Identifying Emotions through Human Speech

Dharmik Shah
MSc in Computer Science Student
Ryerson University
Toronto, Canada
dharmik.shah@ryerson.ca

***Abstract*—Speech and language are abilities that seem trivial to humans, as they have no recollection of learning them in the first place. The same task for a machine however, can prove to be difficult, as many variables must be considered: pitch, loudness, emotion, etc. This paper will consider one of these variables, particularly emotion, and implement a classification model based on a wide variety of speech samples. Using the RAVDESS dataset, the model implemented will attempt to rank a specific speech sample based on the emotion it feels best represented: neutral, calm, happy, sad, angry, fearful, disgust, or surprised.**

***Index Terms*—speech recognition, emotion detection, machine learning, deep learning, mel-frequency cepstral coefficients**

I. Introduction and Motivation

Speech and language are fundamental human skills that are learned during the early stages of childhood. Since it is developed at an earlier time, when two people converse, it is easy for both of them to understand what the other one is saying. The process of taking the information that one person said, understanding it, and replying, is often instantaneous. It does not feel like there is anything to learn from human speech, as the brain processes it at an alarming rate. However, this simply isn't the case, and there is actually much to learn. Although the words spoken are undoubtedly the most important feature when two people converse, they are not the only features that our brain understands. For example, the brain also processes both people's pitch, loudness, and emotions to decipher the true meaning of the speech. For instance, the sentence "you forgot to buy food," said while exhibiting angry emotions, can give rise to very different responses than if said calmly. In this paper, we will explore the classification of speech with regards to emotion. More specifically, it will attempt to classify a speech sample as falling into one of these emotion categories: neutral, calm, happy, sad, angry, fearful, disgust, or surprised, neutral, with a high accuracy rate on the test set.

II. Method

A. Dataset and Preprocessing

For this project, the *RAVDESS* dataset was used for speech processing. It is available freely for download [1]. The RAVDESS dataset contains video and audio recordings of 24 professional actors. These actors, divided equally between male and female, are recorded enunciating two specific sentences. The two sentences are *"Kids are talking by the door"*, and *"Dogs are sitting by the door"*. The actors vocalize each sentence (kids, dogs), with all emotions (neutral, calm, happy, sad, angry, fearful, disgust, surprised), and with both intensities (normal, strong). The RAVDESS dataset also offers video files where the actors are filmed while enunciating all sentences.

The preprocessing steps were as follows:

1) Download the 1440 files where actors are enunciating normally
2) Download the 1012 files where actors are singing, and combine it with the normal dataset
3) Download all videos (normal and singing), and convert them to audio using the FormatFactory software. Then, combine it with the above with a proper folder structure

Overall, we have 4904 audio files that are distributed evenly per class. The audio files were all represented using a specific naming scheme, which is useful for understanding what the file entails. Consider the following audio sample:

03-01-04-02-01-02-19.wav

We can separate the filename into 7 unique properties, split based on the dash. They are as follows:

1) Modality (03) $\rightarrow$ 01 - full audio/video, 03 - audio only
2) Vocal Channel (01) $\rightarrow$ 01 - speech, 02 - song
3) Emotion (04)
   a) 01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised
4) Emotional Intensity (02) $\rightarrow$ 01 - normal, 02 - strong
5) Sentence (01)
   a) 01 - "Kids are talking by the door"
   b) 02 - "Dogs are sitting by the door"
6) Repetition (02) $\rightarrow$ 01 - 1st trial, 02 - 2nd trial
7) Actor (19) $\rightarrow$ 01 - 24

Thus, we can see that the following file is an audio based speech file, where the 19th actor is reading the kids sentence with a sad, strong emotion. They are also on their second trial.

B. Extracting Features from the Sound

In the area of signal and sound processing, there is a recommended method to generate features for a specific waveform, and we have followed that approach. What we aimed to find was the MFCCs, or Mel-frequency cepstrum coefficients for each sample. These are our features.

In speech analysis, one of the most important libraries is *librosa*, and we used this exact library to extract the features of each of the sound samples.

