CS 584 Final Project

Jake Whalen

December 14, 2017

# Description

## Executive Summary

This final project was developed for the GMU CS 584 Spring lecture course at Engility. The goal was to choose an interesting data set and apply the data mining techniques learned in class to the data. There was an additional goal to go above an beyond what was done in class. This was accomplished in several ways during the completion of this project.

The data selected was from [Kaggle](https://www.kaggle.com/primaryobjects/voicegender) and contains over 3,000 pre-processed audio samples. Each sample contains a label for the gender of the voice in the clip. The data was stored in a CSV file and then loaded into both R and Python. R was used for early exploratory data analysis. The data was plotted and run through statistical tests to better understand the underlying distributions. Python was used for applying the data mining techniques learned in class. The [scikit-learn](http://scikit-learn.org/stable/) package was used for all of this work. It was used previously for all three homework assignments so the application of the package was fast and easy for the analyst. The results of the models in python were exported into Excel files and then imported into R for reporting. The final presentation and report were written in R Markdown. This meant that making a slight change to the model parameters would automatically update the results displayed in the presentation and report.

The final results were encouraging. The worst model achieved over 80% classification accuracy while the best reached 98%. All models did better classifying Male voices. The better models achieved higher female classification accuracy. In the end it was a difficult choice between 3 different models. The best model did not achieve the highest rank in all metrics used for evaluation but it was consistently in the top 3 models for every metric.

## Motivation

Using machine learning techniques to understand human audio is a trending topic. Look no further then the in home personal assistants being sold by companies like Apple (Siri), Google (Google Home), and Amazon (Alexa). Answering a question posed by a human however can be quite difficult and would require massive amounts of data and a powerful computer to process and train models. This project reduces the scope to a more manageable level.

This project attempts to develop methods that can classify the gender of a speaker. While not as impressive as responding to a question with an answer this project will introduce how different features of a voice can help group the voice into one or more classes. These features could be used for more then gender classification. This project could potentially be extended to classify the speakers mood or age group. Overall this project was motivated by a curiosity to see how well a model could interpret a speakers voice.

## Data

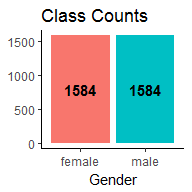
The data was downloaded from [Kaggle](https://www.kaggle.com/primaryobjects/voicegender) and stored in a CSV file. The data consisted of more then 3,000 gender labelled records. Each audio clip was pre-processed in R and the transformed into 20 features by the original poster. The features included measures such as mean frequency, skewness, spectral entropy and the average of the fundamental frequency measured across the acoustic signal. The data values all consisted of floating point values. The features had ranges from 0 up to 1,300 and thus might require some transformations when passed into a model. There were no missing values which allowed the analyst to concentrate on the models and there results rather then preparing the data.

# Methods

## EDA

The first step in building a gender classifier was to inspect the data. This was achieved through data visualization and statistical tests. The main idea of a visualization is to gain insight into the data. Rather then view the raw numbers in a table, a visualization can quickly tell a story.

The first feature in the data to be inspected was the label. This feature is what every model built in the classification step will try to determine. Thus the label may well be the most important feature in the data. In this project the label consists of two values, male and female. The graph below shows the count of each label.



What is important to note is that the classes are evenly distributed. There are an equal number of male and female voice recordings in the data set. Thus a ‘dumb classifier’ that predicts male every time would be correct 50% of the time. In step 2 this will be used as the threshold to measure whether a model is of any use or not. A model that achieves less then 50% gender classification accuracy should be discarded.

The next step after inspecting the class counts was to visualize the data through graphs. Since every feature is numeric each one could be inspected individually and against one another. This step included a lot of trial and error attempting to generate plots that would give insight into the data.

