PROJECT TITLE

MACHINE LEARNING BASED MUSIC GENRE CLASSIFICATION ON SPOTIFY DATA

BY

TEAM NO - 174

TULIKA GARG - 20BDS0348

PESALA NAGA SASHANK - 20BCE2153

ISHANVI SHARMA - 20BCE0394

ARHIT BOSE TAGORE - 20BCE2150

FROM

VELLORE INSTITUTE OF TECHNOLOGY

VELLORE, TAMIL NADU

1. INTRODUCTION

1.1. Overview

The music industry is one of the leading industries in the entertainment world. Every year, 1.6 million new songs are produced in different genres. Day by day, as the popularity of music increased, people's habits of listening to music also changed.

The development of music streaming services has increased the demand for automated music categorization and recommended systems.

Spotify is a Swedish audio streaming and media service provider founded on April 23, 2006, by Daneil Ek and Martin Lorentzon. It is one of the largest music streaming service providers, with over 527 million monthly active users, including 210 million paying subscribers, as of March 2023. Yet, for customers to have a personalized music experience, Spotify must recommend tracks that fit their preferences. Spotify uses machine learning algorithms to guide and categorize music based on genre.

1.2. Purpose

The main purpose of your project is to develop a model that classifies genres and can accurately predict the Genre of a music track on Spotify.

2. LITERATURE SURVEY

2.1. Existing Problem

2.1.1. "Music Genre Classification: A Comparative Study of Feature Selection Methods" by E. Tzanetakis and P. Cook:

This paper presents a comparative study of various feature selection methods for music genre classification. It explores the effectiveness of different feature subsets in improving classification accuracy and identifies key features that contribute to genre discrimination.

2.1.2. "Automatic Musical Genre Classification of Audio Signals" by G. Tzanetakis and P. Cook:

This influential paper discusses the development of a music genre classification system using audio features extracted from songs. It explores the use of various machine learning algorithms and evaluates their performance in genre classification tasks.

2.1.3. **"Deep Content-Based Music Recommendation" by A. van den Oord et al.:**

This paper introduces a deep learning-based approach for music recommendation, including genre classification. It utilizes convolutional neural networks (CNNs) to extract high-level representations of songs and applies them for genre classification and recommendation tasks.

2.1.4. "Music Genre Classification Using Machine Learning Techniques: A Comprehensive Review" by S. Lidy and A. Rauber:

This review paper presents a comprehensive overview of music genre classification techniques, focusing on machine learning algorithms. It discusses feature extraction methods, feature selection, and various classification algorithms employed in the literature.

2.1.5. "Genre Classification of Music Content: A Review" by D. Lee and S. Lee:

This review paper examines the evolution of music genre classification techniques, highlighting different approaches and challenges. It discusses feature extraction methods, classification algorithms, evaluation metrics, and provides insights into future directions.

EXPECTED PROBLEMS:

1. Subjective nature of music genre and the ambiguity in their definitions.
2. The quality and diversity of the datasets used for training and evaluation can significantly impact the performance of genre classification models.
3. Evaluating the performance of music genre classification models poses challenges due to the absence of standardized evaluation protocols and metrics.

2.2. PROPOSED SOLUTION

2.2.1. PROBLEM STATEMENT

The problem we aim to solve is music genre classification using machine learning. Given Spotify data, we want to build a model that can accurately predict the genre of a song based on its features.

2.2.2. DATA PREPARATION

2.2.2.1. The code starts by importing the necessary libraries, including pandas, numpy, matplotlib, seaborn, os, and sys.

2.2.2.2. The dataset is loaded into a DataFrame using pd.read_csv().

2.2.2.3. The unnecessary columns ('title', 'song_name', 'analysis_url', 'track_href', 'uri', 'id', 'Unnamed: 0', 'type') are dropped from the DataFrame.

2.2.2.4. Null values are checked using df.isnull().sum() and duplicates are removed using df.drop_duplicates().

2.2.3. ENCODING AND SPLITTING DATA

2.2.3.1. Label encoding is performed on the 'genre' column using LabelEncoder() from scikit-learn. The data is split into features (X) and the target variable (y).

