# MACHINE LEARNING BASED MUSIC GENRE CLASSIFICATION ON SPOTIFY DATA

Project Report

JUNE 30, 2023

TEAM 477

SMART INTERNZ

# INTRODUCTION

## OVERVIEW

In recent years, the exponential growth of digital music platforms and the vast amount of music available online have posed a significant challenge to music enthusiasts and researchers alike: how to efficiently categorize and classify music into different genres. Music genre classification plays a crucial role in various applications, including personalized music recommendation systems, music discovery platforms, and automated playlist generation. With the advent of machine learning techniques there has been a surge of interest in using these methods to tackle the music genre classification problem. This project aims to explore and analyse machine learning-based approaches for music genre classification using Spotify data, one of the largest music streaming platforms worldwide.

## PURPOSE

The purpose of this project is to develop a system that can accurately predict the genre of a given song based on its audio features. This can be useful in various applications, such as music recommendation systems, music analysis, and music industry insights. By accurately classifying songs into their respective genres, users can discover new music and industry professionals can gain valuable insights into trends and patterns in the music industry.

# LITERATURE SURVEY

## EXISTING PROBLEM

In the field of music genre classification, several approaches have been explored. Existing methods typically involve extracting audio features from songs and using machine learning algorithms to classify them into genres. However, there are challenges in accurately predicting genres due to the subjective nature of music and the presence of overlapping characteristics between genres.

"Music Genre Classification Using Machine Learning Techniques: A Systematic Literature Review" by Santos et al. (2019):
This systematic literature review presents an overview of machine learning techniques employed for music genre classification. It covers a wide range of approaches, including decision trees, support vector machines (SVMs), k-nearest neighbors (k-NN), and ensemble methods. The authors discuss the strengths and weaknesses of each technique, along with their performance on various datasets. Additionally, they highlight potential research directions and open challenges in the field.

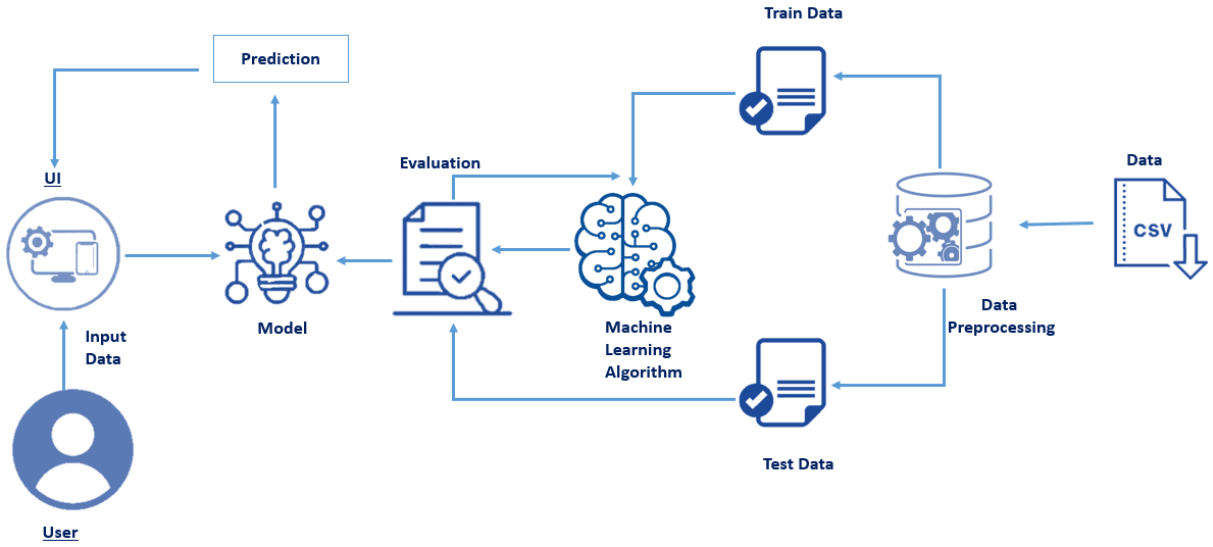
## PROPOSED SYSTEM

The proposed solution in this project involves training machine learning models on a Spotify dataset containing audio features of songs. The project aims to leverage the power of machine learning algorithms, such as decision trees, random forests, or neural

networks, to learn the patterns and relationships between the audio features and the corresponding genre labels. By employing a large and diverse dataset, the project seeks to achieve accurate music genre prediction.

