**Music Recommendation System by Facial Emotion**

**Programs & Output :**

import os

import numpy as np

import pandas as pd

import seaborn as sns

import plotly.express as px

import matplotlib.pyplot as plt

%matplotlib inline

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 collections import defaultdict

import difflib

import warnings

warnings.filterwarnings("ignore")

import os

for dirname, \_, filenames in os.walk('//input'):

    for filename in filenames:

        print(os.path.join(dirname, filename))

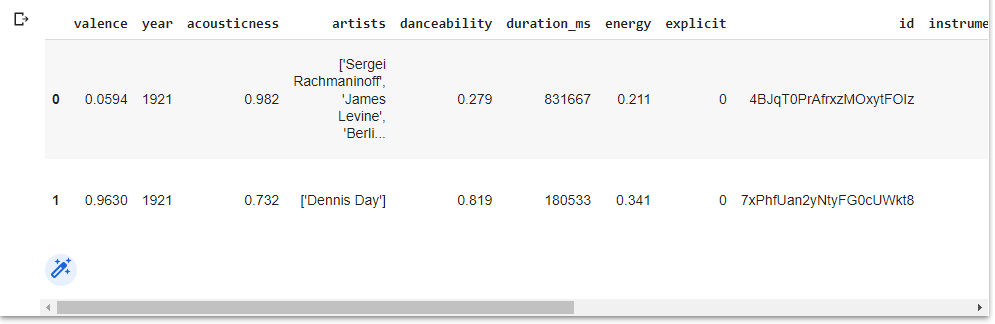
data = pd.read\_csv("/content/drive/MyDrive/DATA ANALYTICS/data.csv")

genre\_data = pd.read\_csv('/content/drive/MyDrive/DATA ANALYTICS/data\_by\_genres.csv')

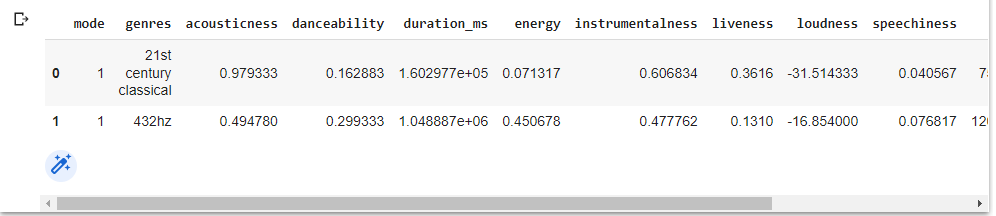
year\_data = pd.read\_csv('/content/drive/MyDrive/DATA ANALYTICS/data\_by\_year.csv')

artist\_data = pd.read\_csv('/content/drive/MyDrive/DATA ANALYTICS/data\_by\_artist.csv')

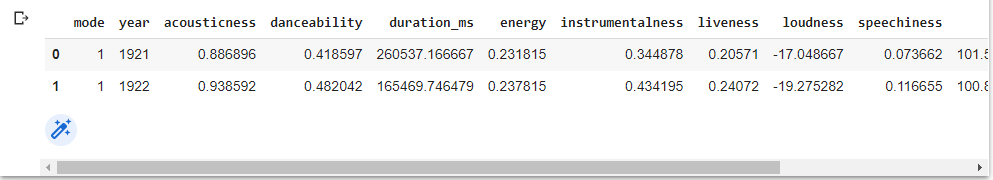
data.head(2)



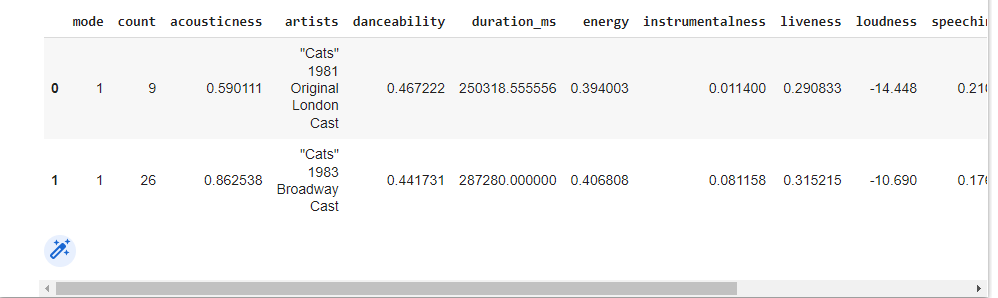
genre\_data.head(2)



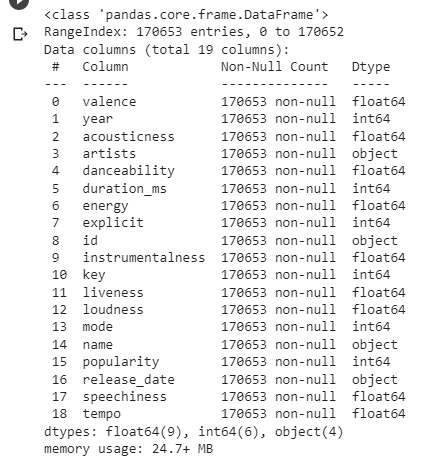
year\_data.head(2)



