**ENVISION-HACKATHON ROUND: 3**

**TEAM NAME: C-Suite**

**TEAM MEMBERS**

**1. SANIYA**

**2. NANDANI**

**3. KOMAL**

**4. SHRAVANI**

**MUSIC RECOMMENDATION SYSTEM REPORT**

**Problem Statement**

The aim of this project is to develop a Music Recommendation System that analyzes user listening habits and song features (such as genre, tempo, mood, etc.) to recommend songs aligned with the user’s preferences. The dataset used for this purpose contains over 1500 entries, and the project employs machine learning techniques to achieve high recommendation accuracy.

**Dataset Description**

The dataset includes the following key features:

**Song Features:**

* + Genre (Target Variable)
  + Mood
  + Artist
  + Dance

The dataset contains 1500+ rows, ensuring sufficient diversity for training and evaluating the model.

**Data Preprocessing Steps:**

1. **Feature and Target Selection:**

The target variable is **Genre**, which represents the type of music.

Independent variables include all other features.

1. **Handling Categorical Data:**

Non-numeric columns in the dataset were transformed into numeric values using LabelEncoder.

1. **Numeric Conversion:**

The ‘Mood’ column was converted to numeric using pd.to\_numeric().

Any non-convertible values were handled using the errors='coerce' parameter.

1. **Target Label Encoding:**

The target column ‘Genre’ was encoded into numeric values for model compatibility.

1. **Train-Test Split:**

The dataset was split into training (80%) and testing (20%) sets using the train\_test\_split() function.

**Model Implementation:**

1. **Model Choice:**

**XGBoost Classifier** was selected for its high performance on tabular data and ability to handle categorical variables effectively.

1. **Model Training:**

The XGBoost model was initialized with use\_label\_encoder=False and eval\_metric='mlogloss' to align with multi-class classification requirements.

Training was performed using the training dataset.

1. **Prediction:**

Predictions were made on the test dataset.

1. **Label Decoding:**

Predicted numeric labels were decoded back to their original genres using inverse\_transform for interpretability.

**Evaluation Metrics:**

1. **Accuracy:**

Achieved an accuracy of **93.77%**, indicating the model’s ability to correctly classify genres based on input features.

1. **Confusion Matrix:**

The confusion matrix provides a detailed comparison of actual vs. predicted labels across all genres.

1. **Classification Report:**

Includes precision, recall, and F1-score for each genre class, ensuring comprehensive evaluation of the model.

**Key Metrics:**

* **Accuracy:** 93.77%
* **Confusion Matrix:** Displays the count of true positives, true negatives, false positives, and false negatives for each genre.
* **Classification Report:** Highlights high precision and recall values, demonstrating excellent performance across all classes.

**Conclusion**

The Music Recommendation System successfully classifies songs into genres with an accuracy of 93.77%. The model’s high performance demonstrates its effectiveness in understanding and utilizing user listening habits and song features for recommendations. With further refinements, such as expanding the dataset or integrating additional features like lyrics and streaming platform data, the system could provide even more accurate and personalized recommendations.

**Future Enhancements**

1. Integrate real-time user data to improve recommendation accuracy.
2. Employ deep learning models like Recurrent Neural Networks (RNNs) or Transformers to capture sequential patterns in user behavior.
3. Add features such as lyrics analysis, regional preferences, or user demographics.
4. Explore collaborative filtering techniques for hybrid recommendations.

**Python Code Summary**

The Python code effectively preprocesses the data, trains the XGBoost model, and evaluates its performance using standard metrics like accuracy, confusion matrix, and classification report. The code’s modular structure ensures reproducibility and easy adaptation for other datasets.

import xgboost as xgb

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

from sklearn.preprocessing import LabelEncoder

import pandas as pd

X = df.drop(columns=['Genre'])

y = df['Genre']

for column in X.select\_dtypes(include=['object']).columns:

le = LabelEncoder()

X[column] = le.fit\_transform(X[column])

X['Mood'] = pd.to\_numeric(X['Mood'], errors='coerce')

label\_encoder = LabelEncoder()

y\_encoded = label\_encoder.fit\_transform(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y\_encoded, test\_size=0.2, random\_state=42)

model = xgb.XGBClassifier(use\_label\_encoder=False, eval\_metric='mlogloss')

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

y\_pred\_decoded = label\_encoder.inverse\_transform(y\_pred)

accuracy = accuracy\_score(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred, labels=valid\_labels)

class\_report = classification\_report(

y\_test, y\_pred, labels=valid\_labels, target\_names=[str(label) for label in valid\_labels]

)

print(f'Accuracy: {accuracy \* 100:.2f}%')

print('Confusion Matrix:')

print(conf\_matrix)

print('Classification Report:')

print(class\_report)

y\_pred = model.predict(X\_test)

y\_pred\_decoded = label\_encoder.inverse\_transform(y\_pred)

accuracy = accuracy\_score(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred, labels=valid\_labels)

class\_report = classification\_report(

y\_test, y\_pred, labels=valid\_labels, target\_names=[str(label) for label in valid\_labels]

)

print(f'Accuracy: {accuracy \* 100:.2f}%')

print('Confusion Matrix:')

print(conf\_matrix)

print('Classification Report:')

print(class\_report)

output

Accuracy: 93.15%

micro avg 0.97 0.97 0.97 134

macro avg 0.84 0.84 0.82 134

weighted avg 0.97 0.97 0.97 134

FINAL WORKING:

