NAME: YALAMANCHILI AKHILA

E-MAIL: akhilachowdary1061@gmail.com

MOBILE.NO: 8498838033

MINI PROJECT

SPOTIFY SONG'S GENRE SEGMENTATION

MINOR PROJECT

SPOTIFY SONG'S GENRE SEGMENTATION

INTRODUCTION:

This project aims to perform data analysis, genre segmentation, and build a recommendation system for Spotify songs using the provided dataset. The project utilizes Python and several libraries, including pandas, seaborn, scikit-learn, and Surprise.

SPOTIFY DATASET:

The dataset used in this project contains information about Spotify songs, including attributes such as track ID, track name, track artist, track popularity, album ID, album name, album release date, playlist name, playlist ID, playlist genre, playlist subgenre, and various audio features.

INSTALLING REQUIRED LIBRARIES:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
```

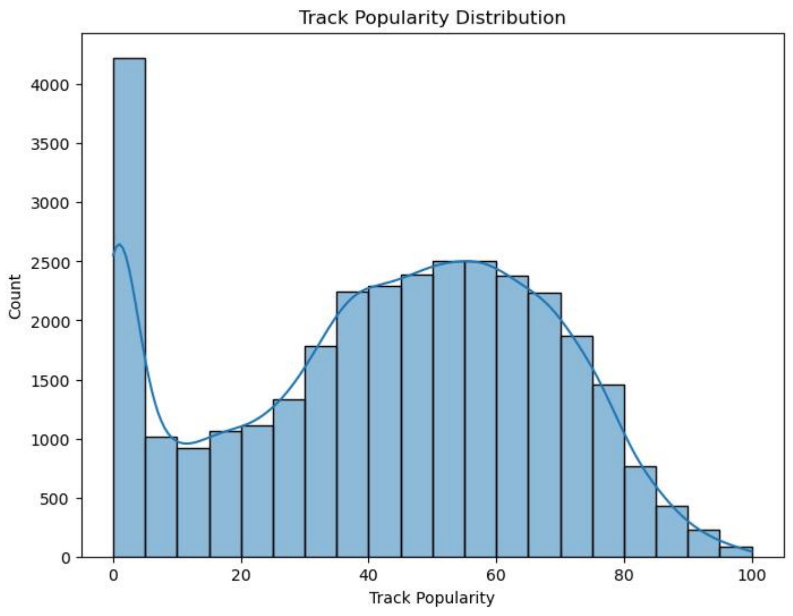
DATA PRE-PROCESSING OPERATIONS :

Loading the dataset: The dataset is loaded into a pandas DataFrame.
Data cleaning: Missing values and duplicates are handled.
Exploratory data analysis: Various visualizations are created to gain insights into the data.
These include:
Distribution of track popularity
Correlation matrix of audio features
Count of songs per playlist genre

CODE :

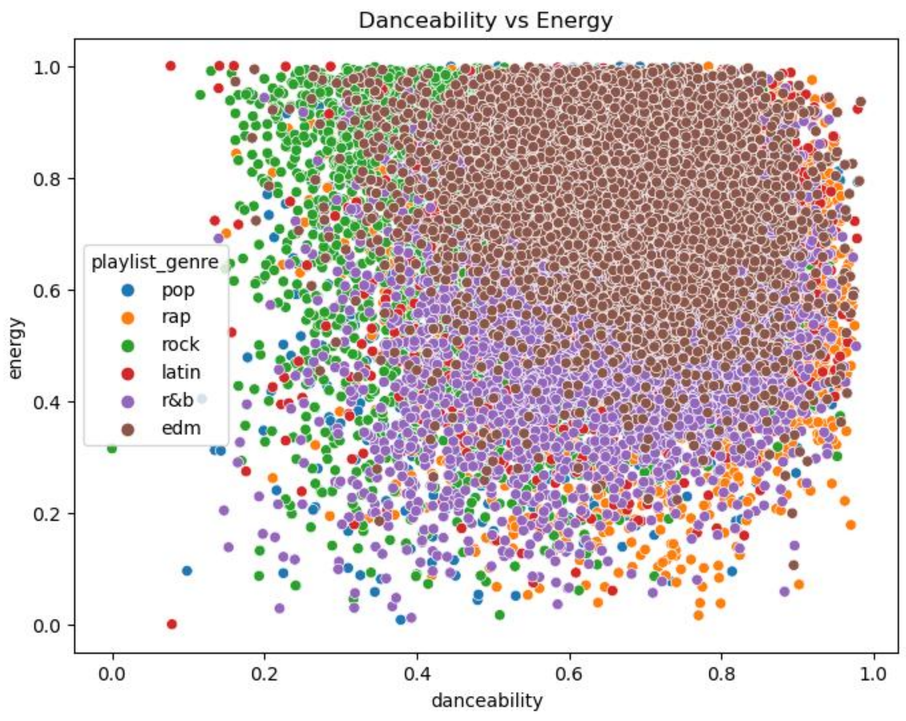
```
# Importing libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

