Moodify

Spotify Mood Playlist - Cluster Analysis of Musical Attributes

Shivani Shrikant Naik, Pulkit Saharan

Introduction

Spotify is a music, podcast, and video streaming service that gives you access to millions of songs and other content from musicians all over the world, as well as a limitless library of performers from various genres and artistic backgrounds. As of 2022, Spotify provides access to over 70 million songs and 2.9 million podcasts to 365 million monthly active users including 165 million paying subscribers. Spotify allows users to find new music and build personalized playlists based on their musical tastes, favorite genres and performers, and even their mood.

Objective of Analysis

The goal and motive of our analysis is to help users create a randomized mood-based playlist, which they can use to generate a playlist with a different set of tracks by using a single track from their library. Case in point, you are a fitness freak who loves working out to start your day off but do not want to listen to remixes or radio during your workout; no worries just enter the name of the song you are feeling that day and let “Moodify” create a personalized playlist for you.

Furthermore, Moodify also gives you an option to explore songs from up and coming artists and even helps you obscurify your playlist. This will assist users in discovering new songs and artists who are similar to their preferences which in turn will also help new and lesser-known artists to build their brand.

Conclusively, in this project we use audio attributes from the Spotify dataset, such as danceability, energy, loudness, and liveness, and construct a model to group similar songs together, which will serve as the foundation for creating mood playlists to the ends of providing users the ability to create a playlist on the fly based on their mood and shuffle through a huge list of comparable songs.

Dataset

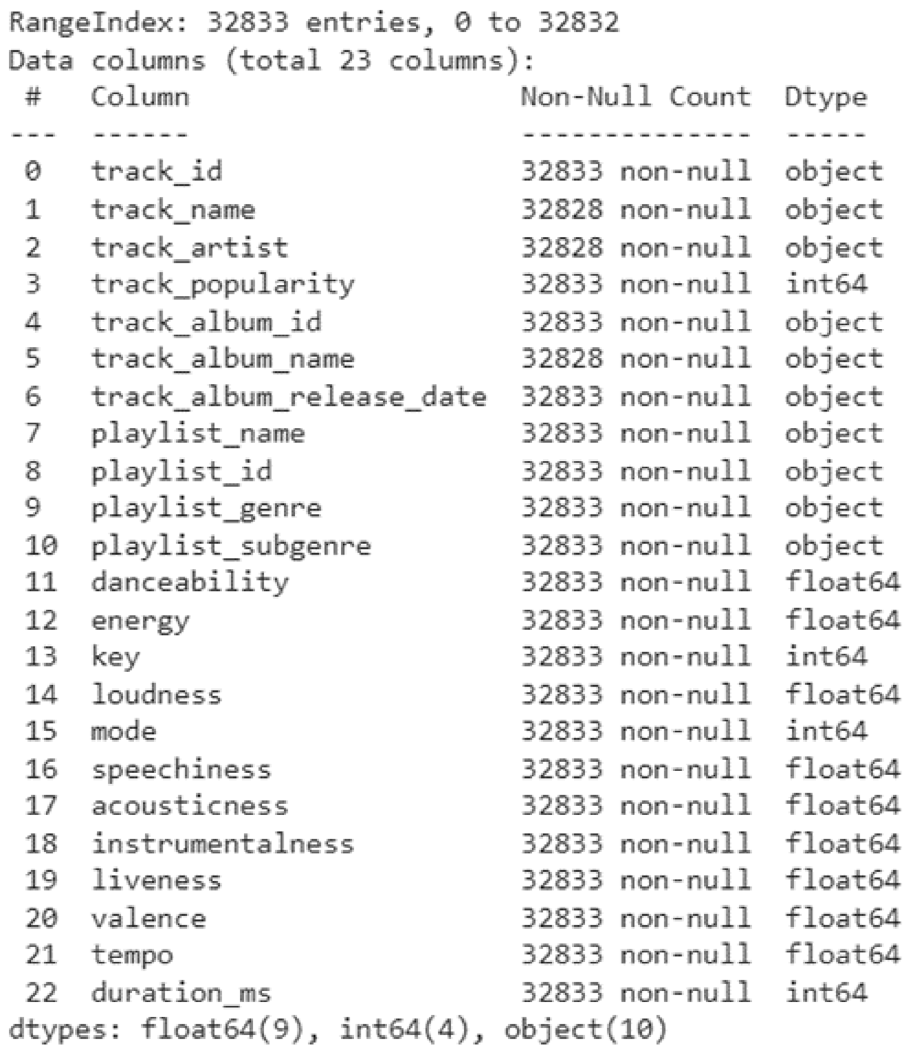
We have used 2 sources of data in this study.

The data for training the model is gathered from a publicly available GitHub repository that was collected from Spotify API. To investigate Spotify-related data, the repository offers an option-access option.

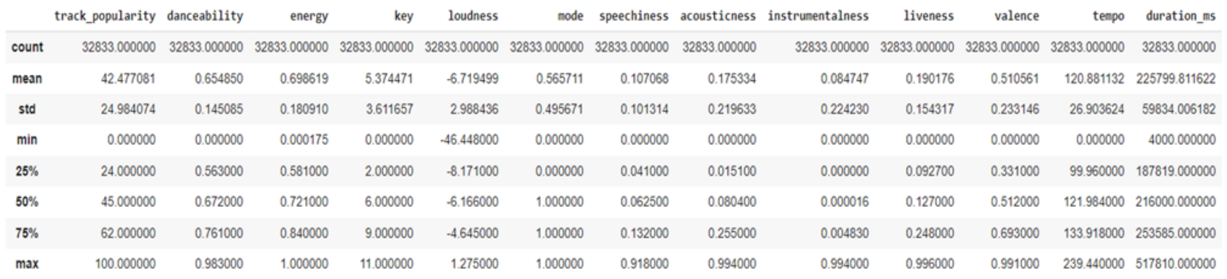
The input data to the service will be pulled from the Spotify API which will contain all the attributes we have used in clustering. Cluster for that particular song will be predicted using a trained model to generate the mood playlist.

The training dataset was collected from the Spotify API from different playlists and contains aggregated data of over 32,000 observations and 23 features, including song information such as track id, track name, and track artist; album and playlist information; and audio attributes of the music such as danceability, energy, loudness, and liveness. The dataset contains a wide variety of songs from more than 10,000 different artists and around 20,000 different albums. The dataset contains 12 numeric features and 11 nominal/categorical features.

Below is the statistical summary of all the numeric features available in the dataset.



```
RangeIndex: 32833 entries, 0 to 32832
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   track_id                  32833 non-null  object 
 1   track_name                32828 non-null  object 
 2   track_artist              32828 non-null  object 
 3   track_popularity          32833 non-null  int64  
 4   track_album_id            32833 non-null  object 
 5   track_album_name          32828 non-null  object 
 6   track_album_release_date  32833 non-null  object 
 7   playlist_name             32833 non-null  object 
 8   playlist_id               32833 non-null  object 
 9   playlist_genre            32833 non-null  object 
 10  playlist_subgenre         32833 non-null  object 
 11  danceability              32833 non-null  float64
 12  energy                    32833 non-null  float64
 13  key                       32833 non-null  int64  
 14  loudness                  32833 non-null  float64
 15  mode                      32833 non-null  int64  
 16  speechiness               32833 non-null  float64
 17  acousticness              32833 non-null  float64
 18  instrumentalness          32833 non-null  float64
 19  liveness                  32833 non-null  float64
 20  valence                   32833 non-null  float64
 21  tempo                     32833 non-null  float64
 22  duration_ms               32833 non-null  int64  
dtypes: float64(9), int64(4), object(10)
```



	track_popularity	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_ms
count	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000	32833.000000
mean	42.477081	0.654850	0.698619	5.374471	-6.719499	0.565711	0.107068	0.175334	0.084747	0.190176	0.510561	120.881132	225799.811622
std	24.984074	0.145085	0.180910	3.611657	2.988436	0.495671	0.101314	0.219633	0.224230	0.154317	0.233146	26.903624	59834.006182
min	0.000000	0.000000	0.000175	0.000000	-46.448000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	4000.000000
25%	24.000000	0.563000	0.581000	2.000000	-8.171000	0.000000	0.041000	0.015100	0.000000	0.092700	0.331000	99.960000	187819.000000
50%	45.000000	0.672000	0.721000	6.000000	-6.166000	1.000000	0.062500	0.080400	0.000016	0.127000	0.512000	121.984000	216000.000000
75%	62.000000	0.761000	0.840000	9.000000	-4.645000	1.000000	0.132000	0.255000	0.004830	0.248000	0.693000	133.918000	253585.000000
max	100.000000	0.983000	1.000000	11.000000	1.275000	1.000000	0.918000	0.994000	0.994000	0.996000	0.991000	239.440000	517810.000000

Feature Descriptions

In this project, we are interested in utilizing the audio features of the songs to use them in our clustering analysis. Each of these features describes some interesting and different aspects of a song. The following table shows the features and their description which have been used in creating the clusters:

.

