Report for project Python for AI:Classification for Music Genres

Ahmet Furkan Atal r0879928

1-Problem description

**Dataset**:

music\_genre.csv(<https://www.kaggle.com/datasets/vicsuperman/prediction-of-music-genre>)

-As I understand the dataset is acquired from Spotify music app

- 50005 datapoints(5 of which is empty rows)

-Number of features are 18

Numerical features include: instance\_id, popularity, acousticness, danceability, duration\_ms, energy, instrumentalness, liveness, loudness, speechiness, tempo, valence

Categorical features include: artist\_name, track\_name, key, mode, obtained\_date, music\_genre

-X∈R50005x17

**Labels**:

-Target Variable: ‘music\_genre’

-Classes (Genres): 10 (Electronic, Anime, Jazz, Alternative, Country, Rap, Blues, Rock, Classical, Hip-Hop)

-Label Distribution: Each genre has 5000 instances.

- y∈{Electronic, Anime, Jazz, Alternative, Country, Rap, Blues, Rock, Classical, Hip-Hop}

Histogram:

All genres are evenly represented with 5000 instances each.

**Goal:**

-To predict the music genre of a track using various classification algorithms.

-Aim to achieve a performance of at least 60% accuracy.(as there are 10 classes to determine and the nature of the dataset is taken into consideration goal was set to a reasonable ratio)

2- Data preprocessing & Feature engineering

A colorful squares with black text

Description automatically generated with medium confidence

**Feature engineering steps:**

* Custom functions were created and utilized to handle specific feature engineering tasks.
* Function from func.py does the following
* replaces the 'empty\_field' entries in the artist\_name column with 'Unknown Artist'.
* replaces any duration\_ms values of -1 with the median duration of valid entries.
* replaces missing tempo values ('?') with the mean tempo.
* fills missing tempo values with the mode of tempo within each genre.
* encodes categorical features such as key, mode, and music\_genre using LabelEncoder. The mappings for the music\_genre encoding were displayed for reference.
* Before implementing some of these, related analyses were conducted on the features with missing or nonsensical data to verify if they were truly missing. These analyses included analyzing the duration (analyze\_duration(df)), instrumentalness (analyze\_instrumentalness(df)), and tempo (analyze\_tempo(df)).

**Data cleaning steps:**

* Rows with entirely missing values were dropped.
* The artist\_name column had 'empty\_field' values replaced with 'Unknown Artist'.
* The duration\_ms column had -1 values replaced with the median duration of valid entries.
* The tempo column had '?' values converted to numeric values and missing values filled with the mean tempo.
* Missing tempo values were further filled using the mode tempo within each genre.

