Music Recommendation Algorithm

Austin Kartheiser
akartheiser@hawk.iit.edu

Cheng Long Wu Yang
cwuyang@hawk.iit.edu

Paul Ojo
pojo@hawk.iit.edu

November 2022

1 Source Code

Our group made the whole source code in Google Colab, all implentation is set for Google Colab environment. For using the code, it will require the use of a Google Drive account for creating the folders and data imported from Kaggle. So, the work may not be reflected in the GitHub. To access our code: Source Code Github.

2 Introduction

Music is a vital part of our daily lives. It has a motivational effect which can encourage people to do some activities. It has also been shown that not all music has an equally strong motivational effect; while some music is very motivational for certain people, other music has no apparent effect on other habits. This is why there is a need for a less complex and more effective music recommendation algorithm based on different factors and actually recommending users songs they will like. A lot of users make their own playlists by liking songs or directly searching them, but we also need recommendation algorithms that make it easier for users to add or search for songs based on their listening habits. Currently, music service providers have generic (and popular) mood-based playlists that are the same for all users. I love listening to music; everywhere I go and anything I do, I'm listening to music. Matter of fact, I am listening to my favorite Spotify playlist as I'm typing this report. And this is the same with a lot of other people. Music, to some, makes doing anything seem less like a chore, especially when working on a big project or even smaller projects. It can brighten or sadden ones day; music can have a big influence on our moods through-out the day. This is why a good recommendation system is vital to the success of any music or video streaming platform.

Like with most modern tech applications and platforms, AI and data science

plays a major role in the development of various algorithms. Most platforms use many complex machine learning models to implement the recommendation system or engine on their platforms. Music and video platforms like Spotify, Apple Music/ITunes, and YouTube have their own algorithms for recommending songs and videos to users. Spotify, for instance, has a "Made For You" category that shows the songs recommended for a specific user. In the last decade, we have seen many music platforms implement AI-driven algorithms in their recommendation engines, and as we move ahead into the 2020s, an increasing share of music consumption and discovery will be mediated by AI-driven recommendation systems [1]. Spotify uses various machine learning models and algorithms to generate track and user representations. The track representation is made up of two components: content-based filtering which describes tracks by examining its contents, and collaborative filtering which describes tracks in their connection with other tracks on the platform by studying user-generated assets [1]. Spotify's AI-driven algorithm analyzes three main features when recommending songs: lyrical content and language, song features, and past listening habits. We don't need Spotify's or any other music streaming platform's AI-driven algorithm or machine learning model to build a good and efficient music recommendation system, and we are building this simple algorithm to show that.

3 Problem Description

Music listeners' taste in music varies a lot, and it can be difficult sometimes to find a song from the huge selection of recommended playlists on Spotify. The playlists may contain songs that do not fit the mood or theme of what the user requires. The lack of user-friendly music recommendation systems makes us believe that there is a need for a less complex and more efficient music recommendation algorithm that should be based on different factors like the "mood" or "theme", and actually recommending users songs they will like. Most people make their own playlists by liking songs or directly searching them, but not everyone can select or find all the songs that they love, due to the fact that they do not have a photographic memory to remember all of them in an instant. In some cases, you just forgot a song that you loved a long time ago, but the moment you hear it, you will remember the name of the song and you can add it to your playlist. As an avid Spotify user, I enjoy listening to songs Spotify recommends to me and it's overall recommendation system, but it could be improved to be more user-friendly.

According to Spotify's research team, users are overwhelmed by the choice of what to watch, buy, read, and listen to online, hence why recommendation systems and algorithms are necessary to help facilitate the decision process [2]. With this idea, Spotify's engine is focused on delivering suggestions based on a user's behavior on the Spotify application, i.e., past listening habits, searches,

created playlists and other songs similar users listen to. Although this is a great method of recommendation, it doesn't really give users the opportunity to listen to similar songs in their playlist based on the "theme." It will only recommend a generic playlist created by Spotify itself for all users, or playlists created by other users. For example, let's say you're on a nice date and you want to set the mood with a smooth romantic song, it would be nice to search for a song and then be recommended similar romantic songs from your created playlist and not a generic Spotify-generated playlist, wouldn't it? This is the main purpose of our recommendation algorithm.

