Analyzing Voice Samples to Predict Age

Final Report

Raj Singh

Problem Statement

The project is for a company called Neurolex Labs. Neurolex analyzes voice samples to help diagnose diseases such as schizophrenia, Parkinson's disease, Alzheimer's disease, and depression. In this case, we are analyzing human voice samples to determine the age of the speaker. We may want to know the age range of the individual so we can get specialized care for people in that age range. Moreover, knowing the age of the individual can give us a better context for the nature of their disease, and how likely they are to have a particular disease.

The Data

The name of the dataset is called the Common Voice Dataset by Mozilla. It contains about 200,000 voice samples. It contains the feature we are interested in, which is the age of the speaker. In order to featurize this audio data, we will be using two Python libraries: pyAudioAnalysis and Librosa.

Data Cleaning

The data can be downloaded as a .csv file from the internet, so we can load in the data as a Pandas Dataframe. Not all entries of the dataset were labeled with age categories, so those entries must be deleted. In order to featurize the data for our multilayer perceptron model, we use pyAudioAnalysis to featurize the data. An example of this can be found here. We also use Librosa to convert the audio files into log-mel-spectrograms, which is essentially image data with two channels. An example of this can be found here. Generally speaking, there were not many data cleaning steps except for those that involved featurizing the audio data.

Modeling

We will try two approaches: a Multilayer Perceptron Model as well as a Convolutional Neural Network.

Multilayer Perceptron Model

Our basic architecture for the model was an input layer with nodes equal to the number of features (~80) and an output layer equal to the number of classes (8). There are four hidden layers. The first one has 256 nodes, the second has 1024, the third has 1024, and the fourth has 256. Because there are is a relatively high number of layers, as well as nodes, the neural network can make more complex and accurate predictions. Of course, this comes at the expense of training time and overfitting. This is why we use a GPU provided by Google Colaboratory and dropouts after the first and third layers (to deal with high training times and overfitting, respectively).

Initial featurization using pyAudioAnalysis yields 170 features for each audio sample. Although the training set has 200,000 samples, only about 73,000 have labels. PyAudioAnalysis yields 35 features that are recorded at each of a certain interval of the sample, and we take the minimum, maximum, mean, median, and standard deviation to get 170 total features. Since there are so many features, we use the L1 regularization with a Support Vector Classifier to select the most important features in the model. This approach was adopted from Jim Schwoebel's book "Introduction to Voice Computing in Python." The code repository can be found here.

We tried two models that were slightly different in nature. Both had dropouts of 0.2 after the first layer, but the second only had a dropout of 0.15 after the third layer. To compensate for the higher dropout in the first model, there were 50 training epochs in the first model and 40 training epochs in the second model. Generally speaking, as we increased the training epochs after this point, the accuracy started to increase by marginal amounts, and the model tended to overfit

more. Thus, this MLP architecture optimized a model with a relatively high accuracy and low tendency to overfit.

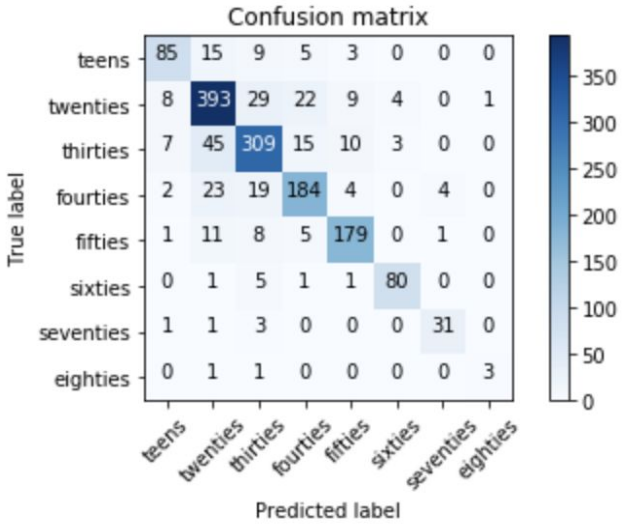
Since there were an uneven number of samples in each age category, we used balanced class weights in our model. That is, we weighted samples that appeared less often higher than samples that appeared more often.

