MACHINE LEARNING

(Gender Recognition by Voice)

Summer Internship Report Submitted in partial fulfillment

of the requirement for undergraduate degree of

Bachelor of Technology

In

Computer Science & Engineering

By

Rajesh Gupta

ABSTRACT

Machine learning algorithms are used to predict the values from the data set by splitting the data set in to train and test and building Machine learning algorithms models of higher accuracy to predict the values is the primary task to be performed on Cereals data set My perception of understanding the given data set has been in the view of undertaking a client’s requirement of overcoming the stagnant point of sales of the products being manufactured by client.

To get a better understanding and work on a strategical approach for solution of the client, I have adapted the view point of looking at ratings of the products and for further deep understanding of the problem, I have taken the stance of a consumer and reasoned out the various factors of choice of the products and they purchase , and my primary objective of this case study was to look up the factors which were dampening the sale of products and corelate them to ratings of products and draft out an outcome report to client regarding the various accepts of a product manufacturing , marketing and sale point determination

Table of Contents:

CHAPTER 1

MACHINE LEARNING

1.1 INTRODUCTION:

Machine Learning(ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of ArtificialIntelligence(AI).

1.2 IMPORTANCE OF MACHINE LEARNING:

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and “more items to consider” and “get yourself a little something” on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today’s data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries. With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that’s in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works

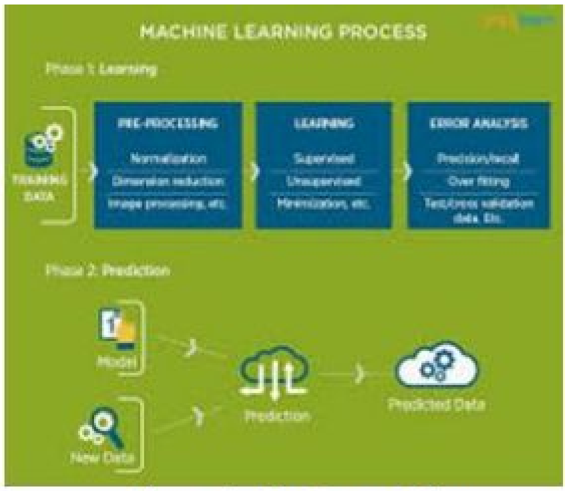


Figure 1 : The Process Flow

1.3 USES OF MACHINE LEARNING:

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let’s consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data.

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data.

By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

1.4 TYPES OF LEARNING ALGORITHMS:

The types of machine learning algorithms differ in their approach, the type of data they Input and output, and the type of task or problem that they are intended to solve.

1.4.1 Supervised Learning :

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to "learn" how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

1.4.2 Unsupervised Learning:

When an algorithm learns from plain examples without any associated response,leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

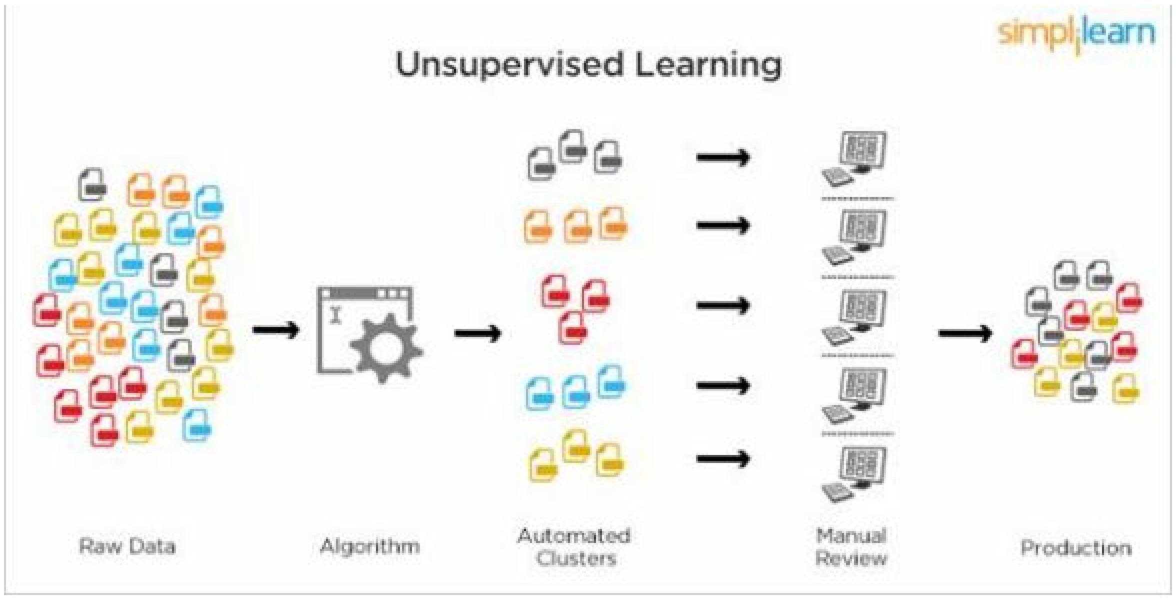


Figure 2 : Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering.
Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

1.4.3 Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

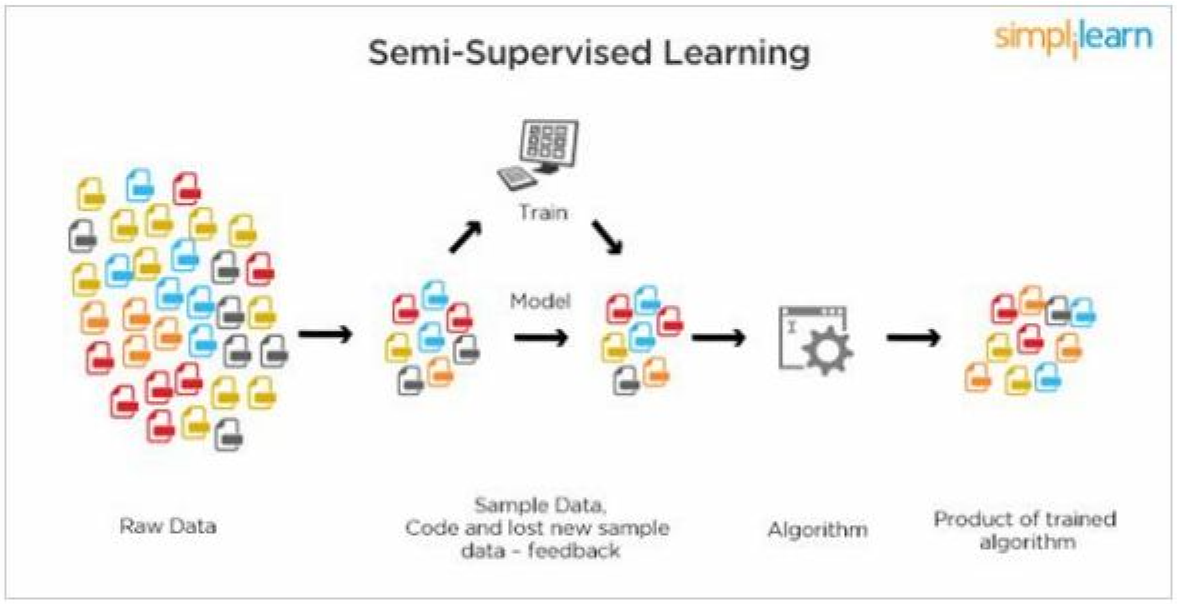

Semi-Supervised Learning
simplilearn
Train
Model
Raw Data
Sample Data, Code and lost new sample data - feedback
Algorithm
Product of trained algorithm

Figure 3 : Semi Supervised Learning

1.5 RELATION BETWEEN DATA MINING,MACHINE LEARNING AND DEEP LEARNING:

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovers previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning.

CHAPTER 2

PYTHON

Basic programming language used for machine learning is : PYTHON

2.1 INTRODUCTION TO PYHTON:

- Python is a high-level, interpreted, interactive and object-oriented scripting Language.
- Python is a general purpose programming language that is often applied in scripting roles
- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
- Python is Interactive: You can sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

2.2 HISTORY OF PYTHON:

- Python was developed by GUIDO VAN ROSSUM in early 1990's
- Its latest version is 3.7 , it is generally called as python3

2.3 FEATURES OF PYTHON:

- Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax,This allows the student to pick up the language quickly.
- Easy-to-read: Python code is more clearly defined and visible to the eyes.

