Big Data Analytics

Project Final Report

Spotify - Trend Analysis and Recommendation System

Rajendran Karthick Sharan

Problem Setting- Motivation and Project Overview

- **Motivation**

  Spotify is a proprietary Swedish audio streaming and media services provider which changed the landscape of the music industry and facilitated the pursuit of a musical career as a viable option for budding artists throughout the world. With 320 million monthly users and 4 billion playlists, Spotify is one of the most popular music and content streaming platforms in existence. With the growth of technology, increasing internet connectivity in even the most remote parts of the globe, and ongoing creative explosions, this number is only expected to increase significantly.

  Much of Spotify's success has been fueled by its data collection and analytics expertise. Several of their Data Analytics features are praised by several customers and proponents of Spotify, such as the Spotify Wrapped, the personalized slideshow that Spotify releases each year, which provides an insight into your favorite artists, musicians, songs and genres over the previous year. While this is but a simple example of the power of real time analytics used by Spotify, it is just the beginning. By collecting and analyzing massive amounts of real-time listener data, Spotify can identify emerging user trends in real-time, and rapidly develop new services and features to capitalize on them.

  This is a win-win for both Spotify and its customers as it helps Spotify gain an edge over other Music Streaming services, and it helps the customers by making Spotify a service that gives its customers a unique musical experience that is tailored to their needs and preferences.

- **Project Overview**

  In the scope of this project, we aim to use a Spotify Developer account and extract a Spotify Dataset, which we will use to develop Machine learning models to analyze, identify and predict the trends behind the lyrics and attributes of a song to improve personalization and generate accurate suggestive algorithms for customers.

During the course of this analysis, we will also provide inferences on whether having certain sentiments or attributes gives a song an edge over others, essentially, classifying various attributes in terms of its importance associated with the popularity/success of a song.

Finally, after the data preprocessing, analysis and model building, we will develop a recommendation system which suggests songs closest to the mood/sentiment conveyed by a track. This system will be able to receive multiple user inputs, which it then processes to generate a similar output. We believe that this system could be of great value to Spotify, and by extension, its customers.

Data Description

- Our working dataset has 200,000 rows which we selected using bootstrap sampling from the 1 million extracted playlists.
- The Dataset has 23 columns, with most of them being numerical, with just two categorical variables of interest- 'Genre' and 'time_signature'.
- We dropped some unnecessary columns such as 'track_id', 'audio_features' during our exploratory data analysis.
- For Trend analysis, we use the 'popularity' score as the target variable, and we use the other attributes such as 'speechiness', 'valence' etc.