2.2.4. MODEL TRAINING AND EVALUATION

2.2.4.1. The data is split into training and testing sets using train_test_split() from scikit-learn.

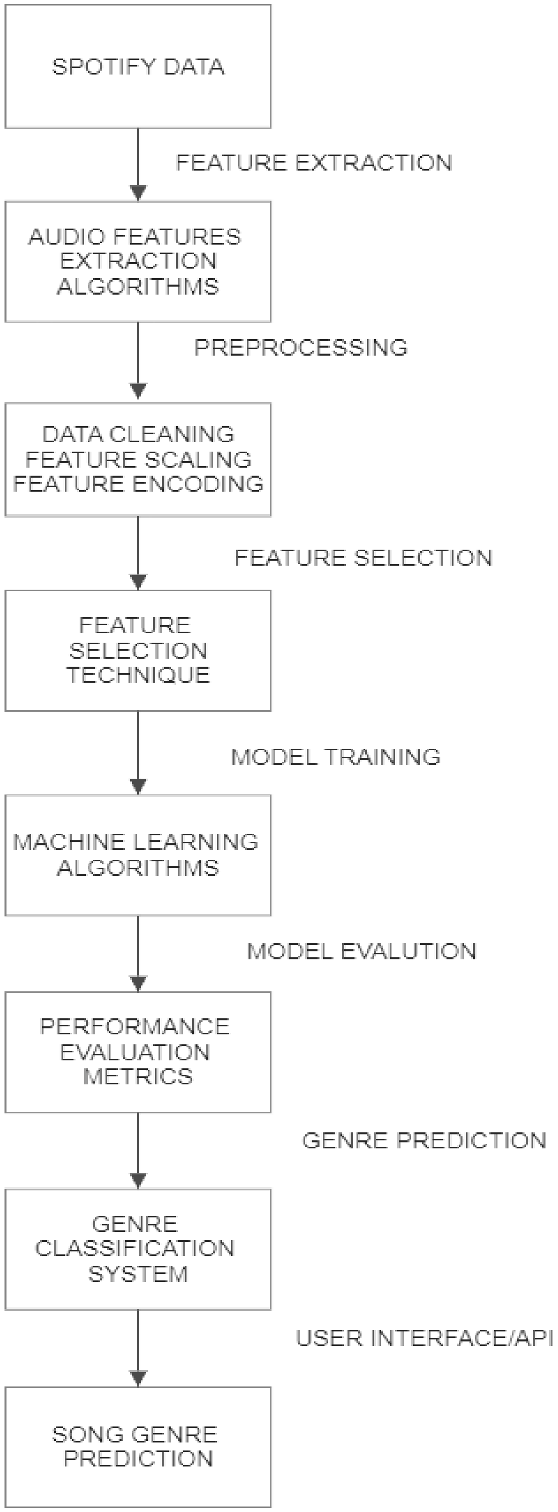
2.2.4.2. The features are standardized using StandardScaler() from scikit-learn.

2.2.4.3. Several machine learning models are trained and evaluated, including Logistic Regression, K Nearest Neighbors, Decision Tree, Random Forest, Gradient Boosting, CatBoost Classifier, and XGBoost Classifier.

