Identification of Musical Instruments from Audio Files (CS419 Course Project Report)

Lyric Khare	Pranav Limaye	Shivam Ambokar	Sneha Kulkarni
20D170022	20D170028	200100145	200100090

May 2022

GitHub Repository Link: `https://github.com/lyrickhare/CS419_Ins_cla.git`

1 Introduction

Musical instrument classification (and audio classification in general) is a common yet challenging application of supervised multiclass machine learning algorithms. A very important step of pre-processing is feature extraction, which in itself requires sufficient knowledge about the mathematics and filetypes of audio files. The aim of this project is to do a comparative study of standard machine learning classifiers, such as Neural Networks, Logistic Regression, Linear Discriminant Analysis, and Decision Tree Classifiers. We will evaluate each algorithm's performance in classifying recordings of musical instruments into the correct instruments based on the model's accuracy.

2 Exploratory Data Analysis on the Dataset

For this project, we have employed the Google NSynth database containing short recordings (in the form of `.wav` files) of plenty of musical instruments being played at plenty of unique combinations of pitch, quality, and timbre. We first downloaded the NSynth's training set containing over 308k files (each 4 seconds long, and sampled at 16kHz). We picked the 5 instruments for which the most files were available: *Synthetic Bass, Electronic Keyboard, Electronic Organ, Acoustic Guitar, Acoustic String.* For each of these, we chose a sample of 3000 random files to be included in our project. **Therefore, *our* collection of audio files consisted of 3000x5=15000 files.** This would ensure that we generate a balanced training set from them. We were also able to use the `librosa` library to visualise the time-domain waveform of any of the `.wav` files, as seen in Figure 1. Here onwards, we moved on to actually extract some mathematical features from these 15000 `.wav` files.

2.1 Feature Extraction

In sound processing, the *mel-frequency cepstrum* (MFC) is a representation of the short-term power spectrum of a sound. We have used the `librosa` library to read each `.wav` file and extract the first 40 mean Mel-Frequency Cepstral Coefficients from them. These acted as our features. In this manner, our dataset has 40 features per entry. Observing that the later MFCCs decayed in magnitude, we realised they would be encoding lesser and lesser information from the audio file. We weighed the trade-off between execution time and model accuracy, and were able to decide that 40 is indeed a reasonable arbitrary choice for number of MFCCs extracted. Check Figure 2 for the dataframe of extracted features and assigned labels.

```
In [ ]: plt.figure()
        data, rate = librosa.load('audio/bass_acoustic_000-024-025.wav')
        librosa.display.waveshow(data, sr=rate)
        display(myAudio(data, rate))
```