Spotify is basically a recommendation engine on its own because almost everything the user sees, except for saved searches, is recommended to the user. New music, artists, podcasts, etc are all recommendations a user sees when they open the Spotify app. But is that really want users need? To an extent, yes, especially when the user just wants to listen to good music. But what if a user wants songs recommended to them from their already created playlist? This is one feature Spotify lacks. As stated in the Introduction above, most music platforms use AI-driven algorithms to develop their recommendation engines. We are building a recommendation algorithm that can provide custom playlists for individual users based on the mood or theme, and will include recommendation features like content-based and metadata filtering like most other platforms, but the main difference is that we will not be using any complex AI-driven machine learning models.

For this recommendation algorithm, we will be using a simpler machine learning algorithm, K-Means. This algorithm will be written in Python and we will import some of Python's libraries such as Pandas, NumPy, Seaborn. We will also be using Spotipy API, a Python client for Spotify Web API that makes it easier for developers to fetch data and query Spotify's song catalog. When we read the sample data, we will be checking for all the analysis with the target as 'popularity'. But before doing that, we will have to check for the Feature Correlation by considering some features of a song like the song's year, mode, key, explicitness, duration/s, valence, tempo, speechiness, loudness, liveness, instrumentalness, energy, danceability, and acousticness. Like mentioned earlier, our algorithm will be based on the content-based filtering approach. This approach allows us to recommend new music to users based on the type of songs (or genre) they listen to. In Figure 1 below, we can see the difference between collaborative filtering and content-based filtering.

3.1 Content-Based Filtering

Now, let's explain how the content-based filtering approach will work for our algorithm. The content-based filtering approach makes recommendations by using specific keywords and attributes assigned to objects in a platform and

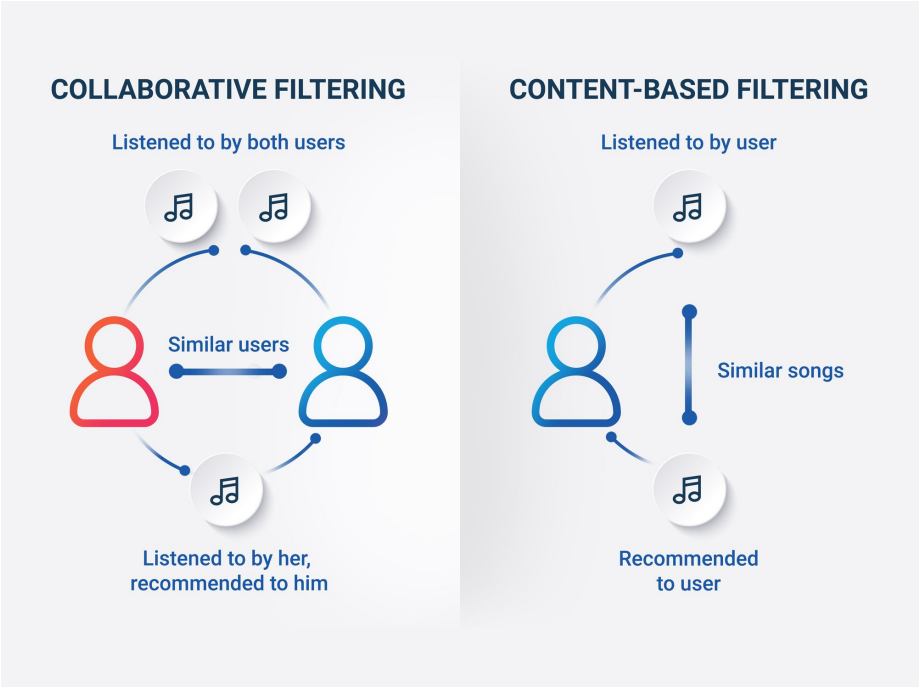

Figure 1: The difference between collaborative filtering and content-based filtering [3].

matching them to a platform [4]. For example, suppose an online marketplace is recommending laptop accessories to a user that just purchased a new laptop. The user's profile will indicate keywords such as the laptop manufacturer, make, model, and prior purchases may include laptop bags. Based on this information, the recommendation system may suggest different brands of laptop bags for the laptop with features like tablet and phone compartments. In this example, the user is expecting recommendations for accessories for his newly-purchased laptop, but the laptop bag with phone/tablet compartment features may not be something they expect, but will probably appreciate. This is how our algorithm will recommend music to users.

