Spotify EDA and Recommendation System Using Pandas and Streamlit

CS 648: Big Data Tools and Methods

Spring 2022

Prof. Roger Whitney

Team Member:

Rutuja Medhekar
RED ID: 826141245

Introduction

Spotify is worlds popular audio streaming platform which provides over 55 million tracks and over 170 million users. Everyone loves listening to music on Spotify. Even I love listening music on Spotify and hence decided to perform various analysis on various audio features of this dataset.

For this project, I had taken Dataset from Kaggle

Spotify Dataset contains information regarding Audio Features of 160000+ songs released in between 1921 and 2020.

In this project, I was interested in analyzing the Spotify dataset by doing Exploratory Data Analysis (EDA) and building recommendation system based on the features selected by doing the analysis.

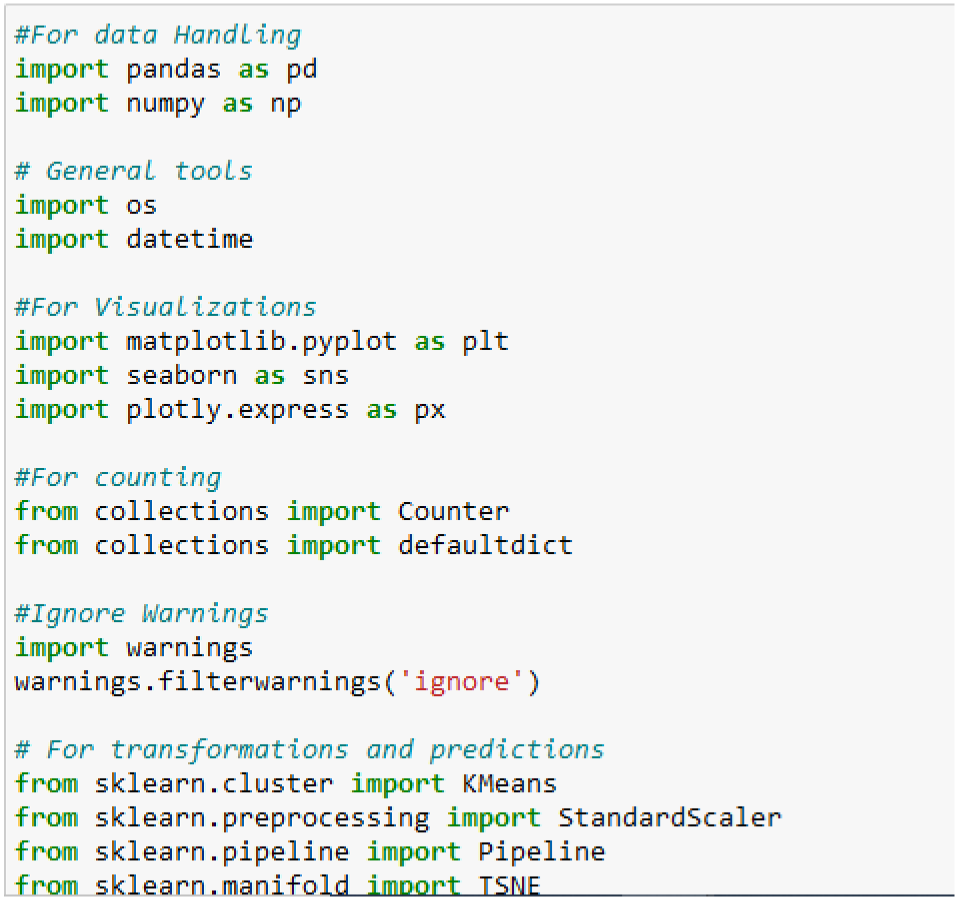
This music recommendation system will recommend songs based on the similar genre. For this we are using K-means clustering, an unsupervised machine Learning algorithm.

Let’s get started!

Methodology

1. Importing Libraries:

We import following necessary libraries:



```
#For data Handling
import pandas as pd
import numpy as np

# General tools
import os
import datetime

#For Visualizations
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

#For counting
from collections import Counter
from collections import defaultdict

#Ignore Warnings
import warnings
warnings.filterwarnings('ignore')

# For transformations and predictions
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.metrics import euclidean_distances
from scipy.spatial.distance import cdist
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer
import difflib

#For metrics Evaluation
from sklearn.metrics import silhouette_score,calinski_harabasz_score,davies_bouldin_score
from sklearn.datasets import make_blobs

#For connecting to spotify web API
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
import spotipy.util

# To create streamlit application
import streamlit as st

# For base64 encoding
import base64
```

2. Data Collection and Understanding of Data:

I. Source of Dataset: Kaggle
   - Dataset1
   - Dataset2

Imported Datasets

- data.csv
- data_by_year.csv
- data_by_genres.csv

The data used in this project was collected from Spotify's Web API. This is a computer algorithm that Spotify has that can estimate various aspects of the audio file.

Below, I will list the various attributes given to every song as referred from Spotify's developer page: **https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/**

Most of the columns are self-explanatory. The audio features are:

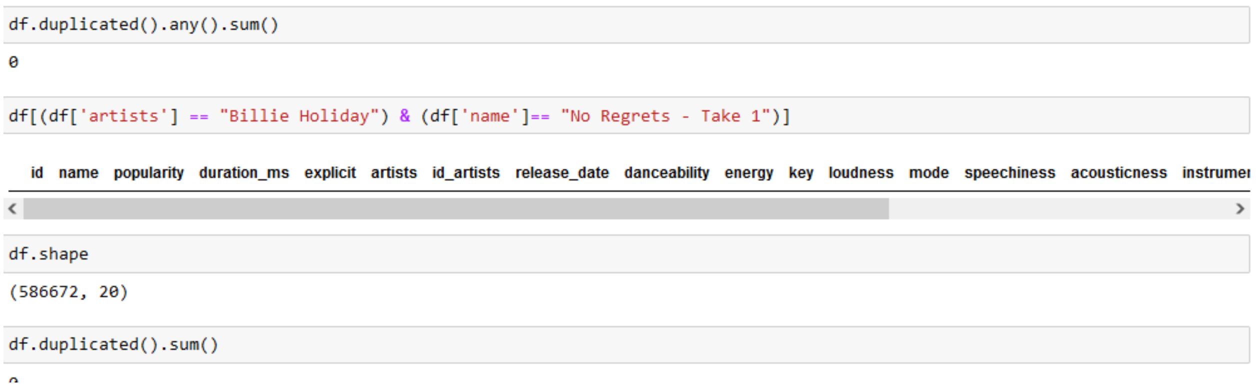
1. **duration_ms**: The duration of the track in milliseconds.
2. **key** - The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1.
3. **id**: Id of track generated by Spotify) Numerical.

4. **popularity** - The popularity of the track. The value will be between 0 and 100, with 100 being the most popular. The popularity is calculated by algorithm and is based, in the most part, on the total number of plays the track has had and how recent those plays are. Generally speaking, songs that are being played a lot now will have a higher popularity than songs that were played a lot in the past. Artist and album popularity is derived mathematically from track popularity. Note that the popularity value may lag actual popularity by a few days: the value is not updated in real time.
5. **danceability**: Danceability is defined as how suitable a track is for dancing based on a combination of musical elements. This depends on the music features, such as tempo, rhythm stability, beat strength, overall regularity, etc. As you can see in the graph, the value of 0.0 is the least danceable and 1.0 is the most danceable.
6. **acousticness**: This value describes how acoustic the song is. A score of 1.0 means the song is most likely an acoustic one.
7. **energy**: Energy is defined as the sense of forwarding motion in music that will keep the listener engaged and listening. We can identify the music's energy when the drums get busier and play louder, and the singer sings higher in an intense tone. An increase in volume and instrumentation, changing of performance style, basic beat, rhythm, and change in lyrics are contributions to the energy of music.
8. **instrumentalness**: This represents the number of vocals in a song. The closer the value to 1.0, the more instrumental the song is.
9. **liveness:** This value denotes the probability that the song is recorded with a live audience. According to the official documentation, "*a value above 0.8 provides a strong likelihood that the track is live*".
10. **loudness**: The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing the relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 dB.
11. **speechiness**: It denotes the presence of spoken words in a song. 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.
12. **tempo**: The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
13. **valence**: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. 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).
14. **time_signature**: An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).