## THEORETICAL ANALYSIS

### BLOCK DIAGRAM




### HARDWARE/SOFTWARE DESIGNING

The project primarily relies on software components such as Anaconda Navigator, Jupyter Notebook, Python libraries (NumPy, Pandas, Matplotlib, Scikit-learn, Flask), and a web browser. The hardware requirements include a computer system capable of running the software components efficiently.

## EXPERIMENTAL INVESTIGATIONS

### DATA PREPROCESSING

The first step begins with preprocessing of the Spotify data. This may involve extracting audio features using the Spotify Web API, such as acousticness, danceability, energy, loudness, tempo, and so on. Further, this includes studying the statistical properties of these features, such as their distribution, variance, and correlation with genre labels.

### FEATURE REPRESENTATION

This step focuses on the selection and representation of features that best capture the characteristics of different music genres. This may involve applying techniques like dimensionality reduction (e.g., Principal Component Analysis or t-SNE) to reduce the feature space while preserving the discriminatory information. This step investigates the information loss and trade-offs associated with different feature representations.

## ALGORITHM SELECTION

Algorithm selection involves studying different machine learning algorithms suitable for music genre classification. This may include traditional algorithms like support vector machines (SVM), random forests, or more advanced techniques like deep learning models (e.g., convolutional neural networks or recurrent neural networks). In this step, theoretical properties of these algorithms, such as their complexity, convergence properties, and generalization capabilities are compared.

