantage of content-based filtering is that it will only propose songs that are similar to what you currently listen to and will not suggest music that are different. For a new user who has only listened to a few songs, the suggestions will be based exclusively on these few listenings, making them incredibly predictable and disappointing. However, if the extracted information is diverse enough and based on a variety of characteristics, the system may be able to make connections between songs that do not sound similar to the human ear.

A two-stage technique is used in content-based approaches to extract traditional audio content elements and forecast user preferences. In order to identify audio perceptual similarity, several studies have focused on extracting and comparing acoustic variables like as timbre and rhythm. Personalized suggestions in the form of item rankings will arise from both collaborative and content-based screening. Because it compares the similarity of characteristics on audio signals, our method may be classified as contentbased music recommendation. We provide music suggestions based on how similar the user's opinions of their favorite music were when they first heard it.

To locate related music, the project uses content-based filtering in addition to user id and real-time parameters.

The difficulty of collecting data does not apply to our project because Spotify offers a wide range of information associated with a track, which will be utilised as the material for comparing songs in our system. However, due to a lack of user data, we will only be able to use three of the features: volume, mode, and tempo. It's impossible to say whether these characteristics are the best for describing a recording, while pace and loudness have been linked to heart rate.

The theory behind content-based filtering is that if a person is interested in one item, they are also likely to be interested in similar products. As a result, content-based filtering employs labels and characteristics to organise and filter objects. Different qualities can be retrieved depending on the object category, and items with comparable attributes are grouped together. Only the target user's history is required this method. The content of the items in the user's history is compared to the content of other domain items, and the domain items with the highest resemblance to the ones in the user's past are recommended. When proposing news articles, content-based filtering is a popular use.

The similarity between the elements is the basis for this strategy. It's a matter of extracting features to best describe the music in order to evaluate similarities. The Machine Learning algorithms then suggest the item that is the most similar to the ones that the user already likes.

A two-stage technique is used in content-based approaches to extract traditional audio content elements and forecast user preferences. In order to identify audio perceptual similarity, several studies have focused on extracting and comparing acoustic variables like as timbre and rhythm. Personalized suggestions in the form of item rankings will arise from both collaborative and content-based screening. Because it compares the similarity of characteristics on audio signals, our method may be classified as contentbased music recommendation. We provide music suggestions based on how similar the user's opinions of their favorite music were when they first heard it.

The main advantage of this approach is that an unknown music is just as likely to be recommended as a currently popular one, or even a timeless one. This allows new artists with a few” views” to be brought up as well. Moreover, the problem of the cold start and in particular of the new items is thus avoided: when new items are introduced into the system, they can be recommended directly, without requiring integration time as is the case for recommendation systems based on a collaborative filtering approach.

### 3) Context-based approach

We listen to music at a specific time, in a specific emotional state, and under specific conditions (party, job, etc.). And these predispositions will have a significant impact on how we feel about music. Although there are several uses for this sort of recommendation, such as tourist guide apps with adaptive ambient music, there are few actual examples.

The basic premise is that other users' opinions may be utilized to make a credible prediction about another user's preferences for an item that they have not yet rated: a user is given recommendations based on other users who share their tastes. Indeed, for years, we've asked our friends, family, and coworkers for recommendations when it came to music, restaurants, movies, and other entertainment. Personalized suggestions in the form of item rankings will arise from both collaborative and content-based screening. Because it compares the similarity of characteristics on audio signals, our method may be classified as content-based music recommendation. We provide music suggestions based on how similar the user's opinions of their favorite music were when they first heard it.

This mechanism is the one that is being attempted to be replicated here. In terms of results, the major goal was to establish a framework for consumers to use in order to assist them find the ideal tunes for them. This project seeks to discover the correlation and similarity between different songs, and then construct a recommendation system framework that suggests new music for your Spotify playlist based on that information.

Many obstacles continue to obstruct study in this subject. Indeed, the type of data to be considered is quite diverse and is dependent on both the environment (time, place, weather, culture, etc.) and the user (motion speed, emotions, heart rate, device luminosity, etc.). An even more serious problem is the scarcity of data for research purposes. In the actual world, retrieving them is difficult as well, because users may not always wish to communicate as much data from their cell phone sensors.

### 4) Hybrid approach

It is also feasible to develop a hybrid recommendation system by combining the preceding complementing strategies. It can also be based on less well-known techniques like location-based recommendations. This strategy can help with cold start and sparsity issues. Several implementations may be put up, the first of which combines the recommendation systems into one.

A hybrid method also allows for the evaluation of more criteria for each proposal. Demographic filtering, a sort of collaborative filtering that puts people in the same demographic together, can also be applied into such a system. Our recommender system uses both content-based and real-time data to filter results, making it a hybrid method. As previously said, we would have preferred to take a collaborative approach to address some of the difficulties with content-based filtering.

Personalized suggestions in the form of item rankings will arise from both collaborative and content-based screening. Because it compares the similarity of characteristics on audio signals, our method may be classified as content-based music recommendation. Using real-time data does not solve the difficulties with content-based filtering; rather, it is meant to improve the suggestions by customizing them depending on the user's present condition.

It's also feasible to maintain many systems distinct and give weights to them, as well as the option to switch between them at anytime. Finally, outputs from one system may be extracted and utilized as input for a subsequent system.