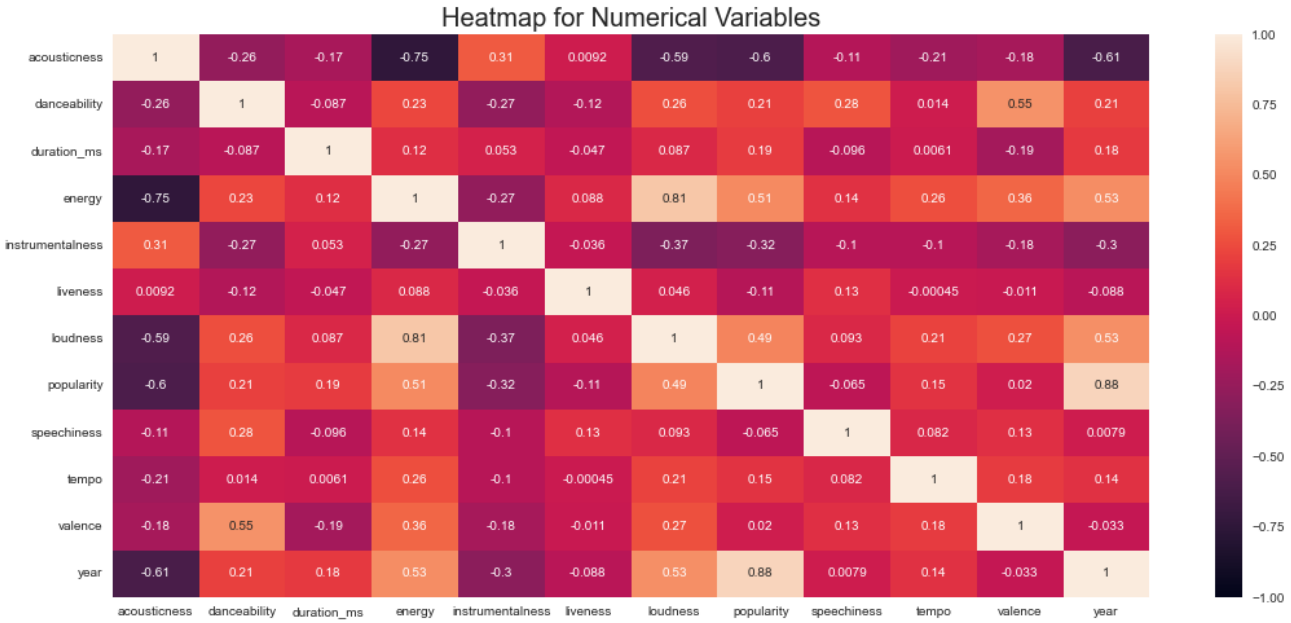
Techniques Used

Through the course of this project, we have used a variety of techniques for different purposes. In order to maintain a logical framework, we have divided the project into multiple components, and will talk about the techniques used in each of the components.

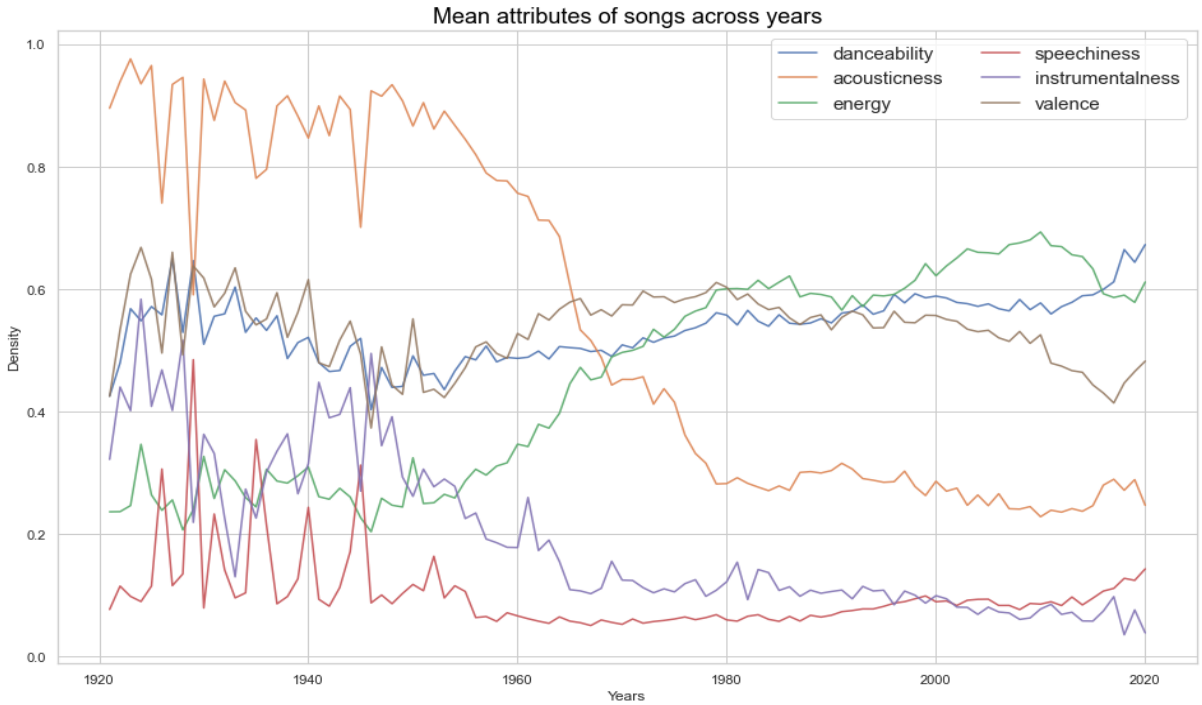
- **Data Extraction**
    1. The first step involved extracting an official dataset from Spotify. We did this using a developer account and the Spotify API.
    2. Once we get the developer credentials (client id, secret), we can use those to authenticate the session and create a 'Spotify' object, which we use to analyze the tracks.
    3. We use a track's URI (Uniform Resource Identifiers) to read the song's attributes.
    4. We extracted 1000 .json files, each containing 1000 playlists, with playlist and track information in each .json file.
    5. We used bootstrap sampling to pick 200k songs from the 1 million, to reduce our data scope.
    6. We extracted the tracks and metadata using code.
    7. We used the "audio_features()" and "track()" function of the Spotipy package to retrieve track attributes. (Please note: The data extraction and cleaning part has been explained in detail along with screenshots in midterm report, hence they were omitted here to keep the report with the page limit)

