Project
On
Gender Recognition by Voice Using Different Supervised Classifier Algorithms

Table of Contents

INTRODUCTION

Speech constitutes one of the most popular and significant means for humans to communicate, express their emotions, cognitive states, and intentions to each other. In general, voice recognition system can be used for gender identification. A natural voice recognition system is the human ear. The human ear has an excellent mechanism which can efficiently distinguish the gender by voice and speech based on attributes like frequency and loud ness. In a similar way, a machine can be taught to do the same thing by choosing and incorporating the right features from voice data on a machine learning algorithms.

Gender recognition is a technique which is often utilized to determine the gender categ ory of a speaker by processing speech signals. Speech signals taken from a recorded speech can be used to acquire acoustic attributes such as duration, intensity, frequency and filtering. Some applications where gender recognition can be useful are speech emotion recognition, human to machine interaction, sorting of telephone calls by gender categorization. As technology is growing in a rapid way, machine learning is a research field which has had major developments; thus it has been widely established as a popular trend.

However, the development of an accurate prediction model for gender recognition by voice is still considered a rather difficult and challenging task. An extensive experimental analysis and pointed out the difficulties of this classification problem since speech signals are highly time-varying and have very high randomness. This is mainly due to the fact that the progress in the field has been hampered by the lack of available labeled data for efficiently training a supervised classifier. Furthermore, in order to train efficiently a classifier and be able to make accurate predictions, it often needs a large amount of labeled data.

Problem Statement

The proposed system, Gender recognition system is design with the purpose of identifying human gender by computer using voice various statistical attributes and provide a user authentication security systems to access personal data and many more. The task of identifying a human's gender by voice seems an easy task when a human identifies it. It becomes difficult when a computer has to identify it whether the voice is of male or female. A human has natural capability of identifying the difference but when it comes to computer we need to teach it by providing inputs, methodology or different training data and make it learn. In this project, the focus is on training computer to identify the gender based on input of acoustic data attributes using different Supervised Machine Learning Classifier algorithms and find the best algorithm results in terms of accuracy.

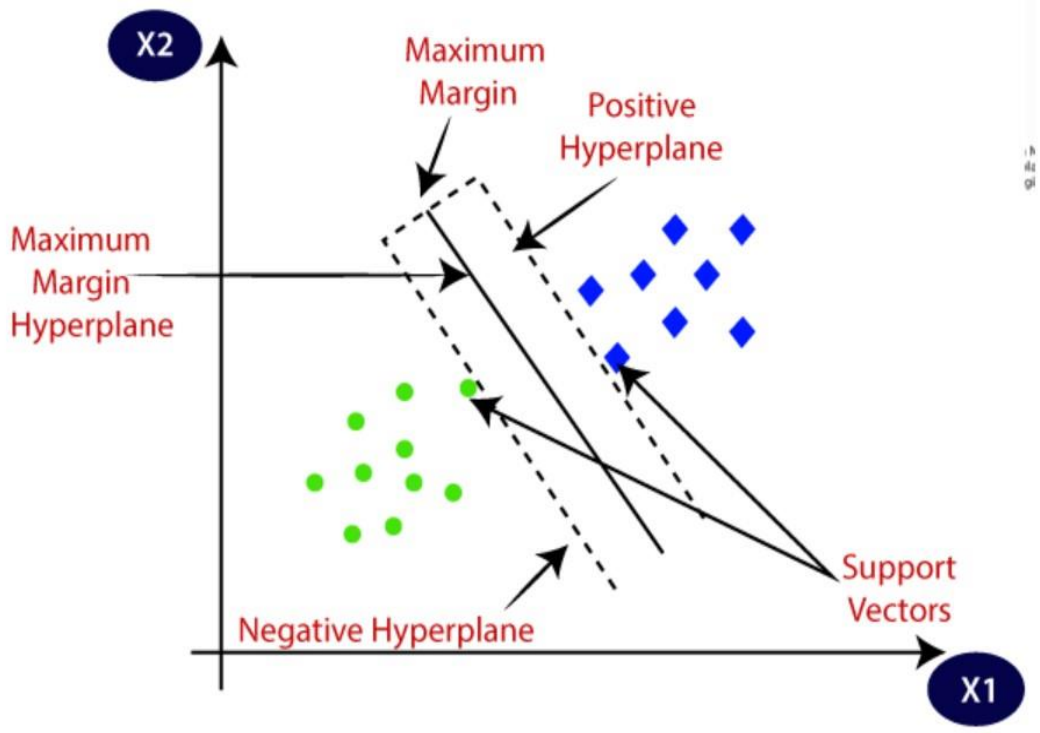
Support Vector Machine (SVM)

Support Vector Machine is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

How to work SVM ?

SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a **hyperplane.** SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as **support vectors**, and hence algorithm is termed as **Support Vector Machine**. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane.

The main goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane.**



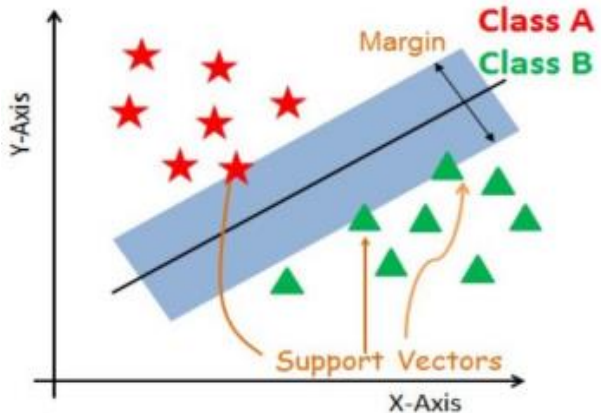
source=https://static.javatpoint.com/tutorial/machine-learning/images/support-vector-machine-algorithm.png

Types of SVM

SVM can be of two types:

Linear SVM:

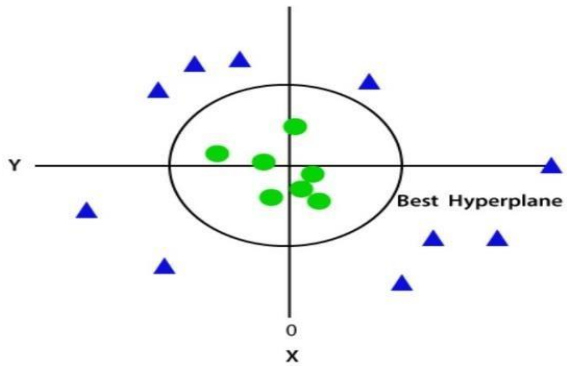
Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.