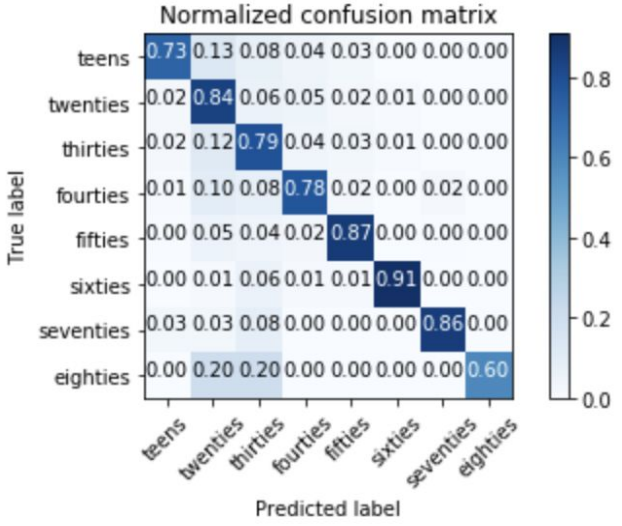
MLP Model 1 Results

Training Accuracy: 84.5%

Testing Accuracy: 82.0%

Confusion Matrices:





Classification Report:

	Precision	**Recall**	**F1-Score**	**# Samples**
Teens	0.82	0.73	0.77	117
Twenties	0.80	0.84	0.82	466
Thirties	0.81	0.79	0.80	389
Fourties	0.79	0.78	0.79	236
Fifties	0.87	0.87	0.87	205
Sixties	0.92	0.91	0.91	88
Seventies	0.86	0.86	0.86	36
Eighties	0.75	0.60	0.67	5
Micro Avg.	0.82	0.82	0.82	1542
Macro Avg.	0.83	0.80	0.81	1542
Weighted Avg.	0.82	0.82	0.82	1542

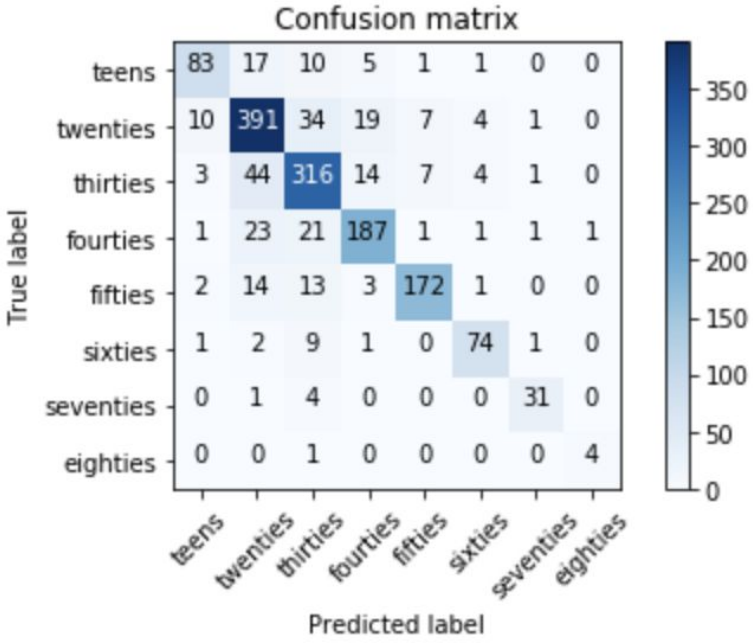
The code for this model can be found here.

