AGE AND/OR GENDER RECOGNITION THROUGH VOICE IN PYTHON

GROUP 19:

MUSKAN RATH (23PGAI0019)

GENDER RECOGNITION

ABSTRACT

Gender identification is a procedure that uses speech cues to determine a person's gender. This technique entails analyzing captured speech to determine acoustic properties such as frequency and interquartile range. It is predicated on the idea that men and women talk differently, which can be quantified through acoustic research. Machine learning models are taught using features extracted from voice signals to identify individuals as masculine or female. Speech recognition, security systems, and forensic inquiries all benefit from gender identification systems. However, these systems' precision can be influenced by variables such as age, speech, and accent. When creating and deploying gender recognition systems, it is critical to consider the ethical consequences. These methods have the ability to perpetuate harmful assumptions and gender biases. As a result, approaching the creation and execution of gender recognition algorithms with a critical perspective is essential.

Introduction:

Human speech includes paralinguistic information that can be used in a variety of voice recognition apps. One of the most important parts of speech recognition is determining the speaker's gender, which can be difficult due to variables such as age, dialect, and accent. Accurate gender identification has useful uses in a variety of areas, including speech processing, security systems, and investigations. Despite the complexities involved, substantial breakthroughs in gender recognition technology have been made, making it a quickly evolving area of study.

Problem Statement:

The issue statement is to identify an individual's gender based on their acoustic vocal characteristics. This is possible with machine learning classification methods, especially supervised learning, where the real output is known and can be compared to predictions.

Techniques/Technologies used:

Data Science tools like pandas, matplotlib, seaborn, numpy, and others. Sklearn, a machine learning framework, was used with many secondary models such as model_selection, classifiers and metrics, and so on. For execution, a Jupyter notepad was used. The dataset is titled "Gender Recognition by Voice," and the URL to it is as follows: https://www.kaggle.com/datasets/primaryobjects/voicegender

Dataset used:

It is made up of 3168 recorded speech excerpts from male and female individuals.

Acoustic analysis was performed on the speech recordings in R using the seewave and tuner packages, with an analysed frequency range of 0Hz-280Hz. (human vocal range).

The provider performed the above acoustic study.

Features in the Dataset:

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: Interquartile range (in kHz)

skew: skewness

kurt: kurtosis

spe.ent: spectral entropy

sfm: spectral flatness

mode: Mode frequency

centroid: frequency centroid

meanfun: average of fundamental frequency measured across acoustic signal

minfun: minimum fundamental frequency measured across fundamental 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

label: male or female

Reading the data:

```
In [55]: # reading the dataframe
         dataframe = pd.read_csv('voice.csv')

In [56]: # Checking first 10 rows in the dataframe
         dataframe.head(10)
```

Out[56]:

	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	0.132786	0.079557	0.119090	0.067958	0.209592	0.141634	1.932562	8.308895	0.963181	0.738307	...	0.132786	0.110132	0.017112	0.253968	0.2982
6	0.150762	0.074463	0.160106	0.092899	0.205718	0.112819	1.530643	5.987498	0.967573	0.762638	...	0.150762	0.105945	0.026230	0.266667	0.4796
7	0.160514	0.076767	0.144337	0.110532	0.231962	0.121430	1.397156	4.766611	0.959255	0.719858	...	0.160514	0.093052	0.017758	0.144144	0.3013
8	0.142239	0.078018	0.138587	0.088206	0.208587	0.120381	1.099746	4.070284	0.970723	0.770992	...	0.142239	0.096729	0.017957	0.250000	0.3364
9	0.134329	0.080350	0.121451	0.075580	0.201957	0.126377	1.190368	4.787310	0.975246	0.804505	...	0.134329	0.105881	0.019300	0.262295	0.3403

10 rows × 21 columns

Upon closing inspecting the data, we observed that the data didn't have null values.

