Mini Project Report

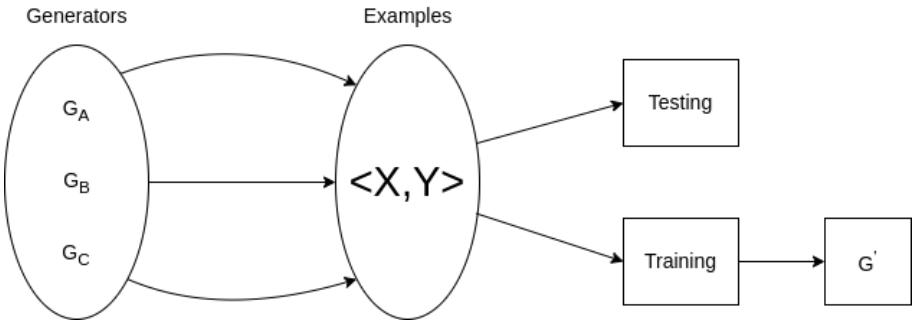
CS 6375 - Machine Learning

Classification of Music Files

Members:
Roshan Rayudu - RXR210122
Abhiram Ambatipudi - AXA210366

Table of Contents

1 Problem Statement



We have three generator functions from Magenta, TensorFlow library: AttentionRNN, BasicRNN, and LookbackRNN. Our task is to predict the generator of the audio file taken from the training examples.

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.

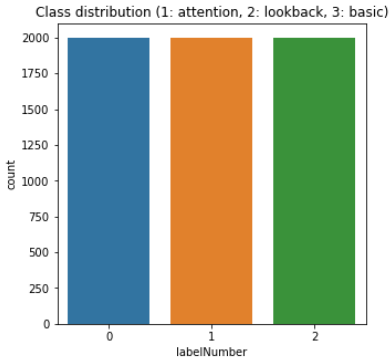
2.2 Using Librosa to extract MFCCs

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

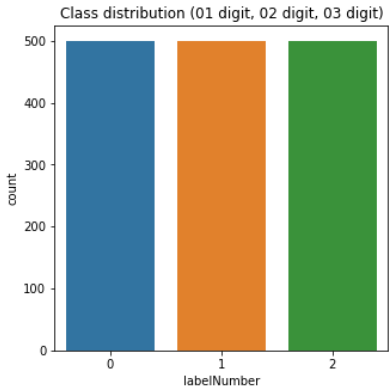
3.1 Synthesized Data

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).



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.



4 Classification Algorithms

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

4.1 Logistic Regression

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

4.2 Decision Trees

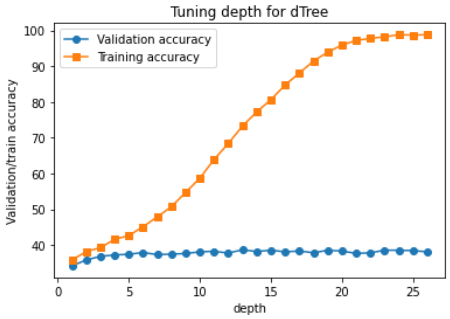


Figure 1: Tuning depth for Decision-Tree

We 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 Figure 1, 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.

4.3 K-Nearest Neighbors

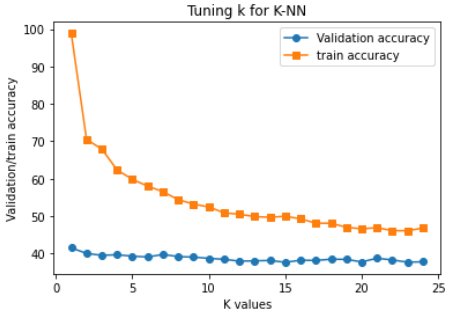


Figure 2: Tuning K for KNN

We 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 in order to avoid overfitting.

4.4 Random Forest

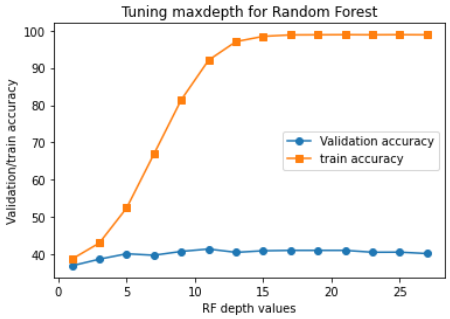


Figure 3: Tuning max depth for Random Forest

We 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%).