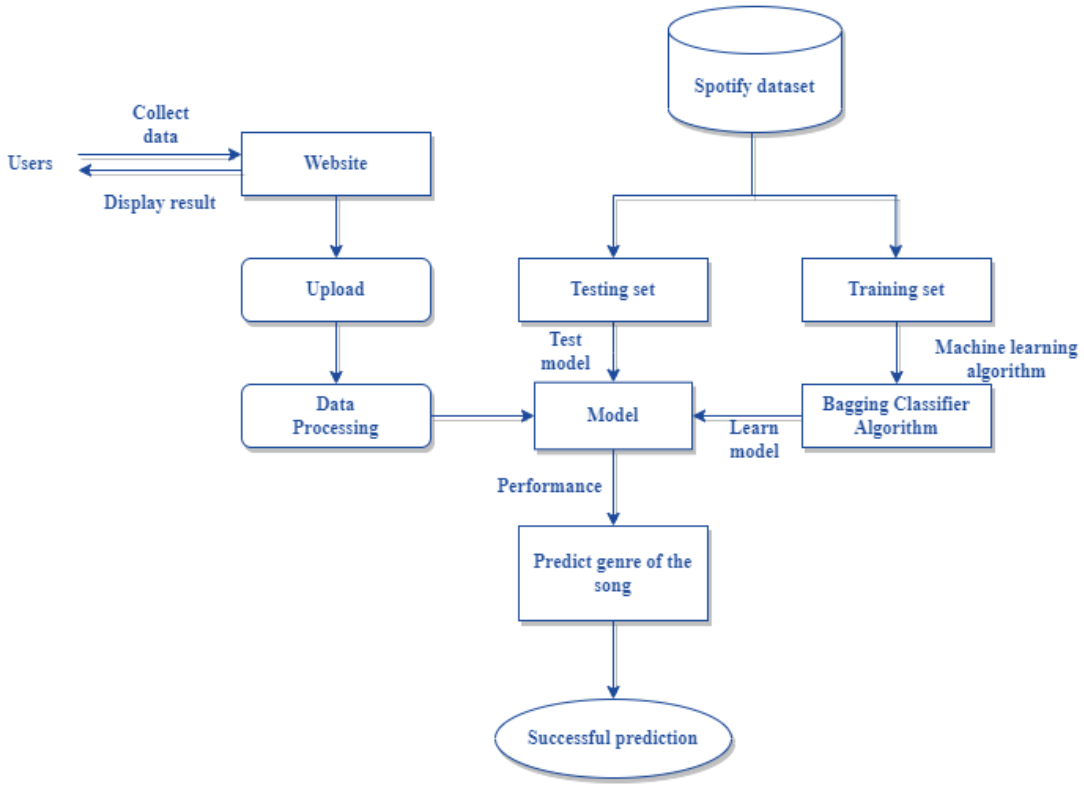
## MODEL TRAINING AND EVALUATION

Model training and evaluation focuses on understanding the training process and performance evaluation of the machine learning models. This includes studying aspects like loss functions, optimization algorithms, regularization techniques, and cross-validation strategies. The model's convergence, overfitting, and bias-variance trade-off are assessed.

## PERFORMANCE METRICS AND COMPARISON

In this step, the performance of the models using appropriate metrics such as accuracy, precision, recall, F1 score, or area under the receiver operating characteristic curve (AUC-ROC) are evaluated. This may include comparing the performance of different algorithms and variations of the models to identify the best approach for music genre classification.

