CLASSIFICATION MODEL TO PREDICT THE GENDER BASED ON ACOUSTIC PARAMETERS

Submitted By:
Piyush Ranjan Srichandan

ABSTRACT:

The speech signal contains a vast spectrum of information about the speaker such as speakers' gender, age, accent, or health state. As a person ages, the acoustic characteristics of the voice change. Understanding how the sound of a voice changes with age may give insight into physiological changes related to vocal function. In this paper, we explored different approaches to automatic speaker's gender classification and age estimation system using speech signals. We applied various Deep Neural Network-based embedder architectures to age estimation and gender classification tasks.

INTRODUCTION:

Speech is a multidimensional phenomenon, the production of which consists of many anatomical structures movements that influence the overall speech quality and voice characteristics. Speech is the main and the easiest source of communication. Beside the linguistic information it also covers speaker-dependent para-linguistic data such as speaker's identity, emotional state, health state, age, or gender. The systems for automatic extraction of these information from speech might be very useful in numerous applications such as in personal identification in banking systems; customer care applications such as call centres; voice bots; and interactive, intelligent voice assistants. Extracting information about age and gender of the speaker may be used by the interactive voice response system (IVR) to redirect the speaker to an appropriate consultant or to play a suitable for a given gender/age group background music. For voice-bots systems, extraction of para-linguistic information may be applied to alter the behaviour of the bot. In the case of voice assistants, such knowledge may be used to target suitable advertisements or select search results that are more fitting for a given age/gender group. All combined, exploiting the para-linguistic content can lead to an improved user experience, and this in turn may generate revenue
for the company that decides to use such systems.

STEPS INCLUDED:

- Remove/handle null values (if any)
- Depict percentage distribution of label on a pie chart
- Considering all the features as independent feature and label as dependent feature, split the dataset training and testing data with test size=20%
- Apply the following classifier models on training dataset and generate predictions for the test dataset
  a. Decision Tree Classifier b. Random Forest Classifier
  c. KNN Classifier d. Logistic Regression e. SVM Classifier
- Also generate confusion-matrix and classification report for each model generated
- Report the model with the best accuracy.

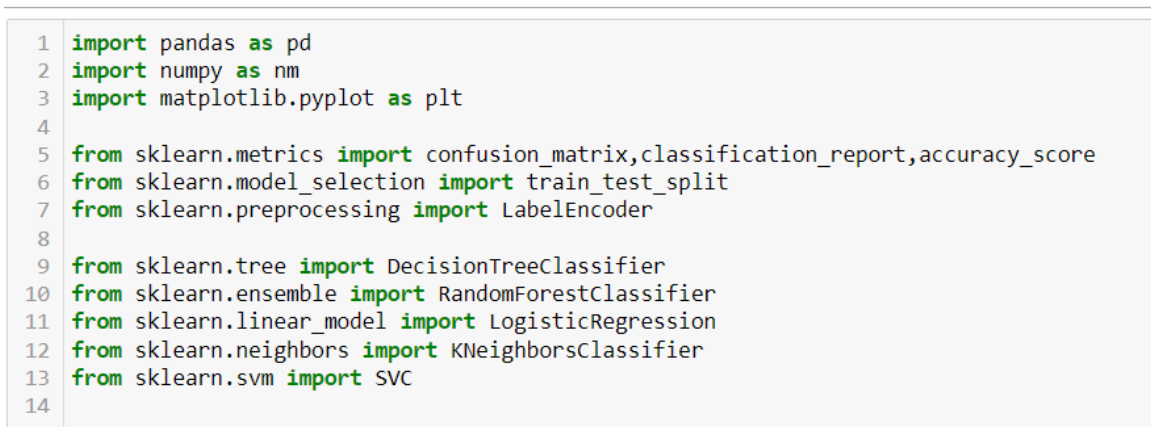
ACOUSTIC PROPERTIES MEASURED:

The following acoustic properties of each voice are measured:

- **duration**: length of signal
- **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)
- **peakf**: peak frequency (frequency with highest energy)
- **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

STEPS WE DID IN THE PROJECT:

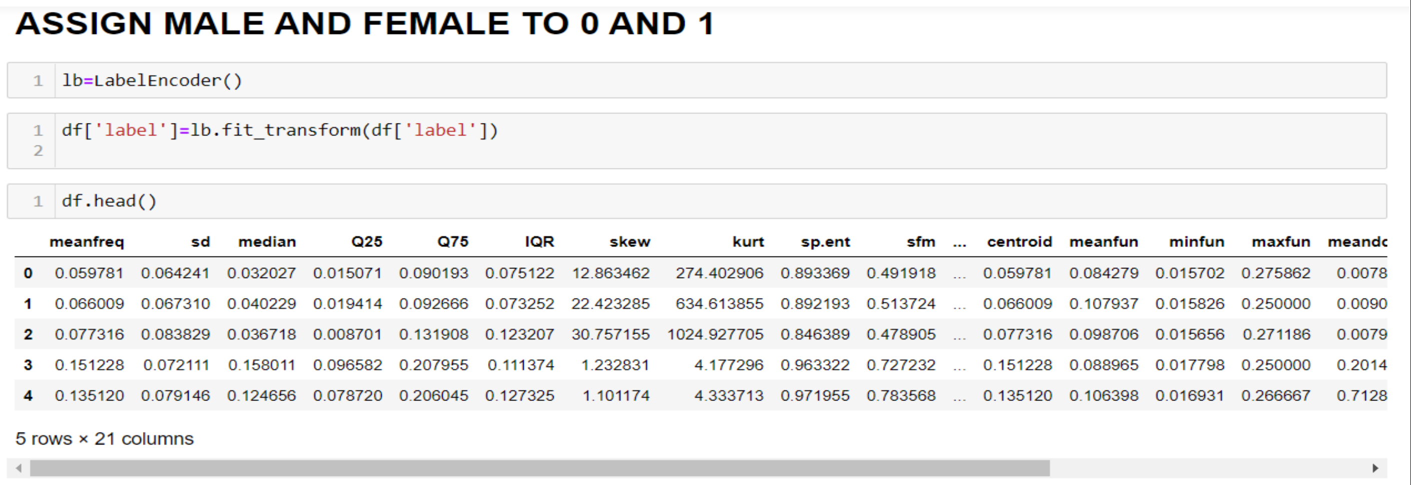
Importing Data Sets:-



```
import pandas as pd
import numpy as nm
import matplotlib.pyplot as plt

from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

```

Applying LevelEncoder: -

LabelEncoder[source] Encode target labels with value between 0 and n_classes-1. This transformer should be used to encode target values, i.e. y , and not the input X . So in our project all the males were assigned as 0 and females were assigned as 1.



ASSIGN MALE AND FEMALE TO 0 AND 1

```
lb=LabelEncoder()
```

```
df['label']=lb.fit_transform(df['label'])