2.2.4.4. The accuracy of each model is printed using the score() method.

3. THEORETICAL ANALYSIS

3.1. BLOCK DIAGRAM



3.2. HARDWARE/SOFTWARE DESIGNING

3.2.1. HARDWARE REQUIREMENTS

1. Computer with 4GB RAM
2. Storage

3.2.2. SOFTWARE REQUIREMENTS

I. IDE: Jupyter Notebook

II. Install Python(if required and update to latest version)

III. Libraries:

A. Pandas: A library for data manipulation and analysis.

B. NumPy: A scientific computer package used for matrix computations.

C. Matplotlib: A plotting library for python programming language.

D. Seaborn: A data visualization library based on matplotlib.

E. Sklearn: A Machine Learning library for building modes of classification.

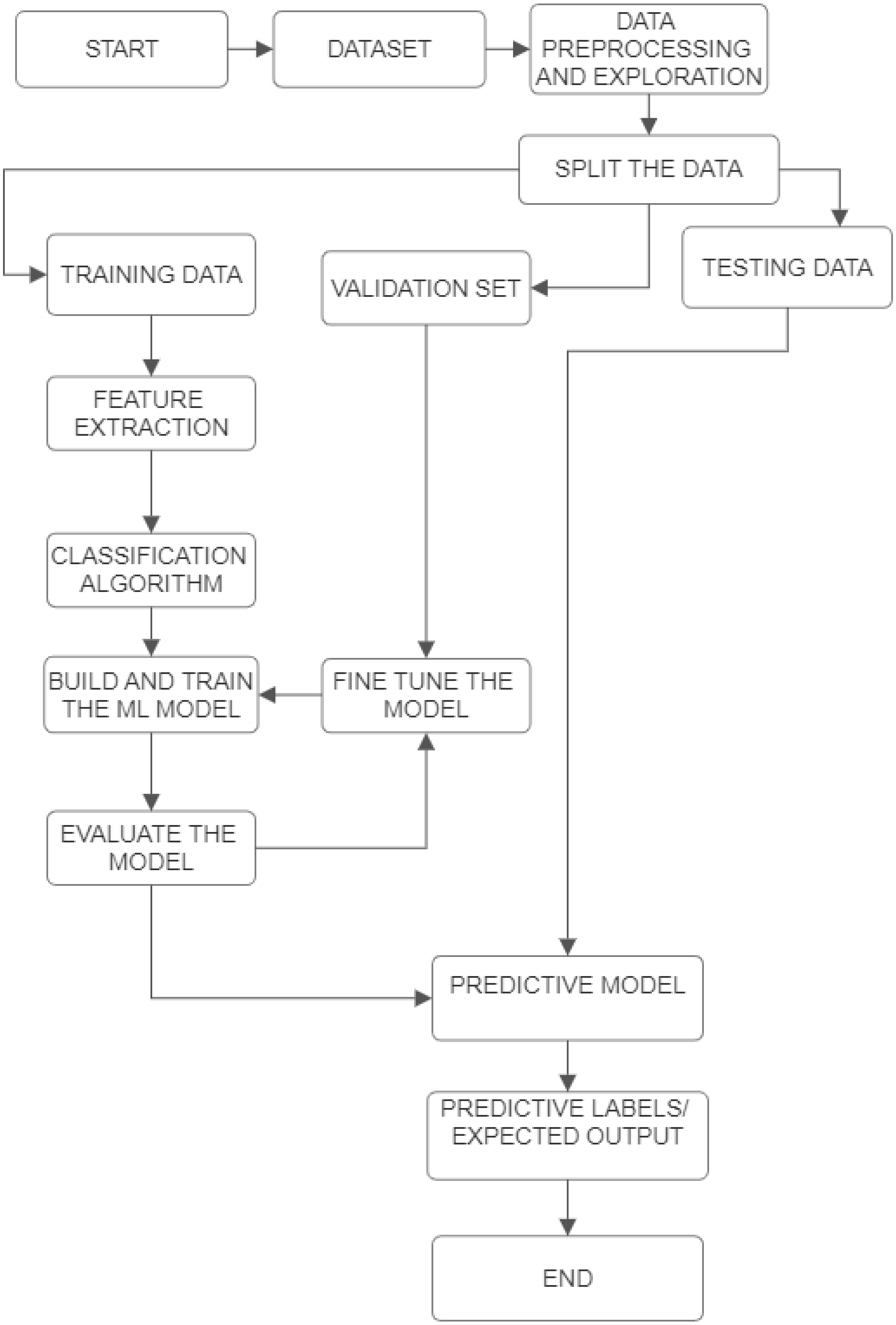
4. EXPERIMENTAL INVESTIGATIONS

Hyperparameter Tuning: Perform hyperparameter tuning for each model to find the optimal combination of hyperparameters that maximizes classification accuracy. This can be done using techniques like grid search or randomized search.

Handling Imbalanced Data: Investigate techniques to handle imbalanced classes if present in the dataset. This could involve oversampling techniques or undersampling techniques to balance the class distribution and potentially improve classification performance.

