MUSIC RECOMMENDER SYSTEM USING MACHINE LEARNING ALGORITHM

FOONG KIN HONG

BACHELOR OF COMPUTER SCIENCE

(SOFTWARE ENGINEERING)

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	: Foong Kin Hong
Date of Birth	
Title	: Music Recommender System Using Machine Learning Content-based Filtering Technique
Academic Session	: Sem 2 2021/2022

I declare that this thesis is classified as:

☐ CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐ RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☑ OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

_______________ (Student's Signature)	_______________ (Supervisor's Signature)
	MOHD ARFIAN BIN ISMAIL
New IC/Passport Number	Name of Supervisor
Date: 01/06/1999	Date: 20/01/2023

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name
Thesis Title

Reasons (i)

(ii)

(iii)

Thank you.

Yours faithfully,

[illegible]

TS. DR. MOHD ARFIAN BIN ISMAIL
SENIOR LECTURER
[illegible]
COLLEGE OF COMPUTING & APPLIED SCIENCES
UNIVERSITI MALAYSIA PAHANG
26600 PEKAN
PAHANG DARUL MAKMUR
TEL : 09-424 4673 FAX : 09-424 4666

____
(Supervisor's Signature)

Date: 20/01/2023

Stamp:

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Bachelor of Computer Science (Software Engineering) with Honors.

[illegible]

(Supervisor's Signature)

Full Name	: MOHD ARFIAN BIN ISMAIL
Position	: Senior Lecturer
Date	: 20/01/2023

Universiti Malaysia PAHANG

Engineering • Technology • Creativity

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

FOONG KIN HONG

____

(Student's Signature)

Full Name	: FOONG KIN HONG
ID Number	: CB19053
Date	: 23 May 2022

MUSIC RECOMMENDER SYSTEM USING MACHINE LEARNING CONTENT-BASED FILTERING TECHNIQUE

FOONG KIN HONG

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Computer Science (Software Engineering)

Faculty of Computing

UNIVERSITI MALAYSIA PAHANG

MAY 2022

ACKNOWLEDGEMENTS

First and foremost, my utmost appreciation goes to my supervisor, Ts. Dr. Mohd Arfian Bin Ismail for his detailed and constructive comments in every stage of my work. Throughout the course of my Final Year Project research, he never failed to provide me continuous encouragement and invaluable advice. His continual assistance and great patience have been the key to the completion of this research. This research would not have been possible without the support from the administrative and teaching staff of Universiti Malaysia Pahang (UMP). I would like to thank all my friends and colleagues for their support and effort in assisting me in my PSM1 and PSM2.

Finally, I'd want to express my gratitude to all of my friends, siblings, and lecturers for their prayers and support throughout the creation of this project, from the beginning to the conclusion, when the system was successfully constructed.

ABSTRAK

Kemajuan berterusan trend dan teknologi pembangunan web telah menghasilkan sejumlah besar sistem web yang kerap dilawati secara tetap. Antara sistem web yang telah diwujudkan, terdapat sistem yang membolehkan pengguna mendengar muzik dalam talian tanpa perlu memuat turunnya ke peranti mereka. Dengan peningkatan populariti penstriman muzik, sistem pengesyor muzik merupakan instrumen penting untuk meningkatkan penggunaan muzik digital. Pembelajaran Mesin ialah satu bentuk Kecerdasan Buatan yang akan menjadikan sistem berfikir seperti manusia. Pembelajaran Mesin membolehkan sistem belajar secara beransur-ansur untuk meningkatkan ketepatannya dalam meramalkan hasil masa hadapan. Objektif projek ini adalah untuk membangunkan sistem pengesyoran muzik menggunakan salah satu teknik Pembelajaran Mesin iaitu teknik penapisan berasaskan kandungan. Matlamat kajian ini adalah untuk mengkaji tentang sistem pengesyor muzik bagaimana ia dilaksanakan dan untuk mereka bentuk dan membangunkan sistem pengesyor muzik. Dalam kajian ini, kaedah K-Mean Clustering, Euclidean Distance dan Cosine Similarity dilaksanakan. Ini adalah algoritma popular untuk pembelajaran tanpa pengawasan, kaedah pembelajaran mesin untuk menganalisis dan kumpulan set data. Algoritma ini mengenal pasti corak tersembunyi atau kumpulan data tanpa bantuan manusia. Ia adalah pilihan terbaik untuk analisis data penerokaan kerana keupayaannya untuk mencari persamaan dan perbezaan maklumat. Berdasarkan analisis pada muzik yang didengari pengguna semasa penggunaan, sistem akan menentukan nilai ciri lagu tersebut. Ini membolehkan algoritma memilih lagu yang serupa selepas pengiraan dalam pangkalan data paling sesuai dengan minat pengguna pada bila-bila masa. K-Mean Clustering akan mengelompokkan data, mengikut persamaan setiap lagu, memisahkannya mengikut kumpulan yang berbeza. Cosinus Similarity akan mengira jarak kosinus dengan data lain dan mengesyorkan satu dengan jarak yang lebih pendek. Jarak Euclidean akan mengira jarak terus antara dua vektor dan mengesyorkan satu dengan jarak yang lebih pendek.

ABSTRACT

The constant advancement of web development trends and technology has resulted in a big number of web systems that are frequently visited on a regular basis. Among the web systems that have been established, there are systems that allow users to listen to music online without having to download it to their devices. With the increasing popularity of music streaming, music recommender systems are important instruments for increasing digital music consumption. Machine Learning is a form of Artificial Intelligence that will make systems to think like human being. Machine Learning allows a system to learn gradually to improve its accuracy in predicting future outcome. The objective of this project is to develop a music recommendation system using one of the Machine Learning techniques which is content-based filtering technique. The aim of this study is to study about music recommender system on how it is implemented and to design and develop a music recommender system. In this study, methods of K-Mean Clustering, Euclidean Distance and Cosine Similarity are implemented. These are the popular algorithm for unsupervised learning, a machine learning method to analyse and cluster datasets. These algorithms identify hidden patterns or data groupings without the assistance of a human. It is the best option for exploratory data analysis because of its ability to find informational similarities and differences. Based on analysis on music user listen to during usage, the system will determine the feature values of that song. This allows the algorithm to select similar songs after calculation in database would best match the user's interests at any given time. K-Mean Clustering will cluster the data, according to the similarities of each song, separate them by different group. Cosine Similarity will calculate the cosine distance with other data and recommend the one with shorter distance. Euclidean Distance will calculate the direct distance between two vectors and recommend the one with shorter distance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

$I(u)$	a set of items rated by user *u*
$I(v)$	a set of items rated by user *u*
w_{uv}	cosine similarity between users *u* and *v*
N_i^u	set of neighbours that are most similar to user *u* and rated item *i*
v	a user that belongs to N_i^u
Sim_{uv}	similarity value between user *u* and *v*
p_{ui}	predicted rate, *k*-quantity of testing rates
r_{ui}	the true rating value that user *u* gave to item *i*
$\hat{r}_{ui}$	predicted rating value that user *u* gave to item *i*
$\lvert\widehat{R}\rvert$	total number of predicted ratings

LIST OF ABBREVIATION

P2P	Peer-to-peer
MP3	MPEG Audio Layer-3
ML	Machine Learning
AI	Artificial Intelligence
i.e.	Id est
URL	Uniform Resource Locator
OS	Operating System
TF	Term Frequency
IDF	Inverse Document Frequency
API	Application Programming Interface
kNN	k-Nearest Neighbour
RMSE	Root Mean Square Error
MAE	Mean Absolute Error

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Over the years, the ever-escalating development of the Internet has changed the lifestyle of current society in various ways related to communication and lifestyle. Systems such as peer-to-peer software that allow complex file sharing with others who also accessing the system becoming more and more popular, as the result of growing of bandwidth. As one of the pioneering peer-to-peer (P2P) file sharing software, Napster is launched as an audio streaming service. This technology allowed people to easily-shared their MP3 file with other users and music sharing starts to become a norm from here. The P2P file sharing system revolutionized the music industry together with the habits of people related to musical collect and playback.

The constant advancement of web development trends and technology has resulted in a big number of web systems that are frequently visited on a regular basis. Among the web systems that have been established, there are systems that allow users to listen to music online without having to download it to their devices. This technology solves some issues that arise from peer-to-peer software, one of them is the requirement of a large storage space in order to download and stream a wide variety of music. The next is the music copyright legal issue, big music distribution companies started legal battles against some peer-to-peer software owner. Despite some peer-to-peer software still operates nowadays, this web music services has become a major way in music sharing.

The software or websites that provide music listening services have large music collections in order to make their services available to a large number of people. The copyrights issue for each nation will be handled by these music listening services. They adapt their musical catalog according to the copy and reproduction rights of the musical label associated to each music distribution companies or individual artists. The majority of these music service companies charge for their services, while some offer free access to the musical collection but not reproduction. Many of the music streaming systems are evolving and has a significant improvement over the years. There exist some simple systems which user provides and shares their own playlist like 8tracks.com, and also complex systems with user's playlist together with recommendation streaming functionality such as Spotify.com.

Recommender systems aim to estimate and predict users' interests or content preference and recommend product items related to their preference. They are the most popular and powerful machine learning systems that implement in almost every field. Music Recommenders have the chance of making accessible to users not only the common, popular music, whilst new emerging groups, minor rare music and some independent label's productions. Recommendations will speed up the search process of the users, leading them to the content they are interested in, and sometimes, surprise them with similar contents they would have never searched for. There are a lot for good reasons to implement a recommender system in music listening service providers. One of them being the promotion of certain artists, whose quality of their musical works are noteworthy.

The importance of implementing recommender system in a music listening service providers is significant. First and foremost, they will handle the duty of filtering and choosing new music for the new listener. It is either users want to explore new music or just simply want to listen to genre they never listen before, recommender system will play the music based on users' selection. On the other hand, for experienced listeners who want to continue explore more about their favorite artists, or even explore similar artists' music, recommender system will also handle their job in suggesting those music. In short, a music recommender system is important to improve user experience while listening to music.

There are a few ways to implement this music recommender system, which include collaborative filtering, content-based filtering and also a hybrid recommender system. Collaborative filtering method recommend musical works to users by considering how someone else rated them. It is based on past interactions recorded between users and items as well as similar decisions made by another users. As an example, suppose that there is a user *A* who likes music *X*. If there are a lot of users who like music *X* also like music *Y*. Then, music *Y* will be recommended to user *A*. As for content-based filtering, it recommends musical works similar to users' favorites in term of musical properties, it uses users' profile item and features as reference. Similar music in the past is grouped based on their properties and will be recommended again in the near future, based on the hypothesis of item interested in the past will be interested again in the future. The last method is the hybrid recommender system which is the combination of both approaches above.

Machine Learning (ML) is a form of Artificial Intelligence that will make systems to think like human being. Machine Learning allows a system to learn gradually to improve its accuracy in predicting future outcome. The machine learning systems have three main parts in general, a decision process which make estimation based on users' input data, an error function to evaluate the prediction of the model, and lastly a model optimization process to improve system's accuracy. Machine Learning is widely implemented in systems like fraud detection systems, spam filtering systems and also recommendation systems.

In this study, Machine Learning will be implemented in the music recommender system. The technique of content-based filtering is chosen in this recommender system. Content-based filtering focus on the relationship between items, it recommends items based on the similarity with other items. It is based on the value of sound features of each song, such as danceability, energy, loudness and etc. Based on the music preferences of the user, the music recommender system will recommend music of the similarity.

1.2 Problem Statements

Although more and more music recommender systems have been developed and implemented in different sites, such systems, however, are still far from flawless. Sometimes, it still generates some unsatisfactory suggestions. This is due to the fact that users' preferences for musical works are influenced by a variety of elements that are not considered sufficient enough in current Music Recommender System approaches, which are centered on interaction between users and items, as well as content-based item descriptors. Therefore, there are still several issues that the field of music recommender systems research is facing.

There are some major aspects that differs music recommender system from other recommendation system like movies recommender system, book recommender system or products recommender system. These aspects are important as it relates to the problems faced while developing a music recommender system. These aspects include: -

i. Duration of items - musical works ranges between 2 and 5 minutes unlike movies of 90 minutes
ii. Magnitudes of item - music catalogs contain millions of music pieces while movies only up to ten thousand

iii. Sequential consumption - music pieces are arranged sequentially, hard to identify the correct arrangements
iv. Recommendation of previously recommended items - same music pieces will be recommended again in the future unlike movies or books
v. Consumption behavior - musical pieces are usually consumed passively in the background
vi. Listening intent & purpose - different intent and purpose of people listening to music

One of the problems faced by music recommender system is the ‘cold start problem’. Most of the music recommender system have this problem. When a new user logs in or a new piece of music is added to the music library, there is insufficient data linked with this user/music in the system, therefore the systems cannot properly recommend other relevant musical pieces.

Another problem of music recommender system is the sparsity problem. Sparsity problems refer to situations in which transactional or feedback data is sparse, number of given ratings is significantly less than the expected ratings, usually happened when there are a large number of users and items. This situation makes the system to have insufficient data to identify and recommend suitable musical pieces to users.

The next problem is the continuation of automatic playlist. A playlist is a collection of music songs that have been prepared and are intended to be listened to by users. In this context, different approaches have been tried and continuously improving, some of them are Markov chain and log-likelihood method. As a variation of automatic playlist generation, the task of automated playlist continuation includes the addition of one or more tracks to a playlist in a way that matches the attributes of original playlist. This variation helps user in listening to a more compelling playlist without any extensive musical familiarity. In short, the main challenge of automatic playlist continuation is to correctly estimate the intended purpose of a specific playlist and recommend music similar to its properties. This is difficult because of the wide range of intended purpose and the diversity in these underlying features. Table 1.1 below give the summary of the problem statements in this study.

Table 1.1- Summary of problem statement

No.	Problem	Description	Effect
1	Cold start problem	When a new user log into the system or a new music piece is added to the music catalog, the system does not have sufficient data associated with this user/ music, therefore the systems cannot properly recommend other relevant musical pieces.	Bad user experience and is time consuming as users need to manually search for songs.
2	Sparsity problem	Situations in which transactional or feedback data is sparse, amount of given ratings is significantly less than the expected ratings, makes the system to have insufficient data to identify and recommend suitable musical pieces to users.	Low accuracy of recommended item meet the users' preference of songs
3	Continuation of automatic playlist	Most of the music streaming systems have difficulties in the continuation of generating a large sequence of music tracks and intended to be listened by users for long hours.	After a certain period of time, songs that played by the system will varies from the users' preferences.

1.3 Aim & Objectives

The aim of this study is to build a music recommender system by using machine learning technique of content-based filtering. There are three objectives that in this project that will be achieved: -

i. To study on music recommender system on how it is implemented.
ii. To design and develop a music recommender system.
iii. To evaluate the performance of the content-based filtering implemented on music recommender system.

1.4 Scope

The scope of this project is outlined below: -

i. This music recommender system is able to estimate users' musical preferences and recommend musical works according to the estimation in a given time.
ii. The users able to access a completely free system which gives users of any age the opportunities to explore new music.
iii. This system itself is a small system in discovering ways and methods in data collecting and machine learning.
iv. This music recommender system does not involve in financial transaction, and elements of commercial use will not be implemented in the system.
v. This music recommender system will be implemented using machine learning in exploring data and identifying pattern of musical pieces.

1.5 Significance of Project

This project is significant due to many reasons, most of them are the advantages of music recommender system itself.

One of the benefits of music recommender system is that it records based on the actual user behavior at that moment, and then recommend musical pieces that fit into that particular mood. The recommendations are not based purely on guesswork, but on an objective realty. Therefore, ones can listen to music that suits in a particular moment.

Next, music recommender system is great for user in terms of discovery. This is apparent when none of the music on the radio is aligned to users' tastes. Recommender system helps to solve this problem by allowing users to discover new things that are similar to what they already liking. It helps users in exploring different field of music, enriching their music playlist, which come in handy in certain mood/ environment.

Last reason for user to use music recommender system is the personalization feature. A recommender system acts like a friend to users. This friend knows users better than anyone else as they know what they like and don't like. Users will be less stressful as the system knows exactly what he/ she needs at the moment.

1.6 Report Organization

There are three chapters in this research thesis and every chapter has its own role.

Chapter 1: Introduction

This chapter describes the general background of music recommender systems. Including the general introduction, problem statements and also objectives and scope for this project.

Chapter 2: Literature Review

This chapter presented and discuss the related existing music recommender systems and related techniques used in the development of music recommender systems.

Chapter 3: Research Framework

This chapter explain the research framework of this project. Steps of development of a music recommender system is discussed and explained, together with the datasets details. The experimental environment including hardware and software that were used also studied.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

This chapter focuses on review and discussion on existing systems and methods that are related to this study. By this chapter, details of music recommender system, on what it is and how to make it works will be discussed. Various methods or techniques of machine learning that are used in recommender system will be compare and contrast in order to find the most suitable one for this study. Details such as techniques used, content, programming and security will be studied in this chapter to help in understanding and building a music recommender system.

2.2 Existing System

The existing systems that are chosen to discuss and compare are Spotify, Deezer and Apple Music. These are few of the most popular music streaming platforms across the world. In this section, these systems will be studied in terms of algorithm, techniques and tools used in their recommendation engine. Comparison will be made between these systems to find which method are more suitable to implement in this study.

2.2.1 Spotify

Spotify is one of the most popular digital music, podcast and video service provider that let us have access to millions of songs from creators all over the world. Spotify can be access through the website https://open.spotify.com. In Spotify, musical can be browsed and searched via different approaches such as artist, album, genre, playlist or record label. Spotify also allows users to create, edit and share their playlists with others. Spotify is an AI recommender system, several Machine Learning models is used and optimized for the key business goals. In order to generate user representations and item representations, different techniques of Machine Learning are used. Figure 2.1 shows the interface of Spotify suggesting daily mixes according to user's preference.

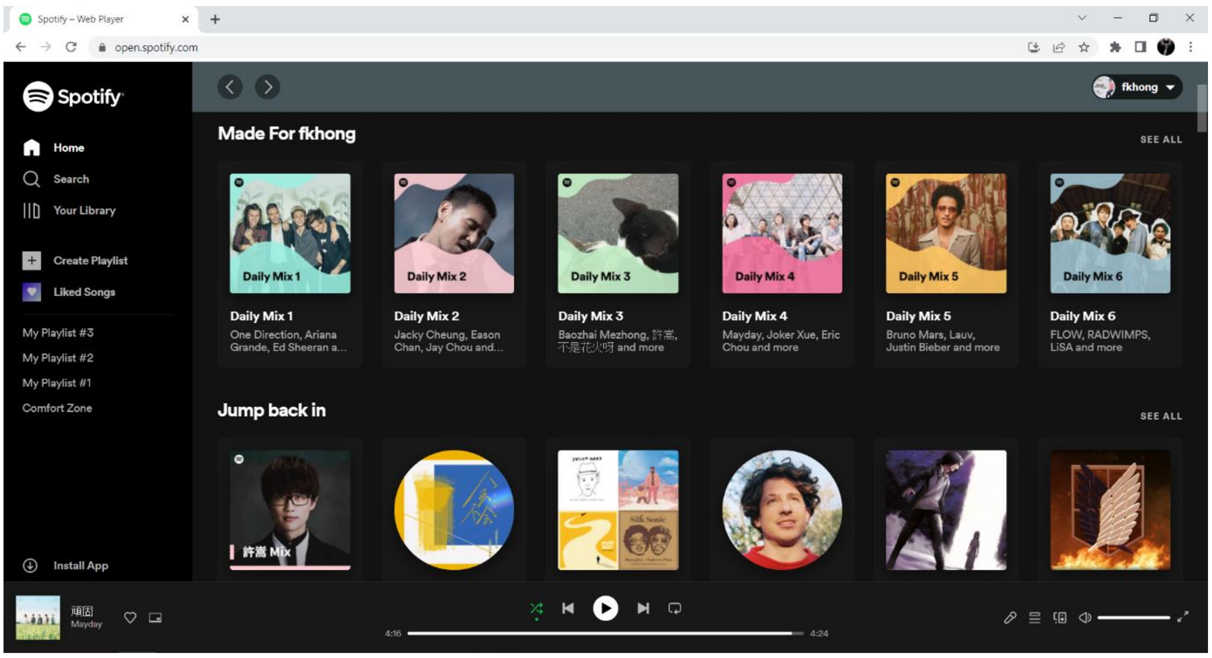


Figure 2.1- Home page interface of Spotify suggesting playlists

Spotify's approach in building its track representation consists two main components, which are content-based filtering and collaborative filtering. Content-based filtering is used to describe the track by examining the content itself while collaborative filtering is used to describe the track in its connection with other tracks exists in the same platform by studying user-generated assets. Both methods are essential as data obtained by both methods will be used to give a clearer view of the content, solving issue like cold start problem.

First is the content-based filtering algorithm of Spotify, it consists of mainly 3 processes which are analysing artist-sourced metadata, analysing raw audio signal and also analysing text with Natural Language Processing models. The processes are simplify as below:

i. Analysing artist-sourced metadata

When a new song is published successfully on Spotify, the metadata given by the distributor will be analysed by an algorithm. When all the metadata is filled correctly, the music will be added to Spotify's database, these data includes track title, release title, artist name, release date, genre tags, primary language and etc. The artist-source metadata will later become one of the input of Spotify recommender system.

ii. Analysing raw audio signal

As soon as audio files and artist-sourced metadata is added into database, raw audio analysis will be performed. The audio features data that are given in Spotify API consists of at least 12 metrics, each of them describes the sonic characteristics of the music track. These features are closely related to objective sonic descriptions. As an example, for the metric of "instrumentalness", it represents the algorithm's confidence that the music track does not have any vocals, the range of score is from 0 to 1. Besides that, Spotify also generates a few high-level features to consider on how the tracks sound like in a different perspective:

- **Danceability**, describing the track's suitability for dancing based on several combined musical aspects such as tempo, rhythm stability, beat strength and overall regularity.
- **Energy**, describing the "intensity and activity" of the music track based on the track's dynamic range, perceived loudness, timbre and onset rate.
- **Valence**, describing musical positiveness of the track, in short, a track is considered happy/ cheerful if it has high valence, a track with low valence is considered sad/ depressed.

In addition, Spotify has another separate algorithm in analysing the tracks temporal structure, splitting them into segments, according to the track's rhythm, verse, chorus, beats, tatums and etc. Figure 2.2 shows the picture of temporal audio analysis for Passenger- Let Her Go:

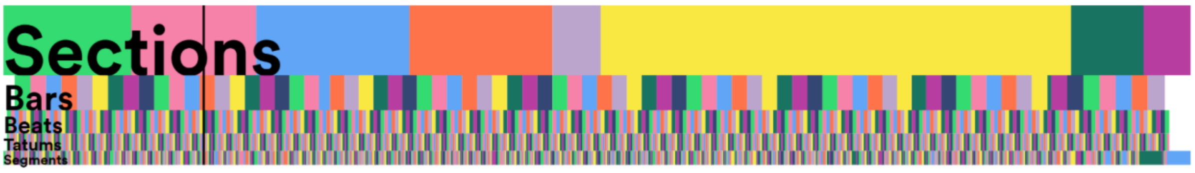


Figure 2.2- Temporal Audio Analysis for Passenger- Let Her Go (Visualization by Spotify Audio Analysis)

In practice, this means that Spotify audio analysis will be able to characterise the recordings that are submitted to the site in considerable detail. The track may be defined by the system's final output along the lines of "this song follows a V-C-V-C-B-V-C structure, builds up in energy towards the bridge and features an aggressive, dissonant guitar solo that resolves into a more melancholic and calm outro".

iii. Analysing text using Natural Language Processing Models

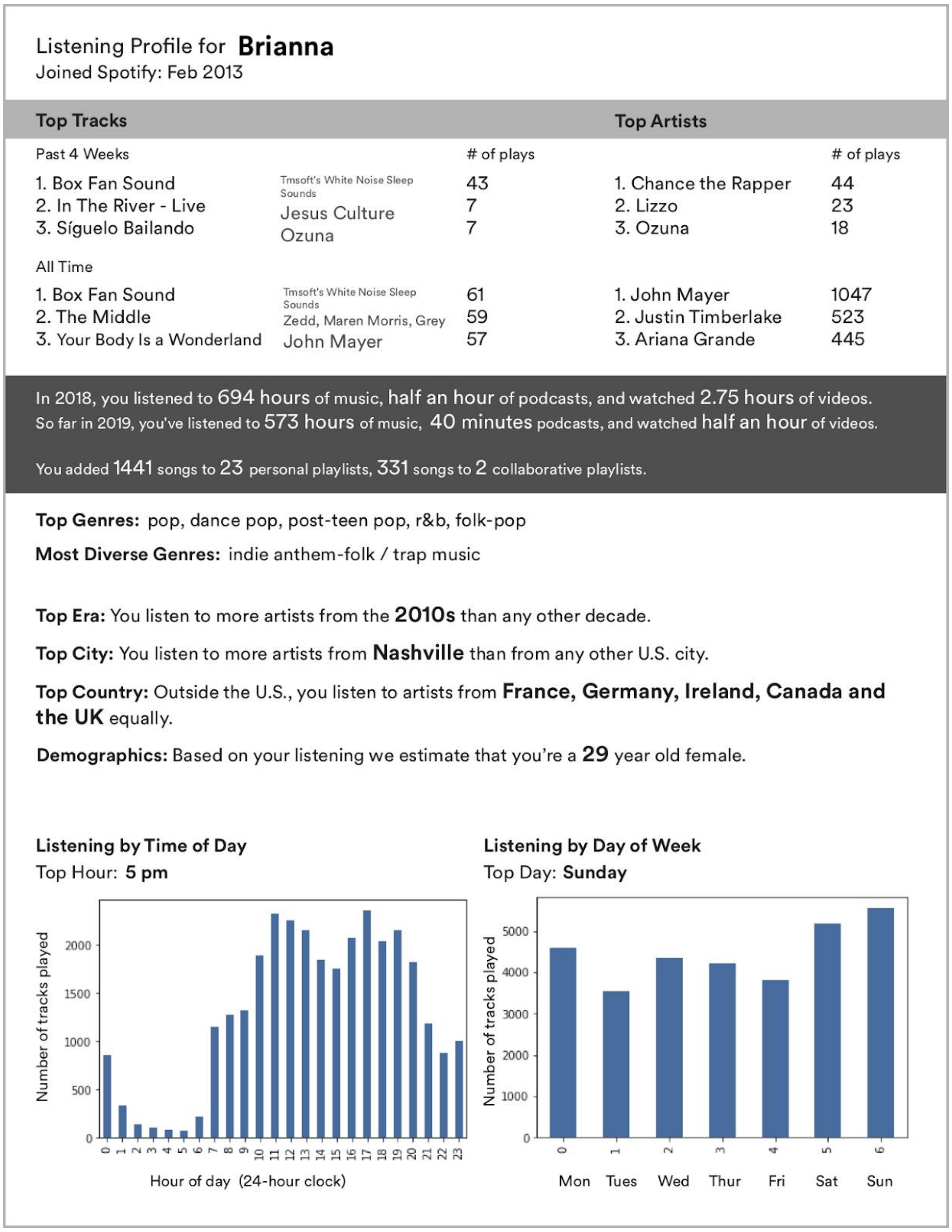
This is the last process part of content-based filtering algorithm. The Natural Language Processing models are implemented to extract data that describes tracks and artists from textual content related to music. These models have 3 primary contexts:

- **Lyric analysis**. To create a general theme and analyse the meaning of song’s lyrics, while also looking for keywords like locations, brands or people mentioned in the song which could be useful later on during recommendation.
- **Web-crawled data**. To learn more about how people describe music on music blogs and online media outlets. Analysing terms and adjectives that are related to song’s title or the name of creator.
- **User-generated playlist**. To have a better understanding on the song’s mood, style and genre by browsing through user-generated playlist on Spotify.