15. **key**: The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1.
16. **mode**: Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
17. **name**: Name of the song
18. **release_date**- Date of release mostly in yyyy-mm-dd format, however precision of date may vary
19. **artists** - List of artists mentioned
20. **id_artists**- Spotify Ids for the artists

2. Data Cleaning

- Checking for duplicates and dropping duplicates if present:



```
df.duplicated().any().sum()
```

```
0
```

```
df[(df['artists'] == "Billie Holiday") & (df['name']== "No Regrets - Take 1")]
```

```
   id  name  popularity  duration_ms  explicit  artists  id_artists  release_date  danceability  energy  key  loudness  mode  speechiness  acousticness  instrumer
```

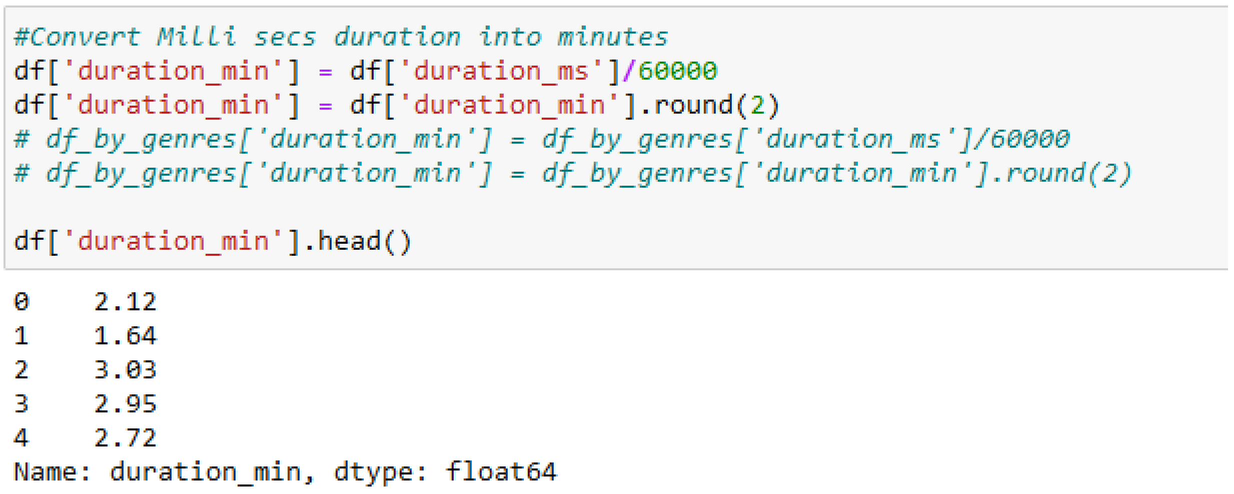
```
df.shape
```

```
(586672, 20)
```

```
df.duplicated().sum()
```

```
0
```

- Converting milliseconds data to minutes

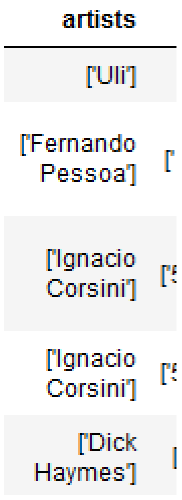


```
#Convert Milli secs duration into minutes
df['duration_min'] = df['duration_ms']/60000
df['duration_min'] = df['duration_min'].round(2)
# df_by_genres['duration_min'] = df_by_genres['duration_ms']/60000
# df_by_genres['duration_min'] = df_by_genres['duration_min'].round(2)

df['duration_min'].head()
```

```
0    2.12
1    1.64
2    3.03
3    2.95
4    2.72
Name: duration_min, dtype: float64
```

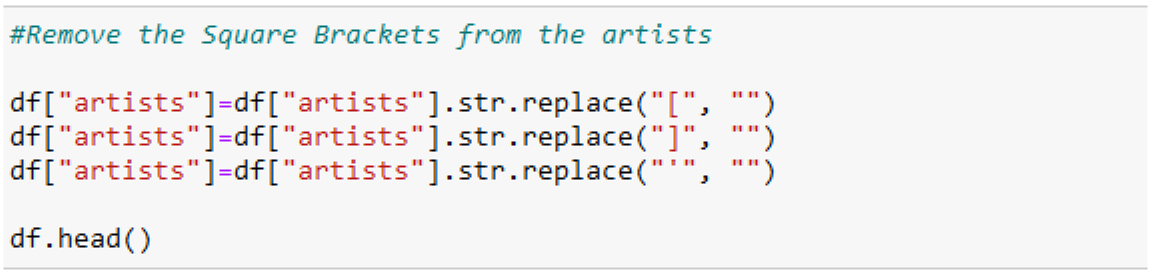
- For proper analysis we remove brackets from the artists columns

Before:

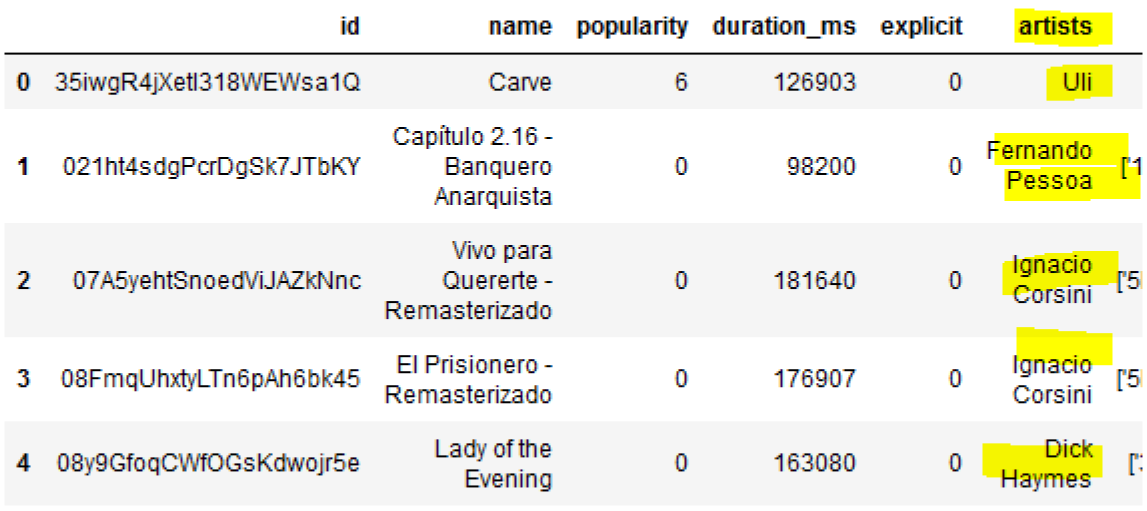


artists	
['Uli']	
['Fernando Pessoa']	['
['Ignacio Corsini']	['5
['Ignacio Corsini']	['5
['Dick Haymes']	[

After:

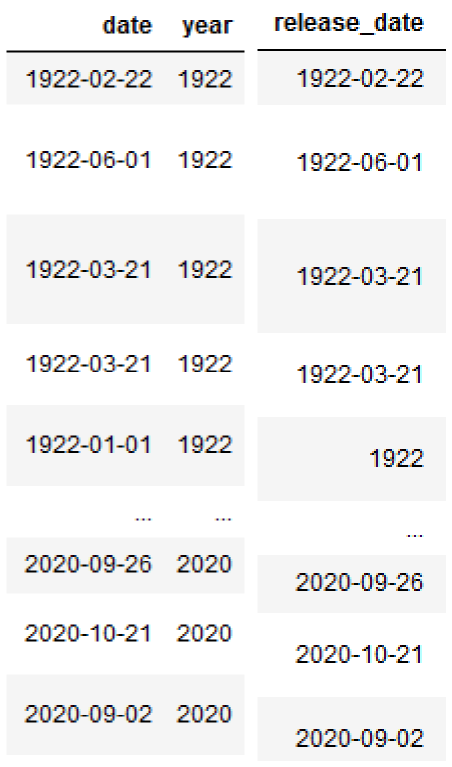


```
#Remove the Square Brackets from the artists

df["artists"]=df["artists"].str.replace("[", "")
df["artists"]=df["artists"].str.replace("]", "")
df["artists"]=df["artists"].str.replace("'", "")

df.head()
```



	id	name	popularity	duration_ms	explicit	artists	
0	35iwgR4jXetI318WEWsa1Q	Carve	6	126903	0	Uli	
1	021ht4sdgPcrDgSk7JTbKY	Capítulo 2.16 - Banquero Anarquista	0	98200	0	Fernando Pessoa	['1
2	07A5yehtSnoedViJAZkNnc	Vivo para Quererte - Remasterizado	0	181640	0	Ignacio Corsini	['5
3	08FmqUhxtyLTn6pAh6bk45	El Prisionero - Remasterizado	0	176907	0	Ignacio Corsini	['5
4	08y9GfoqCWfOGsKdwojr5e	Lady of the Evening	0	163080	0	Dick Haymes	['

- Separating year from release_date and creating another column,'year':