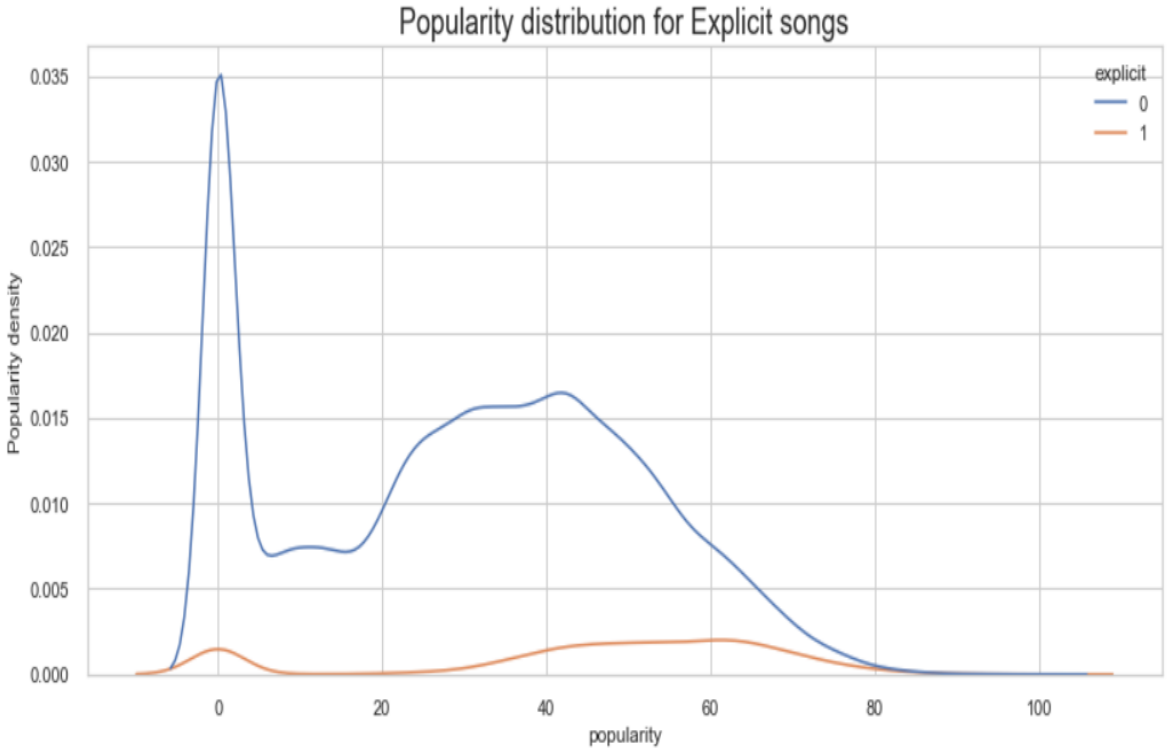
➢ **Exploratory Data Analysis**

1. We used “matplotlib” and “seaborn” packages to illustrate our analyses during the EDA.
2. We used Countplots and Kernel Estimation and Distribution plots to visualize our categorical and numerical variables, with respect to the target variable.
3. Multicollinearity for numerical variables were checked using heatmaps and VIF.
4. We used superimposed line plots to visualize the trend of mean/median attributes of the songs over the years.
5. Checked distribution of target for various categorical variables