# Chapter 2 : Problem Statement and Methodology

## Problem Statement

With commercial music streaming service which can be accessed from mobile devices, the availability of digital music currently is abundant compared to previous era. Sorting out all this digital music is a very time-consuming and causes information fatigue. Therefore, it is very useful to develop a music recommender system that can search in the music libraries automatically and suggest suitable songs to users.

In the long-term, the goal is not only to recommend existing songs but also to generate songs adapted to the musical taste of the user. During this master thesis I focused on the recommendation part while exchanging with a colleague in charge of the generation part. The music which selected by the user is used as the basis music for recommendations. The features of basis music are extraction vector is obtained from the best genre prediction model in previous step. Next, the values of cosine similarity are sorted from the largest to the smallest value. The future of the project will consisting gathering these two parts in order to have a fully functional recommendation system.

Because real-time data changes rapidly, an algorithm based on it must be efficient. We want advice that are relevant to the current situation rather than prior situations. Many researchers are presently focusing on machine learning approaches such as neural networks, and they are also becoming more prominent in the field of recommender systems. In terms of results, the major goal was to establish a framework for consumers to use in order to assist them find the ideal tunes for them. This project seeks to discover the correlation and similarity between different songs, and then construct a recommendation system framework that suggests new music for your Spotify playlist based on that information.

They can not only manage the ever-increasing amount of data, but they also increase in quality in proportion to the amount of data evaluated, thanks to the learning algorithms. Indeed, during years, in order to choose music, restaurants, movies, etc.. We have been asking our friends, family, and colleagues to recommend something they liked. And it is this mechanism that is attempted to be reproduced here. Netflix was a pioneer of this method (based on stars given by other users) but it is now widely used, including for Spotify’s Discover Weekly. Collaborative filtering makes suggestions based on the collaborative power of the available evaluation by users.

Machine learning has become significantly more viable than it has been traditionally as the amount of data has expanded and the processing capacity of computers has improved. The algorithms are made to look for patterns.

Because our music choices and our present emotional state are so closely linked, realtime data sources are extremely important for music suggestions. Certain songs or styles of music can affect our mood in various ways, and our musical choices are frequently linked to our mood.

Music selections are also linked to the listener's current activities. Even if we have a certain musical taste, our tastes will shift depending on what we are doing. When a person is working out at the gym, for example, they will likely listen to different tunes than when they are attempting to go asleep at night. 15 The aim of this thesis is to explore the different recommendation approaches, the available datasets, the ways to take into account the user’s preferences and the machine learning methods in order to build a suitable recommendation system. The music which selected by the user is used as the basis music for recommendations. The features of basis music are extraction vector is obtained from the best genre prediction model in previous step. Next, the values of cosine similarity are sorted from the largest to the smallest value. One important part was only dedicated to determine how to evaluate this recommendation system. This project will be introduced to the members of the company and will take the form of an application. The user will be asked to upload a music and the application will recommend some music to be listened to afterwards.

## Methodology

**Implementation of Music Recommender System**

The music recommendation system is implemented as a hybrid model, leveraging both content-based and collaborative filtering techniques. Below is a breakdown of the key components in the system's implementation:

**1. Data Preparation:**

The dataset is structured as a Pandas DataFrame (df), containing essential information about songs, including the artist, song name, link (if available), and lyrics text.

The lyrics text undergoes preprocessing using the NLTK library, involving tokenization and stemming, to prepare the data for analysis.

**2. Content-Based Filtering:**

Utilizing scikit-learn's TfidfVectorizer, the lyrics text is transformed into a TF-IDF matrix, capturing the importance of words within each document.

Cosine similarity is calculated based on the TF-IDF matrix, resulting in a similarity matrix (similar\_content).

The content\_based\_recommender function uses this matrix to recommend songs based on content similarity when given a specific song name.

**3. Collaborative Filtering:**

The collaborative filtering aspect is implemented using the Surprise library, specifically the KNNBasic algorithm.

The data is structured for collaborative filtering, with a Reader specifying the rating scale, and a Dataset is created from the DataFrame.

The dataset is split into training and testing sets using train\_test\_split.

The collaborative filtering model is trained on the training set using the fit method.

**4. Hybrid Recommendation:**

The hybrid\_recommender function combines recommendations from both content-based and collaborative filtering approaches.

Content-based recommendations are emphasized, and the top 10 collaborative filtering recommendations are included.

This hybrid approach aims to enhance the recommendation quality by leveraging both content information and user-item interactions.

# Chapter 04: Implementation

## 4.1 Date Set Used in the Major Project

### Spotify Million Song Dataset

A dataset containing songs, artists names, link to song and lyrics

Source:<https://www.kaggle.com/datasets/notshrirang/spotify-million-song-dataset/data>

## 4.2 Date Set Features

### 4.2.1 Types of Data Set

The dataset is in the form of csv files which have been taken from kaggle and then various data mining techniques are applied to it to extract the information. The dataset was divided in four parts namely data by link, data by artist, data by song and data by text and were stored in individual csv files to analyze and to train the model.