Multiple plots were generated such as density, box, heat map, and 2D/3D scatter plots. The density and box plots helped show how each feature was distributed and potentially split the gender classes apart. The scatter plots helped to show the interaction between multiple features. In some cases two features would help to split the genders apart to make classification easier. The heat map generated included correlations between features. The correlations were used to remove any features where there was a perfect positive or negative correlation. Removing one of the features from this pair would not cause the data to lose any information while reducing the dimensionality. After inspecting the visualizations and removing some of the features that supplied duplicate information the modeling phase began.

## Models

The modeling phase consisted of two steps using four different algorithms. The first step used only the raw data with no transformation. The models included K-Nearest Neighbors (KNN), Decision Trees, Support Vector Machines (SVM), and Logistic Regression (Log R). The second step applied a transformation on the data prior to fitting the model. In both steps the models were fit using a grid search mechanism which allowed parameters to be varied and the best model chosen based on the cross-validation results.

### K-Nearest Neighbors

The first algorithm used was KNN. The parameters used to attempt to find the best KNN model included algorithm type, number of neighbors, distance metric, and weights. The first model fitted using raw data found that algorithm = auto, n\_neighbors = 11, p = 1, weights = distance resulted in the best cross-validation scores. After predicting genders on the test data the model resulted in an accuracy of 0.8107075.

Next a transformation was used on the data known as Principal Component Analysis (PCA). The idea behind PCA is to reduce the dimensionality data and find a set of vectors to capture a majority of the variance in fewer features. After applying PCA and fitting the KNN model it was determined that 9 components maximized the mean cross-validation test scores. As for the other parameters algorithm = auto, n\_neighbors = 3, p = 1, weights = distance resulted in the best scores. Overall the KNN model with the PCA transform resulted in an accuracy score of 0.9789675.

### Decision Tree

The next model used was a decision tree. The decision tree is simple to understand. At every node the tree makes a decision to split the data based on one features value. Again a grid search was used to find the parameters that build the best tree for the data. It ended up that criterion = gini, max\_depth = 21, presort = TRUE, splitter = random generated the best tree with an accuracy of 0.9608031.

Rather then transform the data since the original model did so well the choice was made to apply more trees and take a majority vote. This method is also known as a Random Forest classifier. The Random Forest classifier found that criterion = entropy, max\_depth = 9, n\_estimators = 15 resulted in the best scores. It’s accuracy when classifying the test data was 0.9770554.

### Support Vector Machine

Next, an SVM was created to classify the data. This algorithm attempts to split the data multiple ways through equations and perfectly divide the genders into their on groups. It was found that by altering the penalty parameter C for SVM the model could achieve better results. The SVM classifier found that C = 48 resulted in the best scores. It’s accuracy when classifying the test data was 0.916826.

Similar to KNN the SVM model then was used with PCA transformed data to attempt to improve the baseline results. The performance was best when 13 principal components were used to fit the model. The SVM classifier with PCA data found that C = 10 resulted in the best scores. It’s accuracy when classifying the test data was 0.9608031.

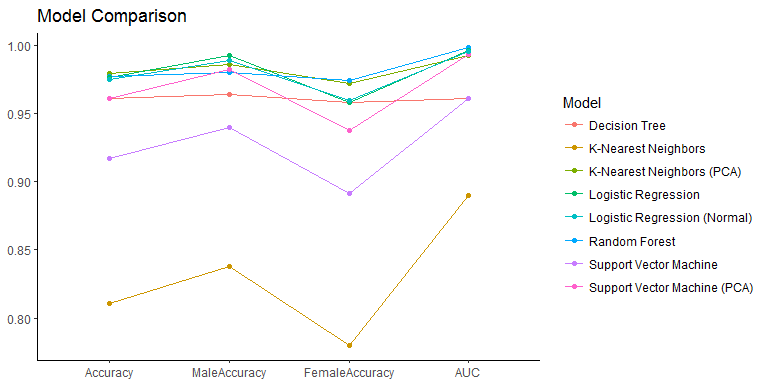
### Logistic Regression

Lastly, a model was built using Logistic Regression. The model was fit first with only the raw data. The Log Regression model accepted numerous parameters in python. To achieve better results this model varied the inverse of regularization strength (C), intercept fit, and a penalty value. A grid search found that C = 0.7000000000000001, fit\_intercept = TRUE, penalty = l1 resulted in the best scores. It’s accuracy when classifying the test data was 0.9760994.