The system will aim to, first of all, recommend new songs to users based on the songs on the playlist the user created themselves, not playlists created by other users, and secondly, recommend songs based on a selected song that will set the theme for the type of songs that should be recommended. This is what K-Means clustering does. The K-Means algorithm will be explained in detail later in this report. One major benefit of using this content-based filtering approach is that no data from other users is required for the system to make recommendations. Once a user searches for a particular song, the content-based

filtering will begin by making relevant recommendations [4] to that song. This filtering approach is highly tailored to the user's interests. Which means the user will only be recommended songs they will be interested in or songs they will like. This is a vital feature in building an efficient music recommendation system or engine. Collaborative filtering has a "cold start" problem when a new platform has few users and there's little-to-no user connections. In this case, it will be hard to make good recommendations because the collaborative filtering approach depends on having enough users that interact with each other to make recommendations. Another benefit of content-based filtering systems is that they are generally easier to create and relatively straightforward compared to collaborative filtering systems.

In the next section, we will do some data analysis on the Spotify data-set we will be using for this project, and perform some data inspection and cleaning.

4 Data Preprocessing and Analysis

Data analysis is the backbone of any data-related project. Before working on a data-related project, we must prepare and do proper analysis to make sure all data-set that will be used for the project are free of errors and inconsistencies. This is what is referred to as the data pre-processing stage of any data-related project. When using data sets to train machine learning models, there's a common phrase, "garbage in, garbage out," which basically means removing bad or dirty data that won't actually be relevant to the overall analysis or training, and may not allow the model to be trained properly. We have to preprocess our data so that the model can perform better.

4.1 Data Inspection

Now, let's explain the dataset we will be using for our recommendation algorithm. We are using a sample Spotify dataset we acquired from Kaggle. The dataset contains 19 columns, but we will be using 18 columns for this algorithm. Each column's value is described below:

acousticness: This is the acoustic measure of a track on a scale of 0.0 to 0.1.

danceability: This describes how suitable a track is for dancing based on a combination of different musical elements including rhythm stability, beat strength, and overall tempo, measured on a scale of 0.0 to 1.0.

duration_ms: This is the duration of the track, measured in milliseconds.

energy: This represents a perceptual measure of intensity and activity of the track. This can be described as whatever keeps the listener engaged and listen-

ing. With louder music, musical energy is easy to identify. We notice the energy more as the drums get busier and play louder, and as the artist(s) sings higher.

instrulmentalness: This predicts whether a track contains vocals or not.

loudness: The loudness of a track in decibels(dB). Influenced by factors such as frequency, timbre, and the environment in which the track was recorded.

popularity: The measure of the track's popularity on Spotify.

speechiness: This detects the presence of an audience in a track. If the speechiness of a song is above 0.66, it is probably made of spoken words. A score between 0.33 and 0.66 is a song that may contain both music and words, and a score below 0.33 means the song does not have any speech.

explicit: The explicitness of lyrics used in a track. Only songs with a value of 1 are tagged as being explicit.

key: This is the key the track is recorded in. Integers will map to pitches using the standard Pitch Class notation. Ranges from 1 to 12.

valence: This describes the "positiveness" of a track, on a scale of 0.0 to 0.1. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

release_date: The release date of a track. Different from Year in that it includes a month and day, if one exists.

tempo: This is the how fast a track is, measured in beats per minute (BPM).

mode: This indicates the modality (major or minor) of a track.

name: The name of the track/song.

artists: The name(s) of all the artists in a track.

liveness: Ranges from 0 to 1. Higher values represent an increased probability that the track was recorded live.

genre: The track genre. Spotify currently has over 5,000 distinct genres.

Each column will contribute to building the algorithm, but the columns we are most interested in are the ***genre*** and ***popularity*** columns.