4.5 Bag of K-Nearest Neighbours

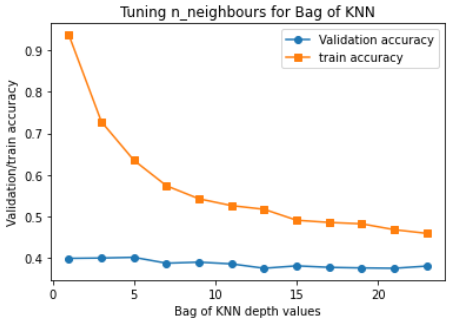


Figure 4: Tuning k for Bag of KNN

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

5 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
```

Figure 5: Classification report for Logistic Regression

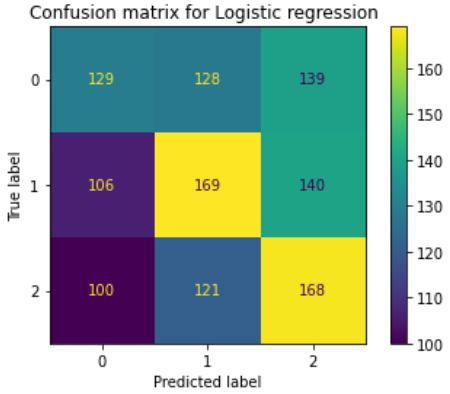


Figure 6: Confusion Matrix for Logistic Regression

- 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
```

