Gender Recognition by Voice - Kaggle

Anirudh Pillai (aanirudh@iu.edu)

1. Introduction

Determining a person's gender as male or female, based upon a sample of their voice seems to initially be an easy task. Often, the human ear can easily detect the difference between a male or female voice within the first few spoken words. However, designing a computer program to do this turns out to be a bit trickier.

This project does detailed analysis of various machine learning models that can be used to identify voice as male or female. The model is constructed using 3,168 recorded samples of male and female voices, speech, and utterances. The samples were processed using acoustic analysis and then applied to machine learning algorithms to learn gender-specific traits.

This paper has been segmented in the following manner. Part I comprises of the Introduction, Part 2 describes the dataset, Part 3 talks about Data preprocessing and feature selection, Part 4 focuses on implementation of algorithms, Part 5 talks about model evaluation and results, Part 6 gives brief conclusion.

2. Dataset

Dataset contains 20 features and target variable - {male, female}. There are 3168 examples

Continuous: - 20
Binary: - 1

Snapshot of Dataset

meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	mode	centroid	meanfun	minfun	maxfun	meandom	mindom	maxdom	dfrange	modindx	label
0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.86346	274.4029	0.893369	0.491918	0	0.059781	0.084279	0.015702	0.275862	0.007813	0.007813	0.007813	0	0	male
0.066009	0.06731	0.040229	0.019414	0.092666	0.073252	22.42329	634.6139	0.892193	0.513724	0	0.066009	0.107937	0.015826	0.25	0.009014	0.007813	0.054688	0.046875	0.052632	male
0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.75715	1024.928	0.846389	0.478905	0	0.077316	0.098706	0.015656	0.271186	0.00799	0.007813	0.015625	0.007813	0.046512	male
0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322	0.727232	0.083878	0.151228	0.088965	0.017798	0.25	0.201497	0.007813	0.5625	0.554688	0.247119	male

Fig1- Overview of Dataset

Attributes or Acoustic Properties Measured

The following acoustic properties of each voice were measured:

- **meanfreq:** mean frequency (in kHz)
- **sd:** standard deviation of frequency
- **median:** median frequency (in kHz)
- **Q25:** first quantile (in kHz)
- **Q75:** third quantile (in kHz)
- **IQR:** interquantile range (in kHz)
- **skew:** skewness (see note in specprop description)
- **kurt:** kurtosis (see note in specprop description)
- **sp.ent:** spectral entropy
- **sfm:** spectral flatness
- **mode:** mode frequency
- **centroid:** frequency centroid (see specprop)
- **meanfun:** average of fundamental frequency measured across acoustic signal
- **minfun:** minimum fundamental frequency measured across acoustic signal
- **maxfun:** maximum fundamental frequency measured across acoustic signal
- **meandom:** average of dominant frequency measured across acoustic signal
- **mindom:** minimum of dominant frequency measured across acoustic signal
- **maxdom:** maximum of dominant frequency measured across acoustic signal
- **dfrange:** range of dominant frequency measured across acoustic signal
- **modindx:** modulation index. Calculated as the accumulated absolute difference between adjacent measurements of fundamental frequencies divided by the frequency range

Target Variable

label: - gender of subject (male or female)

There are no null values in our dataset which makes it easier for preprocessing.

Five point Summary

Table below represents 5 point summary of the data.