5 Data Visualization

Let's visualize our data so far. After importing the data, we will need to check for the feature correlation of each column using yellowbrick package. We can see the graph of feature correlation of each column in Figure 2 below:

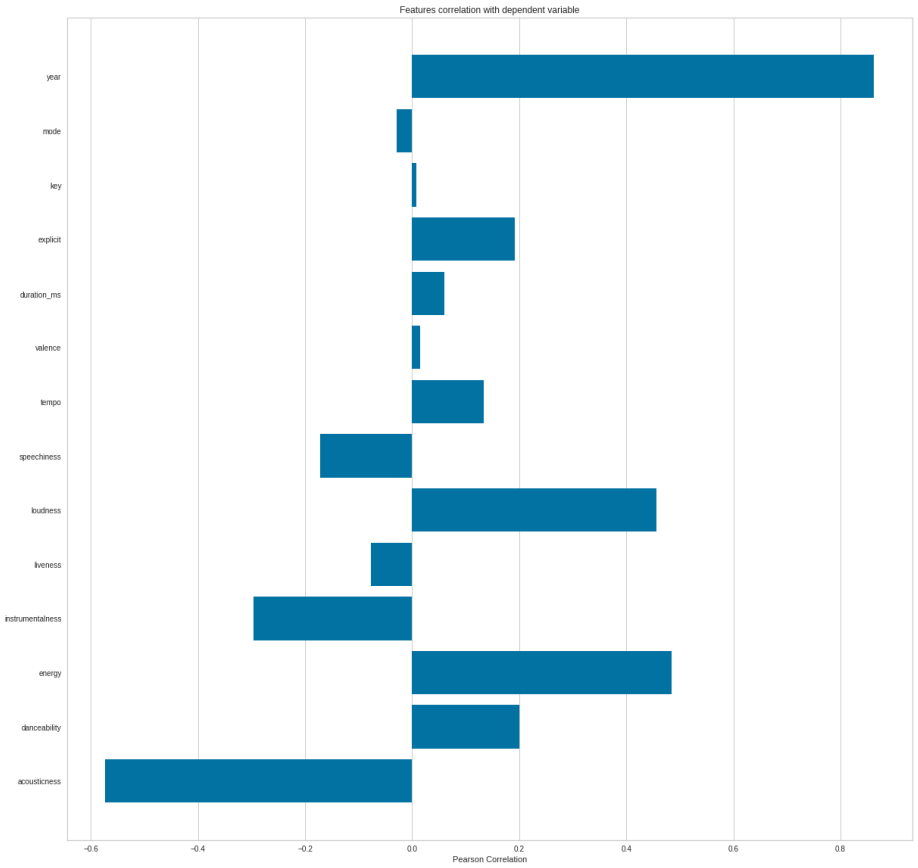

Figure 2: Feature Correlation

We can also see how the sound of music has changed from the 1920s to 2020s below in figures 3 and 4 below. The data is grouped by year:

Our code also allows us to visualize different songs with the audio features for different genres. This information can be used to compare the different types of genres in out Spotify dataset, and understand the uniqueness of their sound. We can see this graph, grouped by genres, in Figure 5 below:

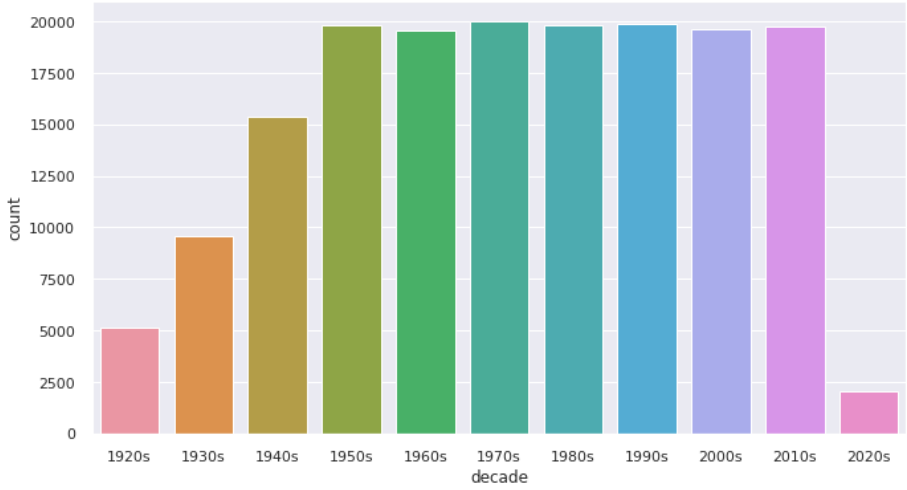

Figure 3: Change of music over time.

