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

A.Shiva ram yadav

221710312005

Under the Guidance of

Mrs.k.Neha

Assistant Professor

Department Of ComputerScience and Engineering

GITAM School of Technology

GITAM (Deemed to be University), Hyderabad-502329

DECLARATION

I submit this industrial training work entitled "GENDER RECOGNITION BY VOICE" to GITAM (Deemed To Be University), Hyderabad in partial fulfilment of the requirements for the award of the degree of "Bachelor of Technology" in "Computer Science Engineering". I declare that it was carried out independently by me under the guidance of Mrs.K.Neha , Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD **A.Shivaram yadav**

Date:13-07-2020 **221710312005**

CERTIFICATE

This is to certify that the Industrial Training Report entitled "GENDER RECOGNITION BY VOICE" is being submitted by Allamula Shivaram yadav (221710312005) in partial fulfilment of the requirement for the award of Bachelor of Technology in Computer Science Engineering at GITAM (Deemed To Be University), Hyderabad during the academic year 2018-19.

It is faithful record work carried out by her at the Computer Science Engineering Department, GITAM University Hyderabad Campus
under my guidance and supervision.

Mrs. K. Neha

Assistant Professor

Department of CSE

Prof. S. Phani Kumar

Professor and HOD

Department of CSE

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank respected **Dr. N. Siva Prasad**, Pro Vice Chancellor,GITAM Hyderabad and **Prof.N.Seetha Ramaiah,** Principal, GITAM Hyderabad.

I would like to thank respected **Prof.S.Phani Kumar**, Head of the Department of Computer Science Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present a internship report. It helped me a lot to realize of what we study for.

I would like to thank the respected faculties **Mrs.K.Neha** who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

A.Shivaram yadav

221710312005

ABSTRACT

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 attributes using various Machine Learning algorithms and get the best results.

Gender identification is one of the major problem speech analysis today. Tracing the gender from acoustic data i.e., pitch, median, frequency etc. Machine learning gives promising results for classification problem in all the research domains. There are several performance metrics to evaluate algorithms of an area. Our Comparative model algorithm for evaluating 5 different machine learning algorithms based on eight different metrics in gender classification from acoustic data. Agenda is to identify gender, with five different algorithms: Linear Discriminant Analysis (LDA), K-Nearest Neighbour (KNN), Classification and Regression Trees (CART), Random Forest (RF), and Support Vector Machine (SVM) on basis of eight different metrics. The main parameter in evaluating any algorithms is its performance. Misclassification rate must be less in classification problems, which says that the accuracy rate must be high. Location and gender of the person have become very crucial in economic markets in the form of AdSense. Here with this comparative model algorithm, we are trying to assess the different ML algorithms and find the best fit for gender classification of acoustic data.

A.Shivaram yadav

221710312005

Table of Contents:

List of Figures:

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 Artificial Intelligence(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

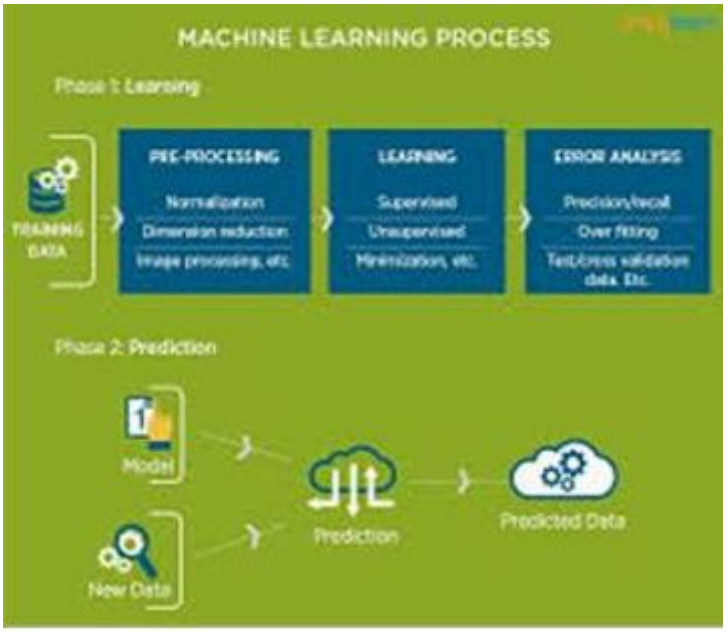


Figure 1 : The Process Flow

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.