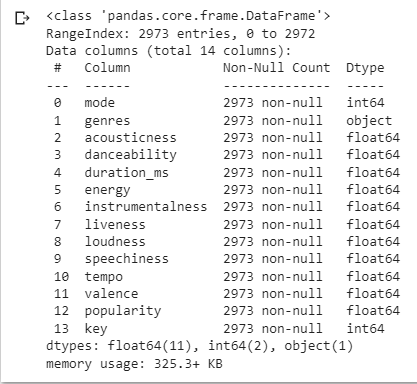
artist\_data.head(2)



data.info()



genre\_data.info()



data['decade'] = data['year'].apply(lambda year : f'{(year//10)\*10}s' )

sound\_features = ['acousticness', 'danceability', 'energy', 'instrumentalness', 'liveness', 'valence']

fig = px.line(year\_data, x='year', y=sound\_features,title='Trend of various sound features over decades')

fig.show()

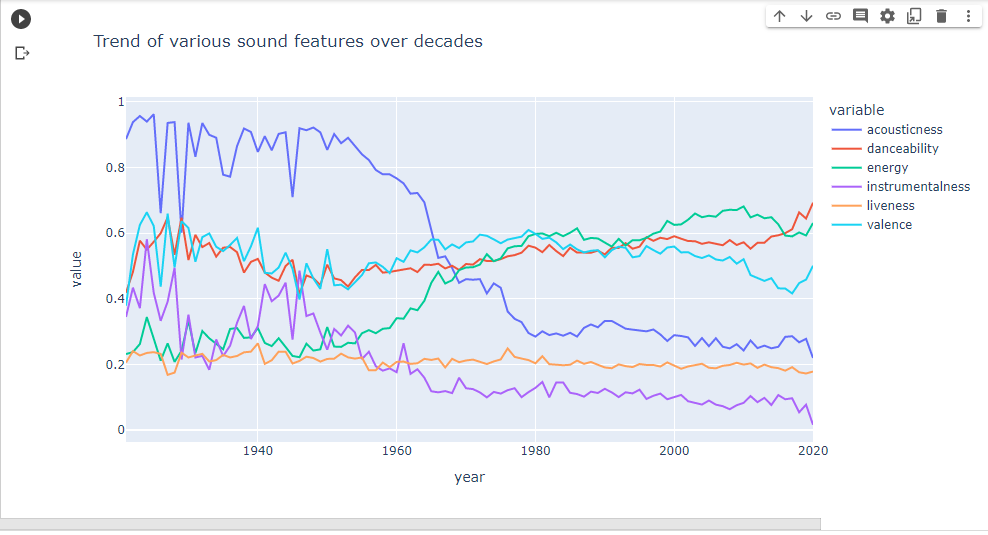
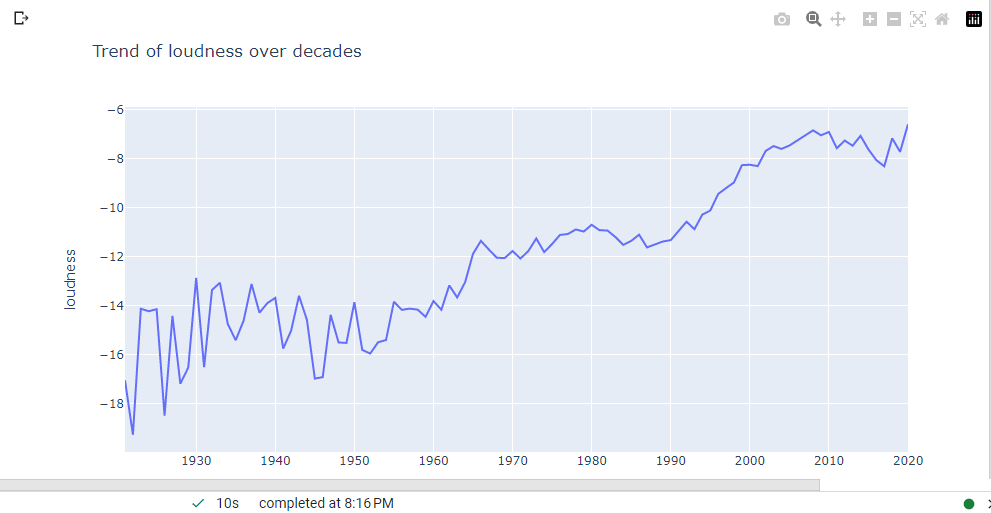


fig = px.line(year\_data, x='year', y='loudness',title='Trend of loudness over decades')

fig.show()



from wordcloud import WordCloud,STOPWORDS

stopwords = set(STOPWORDS)