```

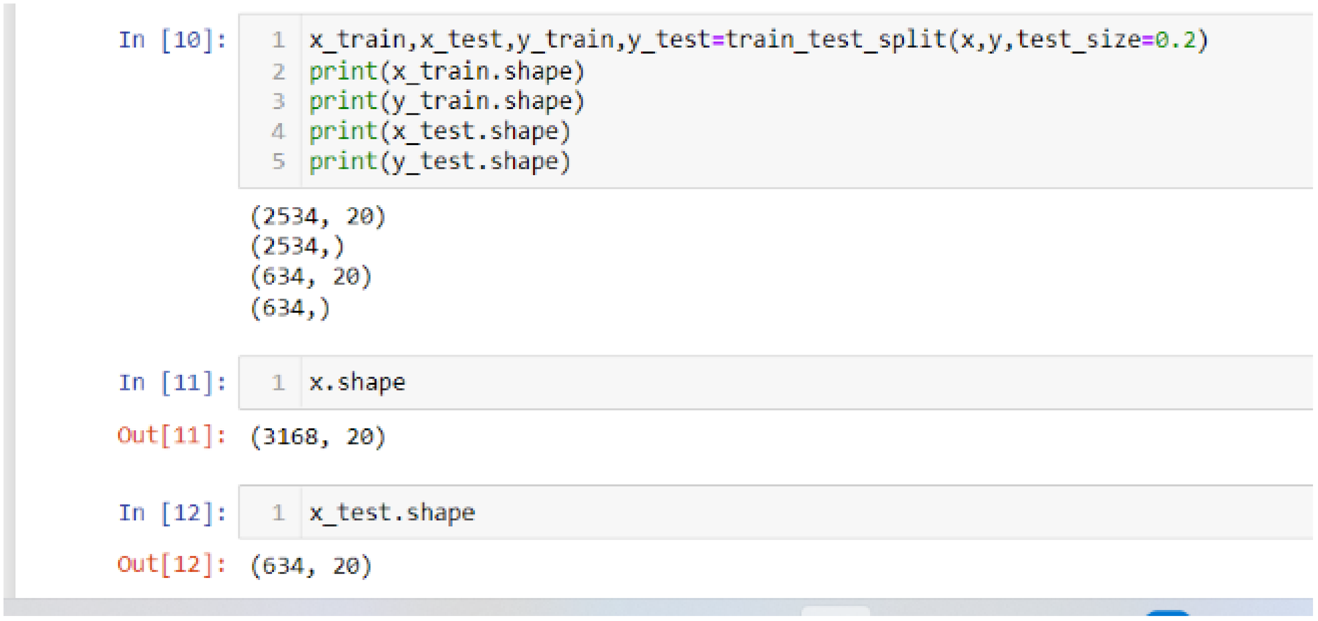
```
df.head()
```

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	meandc
0	0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.863462	274.402906	0.893369	0.491918	...	0.059781	0.084279	0.015702	0.275862	0.0078
1	0.066009	0.067310	0.040229	0.019414	0.092666	0.073252	22.423285	634.613855	0.892193	0.513724	...	0.066009	0.107937	0.015826	0.250000	0.0090
2	0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.757155	1024.927705	0.846389	0.478905	...	0.077316	0.098706	0.015656	0.271186	0.0079
3	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322	0.727232	...	0.151228	0.088965	0.017798	0.250000	0.2014
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.971955	0.783568	...	0.135120	0.106398	0.016931	0.266667	0.7128

5 rows × 21 columns

Splitting Of Dataset: -

Then the dataset was splitted to test dataset and train dataset.



```
In [10]:  x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
          print(x_train.shape)
          print(y_train.shape)
          print(x_test.shape)
          print(y_test.shape)

          (2534, 20)
          (2534,)
          (634, 20)
          (634,)

In [11]:  x.shape

Out[11]:  (3168, 20)

In [12]:  x_test.shape

Out[12]:  (634, 20)
```

Model Application: -

Then we applied different models of machine learning to our dataset to check the accuracy and to get the confusion matrix so that we will get the best model suited foe our project. Different models we used in our projects include: -

