Tech Saksham

Capstone Project Report

“Spotify Music Recommendation System”

“Universal College of Engineering and Technology,vallioor”

NM ID	NAME
au962721103002	Aruna K

Ramar Bose

Sr. AI Master Trainer

ABSTRACT

A recommendation system for music and song recommendations is a project that uses machine learning algorithms K-means clustering algorithms to analyse data on user's listening habits and recommend new songs that they may be interested in. Recommendation systems are widely used in the music industry. One of the reasons they have become so ubiquitous is due to the fact that online listener behavior is characterized by cognitive biases - users prefer to take mental shortcuts rather than evaluate a large range of music choices on a daily basis. It provides these mental shortcuts by offering personalised recommendations based on the user’s preferences..

1. Problem state.
2. Data collection
3. Existing solution
4. Proposed solution with used models
5. Result

INDEX

25 pages

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

In today's competitive world, understanding customer behavior and preferences is crucial for customer retention and revenue generation. However, spotify music recommendations system often face challenges in analyzing customer data due to the sheer volume and velocity of data generated. Traditional data analysis methods are time-consuming and often fail to provide real-time insights. This lack of real-time analysis can lead to missed opportunities for customer engagement and customer satisfaction. Furthermore, the complexity and diversity of customer data, which includes customer details and their preferences.

1.2 Proposed Solution

The proposed solution is to develop a spotify music recommendations system using K-means clustering algorithm. The recommendation system clusters songs by implementing K-Means using sklearn library and generates recommendations according to the clusters. It has the following principal steps:

1.Find optimal number of clusters using the Elbow method
2.Fit the K-means model
3.Add a column with the corresponding clusters
4.Find out the maximum occurring cluster number according to user's favorite track types
5.Sort the cluster numbers and find out the number which occurs the most
6.Get the tracks of that cluster and print the first five rows of the dataframe having that cluster number as their type

1.3 Feature

- **Real-Time Analysis**: The dashboard will provide real-time analysis of customer data.
- **Customer Segmentation**: It will segment customers based on their preference songs
- **Trend Analysis**: The dashboard will identify and display trends in customer behavior.
- **Predictive Analysis**: It will use historical data to predict future customer behavior.

1.4 Advantages

- **Data-Driven Decisions**: Spotify music recommendations system makes decisions based on real-time data analysis.
- **Improved Customer Engagement**: Understanding customer behavior and trends can help spotify music recommendationsystem engage with their customers more effectively.
- **Customer Satisfactions**: By listing their preferences songs, customer will get satisfaction.

1.5 Scope

The scope of this project extends to spotify music recommendation sytem that aim to leverage data for decision-making and customer engagement. The project can be further extended to incorporate more data sources and advanced analytics techniques, such as machine learning and artificial intelligence, to provide more sophisticated insights into customer behavior. The project also has the potential to be adapted for other sectors, such as online marketing where understanding customer behavior is crucial. Furthermore, the project contributes to innovation, and customer satisfaction.

CHAPTER 2

SERVICES AND TOOLS REQUIRED

2.1 LR - Exiting Models

2.1 Required – System config | Spotify Music Recommendations System

2.1 Services Used

- **Data Collection and Storage Services**: Spotify music recommendations system need to collect and store customer data in real-time. During the Extract phase, PySpark was used to read and extract the relevant data from the datasets. The main dataset has been deployed to HerokuSQL Cloud Database and all model-related files retrive data from there. Also, Deta Space cloud database is used to store the Bayesian Personalized Ranking model parameteres.

- **Data Processing Services**: PySpark was used to transform the extracted data into a suitable format for merging and analysis.
- **Machine Learning Services**: The Bayesian Personalized Ranking (BPR) model has been deployed using FastAPI, Github Actions, and Deta as an API hosted on Heroku..

2.2 Tools and Software used

Tools:

1.Docker Python is an open-source tool and a standard shipping container. This tool is used for automating the deployment of any application inside a software container.

2. Visual Studio Code

Software Requirements:

Python uses the following packages for this project.

1. **requests**: Python requests is a library for making HTTP requests. It provides an easy-to-use interface that makes working with HTTP very simple, which means it simplifies the process of sending and receiving data from websites by providing a uniform interface for both GET and POST methods.
2. **Spotipy**:Spotipy is a lightweight Python library for the Spotify Web API. With Spotipy you get full access to all of the music data provided by the Spotify platform.
3. **Streamlit**:Streamlit is an open-source Python framework for data scientists and AI/ML engineers to deliver dynamic data apps

4. **IPython**:IPython itself *is* focused on interactive Python, part of which is providing a Python kernel for Jupyter
5. **Pandas**: Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data

For FrontEnd,the below listed are used.

1.HTML

2.JavaScript File (.js)

CHAPTER 3

PROJECT ARCHITECTURE

3.1 Architecture

1. System flow diagram

2. Data flow diagram

3. Modules

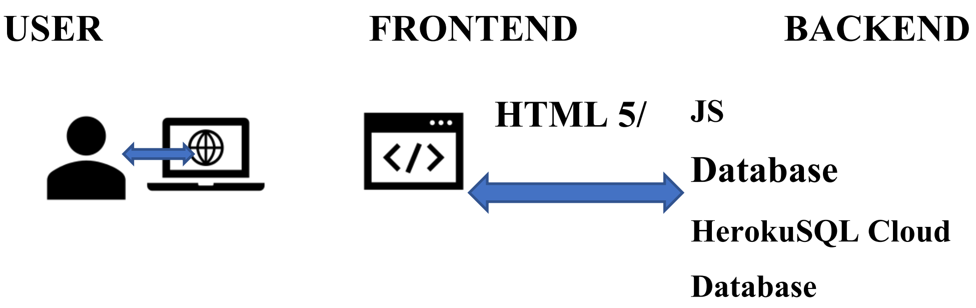
1. User interface

2. Next Module (EDA) flow diagram

3. Training model diagram

4. Predicting model's diagram

5. Model Performance evaluation models



Here’s a high-level architecture for the project:

1. **Data Collection**: During the Extract phase, PySpark was used to read and extract the relevant data from the datasets.

2. **Data Storage**: The main dataset has been deployed to HerokuSQL Cloud Database and all model-related files retrive data from there. Also, Deta Space cloud database is used to store the Bayesian Personalized Ranking model parameteres.