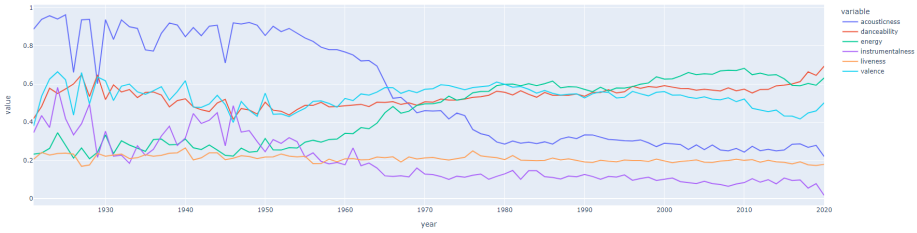

Figure 4: Change of music over time pt. 2.

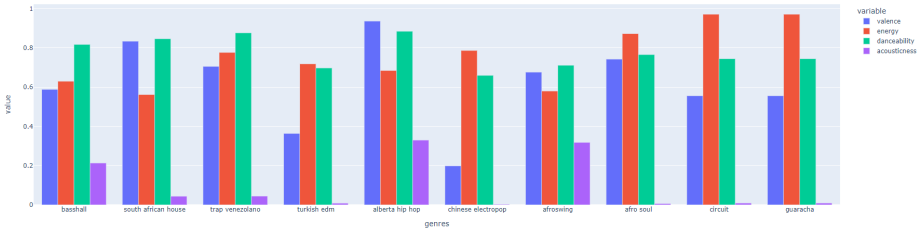

Figure 5: Songs grouped by genres

6 K-Means Algorithm for Recommendation

K-Means clustering is one of the simplest and popular unsupervised learning algorithms. In the K-Means algorithm, a "cluster" refers to a collection of data points with certain similar attributes aggregated together. The goal of using K-Means clustering is to organize the songs in clusters, where similar songs are

put together [2].

6.1 Clustering with K-Means

After using the K-Means clustering to cluster the songs and genres, we can see how the dataset is divided into ten clusters based on the numerical audio features of each genre.

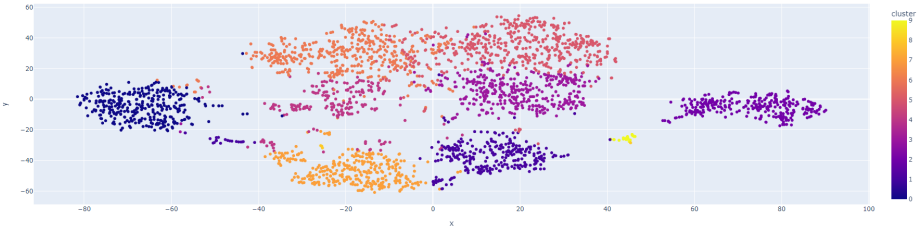

Figure 6: Clustering genres with K-Means.

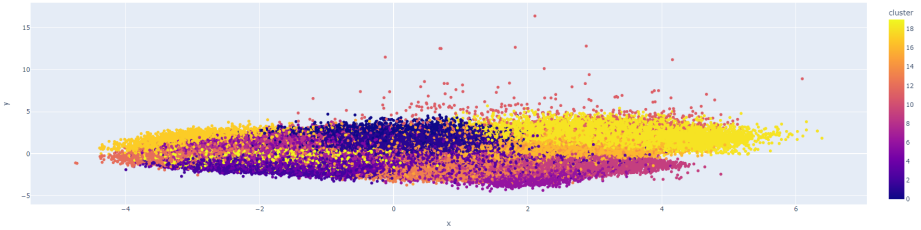

Figure 7: Clustering songs with K-Means.

7 Replication

8 Results

So our program to predict songs, we have created a small front-end user interface as shown in the following: The — is shown when an input is needed from the user.

