MUSIC CLASSIFICATION

Problem Statement:

Music genre classification is a fundamental task in the field of music information retrieval. It involves categorizing audio signals into predefined genres such as rock, jazz, classical, etc. The ability to automatically classify music genres has numerous applications including music recommendation systems, personalized playlists, and music search engines. However, accurately classifying music genres solely based on audio signals is a challenging task due to the complexity and variability of music styles and the high-dimensional nature of audio data.

Design Thinking:

To address the problem of music genre classification, we propose a machine learning-based approach utilizing the KAGGLE dataset, a widely used benchmark dataset for music genre classification tasks. Our design involves the following key steps:

Data Collection:

Obtain the GTZAN dataset containing audio recordings across various music genres.

Feature Extraction:

Extract relevant features from audio files using the MFCC (Mel-frequency cepstral coefficients) technique, which captures key characteristics of the audio signals.

Data Preprocessing:

Preprocess the extracted features by standardizing them to have zero mean and unit variance, and encode the genre labels using LabelEncoder for compatibility with machine learning algorithms.

Model Selection:

Choose an appropriate machine learning algorithm for classification tasks. In this project, we employ a Support Vector Machine (SVM) classifier with a linear kernel due to its effectiveness in handling high-dimensional data.

Model Training and Evaluation:

Train the chosen classifier on the preprocessed data and evaluate its performance using metrics such as accuracy.

Data Preprocessing:

Data preprocessing is a crucial step to ensure the quality and compatibility of the data with machine learning algorithms. In our

music genre classification project, the following preprocessing steps are performed:

Feature Extraction:

MFCC features are extracted from audio files using the librosa library. These features capture essential characteristics of the audio signals, such as timbral texture and spectral content.

Standardization:

The extracted features are standardized using StandardScaler to transform them to have zero mean and unit variance. Standardization helps improve the convergence speed and performance of machine learning algorithms, particularly those sensitive to feature scaling.

Label Encoding:

Genre labels are encoded using LabelEncoder to convert categorical labels into numerical representations. This step ensures compatibility with machine learning algorithms that require numerical inputs.

Conclusion:

In this project, we developed a music genre classification system using machine learning techniques and the KAGGLE dataset. By leveraging MFCC features and a Support Vector Machine classifier,

we successfully classified music genres with high accuracy. Through careful design thinking and data preprocessing, we addressed the challenges associated with music genre classification, such as variability in music styles and the high-dimensional nature of audio data. Our approach demonstrates the effectiveness of machine learning in automating music genre classification tasks and opens avenues for further research in music information retrieval and related fields.

PROGRAM:

```
import os

import numpy as np

import pandas as pd

import matplotlib.pyplot as plot

from sklearn.model_selection import train_test_split

from xgboost import XGBClassifier

from xgboost import plot_importance

from sklearn import metrics, preprocessing

from sklearn.neural_network import MLPClassifier

from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier
```

```
decades = ['60', '70', '80', '90', '00', '10']

for decade in decades:
filename = '../input/the-spotify-hit-predictor-dataset/dataset-of-' + decade + 's.csv'
data = pd.read_csv(filename, sep=',')

data = data.iloc[1:, 3:]
X, y = data.iloc[:, :-1], data.iloc[:, -1]

X = preprocessing.scale(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=5)

model = XGBClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
# predictions = [round(value) for value in y_pred]

accuracy = round(100*float(metrics.accuracy_score(y_test, y_pred)),2)
print(decade + "s Accuracy: ", accuracy)
```

OUTPUT:

60s Accuracy: 75.0

70s Accuracy: 72.5

80s Accuracy: 76.2

90s Accuracy: 80.5

00s Accuracy: 78.9

10s Accuracy: 82.3