Project Report

Title : Classification of Digits Data Using Audio Signal Processing

Submitted By:

Afsaruddin Mohammed
Net Id: Axm210415

Problem Statement:

Here we are trying to classify audio sample contain of spoken digits (0-2) of 60 different speakers. Hence, we have 3 categories that we are trying to classify using machine learning algorithms. To Compare these results with synthesized music (machine made music) we generated music files from three generator functions from Magenta, TensorFlow library: Attention RNN, Basic RNN, and Lookback RNN. And we tried to classify using same machine learning algorithms and provided analysis of the results.

2 Feature Extraction from Audio Files

2.1 Audio file generation

We have used the pre-trained models available for basic RNN, lookback RNN, and attention RNN and run them 200 times, each by varying the primer-melody sequences to produce distinct MIDI piano-roll files. The MIDI files thus extracted are converted to WAV files for further processing.

2.2 Using Librosa to extract MFCCs.

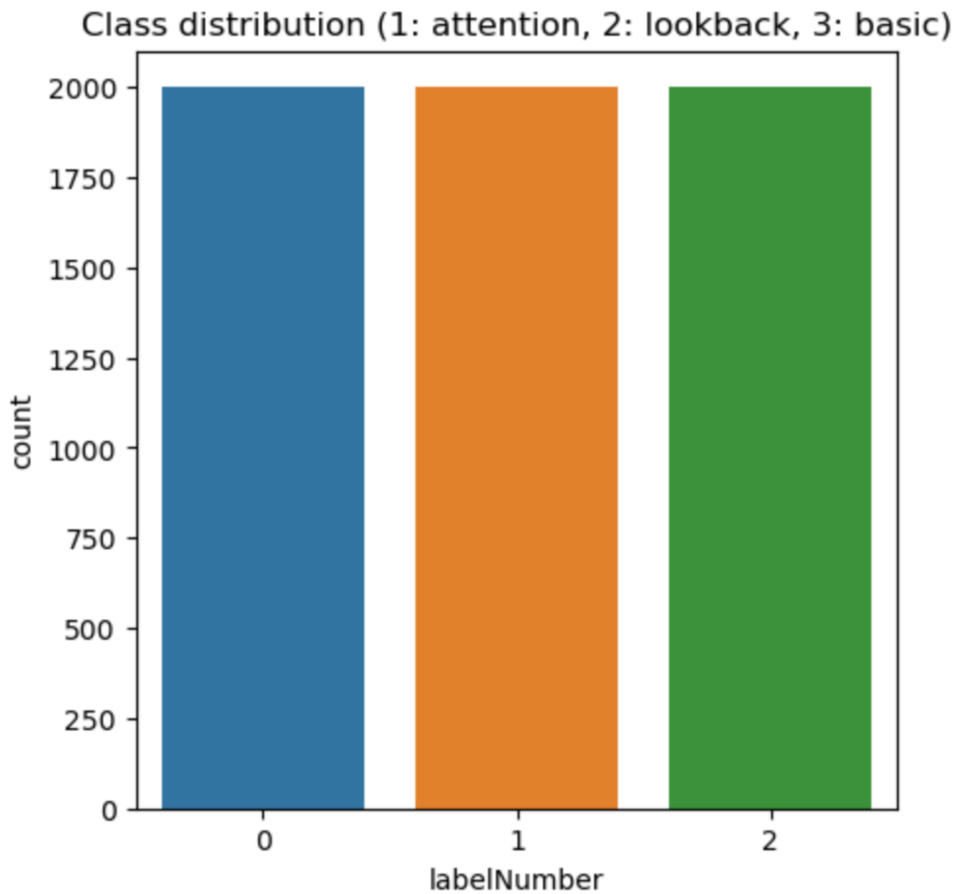
Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. The MFCC algorithm processes the entire speech data in a batch. Based on the number of input rows, the window length, and the overlap length, MFCC partitions the speech into frames and computes the cepstral features for each frame. Using librosa package, the synthesized audio file is divided into audio segments of 1 second each and then MFCC are extracted from each segment to minimize the information loss.

3 Data Set Description:

3.1 Synthesized Data

After performing feature extraction on 600 audio files generated from the magenta library. The data set thus obtained had 6000 examples and each example had 13 features. Each example will belong to one of 3 classes (attention, lookback, basic).

