19-202-0604 Data Mining

Assignment Report

Aparna Nair

Predicting Song Popularity Based On Spotify Classifying Features Using Logistic Regression

B.Tech Computer Science

CS-A Register Number: 20220021

July 30, 2023

Introduction

Predicting song popularity using spotify classifying features. These features are: Title, Artist, Top Genre, Year, Beats Per Minute (BPM), Energy, Danceability, Loudness (dB), Liveness, Valence, Length (Duration), Acousticness, Speechiness, Popularity.

Dataset

First, find the dataset required on Kaggle: Spotify - All Time Top 2000s Mega Dataset. This is a dataset containing audio features of Top 2000 songs on Spotify of all time. The data contains about 15 columns each describing the track and it's qualities. Songs released from 1956 to 2019 are included from some notable and famous artists like Queen, The Beatles, Guns N' Roses, etc.

Content

- Index: ID
- Title: Name of the Track
- Artist: Name of the Artist
- Top Genre: Genre of the track
- Year: Release Year of the track
- Beats per Minute(BPM): The tempo of the song
- Energy: The energy of a song - the higher the value, the more energtic. song
- Danceability: The higher the value, the easier it is to dance to this song.
- Loudness: The higher the value, the louder the song.
- Valence: The higher the value, the more positive mood for the song.
- Length: The duration of the song.
- Acoustic: The higher the value the more acoustic the song is.
- Speechiness: The higher the value the more spoken words the song contains.

These are the Spotify classifying features.

Implementation in Google Colab

```
#Mount the google drive
from google.colab import drive
drive.mount('/content/drive')

#Import nrequired libraries
# Visualizations
import pandas as pd
import numpy as np
import datetime
import plotly.express as px
import seaborn as sns
from matplotlib import pyplot as plt
%matplotlib inline
sns.set_style("whitegrid")
import matplotlib.pyplot as plt


# For transformations and predictions
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Models to be used, all from sklearn
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier

# Transformers
from scipy.optimize import curve_fit
from sklearn.preprocessing import FunctionTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import pairwise_distances
from six import StringIO

# For comparing metrics
from sklearn.metrics import mean_squared_error as mse

# For validation
from sklearn.model_selection import train_test_split as split



# Step 1: Data Preprocessing
# Load the dataset
df = pd.read_csv("/content/drive/MyDrive/Spotify-2000.csv")

# Returns the first 1000 rows
```

```
48 df.head(1000)
49
50 df.isnull().sum()
51 # no nulls, allgood
52
53 # print(len(df.index))
54
55 df = df.iloc[0:2000]
56
57 # There were some non-numeric values contained in the duration column,
58 # that should have been measured in milliseconds
59 df[df["Length (Duration)"].str.contains(",")==True]
60
61 df["Length (Duration)"] = df["Length (Duration)"].replace(["1,412","
      1,121","1,367","1,292"],['1412','1121','1367','1292'])
62 #These were corrected manually
63
64 df["Length (Duration)"] = df["Length (Duration)"].astype(np.int64)
65 #Casting all values to integers
66
67 columns = ['Index', 'Title']
68 # for col in columns:
69   # print(f'{col:<15}: {df[col].nunique()} unique values')
70
71 df = df.drop(labels=['Index', 'Title'], axis=1)
72 df.shape
73
74 #No duplicates
75 df.duplicated().sum()
76
77 #No null values
78 df.isnull().sum().sum()
79
80 #As danceability increases, popularity increases according to spotify
81 fig, ax = plt.subplots(1, figsize=(15, 6), sharey=True, sharex = True)
82 ax_data =  df.groupby('Danceability')['Popularity'].mean().to_frame().
      reset_index()
83 ax = sns.scatterplot(x='Danceability', y='Popularity', data=ax_data,
      color='blue', ax=ax)
84 ax.set_title('Danceability')
85 ax.set_ylabel('Mean Popularity', fontsize=12)
86 plt.tight_layout()
87 plt.show()

88 #Results of initial numeric correlation, shows the features are not good
       indicators of target varible, popularity
89 corr = np.abs(numeric_df.corr())
90 series = np.abs(corr['Popularity']).sort_values(ascending=False)
91 print('The most linear correlated features to POPULARITY are:')
```