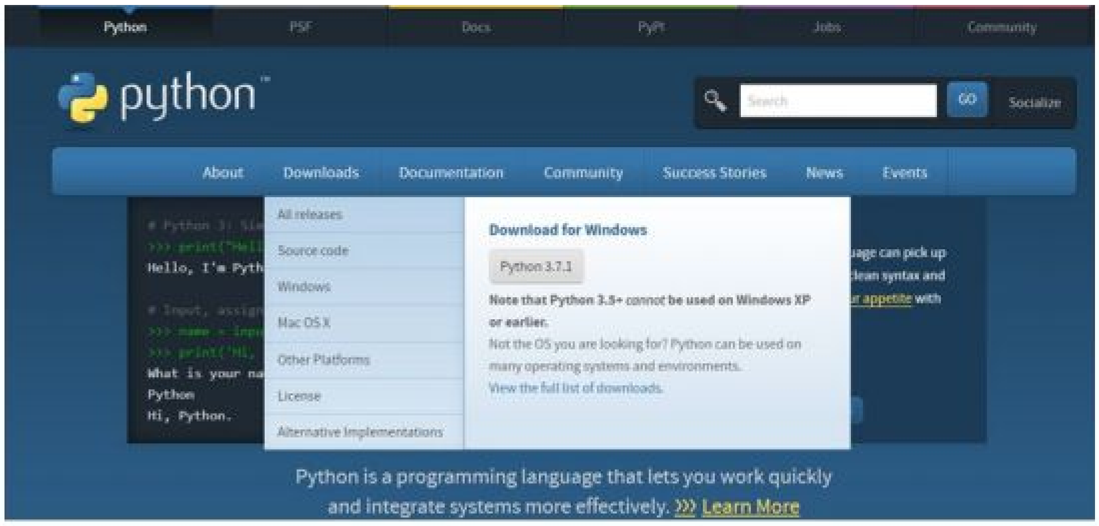
- Easy-to-maintain: Python's source code is fairly easy-to-maintaining.
- A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases: Python provides interfaces to all major commercial databases.
- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

2.4 HOW TO SETUP PYTHON:

- Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.
- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

2.4.1 Installation(using python IDLE):

- Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.
- Download python from www.python.org
- When the download is completed, double click the file and follow the instructions to install it.
- When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.



Note that Python 3.5+ cannot be used on Windows XP or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments.

View the full list of downloads.

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Figure 4 : Python download

2.4.2 Installation(using Anaconda):

- Python programs are also executed using Anaconda.
- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.
- Conda is a package manager quickly installs and manages packages.
- In WINDOWS:
- In windows
  - Step 1: Open Anaconda.com/downloads in web browser.
  - Step 2: Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)
  - Step 3: select installation type(all users)

- Step 4: Select path(i.e. add anaconda to path & register anaconda as

  Default python 3.4) next click install and next click finish
- Step 5: Open jupyter notebook (it opens in default browser)

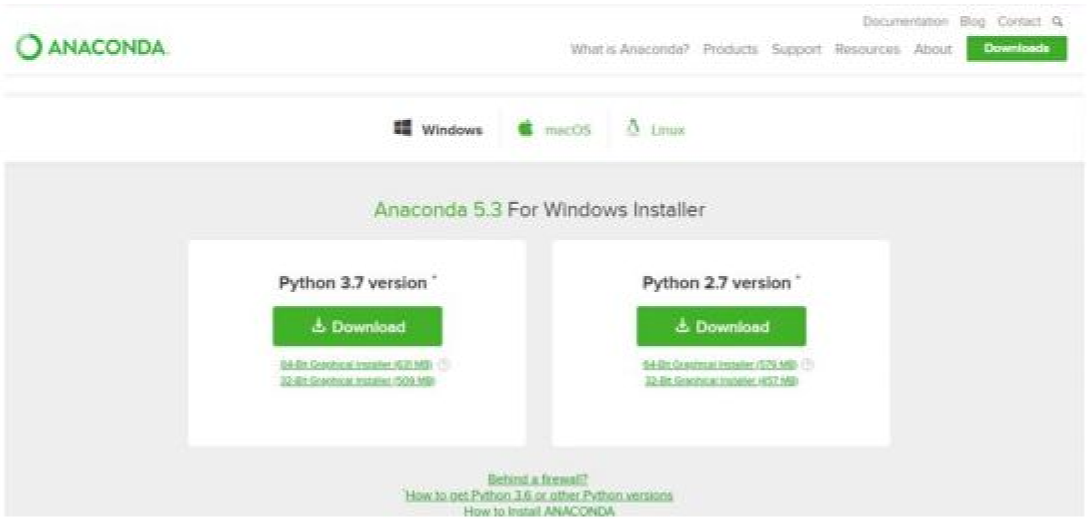

ANACONDA.
Documentation Blog Contact
What is Anaconda? Products Support Resources About Downloads
Windows macOS Linux
Anaconda 5.3 For Windows Installer
Python 3.7 version *
Download
64-Bit Graphical Installer (631 MB)
32-Bit Graphical Installer (509 MB)
Python 2.7 version *
Download
64-Bit Graphical Installer (579 MB)
32-Bit Graphical Installer (457 MB)
Behind a firewall?
* How to get Python 3.6 or other Python versions
How to Install ANACONDA

Figure 5 : Anaconda download

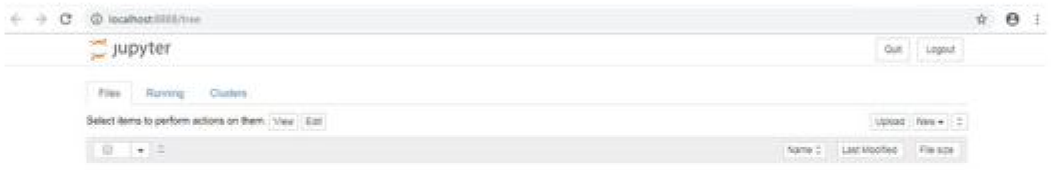

localhost:8888/tree
jupyter
Quit Logout
Files Running Clusters
Select items to perform actions on them. View Edit
Upload New
Name Last Modified File size

Figure 6 : Jupyter notebook

2.5 PYTHON VARIABLE TYPES:

● Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

● Variables are nothing but reserved memory locations to store values
.
● Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.

● Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.

● Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

● Python has five standard data types –

- o Numbers
- o Strings
- o Lists
- o Tuples
- o Dictionary

2.5.1 Python Numbers:

- Number data types store numeric values. Number objects are created when you assign a value to them.
- Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

2.5.2 Python Strings:

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

2.5.3 Python Lists:

- Lists are the most versatile of Python's compound data types.
- A list contains items separated by commas and enclosed within square brackets ([]).
- To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([ ] and [:])

with indexes starting at 0 in the beginning of the list and working their way to end -1.

● The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

2.5.4 Python Tuples:

● A tuple is another sequence data type that is similar to the list.

● A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

● The main differences between lists and tuples are: Lists are enclosed in brackets ([]) and their elements and size can be changed, while tuple are enclosed in parentheses (( )) and cannot be updated.

● Tuples can be thought of as read-only lists.

● For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

2.5.5 Python Dictionary:

● Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

● Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

● You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.

● What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

2.6 PYTHON FUNCTION:

2.6.1 Defining a Function:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e.()).

Any input parameters or arguments should be placed within these parentheses.You can also define parameters inside these parentheses

The code block within every function starts with a colon (:) and is indented.The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

2.6.2 Calling a Function:

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

2.7 PYTHON USING OOP's CONCEPTS:

2.7.1 Class:

- Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot Notation.

- Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.

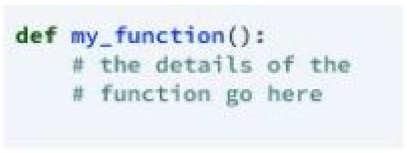
- Data member: A class variable or instance variable that holds data associated with a class and its objects.

- Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.

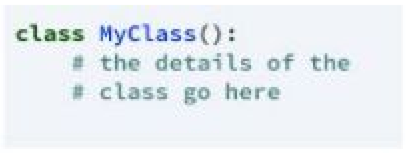
- Defining a Class:

    - We define a class in a very similar way how we define a function.

    - Just like a function ,we use parentheses and a colon after the class name(i.e. ():) when we define a class. Similarly, the body of our class is indented like a functions body is.



```
def my_function():
    # the details of the
    # function go here
```



```
class MyClass():
    # the details of the
    # class go here
```

Figure 7 : Defining a Class

2.7.2 __init__ method in Class:

- The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.

- The init method has a special name that starts and ends with two underscores:__init__().

