Gender and Age Recognition Through Voice Data

Noah Bardenstein, Evan Donohoe, Vishakh Subramanian, Kyle Takeuchi

ECE 4424

Data Collection:

We started our design process by collecting mp3 files from a kaggle dataset: https://www.kaggle.com/datasets/mozillaorg/common-voice. The original dataset had around 185,000 files of 2-3 second voice recordings, with only some of these recordings having labels for age, gender, and accents. The ‘gender’ label had a value of either ‘male’ or ‘female’, and the ‘age’ labels were ‘teens’, ‘twenties’, ‘thirties’, ‘fourties’, ‘fifties’, ‘sixties’, ‘seventies’, and ‘eighties’. For the purpose of this project, we have omitted the 'accent' label, as we decided to only classify the audio files based on their ‘gender’ and ‘age’. The original speaker had the option to report these attributes, which is why most of the .mp3 files did not have age or gender labels. We initially downloaded all of the mp3 files through the folder “cv-valid-train”, with ‘valid’ meaning each recording has been listened to by at least 2 other people that have confirmed the audio can be correctly interpreted.

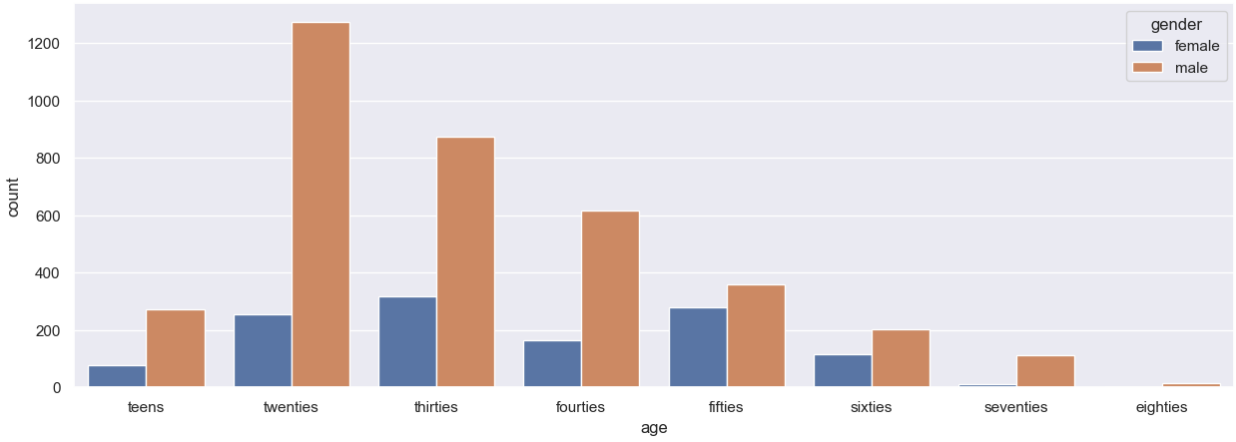

Figure 1: Distribution of ages of voice samples.

Data Transformation and Preprocessing:

In order to train our models, we needed to first gather all of the .mp3 files that contained labels for the gender and age. To do so, we downloaded all 185,000 .mp3 files and created a Python script that identified which .mp3 files contained a label. This created an output folder of 10,000 .mp3 files. We then created another Python script that moved all 10,000 labeled .mp3 files into a folder called mp3_files. We then used a script that would convert all files within mp3_files into .wav files in a different folder called wav_files. We initially ran the R script to extract audio properties of all 10,000 files, but with each set of 1,000 wav files taking an hour and thirty minutes to complete, we decided to extract a total of 5,000 files due to time constraint, partitioning them into 5 separate files (1,000 files extracted properties in one execution of R script) for each member to run on their own.

Feature Extraction:

From our research, we determined that the use of R was the best method to extract unique audio properties. Transforming our .mp3 files into .wav files allowed us to use an extension of a seewave library function in an R function called specan3 that would extract the following features: meanfreq (mean frequency of sound signal in kHz), sd (standard deviation of frequency), median (median frequency in kHz), Q25 (first quantile in kHz), Q75 (third quantile in kHz), IQR (interquartile range in kHz), skew (skewness), kurt (kurtosis, measure of the tailedness of the probability distribution), sp.ent (spectral entropy, distribution of power in the frequency domain), sfm (spectral flatness measure, measure of how close a sound is to being noise), mode (mode frequency in kHz), centroid (frequency centroid), meanfun (average fundamental frequency in kHz), minfun (minimum fundamental frequency in kHz), maxfun

(maximum fundamental frequency in kHz), meandom (average of dominant frequency in kHz), mindom (minimum of dominant frequency in kHz), maxdom (maximum of dominant frequency in kHz), dfrange (range of dominant frequency in kHz), and modindx (modulation index, determines the harmonic content of sound). We formed a general relationship between voice and gender/age using these properties and began training our model.