feature	VIF
acousticness	2.929021
danceability	1.916783
duration_ms	1.116869
energy	5.373507
instrumentalness	1.292138
liveness	1.093536
loudness	3.369192
popularity	4.815591
speechiness	1.214077
tempo	1.118318
valence	2.056626
year	5.119900





Variable	skew
acousticness	-0.068277
danceability	-0.034222
duration_ms	0.020586
energy	-0.035967
instrumentalness	1.163562
liveness	0.327066
loudness	0.010805
popularity	-0.469333
speechiness	0.950062
tempo	0.087510
valence	-0.076540
year	-0.030269

- **Data Pre-processing**
  1. We used Power Transformations to treat the variables that were skewed.
  2. Standard scalar was used to scale the data for linear models.

- **Model Building**
  1. After Pre-processing and Train-test split, 3 models were built:
     - KNN Regressor
     - Random Forest Regressor
     - Gradient Boost Regressor
  2. We chose a combination of high R2 score and low RMSW, MAPE scores as the relevant metric to choose the best model, hence RF was selected as the best model.
  3. Important features were retrieved for the best model.

- **Recommendation System**
  1. As there are literally thousands of available genres, we clustered the available sub-genres into 9 main genres.
  2. Based on the above classification of genres and the song attributes, t-SNE was used for clustering songs into 10 clusters.
  3. We also reduced the dimensions of the attributes using Principal Component Analysis (PCA), and silhouette score and plot was used to evaluated best number of clusters.
  4. Even though k=2 was giving the best silhouette score, we decided to stick with 10 clusters for songs as that'd be more helpful for our recommendation system.
  5. Then, we developed a recommendation system to get the closest matching song with respect to song attributes and clusters.

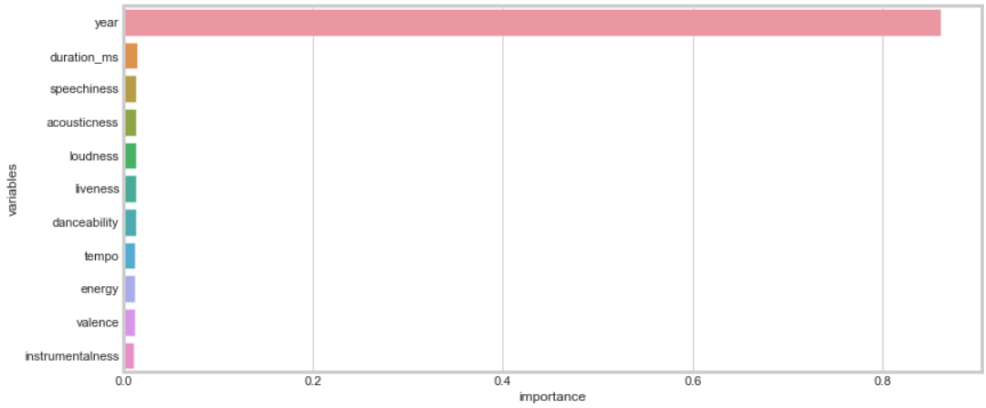
Results

Based on the 3 Supervised models that we developed, we established that the best model must be selected based on the combination of a high R2 score, and a low RMSW and MAPE score.