CHAPTER 3

CASE STUDY

3.1 PROBLEM STATEMENT:

To predict the Gender Recognition by Voice using Machine Learning algorithm called CLASSIFICATION.

3.2 DATA SET:

The given data set consists of the following parameters:

A - meanfreq: mean frequency (in kHz)

B - sd: standard deviation of frequency

C - median: median frequency (in kHz)

D - Q25: first quantile (in kHz)

E - Q75: third quantile (in kHz)

F - IQR: interquantile range (in kHz)

G - skew: skewness (see note in specprop description)

H - kurt: kurtosis (see note in specprop description)

I - sp.ent: spectral entropy

J - sfm: spectral flatness

K - mode: mode frequency

L - centroid: frequency centroid (see specprop)

M - peakf: peak frequency (frequency with highest energy)

N - meanfun: average of fundamental frequency measured across acoustic signal

P - minfun: minimum fundamental frequency measured across acoustic signal

Q - maxfun: maximum fundamental frequency measured across acoustic signal

R - mindom: minimum of dominant frequency measured across acoustic signal

S - maxdom: maximum of dominant frequency measured across acoustic signal

T - dfrange: range of dominant frequency measured across acoustic signal

U - modindx: modulation index. Calculated as the accumulated absolute difference between

V - adjacent measurements of fundamental frequencies divided by the frequency range

W - label: male or female

3.3 OBJECTIVE OF THE CASE STUDY:

it appears that looking at the mean fundamental frequency might be enough to accurately classify a voice. However, some male voices use a higher frequency, even though their resonance differs from female voices, and may be incorrectly classified as female. To the human ear, there is apparently more than simple frequency,that determines a voice’s gender.

CHAPTER 4

DATA PREPROCESSING

4.1 Statistical Analysis :

Preprocessing of the data actually involves the following steps:

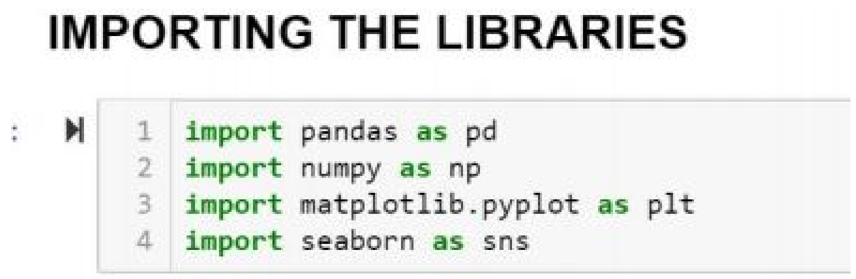
4.1.1 GETTING THE DATASET:

We can get the data set from the database or

we can get the data from client.

4.1.2 IMPORTING THE LIBRARIES:

We have to import the libraries as per the requirement of the Algorithm.



IMPORTING THE LIBRARIES

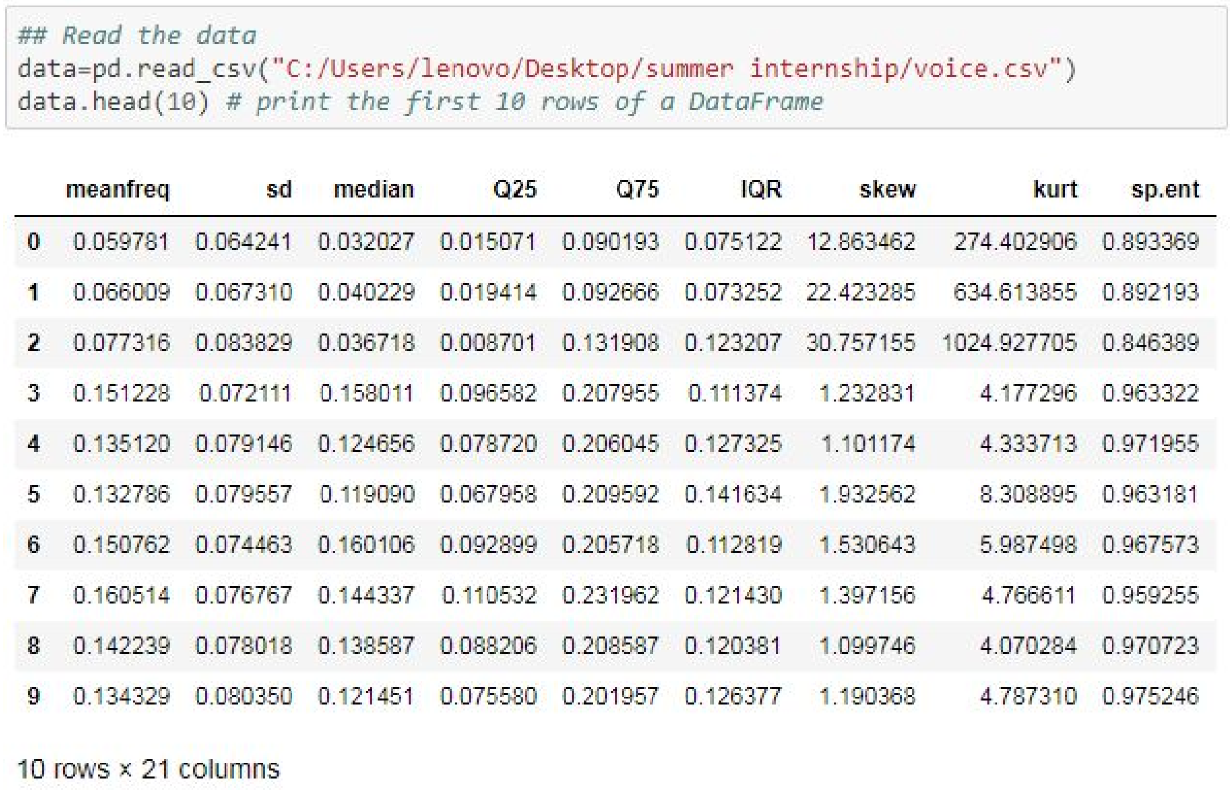
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Figure 8 : Importing Libraries

4.1.3 IMPORTING THE DATA-SET:

Pandas in python provide an interesting method read_csv(). The read_csv function reads the entire dataset from a comma separated values file and we can assign it to a DataFrame to which all the operations can be performed. It helps us to access each and every row as well as columns and each and every value can be access using the dataframe. Any missing value or NaN value have to be cleaned.

4.1.4 READING THE DATA-SET



```
## Read the data
data=pd.read_csv("C:/Users/lenovo/Desktop/summer internship/voice.csv")
data.head(10) # print the first 10 rows of a DataFrame
```

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent
0	0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.863462	274.402906	0.893369
1	0.066009	0.067310	0.040229	0.019414	0.092666	0.073252	22.423285	634.613855	0.892193
2	0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.757155	1024.927705	0.846389
3	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.971955
5	0.132786	0.079557	0.119090	0.067958	0.209592	0.141634	1.932562	8.308895	0.963181
6	0.150762	0.074463	0.160106	0.092899	0.205718	0.112819	1.530643	5.987498	0.967573
7	0.160514	0.076767	0.144337	0.110532	0.231962	0.121430	1.397156	4.766611	0.959255
8	0.142239	0.078018	0.138587	0.088206	0.208587	0.120381	1.099746	4.070284	0.970723
9	0.134329	0.080350	0.121451	0.075580	0.201957	0.126377	1.190368	4.787310	0.975246

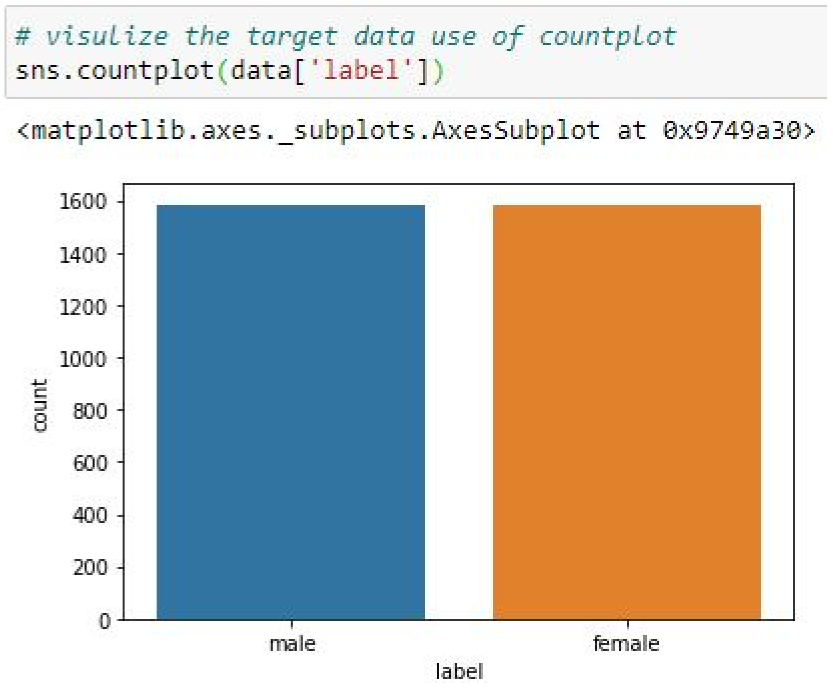
10 rows × 21 columns

Figure 9 : Reading the dataset

4.2 Generating Plots

4.2.1 -Visualize the data between Target and the Features :