### 4.2.2 Number of Attributes, fields, description of the data set

● audio features of song name44824 unique values

● audio features of artists, 643 unique values

● audio features of link ,57650 unique values

● audio features of text , 57494 unique values

****

## 4.3 Design of Problem Statement

We used numerous methods to create, build, and assess music recommendation systems in this study. Music suggestion is a difficult challenge since we must structure music in such a way that we can propose users' favorite songs, which is never a sure thing. It is dynamic, and it is occasionally influenced by variables other than the listening history of users or songs. Music suggestion is a difficult challenge since we must structure music in such a way that we can propose users' favorite songs, which is never a sure thing. It is dynamic, and it is occasionally influenced by variables other than the listening history of users or songs. Music suggestion is a difficult challenge since we must structure music in such a way that we can propose users' favorite songs, which is never a sure thing. It is dynamic, and it is occasionally influenced by variables other than the listening history of users or songs. Music suggestion is a difficult challenge since we must structure music in such a way that we can propose 30 users' favorite songs, which is never a sure thing. It is dynamic, and it is occasionally influenced by variables other than the listening history of users or songs.

We'll take the following method to solving the problem:

i. Collecting information (choosing dataset)

ii. Using data mining techniques to analyze a dataset (data pre-processing and data cleaning)

iii. Preparing the dataset for usage in various methods

iv. Choosing the best algorithms for the job

v. Using different machine learning algorithms

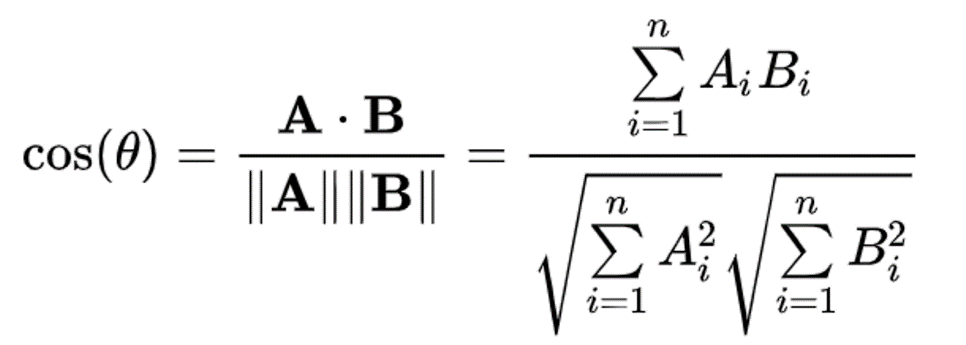
vi. Analyzing the data and selecting the best model

## 4.4 Algorithm / Pseudo code of the Project Problem

The algorithm used for the Project Problems is:

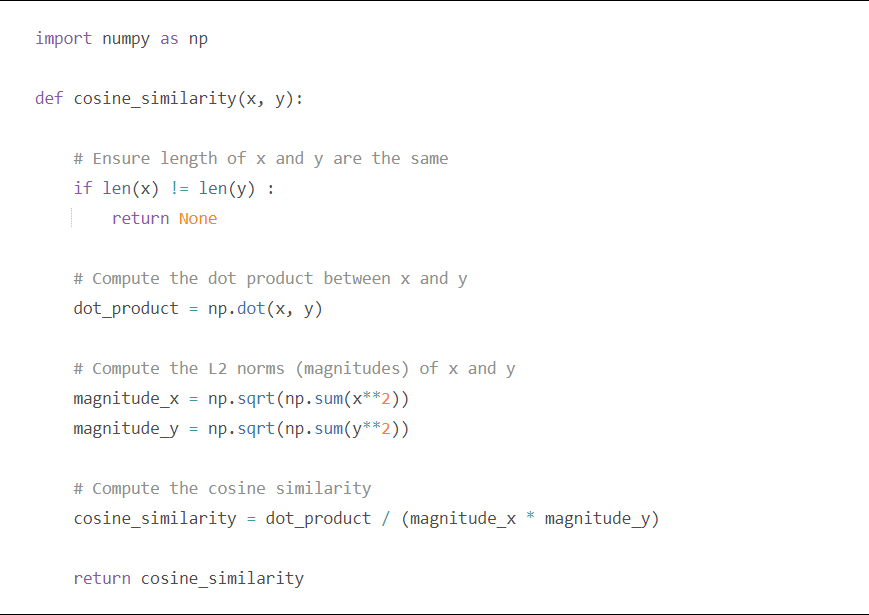
**Cosine Similarity**

* **Introduction to Cosine Similarity:**
  + Cosine similarity is a metric used to measure the cosine of the angle between two non-zero vectors in an inner product space.
  + In the context of the song recommender system, each song's TF-IDF vector is considered a multi-dimensional vector, and the cosine similarity between these vectors reflects the similarity of their lyrical content.
* **Cosine Similarity Matrix:**
  + The cosine similarity matrix is a square matrix where each element represents the cosine similarity between two songs.
  + The matrix is symmetric, as the similarity between Song A and Song B is the same as the similarity between Song B and Song A.
  + A higher cosine similarity indicates greater similarity in the lyrical content of the songs.