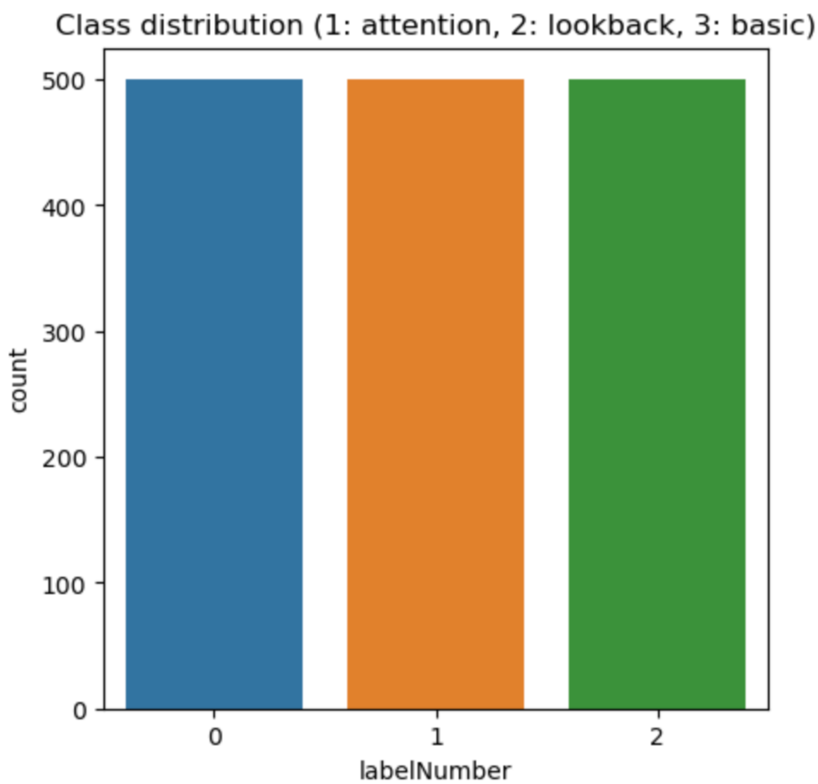
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6000 entries, 0 to 5999
Data columns (total 16 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0   6000 non-null   int64
 1   className    6000 non-null   object
 2   labelNumber  6000 non-null   int64
 3   team1        6000 non-null   float64
 4   team2        6000 non-null   float64
 5   team3        6000 non-null   float64
 6   team4        6000 non-null   float64
 7   team5        6000 non-null   float64
 8   team6        6000 non-null   float64
 9   team7        6000 non-null   float64
 10  team8        6000 non-null   float64
 11  team9        6000 non-null   float64
 12  team10       6000 non-null   float64
 13  team11       6000 non-null   float64
 14  team12       6000 non-null   float64
 15  team13       6000 non-null   float64
dtypes: float64(13), int64(2), object(1)
memory usage: 750.1+ KB
```



3.2 Real World Digit Data:

A subset of Audio MNIST data is taken which has samples of spoken digits (1-3) of different speakers. The dataset contains 1500 audio files (500 in each class/digit). This data is collected to compare the results of classification algorithms used on Synthesized data.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 16 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0   1500 non-null   int64
 1   className    1500 non-null   int64
 2   labelNumber  1500 non-null   int64
 3   team1        1500 non-null   float64
 4   team2        1500 non-null   float64
 5   team3        1500 non-null   float64
 6   team4        1500 non-null   float64
 7   team5        1500 non-null   float64
 8   team6        1500 non-null   float64
 9   team7        1500 non-null   float64
 10  team8        1500 non-null   float64
 11  team9        1500 non-null   float64
 12  team10       1500 non-null   float64
 13  team11       1500 non-null   float64
 14  team12       1500 non-null   float64
 15  team13       1500 non-null   float64
dtypes: float64(13), int64(3)
memory usage: 187.6 KB
```