- Step 1.- How many songs would you like to enter? —
  - Success: To Step 2:
  - Unsuccessfully: Invalid Entry: Please enter a positive integer
- Step 2.- How many recommendations would you like to receive? —
  - Success: To Step 3:

- Unsuccessfully: Invalid Entry: Please enter a positive integer

- Step 3:- Enter the name of song : —
- Step 4:- Enter artist of song : —
- 'name': 'xxxxxxx', 'year': xxxx, 'artists': "['xxxxxxxx']"

Here is an actual example:

- Step 1.- How many songs would you like to enter?
  Can't Help Falling in Love
  - Unsuccessfully: Invalid Entry: Please enter a positive integer
- Step 2.- How many songs would you like to enter?
  1
- Step 3.- How many recommendations would you like to receive? help
  - Unsuccessfully: Invalid Entry: Please enter a positive integer
- Step 4:- Enter the name of song 1:
  Can't Help Falling in Love
- Step 5:- Enter artist 1 of song 1:
  1961

8.1 Output

1. {'name': 'The Christmas Song (Merry Christmas To You)', 'year': 1962, 'artists': "['Nat King Cole']"}
2. {'name': 'Everything I Own', 'year': 1972, 'artists': "['Bread']"}
3. {'name': 'Yesterday - Remastered 2009', 'year': 1965, 'artists': "['The Beatles']"}
4. {'name': 'The Last Waltz', 'year': 1967, 'artists': "['Engelbert Humperdinck']"}
5. {'name': 'Homeward Bound', 'year': 1966, 'artists': "['Simon Garfunkel']"}
6. {'name': 'Helpless', 'year': 1970, 'artists': "['Crosby, Stills, Nash Young']"}
7. {'name': "I Won't Last A Day Without You", 'year': 1972, 'artists': "['Carpenters']"}
8. {'name': 'Just My Imagination (Running Away With Me)', 'year': 1971, 'artists': "['The Temptations']"}
9. {'name': 'Blue Christmas', 'year': 1957, 'artists': "['Elvis Presley']"}

10. {'name': 'Dream A Little Dream Of Me - Single Version', 'year': 1957, 'artists': "['Ella Fitzgerald', 'Louis Armstrong']"

11. {'name': "What A Diff'rence A Day Made", 'year': 1959, 'artists': "['Dinah Washington']"}

12. {'name': 'Beth', 'year': 1976, 'artists': "['KISS']"}

13. {'name': 'Going to California - Remaster', 'year': 1971, 'artists': "['Led Zeppelin']"}

14. {'name': 'Everything I Own', 'year': 1973, 'artists': "['Bread']"}

15. {'name': "Can't Find My Way Home", 'year': 1969, 'artists': "['Blind Faith']"}

16. {'name': 'Christmas Time', 'year': 1963, 'artists': "['The Platters']"}

17. {'name': 'The First Noel - Remastered 1999', 'year': 1957, 'artists': "['Frank Sinatra']"}

18. {'name': 'Black Coffee', 'year': 1955, 'artists': "['Sarah Vaughan']"}

19. {'name': 'O Little Town of Bethlehem', 'year': 1957, 'artists': "['Elvis Presley']"}

20. {'name': "You've Lost That Lovin' Feelin' - Single Version", 'year': 1965, 'artists': "['The Righteous Brothers']"}

21. {'name': 'Pocketful of Rainbows', 'year': 1960, 'artists': "['Elvis Presley']"}

22. {'name': 'Crying', 'year': 1962, 'artists': "['Roy Orbison']"}

23. {'name': "She's Leaving Home - Remastered 2009", 'year': 1967, 'artists': "['The Beatles']"}

24. {'name': 'Donna - Single Version', 'year': 1959, 'artists': "['Ritchie Valens']"}

25. {'name': 'The Sound of Silence - Acoustic Version', 'year': 1964, 'artists': "['Simon Garfunkel']"}

26. {'name': 'O Tannenbaum', 'year': 1962, 'artists': "['Nat King Cole']"}

27. {'name': 'Bleecker Street', 'year': 1964, 'artists': "['Simon Garfunkel']"}

28. {'name': "I'll Be Seeing You", 'year': 1957, 'artists': "['Billie Holiday']"}