Next is the Collaborative Filtering method that are used by Spotify. Throughout the years, Spotify's recommender system has become associated with collaborative filtering. Collaborative Filtering algorithm is an algorithm which compares the listening history of users, that is: if user *A* has enjoyed listening to songs *X*, *Y* and *Z*, while another user *B* has enjoyed listening to songs *X* and *Y* (but haven't heard *Z* yet), the system will recommend song *Z* to user *B*. Spotify will identify whether two songs are similar or two users are similar by keeping a massive user-item interaction matrix up to date. In the latest version of collaborative filtering in Spotify, it emphasises on the track's organizational similarity instead of consumption-based filtering, i.e. “two songs are similar if a user puts them on the same playlist”. Collaborative filtering algorithm get access to a higher level of details and capture user signals that are well-defined.

The combination of content-based and collaborative allows Spotify to create a good track representation. Now, the machine must match the tracks with another dataset describing users in order to transform this data into meaningful suggestions. Spotify recommender engine will logs all the users’ listening activities, splitting them into separate context-rich listening sessions. These are feedbacks from users which is important in generating user taste profile. For example, if the user interacts with Spotify's "What's New" tab, the main purpose of the listening session is to have a brief explore at new music that has been introduced recently. In this situation, high skip rates are to be expected, as users will skimming through the feed, looking to store some of the songs recommended for future — that is to say track skipping should not be interpreted as a definite negative signal. Given another example, if user skips a track while listening to a "Deep Focus" playlist designed to be listened for long hours in the background, it is clear that the skip is a far more powerful sign of consumer dissatisfaction.

User taste profile is further subdivided based on consumption context, Spotify will then creating a context-aware user profile. Figure 2.3 shows the example of context-aware user profile by Spotify Research:


Listening Profile for **Brianna**
Joined Spotify: Feb 2013

Top Tracks | **Top Artists**

Top Tracks		# of plays	Top Artists	# of plays
Past 4 Weeks				
1. Box Fan Sound	Tmsoft's White Noise Sleep Sounds	43	1. Chance the Rapper	44
2. In The River - Live	Jesus Culture	7	2. Lizzo	23
3. Síguelo Bailando	Ozuna	7	3. Ozuna	18
All Time				
1. Box Fan Sound	Tmsoft's White Noise Sleep Sounds	61	1. John Mayer	1047
2. The Middle	Zedd, Maren Morris, Grey	59	2. Justin Timberlake	523
3. Your Body Is a Wonderland	John Mayer	57	3. Ariana Grande	445

In 2018, you listened to 694 hours of music, half an hour of podcasts, and watched 2.75 hours of videos.
So far in 2019, you've listened to 573 hours of music, 40 minutes podcasts, and watched half an hour of videos.

You added 1441 songs to 23 personal playlists, 331 songs to 2 collaborative playlists.

Top Genres: pop, dance pop, post-teen pop, r&b, folk-pop

Most Diverse Genres: indie anthem-folk / trap music

Top Era: You listen to more artists from the **2010s** than any other decade.

Top City: You listen to more artists from **Nashville** than from any other U.S. city.

Top Country: Outside the U.S., you listen to artists from **France, Germany, Ireland, Canada and the UK** equally.

Demographics: Based on your listening we estimate that you're a **29** year old female.

Listening by Time of Day
Top Hour: **5 pm**

Listening by Day of Week
Top Day: **Sunday**

Figure 2.3- Context-aware listening profile of Spotify user

After the production of track representations and user representations, Spotify recommendation system is ready to serve relevant music to users based on features such as user-entity affinity, item similarity and item clustering.

2.2.2 Deezer

Deezer is also a music streaming platform which can be accessed on websites with https://www.deezer.com/en/. The algorithm that are used by Deezer is called Flow. Flow is an algorithm that learns on users' music listening habits, while taking details such as time of day and location into account and then recommends users' favourite tunes without the needs to move a single muscle. Flow is being called as "lean back experience". While other music streaming system like Spotify and Apple Music, users must scroll through playlists and individual songs to find what they truly want to hear, as for Deezer's Flow algorithm, it hopes that it is capable of figuring it out on its own.

Some creative ways are implemented to make Flow successfully achieving its purpose. Flow refers to what users have previously listened to and by using metadata tags, it recommends similar artists and music based on what users have liked in the past. On top of this basic approach, a more complex idea is implemented. Flow also looks on what music is being listened to by other people with similar musical tastes, and recommend some of them to users to check out. For example, if you have been listening to Avicii songs, based on the data collected, Flow know that people who liked Avicii also like Martin Garrix, so you will expect that Flow will recommends you songs by Martin Garrix. Figure 2.4 and Figure 2.5 showing Deezer recommending song based on user's favourite artists:

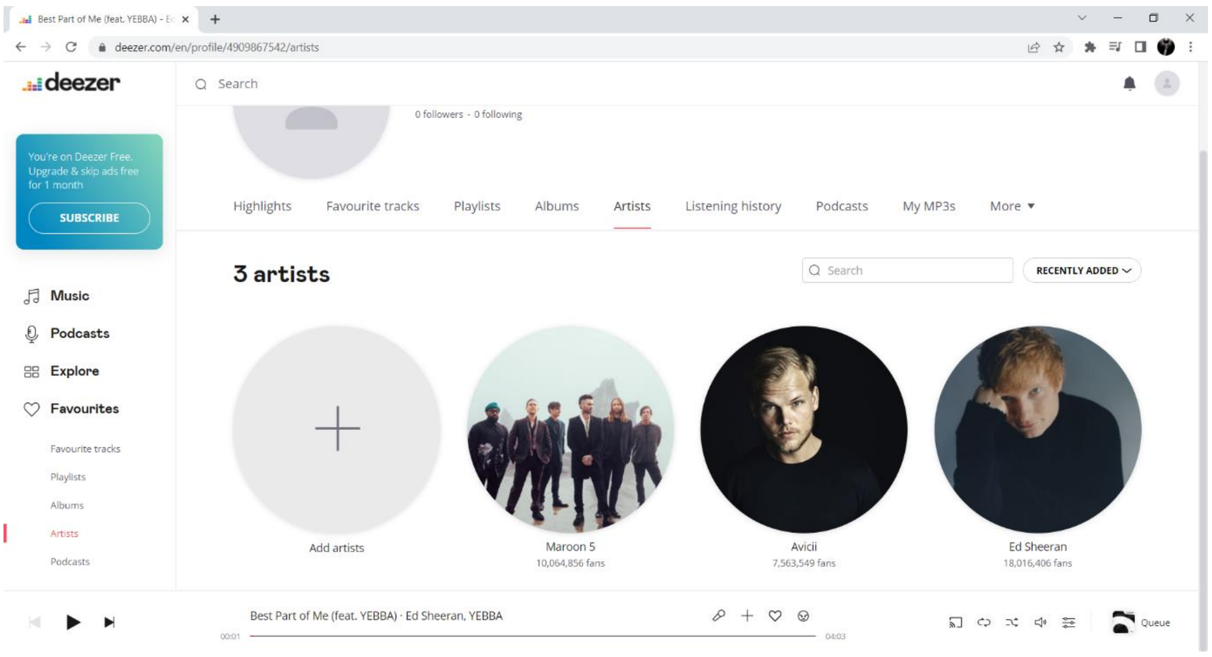


Figure 2.4 – Interface showing user's favourite artists

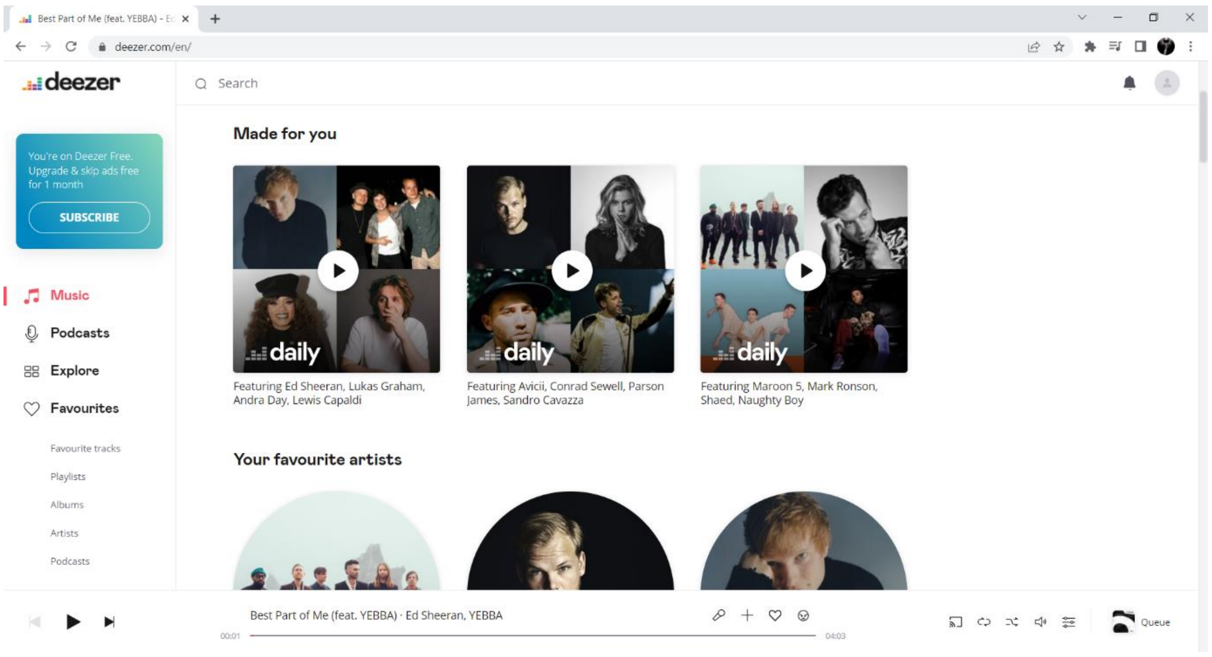

Best Part of Me (feat. YEBBA) - E
deezer.com/en/
deezer
Search
You're on Deezer Free. Upgrade & skip ads free for 1 month
SUBSCRIBE
Music
Podcasts
Explore
Favourites
Favourite tracks
Playlists
Albums
Artists
Podcasts
Made for you
daily
Featuring Ed Sheeran, Lukas Graham, Andra Day, Lewis Capaldi
daily
Featuring Avicii, Conrad Sewell, Parson James, Sandro Cavazza
daily
Featuring Maroon 5, Mark Ronson, Shaed, Naughty Boy
Your favourite artists
Best Part of Me (feat. YEBBA) · Ed Sheeran, YEBBA
00:01
04:03
Queue

Figure 2.5– Playlists recommended by Deezer based on user's favourite artists

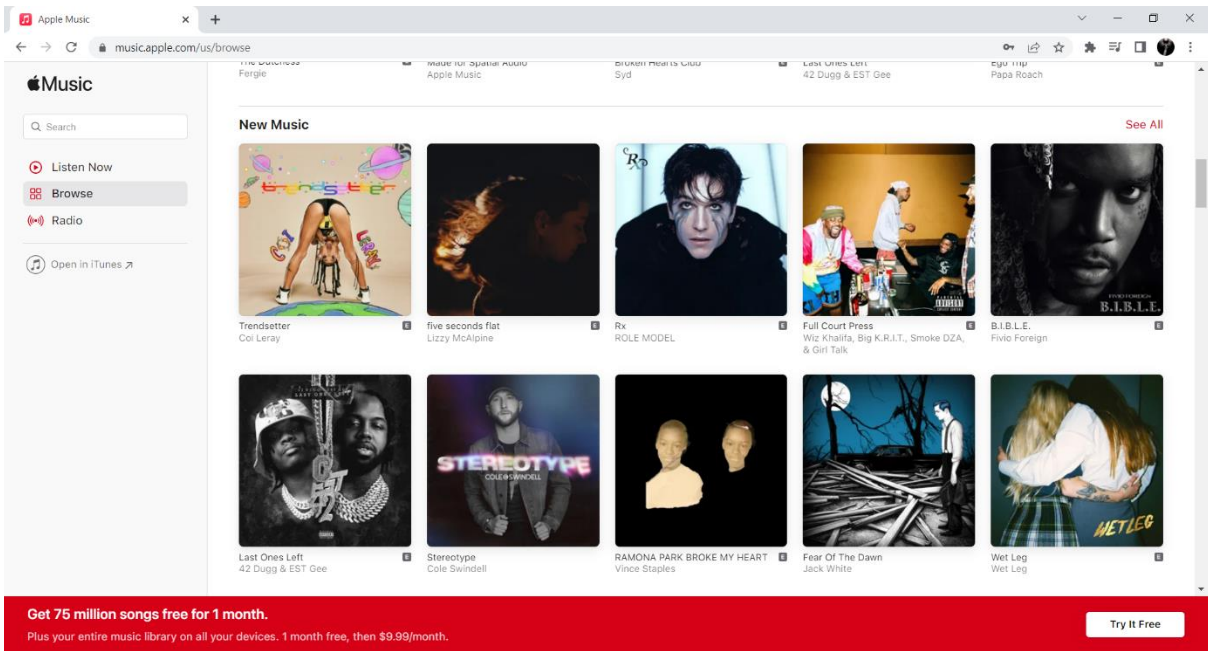
Another feature of Flow is that it considers also on users' location, users' scheduled item on calendar, and current users' velocity. And yes, Deezer's music algorithm seriously uses your phone's accelerometer to pick out users' next song. As examples, if Deezer detected that you were riding the train to work in the morning, it may send you some more uplifting music to get you in the right state of mind. On the other hand, if Flow detects that user were at home vegetating on the couch, it will likely play something mellower – like jazz or classical. Last but not least, if Flow determines that user is at the gym based on the data from user's accelerometer, the next song recommended might be a high-octane rap or rock anthem to motivate the user to push further.

2.2.3 Apple Music

Apple Music can be assessed on browser using the url of https://music.apple.com/us/browse. Apple music's recommendation system is basically a supercharged Genius recommender system which iTunes has been using for years now. Apple Music has a section called For You where system provides a personalized list of music based on users' listening habit. Apple Music's recommendation engine takes the following features into account:

i. Hearts. Users interaction with songs played of hearting them helps the system in identifying users' taste of music.
ii. Plays. Apple Music's engine plays attention closely on what users actually listening to help surfacing similar content. The engine only take into account if and only if a song is fully-played, and discards skips of music.
iii. Users' Library. Songs that are downloaded from iTunes Store or imported into iTunes from other sources will be analysed. Users' personal library data, together with music playlist that are manually added or created in Apple Music will influence music that recommended by the system.
iv. Genres and bands. It is under the start-up procedure of Apple Music. After selecting which songs and genres, system will start recommend them to users.

The recommendation of Apple Music is in its own unique way. The algorithm uses song by song, artist by artist, playlist by playlist and genre by genre approaches. If users add a playlist to the library, Apple Music will recommends other playlist with similar energy, which also contains some songs from users' library. Similarly, if users add a specific genre of songs frequently into library, the system will recommend other song playlist of that genre, also containing some of the original songs. Another playlist of other users with overlapping music of yours will also be recommended. As the system will expect the other user has the same taste in music as yours, thus the music he likes you might also suitable to listen. Figure 2.6 shows the home page of Apple Music with recommended playlists for user.



Get 75 million songs free for 1 month.

Plus your entire music library on all your devices. 1 month free, then $9.99/month.

Figure 2.6 – Home page of Apple Music

2.3 Comparative Analysis of Existing System

Although these system serve the same purpose of recommending musical pieces to users, there are some differences between the systems. In this section, comparative analysis will be made to compare the 3 existing systems, Spotify, Deezer and Apple Music. Table 2.1 shows the differences between each system in terms of algorithm used, interoperability, feedback collection, etc.:

	Spotify	Deezer	Apple Music
Algorithm Used	Content-based Filtering, Collaborative Filtering	Content-based Filtering	Content-based Filtering, Collaborative Filtering
Interoperability	Mobile app, Desktop app, Web player	Mobile app, Desktop app, Web player	Mobile app, Web player

Feedback Collection	Implicit rating	Implicit rating	Implicit rating, Explicit rating
Smartphone OS support	Android, IOS		

Table 2.1 - Comparison between existing systems

There are pros and cons in each existing system, which makes them unique from other existing systems. In Spotify, the most significant advantage is that Spotify is easy to use. The interface of Spotify is clear and simple, first-time users usually will not have difficulties in searching for songs. Another advantages of Spotify is the high compatibility. Aside from common devices like desktop, mobile and tablet, Spotify is also available in some cars, TVs, game consoles and smart watches. Music streaming of over 40 million tracks is made easy for users at anywhere and anytime. With all these advantages, the cons of Spotify are also easily spotted. The first disadvantages of Spotify is lack of lyrical features. Most of the songs in Spotify do not come with lyrics, especially songs which not in English. Another disadvantage of Spotify is that free users will have audio advertisements in between songs, only paid users will have full access in music streaming without ads.

For Deezer, the first advantage is that the app is available for over 180 countries, over 90 percent of countries in the world will have access to the Deezer app, giving a lot of users who would otherwise be locked out because of their geographical location a chance to try it. Another advantage of Deezer is that Deezer has a massive music library. Deezer has over 73 million music tracks in its library. In addition to music, Deezer also offers features like radio, podcasts and audiobooks. The disadvantages of Deezer includes bad user experience for free users, and poor customer support service. For free tier users, the songs played come with ads that keep interrupting the playlists. In free version, the sound quality is relatively poor, as compared to the premium version. The customer support base of Deezer is hard to reach. On their website's Help page, Deezer does not provide a contact list that includes phone numbers or even an email address. It causes the user feedbacks process to be delayed.

The first advantage of Apple Music is the availability on Android smartphone. Being an Apple's application, Android phone users can also have access in the app. Users only need to log in the app using their Apple ID and their library of songs will now available on Android phones. Another unique advantage of Apple Music is that Apple Music accepts music uploads from artists. For some small musicians that are looking to get noticed, Apple will accept submissions for inclusion on Apple Music with terms and conditions. For disadvantages, Apple Music does not have a free, ad-supported option. Users are strictly need to pay before start listening to music on the app. The next disadvantages of Apple Music is that it only supported limited devices on iOS, Apple Watch, Macs, PCs, and Android devices unlike Spotify which covers platforms like TV, PlayStaion3, 4 and etc.

2.4 Techniques of Music Recommender System

In this section, different techniques of music recommender system, which are content-based filtering, collaborative filtering and hybrid filtering will be discussed in details on how each technique recommends musical pieces to users. Comparison between each technique will also be made at the end of this section.

2.4.1 Content-based Filtering

Content-based filtering is a type of method that uses item features such as item keywords and attributes to recommend other similar items to what the users likes, based on users' previous actions and feedbacks. In the example of music recommendations, a recommender system will consider whether a song belongs to a specific genre, analyse the song on its lyrics, artists and so on before recommends it to users according to profiles created.

One of the most straightforward ways in developing a content-based filtering music recommender system is the keyword matching. The ideology behind it is to extract essential keywords from a song's description. Based on user's activity on likes and search history, the system will find other music with the same keywords, calculate the similarities between songs and based on that, suggests musical pieces to the user. In short, content-based recommendation algorithm is performing 2 steps. First step is to extract characteristics from the song descriptions to generate an object representation. The next step is to define similarity among the object representations created mimicking human understanding on item-item similarity. Figure 2.7 illustrates the recommendation of content-based filtering where the system recommends music of similar attributes with music listened by users.

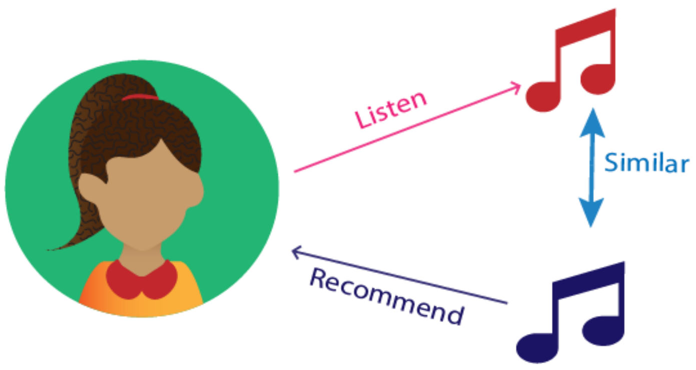


Figure 2.7 – Content-based filtering method recommends music with similar attributes

The key idea of content-based filtering is to recommends items with similar attributes. Similarity can be derived from description of items using Term Frequency-Inverse Document Frequency (TF-IDF) technique. This method is used to count the number of times each word appears in a document and weight its relevance, and generate a score for that item.

Term Frequency (TF). The Term frequency of a word in the present document refers to the number of times that it appears to the total number of words in a document. As an example, for the phrase “music” in the data “I love music because it helps me in releasing stress.” Inverse document frequency (IDF) is the metric of how important that term is over the whole database. It is defined as Total Number of Documents to the frequency occurrence of documents containing the word. The lower the number of documents containing the terms, the higher the value of IDF, indicates that the term is rare. With TF and IDF values calculated, a TF-IDF vector will be calculated. And in order to use this vector matrix for a recommendation, similarity of one data to another need to be calculated. Different metrics can be used to compute the similarity between items, such as Cosine Similarity, Euclidean Distance and K-Mean Clustering.

2.4.1.1 Euclidean Distance

The Euclidean distance is calculated between the centroid of each neighbouring cell and the source cell. Each distance tool calculates the actual Euclidean distance. The Euclidean algorithm conceptually operates as follows: for each cell, the distance to each source cell is calculated by finding the hypotenuse of the triangle, with x-max and y-max serving as the other two legs. Instead of using the cell distance, this technique derives the actual Euclidean distance. Figure 2.8 illustrate how to calculate Euclidean Distance (d) between two points P and Q.

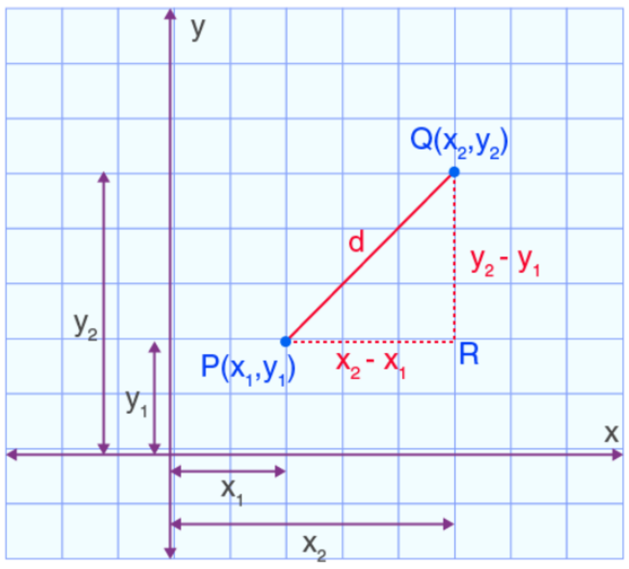


Figure 2.8 – Euclidean Distance between point P and point Q

2.4.1.2 Cosine Similarity

Regardless of the size of the documents, cosine similarity is a metric used to determine how similar the data is. It calculates the cosine of the angle created by two vectors that are projected onto a multidimensional space. The cosine similarity is advantageous since it increases the chances that the two similar documents will be oriented closer together, even if they are separated by a large Euclidean distance because of the size of the documents. The cosine similarity increases with decreasing angle. Figure 2.8 illustrate how cosine distance is calculated. Content-based filtering system will then recommend items based on the distance calculated.

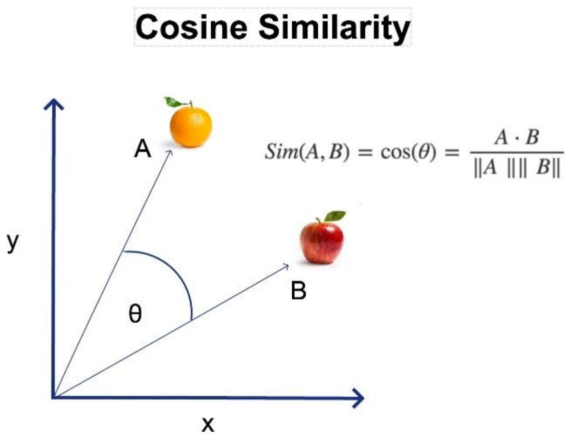


Figure 2.9 – Cosine Similarity of two items is calculated

2.4.1.3 K-Mean Clustering

Without an output variable to guide the learning process, algorithms analyse data to find patterns for unsupervised learning. The programme finds similarities among the data points and clusters them since the data does not have labels. Each cluster in K-means is represented by its centroid, or centre, which is the arithmetic mean of the data points assigned to the cluster. A centroid is a data point that represents the centre of the cluster (the mean), and it might not necessarily be a member of the dataset. This way, the algorithm works through an iterative process until each data point is closer to its own cluster’s centroid than to other clusters’ centroids, minimizing intra-cluster distance at each step.

2.4.2 Collaborative Filtering

Collaborative method is the most common and widely-used method for generating recommendations in music streaming services. This algorithm relying on set of songs which users preferred in the past to predict which song that user would like to listen to. To know preferences from users, users’ rating is collected using two ways. First way is the explicit rating, where a system ask users directly for rating on songs recommended. The second way is the implicit rating, where system takes the duration and number of times of the songs are played, to know whether users like the songs or not. These ratings are then translate to binary to generate interaction metrics.

Now that interaction metrics have been created, here comes the part where the system starts to recommend songs to a particular user. In collaborative filtering, there are two approaches, one is user-based approach and another one is item-based approach. For user-based approach, system will finds users with similar interests and behaviour, considering which songs are the similar users frequently listened to, and make recommendations. For item-based approach, songs that are listened in the past are taken into account and recommendation are made based on that. The main idea of collaborative filtering is to recommend new song based on the closeness in the behaviour of similar users. For example, suppose that there is a user *A* who likes music *X*. If there are a lot of users who like music *X* also like music *Y*. Then, music *Y* will be recommend to user *A*. Thus, collaborative filtering primarily focus on relationships between items and users, items' similarity among users is determined by users who rated them.