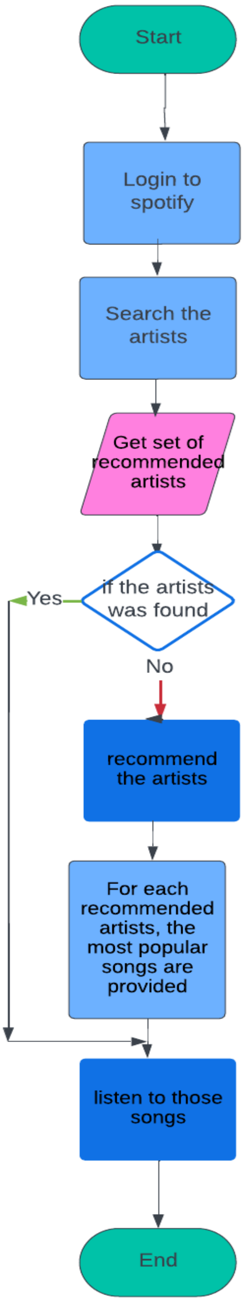
3. **Data Processing**: PySpark was used to transform the extracted data into a suitable format for merging and analysis

4. **Machine Learning**: The Bayesian Personalized Ranking (BPR) model has been deployed using FastAPI, Github Actions, and Deta as an API hosted on Heroku.
.

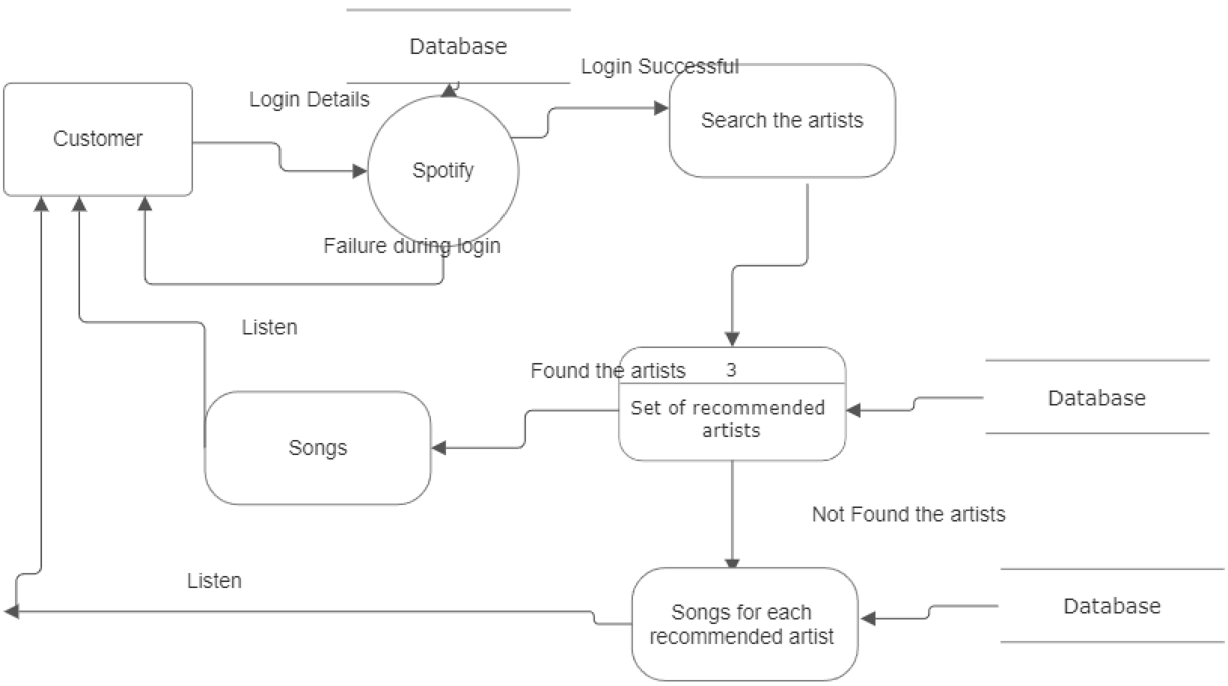
5. **Data Visualization& Data Access**:

- Log In to Spotify
- Search the artist.
- Get set of recommended artists using Bayesian Personalized Ranking (BPR) model deployed.
  - If artist was not found in the database, recommendation using Spotify recommender is retrieved.
- For each recommended artists, the most popular songs are provided.
- User can listen to those songs directly inside the web application.

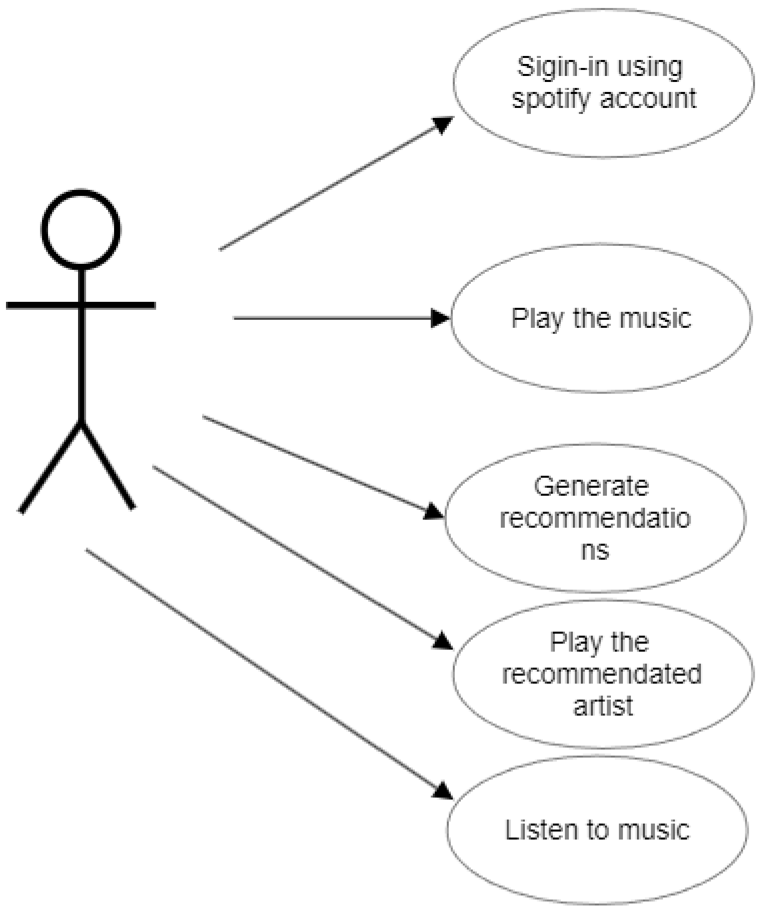
1.System flow diagram:


Start
Login to spotify
Search the artists
Get set of recommended artists
if the artists was found
Yes
No
recommend the artists
For each recommended artists, the most popular songs are provided
listen to those songs
End

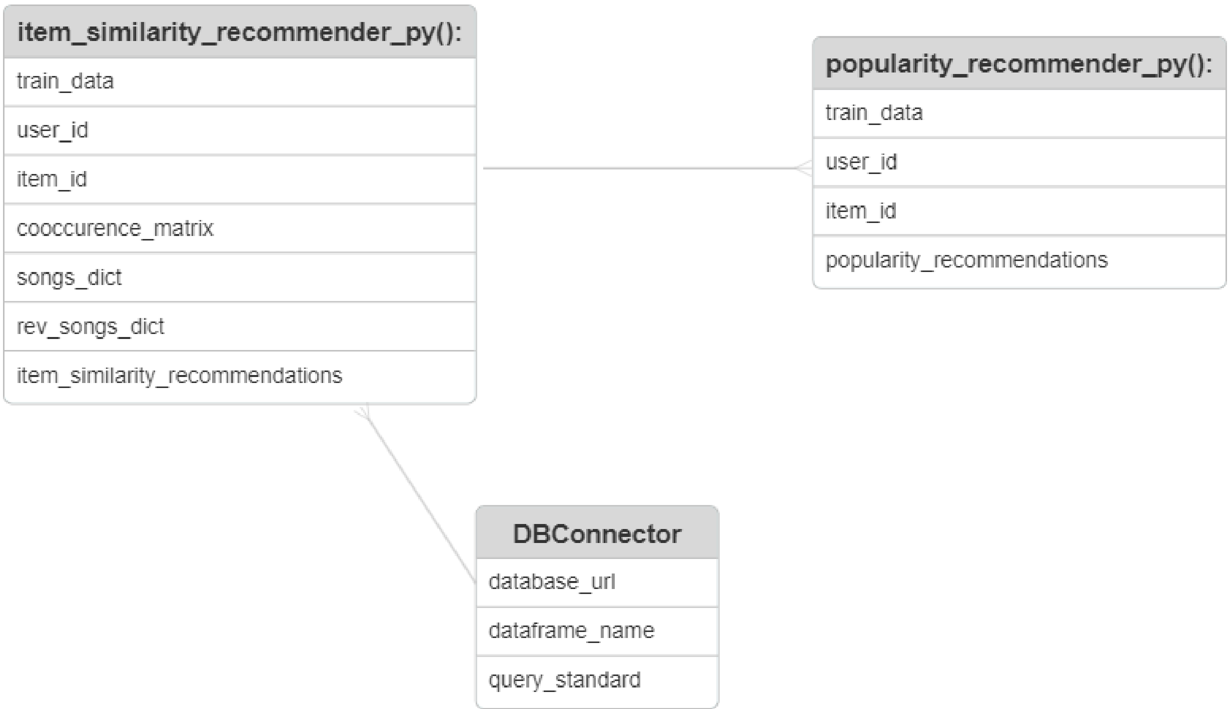
2.Data flow diagrams



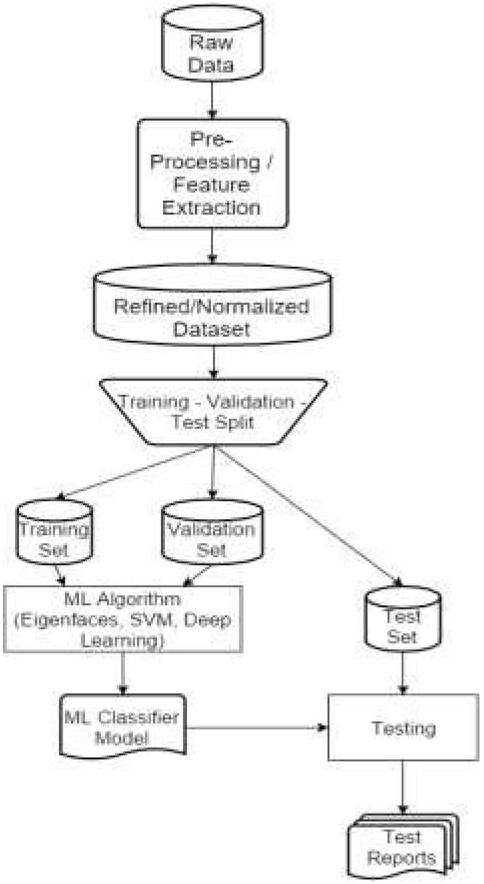
1.User Interface



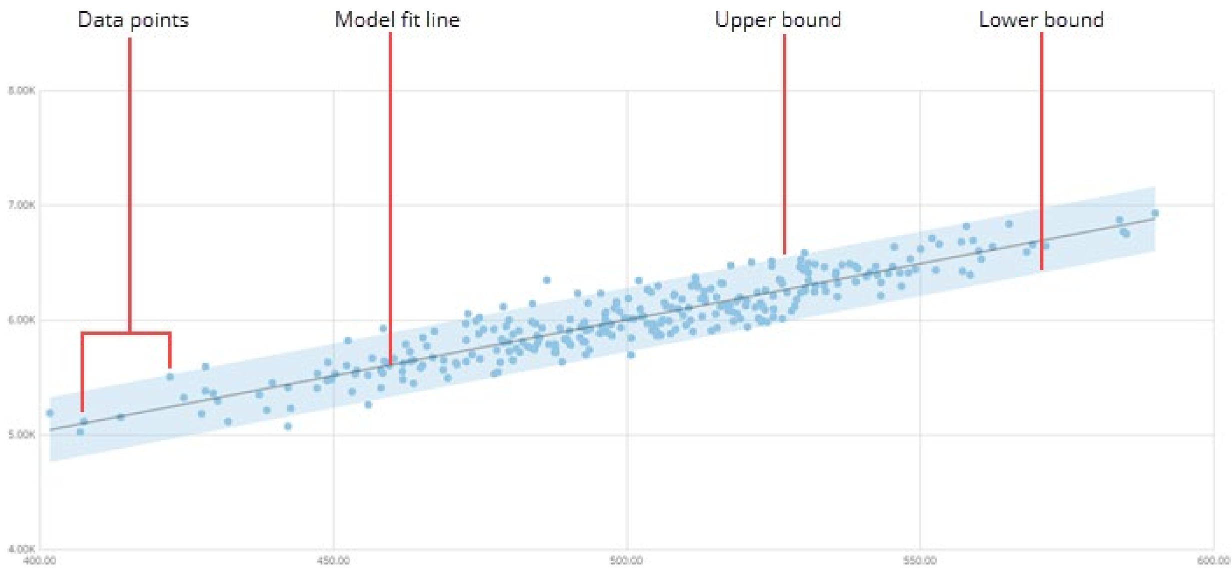
2. Next Module (EDA) flow diagram:



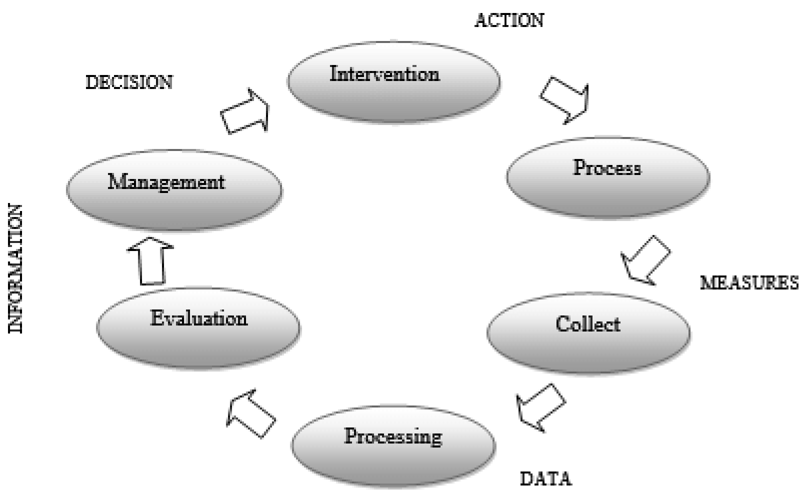
3.Training Model Diagram



4.Predicting model's diagram



6. Model Performance evaluation models



CHAPTER 4

MODELING AND PROJECT OUTCOME

(code& result)

Import libraries:

```
importos
import numpy asnp
import pandas aspd

import seaborn assns
import plotly.express aspx
import matplotlib.pyplot asplt
%matplotlib inline

from sklearn.cluster importKMeans
from sklearn.preprocessing importStandardScaler
from sklearn.pipeline importPipeline
from sklearn.manifold importTSNE
from sklearn.decomposition importPCA
from sklearn.metrics importeuclidean_distances
from scipy.spatial.distance importcdist

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

Data load:

Code:

```
data = pd.read_csv("../input/spotify-dataset/data/data.csv")
genre_data = pd.read_csv('../input/spotify-dataset/data/data_by_genres.csv')
year_data = pd.read_csv('../input/spotify-dataset/data/data_by_year.csv')
print(data.info())
```

Output:

```
RangeIndex: 170653 entries, 0 to 170652
Data columns (total 19 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   valence           170653 non-null  float64
 1   year              170653 non-null  int64
 2   acousticness      170653 non-null  float64
 3   artists           170653 non-null  object
 4   danceability      170653 non-null  float64
 5   duration_ms       170653 non-null  int64
 6   energy            170653 non-null  float64
 7   explicit          170653 non-null  int64
 8   id                170653 non-null  object
 9   instrumentalness  170653 non-null  float64
 10  key               170653 non-null  int64
 11  liveness          170653 non-null  float64
 12  loudness          170653 non-null  float64
 13  mode              170653 non-null  int64
 14  name              170653 non-null  object
 15  popularity        170653 non-null  int64
 16  release_date      170653 non-null  object
 17  speechiness       170653 non-null  float64
 18  tempo             170653 non-null  float64
dtypes: float64(9), int64(6), object(4)
memory usage: 24.7+ MB
```

Code:

```
print(genre_data.info())
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2973 entries, 0 to 2972
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   mode              2973 non-null   int64
 1   genres            2973 non-null   object
 2   acousticness      2973 non-null   float64
 3   danceability      2973 non-null   float64
```



```
 4   duration_ms       2973 non-null   float64
 5   energy            2973 non-null   float64
 6   instrumentalness  2973 non-null   float64
 7   liveness          2973 non-null   float64
 8   loudness          2973 non-null   float64
 9   speechiness       2973 non-null   float64
 10  tempo             2973 non-null   float64
 11  valence           2973 non-null   float64
 12  popularity        2973 non-null   float64
 13  key               2973 non-null   int64
dtypes: float64(11), int64(2), object(1)
memory usage: 325.3+ KB
None
```

Code:

```
print(year_data.info())
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   mode              100 non-null    int64
 1   year              100 non-null    int64
 2   acousticness      100 non-null    float64
 3   danceability      100 non-null    float64
 4   duration_ms       100 non-null    float64
 5   energy            100 non-null    float64
 6   instrumentalness  100 non-null    float64
 7   liveness          100 non-null    float64
 8   loudness          100 non-null    float64
 9   speechiness       100 non-null    float64
 10  tempo             100 non-null    float64
 11  valence           100 non-null    float64
 12  popularity        100 non-null    float64
 13  key               100 non-null    int64
dtypes: float64(11), int64(3)
memory usage: 11.1 KB
None
```

EDA – analysis report:

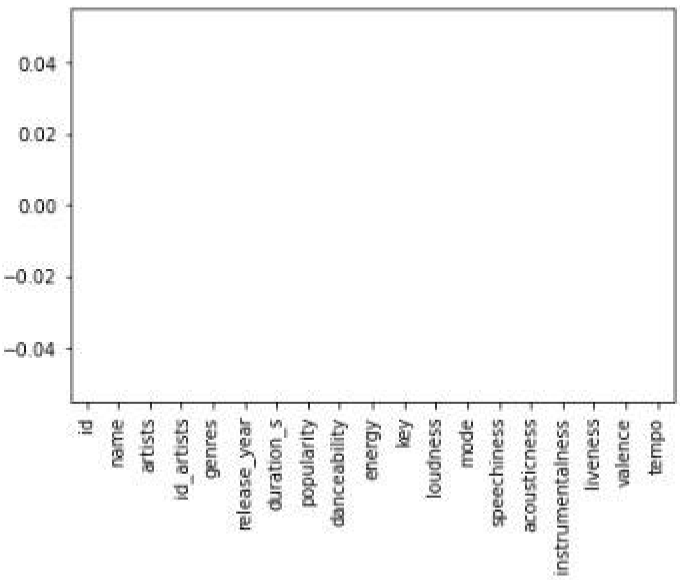
1. Missing

Code:

```
tracks.dropna(inplace = True)

tracks.isnull().sum().plot.bar()

plt.show()
```

Output:

:



2. Duplicate

Code:

```
tracks['name'].nunique(), tracks.shape
```

Output:

Output:

```
(408902, (536847, 17))
```



```
tracks =tracks.sort_values(by=['popularity'], ascending=False)

tracks.drop_duplicates(subset=['name'], keep='first', inplace=True)
```

The above code removed the duplicate rows.

3. Normalization

Code:

plt.subplots(figsize = (15, 5))

for i, col in enumerate(floats):

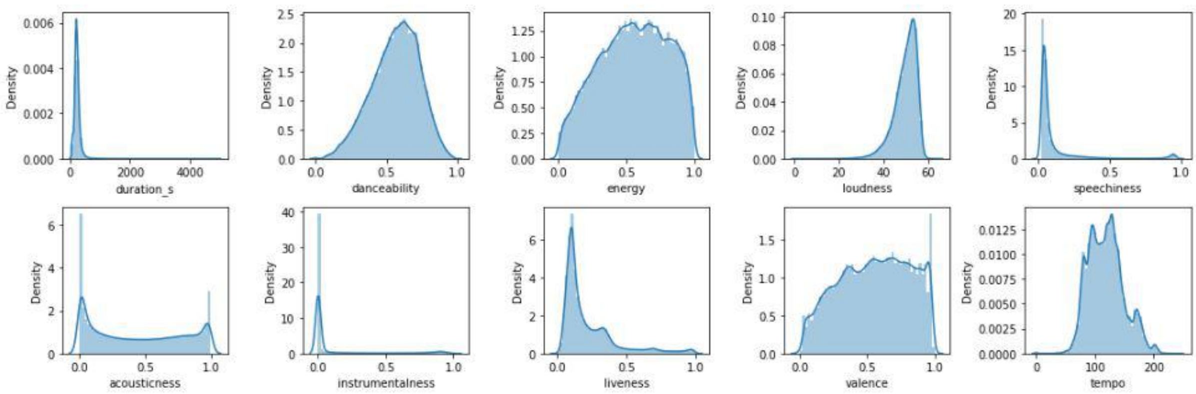
plt.subplot(2, 5, i + 1)

sb.distplot(tracks[col])

plt.tight_layout()

plt.show()

Output:



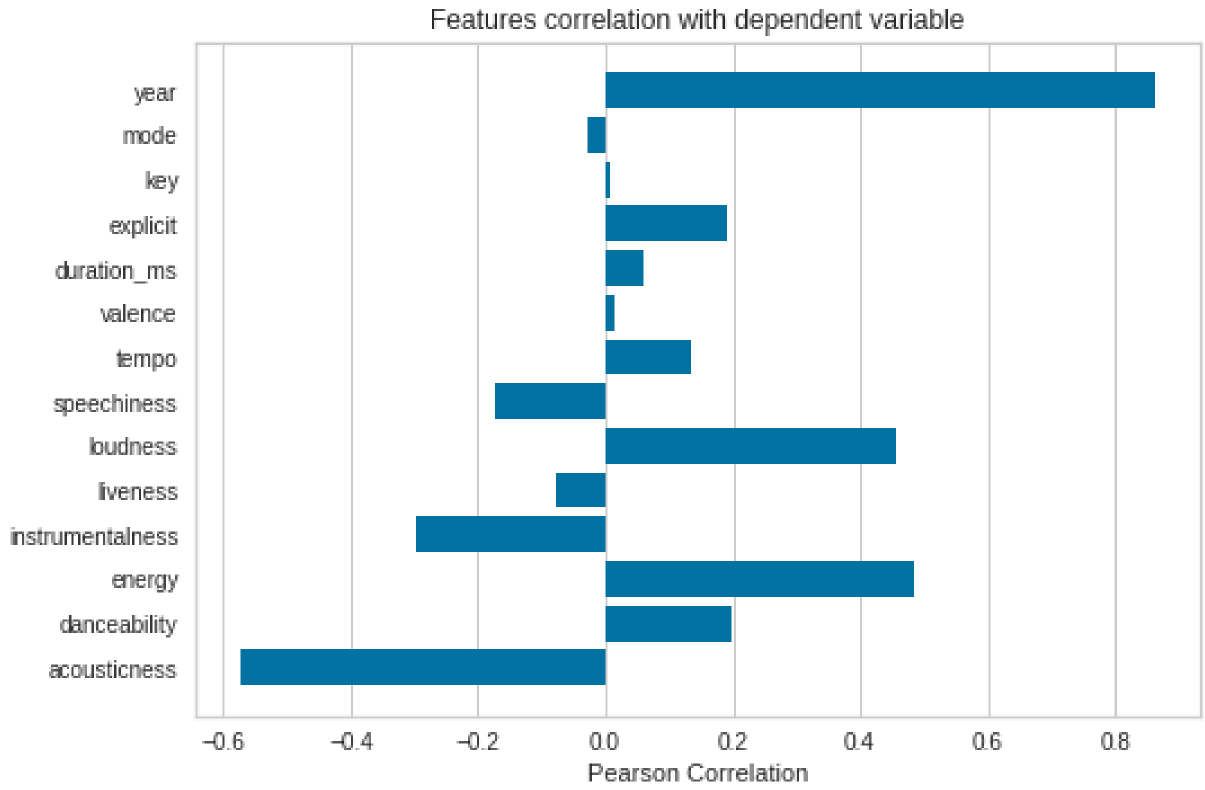
Some of them have Normal distribution.

4. Correlation

Code:

```
from yellowbrick.target importFeatureCorrelation

feature_names = ['acousticness', 'danceability', 'energy', 'instrumentalness',
'liveness', 'loudness', 'speechiness', 'tempo',
'valence','duration_ms','explicit','key','mode','year']

X, y = data[feature_names], data['popularity']