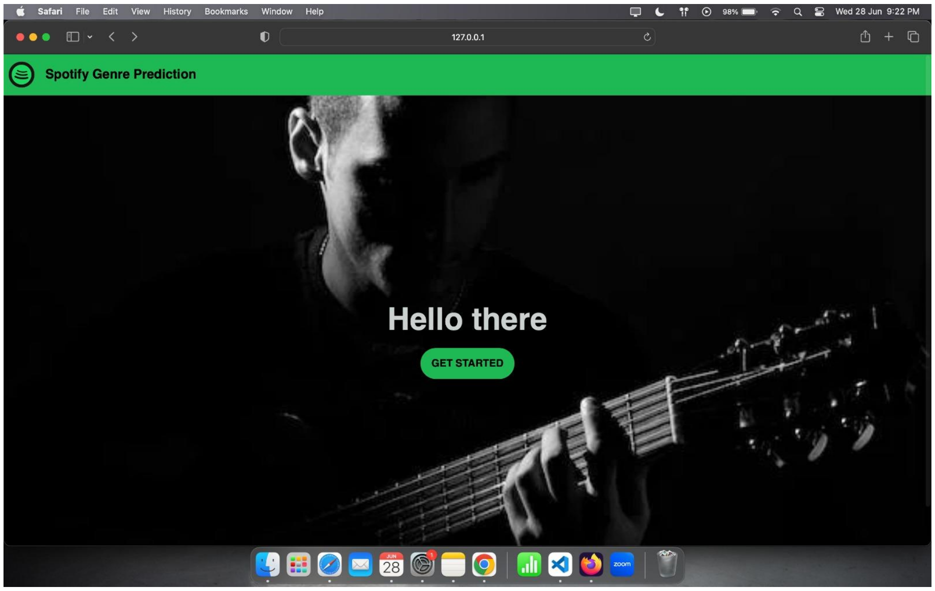
External Data Sources: Consider incorporating external data sources, such as lyrics or audio features from other platforms, to enrich the feature set and potentially improve genre classification accuracy. This could involve merging datasets or extracting additional features from external sources.

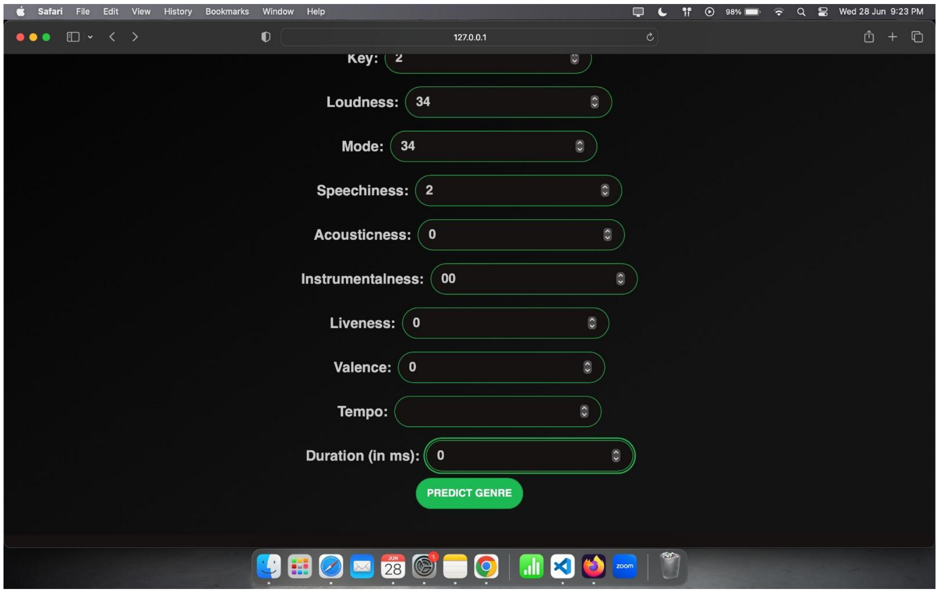
5. FLOW CHART


START
DATASET
DATA PREPROCESSING AND EXPLORATION
SPLIT THE DATA
TRAINING DATA
VALIDATION SET
TESTING DATA
FEATURE EXTRACTION
CLASSIFICATION ALGORITHM
BUILD AND TRAIN THE ML MODEL
FINE TUNE THE MODEL
EVALUATE THE MODEL
PREDICTIVE MODEL
PREDICTIVE LABELS/ EXPECTED OUTPUT
END