However, given that we were able to get the features with just one function call, we wanted to explore more deeper into exactly was happening internally. Thus, we first experimented by going through the full procedure of finding the MFCCs, and only then did we use the function call.

When we read a music file into our program, it is represented as an array of amplitude points. This is our starting point.

We firstly converted the sound waveform into a frequency waveform by performing a Fast Fourier Transform [2]. The transformation helped us understand the frequency distribution in a specific emotion. However, this transformation displays frequency distribution as a whole, and thus, we do not have frequency information for a specific time interval. In order to retain time information while still having frequency information, we instead performed a *Short-Time* Fourier Transform. From this, we then computed what is known as the Mel spectrum, which is a plotting of Time, Frequency, and Intensity on the Mel Scale (what humans perceive). The final step was to compute another transformation on top of the Mel spectrum to retrieve the MFCCs, or the Mel-frequency cepstrum coefficients [3]. Although we could have stopped at the Mel spectrum, the MFCCs are an industry leader and are used in many real-world speech algorithms. The MFCCs represent amplitude information about the sound wave in a vector format, which were then be fed into the machine learning model. While experimenting with different amount of MFCC coefficients, we decided to stick with 40. Typically, speech projects might take as little as 12-13 coefficients or as much as 38-40 [2].

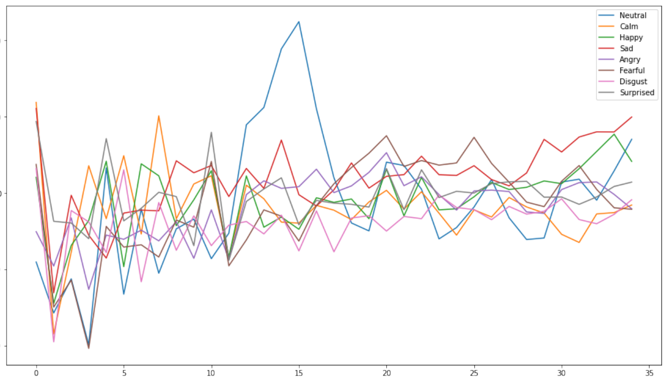


Fig. 1. 35 MFCC coefficients for each of the 8 classes

C. *Training and Testing Split*

As we have a total of 4904 samples, we can afford to split the data more generously, utilizing a 70-30 train-test split function. This means that around 3400 samples will be used on training, while the rest will be used in validation.

III. Machine Learning Techniques

The project trained the sound samples on a variety of machine learning algorithms, and the results of each one will be presented in the next section.

In this section, we go through our chosen models, and justification for why we have chosen them.

1) Decision Tree – Often one of the simplest algorithms to use for classification and regression problems, decision trees perform well on numeric attributes, and non-linear data. It works by constructing a tree-like structure which rules, which dictate how an example is to be classified. Decision Tree's are very vulnerable to overfitting, and so this was not our best choice [4]
2) KNN (K Nearest Neighbours) – Suitable for when the data consists of more than two categories, the KNN is a suitable algorithm for classification problems. It works by classifying new data points based on how it's neighbors are classified. KNN is often slow with large sets of data like ours, and if thus computationally expensive [5]
3) Random Forest – Preferred for when we have a large dataset, the random forest is an advanced version of the decision tree algorithm, and often performs much better. It works by constructing a series of decision trees, similiar to how a forest houses many trees. The random forest, similar to KNN requires quite a bit of computational power as it builds numerous trees and combines their individual outputs [6]
4) Neural Network – The recommended model of choice in many classification and regression problems, the neural network is able to analyze complex relationships between the input features and make very highly accurate predictions. It works by learning more and more each layer, and outputs the prediction in terms of neuron activation. One disadvantage is that neural networks may need to have multiple layers and activation's in order to learn the input data well, and thus require many rounds of trial and error [7]

With all this being said, we will now showcase the results of training and the relevant evaluation metrics we used to understand which model performed the best.

IV. Results

A. *Evaluation Metrics*

The most popular evaluation metrics are Accuracy, Precision, Recall and F1 Score. We often use precision and recall for binary classification, although they can still be extended to multi-class [8]. In our scenario, we only cared about correctly classifying a speech sample amongst the eight classes. Accuracy is the metric that does this, and is given by:

$$\frac{TP + TN}{(TP + FP) + (TN + FN)}$$

We will use accuracy as our metric of evaluating which model performed the best. In each of the learning algorithms, we will first illustrate what the model looks like internally. We will then output it's confusion matrix, calculate the accuracy, and say a few words on the results. After all algorithms are covered, we will summarize the results and determine which was our best algorithm based on the accuracy metric.

B. *Decision Tree*

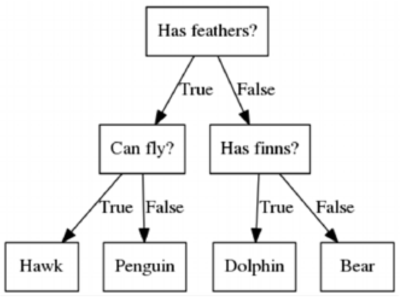
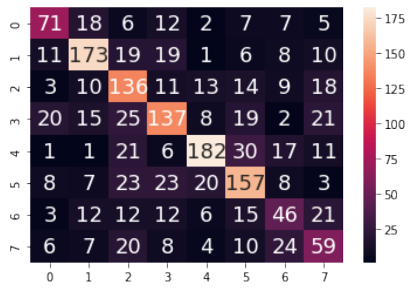


Fig. 2. Simple decision tree that classifies 4 types of animals

The decision tree was our first algorithm of choice. We did not expect very good results from it since prone to overfitting. We come up with the following confusion matrix.



	0	1	2	3	4	5	6	7
0	71	18	6	12	2	7	7	5
1	11	173	19	19	1	6	8	10
2	3	10	136	11	13	14	9	18
3	20	15	25	137	8	19	2	21
4	1	1	21	6	182	30	17	11
5	8	7	23	23	20	157	8	3
6	3	12	12	12	6	15	46	21
7	6	7	20	8	4	10	24	59

Fig. 3. Decision Tree confusion matrix (test set)

The overall accuracy of the model is 59%, and on a per-class basis, it is [0.55%, 0.7%, 0.64%, 0.55%, 0.68%, 0.63%, 0.36%, 0.43%]. We can see that the accuracy is pretty low overall. Even though the last two classes do not have as many testing examples as the rest, from the accuracy alone, we can see that they are classified the poorest. We expect this to be a baseline, and the algorithms to only improve from here on out.

C. *KNN*

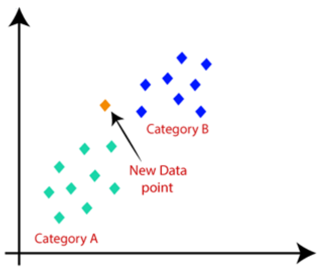


Fig. 4. KNN classifies the new data point based on the closest neighbor classes

The KNN algorithm was our second choice, and the challenge here was picking an appropriate K value. If we chose K to be too small, then our algorithm can very easily overfit. This is because of potential outliers in the data, that might be close to our test example, but overall is not the correct classification. If we choose K as a very large number, we may underfit, because we are considering too many neighbors to decide a class, and thus the model has not really learned well. We need to choose K relative to the amount of data points we have. We have found that with over 3000 samples in the training set alone, choosing K = 5 realized the best accuracy. Below is a graph comparing many K values and their relative accuracy scores.