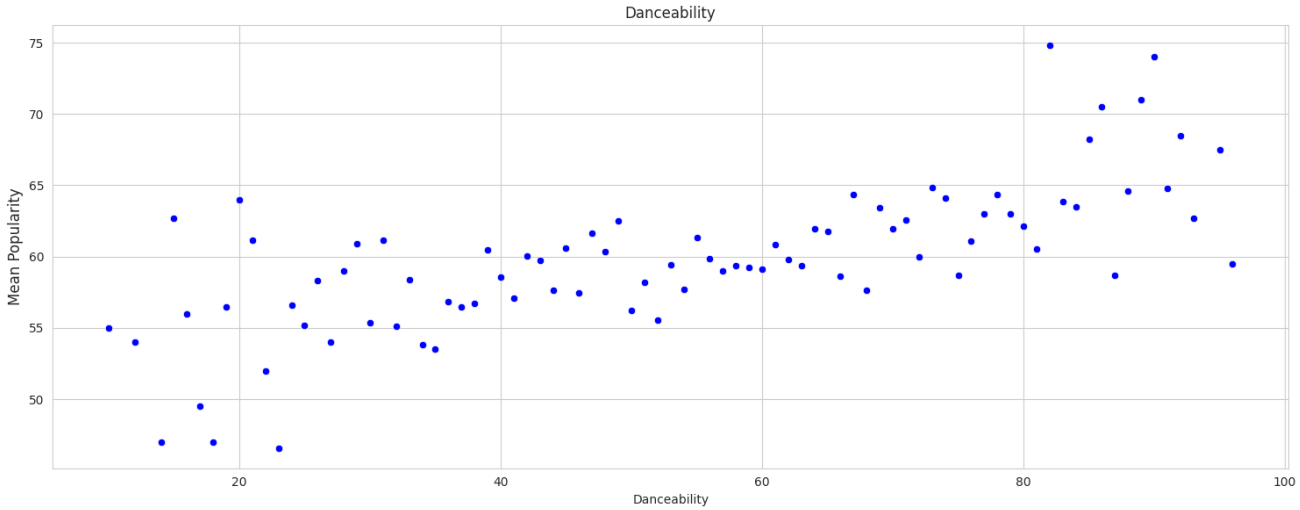

Figure 1: Scatter graph of Danceability vs Mean Popularity

```
for i, row in enumerate(series):
    if 0.1 <= row < 1:
      print(f'{series.index[i]:17} --> {row: .2f} (abs)')
```

The most linear correlated features to POPULARITY are:
Loudness (dB) –> 0.17 (abs)
Year –> 0.16 (abs)
Danceability –> 0.14 (abs)
Liveness –> 0.11 (abs)
Speechiness –> 0.11 (abs)
Energy –> 0.10 (abs)

```
#feature engineering, classify popular songs as 1, more unpopular songs
   as 0
df['Popularity'].describe()
df['Popularity'] = pd.qcut(df['Popularity'], q=2, labels=[0, 1])
df["Popularity"].value_counts()
df['Popularity']

#One-hot encoding to get dummies function for the non-numeric values.
def onehot_encode(df, column, prefix):
    df = df.copy()
    dummies = pd.get_dummies(df[column], prefix=prefix)
    df = pd.concat([df, dummies], axis=1)
    df = df.drop(column, axis=1)
    return df

#one hot encoding the genre and artists
df = onehot_encode(df, 'Artist', 'artist')
df = onehot_encode(df, 'Top Genre', 'genre')
```

```
df

#splitting the data
y = df.loc[:, 'Popularity']
X = df.drop('Popularity', axis=1)
# X = X.drop('Top Genre', axis=1)

scaler = StandardScaler()
X = scaler.fit_transform(X)

#random = 369
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size
   =0.7, random_state=369)

print("X_train: ",X_train.shape)
print("X_test: ",X_test.shape)
print("y_train: ",y_train.shape)
print("y_test: ",y_test.shape)
```

$X_train : (1395, 890)$
$X_test : (599, 890)$
$y_train : (1395,)$
$y_test : (599,)$

```
#Accuracy
log_model = LogisticRegression()
log_model.fit(X_train, y_train)
log_acc = log_model.score(X_test, y_test)
print("Logistic Regression Accuracy:", log_acc)
```

Logistic Regression Accuracy: 0.7161936560934892

Results and interpretation

From the accuracy count which is 0.71 using Logistic Regression, we can safely say that this model can be used to predict the popularity of a song against spotify classifying features.

My interpretation from this case study is that we can not in fact, successfully predict the popularity of a song from spotify features alone. While some features were largely useful, such as Danceability, a combination of BPM over duration, most were not, Valence, Energy, Loudness, etc.

Spotify Feature	Target Variable, (Popularity)
Loudness (dB)	0.17 (abs)
Year	0.16 (abs)
Danceability	0.14 (abs)
Liveness	0.11 (abs)
Speechiness	0.11 (abs)
Energy	0.10 (abs)

Table 1: Shows all the features that are not required for explaining song popularity