* **Significance for Recommender Systems:**
  + Cosine similarity is particularly useful in text-based recommender systems because it captures the orientation (direction) rather than the magnitude of vectors.
  + For song lyrics represented as vectors, cosine similarity measures the cosine of the angle between the vectors, providing a measure of similarity regardless of the vector's magnitude.
* **Calculation Process:**
  + The **cosine\_similarity** function in scikit-learn calculates the cosine similarity between all pairs of songs.
  + For each pair of songs, it computes the dot product of their TF-IDF vectors and divides it by the product of their vector magnitudes.
  + The result is a similarity score ranging from -1 (completely dissimilar) to 1 (perfectly similar).

In Swift, here's a fast implementation of the cosine similarity logic.



* **Sorting and Recommendation:**
  + After calculating the cosine similarity matrix, the system sorts the similarities for a given song in descending order.
  + The sorted list represents the songs in the dataset, ordered by their similarity to the input song.
  + The recommender function then extracts the indices of similar songs, excluding the input song itself.
  + Finally, it retrieves the names of these recommended songs based on their indices.
* **Adjusting the Number of Recommendations:**
  + The number of recommended songs (e.g., 23 in the provided code) can be adjusted based on the desired level of granularity in the recommendations.
  + A higher number may provide more diverse recommendations, while a lower number may focus on the most closely related songs.
* **Limitations:**
  + While cosine similarity is effective in capturing textual similarity, it may not consider the semantics or context of the lyrics.
  + It assumes that songs with similar words are similar in meaning, which may not always be the case.
  + It's crucial to consider user feedback and potentially explore more advanced techniques, such as collaborative filtering, to enhance recommendation accuracy.

Understanding the nuances of cosine similarity helps grasp how the recommender system identifies and recommends songs based on the textual similarity of their lyrics. This metric plays a central role in shaping the personalized song recommendations provided to users.

## TF-IDF

The abbreviation for term frequency is TF-IDF. Records having a Document Frequency that is inverse. It is the process of identifying the importance of a word in a sequence or corpus to a text. The number of times a word appears in the text increases its significance, although this is countered by the corpus's word frequency (data-set).

The TF-IDF determines the relevance of a phrase by considering its value in a single text and scaling it by its importance across all documents.

The most commonly used recommendation algorithm . We call it a “user-user” algorithm because it recommends an item to a user if similar users liked this item before. The similarity between two users is computed from the amount of items they have in common in the dataset.

The term Frequency is shortened as TF-IDF. Records with an inverted Document Frequency. It is the process of identifying the importance of a word in a sequence or corpus to a text. The meaning of a word rises in proportion to how many times it appears in the text, but this is neutralised by the corpus's word frequency.

The TF-IDF determines the relevance of a phrase by considering its value in a single text and scaling it by its importance across all documents.

**Term Frequency**: The acronym TF-IDF stands for Frequency. Records with a Document Frequency that is reversed. It is the process of determining the significance of a word in a sequence or corpus in relation to a text. The number of times a word appears in the text increases its significance, although this is offset by the corpus's word frequency.

The TF-IDF determines the relevance of a phrase by considering its value in a single text and scaling it by its importance across all documents.

The acronym TF-IDF stands for Frequency. Records with a Document Frequency that is reversed. It is the process of determining the significance of a word in a sequence or corpus in relation to a text. The significance of a word increases in proportion to how many times it appears in the text, but this is offset by the corpus's word frequency.

**Document Frequency**: This is similar to TF in that it verifies the meaning of the text across the full corpus collection. The only difference is that in document d, TF represents the frequency counter for a term t, but in document set N, df reflects the number of times the phrase t appears. To put it another way, the number of times the phrase appears in publications is DF.

df(t) = occurrence of t in documents

**Inverse Document Frequency (IDF)** is a test that assesses a word's relevance. The primary purpose of the search is to locate records that are related to the requirement. The term frequencies cannot be used to assess the weight of a phrase in the document since tf 37 considers all terms to be equally significant. Music suggestion is a difficult challenge since we must structure music in such a way that we can propose users' favorite songs, which is never a sure thing. It is dynamic, and it is occasionally influenced by variables other than the listening history of users or songs. Thanks to significant improvements in the disciplines of music information retrieval and (audio) signal processing over the last decades, content-based characteristics retrieved from music audio signals have traditionally played a significantly larger role than in other domains. DL approaches can work on a significantly wider and more comprehensive collection of low- and mid-level audio properties, thanks to established tools and expertise from these domains.

In contrast to the movie or product domains, where people frequently dislike repetitive suggestions of the same products, listeners can like repeated recommendations. The network's output is commonly a vector over items (or playlists) that provides the probability of fit, thanks to the probabilistic handling of items in DL architectures.

Music suggestion is a difficult challenge since we must structure music in such a way that we can propose users' favorite songs, which is never a sure thing. It is dynamic, and it is occasionally influenced by variables other than the listening history of users or songs. To find the document frequency of the keyword t, start by counting the number of documents that include it.