In collaborative filtering, some machine learning algorithms can also be implemented, such as k-nearest-neighbour, clustering and matrix factorization. K-Nearest Neighbours (kNN) is regarded as the standard method in both user-based and item-based collaborative filtering (Euge Inzaugarat, 2020). The kNN algorithm is a supervised non-parametric lazy learning approach that may be used for both classification and regression. This technique is based on feature similarity, it makes assumptions that items that have similarities are close to each other.

Figure 2.9 shows original data points in the database that are classified into two groups, *classA* (yellow) and *classB* (Green) and every points have different attributes value.

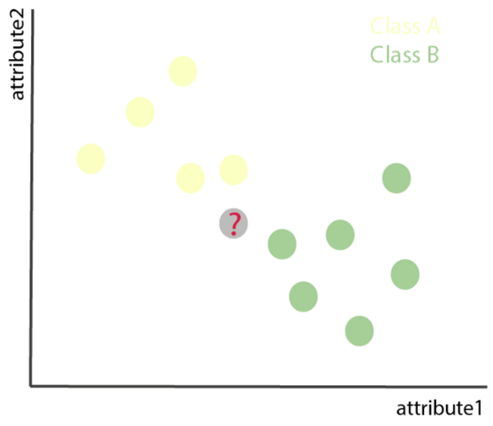


Figure 2.10 – Data points that are classified into two class

If a new item is added, the algorithm will first look at its k nearest neighbours to classify where that new point belongs by choosing the most popular class among these k-points. Below are the illustration of k-nearest neighbour algorithm in classifying classes of new point data. The k-value is important and need to be carefully defined. Figure 2.10 illustrates the situation of when *k=5*, the new point data is classify as class *B*. This is because when *k=5*, 5 nearest neighbours data points are take into accounts and the 5 nearest neighbours are mostly consists of data points of class *B*.

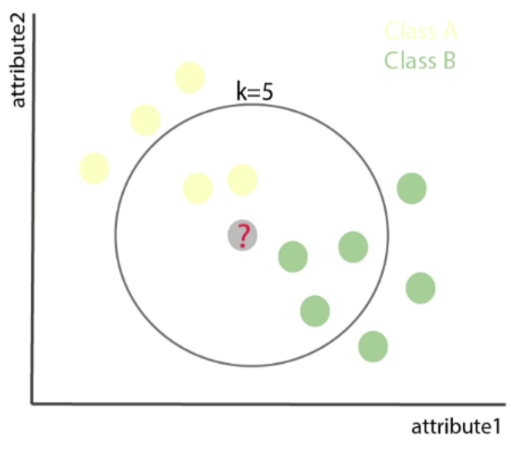


Figure 2.11 – New data point added is classified as class B

In figure 2.11, it illustrates another case where *k=3*, the new data point is classified as class *A*, as majority of the 3 nearest neighbours are belong to class *A*. These 2 scenario shows the importance of *k*-value in affecting the classification of items and recommendations.

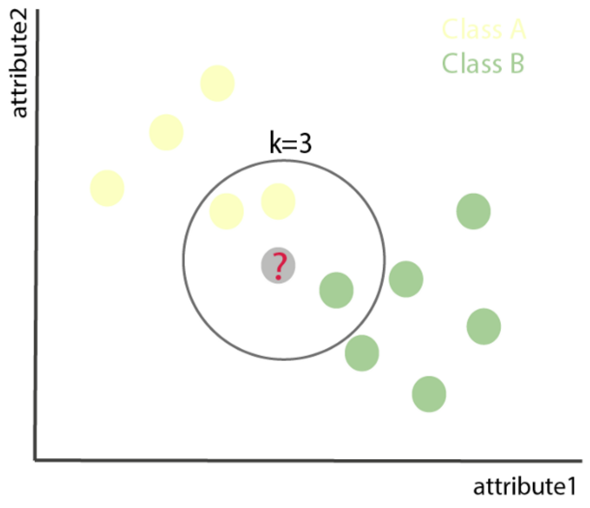


Figure 2.12 – New data point added is classified as class *A*

When kNN algorithm is used to make prediction and recommendation on songs, this algorithm will make calculations on distance between selected songs with all songs available in database. The algorithm will then make rankings based on distances between them. Lastly, it will returns the top k nearest neighbour musical pieces as recommendation to users.

2.4.3 Hybrid Filtering Recommender System

Hybrid filtering approach is the mixture of both content-based and collaborative filtering in making recommendations. In this type of recommenders system, both user-to-item relation and user-to-user relation is important. The data collection is similar to the above where it is either collect data explicitly or implicitly. The data consists of collecting similar calculations, and the results are produced using both methods. This hybrid system has a higher suggestions accuracy because it covers the absent part of other recommender system. For example, people's interest is not considered in content-based filtering but in hybrid filtering, people's interest is considered. When two approaches works together, it explores new paths to significant underlying. The hybrid system implement both methods, overcoming most of the weaknesses exists in both algorithms and improves systems' performance. Classification and cluster techniques can also be included to get a more excellent recommendations.

2.4.4 Comparison of techniques in music recommender system

In this section, comparison will be made between three techniques of music recommender system. Advantages and disadvantages of content-based filtering, collaborative filtering and hybrid filtering will be discussed.

The first advantage of content-based filtering approach is that this model does not need any data about other users, since the recommendations made are based on items specific to one user. Once a user has searched on a few items, a content-based filtering system can begin making relevant recommendations. This makes it ideal for businesses that don’t have an enormous pool of users to sample. The next advantage of content-based filtering is recommendations made are highly relevant to users. This filtering method is highly tailored to users’ interests, including recommendations for the niche items. The only disadvantage of content-based filtering is that the model only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

The advantage of collaborative filtering is that this model can help users to discover new interests. The music recommender system may not know the user is interested in a given item, but the model might still recommend it because similar users are interested in that item. For disadvantages, collaborative filtering method have the cold start problem. The system will has difficulty of making recommendations when the users are new. This is because the operation of collaborative filtering is based on historical data of site interactions between users and items. But new users and items simply do not have enough historical data (data sparsity) to make it work. Another disadvantage of collaborative filtering is that it suffer serious scalability problems. As the number of users increases and the amount of data expands, collaborative algorithms will begin to suffer a decrease in performances simply due to the sheer increase in data volume.

The advantage of hybrid filtering is that it has a higher performance and accuracy compared to other recommender systems as this filtering technique as it combines two or more recommendation techniques to gain performance with fewer of the drawbacks of any of them. The disadvantage of hybrid filtering is the difficulty in implementation. The combination of different feature selection method caused the increased in complexity and thus makes it difficult to implement.

2.5 Summary

As summary, this chapter discussed about the existing system in the real world. We have study on how each existing system works on recommending music to users, studying methods and algorithms used by each system. Each of them has its own unique way of recommending musical pieces. We also studied on different approaches while implementing a music recommender system. The content-based filtering, collaborative filtering and also hybrid filtering, each having its own advantages and disadvantages. Each method are good to implement in their own ways.

CHAPTER 3

METHODOLOGY

3.1 Overview

This chapter focuses on the research framework and methodology of content-based filtering technique which is going to implement on a music recommender system. The research framework is conducted step by step to achieve the objectives set. This discussion includes literature review, data collection, data normalization, content-based filtering design and music recommender system implementation.

3.2 Research Framework

This study consists of four major steps. The first phase of this study is the literature review, where different approaches of building a music recommender system is discussed and review again. The next phase is the dataset collection for the system. The third phase is the research and design of content-based filtering music recommender system. The last phase is the development of actual music recommender system using suitable algorithm. Figure 3.1 shows the overall phases in the research methodology.

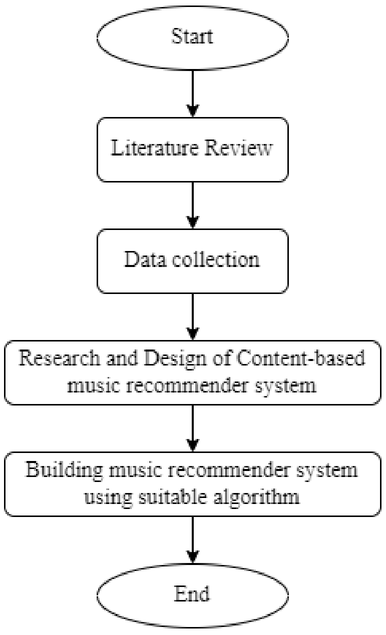

Start
Literature Review
Data collection
Research and Design of Content-based music recommender system
Building music recommender system using suitable algorithm
End

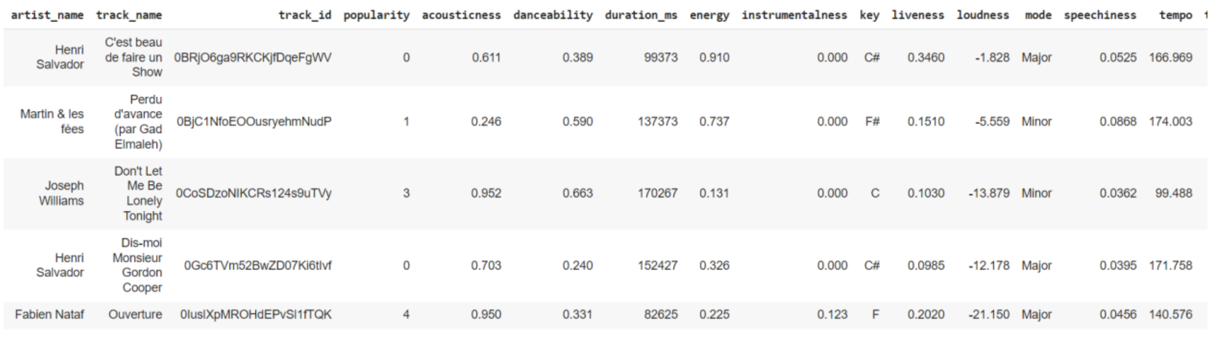
Figure 3.1 – Overview of phases in the research methodology

3.2.1 Literature Review

There are different approaches when a music recommender system is intended to be implemented. The approaches are content-based filtering, collaborative filtering and hybrid filtering. Each approaches have its own unique way of recommending musical items to users. Among the approaches mentioned, in this study, content-based filtering technique is used while making a music recommender system. Content-based filtering approach does not need any data about other users, since the recommendations made are based on items specific to one user. This approach only needs to make recommendation based on the songs in datasets, using song features to calculate similarities between the data. Comparison between three techniques of content-based filtering, collaborative filtering and hybrid filtering and comparison between three existing system of Spotify, Deezer and Apple Music have been made. The details of comparison are written in Chapter 2.

3.2.2 Data Collection

The dataset that are used in this study is the Spotify Dataset, provided by Spotify itself and can be accessed through the website with address https://www.aicrowd.com/challenges/spotify-million-playlist-dataset-challenge#dataset. Spotify uses very advanced technology to track and identify each song uploaded to its platform. The Spotify dataset provides insight into user's data about which songs they listen to, and not just the popularity of tracks, but also features of the tracks they have in their library is recorded in their database. Figure 3.2 shows some examples of data fetched from the datasets. Attributes such as artist_name, track_name, and other song features such as acousticness, energy, liveness, loudness, etc.



artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo
Henri Salvador	C'est beau de faire un Show	0BRjO6ga9RKCKjfDqeFgWV	0	0.611	0.389	99373	0.910	0.000	C#	0.3460	-1.828	Major	0.0525	166.969
Martin & les fées	Perdu d'avance (par Gad Elmaleh)	0BjC1NfoEOOusryehmNudP	1	0.246	0.590	137373	0.737	0.000	F#	0.1510	-5.559	Minor	0.0868	174.003
Joseph Williams	Don't Let Me Be Lonely Tonight	0CoSDzoNIKCRs124s9uTVy	3	0.952	0.663	170267	0.131	0.000	C	0.1030	-13.879	Minor	0.0362	99.488
Henri Salvador	Dis-moi Monsieur Gordon Cooper	0Gc6TVm52BwZD07Ki6tlvf	0	0.703	0.240	152427	0.326	0.000	C#	0.0985	-12.178	Major	0.0395	171.758
Fabien Nataf	Ouverture	0IusIXpMROHdEPvSI1fTQK	4	0.950	0.331	82625	0.225	0.123	F	0.2020	-21.150	Major	0.0456	140.576

Figure 3.2 – Examples of music data with its attributes

3.2.3 Content-based Filtering Technique Design

Content-based filtering is a type of method that uses item features such as item keywords and attributes to recommend other similar items to what the users likes, based on users' previous actions. In this study, K-mean Clustering is used as the algorithm for the system to recommends song. The K-means algorithm is chosen because it is relatively simple to implement, and it scales to large datasets. Figure 3.3 shows the overall phases in building a content-based filtering music recommender system. Next is the detail description on each phase.

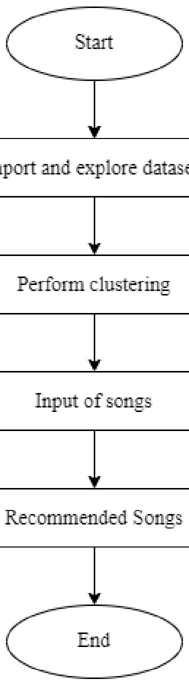


Figure 3.3 – Overview of phases in content-based filtering music recommender system

Phase 1 – Import and explore datasets. Datasets of Spotify Datasets is imported and the data is explored. The attributes of songs are analysed and are categorized to different groups. Analyzation is done each of the song features, as well as year of released and popularity. Figure 3.4 shows the results of comparing songs popularity with other song attributes such as year, loudness, energy, and etc. which can be used later as recommendations' reference. Figure 3.5 illustrate the numbers of songs that are released throughout the decade.

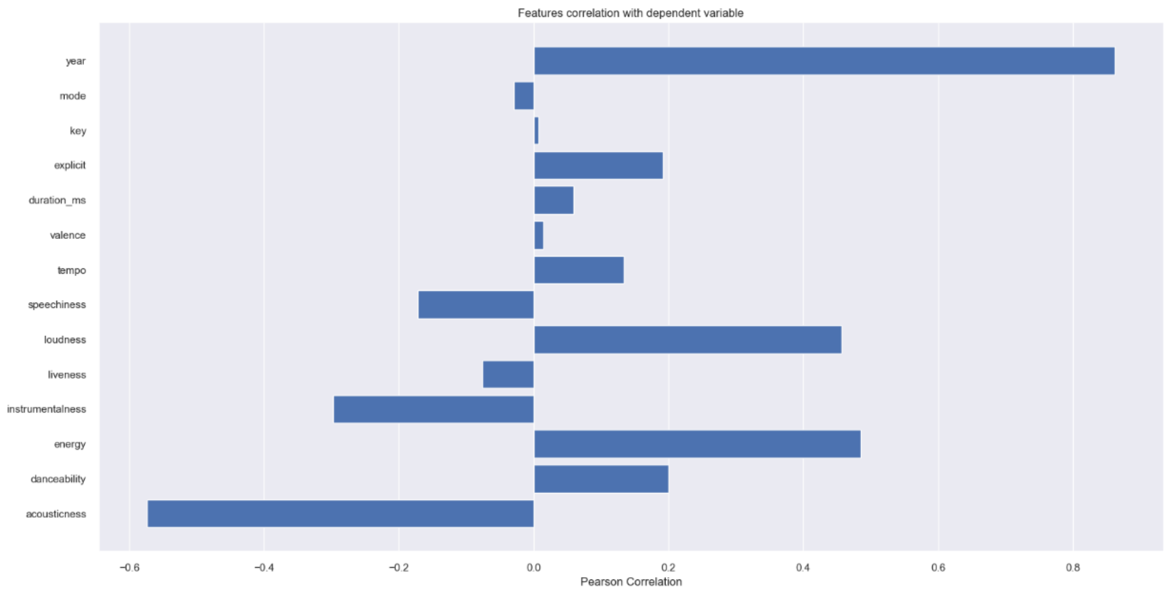


Figure 3.4 – Songs attributes are calculated against popularity

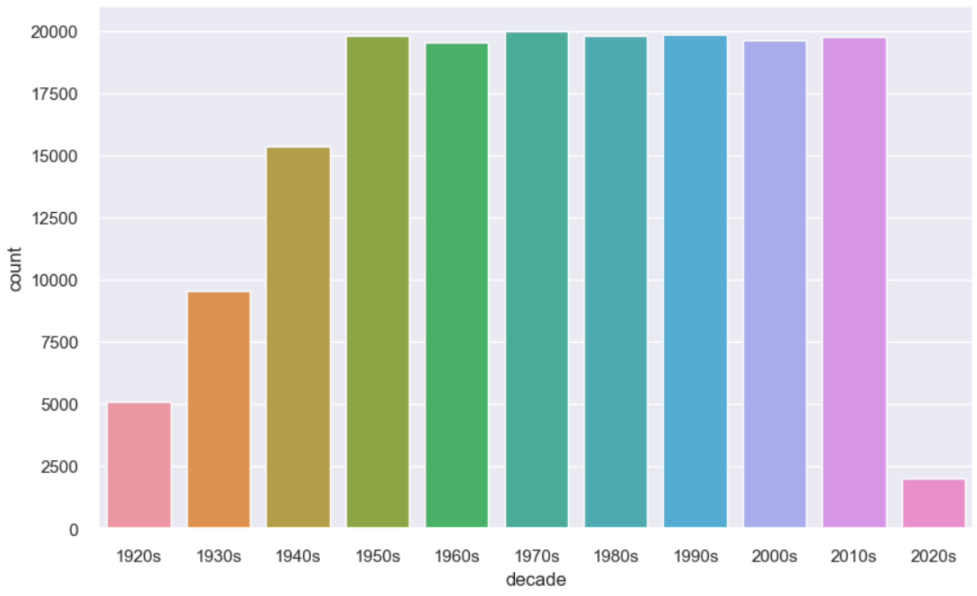


Figure 3.5 – Songs released throughout the decades

Phase 2- Perform Clustering. K-means searches for a predetermined number of clusters within an unlabelled dataset by using an iterative method to produce a final clustering based on the number of clusters defined by the user (represented by the variable K). For example, by setting “k” equal to 2, your dataset will be grouped in 2 clusters, while if you set “k” equal to 4 you will group the data in 4 clusters. K-means triggers its process with arbitrarily chosen data points as proposed centroids of the groups and iteratively recalculates new centroids in order to converge to a final clustering of the data points. Specifically, the process works as follows:

1. The algorithm randomly chooses a centroid for each cluster. For example, if we choose a “k” of 3, the algorithm randomly picks 3 centroids.
2. K-means assigns every data point in the dataset to the nearest centroid, meaning that a data point is considered to be in a particular cluster if it is closer to that cluster’s centroid than any other centroid.
3. For every cluster, the algorithm recomputes the centroid by taking the average of all points in the cluster, reducing the total intra-cluster variance in relation to the previous step. Since the centroids change, the algorithm re-assigns the points to the closest centroid.
4. The algorithm repeats the calculation of centroids and assignment of points until the sum of distances between the data points and their corresponding centroid is minimized, a maximum number of iterations is reached, or no changes in centroids value are produced.

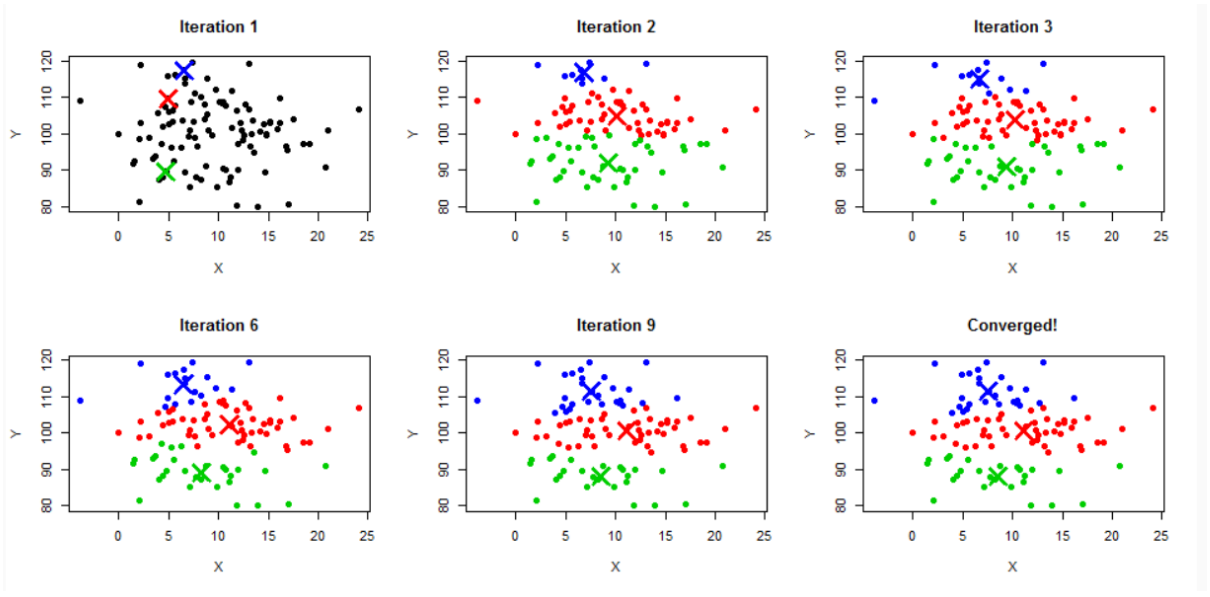


Figure 3.6 – Illustration of K-mean clustering makes clusters from scratch

Phase 3 – Input & output of songs. When user enter a song name as input, the recommender system will first search the datasets on whether the song exists. If the song exists, the system will then search for the cluster where the song is located. According to user’s input on how many songs to recommend, the system will take songs within the same cluster and is close to user's song as recommendation, display them to user as output.

3.2.4 Evaluation of Content-based filtering model

Both online and offline approaches may be used to evaluate recommender systems. The user reactions to the offered recommendations are measured via an online system. Offline approaches are the most prevalent ways for testing recommendation systems. It is critical to properly build the evaluation system such that the measured metrics accurately represent the system's efficacy from user's perspective. Accuracy metrics are the most often used measure of effectiveness. It is used to evaluate the prediction accuracy of the top-k ranking predicted by algorithm in recommender systems. The root mean squared error, RMSE, is one of the most often used ways of calculating accuracy metrics as shown in Equation 3.3:

$$RMSE = \sqrt{\frac{1}{|k|}\sum_{(u,i)\in k}(p_{ui} - r_{ui})^2} \tag{3.3}$$

where p_{ui} is predicted rate, k-quantity of testing rates. A lower RMSE number indicates that the algorithm's predictions are more accurate, results in a better performance.

Another evaluation metrics that also can be used is the Mean Absolute Error (MAE). The average absolute difference between observed and anticipated ratings is calculated using MAE. The MAE is shown in Equation 3.4.

$$MAE = \frac{1}{|\hat{R}|}\sum_{\hat{r}_{ui}\in\hat{R}} |r_{ui} - \hat{r}_{ui}| \tag{3.4}$$

where $|\widehat{R}|$ is the total number of predicted ratings, r_{ui} is is the true rating value that user u gave to item i and $\hat{r}_{ui}$ is the predicted rating value that user u gave to item i. A lower MAE score indicates that the algorithm's predictions are more accurate, therefore it can be said that it has a better performance.

3.3 Hardware and Software

The hardware and software requirements written are according to the development of music recommender system. The hardware and software specification are written in Table 3.1 and Table 3.2.

Table 3.1- Hardware Specifications

Hardware	**Specification**
ROG STRIX LAPTOP-8EQBG765	- Intel(R) Core(TM) i7-9750H CPU@2.60GHz processor - Intel(R) UHD Graphics 630 graphic card - Intel(R) Wireless-AC 9560 wireless card

Table 3.2 - Software Specifications

Software	**Specification**
Operating System	Microsoft Windows 10 Home Single Language 64-bit
Microsoft Office Word 2013	Documentation of development from chapter 1 to chapter 5
Microsoft Office Project 2013	Design of Gantt chart
Microsoft Office Power Point 2013	Preparation of presentation slide

Jupyter Notebook	Development of music recommender system
Zoom	Recording of presentation video

3.4 Conclusion

In this chapter, content-based filtering methods is reviewed and discussed. Evaluation metrics are also discussed to estimate effectiveness of recommender systems. A theoretical basis is prepared for the implementation of content-based filtering techniques for a music recommender system. Vary different parameters such as similarity measure, predicted ratings are planned to be implemented to enhance the efficacy of recommender systems.

CHAPTER 4

IMPLEMENTATION, RESULTS AND DISCUSSION

4.1 Introduction

This chapter is briefly discussed about the implementation of the research. The implementation is defined to meet the objectives that were stated to ensure that the results is relevance to the research. The datasets involved is used to perform analysis and from it, songs will be recommended according to the song's attributes. In this chapter, steps of building a music recommender system starting from analyzation of datasets, calculation of similarity to making of recommender system will be discussed.

4.2 Basic Steps on Building a Music Recommender System Using Google Colaboratory

In this section, seven basic steps of building a music recommender system using Google Colab is explained. Figure 4.1 shows the flowchart of steps involved in building this recommender system. Further in this section, each phases involved will be explained in details together with its codes. There are seven main steps which are Data Preparation, Mutual Features Calculation, Sound Features Analyzation, Perform K-Mean Clustering Algorithm, Building Recommendation Engine, Input and Output of songs. Besides K-Mean Clustering, other methods are also implemented such as cosine similarity and Euclidean distance.

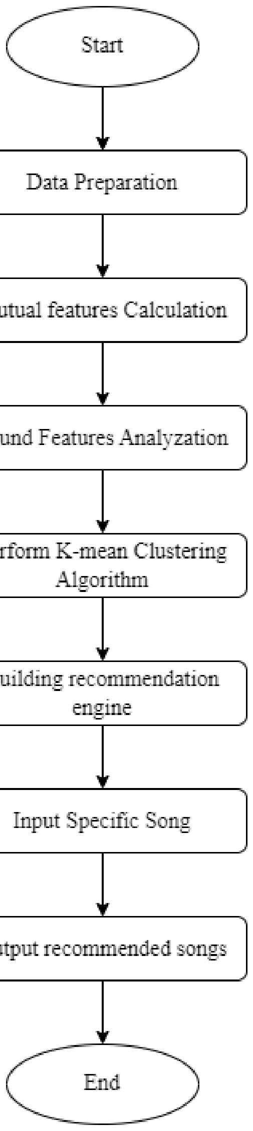

Start
Data Preparation
Mutual features Calculation
Sound Features Analyzation
Perform K-mean Clustering Algorithm
Building recommendation engine
Input Specific Song
Output recommended songs
End

Figure 4.1 – Flowchart of steps involved in building Music Recommender System

Phase 1: Data Preparation. In this study, Spotify dataset is chosen and implemented. The Spotify Million Playlist Dataset Challenge consists of a dataset and evaluation to enable research in music recommendations. It is a continuation of the RecSys Challenge 2018, which ran from January to July 2018. The dataset contains 1,000,000 playlists, including playlist titles and track titles, created by users on the Spotify platform between January 2010 and October 2020.The evaluation task is automatic playlist continuation: given a seed playlist title and/or initial set of tracks in a playlist, to predict the subsequent tracks in that playlist.