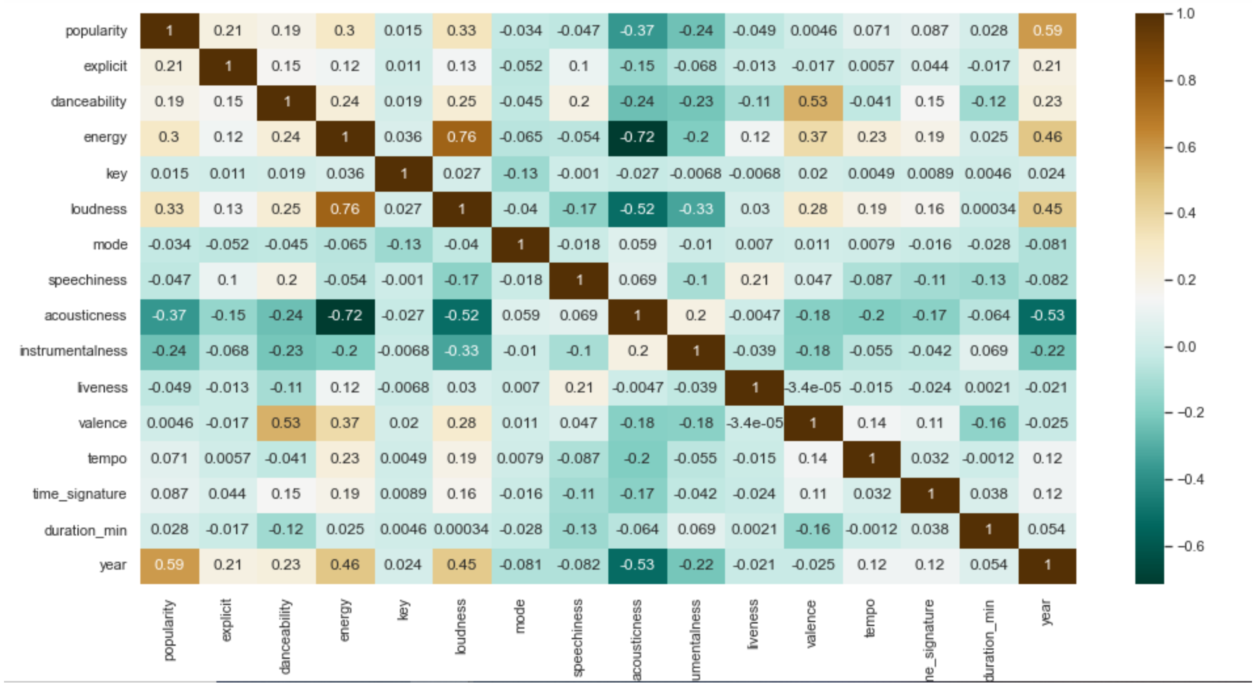


date	year	release_date
1922-02-22	1922	1922-02-22
1922-06-01	1922	1922-06-01
1922-03-21	1922	1922-03-21
1922-03-21	1922	1922-03-21
1922-01-01	1922	1922
...	...	...
2020-09-26	2020	2020-09-26
2020-10-21	2020	2020-10-21
2020-09-02	2020	2020-09-02

3. Data Exploration and Analysis:

Below are some snapshots from Jupyter notebook and and my working project on Streamlit showcasing the Exploratory data analysis:

- HeatMap: Understanding which specific attributes make the a song more popular by using a heatmap



	popularity	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	umentalness	liveness	valence	tempo	ne_signature	uration_min	year
popularity	1	0.21	0.19	0.3	0.015	0.33	-0.034	-0.047	-0.37	-0.24	-0.049	0.0046	0.071	0.087	0.028	0.59
explicit	0.21	1	0.15	0.12	0.011	0.13	-0.052	0.1	-0.15	-0.068	-0.013	-0.017	0.0057	0.044	-0.017	0.21
danceability	0.19	0.15	1	0.24	0.019	0.25	-0.045	0.2	-0.24	-0.23	-0.11	0.53	-0.041	0.15	-0.12	0.23
energy	0.3	0.12	0.24	1	0.036	0.76	-0.065	-0.054	-0.72	-0.2	0.12	0.37	0.23	0.19	0.025	0.46
key	0.015	0.011	0.019	0.036	1	0.027	-0.13	-0.001	-0.027	-0.0068	-0.0068	0.02	0.0049	0.0089	0.0046	0.024
loudness	0.33	0.13	0.25	0.76	0.027	1	-0.04	-0.17	-0.52	-0.33	0.03	0.28	0.19	0.16	0.00034	0.45
mode	-0.034	-0.052	-0.045	-0.065	-0.13	-0.04	1	-0.018	0.059	-0.01	0.007	0.011	0.0079	-0.016	-0.028	-0.081
speechiness	-0.047	0.1	0.2	-0.054	-0.001	-0.17	-0.018	1	0.069	-0.1	0.21	0.047	-0.087	-0.11	-0.13	-0.082
acousticness	-0.37	-0.15	-0.24	-0.72	-0.027	-0.52	0.059	0.069	1	0.2	-0.0047	-0.18	-0.2	-0.17	-0.064	-0.53
instrumentalness	-0.24	-0.068	-0.23	-0.2	-0.0068	-0.33	-0.01	-0.1	0.2	1	-0.039	-0.18	-0.055	-0.042	0.069	-0.22
liveness	-0.049	-0.013	-0.11	0.12	-0.0068	0.03	0.007	0.21	-0.0047	-0.039	1	-3.4e-05	-0.015	-0.024	0.0021	-0.021
valence	0.0046	-0.017	0.53	0.37	0.02	0.28	0.011	0.047	-0.18	-0.18	-3.4e-05	1	0.14	0.11	-0.16	-0.025
tempo	0.071	0.0057	-0.041	0.23	0.0049	0.19	0.0079	-0.087	-0.2	-0.055	-0.015	0.14	1	0.032	-0.0012	0.12
time_signature	0.087	0.044	0.15	0.19	0.0089	0.16	-0.016	-0.11	-0.17	-0.042	-0.024	0.11	0.032	1	0.038	0.12
duration_min	0.028	-0.017	-0.12	0.025	0.0046	0.00034	-0.028	-0.13	-0.064	0.069	0.0021	-0.16	-0.0012	0.038	1	0.054
year	0.59	0.21	0.23	0.46	0.024	0.45	-0.081	-0.082	-0.53	-0.22	-0.021	-0.025	0.12	0.12	0.054	1

Observations from heatmap are:

1.Loudness and energy are highly correlated. This makes some sense as energy is definately influence by the volume the music is being played at.

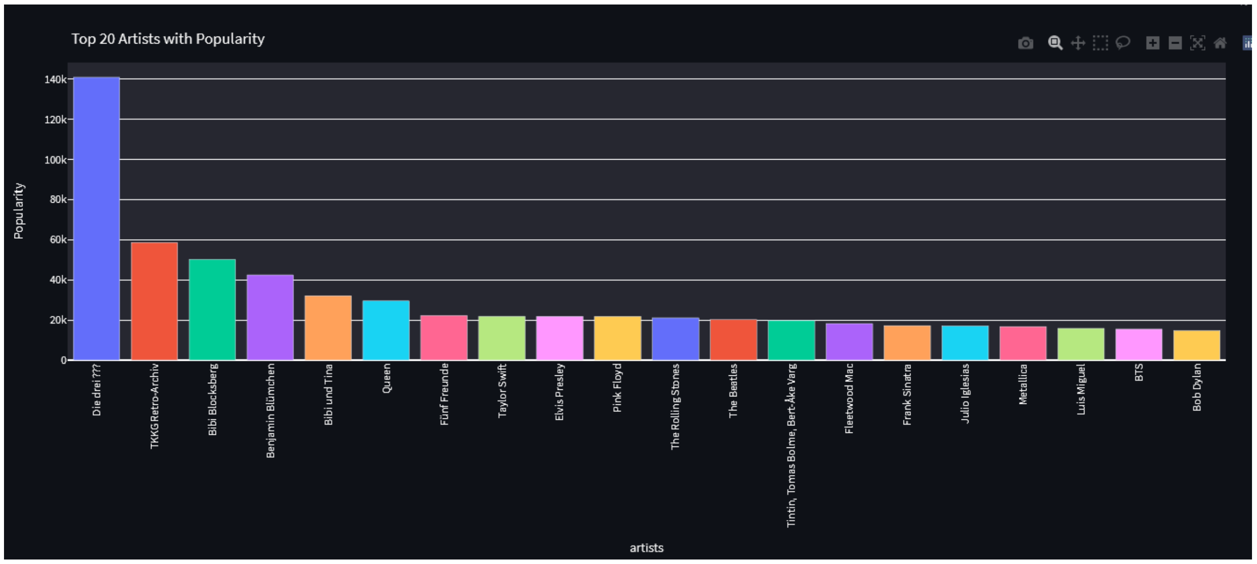
2.Valence and dancability are highly coorelated. Dance songs are usually happier and in a major key.