Source=https://static.javatpoint.com/tutorial/machine-learning/images/support-vector-machine-algorithm5.png

Non-linear SVM:

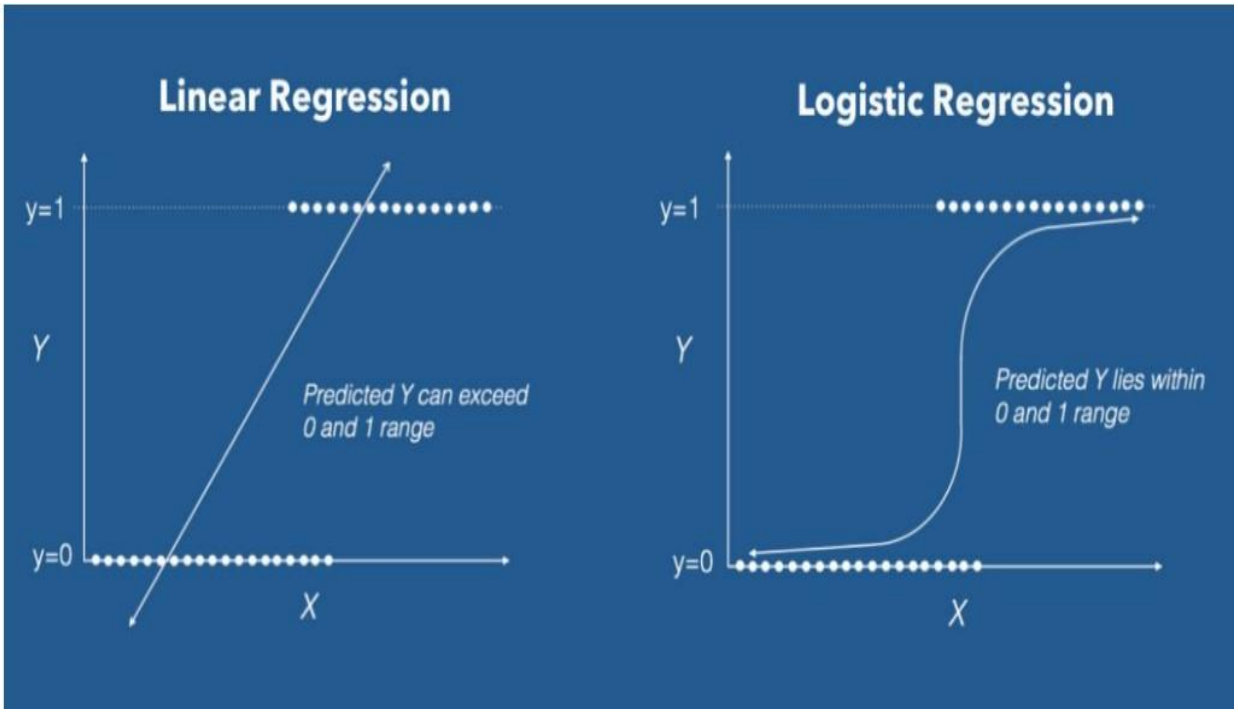
Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line. But, another question which arises is, how to classify these data. SVM algorithm has a technique called the **kernel trick**. The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space i.e. it converts not separable problem to separable problem.



Source=https://static.javatpoint.com/tutorial/machine-learning/images/support-vector-machine-algorithm9.png

Logistic Regression

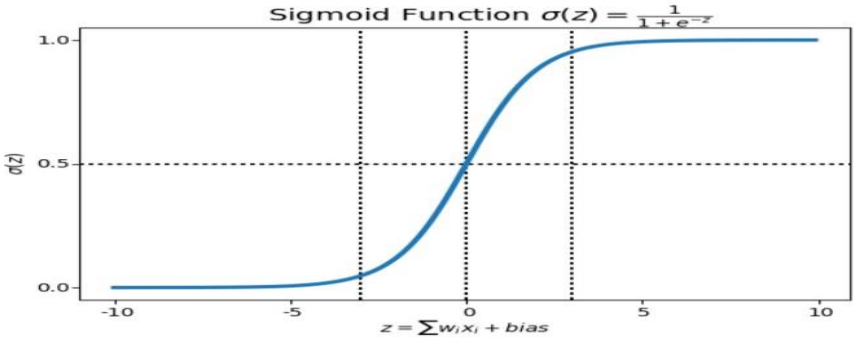
- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification.



Source=https://miro.medium.com/max/840/1*dm6ZaX5fuSmuVvM4Ds-vcg.jpeg

Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.



Sigmoid Function Graph

$$f(x) = \frac{1}{1 + e^{-(x)}}$$

Source=https://miro.medium.com/max/768/1*OUOB_YF41M-O4GgZH_F2rw.png

Hypothesis Representation

When using linear regression we used a formula of the hypothesis i.e.

$h\Theta(x) = \beta_o + \beta_1 X$

For logistic regression we are going to modify it a little bit i.e.

$\sigma(Z) = \sigma(\beta_o + \beta_1 X)$

We have expected that our hypothesis will give values between 0 and 1.

