Music Genre Classification

Name:	**Abhinav Narayan**
Registration No./Roll No.:	17005
Institute/University Name:	IISER Bhopal
Program/Stream:	Physics
Problem Release date:	February 02, 2022
Date of Submission:	April 24, 2022

1 Introduction

The goal of this project is to build a machine learning classifier that can identify the genre of a music sample from a given data set of its spectral features. I have worked on this project in a group with Aditya Batra and Jayant Singh Bhati.

Data

The given data comprises of a set of spectral features of 900 music samples which are 30 second long divided into 10 genre classes.

Each data point contains 57 spectral features which contain information about the mean and variance of the following:

1. Chroma short-term Fourier transformation
2. Root Mean Square level
3. Spectral Centroid
4. Spectral Bandwidth
5. Spectral Rolloff
6. Zero Crossing Rate
7. Harmony
8. Percussion
9. Tempo
10. Mel-frequency cepstral coefficients (MFCCs): 20

2 Methods

To find the best performing classification framework for the given data, we assemble a pipeline that cross-validates various classifiers while setting different parameters using a suitable parameter tuning method.

2.1 Pre-Processing

For preprocessing the data to make it easier to fit with the classifier we have used StandardScaler to normalise the data.

2.2 Classification Techniques

We use Decision Tree, Random Forest, K-Nearest Neighbors, Support Vector Machine(SVM), Multi-layer Perceptron (MLP) and Logistic Regression classifiers. I implemented the Random forest and KNN which are described below.

2.2.1 Random Forest Classifier

A Random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

2.2.2 K-nearest Neighbours

The K nearest neighbour classifier classifies data by ranking nearest neighbours according to their proximity.

2.3 Boosting

We have used adaboost on the classifiers with bad performance and have done the gridsearch on best estimators.

2.3.1 AdaBoost

```
be1 = svm.SVC(kernel='linear', class_weight='balanced',probability=True)
be2 = LogisticRegression(solver='liblinear',class_weight='balanced')
be3 = DecisionTreeClassifier(max_depth=50)
#            clf = AdaBoostClassifier(algorithm='SAMME',n_estimators=100)
clf = AdaBoostClassifier(algorithm='SAMME.R',n_estimators=100)
clf_parameters = {
'clf__base_estimator':(be1,be2,be3),
'clf__random_state':(0,10),
}
```

2.4 Feature Selection

Feature selection refers to techniques that select a subset of the most relevant features (columns) for a dataset. Fewer features can allow machine learning algorithms to run more efficiently (less space or time complexity) and be more effective.

We use χ^2 statistic, mutual information gain and Recursive Feature Elimination (RFE) feature engineering techniques. I implemented mutual information classification.

2.4.1 Mutual information

Mutual information (MI) between two random variables is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency.

The function relies on nonparametric methods based on entropy estimation from k-nearest neighbors distances.

2.5 Parameter Tuning

We have used the Grid Search to tune the parameters. We passed the classifiers one by one through the pipeline with different grid parameters.

2.5.1 Grid Search

```
grid = GridSearchCV(pipeline,clf_parameters,scoring='f1_micro',cv=10)
grid.fit(X_train,y_train)
clf= grid.best_estimator_
```

3 Experimental Analysis

3.1 Feature selection

Using χ^2 statistics, mutual information gain and Recursive Feature Elimination (RFE) we found the best number of features to be 45 with mutual information classifier.

3.2 Evaluation Criteria

To evaluate the performance of each classifier, we use the k-fold cross-validation technique.

```
skf = StratifiedKFold(n_splits=3)
```

to and precision, recall, f-measure, micro and macro averaged techniques as evaluation criteria. We ran different number of splits for each classifier to analyze their performance and get confidence score. For the best parameters with the highest confidence score we ran them on the test split part of the training data.

