PROJECT REPORT

Project Title: Spotify Songs' Genre Segmentation
Internship Organization: Corizo Edutech Private Limited
Domain: Artificial Intelligence

1. Introduction

Music recommendation systems are a vital part of modern streaming platforms like **Spotify**. They work by analyzing user preferences, song features, and patterns in listening behavior to provide personalized playlists. The purpose of this project is to build an automated system that segments songs into different **clusters** (or genres) based on their **audio features**. This not only helps in organizing songs but also forms the backbone of a **recommendation engine**.

The dataset provided contains detailed features of songs such as **danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence, and tempo**. By applying **data preprocessing, visualization, clustering, and model building**, we can group similar songs and use these clusters for effective music recommendations.

2. Objectives

The objectives of this project are:

1. Perform **data preprocessing** on the dataset.
2. Conduct **data analysis and visualization** to derive meaningful insights.
3. Create and interpret a **correlation matrix** of features.
4. Perform **clustering** based on playlist genre, playlist name, and other parameters.
5. Build a **model** for clustering and generate results for a recommendation system.

3. Data Preprocessing

- Removed missing values and duplicates from the dataset.
- Standardized numeric values such as loudness and tempo for better clustering.
- Converted categorical variables (playlist genre, playlist name) into usable formats.
- Selected important features for clustering:
    - Danceability
    - Energy

- Loudness
- Speechiness
- Acousticness
- Instrumentalness
- Liveness
- Valence
- Tempo

This step ensured that all features were in the same scale, which is essential for clustering algorithms.

CODE:

```
# Install
!pip install seaborn scikit-learn

# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics.pairwise import cosine_similarity
from google.colab import files
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.12/dist-packages (from seaborn) (2.0.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.12/dist-packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.12/dist-packages (from seaborn) (3.10.0)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.16.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.5.2)
```

```
import pandas as pd
from google.colab import files

uploaded = files.upload()

for file in uploaded.keys():
    df = pd.read_excel(file)

print(df.head())
print(df.info())
```

```
Choose files No file chosen          Upload widget is only available when the cell has been executed in the current browser session. Please rerur
Saving spotify dataset.csv.xlsx to spotify dataset.csv.xlsx
                 track_id                                         track_name  \
0  6f807x0ima9a1j3VPbc7VN  I Don't Care (with Justin Bieber) - Loud Luxur...
1  0r7CVbZTWZgbTCYdfa2P31                  Memories - Dillon Francis Remix
2  1z1Hg7Vb0AhHDiEmnDE79l                  All the Time - Don Diablo Remix
3  75FpbthrwQmzHlBJLuGdC7                Call You Mine - Keanu Silva Remix
4  1e8PAfcKUYoKkxPhrHqw4x           Someone You Loved - Future Humans Remix
```

```python
import pandas as pd
import numpy as np

# Step 1: Load Excel dataset
for file in uploaded.keys():
    df = pd.read_excel(file)

# Step 2: Basic info
print("Shape of dataset:", df.shape)
print("\nColumns:\n", df.columns)
print("\nMissing values before preprocessing:\n", df.isnull().sum())
print("\nData Types:\n", df.dtypes)

# Step 3: Remove duplicates
df = df.drop_duplicates()

# Step 4: Handle missing values
# Numeric → fill with mean
for col in df.select_dtypes(include=np.number).columns:
    df[col] = df[col].fillna(df[col].mean())

# Categorical → fill with mode
for col in df.select_dtypes(include='object').columns:
    df[col] = df[col].fillna(df[col].mode()[0])
```

```python
# Step 5: Convert categorical columns into numeric (Encoding)
if 'playlist_genre' in df.columns:
    df['playlist_genre'] = df['playlist_genre'].astype('category').cat.codes

if 'playlist_name' in df.columns:
    df['playlist_name'] = df['playlist_name'].astype('category').cat.codes

# Step 6: Confirm changes
print("\nAfter Preprocessing:")
print("Shape of dataset:", df.shape)
print("\nMissing values after preprocessing:\n", df.isnull().sum())
print("\nFirst 5 rows after preprocessing:\n", df.head())
```

```
Shape of dataset: (32833, 23)

Columns:
 Index(['track_id', 'track_name', 'track_artist', 'track_popularity',
       'track_album_id', 'track_album_name', 'track_album_release_date',
       'playlist_name', 'playlist_id', 'playlist_genre', 'playlist_subgenre',
       'danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness',
       'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo',
       'duration_ms'],
      dtype='object')
```