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	mode	centroid	meanfun	minfun	maxfun	meandom	mindom	maxdom	dfrange	modindx
count	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000
mean	0.180907	0.057126	0.185621	0.140456	0.224765	0.084309	3.140168	36.568461	0.895127	0.408216	0.165282	0.180907	0.142807	0.036802	0.258842	0.829211	0.052647	5.047277	4.994630	0.173752
std	0.029918	0.016652	0.036360	0.048680	0.023639	0.042783	4.240529	134.928661	0.044980	0.177521	0.077203	0.029918	0.032304	0.019220	0.030077	0.525205	0.063299	3.521157	3.520039	0.119454
min	0.039363	0.018363	0.010975	0.000229	0.042946	0.014558	0.141735	2.068455	0.738651	0.036876	0.000000	0.039363	0.055565	0.009775	0.103093	0.007812	0.004883	0.007812	0.000000	0.000000
25%	0.163662	0.041954	0.169593	0.111087	0.208747	0.042560	1.649569	5.669547	0.861811	0.258041	0.118016	0.163662	0.116998	0.018223	0.253968	0.419828	0.007812	2.070312	2.044922	0.099766
50%	0.184838	0.059155	0.190032	0.140286	0.225684	0.094280	2.197101	8.318463	0.901767	0.396335	0.186599	0.184838	0.140519	0.046110	0.271186	0.765795	0.023438	4.992188	4.945312	0.139357
75%	0.199146	0.067020	0.210618	0.175939	0.243660	0.114175	2.931694	13.648905	0.928713	0.533676	0.221104	0.199146	0.169581	0.047904	0.277457	1.177166	0.070312	7.007812	6.992188	0.209183
max	0.251124	0.115273	0.261224	0.247347	0.273469	0.252225	34.725453	1309.612887	0.981997	0.842936	0.280000	0.251124	0.237636	0.204082	0.279114	2.957682	0.458984	21.867188	21.843750	0.932374

Fig2: - Five point Summary of Dataset

Let’s check label distribution to see if our dataset is biased.

Number of male = 1584

Number of female = 1584

Dataset contains equal number of examples for both male and female hence it is not biased and we can achieve good accuracy during classification.

3. Data Preprocessing and Feature Selection

Data Cleaning

Male and female under label was replaced with binary values. {male :1 , female: 0}

Dataset contains no null or missing values, nor any redundant or irrelevant information hence there was not much data processing needed.

Exploratory Analysis

Relational plot with few variables is plotted using pairplot function to see correlation between them. It can be seen that centroid and meanfreq seems to be correlated.

Fig 3 : Pairplot between few features

For feature selection, Heat map with correlation between variables was created. It can be seen that Meanfreq and median appears to be correlated.

