CollabMP3

Importing Libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler

import warnings
warnings.filterwarnings("ignore")
```

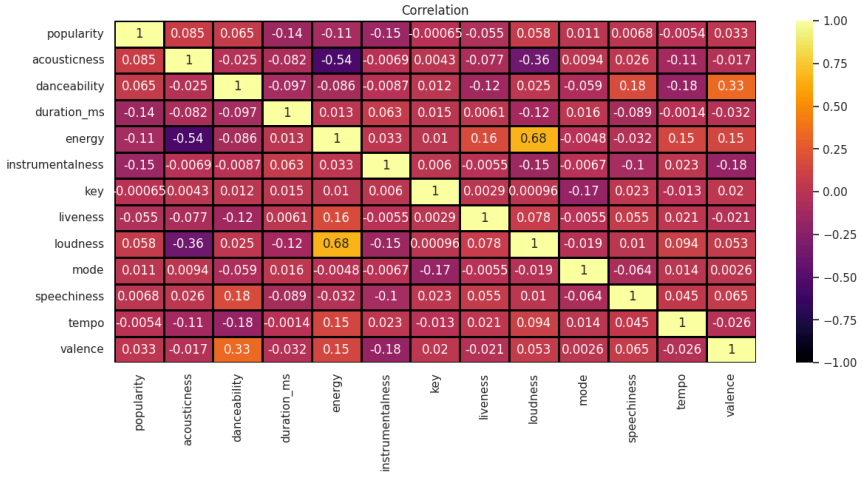
Data Preprocessing

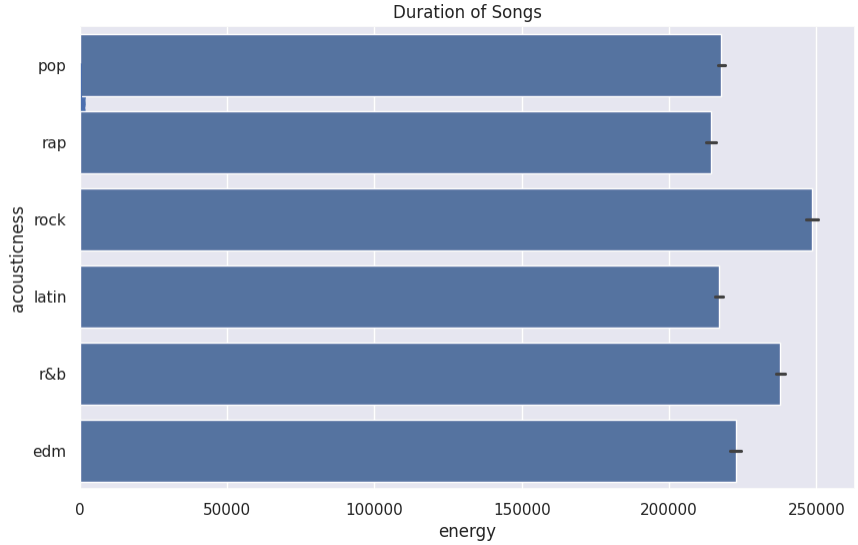
```
data = pd.read_csv("SpotifyFeatures.csv")
data.head()
```

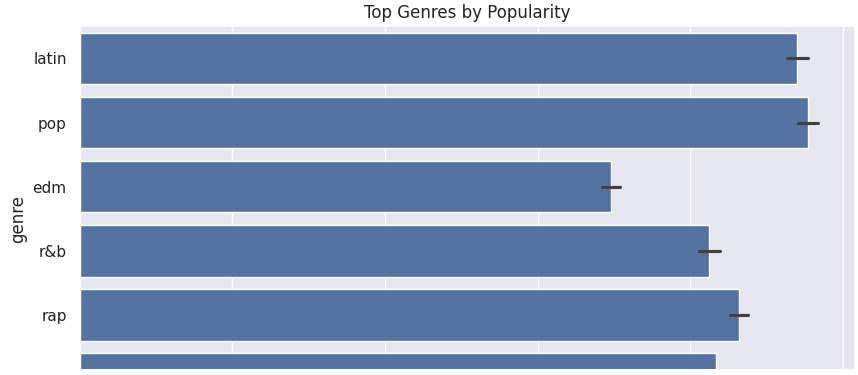
	genre	artist_name	track_name	track_id	popularity	acousticness	c
0	pop	Ed Sheeran	I Don't Care (with Justin Bieber) - Loud Luxur...	6f807x0ima9a1j3VPbc7VN	66	0.1020	
1	pop	Maroon 5	Memories - Dillon Francis Remix	0r7CVbZTWZgbTCYdfa2P31	67	0.0724	
2	pop	Zara Larsson	All the Time - Don Diablo Remix	1z1Hg7Vb0AhHDiEmnDE79l	70	0.0794	