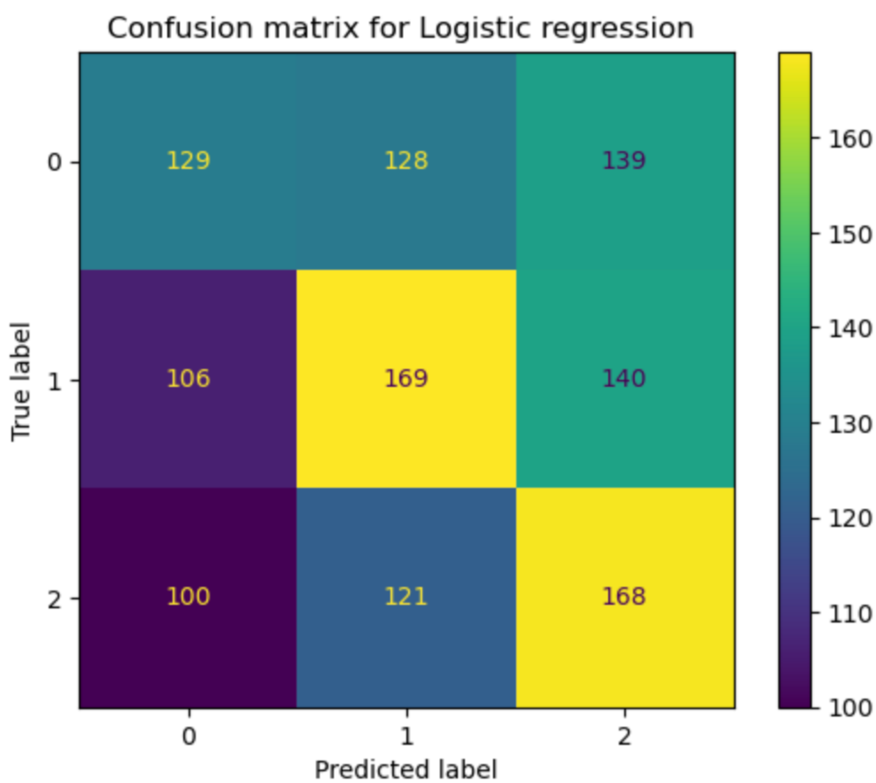
Classification Algorithms:

We have run logistic regression as our baseline learning algorithm to see if the data is linearly separ- able. The algorithm did not converge as the values are not scaled. Hence we used MinMaxScaler() to bring all the values to the same scale.

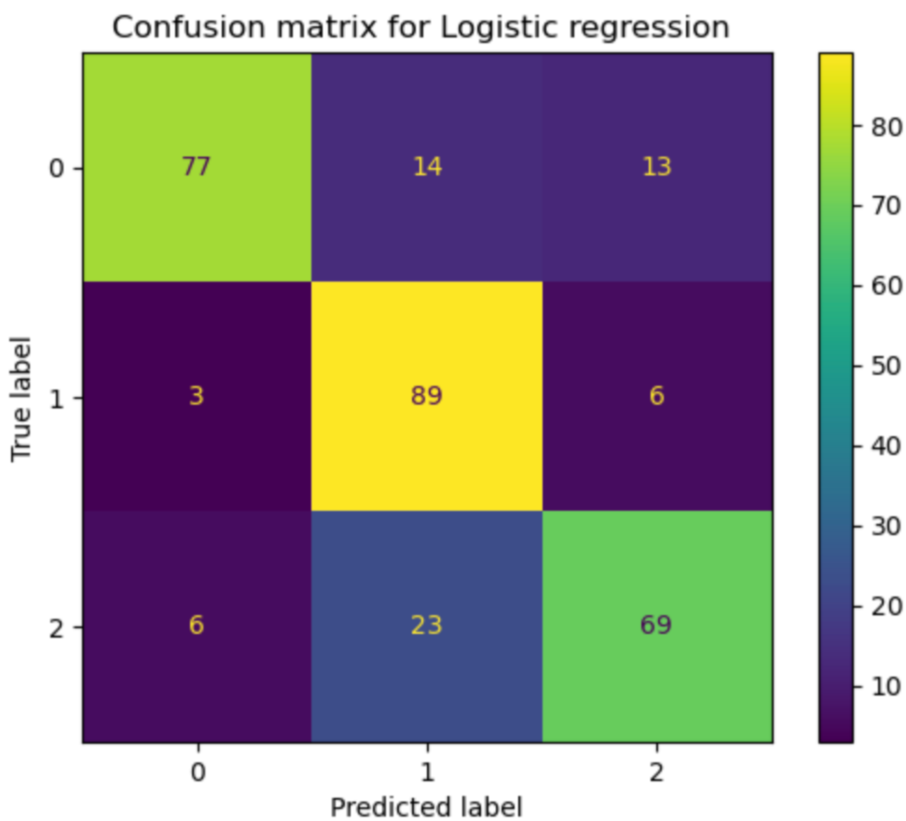
Logistic Regression:

Logistic regression is used as our baseline learner algorithm. On running algorithm on normalized data it converged.

```
------------LOG REG-------------------
0.39 accuracy for logistic regression
              precision    recall  f1-score   support

           0       0.39      0.33      0.35       396
           1       0.40      0.41      0.41       415
           2       0.38      0.43      0.40       389

    accuracy                           0.39      1200
   macro avg       0.39      0.39      0.39      1200
weighted avg       0.39      0.39      0.39      1200
```



```
------------LOG REG-------------------
0.78 accuracy for logistic regression
              precision    recall  f1-score   support

           0       0.90      0.74      0.81       104
           1       0.71      0.91      0.79        98
           2       0.78      0.70      0.74        98

    accuracy                           0.78       300
   macro avg       0.80      0.78      0.78       300
weighted avg       0.80      0.78      0.78       300
```



Decision Trees:

I have tried to tune the decision tree by varying depth from 1 to 25 using the criterion - 'gini'. Using 10-fold cross-validation, validation scores are plotted against depth values. From the figure, we can infer as the depth increases the validation set accuracy remained the same and training accuracy increased drastically. Since we wanted the decision tree not to overfit, we pruned at depth =7.

K-Nearest Neighbors:

I have tuned K nearest neighbors by varying k from 1 to 25. Using 10-fold cross-validation, validation scores are plotted against depth values. From figure 2, we can infer that the validation set accuracy remained the same for most k values. Therefore, k = 8 is chosen to avoid overfitting.

Random Forest:

I tried to exploit the bagging property of reducing the variance. Therefore, we chose max depth = 27 and tried to create an ensemble of random trees (number of estimators = 100).
The validation score obtained was better than the rest algorithms (validation set accuracy = 41%).

Bag of K-Nearest Neighbors:

Similarly, I tried to bag an overfitted K-nearest neighbor (k=3) to improve the validation, set accuracy.

Results:

Test set performance of Logistic Regression.

```
------------LOG REG-------------------
0.39 accuracy for logistic regression
              precision    recall  f1-score   support

           0       0.39      0.33      0.35       396
           1       0.40      0.41      0.41       415
           2       0.38      0.43      0.40       389

    accuracy                           0.39      1200
   macro avg       0.39      0.39      0.39      1200
weighted avg       0.39      0.39      0.39      1200
```

Test set performance of decision tree with d = 7.

```
------------ Dtree with depth=7 Performance-------------------
0.36 accuracy for dTree with depth =  7
              precision    recall  f1-score   support

           0       0.36      0.54      0.43       396
           1       0.36      0.29      0.32       415
           2       0.37      0.25      0.30       389

    accuracy                           0.36      1200
   macro avg       0.36      0.36      0.35      1200
weighted avg       0.36      0.36      0.35      1200
```

Test set performance of k-Nearest Neighbors with k = 8

------------KNN with k=8 Performance-------------------

0.39 accuracy KNN with k=8

	precision	recall	f1-score	support
0	0.38	0.45	0.41	396
1	0.43	0.40	0.41	415
2	0.37	0.33	0.35	389
accuracy			0.39	1200
macro avg	0.39	0.39	0.39	1200
weighted avg	0.39	0.39	0.39	1200