- Here we are plotting the target data using countplot then find Data 50% of male and 50% of Female in target data



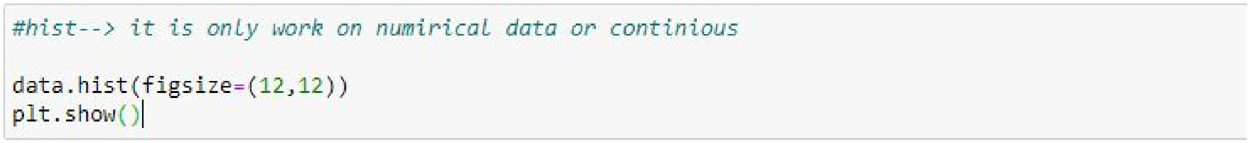
```
# visulize the target data use of countplot
sns.countplot(data['label'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x9749a30>
```

Figure 10 : Visualize the Target Value

4.2.2 - Visualize the data between all the Features ：

- Here we are plotting all columns using histogram
- It is only work on Numiracal and Continious Data



```
#hist--> it is only work on numirical data or continious

data.hist(figsize=(12,12))
plt.show()
```

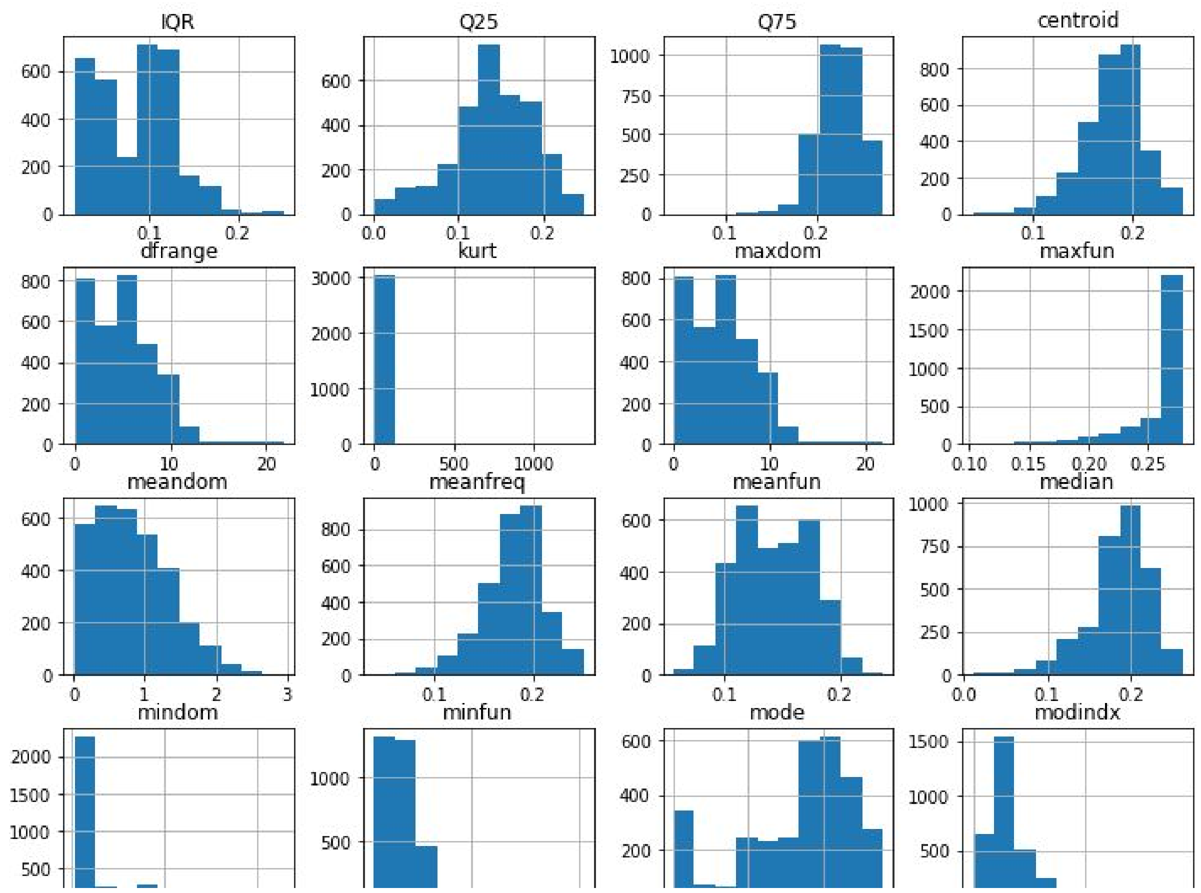


Figure 11 : Visualize the feature using histogram

4.3 Detection of Outliers

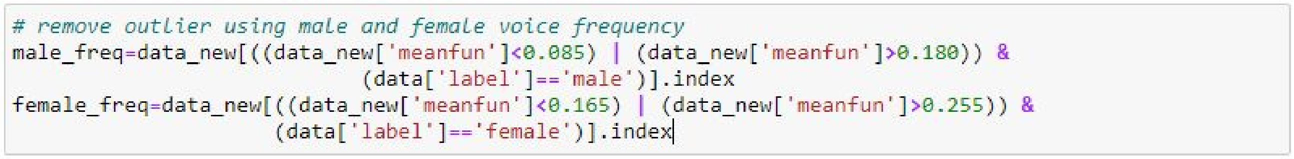
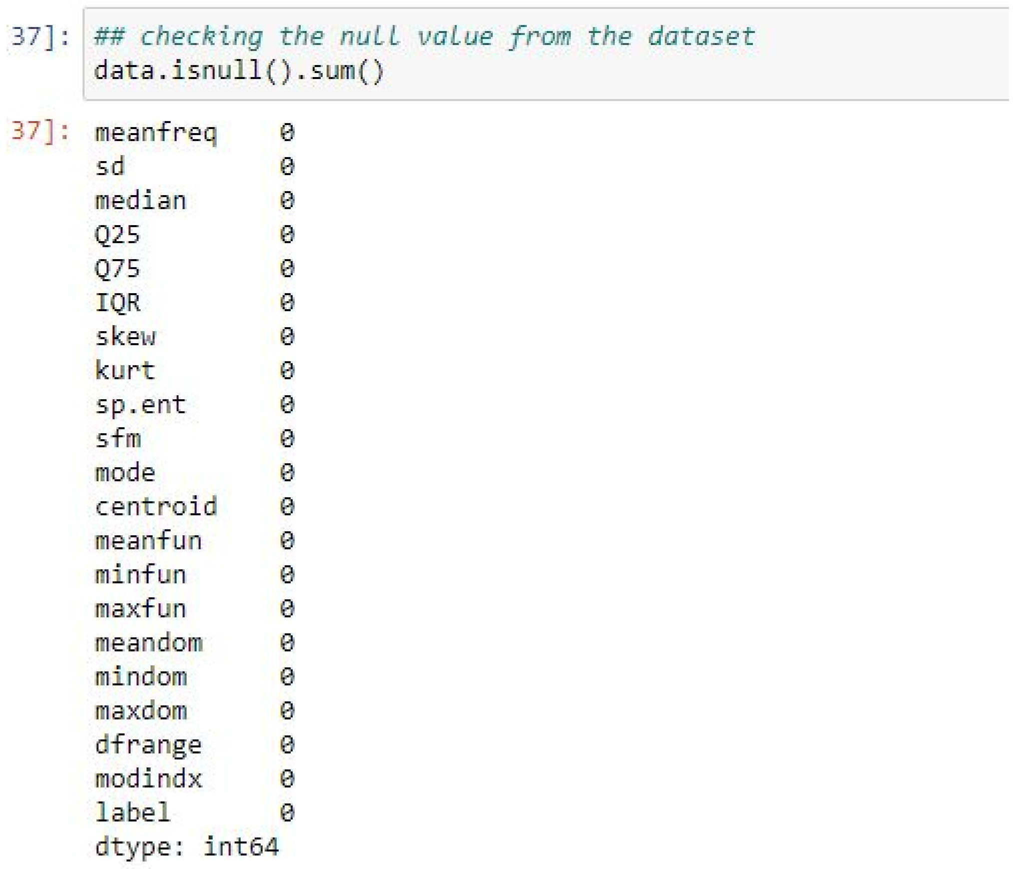

```
# remove outlier using male and female voice frequency
male_freq=data_new[((data_new['meanfun']<0.085) | (data_new['meanfun']>0.180)) &
                   (data['label']=='male')].index
female_freq=data_new[((data_new['meanfun']<0.165) | (data_new['meanfun']>0.255)) &
                     (data['label']=='female')].index
```