1)Random Forest Classifier

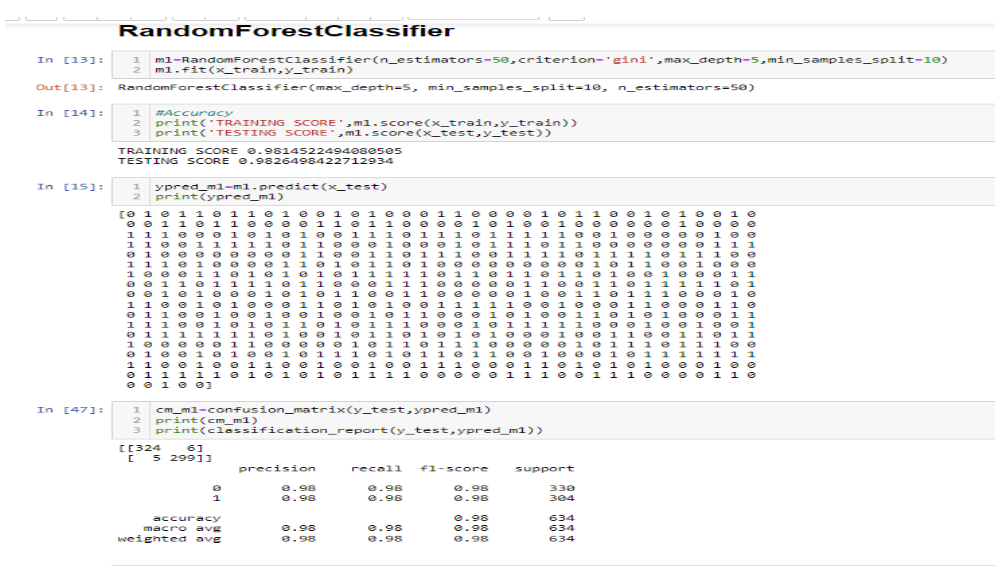
2)Decision Tree Classifier

3)KNeighbour Classifier

4)SVM

5)Logistic Regression

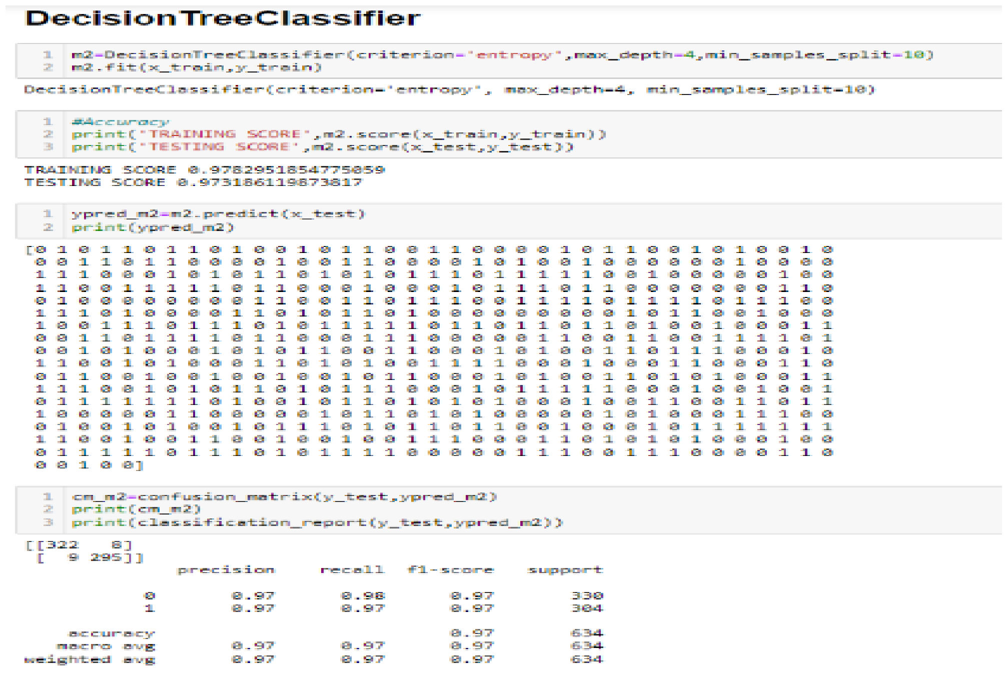
Random Forest Classifier



```
RandomForestClassifier

In [13]:  m1=RandomForestClassifier(n_estimators=50,criterion='gini',max_depth=5,min_samples_split=10)
          m1.fit(x_train,y_train)
Out[13]: RandomForestClassifier(max_depth=5, min_samples_split=10, n_estimators=50)

In [14]:  #Accuracy
          print('TRAINING SCORE',m1.score(x_train,y_train))
          print('TESTING SCORE',m1.score(x_test,y_test))

TRAINING SCORE 0.9814522494080505
TESTING SCORE 0.9826498422712934

In [15]:  ypred_m1=m1.predict(x_test)
          print(ypred_m1)

[0 1 0 1 1 0 1 1 0 1 0 0 1 0 1 0 0 0 1 1 0 0 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0
 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0
 1 1 1 0 0 0 1 0 1 0 1 0 0 1 1 1 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0
 1 1 0 0 1 1 1 1 1 0 1 1 0 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 0 0 0 0 0 0 1 1 1
 0 1 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 1 1 1 0 0 1 1 1 1 0 1 1 1 1 0 1 1 1 0 0
 1 1 1 0 1 0 0 0 0 1 1 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 0 0 0
 1 0 0 0 1 1 0 1 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 1 0 1 1 0 1 0 0 1 0 0 0 1 1
 0 0 1 1 0 1 1 1 1 0 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0 1 1 0 1 1 1 1 1 0 1
 0 0 1 0 1 0 0 0 1 0 1 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 0 0 0 1 0
 1 1 0 0 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0 0 0 1 1 0 0 0 1 1 0
 0 1 1 0 0 1 0 0 1 0 0 1 0 0 1 0 1 1 0 0 0 1 0 1 0 0 1 1 0 1 0 1 0 0 0 1 1
 1 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 0 0 0 1 0 1 1 1 1 1 0 0 0 1 0 0 1 0 0 1
 0 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 0 1 0 1 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 1 1
 1 0 0 0 0 0 1 1 0 0 0 0 0 1 0 1 1 0 1 1 1 0 0 0 0 0 1 0 1 1 1 0 1 1 1 0 0
 0 1 0 0 1 0 1 0 0 1 0 1 1 1 0 1 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1 1 1 1 1
 1 1 0 0 1 0 0 1 1 0 0 1 0 0 1 0 0 1 1 1 0 0 0 1 1 0 1 0 1 0 1 0 0 0 1 0 0
 0 1 1 1 1 1 0 1 0 1 0 1 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 0 0 1 1 0
 0 0 1 0 0]

In [47]:  cm_m1=confusion_matrix(y_test,ypred_m1)
          print(cm_m1)
          print(classification_report(y_test,ypred_m1))

[[324   6]
 [  5 299]]
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       330
           1       0.98      0.98      0.98       304

    accuracy                           0.98       634
   macro avg       0.98      0.98      0.98       634
weighted avg       0.98      0.98      0.98       634
```

We applied a random forest model. This achieves an accuracy of 98% on the training set and 98% on the test set.