Phase 2: Mutual Features Calculation. A visualizer is used to calculate Pearson correlation coefficients and mutual information between features and the dependent variable. This visualization can be used in feature selection to identify features with high correlation or large mutual information with the dependent variable.

Phase 3: Analysis on sound features. The music data are read and analyze based on its sound features, such as acousticness, danceability, energy, instrumentalness, liveness and valence. Songs are also categorized into different category based on its popularity and released year.

Phase 4: Perform K-mean clustering algorithm. K-means algorithm is an iterative algorithm that tries to partition the dataset into K-pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

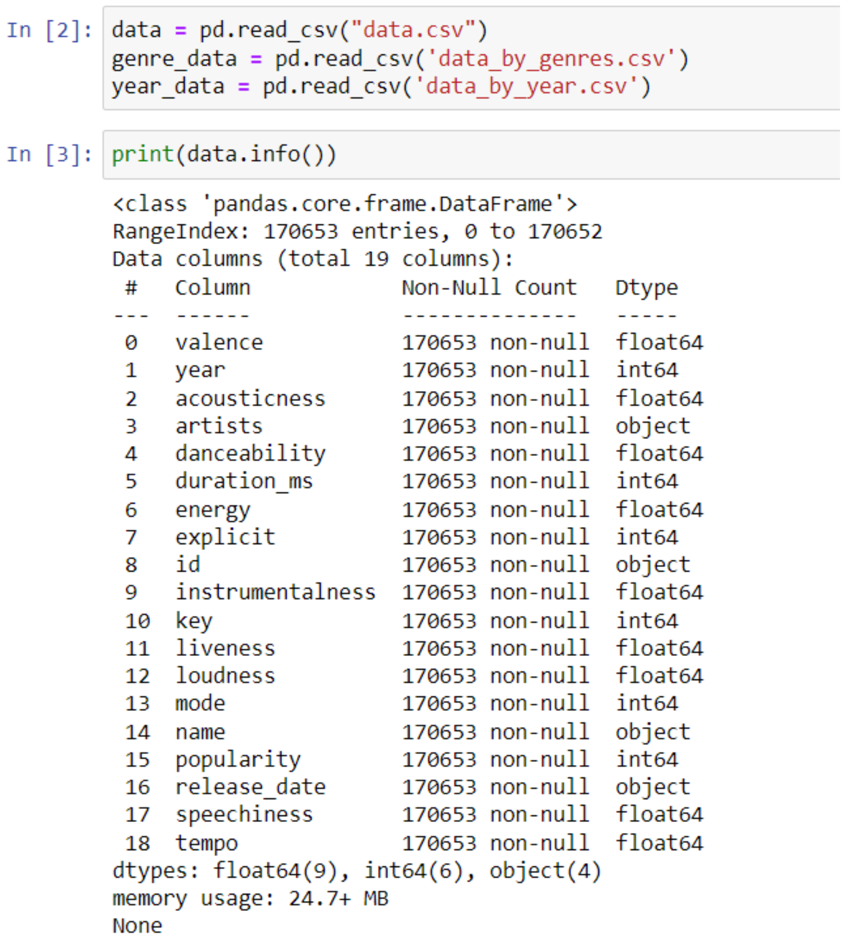
Phase 5: Building recommendation engine. The recommendation engine based on the learned data is built in this phase. The mean vector will be calculated between input songs and existing songs in datasets. After that the engine will recommend songs with similar attributes to the user.

Phase 6 & 7: Input and Output of songs. User can select to recommend based on a list of music or just select one song to recommend as output.

4.3 Implementation Process

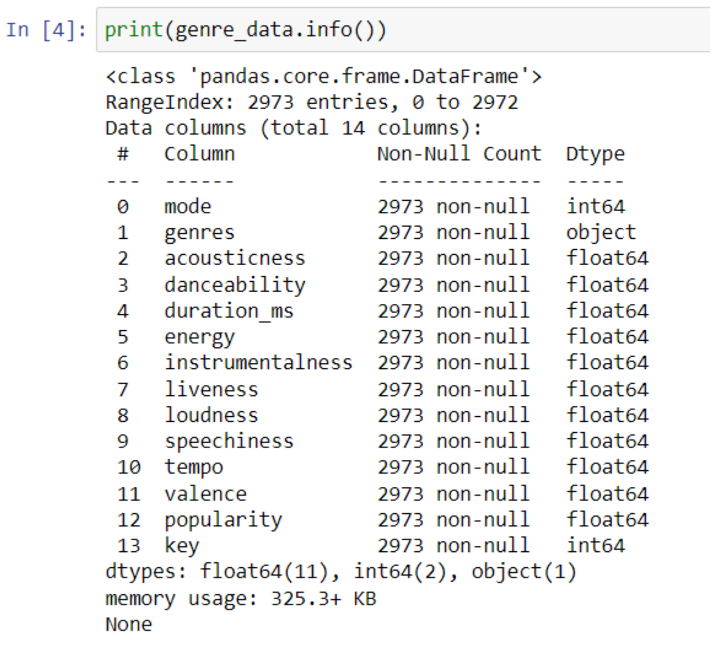
4.3.1 Main Method: K-Mean Clustering

Phase 1: Data Preparation. The Spotify datasets imported have three parts, one is the overall data, one is the data of songs' genre and lastly the data of songs by year. Figure 4.2 shows the import of data and overall data info. Figure 4.3 shows the info of datasets involving the songs' genre. Figure 4.4 shows the info of datasets of songs by years.



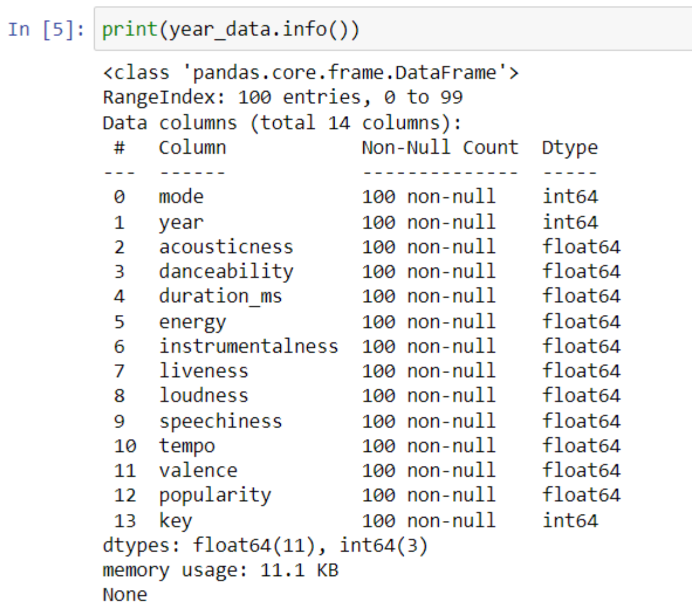
```
In [2]: data = pd.read_csv("data.csv")
        genre_data = pd.read_csv('data_by_genres.csv')
        year_data = pd.read_csv('data_by_year.csv')

In [3]: print(data.info())

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 170653 entries, 0 to 170652
        Data columns (total 19 columns):
         #   Column            Non-Null Count   Dtype
        ---  ------            --------------   -----
         0   valence           170653 non-null  float64
         1   year              170653 non-null  int64
         2   acousticness      170653 non-null  float64
         3   artists           170653 non-null  object
         4   danceability      170653 non-null  float64
         5   duration_ms       170653 non-null  int64
         6   energy            170653 non-null  float64
         7   explicit          170653 non-null  int64
         8   id                170653 non-null  object
         9   instrumentalness  170653 non-null  float64
         10  key               170653 non-null  int64
         11  liveness          170653 non-null  float64
         12  loudness          170653 non-null  float64
         13  mode              170653 non-null  int64
         14  name              170653 non-null  object
         15  popularity        170653 non-null  int64
         16  release_date      170653 non-null  object
         17  speechiness       170653 non-null  float64
         18  tempo             170653 non-null  float64
        dtypes: float64(9), int64(6), object(4)
        memory usage: 24.7+ MB
        None
```

Figure 4.2 – Import of data and overall data info



```
In [4]: print(genre_data.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2973 entries, 0 to 2972
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   mode              2973 non-null   int64  
 1   genres            2973 non-null   object 
 2   acousticness      2973 non-null   float64
 3   danceability      2973 non-null   float64
 4   duration_ms       2973 non-null   float64
 5   energy            2973 non-null   float64
 6   instrumentalness  2973 non-null   float64
 7   liveness          2973 non-null   float64
 8   loudness          2973 non-null   float64
 9   speechiness       2973 non-null   float64
 10  tempo             2973 non-null   float64
 11  valence           2973 non-null   float64
 12  popularity        2973 non-null   float64
 13  key               2973 non-null   int64  
dtypes: float64(11), int64(2), object(1)
memory usage: 325.3+ KB
None
```

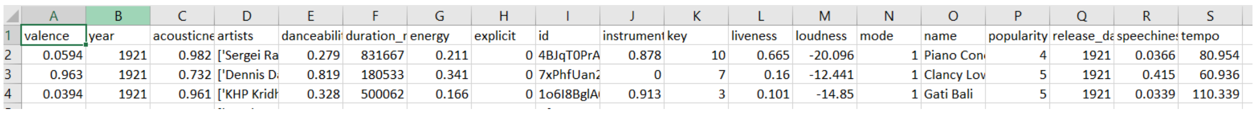
Figure 4.3 – Info of songs by genre



```
In [5]: print(year_data.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   mode              100 non-null    int64  
 1   year              100 non-null    int64  
 2   acousticness      100 non-null    float64
 3   danceability      100 non-null    float64
 4   duration_ms       100 non-null    float64
 5   energy            100 non-null    float64
 6   instrumentalness  100 non-null    float64
 7   liveness          100 non-null    float64
 8   loudness          100 non-null    float64
 9   speechiness       100 non-null    float64
 10  tempo             100 non-null    float64
 11  valence           100 non-null    float64
 12  popularity        100 non-null    float64
 13  key               100 non-null    int64  
dtypes: float64(11), int64(3)
memory usage: 11.1 KB
None
```

Figure 4.4 – Info of songs by year

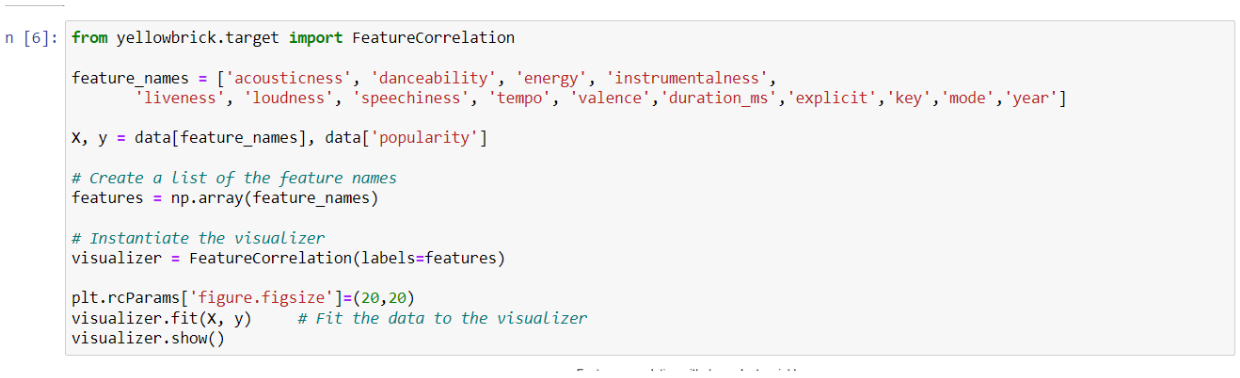
The details of datasets are being printed, showing that there are a total of 170653 songs in the *data.csv* file. Besides that, there are a total of 2973 types of songs' genre in *data_by_genres.csv* file. Lastly, there are a total of 100 entries of years in *data_by_year.csv* file. All of the data above each have their own value of songs’ features, all of the data are break down according to each category. For example, in the first datasets, each song is breakdown to have values on different features as shown in Figure 4.5. It is the same for data by genre and data by year, after categorizing song into different genre and different year, each genre and year is also given values according to different song feature.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	valence	year	acousticne	artists	danceabili	duration_r	energy	explicit	id	instrument	key	liveness	loudness	mode	name	popularity	release_da	speechine	tempo
2	0.0594	1921	0.982	['Sergei Ra	0.279	831667	0.211	0	4BJqT0PrA	0.878	10	0.665	-20.096	1	Piano Con	4	1921	0.0366	80.954
3	0.963	1921	0.732	['Dennis D	0.819	180533	0.341	0	7xPhfUan2	0	7	0.16	-12.441	1	Clancy Lov	5	1921	0.415	60.936
4	0.0394	1921	0.961	['KHP Kridh	0.328	500062	0.166	0	1o6I8BglA	0.913	3	0.101	-14.85	1	Gati Bali	5	1921	0.0339	110.339

Figure 4.5 – Values of song feature for each song in dataset

Phase 2: Mutual Feature Calculation. This visualizer calculates Pearson correlation coefficients and mutual information between features and the dependent variable. This visualization can be used in feature selection to identify features with high correlation or large mutual information with the dependent variable. In this case, variable of songs’ popularity is used to calculate correlation against other songs’ features. If the two variables tend to increase and decrease together, the correlation value is positive. If one variable increases while the other variable decreases, the correlation value is negative. Figure 4.6 shows the code to perform Pearson correlation calculation while Figure 4.7 shows the results in graph.



```
n [6]: from yellowbrick.target import FeatureCorrelation

       feature_names = ['acousticness', 'danceability', 'energy', 'instrumentalness',
              'liveness', 'loudness', 'speechiness', 'tempo', 'valence','duration_ms','explicit','key','mode','year']

       X, y = data[feature_names], data['popularity']

       # Create a list of the feature names
       features = np.array(feature_names)

       # Instantiate the visualizer
       visualizer = FeatureCorrelation(labels=features)

       plt.rcParams['figure.figsize']=(20,20)
       visualizer.fit(X, y)      # Fit the data to the visualizer
       visualizer.show()
```

Figure 4.6 – Code of calculating and displaying Pearson Correlation

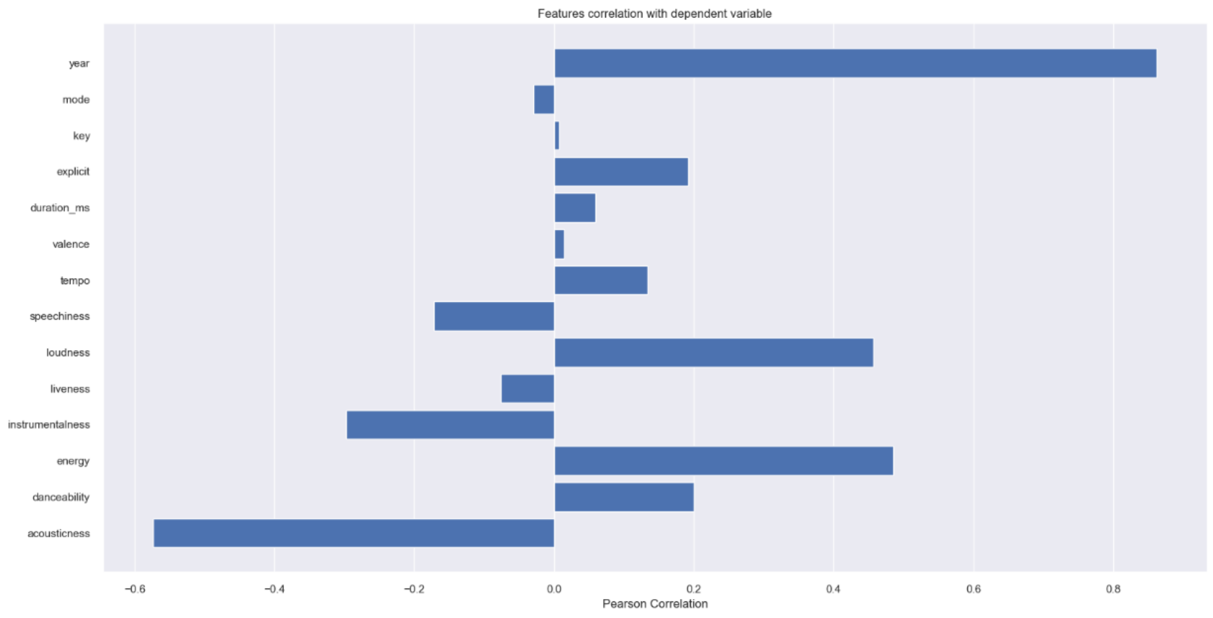
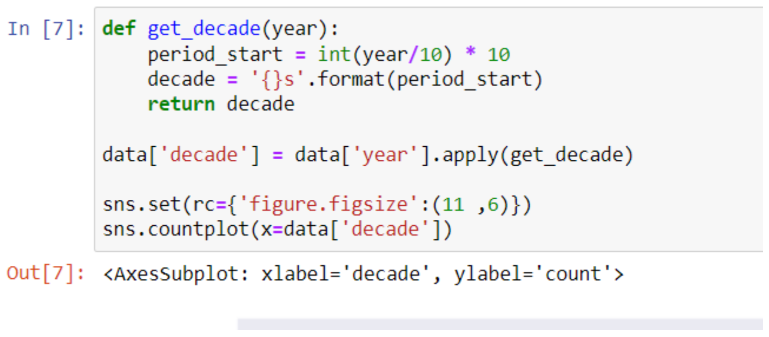


Figure 4.7 – Graph showing correlation value between popularity and other features

Phase 3: Sound Feature Analyzation. Using the data grouped by year, how the overall sound of music has changed from 1921 to 2020 can be understand. Figure 4.8 and Figure 4.9 shows the total number of songs released over the years from the 1920s to 2020s.



```
In [7]: def get_decade(year):
            period_start = int(year/10) * 10
            decade = '{}s'.format(period_start)
            return decade

        data['decade'] = data['year'].apply(get_decade)

        sns.set(rc={'figure.figsize':(11 ,6)})
        sns.countplot(x=data['decade'])

Out[7]: <AxesSubplot: xlabel='decade', ylabel='count'>
```

Figure 4.8 – Code of plotting graph showing total numbers of songs over decades

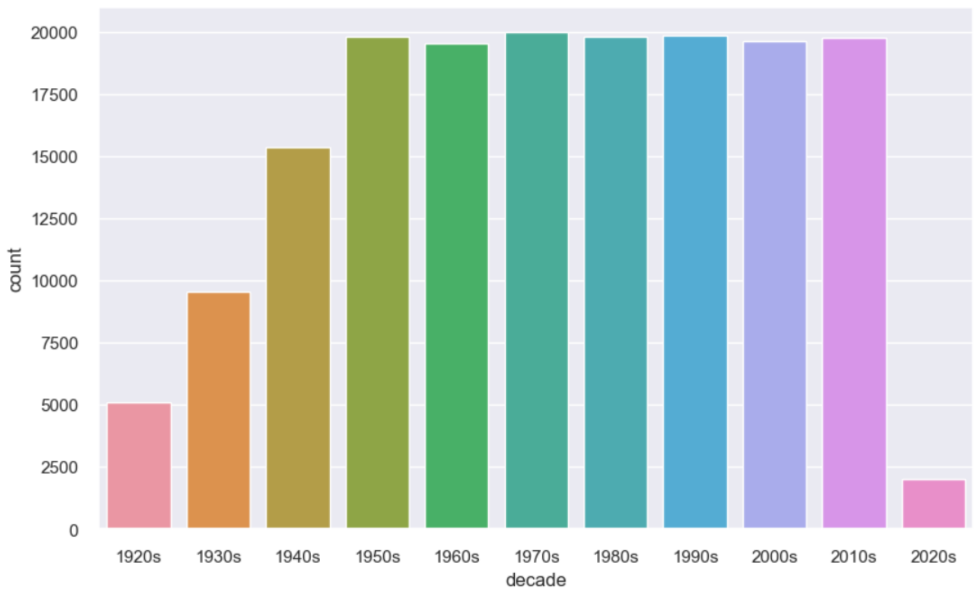
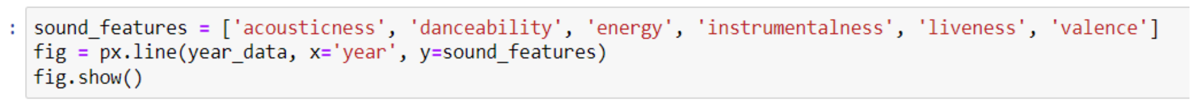


Figure 4.9 – Graph showing total numbers of songs over decades

Over the years, the feature of songs released are also changing. Figure 4.10 and Figure 4.11 shows the song features value of songs over the year.



```
sound_features = ['acousticness', 'danceability', 'energy', 'instrumentalness', 'liveness', 'valence']
fig = px.line(year_data, x='year', y=sound_features)
fig.show()
```

Figure 4.10 – Code of plotting graph showing song features over the years

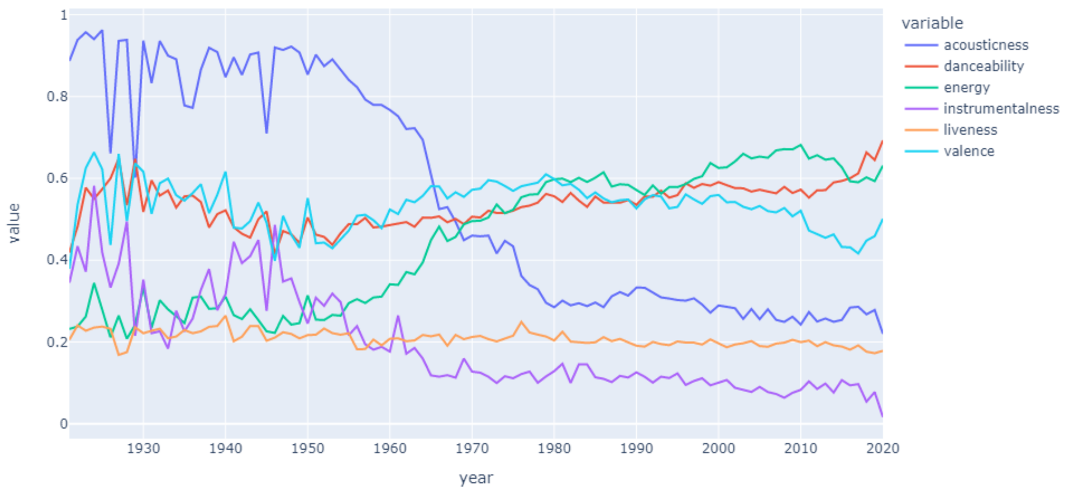
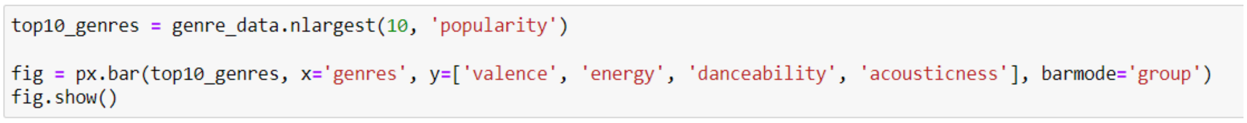


Figure 4.11 – Graph showing song features over the years

The dataset used contains the audio features for different songs along with the audio features for different genres. This information is used to compare different genres and understand their unique differences in sound. Figure 4.12 and Figure 4.13 show the top 10 genre (picked according to popularity) with their value of four most common sound feature.



```
top10_genres = genre_data.nlargest(10, 'popularity')

fig = px.bar(top10_genres, x='genres', y=['valence', 'energy', 'danceability', 'acousticness'], barmode='group')
fig.show()
```

Figure 4.12 – Code of plotting graph showing top 10 genres with its audio feature values

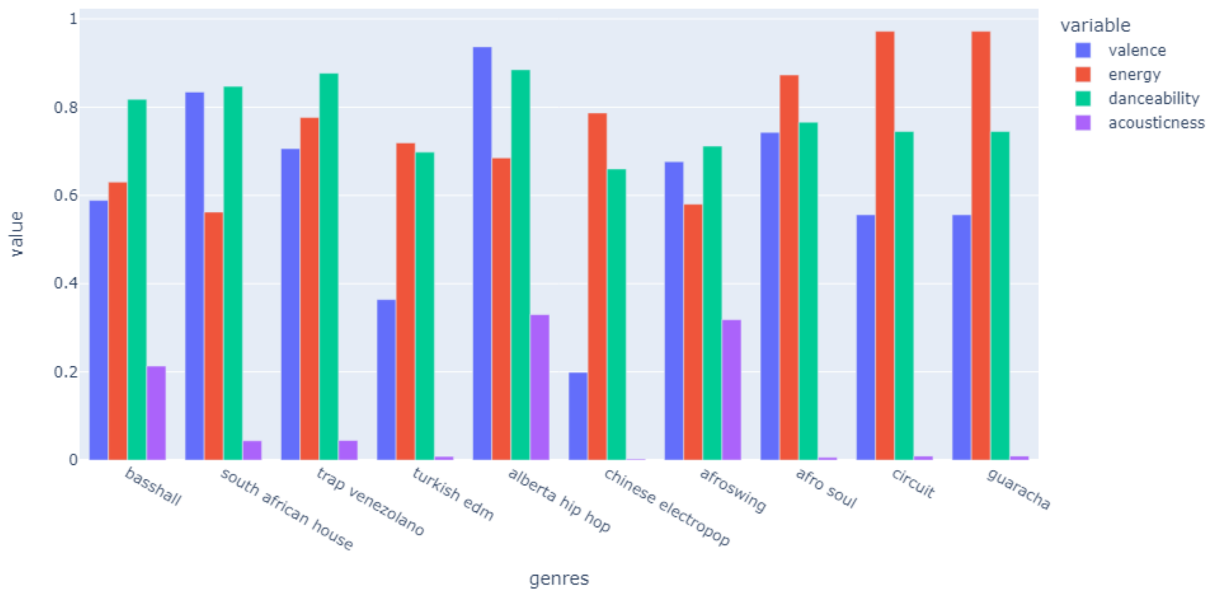
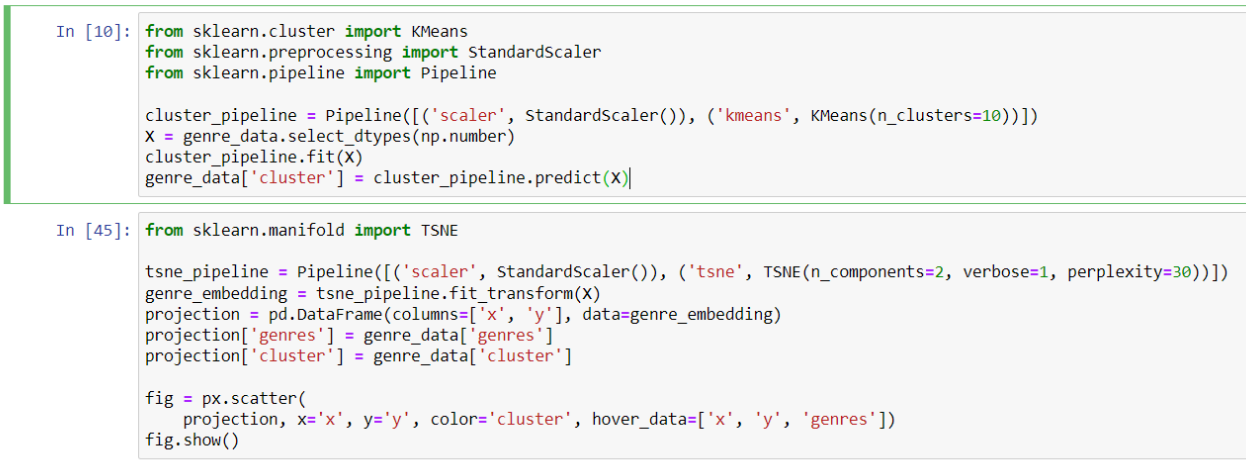