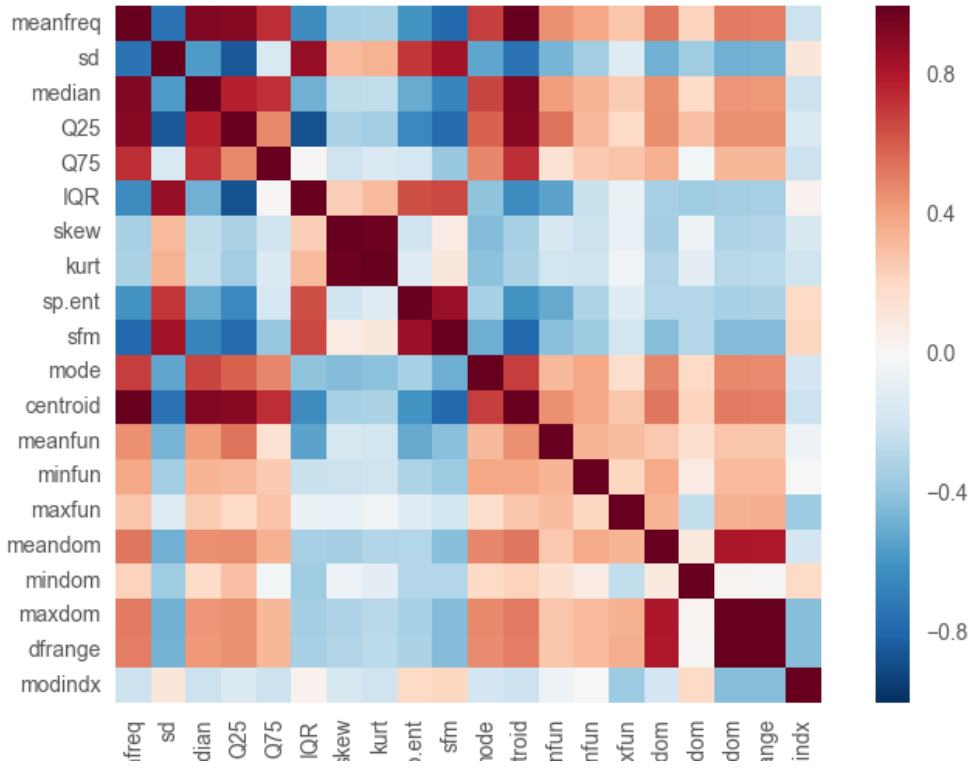

Fig 4: - Heat map indicating correlation between features

Feature Selection

i) Removing features with low variance: - VarainceThreshold was used which removes all variance whose variance doesn't meet some threshold. (80% in this case)
Variance for all the features were calculated and observed that variance is good for all the features hence none were removed.

ii) Univariate feature selection: - Univariate feature selection works by selecting the best features based on univariate statistical tests. SelectKBest removes all but k-high scoring features. Looking at the five point summary, I felt that 4-5 features could be dropped.

So, 16 best features were finalized after performing univariate feature selection.

Dataset Split: -

Dataset was divided into 80% training and 20 % tuning data.

Training set: - 2534

Tuning set: - 634

4. Algorithms

4.1 K-Nearest Neighbor

K-nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions).Since it is simple and intuitive with very flexible decision boundary and variable-sized hypothesis space, it is my first choice for classifying this problem.

KNN accuracy: 0.716088

We get an accuracy of 71% on test data with K-nearest neighbor which is more than random (50%).

4.2 Logistic Regression

Since we have binary classification, Logistic regression is my second choice because it is comparatively quicker and simple to implement and has few parameters to optimize. It assumes a smooth linear boundary of classification which is practically pretty intuitive. Logistic regression learns conditional probability P(y|x).

Logistic Regression score: 0.899054

We get accuracy of 89.9% on test data which almost 90% which is better than KNN.

4.3 SVM (Support Vector Machine)

SVM finds the best line of separation by the concept of maximal margin and I expected it to perform better than Logistic Regression. My data size is small and it can also be applied to very complex data. Kernels allow very flexible hypothesis and different kernel functions can be used to fit data properly.

I applied SVM with linear and RBF kernel and observed following results:

Kernel Comparison

Kernel	Accuracy
Linear C = 1	0.90782828
RBF C = 1	0.65877525
Linear C = 10	0.95391414
RBF C = 10	0.7427399

SVM Linear/RBF kernel analysis

After comparing results between RBF and Linear, I observed that linear kernel works better and we can further improve accuracy by selecting right C value by applying grid search. I tried polynomial kernel since it was consuming lot of time to fetch results, it was taken out of consideration.

To further fine-tune our model, I applied K-fold cross-validation and grid search to get right value of C.

i) **K-fold cross validation: -** The data set is divided into k subsets, and the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other k-1 subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided. K =5 was used to improve results.

ii) **C- value:** - The C parameter trades off misclassification of training examples against simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly by the model freedom to select more samples as support vectors.

Grid search results for tuning parameters

I tried following C values on linear and rbf kernel and received following results on train and test data:

parameters = {'kernel'☹'linear', 'rbf'), 'C':[0.01, 0.1, 1, 10, 100, 1000]}

({'kernel': 'linear', 'C': 1}, {'kernel': 'rbf', 'C': 1},

{'kernel': 'linear', 'C': 0.001}, {'kernel': 'rbf', 'C': 0.001}, {'kernel': 'linear', 'C': 0.01}, {'kernel': 'rbf', 'C':

0.01}, {‘kernel’: ‘linear’, ‘C’: 0.1}, {‘kernel’: ‘rbf’, ‘C’: 0.1}, {‘kernel’: ‘linear’, ‘C’: 1}, {‘kernel’: ‘rbf’, ‘C’:

1}, {‘kernel’: ‘linear’, ‘C’: 10}, {‘kernel’: ‘rbf’, ‘C’: 10}, {‘kernel’: ‘linear’, ‘C’: 100}, {‘kernel’: ‘rbf’, ‘C’:

100}, {‘kernel’: ‘linear’, ‘C’: 1000}, {‘kernel’: ‘rbf’, ‘C’: 1000}, {‘kernel’: ‘linear’, ‘C’: 10000}, {‘kernel’: ‘rbf’,

‘mean_train_score’: array([0.92818931, 0.76096775, 0.67463531,

0.65309395, 0.73153373,
0.65514411, 0.82528371, 0.69294323, 0.92818931, 0.76096775,
0.97285379, 0.8843917 , 0.97238023, 0.9705653 , 0.96393696,
0.99250365, 0.96030689, 0.99897414]),

‘mean_test_score’: array([0.91729798, 0.67487374, 0.64520202,
0.6395202 , 0.69917929,
0.6395202 , 0.79482323, 0.62752525, 0.91729798, 0.67487374,
0.96496212, 0.77114899, 0.95643939, 0.86805556, 0.95044192,
0.88257576, 0.94349747, 0.87089646]),

Best parameters: - {‘kernel’: ‘linear’, ‘C’: 10}
Best Accuracy: 0.96496212

I used the above parameters obtained after tuning for SVM on my Test Set and observed an accuracy of 97%

Accuracy on Test Dataset = 0.9700

After seeing the train and test score, I observed that there is not much variance between the score and our model doesn’t seem to overfit.

'mean_train_score': array([0.92818931, 0.76096775, 0.67463531,

0.65309395, 0.73153373,0.65514411, 0.82528371, 0.69294323, 0.92818931, 0.76096775, 0.97285379, 0.8843917 , 0.97238023, 0.9705653 , 0.96393696, 0.99250365, 0.96030689, 0.99897414]),

'mean_test_score': array([0.91729798, 0.67487374, 0.64520202, 0.6395202 , 0.69917929, 0.6395202 , 0.79482323, 0.62752525, 0.91729798, 0.67487374,0.96496212, 0.77114899, 0.95643939, 0.86805556, 0.95044192,0.88257576, 0.94349747, 0.87089646])

I believe SVM is the best model among all. I didn't try to further boost my performance using ensemble methods as I was able to get good performance with SVM and I don't see that my model hypothesis has very high variance/bias.

5. Results

We get following accuracies on our test dataset with each algorithm

Accuracy for KNN = 71.60 %
Accuracy for Logistic Regression = 89.9
Accuracy for SVM with linear kernel = 97 %