```
def visualize(data):
    corr = data.corr(method="pearson")
    plt.figure(figsize=(14,6))
    heatmap = sns.heatmap(corr, annot=True,vmin=-1, vmax=1, center=0, cmap="inferno", linewidths=1, linecolor="Black")
    heatmap.set_title("Correlation")
    plt.savefig('Plots/Heatmap.png')

    sample = data.sample(int(0.01*len(data)))
    print("Number of samples taken: ",len(sample))

    plt.figure(figsize=(10,6))
    sns.regplot(data=sample, y="acousticness", x="energy").set(title="Acousticness vs Energy")

    sns.set_style(style="darkgrid")
    plt.title("Duration of Songs")
    sns.color_palette("rocket", as_cmap = True)
    sns.barplot(y="genre", x="duration_ms", data = data)
    plt.savefig('Plots/DurationOfSongs.png')

    sns.set_style(style = "darkgrid")
    plt.figure(figsize=(10,5))
    famous = data.sort_values("popularity", ascending=False)
    sns.barplot(y="genre", x="popularity", data = famous).set(title="Top Genres by Popularity")
    plt.savefig('Plots/TopGenresByPopularity.png')


visualize(data)
```

```
Number of samples taken:  328
```

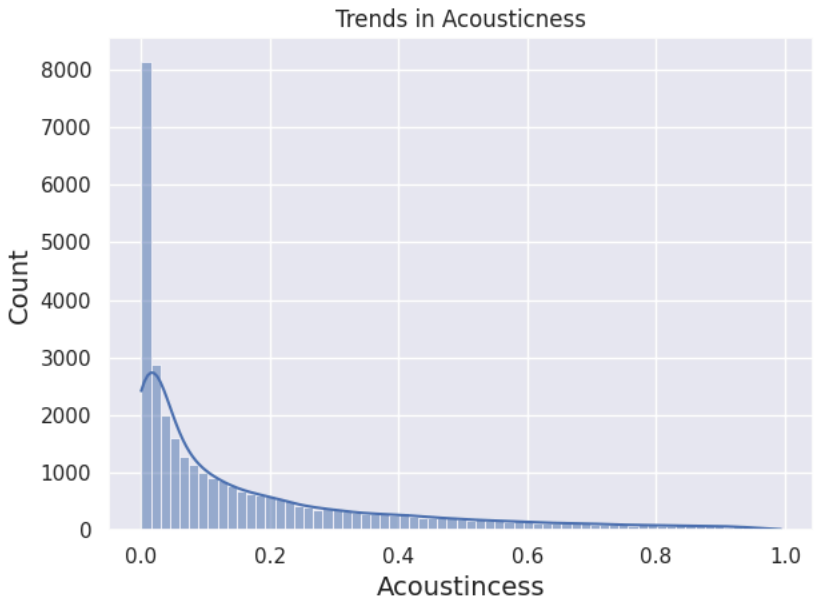






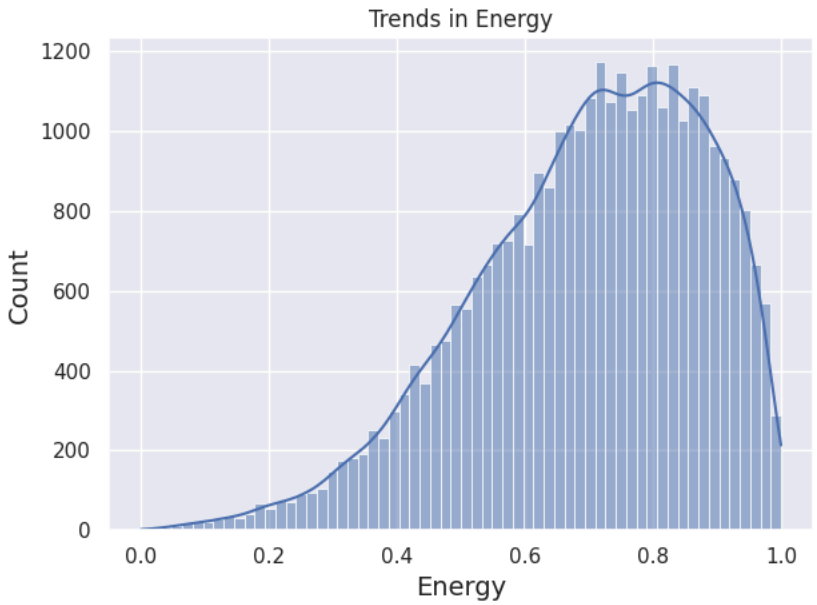
```
def plot1(data):
    print("Mean value of acousticness:", data['acousticness'].mean())
    sns.histplot(x='acousticness', data=data, kde=True)
    plt.title("Trends in Acousticness")
    plt.xlabel('Acoustincess', fontsize=14)
    plt.ylabel('Count', fontsize=14)
    plt.tight_layout()
    plt.savefig('Plots/TrendsAcousticness.png')