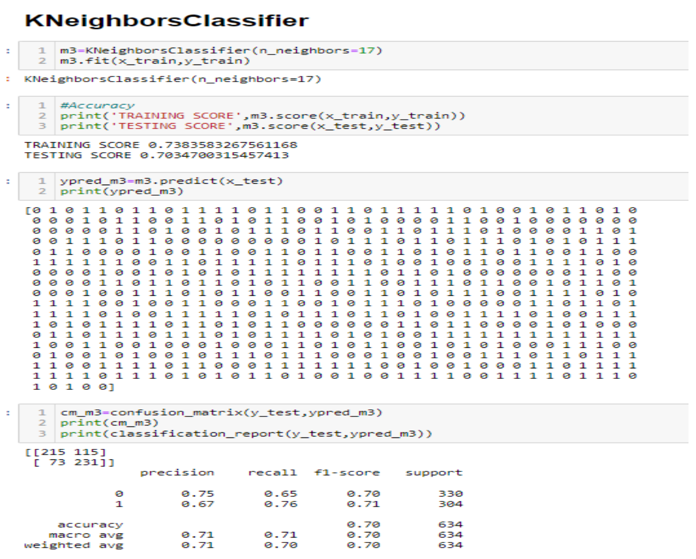
Decision Tree Classifier



```
DecisionTreeClassifier

m2=DecisionTreeClassifier(criterion="entropy",max_depth=4,min_samples_split=10)
m2.fit(x_train,y_train)
DecisionTreeClassifier(criterion='entropy', max_depth=4, min_samples_split=10)

#Accuracy
print('TRAINING SCORE',m2.score(x_train,y_train))
print('TESTING SCORE',m2.score(x_test,y_test))

TRAINING SCORE 0.9782951854775859
TESTING SCORE 0.973186119873817

ypred_m2=m2.predict(x_test)
print(ypred_m2)

[0 1 0 1 1 0 1 1 0 1 0 0 1 0 1 1 0 0 1 1 0 0 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0
 0 0 1 1 0 1 1 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0
 1 1 1 0 0 0 1 0 1 0 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0
 1 1 0 0 1 1 1 1 1 0 1 1 0 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 0 0 0 0 0 0 1 1 0
 0 1 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 1 1 1 0 0 1 1 1 1 0 1 1 1 1 0 1 1 1 0 0
 1 1 1 0 1 0 0 0 0 1 1 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 0 0 0
 1 0 0 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 0 1 1 0 1 1 0 1 1 0 1 0 0 1 0 0 0 1 1
 0 0 1 1 0 1 1 1 1 0 1 1 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1
 0 0 1 0 1 0 0 0 1 0 1 0 1 1 0 0 1 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 1 0
 1 1 0 0 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1 0 0 0 1 0 0 0 1 1 0 0 0 1 1 0
 0 1 1 0 0 1 0 0 1 0 0 1 0 0 1 0 1 1 0 0 0 1 0 1 0 0 1 1 0 1 0 1 0 0 0 1 1
 1 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 0 0 0 1 0 1 1 1 1 1 0 0 0 1 0 0 1 0 0 1
 0 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 0 1 0 1 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 1 1
 1 0 0 0 0 0 1 1 0 0 0 0 0 1 0 1 1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 1 1 1 0 0
 0 1 0 0 1 0 1 0 0 1 0 1 1 1 0 1 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1 1 1 1 1
 1 1 0 0 1 0 0 1 1 0 0 1 0 0 1 0 0 1 1 1 0 0 0 1 1 0 1 0 1 0 1 0 0 0 1 0 0
 0 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 0 0 1 1 0
 0 0 1 0 0]

cm_m2=confusion_matrix(y_test,ypred_m2)
print(cm_m2)
print(classification_report(y_test,ypred_m2))

[[322   8]
 [  9 295]]
              precision    recall  f1-score   support

           0       0.97      0.98      0.97       330
           1       0.97      0.97      0.97       304

    accuracy                           0.97       634
   macro avg       0.97      0.97      0.97       634
weighted avg       0.97      0.97      0.97       634
```

We applied a Decision Tree Classifier model. This achieves an accuracy of 97% on the training set and 97% on the test set, which is slightly less than Random Forest Classifier model. So, this model can be considered for deployment.