Data Split:

We decided to use a 70/30 split for training and test data due to our relatively small dataset, as we wanted to ensure that there is enough data for the model to learn from. With an 80/20 split, there is an inherent risk of overfitting, and when we tested the differences, it led to our results being less accurate than that of a 70/30 split. Both models executed quicker and had higher accuracies when a 70/30 data split was used.

Model Tuning:

For the most accurate SVM model, the hyperparameters used were a radial basis function (RBF) kernel and regularization parameter (c) of 1. These hyperparameters allowed a gender accuracy of around ~93% and an age accuracy of about ~68%. Initially, we were only able to get an age accuracy of about 40% for both male and female points; however, given a tolerance of plus or minus 1 classification bin, we were able to overcome this shortcoming. For example, if the model predicted an age label of “thirties” when the correct label was “forties”, the model would consider itself accurate for that point as it falls within the tolerance threshold. This allowed the accuracy for age to be greatly improved and didn’t interfere with the remaining results. For our random forest algorithm, we experimented with several different values for

n_estimators which is the hyperparameter that determines how many decision trees are used in the random forest ensemble model. N = 1000 estimators gave us the best performance.

Performance Evaluation:

Our implementation resulted in SVM models with gender accuracy of around ~93% and an age accuracy of about ~68%. The Random Forest models resulted in similar performance. It should be noted that running our random forest model with 1000 estimators took about 30 seconds as opposed to the SVMs 1.3 seconds, while giving nearly the same accuracy. With ML model selection, prediction time can be an important factor in deciding which model to use, thus comparing the two models, SVM using an RBF kernel. In **Figure 2**, we show a test of several selected samples. The first four are voice recordings of our group members; luckily all of us were correctly classified. An additional female sample using the same method was also used. The last three were taken from our train data. As we can see the twenties males were correctly classified; however, the female samples were not as successful. This is likely due to the skewed distribution of samples, as seen in **Figure 1** with significantly fewer female samples.

Labeled Custom Inputs	SVM (RBF, c = 1)	Random Forest (N=1000)
age gender twenties male twenties male twenties male twenties male twenties female twenties female teens female fifties female	0 : ('twenties', 'male') 1 : ('twenties', 'male') 2 : ('twenties', 'male') 3 : ('twenties', 'male') 4 : ('thirties', 'female') 5 : ('twenties', 'female') 6 : ('twenties', 'female') 7 : ('thirties', 'male')	0 : ('twenties', 'male') 1 : ('twenties', 'male') 2 : ('twenties', 'male') 3 : ('twenties', 'male') 4 : ('thirties', 'female') 5 : ('twenties', 'female') 6 : ('twenties', 'female') 7 : ('thirties', 'male')
Accuracy values	Gender: 92.2559% Male age: 68.4540% Female age: 69.3989%	Gender: 92.3906% Male age: 68.2583% Female age: 69.3989%

Figure 2: SVM results and Random Forest results of the same custom data points.

Contributions:

Evan and Vishakh found sci-kit implementations for SVMs and Random Forests. Vishakh focused more on the implementation of the Random Forest, while Evan focused more on the SVM. They both worked on formatting the input .csv data to allow as an input to both functions. They worked on building a gender, male age, and female age dataset. With these, they created independent training and testing data sets. They used a 70-30 split as mentioned above. They normalized the features and replaced the string labels with integers as discussed. With these normalized sets, they were able to pass in the training data and testing data and found the accuracy on the testing data set for gender, male age, and female age. Noah and Kyle worked on the data preprocessing and feature extraction scripts. Kyle worked with Noah on the development and execution of the different python scripts. They implemented two python scripts required for the preprocessing and the R script required to extract the unique audio properties which would then be used for the model. They also worked with Evan and Vishakh on the implementations of the two algorithms and the logic of the project.