Play

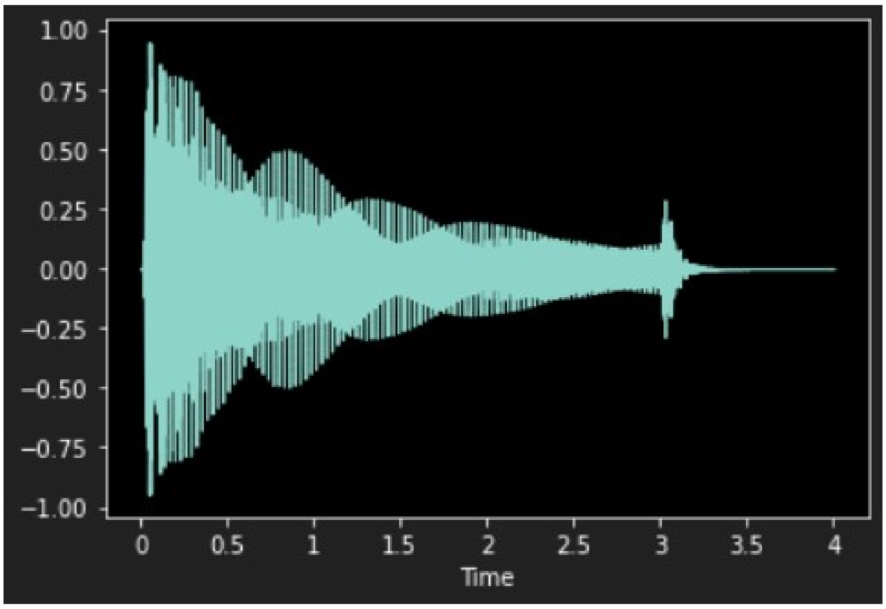
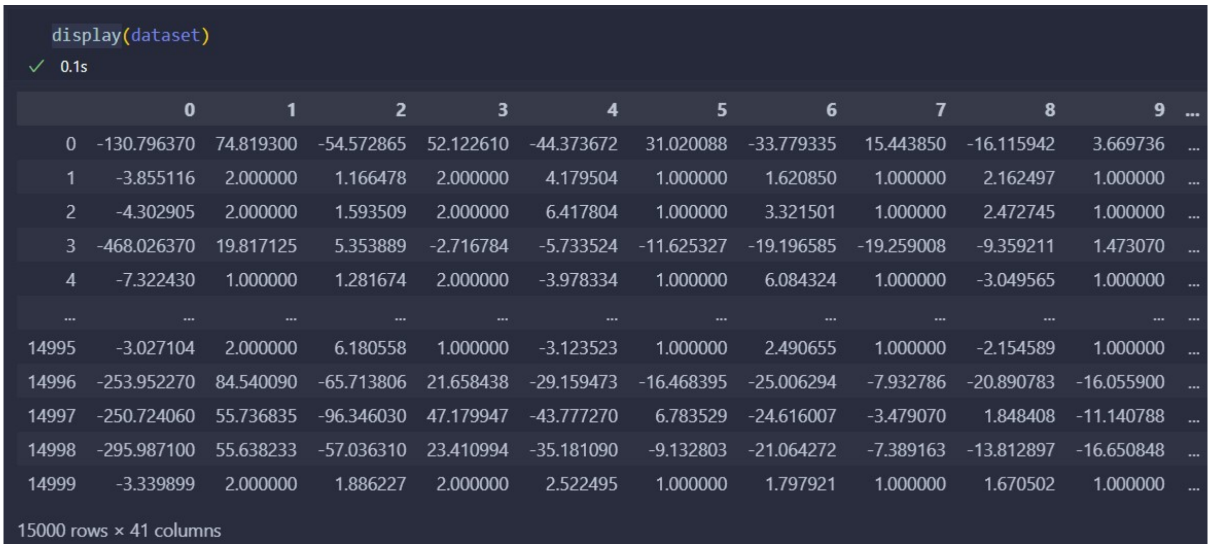


Figure 1: *Waveform of the audio input*



```
display(dataset)
✓ 0.1s
```

	0	1	2	3	4	5	6	7	8	9	...
0	-130.796370	74.819300	-54.572865	52.122610	-44.373672	31.020088	-33.779335	15.443850	-16.115942	3.669736	...
1	-3.855116	2.000000	1.166478	2.000000	4.179504	1.000000	1.620850	1.000000	2.162497	1.000000	...
2	-4.302905	2.000000	1.593509	2.000000	6.417804	1.000000	3.321501	1.000000	2.472745	1.000000	...
3	-468.026370	19.817125	5.353889	-2.716784	-5.733524	-11.625327	-19.196585	-19.259008	-9.359211	1.473070	...
4	-7.322430	1.000000	1.281674	2.000000	-3.978334	1.000000	6.084324	1.000000	-3.049565	1.000000	...
...	...	...	...	...	...	...	...	...	...	...	...
14995	-3.027104	2.000000	6.180558	1.000000	-3.123523	1.000000	2.490655	1.000000	-2.154589	1.000000	...
14996	-253.952270	84.540090	-65.713806	21.658438	-29.159473	-16.468395	-25.006294	-7.932786	-20.890783	-16.055900	...
14997	-250.724060	55.736835	-96.346030	47.179947	-43.777270	6.783529	-24.616007	-3.479070	1.848408	-11.140788	...
14998	-295.987100	55.638233	-57.036310	23.410994	-35.181090	-9.132803	-21.064272	-7.389163	-13.812897	-16.650848	...
14999	-3.339899	2.000000	1.886227	2.000000	2.522495	1.000000	1.797921	1.000000	1.670502	1.000000	...