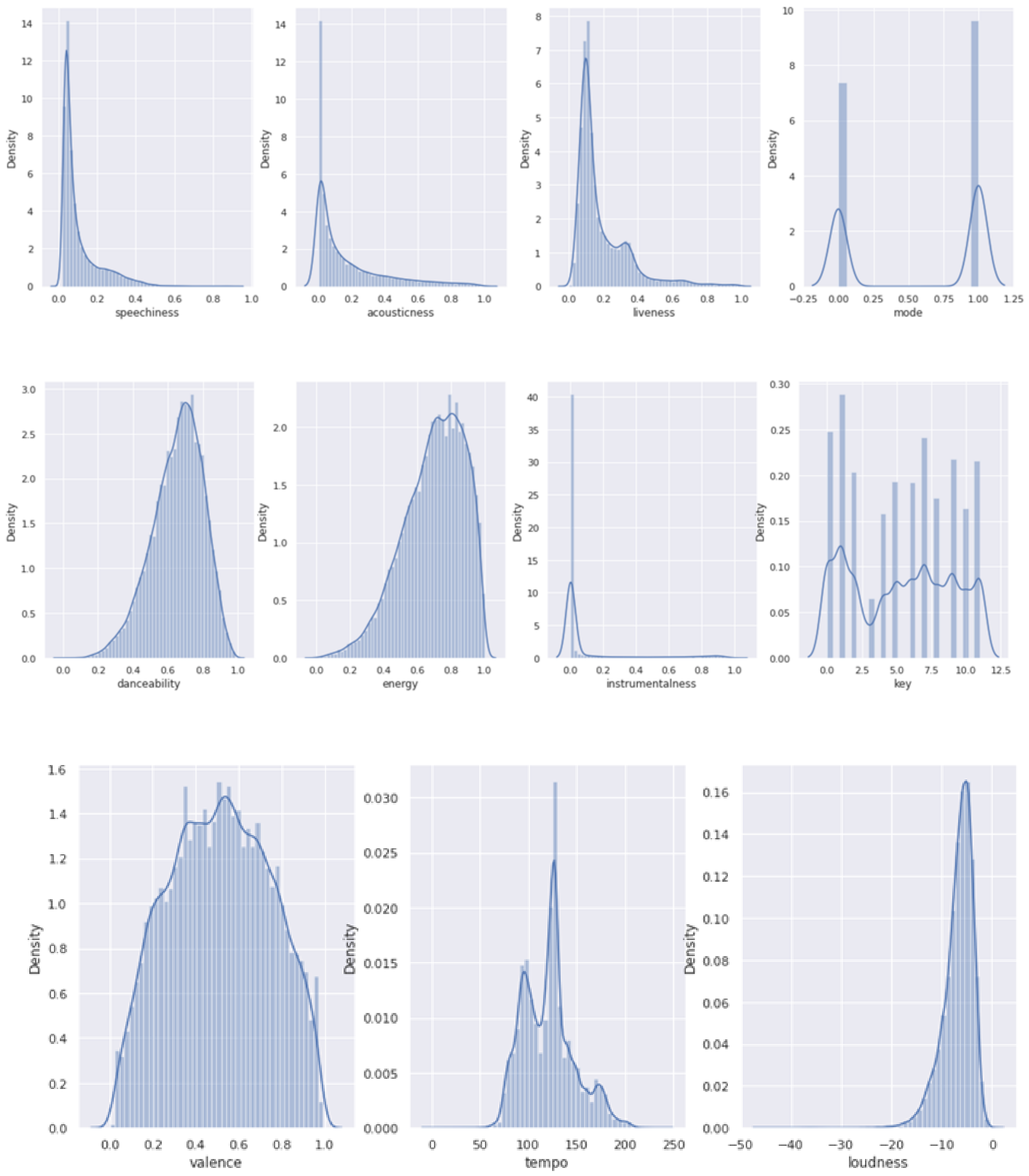
Features	Description
Danceability	Describes how suitable a track is for dancing. A value of 0.0 is least danceable, and 1.0 is most danceable.
Energy	A scale ranging from 0.0 to 1.0 that represents a subjective measure of activity and intensity. Typical energetic tunes have a quick, loud, and rowdy feel to them. For example, death metal has high energy, while a Bach prelude scores low on the scale.
Key	The track's projected overall key. Pitch Class notation is used to relate integers to pitches. It ranges between 0 to 11.
Loudness	An auditory sensation characterized by the ability to rank sounds on a scale ranging from quiet to loud. Minimum value of loudness is -46.4 and maximum value is 1.27 in the dataset.
Mode	The type of scale from which the melodic content of a music is formed (major or minor). The major is represented by 1 while the minor is represented by 0.
Speechiness	The existence of spoken words in a track is detected. If a song's speechiness is greater than 0.66, it is most likely made up of spoken words; a score between 0.33 and 0.66 indicates that the song may contain both music and words (e.g. rap music); while a score below 0.33 indicates that the song has no speech.

Acousticness	A scale from 0.0 to 1.0 that indicates whether or not the track is acoustic. The number 1.0 denotes a high level of certainty that the track is acoustic.
Instrumentalness	Represents the number of vocals in the song. The closer it is to 1.0, the greater likelihood the song contains no vocal content
Liveness	Describes the probability that the song was recorded with a live audience. A value above closer to 1 provides a strong likelihood that the track is live.
Valence	With a scale of 0.0 to 1.0, this metric describes the musical positivity expressed by a track. Tracks with a high valence sound more positive (e.g., joyful, cheerful, euphoric), while tracks with a low valence sound more negative (e.g., sad, sad, sad, sad, sad, sad, sad, sad, sad, sad, sad, sad, sad, sad, sad, sad, terrible (e.g., sad, depressed, angry) The number of vocals in the song is represented by this number.
Tempo	Describes the speed at which a piece of music is played or the timing of the song.

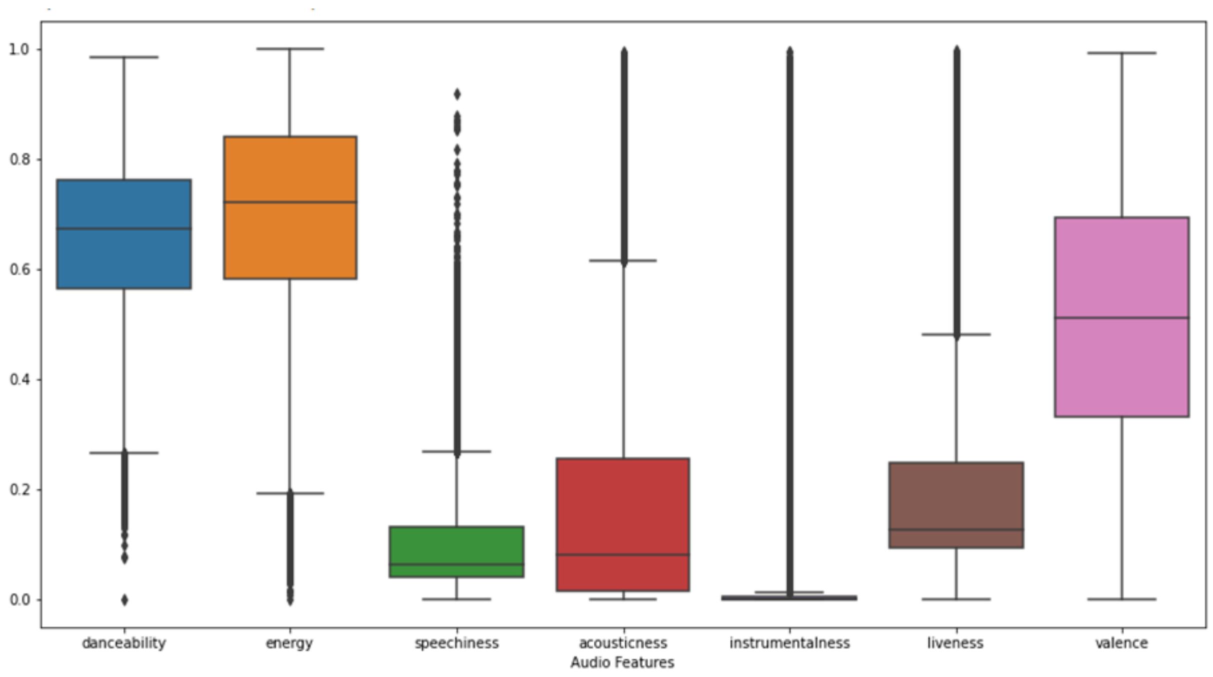
Exploratory Data Analysis

Data Distribution

Let's have a look at the distribution of the aforementioned attributes across the dataset which are to be used in our modeling.

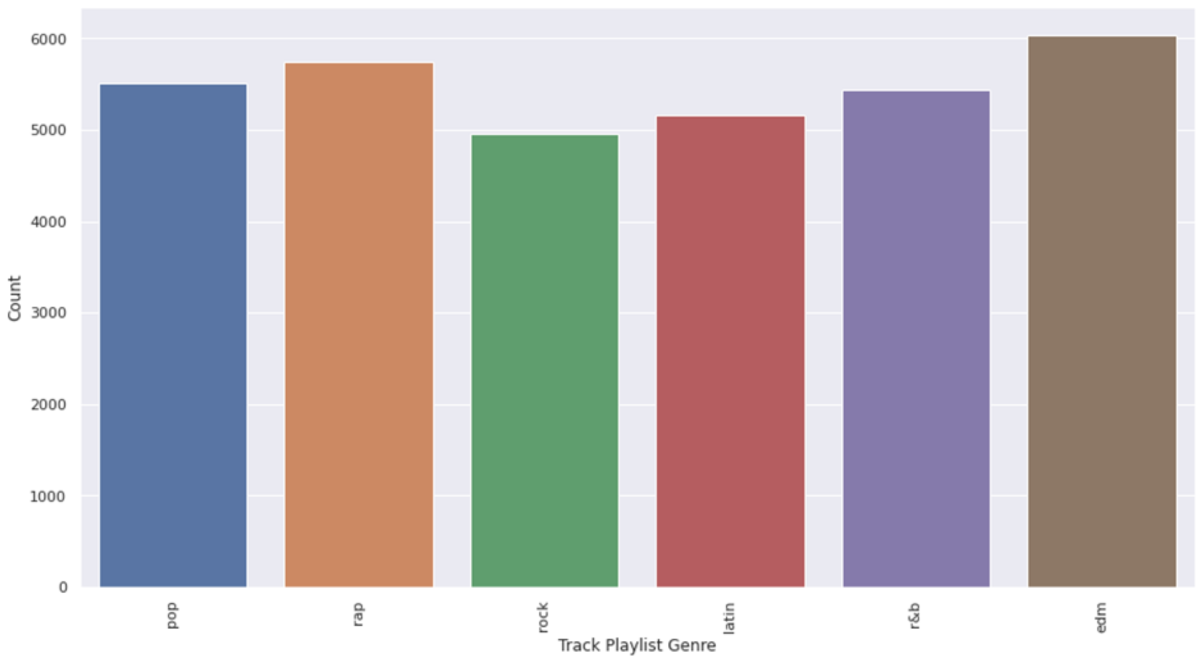


Density Plot interpretation: As we can see from the above plots, the distribution of the attributes across the data is highly skewed in the cases of attributes such as tempo, loudness, speechiness, acousticness and liveness whereas in case of valence, energy and danceability we see the distribution is somewhat normal.

