Song Recommendation System

Team Members:

Student Name: **Student I'D:**

- **Raghu varan sharma Bhagavatham** **11543836**
- **Ainavolu Venkata Anoop Raj** **11518650**
- **Vamsi Krishna Kunta** **11525411**
- **Revathi Mudduluru** **11545180**
- **kaushic kolla** **11520251**

- **Executive Summary**

  People in our day and age are highly used to recommendation systems in numerous sectors, including but not limited to news feeds, streaming video and audio services, and online retailers. Given the consumption patterns that are prevalent in our contemporary environment, a recommendation system is an absolute necessity for a music streaming service.

  The individual songs that make up a music subscription service's comprehensive playlists are curated by recommendation algorithms. These playlists are based on certain criteria. That is done to help the user and make the consumer happier.

  Platforms that sell individual songs or albums sometimes use recommendation algorithms in an effort to boost revenue. The way it works is as follows: when a user purchases a certain track, a recommendation system will propose some more songs for him to purchase that are similar to the one he just purchased. This will encourage the user to make further purchases.

  If you want to listen to a great variety of songs you might like, then you'll need a music recommendation system. Some Reasons are

  - Enhance the level of contentment and participation of your clients.
  - Personalize your platform to the fullest extent possible.
  - Offer a streaming experience that is of high quality and completely immersive to customers.

- Develop your users' listening skills and their faith in you.
- Make it easy to utilize your service by eliminating the need to search for new music and wasting time doing so.
- Increase both your sales and your rate of subscriptions
- Improve click-through rate and conversion while reducing customer turnover.
- Foster both upselling and cross-selling opportunities.
- Automate the process of audio curation and playlisting
- Gain insights into the behavior of users, and base marketing choices on the collected data.
- The way their users listen to music has changed because of how many music streaming apps there are and how music recommendation algorithms work. Nielsen did a survey that found that more than 75 percent of the user experience is deeply connected to playlisting.

- **Introduction**

  A recommendation system is a filtering system, the objective of which is to forecast the preference that a user would give to a certain element, in our example, music. This may be accomplished by analyzing the user's past interactions with the system and comparing them to those of similar users. It is the central component of a large number of engines that are driven by particular recommender algorithms in order to provide consumers with recommendations for a single item or a collection of things based on such predictions.

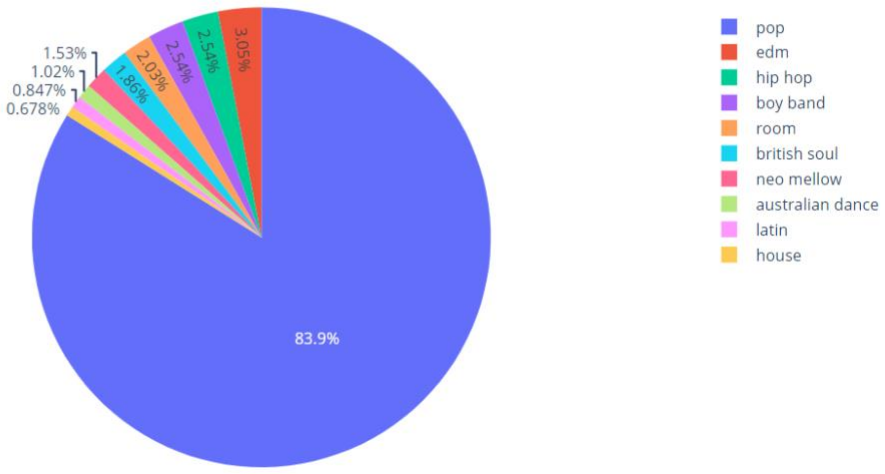
  Recently, a number of different recommendation systems have become an essential component of our regular routines, regardless of whether we are conscious of this development. Recommendation systems seem to be permeating our daily lives from practically every aspect of the digital environment, beginning with advertising product recommendations that are correctly targeted and culminating with tailored video or music playlists that have been produced just for us.

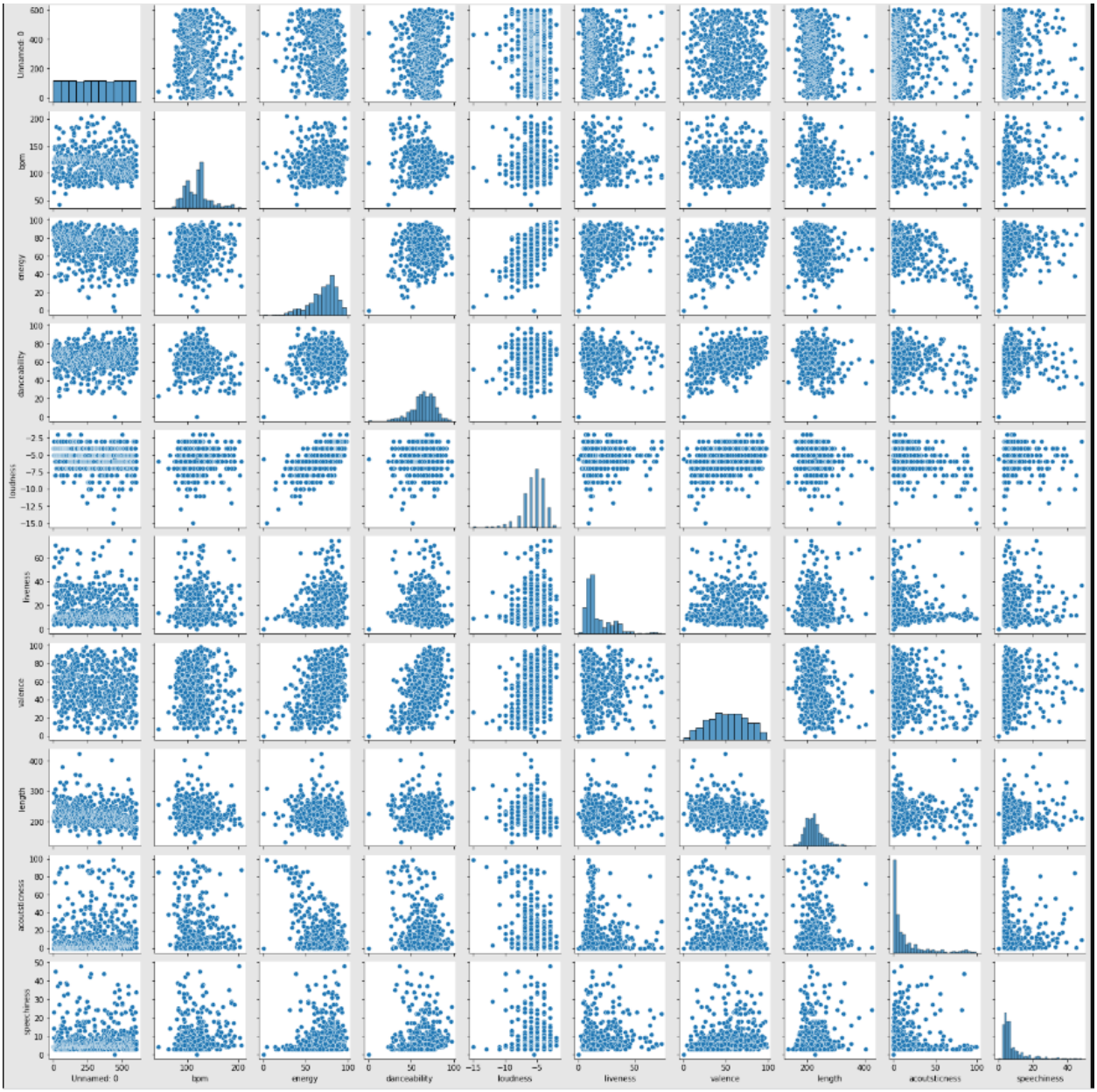
  In the field of music, recommendation algorithms are an integral component of the massive engines that power streaming music services such as Spotify, YouTube Music, Deezer, and Tidal, amongst others.