KNeighbour Classifier



KNeighborsClassifier

```
m3=KNeighborsClassifier(n_neighbors=17)
m3.fit(x_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=17)
```

```
#Accuracy
print('TRAINING SCORE',m3.score(x_train,y_train))
print('TESTING SCORE',m3.score(x_test,y_test))
```

```
TRAINING SCORE 0.7383583267561168
TESTING SCORE 0.7034700315457413
```

```
ypred_m3=m3.predict(x_test)
print(ypred_m3)
```

```
[0 1 0 1 1 0 1 1 0 1 1 1 1 0 1 1 0 0 1 1 0 1 1 1 1 1 0 1 0 0 1 0 1 1 0 1 0
 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 1 0 0 1 0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 1 1 0 1 0 0 1 0 1 1 1 0 1 1 0 0 1 1 0 1 1 1 0 1 0 0 0 0 1 1 0 1
 0 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 1 1 0 1 1 1 0 1 0 1 0 1 1 1
 0 1 1 0 0 0 0 1 0 0 1 1 0 0 1 1 0 1 1 0 0 1 1 0 1 0 1 1 0 1 1 0 0 1 1 0 0
 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 1 1 1 0 1 0
 0 0 0 0 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 0 0 0 0 0 0 0 1 1 0 0
 0 0 0 0 1 1 0 1 1 0 1 1 0 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 0 0 1 0 1 1 0 1
 0 0 0 1 0 0 1 1 1 0 1 0 1 1 0 0 1 1 0 0 1 1 0 1 0 1 1 1 0 0 1 1 1 1 0 1 0
 1 1 1 1 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 0 1 0 0 0 0 0 1 1 0 1 1 0 1
 1 1 1 1 0 1 0 0 1 1 1 1 1 0 1 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1
 1 0 1 0 1 1 1 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 1 0 0 0
 0 1 1 0 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 0 1 1 0 0 1 0 0 0 1 0 0 0 1 1 0 1 0 1 1 0 0 1 0 1 0 1 0 0 0 1 1 1 0 0
 0 1 0 0 1 0 1 0 0 1 0 1 1 1 0 1 1 1 0 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0 1 1 1
 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 1 1
 1 1 1 1 0 1 1 1 0 1 0 1 0 1 1 0 1 0 0 1 0 0 1 1 1 1 0 0 1 1 1 1 0 1 1 1 0
 1 0 1 0 0]
```

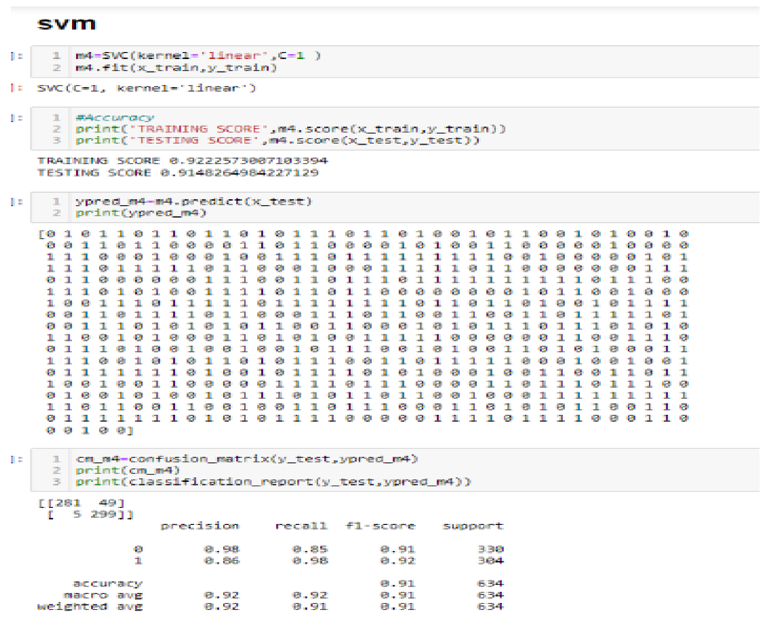
```
cm_m3=confusion_matrix(y_test,ypred_m3)
print(cm_m3)
print(classification_report(y_test,ypred_m3))
```

```
[[215 115]
 [ 73 231]]
              precision    recall  f1-score   support

           0       0.75      0.65      0.70       330
           1       0.67      0.76      0.71       304

    accuracy                           0.70       634
   macro avg       0.71      0.71      0.70       634
weighted avg       0.71      0.70      0.70       634
```

We applied a random forest model. This achieves an accuracy of 73% on the training set and 70% on the test set, which is much less than Random Forest Classifier model and Decision Tree Classifier model. So, this can't be the best suited model.