They were already pre-processed by the source and all the data values are normalized.

```
In [7]: # Checking if there are any null or missing values
        dataframe.isnull().sum()
```

```
Out[7]: meanfreq    0
        sd          0
        median      0
        Q25         0
        Q75         0
        IQR         0
        skew        0
        kurt        0
        sp.ent      0
        sfm         0
        mode        0
        centroid    0
        meanfun     0
        minfun      0
        maxfun      0
        meandom     0
        mindom      0
        maxdom      0
        dfrange     0
        modindx     0
        label       0
        dtype: int64
```

Exploratory Data Analysis:

The dataset has 3168 rows and 21 features (columns). All of the attributes mentioned below are relevant ones.

```
: # Printing different columns
  print(dataframe.columns)
```

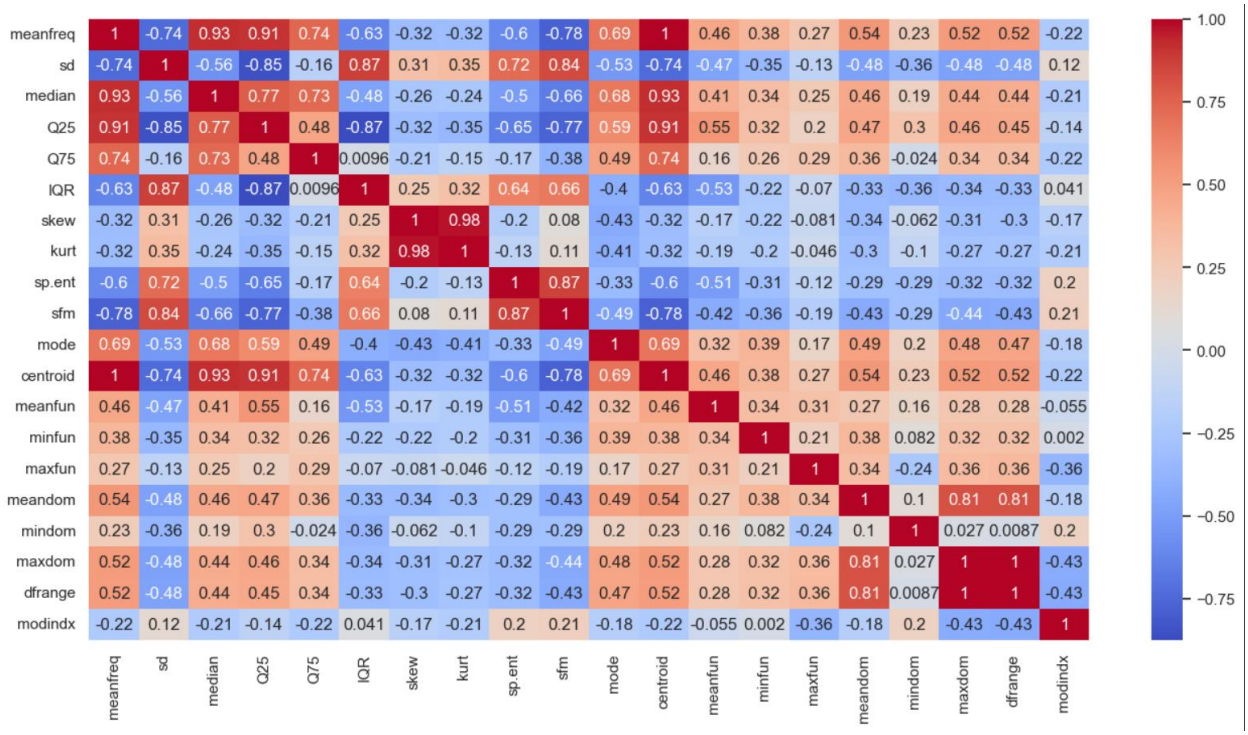
```
Index(['meanfreq', 'sd', 'median', 'Q25', 'Q75', 'IQR', 'skew', 'kurt',
       'sp.ent', 'sfm', 'mode', 'centroid', 'meanfun', 'minfun', 'maxfun',
       'meandom', 'mindom', 'maxdom', 'dfrange', 'modindx', 'label'],
      dtype='object')
```

In the age detection portion of the project, we had to predict whether the voice is that of male or female based on the features. So, the given problem is a classification problem and machine learning models like SVM, Random Forest, Decision Tree and Naïve Bayes were used for the same.

For the column 'label', there were in total 1584 values for male and 1584 values for female.

```
male      1584
female    1584
Name: label, dtype: int64
Count total  3168
```

The correlation between the features using heatmap of seaborn library was tried to be found:



Feature Engineering:

All the features were taken into consideration for the model.

The input to the models is:

meanfreq, sd, median, Q25, Q75, IQR, skew, kurt, sp.ent, sfm, mode, centroid, meanfun, minfun, maxfun, meandom, mindom, maxdom, dfrange, modindx

Output to the model: label

The encoding was added to the label column. The column female was given the value 0 and the column for male was given the value ‘1’.