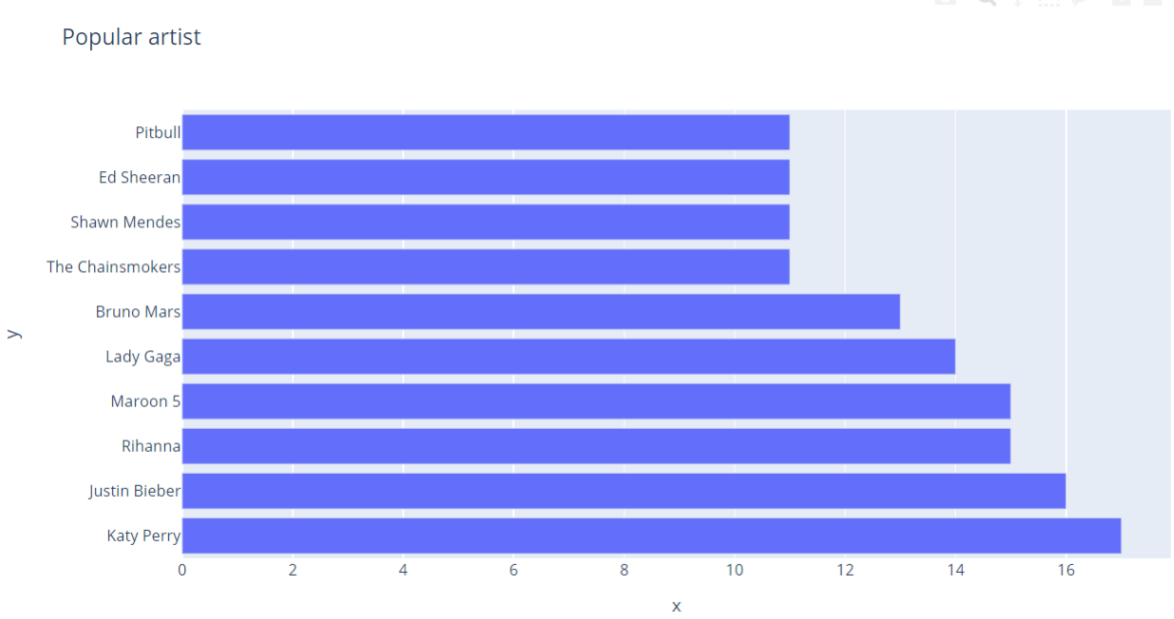
- **Architecture**

  Exploratory Data Analysis - Explored the dataset on various terms to find out what the dataset looks like. Used multiple libraries like Matplotlib, Seaborn and Plotly.