$Z = \beta_o + \beta_1 X$

$h\Theta(x) = \text{sigmoid}(Z)$

i.e. $h\Theta(x) = 1/(1 + e^{-(\beta_o + \beta_1 X)})$

$$h\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

The Hypothesis of logistic regression

Source=https://miro.medium.com/max/628/1*I59BUnPwWHMf1H-GNxgZHQ.png

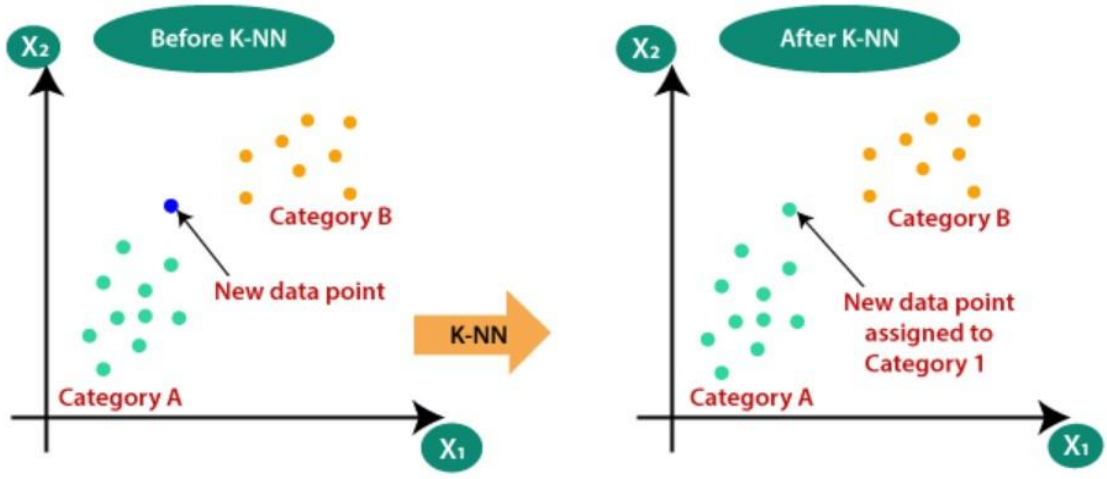
K-Nearest Neighbor(KNN)

- K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Working process of KNN:-

The K-NN working can be explained on the basis of the below algorithm:

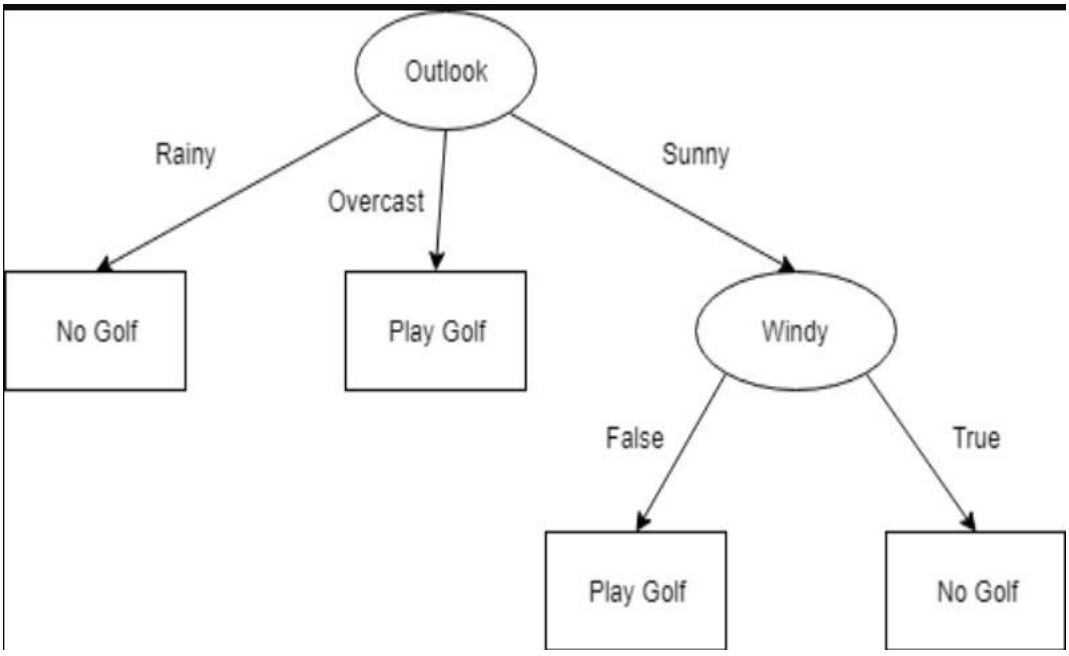
- Select the number K of the neighbors.
- Calculate the Euclidean distance of **K number of neighbors.**
- Take the K nearest neighbors as per the calculated Euclidean distance.
- Among these k neighbors, count the number of the data points in each category.
- Assign the new data points to that category for which the number of the neighbor is maximum.
- Our model is ready.



src=https://static.javatpoint.com/tutorial/machine-learning/images/k-nearest-neighbor-algorithm-for-machine-learning2.png

Decision Tree Classification Algorithm

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.



Src=https://miro.medium.com/max/637/1*Qqr1ZsGqOlSKv-dQR0N1rA.png

Working Process of Decision Tree :-

- Begin the tree with the root node, says S, which contains the complete dataset.
- Find the best attribute in the dataset using Attribute Selection Measure (ASM).
- Divide the S into subsets that contains possible values for the best attributes.
- Generate the decision tree node, which contains the best attribute.
- Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

i.) Information Gain:

Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.

It calculates how much information a feature provides us about a class.

According to the value of information gain, we split the node and build the decision tree.

A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

S= Total number of samples

P(yes)= probability of yes

P(no)= probability of no

ii.) Gini Index:

Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.

An attribute with the low Gini index should be preferred as compared to the high Gini index.

It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.

