CREATING SPOTIFY PLAYLISTS BASED ON CLUSTERED SONGS SHARING SIMILAR MUSICAL ATTRIBUTES

Jake Darby

DATA 470: Data Science Capstone

December 15, 2020

**ABSTRACT:**

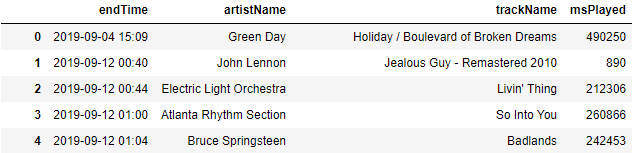
Music listening has grown increasingly digital over the past few decades, and popular streaming services such as Spotify, Apple Music, and Soundcloud have opened the door to countless users to stream their music digitally. Spotify, among these, allows users to create their own playlists consisting of songs they personally choose and enjoy. This process of creating playlists may become tedious for a user, as it can be difficult to find similar sounding songs for their playlists, and thus becomes more time consuming the longer the playlist is desired to be. Using a data science approach, I have created an algorithm that utilizes a user’s streaming history, which includes all of the songs a user has ever played, and groups the songs that share similar musical attributes together. I achieve this grouping of songs using KMeans clustering, where clusters are formed from the grouping of similar attributes within the songs of a user’s streaming history, namely acousticness, tempo, energy, instrumentalness, etc. The clusters created by KMeans include songs sharing similar attributes within their respective cluster. I then can transfer the songs from each cluster to Spotify through the Spotify Web API. In Spotify, these clusters of songs will then make up multiple newly made playlists consisting of similar sounding songs, and, because they are songs that the user has listened to themselves, they are more likely to be satisfactory to the listener. To demonstrate this, I have utilized my own streaming history for my project, clustering together similar songs that I have listened to, and creating 9 resulting playlists on my own Spotify account that each consist of similar sounding songs.

**INTRODUCTION:**

Using the online music streaming service Spotify, users are able to search for practically any song, artist, album, or genre that they desire within the service’s expansive collection of streamable music. After listening to more and more music, Spotify is able to determine the unique music preferences of a user, which in turn allows Spotify to create daily playlists for users titled Daily Mixes. Within these daily mixes are songs commonly listened to by the user, along with new songs that Spotify has determined are similar to the songs listened to by the user. Daily Mixes, however, are only about 50 songs in size, and tend to present users with songs they already are listening to very often. As a result, the Daily Mixes are useful insofar as the user does not get tired of the provided songs within the playlists. To address this problem of having both overplayed and limited songs from Spotify’s Daily Mixes, I have created a process that utilizes all of the songs a user has listened to on Spotify to help create playlists of similar songs within this entire history. These playlists will not be limited and will only vary in length based on the category of songs that is represented by each playlist.

**DATA SOURCES:**

The data that I used within my project is my personal Spotify streaming history, which I requested directly from Spotify. The file containing this streaming history consisted of 17,656 rows when expressed in a dataframe, or table, that each represent a song that I have at some point listened to. I am able to view this data in a dataframe using the Pandas library in Python. The dataframe also consisted of 4 columns, including the date which the song was played, the artist, the title of the song, and the amount of time that I spent listening to the song in that instance. Shown in Figure 1 below, the first five songs of this dataframe are listed, along with the columns that correspond with each song’s row.



**FIGURE 1: The first 5 rows of the dataframe showing my Spotify streaming history.**

**PREVIOUS RESEARCH:**

There are numerous instances of previous research that share very similar aspects of musical classification as my own project that have occurred in the past. One example of this use of musical classification appears in Google’s patent for a “Music Mapper”, which was first applied for in 2004. This “Music Mapper” is an algorithm that essentially was intended to strategically “map out” songs through assigned coordinate vectors, creating a “music similarity table” of sorts. By plotting out songs by their attributes in this way, the “Music Mapper” can be used to “[infer] similarity between various pieces of music”. The patent application describes this process, stating:

[The] Music Mapper applies a recursive embedding process to embed each of the graph’s music entries into a multi-dimensional space. This recursive embedding process also embeds new music items added to the music similarity graph without reembedding existing entries so long a convergent embedding solution is achieved. Given this embedding, coordinate vectors are then computed for each of the embedded musical items. The similarity between any two musical items is then determined as either a function of the distance between the two corresponding vectors.[[1]](#footnote-1)

Google’s “Music Mapper” can therefore accurately determine similarity between songs, and subsequentially group like songs together based on their proximity to each other on the graph. The patent further explains that the similarities between songs can be used for “constructing music playlists from one or more random or user selected seed songs, or for clustering similar songs, albums, or artists”. Google’s “Music Mapper” therefore uses a very similar strategy for grouping together songs from a user’s library as I have within my own project. A drawback of this patent is that it is not practical for most users, as users do not have the available resources to plot every song. However, using a very large selection of my own personal streaming history of songs and artists that I have, over time, explored and enjoyed on Spotify, I can take advatange of a large amount of songs to test cluster together similar songs based on their common musical attributes to then create playlists.

In addition to Google’s “Music Mapper”, another 2018 study by four students at ICT Research Institute titled “Automated Mood Based Music Playlist Generation By Clustering The Audio Features” aimed to achieve another grouping of like music in a process similar to that of Google’s patent. This ICT study stated its approach to music clustering as such:

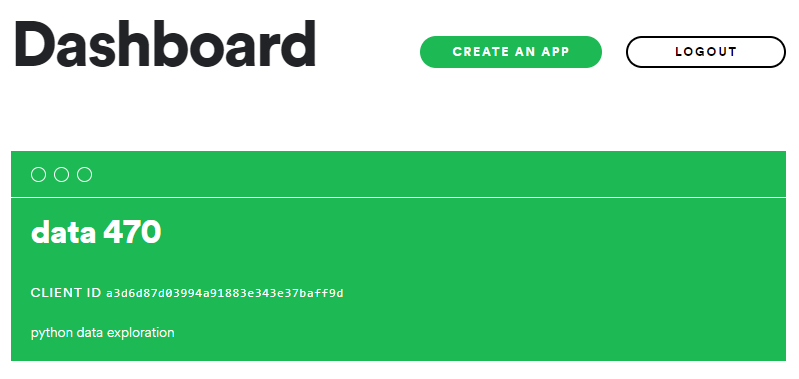
One of the main goals is to generate personalized playlists automatically for each user. Beyond that, an appropriate switching among these playlists to play the tracks based on the current mood of the user would certainly lead to the development of more advanced and personalized music player apps. In this paper, a data scientific approach is provided to model the music moods which are created by clustering the tracks extracted from users' listening. Each Cluster consists of music tracks with similar audio features existing in the user's listening history.[[2]](#footnote-2)

Here, the study describes its grouping of similar songs through clusters by comparing similar attributes commonly shared by songs. Certain songs may convey particular moods or emotions that listeners can resonate with; for example, a “sad” song might have a slower tempo, lower energy, and possibly a minor tonality in its key, while a “happy” song might have an upbeat tempo, higher energy, and a bright, major tonality in its key. By grouping the songs of a user’s music library into clusters based on the emotions represented by the songs’ attributes, this study was able to recommend songs sharing the same “mood” as the song that the user is currently listening to. Further, playlists based on the emotions or moods discovered within the clustering of a user’s library could also be created and stored for the user’s convenience for the future. A strength of this study is obviously the benefit for users to find similar songs to that which they are already actively listening to. However, a weakness of this study is that “emotions” of songs may be subjective, and thus a difficulty that could arise is either inconsistent or non-cohesive playlists. As a whole, this study is very similar to mine, insofar as it takes advantage of Spotify’s user streaming history and is aimed at creating playlists from clustered songs that share similar attributes.