	Model	R2 Score	RMSE	MAPE
0	KNN Regressor	0.781530	0.467054	1.471722
1	Random Forest Regressor	0.866156	0.365570	1.187709
2	Gradient Boost Regressor	0.860244	0.373557	1.194587

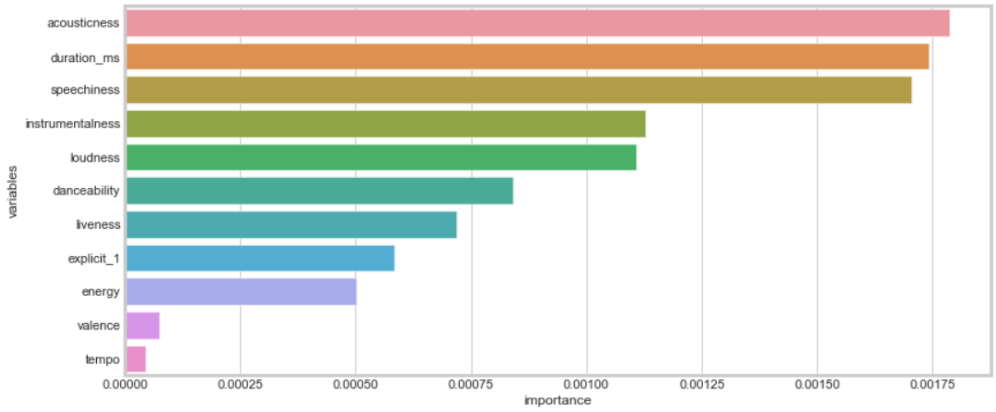
As we can see from the output above, the Random Forest Regressor Model fits those criteria, and hence, it can be concluded that the Random Forest Regressor is the best model for predicting the popularity of a song, based on our dataset.

We have also derived other inferences based on the model and the data analysis.



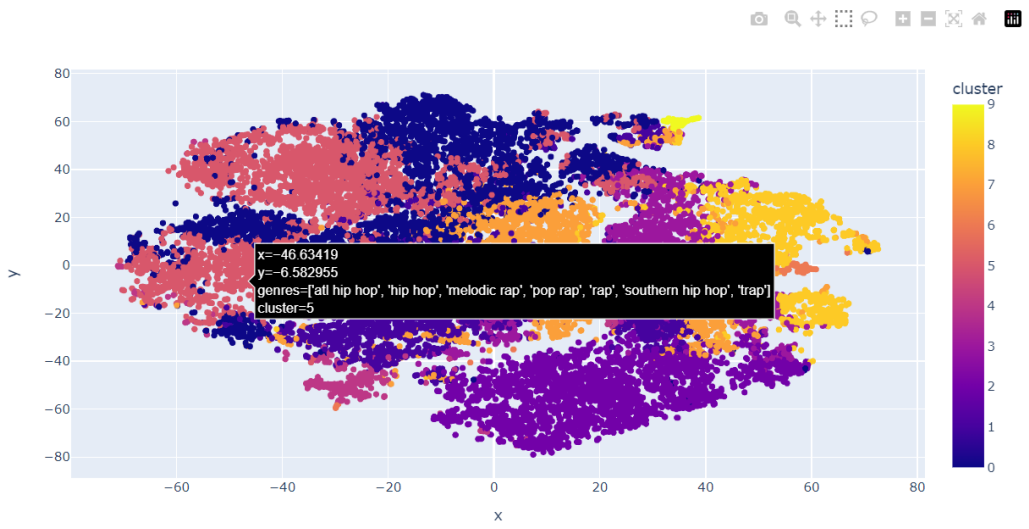
It can be observed from the graph above that the variable 'Year' has a high impact on a song's popularity. However, this is natural and to be expected as a higher population of newer songs are more likely to be popular than older songs.

Hence, we would like to observe the importance of the other variables without the 'Year'. The next most important variables in determining a songs popularity can be visualized as shown in the figure above.



As we move on to building the recommendation system, we use Clustering using K-Means to cluster the available 1000+ genres into 9 main categories of genres.