Gini index can be calculated using the below formula:

$$\text{Gini Index} = 1 - \sum_j P_j^2$$

About the Voice Dataset

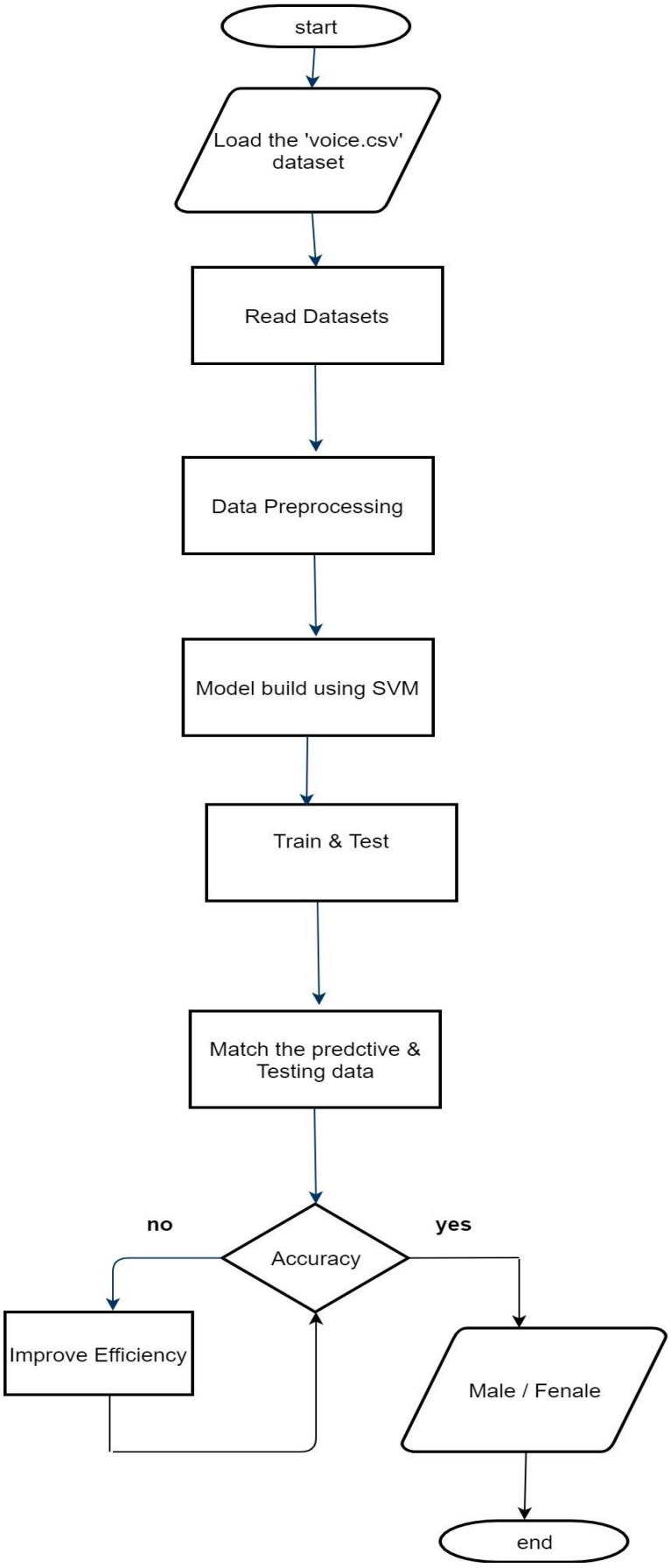
This database is identify a voice as male or female, based upon acoustic properties of the voice and speech. The voice dataset consists of 21 parameters (which are explained below), in which 20 input parameter which help to analysis and find result who one male or female according to their parameter values and 1 output parameter. In this dataset total 3168 recorded voice samples, collected from male and female speakers. The voiced speech of a typical adult male will have a fundamental frequency from 85 to 180 Hz, and that of a typical adult female from 165 to 255 Hz The voice samples are pre-processed by acoustic analysis with an analyzed frequency range of 0hz-280hz.

These following acoustic properties of each voice are measured and included within the datasets:-

1. **meanfreq**: mean frequency (in kHz)
2. **median**: median frequency (in kHz)
3. **mode**: mode frequency
4. **Standard Deviation** - By calculating the standard deviation we could get how much the values in the dominating frequency list varies from the mean value of the list. median: median frequency (in kHz)
5. **First Quartile(Q25)** - Q25 is the median of the lower half of the dominating frequency list. It means that about 25% of the frequency values in the dominating frequency list lie below Q25 and 75% of dominating frequency values larger than Q25.
6. **Third Quartile(Q75)** - Q75 is the median of the upper half of the dominating frequency list. It suggests that about 75% of the frequency values in the dominating frequency list lie below Q75 and only 25% of dominating frequency values lie above Q75.
7. **Interquartile Range(IQR)** - IQR means the frequency ranging between Q25 and Q75 and the value is the difference between Q75 and Q25 of an audio.
8. **Skewness** - Skewness shows the asymmetry of the voice frequency spectrum around the sample means. If skewness is positive, the spectrum spreads out more to the right of the mean than to the left side, and vice versa.

9. **kurt**: Kurtosis shows how much outlier-prone a distribution is. The frequency spectrum will exhibit a normal shape if kurtosis value is 3.Otherwise if the value is less than 3, the frequency spectrum will have fewer items at the center and the tails than the normal curve but with more items in the shoulders. If the frequency spectrum is having more items near the center and at the tails, and few items in the shoulders compared to a normal distribution with the same mean and variance, then the kurtosis value is greater than 3.
10. **sp.ent**: spectral entropy
11. **sfm**: spectral flatness
12. **centroid**: frequency centroid (see specprop)
13. **meanfun**: average of fundamental frequency measured across acoustic signal
14. **minfun**: minimum fundamental frequency measured across acoustic signal
15. **maxfun**: maximum fundamental frequency measured across acoustic signal
16. **meandom**: average of dominant frequency measured across acoustic signal
17. **mindom**: minimum of dominant frequency measured across acoustic signal
18. **maxdom**: maximum of dominant frequency measured across acoustic signal
19. **dfrange**: range of dominant frequency measured across acoustic signal
20. **modindx**: modulation index. Calculated as the accumulated absolute difference between adjacent measurements of fundamental frequencies divided by the frequency range
21. **label**: male or female

