The first thing I did with the data was to read the training labels from the training dataset. And then I saved it as the table with the individual path of each audio and its label as Table 1. below.

	audio	Label
0	/Competition/train/train_new/train_0.wav	21
1	/Competition/train/train_new/train_1.wav	32
2	/Competition/train/train_new/train_2.wav	31
3	/Competition/train/train_new/train_3.wav	31
4	/Competition/train/train_new/train_4.wav	41

Table 1. Audio paths with its labels.

After I had the path and label table, I examined the single audio path and extracted the audio data into an array form, with the sampling rate of 8000 as a first peek of the audio data analysis.

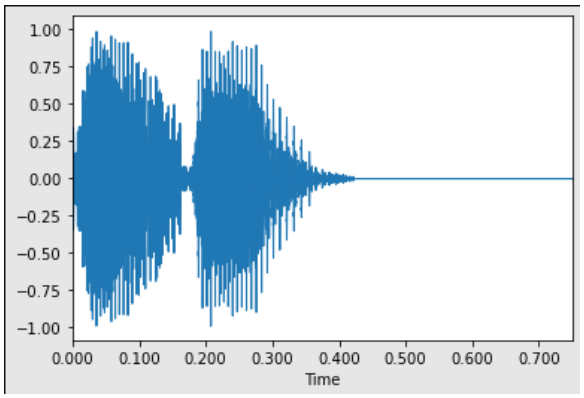

Figure 1. Sample audio data extraction.

The audio is in wav format, which I coud extract the features of an audio with Fast Fourier Transformation (FFT). First, we can transform it from time domain into frequency domain. Then we can transform it into a spectrogram as the training features for each pixel. I used the librosa library in Python and implement its feature extraction methods. The first feature I extracted is the MFCC, other features that I extracted are the Mel spectrogram and chroma_stft. Mel spectrogram is the spectrogram of the Mel ceptrum, and chroma_stft feature is the function that computes the chromagram from a waveform for audio hashing.

After the preprocessing and feature extraction, we can start the training process. In this stage, I tested two kinds of approaches. But before I start training, I had to separate the dataset into training set and testing set. I perform an 8:2 train-test split from the dataset and encoded the labels into distinct numbers. For the later training

part, the first approach I used is the neural network. I implemented from the Tenserflow library with two hidden layers, and the activation function I used is relu. For the output layer, I used softmax as the loss function. I had tried a few attempts of choosing the iterations and feature extractions for the best testing accuracy. The final iterations I chose was 1500.

For my second approach, I used MFCC features to use a two-dimension Convolutional Neural Network (CNN). I set three hidden layers with different filters but remain the same activation function, and the output activation function was also softmax as in my first approach.

For the test result, I had my best score of the first attempt was around 0.904, and with the CNN, I had a better performance with the accuracy approximately 0.9057.

After that, I started using different methods that I have learned in the class to see its performance. The first thing I tried is the Naïve Bayes.

```
Number of mislabeled points out of a total 36000 points : 11218
Accuracy = 0.688389
```

Figure 2. Mislabeled points and accuracy using Naïve Bayes

Later on, I used logistic regression with regularization and the result came out a better score compared to the Naïve Bayes, which is around 88% accuracy.

```
The training accuracy is: 0.8865555555555555.
The testing accuracy is: 0.8834444444444445.
```

Figure 3. Accuracy using logistic regression.

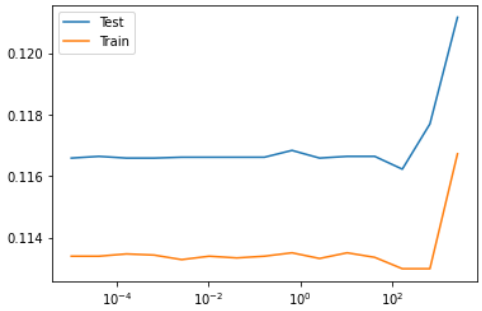

Figure 4. Loss from logistic regression.

The next classifier I tested was the decision tree. I chose information gain as my classification criterion.

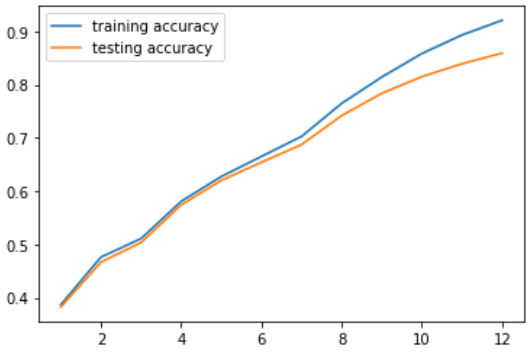

Figure 5. Accuracy using decision tree.

The last classifier I tested was the support vector machine (SVM). SVM performed with the testing accuracy 93%.

```
The maximum testing accuracy achieved with Linear SVM is: 0.9301944444444444
```

Figure 6. Testing accuracy using linear SVM.

I find that I can still improve the accuracy by separating the labels from the dataset since I have found out the combinations of the label in the training set are only five, but the outcome labels should be six. The missing one label be the obstruction of the accuracy improvement. For future improvement, I may preprocess the single label into multiple labels then train. In conclusion, I really learned a lot from this competition.

```
y_trn['Label'].value_counts()
```

```
42    18000
41    18000
32    18000
31    18000
21    18000
Name: Label, dtype: int64
```

Figure 7. Only five labels in the training dataset.