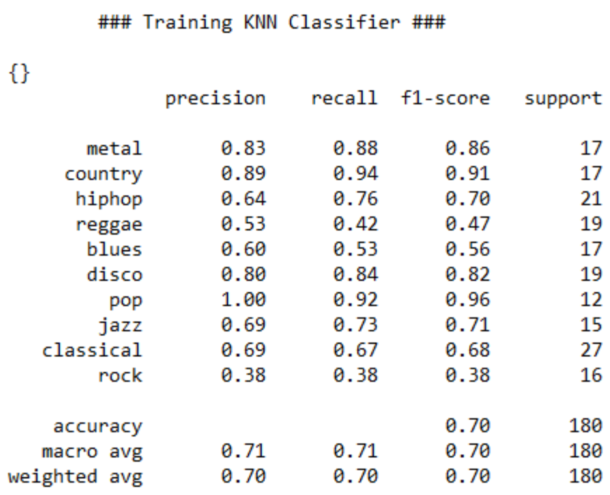
4 Analysis of Results

The results on training data for Random forest and KNN classifiers are highlighted below. The best grid parameters for Random forest we got was:

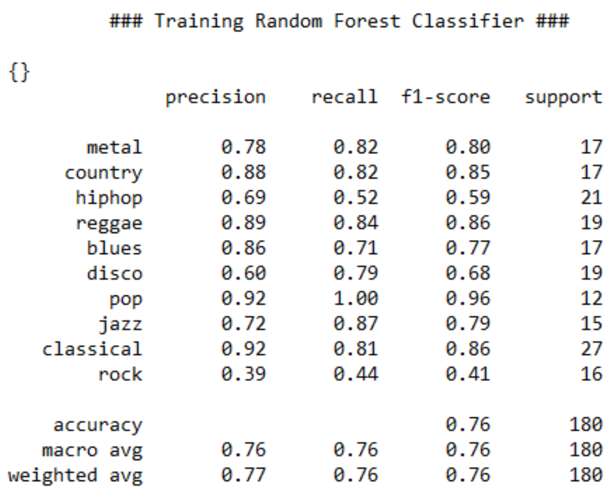
```
  clf=RandomForestClassifier(class_weight='balanced',criterion='entropy'
   ,max_depth=50,n_estimators=200)
```

The best grid parameters for KNN was:

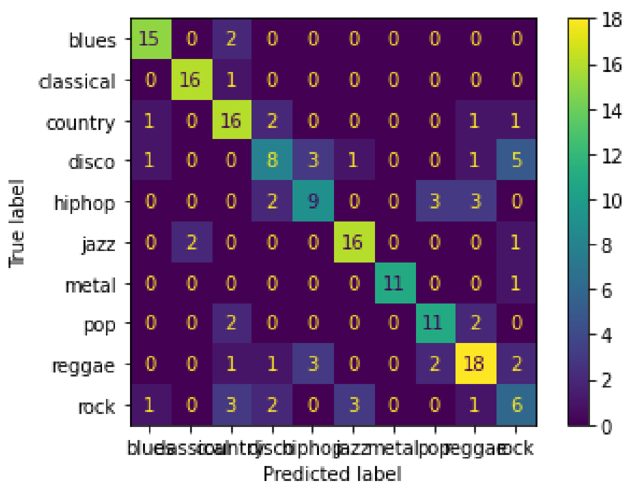
```
 clf=  KNeighborsClassifier(n_neighbors=3,weights='distance',metric='manhattan')
```

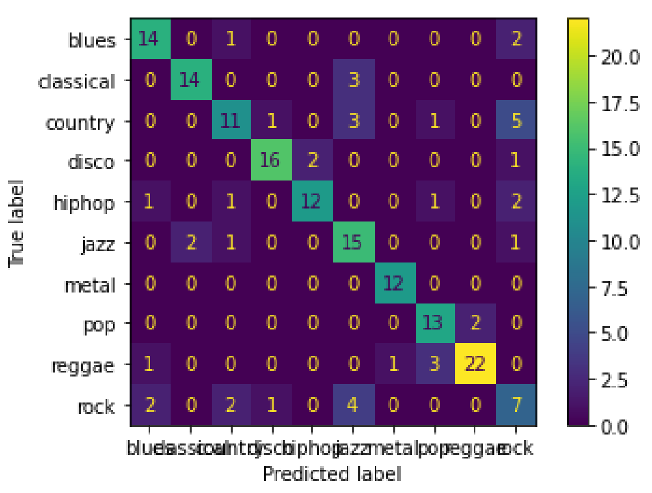


```
        ### Training KNN Classifier ###

{}
              precision    recall  f1-score   support

       metal       0.83      0.88      0.86        17
     country       0.89      0.94      0.91        17
      hiphop       0.64      0.76      0.70        21
      reggae       0.53      0.42      0.47        19
       blues       0.60      0.53      0.56        17
       disco       0.80      0.84      0.82        19
         pop       1.00      0.92      0.96        12
        jazz       0.69      0.73      0.71        15
   classical       0.69      0.67      0.68        27
        rock       0.38      0.38      0.38        16

    accuracy                           0.70       180
   macro avg       0.71      0.71      0.70       180
weighted avg       0.70      0.70      0.70       180
```