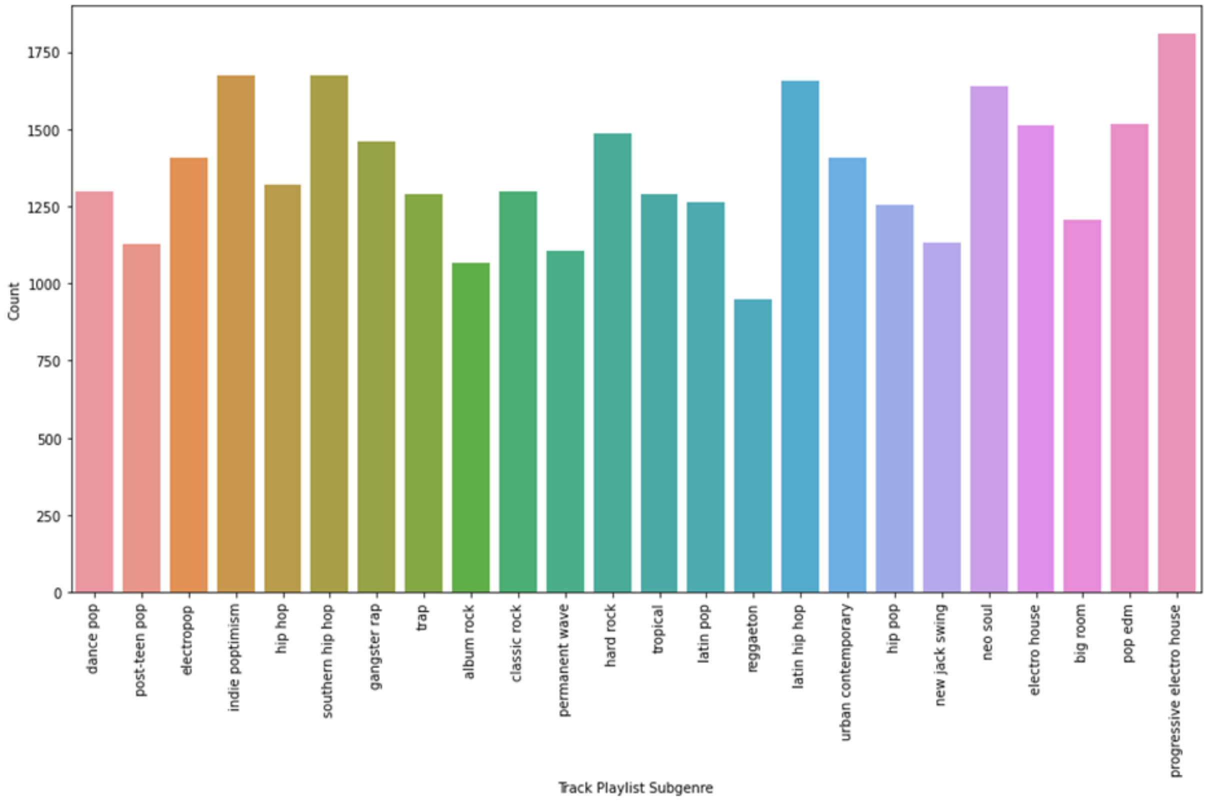


Boxplot Interpretation: In the figure above we can see the boxplots for all the attributes which have a scale of 0-1, here we can observe that the distribution has a few outliers and we'll address these outliers in the data preprocessing section below, before we start our clustering. Instrumentalness has most of the values close to 0 in our dataset.

Bar Graph Information: The songs in the dataset were assigned to pre-existing playlists, yielding six different playlist genres such as pop, rock, and rap, as well as 24 different playlist subgenres such as dance pop, hip hop, and gangster rap. The following plots show the frequency distribution of songs in the Playlist genres and subgenres, respectively. The dataset has a good distribution of songs overall, and there is no dominating genre/subgenre.

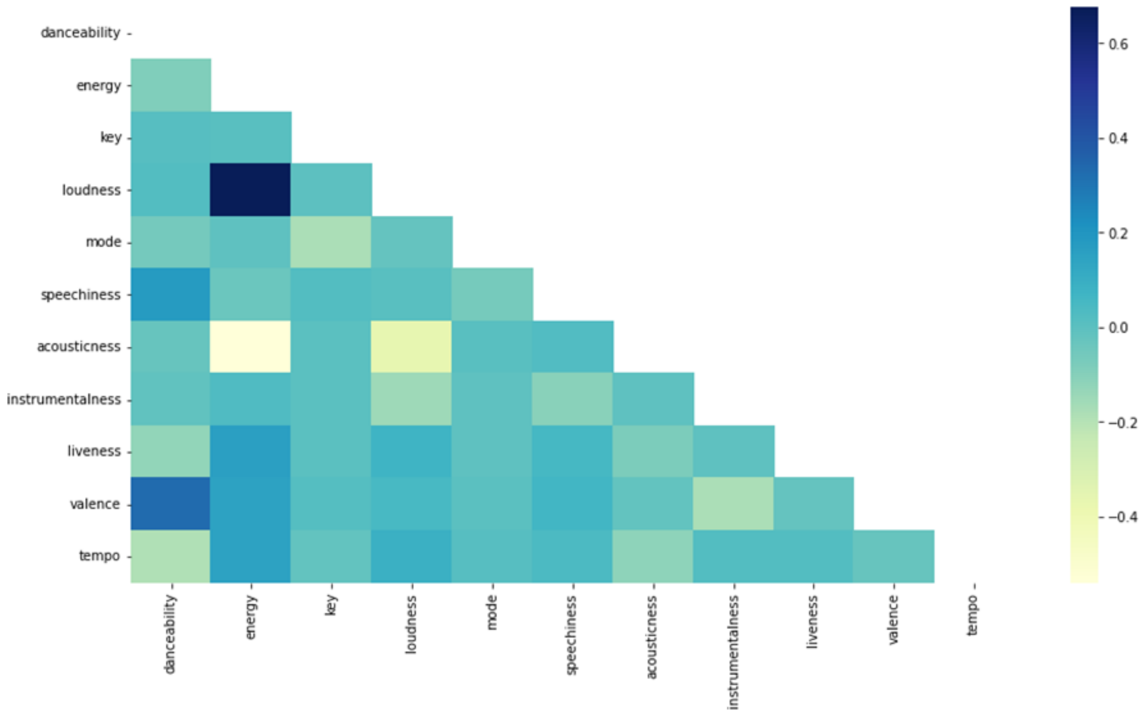


Count
6000
5000
4000
3000
2000
1000
0
pop
rap
rock
latin
r&b
edm
Track Playlist Genre



Count
1750
1500
1250
1000
750
500
250
0
dance pop
post-teen pop
electropop
indie poptimism
hip hop
southern hip hop
gangster rap
trap
album rock
classic rock
permanent wave
hard rock
tropical
latin pop
reggaeton
latin hip hop
urban contemporary
hip pop
new jack swing
neo soul
electro house
big room
pop edm
progressive electro house
Track Playlist Subgenre

Correlation Heatmap analysis and interpretation: We evaluated the correlations between different audio features that were used for clustering. Except for loudness and energy(0.68), which have a slight correlation, there is no major association among the audio features used in the analysis. Valence and danceability also have some correlation, but overall we do not have highly correlated features. Because our research is based on a clustering technique that focuses on distance metrics, this will not be an issue.



Why Data Science?

Spotify uses an extremely well-built recommendation system and playlist generation option. To create mood-based playlist from audio features of songs having a variety of individual ranges, Data Science can be leveraged in data pre-processing and cluster building.

1. Pre-processing of Data - The units and sizes of the features in the dataset vary. We can use Min-Max Scaler to transform all of the features to the same dimension.

2. Grouping of similar songs – Because we need to create a mood playlist based on song similarity, we can use clustering algorithms such as K-means or Hierarchical to group similar songs together.

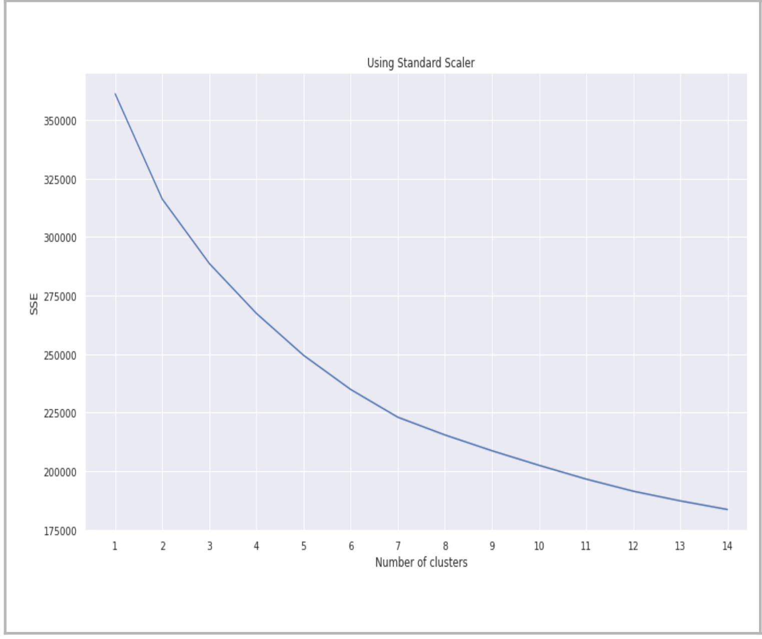
Models and Training

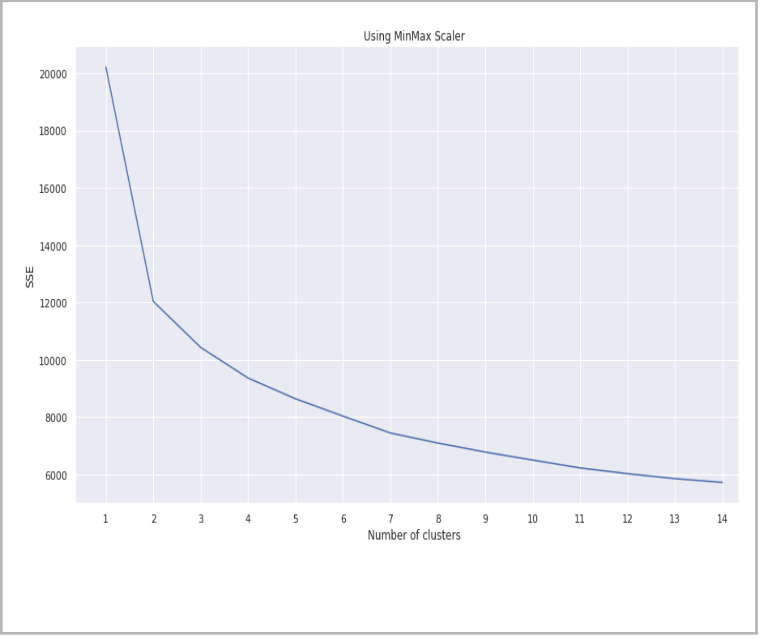
After studying our data and essential features using exploratory data analysis, we experimented with two different and most prominent clustering approaches, K-Means clustering and Hierarchical Clustering, since the goal of this study is to construct various groupings. After experimenting with these two techniques, we have chosen to train the model and build final clusters using the K-Means Clustering approach.

