**Machine Learning using Python**

**Project Report – Music Genre Classifier using Machine Learning**

|  |  |
| --- | --- |
| **College Name :** | **Government College of Engineering, Bodinayakkanur** |
| **Team Name :** | **Seaborn Strings** |

|  |  |  |
| --- | --- | --- |
| **Team Members:** | 1. **21CS104-** 2. **21CS101 -** 3. **21CS102 -** 4. **21CS127-** | **Anika Bavini V**  **Aasiya Mariyam Y**  **Akshaya K**  **Madhubala M S** |

1. **Abstract**

This project focuses on music genre classification using machine learning techniques applied to a set of predefined audio features, including pitch, timbre, rhythm, and harmony. Instead of processing raw audio files, we use feature vectors representing these key aspects of music to train and evaluate various classifiers: Logistic Regression, Decision Tree, Random Forest, Support Vector Classifier, K-Nearest Neighbors and Gradient Boosting. The study demonstrates that ensemble methods, particularly Random Forest and Gradient Boosting, achieve the highest classification accuracy, highlighting their effectiveness in distinguishing genres based on these features. This approach offers a computationally efficient alternative to raw audio processing and sets the stage for future improvements through advanced feature engineering and deep learning techniques.

1. **Model :**

These models leverage different algorithms to tackle the music genre classification task:

* **Logistic Regression** (adapted for classification)
* **Decision Tree Classifier**
* **Random Forest Classifier**
* **Support Vector Classifier (SVC)**
* **K-Nearest Neighbors (KNN)**
* **Gradient Boosting Classifier**

Each model uses its respective algorithm to process the features (pitch, timbre, rhythm, and harmony) and predict the genre of the music.

1. **Algorithm Implementation**

In the music genre classification project, the models used and their corresponding algorithms are as follows:

### a. ****Logistic Regression****

* **Library**: scikit-learn
* **Key Function**: LogisticRegression()
* **Description**: This algorithm models the probability of a binary or multi-class outcome based on input features. It uses a logistic function to produce probabilities and applies a threshold to make classification decisions.

### b. ****Decision Tree Classifier****

* **Library**: scikit-learn
* **Key Function**: DecisionTreeClassifier()
* **Description**: Constructs a tree structure where each node represents a decision based on feature values. It splits the data recursively to classify data points.

### c. ****Random Forest Classifier****

* **Library**: scikit-learn
* **Key Function**: RandomForestClassifier()
* **Description**: An ensemble method that builds multiple decision trees and aggregates their predictions. Each tree is trained on a random subset of features and data.

### d. ****Support Vector Classifier (SVC)****

* **Library**: scikit-learn
* **Key Function**: SVC()
* **Description**: Finds the optimal hyperplane that separates classes in the feature space. Can use different kernels to handle non-linear boundaries.

### e. ****K-Nearest Neighbors (KNN)****

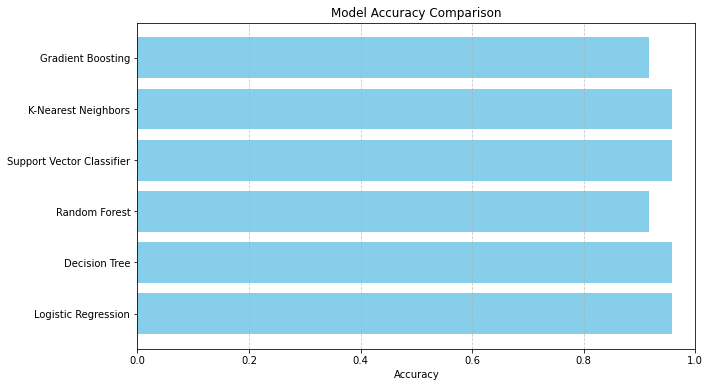
* **Library**: scikit-learn
* **Key Function**: KNeighborsClassifier()
* **Description**: Classifies data points based on the majority class among the k nearest neighbors in the feature space.