Rather then apply a PCA transform which resulted in no change in accuracy for the Log Regression model a normalization was applied to the data. Each columns values were transformed so that the max value was set to 1 and minimum value was 0. This resulted in the model accuracy decreasing to 0.9751434. It seemed that the Log Regression model performed best on the raw data.

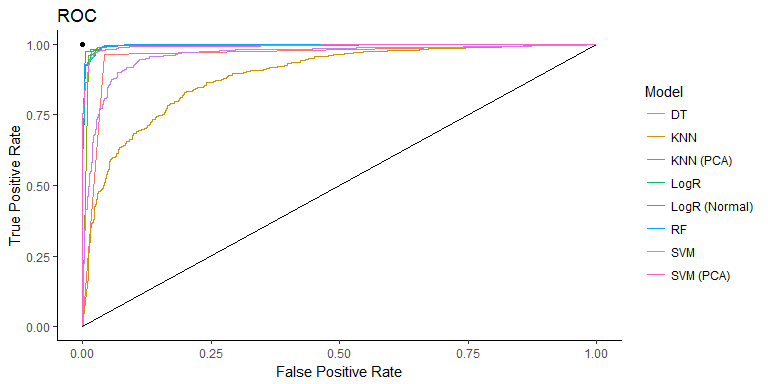
# Results

Evaluating the models required choosing the proper metrics to compare the model results. These metrics included overall accuracy, class specific accuracy, and the area under the ROC curve (AUC). The graph below shows the metrics for each of the models. No single model was ranked number one for every metric, thus there is no clear cut best model. No model ranked in the top three of every metric either. Instead the best model was chosen based on how it faired at each metric.



As evidenced by the graph above there were a handful of models that did well across all four metrics. The best model ended up being the Random Forest classifier. This model was chosen as the best because of it’s high marks for Overall Accuracy (2nd), Female Accuracy (1st), and AUC (1st). While other models such as KNN using PCA also achieved high scores it came down to the AUC of the Random Forest being so high (0.9979454) and it’s high Female accuracy. Classifying females voices proved to be a challenge for many of the models as seen by the lower values in the chart above.

The ROC chart below shows how close it was among the better models. While the first two models built (KNN & SVM) were okay, the rest of the models approached the black dot indicating a perfect classification.



# Conclusions

This project wound up being a success not only through the learning achieved but the results produced. Multiple models were generated and even achieved good results based on the metrics used for evaluation. During the execution of this project however a few lessons were learned.

The first was during the EDA phase. I wound up spending more time on this step then was necessary. This was partly due to finding a data set that was already clean and did not require much work. The second lesson was learned while writing the code for the models in python. Near the end of the project when time was becoming a factor I realized I might have spread myself thin with models to review. A better method could have been to run the base models and then move forward with the best two. Improving only these with the transformations. Given this is a school project done outside of work hours I found myself having to rush through reviewing models just to get them all accounted for.

Moving forward I would continue to refine the Random Forest and KNN (PCA) models. The challenge would be improving the female accuracy first since that is the area that most models struggled in. I would also like to gather the raw data and apply the models to it as well. This may prove to make better models if the raw data were used and transformed via PCA or a Fast Fourier transform.

A spin off of this project would be to also identify voice types. Clustering the data into groups and attempting to identify which clusters are tenor or baritone etc. I believe this would result in good results due to the fact that those voice types are divided up based on specific voice features. A machine learning model would simply automate the process and potentially beat out the human ear for accuracy.