Figure 4.13 – Graph showing top 10 genres with its audio feature values

Phase 4: Perform K-mean clustering algorithm. Based on the numerical audio attributes of each genre, the dataset's genres are divided into ten clusters using the simple K-means clustering technique. T-SNE method is used to perform the clustering process. By comparing the distances between nearby or local points, T-SNE assesses how similar they are (Euclidean distance). Points that are close to each other are considered similar. t-SNE then converts this similarity distance for each pair of points into a probability for each pair of points. If two points are close to each other in the high-dimensional space they will have a high probability value and vice versa. This way the probability of picking a set of points is proportional to their similarity. "Perplexity" determines how broad or how tight of a space t-SNE captures similarities between points. If your perplexity is low (perhaps 2), t-SNE will only use two similar points and produce a plot with many scattered clusters. However, when we increase the perplexity to 10, t-SNE will consider 10 neighbor points as similar and cluster them together resulting in larger clusters of points. In this case, “Perplexity” value of 30 is used. Figure 4.14 and Figure 4.15 shows the system performing clustering categorizing each genre data in *data_by_genres.csv* to similar clusters.



```
In [10]: from sklearn.cluster import KMeans
         from sklearn.preprocessing import StandardScaler
         from sklearn.pipeline import Pipeline

         cluster_pipeline = Pipeline([('scaler', StandardScaler()), ('kmeans', KMeans(n_clusters=10))])
         X = genre_data.select_dtypes(np.number)
         cluster_pipeline.fit(X)
         genre_data['cluster'] = cluster_pipeline.predict(X)
```

```
In [45]: from sklearn.manifold import TSNE

         tsne_pipeline = Pipeline([('scaler', StandardScaler()), ('tsne', TSNE(n_components=2, verbose=1, perplexity=30))])
         genre_embedding = tsne_pipeline.fit_transform(X)
         projection = pd.DataFrame(columns=['x', 'y'], data=genre_embedding)
         projection['genres'] = genre_data['genres']
         projection['cluster'] = genre_data['cluster']

         fig = px.scatter(
             projection, x='x', y='y', color='cluster', hover_data=['x', 'y', 'genres'])
         fig.show()
```

Figure 4.14 – Code of plotting graph after performing K-Mean Clustering

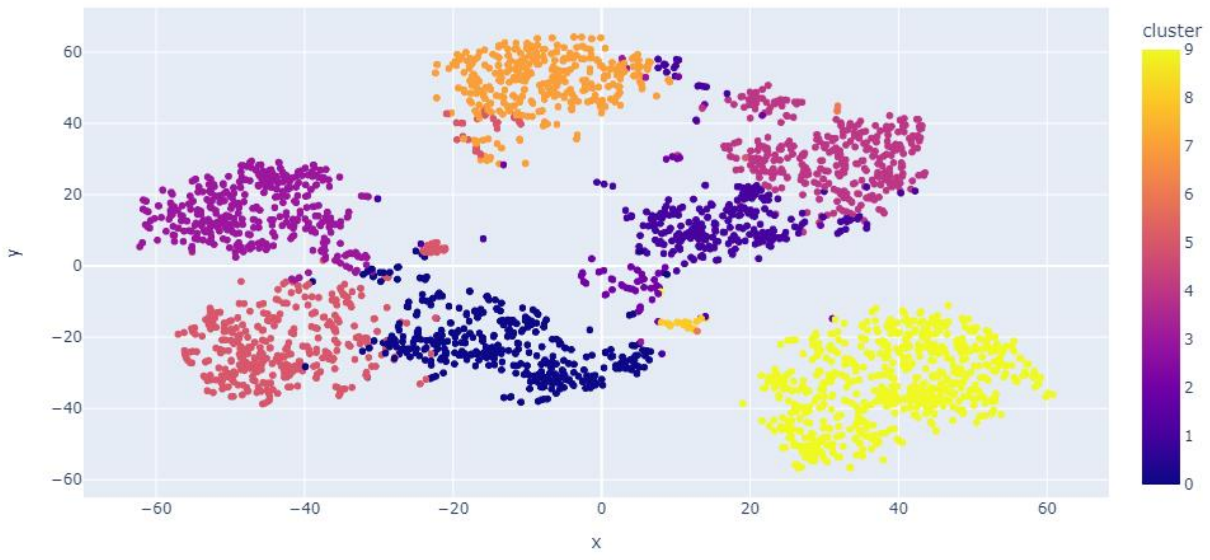
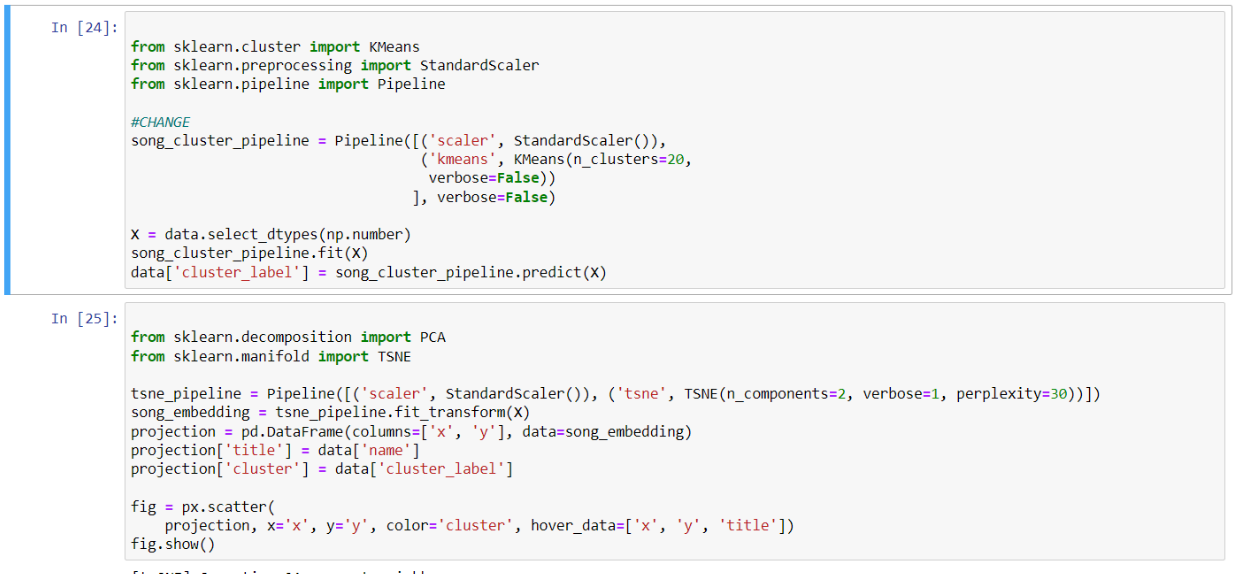


Figure 4.15 – Graph after performing K-Mean Clustering

Figure 4.16 and Figure 4.17 shows the system performing clustering categorizing each song data in *data.csv* to similar clusters.



```
In [24]:
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

#CHANGE
song_cluster_pipeline = Pipeline([('scaler', StandardScaler()),
                                  ('kmeans', KMeans(n_clusters=20,
                                   verbose=False))
                                 ], verbose=False)

X = data.select_dtypes(np.number)
song_cluster_pipeline.fit(X)
data['cluster_label'] = song_cluster_pipeline.predict(X)
```

```
In [25]:
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE

tsne_pipeline = Pipeline([('scaler', StandardScaler()), ('tsne', TSNE(n_components=2, verbose=1, perplexity=30))])
song_embedding = tsne_pipeline.fit_transform(X)
projection = pd.DataFrame(columns=['x', 'y'], data=song_embedding)
projection['title'] = data['name']
projection['cluster'] = data['cluster_label']

fig = px.scatter(
    projection, x='x', y='y', color='cluster', hover_data=['x', 'y', 'title'])
fig.show()
```

Figure 4.16 – Code of plotting graph after performing K-Mean Clustering

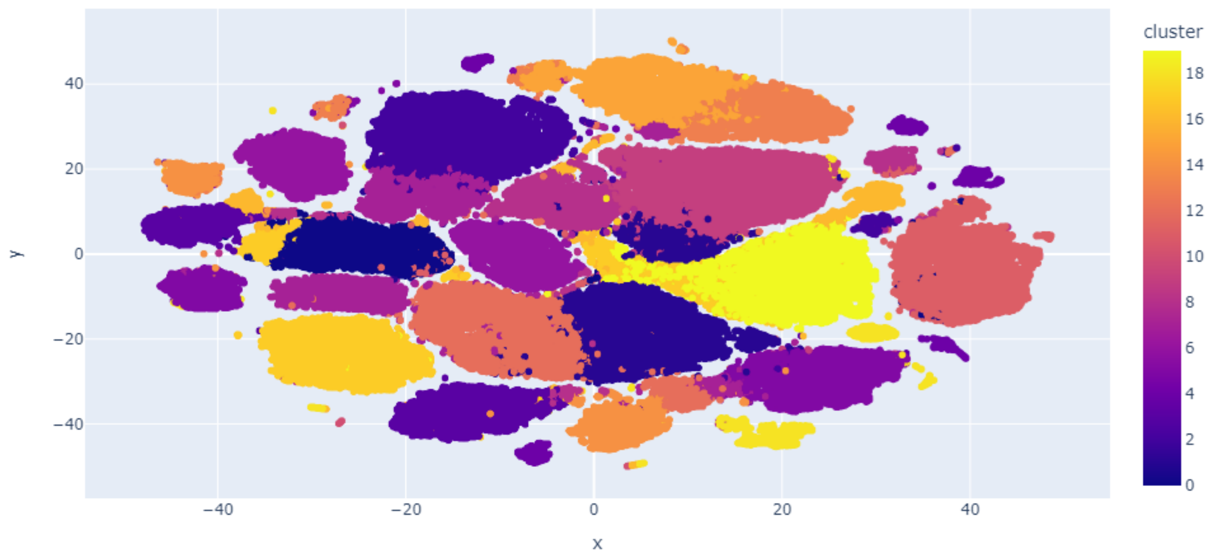
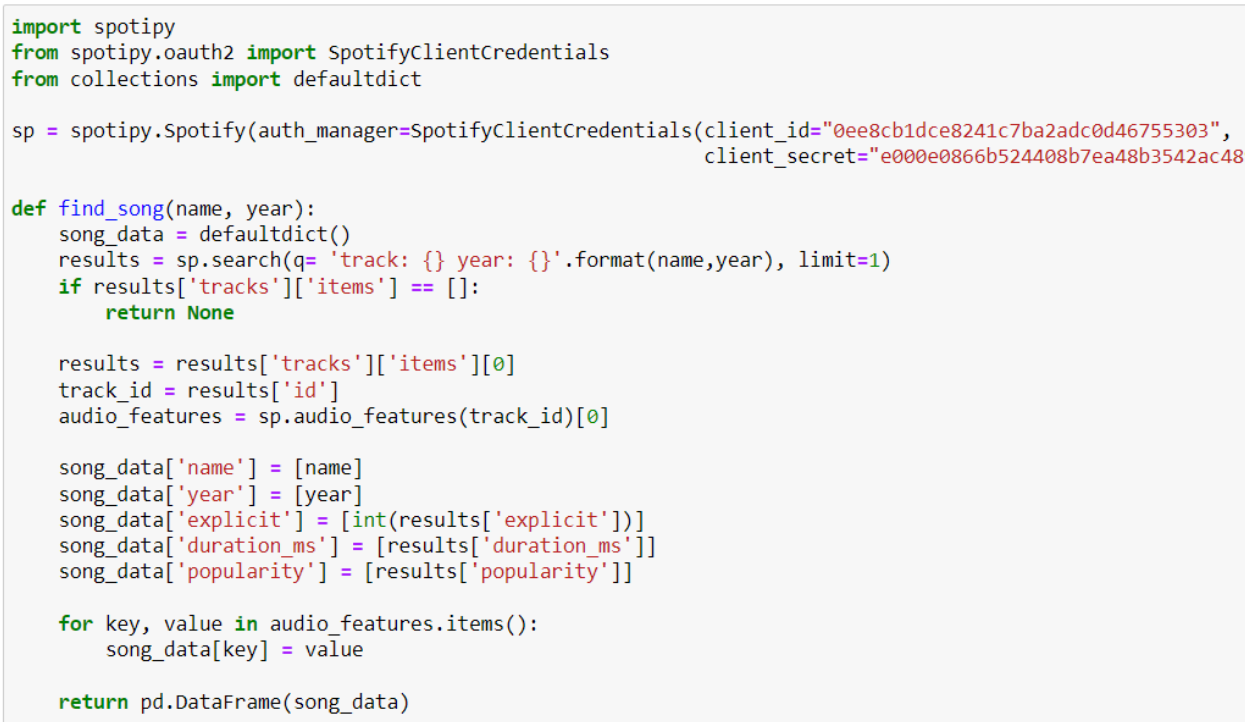
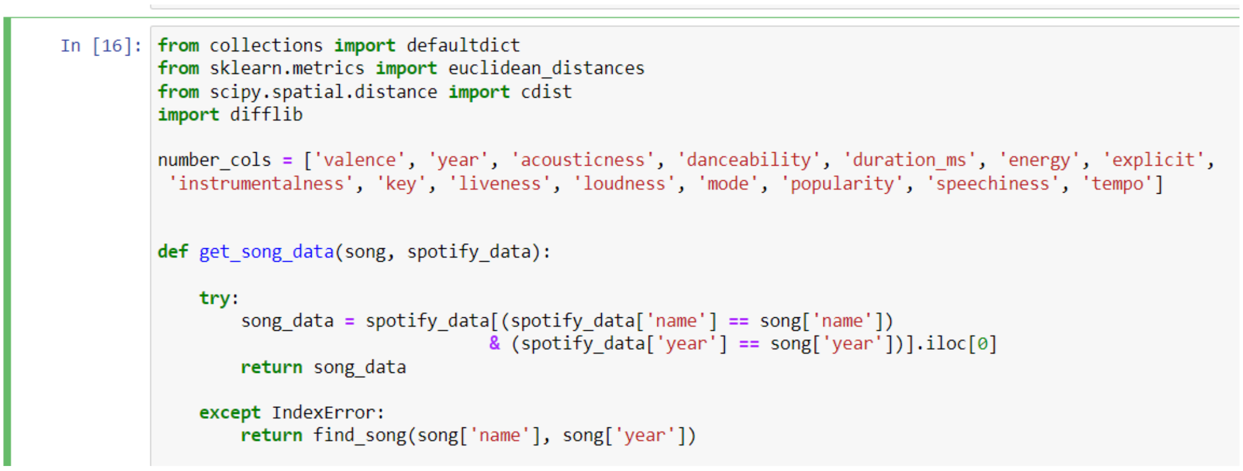


Figure 4.17 – Graph after performing K-Mean Clustering

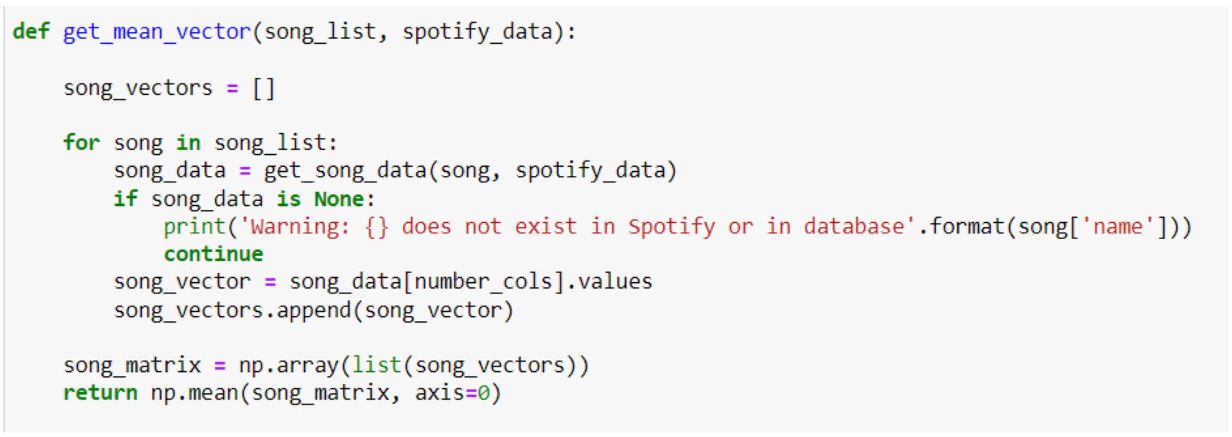
Phase 5: Building recommendation engine. Based on the analysis and visualizations, It is clear that identical genres prefer to have data points close to one another and that songs of the same type likewise tend to group together. This observation does makes sense. Similar genres will sound similar and will come from similar time periods while the same can be said for songs within those genres. We can use this idea to make a recommendation system by taking the data points of the songs a user has listened to and make recommendation of songs corresponding to nearby data points. Figures below show the code for building a recommender engine.



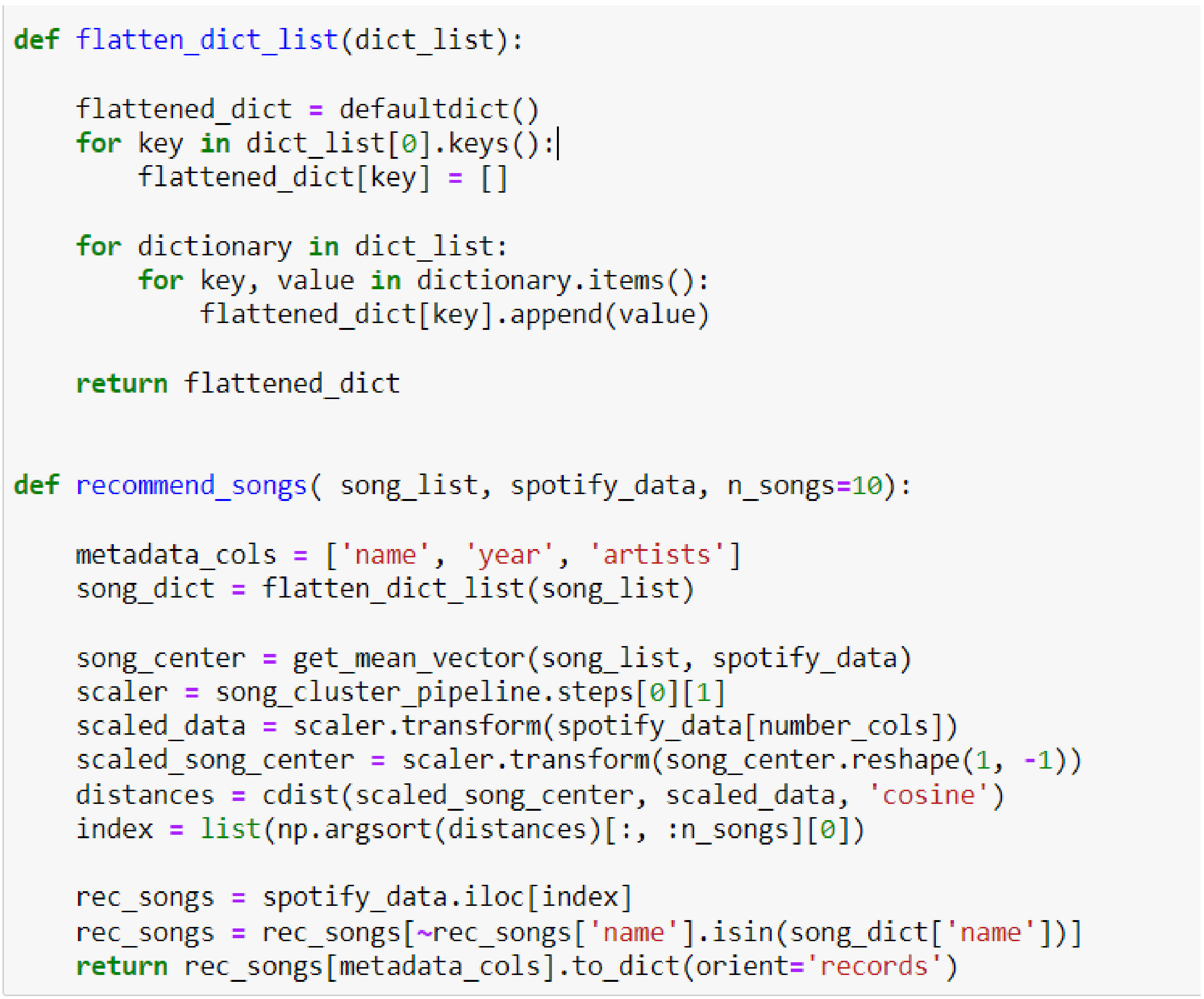
```python
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
from collections import defaultdict

sp = spotipy.Spotify(auth_manager=SpotifyClientCredentials(client_id="0ee8cb1dce8241c7ba2adc0d46755303",
                                                           client_secret="e000e0866b524408b7ea48b3542ac48

def find_song(name, year):
    song_data = defaultdict()
    results = sp.search(q= 'track: {} year: {}'.format(name,year), limit=1)
    if results['tracks']['items'] == []:
        return None

    results = results['tracks']['items'][0]
    track_id = results['id']
    audio_features = sp.audio_features(track_id)[0]

    song_data['name'] = [name]
    song_data['year'] = [year]
    song_data['explicit'] = [int(results['explicit'])]
    song_data['duration_ms'] = [results['duration_ms']]
    song_data['popularity'] = [results['popularity']]

    for key, value in audio_features.items():
        song_data[key] = value

    return pd.DataFrame(song_data)
```



```python
In [16]: from collections import defaultdict
         from sklearn.metrics import euclidean_distances
         from scipy.spatial.distance import cdist
         import difflib

         number_cols = ['valence', 'year', 'acousticness', 'danceability', 'duration_ms', 'energy', 'explicit',
          'instrumentalness', 'key', 'liveness', 'loudness', 'mode', 'popularity', 'speechiness', 'tempo']


         def get_song_data(song, spotify_data):

             try:
                 song_data = spotify_data[(spotify_data['name'] == song['name'])
                                         & (spotify_data['year'] == song['year'])].iloc[0]
                 return song_data

             except IndexError:
                 return find_song(song['name'], song['year'])
```



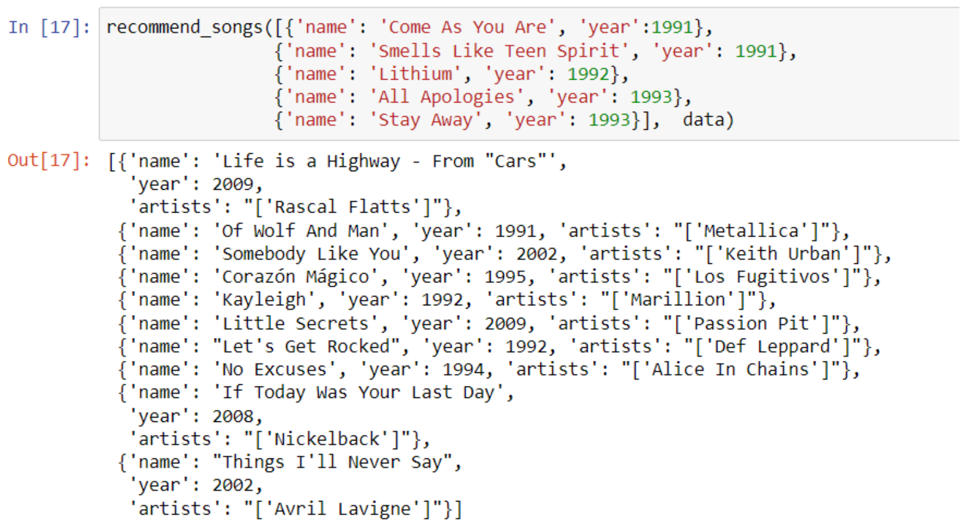
```python
def get_mean_vector(song_list, spotify_data):

    song_vectors = []

    for song in song_list:
        song_data = get_song_data(song, spotify_data)
        if song_data is None:
            print('Warning: {} does not exist in Spotify or in database'.format(song['name']))
            continue
        song_vector = song_data[number_cols].values
        song_vectors.append(song_vector)

    song_matrix = np.array(list(song_vectors))
    return np.mean(song_matrix, axis=0)
```



```python
def flatten_dict_list(dict_list):

    flattened_dict = defaultdict()
    for key in dict_list[0].keys():
        flattened_dict[key] = []

    for dictionary in dict_list:
        for key, value in dictionary.items():
            flattened_dict[key].append(value)

    return flattened_dict


def recommend_songs( song_list, spotify_data, n_songs=10):

    metadata_cols = ['name', 'year', 'artists']
    song_dict = flatten_dict_list(song_list)

    song_center = get_mean_vector(song_list, spotify_data)
    scaler = song_cluster_pipeline.steps[0][1]
    scaled_data = scaler.transform(spotify_data[number_cols])
    scaled_song_center = scaler.transform(song_center.reshape(1, -1))
    distances = cdist(scaled_song_center, scaled_data, 'cosine')
    index = list(np.argsort(distances)[:, :n_songs][0])

    rec_songs = spotify_data.iloc[index]
    rec_songs = rec_songs[~rec_songs['name'].isin(song_dict['name'])]
    return rec_songs[metadata_cols].to_dict(orient='records')
```