Data Preprocessing

It is critical to preprocess our data in order to improve the model's performance. Because k-Means distance computation equally weights each dimension, effort must be made to ensure that the unit of dimension does not misrepresent the relative closeness of observations.

After thorough testing, we discovered that the MinMax Scaler outperformed the Standard Scaler. The range of SSE, or Sum of Squared Error, when obtaining the best K value is substantially higher in the case of Standard Scaler than in the case of MinMax Scaler, as shown in the figures below.





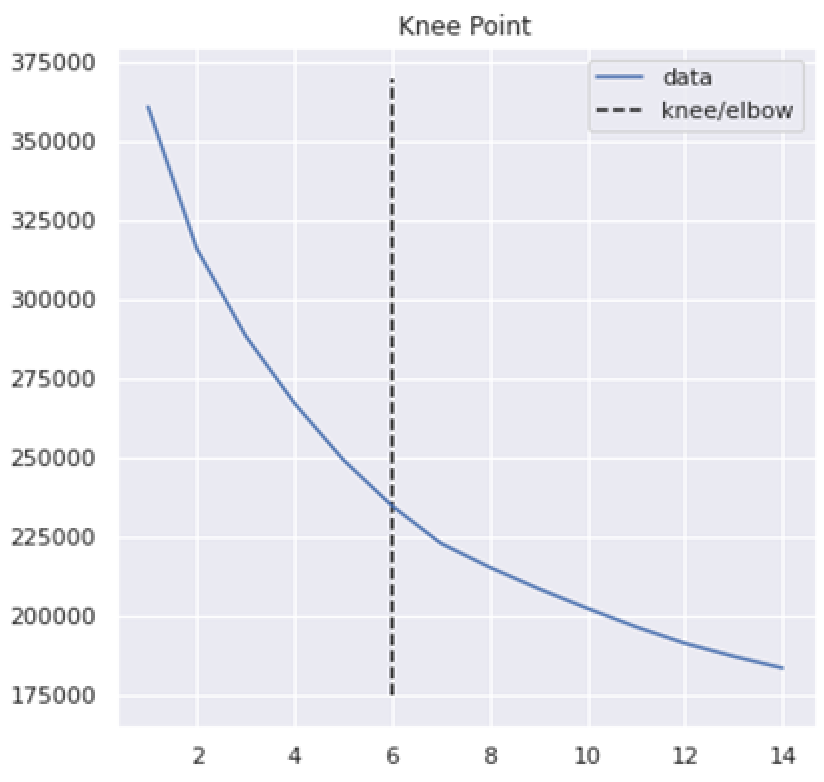
K-Means Clustering

K-Means assigns records to each cluster using a pre-specified number of clusters to discover the mutually exclusive spherical shape cluster based on distance. K-Means Clustering necessitates prior knowledge of K, i.e. the number of clusters into which we intend to divide our data.

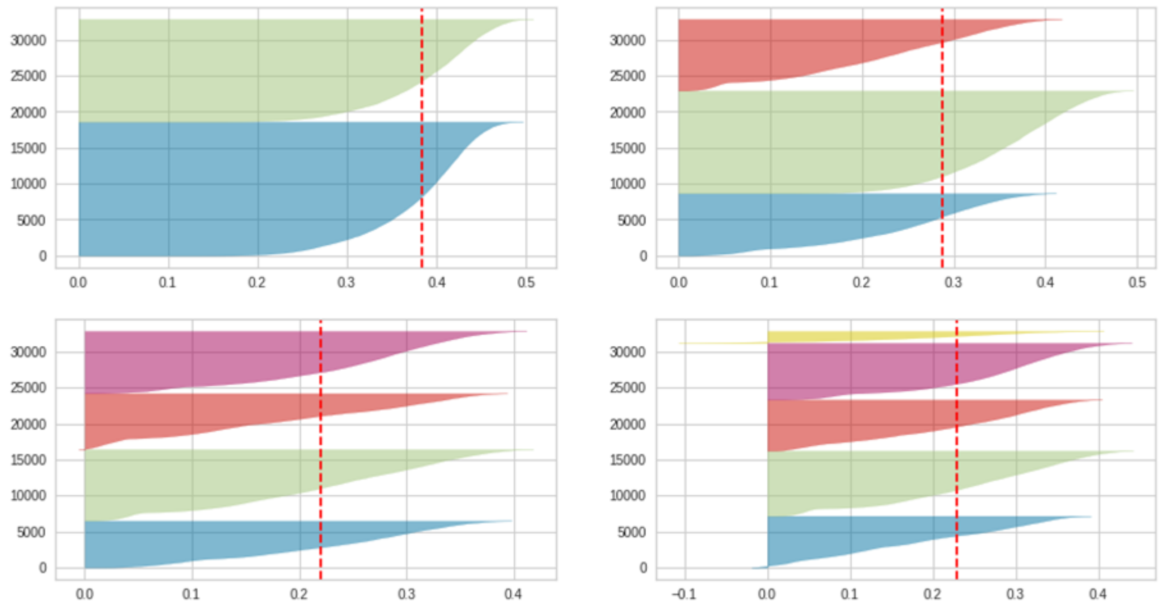
Finding the ideal number of clusters

Elbow/Knee method and Silhouette Score are one of the most popular ways to find the optimal and efficient number of clusters.

The elbow method is used to find the “elbow” point, where increasing the number of clusters does not improve the error too much. We can see from the below Elbow/Knee plot, 6 looks like an elbow point for the plot.



The silhouette score indicates if there are significant gaps between each sample and all other samples within or across clusters. A point's silhouette score indicates how close it is to its closest neighbor points across all clusters. Below are the Silhouette plots for different numbers of clusters ranging from 2 to 5 clusters.

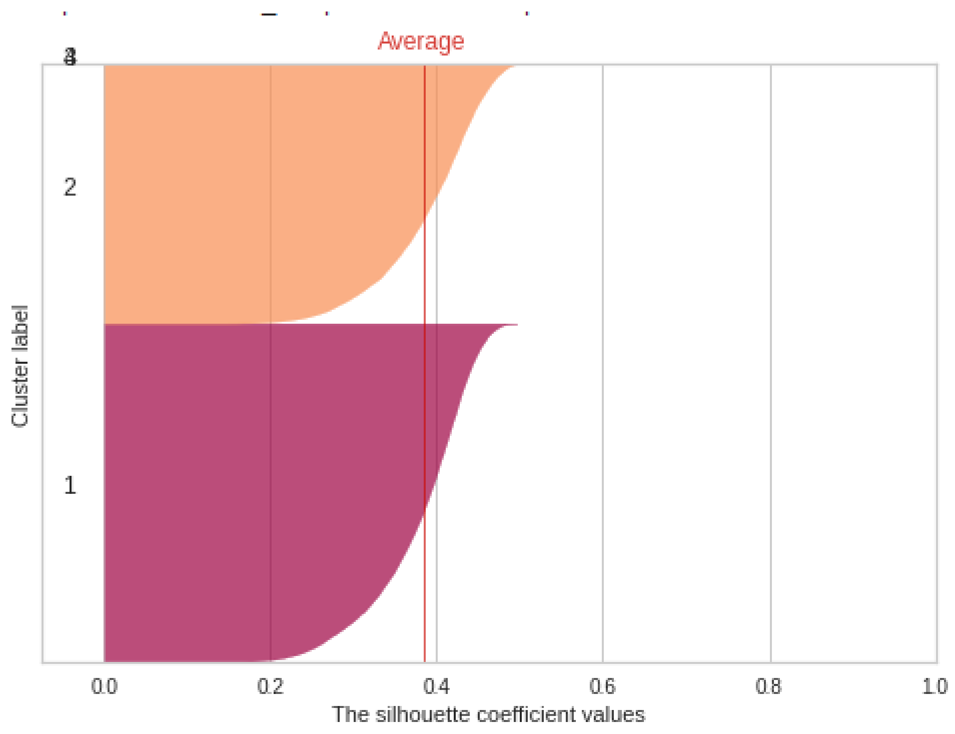


Using the concepts stated above, we initially created 4 clusters, but we decided against using this as our objective was creating mood-based clusters with a highly streamlined analysis to get focused playlists but using the aforementioned methods we could only create a few of them which lacked the exclusivity needed for our analysis.

Hierarchical Clustering

A hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters without having a fixed number of clusters. It is a set of nested clusters that are arranged as a tree.

Hence after implementing K-means method we tried to cluster the data by Hierarchical method but as shown in the plot below the result was sub par as we could only create two clear clusters with songs and the other two came out to be empty.



Cluster Number (K) Choice

From the clustering analysis mentioned above, the optimal number of clusters actually present in our data are around 2-4. If K is between this range, we will get a small number of clusters with a large amount of data.

But our use case for clustering is slightly different than trying to figure out the pattern present in the data. We need to make sure we get a variable playlist for many different moods, and not similar ones. Thus, we prefer to have a large number of clusters with relatively less amount of songs assigned to each cluster.