K-Nearest Neighbour

* K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
* K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
* K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
* K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
* K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
* It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

The K-NN working can be explained on the basis of the below algorithm:

* **Step-1:** Select the number K of the neighbors
* **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
* **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
* **Step-4:** Among these k neighbors, count the number of the data points in each category.
* **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
* **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:



* Firstly, we will choose the number of neighbors, so we will choose the k=5.
* Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



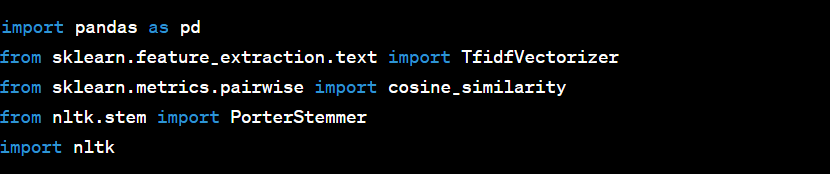
* By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:

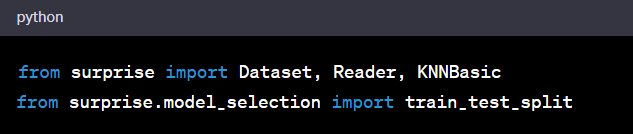


* As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

## 4.5 Screenshots of the various stages of the Project

**Step 1- Importing libraries**

****

****

**Libraries used**

1. Pandas

2. scikit-learn

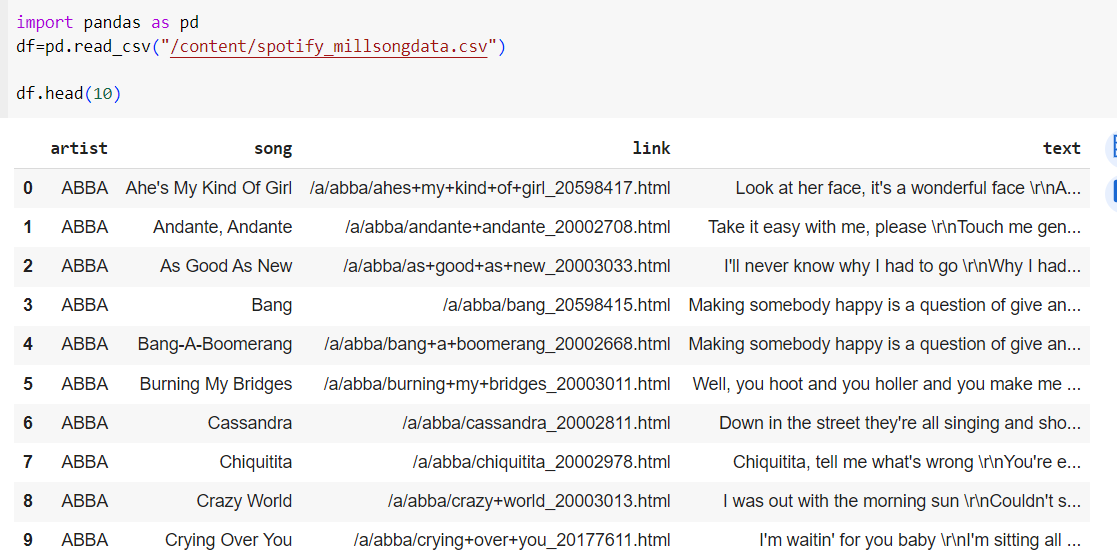
3. sklearn.metrics

4. sklearn.feature\_extraction

5. nltk.stem

6. surprise

**Step-2 Uploading**

****

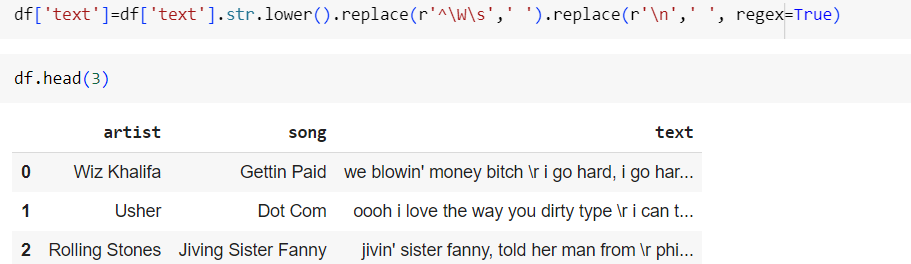
**Step-3 Cleaning the data**

Choosing a sample size of 5000 and dropping the *link* column

****

**Step-4 Preprocessing the data**

Remove the regular expressions and escape sequences

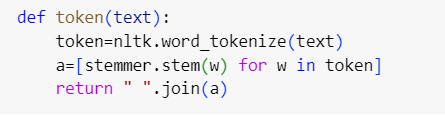
****

**Step-5 Text Tokenization:**

Define a comprehensive tokenization function (token) utilizing the NLTK library and the Porter Stemmer.

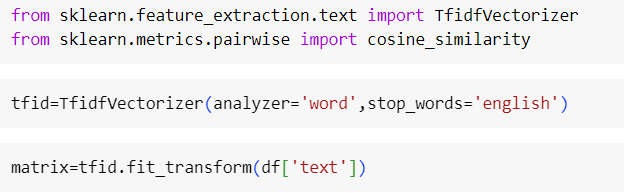
Tokenization involves breaking down lyrics into individual words and then stemming them to their root form.

Apply the tokenization function to the 'text' column, transforming raw lyrics into stemmed tokens for further analysis.

****

**Step-6 Use of TF-IDF**

It assigns a value to a term based on its importance in a document scaled by its importance across all documents in your corpus, TF-IDF is a popular approach for NLP tasks because it mathematically eliminates naturally occurring words in the English language and selects words that are more descriptive of your text. Text summarization, information retrieval, and sentiment classification are just a few of the NLP tasks that make use of TF-strong IDF's weighting operation.