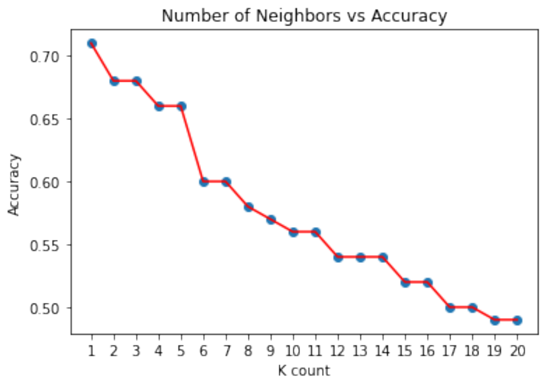
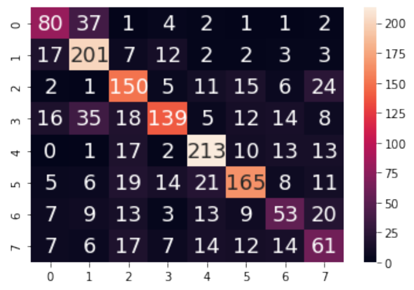


Fig. 5. Plot of possible K values and their respective accuracy

Above we see that K = 1 has the best accuracy, but as we mentioned before, this is wildly overfitting, and thus not the best choice. From K = 2 - 4, we notice a steady accuracy, and at K = 5, there is a much larger drop. This means that the range between K = 4 and K = 5 are quite important, and so we select K = 5 as our K value.

We can see the accompanying confusion matrix based on this model below.



	0	1	2	3	4	5	6	7
0	80	37	1	4	2	1	1	2
1	17	201	7	12	2	2	3	3
2	2	1	150	5	11	15	6	24
3	16	35	18	139	5	12	14	8
4	0	1	17	2	213	10	13	13
5	5	6	19	14	21	165	8	11
6	7	9	13	3	13	9	53	20
7	7	6	17	7	14	12	14	61

Fig. 6. KNN (K=5) confusion matrix (test set)

If we compare the K nearest neighbors confusion matrix with what we saw previously in the decision tree, we immediately notice that we have more accurate classifications along the diagonal. Every single one of the classes is classified more accurately now, and the last two classes have improved slightly as well.

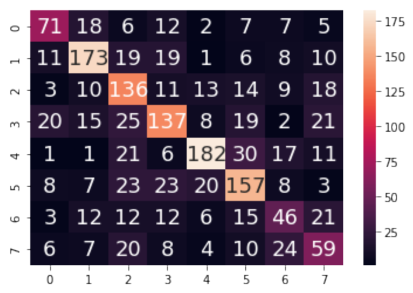


Fig. 7. Decision Tree confusion matrix (test set)

Overall, we have a 66% accuracy compared to 59% in the decision tree, and on a per-class basis, it is [0.62%, 0.81%, 0.7%, 0.56%, 0.79%, 0.66%, 0.42%, 0.44%]. We see that the 7th class (disgust) is still pretty behind on accuracy, but this will hopefully become better in our next algorithm.

D. Random Forest

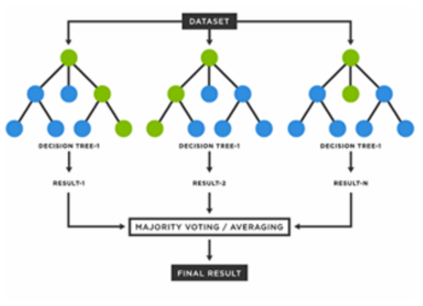


Fig. 8. Random forest with 3 decision trees

The random forest algorithm was our third choice, and the hope was that it is a much better version of the decision tree. Similarly to KNN, we needed to provide a hyperparameter here, more specifically, the number of trees that we want in the forest. The more trees you use, the more memory usage the algorithm will require. Thus, the challenge here to figure out a good number of trees that has a high accuracy and minimizes the training time.

We take a similar approach as the last section, and create a plot of different tree values and compare the accuracy. This time however, we also note the time taken to complete each algorithm, to determine what the best tree value is for our case.

We see above that the accuracy peaks around 100 trees in the forest, and seems to plateau after that, only decreasing a single percentage after that. We also note the time increase each time we choose a higher tree value. Based on these two factors, the best choice is 100 trees in our forest.

Below, we show the confusion matrix associated with the random forest.

The respective accuracy values for each class are [0.63%, 0.94%, 0.83%, 0.7%, 0.81%, 0.79%, 0.57%, 0.69%].