Flowchart


start
Load the 'voice.csv' dataset
Read Datasets
Data Preprocessing
Model build using SVM
Train & Test
Match the predctive & Testing data
Accuracy
no
yes
Improve Efficiency
Male / Fenale
end

Hardware and Software Requirement

Hardware Requirement:

- Minimum RAM: 2GB
- Hard Disk: 128GB
- Processor: Intel i3 or above

Software Requirement:

- Operating System: Windows 7 or above
- IDE: Jupyter Notebook
- Programming Language: Python 3
- Libraries: Numpy, Pandas, Matplotlib, sklearn
- Other: Anaconda

Project Implementation Screenshot

a.) Importing necessary libraries :-

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pylab as plt
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score
```

b.) Reading .csv File :-

```
In [2]: df=pd.read_csv("D:\\ML\\ML projects\\Gender Recognition\\voice.csv")

In [3]: df.head()

Out[3]:
```

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	mean
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	0.00
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	0.00
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	0.00
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	0.20
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	0.71

5 rows × 21 columns

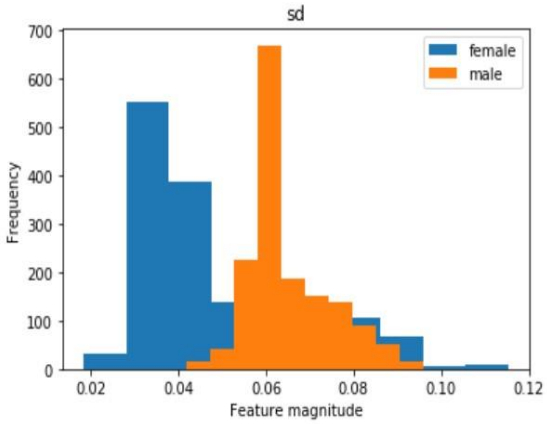
```
In [2]: df=pd.read_csv("D:\\ML\\ML projects\\Gender Recognition\\voice.csv")

In [3]: df.head()

Out[3]:
```

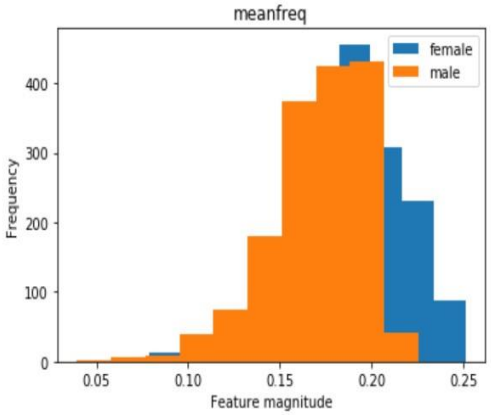
	i75	IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	meandom	mindom	maxdom	dfrange	modindx	label
	193	0.075122	12.863462	274.402906	0.893369	0.491918	...	0.059781	0.084279	0.015702	0.275862	0.007812	0.007812	0.007812	0.000000	0.000000	male
	166	0.073252	22.423285	634.613855	0.892193	0.513724	...	0.066009	0.107937	0.015826	0.250000	0.009014	0.007812	0.054688	0.046875	0.052632	male
	108	0.123207	30.757155	1024.927705	0.846389	0.478905	...	0.077316	0.098706	0.015656	0.271186	0.007990	0.007812	0.015625	0.007812	0.046512	male
	155	0.111374	1.232831	4.177296	0.963322	0.727232	...	0.151228	0.088965	0.017798	0.250000	0.201497	0.007812	0.562500	0.554688	0.247119	male
	145	0.127325	1.101174	4.333713	0.971955	0.783568	...	0.135120	0.106398	0.016931	0.266667	0.712812	0.007812	5.484375	5.476562	0.208274	male

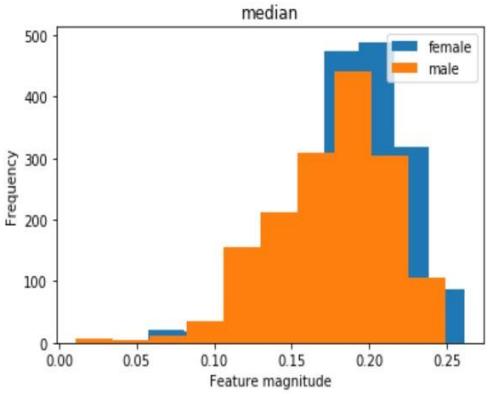
c.) Count the no. of different class data :-

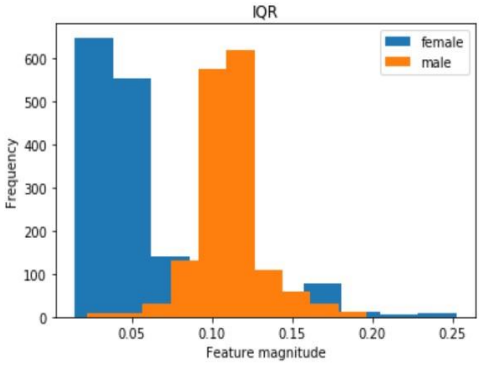
```
In [4]: label_val_cnt=df.label.value_counts()
        print(label_val_cnt)