def plot2(data):
    # mean value and histplot for for energy feature
    print("Mean value of energy:", data['energy'].mean())
    sns.histplot(x='energy', data=data, kde=True)
    plt.title("Trends in Energy")
    plt.xlabel('Energy', fontsize=14)
    plt.ylabel('Count', fontsize=14)
    plt.tight_layout()
    plt.savefig('Plots/TrendsEnergy.png')


plot1(data)
```

```
Mean value of acousticness: 0.1753337150793409
```




```
plot2(data)
```

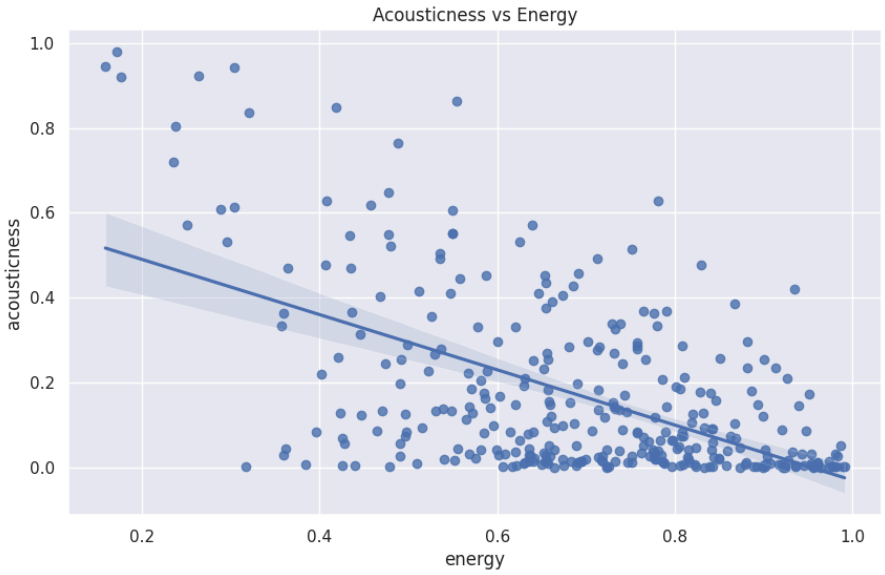
```
Mean value of energy: 0.6986192707032558
```