6. RESULT

RESULTS FROM THE WEBSITE





Key: 2

Loudness: 34

Mode: 34

Speechiness: 2

Acousticness: 0

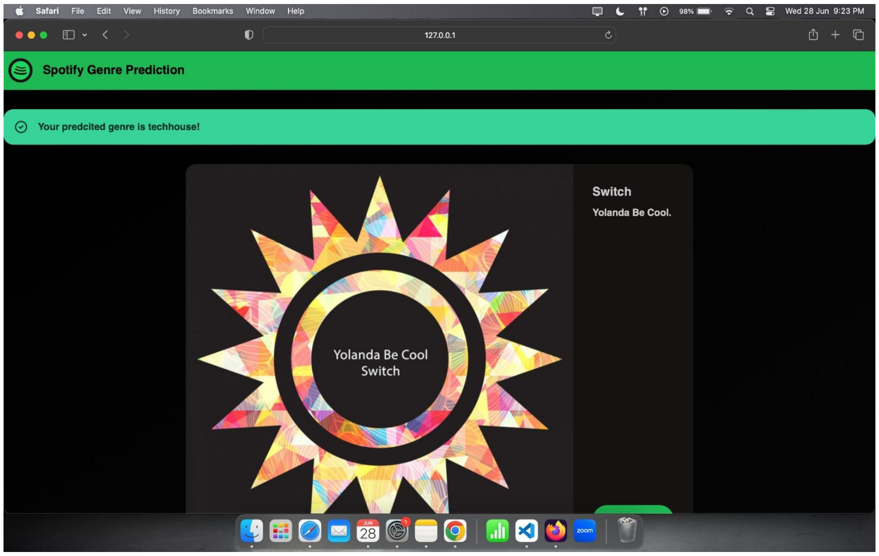
Instrumentalness: 00

Liveness: 0

Valence: 0

Tempo:

Duration (in ms): 0

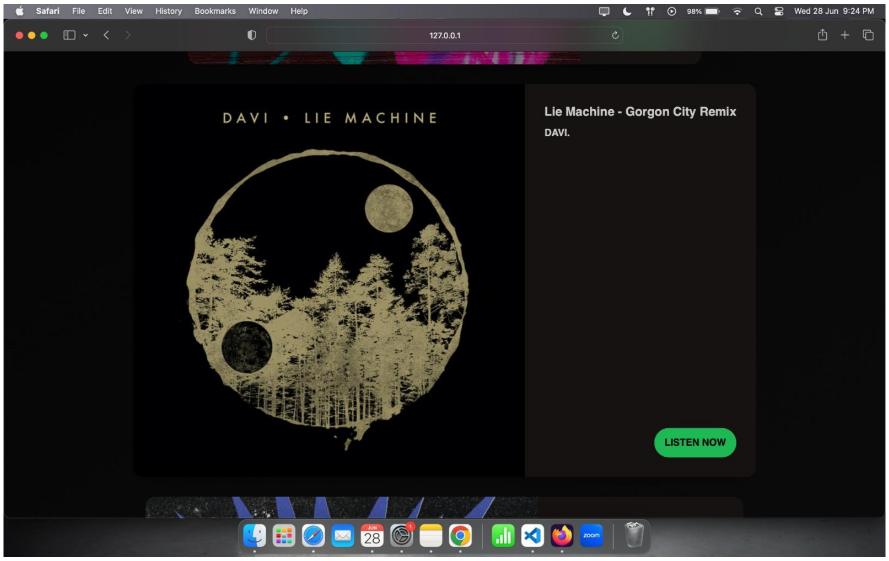


Spotify Genre Prediction

Your predcited genre is techhouse!

Switch

Yolanda Be Cool.



Lie Machine - Gorgon City Remix

DAVI.

7. ADVANTAGES & DISADVANTAGES

7.1. ADVANTAGES

7.1.1. Simple and clear Implementation of code.

7.1.2. Multi Model Options

7.1.3. Evaluation and Comparisons.

7.1.4. Preprocessing Techniques.

7.1.5. Model Persistence.

7.2. DISADVANTAGES

7.2.1. Limited Hyperparameter Tuning

7.2.2. Potential Overfitting

8. APPLICATIONS

8.1. Some Key Applications are:

8.1.1. Music Recommendation
8.1.2. Playlist Generation
8.1.3. User Personalization
8.1.4. Music Organization and Tagging
8.1.5. Music Listening and Royalty Distribution
8.1.6. Music Genre Analysis and Insights
8.1.7. Music Education and Research

9. CONCLUSION

In conclusion, we could categorize the Spotify music genres with an accuracy of around **84%** .

The machine learning-based music genre classification system on Spotify data is a complex and challenging task that requires the integration of various components and techniques. By leveraging the vast amount of audio features and metadata provided by Spotify, the system aims to accurately classify songs into their respective genres.