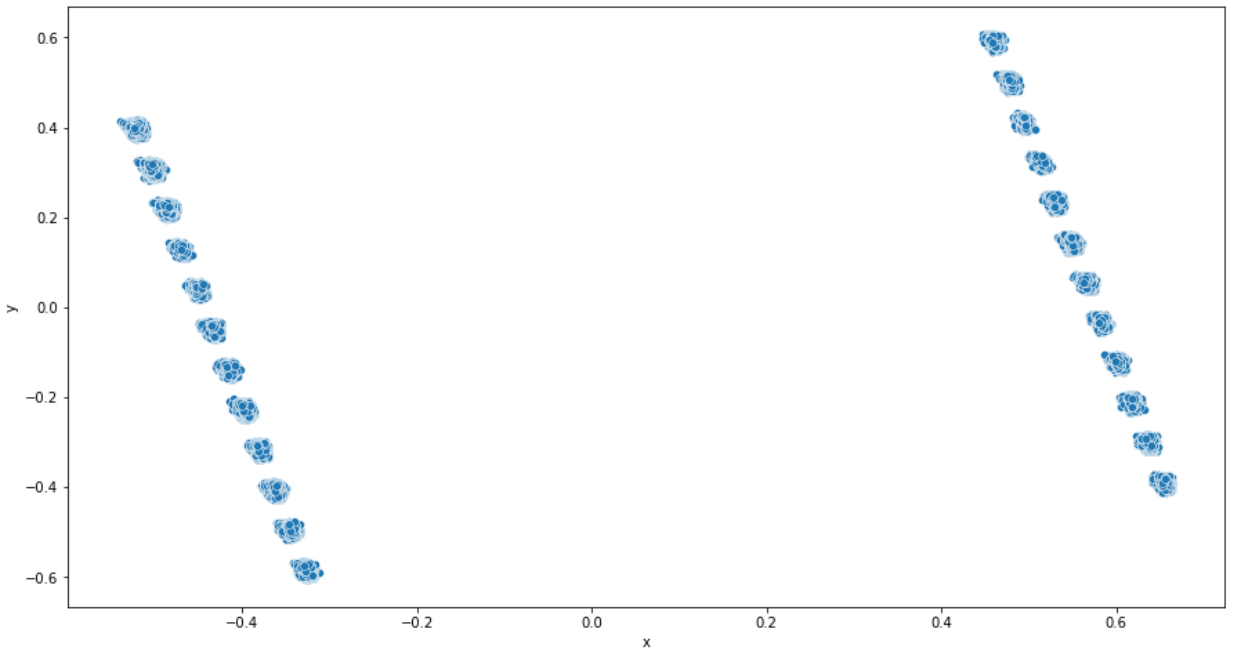
To get intuition of the data and the number of clusters we could use, we used PCA to view the data in 2 dimensions and determine the value for K.

Principal Component Analysis (PCA)

PCA (Principal Component Analysis) is a statistical technique for reducing data without sacrificing its features. In addition, the interpretation of main components might reveal correlations between variables that are not immediately apparent. It assists in the analysis of the scattering of observations and the identification of the variables responsible for distribution.

We have used PCA as a visualization technique, and not for the actual clustering process. By reducing the dimensions to 2, we can plot the data and see the presence of clusters in the data.

We have a small number of features, so we do not need to perform dimensionality reduction for the actual clustering.



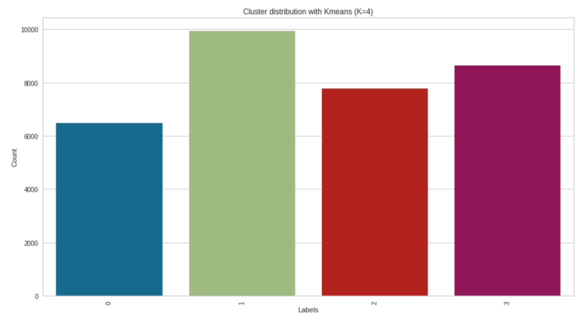
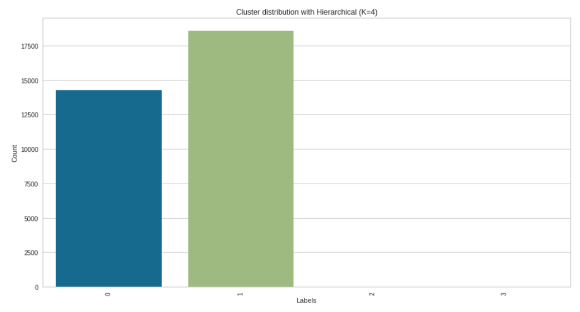
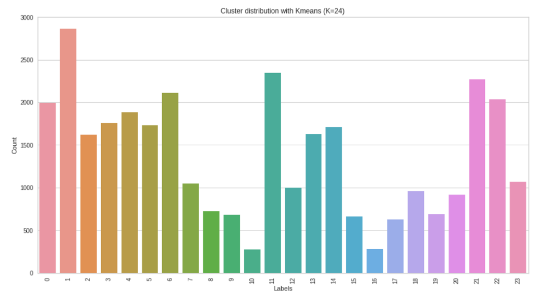
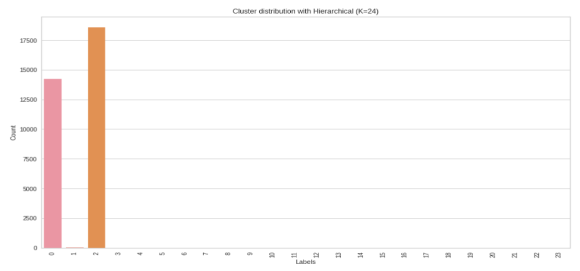
Scatter plot after PCA

Above plot shows the data in reduced 2-dimensions after performing PCA. We can clearly see, there are 2 main clusters present in the data, which we have already validated from the silhouette plot.

But on closer examination, we also see 24 smaller clusters in these clusters. Upon experimenting with more values for K and based on this observation, we decided to use K=24.

K-Means Clustering Vs Hierarchical Clustering

K-Means clustering is usually less computationally intensive and can handle huge datasets. Specialized to clusters of different sizes and shapes.

K-Means Clustering	Hierarchical Clustering
	
	

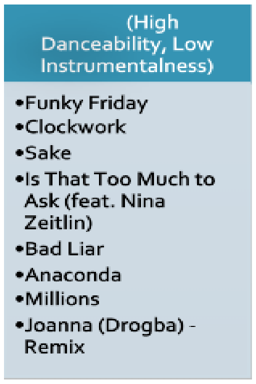
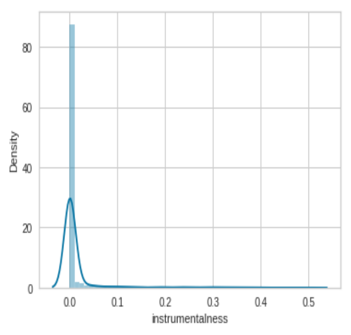
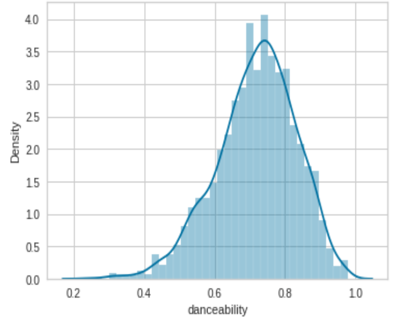
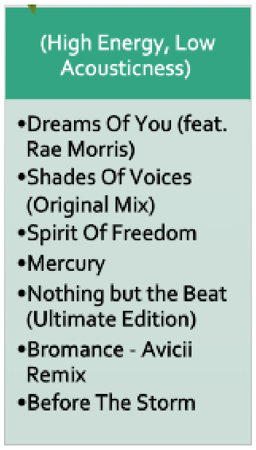
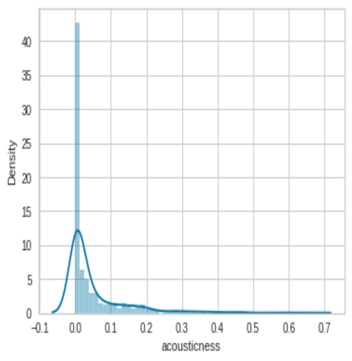
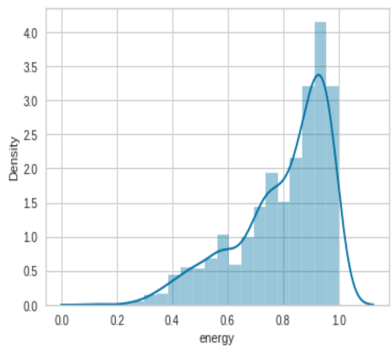
As shown in the comparison table above, we can see when we make four clusters using both the techniques we get 4 filled clusters from the K-means technique but in Hierarchical we only get two and then when we set our K=24 we again observe that using K-means all the 24 clusters have been formed with significant number of songs whereas we again only see 2 significant clusters using hierarchical method. Accordingly we choose K-means as our clustering technique for this analysis.

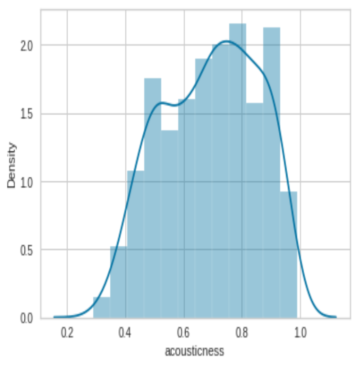
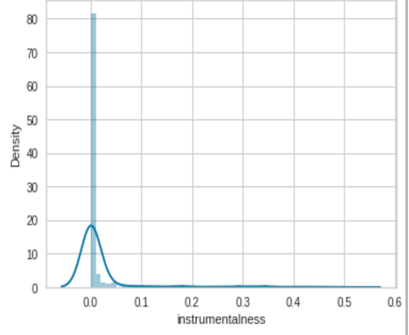
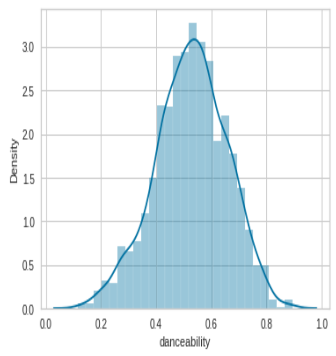
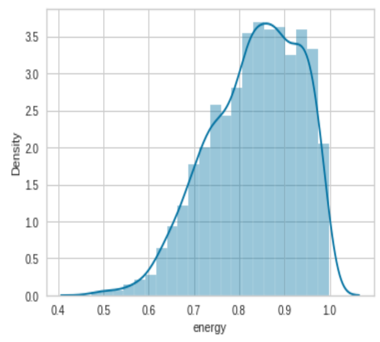
Note: In the plots shown above we have used linkage as “single” when we tried Hierarchical method but we tried other types of linkages as well; complete and average but the results were still dissatisfactory as when linkage was “average” there was no significant difference between the results obtained while performing “single” linkage Hierarchical Clustering and when linkage was “complete” we encountered a RAM insufficiency error as this method is computationally inefficient as compared to K-means.