A third study conducted by two students at California State Polytechnic University in 2018 titled “Automated Playlist Generation from Personal Music Libraries” partakes in very similar strategies used both by the aforementioned studies as well as my own project. This third study aims to use the songs stored in a user’s music library and sort them into clusters by similar song attributes that can then be placed into specific user playlists.[[3]](#footnote-3) This study seems to parallel my own project the most out of the three, and thus I will be able to explain this study in more detail as I describe my own.

**PROBLEM FORMULATION:**

When considering Spotify’s Daily Mixes having both overplayed and limited songs, I chose to create a way to create playlists that, instead of having a small amount of very familiar songs, instead would utilize all of the songs I have listened to personally on Spotify. In order to be able to group together any of the songs in my dataframe, I needed to know more about each song other than the date, artist, title, and amount of time I listened to it. To retrieve more descriptive information about the songs in my dataframe, I needed to use the Spotify Web API, which would allow me to retrieve the musical attributes my data exploration. The first step to take was to create a profile for my project on the Spotify Developer page, shown below in Figure 2, which would become the first link between my personal account on Spotify and the project I was to develop using a Python. This Spotify Developer page provides two unique codes for my project, namely the Client ID and the Client Secret ID, which, when combined, allowed for the use of the Spotify Web API within my notebook; these can be seen below in Figure 2. To make using the Spotify Web API easier, I also utilized the SpotiPy library, a Python library that simplifies processes used by the Spotify Web API.



**FIGURE 2: The Spotify Developer Page and my project’s application.**

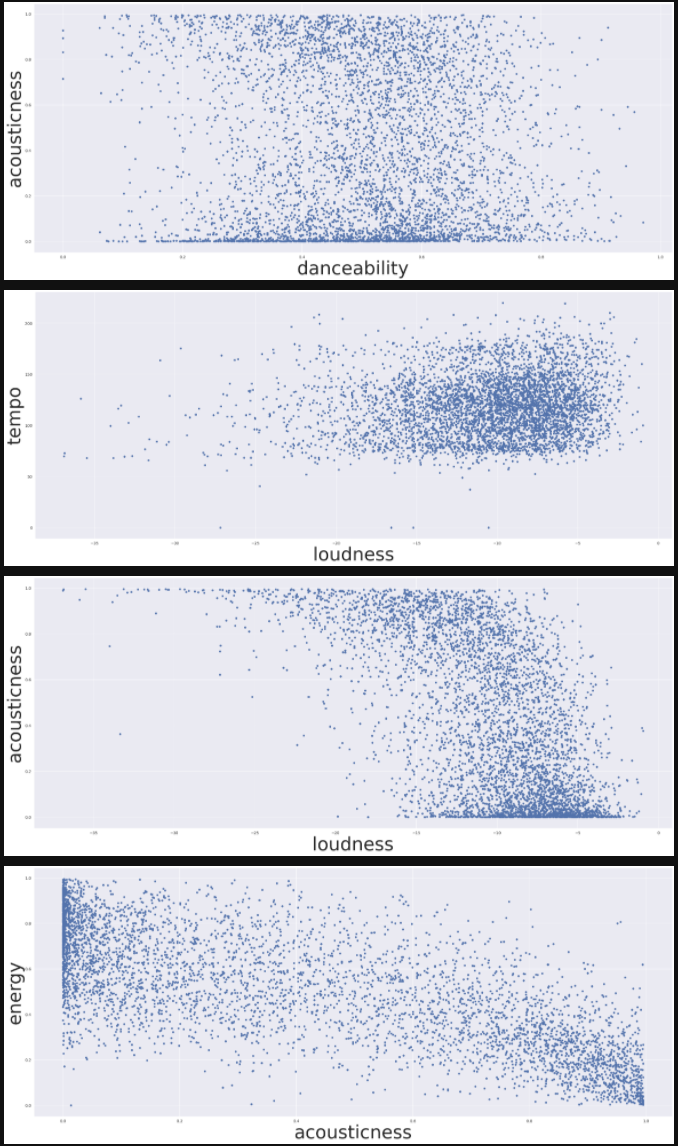
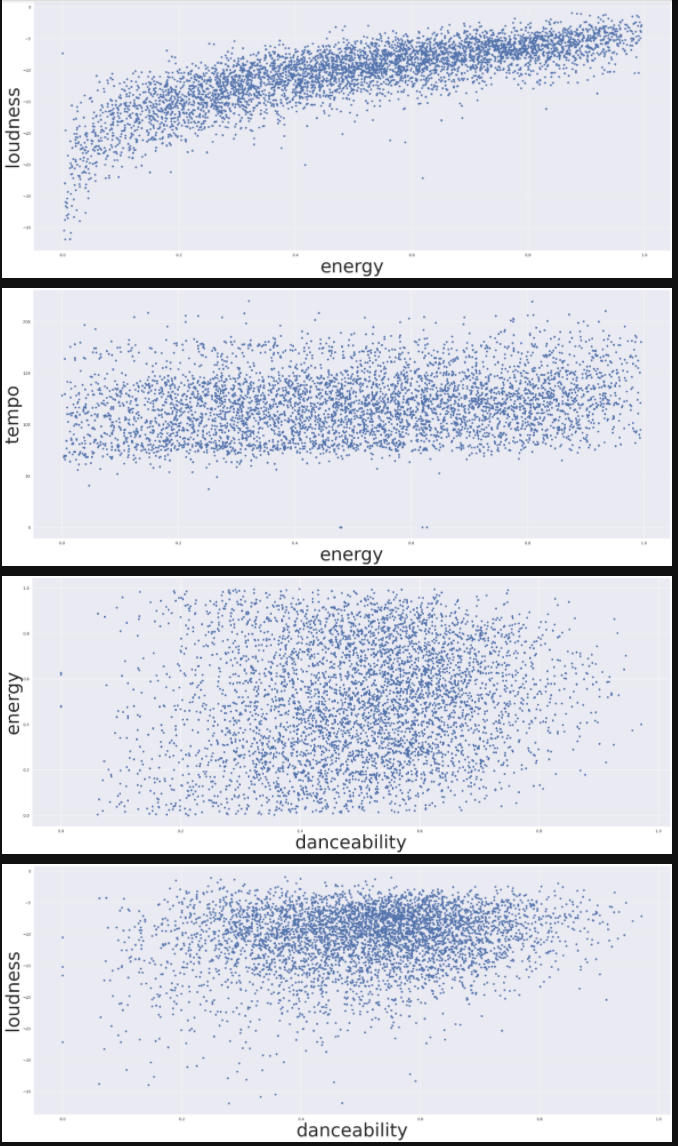
**METHODS AND RESULTS:**