15000 rows × 41 columns

Figure 2: *Final dataset*

2.2 Data Pre-Processing

This whole dataset was subjected to feature scaling (min-max scaling) in order to speed up convergence before proceeding further (except for Decision tree and Random Forest Classification algorithms). Thereafter, with a ratio of 67:33, the data was split into 'train' and 'test' sets. The number of instruments in the dataset were 5. Thus, the instruments were assigned certain numbers (from 0 to 4) by using a dictionary.

3 Neural Network

3.1 What is Neural Network?

Neural Networks comprise of layers of nodes which begin with an input layer, followed by one or more hidden layers and finally the output layer. A neuron is a building block of a neural network which performs two tasks, linear combination of the inputs followed by activation.

3.2 Maths behind Neural Networks

Suppose a neuron has input x, weight matrix as w and bias b, then z = w.x + b. If Relu(x) is the activation function used, then Relu(z) = max(0, z) is the output for that particular neuron. Generally, for multi-class classification probelms, Relu activation is used for hidden layers and 'softmax'(which calculates the relative probabilities) is used for the final layer. The formula for softmax activation is:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

where z represents the value from the neurons in the output layer. The loss function used in multi-class classification problems is 'Categorical Cross Entropy Loss' which is given by:

$$L(y_i, \hat{y}_i) = -\sum_i y_i \log(\hat{y}_i)$$

where $\hat{y}_i$ represents the i-th scalar value in the model output and y_i represents the corresponding target value. In neural networks, the weights are updated by back propagation in which, chain rule is implemented to calculate the gradient of loss function with respect to a particular weight value. The gradient descent algorithm for the same is given by:

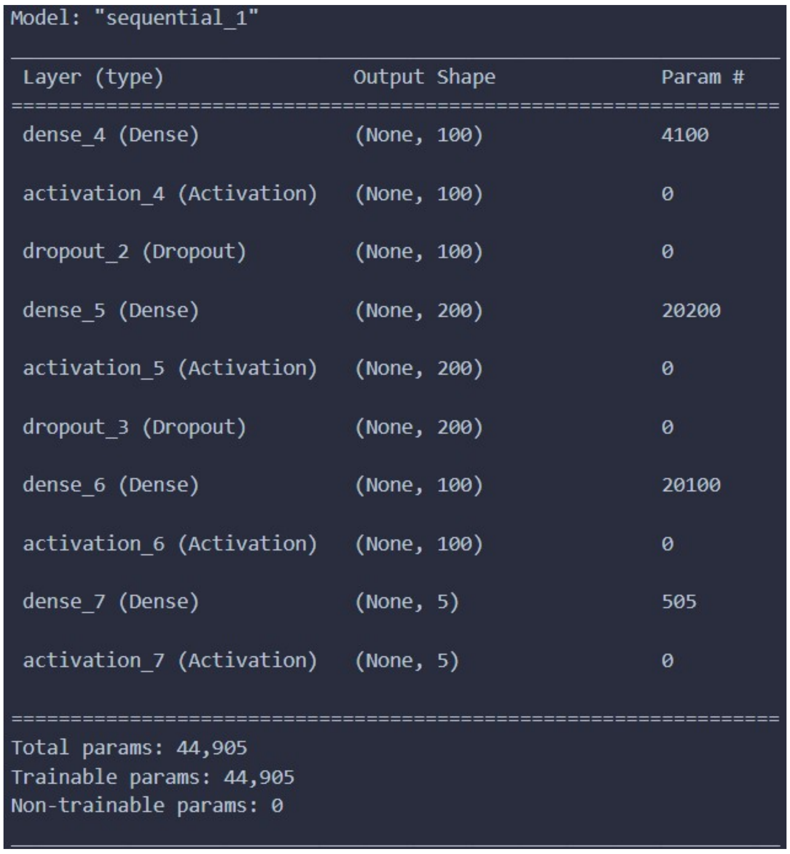
$$\text{w}_i = w_i - \alpha\frac{\partial L}{\partial w_i}$$

$$\text{b} = b - \alpha\frac{\partial L}{\partial b}$$

where α is the learning rate.