Splitting the Data into Train and Test

The data was split into train and test in the ratio 7:3. In other words, 70% of the data was train data and 30% of the data was test data. Random state was set to an integer value so that every time the kernel restarts, the splitting is in the same random order as the previous one.

```
X_train.shape
```

```
(2217, 20)
```

```
X_test.shape
```

```
(951, 20)
```

Applying Machine Learning Models:

1) **Decision Tree Model:** It is a classification method used in data mining and machine learning in which decision trees are built using an algorithmic approach that finds ways to divide a data collection based on various circumstances.

Decision Tree

```
decision_tree = DecisionTreeClassifier(random_state=42)
decision_tree.fit(X_train, Y_train)
print("Decision Tree Classification Score: ",decision_tree.score(X_test, Y_test))
name_algorithm.append("Decision Tree")
score_algorithm.append(decision_tree.score(X_test, Y_test))
# Values predicted
pred1 = decision_tree.predict(X_test)
```

```
Decision Tree Classification Score:  0.9621451104100947
```

2) **Random Forest Model:** The random forest combines hundreds or thousands of decision trees, each of which is trained on a slightly different collection of data and splits nodes in each tree based on a restricted number of features. The random forest's ultimate forecasts are generated by averaging the predictions of each unique tree.

Random Forest

```
random_forest = RandomForestClassifier(random_state = 42)
random_forest.fit(X_train, Y_train)
print("Random Forest Classification Score ", random_forest.score(X_test, Y_test))
name_algorithm.append("Random Forest")
score_algorithm.append(random_forest.score(X_test, Y_test))
pred2 = random_forest.predict(X_test)
```

```
Random Forest Classification Score  0.982124079915878
```

3) **Support Vector Machine:** SVM is a type of supervised machine learning method that can be used to solve classification or regression issues. It transforms our data using a method known as the kernel trick and then discovers an optimum boundary between the possible

outputs based on these changes.

SVM

```
svm = SVC(random_state = 42)
svm.fit(X_train, Y_train)
print("SVM Classification Score is: ",svm.score(X_test, Y_test))
name_algorithm.append("SVM")
score_algorithm.append(svm.score(X_test, Y_test))
pred3 = svm.predict(X_test)
```

```
SVM Classification Score is:  0.6624605678233438
```

4) **Naïve Bayes Model:** It is a classification technique based on the Bayes' Theorem with an assumption of independence among predictors. A Naïve classifier, in basic words, asserts that the existence of one feature in a class is unrelated to the presence of any other feature.

Naive Bayes

```
naive_bayes = GaussianNB()
naive_bayes.fit(X_test, Y_test)
print("Naive Bayes Classification Score: ", naive_bayes.score(X_test, Y_test))
name_algorithm.append("Naive Bayes")
score_algorithm.append(naive_bayes.score(X_test, Y_test))
pred4 = naive_bayes.predict(X_test)
```

```
Naive Bayes Classification Score:  0.8706624605678234
```

We can observe that the classification score of different models being shown in the output.

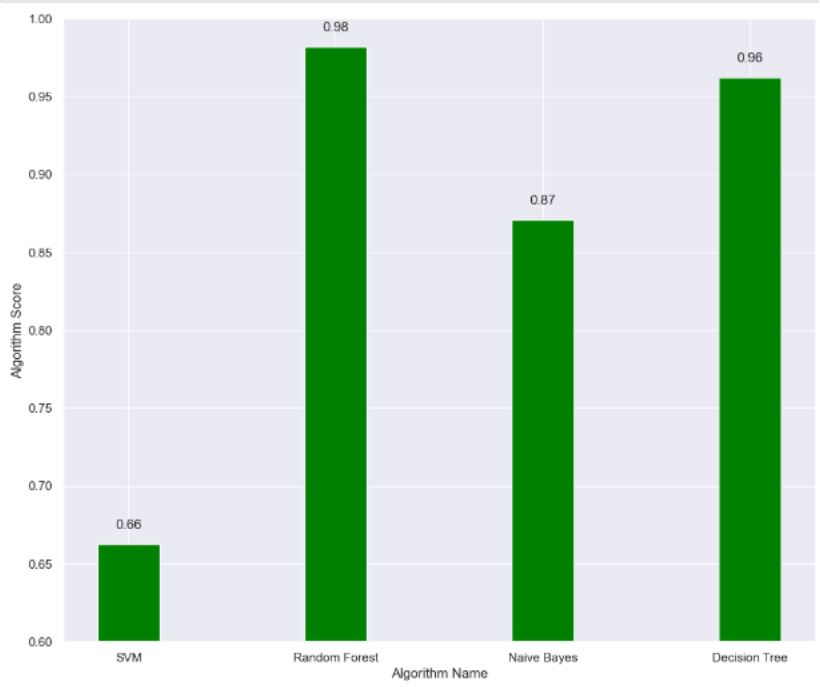
The comparison between classification scores of different models have been illustrated through the bar chart.