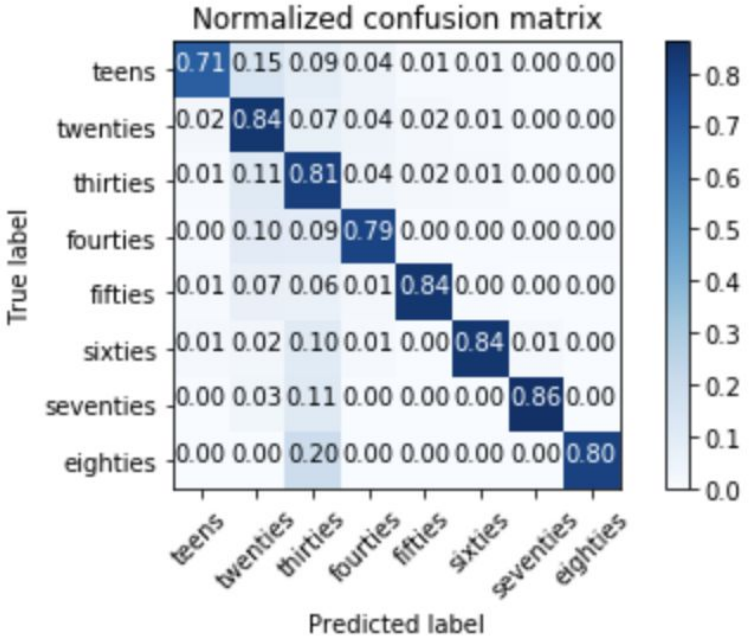
MLP Model 2 Results

Training Accuracy: 85.1%

Testing Accuracy: 81.6%

Confusion Matrices:





Classification Report:

	Precision	**Recall**	**F1-Score**	**# Samples**
Teens	0.83	0.71	0.76	117
Twenties	0.79	0.84	0.82	466
Thirties	0.77	0.81	0.79	389
Fourties	0.82	0.79	0.80	236
Fifties	0.91	0.84	0.88	205
Sixties	0.87	0.84	0.86	88
Seventies	0.89	0.86	0.87	36
Eighties	0.80	0.80	0.80	5
Micro Avg.	0.82	0.82	0.82	1542
Macro Avg.	0.84	0.81	0.82	1542
Weighted Avg.	0.82	0.82	0.82	1542

We can see that the average precisions, recalls, and f1-scores are very similar between the two models. The second model has a slightly higher accuracy on the test set. We would prefer the second model because it overfit less. Thus, perhaps having more dropout and more epochs works better for this particular problem. There seems to be some variability between the results for those in their eighties, and this is likely because there are so few samples that have those labels. We would likely benefit from more training data in this particular category. The code for this model can be found here.

Convolutional Neural Network Model

One common way of featurizing audio files is to convert it into a log-mel spectrogram, which is essentially converting it into an image with two channels. Instead of the axes being height and width, the axes are time and frequency and the values are the amplitudes of the wave instead of an RGB value. Information regarding log-mel spectrograms can be found here. Details of featurizing the audio data into log-mel spectrograms can be found here.

Featurization of the audio files into log-mel spectrograms resulted in huge files which were very hard to process with Keras. Moreover, even when the data was able to be successfully loaded, the CNN models had extremely low accuracies and took significantly longer than the MLP model. The architecture of the CNN can be found here.

It is not completely clear why the CNN model did not work, but there are a few possible explanations. The files were featurized into 60x41 spectrograms, and this may have been too small of an image for the CNN to process effectively. Furthermore, the exact architecture of the CNN may not be optimal for the problem we are dealing with. It may be helpful to learn more about how many bands and frames were used (what are the optimal dimensions for the spectrogram?) with typical audio classification models that use the log-mel spectrogram. Thus, reading a research paper on the topic could be beneficial.

Conclusion

In this deep learning project, we featurized audio samples from the Common Voice Dataset to predict the age category of the speaker. Because there were so many categories, and we don't have intuition about the features that the Python libraries are capturing, using a deep learning model is preferable over a typical machine learning approach.

Generally speaking, we got very good results by doing some basic LASSO feature selection and using a normal artificial neural network. When the models trained on a GPU, training was fast. Additionally, the model did not overfit. Precision, recall, and f1-scores were relatively high.

Another common approach for audio classification problems is using a log-mel spectrogram to turn the problem into an image classification problem, and accordingly using a CNN. However, it was very difficult to work with the 4D Numpy array (N x 61 x 40 x 2) and so running the models became very tedious. The accuracies were also very low (around 30%), so using this approach for this particular problem may not be useful. As a result, it is not recommended to use a CNN for this problem.

Though we got adequate results from our MLP model, our performance metrics could always be better. Perhaps, we should see what happens when continuing to add more epochs, and balancing any overfitting by adding more dropout. We also did not try regularization as a way of dealing with overfitting. This could be a more effective way than using dropout. We could also experiment with deeper networks and/or more nodes. Finally, using batch normalization might be an effective way of stabilizing the neural network.