male      1584
female    1584
Name: label, dtype: int64
```

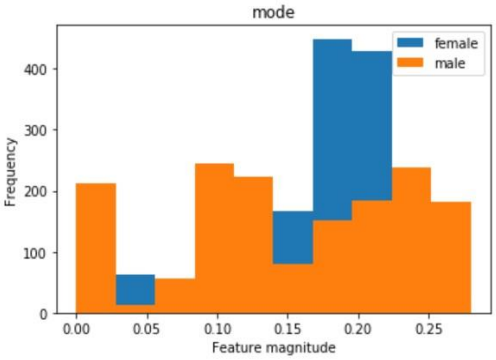
d.) Visualize the distribution of male and female in some feature columns :

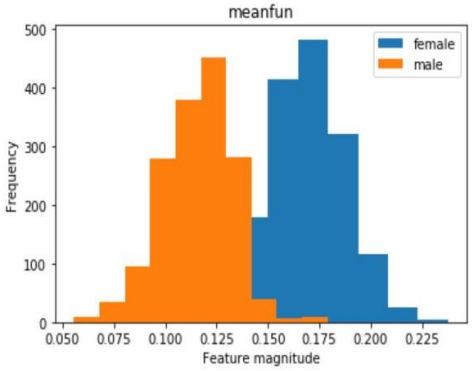


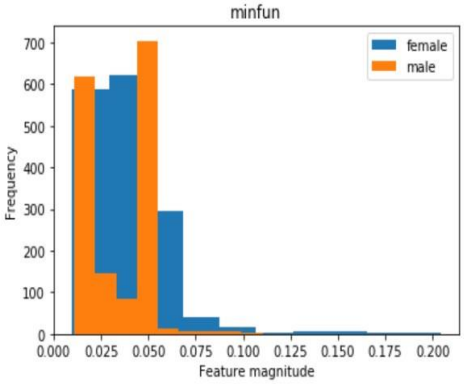


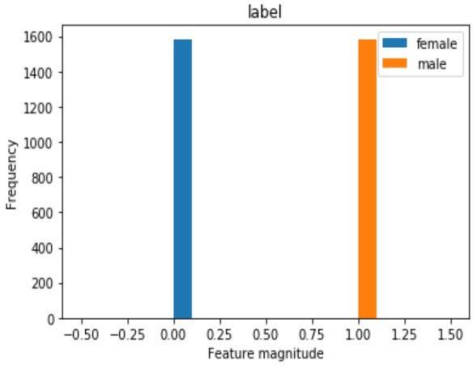












e.) Splitting the data input & output parameter:-

```
In [8]: x=df.loc[:,df.columns!='label']
        y=df.loc[:,'label']
```

f.) Replacing labels for male as 1 and for females as 0 :-

```
In [6]: #Replacing labels for male as 1 and for females as 0
        dict = {'label':{'male':1,'female':0}}
        df.replace(dict,inplace = True)
```

```
In [7]: df.head(10)
```

Out[7]:

ı75	IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	meandom	mindom	maxdom	dfrange	modindx	label
193	0.075122	12.863462	274.402906	0.893369	0.491918	...	0.059781	0.084279	0.015702	0.275862	0.007812	0.007812	0.007812	0.000000	0.000000	1
666	0.073252	22.423285	634.613855	0.892193	0.513724	...	0.066009	0.107937	0.015826	0.250000	0.009014	0.007812	0.054688	0.046875	0.052632	1
908	0.123207	30.757155	1024.927705	0.846389	0.478905	...	0.077316	0.098706	0.015656	0.271186	0.007990	0.007812	0.015625	0.007812	0.046512	1
655	0.111374	1.232831	4.177296	0.963322	0.727232	...	0.151228	0.088965	0.017798	0.250000	0.201497	0.007812	0.562500	0.554688	0.247119	1
945	0.127325	1.101174	4.333713	0.971955	0.783568	...	0.135120	0.106398	0.016931	0.266667	0.712812	0.007812	5.484375	5.476562	0.208274	1
692	0.141634	1.932562	8.308895	0.963181	0.738307	...	0.132786	0.110132	0.017112	0.253968	0.298222	0.007812	2.726562	2.718750	0.125160	1
718	0.112819	1.530643	5.987498	0.967573	0.762638	...	0.150762	0.105945	0.026230	0.266667	0.479620	0.007812	5.312500	5.304688	0.123992	1
962	0.121430	1.397156	4.766611	0.959255	0.719858	...	0.160514	0.093052	0.017758	0.144144	0.301339	0.007812	0.539062	0.531250	0.283937	1
887	0.120381	1.099746	4.070284	0.970723	0.770992	...	0.142239	0.096729	0.017957	0.250000	0.336476	0.007812	2.164062	2.156250	0.148272	1
957	0.126377	1.190368	4.787310	0.975246	0.804505	...	0.134329	0.105881	0.019300	0.262295	0.340365	0.015625	4.695312	4.679688	0.089920	1

g.) Splitting The data into Training And Testing set :-

```
In [10]: #Creating X,y and splitting the dataset into training and testing

         from sklearn.model_selection import train_test_split
         X_train,X_test,y_train,y_test=train_test_split(data_x, data_y, test_size=0.2, random_state=7)
```

h.) Scaling the features

```
In [11]: from sklearn.preprocessing import StandardScaler
         stdsc = StandardScaler()
         X_train_std = stdsc.fit_transform(X_train)
         X_test_std = stdsc.transform(X_test)
```

i.) Training the model using Logistic Regression Algorithm & also check accuracy :-

In [12]:
```
#Train logistic regression model
from sklearn.linear_model import LogisticRegression
logit = LogisticRegression()
logit.fit(X_train_std, y_train)
print("Logistic Regression")
print("Accuracy on training set: {:.3f}".format(logit.score(X_train_std, y_train)))
print("Accuracy on test set: {:.3f}".format(logit.score(X_test_std, y_test)))

y_pred_logit = logit.predict(X_test_std)
print("Predicted value: ",y_pred_logit)
```