3.Energy also seems to influence a song's popularity. Many popular songs are energetic, though not necessarily dance songs. Because the correlation here is not too high, low energy songs do have some potential to be more popular.

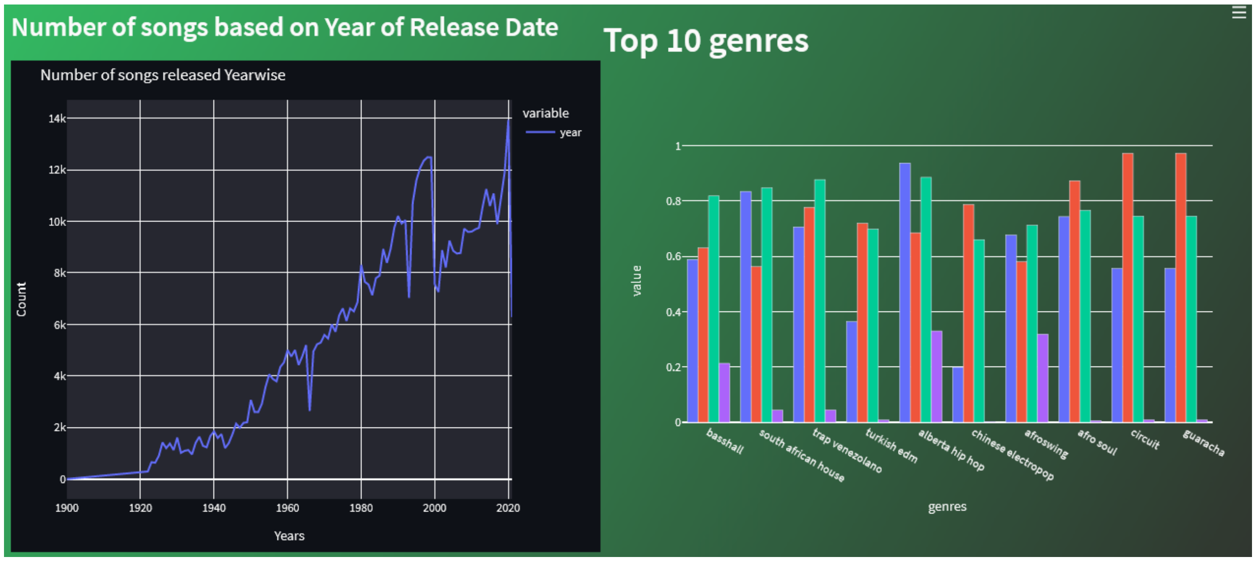
4.As expected popularity is highly correlated with the year released. This makes sense as the Spotify algorithm which makes this decision generates it's "popularity" metric by not just how many streams a song receives, but also how recent those streams are.

5.Acousticness seems to be uncorrelated with popularity. Most popular songs today have either electronic or electric instruments in them. It is very rare that a piece of music played by a chamber orchestra or purely acoustic band becomes immesely popular (though, again, not impossible).

Few snapshots of analysis and visualizations:

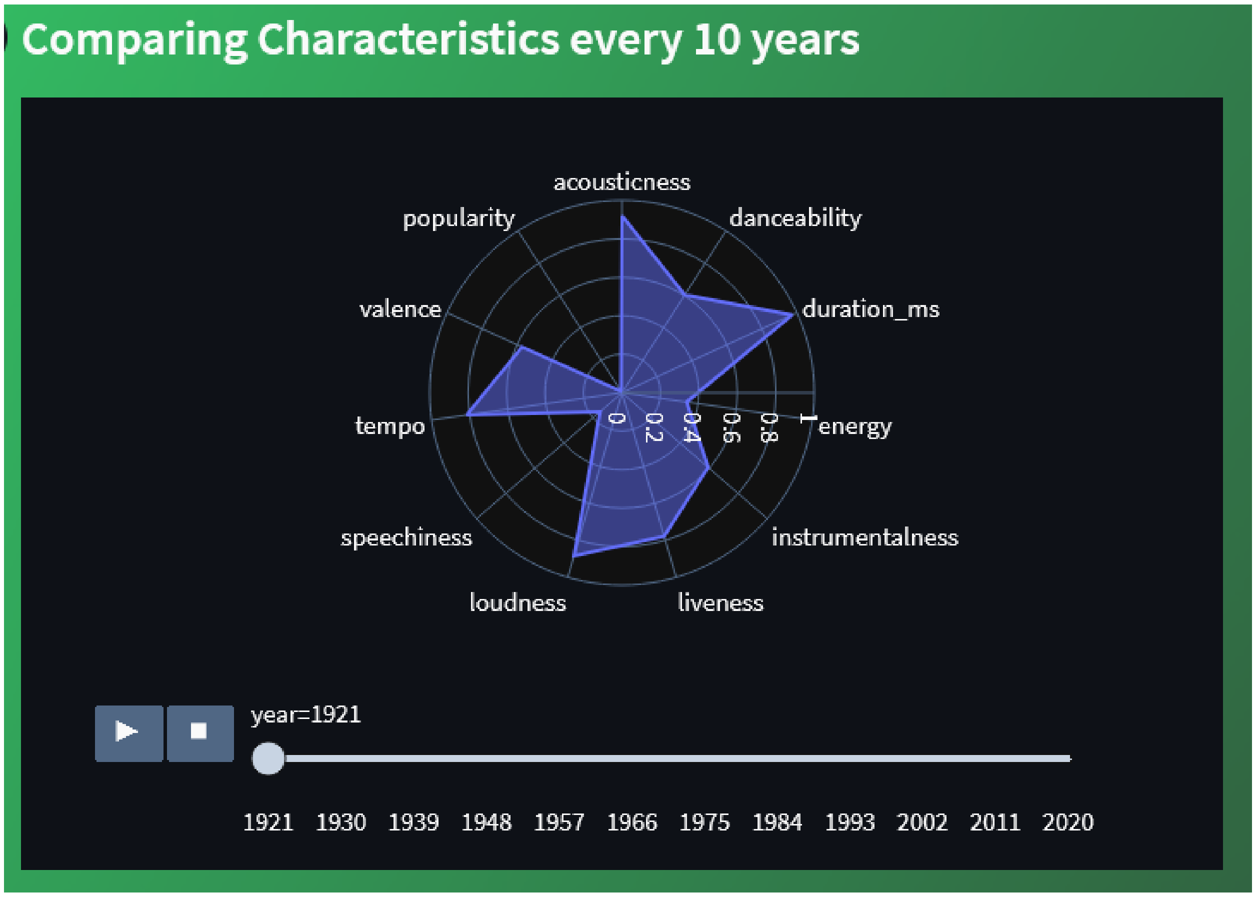


Observation: Die drei??? is most popular



Observation:

- 13937 songs are released in year 2020
- Data_by_genres dataset contains the audio features for different songs along with the audio features for different genres. We used this information to compare different genres and understand their unique differences in sound by having top 10 genres by popularity.