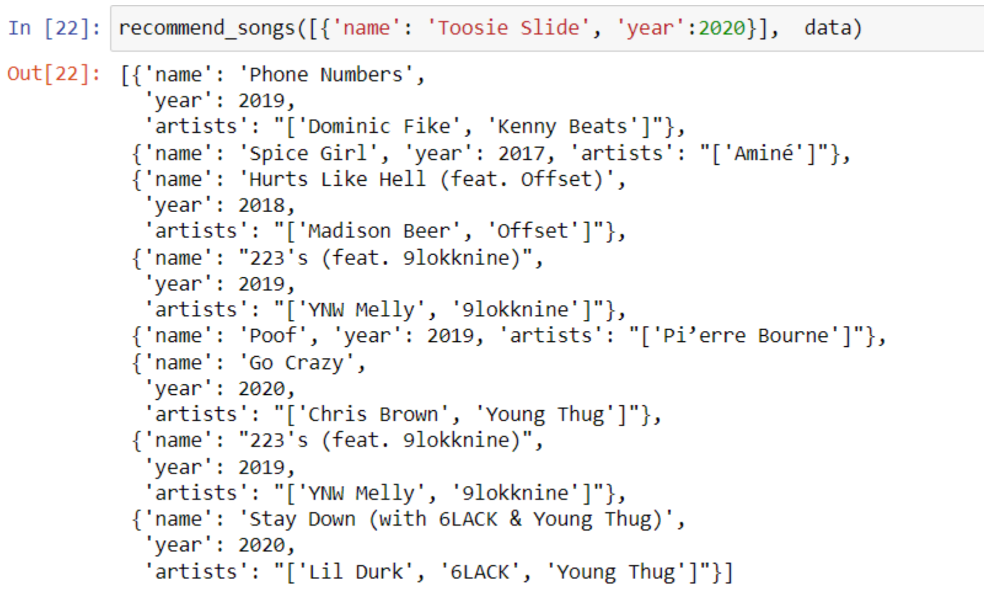
Figure 4.18 – Code to build recommendation engine

Phase 6 & 7: Input and Output of songs. User can input the song title together with the year of release in order to let recommendation engine recommends similar songs. The input can be either a list of song or single song. The output is a list of songs recommend to users, each song shows the details of its title, released year and artists. Figure 4.19 shows recommendation result by inputing a list of songs. Figure 4.20 shows recommendation results based on a single song.



```
In [17]: recommend_songs([{'name': 'Come As You Are', 'year':1991},
                          {'name': 'Smells Like Teen Spirit', 'year': 1991},
                          {'name': 'Lithium', 'year': 1992},
                          {'name': 'All Apologies', 'year': 1993},
                          {'name': 'Stay Away', 'year': 1993}],  data)

Out[17]: [{'name': 'Life is a Highway - From "Cars"',
           'year': 2009,
           'artists': "['Rascal Flatts']"},
          {'name': 'Of Wolf And Man', 'year': 1991, 'artists': "['Metallica']"},
          {'name': 'Somebody Like You', 'year': 2002, 'artists': "['Keith Urban']"},
          {'name': 'Corazón Mágico', 'year': 1995, 'artists': "['Los Fugitivos']"},
          {'name': 'Kayleigh', 'year': 1992, 'artists': "['Marillion']"},
          {'name': 'Little Secrets', 'year': 2009, 'artists': "['Passion Pit']"},
          {'name': "Let's Get Rocked", 'year': 1992, 'artists': "['Def Leppard']"},
          {'name': 'No Excuses', 'year': 1994, 'artists': "['Alice In Chains']"},
          {'name': 'If Today Was Your Last Day',
           'year': 2008,
           'artists': "['Nickelback']"},
          {'name': "Things I'll Never Say",
           'year': 2002,
           'artists': "['Avril Lavigne']"}]
```

Figure 4.19 – Recommendation results based on a list of songs

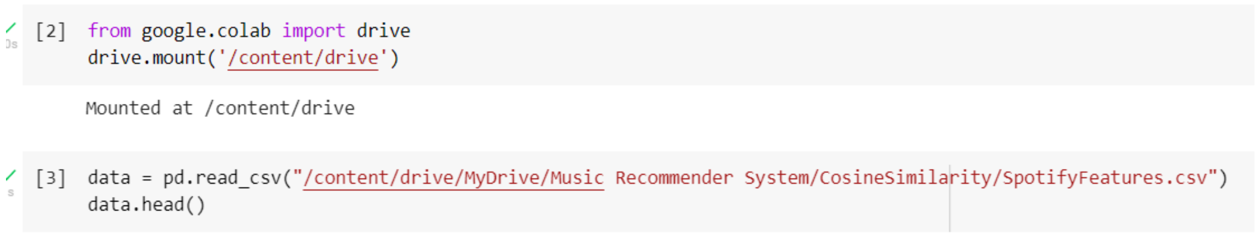


```
In [22]: recommend_songs([{'name': 'Toosie Slide', 'year':2020}],  data)

Out[22]: [{'name': 'Phone Numbers',
           'year': 2019,
           'artists': "['Dominic Fike', 'Kenny Beats']"},
          {'name': 'Spice Girl', 'year': 2017, 'artists': "['Aminé']"},
          {'name': 'Hurts Like Hell (feat. Offset)',
           'year': 2018,
           'artists': "['Madison Beer', 'Offset']"},
          {'name': "223's (feat. 9lokknine)",
           'year': 2019,
           'artists': "['YNW Melly', '9lokknine']"},
          {'name': 'Poof', 'year': 2019, 'artists': "['Pi'erre Bourne']"},
          {'name': 'Go Crazy',
           'year': 2020,
           'artists': "['Chris Brown', 'Young Thug']"},
          {'name': "223's (feat. 9lokknine)",
           'year': 2019,
           'artists': "['YNW Melly', '9lokknine']"},
          {'name': 'Stay Down (with 6LACK & Young Thug)',
           'year': 2020,
           'artists': "['Lil Durk', '6LACK', 'Young Thug']"}]
```

Figure 4.20 – Recommendation result based on single song

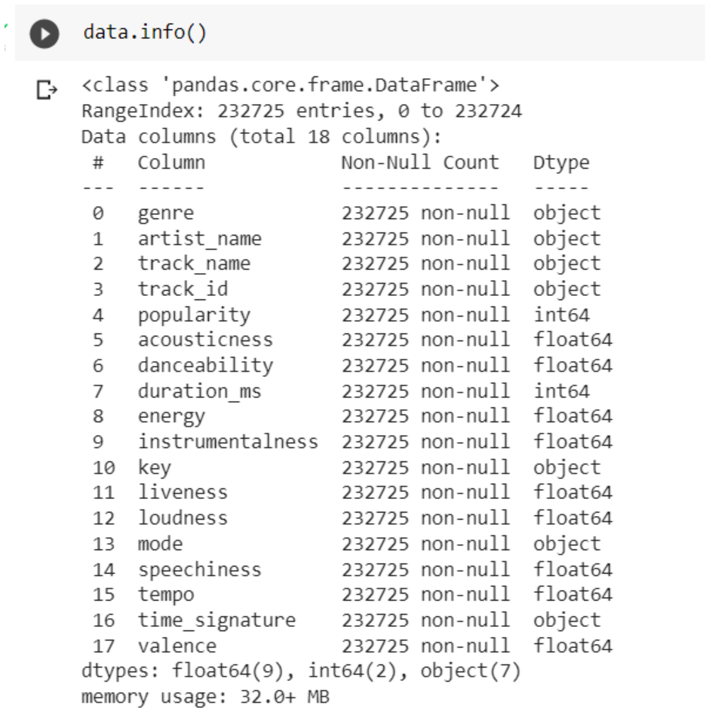
4.3.2 Other Method 1: Cosine Similarity

Phase 1: Data Preparation. The Spotify datasets are also being imported for this method. As shown in Figure 4.21 and Figure 4.22, there are a total of 232725 entries of song in the dataset.



```
[2] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive

[3] data = pd.read_csv("/content/drive/MyDrive/Music Recommender System/CosineSimilarity/SpotifyFeatures.csv")
    data.head()
```

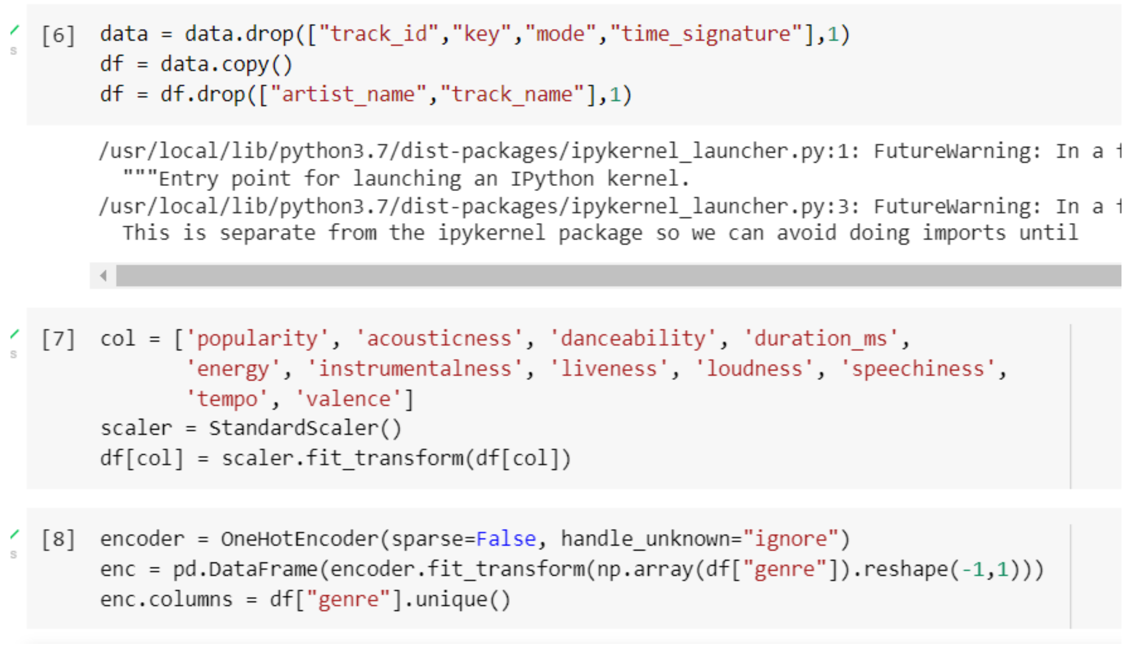
Figure 4.21 – Fetching datasets from Google Drive



```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 232725 entries, 0 to 232724
Data columns (total 18 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   genre             232725 non-null  object
 1   artist_name       232725 non-null  object
 2   track_name        232725 non-null  object
 3   track_id          232725 non-null  object
 4   popularity        232725 non-null  int64
 5   acousticness      232725 non-null  float64
 6   danceability      232725 non-null  float64
 7   duration_ms       232725 non-null  int64
 8   energy            232725 non-null  float64
 9   instrumentalness  232725 non-null  float64
 10  key               232725 non-null  object
 11  liveness          232725 non-null  float64
 12  loudness          232725 non-null  float64
 13  mode              232725 non-null  object
 14  speechiness       232725 non-null  float64
 15  tempo             232725 non-null  float64
 16  time_signature    232725 non-null  object
 17  valence           232725 non-null  float64
dtypes: float64(9), int64(2), object(7)
memory usage: 32.0+ MB
```

Figure 4.22 – Table showing a total of 232725 entries of song

Phase 2: Data Normalization and One Hot Encoding. In this part, data will be normalized and applying one hot encoding to the 'genre' column. Most machine learning algorithm require inputs and outputs variables to be a number, or a numeric in value. This means that any categorical data must be mapped to integers. One hot encoding is one method of converting data to prepare it for an algorithm and get a better prediction. With one-hot, we convert each categorical value into a new categorical column and assign a binary value of 1 or 0 to those columns. Figure 4.23 shows the one-hot encoding process on 'genre' column of the dataset.



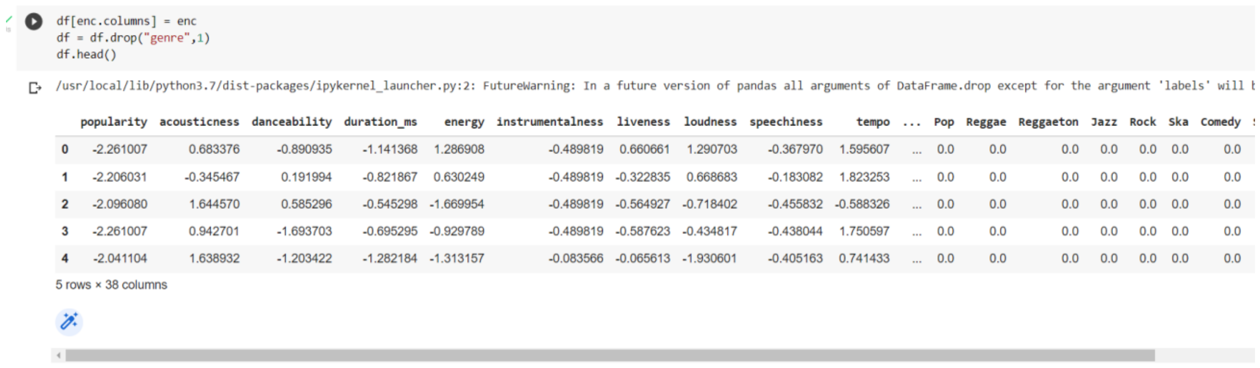
```
[6]  data = data.drop(["track_id","key","mode","time_signature"],1)
     df = data.copy()
     df = df.drop(["artist_name","track_name"],1)

     /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: In a f
       """Entry point for launching an IPython kernel.
     /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: FutureWarning: In a f
       This is separate from the ipykernel package so we can avoid doing imports until
```

```
[7]  col = ['popularity', 'acousticness', 'danceability', 'duration_ms',
            'energy', 'instrumentalness', 'liveness', 'loudness', 'speechiness',
            'tempo', 'valence']
     scaler = StandardScaler()
     df[col] = scaler.fit_transform(df[col])
```

```
[8]  encoder = OneHotEncoder(sparse=False, handle_unknown="ignore")
     enc = pd.DataFrame(encoder.fit_transform(np.array(df["genre"]).reshape(-1,1)))
     enc.columns = df["genre"].unique()
```

Figure 4.23 – Codes of implementing one-hot encoding

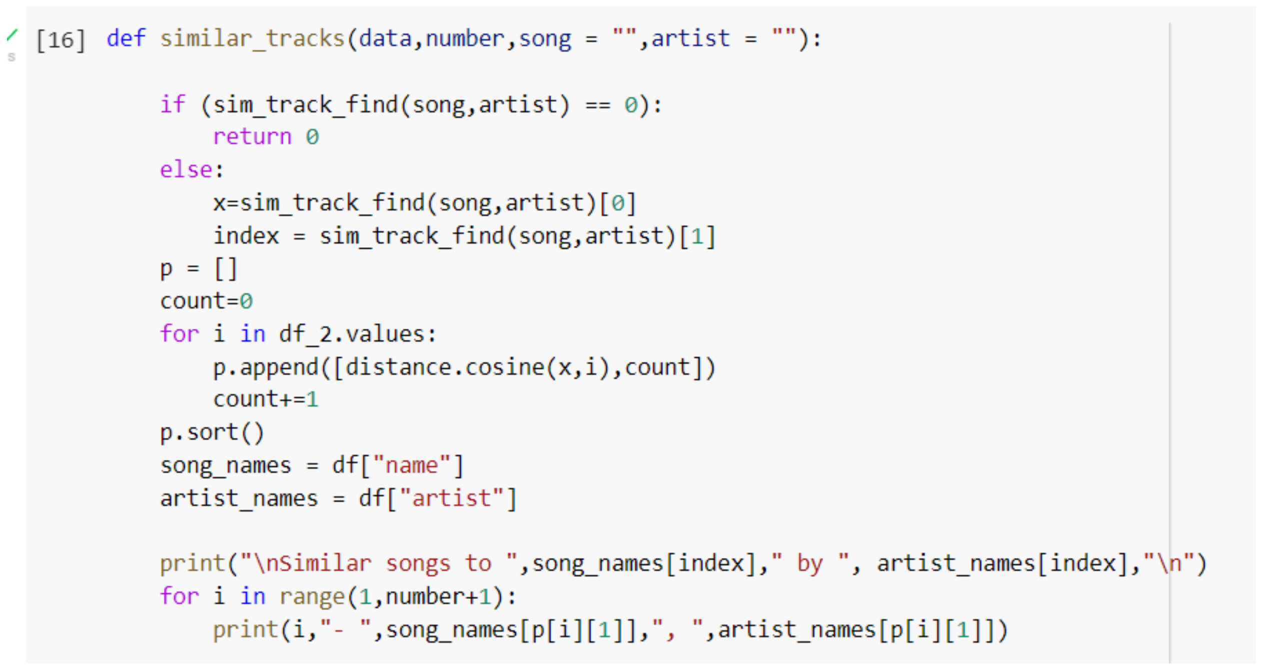
After performing one hot encoding, the original ‘genre’ column is replaced with the encoded one. Furthermore, the column name of ‘track_name’ and ‘artists_name’ have been normalized to artist and name for better understanding. Figure 4.24 shows the combination of encoded ‘genre’ column with other columns in datasets.



```
df[enc.columns] = enc
df = df.drop("genre",1)
df.head()

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will b

   popularity  acousticness  danceability  duration_ms     energy  instrumentalness  liveness  loudness  speechiness     tempo  ...  Pop  Reggae  Reggaeton  Jazz  Rock  Ska  Comedy  :
0   -2.261007      0.683376     -0.890935    -1.141368   1.286908         -0.489819  0.660661  1.290703    -0.367970  1.595607  ...  0.0     0.0        0.0   0.0   0.0  0.0     0.0
1   -2.206031     -0.345467      0.191994    -0.821867   0.630249         -0.489819 -0.322835  0.668683    -0.183082  1.823253  ...  0.0     0.0        0.0   0.0   0.0  0.0     0.0
2   -2.096080      1.644570      0.585296    -0.545298  -1.669954         -0.489819 -0.564927 -0.718402    -0.455832 -0.588326  ...  0.0     0.0        0.0   0.0   0.0  0.0     0.0
3   -2.261007      0.942701     -1.693703    -0.695295  -0.929789         -0.489819 -0.587623 -0.434817    -0.438044  1.750597  ...  0.0     0.0        0.0   0.0   0.0  0.0     0.0
4   -2.041104      1.638932     -1.203422    -1.282184  -1.313157         -0.083566 -0.065613 -1.930601    -0.405163  0.741433  ...  0.0     0.0        0.0   0.0   0.0  0.0     0.0
5 rows × 38 columns
```

Figure 4.24 – Combination of original and encoded columns

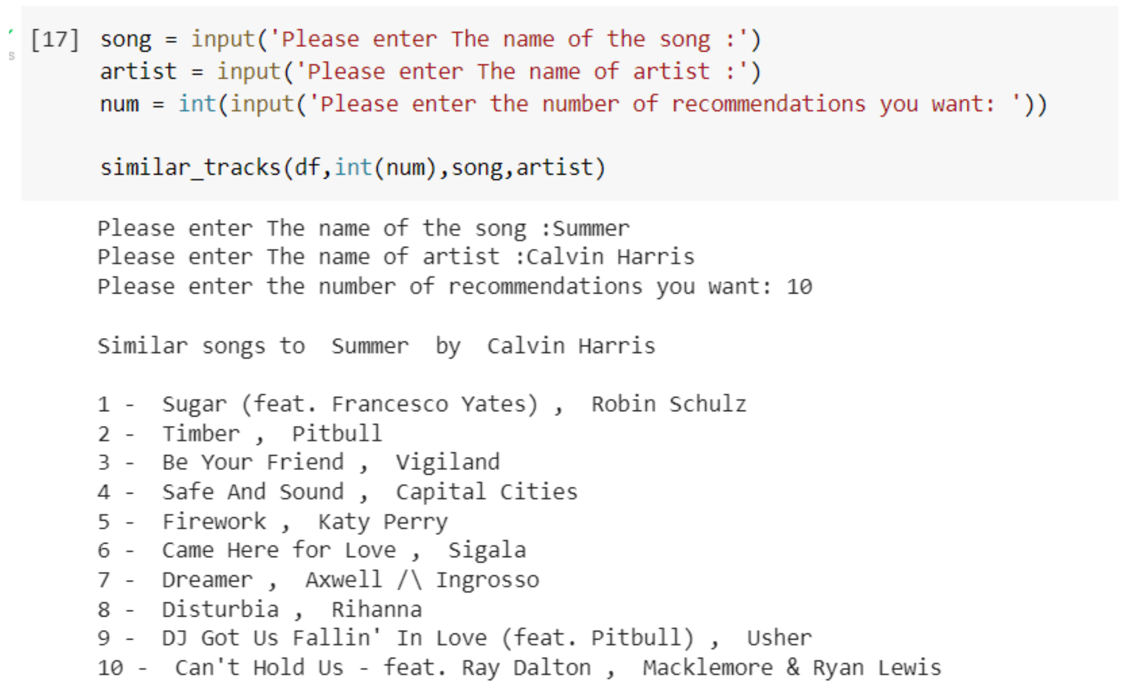
Phase 3: Building system using cosine similarity. Figure 4.25 shows the function that finds similar tracks based on user input. Cosine distance is used to measure the distances between songs in dataset. Cosine similarity measures the similarity between two vectors of an inner product space based on its cosine distance. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It will then recommends songs with high similarity to users.



```
[16] def similar_tracks(data,number,song = "",artist = ""):

         if (sim_track_find(song,artist) == 0):
             return 0
         else:
             x=sim_track_find(song,artist)[0]
             index = sim_track_find(song,artist)[1]
         p = []
         count=0
         for i in df_2.values:
             p.append([distance.cosine(x,i),count])
             count+=1
         p.sort()
         song_names = df["name"]
         artist_names = df["artist"]

         print("\nSimilar songs to ",song_names[index]," by ", artist_names[index],"\n")
         for i in range(1,number+1):
             print(i,"- ",song_names[p[i][1]],", ",artist_names[p[i][1]])
```

Figure 4.25 – Code of calculating cosine distance based on user input

Phase 4: Input and Output. User can input the song name, artist of the song and desired number of recommendations. Based on that, the recommendation engine will print out the recommended songs. Figure 4.26 shows the input and output of user and system.

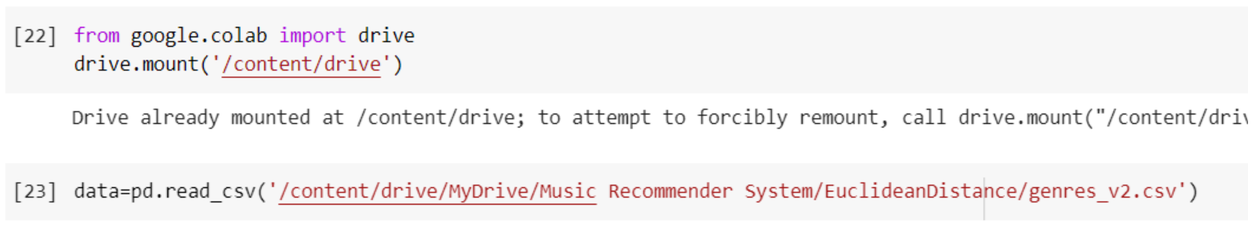


```
[17] song = input('Please enter The name of the song :')
     artist = input('Please enter The name of artist :')
     num = int(input('Please enter the number of recommendations you want: '))

     similar_tracks(df,int(num),song,artist)

     Please enter The name of the song :Summer
     Please enter The name of artist :Calvin Harris
     Please enter the number of recommendations you want: 10

     Similar songs to  Summer  by  Calvin Harris

     1 -  Sugar (feat. Francesco Yates) ,  Robin Schulz
     2 -  Timber ,  Pitbull
     3 -  Be Your Friend ,  Vigiland
     4 -  Safe And Sound ,  Capital Cities
     5 -  Firework ,  Katy Perry
     6 -  Came Here for Love ,  Sigala
     7 -  Dreamer ,  Axwell /\ Ingrosso
     8 -  Disturbia ,  Rihanna
     9 -  DJ Got Us Fallin' In Love (feat. Pitbull) ,  Usher
     10 -  Can't Hold Us - feat. Ray Dalton ,  Macklemore & Ryan Lewis
```

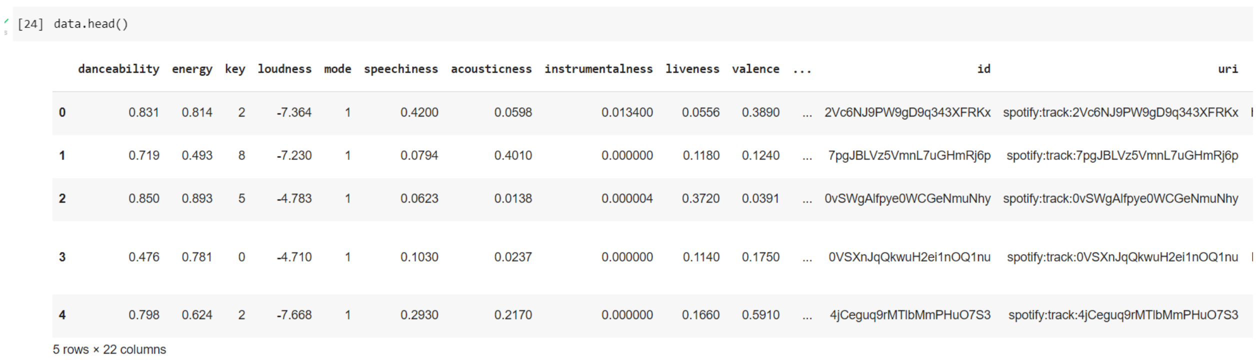
Figure 4.26 – Input and output of system

4.3.3 Other Method 2: Euclidean Distance

Phase 1: Data Preparation. Figure 4.27 shows that the data is imported from Google Drive. The full list of genres included in the CSV are Trap, Techno, Techhouse, Trance, Psytrance, Dark Trap, DnB (drums and bass), Hardstyle, Underground Rap, Trap Metal, Emo, Rap, RnB, Pop and Hiphop.



```
[22] from google.colab import drive
     drive.mount('/content/drive')

     Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/dri

[23] data=pd.read_csv('/content/drive/MyDrive/Music Recommender System/EuclideanDistance/genres_v2.csv')
```



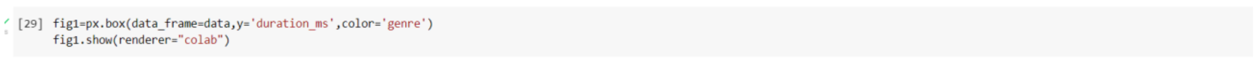
```
[24] data.head()
```

	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	...	id	uri
0	0.831	0.814	2	-7.364	1	0.4200	0.0598	0.013400	0.0556	0.3890	...	2Vc6NJ9PW9gD9q343XFRKx	spotify:track:2Vc6NJ9PW9gD9q343XFRKx
1	0.719	0.493	8	-7.230	1	0.0794	0.4010	0.000000	0.1180	0.1240	...	7pgJBLVz5VmnL7uGHmRj6p	spotify:track:7pgJBLVz5VmnL7uGHmRj6p
2	0.850	0.893	5	-4.783	1	0.0623	0.0138	0.000004	0.3720	0.0391	...	0vSWgAlfpye0WCGeNmuNhy	spotify:track:0vSWgAlfpye0WCGeNmuNhy
3	0.476	0.781	0	-4.710	1	0.1030	0.0237	0.000000	0.1140	0.1750	...	0VSXnJqQkwuH2ei1nOQ1nu	spotify:track:0VSXnJqQkwuH2ei1nOQ1nu
4	0.798	0.624	2	-7.668	1	0.2930	0.2170	0.000000	0.1660	0.5910	...	4jCeguq9rMTlbMmPHuO7S3	spotify:track:4jCeguq9rMTlbMmPHuO7S3