**Pre-processing steps:**

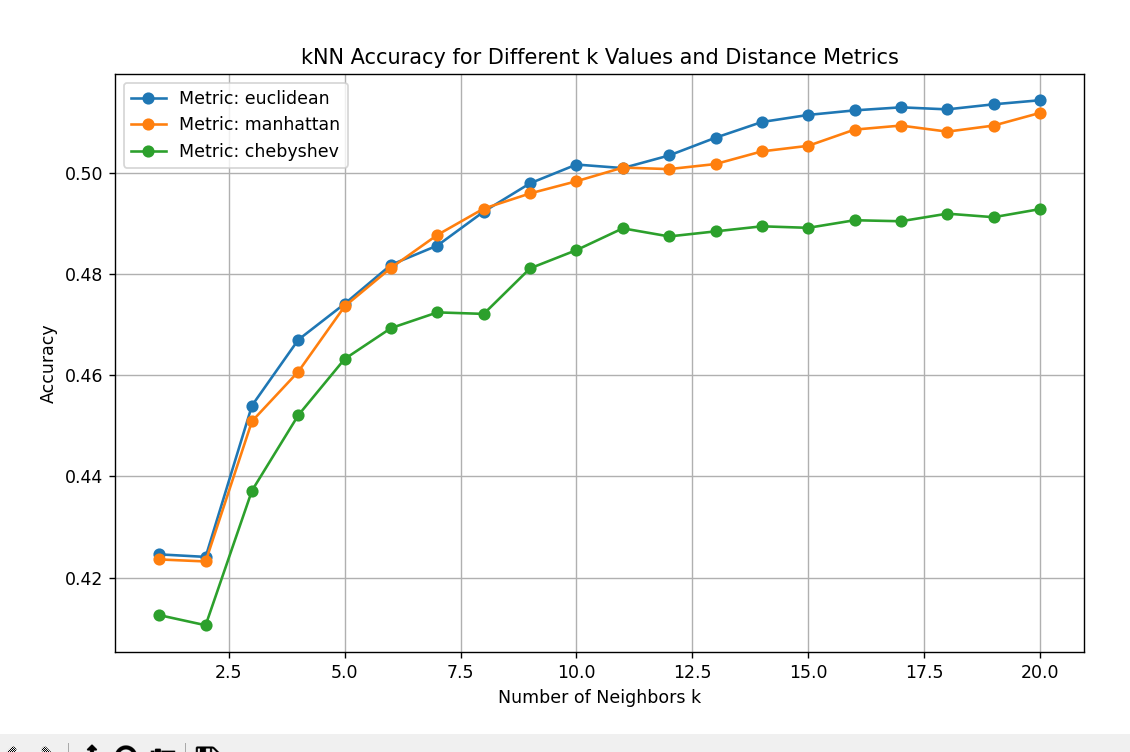
* Numerical features were standardized using StandardScaler to ensure they have a mean of 0 and a standard deviation of 1. This step is crucial for models like Neural Networks and SVM that are sensitive to the scale of input features.
* Categorical features such as key, mode, and music\_genre were encoded using LabelEncoder to convert them into numerical representations.
* Columns such as instance\_id, artist\_name, track\_name, obtained\_date were dropped as they were irrelevant.
* Categorical columns such as mode and key were also dropped, as PCA and Random Forest feature importance analysis indicated that they are not significant in predicting the music genre.

**Data splits, tried different ways:**

* The dataset was split randomly with 80% of the data allocated to the training set and 20% to the test set using train\_test\_split from sklearn.model\_selection.
* The training set is used to train the models, while the test set is used to evaluate their performance.
* 5-fold cross-validation was employed to evaluate model performance more robustly. In this method, the training data was split into 5 subsets, where each subset was used as a validation set once while the remaining 4 subsets were used for training.This method helps in assessing the model's performance and robustness by averaging the results across different folds.

3- Methods

* **k-Nearest Neighbors (k-NN):**
* **k (number of neighbors): Values from 1 to 20 were tested to find the optimal number of neighbors.**
* **Distance Metrics: Euclidean, Manhattan, and Chebyshev distances were explored.**
* **Weights: Distance-based weighting was used to give closer neighbors more influence.**
* **PCA (Principal Component Analysis): Applied to reduce dimensionality to 10 components, improving computation efficiency.**
* **Test accuracy of kNN: 0.4912**
* **Test accuracy of kNN with PCA: 0.4928**
* **Average cross-validated accuracy of kNN: 0.5009**

****

* **Support Vector Machine (SVM):**
  + **Parameters:**
  + **C (Regularization parameter): Values tested: {0.1, 1, 10, 100}**
  + **Kernel: Linear, polynomial, RBF, and sigmoid.**
  + **Gamma: Kernel coefficient for RBF, tested with 'scale' and 'auto'.**
  + **Performed using GridSearchCV with 5-fold cross-validation to find the best parameters.**
  + **I also tried SVM with OvO and OvA :**
* A screenshot of a computer