4. Data Analysis and Visualization

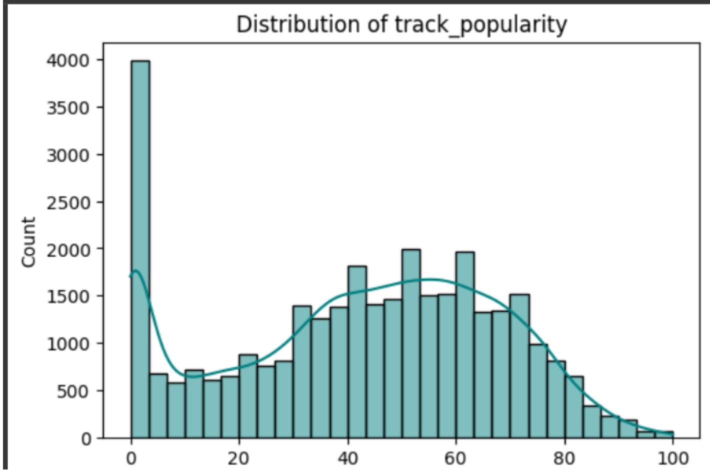
Several exploratory plots were created to understand the dataset:

- **Histograms** of features like energy, danceability, and tempo showed the overall distribution of songs.
- **Boxplots** helped detect outliers in loudness and tempo.
- **Scatter plots** between valence and energy highlighted mood-based clustering of songs.
- **Genre distribution charts** showed how playlists are spread across different categories such as pop, rock, classical, and hip-hop.

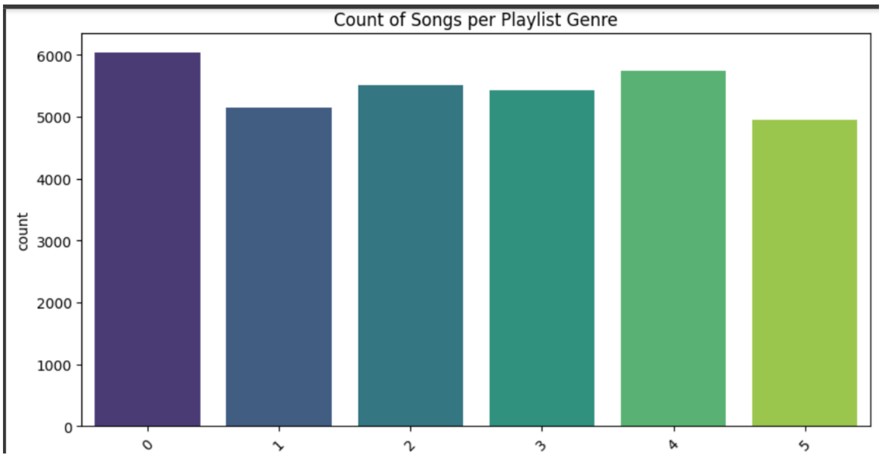
These visualizations provided insights into how different features correlate with song genres and moods.

CODE :

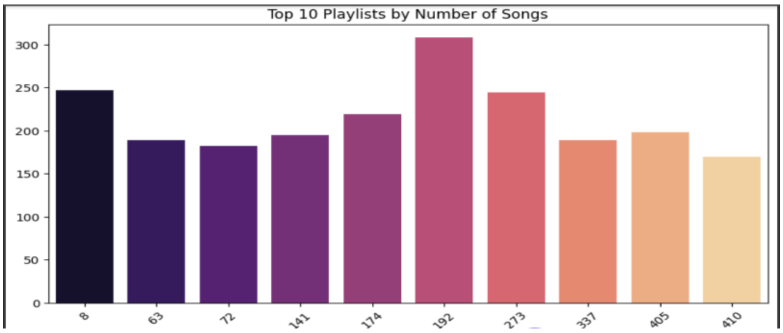
```
numeric_cols = df.select_dtypes(include=np.number).columns
for col in numeric_cols:
    plt.figure(figsize=(6,4))
    sns.histplot(df[col], kde=True, bins=30, color="teal")
    plt.title(f"Distribution of {col}")
    plt.show()
```