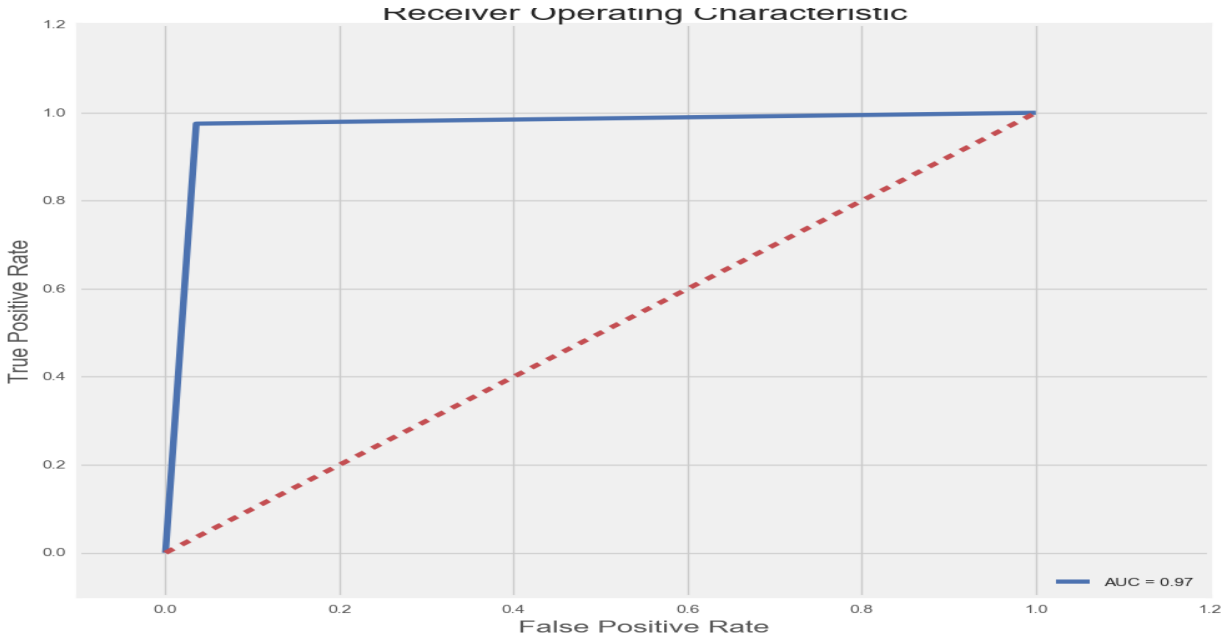

Fig 5:- Receiver Operating Characteristics Curve

Figure above shows ROC Curve which is very close to ideal spot.

Confusion Matrix

	Predicted No	**Predicted Yes**
Actual No	TN = 315	FN = 8
Actual Yes	FP = 11	TP = 300

6. Conclusion

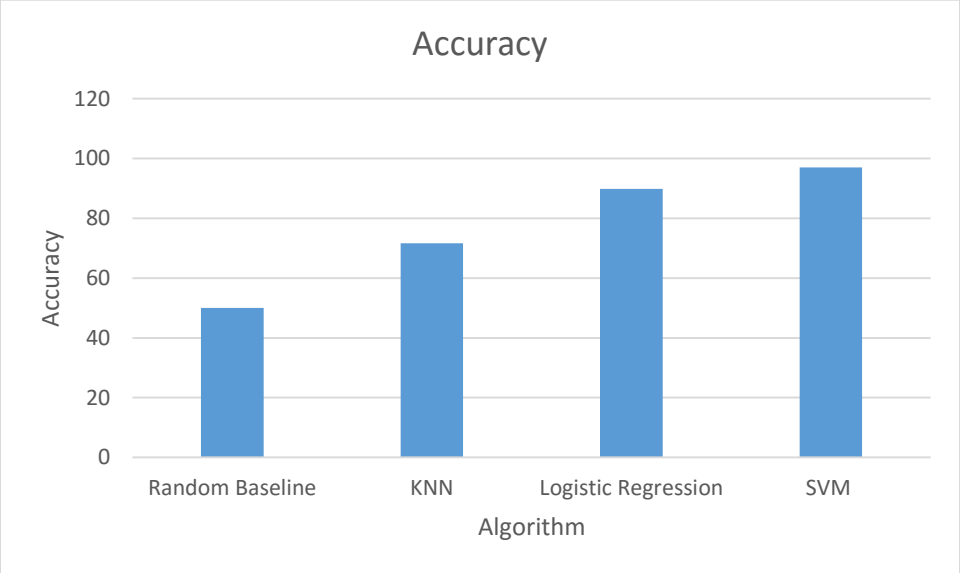

Fig 6: - Comparison between Algorithms

- **Comparison of Algorithms**

1. **Logistic regression vs K nearest neighbor:** - Logistic regression works better than K nearest neighbor.
2. **Logistic regression vs SVM**: - SVM works better than Logistic regression.

   We achieved accuracy of 97.23% on our training data and 95.64% on dataset after applying cross-validation which proves that model didn't overfit.

   Linear kernel with C=10 and with 5 k-folds gives 97% accuracy on our test dataset.

   Reasons why SVM works better than Logistic regression:

1. Minimizes generalization error with max-margin and works well on future data.
2. Minimizes complexity with fewer support vectors.
3. Minimizes the capacity of the classifier hence avoids overfitting.