10. FUTURE SCOPE

The future scope of machine learning based music genre classification on Spotify Data is:

1. By using the Deep Learning Models such as CNNs and RNNs.
2. Hybrids Models combining multiple machine learning models.
3. Transfer learning, where pre-trained models from related tasks or datasets are used as a starting point, can be explored in the context of music genre classification.

11. BIBLIOGRAPHY

1. "Music Genre Classification: A Comparative Study of Feature Selection Methods" by E. Tzanetakis and P. Cook
2. "Automatic Musical Genre Classification of Audio Signals" by G. Tzanetakis and P. Cook
3. "Deep Content-Based Music Recommendation" by A. van den Oord et al.
4. "Music Genre Classification Using Machine Learning Techniques: A Comprehensive Review" by S. Lidy and A. Rauber
5. "Genre Classification of Music Content: A Review" by D. Lee and S. Lee

APPENDIX:

A. SOURCE CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import sys

df = pd.read_csv('genres_v2.csv')
df

for i in df.columns:
    print(i)

# Remove title, song_name, analysis_url, track_href, uri, id,
Unamed: 0
df = df.drop(['title', 'song_name', 'analysis_url', 'track_href',
'uri', 'id', 'Unnamed: 0', 'type'], axis=1)
```

```
df

# Check for null values
df.isnull().sum()

# Check for duplicates
df.duplicated().sum()

# Remove duplicates
df = df.drop_duplicates()
df

# Perform encoding for type and genre
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['genre'] = le.fit_transform(df['genre'])
df

# Separate the data into features and target
X = df.drop(['genre'], axis=1)
y = df['genre']

# Split the data into training and testing data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)
# Standardize the data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Model 1: Logistic Regression

from sklearn.linear_model import LogisticRegression
```

```
lr = LogisticRegression()
lr.fit(X_train, y_train)
lr.score(X_test, y_test)

print("Accuracy of Logistic Regression: ", lr.score(X_test,
y_test))

# Model 2: K Nearest Neighbors

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
knn.score(X_test, y_test)

print("Accuracy of K Nearest Neighbors: ", knn.score(X_test,
y_test))

# Model 3: Decision Tree

from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
dt.score(X_test, y_test)

print("Accuracy of Decision Tree: ", dt.score(X_test, y_test))

# Model 4: Random Forest

from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
rf.score(X_test, y_test)
```

```
print("Accuracy of Random Forest: ", rf.score(X_test, y_test))

# Gradient Boosting
from sklearn.ensemble import GradientBoostingClassifier
gb = GradientBoostingClassifier()
gb.fit(X_train, y_train)
gb.score(X_test, y_test)

print("Accuracy of Gradient Boosting: ", gb.score(X_test,
y_test))

# CatBoost Classifier
from catboost import CatBoostClassifier
cb = CatBoostClassifier()
cb.fit(X_train, y_train)
cb.score(X_test, y_test)

print("Accuracy of CatBoost Classifier: ", cb.score(X_test,
y_test))

# XGBoost Classifier
from xgboost import XGBClassifier
xgb = XGBClassifier()
xgb.fit(X_train, y_train)
xgb.score(X_test, y_test)

print("Accuracy of XGBoost Classifier: ", xgb.score(X_test,
y_test))

# Pickle the model
import pickle
pickle.dump(lr, open('model.pkl', 'wb'))
pickle.dump(le, open('encoder.pkl', 'wb'))
pickle.dump(scaler, open('scaler.pkl', 'wb'))


# Load the encoder, scaler, and model
model = pickle.load(open('model.pkl', 'rb'))
encoder = pickle.load(open('encoder.pkl', 'rb'))
scaler = pickle.load(open('scaler.pkl', 'rb'))

# Create a function to predict the genre
def predict_genre(danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, duration_ms):
   x = [[danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, duration_ms]]
   x = scaler.transform(x)
   x = pd.DataFrame(x)
   x.columns = ['danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo', 'duration_ms']
   prediction = model.predict(x)
   prediction = encoder.inverse_transform(prediction)
   return prediction

# Test the function
predict_genre(0.831, 0.814, 1, -7.364, 1, 0.420, 0.0598, 0.0134, 0.0556, 0.389, 98.002, 201907)
```