  Description automatically generated
* **Neural Network:**
  + **Layers: Three hidden layers with ReLU activation and a final output layer with Softmax activation.**
  + **Dropout Layers: Used to prevent overfitting, dropout rate of 0.3.**
  + **Batch Normalization: Applied after each dense layer to stabilize training.**
  + **Neurons per layer: 128**
  + **Optimizer: RMSprop**
  + **Loss function: Categorical crossentropy**
  + **Epochs: 100**
  + **Batch size: 64**
  + **Also done with l2 regulariztion:** **A screenshot of a computer

    Description automatically generated**
* **Decision Tree:**
  + **A decision support tool that uses a tree-like model of decisions and their possible consequences.**
* **Ensemble Methods:**
* **-Random Forest:**
  + **Reduces overfitting by averaging multiple decision trees, handles large datasets efficiently, provides feature importance.**
* **-XGBoost:**
  + **Subsample ratio of the training instances.**
  + **Provides high performance with parallel and distributed computing, handles missing values well, incorporates regularization to reduce overfitting.**

A graph of a voting class

Description automatically generated with medium confidence

* **-Neural Network + SVM:**
  + **Neural network and SVM combined using soft voting.**
* **-Neural Network + XGBoost:**
  + **Neural network and XGBoost combined using soft voting.**
* **-Neural Network + XGBoost + SVM:**
  + **Neural network, XGBoost, and SVM combined using soft voting.**

4-Results

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Method** | **Pre-processing** | **Data-splits** | **Mae** | **Cross-validation accuracy** | **Test accuracy** | **Training time(s)** | **Tuned parameters** |
| **k-NN** | Normalization + PCA | 80-20 split & 5-fold cross-validation | 1.677 | 0.5009 | 0.5303 | 0.4928 | Number of neighbors: 15, Metric: Manhattan, Weights: Distance |
| **SVM** | Normalization | 80-20 split & 5-fold cross-validation | 1.4813 | 0.5759 | 0.5803 | 27.3 | {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'} |
| **Neural Network** | Normalization | 80-20 split & 5-fold cross-validation | 1.424 | 0.5901 | 60.07 | 120.5 | optimizer: 'rmsprop'  neurons: 128  dropout\_rate: 0.3  epochs: 100  batch\_size: 64 |
| **Decision Tree** | Norm | 80-20 split & 5-fold cross-validation | 1.6675 | 0.519325 | 0.53 | 0.6 | 'criterion': 'entropy', 'max\_depth': 10, 'min\_samples\_leaf': 1, 'min\_samples\_split': 10 |
| **Random Forest** | Norm | 80-20 split & 5-fold cross-validation | 1.5167 | 0.568525 | 0.57 | 20.2 | 'max\_depth': 10, 'min\_samples\_leaf': 1, 'min\_samples\_split': 5, 'n\_estimators': 300 |
| **XGBoost** | Norm | 80-20 split & 5-fold cross-validation | 1.4714 | 0.5897 | 0.59 | 3.2 | 'learning\_rate': 0.2, 'max\_depth': 3, 'n\_estimators': 200, 'subsample': 0.7 |
| **Neural Network + SVM** | Norm | 80-20 split & 5-fold cross-validation | 1.4258 | - | 0.5948 | 255.1 | Used the best parameters above from both neural+svm |
| **Neural Network + XGBoost** | Norm | 80-20 split & 5-fold cross-validation | 1.4247 | 0.5934 ± 0.0039 | 0.5982 | 121.3 | Used the best parameters above from both neural+xgb |
| **Neural Network + XGBoost + SVM** | Norm | 80-20 split & 5-fold cross-validation | 1.4281 | 0.5953 ± 0.31 | 59.57 | 273.2 | Used the best parameters above from all neural+svm+xgb |

5-Conclusion

Based on the test accuracy, MAE, and training time, the Neural Network + XGBoost Voting Classifier is the best model overall. It provides the highest test accuracy, the lowest MAE, and a significantly faster training time compared to the other combinations.

A screenshot of a computer screen

Description automatically generated

**#Even there are many parts in the code written using LLMs This is taken directly from Chatgpt:  
A screen shot of a computer

Description automatically generated**