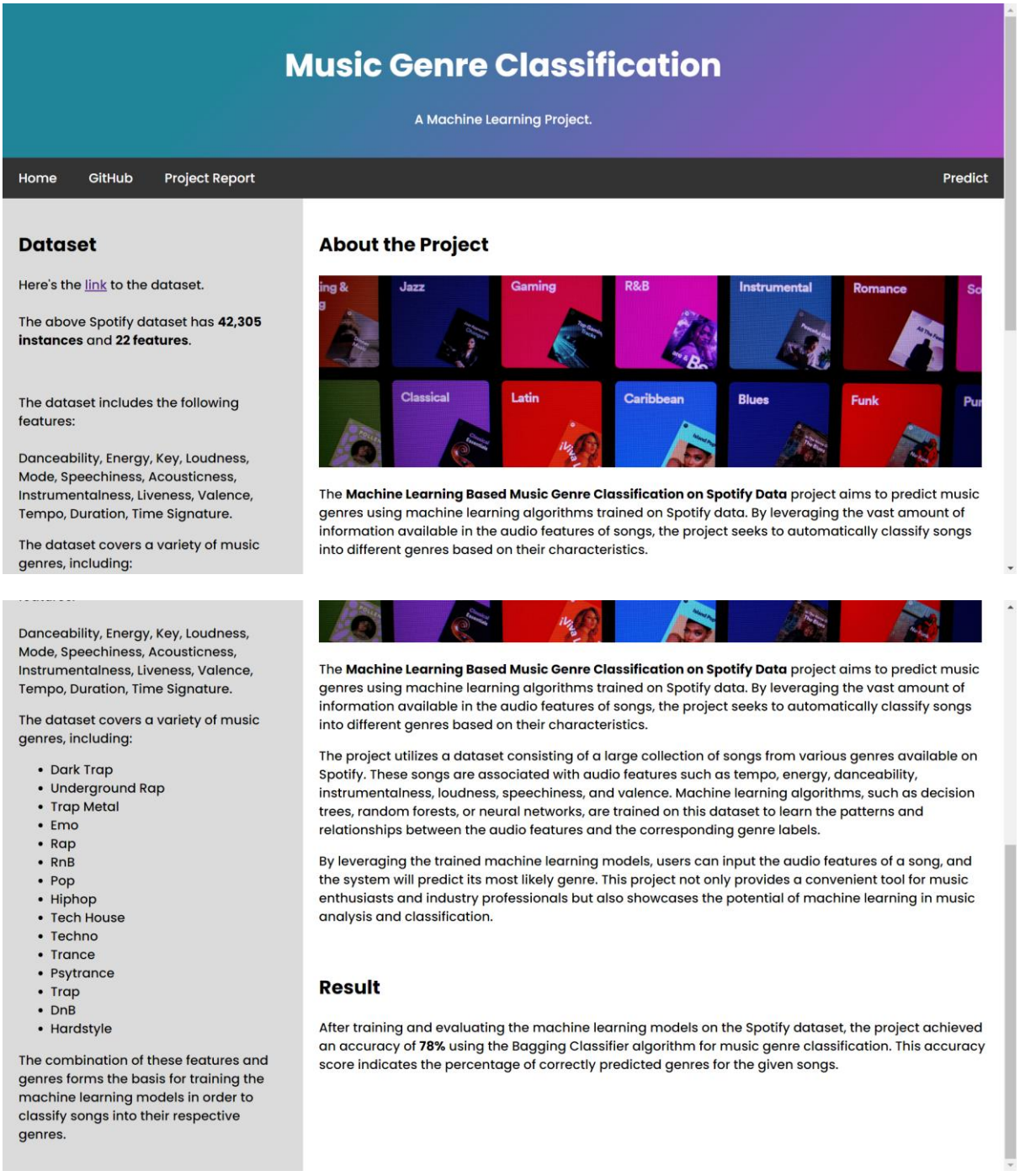
## FLOWCHART

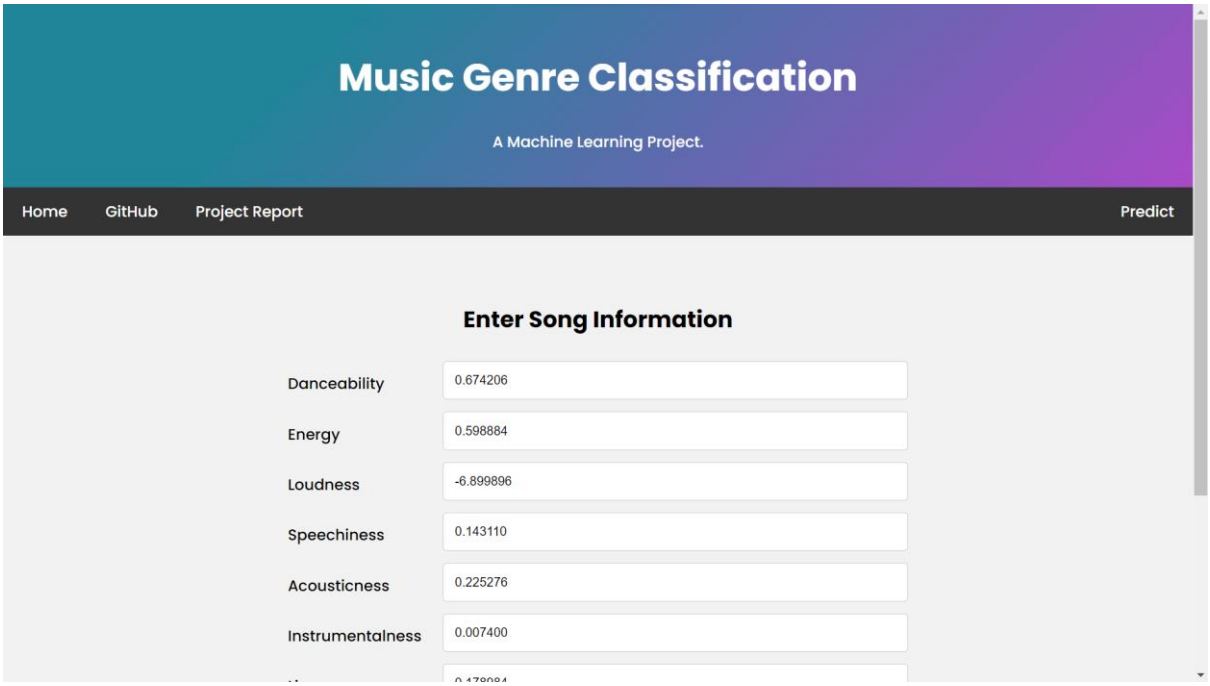


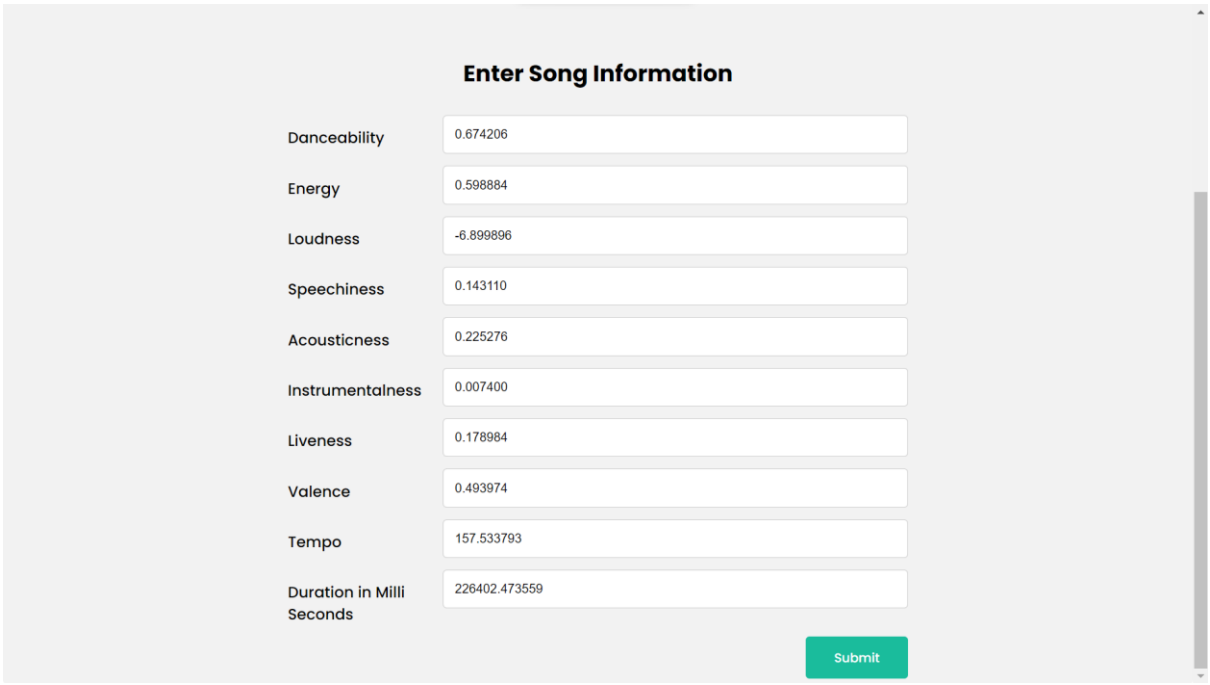
## RESULT

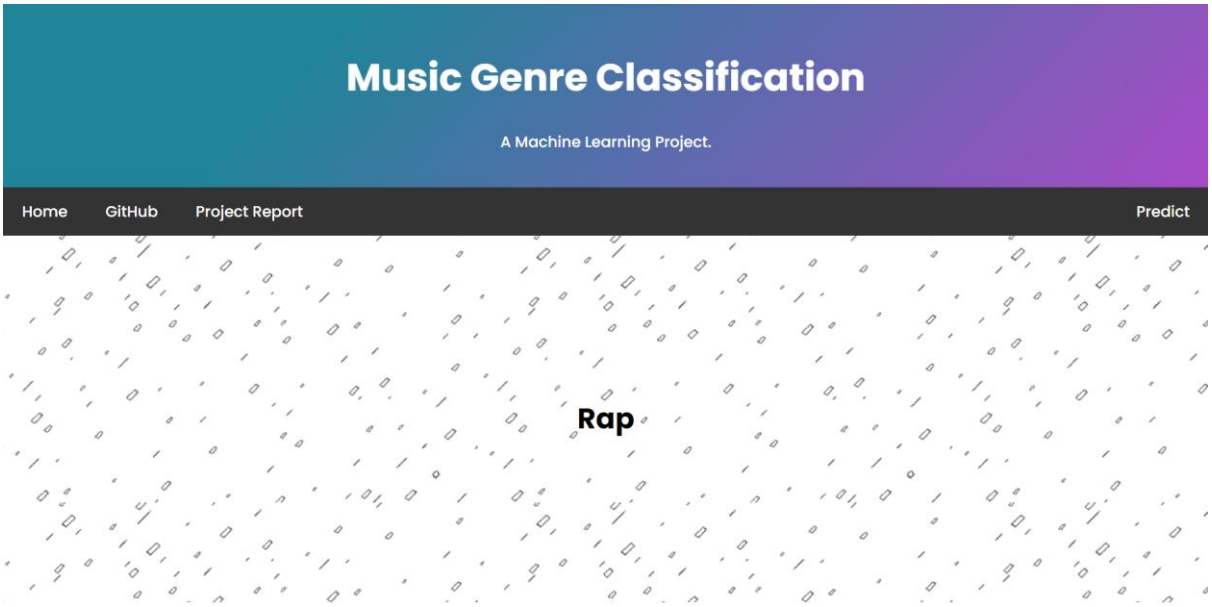
After training and evaluating the machine learning models on the Spotify dataset, the project achieved an accuracy of 78% using the Bagging Classifier algorithm for music genre classification. This accuracy score indicates the percentage of correctly predicted genres for the given songs.