5 rows × 22 columns

Figure 4.27 – Table showing details of dataset

Figure 4.28 shows the average play time of a song. It is clearly see that most of the genres have their own time ranges pystrance is mostly longer and can see that trap music is of lower timespan.



```
[29] fig1=px.box(data_frame=data,y='duration_ms',color='genre')
     fig1.show(renderer="colab")
```

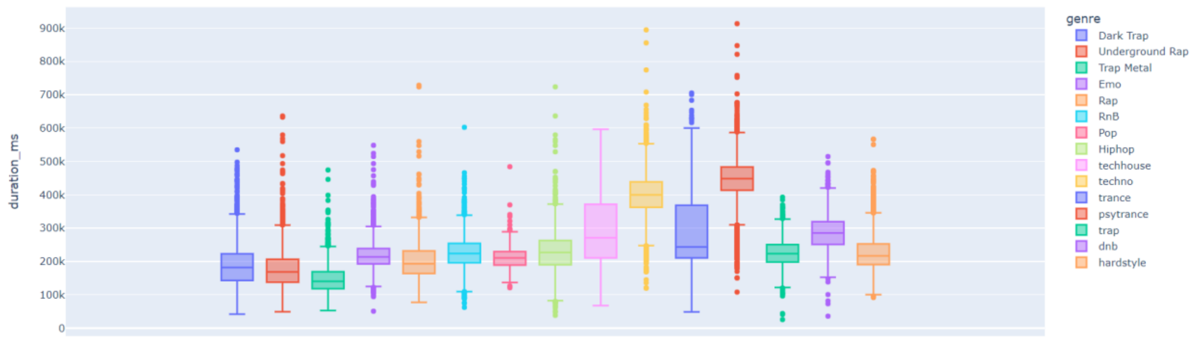
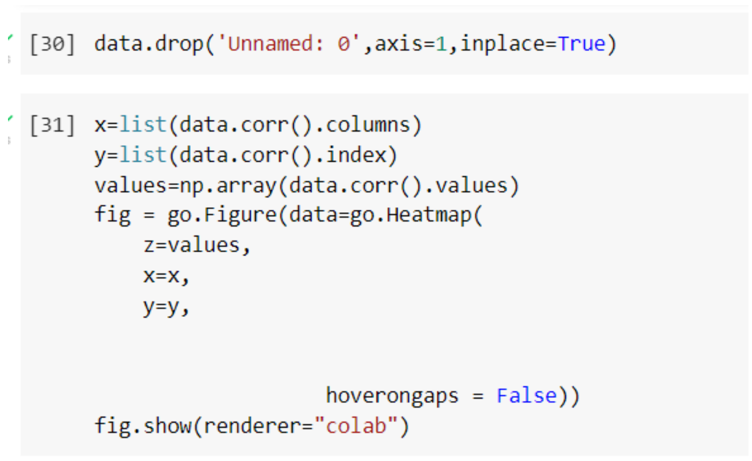


Figure 4.28 – Average playtime of songs according to different genre

Heat Maps are graphical representations of data that utilize color-coded systems. The primary purpose of Heat Maps is to better visualize the volume of features within a dataset and assist in directing viewers towards areas on data visualizations that matter most. Figure 4.29 shows the heat map of the dataset.



```
[30] data.drop('Unnamed: 0',axis=1,inplace=True)
```

```
[31] x=list(data.corr().columns)
     y=list(data.corr().index)
     values=np.array(data.corr().values)
     fig = go.Figure(data=go.Heatmap(
         z=values,
         x=x,
         y=y,


                             hoverongaps = False))
     fig.show(renderer="colab")
```

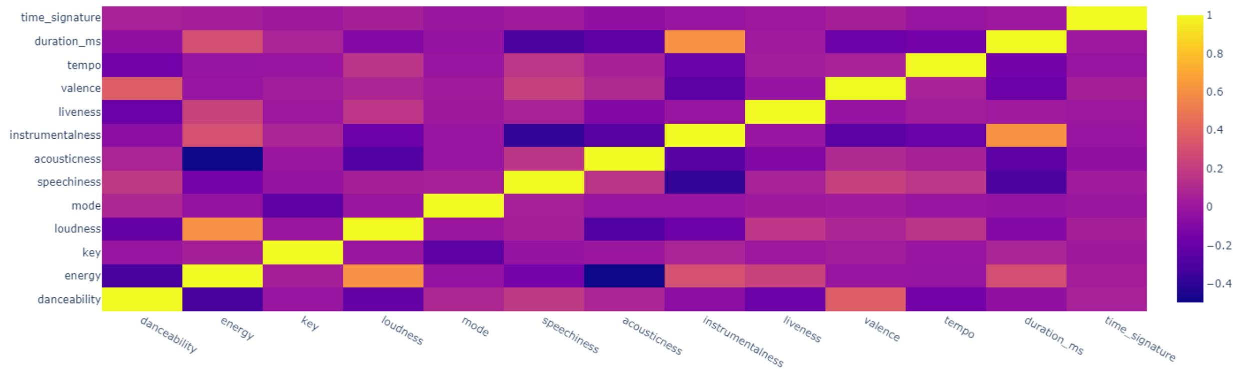
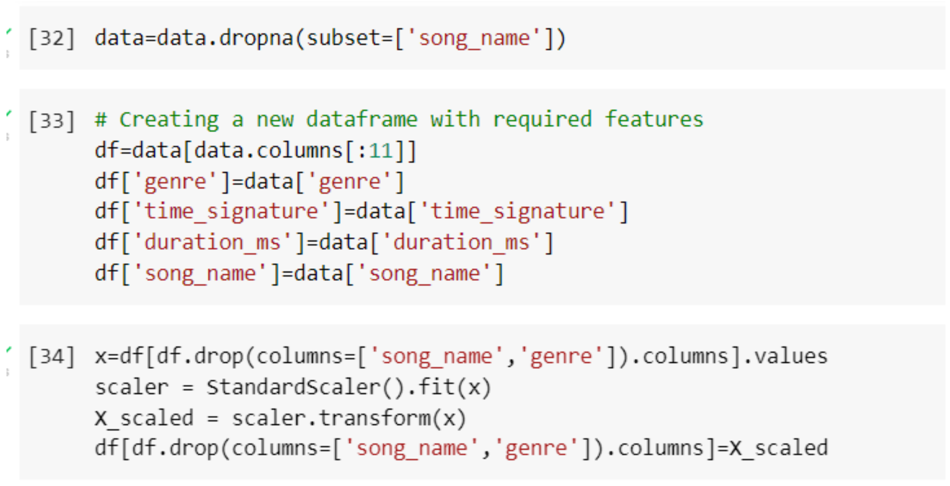


Figure 4.29 – Heatmap of dataset

Phase 2: Data pre-process and standardization. It can be seen that the data has columns like duration ms whose distance difference can be high causing lot of change in the answer we want every field to contribute the same to the distance (Euclidean) hence we have to standardize the data. Figure 4.30 shows the code on how to standardize data.



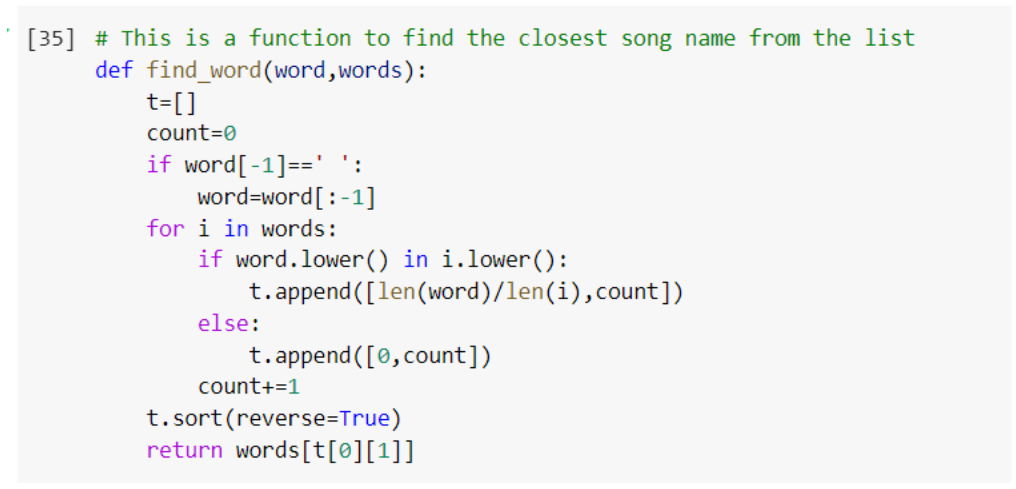
```
[32]  data=data.dropna(subset=['song_name'])
```

```
[33]  # Creating a new dataframe with required features
      df=data[data.columns[:11]]
      df['genre']=data['genre']
      df['time_signature']=data['time_signature']
      df['duration_ms']=data['duration_ms']
      df['song_name']=data['song_name']
```

```
[34]  x=df[df.drop(columns=['song_name','genre']).columns].values
      scaler = StandardScaler().fit(x)
      X_scaled = scaler.transform(x)
      df[df.drop(columns=['song_name','genre']).columns]=X_scaled
```

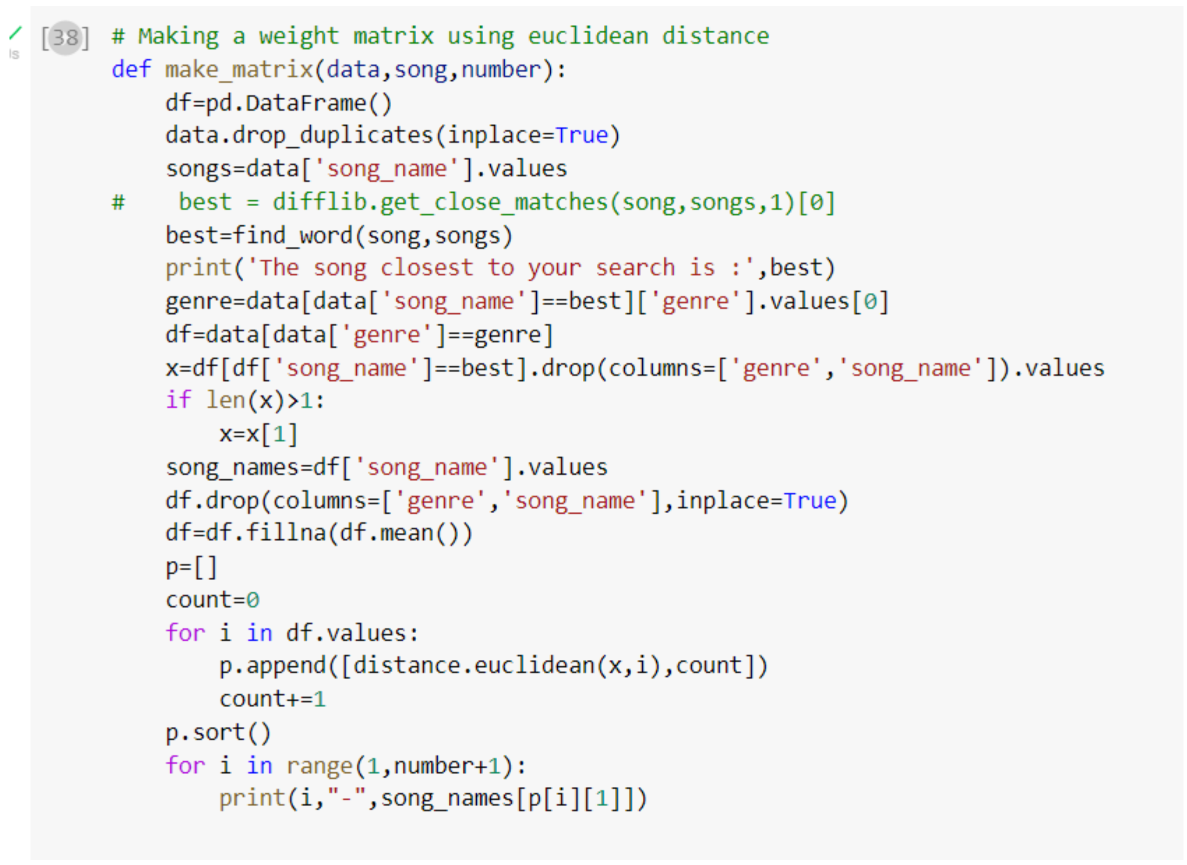
Figure 4.30 – Standardization of data

Phase 3: Building recommender system. Euclidean distance is a measure of the straight-line distance between two points. In data science and machine learning, it is often used to measure the similarity between two data points. Figure 4.31 and Figure 4.32 shows code of building a recommender system based on Euclidean distance.



```
[35]  # This is a function to find the closest song name from the list
      def find_word(word,words):
          t=[]
          count=0
          if word[-1]==' ':
              word=word[:-1]
          for i in words:
              if word.lower() in i.lower():
                  t.append([len(word)/len(i),count])
              else:
                  t.append([0,count])
              count+=1
          t.sort(reverse=True)
          return words[t[0][1]]
```

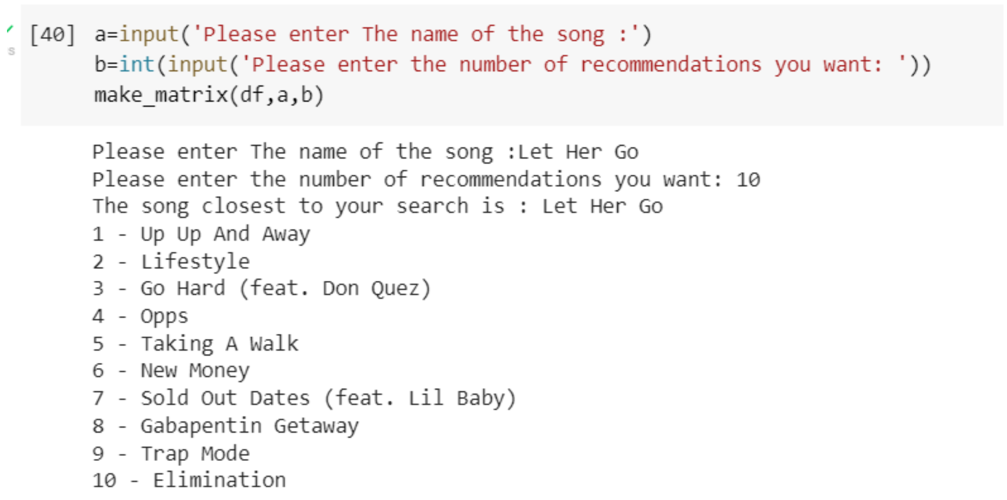
Figure 4.31 – Codes to find closest song name based on user input



```
[38] # Making a weight matrix using euclidean distance
     def make_matrix(data,song,number):
         df=pd.DataFrame()
         data.drop_duplicates(inplace=True)
         songs=data['song_name'].values
     #    best = difflib.get_close_matches(song,songs,1)[0]
         best=find_word(song,songs)
         print('The song closest to your search is :',best)
         genre=data[data['song_name']==best]['genre'].values[0]
         df=data[data['genre']==genre]
         x=df[df['song_name']==best].drop(columns=['genre','song_name']).values
         if len(x)>1:
             x=x[1]
         song_names=df['song_name'].values
         df.drop(columns=['genre','song_name'],inplace=True)
         df=df.fillna(df.mean())
         p=[]
         count=0
         for i in df.values:
             p.append([distance.euclidean(x,i),count])
             count+=1
         p.sort()
         for i in range(1,number+1):
             print(i,"-",song_names[p[i][1]])
```

Figure 4.32 – Codes of making recommendation based on Euclidean Distance

Phase 4: Input and Output. Users are required to enter song name and how many recommendations in order for system to recommends. Figure 4.33 shows the input and output of system.



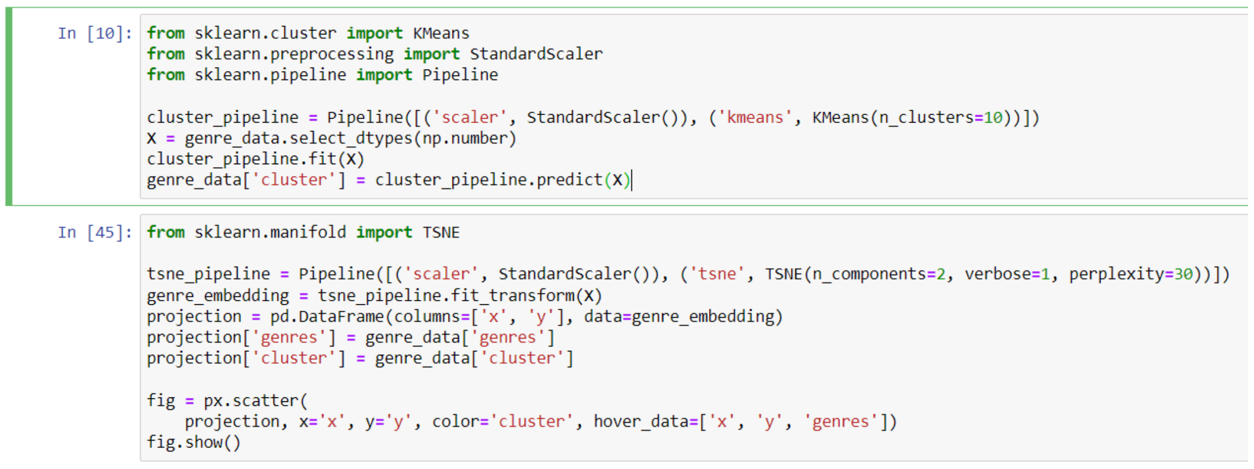
```
[40] a=input('Please enter The name of the song :')
     b=int(input('Please enter the number of recommendations you want: '))
     make_matrix(df,a,b)
```

```
Please enter The name of the song :Let Her Go
Please enter the number of recommendations you want: 10
The song closest to your search is : Let Her Go
1 - Up Up And Away
2 - Lifestyle
3 - Go Hard (feat. Don Quez)
4 - Opps
5 - Taking A Walk
6 - New Money
7 - Sold Out Dates (feat. Lil Baby)
8 - Gabapentin Getaway
9 - Trap Mode
10 - Elimination
```

Figure 4.33 – Input and output of the system

4.4 Discussion

4.4.1 K-Mean Clustering



```
In [10]: from sklearn.cluster import KMeans
         from sklearn.preprocessing import StandardScaler
         from sklearn.pipeline import Pipeline

         cluster_pipeline = Pipeline([('scaler', StandardScaler()), ('kmeans', KMeans(n_clusters=10))])
         X = genre_data.select_dtypes(np.number)
         cluster_pipeline.fit(X)
         genre_data['cluster'] = cluster_pipeline.predict(X)
```

```
In [45]: from sklearn.manifold import TSNE

         tsne_pipeline = Pipeline([('scaler', StandardScaler()), ('tsne', TSNE(n_components=2, verbose=1, perplexity=30))])
         genre_embedding = tsne_pipeline.fit_transform(X)
         projection = pd.DataFrame(columns=['x', 'y'], data=genre_embedding)
         projection['genres'] = genre_data['genres']
         projection['cluster'] = genre_data['cluster']

         fig = px.scatter(
             projection, x='x', y='y', color='cluster', hover_data=['x', 'y', 'genres'])
         fig.show()
```

Figure 4.34 – Codes of performing K-Mean Clustering

One of the clustering visualization method, t-SNE is used in this case instead of PCA, as for a small dataset, t-SNE tends to handle non linear data efficiently, it is able to interpret complex polynomial relationships between features comparatively better. In K-Mean Clustering, there is no fix number or to be said the best number of clusters to be used, therefore, in this case, 10 clusters is being used. That is to say the dataset is being divided into 10 clusters with similar attributes according to their feature values (i.e. energy, liveness, loudness, etc.). The perplexity is set to 30, which means 30 nearest neighbors will be taken into calculation at a time, it will produce denser clusters. In this case, default number of iterations is used, which is 1000, the algorithm will redefine data values with the centroid of clusters 100 times before finalize the clusters. Figure 4.35, 4.36, 4.37, 4.38 each showing different clustering result after running the codes for 4 separate times.

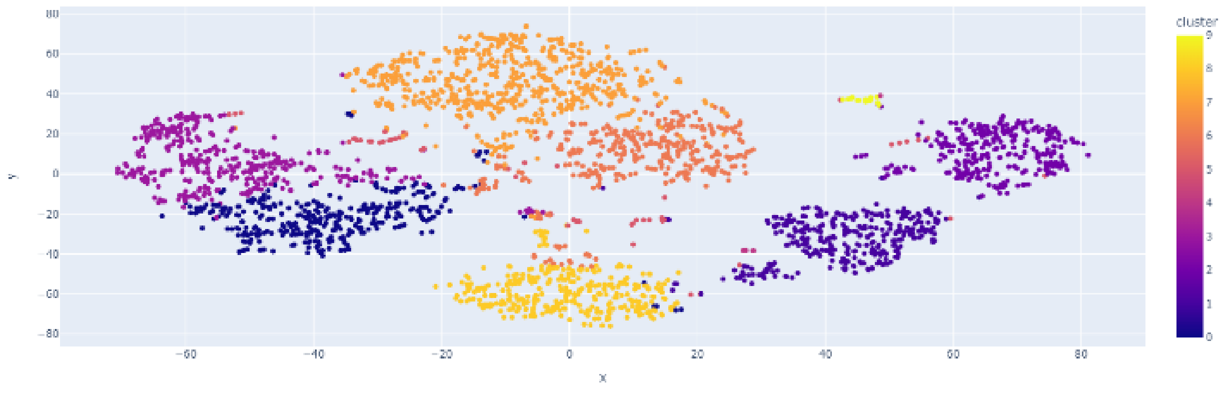


Figure 4.35 – First run of K-Mean Clustering

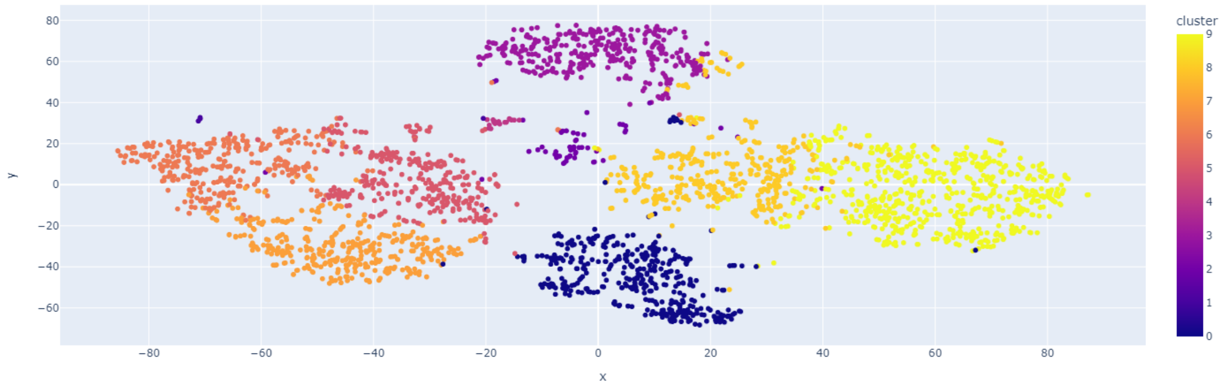


Figure 4.36 – Second run of K-Mean Clustering

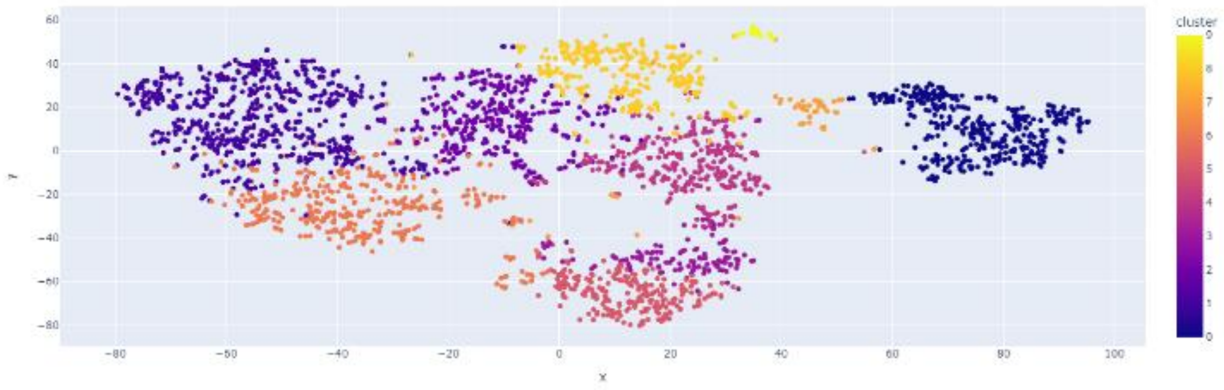


Figure 4.37 – Third run of K-Mean Clustering

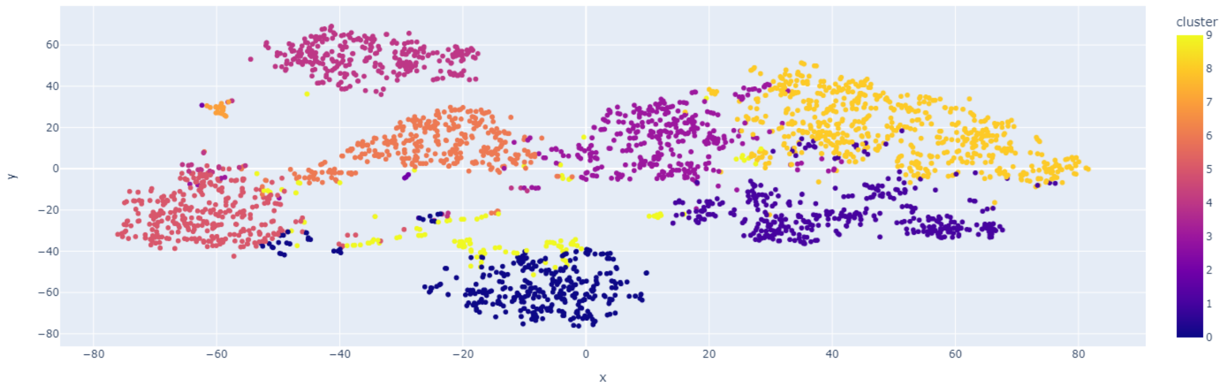
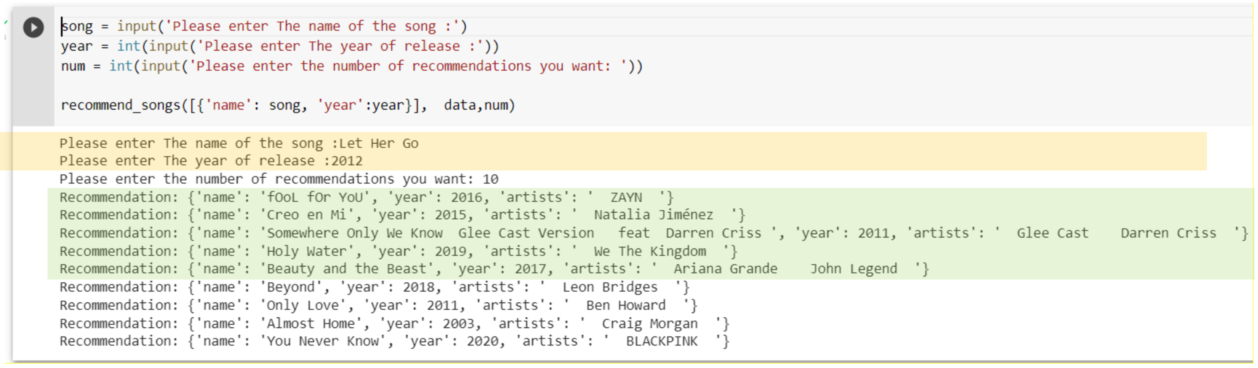