Figure 7: Classification report for Decision Tree with depth = 7

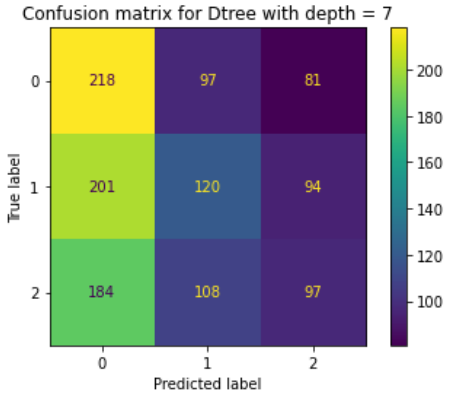


Figure 8: Confusion Matrix for Decision Tree with depth = 7

- 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
```

Figure 9: Classification report for KNN with k=8

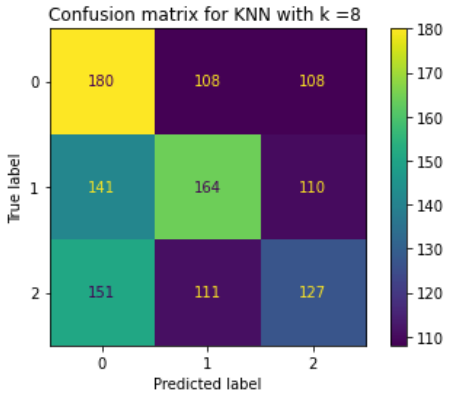
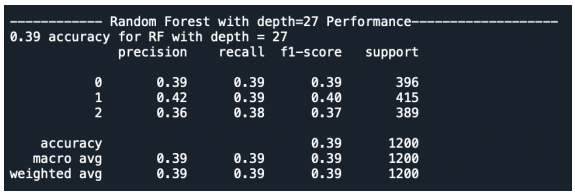


Figure 10: Confusion Matrix for k-Nearest Neighbors with k = 8

- Test set performance of Random Forest with max depth = 27.



---------- Random Forest with depth=27 Performance----------
0.39 accuracy for RF with depth = 27

	precision	recall	f1-score	support
0	0.39	0.39	0.39	396
1	0.42	0.39	0.40	415
2	0.36	0.38	0.37	389
accuracy			0.39	1200
macro avg	0.39	0.39	0.39	1200
weighted avg	0.39	0.39	0.39	1200

Figure 11: Classification report for Random Forest with max depth = 27

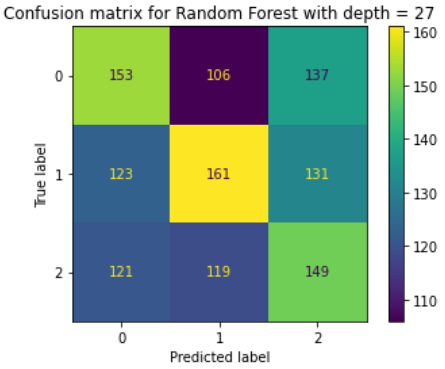
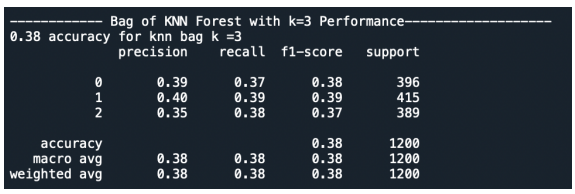


Figure 12: Confusion Matrix for Random Forest with maxdepth = 27

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



---------- Bag of KNN Forest with k=3 Performance----------
0.38 accuracy for knn bag k =3

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

Figure 13: Classification report for bag of KNN with k = 3

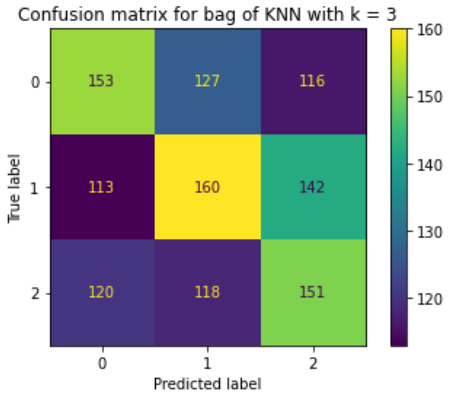


Figure 14: Confusion Matrix for bag of KNN with k = 3

6 Comparision and Analysis with Real Life Digit Dataset

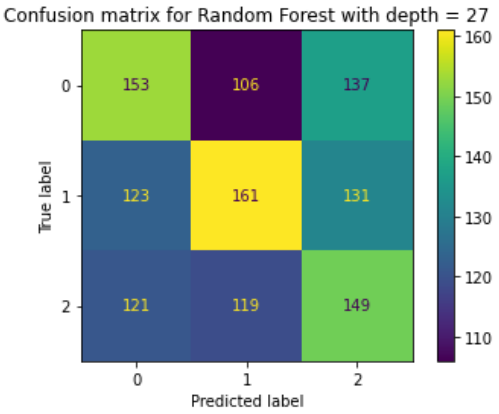


Figure 15: For Synthesized data

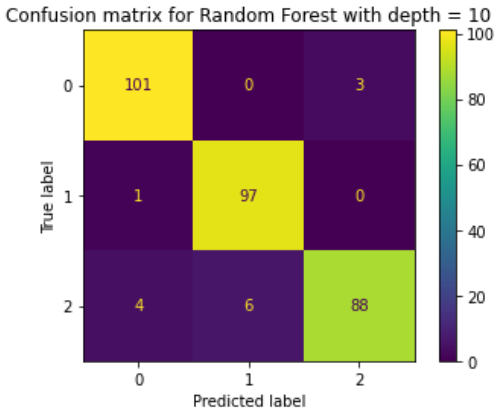


Figure 16: For Real Life data

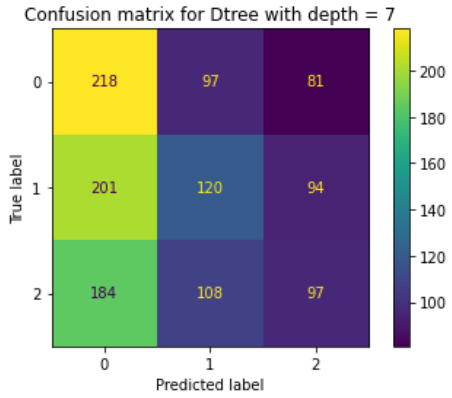


Figure 17: For Synthesized data

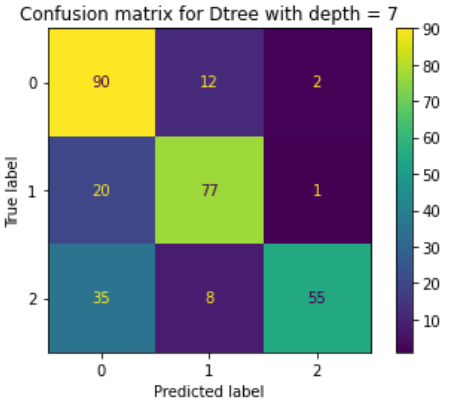


Figure 18: For Real Life data

- 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\Dataset	Random Forest	Decision Tree
Dataset 1	0.39	0.36
Dataset 2	0.96	0.76

7 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.

References

[1] Magenta Melody Generation, Github
https://github.com/magenta/magenta/tree/main/magenta/models/melody_rnn

[2] Audio MNIST Dataset
https://www.kaggle.com/datasets/sripaadsrinivasan/audio-mnist