Test set performance of Random Forest with max depth = 27

------------ Random Forest with depth=27 Performance------------------

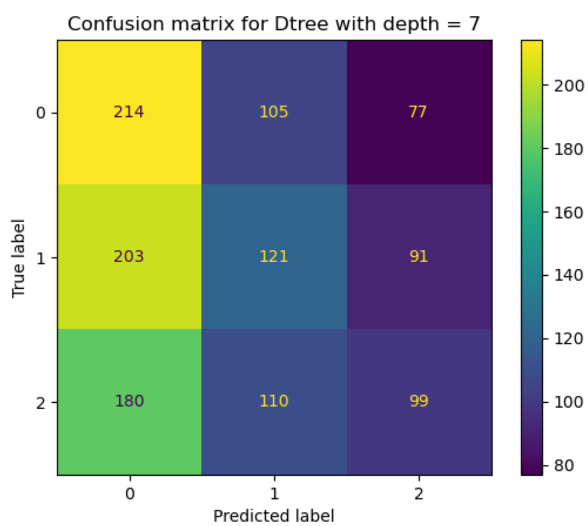
0.38 accuracy for RF with depth = 27

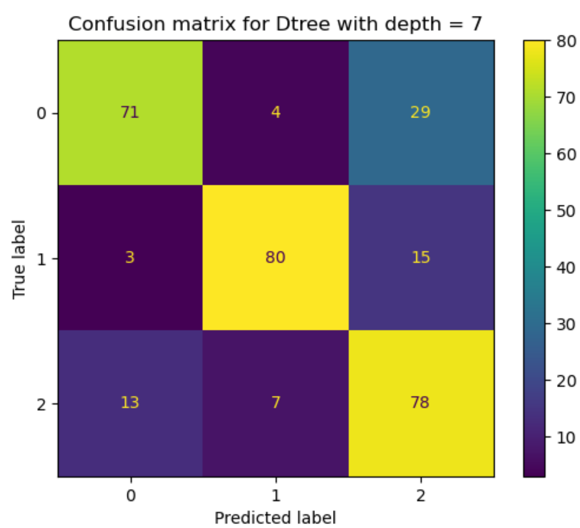
	precision	recall	f1-score	support
0	0.37	0.38	0.37	396
1	0.39	0.33	0.36	415
2	0.37	0.42	0.39	389
accuracy			0.38	1200
macro avg	0.38	0.38	0.38	1200
weighted avg	0.38	0.38	0.38	1200

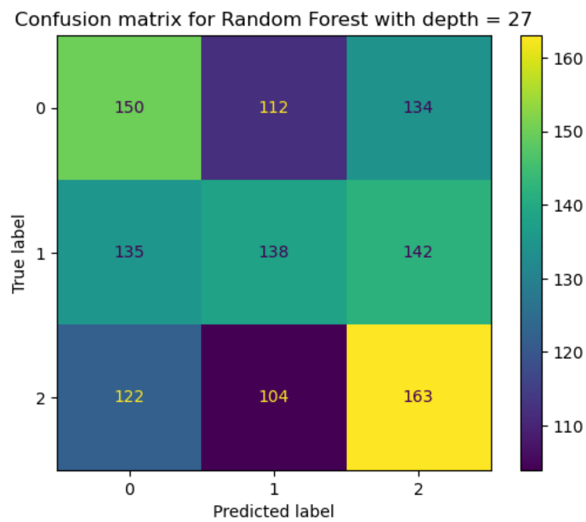
Test set performance of Bag of KNN with k = 3.

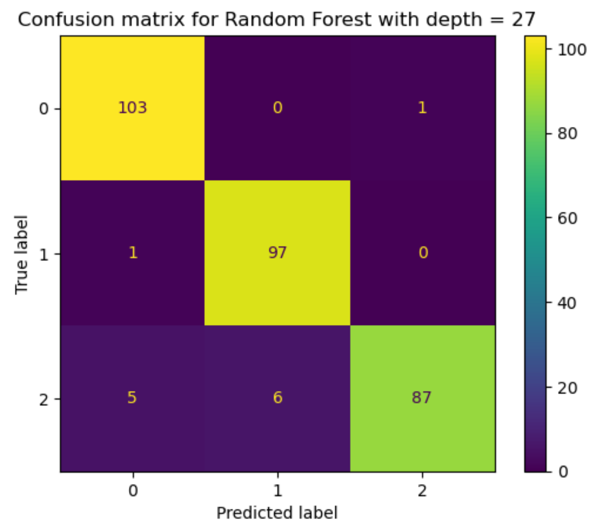
```
------------ Bag of KNN Forest with k=3 Performance------------------
0.39 accuracy for knn bag k =3
              precision    recall  f1-score   support

           0       0.40      0.39      0.39       396
           1       0.38      0.39      0.39       415
           2       0.38      0.38      0.38       389

    accuracy                           0.39      1200
   macro avg       0.39      0.39      0.39      1200
weighted avg       0.39      0.39      0.39      1200
```

Comparison and Analysis with Real Life Digit Dataset :









Synthesized data **Rea Life Data**

In the synthesized data set, the classification algorithms assign classes to the examples differently i.e for the random forest it is closer but for the decision tree it favors class 0.

In both the data sets, Random Forest gave better results when compared to other algorithms implemented. This indicated that an ensemble of decision trees worked better than a single decision tree.

For data set 2, we have run the Decision tree and Random Forest and tuned their depths. From the above figures (15,16,17,18), we can see that the number of wrongly classified are very low in real-life data then compared to synthesized data. Hence for the same learner, we get better accuracy.

Classifiers	**Random Forest**	**Decision Tree**
Dataset1	**0.39**	**0.36**
Dataset2	**0.96**	**0.76**

Conclusion:

The drastic change in accuracy from real-life data set to synthesized data set indicates that there is high inter-class similarity. This is true as the audio files generated from the magenta library have only piano notes. Hence the learner algorithms used for classification could not discriminate between classes from the extracted features.