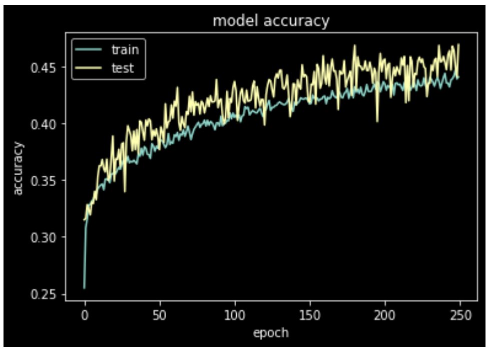
3.3 The model used

In this classification problem, we have 40 features for each datapoint. Thus the input layer consists of 40 neurons. The output layer consists of 5 neurons since the target variable can take 5 discrete values. The architecture of the neural network is shown in the figure.

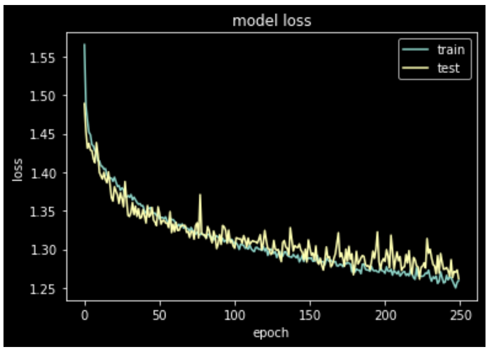


```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 100)               4100

 activation_4 (Activation)   (None, 100)               0

 dropout_2 (Dropout)         (None, 100)               0

 dense_5 (Dense)             (None, 200)               20200

 activation_5 (Activation)   (None, 200)               0

 dropout_3 (Dropout)         (None, 200)               0

 dense_6 (Dense)             (None, 100)               20100

 activation_6 (Activation)   (None, 100)               0

 dense_7 (Dense)             (None, 5)                 505

 activation_7 (Activation)   (None, 5)                 0

=================================================================
Total params: 44,905
Trainable params: 44,905
Non-trainable params: 0
_________________________________________________________________
```

Figure 3: *Architecture of the Neural Network*



(a) Accuracy of the NN Model



(b) Loss of the NN Model

Figure 4: *Results of the Neural Network*

4 Logistic Regression

4.1 What is Logistic Regression?

A statistical model typically used to model a binary dependent variable with the help of logistic function. Another name for the logistic function is a sigmoid function and is given by:

$$F(x) = \frac{1}{1+e^{-x}}$$

4.2 Maths behind Logistic Regression

For a binary classification problem, let x be the independent variable and y be the dependent variable and $y \in 0, 1$ and predicted value is $h_\theta(x)$; then the cost function is given by:

$$\text{Cost}(h_\theta(x), y) = -log(h_\theta(x)) \text{ if } y = 1 \text{ and } -log(1 - h_\theta(x)) \text{ if } y = 1 \tag{1}$$

$$J(\theta) = \frac{1}{m} \sum_{1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)}) \tag{2}$$

After this you can easily use gradient descent method using $J(\theta)$ as loss function.

4.3 Multi-class Classification

Now if we want to do multi-class classification (which is there in our data-set), we can use one v/s all method which says: Train a logistic regression classifier h_θ for each class i to predict the probability that $y = i$. On a new input x, to make a prediction, pick the class i that maximizes the following:

$$\max_{(i)} h_\theta^{(i)}(x)$$

5 Linear Discriminant Analysis

5.1 What is Linear Discriminant Analysis?

Linear Discriminant Analysis or Normal Discriminant Analysis or Discriminant Function Analysis is a dimensionality reduction technique that is commonly used for supervised classification problems. It projects an n-dimensional feature space onto one with fewer dimensions, so as to maximise the separability between different labelled classes. It is closely linked with Principal Component Analysis (PCA).