```
Logistic Regression
Accuracy on training set: 0.976
Accuracy on test set: 0.968
Predicted value:  [0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1 0 0 0
 1 1 0 1 0 0 0 1 0 0 0 1 1 0 1 1 0 0 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1
 0 0 1 1 0 0 0 1 1 0 1 1 1 1 1 1 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0
 0 1 1 1 0 0 1 1 1 0 1 1 1 0 0 0 1 1 1 1 0 1 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0
 0 0 0 1 0 0 1 1 0 1 0 1 0 1 1 1 1 1 0 0 0 0 0 1 1 0 1 0 0 1 0 0 1 1 1 1 1 0
 1 1 1 0 1 1 0 0 0 0 1 1 1 1 1 1 0 1 1 1 0 0 1 1 0 0 0 1 0 1 0 0 1 0 1 1 0 0
 1 0 1 0 0 1 0 1 1 0 1 1 0 1 0 0 1 0 1 1 1 0 0 0 0 0 1 1 0 0 0 1 1 0 1 1 0 0
 0 0 0 0 1 0 0 1 1 1 1 0 1 0 0 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0
 1 0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 0 0 1 0 0 0 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1
 1 0 1 0 1 0 0 0 0 0 0 1 0 1 0 1 1 1 0 0 1 0 1 0 0 0 0 1 0 1 1 0 1 0 1 1 0
 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 0 1 0 0
 1 0 0 0 1 0 0 1 0 0 0 1 1 1 1 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1 1 0 0 1 1 0
 0 1 0 1 1 1 0 0 1 1 0 1 0 0 1 0 0 0 0 1 0 1 1 1 0 0 0 1 1 1 1 0 0 1 1 1 0
 1 0 1 0 1 0 1 1 1 0 1 1 1 0 0 0 1 1 0 1 0 1 1 1 1 0 1 1 0 1 1 0 0 1 1 0 0
 1 0 1 1 1 1 1 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1 1 1 0 0 0 1 1 1 0 1 1 1 1 0 1
 1 1 0 0 1 1 0 1 0 1 0 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 1 0 1 0 0 1 1 1 0
 0 0 0 0 0 1 1 0 0 0 1 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 1
 0 0 0 1 1]
```

j.) Training the model using K-Nearest Neighbours Algorithm & also check accuracy :-

In [13]:
```
# Train K Neighbors Classifier model
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier()
clf.fit(X_train_std, y_train)
print("K Neighbors Classifier")
print("Accuracy on training set: {:.3f}".format(clf.score(X_train_std, y_train)))
print("Accuracy on test set: {:.3f}".format(clf.score(X_test_std, y_test)))

y_pred_knn = clf.predict(X_test_std)
print("Predicted value: ",y_pred_knn)
```

```
K Neighbors Classifier
Accuracy on training set: 0.984
Accuracy on test set: 0.964
Predicted value:  [0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1 0 0 0
 1 1 0 1 0 0 0 1 0 0 0 1 1 0 1 1 0 0 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1
 0 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 1 1 0
 0 1 1 1 0 0 1 1 1 0 1 1 1 0 0 0 1 1 1 1 0 1 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0
 0 0 0 1 0 0 1 1 0 1 0 1 0 1 1 1 1 1 0 0 0 0 0 1 1 0 1 0 0 1 0 0 1 1 1 1 1 0
 1 1 1 0 1 0 0 0 0 0 1 1 1 1 1 1 0 1 1 1 0 0 1 1 0 0 0 1 0 1 0 0 1 0 1 1 0 0
 1 0 1 0 0 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 1 1 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 0
 0 0 0 0 1 0 0 1 1 1 1 0 1 0 0 0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0
 1 0 0 0 0 1 0 0 1 0 0 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 1 1 1 1 0 0 1 0 1 0 1
 1 0 1 1 1 0 0 0 0 0 0 1 0 1 0 1 1 1 0 0 1 0 1 0 0 0 0 1 0 1 1 0 1 0 1 1 0
 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 0 1 0 0
 1 0 0 0 1 0 0 1 0 0 0 1 1 1 1 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1 1 1 0 1 1 0
 0 1 0 1 1 1 0 0 1 1 0 1 0 0 1 0 0 0 0 1 0 1 1 1 0 0 0 1 1 1 1 0 1 1 1 1 0
 1 0 1 0 1 0 1 1 1 0 1 1 1 0 0 0 1 1 0 1 0 1 1 1 1 0 1 1 0 1 1 0 0 1 1 0 0
 1 0 1 1 1 1 1 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1 1 1 0 0 0 1 1 1 0 1 1 1 1 1 0 1
 1 1 0 0 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 0 0 0 0 1 1 1 0 1 0 0 1 1 1 0
 0 0 0 0 0 1 1 0 0 0 1 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 1
 0 0 0 1 1]
```

k.) Training the model using Decision Tree Algorithm & also check accuracy:-

In [14]:
```
#Train decision tree model
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(random_state=0,max_depth=4)
tree.fit(X_train_std, y_train)
print("Decision Tree")
print("Accuracy on training set: {:.3f}".format(tree.score(X_train_std, y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X_test_std, y_test)))

y_pred_tree = tree.predict(X_test_std)
print("Predicted value: ",y_pred_tree)
```

```
Decision Tree
Accuracy on training set: 0.981
Accuracy on test set: 0.956
Predicted value:  [0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 0 1 0 0 1 0 0 1 1 0 1 1 1 0 1 0 0 1
 1 1 0 1 0 0 0 1 0 0 0 1 1 0 1 1 0 0 1 0 0 1 0 1 1 0 1 0 1 1 0 1 1 1 1 1 0 1
 0 0 1 1 0 0 0 1 1 0 1 1 1 1 1 1 0 1 0 0 0 1 0 1 0 1 0 1 0 0 0 0 1 1 0 1 1 0
 0 1 1 1 0 0 1 1 1 0 1 1 1 0 0 0 1 1 1 1 1 0 1 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0
 0 1 0 1 0 0 1 1 0 1 0 1 0 1 1 1 1 0 0 0 0 0 0 1 1 0 1 0 0 1 0 0 1 1 1 1 1 1
 1 1 1 0 1 1 0 0 0 0 1 1 1 1 1 1 0 1 1 1 0 0 1 1 0 0 0 1 0 1 0 0 1 0 1 1 0 0
 1 0 1 0 0 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 1 0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0
 0 0 0 0 1 0 0 1 1 1 1 1 0 1 0 0 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0
 1 0 0 0 0 1 0 0 1 0 0 1 1 1 0 0 0 0 1 0 0 0 1 1 0 0 1 1 1 1 1 0 0 1 0 1 0 1
 1 0 1 1 1 0 0 0 0 0 0 1 0 1 0 1 1 1 0 0 1 0 0 0 0 0 0 1 0 1 1 0 1 0 1 1 0
 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 0 1 0 0
 1 0 0 0 1 0 0 1 0 0 0 1 1 1 1 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1 1 1 0 0 1 1 0
 0 1 0 1 1 1 0 0 1 1 0 1 0 0 1 0 0 0 0 1 0 1 1 1 0 0 0 1 1 1 1 0 1 1 1 1 1 0
 1 0 1 0 1 0 1 1 1 0 1 1 1 0 0 0 1 1 0 1 0 1 1 1 1 0 1 1 0 1 1 0 0 1 1 0 0
 1 0 1 1 1 1 1 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1 1 1 0 0 0 1 1 1 0 1 1 1 1 1 0 1
 1 1 0 0 1 1 0 1 0 1 0 0 1 1 0 1 0 0 0 1 1 1 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0
 0 0 0 0 0 1 1 0 0 0 1 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 1
 0 0 0 1 1]
```

l.) Training the model using Support Vector Machine Algorithm & also check accuracy :-

In [15]:
```
#Train support vector machine model
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train_std, y_train)
print("Support Vector Machine")
print("Accuracy on training set: {:.3f}".format(svm.score(X_train_std, y_train)))
print("Accuracy on test set: {:.3f}".format(svm.score(X_test_std, y_test)))
y_pred_sm = svm.predict(X_test_std)
print("Predicted value: ",y_pred_sm)
```