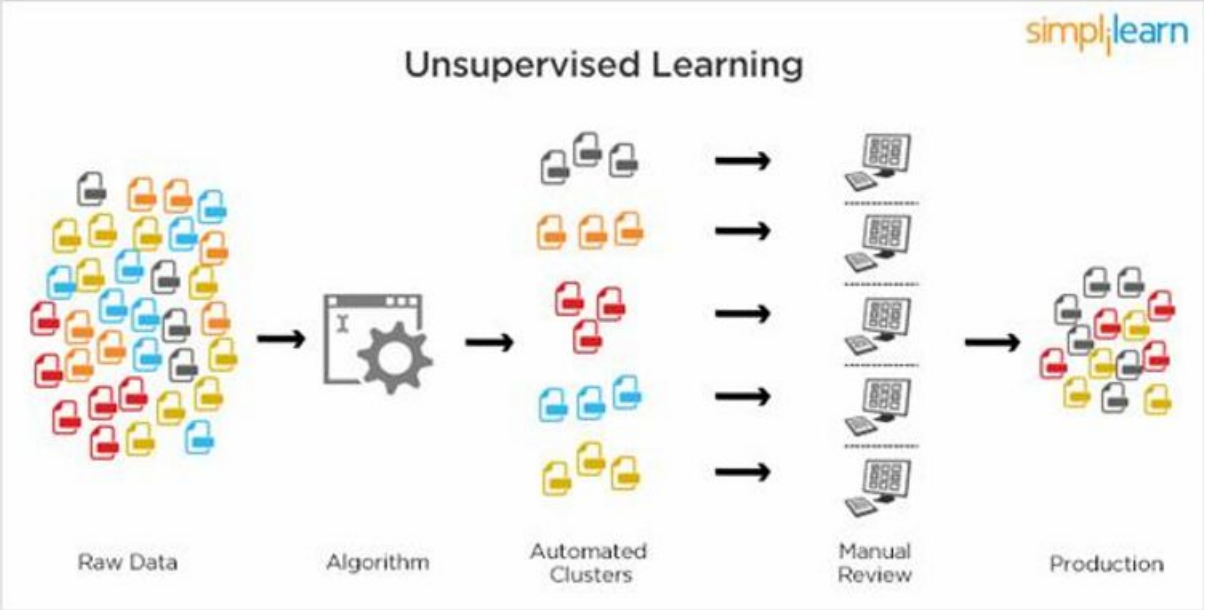


Figure 2 : Unsupervised Learning.

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.