Figure 12 : Remove Outlier

4.4 HANDLING MISSING VALUES:

o There is a method called isnull() which gives the number of missing values in each and every column.
o Using fillna() method each and every missing value is replaced by 0.



```
37]: ## checking the null value from the dataset
     data.isnull().sum()

37]: 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
```

Figure 13 : Checking Null Value

4.5 CATEGORICAL DATA:

● Machine Learning models are based on equations, we need to replace the text by numbers. So that we can include the numbers in the equations.

● Categorical Variables are of two types: Nominal and Ordinal
● **Nominal**: The categories do not have any numeric ordering in between them. They don't have any ordered relationship between each of them. Examples: Male or Female, any colour

- **Ordinal**: The categories have a numerical ordering in between them. Example:Graduate is less than Post Graduate, Post Graduate is less than Ph.D. customer satisfaction survey, high low medium

- Categorical data can be handled by using dummy variables, which are also called as indicator variables.

- Handling categorical data using dummies:

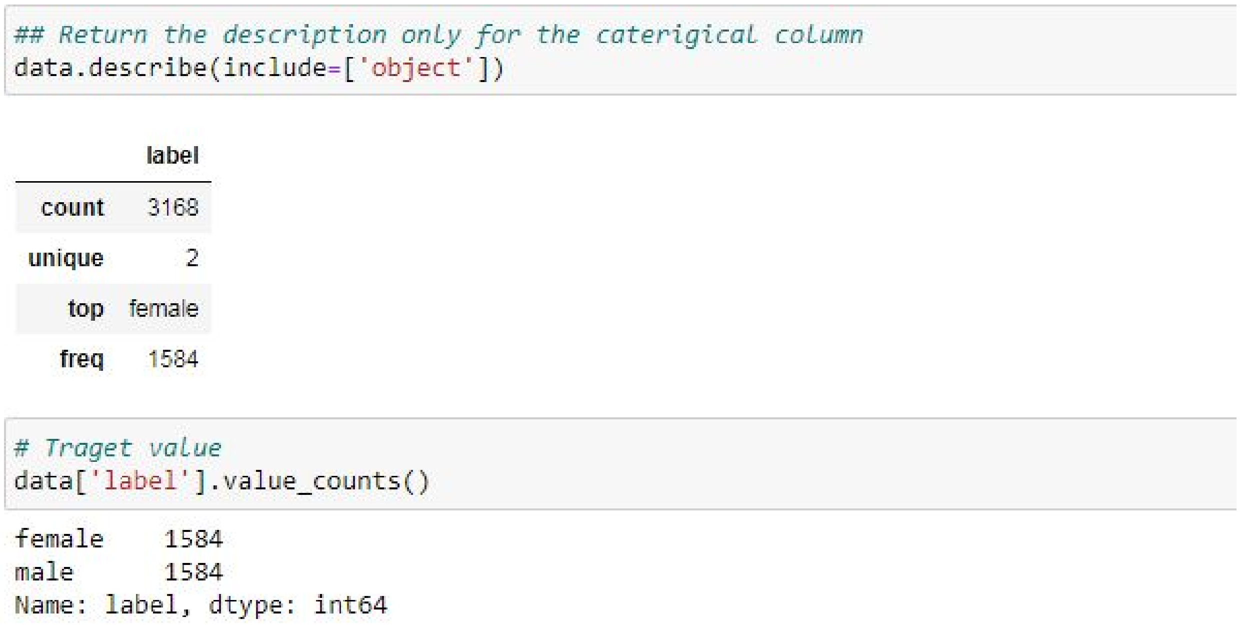
  In pandas library we have a method called get_dummies() which creates dummy variables for those categorical data in the form of 0's and 1's.

Once these dummies got created we have to concat this dummy set to our dataframe or we can add that dummy set to the dataframe.

- Getting dummies using label encoder from scikit learn package

We have a method called label encoder in scikit learn package .we need to import the label encoder method from scikitlearn package and after that we have to fit and transform the data frame to make the categorical data into dummies.

If we use this method to get dummies then in place of categorical data we get the numerical values (0,1,2….)



```
## Return the description only for the caterigical column
data.describe(include=['object'])
```

	label
count	3168
unique	2
top	female
freq	1584

```
# Traget value
data['label'].value_counts()
```

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

Figure 14 : analyse the Categorical Data

CHAPTER 5

FEATURE SELECTION

5.1 Select relevant features for the analysis

● Correlation: Correlation is a statistical technique that can show whether and how strongly pairs of variables are related. Correlation is described as the analysis which lets us know the association or the absence of the relationship between two variables ‘x’ and ‘y’. It is a statistical measure that represents the strength of the connection between pairs of variables.

● In pair plot we need to find the correlation between two variables and we can do some

```
plt.figure(figsize=(15,10))
sns.heatmap(data.corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x9780ad0>
```

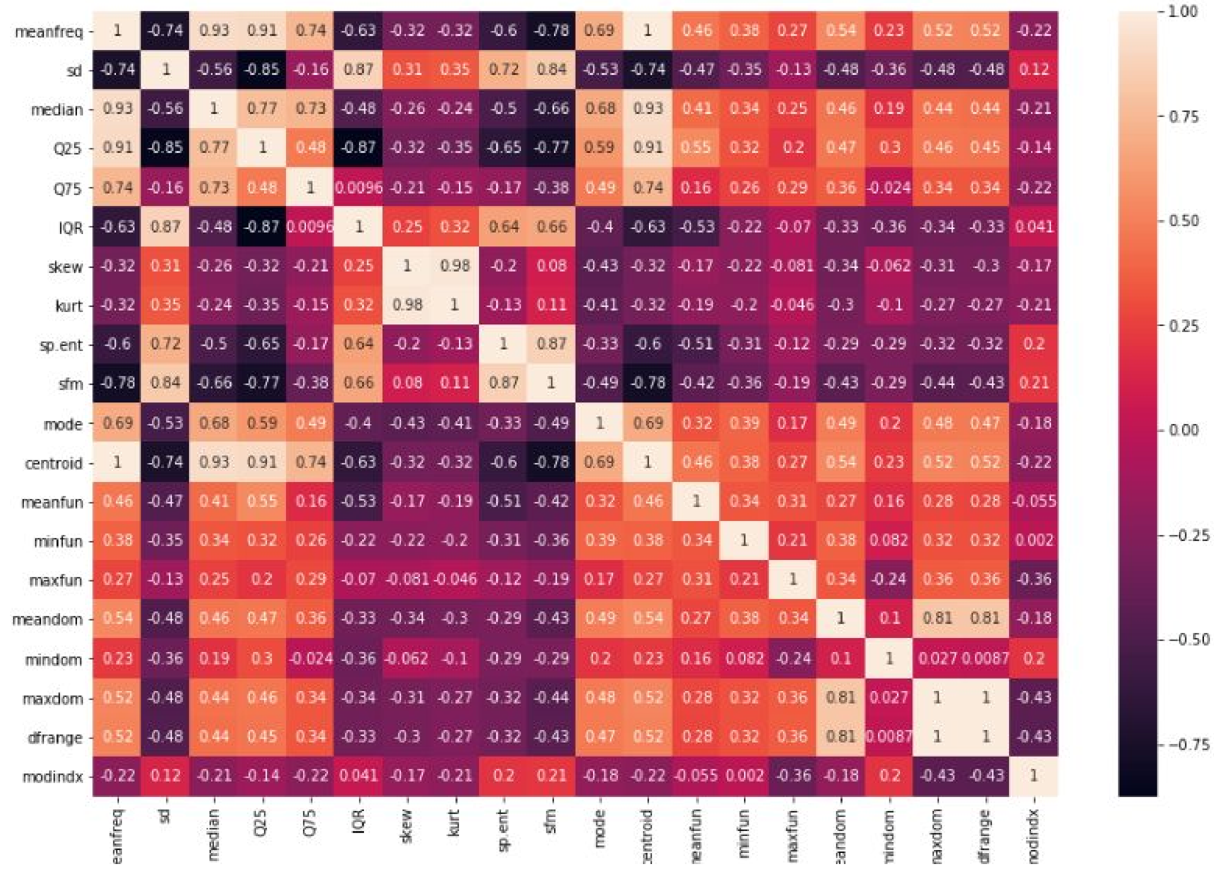
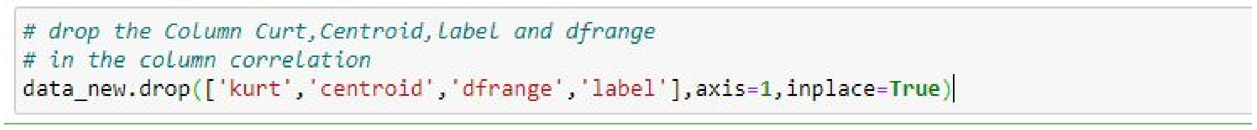