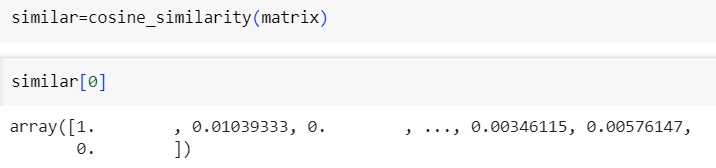
****

**Step- 7 Cosine Similarity**

Utilize the cosine\_similarity function from scikit-learn to compute the cosine similarity matrix.

The matrix serves as a representation of the similarity between each pair of songs based on their TF-IDF vectors.

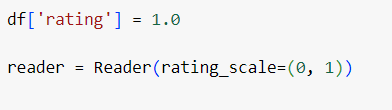
This step establishes the foundation for identifying songs with similar lyrical content.

****

**Step- 8 Add a rating column for Collaborative filtering**

The Reader class from the surprise library is used to specify how to interpret the data file when loading it into a Dataset. In this case, the rating\_scale parameter is set to (0, 1), indicating that the expected range of ratings in the dataset is from 0 to 1.

The rating\_scale parameter is useful for defining the range of possible ratings, especially when working with explicit feedback data where ratings typically fall within a specific numerical range.

****

**Step -9 Load the Data\_Frame into a Surprise Dataset**

The Dataset is loaded from the DataFrame, where ratings are set to 1.0, and the Reader is used to specify the expected rating scale**.**

**Screenshot 2023-12-20 010849.png**

**Step-10 Split the test data and train data and perform KNN to fit into the model**

When you use **train\_test\_split** from scikit-learn with **test\_size**=0.2, the function randomly splits the dataset into a training set and a testing set. The training set contains 80% of the data, and the testing set contains 20% of the data.

The **sim\_options** dictionary is used to specify options or parameters for similarity-based collaborative filtering algorithms in the surprise library. In this particular example, it is configuring the options for a collaborative filtering algorithm based on cosine similarity. Let's break down the key components of **sim\_options**:

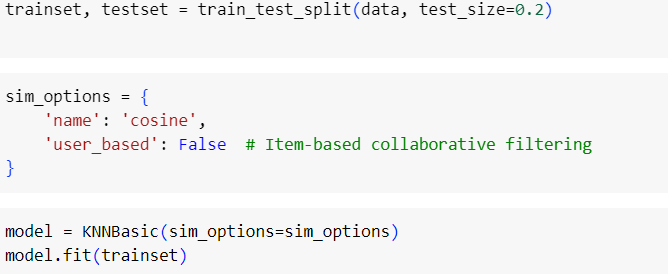
*'name':* ***'cosine'****:*

This indicates the similarity metric to be used. In this case, it's set to 'cosine', meaning that the algorithm will use cosine similarity to measure the similarity between items (or users, depending on the context).

*'user\_based':* ***False****:*

This parameter determines whether the collaborative filtering should be user-based or item-based. When set to False, as it is here, it means item-based collaborative filtering. In item-based collaborative filtering, the similarity between items is calculated based on user ratings for those items. In other words, it recommends items similar to those a user has liked or interacted with.

The fit method is where the actual training of the collaborative filtering model occurs. During training, the algorithm computes the similarity between items based on the training set. The resulting model is then capable of making predictions for unseen items based on their similarity to items in the training set.

****

**Step-11 Content Based Recommendation Function:**

Develop a recommendation function (recommender) that takes a song name as input.

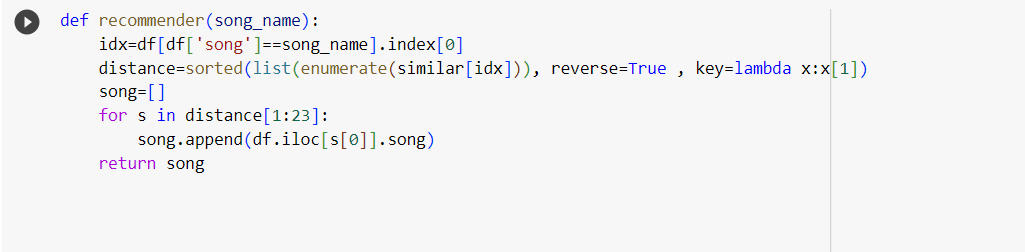
Locate the index of the input song within the DataFrame to identify its position in the similarity matrix.

Calculate cosine similarity with all other songs, sorting them in descending order to identify the most similar ones.

Extract the indices of similar songs, excluding the input song.

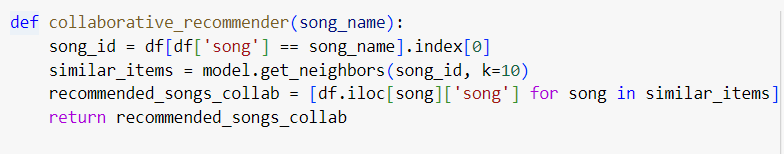
Retrieve the names of recommended songs based on their indices.

Return a list of recommended songs for the user's consideration.