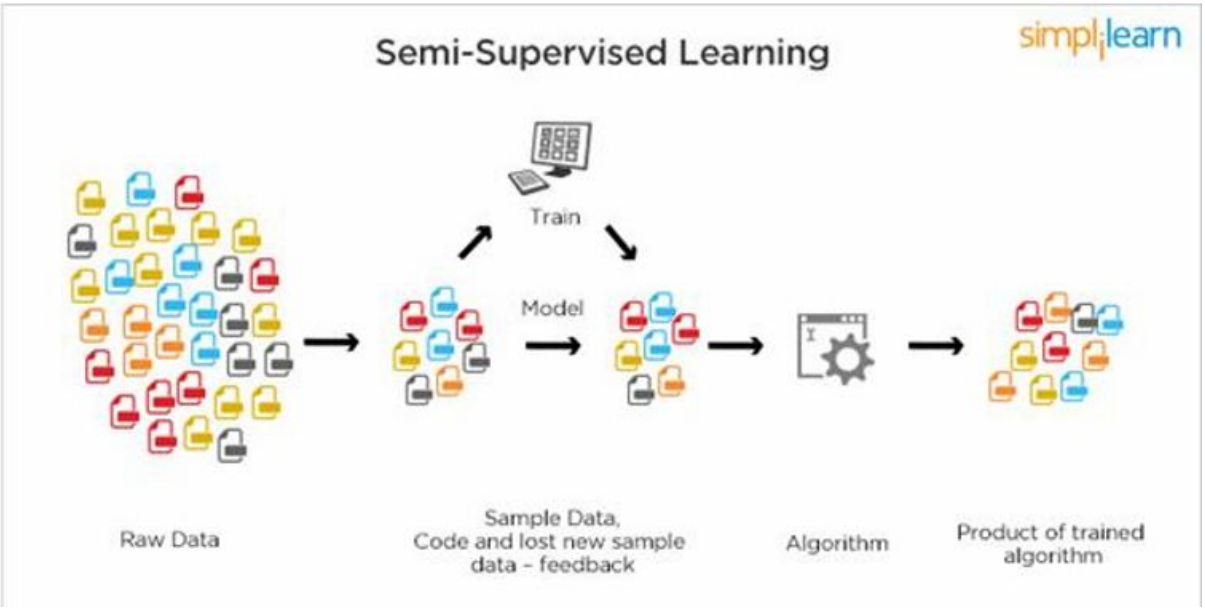


Figure 3 : Semi Supervised Learning

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

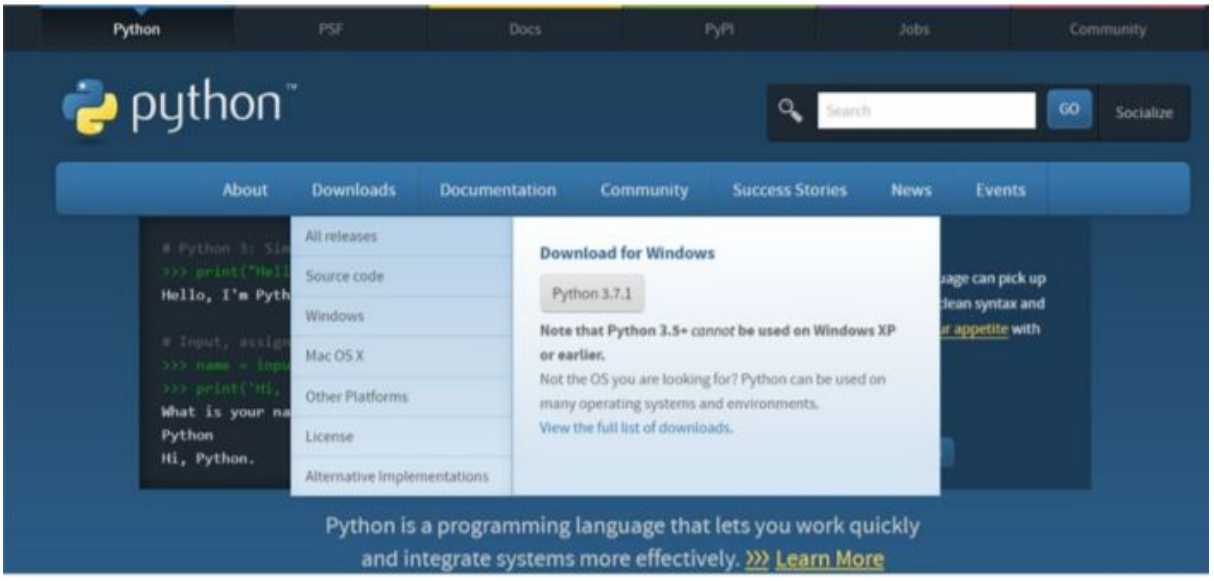


Figure 4 : Python download

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:
- Step 1: Open Anaconda.com/downloads in a 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)