```
def plot3(data):
    sample = data.sample(int(0.01*len(data)))
    print("Number of samples taken: ",len(sample))

    plt.figure(figsize=(10,6))
    sns.regplot(data=sample, y="acousticness", x="energy").set(title="Acousticness vs Energy")
    plt.savefig('Plots/AcouticnessVsEnergy.png')
```

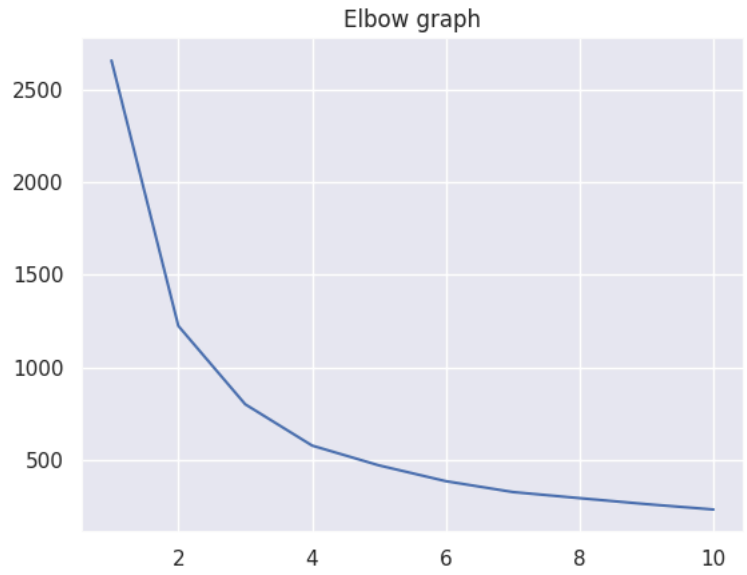
```
plot3(data)
```

```
Number of samples taken:  328
```

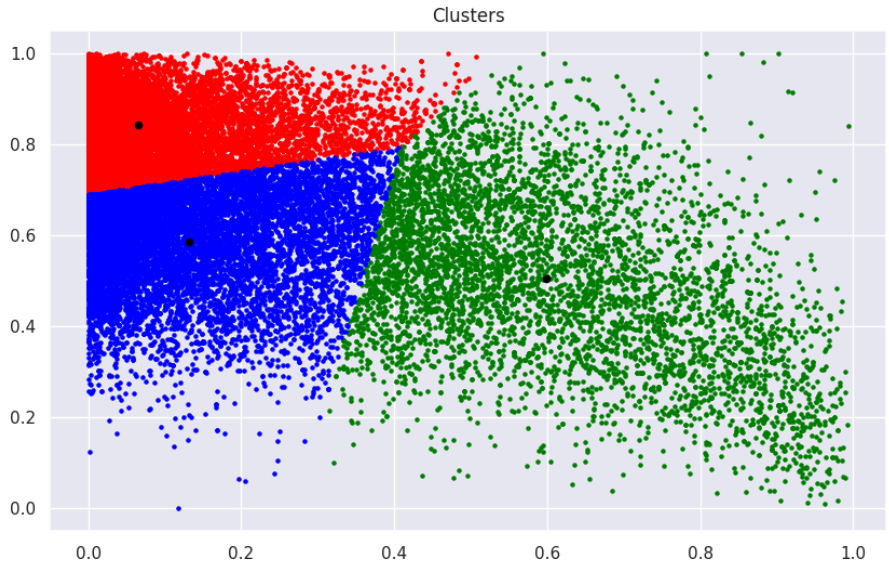


Cluster creation