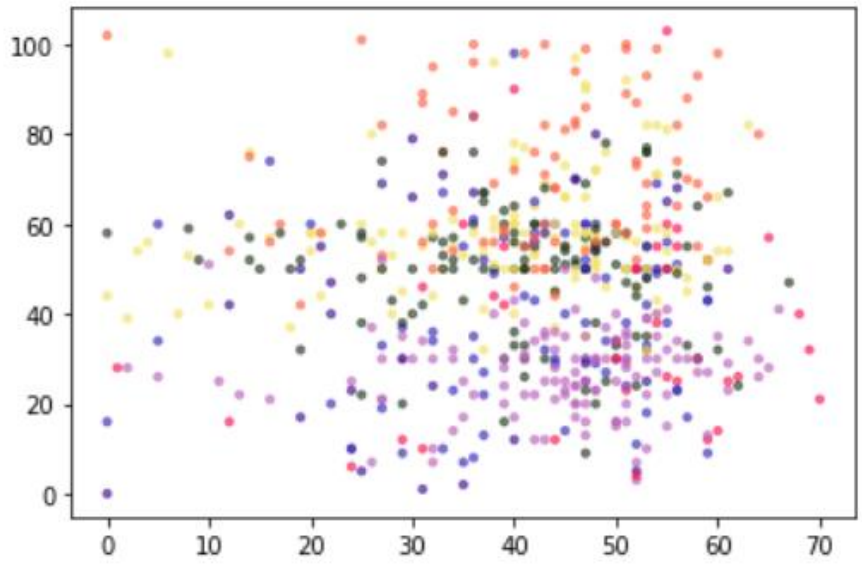
Popular genre

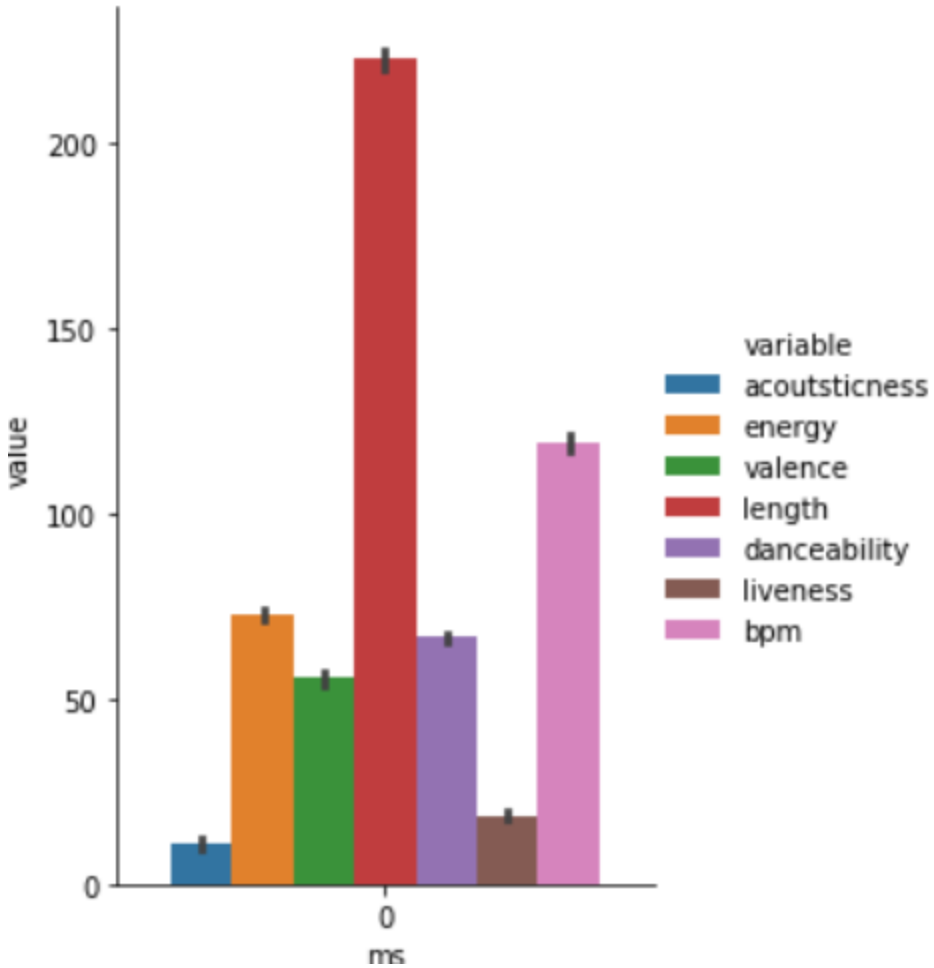



Unnamed: 0
bpm
energy
danceability
loudness
liveness
valence
length
acousticness
speechiness



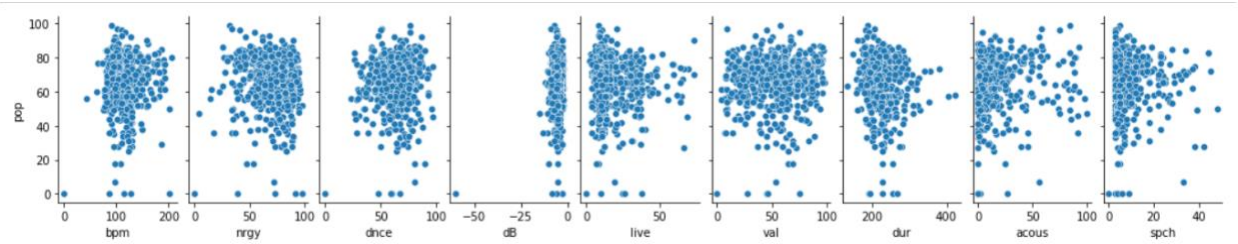
KMeans - used the Kmeans model to best fit our data so as to recommend songs based on how you are feeling.




200
150
100
50
0
value
0
ms
variable
acoutsticness
energy
valence
length
danceability
liveness
bpm

- **Modeling**

Data properties -



The songs have various attributes like beats per minute (bpm), energy, etc. The above picture shows one comparison between pop and the other properties.