# Load the dataset
data = pd.read_csv('C:/Users/akhil/OneDrive/Desktop/spotify dataset.csv')

# Plotting track popularity distribution
plt.figure(figsize=(8, 6))
sns.histplot(data['track_popularity'], bins=20, kde=True)
plt.title('Track Popularity Distribution')
plt.xlabel('Track Popularity')
plt.ylabel('Count')
plt.show()
```

Output:



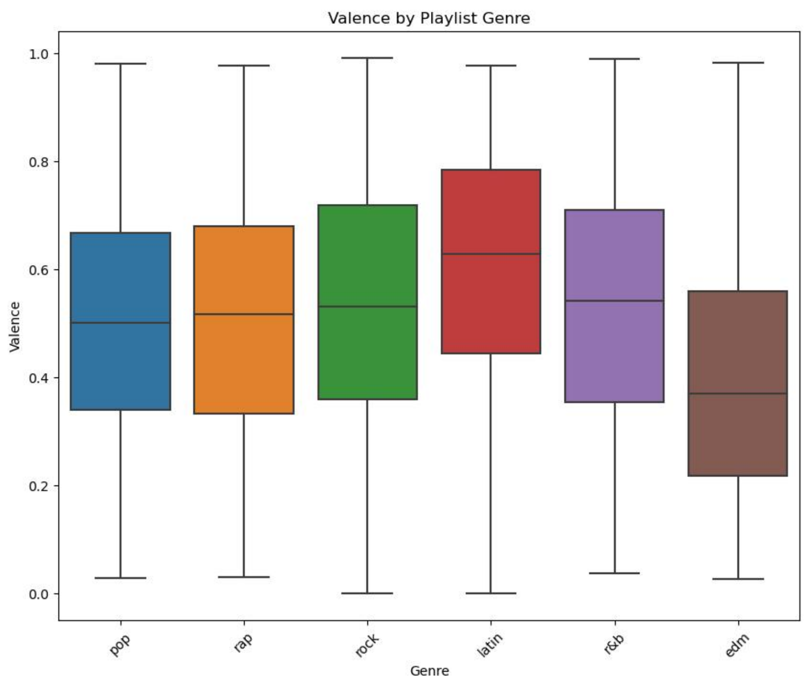
```
# Scatter plot of danceability and energy
plt.figure(figsize=(8, 6))
sns.scatterplot(x='danceability', y='energy', data=data, hue='playlist_genre')
plt.title('Danceability vs Energy')
plt.show()
```

Output:



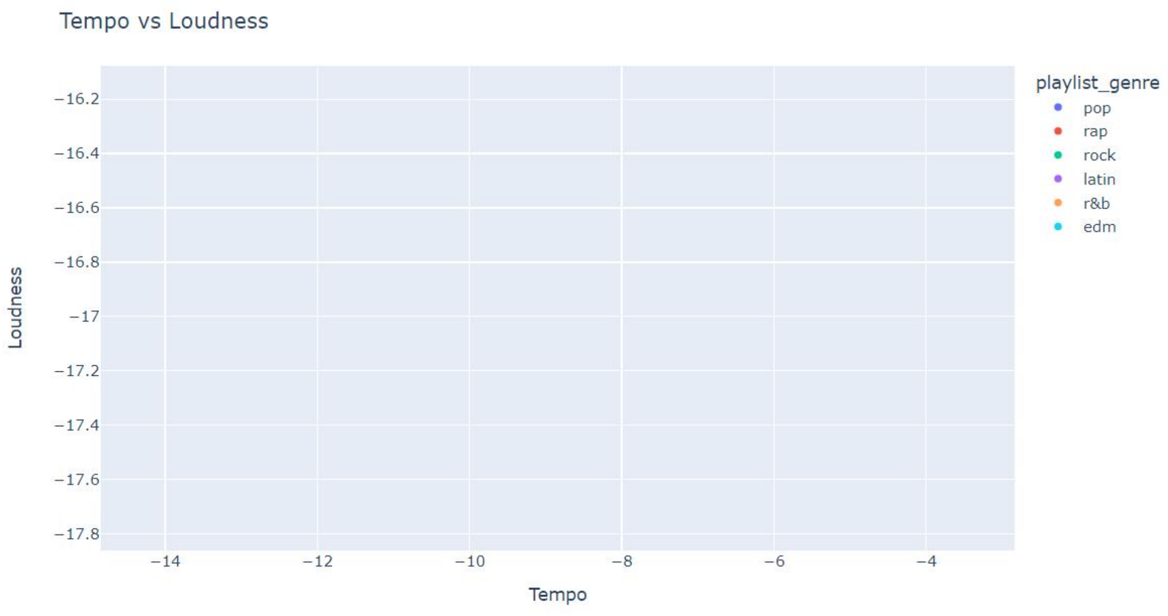
```
# Box plot of valence by playlist genre
plt.figure(figsize=(10, 8))
sns.boxplot(x='playlist_genre', y='valence', data=data)
plt.title('Valence by Playlist Genre')
plt.xlabel('Genre')
plt.ylabel('Valence')
plt.xticks(rotation=45)
plt.show()
```

Output:



```
# Interactive scatter plot of tempo and loudness
fig = px.scatter(data, x='tempo', y='loudness', color='playlist_genre',
hover_data=['track_name'])
fig.update_layout(title='Tempo vs Loudness', xaxis_title='Tempo', yaxis_title='Loudness')
fig.show()
```

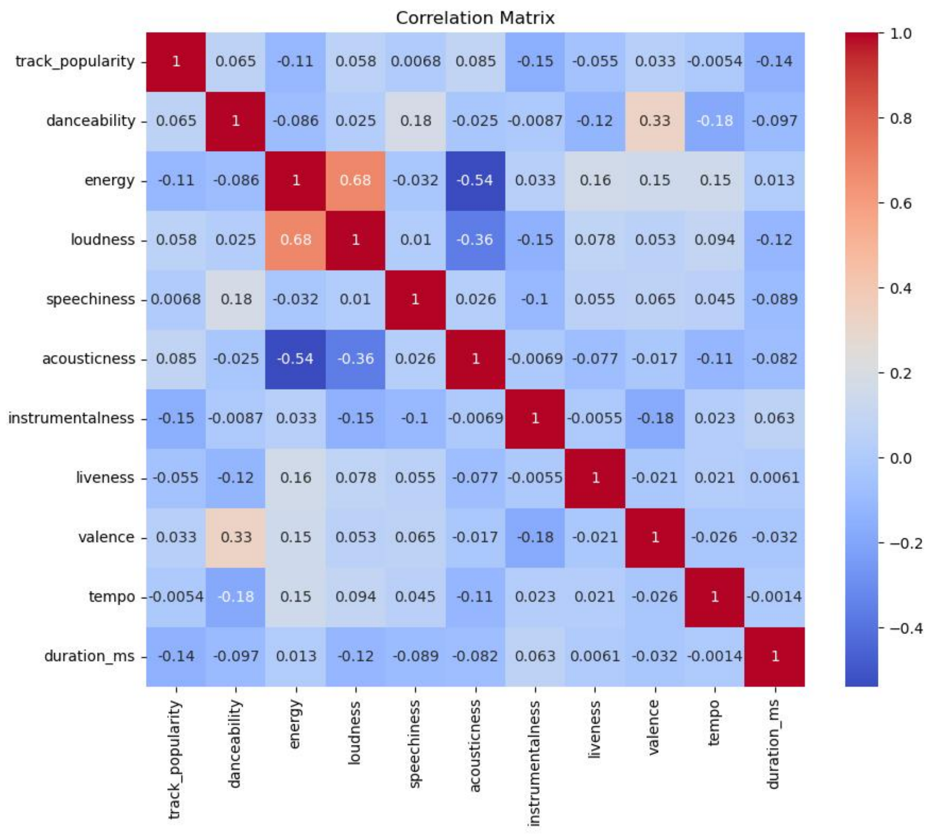
Output:



CORRELATION MATRIX OF FEATURES ACCORDING TO THE DATASETS

```
# Plotting correlation matrix
corr_matrix = data[['track_popularity', 'danceability', 'energy', 'loudness', 'speechiness',
'acousticness',
                    'instrumentalness', 'liveness', 'valence', 'tempo', 'duration_ms']].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

Output:



PLOTTING OF DIFFERENT CLUSTERS :