29. {'name': 'Imagine - Remastered 2010', 'year': 1971, 'artists': "['John Lennon']"}

30. {'name': 'Moon River', 'year': 1964, 'artists': "['Louis Armstrong']"}

31. {'name': 'Scarborough Fair / Canticle', 'year': 1966, 'artists': "['Simon Garfunkel']"}

32. {'name': 'Open Arms', 'year': 1981, 'artists': "['Journey']"}

33. {'name': 'Only You (And You Alone)', 'year': 1986, 'artists': "['The Platters']"}

34. {'name': 'I Only Have Eyes for You', 'year': 1959, 'artists': "['The Flamingos']"}

35. {'name': 'River', 'year': 1971, 'artists': "['Joni Mitchell']"}

36. {'name': 'The Sound of Silence', 'year': 1965, 'artists': "['Paul Simon']"}

37. {'name': 'Because - Remastered 2009', 'year': 1969, 'artists': "['The Beatles']"}

38. {'name': 'April In Paris', 'year': 1958, 'artists': "['Billie Holiday']"}

39. {'name': 'Turandot / Act 3: "Nessun dorma!"', 'year': 1973, 'artists': "['Giacomo Puccini', 'Luciano Pavarotti', 'John Alldis Choir', 'Wandsworth School Boys Choir', 'London Philharmonic Orchestra', 'Zubin Mehta']"}

40. {'name': 'Your Song', 'year': 1970, 'artists': "['Elton John']"}

41. {'name': 'Hark! The Herald Angels Sing', 'year': 1962, 'artists': "['Nat King Cole']"}

42. {'name': 'In The Wee Small Hours Of The Morning - Remastered 1998', 'year': 1955, 'artists': "['Frank Sinatra']"}

43. {'name': 'It Had to Be You', 'year': 1959, 'artists': "['Ray Charles']"}

44. {'name': 'Moon River (From "Breakfast at Tiffanýs")', 'year': 1962, 'artists': "['Andy Williams']"}

45. {'name': "I'll Look Around", 'year': 1956, 'artists': "['Billie Holiday']"}

46. {'name': 'Tennessee Waltz', 'year': 1960, 'artists': "['Patti Page']"}

47. {'name': "Baby, It's Cold Outside", 'year': 1959, 'artists': "['Dean Martin']"}

48. {'name': 'Return To Me (Ritorna-Me) - 1997 Remaster', 'year': 1958, 'artists': "['Dean Martin']"}

49. {'name': 'Dream A Little Dream Of Me - With Introduction', 'year': 1968, 'artists': "['The Mamas The Papas']"}

50. {'name': 'Baby', 'year': 1979, 'artists': "['Donnie Joe Emerson']"}

9 Conclusion

In general, in order to measure the accuracy, unfortunately, we cannot because that is someone's preference. But according to our perspective, we get some similarities between them according to their genre and year. Overall, we tried some inputs according to our taste and the majority of the music recommended was interesting. But as I said, it depends on the user preferences. All three, agree that making this project has made us gain more knowledge as well in the research aspect of it and implementation methods.

10 References

[1] D. Pastukhov. "Inside Spotify's Recommender System: A Complete Guide to Spotify Recommendation Algorithms." Music Tomorrow. https://www.music-tomorrow.com/blog/how-spotify-recommendation-system-works-a-complete-guide-2022 (accessed Nov. 17, 2022).

[2] R. Duarte. "K-Means Clustering Using Spotify Data." Medium. https://medium.datadriveninvestor.com/k-means-clustering-using-spotify-data-45435a7cd32a (accessed Nov. 28, 2022).

[3] S. Ripenko and N. Hasuik. " Music recommendation system: ElifTech." Eliftech. https://www.eliftech.com/insights/all-you-need-to-know-about-a-music-recommendation-system-with-a-step-by-step-guide-to-creating-it/ (accessed Nov. 29, 2022).

[4] The Upwork Team. "What Content-Based Filtering is and Why You Should Use It." Upwork. https://www.upwork.com/resources/what-is-content-based-filtering (accessed Nov. 20, 2022).