Pre-Processing -

Dropping null values

```
df = flight.dropna()
```

Scaling the dataset

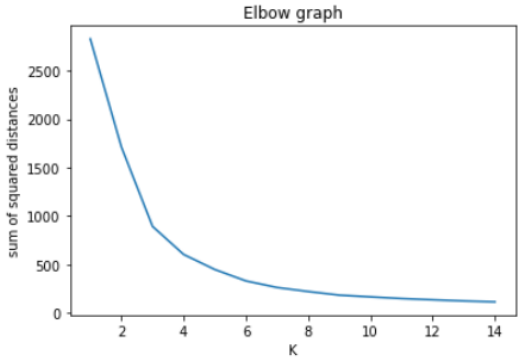
```
X_std = StandardScaler().fit_transform(new_df)

pca = PCA(n_components=.95)
principalComponents = pca.fit_transform(X_std)
```

Models Used

KMeans

THe best number of clusters comes out to be 3 and we use KMeans with 3 clusters.




```
#k-means
k=3

model = KMeans(n_clusters = k, algorithm='auto')
model.fit(pca_df)
predict = pd.DataFrame(model.predict(pca_df))
predict.columns = ['predict']
label = model.fit(pca_df).labels_

n_clusters_ = len(set(label))
print('Estimated number of clusters: %d' % n_clusters_)


```

```
Estimated number of clusters: 3
```

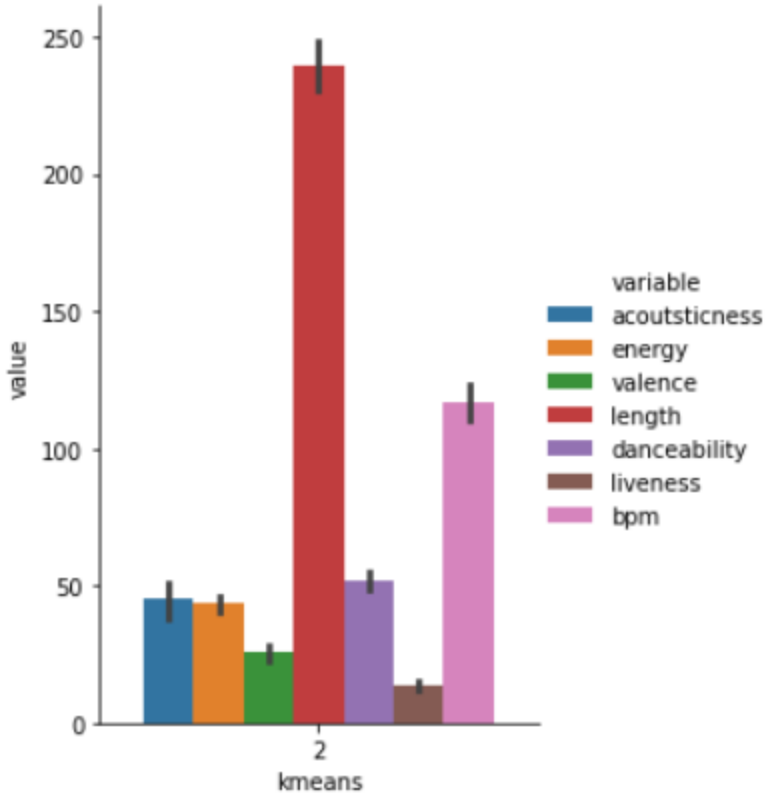
Hyperparameter Tuning -

```
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X_std)
pca_df = pd.DataFrame(principalComponents)
```

Performance metrics used - Silhouette Score, Cluster result

```
#silhouette_score
label = pd.DataFrame(model.fit(pca_df).labels_)
silhouette_score(pca_df, model.fit(pca_df).labels_, metric='euclidean')
```

```
0.6072213217980009
```



Test Results and Significance

Top Recommendations for party songs

```
TiK ToK by Kesha
Dynamite by Taio Cruz
Eenie Meenie by Sean Kingston
Your Love Is My Drug by Kesha
Take It Off by Kesha
```

- **Implementation**
  React in javascript and its libraries are used for making a website that will help you choose recommendations. The React.js framework is an open-source JavaScript framework and library developed by Facebook. It's used for building interactive user interfaces and web applications quickly and efficiently with significantly less code than you would with vanilla JavaScript. We will use React JS to build our front end.

- **Conclusion**

  Achievement

  We were able to successfully build a song recommendation system that recommends us songs based on which type of songs we want to listen to (happy, sad, partying, etc)

  Limitations of work

  We are using the dataset of the top 10 songs in multiple genres of Spotify, hence not covering the entire plethora of songs that exist in the world.