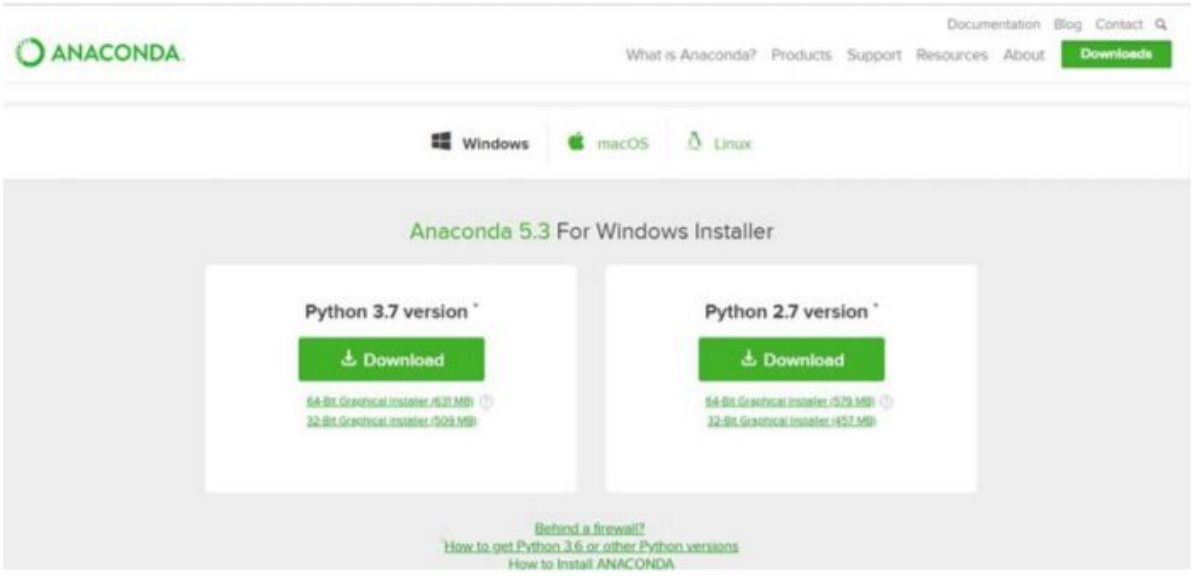


Figure 5 : Anaconda download

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 –
    - Numbers
    - Strings
    - Lists
    - Tuples
    - 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 tuples 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: o We define a class in a very similar way how we define a function. o 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 function 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 whether the voice is male or female.

3.2 DATA SET:

The given dataset contains following parameters:

Voice Gender

The **dataset** consists of 3,168 recorded **voice** samples, collected from male and female speakers. The **voice** samples are pre-processed by acoustic analysis in R using the seewave and tuneR packages, with an analyzed frequency range of 0hz-280hz (human **vocal** range).

CHAPTER-4

MODEL BUILDING

4.1 PREPROCESSING OF THE DATA:

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 the client.

4.1.2 IMPORTING THE LIBRARIES:

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

```
#importing libraries
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

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.