SVM

Our next model is a support vector machine, tuned with the best values for cost and gamma. To determine the best fit for an SVM model, the model was initially run with default parameters. A plot of the SVM error rate is then printed, with the darkest shades of blue indicating the best (ie., lowest) error rates. This is the best place to choose a cost and gamma value. You can fine-tune the SVM by narrowing in on the darkest blue range and performing further tuning. This essentially focuses in on the section, yielding a finer value for cost and gamma, and thus, a lower error rate and higher accuracy.



svm

```
m4=SVC(kernel='linear',C=1 )
m4.fit(x_train,y_train)
```

```
SVC(C=1, kernel='linear')
```

```
#Accuracy
print('TRAINING SCORE',m4.score(x_train,y_train))
print('TESTING SCORE',m4.score(x_test,y_test))
```

```
TRAINING SCORE 0.9222573007103394
TESTING SCORE 0.9148264984227129
```

```
ypred_m4=m4.predict(x_test)
print(ypred_m4)
```

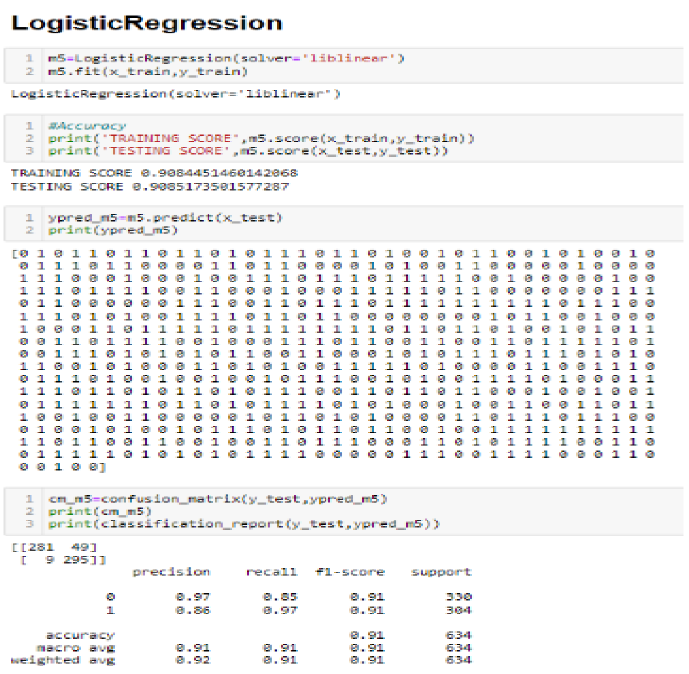
```
[0 1 0 1 1 0 1 1 0 1 1 0 1 0 1 1 1 0 1 1 0 1 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0
 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0
 1 1 1 0 0 0 1 0 0 0 1 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 1
 1 1 1 0 1 1 1 1 1 0 1 1 0 0 0 1 0 0 0 1 1 1 1 1 0 1 1 0 0 0 0 0 0 0 1 1 1
 0 1 1 0 0 0 0 0 0 1 1 1 0 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0
 1 1 1 0 1 0 1 0 0 1 1 1 1 0 1 1 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 0 0 0
 1 0 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 0 0 1 0 1 1 1 1
 0 0 1 1 0 1 1 1 1 0 1 1 0 0 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 0 1 1 1 1 1 0 1
 0 0 1 1 1 0 1 0 1 0 1 0 1 1 0 0 1 1 0 0 0 1 0 1 0 1 1 1 0 1 1 1 0 1 0 1 0
 1 1 0 0 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 1 1 0 0 1 1 1 0
 0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 1 0 1 0 1 0 0 0 1 1
 1 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 0 0 1 1 0 1 1 1 1 1 0 0 0 1 0 0 1 0 0 1
 0 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 0 1 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 1 1
 1 0 0 1 0 0 1 1 0 0 0 0 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1 0 1 1 1 0 1 1 1 0 0
 0 1 0 0 1 0 1 0 0 1 0 1 1 1 0 1 0 1 1 0 1 1 0 0 1 0 0 0 1 1 1 1 1 1 1 1 1
 1 1 0 1 1 0 0 1 1 0 0 1 0 0 1 1 0 1 1 1 0 0 0 1 1 0 1 0 1 0 1 1 0 0 1 1 0
 0 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0 1 1 0
 0 0 1 0 0]
```

```
cm_m4=confusion_matrix(y_test,ypred_m4)
print(cm_m4)
print(classification_report(y_test,ypred_m4))
```

```
[[281  49]
 [  5 299]]
              precision    recall  f1-score   support

           0       0.98      0.85      0.91       330
           1       0.86      0.98      0.92       304

    accuracy                           0.91       634
   macro avg       0.92      0.92      0.91       634
weighted avg       0.92      0.91      0.91       634
```

SVM achieves an accuracy of 92% on the training set and 91% on the test set. Although this is just slightly less accuracy than the random forest model, we'll still be able to use the SVM in a stacked ensemble model.