Now, by using the Spotify Web API, I am able to create a method that allows me to retrieve the specific song “IDs” that Spotify has uniquely for each song it has available for streaming. With the song ID, I am also able to retrieve all of the attributes that Spotify also stores for each song. These attributes, listed in depth on the Spotify Developer webpage[[4]](#footnote-4), are included in Figure 4 and are now available for my use in my project. With these attributes, I am able to start exploration the correlation between the songs in my own dataframe.

|  |  |
| --- | --- |
| Attribute | Description |
| Acousticness | A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic. |
| Danceability | Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable. |
| Energy | Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. |
| Instrumentalness | Predicts whether a track contains no vocals. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content, where anything over 0.5 can be considered instrumental. |
| Liveness | Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. |
| Loudness | The overall loudness of a track in decibels (dB). Values typical range between -60 and 0 db. |
| Popularity | A measure of an artist’s popularity on Spotify, ranging from a scale from 0 to 100, from least to most popular. |
| Speechiness | Detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. |
| Tempo | The overall estimated tempo of a track in beats per minute (BPM). |

**FIGURE 3: List of Spotify song attributes included in my dataframe.**

Looking at the attributes I now have in my dataframe, I want to find out if there exist any significant correlations between any of them across all of my songs, so that I can start to learn how my song data were to be split up later on. Using the Seaborns library on Python, I am able to create scatterplots that plot each attribute against a different one in the dataframe. To achieve this, I created a subset of my data, including only the attributes that I wished to visualize. In Figure 4 below, it is evident which attributes are correlated and which are not.