Cluster Validation and Labeling

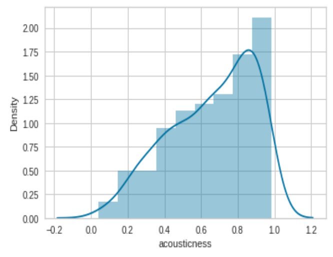
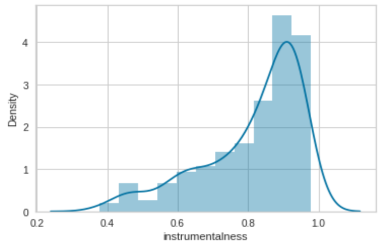
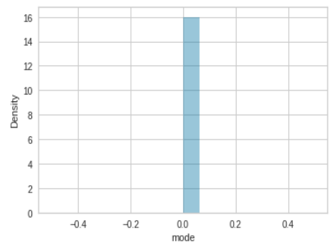
The clusters were validated and labeled after identifying the main features of the cluster and manually going through the songs. For example, the cluster shown in the first row of the table below has a high danceability and low instrumentality, and we recognized one of our favorite Nicki Minaj hits - "Anaconda" - and cross-referenced the other songs to see if they were similar to each other.

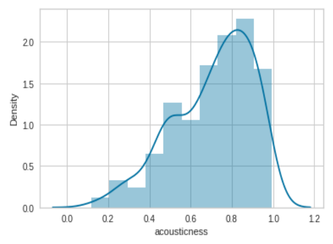
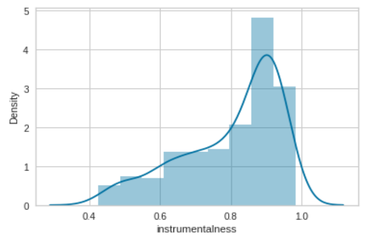
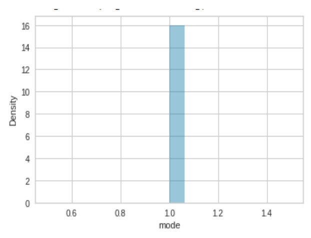
Sample Mood Cluster Snapshots

Cluster Songs	Distinctive features distribution		Explanation
 (High Danceability, Low Instrumentalness) •Funky Friday •Clockwork •Sake •Is That Too Much to Ask (feat. Nina Zeitlin) •Bad Liar •Anaconda •Millions •Joanna (Drogba) - Remix			In this cluster we can observe from the graph that the Danceability is high but the instrumentalness is low. For example- The Nicki Minaj song Anaconda has minimal use of instruments but has high danceability.
 (High Energy, Low Acousticness) •Dreams Of You (feat. Rae Morris) •Shades Of Voices (Original Mix) •Spirit Of Freedom •Mercury •Nothing but the Beat (Ultimate Edition) •Bromance - Avicii Remix •Before The Storm			In this cluster we can observe from the graph that the Energy is high but the Acousticness is low. Here we cross referenced the songs with one of the songs of well-known artist Avicii which we know has the features mentioned above.

(High Acousticness, Low Instrumentalness •New Shoes - feat.Shun Murakami •Memories •When You Wish Upon A Star •Gravity •Zebra •(No One Knows Me) Like the Piano •Someone Like You •Sweet Dreams			In this cluster we can observe from the graph that the Acousticness is high but the Instrumentalness is low.Here we choose the song memories which was one of most well known songs in this cluster for the purpose of manually validating the cluster.
(Moderate Danceability, High Energy) •Feed the Machine •Hardwired •Creep - The Four Performance •Spit Out the Bone •Always Forever •Animal •Legend •Mockingbird			In this cluster we can observe from the graph that the Danceability is relatively moderate h but the Energy is quite high. We used one of the hits of Maroon 5 Animals to cross reference the songs in this cluster.

Note: The clusters shown in the table below are the ones with the most similar songs amongst them which have high instrumentalness and high acousticness but they differ on the basis of Mode; as in one cluster the songs are the ones which have the minor notes and in the other one the songs have major notes.

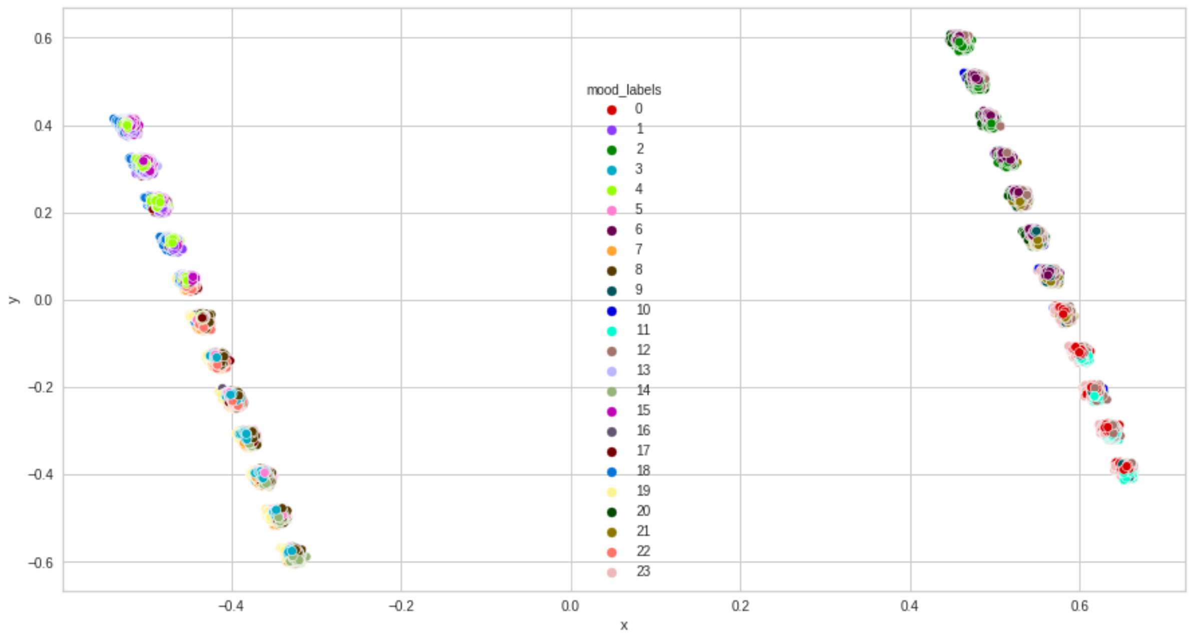
Cluster	**Acousticness**	**Instrumentalness**	**Mode**
10			

16			

Clustering Output

Now that we've completed our model, which used the K-Means clustering approach to build 24 clusters. Let's move on to forecasting new song clusters and creating playlists based on mood and obscurity.

Labels	Count	Labels	Count
0	1995	12	995
1	2860	13	1626
2	1618	14	1709
3	1758	15	658
4	1881	16	280
5	1730	17	626
6	2110	18	956
7	1048	19	686
8	719	20	914
9	681	21	2268
10	272	22	2037
11	2342	23	1064

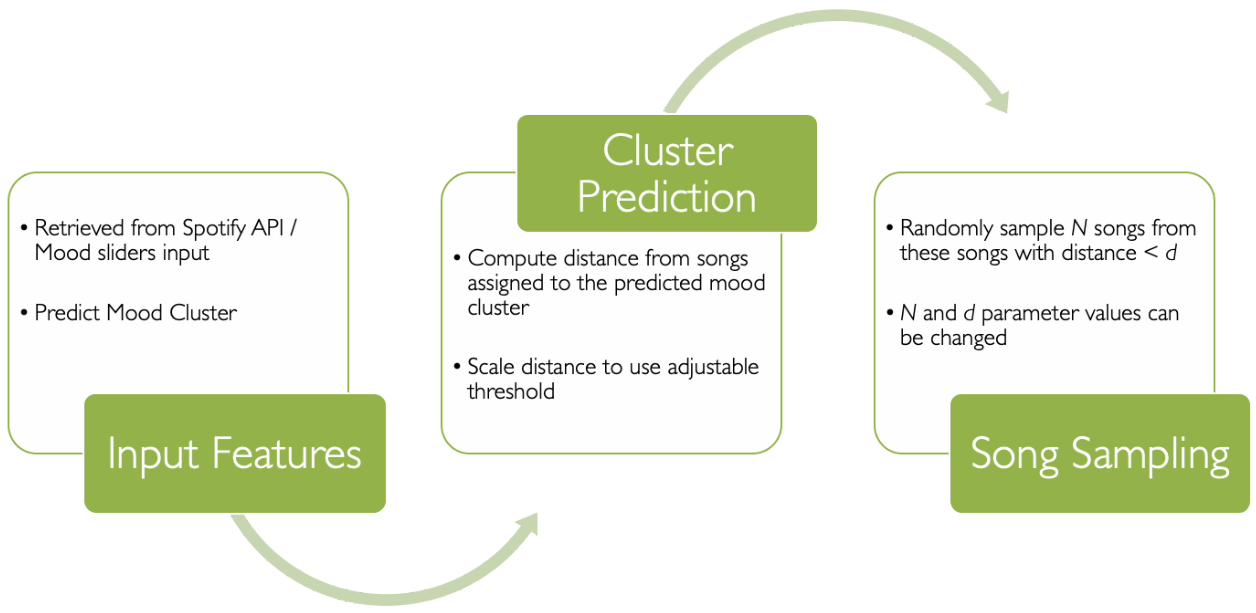


Playlist Generation

1. Normal Playlist

Song attributes will be retrieved via Spotify API or by manually inputting the values of attributes to create the playlist. The Model.predict() command will predict the new song's mood cluster. After the song cluster has been predicted, the algorithm will work on creating a playlist.