```
def plot_clus(X, Y, kmeans):
    plt.figure(figsize=(10,6))
    plt.scatter(X[Y==0,0], X[Y==0,1], s=5, c='red', label="Cluster 1")
    plt.scatter(X[Y==1,0], X[Y==1,1], s=5, c='green', label="Cluster 2")
    plt.scatter(X[Y==2,0], X[Y==2,1], s=5, c='blue', label="Cluster 3")
    plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=20, c="black", label='Centroids')
    plt.title("Clusters")
    plt.savefig('Plots/Clusters.png')

def cluster(data):
    X = data.iloc[:, [5,8]].values

    wcss = []
    for i in range(1,11):
      kmeans = KMeans(n_clusters=i, init='k-means++', random_state=30)
      kmeans.fit(X)
      wcss.append(kmeans.inertia_)

    sns.set()
    plt.plot(range(1,11), wcss)
    plt.title("Elbow graph")
    plt.show()
    plt.savefig('Plots/ElbowGraph.png')

    kmeans = KMeans(n_clusters=3, init='k-means++', random_state=0)

    Y = kmeans.fit_predict(X)

    plot_clus(X, Y, kmeans)


cluster(data)
```




```
<Figure size 640x480 with 0 Axes>
```



Most Popular Songs

```
def most_popular(data):
    df2 = data.copy()
    df2.drop_duplicates(subset = "track_name", inplace = True) #dropping duplicate songs
    df2.head()

    rslt_df = df2.sort_values(by = 'popularity', ascending = False)
    rslt_df = rslt_df[['genre', 'artist_name', 'track_name']]

    print("Top 10 most popular songs:\n")
    for i in range(10):
        row_list = rslt_df.loc[i, :].values.flatten().tolist()
        print(row_list[1], "-", row_list[2])


most_popular(data)
```

```
Top 10 most popular songs:

Ed Sheeran - I Don't Care (with Justin Bieber) - Loud Luxury Remix
Maroon 5 - Memories - Dillon Francis Remix
Zara Larsson - All the Time - Don Diablo Remix
```

```
The Chainsmokers - Call You Mine - Keanu Silva Remix
Lewis Capaldi - Someone You Loved - Future Humans Remix
Ed Sheeran - Beautiful People (feat. Khalid) - Jack Wins Remix
Katy Perry - Never Really Over - R3HAB Remix
Sam Feldt - Post Malone (feat. RANI) - GATTÜSO Remix
Avicii - Tough Love - Tiësto Remix / Radio Edit
Shawn Mendes - If I Can't Have You - Gryffin Remix
```

Code for single track that is being playing in real time by the user

Plotting a random song on the plot

```
import random

num_rows = data.shape[0]
random_index = random.randint(0, num_rows - 1)
print(random_index)
df = data.iloc[[18187]].reset_index(drop=True)

#example for red cluster
# df = data.iloc[[18187]].reset_index(drop=True)

#example for green cluster
# df = data.iloc[[25598]].reset_index(drop=True)

#example for blue cluster
# df = data.iloc[[23641]].reset_index(drop=True)

df.head()
```

```
9257
```

	genre	artist_name	track_name	track_id	popularity	acousticness	danceability	duration_ms	energy	instrumentalnes
0	latin	Jarabe De Palo	La Flaca - Acousique	4EIX0PX2miGsMVsDJS2qqb	27	0.794	0.611	220200	0.231	0.

```
A = []
def song_features(data):
    B = []

    data = data.values.tolist()
    # print(data[0][8])
    # print(data[0][5])

    B.append(data[0][8])
    B.append(data[0][5])

    A.append(B)
    # print(A)


song_features(df)
```

```
from scipy.spatial import distance

def AddPoint(plot, x, y, color):
    plt.scatter(x, y, c=color)
    plt.figure(figsize=(10,6))
    plt.show()


#determining which cluster the given song is in
num = cal_cluster(A)

X = data.iloc[:, [5,8]].values
kmeans = KMeans(n_clusters=3, init='k-means++', random_state=0)
Y = kmeans.fit_predict(X)

#plotting the song in the scatter plot
plot_clus(X, Y, kmeans)
AddPoint(plt, A[0][0], A[0][1], 'yellow')
```

```
Song is in RED cluster
```