5.2 Maths behind Multi-class Linear Discriminant Analysis

Suppose we have n samples (as $(x_i,\, y_i)$), C classes (each k^{th} class having n_k samples and mean μ_k), and μ is the overall sample mean. Then, the within-class scatter matrix is:

$$S_w = \sum_{i=1}^{n} (x_i - \mu_{y_i})(x_i - \mu_{y_i})^T$$

Similarly, the between-class scatter matrix is:

$$S_b = \sum_{k=1}^{C} n_k(\mu_k - \mu)(\mu_k - \mu)^T$$

Then, multi-class LDA can be formulated as an optimization problem to find a set of linear combinations (with coefficients w) that maximizes the ratio of the between-class scattering to the within-class scattering, as:

$$\hat{w} = \underset{w}{\operatorname{argmax}} \frac{w^T S_b w}{w^T S_w w}$$

Thus, the solution is given by this generalised eigenvalue problem:

$$S_b w = \lambda S_w w$$

This generates at most $(C - 1)$ useful eigenvectors, along which class separability is maximised.

6 Decision Tree Classification

6.1 What is Decision Tree Classification?

Decision tree algorithm is one of the most popular machine learning algorithm. It is a supervised machine learning algorithm, used for both classification and regression task. It is a model that uses set of rules to classify something.

6.2 Maths behind Decision Tree Classification

The maths in decision trees occurs in the learning process. We initially start with a dataset $D = (X, y)$ from which we need to find a tree structure and decision rules at each node. Each node will split out dataset into two or more disjoint subsets $D^*_{l,i}$, where l is the layer number and i denotes each individual subset. If all our labels in this subset are of the same class, the subset is said to be pure and this node will be declared a leaf node and this section of the tree has reached termination. If this is not the case, the splitting criteria will continue.
As long as the error on the validation set continues to fall, we continue to split our subsets. When the error stops falling, we stop the generation of new levels to our decision tree and this is our final model.
We have employed this algorithm at maximum depths ranging from 20 to 300 splits and the performance did not vary very significantly due to this change.

7 K-Nearest Neighbours

7.1 What is K-Nearest Neighbours Classification?

K-Nearest Neighbours classification is among the most basic classification algorithms in machine learning. It is a supervised learning algorithm and is non-parametric in nature (no underlying assumptions regarding the distribution of data).

7.2 The Algorithm

For a point `p` in consideration, we find the 'distances' of this point with respect to all the points. This distance can be chosen based on the type of variables in the dataset.Hereafter, make a set `S` of `K` smallest distances obtained and return the majority label among **S**. The accuracy of this algorithm increases with an increasing number of elements and also with feature scaling. However, there are some drawbacks of this algorithm, such as it getting slower for large datasets and in case of skewed distributions.

8 Random Forest Classification

8.1 What is Random Forest Classification?

Random Forest classification is a supervised nonlinear classification algorithm and is an ensemble learning (multiple learning algorithms) method. A collection of decision trees is used, which specify the categories with way higher probability. It yields higher accuracy than decision trees alone.

8.2 The Algorithm

Subsets of the original data are made by virtue of row and feature sampling. This way, subsets of the training data are created. For each subset, individual decision trees are created and the outputs of each of the decision tree is noted. The final output is considered based on 'Majority voting' for the classification problem.
This algorithm can also be used in feature selection as well After applying this algorithm to our dataset, the accuracy was found to be around 0.721, with an *out-of-bag* score of **0.69** for a **67:33** test-validation split