Mood Playlist creation algorithm works in the following manner:



Input Features
• Retrieved from Spotify API / Mood sliders input
• Predict Mood Cluster
Cluster Prediction
• Compute distance from songs assigned to the predicted mood cluster
• Scale distance to use adjustable threshold
Song Sampling
• Randomly sample *N* songs from these songs with distance < *d*
• *N* and *d* parameter values can be changed

Step 1 - Input Features: Predict mood cluster for user's input song. The audio features for this song are retrieved from Spotify API using track name and artist name, or taken as input from "mood sliders" in the UI

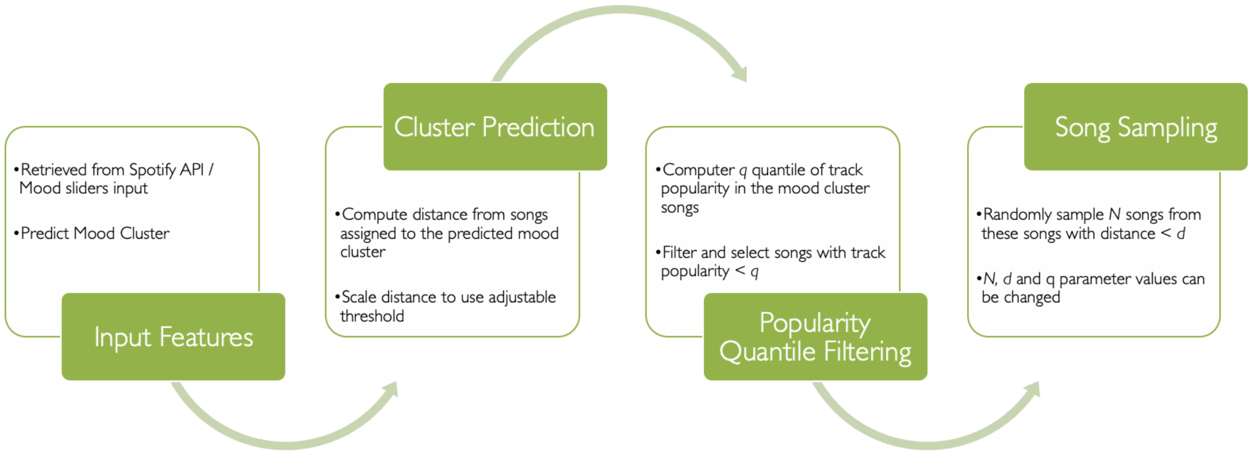
Step 2 - Cluster Prediction: Calculate the Euclidean distance of a new song from all songs in the predicted cluster. Normalize the distance using MinMax Scaler, so we can use threshold (0 - 1) as a parameter instead of absolute distance.

Step 3 - Song Sampling: Filter out only the nearest songs to the new song using an adjustable threshold value d. Randomly sample N number of songs to create the playlist

2. Obscure Playlist

If a user wants to listen to songs from the predicted cluster that are less popular, the algorithm takes a few extra steps to create a mood playlist with less popular songs.

Obscure Mood Playlist creation algorithm works in the following manner:



Input Features
•Retrieved from Spotify API / Mood sliders input
•Predict Mood Cluster
Cluster Prediction
•Compute distance from songs assigned to the predicted mood cluster
•Scale distance to use adjustable threshold
Popularity Quantile Filtering
•Computer *q* quantile of track popularity in the mood cluster songs
•Filter and select songs with track popularity < *q*
Song Sampling
•Randomly sample *N* songs from these songs with distance < *d*
•*N*, *d* and q parameter values can be changed

Step 1 - Input Features: Predict mood cluster for user's input song. The audio features for this song are retrieved from Spotify API using track name and artist name, or taken as input from "mood sliders" in the UI

Step 2 - Cluster Prediction: Calculate the Euclidean distance of a new song from all songs in the predicted cluster. Normalize the distance using MinMax Scaler, so we can use threshold (0 - 1) as a parameter instead of absolute distance

Step 3 - Popularity Quantile Filtering: Compute *q*th quantiles of track popularity among the predicted cluster. Filter out the songs with track popularity less than a threshold value of quantile (q)

Step 4 - Song Sampling: Randomly sample N number of songs to create the playlist

Service

We have created a Streamlit application that requires minimal user input in order to maintain a simple and user-friendly user interface. We used shades of green for our interface to match Spotify's theme. We load our pre-trained clustering model on starting the service. The following section explains the key features of our application.

Input Types

The user can specify the track name and artist name of the input song, which will be used to retrieve the input audio features from the Spotify API. This is the simplest way to interact with our service. Alternatively, if the user simply wants to experiment with "mood sliders," he can do so to generate a mood playlist. The user must select a checkbox to activate the mood sliders. To avoid overwhelming the user with too many input options, the sliders are only displayed if this checkbox is selected.

Obscure Playlist Input

A checkbox can be selected as a unique feature to help the user discover songs and artists that are less well-known. If this option is chosen, we will return an "obscure playlist" from our quantile-filtered set of songs.

Multiple Shuffled Playlists Feature

The UI provides a “Moodify!” button, that can be clicked multiple times to get different shuffled playlists each time.

Closest Song Display Feature

This section is activated when the user selects to provide input using mood sliders. If the user is unsure what the various sliders represent, he can experiment with them to see which song best represents the input. This makes it easier for the user to use the service.

Chosen Mood Attributes Feature

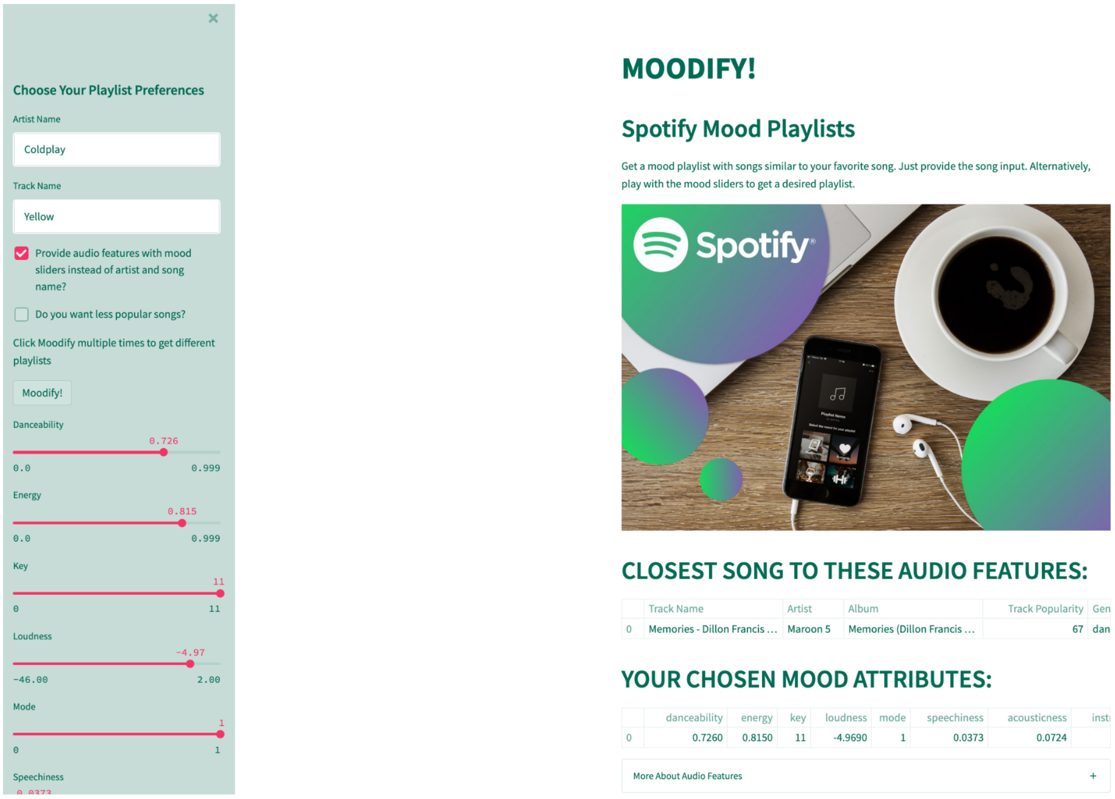
We display the audio attributes selected by the user here. This can be either retrieved from Spotify API or from the mood sliders.

Expandable Tabs

To provide more information to users, we have used two expandable tabs.

One tab explains what each mood feature stands for using examples and ranges. The user can utilize this tab to understand each feature.

Second tab displays a snapshot of the data used for clustering.



Choose Your Playlist Preferences

Artist Name: Coldplay

Track Name: Yellow

[✓] Provide audio features with mood sliders instead of artist and song name?

[ ] Do you want less popular songs?

Click Moodify multiple times to get different playlists

Moodify!

Danceability: 0.726 (0.0 – 0.999)

Energy: 0.815 (0.0 – 0.999)

Key: 11 (0 – 11)

Loudness: -4.97 (-46.00 – 2.00)

Mode: 1 (0 – 1)

Speechiness: 0.0373

MOODIFY!

Spotify Mood Playlists

Get a mood playlist with songs similar to your favorite song. Just provide the song input. Alternatively, play with the mood sliders to get a desired playlist.

CLOSEST SONG TO THESE AUDIO FEATURES:

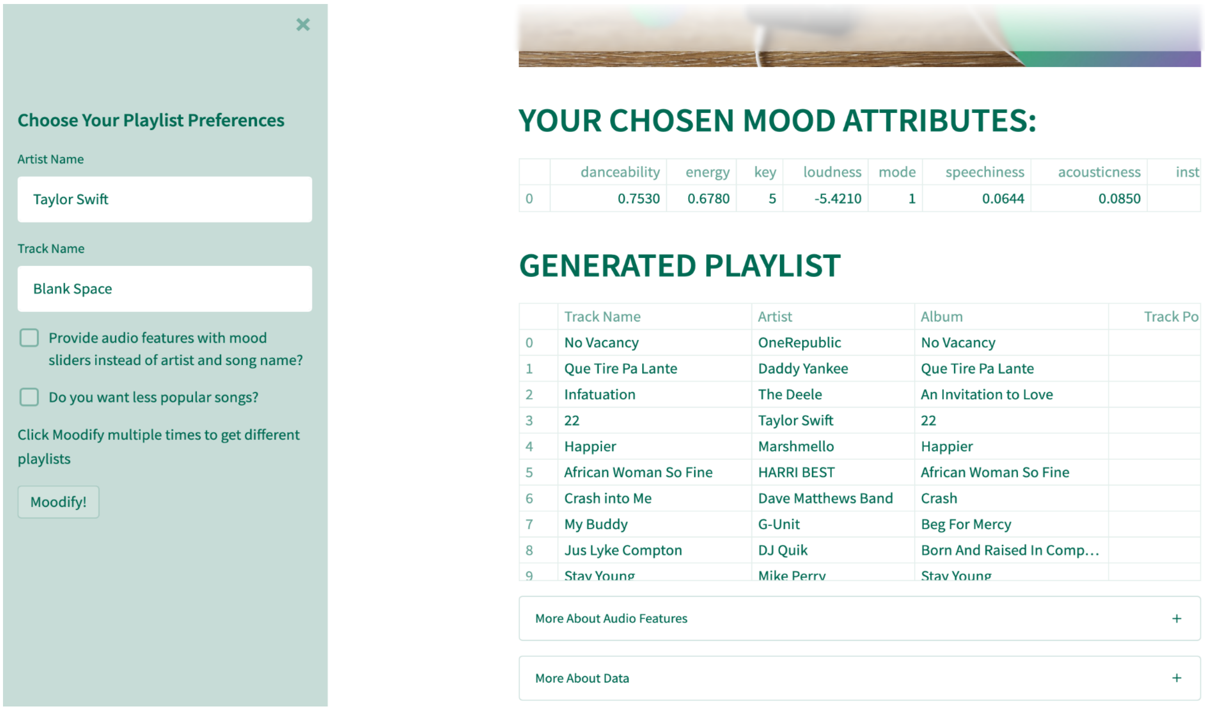
	Track Name	Artist	Album	Track Popularity	Gen
0	Memories - Dillon Francis …	Maroon 5	Memories (Dillon Francis …	67	dan