# Create a list of the feature names
features = np.array(feature_names)

# Instantiate the visualizer
visualizer = FeatureCorrelation(labels=features)

plt.rcParams['figure.figsize']=(20,20)
visualizer.fit(X, y)      # Fit the data to the visualizer
visualizer.show()
```

Output:



5. Outlier

Code:

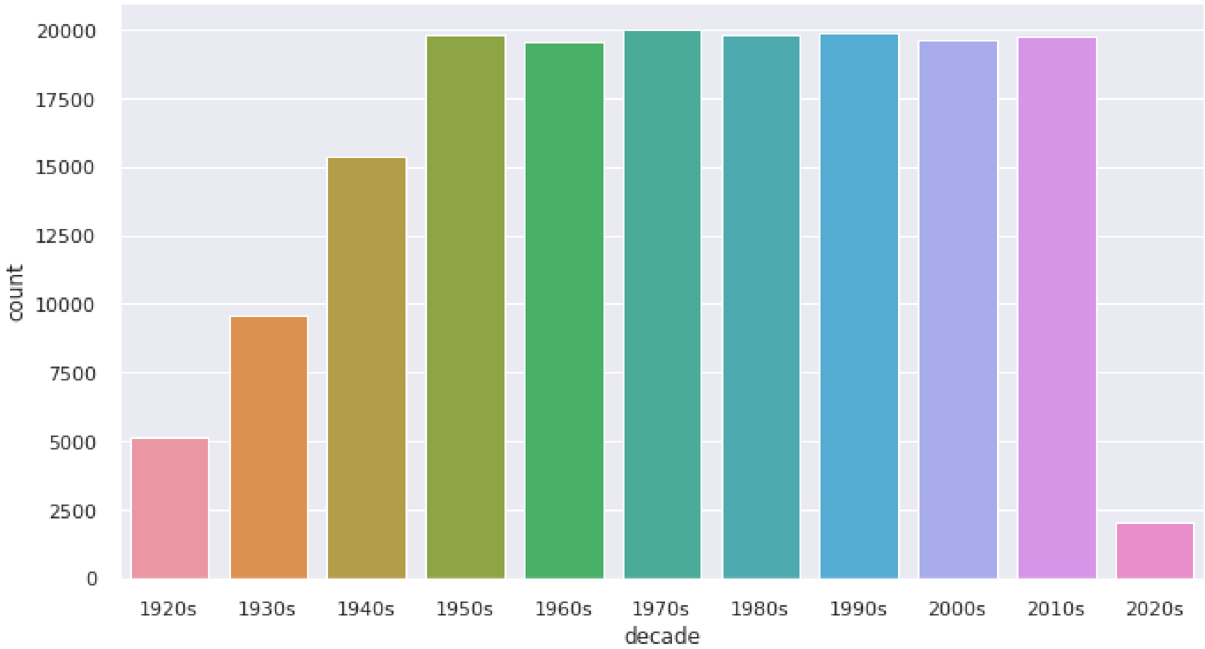
tracks = tracks.drop(['id', 'id_artists'], axis = 1)

6. Data Visualizations (5 v)

Code:

```
defget_decade(year):
period_start =int(year/10) *10
decade ='{}s'.format(period_start)
returndecade

data['decade'] = data['year'].apply(get_decade)

sns.set(rc={'figure.figsize':(11 ,6)})
sns.countplot(data['decade'])
```

Output:



Model Output:

Clustering Genres with K-Means:

Here, the simple K-means clustering algorithm is used to divide the genres in this dataset into ten clusters based on the numerical audio features of each genres.

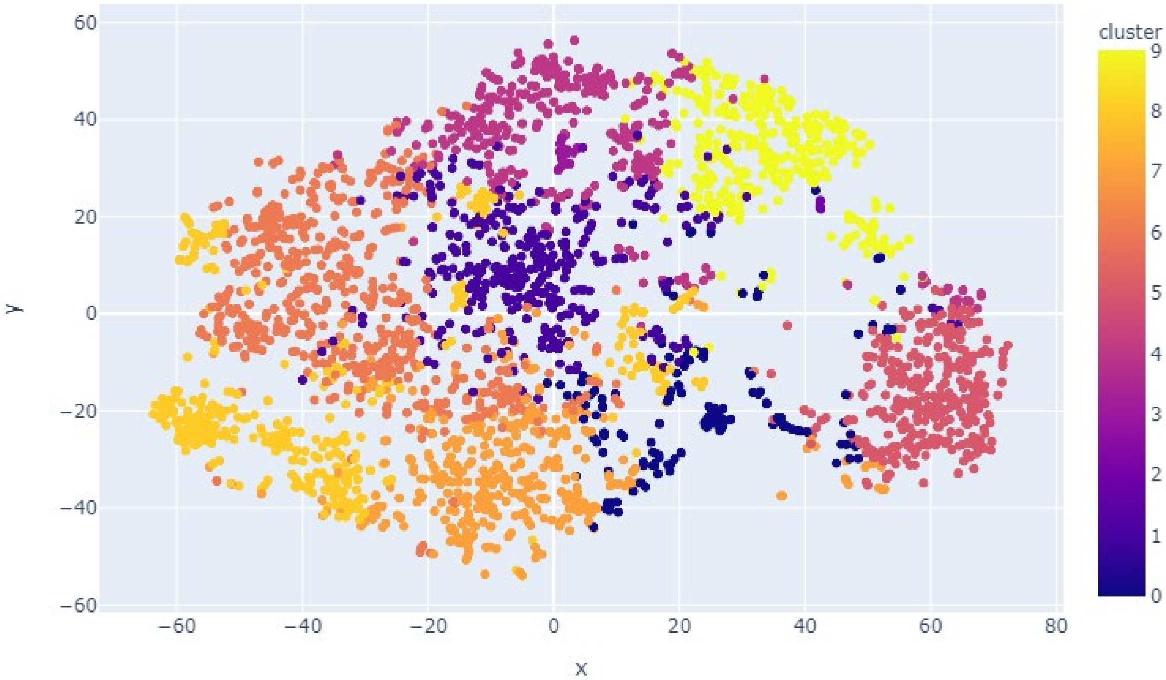
Code:

```
from sklearn.cluster importKMeans
from sklearn.preprocessing importStandardScaler
from sklearn.pipeline importPipeline

cluster_pipeline = Pipeline([('scaler', StandardScaler()), ('kmeans',
KMeans(n_clusters=10, n_jobs=-1))])
X = genre_data.select_dtypes(np.number)
cluster_pipeline.fit(X)
genre_data['cluster'] = cluster_pipeline.predict(X)
```