```
Support Vector Machine
Accuracy on training set: 0.987
Accuracy on test set: 0.972
Predicted value:  [0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 0 1 0 0 1 1 0 1 1 0 1 1 1 0 1 0 0 0
 1 1 0 1 0 0 0 1 0 0 0 1 1 0 1 1 0 0 1 0 0 1 0 1 1 0 1 0 1 1 0 1 1 1 1 1 0 1
 0 0 1 1 0 0 0 1 1 0 1 1 1 1 1 1 0 1 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 0 1 1 0 1 1 0
 0 1 1 1 0 0 1 1 1 0 1 1 1 0 0 0 1 1 1 1 1 0 1 0 1 0 1 0 0 1 0 1 0 0 1 1 1 1 0
 0 0 0 1 0 0 1 1 0 1 0 1 0 1 1 1 1 0 0 0 0 0 1 1 0 1 0 0 1 0 0 1 1 1 1 1 1 0
 1 1 1 0 1 0 0 0 0 0 1 1 1 1 1 0 1 1 1 0 0 1 1 0 0 0 1 0 1 0 0 1 0 1 1 0 0
 1 0 1 0 0 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 1 0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0
 0 0 0 0 1 0 0 1 1 1 1 1 0 1 0 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0
 1 0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 0 1 1 0 0 0 1 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1
 1 0 1 0 1 0 0 0 0 0 0 1 0 1 0 1 1 1 0 0 1 0 1 0 0 0 0 1 0 1 1 0 1 0 1 1 0
 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 0 1 0 0
 1 0 0 0 1 0 0 1 0 0 0 1 1 1 1 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1 1 1 0 1 1 0
 0 1 0 1 1 1 0 0 1 1 0 1 0 0 1 0 0 0 0 1 0 1 1 1 0 0 0 1 1 1 1 0 1 1 1 1 0
 1 0 1 0 1 0 1 1 1 0 1 1 1 0 0 0 1 1 0 1 0 1 1 1 1 0 1 1 0 1 1 0 0 1 1 0 0
 1 0 1 1 1 1 1 0 0 0 0 0 1 0 1 1 0 0 1 0 1 1 1 0 0 0 0 1 1 1 0 1 1 1 1 1 0 1
 1 1 0 0 1 1 0 1 0 1 0 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 1 0 1 0 0 1 1 1 0
 0 0 0 0 0 1 1 0 0 0 1 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 1
 0 0 0 1 1]
```

Result

The result is the calculated accuracies from our trained models help of different supervised classifier algorithms. The accuracies with the different models are compared and the best model is the one which has the highest accuracy with the dataset. The aim of using multiple models is to identify the best gender recognition model, which can be helpful in developing future applications.

The table below shows the obtained accuracies of different models.

Accuracy Table		
Algorithm Used	**Training**	**Testing**
Logistic Regression	97.6	96.8
KNN	98.4	96.4
Decision Tree	98.1	95.6
Support Vector Machine	98.7	97.2

Conclusion

Our presented statistical analysis and project implementation work demonstrate the efficiency of the proposed comparative algorithm for the gender recognition by voice. Moreover, it presents better performance and better accuracy of our trained Models which help to choose stable and robust supervised classification algorithms.

Accuracy of our model which is trained with help of SVM algorithm comes out to be **98.7% on training dataset** and **97.2% on testing dataset** has successfully achieved. The result points out that the algorithm with the highest accuracy is Support Vector Machine. So the best algorithm for Voice Based Gender Classification is **Support Vector Machine (SVM).**

Future Scope

Now a day's gender recognition has different various applications which provide a different way of security system and interaction like human computer interaction and authentication security systems by voice parameter to access data and many more. Some applications where gender recognition can be useful are speech emotion recognition, human to machine interaction, sorting of telephone calls by gender categorization. This voice recognition system also help to detecting the criminal gender when voice is recorded but image cannot be seen, used for recognize the emotion of a person , also used to classify the gender for audio samples and so on.

References

1. https://en.wikipedia.org/wiki/Voice_frequency
2. **Dataset:** https://www.kaggle.com/primaryobjects/voicegender
3. https://www.javatpoint.com/logistic-regression-in-machine-learning
4. https://www.analyticsvidhya.com/blog/2021/02/machine-learning-101-decision-tree-algorithm-for-classification/
5. https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
6. https://scikit-learn.org/stable/modules/svm.html
7. https://towardsdatascience.com/support-vector-machine-
8. https://iopscience.iop.org/article/10.1088/1757- 899X/263/4/042083/pdf
9. https://www.udemy.com/course/machinelearning/