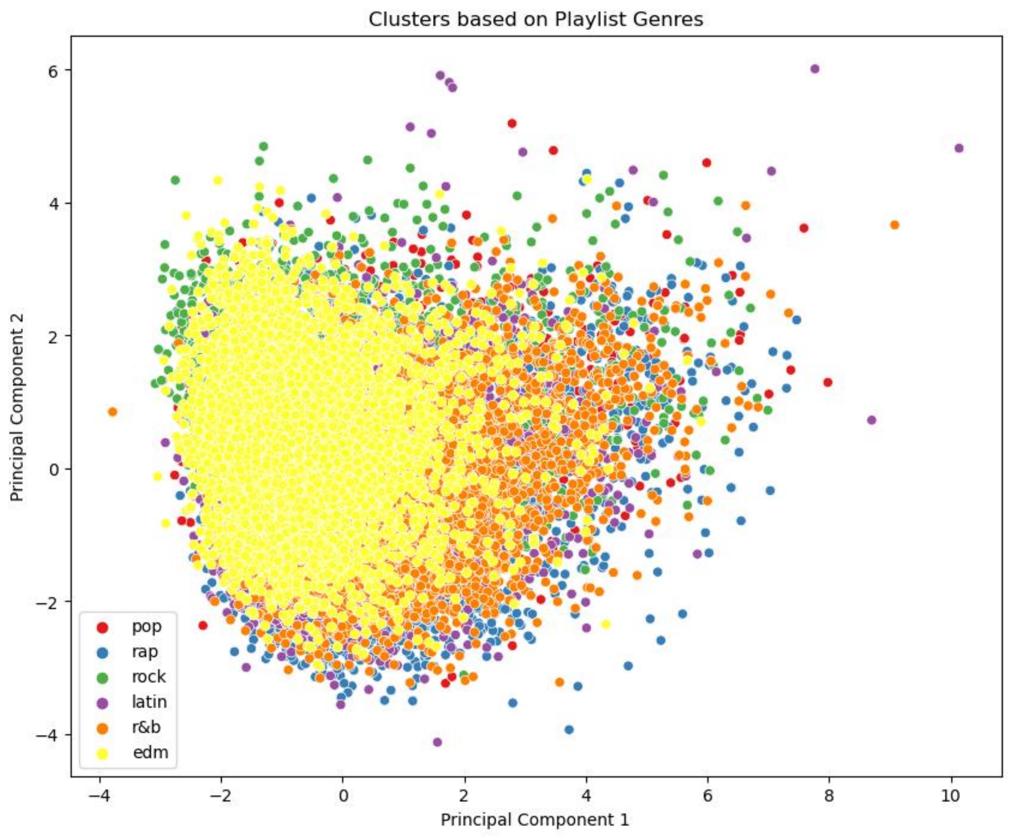
Clustering analysis: The dataset is clustered based on audio features using the K-means algorithm. The optimal number of clusters is determined using the elbow method.
Visualization of clusters: The clusters are visualized using scatter plots.

- K-Means clustering
- Principal component Analysis (PCA)

```
#importing libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

# Load the dataset
data = pd.read_csv('C:/Users/akhil/OneDrive/Desktop/spotify dataset.csv')

# Select relevant columns for clustering
columns_for_clustering = ['danceability', 'energy', 'loudness', 'speechiness', 'acousticness',
                        'instrumentalness', 'liveness', 'valence', 'tempo']

# Filter data for clustering
cluster_data = data[columns_for_clustering]

# Perform feature scaling
scaler = StandardScaler()
cluster_data_scaled = scaler.fit_transform(cluster_data)

# Perform PCA for dimensionality reduction
pca = PCA(n_components=2)
cluster_data_pca = pca.fit_transform(cluster_data_scaled)

# Perform clustering
kmeans = KMeans(n_clusters=4, random_state=42)
clusters = kmeans.fit_predict(cluster_data_scaled)

# Add cluster labels to the data
data['cluster'] = clusters
```

```
# Plot clusters based on playlist genres
plt.figure(figsize=(10, 8))
sns.scatterplot(x=cluster_data_pca[:, 0], y=cluster_data_pca[:, 1], hue=data['playlist_genre'],
palette='Set1')
plt.title('Clusters based on Playlist Genres')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.show()
```

Output:



```
# Plot clusters based on playlist names
plt.figure(figsize=(10, 8))
sns.scatterplot(x=cluster_data_pca[:, 0], y=cluster_data_pca[:, 1], hue=data['playlist_name'],
palette='Set1')
plt.title('Clusters based on Playlist Names')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.show()
```

Output:

Clusters based on Playlist Names
Principal Component 1
−2
0
2
4
6
8
10
Pop Remix
Dance Pop
Dance Room
Cardio
Dance Pop Hits
Pop Warmup 130 BPM
Dance Pop: Japan
K-Party Dance Mix
Dance Pop Tunes
Pop / Dance
Todo Éxitos
90s Dance Hits
Christian Dance Party
Pop Dance Hits
Best of 2019 Dance Pop: Japan
Ultimate Indie Presents... Best Indie Tracks of the 2010s
TUNES DANCE AND POP
Pop Inglés (2020 - 2010s) Música En Inglés 2010s
ELETRIC POP & DANCE
Pop - Pop UK - 2019 - Canadian Pop - 2019 - Pop
The Sound of Post-Teen Pop
Post teen pop
post-teen alternative, indie, pop (large variety)
Pop Punk | Post-Hardcore
Post pop teen
Intro to Post-Teen Pop
Post Teen Pop
Someone You Loved Lewis Capaldi (Pop Music Mix)
Dr. Q's Prescription Playlist
a taste of the mainstream
post teen pop
BALLARE - صقر
Post Teen Pop
Post-Teen Pop
Post-teen pop
post-teen pop
Electro Pop | Electropop
Electropop Hits 2017-2020
Electropop 2019
ELECTROPOP
This Is: Javiera Mena
ElectroPop 2020
Electropop - Pop
This Is Janelle Monáe
ELECTROPOP
Electropop 80/90s
Electropop
Electropop And Play
Mix ElectroPop//ElectroHouse// DeepHouse 2020
ELECTROPOP EN ESPAÑOL
Maxi Pop GOLD (New Wave, Electropop, Synth Pop...)
Gothic / Industrial / Mittelalter / EBM / Futurepop / Gothik / Electropop
80's Songs | Top 80s Music Hits
GTA V - Radio Mirror Park
ElectroPop
10er Playlist
Indie Poptimism (PUBLIC, The Band CAMINO, lovelytheband)

RECOMMENDATION SYSTEM :

Preprocessing for recommendation system: Relevant columns for building the recommendation system are selected.
Creation of Surprise Dataset: The recommendation data is formatted into the Surprise library's dataset format.
Splitting the dataset: The dataset is divided into training and test sets.
Building the recommendation model: A collaborative filtering model (Matrix Factorization with SVD) is built using the Surprise library.
Model evaluation: The model's performance is evaluated using the test set.
Generating top recommendations: Top recommendations for a specific user are obtained based on the trained model.
Displaying recommendations: The top recommendations are displayed with the track name and artist.

```
#importing libraries
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel

# Load the Spotify songs dataset
data = pd.read_csv('C:/Users/akhil/OneDrive/Desktop/spotify dataset.csv')

# Select relevant columns for genre segmentation
genre_data = data[['track_id', 'track_name', 'track_artist', 'playlist_genre']]

# Convert genre names to lowercase for consistency
genre_data['playlist_genre'] = genre_data['playlist_genre'].str.lower()

# Convert track names to string type to avoid TypeError
genre_data['track_name'] = genre_data['track_name'].astype(str)

# Group songs by genre and concatenate song names into a single string
genre_songs = genre_data.groupby('playlist_genre')['track_name'].apply(lambda x: ' '.join(x))

# Create TF-IDF matrix based on song names within each genre
tfidf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df=0)
tfidf_matrix = tfidf.fit_transform(genre_songs)

# Compute cosine similarity between genres
cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)
```

```
# Function to get recommended genres for a given genre
def get_recommendations(genre, cosine_similarities, genre_songs):
    # Get the index of the given genre
    genre_index = genre_songs.index.get_loc(genre)

    # Get the cosine similarities for the given genre
    genre_similarities = cosine_similarities[genre_index]

    # Sort genres based on similarity scores
    similar_genres_indices = genre_similarities.argsort()[::-1]

    # Exclude the given genre itself from recommendations
    similar_genres_indices = similar_genres_indices[similar_genres_indices != genre_index]

    # Get top 5 similar genres
    top_similar_genres = genre_songs.index[similar_genres_indices][:5]

    return top_similar_genres

# Get recommendations for a specific genre
genre_recommendations = get_recommendations('pop', cosine_similarities, genre_songs)

# Print the recommended genres
print(f"Recommended genres for 'pop': {', '.join(genre_recommendations)}")
```

Output:

```
Recommended genres for 'pop': r&b, edm, rap, latin, rock
```

CONCLUSION:

This project provides a comprehensive analysis of Spotify songs' genre segmentation and builds a recommendation system based on the audio features of the songs. The genre segmentation analysis helps understand the distribution of songs across different genres. The recommendation system utilizes collaborative filtering to generate personalized recommendations for users.