comment\_words = " ".join(genre\_data['genres'])+" "

wordcloud = WordCloud(width = 800, height = 800,

                background\_color ='white',

                stopwords = stopwords,

                max\_words=40,

                min\_font\_size = 10).generate(comment\_words)

plt.figure(figsize = (8, 8), facecolor = None)

plt.imshow(wordcloud)

plt.axis("off")

plt.tight\_layout(pad = 0)

plt.title("Genres Wordcloud")

plt.show()



stopwords = set(STOPWORDS)

comment\_words = " ".join(artist\_data['artists'])+" "

wordcloud = WordCloud(width = 800, height = 800,

                background\_color ='white',

                stopwords = stopwords,

                min\_word\_length=3,

                max\_words=40,

                min\_font\_size = 10).generate(comment\_words)

plt.figure(figsize = (8, 8), facecolor = None)

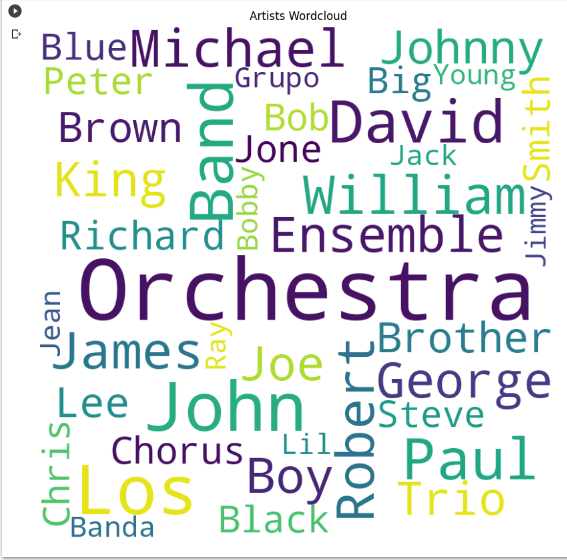
plt.imshow(wordcloud)

plt.axis("off")

plt.title("Artists Wordcloud")

plt.tight\_layout(pad = 0)

plt.show()

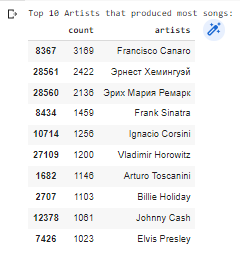


top10\_popular\_artists = artist\_data.nlargest(10, 'popularity')

top10\_most\_song\_produced\_artists = artist\_data.nlargest(10, 'count')

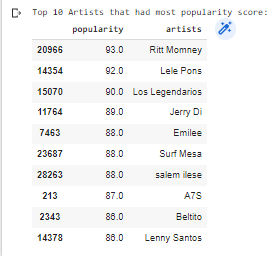
print('Top 10 Artists that produced most songs:')

top10\_most\_song\_produced\_artists[['count','artists']].sort\_values('count',ascending=False)



print('Top 10 Artists that had most popularity score:')

top10\_popular\_artists[['popularity','artists']].sort\_values('popularity',ascending=False)



from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import Pipeline

cluster\_pipeline = Pipeline([('scaler', StandardScaler()), ('kmeans', KMeans(n\_clusters=12))])

X = genre\_data.select\_dtypes(np.number)

cluster\_pipeline.fit(X)

genre\_data['cluster'] = cluster\_pipeline.predict(X)

from sklearn.manifold import TSNE

tsne\_pipeline = Pipeline([('scaler', StandardScaler()), ('tsne', TSNE(n\_components=2, verbose=1))])

genre\_embedding = tsne\_pipeline.fit\_transform(X) # returns np-array of coordinates(x,y) for each record after TSNE.

projection = pd.DataFrame(columns=['x', 'y'], data=genre\_embedding)

projection['genres'] = genre\_data['genres']

projection['cluster'] = genre\_data['cluster']

fig = px.scatter(

    projection, x='x', y='y', color='cluster', hover\_data=['x', 'y', 'genres'],title='Clusters of genres')

fig.show()



song\_cluster\_pipeline = Pipeline([('scaler', StandardScaler()),

                                  ('kmeans', KMeans(n\_clusters=25,

                                   verbose=False))

                                 ], verbose=False)

X = data.select\_dtypes(np.number)

song\_cluster\_pipeline.fit(X)

song\_cluster\_labels = song\_cluster\_pipeline.predict(X)

data['cluster\_label'] = song\_cluster\_labels

from sklearn.decomposition import PCA

pca\_pipeline = Pipeline([('scaler', StandardScaler()), ('PCA', PCA(n\_components=2))])

song\_embedding = pca\_pipeline.fit\_transform(X)

projection = pd.DataFrame(columns=['x', 'y'], data=song\_embedding)

projection['title'] = data['name']

projection['cluster'] = data['cluster\_label']

fig = px.scatter(

    projection, x='x', y='y', color='cluster', hover\_data=['x', 'y', 'title'],title='Clusters of songs')

fig.show()