RESULTS (EVALUATION METRIC):-

1) EVALUATION METRIC: Classification Score Comparison of different Machine Learning models used for classification purposes (using Bar Chart)

A classification score is any score or measure that the algorithm uses (or that the user has defined) to calculate the classifier's success, how well it functions, and its predictive ability. Based on the

method and measure used, each occurrence of data is assigned its own classification score.



2) Classification Report for all the models

SVM Classification Report

```
svm_report = classification_report(Y_test, pred3)
print(svm_report)
```

	precision	recall	f1-score	support
0	0.67	0.58	0.62	452
1	0.66	0.74	0.70	499
accuracy			0.66	951
macro avg	0.66	0.66	0.66	951
weighted avg	0.66	0.66	0.66	951

Random Forest Classification Report

```
random_forest_report = classification_report(Y_test, pred2)
print(random_forest_report)
```

	precision	recall	f1-score	support
0	0.98	0.99	0.98	452
1	0.99	0.98	0.98	499
accuracy			0.98	951
macro avg	0.98	0.98	0.98	951
weighted avg	0.98	0.98	0.98	951

```
naive_bayes_report = classification_report(Y_test,pred4)
print(naive_bayes_report)
```

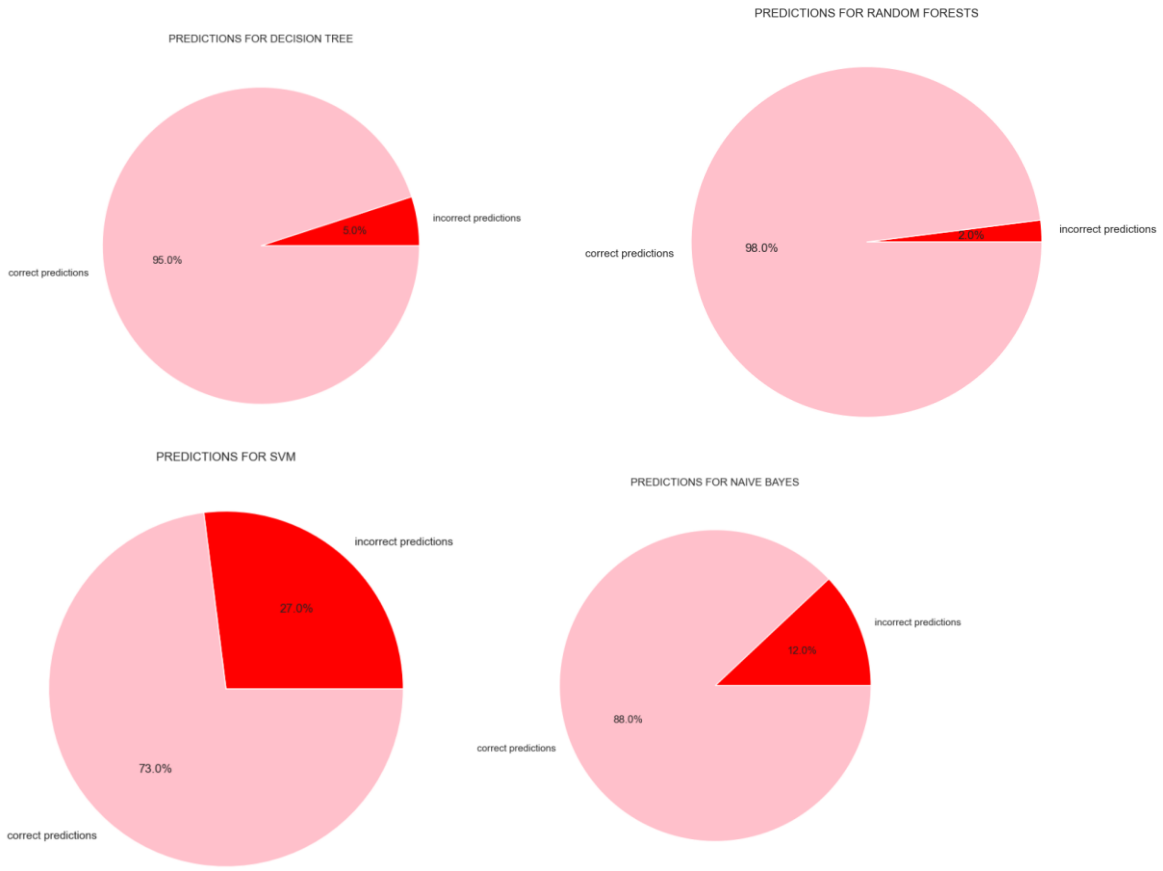
	precision	recall	f1-score	support
0	0.86	0.87	0.86	452
1	0.88	0.87	0.88	499
accuracy			0.87	951
macro avg	0.87	0.87	0.87	951
weighted avg	0.87	0.87	0.87	951