Figure 15 : Visualise the Correlation

5.2 Drop irrelevant features:

5.2.1 Drop Manually:

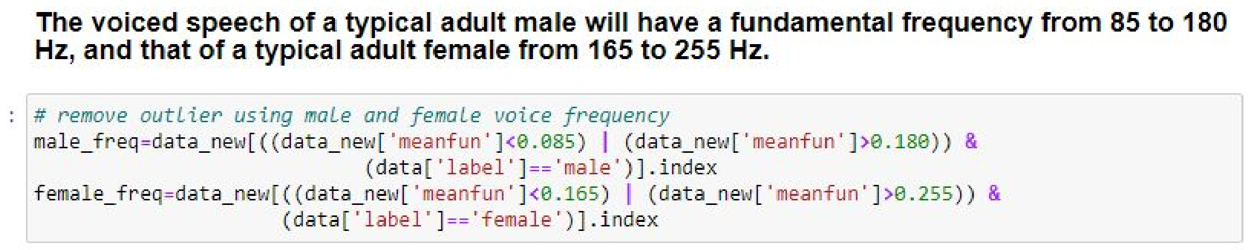
- Removing the columns which are not required using drop method



```
# drop the Column Curt,Centroid,label and dfrange
# in the column correlation
data_new.drop(['kurt','centroid','dfrange','label'],axis=1,inplace=True)
```

Figure 16 : Drop Data

5.2.2 Based on Correlation



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.

```
# remove outlier using male and female voice frequency
male_freq=data_new[((data_new['meanfun']<0.085) | (data_new['meanfun']>0.180)) &
                             (data['label']=='male')].index
female_freq=data_new[((data_new['meanfun']<0.165) | (data_new['meanfun']>0.255)) &
                     (data['label']=='female')].index
```

Figure 17 : Remove Outlier

5.3 Train-Test-Split:

- Splitting the data : after the preprocessing is done then the data is split into train and test sets

- In Machine Learning in order to access the performance of the classifier. You train the classifier using 'training set' and then test the performance of your classifier on unseen 'test set'. An important point to note is that during training the classifier only uses the training set . The test set must not be used during training the classifier. The test set will only be available during testing the classifier.

- training set - a subset to train a model.(Model learns patterns between Input and Output)

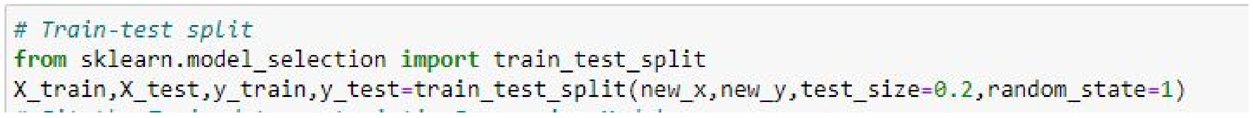
- test set - a subset to test the trained model.(To test whether the model has correctly learnt)

- The amount or percentage of Splitting can be taken as specified (i.e. train data = 75%, test data =25% or train data = 80% , test data= 20%)

- First we need to identify the input and output variables and we need to separate the input set and output set

- In scikit learn library we have a package called model_selection in which train_test_split method is available .we need to import this method

- This method splits the input and output data to train and test based on the percentage specified by the user and assigns them to four different variables(we need to mention the variables)



```
# Train-test split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(new_x,new_y,test_size=0.2,random_state=1)
```

Figure 18 : Split Data

CHAPTER 6

MODEL BUILDING AND EVALUATION:

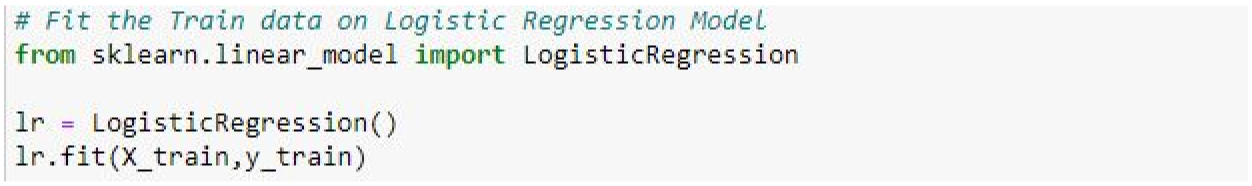
6.1 Brief about the algorithms used

Here i am using the Different types of model:-

6.1.1 Logistic Regression Model :

- Import Logistic regression method which is available in linear_model package From scikit learn library

6.1.1.1 Train the Models



```
# Fit the Train data on Logistic Regression Model
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr.fit(X_train,y_train)
```

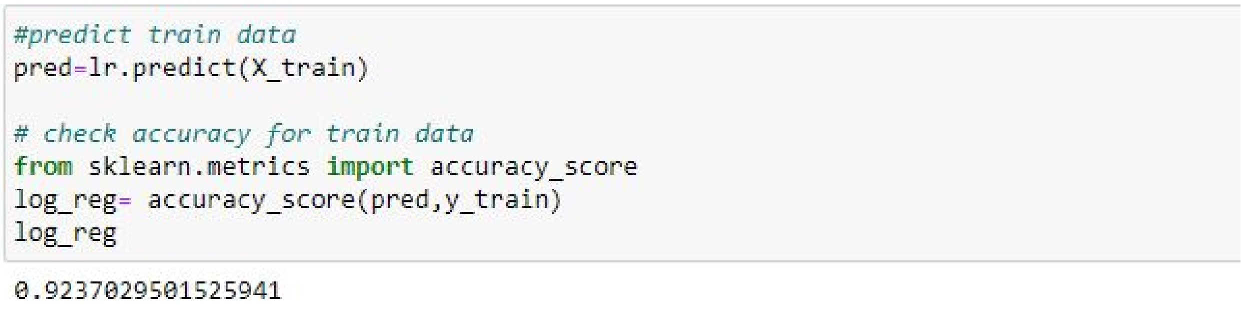
Figure 19 : Apply Logistic Regression Model

6.1.1.2 Make Predictions :

- Once the model is built we need to check for accuracy.
- This can be done using predict method which is used to predict the

output for input test set, and compare the predicted output with original output train set.