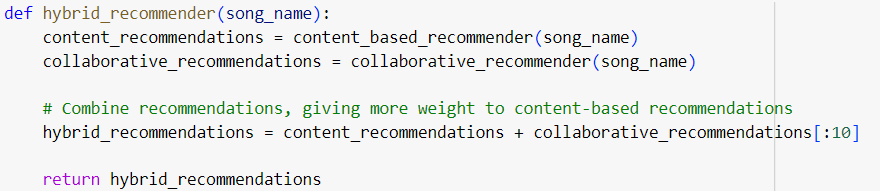


**Step-12 Collaborative Filtering Based Recommender Function**

* **idx** = df[df['song'] == **song\_name**].index[0]: This line finds the index of the input song **(song\_name**) in the **DataFrame** (df). It assumes that the 'song' column contains unique song names, and it retrieves the index of the first match.
* **distance** = sorted(list(enumerate(similar\_content[idx])), reverse=True, key=lambda x: x[1]): The variable similar\_content is assumed to be a cosine similarity matrix calculated based on the content (e.g., TF-IDF) of the songs.
* **enumerate**(**similar\_content[idx]**) pairs each song's similarity score with its index in the similarity matrix. sorted then sorts these pairs in descending order based on similarity scores.
* **recommended\_song\_indices** = [s[0] for s in distance[1:23]]:This line extracts the indices of the most similar songs, excluding the input song itself. It takes the top 22 similar songs (indices 1 to 22) because index 0 corresponds to the input song.
* **recommended\_songs\_content** = [df.iloc[s]['song'] for s in recommended\_song\_indices]:This line retrieves the names of the recommended songs based on their indices. It uses df.iloc[s]['song'] to get the 'song' column value for each index s.

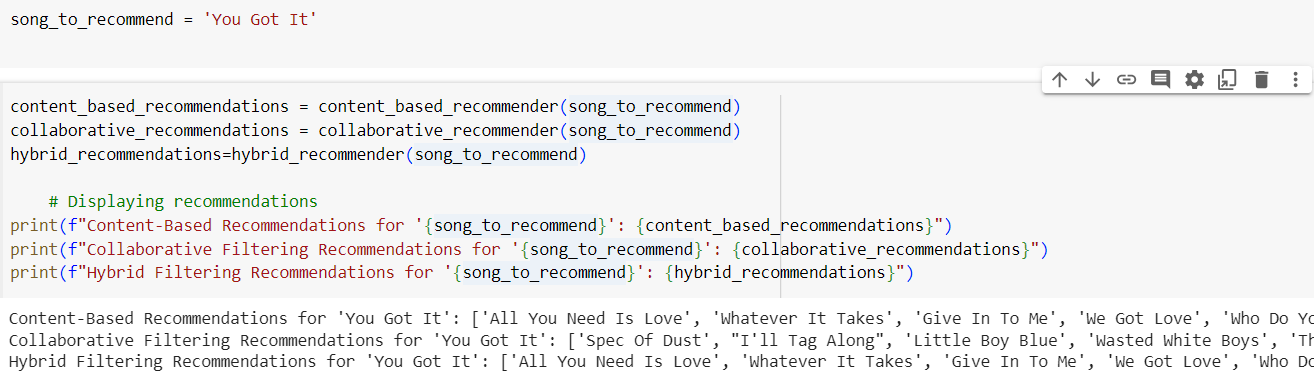


**Step-13 Hybrid Filtering**



**Step - 14 Recommend Songs**

The music supplier can forecast and then give acceptable songs to its consumers using a music recommender system based on the qualities of previously heard music. for each song listened to by the user, the average vector of audio and metadata attributes Find the n-closest data points in the dataset (excluding points from the user's listening history) to this average vector. Take these n points and come up with some tunes to go with them.



# Chapter 05: CONCLUSION

## 5.1 Application of the Project

The use of this project is that the application will also be able to record an extract of a music being played. The music supplier can forecast and then give acceptable songs to its consumers using a music recommender system based on the qualities of previously heard music. for each song listened to by the user, the average vector of audio and metadata attributes Find the n-closest data points in the dataset (excluding points from the user's listening history) to this average vector. Take these n points and come up with some tunes to go with them. The music supplier can forecast and then give acceptable songs to its consumers using a music recommender system based on the qualities of previously heard music. From the music or extract, the application will offer the possibility to listen to recommended songs by the algorithm developed in project. The program will eventually be able to record an excerpt of music being played. Using a music recommender system based on the attributes of previously heard music, the music provider may foresee and then provide suitable songs to its customers. The application will allow users to listen to songs selected by the algorithm built in the project based on the music or extract.The music supplier can forecast and then give acceptable songs to its consumers using a music recommender system based on the qualities of previously heard music. for each song listened to by the user, the average vector of audio and metadata attributes Find the n closest data points in the dataset (excluding points from the user's listening history) to this average vector. Take these n points and come up with some tunes to go with them.

A research on the limits of an interactive music recommendation service based on artificial audio similarity calculation was provided. A number of computer experiments, as well as a review of real download data, reveal that a large chunk of the audio collection is only never or never suggested. A number of computer experiments are used to investigate this issue in depth, including the investigation of various audio similarity functions and comparisons with real download data. Our music recommendation service uses Gaussian mixtures as statistical models to determine timbre similarity. This is the de facto standard method for computing audiosimilarity, and it is recognised to produce high-quality results.