```
#reading the data set
df=pd.read_csv('voice.csv')
df.head(10)
```

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	meand
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	0.007
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	0.009
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	0.007
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	0.201
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	0.712
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	0.298
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	0.479
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	0.301
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	0.336
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	0.340

10 rows × 21 columns

Figure 9 : Reading the dataset

4.1.4 HANDLING MISSING VALUES:

Missing values can be handled in many ways using some inbuilt methods:

1. dropna()

2. fillna()
3. interpolate()
4. mean imputation and median imputation .

1. dropna():

dropna() is a function which drops all the rows and columns which are having the missing values(i.e. NaN).

dropna() function has a parameter called how which works as follows:

- if how = 'all' is passed then it drops the rows where all the columns of the particular row are missing.
- if how = 'any' is passed then it drops the rows where all the columns of the particular row are missing.

2. fillna():

fillna() is a function which replaces all the missing values using different ways

fillna() also have parameters called method and axis.

- if we use method = 'ffill' where ffill is a method called forward fill, which carry forwards the previous row's value .
- if we use method = 'bfill' where bfill is a method called backward fill, which carry backward the next row's value .
- if we use method = 'ffill' , axis = 'columns' then it carry forwards the previous column's value .
- if we use method = 'bfill' , axis = 'columns' then it carry backward the next column's value .

3. interpolate():

- interpolate() is a function which comes up with a guess value based on the other values in the dataset and fills those guess values in the place of missing values .

4. mean and median imputation

● mean and median imputation can be performed by using fillna().

● mean imputation calculates the mean for the entire column and replaces the missing values in that column with the calculated mean.

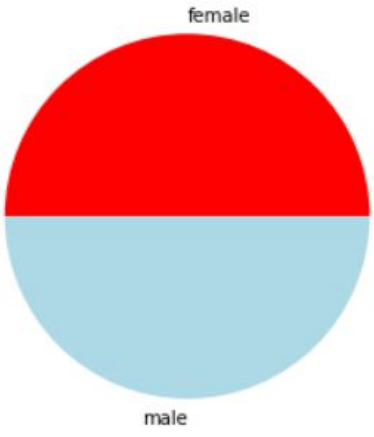
● median imputation calculates the median for the entire column and replaces the missing values in that column with the calculated median.

Missing values can be checked using isna() or isnull() functions which returns the output in a boolean format.

Total number of missing values in each column can be calculated using isna().sum() or isnull().sum().

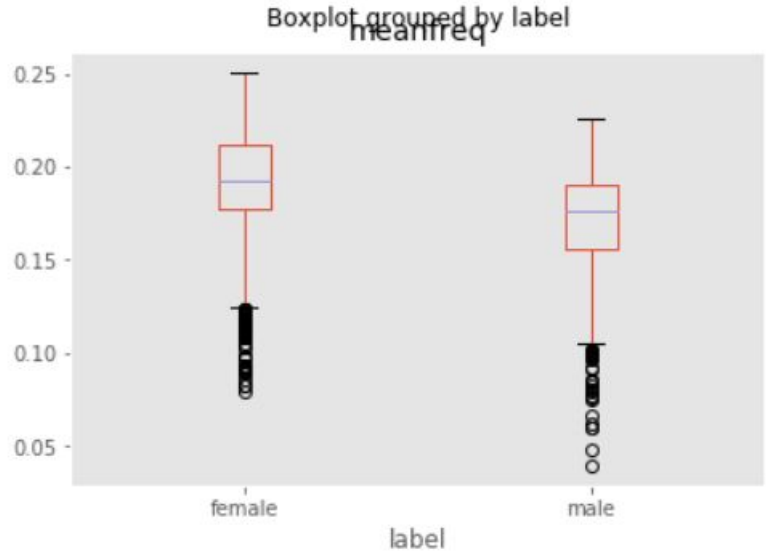
```
#distribution of target variables
colors=['red','Lightblue']
data_y=df[df.columns[-1]]
plt.pie(data_y.value_counts(),colors=colors,labels=['female','male'])
plt.axis('equal')
print(df['label'].value_counts())
```

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

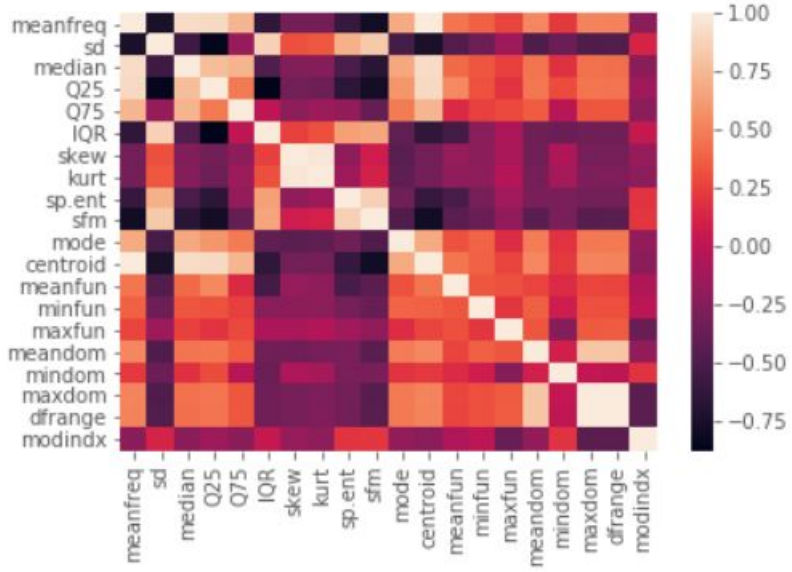