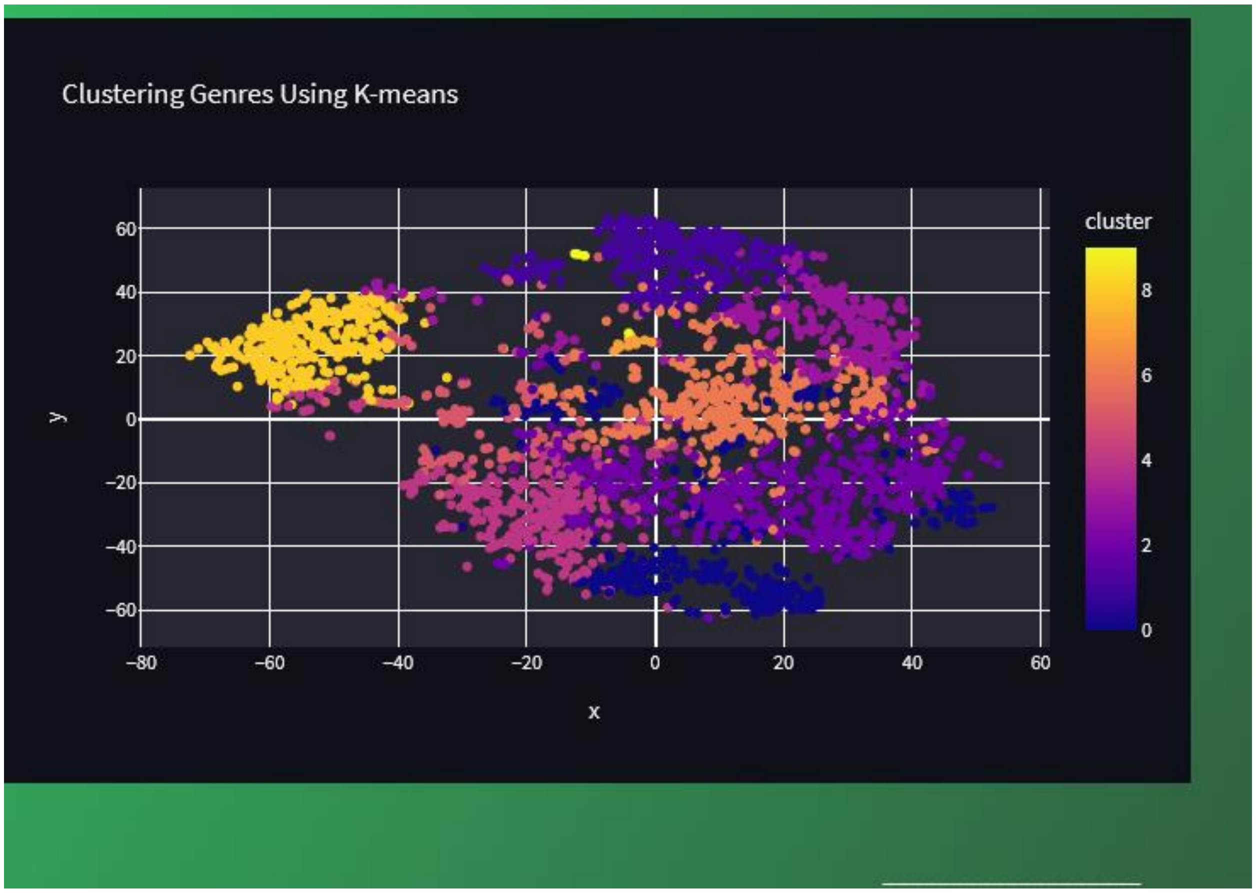
Clustering

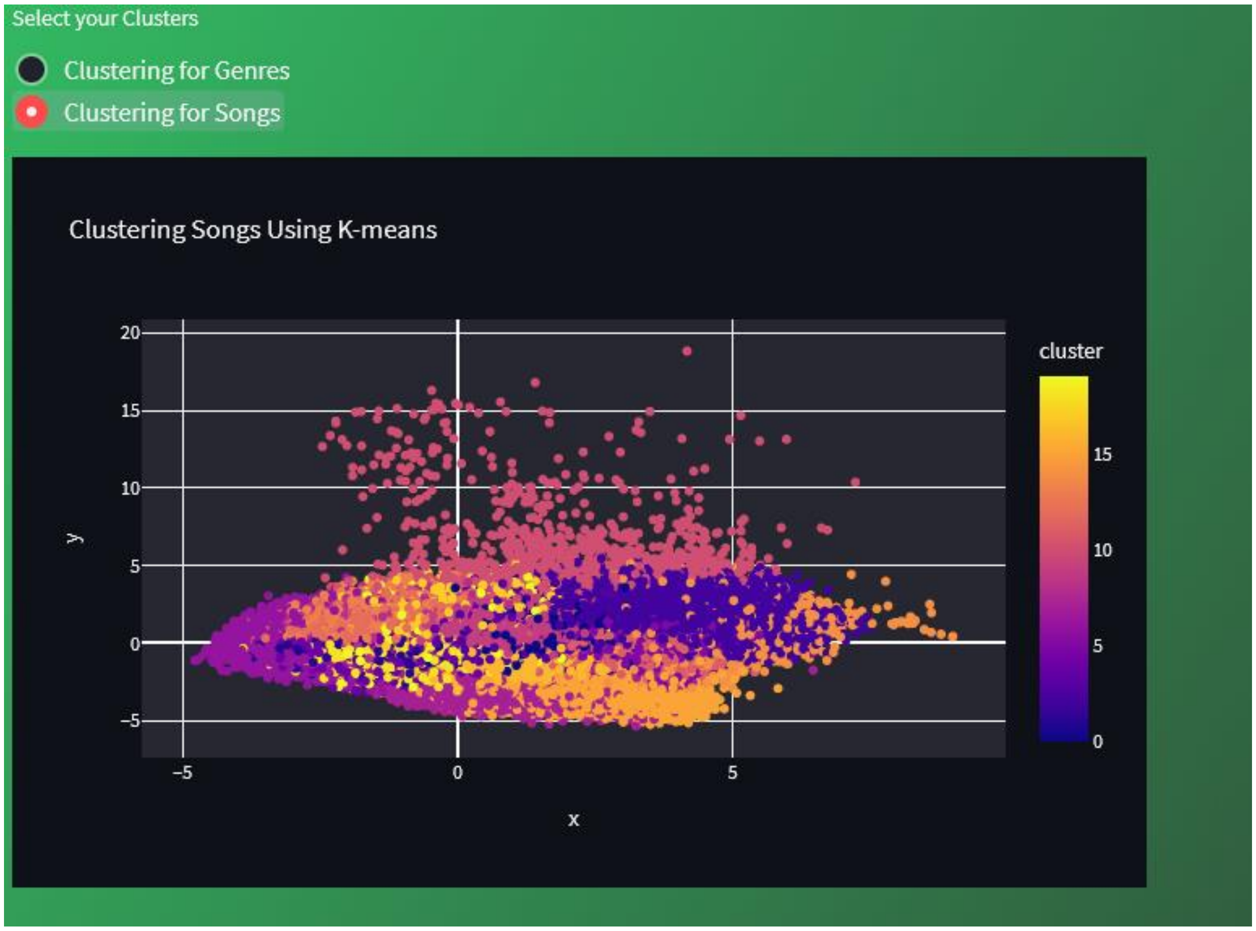
- In this project, I have taken the approach of using K-means Clustering on the dataset to find the key aspects that are common.
- Here, the simple K-means clustering algorithm is used to divide the genres in this dataset into ten clusters based on the numerical audio features of each genres
- Before proceeding further with clustering, we have to do something called standardization.
- Standardization is the process of putting different features on the same scale, with each scaled feature having a mean of 0 and a standard deviation of 1.
- This is important because the model is not familiar with the context of the data. If we do not standardize our data, the model would place a much larger weight on tempo and loudness, since those variables vary by much more than the variables that are distributed in the range from 0 to 1.
- Standardization allows for all features to be treated equally by the model. Here, I have imported sklearn’s StandardScaler.

- Principal Component Analysis(PCA) is a dimensionality reduction approach which I have used while creating clusters for songs. This approach attempts to find best possible subspace that tells us more about variance of the data.
- After fitting a PCA object to the standardized matrix, I have chosen suitable number of principle components ie n_components=2 to preserve 80% of variance and incorporate in K-means clustering algorithm.

```
# Visualizing the Clusters with PCA

from sklearn.decomposition import PCA

pca_pipeline = Pipeline([('scaler', StandardScaler()), ('PCA', PCA(n_components=2))])
song_embedding = pca_pipeline.fit_transform(X)
```

Below are the two clustering:

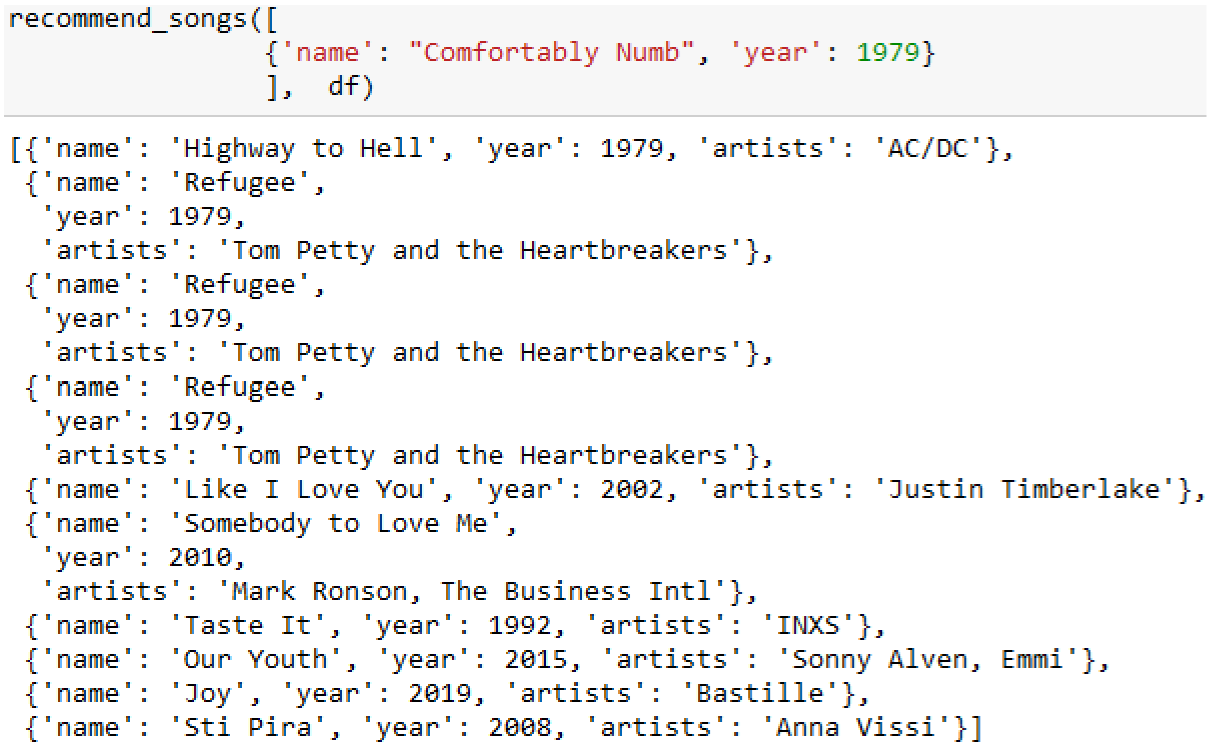




Now based on the clustering, I have proceeded further with building Recommendation System:

- Based on the above analysis and visualizations, it's evident genres that are similar tend to have data points that are located close to each other while similar types of songs are also clustered together.
- This observation makes perfect sense. Similar genres will sound similar and come from similar time periods, and songs within those genres will sound similar as well.
- By gathering the data points from the songs a user has listened to and recommending songs that correlate to nearby data points, we can develop a recommendation engine..
- Spotipy is a Python client for the Spotify Web API that makes it easy for developers to fetch data and query Spotify's catalog for songs. You have to install using pip install spotipy
- After installing Spotipy, you will need to create an app on the Spotify Developer's page and save your Client ID and secret key.
- By providing this information, we will get the playlist and thus b running the recommendation system we will be getting the 10 recommendations of song which will be based on the similar genres to the song provided.

Testing 1:



```
recommend_songs([
                {'name': "Comfortably Numb", 'year': 1979}
                ],  df)
```

```
[{'name': 'Highway to Hell', 'year': 1979, 'artists': 'AC/DC'},
 {'name': 'Refugee',
  'year': 1979,
  'artists': 'Tom Petty and the Heartbreakers'},
 {'name': 'Refugee',
  'year': 1979,
  'artists': 'Tom Petty and the Heartbreakers'},
 {'name': 'Refugee',
  'year': 1979,
  'artists': 'Tom Petty and the Heartbreakers'},
 {'name': 'Like I Love You', 'year': 2002, 'artists': 'Justin Timberlake'},
 {'name': 'Somebody to Love Me',
  'year': 2010,
  'artists': 'Mark Ronson, The Business Intl'},
 {'name': 'Taste It', 'year': 1992, 'artists': 'INXS'},
 {'name': 'Our Youth', 'year': 2015, 'artists': 'Sonny Alven, Emmi'},
 {'name': 'Joy', 'year': 2019, 'artists': 'Bastille'},
 {'name': 'Sti Pira', 'year': 2008, 'artists': 'Anna Vissi'}]
```

Observation:

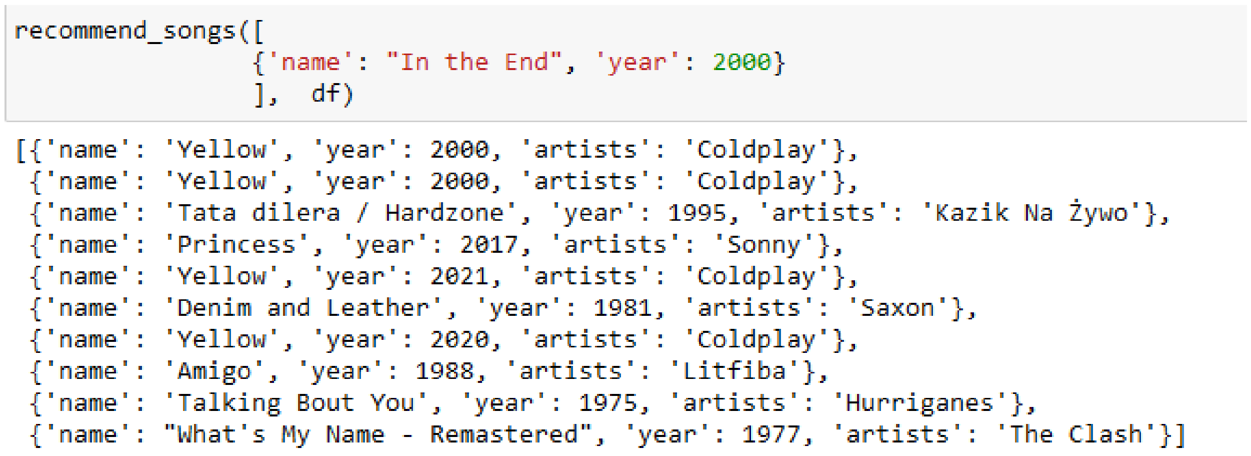
Here, I had provided the song which belongs to Progressive Rock,Pop.

So, the recommendation System provided me with the 10 songs as a recommendation which are similar to or belong to Pop and Progressive Rock.

From the above recommendation songs list :

Highway to hell – Rock, Refugee by Tom Petty -Rock, Somebody to Love Me- Pop

Testing 2:



```
recommend_songs([
                {'name': "In the End", 'year': 2000}
                ],  df)
```

```
[{'name': 'Yellow', 'year': 2000, 'artists': 'Coldplay'},
 {'name': 'Yellow', 'year': 2000, 'artists': 'Coldplay'},
 {'name': 'Tata dilera / Hardzone', 'year': 1995, 'artists': 'Kazik Na Żywo'},
 {'name': 'Princess', 'year': 2017, 'artists': 'Sonny'},
 {'name': 'Yellow', 'year': 2021, 'artists': 'Coldplay'},
 {'name': 'Denim and Leather', 'year': 1981, 'artists': 'Saxon'},
 {'name': 'Yellow', 'year': 2020, 'artists': 'Coldplay'},
 {'name': 'Amigo', 'year': 1988, 'artists': 'Litfiba'},
 {'name': 'Talking Bout You', 'year': 1975, 'artists': 'Hurriganes'},
 {'name': "What's My Name - Remastered", 'year': 1977, 'artists': 'The Clash'}]
```

Observation:

Here, I had provided the song which belongs to Rap Rock,Alternative/Indie/Metal.

So, the recommendation System provided me with the 10 songs as a recommendation which are similar to or belong to Rap Rock,Alternative/Indie/Metal.

From the above recommendation songs list : Yellow by Coldplay – Alternative/Indie, Denim and Leathe by Saxon -Rock& Metal

Testing 3:

```
recommend_songs([{'name': 'Life is a Highway - From "Cars', 'year':2009},
                {'name': 'Smells Like Teen Spirit', 'year': 1991},
                {'name': 'Lithium', 'year': 1992},
                {'name': 'All Apologies', 'year': 1993},
                {'name': 'Stay Away', 'year': 1993}],  df)
```

```
[{'name': 'Star - 2019 - Remaster', 'year': 1989, 'artists': 'Erasure'},
 {'name': 'All For A Reason', 'year': 1998, 'artists': 'Alessi Brothers'},
 {'name': 'Luka', 'year': 1987, 'artists': 'Suzanne Vega'},
 {'name': 'Ghostrider in the Sky',
  'year': 1995,
  'artists': 'Foggy Mountain Rockers'},
 {'name': "That's Love", 'year': 1994, 'artists': 'Blue System'},
 {'name': 'Liberty!', 'year': 1990, 'artists': 'Kon Kan'},
 {'name': 'Si Tienes Fe', 'year': 1993, 'artists': 'Óscar Medina'},
 {'name': 'Sempre Será', 'year': 1995, 'artists': 'Ara Ketu'},
 {'name': '愛を止めないで', 'year': 1996, 'artists': 'Kazumasa Oda'},
 {'name': 'Always on My Mind', 'year': 1998, 'artists': 'Pet Shop Boys'}]
```

Observation:

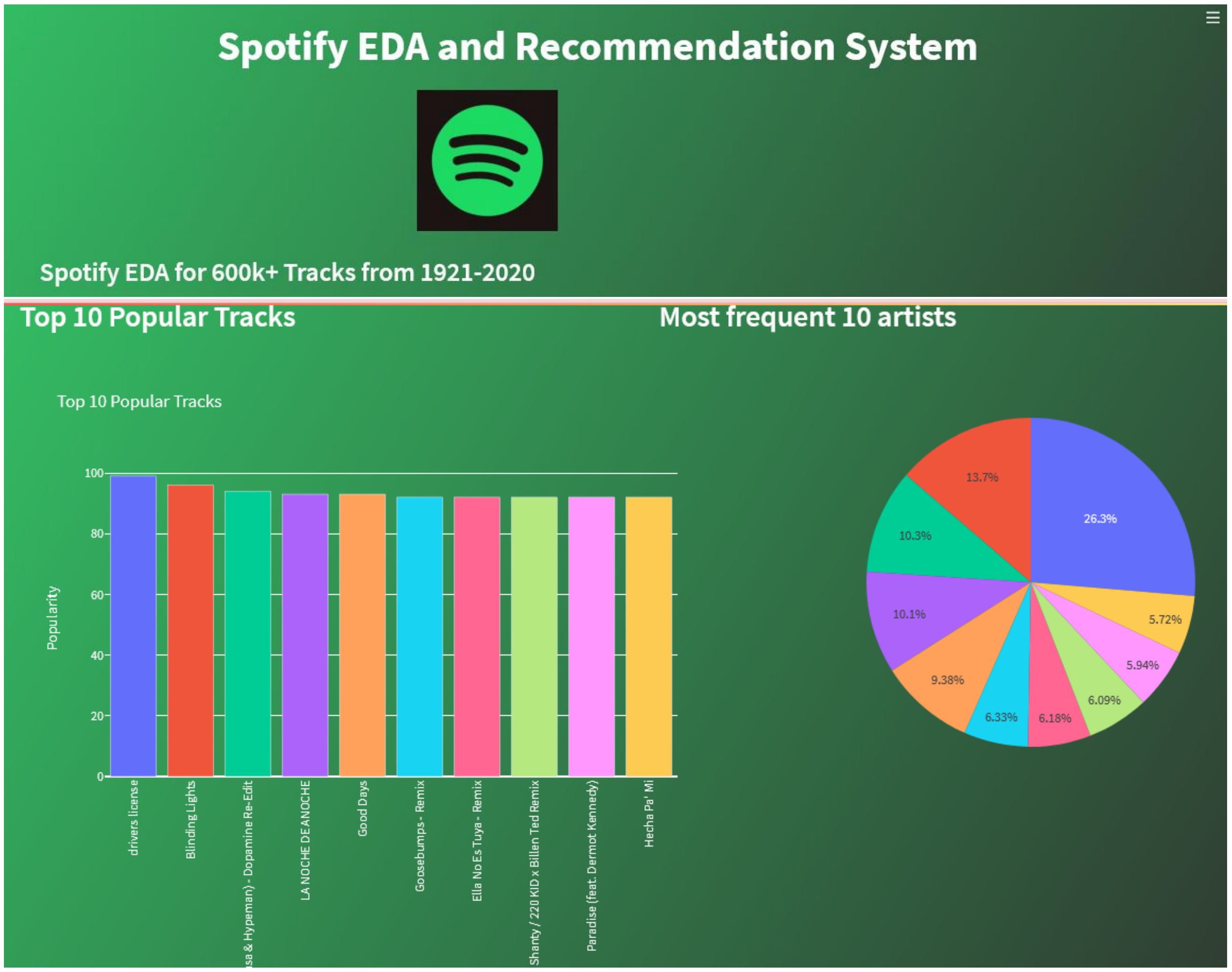
Here, I had provided the songs belongs to Rock, Heartland rock, Country, Alternative/Indie, pop

So, the recommendation System provided me with the 10 songs as a recommendation which are similar to or belong to Rock, Heartland rock, Country, Alternative/Indie, pop,R&B

From the above recommendation songs list : Star -2019 , Luka, That's Love belong to Pop, R&B, Country.

Results and Conclusion

- I learned how to use clustering to find interesting patterns in the dataset.
- I got hands-on experience on building recommendation systema and learnt how to use Spotify web API.
- One of the limitations that I faced while working on the project was that Spotify web API does not have genre data which majority determines the popularity of songs/albums. I incorporated genre data from other sources for the analysis purpose.
- I successfully analyzed the Spotify dataset and answered many questions like Top artists, top 10 songs, artists with most songs, etc.
- Elbow method was also implemented to find the optimal value of k which can be checked from Jupyter notebook.
- Running Visualization Streamlit App:



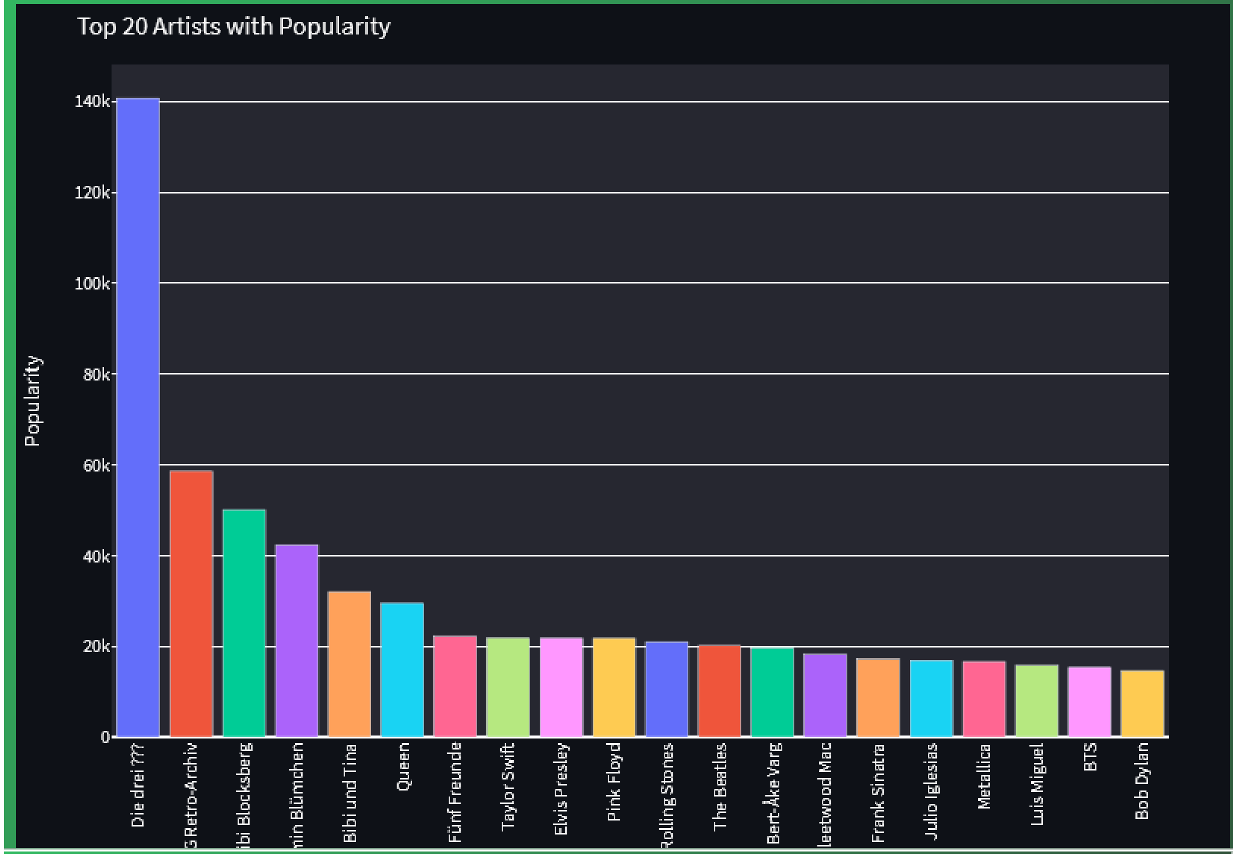
Spotify EDA and Recommendation System

Spotify EDA for 600k+ Tracks from 1921-2020

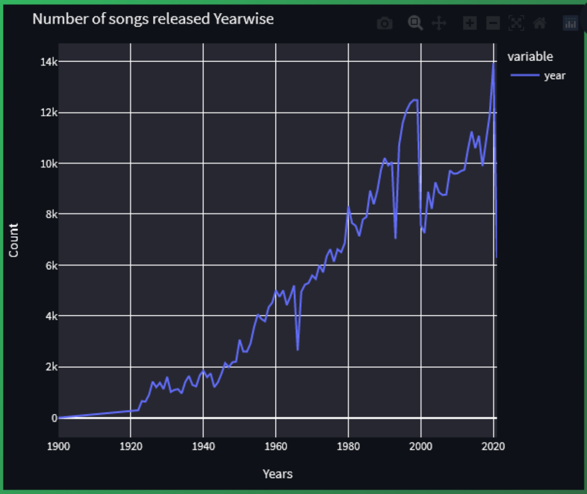
Top 10 Popular Tracks

Most frequent 10 artists

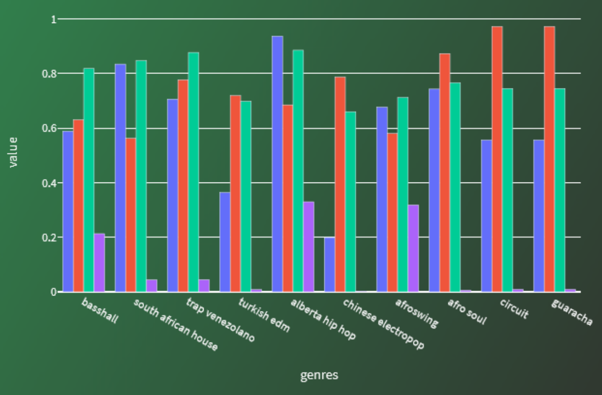
Most Popular Artists

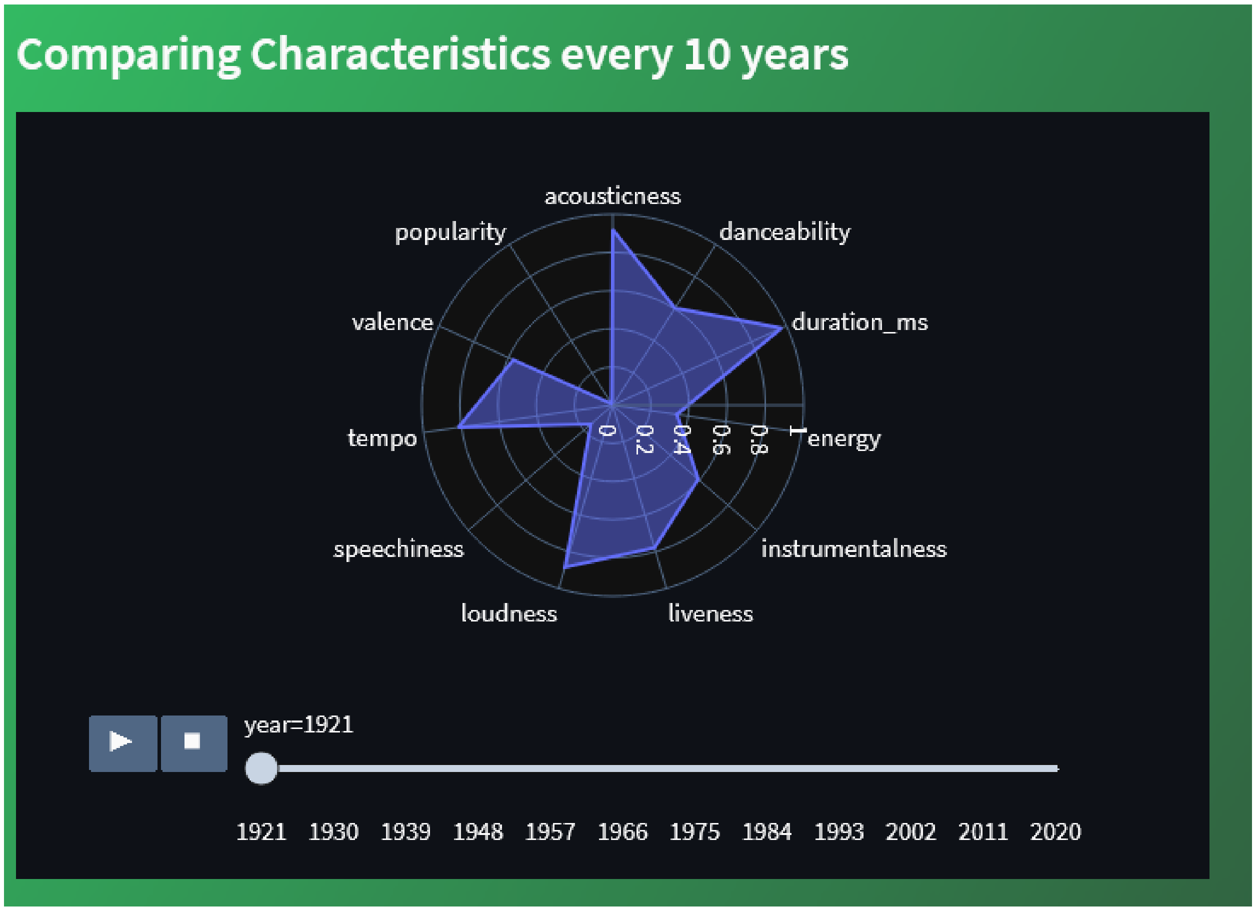


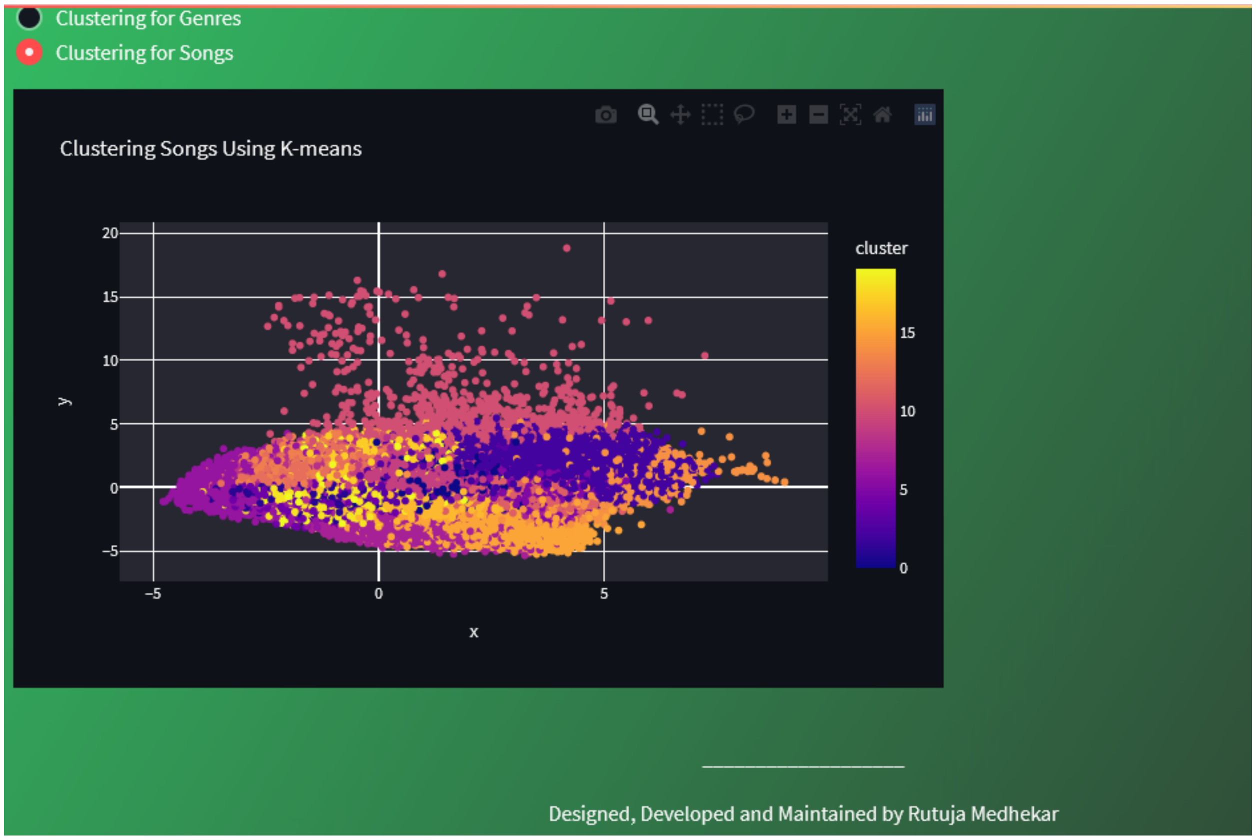
Number of songs based on Year of Release Date



Top 10 genres




Comparing Characteristics every 10 years



Designed, Developed and Maintained by Rutuja Medhekar

References

[1] Spotify Dataset : https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks?datasetId=1993933&sortBy=dateRun&tab=bookmarked&select=tracks.csv

[2] Spotify Web API: https://developer.spotify.com/dashboard/

[3] https://towardsdatascience.com/k-means-clustering-and-pca-to-categorize-music-by-similar-audio-features-df09c93e8b64

[4] https://betterprogramming.pub/how-to-extract-any-artists-data-using-spotify-s-api-python-and-spotipy-4c079401bc37

[5] Steamlit Documentation: https://docs.streamlit.io/