  Future Avenues

  Personalization of the song recommendation system means it will automatically recommend songs based on songs you have been listening to.

- **Repository/Archive**

  Add github repository here.

- **Code**

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly as py
import plotly.express as px
import plotly.graph_objects as go
from plotly.offline import plot
import numpy as np
import seaborn as sb
from matplotlib import rcParams
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from scipy.cluster.hierarchy import linkage, dendrogram
from sklearn.cluster import DBSCAN
from sklearn.cluster import AgglomerativeClustering
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
import warnings
warnings.filterwarnings("ignore")
```

```
df10 = pd.read_csv('top10s.csv',encoding='ISO-8859-1')
yearless_df = df10.drop(['year', 'pop'], axis=1)
yearless_df=yearless_df.drop_duplicates()
yearless_df['top genre'].value_counts()
for i in yearless_df['top genre']:
    if 'pop' in i :
        yearless_df['top genre'] = yearless_df['top genre'].replace(i, 'pop')
    elif 'hip hop' in i :
        yearless_df['top genre'] = yearless_df['top genre'].replace(i, 'hip hop')
    elif 'edm' in i:
        yearless_df['top genre'] = yearless_df['top genre'].replace(i, 'edm')
    elif 'r&b' in i:
        yearless_df['top genre'] = yearless_df['top genre'].replace(i, 'pop')
    elif 'latin' in i:
        yearless_df['top genre'] = yearless_df['top genre'].replace(i, 'latin')
    elif 'room' in i:
        yearless_df['top genre'] = yearless_df['top genre'].replace(i, 'room')
    elif 'electro' in i:
        yearless_df['top genre'] = yearless_df['top genre'].replace(i, 'edm')
    elif 'house' in i:
        yearless_df['top genre'] = yearless_df['top genre'].replace(i, 'house')

for i in df10['top genre']:
    if 'pop' in i :
        df10['top genre'] = df10['top genre'].replace(i, 'pop')
    elif 'hip hop' in i :
        df10['top genre'] = df10['top genre'].replace(i, 'hip hop')
    elif 'edm' in i:
        df10['top genre'] = df10['top genre'].replace(i, 'edm')
    elif 'r&b' in i:
        df10['top genre'] = df10['top genre'].replace(i, 'pop')
    elif 'latin' in i:
        df10['top genre'] = df10['top genre'].replace(i, 'latin')
    elif 'room' in i:
        df10['top genre'] = df10['top genre'].replace(i, 'room')
    elif 'electro' in i:
        df10['top genre'] = df10['top genre'].replace(i, 'edm')
    elif 'house' in i:
        df10['top genre'] = df10['top genre'].replace(i, 'house')

yearless_df['top genre'] = yearless_df['top genre'].replace('complextro', 'edm')
yearless_df['top genre'] = yearless_df['top genre'].replace('chicago rap', 'hip hop')

df10['top genre'] = df10['top genre'].replace('complextro', 'edm')
df10['top genre'] = df10['top genre'].replace('chicago rap', 'hip hop')

#Change columns name
df = yearless_df
```

```
df.rename(columns={'title': 'song',
                   'artist':'artist',
                   'top genre':'genre',
                   'bpm':'bpm',
                   'nrgy': 'energy',
                   'dnce': 'danceability',
                   'dB':'loudness',
                   'live': 'liveness',
                   'val': 'valence',
                   'dur': 'length',
                   'acous': 'acoutsticness',
                   'spch': 'speechiness'}, inplace=True)

df10.rename(columns={'title': 'song',
                   'artist':'artist',
                   'top genre':'genre',
                   'bpm':'bpm',
                   'nrgy': 'energy',
                   'dnce': 'danceability',
                   'dB':'loudness',
                   'live': 'liveness',
                   'val': 'valence',
                   'dur': 'length',
                   'acous': 'acoutsticness',
                   'spch': 'speechiness'}, inplace=True)
df1=df['genre'].value_counts().head(10)
df1.index
fig = px.pie(df1, names=df1.index, values='genre', title = 'Popular genre',labels='genre')
fig.show()
#Popular artist
fig=px.bar(df, x = df['artist'].value_counts().head(10), y=df['artist'].value_counts().head(10).index,
          title = 'Popular artist')
fig.show()
sns.pairplot(df, height = 2)
plt.show()
#Variable relationships
fig = px.scatter(df10,x='loudness',
y='energy',color='energy',hover_name='song',hover_data=['artist','year'],title='Relationship loudness and
energy')
fig.show()

fig = px.scatter(df10,x='danceability', y='bpm',color='bpm',hover_name='song',hover_data=['artist','year'],
title='Relationship Danceability and bpm')
fig.show()

fig = px.scatter(df10,x='danceability',
y='valence',color='valence',hover_name='song',hover_data=['artist','year'],title='Relationship Danceability
and valence')
fig.show()
```