```
#Box plot comparision in labels by other features
df.boxplot(column='meanfreq',by='label',grid=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x13c547986c8>
```




```
#plot correlation matrix
correlation=df.corr()
sns.heatmap(correlation)
plt.show()
```



4.1.5 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.

```
df.dtypes
```

```
meanfreq    float64
sd          float64
median      float64
Q25         float64
Q75         float64
IQR         float64
skew        float64
kurt        float64
sp.ent      float64
sfm         float64
mode        float64
centroid    float64
meanfun     float64
minfun      float64
maxfun      float64
meandom     float64
mindom      float64
maxdom      float64
dfrange     float64
modindx     float64
label        object
dtype: object
```

4.2 TRAINING THE MODEL:

In Machine Learning and Data Science we often come across a term called Imbalanced Data Distribution, generally happens when observations in one of the class are much higher or lower than the other classes. As Machine Learning algorithms tend to increase accuracy by reducing the error, they do not consider the class distribution. This problem is prevalent in examples such as Fraud Detection, Anomaly Detection, Facial recognition etc.

Standard ML techniques such as Decision Tree and Logistic Regression have a bias towards the majority class, and they tend to ignore the minority class. They tend only to predict the majority class, hence, having major misclassification of the minority class in comparison with the majority class. In more technical words, if we have imbalanced data distribution in our dataset then our model becomes more prone to the case when minority class has negligible or very lesser recall.

Imbalanced Data Handling Techniques: There are algorithms that are widely used for handling imbalanced class distribution

1.linear regression classifier
2.LogisticRegression
3.Baysion Classifier
4.GaussianNB
5.DecisionTreeClassifier
6.RandomForestClassifier
7.confusion matrix
8.LabelEncoder

```
# train_test_split is responsible to split the data into (train and test data)
from sklearn.model_selection import train_test_split
#importing linear regression classifier
from sklearn.linear_model import LogisticRegression
#importing Baysion Classifier
from sklearn.naive_bayes import GaussianNB
#importing decision tree classifier
from sklearn.tree import DecisionTreeClassifier
#importing random forest classifier
from sklearn.ensemble import RandomForestClassifier
#importing theclassification report results
from sklearn.metrics import classification_report
#importing confusion matrix
from sklearn.metrics import confusion_matrix
#import matrix to calculate accuacy
from sklearn import metrics, neighbors
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
```

4.2.1 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) .

```
#Splitting dataset into (train,test)
X=df[df.columns[:-1]].values
#y=df['label']
y=df[df.columns[-1]].values
#70-30% of train and test
Xtrain, Xtest, ytrain, ytest = train_test_split(X,y, test_size=0.30)
```

Figure 12 :splitting data

- Then we need to import logistic regression method from linear_model package from scikit learn library
- We need to train the model based on our train set (that we have obtained from splitting)
- Then we have to test the model for the test set ,that is done as follows
  - We have a method called predict , using this method we need to predict the output for the input test set and we need to compare the output with the output test data.

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

4.3 Model Building and Evaluation

4.3.1 Logistic regression

Logistic Regression is used when the dependent variable(target) is categorical.

Logistic Regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis and it predicts the probability

Example: Yes or No, get a disease or not, pass or fail, defective or non-defective, etc.,

Also called a classification algorithm, because we are classifying the data. It predicts the probability associated with each dependent variable category.