Figure 4.38 – Fourth run of K-Mean Clustering

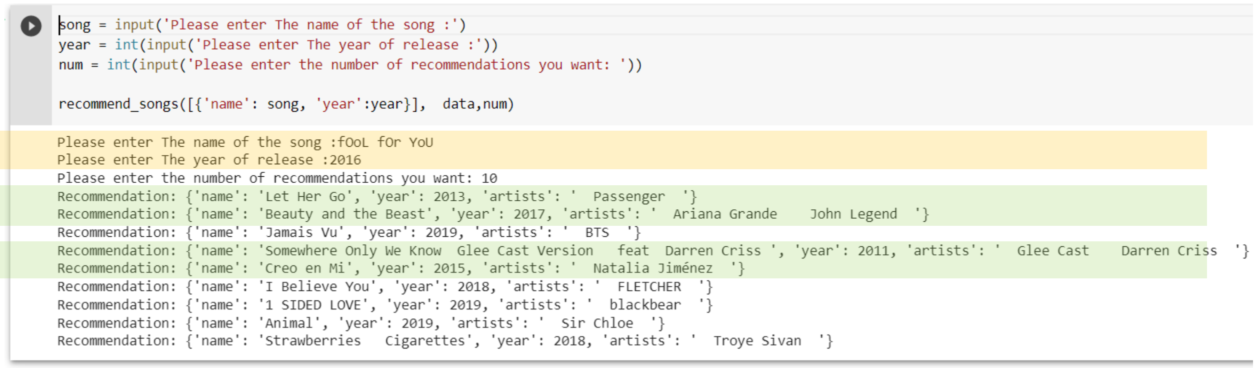
As shown in the figures, each run of K-Mean clustering algorithm will give different clusters result. This is because K-means clustering does involve a random selection process for the initial centroid guesses, after that, it selects k number of random points and calculates the distance between points and further minimize the distance by finding new centroids. So, it may get different results from different runs as the centroids location is different each run.



```
song = input('Please enter The name of the song :')
year = int(input('Please enter The year of release :'))
num = int(input('Please enter the number of recommendations you want: '))

recommend_songs([{'name': song, 'year':year}],  data,num)
```

```
Please enter The name of the song :Let Her Go
Please enter The year of release :2012
Please enter the number of recommendations you want: 10
Recommendation: {'name': 'fOoL fOr YoU', 'year': 2016, 'artists': '  ZAYN  '}
Recommendation: {'name': 'Creo en Mi', 'year': 2015, 'artists': '  Natalia Jiménez  '}
Recommendation: {'name': 'Somewhere Only We Know  Glee Cast Version   feat  Darren Criss ', 'year': 2011, 'artists': '  Glee Cast    Darren Criss  '}
Recommendation: {'name': 'Holy Water', 'year': 2019, 'artists': '  We The Kingdom  '}
Recommendation: {'name': 'Beauty and the Beast', 'year': 2017, 'artists': '  Ariana Grande    John Legend  '}
Recommendation: {'name': 'Beyond', 'year': 2018, 'artists': '  Leon Bridges  '}
Recommendation: {'name': 'Only Love', 'year': 2011, 'artists': '  Ben Howard  '}
Recommendation: {'name': 'Almost Home', 'year': 2003, 'artists': '  Craig Morgan  '}
Recommendation: {'name': 'You Never Know', 'year': 2020, 'artists': '  BLACKPINK  '}
```

Figure 4.39 – Recommendation based on ‘Let Her Go’



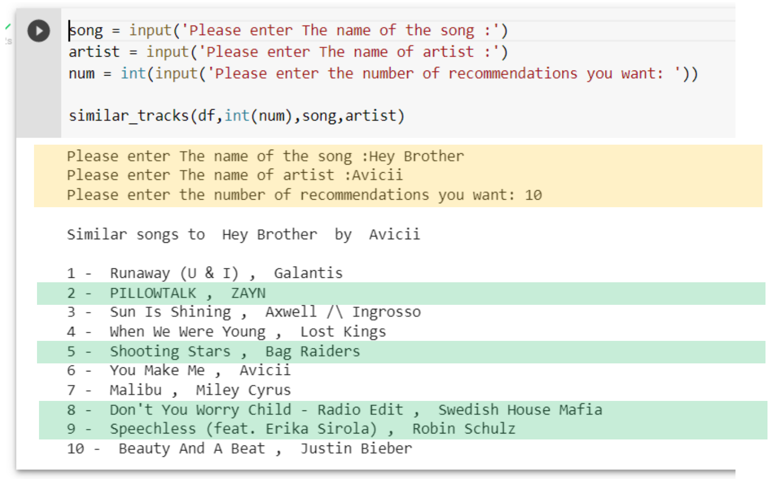
```
song = input('Please enter The name of the song :')
year = int(input('Please enter The year of release :'))
num = int(input('Please enter the number of recommendations you want: '))

recommend_songs([{'name': song, 'year':year}],  data,num)
```

```
Please enter The name of the song :fOoL fOr YoU
Please enter The year of release :2016
Please enter the number of recommendations you want: 10
Recommendation: {'name': 'Let Her Go', 'year': 2013, 'artists': '  Passenger  '}
Recommendation: {'name': 'Beauty and the Beast', 'year': 2017, 'artists': '  Ariana Grande    John Legend  '}
Recommendation: {'name': 'Jamais Vu', 'year': 2019, 'artists': '  BTS  '}
Recommendation: {'name': 'Somewhere Only We Know  Glee Cast Version   feat  Darren Criss ', 'year': 2011, 'artists': '  Glee Cast    Darren Criss  '}
Recommendation: {'name': 'Creo en Mi', 'year': 2015, 'artists': '  Natalia Jiménez  '}
Recommendation: {'name': 'I Believe You', 'year': 2018, 'artists': '  FLETCHER  '}
Recommendation: {'name': '1 SIDED LOVE', 'year': 2019, 'artists': '  blackbear  '}
Recommendation: {'name': 'Animal', 'year': 2019, 'artists': '  Sir Chloe  '}
Recommendation: {'name': 'Strawberries   Cigarettes', 'year': 2018, 'artists': '  Troye Sivan  '}
```

Figure 4.40 – Recommendation based on ‘fOoL fOr YoU’

Figure 4.39 shows the results of recommendation based on song input of ‘Let Her Go’ while Figure 4.40 shows the results of recommendation based on song input ‘fOoL fOr YoU’. As shown in both figures, the recommended songs for each case overlapped with each other (songs highlighted with green color). As ‘fOoL fOr YoU’ is the first song recommended based on song input of ‘Let Her Go’, ‘Let Her Go’ is also being recommended when song input changed to ‘fOoL fOr YoU’, the same goes to the other overlapping songs. This means that the recommendation engine has categorized them into the same cluster, and from the same cluster, recommendation is made to the user.

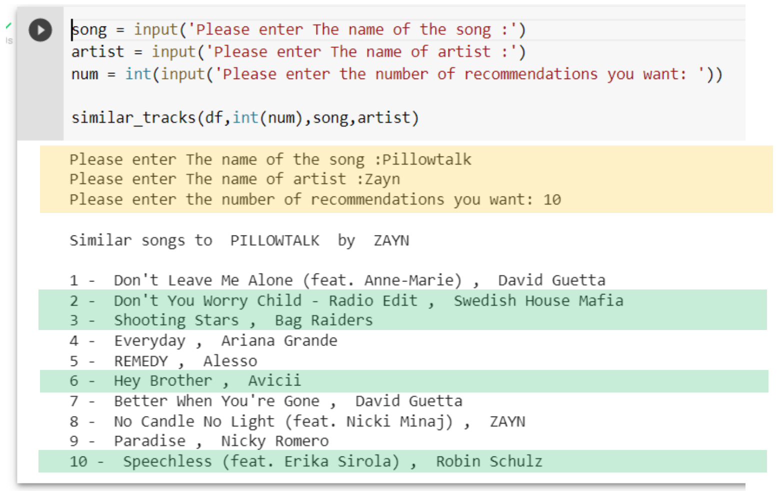
4.4.2 Cosine Similarity



```
song = input('Please enter The name of the song :')
artist = input('Please enter The name of artist :')
num = int(input('Please enter the number of recommendations you want: '))

similar_tracks(df,int(num),song,artist)
```

```
Please enter The name of the song :Hey Brother
Please enter The name of artist :Avicii
Please enter the number of recommendations you want: 10

Similar songs to  Hey Brother  by  Avicii

1 -  Runaway (U & I) ,  Galantis
2 -  PILLOWTALK ,  ZAYN
3 -  Sun Is Shining ,  Axwell /\ Ingrosso
4 -  When We Were Young ,  Lost Kings
5 -  Shooting Stars ,  Bag Raiders
6 -  You Make Me ,  Avicii
7 -  Malibu ,  Miley Cyrus
8 -  Don't You Worry Child - Radio Edit ,  Swedish House Mafia
9 -  Speechless (feat. Erika Sirola) ,  Robin Schulz
10 -  Beauty And A Beat ,  Justin Bieber
```

Figure 4.41 – Recommendation based on 'Hey Brother'



```
song = input('Please enter The name of the song :')
artist = input('Please enter The name of artist :')
num = int(input('Please enter the number of recommendations you want: '))

similar_tracks(df,int(num),song,artist)
```

```
Please enter The name of the song :Pillowtalk
Please enter The name of artist :Zayn
Please enter the number of recommendations you want: 10

Similar songs to  PILLOWTALK  by  ZAYN

1 -  Don't Leave Me Alone (feat. Anne-Marie) ,  David Guetta
2 -  Don't You Worry Child - Radio Edit ,  Swedish House Mafia
3 -  Shooting Stars ,  Bag Raiders
4 -  Everyday ,  Ariana Grande
5 -  REMEDY ,  Alesso
6 -  Hey Brother ,  Avicii
7 -  Better When You're Gone ,  David Guetta
8 -  No Candle No Light (feat. Nicki Minaj) ,  ZAYN
9 -  Paradise ,  Nicky Romero
10 -  Speechless (feat. Erika Sirola) ,  Robin Schulz
```

Figure 4.42 – Recommendation based on 'Pillowtalk'

Figure 4.41 shows the recommendation made from cosine distance based on 'Hey Brother' by Avicii while Figure 4.42 shows the recommendation made based on 'Pillowtalk' by Zayn. As shown in the figure, both songs appear in each other recommendation list, which means that their cosine distance with each other is short. There are also other overlapping songs (highlighted in green) in recommendation means that all of these highlighted songs are close to each other, measuring using cosine distance.

4.4.3 Euclidean Distance

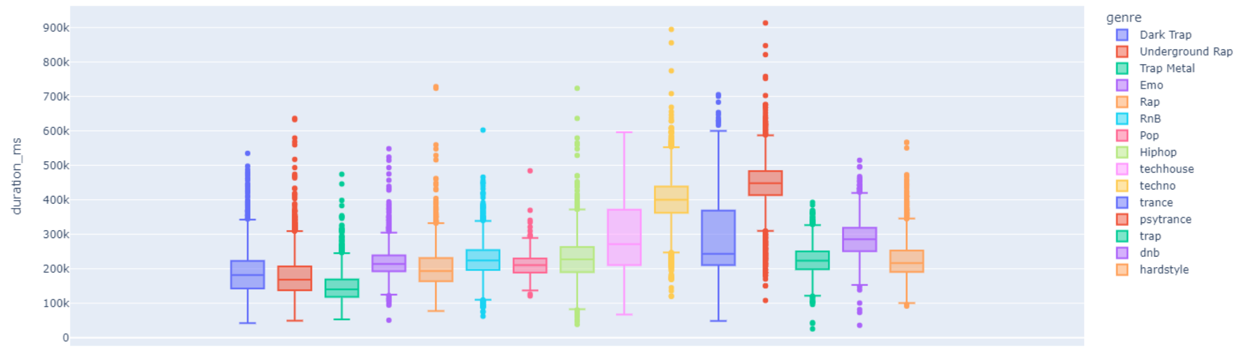


Figure 4.43 – Graph showing average time of song according to genre

Figure 4.43 shows the average time of song according to different genre. It is clear that all genres have their own time ranges. Genre of pytrance (red) is the longest and on the other hand trap music (Light Green) has the shortest timespan.

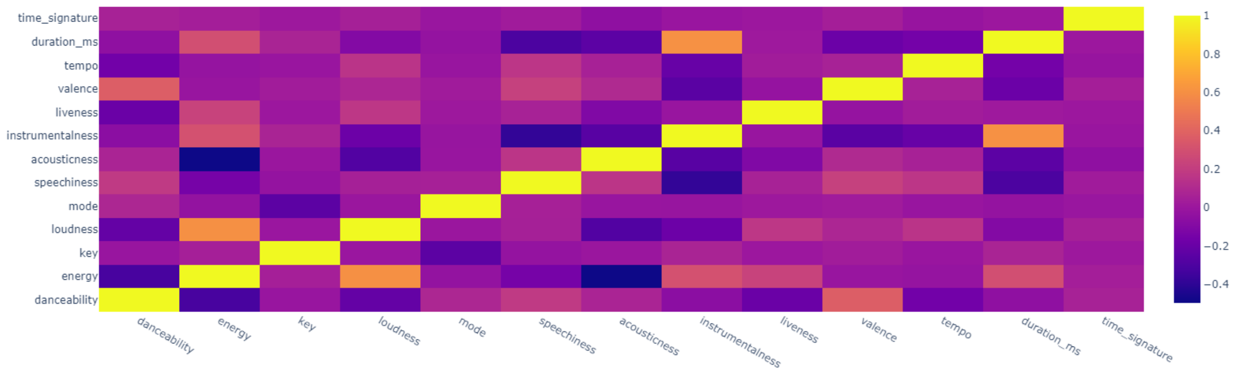
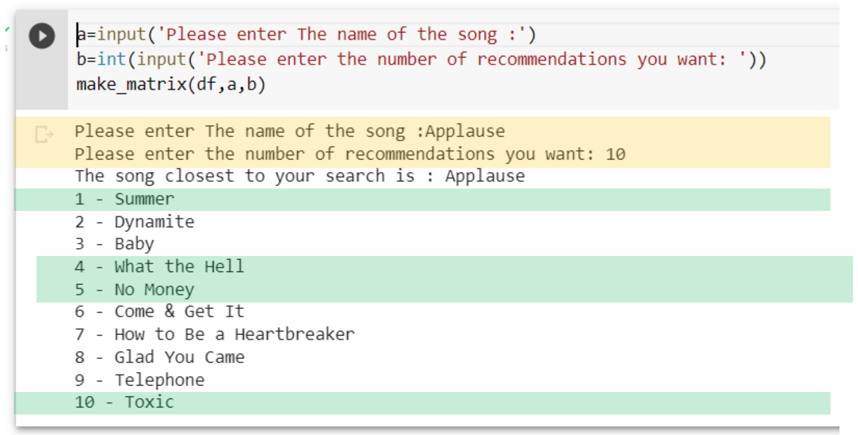


Figure 4.44 – HeatMap showing relationship between each feature

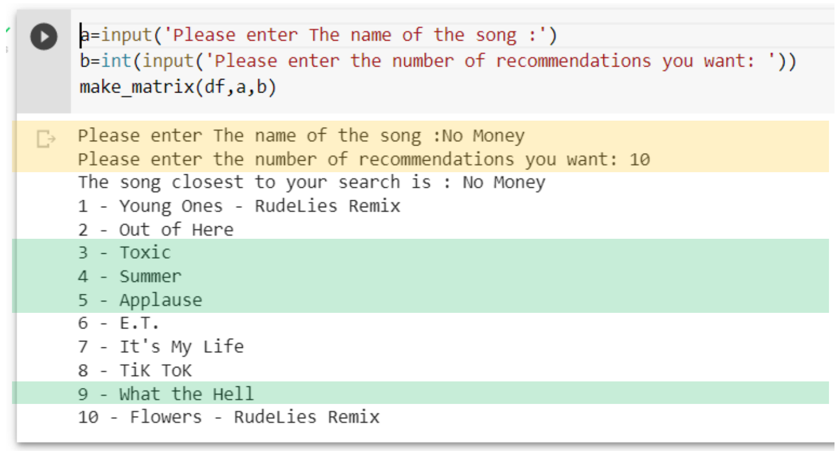
Figure 4.44 shows a heatmap of dataset, it is a type of plot that visualize the strength of relationships between the numerical features. Heat Map plots are used to understand which features are related to each other and the strength of this relationship. This Heat Map plot contains a number of numerical features, with each feature represented by columns. The rows represent the relationship between each pair of features. The values in the cells indicate the strength of the relationship, with positive values indicating a positive relationship (increase linearly together) negative values indicating a negative relationship (does not increase linearly with each other). For example, the intersection of ‘accousticness’ and ‘energy’ is in dark blue color (has a negative value), which means that as the ‘accousticness’ value increases, the ‘energy’ value does not increase with ‘accousticness’. Correlation heatmaps can be used to find potential relationships between features and to understand the strength of these relationships.



```
a=input('Please enter The name of the song :')
b=int(input('Please enter the number of recommendations you want: '))
make_matrix(df,a,b)
```

```
Please enter The name of the song :Applause
Please enter the number of recommendations you want: 10
The song closest to your search is : Applause
1 - Summer
2 - Dynamite
3 - Baby
4 - What the Hell
5 - No Money
6 - Come & Get It
7 - How to Be a Heartbreaker
8 - Glad You Came
9 - Telephone
10 - Toxic
```

Figure 4.45 – Recommendation based on ‘Applause’



```
a=input('Please enter The name of the song :')
b=int(input('Please enter the number of recommendations you want: '))
make_matrix(df,a,b)
```

```
Please enter The name of the song :No Money
Please enter the number of recommendations you want: 10
The song closest to your search is : No Money
1 - Young Ones - RudeLies Remix
2 - Out of Here
3 - Toxic
4 - Summer
5 - Applause
6 - E.T.
7 - It's My Life
8 - TiK ToK
9 - What the Hell
10 - Flowers - RudeLies Remix
```

Figure 4.46 – Recommendation based on ‘Telephone’

Figure 4.45 shows the recommendation based on the song ‘Applause’ while Figure 4.46 shows the recommendation based on song ‘Telephone’. As shown in the figure both songs appear in each other’s recommendation, and furthermore there are also other overlapped songs recommended. This means that these songs are closed to each other calculates using Euclidean Distance.

4.4.4 Comparison between Cosine Distance and Euclidean Distance

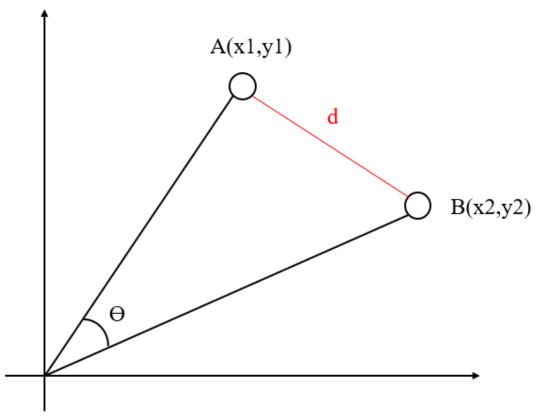


Figure 4.47 – Visual representation of Euclidean Distance (d) and cosine similarity (θ)

While cosine looks at the angle between vectors (thus not taking into regard their weight or magnitude), Euclidean Distance is similar to using a ruler to actually measure the distance. Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them whereas Euclidean Distance is simply the distance between two vectors *A* and *B* in some k-dimensional hyperspace.

CHAPTER 5

CONCLUSION

5.1 Concluding Remarks

K-Mean Clustering, Euclidean Distance and Cosine Similarity are methods that had been widely used in building recommender system. These are the popular algorithm for unsupervised learning, a machine learning method to analyze and cluster datasets. These algorithms identify hidden patterns or data groupings without the assistance of a human. It is the best option for exploratory data analysis because of its ability to find informational similarities and differences. In this study, all the three algorithms (K-Mean Clustering, Euclidean Distance and Cosine Similarity) are explored and separate music recommender system are built based on each algorithm using Google Colaboratory.

Data preparation, Mutual features calculation, Sound features analyzation, Perform K-mean clustering algorithm, Building recommendation engine, Input and Output are the seven essential phase for building K-Mean Clustering based Music Recommender System (MRS). Starting with data preparation, Spotify dataset is used because Spotify datasets is the largest songs dataset that provides a large variety of complete data (comes together with features and attributes) from 2010 to 2020. After that the system will perform mutual features calculation and sound features analyzation, collecting data of each song, comparing and contrasting among each other for clustering process later. The data will then be clustered by the algorithm, according to the similarities of each song, separate them by different group. Based on those clusters, the recommendation engine is built. The system will take user's input of song, search for cluster belongs to that song, and make recommendation of songs nearby or close to it within the same cluster. The result is then generated and show to user.

For Cosine Similarity algorithm, there are four main phases, which are Data preparation, Data normalization and One Hot Encoding, Building system using cosine similarity and lastly Input and output. For the data preparation it is the same as above where Spotify dataset is used. In the data normalization part, data will be normalized and applying one hot encoding to the 'genre' column. Most machine learning algorithm require inputs and outputs variables to be a number, or a numeric in value. With one-hot, we convert each categorical value into a new categorical column and assign a binary value of 1 or 0 to those columns. The column name of 'track_name' and 'artists_name' have also been normalized to 'artist' and 'name' for better understanding. The next phase is to build the recommender system. Cosine similarity measures the similarity between two vectors of an inner product space based on its cosine distance. It will then recommend songs with high similarity to users.

Lastly is the Euclidean Distance algorithm. There are a total of 4 main phases in this system. The first phase is the data preparation, followed by data preprocess and standardization, then is building recommender system and lastly input and output. In this system we used Heat Maps is to better visualize the volume of features within a dataset and assist in directing viewers towards areas on data visualizations that matter most. In the next phase of standardization, all the data of same category is set to the standard form, i.e., all the songs' duration are set to seconds in units. It is easier for the system to calculate similarity later on. Next is the building of Euclidean Distance based recommender system, Euclidean distance is a measure of the straight-line distance between two points based on its attributes. According to the distance measured, the similarity is known and recommendation can be made based on that.

5.2 Research Constraints and Challenges

Throughout the research, there are a few constraints and challenges that need to be faced. Firstly, the limited number of resources like journals and books that related to the topics of the research. Although there are a few leading companies equipped with a more mature recommender system, their recommendation systems are too complicated to do as research, the other online contributors are just building their personal music recommender system, therefore it is hard to find formal article about the topic. Thus, this become excruciatingly challenging to implement this research. Secondly, the limited amount of time to prepare the research report also become one of the challenges in this research. As the amount of effort to do research is huge, the time given must be use efficiently in order to complete on time.

5.3 Future Work

In the future, several improvement and implementation can be done for future works. First and foremost is the improvement on data interpretation, the system built and the algorithm studied is done individually, only one type of algorithm is used for analyzation of dataset in each system. Combining two or more algorithm into one recommender system will results in a more precise and accurate interpretation of data, and thus, a better and more complete music recommender system. The next improvement that can be made to these systems are to create interfaces for each system. Due to complexity of connecting Google Colaboratory lines of code to Graphical User Interface, it is hard and not enough time to do research within the time limit given. In the future, if interactive interfaces is created, it is no doubt that it will further enhance user experience in terms of operationality.

REFERENCES

Kevin Luk. (2019, Feb 3). Introduction to TWO approaches of Content-based Recommendation System. Towards Data Science. https://towardsdatascience.com/introduction-to-two-approaches-of-content-based-recommendation-system-fc797460c18c

Euge Inzaugarat. (2020, Feb 19). The ABC of building a content-based music recommender system. Towards Data Science. https://towardsdatascience.com/the-abc-of-building-a-music-recommender-system-part-i-230e99da9cad

Iterators. (2021, July 15). Collaborative Filtering In Recommender Systems: Learn All You Need To Know. Iterators. https://www.iteratorshq.com/blog/collaborative-filtering-in-recommender-systems/

Carlos Álvarez Angulo. (2010). Music Recommender System. [Unpublished master graduation thesis]. Polytechnic University of Milan.

Erion Çano, Maurizio Morisio. (2019, Jan 12). Hybrid Recommender Systems: A Systematic Literature Review. Arxiv. https://arxiv.org/abs/1901.03888

P. Darshna, "Music recommendation based on content and collaborative approach & reducing cold start problem," 2018 2nd International Conference on Inventive Systems and Control (ICISC), 2018, pp. 1033-1037, doi: 10.1109/ICISC.2018.8398959.

E. Shakirova, "Collaborative filtering for music recommender system," 2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2017, pp. 548-550, doi: 10.1109/EIConRus.2017.7910613.

Kevin Liao. (2018, Nov 11). Prototyping a Recommender System Step by Step Part 1: KNN Item-Based Collaborative Filtering. Towards Data Science. https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-1-knn-item-based-collaborative-filtering-637969614ea

Vuong Khuat. (2018, Dec). Music Recommendation Using Collaborative Filtering. https://portfolios.cs.earlham.edu/wp-content/uploads/2019/08/final_paper_draft_Vuong.pdf

Fkih, Fethi. (2021). Similarity Measures for Collaborative Filtering-based Recommender Systems: Review and Experimental Comparison. Journal of King Saud University - Computer and Information Sciences. 10.1016/j.jksuci.2021.09.014.

Selva Prabhakaran. (2018, October 22). Cosine Similarity – Understanding the math and how it works. https://www.machinelearningplus.com/nlp/cosine-similarity/#:~:text=The%20cosine%20similarity%20is%20advantageous,angle%2C%20higher%20the%20cosine%20similarity.

Anuus Soni. (2020, July 3). Advantages And Disadvantages of KNN. https://medium.com/@anuuz.soni/advantages-and-disadvantages-of-knn-ee06599b9336

Education Ecosystem (LEDU). (2018, September 13). Understanding K-means Clustering in Machine Learning. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

Pulkit Sharma. (2019, August 19). K-Means Clustering : Definition, Methods and Applications of K-Means Clustering Algorithm. https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

Byju. (2019, July 7). Euclidean Distance - Definition, Formula, Derivation. https://byjus.com/maths/euclidean-distance/

Richmond Alake. (2020, September 15). Understanding Cosine Similarity And Its Application. https://towardsdatascience.com/understanding-cosine-similarity-and-its-application-fd42f585296a