Decision Tree Classification Report

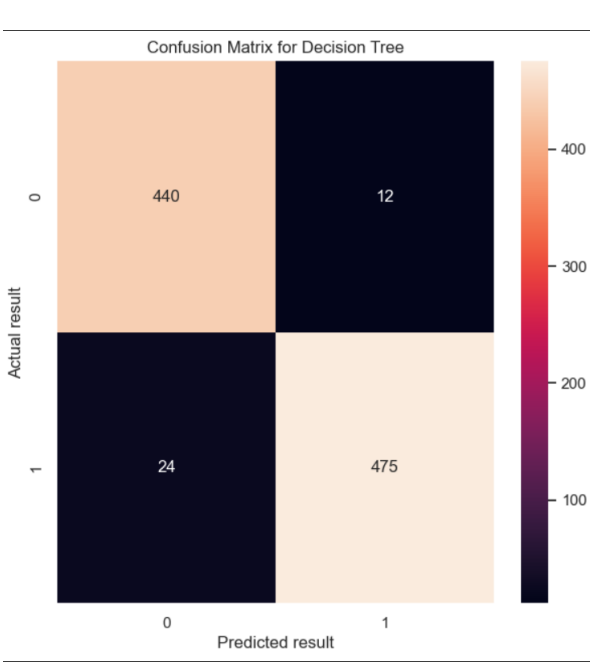
```
decision_tree_report = classification_report(Y_test, pred1)
print(decision_tree_report)
```

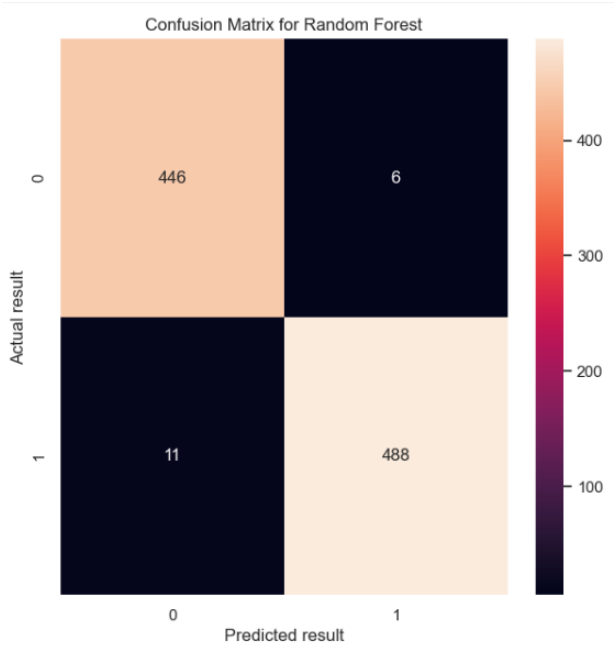
	precision	recall	f1-score	support
0	0.95	0.97	0.96	452
1	0.98	0.95	0.96	499
accuracy			0.96	951
macro avg	0.96	0.96	0.96	951
weighted avg	0.96	0.96	0.96	951