```
clf4=LogisticRegression()
clf4.fit(Xtrain,ytrain)
y_predict4=clf4.predict(Xtest)
print(metrics.accuracy_score(ytest,y_predict4))
```

Instead of directly predicting on test data, let us see how well the model predicts the training data.

Predicting on training data

```
data_x=df[df.columns[0:20]].copy()
data2=data_x.drop(['kurt','centroid','dfrange'],axis=1).copy()
data2.head(3)
data2=data2.drop(index_to_remove,axis=0)

#y=df['label']
data_y=pd.Series(y).drop(index_to_remove,axis=0)
Xtrain, Xtest ,ytrain ,ytest = train_test_split(data2,data_y,test_size=0.30 )
clf1=RandomForestClassifier()
clf1.fit(Xtrain, ytrain)
y_pred = clf1.predict(Xtest)
print(metrics.accuracy_score(ytest, y_pred))
```

```
0.991869918699187
```

```
clf2=DecisionTreeClassifier()
clf2.fit(Xtrain,ytrain)
y_predict=clf2.predict(Xtest)
print(metrics.accuracy_score(ytest,y_pred))
```

```
0.991869918699187
```

```
clf3=GaussianNB()
clf3=clf3.fit(Xtrain,ytrain)
y_predd=clf3.predict(Xtest)
print(metrics.accuracy_score(ytest,y_predd))
```

```
0.9701897018970189
```

```
clf4=LogisticRegression()
clf4.fit(Xtrain,ytrain)
y_predict4=clf4.predict(Xtest)
print(metrics.accuracy_score(ytest,y_predict4))
```

```
0.9281842818428184
```

Test Result:

```
#cross validation with same classify as first time
test_result = cross_val_score(clf3,data2,data_y,cv=10,scoring='accuracy')
print('Accuracy obtained from 10-fold cross validation is:',test_result.mean())
```

```
Accuracy obtained from 10-fold cross validation is: 0.9637979094076655
```

```
#cross validation on the best result
test_result = cross_val_score(clf2,data2,data_y,cv=10,scoring='accuracy')
print('Accuracy obtained from 10-fold cross validation is:',test_result.mean())
```

```
Accuracy obtained from 10-fold cross validation is: 0.9906404513024721
```

4.3.2 Random forest classification

Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or the same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithms of the same type i.e. multiple decision trees, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.

The following are the basic steps involved in performing the random forest algorithm:

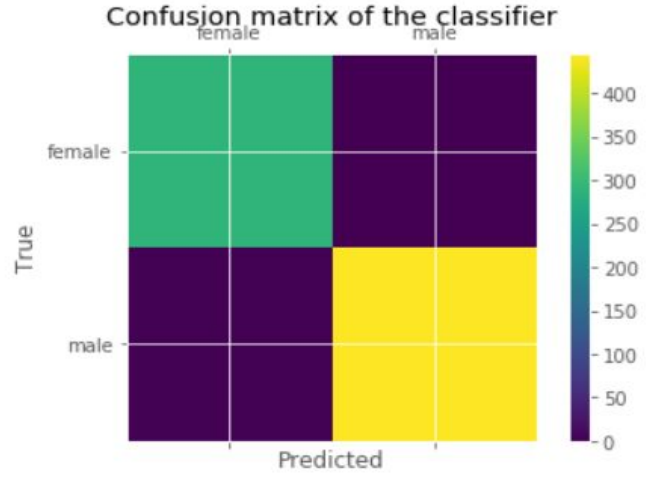
1. Pick N random records from the dataset.
2. Build a decision tree based on these N records.

3. Choose the number of trees you want in your algorithm and repeat steps 1 and 2.
4. In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

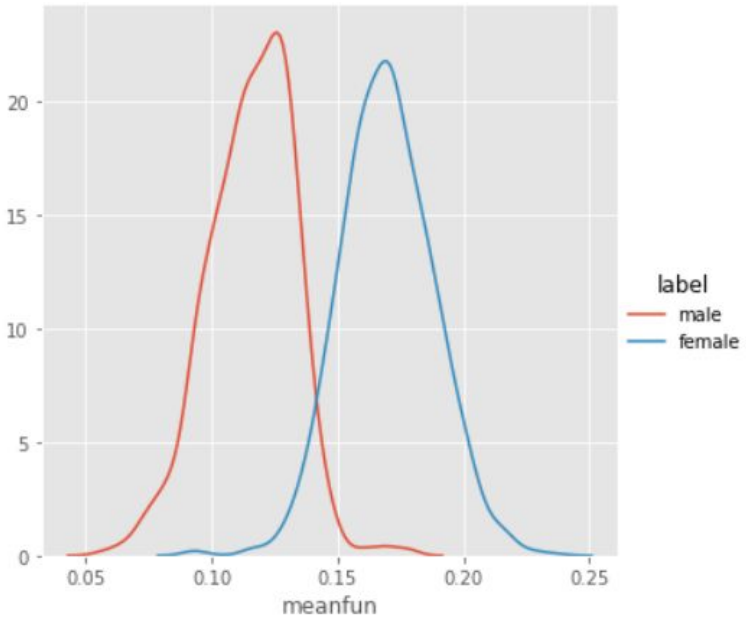
```
#Random forest classifier
rand_forest = RandomForestClassifier()
rand_forest.fit(Xtrain, ytrain)
y_pred = rand_forest.predict(Xtest)
```