```
In [11]:
linkcode
# Visualizing the Clusters with t-SNE

from sklearn.manifold importTSNE

tsne_pipeline = Pipeline([('scaler', StandardScaler()), ('tsne',
TSNE(n_components=2, verbose=1))])
genre_embedding = tsne_pipeline.fit_transform(X)
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'])
fig.show()
```

Output:

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 2973 samples in 0.005s...
[t-SNE] Computed neighbors for 2973 samples in 0.322s...
[t-SNE] Computed conditional probabilities for sample 1000 / 2973
[t-SNE] Computed conditional probabilities for sample 2000 / 2973
[t-SNE] Computed conditional probabilities for sample 2973 / 2973
[t-SNE] Mean sigma: 0.777516
[t-SNE] KL divergence after 250 iterations with early exaggeration: 76.115768
[t-SNE] KL divergence after 1000 iterations: 1.392461
```

Clustering Songs with K-Means

Code:

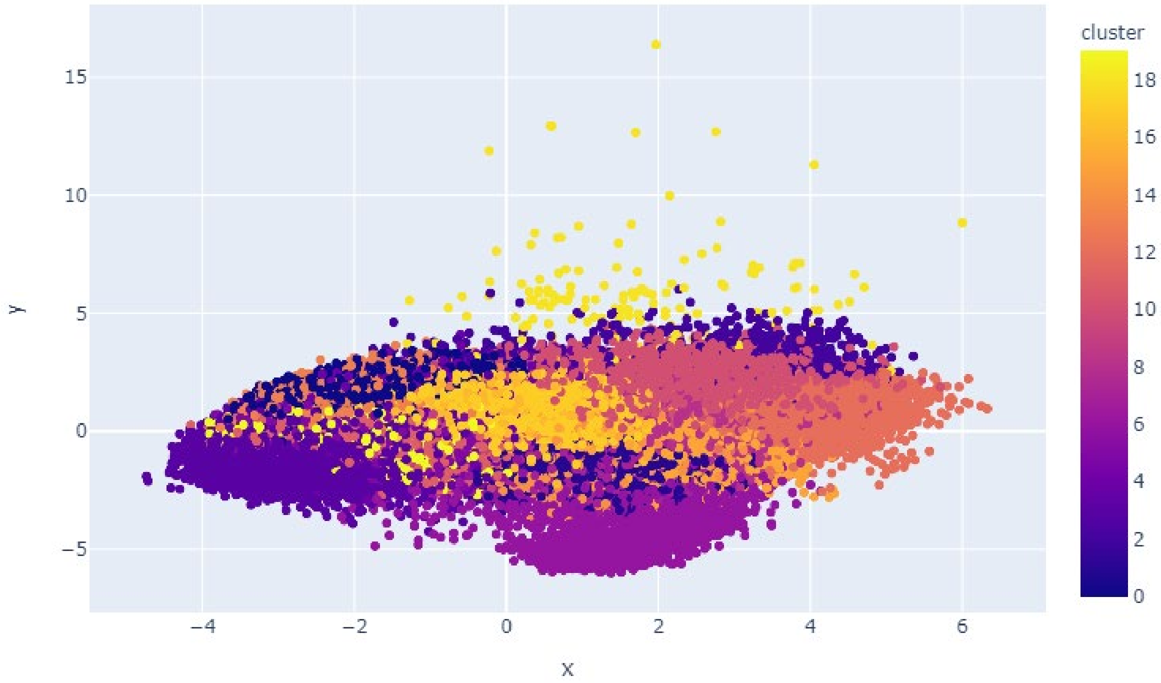
```
song_cluster_pipeline = Pipeline([('scaler', StandardScaler()),
                                  ('kmeans', KMeans(n_clusters=20,
                                   verbose=False, n_jobs=4))
                                 ], verbose=False)

X = data.select_dtypes(np.number)
number_cols =list(X.columns)
song_cluster_pipeline.fit(X)
song_cluster_labels = song_cluster_pipeline.predict(X)
data['cluster_label'] = song_cluster_labels
```

```
# Visualizing the Clusters with PCA

from sklearn.decomposition importPCA

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'])
fig.show()
```

OutPut:

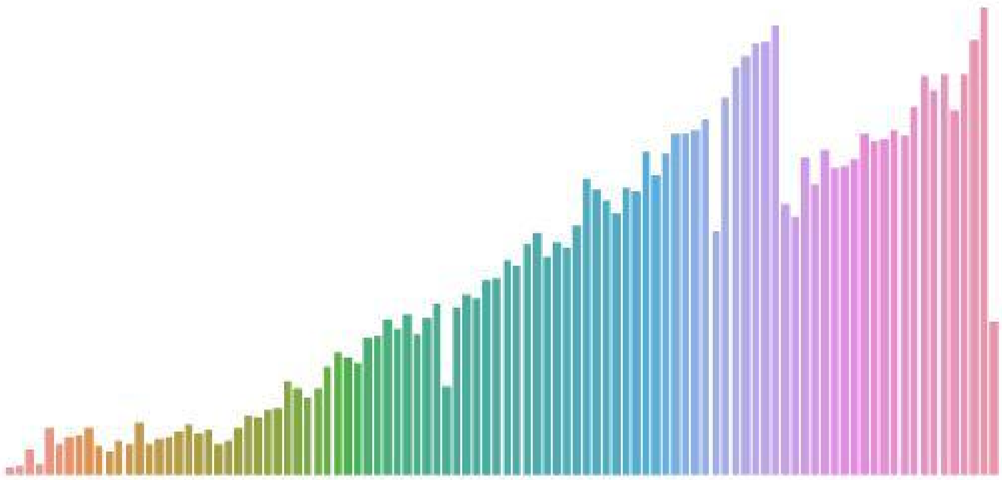


Let’s visualize the number of songs released each year.