**FIGURE 5: Scatterplots of song attributes.**

Namely, there is a positive correlation between the plots “loudness and energy”, “tempo and energy”, and “loudness and danceability”, while there is a negative correlation between the plots “acousticness and loudness” and “energy and acousticness”.With these correlations in mind, I am able to expect how the songs in my data are to be clustered together. In addition to visualizing the relationship between each attribute, I want to know how each attribute is distributed within the data. To visualize this distribution, I utilize the histogram function found in the Pandas library. This function allows me to plot out and visualize how frequently each attribute is distributed within each quantile, or equally sized subgroups, of the data. The histograms represented by this function are shown below in Figure 6.

Chart, bar chart

Description automatically generated

**FIGURE 6: Histograms showing the distribution of each attribute of the dataframe.**

It is evident from these histograms that there are normal distributions for the attributes popularity, danceability, energy, and tempo, while there are non-normal distributions for the attributes acousticness, instrumentalness, liveness, loudness, and speechiness. Now that I understand how my data are both related and distributed, I can begin to cluster my data by their similarities. To set all of my data at a scale from 0 to 1, I use the package MinMaxScalar included in the SKLearn Python library. Because clusters are determined by how far apart points are between each other when plotted, it helps to scale the attributes in order to create an evenly balanced clustering, where no attribute becomes more important than another. Now that my data are scaled, I need to determine how many clusters would be optimal for my data. To achieve this, Chart, line chart

Description automatically generatedcan utilize the “elbow method”, where “If the line chart resembles an arm, then the “elbow” (the point of inflection on the curve) is a good indication that the underlying model fits best at that point.”[[5]](#footnote-5). After creating a curve to determine the number of clusters I should divide my data into, shown in Figure 7, it is evident that I should use 4 clusters for my data.

**FIGURE 7: Elbow Method for my data.**

**CONCLUSION:**

**BIBLIOGRAPHY**

Bakhshizadeh, Mahta, Ali Moeini, Mina Latifi, and Maryam Tayefeh Mahmoudi. “Automated Mood Based Music Playlist Generation By Clustering The Audio Features.” Automated Mood Based Music Playlist Generation By Clustering The Audio Features - IEEE Conference Publication, October 24, 2018. https://ieeexplore.ieee.org/document/8965190.

Lin, Diana, and Sampath Jayaranthna. “Automated Playlist Generation from Personal Music Libraries.” Automated Playlist Generation from Personal Music Libraries - IEEE Conference Publication, July 6, 2018. https://ieeexplore.ieee.org/abstract/document/8424710.

Platt, John, Erin Renshaw, Max Chickering, and Cormac Herley. “US7777125B2 - Constructing a Table of Music Similarity Vectors from a Music Similarity Graph.” Google Patents. Google, November 19, 2004. https://patents.google.com/patent/US7777125B2/en?q=music+mapper.

1. Platt, John, Erin Renshaw, Max Chickering, and Cormac Herley. “US7777125B2 - Constructing a Table of Music Similarity Vectors from a Music Similarity Graph.” Google Patents. Google, November 19, 2004. [↑](#footnote-ref-1)
2. Bakhshizadeh, Mahta, Ali Moeini, Mina Latifi, and Maryam Tayefeh Mahmoudi. “Automated Mood Based Music Playlist Generation By Clustering The Audio Features.” Automated Mood Based Music Playlist Generation By Clustering The Audio Features - IEEE [↑](#footnote-ref-2)
3. Lin, Diana, and Sampath Jayaranthna. “Automated Playlist Generation from Personal Music Libraries.” Automated Playlist Generation from Personal Music Libraries - IEEE Conference Publication, July 6, 2018. [↑](#footnote-ref-3)
4. Spotify Dev [↑](#footnote-ref-4)
5. https://www.scikit-yb.org/en/latest/api/cluster/elbow.html [↑](#footnote-ref-5)