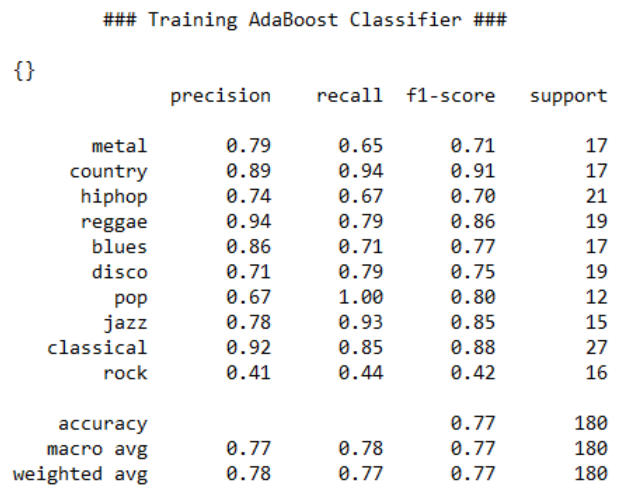
```
         ### Training Random Forest Classifier ###

{}
              precision    recall  f1-score   support

       metal       0.78      0.82      0.80        17
     country       0.88      0.82      0.85        17
      hiphop       0.69      0.52      0.59        21
      reggae       0.89      0.84      0.86        19
       blues       0.86      0.71      0.77        17
       disco       0.60      0.79      0.68        19
         pop       0.92      1.00      0.96        12
        jazz       0.72      0.87      0.79        15
   classical       0.92      0.81      0.86        27
        rock       0.39      0.44      0.41        16

    accuracy                           0.76       180
   macro avg       0.76      0.76      0.76       180
weighted avg       0.77      0.76      0.76       180
```



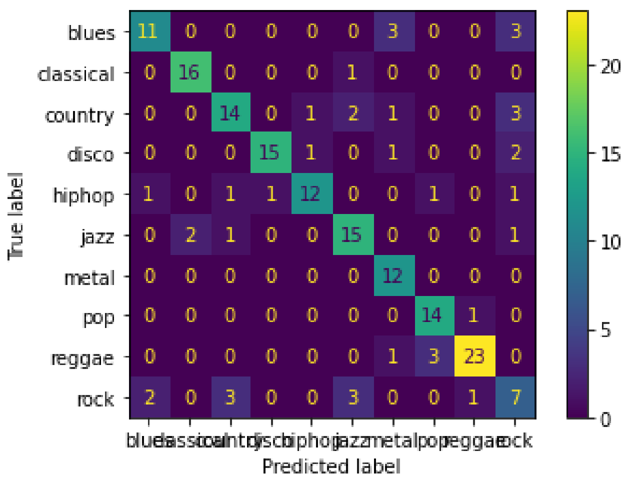


The best esimator for our model was Adaboost using decision tree as the base algorithm. The best parameters are:

```
clf = AdaBoostClassifier( DecisionTreeClassifier(max_depth=50),algorithm='SAMME.R',
 n_estimators=100,random_state=10)
```



```
        ### Training AdaBoost Classifier ###

{}
              precision    recall  f1-score   support

       metal       0.79      0.65      0.71        17
     country       0.89      0.94      0.91        17
      hiphop       0.74      0.67      0.70        21
      reggae       0.94      0.79      0.86        19
       blues       0.86      0.71      0.77        17
       disco       0.71      0.79      0.75        19
         pop       0.67      1.00      0.80        12
        jazz       0.78      0.93      0.85        15
   classical       0.92      0.85      0.88        27
        rock       0.41      0.44      0.42        16

    accuracy                           0.77       180
   macro avg       0.77      0.78      0.77       180
weighted avg       0.78      0.77      0.77       180
```



5 Discussions and Conclusion

We got the best results with the Adaboost classifier because it assigns weights to the misclassified data points for a poorly performing classifier like decision tree. The musical data has a lot of features and low variance over the different classes so a tree algorithm like random forest will work well, which is also what we got. The code is uploaded in the following repository.
https://github.com/Relativestein/Music-Genre-Classification