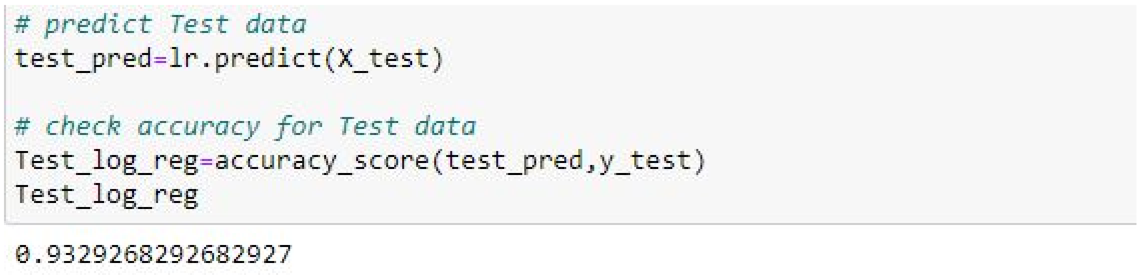
o If the predicted values and the original values are close then we can say that model is trained with good accuracy



```
#predict train data
pred=lr.predict(X_train)

# check accuracy for train data
from sklearn.metrics import accuracy_score
log_reg= accuracy_score(pred,y_train)
log_reg
```

```
0.9237029501525941
```

Figure 20 : Predict Train Data



```
# predict Test data
test_pred=lr.predict(X_test)

# check accuracy for Test data
Test_log_reg=accuracy_score(test_pred,y_test)
Test_log_reg
```

```
0.9329268292682927
```

Figure 21 : Predict Test Data

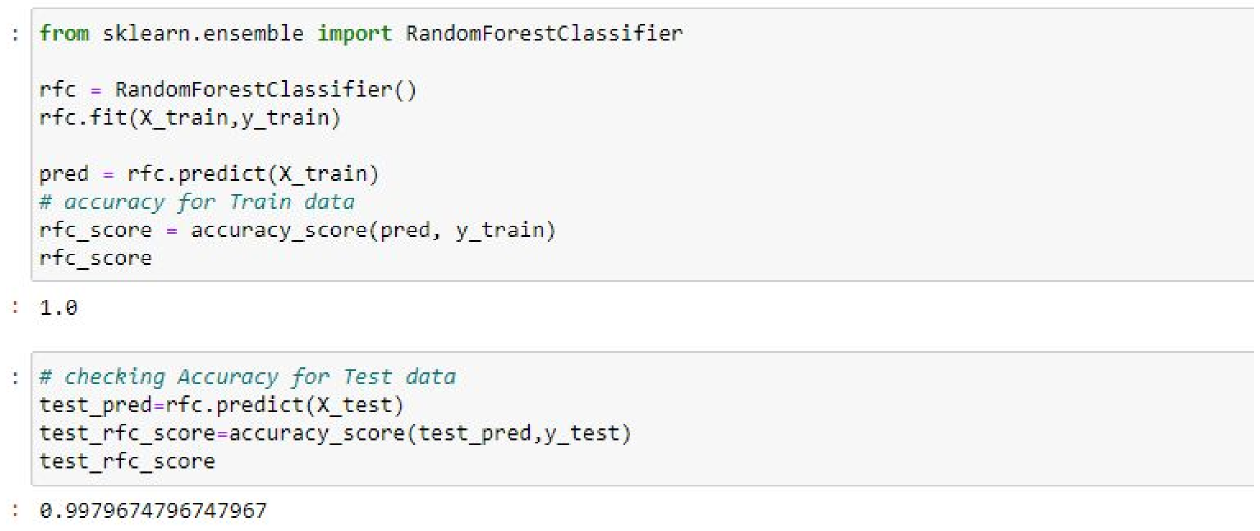
6.1.2 Random Forest Classifier Model:

- Import Random Forest Classifier method which is available in ensemble Package From scikit learn library

- Once the model is built we need to check for accuracy.

- This can be done using predict method which is used to predict the output for input test set, and compare the predicted output with original output train and test set.

o If the predicted values and the original values are close then we can say that model is trained with good accuracy



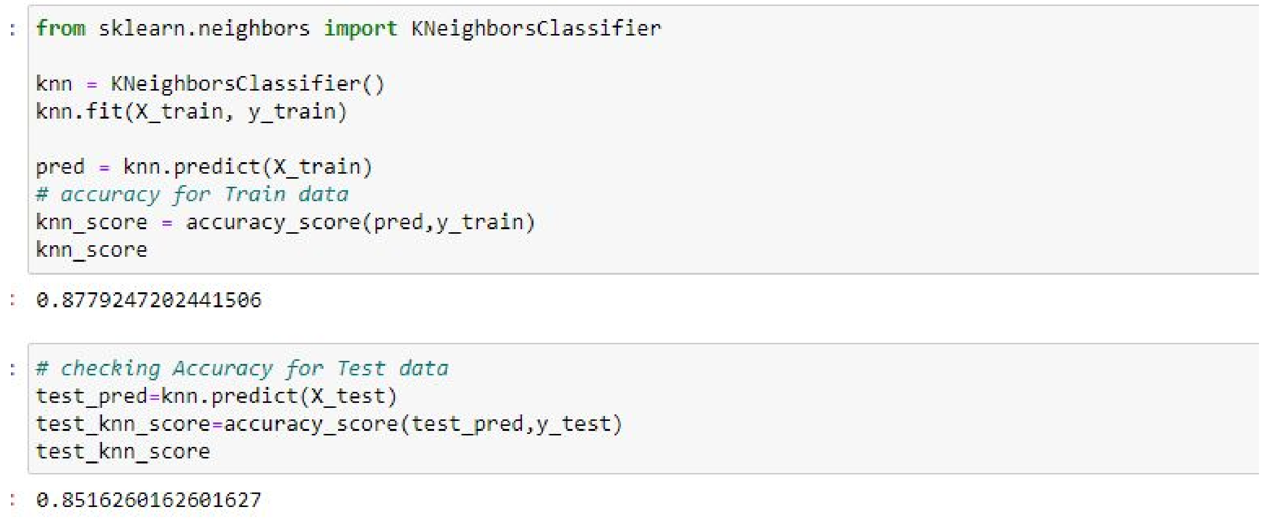
```
: from sklearn.ensemble import RandomForestClassifier

  rfc = RandomForestClassifier()
  rfc.fit(X_train,y_train)

  pred = rfc.predict(X_train)
  # accuracy for Train data
  rfc_score = accuracy_score(pred, y_train)
  rfc_score

: 1.0

: # checking Accuracy for Test data
  test_pred=rfc.predict(X_test)
  test_rfc_score=accuracy_score(test_pred,y_test)
  test_rfc_score

: 0.9979674796747967
```

Figure 22 : Apply Random Forest Model

6.1. 3 K-Nearest Neighbours Model :

- Import KNeighbors Classifier method which is available in neighbors Package From scikit learn library
- Once the model is built we need to check for accuracy.
- This can be done using predict method which is used to predict the output for input test set, and compare the predicted output with original output train and test set.

o If the predicted values and the original values are close then we can say that model is trained with good accuracy



```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(X_train, y_train)

pred = knn.predict(X_train)
# accuracy for Train data
knn_score = accuracy_score(pred,y_train)
knn_score
```

```
0.8779247202441506
```

```
# checking Accuracy for Test data
test_pred=knn.predict(X_test)
test_knn_score=accuracy_score(test_pred,y_test)
test_knn_score
```

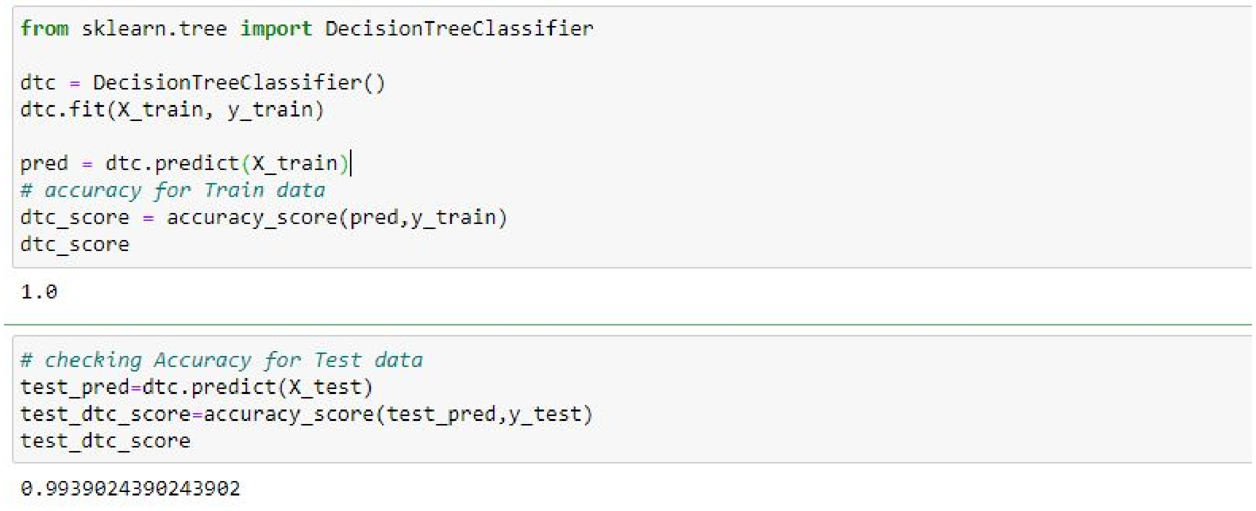
```
0.8516260162601627
```

Figure 23 : Apply K-Nearest Neighbours Model

6.1.4 Decision Tree Classifier Model :

- Import DecisionTree Classifier method which is available in tree Package From scikit learn library
- Once the model is built we need to check for accuracy.
- This can be done using predict method which is used to predict the output for input test set, and compare the predicted output with original output train and test set.