```
import pylab as pl
labels =['female', 'male']
cm = confusion_matrix(ytest, y_pred,labels)
print(cm)
fig = plt.figure()
ax=fig.add_subplot(111)
cax=ax.matshow(cm)
pl.title('Confusion matrix of the classifier')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
pl.xlabel('Predicted')
pl.ylabel('True')
pl.show()
```

```
[[291   0]
 [  2 445]]
```

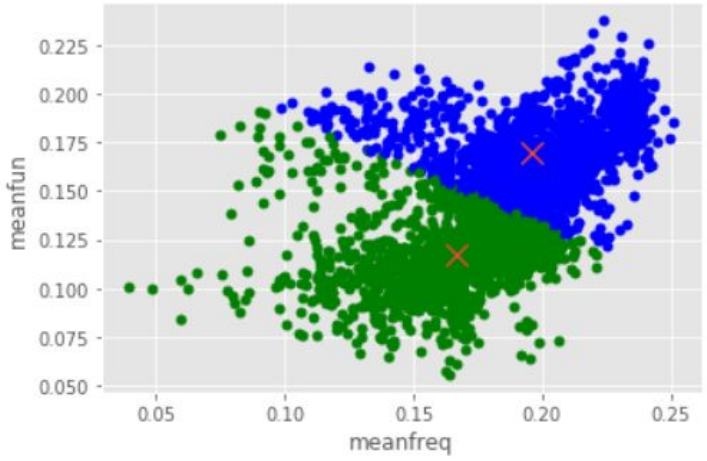


```
#Distributionof male and female
sns.FacetGrid(df, hue="label", size=5).map(sns.kdeplot, "meanfun").add_legend()
plt.show()
```



```
#Doing flat-clustering
from sklearn.cluster import KMeans
from matplotlib import style
style.use('ggplot')
data_x=np.array(df[['meanfreq','meanfun']])
kmeans=KMeans(n_clusters=2)
kmeans.fit(data_x)
centroids=kmeans.cluster_centers_
labels=kmeans.labels_
#print(centroids)
#print(labels) #0-male,1-female(the machine has assigned on its own)

colors=["g.","b."] #green=male
for i in range(len(data_x)):
    plt.plot(data_x[i][0],data_x[i][1],colors[labels[i]],markersize=10)
plt.scatter(centroids[:,0],centroids[:,1],marker="x",s=150,linewidths=5,zorder=10)
plt.ylabel('meanfun')
plt.xlabel('meanfreq')
plt.show()
```



5.Conclusion:

From the above model building and evaluation we can predict that random forest classifier is Random Forest.So the best algorithm for Voice Based Gender Classification is Random Forest.