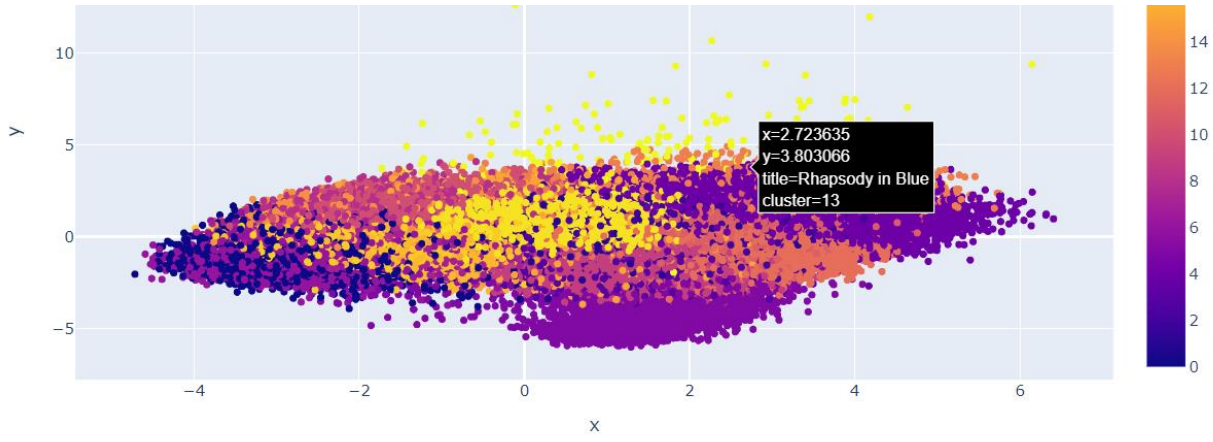
These genres are hip hop, rock, indie, jazz, classical/opera, house, etc. Below, is an interactive plot of the various clusters formed using t-SNE.



After this before moving on to clustering the songs, we decided to reduce the dimensions first. Principal Component Analysis (PCA) was used to achieve this. Below is a snapshot of the code used to perform PCA.

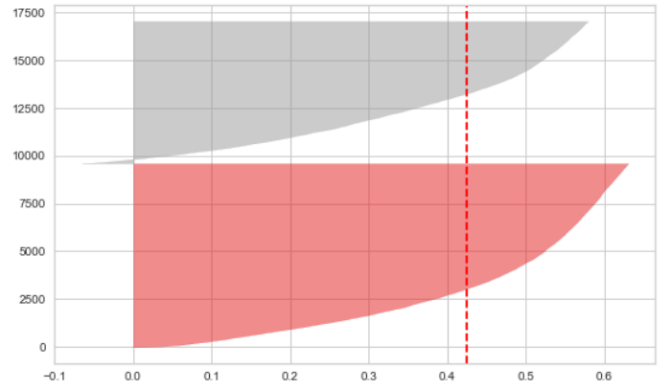
```
#Dimensionality Reduction with PCA

from sklearn.decomposition import PCA

#Implementing PCA
pipeline_PCA = Pipeline([('scaler', StandardScaler()), ('PCA', PCA(n_components=2))])
song_insertion = pipeline_PCA.fit_transform(X)
pca_data = pd.DataFrame(columns=['x', 'y'], data=song_insertion)
pca_data['title'] = data['name']
pca_data['cluster'] = data['labels_cluster']

#Plotting Inetractive plot after PCA
fig = px.scatter(pca_data, x='x', y='y', color='cluster', hover_data=['x', 'y', 'title'])
fig.show()
fig.write_image('song_clusters.png')
```

Now that the code for PCA is complete, let us take a look at the clusters of songs with reduced dimensions (x and y), below.