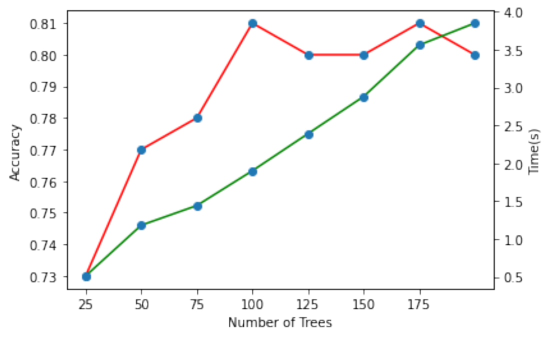


Fig. 9. Plot of possible tree values and their respective accuracy

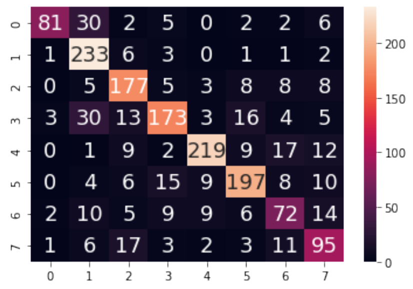


Fig. 10. Random Forest confusion matrix (test set)

Compared to the algorithms shown prior, random forest has performed the best. It has an overall accuracy of 77%, which is 11% better than our KNN, and has classified the 7th class (disgust), approximately 15% better than previous.

E. Neural Network

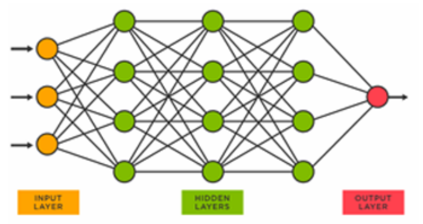
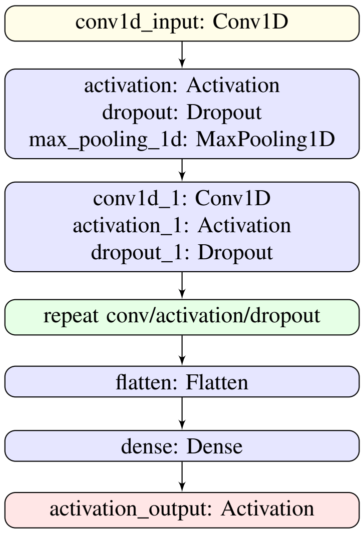


Fig. 11. Neural Network with 3 input, 3 hidden, and 1 output layer

The neural network was our fourth and final algorithm, and the desire was that it would outperform the previous three. Neural networks are very good at determining complex relationships between the features, and in the case of the MFCCs, where there isn't any clear connection between the samples, a neural network would be our go-to model.

Below, we present an architecture of our model.

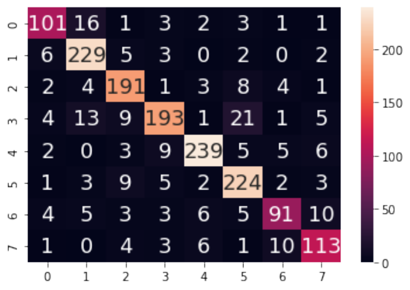


We have implemented a deep learning strategy based on convolutional neural networks (CNNs). We use convolutional, max-pooling, dropout and dense layers. Convolutional layers perform a linear combination between the input and the weights. The dimension of the convolution is given by the kernel size. We can repeat this process for any number of convolutions [9]. Pooling layers reduce the input size by only selecting the features that are the most vivid. It thus also helps to prevent overfitting, as it simplifies the model [10]. Dropout layers randomly drop a set number of neurons from the layers that they are applied to, thus simplifying the model once more, and helping to reduce overfitting [11]. The ReLU activation function changes all negative outputs to 0, and if not 0, leaves the output as is. The overall steps of the neural network are:

1) Neural network receives as input, a single MFCCs sample of dimension (40, 1)
2) It is passed through a convolutional neural network, that has 128 filters, and a kernel size of (5,5)
3) We then perform a reLU activation on it, and dropout 10% of the neurons to combat overfitting
4) To simplify the input more, we add a pooling operation with a pool size of (8,8) to select the main features
5) The convolution, activation and dropouts are repeated two more times, to determine more and more features from the input data
6) Once the learning layers are done, we flatten the layer, which now has 128 neurons
7) A dense layer is added to map the 128 neurons to 8, and finally a softmax activation is applied to determine the final class

Now that we understand how the neural network is constructed, we can discuss the results.