## 5.2 Limitation

This project due to the nature of the dataset fails to provide accurate recommendations as the dataset does not consist of all of the songs in the playlist. This is noteworthy that a dataset can be built and versioned entirely from one data source for user convenience and technical simplicity. That is, data sources in a dataset cannot presently be mixed and matched (for example, a dataset built from a GitHub repository cannot include files uploaded from your local system). You can build numerous datasets and add them both to a Notebook if you want to leverage multiple distinct data sources in it. Songs that are identical to a large number of other songs and hence appear unnecessarily frequently in recommendation lists prevent a big section of the audio library from being recommended at all. A number of computer experiments are used to investigate this issue in depth, including the investigation of various audio similarity functions and comparisons with real download data. For user convenience and technological simplicity, a dataset can be generated and versioned fully from a single data source. That is, data sources in a dataset cannot be mixed and matched at this time (for example, a dataset built from a GitHub repository cannot include files uploaded from your local system).

The programme will allow users to listen to recommended tracks based on the music or extract. A dataset can be built and versioned entirely from one data source for user experience and technical simplicity.The music supplier can forecast and then give acceptable songs to its consumers using a music recommender system based on the qualities of previously heard music. for each song listened to by the user, the average vector of audio and metadata attributes Find the n-closest data points in the dataset (excluding points from the user's listening history) to this average vector. Take these n points and come up with some tunes to go with them. That is, data sources in a dataset cannot presently be mixed and matched (for example, a dataset built from a GitHub repository cannot also include files uploaded from your local workstation). You may build numerous datasets and add them both to a Notebook if you want to leverage different data sources in it.

## 5.3 Future Work

The range of characteristics covered by the recommender system is extensive. In today's generation of e-services and commerce, it is growing and evolving. However, there is a requirement to create and optimize the working and output of the recommender system at the same time. For user convenience and technological simplicity, a dataset can be generated and versioned fully from a single data source. That is, data sources in a dataset cannot be mixed and matched at this time (for example, a dataset built from a GitHub repository cannot include files uploaded from your local system).

The programme will allow users to listen to recommended tracks based on the music or extract. 57 Several service providers provide consumers with a shopping list. However, this is insufficient since consumers have varying preferences and decisions that are influenced by a variety of circumstances and restrictions. It may also be impossible to propose specific things to individual users in many circumstances. As a result, there is potential for combining several dimensions into music recommender systems in particular.

We were unable to create a model utilising singular value decomposition and support vector machines due to a lack of time. Because popularity-based models are adept at making suggestions, we'll aim to utilise it to forecast the top-N songs for the users who are most popular at any given time. ndeed, during years, in order to choose music, restaurants, movies, etc.. We have been asking our friends, family, and colleagues to recommend something they liked. And it is this mechanism that is attempted to be reproduced here. Netflix was a pioneer of this method (based on stars given by other users) but it is now widely used, including for Spotify’s Discover Weekly. Collaborative filtering makes suggestions based on the collaborative power of the available evaluation by users.

Customers are less likely to use the majority of the items and services offered by various e commerce sites since they are pricey. As a result, you won't be able to accurately and properly rank an item or collection of things. As a result, typical recommender system strategies are inadequate. This paves the path for more research and development in the form of an efficient recommender system that also considers constraints. We'll aim to utilize it to forecast the top-N songs for the users who are most popular at any given time.

Discover Weekly is a 30-song playlist that includes music that are similar to what the user is listening to. This, like its daily mixes and tailored playlists, is made possible by AI and big data. The system also considers the user's streaming history and playlists, as well as their current music preferences, to improve this suggestion.

Spotify is planning to launch a live audio streaming function in order to improve its users' 'tailored' experience. Locker Room, a live audio app for creating conversations about music and culture, was recently bought by Spotify.

It will allow users to listen to recommended tracks based on the music or extract. Several service providers provide consumers with a shopping list. However, this is insufficient since consumers have varying preferences and decisions that are influenced by a variety of circumstances and restrictions. It may also be impossible to propose specific things to individual users in many circumstances. As a result, there is potential for combining several dimensions into music recommender systems in particular.

## REFERENCES

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

2. Spotify. (2013, March) Spotify Music for everyone. [Online].

3. https://www.spotify.com/us/ [10] (2013, March) Wikipedia. [Online].

4. Peter Knees and Markus Schedl. Music retrieval and recommendation – a tutorial overview. In Proceedings of the 38th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), Santiago, Chile, August 2015.

5. Peter Knees and Markus Schedl. Music Similarity and Retrieval: An Introduction to Audioand Web-based Strategies. 2016.

6. Chris Johnson. From idea to execution: Spotify’s discover weekly. November 2015.

7. Zhiwei Gu Li Guo and Tianchi Liu. Music genre classification via machine learning.

8. [11] Vikas Sindhwani Prem Melville, "Recommender Systems," in Encyclopedia of Machine Learning. Yorktown Heights, USA: IBM T. J. Watson Research Center, 2010, ch. 00338.

9. Beatriz C. F. de Azevedo Glaucia M. Bressan and Elisangela Ap. S. Lizzi. 60