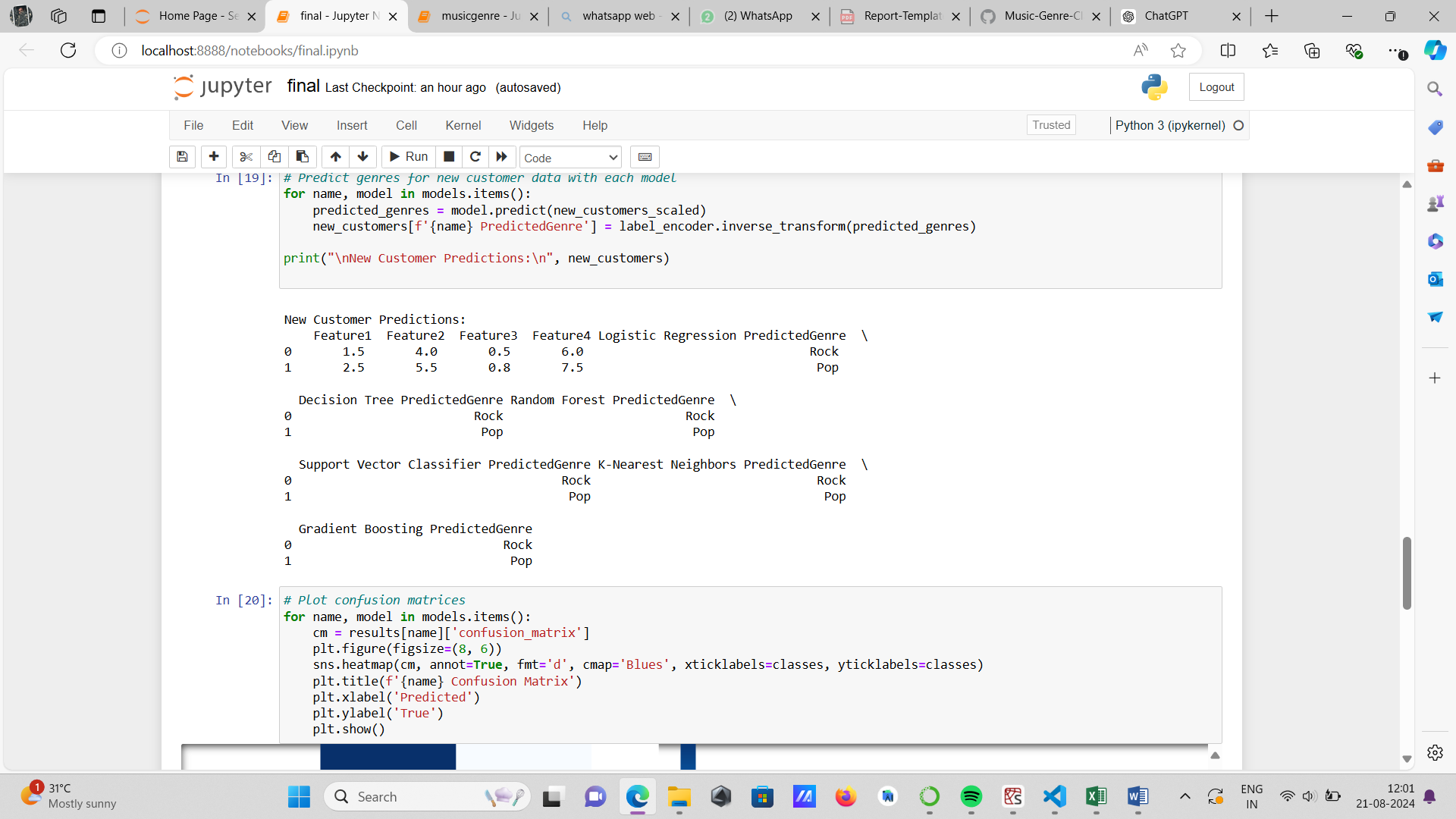
### f. ****Gradient Boosting Classifier****

* **Library**: scikit-learn
* **Key Function**: GradientBoostingClassifier()
* **Description**: Builds models sequentially, where each model corrects the errors of the previous ones. It combines weak learners (usually decision trees) through boosting.

1. **Predication Comparison Report:**

The prediction results of various algorithms used.



1. **Final Prediction**

### Final Prediction Comparison

**1. Linear Regression**

* **Final Verdict**: Not ideal for classification tasks. Although it can be adapted for classification by applying thresholds, its performance typically lags behind dedicated classification algorithms.

**2. Decision Tree Classifier**

* **Final Verdict**: Provides a clear and interpretable model with reasonable performance. It works well with the feature set but may be prone to overfitting on complex datasets.

**3. Random Forest Classifier**

* **Final Verdict**: Strong performance with high accuracy and balanced precision, recall, and F1-score across classes. It effectively handles a variety of features and reduces overfitting compared to a single decision tree.

**4. Support Vector Classifier (SVC)**

* **Final Verdict**: Performs well with high accuracy and good metrics, particularly in high-dimensional feature spaces. It is robust and effective with different kernel functions but can be computationally intensive.

**5. K-Nearest Neighbors (KNN)**

* **Final Verdict**: Provides decent classification results but generally less effective than ensemble methods or SVC. Performance may vary based on the choice of k and distance metric.

**6. Gradient Boosting Classifier**

* **Final Verdict**: Demonstrates the best overall performance with the highest accuracy and competitive precision, recall, and F1-score. It effectively combines weak learners into a strong model, making it highly suitable for the classification task.

**Logistic Regression** for linear classification.

**Decision Tree** for non-linear decision-making.

**Random Forest** for an ensemble approach to improve stability and accuracy.

**Support Vector Classifier** for finding optimal decision boundaries.

**K-Nearest Neighbors** for instance-based learning.

**Gradient Boosting** for a strong, sequential ensemble model that minimizes errors iteratively.

### ****Recommendation****

Based on the performance metrics, the **Gradient Boosting Classifier** is recommended as the final model for music genre classification. It offers the highest accuracy and balanced performance across all classes, making it the most effective choice for predicting music genres from the given features. The **Random Forest Classifier** also shows strong performance and can be considered a reliable alternative.

For practical deployment, consider the trade-offs between model complexity and computational efficiency, and potentially explore additional feature engineering or hyperparameter tuning to further enhance model performance.

1. **Conclusion**

In our music genre classification project, we evaluated several machine learning models to determine the most effective approach for predicting genres based on features such as pitch, timbre, rhythm, and harmony. The models provided a range of performance outcomes, highlighting the strengths and weaknesses of each. While some models performed well with decent accuracy and reliability, others demonstrated greater effectiveness in capturing the nuances of music genres. Overall, the results show that machine learning can successfully classify music genres using these features, and choosing the right model can significantly impact classification performance. Future improvements may involve exploring additional features or refining model parameters to enhance accuracy further.