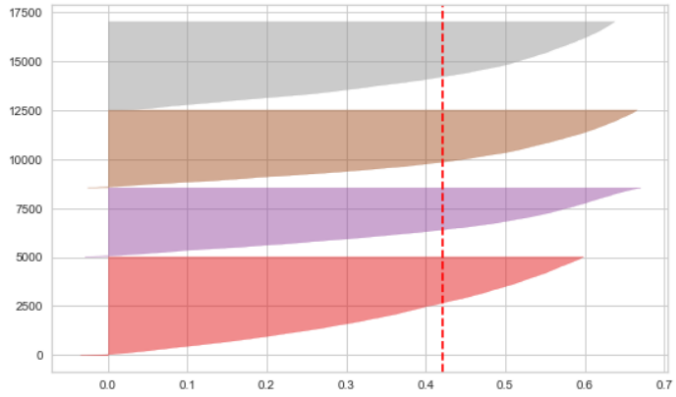


Determining best number of clusters using Silhouette score and Visualizations:

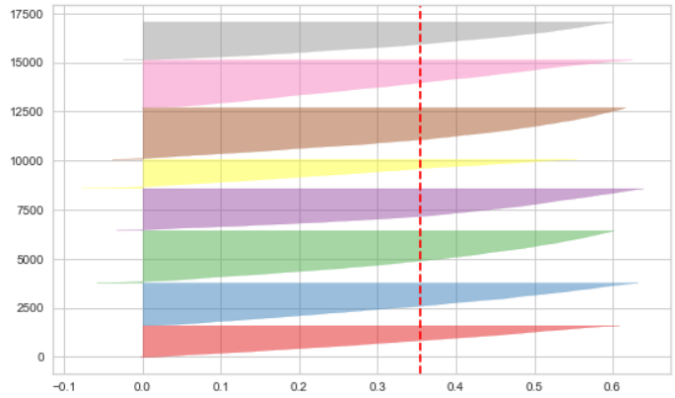
For (k= 2) the silhouette score is : 0.4239501648642863
Silhouette Plot:



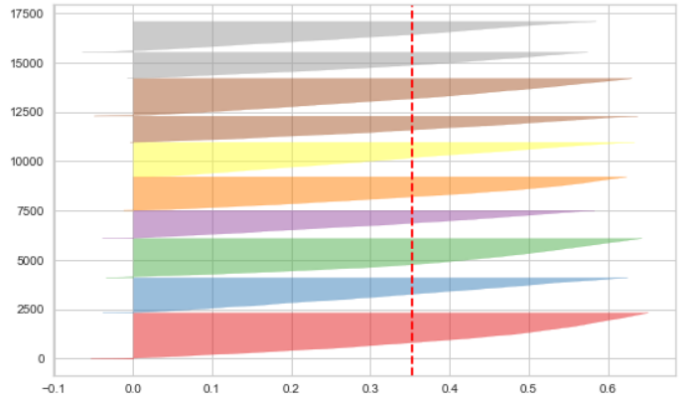
For (k= 4) the silhouette score is : 0.4211367790831248
Silhouette Plot:



For (k= 8) the silhouette score is : 0.3547341853216577
Silhouette Plot:



For (k= 10) the silhouette score is : 0.3522192394461182
Silhouette Plot:



We can observe that k=2 gives the best results, but this'll only form 2 clusters with 2 extreme genres like classical and metal which have very different song attributes.

This however won't be helpful for our recommendation system; hence we'll stick with 10 clusters for now.

Now that we have established the clusters and performed PCA to reduce its dimensions, we can move on to the recommendation system.

To build the recommendation system, multiple nested user defined functions were created in order to generate the mean vector and based on spatial distance of songs with respect to their various numeric attributes such as acousticness, danceability, tempo, energy etc.

In addition, categorical variables such as explicitness, time signature, classified cluster label and also the mean popularity of the artist was considered when fetching the closest match.

A snippet of the code for our recommendation system is shown below:

```
def mean_vector(song_list, spotify_data):
    vectors = []

    for song in song_list:
        song_info = get_song_info(song, spotify_data)
        if song_info is None:
            print('Warning: {} does not exist in Spotify or in database'.format(song['name']))
            continue
        vector = song_info[num_cols].values
        vectors.append(vector)

    matrix = np.array(list(vectors))
    return np.mean(matrix, axis=0)


#Final function to retreive the closest match using earlier function and returns the finds
def song_recommendation(song_list, spotify_data, n_songs=10):
    metadata_col = ['name', 'year', 'artists']
    dict_song = dict_flatten(song_list)

    center_song = mean_vector(song_list, spotify_data)
    scaler = pipeline_song_cluster.steps[0][1]
    scaled_data = scaler.transform(spotify_data[num_cols])
    scaled_center_song = scaler.transform(center_song.reshape(1, -1))
    distances = cdist(scaled_center_song, scaled_data, 'cosine')
    index = list(np.argsort(distances)[:, :n_songs][0])

    songs_rec = spotify_data.iloc[index]
    songs_rec = songs_rec[~songs_rec['name'].isin(dict_song['name'])]
    return songs_rec[metadata_col].to_dict(orient='records')
```

The system takes as input the name of a song and its year, to uniquely identify a track. Based on this input, the system generates recommendations of songs that are good matches to the inputted song, with respect to the songs attributes, labelled clusters and the artist's mean popularity score.

```
#Calling recommendation system function
#Passing song name and year to uniquely indetify a song
song_recommendation([{'name': 'Blinding Lights', 'year':2020}],  data)
```

```
[{'name': 'Acapella', 'year': 2014, 'artists': "['Karmin']"},
 {'name': 'Oh My God', 'year': 2008, 'artists': "['Ida Maria']"},
 {'name': 'Collar Full', 'year': 2013, 'artists': "['Panic! At The Disco']"},
 {'name': "What's Your Name", 'year': 2014, 'artists': "['Chase Rice']"},
 {'name': 'Juicy', 'year': 2019, 'artists': "['Doja Cat']"},
 {'name': "Listen to D.J.'s",
  'year': 2001,
  'artists': "['Long Beach Dub Allstars']"},
 {'name': 'Bježi Kišo S Prozora',
  'year': 1986,
  'artists': "['Crvena Jabuka']"},
 {'name': "I'd Do Anything", 'year': 2018, 'artists': "['Simple Plan']"}]
```

An interesting facet of our recommendation system is that it can receive multiple inputs to generate recommendations that are a match with the multiple inputs received.

```
#If 2 songs are passed, the closest match to both will be retreived
song_recommendation([{'name': 'Say So', 'year':2019},
                     {'name': "Don't Start Now", 'year':2019}],  data)
```


```
[{'name': 'Home',
  'year': 2018,
  'artists': "['Vince Staples', 'Richie Kohan']"},
 {'name': 'Oscillate Wildly - 2011 Remaster',
  'year': 1987,
  'artists': "['The Smiths']"},
 {'name': 'New Rules', 'year': 2017, 'artists': "['Dua Lipa']"},
 {'name': 'Ocean Drive', 'year': 2015, 'artists': "['Duke Dumont']"},
 {'name': 'beibs in the trap', 'year': 2016, 'artists': "['Travis Scott']"},
 {'name': 'New Rules', 'year': 2017, 'artists': "['Dua Lipa']"},
 {'name': 'Glycerine - Acoustic', 'year': 2005, 'artists': "['Bush']"},
 {'name': 'Highway to Hell', 'year': 1979, 'artists': "['AC/DC']"},
 {'name': 'Smack That', 'year': 2006, 'artists': "['Akon', 'Eminem']"},
 {'name': 'Fairly Local', 'year': 2015, 'artists': "['Twenty One Pilots']"}]
```


So, to conclude things, we developed multiple supervised models to predict the popularity score of a song. Our Random Forest model was able to achieve 0.866 R2 score and a relatively low RMSE and MAPE value. The observed features which impact the popularity the most other than ‘year’ of release are:

- Acousticness
- Duration_ms
- Speechiness
- Instrumentality
- Danceability
- Liveliness
- Energy

For our Recommendation system, we first used unsupervised models like t-SNE and K-means to broadly classify genres and labels songs into clusters respectively. Based on these clusters, an artists’ mean popularity and the various attributes of a song, a recommendation system was developed to get the closest match for an or multiple uniquely identifiable song.