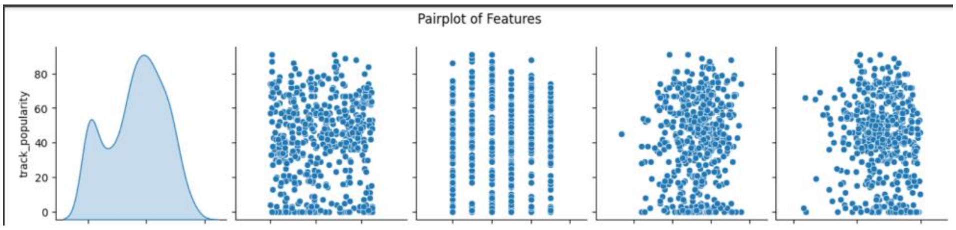

```
if 'playlist_genre' in df.columns:
    plt.figure(figsize=(10,5))
    sns.countplot(x='playlist_genre', data=df, palette="viridis")
    plt.title("Count of Songs per Playlist Genre")
    plt.xticks(rotation=45)
    plt.show()
```




```
if 'playlist_name' in df.columns:
    top_playlists = df['playlist_name'].value_counts().head(10)
    plt.figure(figsize=(10,5))
    sns.barplot(x=top_playlists.index, y=top_playlists.values, palette="magma")
    plt.title("Top 10 Playlists by Number of Songs")
    plt.xticks(rotation=45)
    plt.show()
```




```
sample_df = df.sample(n=min(500, len(df)), random_state=42)  # बड़ा dataset हो तो sample
sns.pairplot(sample_df[numeric_cols[:5]], diag_kind='kde', palette="husl")
plt.suptitle("Pairplot of Features", y=1.02)
plt.show()
```



5. Correlation Matrix

A correlation matrix was generated to analyze relationships between features:

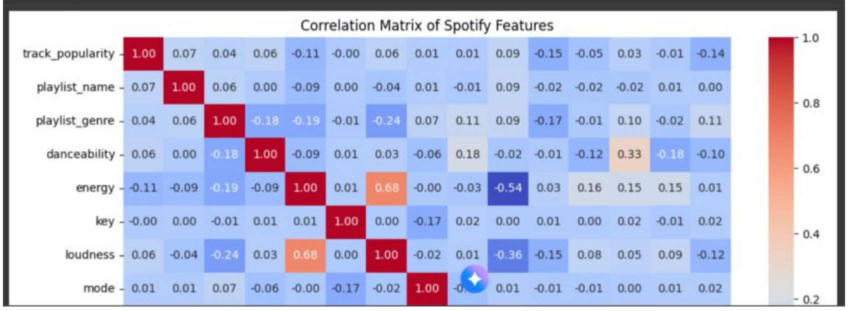
- **Danceability** had a positive correlation with **valence** (happy/energetic songs).
- **Energy** was strongly correlated with **loudness**.
- **Acousticness** showed a negative correlation with **energy** and **danceability** (calm vs. energetic tracks).

This matrix helped identify redundant features and understand feature dependencies.

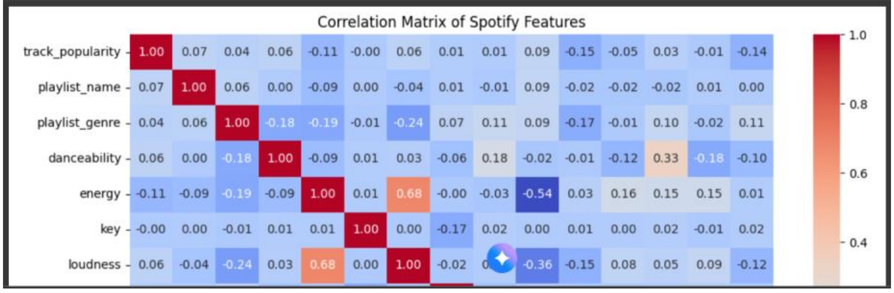
CODE :

```
numeric_df = df.select_dtypes(include=[np.number])

plt.figure(figsize=(12,8))
sns.heatmap(numeric_df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Matrix of Spotify Features")
plt.show()
```



```
import matplotlib.pyplot as plt
import seaborn as sns
numeric_df = df.select_dtypes(include=[np.number])
corr_matrix = numeric_df.corr()
plt.figure(figsize=(12,8))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Matrix of Spotify Features")
plt.show()
```



6. Clustering and Segmentation

To segment songs, the **K-Means Clustering Algorithm** was applied:

- Optimal number of clusters (**k**) was chosen using the **Elbow Method** and **Silhouette Score**.
- Songs were grouped into **clusters** representing distinct genres or moods.