Code:

```
plt.figure(figsize = (10, 5))
    sb.countplot(tracks['release_year'])
    plt.axis('off')
    plt.show()
```

OutPut:



Code:

recommend_songs('Shape of You')

Output:

	name	artists
90082	Supermarket Flowers	Ed Sheeran
91939	Bruises	Lewis Capaldi
91899	Before You Go	Lewis Capaldi
90741	What Do I Know?	Ed Sheeran
119306	Hearts Don't Break Around Here	Ed Sheeran

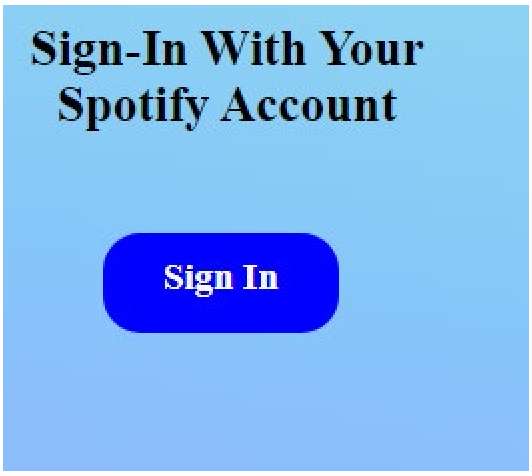
Code:

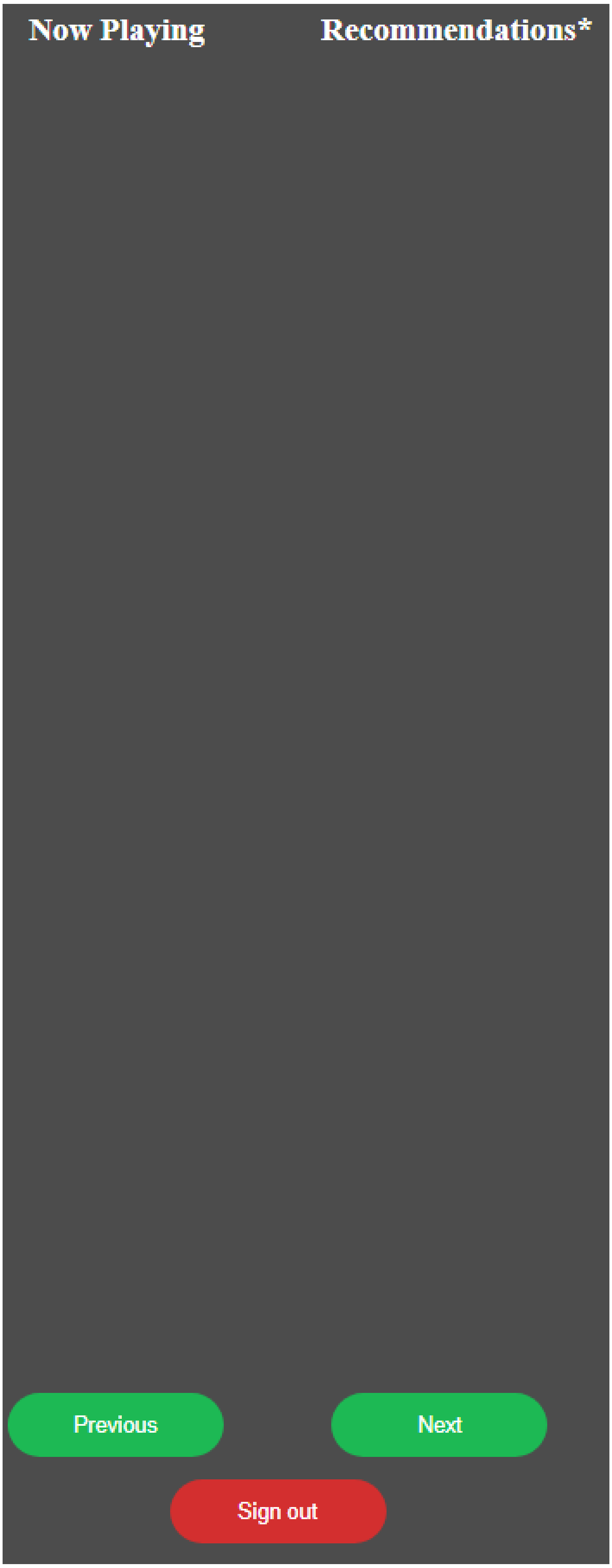
recommend_songs('Love me like you do')

OutPut:

```
This song is either not so popular or you have entered invalid_name.
 Some songs you may like:

Eu Tenho Medo
Alaz Alaz
Crimen
Red Lights
Lonely Together (feat. Rita Ora)
```


App interface / project result




Now Playing
Recommendations*
Previous
Next
Sign out

```
This song is either not so popular or you have entered invalid_name.
 Some songs you may like:

Eu Tenho Medo
Alaz Alaz
Crimen
Red Lights
Lonely Together (feat. Rita Ora)
```

CONCLUSION

The project "Spotify Music Recommendations System" has successfully demonstrated the potential of data analytics in the recommendations system. The real-time analysis of customer data has provided valuable insights into customer behavior, preferences, and trends, thereby facilitating informed decision-making. The interactive dashboards and reports have offered a comprehensive view of customer data, enabling the identification of patterns and correlations. This has not only improved the efficiency of data analysis but also enhanced the recommendation's ability to provide personalized services to its customers. The project has also highlighted the importance of data visualization in making complex data more understandable and accessible. The use of Jupiter Notebook has made it possible to present data in a visually appealing and easy-to-understand format, thereby aiding in better decision-making.

FUTURE SCOPE

The future scope of this project is vast. With the advent of advanced analytics and machine learningcan be leveraged to predict future trends based on historical data. Integrating these predictive analytics into the project could enable the recommendations system to anticipate customer needs and proactively offer solutions. Furthermore, It has the capability to integrate with various data sources opens up the possibility of incorporating more diverse datasets for a more holistic view of customers. As data privacy and security become increasingly important, future iterations of this project should focus on implementing robust data governance strategies. This would ensure the secure handling of sensitive customer data while complying with data protection regulations. Additionally, the project could explore the integration of real-time data streams to provide even more timely and relevant insights. This could potentially transform the way spotify music recommendations system interact with their customers, leading to improved customer satisfaction and loyalty.

REFERENCES

1. https://github.com/arunajeyarani/Aruna-/tree/main/Code, Aruna K , 2024
2.https://github.com/arunajeyarani/Aruna-/tree/main/Video,Aruna K ,2024
3.https:://github.com/arunajeyarani/Aruna-/tree/main/ProjectReport.Aruna K , 2024

https://medium.com/analytics-vidhya/analysis-of-bank-customers-using-dashboard-in-power-bi-a366f2b3e563

GIT Hub Link of Project Code:

https://github.com/githubtraining/hellogitworld.git