We see that the neural network performs the best overall, evident from the large diagonal values. It has an overall accuracy of 86%, which is higher than the random forest, which had 79%. On a per-class basis, we measure the accuracies as [0.84%, 0.89%, 0.87%, 0.91%, 0.86%, 0.85%, 0.82%, 0.78%]. The neural network was able to learn the surprised and disgust classes well, evident from their accuracies.



	0	1	2	3	4	5	6	7
0	101	16	1	3	2	3	1	1
1	6	229	5	3	0	2	0	2
2	2	4	191	1	3	8	4	1
3	4	13	9	193	1	21	1	5
4	2	0	3	9	239	5	5	6
5	1	3	9	5	2	224	2	3
6	4	5	3	3	6	5	91	10
7	1	0	4	3	6	1	10	113

Fig. 12. Neural Network confusion matrix (test set)

We now will discuss the learning curves of the neural network model, to see how it learned over time.

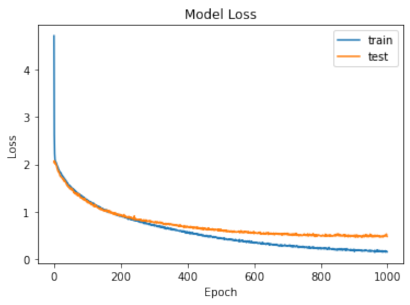


Fig. 13. Neural Network model loss learning curve)

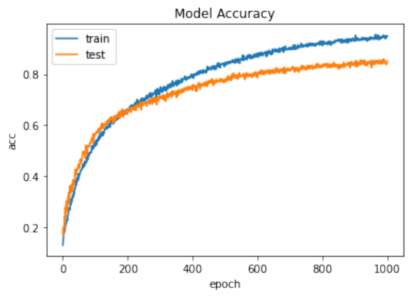


Fig. 14. Neural Network model accuracy learning curve

In these graphs, we are plotting how the loss and accuracy of the model changes as it continues to train over many epochs. We plot both the loss/accuracy on the training set as well as the test set. We can clearly see that we are not underiftting, nor are we overfitting. We have a well established model, and the learning curves reflect this [12]. In the model loss curve, we see that as the number of epochs increases, the training loss reaches a very low number, while the testing loss is slightly above, and it seems to remain there. Plateau's are often common when training over large number of epochs, and if we were to train more, it would just result in overfitting, so we left it at this. Similarly for the model accuracy, we see that the model has learned very well on the training and testing set, and the testing set is slowly beginning to level off. Any more training would not result in a better accuracy.

V. Bias-Variance Tradeoff

A final note to discuss is how we tackled the bias-variance tradeoff within the models. We will use the neural network model as our example. Initially, as we had no idea on how simple or complex the network needed to be, we started off simple, with a single dense layer consisting of 64 neurons. After training for 300 epochs, we saw that the training and validation set had extremely low accuracies on both. This meant that we had high bias, high variance, and were underfitting. The solution was to add complexity to the model, and we did this next by introducing convolutional and max pooling layers. While keeping the epochs the same, we noticed that we improved drastically on the training and validation set, but they were still less than 80% on both. In this case, we were in a lower bias, lower variance state, but our accuracy was still not where we wanted. We decided to extend out out the neural network by repeating our convolutional and max pooling layers, increased our epochs, and this resulted in a good accuracy for our training set, but our validation set remained around the same. From the learning curves, it was apparent that we were overfitting, and thus in a low bias, medium variance range. To combat overfitting, we introduced an L2 kernel regularizer and compared the results of it against that of a dropout layer. The dropout layer resulted in a better accuracy on the validation set, and thus we were eventually in the low bias, low variance state. From this state, if we increased the complexity of the model, we went back to the overfitting state, and if we simplified it, we lost accuracy on the training and validation set. We realized that this was the best state for our model. We followed the same approach for the remaining models.