o If the predicted values and the original values are close then we can say that model is trained with good accuracy



```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)

pred = dtc.predict(X_train)
# accuracy for Train data
dtc_score = accuracy_score(pred,y_train)
dtc_score
```

```
1.0
```

```
# checking Accuracy for Test data
test_pred=dtc.predict(X_test)
test_dtc_score=accuracy_score(test_pred,y_test)
test_dtc_score
```

```
0.9939024390243902
```

Figure 24 : Apply Decision Tree Classifier Model

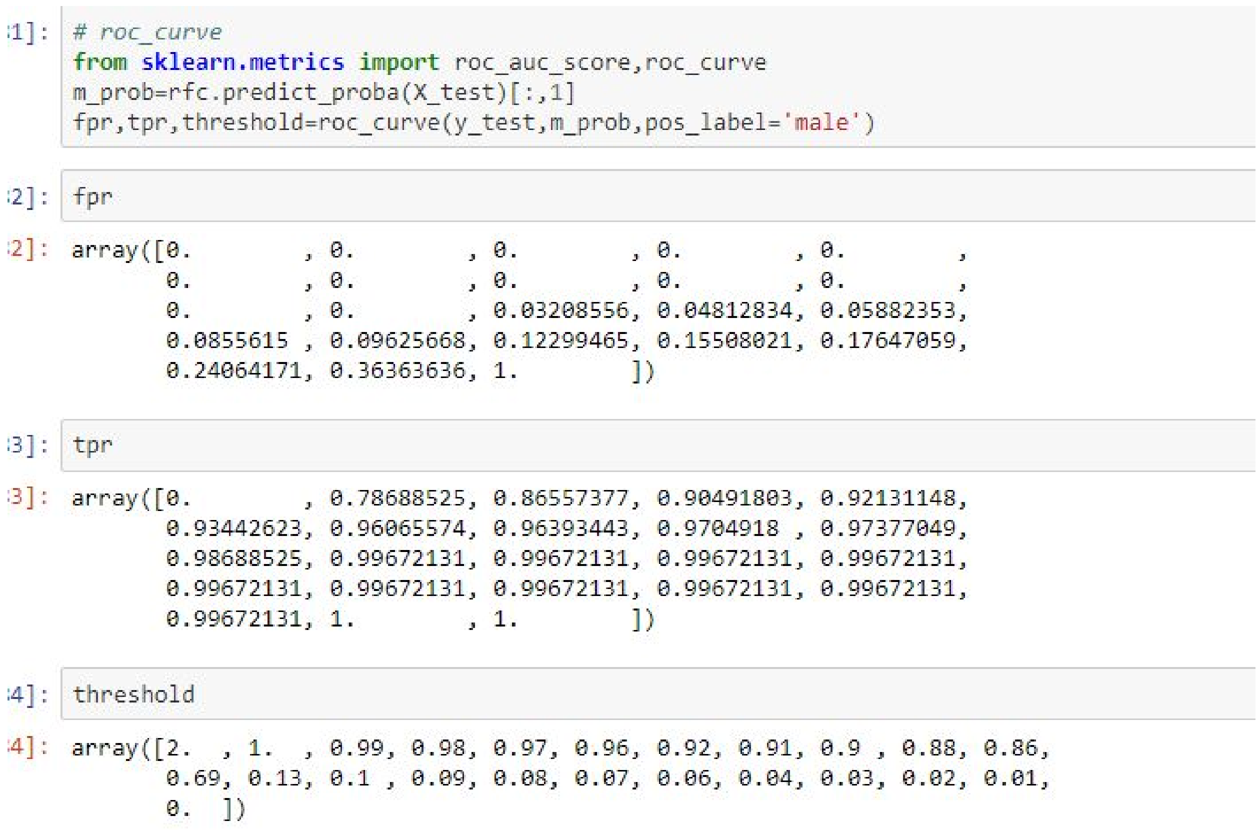
6.2 Use of ROC and ROC AUC CURVE :

6.2.1 ROC

An ROC curve (Receiver Operating Characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

- True Positive Rate
- False Positive Rate

It is a probability curve that plots the TPR against FPR at various threshold values.



```
:1]: # roc_curve
     from sklearn.metrics import roc_auc_score,roc_curve
     m_prob=rfc.predict_proba(X_test)[:,1]
     fpr,tpr,threshold=roc_curve(y_test,m_prob,pos_label='male')

:2]: fpr

:2]: array([0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.        , 0.        , 0.        ,
            0.        , 0.        , 0.03208556, 0.04812834, 0.05882353,
            0.0855615 , 0.09625668, 0.12299465, 0.15508021, 0.17647059,
            0.24064171, 0.36363636, 1.        ])

:3]: tpr

:3]: array([0.        , 0.78688525, 0.86557377, 0.90491803, 0.92131148,
            0.93442623, 0.96065574, 0.96393443, 0.9704918 , 0.97377049,
            0.98688525, 0.99672131, 0.99672131, 0.99672131, 0.99672131,
            0.99672131, 0.99672131, 0.99672131, 0.99672131, 0.99672131,
            0.99672131, 1.        , 1.        ])

:4]: threshold

:4]: array([2.  , 1.  , 0.99, 0.98, 0.97, 0.96, 0.92, 0.91, 0.9 , 0.88, 0.86,
            0.69, 0.13, 0.1 , 0.09, 0.08, 0.07, 0.06, 0.04, 0.03, 0.02, 0.01,
            0.  ])
```

Figure 25 : Apply ROC CURVE

6.2.2 AUC CURVE

The **Area Under the Curve (AUC)** is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

Setting different thresholds for classifying positive classes for data points will inadvertently change the Sensitivity and Specificity of the model. And one of these thresholds will probably give a better result than the others, depending on whether we are aiming to lower the number of False Negatives or False Positives.

● Mathematically, it represents the various confusion matrices for various thresholds. Each black dot is one confusion matrix.
● The green dotted line represents the scenario when the true positive rate equals the false positive rate.
● As evident from the curve, as we move from the rightmost dot towards left, after a certain threshold, the false positive rate decreases.
● After some time, the false positive rate becomes zero.
● The point encircled in green is the best point as it predicts all the values correctly and keeps the False positive as a minimum.
● But that is not a rule of thumb. Based on the requirement, we need to select the point of a threshold.
● The ROC curve answers our question of which threshold to choose.

```
]: plt.plot(fpr,tpr)

]: [<matplotlib.lines.Line2D at 0xa575830>]
```

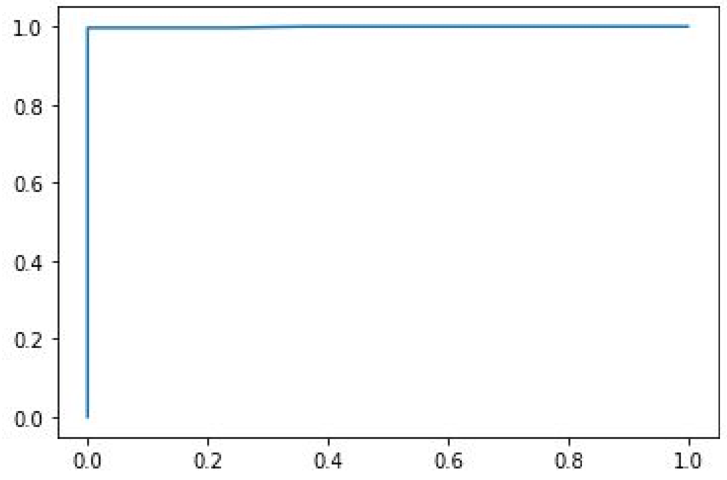


Figure 26 : roc curve

When AUC = 1, then the classifier is able to perfectly distinguish between all the Positive and the Negative class points correctly. If, however, the AUC had been 0, then the classifier would be predicting all Negatives as Positives, and all Positives as Negatives.

7. CONCLUSION :

It is concluded after performing thorough Exploratory Data analysis which include Random Forest Classifier Model and Decision Tree Classifier Model which are computed to get accurate accuracy.

REFERENCES:

[1] **https://www.kaggle.com/primaryobjects/voicegender**

[2] **https://github.com/Rajeshgupta143**

[3] https://en.wikipedia.org/wiki/Machine_learning