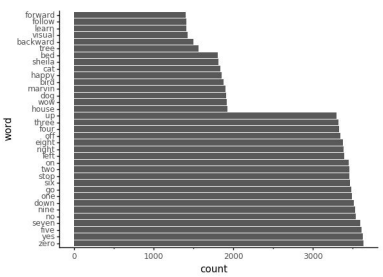
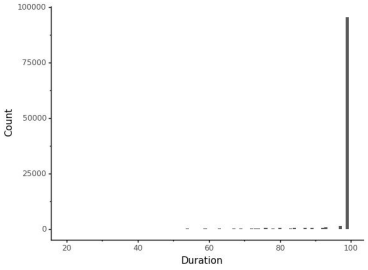
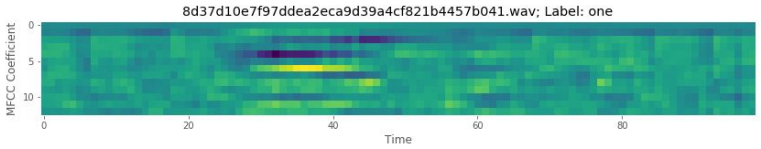
EDA: The number of unique labels to classify were 35, making this a multiclass classification problem. The distribution of the labels is unbalanced but the least common label still has 1402 data points, which appears sufficient for learning.

The majority of data points had a duration of 99, with some exceptions. These were zero padded to bring their lengths to 99 in order to create a 3 dimensional matrix.

Six wav files were removed because they were either not found in the train or test set files or because they were not recordings of words (noise). This leaves the dataset with 105829 files.







Feature Extraction: Two sets of features were extracted from the provided files. The first is a set of aggregate features which comprise the mean, maximum, minimum, skewness, kurtosis and standard deviation of each file for each coefficient. This resulted in 78 features. The second set of features was the complete padded data of 1287 data points in a 2 dimension format. By adding another dimension representing the audio channel this created a 4 dimension data matrix.

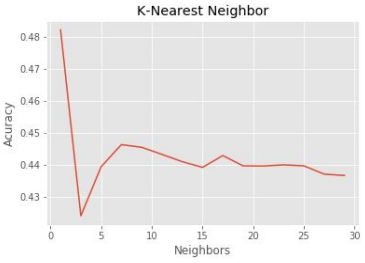
Algorithms: We attempted KNN, random forests, NN and CNN algorithms. Out of these the CNN achieved the highest accuracy score with 86% on a validation set and 85% on the final test data.

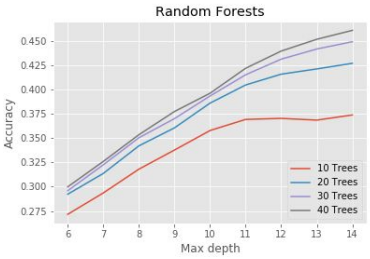
Tuning: We tuned the *KNN* with different levels of K ranging from 1 to 30. Surprisingly the best K was 1 with the second best being 7. KNN never surpassed the initial 48% accuracy and was thus dropped.

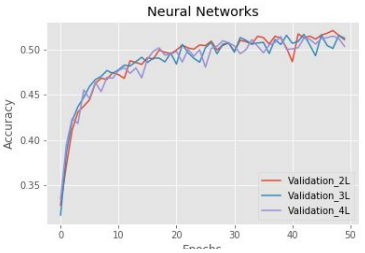
The *random forests* were tuned by changing the number of forests used and the max depth of these trees. Increasing the max depth improved the accuracy of the forests. For larger forests, overfitting was less of an issue. As the parameters increased there were diminishing returns, however, and the computation intensity outweighed any further accuracy gains.

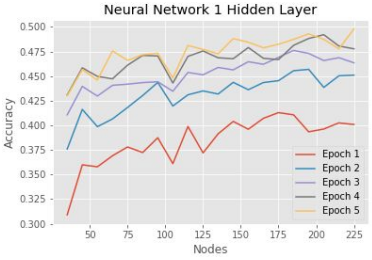
The *neural networks* were tuned by adding more layers (35 nodes each). There appeared to be no increase in validation set accuracy by adding more layers of the same size. Next the NN was tuned by changing the number of nodes in an NN with only one hidden layer. It appears that as the number of nodes increases, the validation set accuracy increases for the first 5 epochs. The additional nodes did not seem to have captured significant abstractions/features of the data as the increases are minimal.

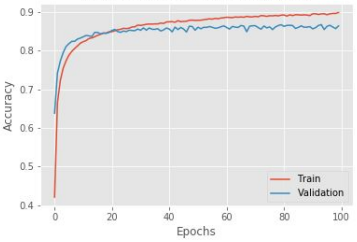
The *convolutional neural network* was tuned initially by changing the kernel size of one layer of 2D convolution (3 by 3 up to 11 by 11 in steps of 2). Although the accuracy improved the largest gains were achieved when layering multiple convolutions (3 by 3) with pooling and 20% dropouts. These helped against overfitting and allowed for proper feature extraction over multiple layers. The convolution output was fed through some more dense layers. These layers performed the necessary calculations on the features extracted in the layers before. This method resulted in the highest validation accuracy and was chosen as the final method.











Account Name used for submission: randomstate666

Specification of work task division

Prathna Vaswani

Research on theory (What are MFCCs? What format do they come in?).
Research into best approaches to tackle audio classification (Proposed CNN, RNN, HMM[1]).
Creation of aggregate features (e.g. mean, min, max, etc.)

Noah Smeets

Implementation and tuning of KNN, random forest and CNN algorithms.

Marco Wedemeyer

Implementation and tuning of NN and CNN algorithms.
Creation of 3D features for CNN with padding (zero and mean padding).

References

https://github.com/lukas/ml-class/blob/master/videos/cnn-audio/preprocess.py
https://www.heatonresearch.com/2017/07/22/keras-getting-started.html
https://keras.io/getting-started/sequential-model-guide/
https://librosa.github.io/librosa/feature.html
https://librosa.github.io/librosa/generated/librosa.feature.mfcc.html
https://medium.com/prathena/the-dummys-guide-to-mfcc-aceab2450fd
https://medium.com/@mikesmales/sound-classification-using-deep-learning-8bc2aa1990b7
https://opensource.com/article/19/9/audio-processing-machine-learning-python
https://plotnine.readthedocs.io/en/stable/tutorials/miscellaneous-order-plot-series.html
https://www.pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/
https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw
https://stackoverflow.com/questions/27516849/how-to-convert-list-of-numpy-arrays-into-single-numpy-array
https://stackoverflow.com/questions/8251541/numpy-for-every-element-in-one-array-find-the-index-in-another-array
https://towardsdatascience.com/how-to-use-ggplot2-in-python-74ab8adec129
https://www.youtube.com/watch?v=Qf4YJcHXtcY
https://www.youtube.com/watch?v=Z7YM-HAz-IY

[1] Hidden Markov Model