YOUR CHOSEN MOOD ATTRIBUTES:

	danceability	energy	key	loudness	mode	speechiness	acousticness	inst
0	0.7260	0.8150	11	-4.9690	1	0.0373	0.0724	

More About Audio Features +

Snapshot of application



Choose Your Playlist Preferences

Artist Name: Taylor Swift

Track Name: Blank Space

[ ] Provide audio features with mood sliders instead of artist and song name?

[ ] Do you want less popular songs?

Click Moodify multiple times to get different playlists

Moodify!

YOUR CHOSEN MOOD ATTRIBUTES:

	danceability	energy	key	loudness	mode	speechiness	acousticness	inst
0	0.7530	0.6780	5	-5.4210	1	0.0644	0.0850	

GENERATED PLAYLIST

	Track Name	Artist	Album	Track Po
0	No Vacancy	OneRepublic	No Vacancy	
1	Que Tire Pa Lante	Daddy Yankee	Que Tire Pa Lante	
2	Infatuation	The Deele	An Invitation to Love	
3	22	Taylor Swift	22	
4	Happier	Marshmello	Happier	
5	African Woman So Fine	HARRI BEST	African Woman So Fine	
6	Crash into Me	Dave Matthews Band	Crash	
7	My Buddy	G-Unit	Beg For Mercy	
8	Jus Lyke Compton	DJ Quik	Born And Raised In Comp…	
9	Stay Young	Mike Perry	Stay Young	

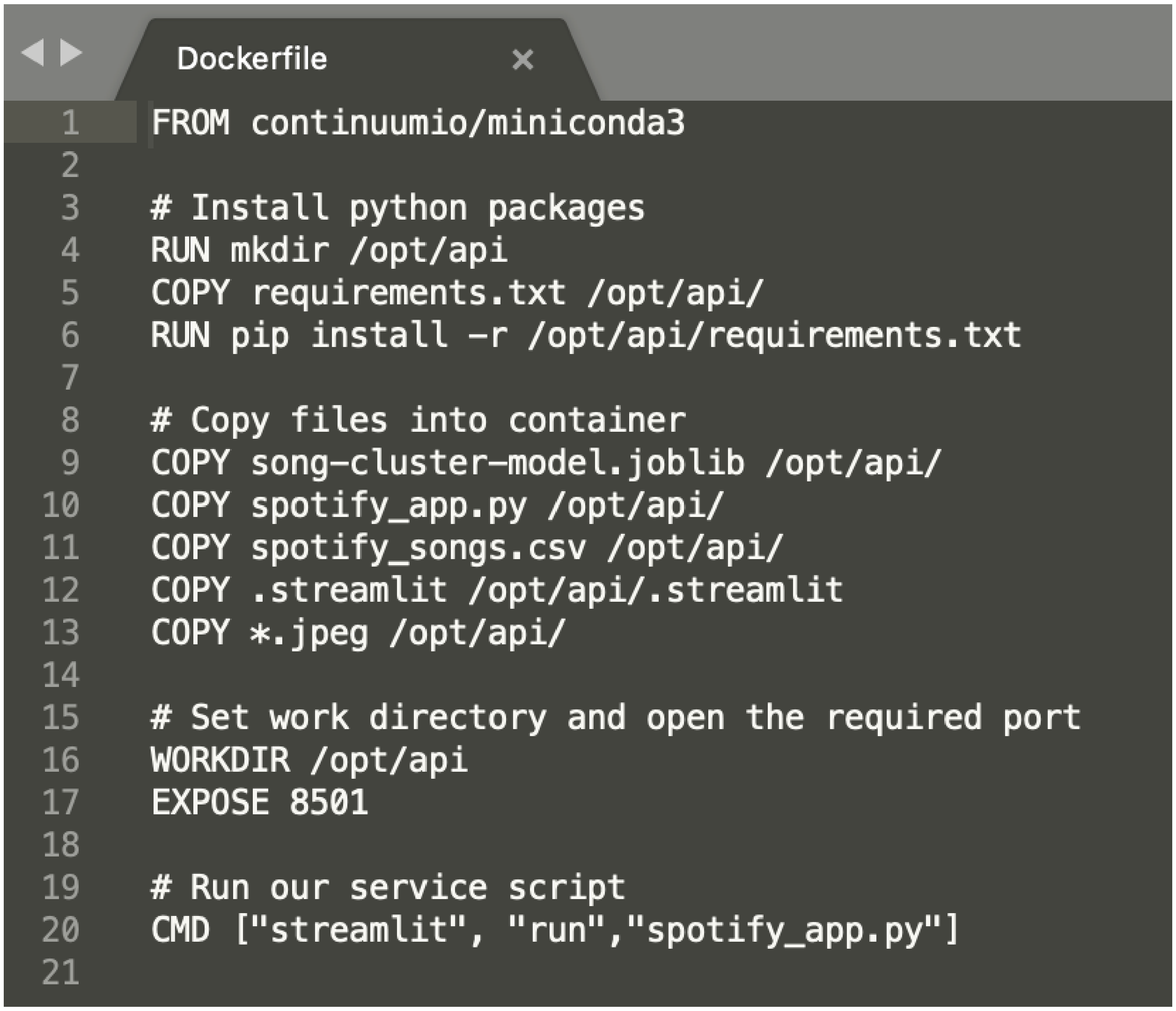
More About Audio Features +

More About Data +

In the screenshot, we see a playlist that matches the mood of “Blank Space by Taylor Swift”. Blank space has a very high danceability with an upbeat mood. “No Vacancy by OneRepublic” has a very similar mood. The playlist also has “22 by Taylor Swift”. Another song that stands out to us is “Happier by Marshmello”. All of these songs have very similar sounding beats patterns.

Docker:

We have created a Docker image of our application to ensure portability and flexible sharing and have used the miniconda3 base image from dockerhub. Below is a snapshot of the dockerfile.



```
FROM continuumio/miniconda3

# Install python packages
RUN mkdir /opt/api
COPY requirements.txt /opt/api/
RUN pip install -r /opt/api/requirements.txt

# Copy files into container
COPY song-cluster-model.joblib /opt/api/
COPY spotify_app.py /opt/api/
COPY spotify_songs.csv /opt/api/
COPY .streamlit /opt/api/.streamlit
COPY *.jpeg /opt/api/

# Set work directory and open the required port
WORKDIR /opt/api
EXPOSE 8501

# Run our service script
CMD ["streamlit", "run","spotify_app.py"]

```

Conclusion & Future Scope

This project gave us a golden opportunity to work on a live API to solve the problem we as music listeners have encountered countless times, be it during house-parties or taking a road trip with friends and family. Playlist creation has always been a menace, we think this project of ours is a small step to make this menace disappear and help people enjoy the music they crave for.

1. Things we learned

Working on this project allowed us to learn more about unsupervised machine learning techniques such as clustering and PCA and how they can be applied to an interesting problem statement. Few of the key things we learned are:

- Working with a dataset containing 30K+ data points
- Exploratory data analysis techniques to understand more about data
- Exposure to data collection from the live Spotify API. This could be used to collect more data in the future to generate a wider range of mood playlists
- Working of K-means algorithm and Hierarchical algorithm in real world data grouping
- Importance of business case in terms of choosing final parameters
- Creating a Streamlit application with a trained machine learning model that users can easily interact with
- Creating a Docker image for our application to allow for flexibility and portability.

2. Challenges

A very significant aspect of this project is determining K, and widely used methods such as the elbow method and silhouette score/plot were ineffective in suggesting an optimal value of K due to the use case of our problem statement. We were able to overcome this challenge by using intuition from the feature reduced scatter plot as well as manual experiments.

Another challenge was validating the goodness of the clusters that were created, which required manually verifying whether or not the songs assigned to clusters were similar. We also investigated some statistical features and distribution plots for various clusters in order to validate the clusters generated and make more sense of them.

3. Future Enhancements

There are numerous avenues for this project that can be explored in the future. Some of the most significant improvements that can be made on this application include leaving the playlist size selection to users in order to create user-friendly customized playlists that can be used for workout sessions, parties, or even weddings, thus leaving the option "duration of the playlist" open to users by selecting the number of songs they want in the playlist.

To improve the model, we could include more features such as the genres with which the song artist is typically associated, the language of the song, and so on.

Next, we can combine data from different regions to create a mixed playlist that includes songs in multiple languages. For example, we could create an anglo-hispanic playlist for people who enjoy upbeat Spanish songs with English adaptations, such as Despacito.

Furthermore, by providing different ranges of track popularity, we can improve the user interface of this application by giving users the option of editing the thresholds so that they can select how streamlined playlists they want to make and what kind of artists they want to discover.

References:

1. Web API Reference | Spotify for Developers

2. tidytuesday/data/2020/2020-01-21 at master · rfordatascience/tidytuesday · GitHub

3. Gallery • Streamlit

4. sklearn.cluster.AgglomerativeClustering — scikit-learn 1.0.2 documentation

5. sklearn.cluster.KMeans — scikit-learn 1.0.2 documentation