VI. Overall Analysis

To summarize our results, below is a plotting of different models and their relevant accuracies.

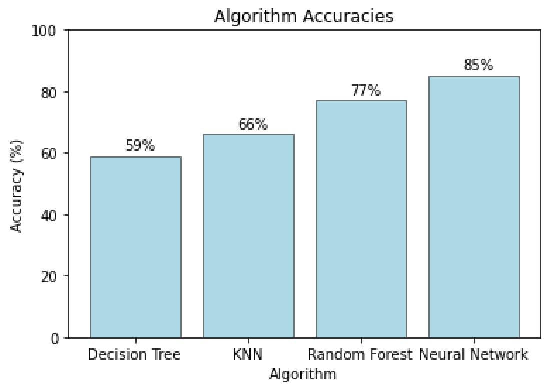


Fig. 15. Accuracies for all models

We can see that the neural network performed the best, while the decision tree performed the worst.

VII. Implementation and Code

In this project, we used a variety of machine learning libraries to help us analyze the data. We started off by using *numpy* to transform our data into the relevant shapes, and used *librosa* extensively to generate our features for our speech samples. Along the way, *matplotlib* was used to plot various curves, including waveforms, graphs, and some results from our validation set. Similarly, *seaborn* was used to plot the confusion matrix, which we relied on heavily for our analysis. For the models themselves, *sklearn* was used for splitting the data, and for all models except the neural network. The neural network model heavily utilized *keras*, and with all these packages, our project was completed. Research papers were utilized to sufficiently understand how speech analysis can be done through MFCCs, and librosa documentation was largely used to implement what had been read.

References

[1] Livingstone, Steven R., and Frank A. Russo. ”The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS).” Zenodo. 05 Apr. 2018. Web. 20 Sept. 2021.

[2] Logan, Beth. ”Mel Frequency Cepstral Coefficients for MUSIC Modeling.” Zenodo. 23 Oct. 2000. Web. 20 Sept. 2021.

[3] Muda, Lindasalwa, Mumtaj Begam, and I. Elamvazuthi. ”Voice Recognition Algorithms Using Mel FREQUENCY CEPSTRAL COEFFICIENT (mfcc) and Dynamic Time Warping (DTW) Techniques.” ArXiv.org. 22 Mar. 2010. Web. 20 Sept. 2021.

[4] Harsh H. Patel, Purvi Prajapati ”Study and Analysis of Decision Tree Based Classification Algorithms.” International Journal of Computer Sciences and Engineering 6.10 (2018): 74-78.

[5] Gongde Guo Hui, et al. ”KNN Model-Based Approach in Classification.” (2003).

[6] Ali, Jehad et al. ”Random Forests and Decision Trees”. International Journal of Computer Science Issues(IJCSI) 9. (2012).

[7] Grossi, Enzo et al. ”Introduction to artificial neural networks”. European journal of gastroenterology & hepatology 19. (2008): 1046-54.

[8] Hossin, Mohammad et al. ”A Review on Evaluation Metrics for Data Classification Evaluations”. International Journal of Data Mining & Knowledge Management Process 5. (2015): 01-11.

[9] Albawi, Saad et al. ”Understanding of a convolutional neural network.” 2017 International Conference on Engineering and Technology (ICET).

[10] Yu, Dingjun et al. ”Mixed pooling for convolutional neural networks.” International conference on rough sets and knowledge technology.

[11] Srivastava, Nitish et al. ”Dropout: a simple way to prevent neural networks from overfitting”. The journal of machine learning research 15. 1(2014): 1929–1958.

[12] Murata, Noboru et al. ”Learning curves, model selection and complexity of neural networks”. Advances in Neural Information Processing Systems. (1993): 607–607.