Below are the screenshots of the implementation:




Music Genre Classification
A Machine Learning Project.
Home
GitHub
Project Report
Predict
Enter Song Information
Danceability
0.674206
Energy
0.598884
Loudness
-6.899896
Speechiness
0.143110
Acousticness
0.225276
Instrumentalness
0.007400


Enter Song Information
Danceability
0.674206
Energy
0.598884
Loudness
-6.899896
Speechiness
0.143110
Acousticness
0.225276
Instrumentalness
0.007400
Liveness
0.178984
Valence
0.493974
Tempo
157.533793
Duration in Milli Seconds
226402.473559
Submit




## ADVANTAGES & DISADVANTAGES

The accurate prediction of music genres based on audio features offers several advantages in various domains. One notable advantage is the potential it holds for music recommendation systems, allowing for personalized and tailored music suggestions to users based on their preferences. Additionally, this accurate prediction can provide valuable insights to the music industry, aiding in market research, trend analysis, and identification of emerging genres. Leveraging machine learning algorithms to analyze and classify music enables efficient processing of large music libraries, making it easier to organize and categorize vast amounts of audio data. However, there are also a few disadvantages to consider. One drawback is the presence of data imbalance among different genres, where certain genres may have significantly more representation in the dataset than others, potentially leading to biased predictions. Moreover, the subjective nature of music genres poses a challenge, as individual perceptions and cultural influences can lead to differing interpretations and classifications. Lastly, it is important to acknowledge that machine learning algorithms have limitations in capturing all the nuanced aspects of music, such as emotional depth, cultural context, and artistic expression, which may require human expertise for a comprehensive understanding.

## APPLICATIONS

The proposed solution has various applications in the music industry and related fields. Some potential applications include:

1. Music recommendation systems for personalized music discovery.
2. Music analysis and insights for industry professionals.
3. Genre-based playlists and radio stations for streaming platforms.
4. Music classification as a product, e.g., Shazam-like applications.

## CONCLUSION

The Machine Learning Based Music Genre Classification on Spotify Data project successfully developed a system capable of predicting music genres based on audio features. By leveraging machine learning algorithms and a large dataset, the project achieved a 78% accuracy in genre classification. This solution can have significant implications for music enthusiasts, industry professionals, and music platforms by providing accurate genre predictions and enabling personalized music experiences.

## FUTURE SCOPE

The project can be further improved and expanded in several ways:

1. Incorporating more diverse datasets to address data imbalance issues.
2. Exploring advanced machine learning algorithms and ensemble techniques.
3. Enhancing the web application with additional features and user interactions.
4. Integrating the system with streaming platforms for real-time genre predictions.
5. Investigating the influence of audio features on subgenres within each genre.