```
#most popular artist in 2019
fig = px.scatter(df10.query('year==2019'), y='pop', x='artist', hover_name='song', color='pop', title =
'Popular artist in 2019' )
fig.show()

#most popular artist in 2018
fig = px.scatter(df10.query('year==2018'), y='pop', x='artist', hover_name='song', color='pop', title =
'Popular artist in 2018' )
fig.show()

#most popular artist in 2017
fig = px.scatter(df10.query('year==2017'), y='pop', x='artist', hover_name='song', color='pop', title =
'Popular artist in 2017' )
fig.show()

#most popular artist in 2016
fig = px.scatter(df10.query('year==2016'), y='pop', x='artist', hover_name='song', color='pop', title =
'Popular artist in 2016' )
fig.show()
#Change null values
df.bpm = df.bpm.replace(0, df.bpm.mean())
df.bpm.unique()

df.loudness = df.loudness.replace(-60, df.loudness.mean())
df.loudness.unique()
plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_), c='r')
plt.xlabel('Number of Components')
plt.ylabel('Variance (%)') #for each component
plt.title('Pulsar Dataset Explained Variance')
plt.show()
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X_std)
pca_df = pd.DataFrame(principalComponents)
# SSR elbow
sum_of_squared_distances = []

K = range(1,15)
for k in K:
   km = KMeans(n_clusters=k)
   km = km.fit(pca_df)
   sum_of_squared_distances.append(km.inertia_)

ax = sns.lineplot(x=K, y = sum_of_squared_distances)
ax.set(xlabel='K', ylabel='sum of squared distances', title='Elbow graph')

#k-means
k=3
```

```
model = KMeans(n_clusters = k, algorithm='auto')
model.fit(pca_df)
predict = pd.DataFrame(model.predict(pca_df))
predict.columns = ['predict']
label = model.fit(pca_df).labels_

n_clusters_ = len(set(label))
print('Estimated number of clusters: %d' % n_clusters_)

#silhouette_score
label = pd.DataFrame(model.fit(pca_df).labels_)
silhouette_score(pca_df, model.fit(pca_df).labels_, metric='euclidean')
#cluster result
df_scaled = pd.DataFrame(new_df)
df_scaled['kmeans'] = model.fit(pca_df).labels_
df_mean = (df_scaled.loc[df_scaled.kmeans!=-1, :].groupby('kmeans').mean())
results = pd.DataFrame(columns=['Variable', 'Var'])
for column in df_mean.columns[1:]:
    results.loc[len(results), :] = [column, np.var(df_mean[column])]
    selected_columns = list(results.sort_values('Var', ascending=False,).head(7).Variable.values) +
['kmeans']
    tidy = df_scaled[selected_columns].melt(id_vars='kmeans')

for i in range(3):
    sns.catplot(x='kmeans', y='value', hue='variable', data=tidy[tidy['kmeans']==i], height=5, aspect=.7,
kind='bar')

#means shift clustering
from sklearn.cluster import MeanShift
ms = MeanShift(bandwidth=2)
ms.fit(pca_df)
predict = pd.DataFrame(ms.predict(pca_df))

predict.columns = ['predict']
label3 = ms.fit(pca_df).labels_

n_clusters_ = len(set(label3))
print('Estimated number of clusters: %d' % n_clusters_)
#silhouette_score
label3 = pd.DataFrame(ms.fit(pca_df).labels_)
silhouette_score(pca_df, ms.fit(pca_df).labels_, metric='euclidean')
#cluster result
df_scaled['ms'] = ms.fit(pca_df).labels_
df_mean = (df_scaled.loc[df_scaled.kmeans!=-1, :].groupby('ms').mean())
results = pd.DataFrame(columns=['Variable', 'Var'])
for column in df_mean.columns[1:]:
    results.loc[len(results), :] = [column, np.var(df_mean[column])]
    selected_columns = list(results.sort_values('Var', ascending=False,).head(7).Variable.values) + ['ms']
```