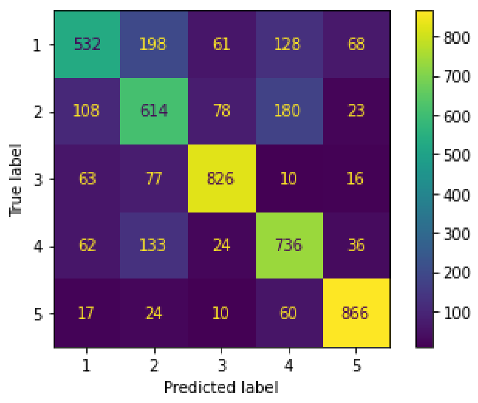


Figure 5: *Confusion Matrix for Random Forest Classifier*

9 Performance Comparison and Results

The metric used for evaluating all the models is Accuracy which gives the ratio of total number of correct predictions to the total number of datapoints for which predictions have been made.

#	Model	Training Accuracy	Validation Accuracy
1	Random Forest	1.00	0.72
2	Decision Tree Classifier	1.00	0.56
3	KNeighbours Classifier	0.68	0.52
4	Neural Network	0.44	0.47
5	Linear Discriminant Analysis	0.38	0.37
6	Logistic Regression	0.39	0.38

Thus, from the above table, we conclude that Random Forest Classifier gives comparatively the best results. On the other hand, Logistic Regression and Linear Discriminant Analysis are seen to be poorly performing on the data. The graph for the accuracy in the case of neural network is shown below.

10 Predicting Instruments in Overlay .wav files

10.1 Overlay Data-set

The Google NSynth data-set has `.wav` files of 4 sec duration having a mono instrument sound playing in it, but we tried to make a new test data in which the `.wav` files were generated by overlaying two instruments using `pydub` library.

10.2 Algorithm

We used the most basic classification algorithm, 'logistic regression' to find the instruments being played in the data set. We trained our data on 3000 mono instrument .wav file samples having 1000 samples of 'flute:synthetic', 'bass:electronic' and 'vocal:acoustic' each.
For generating overlay files we used 50 files from each of 'flute:synthetic', 'bass:electronic' and 'vocal:acoustic'. So total 150 overlay files were generated.
Now we predicted probability of each instrument using `model.predict_proba(test)` and checked for two highest probabilities to find the 2 instruments being played in that file.
We achieved an accuracy of 66.6% using this method.

References, Appendix, and Acknowledgements

Database Used for Audio Files

NSynth, a part of Project Magenta (built upon Google's TensorFlow)
Publicly available here: `https://magenta.tensorflow.org/datasets/nsynth`

Python Libraries Used

`numpy`, `scipy`, `pandas`, `matplotlib`, `sklearn`, `tensorflow.keras`, `librosa`, `IPython`, `re`, `os`

References

1. `https://github.com/krishnaik06/Audio-Classification`
2. `https://stackoverflow.com/questions/4289331/how-to-extract-numbers-from-a-string-in-python`
3. `https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/`
4. `https://towardsdatascience.com/introduction-to-math-behind-neural-networks-e8b60dbbdeba`
5. `https://medium.com/analytics-vidhya/the-math-behind-logistic-regression-c2f04ca27bca`
6. `https://www.coursera.org/learn/machine-learning`
7. `https://en.wikipedia.org/wiki/Linear_discriminant_analysis`
8. `https://multivariatestatsjl.readthedocs.io/en/latest/mclda.html`
9. `https://towardsdatascience.com/decision-trees-the-maths-the-theory-the-benefits-6315abd2f10a`
10. `https://www.geeksforgeeks.org/k-nearest-neighbours/`

11. https://towardsdatascience.com/decision-trees-the-maths-the-theory-the-benefits-6315abd2f10a

Division of Work

Lyric	Testing on multi-instrument dataset, Logistic Regression, Compiling the work
Pranav	Exploratory Data Analysis, Feature Extraction, Linear Discriminant Analysis
Shivam	Feature Extraction and Processing, Random Forests Classifier, K-Nearest Neighbours
Sneha	Data Pre-Processing, Decision Tree Classifier, Neural Networks