## BIBILOGRAPHY


1. https://towardsdatascience.com/music-genre-prediction-with-spotifys-audio-features-8a2c81f1a22e
2. https://www.kaggle.com/code/danielsheen/spotify-song-genre-classification
3. https://towardsdatascience.com/spotify-genre-classification-algorithm-88051db23d42
4. https://cs229.stanford.edu/proj2017/final-reports/5242682.pdf
5. https://www.kaggle.com/code/iqbalbasyar/spotify-genre-classification


## APPENDIX

### MUSIC-GENRE-PREDICTION.IPYNB

```
#!/usr/bin/env python
# coding: utf-8


# Import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split,
RepeatedStratifiedKFold, cross_val_score
from sklearn.preprocessing import StandardScaler, LabelEncoder
from imblearn.over_sampling import SMOTE
import pickle
```

```
from sklearn.ensemble import BaggingClassifier
from sklearn.metrics import accuracy_score, f1_score

# Load the dataset
data = pd.read_excel('genres_v2_1.xlsx')

# Drop irrelevant columns
data = data.drop(['song_name', 'Unnamed: 0', 'title', 'type', 'id',
'uri', 'track_href', 'analysis_url'], axis=1)

# Exploratory Data Analysis
sns.set_theme(style='darkgrid', palette='pastel')
numeric = data._get_numeric_data()
hist = numeric.hist(layout=(3,5), figsize=(20,10))
plt.show()

# Histogram - Danceability
plt.figure(figsize=(8, 6))
sns.histplot(data=data, x='danceability', kde=True, color='#E458CC')
plt.title('Distribution of Danceability')
plt.xlabel('Danceability')
plt.ylabel('Count')
plt.show()

# Histogram - Energy
plt.figure(figsize=(8, 6))
sns.histplot(data=data, x='energy', kde=True, color='#6402A9')
plt.title('Distribution of Energy')
plt.xlabel('Energy')
plt.ylabel('Count')
plt.show()

# Histogram - Key
plt.figure(figsize=(8, 6))
sns.histplot(data=data, x='key', kde=True, color='#EC9C75')
plt.title('Distribution of Key')
```

```
plt.xlabel('Key')
plt.ylabel('Count')
plt.show()

# Histogram - Loudness
plt.figure(figsize=(8, 6))
sns.histplot(data=data, x='loudness', kde=True, color='#AAE12A')
plt.title('Distribution of Loudness')
plt.xlabel('Loudness')
plt.ylabel('Count')
plt.show()

# Histogram - Speechiness
plt.figure(figsize=(8, 6))
sns.histplot(data=data, x='speechiness', kde=True, color='#08D0C1')
plt.title('Distribution of Speechiness')
plt.xlabel('Speechiness')
plt.ylabel('Count')
plt.show()

# Histogram - Acousticness
plt.figure(figsize=(8, 6))
sns.histplot(data=data, x='acousticness', kde=True, color='#005F6F')
plt.title('Distribution of Acousticness')
plt.xlabel('Acousticness')
plt.ylabel('Count')
plt.show()

# Histogram - Instrumentalness
plt.figure(figsize=(8, 6))
sns.histplot(data=data, x='instrumentalness', color='#F72F3E', kde=True)
plt.title('Distribution of Instrumentalness')
plt.xlabel('Instrumentalness')
plt.ylabel('Count')
plt.show()
```

```
# Histogram - Liveness
plt.figure(figsize=(8, 6))
sns.histplot(data=data, x='liveness', color='#DFE607', kde=True)
plt.title('Distribution of Liveness')
plt.xlabel('Liveness')
plt.ylabel('Count')
plt.show()

# Histogram - Valence
plt.figure(figsize=(8, 6))
sns.histplot(data=data, x='valence', color='#CDA5E2', kde=True)
plt.title('Distribution of Valence')
plt.xlabel('Valence')
plt.ylabel('Count')
plt.show()

# Histogram - Tempo
plt.figure(figsize=(8, 6))
sns.histplot(data=data, x='tempo', color='#715266', kde=True)
plt.title('Distribution of Tempo')
plt.xlabel('Tempo')
plt.ylabel('Count')
plt.show()

# Histogram - Duration in Milli Seconds
plt.figure(figsize=(8, 6))
sns.histplot(data=data, x='duration_ms', color='#C5807D', kde=True)
plt.title('Distribution of Duration in Milli Seconds')
plt.xlabel('Duration in Milli Seconds')
plt.ylabel('Count')
plt.show()

# Histogram - Time Signature
plt.figure(figsize=(8, 6))
sns.histplot(data=data, x='time_signature', color='#054F6C', kde=True)
plt.title('Distribution of Time Signature')
```

```
plt.xlabel('Time Signature')
plt.ylabel('Count')
plt.show()

# Features of Each Genre
grouped_genre = data.groupby('genre')
for col in numeric.columns:
    fig, ax = plt.subplots()
    for i, d in grouped_genre:
        d[col].hist(alpha=0.4, ax=ax, label=i, figsize=(7,5))
        ax.set_title(col)
    ax.legend()
    plt.show()

# Mean of different features in a Genre
grouped_genre.mean()

# Standard Deviation of different features in a Genre
grouped_genre.std()

# Minimum values of different features in a Genre
grouped_genre.min()

# Maximum values of different features in a Genre
grouped_genre.max()

# Songs count for all the Genres
genre = data['genre']
genre_count = {}
for gen in np.unique(genre):
    genre_count[gen] = len(data[data['genre'] == gen])
genre_count

fig = plt.figure(figsize=(20,10))
plt.bar(height=data['genre'].value_counts().values,
x=data['genre'].value_counts().index, color='purple')
```

```
plt.show()

# Check for missing values
data.isnull().any()

data['genre'].unique()

# Correlation Analysis
le = LabelEncoder().fit(data['genre'])
data['genre'] = le.transform(data['genre'])
corr = data.corr()
plt.figure(figsize=(15, 15))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

# Dropping highly correlated columns
features = data.drop(['key', 'mode', 'time_signature', 'genre'], axis=1)
labels = data['genre']

# Feature Scaling
sc = StandardScaler()
features = sc.fit_transform(features)
pickle.dump(sc, open('scalar.plk', 'wb'))

# Oversampling
oversample = SMOTE()
features, labels = oversample.fit_resample(features, labels)

# Splitting into train and test data
X_train, X_test, y_train, y_test = train_test_split(features, labels,
test_size=0.2, shuffle=True)

# Building Bagging Classifier
model = BaggingClassifier()
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
```

```
n_scores = cross_val_score(model, X_train, y_train, scoring='accuracy',
cv=cv, n_jobs=-1, error_score='raise')


n_scores.mean()


model.fit(X_train, y_train)
pred = model.predict(X_test)
le.inverse_transform(pred)

# Evaluating the model and saving the model
accuracy_score(y_test, pred)
f1_score(y_test, pred, average='weighted')

pickle.dump(model, open('model.plk', 'wb'))
```