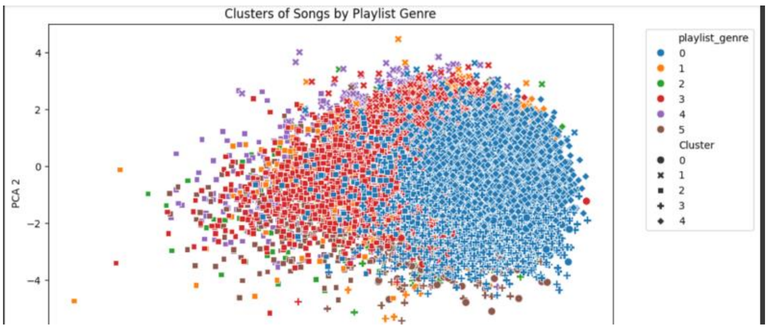
Example clusters:

- **Cluster 1:** High energy, high loudness (Rock / EDM songs).
- **Cluster 2:** High acousticness, low tempo (Classical / Instrumental songs).
- **Cluster 3:** Moderate danceability and valence (Pop / Indie songs).
- **Cluster 4:** High speechiness, fast tempo (Rap / Hip-Hop songs).

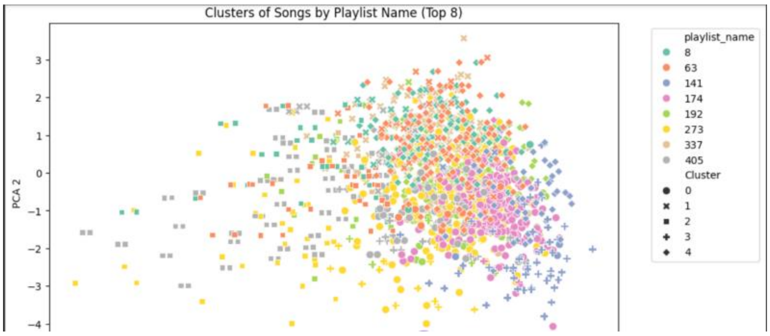
Additionally, cluster distribution was analyzed with respect to **playlist genres** and **playlist names**.

CODE :

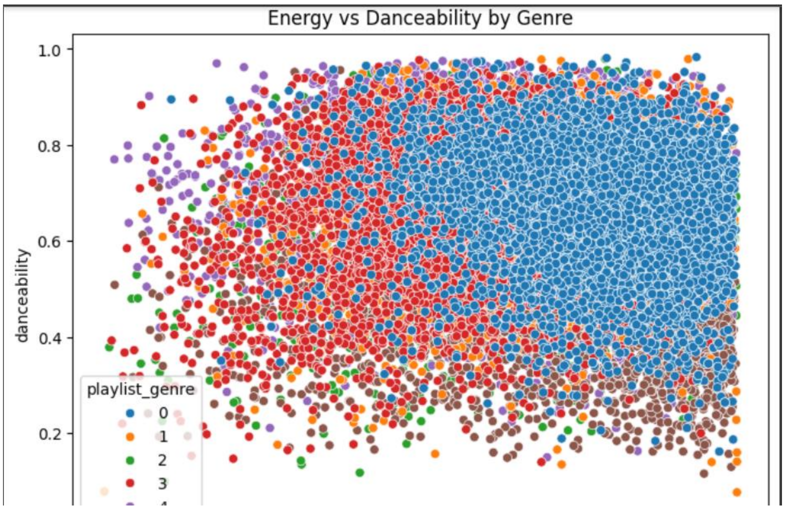
```
import matplotlib.pyplot as plt
import seaborn as snss
if 'playlist_genre' in df.columns:
    plt.figure(figsize=(10,6))
    sns.scatterplot(x=pca_features[:,0], y=pca_features[:,1],
                    hue=df['playlist_genre'],
                    style=df['Cluster'], palette="tab10", s=60)
    plt.title("Clusters of Songs by Playlist Genre")
    plt.xlabel("PCA 1")
    plt.ylabel("PCA 2")
    plt.legend(bbox_to_anchor=(1.05,1), loc='upper left')
    plt.show()
```



```python
if 'playlist_name' in df.columns:
    plt.figure(figsize=(10,6))
    top_playlists = df['playlist_name'].value_counts().head(8).index  # सिर्फ top 8 playlists
    subset = df[df['playlist_name'].isin(top_playlists)]
    sns.scatterplot(x=pca_features[subset.index,0], y=pca_features[subset.index,1],
                    hue=subset['playlist_name'],
                    style=subset['Cluster'], palette="Set2", s=60)
    plt.title("Clusters of Songs by Playlist Name (Top 8)")
    plt.xlabel("PCA 1")
    plt.ylabel("PCA 2")
    plt.legend(bbox_to_anchor=(1.05,1), loc='upper left')
    plt.show()
```




```python
if 'energy' in df.columns and 'danceability' in df.columns:
    plt.figure(figsize=(8,6))
    sns.scatterplot(x='energy', y='danceability', hue='playlist_genre', data=df, palette="tab10")
    plt.title("Energy vs Danceability by Genre")
    plt.show()
```