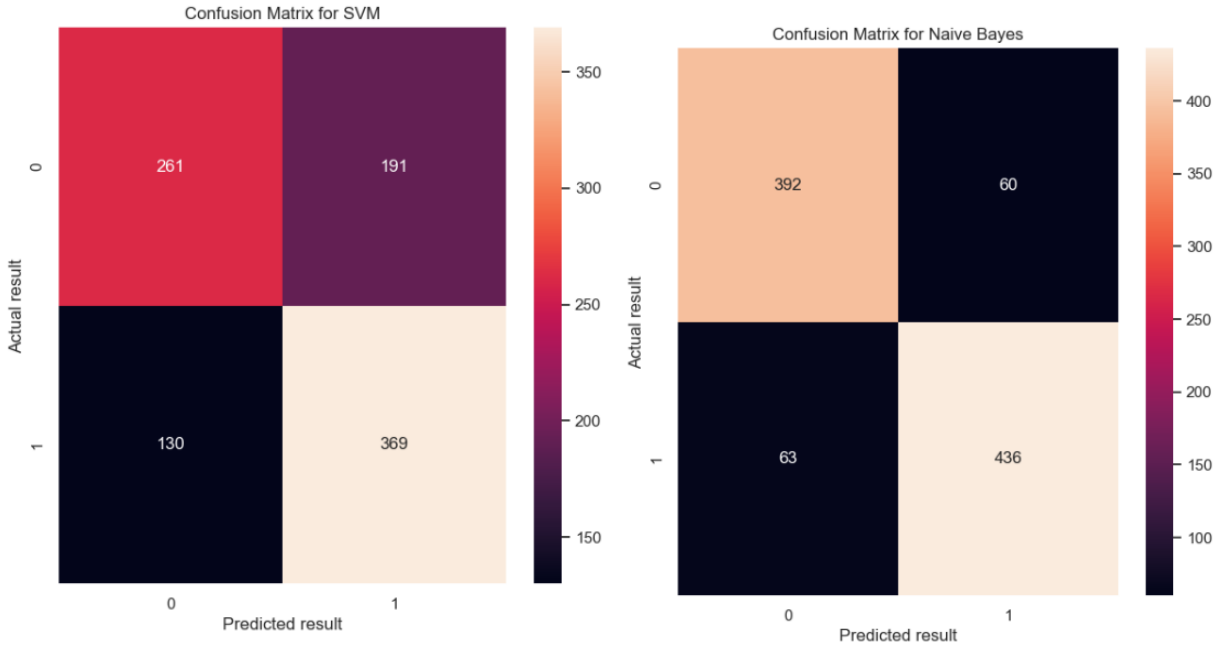
3) Pie Charts for Model Predictions:



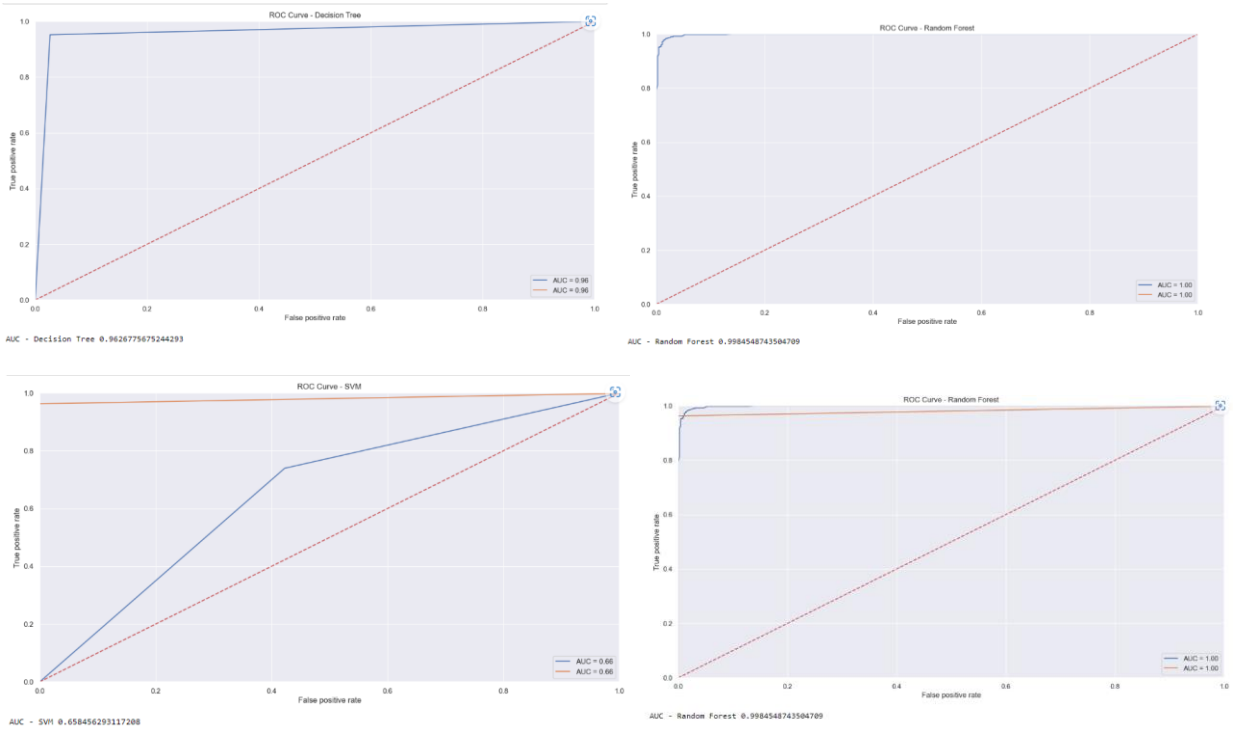
4) Confusion Matrix for all the Models



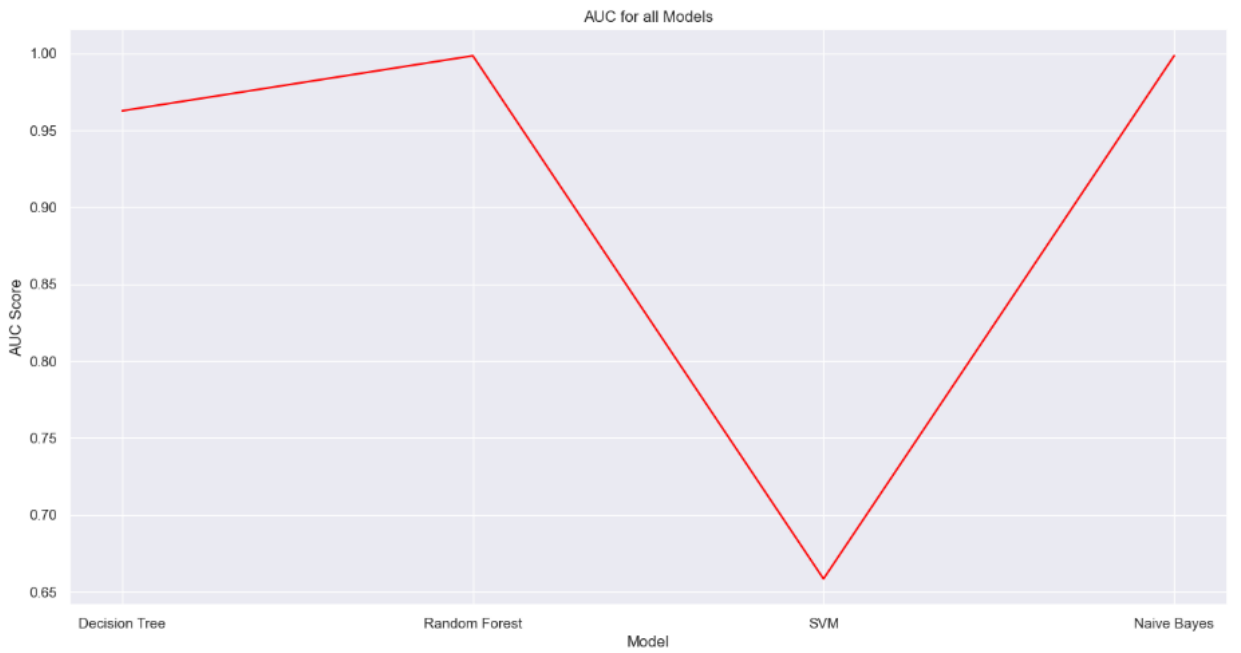




5) ROC Curves for all models:



6) AUC Line Graph



CONCLUSION:

Based on precision, F1-score, Recall Values, Prediction Pie Chart, Accuracy Score, Confusion Matrix, ROC Curve and AUC Line Graph; we can say that random forest classification is best suited for this problem where we are predicting the person as Male/Female based upon their voice signals. Support Vector Machine is least suitable based on the same analysis.