**APP.PY**

```
# Importing the necessary dependencies
from flask import Flask, request, render_template
import numpy as np
import pickle

# Loading the model
app = Flask(__name__) # initializing a flask app
with open('D:\Applied Data Science\Flask\model.plk','rb') as f:
    model = pickle.load(f)
with open('D:\Applied Data Science\Flask\scalar.plk','rb') as f:
    sc = pickle.load(f)

# Loading the home page
@app.route('/') # route to display home page
def home():
    return render_template('home.html') # rendering the home page

# Loading the Music Genre Prediction page
@app.route('/prediction', methods=['POST', 'GET'])
def prediction():
```

```
    return render_template('index.html')

@app.route('/home', methods=['POST', 'GET'])
def my_home():
    return render_template('home.html')

@app.route('/predict', methods=['POST', 'GET'])
def predict():
    # reading the inputs by the user
    input_features = [float(x) for x in request.form.values()]
    x = [np.array(input_features)]
    x = sc.transform(x)

    # prediction using the loaded model
    prediction = model.predict(x)
    labels = ['Dark Trap', 'Underground Rap', 'Trap Metal', 'Emo', 'Rap',
'RnB',
       'Pop', 'Hiphop', 'Tech House', 'Techno', 'Trance', 'Psytrance',
       'Trap', 'DnB', 'Hardstyle']

    # showing the prediction result
    return render_template('result.html', prediction =
labels[prediction[0]])

# running the app
if __name__=='__main__':
    app.run(host='0.0.0.0', port=8000, debug=True)
```