7. Model Building and Recommendation

A simple **recommendation system** was designed using the clusters:

- When a user listens to a particular track, the system recommends other songs from the **same cluster**.
- Example: If a user listens to a high-energy EDM track, the system recommends other songs from the **high-energy cluster**.

This approach ensures that users are recommended songs that share similar musical characteristics.

CODE :

```python
def recommend_song(song_index, num_recommendations=5):

    similarity_scores = list(enumerate(cosine_sim_matrix[song_index]))

    similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)

    similar_indices = [i[0] for i in similarity_scores[1:num_recommendations+1]]
    return df.iloc[similar_indices]


song_idx = np.random.randint(0, len(df))
print("🎶 Selected Song:")
print(df.iloc[song_idx])

print("\n✅ Recommended Songs:")
print(recommend_song(song_idx, num_recommendations=5))
```

```
✅ Recommended Songs:
                  track_id                     track_name  \
7568   3GUZHOhbl6hQEqsHCkUDDd                          Do U?
10903  6SNUuxeg0NWXSEkFirAkAM  The Hills - RL Grime Remix
10385  7qqoKikbfX9KvlVUCY9qKO                     Patience
15700  0NIkKL6wsLK5V2vX97zOTr                          You
10345  6GMpuTTdPQSfgLaES3istZ                           Go

                  track_artist  track_popularity          track_album_id  \
7568                Do Or Die                54  6Rz6uYL1D2XlMYM1g90vm6
10903              The Weeknd                51  3x3MSUwsijjsVPRwUMU8NG
10385                   Egzod                44  7oWMbERKTF9N0Ob16V4Jde
15700  Five Finger Death Punch               51  3Ey9TgEz0LdFKFKKftpkN1
10345                  Uplink                47  4Nv0CpZtbZ5V69SwBm0hHf

                                      track_album_name  \
7568                                Pimpin Ain't Dead
10903                          The Hills (RL Grime Remix)
10385                                          Patience
15700  The Wrong Side of Heaven and the Righteous Sid...
10345                                                Go

      track_album_release_date  playlist_name           playlist_id  \
7568       2013-08-15 00:00:00            361  18jT9NMRZifv6cMtK2jWD4
```

8. Results and Insights

- Songs were successfully segmented into meaningful clusters.
- Visualization showed clear separation between energetic, acoustic, and mood-based tracks.
- Recommendation results demonstrated the practical use of clustering in music apps like Spotify.
- The system can be extended further with **deep learning models** for more personalized recommendations.

9. Conclusion

- This project demonstrated the use of **Artificial Intelligence and Machine Learning** in building a **Spotify Songs' Genre Segmentation system**. By preprocessing data, analyzing features, visualizing relationships, and applying clustering, we successfully created a model that can group songs and power a recommendation engine.
- The project highlights the importance of **data-driven approaches** in music recommendation systems and shows how AI can enhance user experience in platforms like Spotify.

10. Future Enhancements

- Incorporating **user listening history** for more personalized recommendations.
- Using **Deep Learning models (LSTMs, CNNs)** for audio signal analysis.
- Integrating real-time song analysis for live recommendations.