Logistic Regression



LogisticRegression

```
m5=LogisticRegression(solver='liblinear')
m5.fit(x_train,y_train)
```

```
LogisticRegression(solver='liblinear')
```

```
#Accuracy
print('TRAINING SCORE',m5.score(x_train,y_train))
print('TESTING SCORE',m5.score(x_test,y_test))
```

```
TRAINING SCORE 0.9084451460142068
TESTING SCORE 0.9085173501577287
```

```
ypred_m5=m5.predict(x_test)
print(ypred_m5)
```

```
[0 1 0 1 1 0 1 1 0 1 1 0 1 0 1 1 1 0 1 1 0 1 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0
 0 1 1 1 0 1 1 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0
 1 1 1 0 0 0 1 0 0 0 1 0 0 1 1 1 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0
 1 1 1 0 1 1 1 1 0 0 1 1 0 0 0 1 0 0 0 1 1 1 1 1 0 1 1 0 0 0 0 0 0 0 1 1 1
 0 1 1 0 0 0 0 0 0 1 1 1 0 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0
 1 1 1 0 1 0 1 0 0 1 1 1 1 0 1 1 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 0 0 0
 1 0 0 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 0 0 1 0 1 0 1 1
 0 0 1 1 0 1 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 0 1 1 1 1 1 0 1
 0 0 1 1 1 0 1 0 1 0 1 0 1 1 0 0 1 1 0 0 0 1 0 1 0 1 1 1 0 1 1 1 0 1 0 1 0
 1 1 0 0 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 0 1 1 1 0
 0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 1 1 1 0 1 0 0 0 1 1
 1 1 1 0 1 1 0 1 0 1 1 0 1 0 1 1 1 0 0 1 1 0 1 1 0 1 1 0 0 0 1 0 0 1 0 0 1
 0 1 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 1 1
 1 0 0 1 0 0 1 1 0 0 0 0 0 1 0 1 1 0 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 1 0 0
 0 1 0 0 1 0 1 0 0 1 0 1 1 1 0 1 0 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1 1
 1 1 0 1 1 0 0 1 1 0 0 1 0 0 1 1 0 1 1 1 0 0 0 1 1 0 1 0 1 1 1 1 0 0 1 1 0
 0 1 1 1 1 1 0 1 0 1 0 1 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 1 1 1 1 0 0 0 1 1 0
 0 0 1 0 0]
```

```
cm_m5=confusion_matrix(y_test,ypred_m5)
print(cm_m5)
print(classification_report(y_test,ypred_m5))
```

```
[[281  49]
 [  9 295]]
              precision    recall  f1-score   support

           0       0.97      0.85      0.91       330
           1       0.86      0.97      0.91       304

    accuracy                           0.91       634
   macro avg       0.91      0.91      0.91       634
weighted avg       0.92      0.91      0.91       634
```

We applied a Logistic Regression Model. This achieves an accuracy of 90% on the training set and 90% on the test set.

So, according to our model, we concluded that Random Forest Classifier has the best output.

CONCLUSION: -

We've seen in the above models how the accuracy of classifying male or female voices was increased by including all available acoustic properties of the voices and speech. Determining a male or female voice does, indeed, utilize more than a simple measurement of average frequency. To demonstrate this, several new voice samples were applied to the model, each using different intonation. For example, the first voice sample used flat or dropping frequency at the end of sentences. A second sample used a rising frequency at the end of sentences. When combined with voice frequency and pitch (ie., male vs female voice range), this difference in lowering or rising of the voice at the end of a sentence would occasionally signify the difference in a classification of male or female. This is especially true when the male and female voice samples were within a similar, androgynous, frequency range.

The above described type of classification makes sense, as male and female speakers will often use changing intonations to express parts of speech. Female voices tend to rise and fall more dramatically than their male counterparts, which might account for this difference.

A larger dataset of voice samples from both male and female subjects could help minimize incorrect classifications from intonation.