```
    tidy = df_scaled[selected_columns].melt(id_vars='ms')
df_mean.columns[1:]
for i in range(4):
    sns.catplot(x='ms', y='value', hue='variable', data=tidy[tidy['ms']==i], height=5, aspect=.7, kind='bar')
# AgglomerativeClustering
ac = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='ward')
ac.fit_predict(pca_df)
label2 = ac.fit(pca_df).labels_

n_clusters_ = len(set(label2))
print('Estimated number of clusters: %d' % n_clusters_)
#silhouette_score
label2 = pd.DataFrame(ac.fit(pca_df).labels_)
silhouette_score(pca_df, ac.fit(pca_df).labels_, metric='euclidean')
#clustering result
df_scaled['ac'] = ac.fit(pca_df).labels_
df_mean = (df_scaled.loc[df_scaled.kmeans!=-1, :].groupby('ac').mean())
results = pd.DataFrame(columns=['Variable', 'Var'])
for column in df_mean.columns[1:]:
    results.loc[len(results), :] = [column, np.var(df_mean[column])]
    selected_columns = list(results.sort_values('Var', ascending=False,).head(7).Variable.values) + ['ac']
    tidy = df_scaled[selected_columns].melt(id_vars='ac')

for i in range(3):
    sns.catplot(x='ac', y='value', hue='variable', data=tidy[tidy['ac']==i], height=5, aspect=.7, kind='bar')
cen_0 = [i[0] for i in centroids]
cen_1 = [i[1] for i in centroids]
cen_2 = [i[2] for i in centroids]
cen_3 = [i[3] for i in centroids]
cen_4 = [i[4] for i in centroids]
cen_5 = [i[5] for i in centroids]
cen_6 = [i[6] for i in centroids]

## add to df
df['cen_0'] = df.cluster.map({0:cen_0[0], 1:cen_0[1], 2:cen_0[2], 3:cen_0[3], 4:cen_0[4], 5:cen_0[5],
6:cen_0[6]})
df['cen_1'] = df.cluster.map({0:cen_1[0], 1:cen_1[1], 2:cen_1[2], 3:cen_1[3], 4:cen_1[4], 5:cen_1[5],
6:cen_1[6]})
df['cen_2'] = df.cluster.map({0:cen_2[0], 1:cen_2[1], 2:cen_2[2], 3:cen_2[3], 4:cen_2[4], 5:cen_2[5],
6:cen_2[6]})
df['cen_3'] = df.cluster.map({0:cen_3[0], 1:cen_3[1], 2:cen_3[2], 3:cen_3[3], 4:cen_3[4], 5:cen_3[5],
6:cen_3[6]})
df['cen_4'] = df.cluster.map({0:cen_4[0], 1:cen_4[1], 2:cen_4[2], 3:cen_4[3], 4:cen_4[4], 5:cen_4[5],
6:cen_4[6]})
df['cen_5'] = df.cluster.map({0:cen_5[0], 1:cen_5[1], 2:cen_5[2], 3:cen_5[3], 4:cen_5[4], 5:cen_5[5],
6:cen_5[6]})
df['cen_6'] = df.cluster.map({0:cen_6[0], 1:cen_6[1], 2:cen_6[2], 3:cen_6[3], 4:cen_6[4], 5:cen_6[5],
6:cen_6[6]})
# define and map colors
```

```
colors = ['#21351d', '#bd60be', '#f5e261', '#3a0892', '#fe1054', '#3627bd', '#ff603f']
df['c'] = df.cluster.map({0:colors[0], 1:colors[1], 2:colors[2], 3:colors[3], 4:colors[4], 5:colors[5],
6:colors[6]})
#Recommendation
"""
0 to 6
0 being the lightest and happiest mood while 6 being saddest
Change x to change recommendation with different mood
Change y for number of recommendations
"""

x = 0
y = 5
for i in range(y):
    print(df_.loc[df[df.cluster == x].iloc()[i].name].title + " by " + df_.loc[df[df.cluster ==
x].iloc()[i].name].artist)
```

Git Hub link:
https://github.com/UNT-CSCE5214/project-repository-team_song-recommendation-system