

# **A SYSTEMATIC APPROACH TO DETECT PARKINSON'S DISEASE USING TRADITIONAL AND ENSEMBLE MACHINE LEARNING TECHNIQUES**

*Submitted for partial fulfillment of the requirements*

*for the award of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE & ENGINEERING**

**by**

<b>GOLI HEMANTH</b>	<b>-</b>	<b>19BQ1A0553</b>
<b>ELURI JAHNAVI</b>	<b>-</b>	<b>19BQ1A0547</b>
<b>ARAMALLA VASUNDHARA DEVI</b>	<b>-</b>	<b>19BQ1A0505</b>
<b>BHARATA RAJ KUMAR</b>	<b>-</b>	<b>20BQ5A0505</b>

Under the guidance of

**Dr. V. RAMA CHANDRAN, Ph.D.**

**Professor & HOD, Dept of CSE**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

(B. Tech Program is Accredited by NBA)

**VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY**

Permanently Affiliated to JNTU Kakinada, Approved by AICTE

Accredited by NAAC with 'A' Grade, ISO 9001:2008 Certified

NAMBUR (V), PEDAKAKANI (M), GUNTUR – 522 508

Tel no: 0863-2118036, url: [www.vvitguntur.com](http://www.vvitguntur.com)

April 2023

## **DECLARATION**

We, G. HEMANTH, E. JAHNAVI, A. VASUNDHARA DEVI, B.RAJ KUMAR, hereby declare that the Project Report entitled “**A SYSTEMATIC APPROACH TO DETECT PARKINSON’S DISEASE USING TRADITIONAL AND ENSEMBLE MACHINE LEARNING TECHNIQUES**” done by us under the guidance of Dr. V. Rama Chandran, **Professor & HOD, Dept of CSE** at Vasireddy Venkatadri Institute of Technology is submitted for partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science & Engineering. The results embodied in this report have not been submitted to any other University for the award of any degree.

DATE :

PLACE :

**SIGNATURE OF THE CANDIDATE (S)**

G. HEMANTH (19BQ1A0553)

E. JAHNAVI (19BQ1A0547)

A.VASUNDHARA DEVI (19BQ1A0505)

B. RAJ KUMAR (20BQ5A0505)

## ACKNOWLEDGEMENT

We take this opportunity to express my deepest gratitude and appreciation to all those people who made this project work easier with words of encouragement, motivation, discipline, and faith by offering different places to look to expand my ideas and helped me towards the successful completion of this project work.

First and foremost, we express my deep gratitude to **Mr. Vasireddy Vidya Sagar**, Chairman, Vasireddy Venkatadri Institute of Technology for providing necessary facilities throughout the B.Tech programme.

We express my sincere thanks to **Dr. Y. Mallikarjuna Reddy**, Principal, Vasireddy Venkatadri Institute of Technology for his constant support and cooperation throughout the B.Tech programme.

We express my sincere gratitude to **Dr. V. Rama Chandran**, Professor & HOD, Computer Science & Engineering, Vasireddy Venkatadri Institute of Technology for his constant encouragement, motivation and faith by offering different places to look to expand my ideas.

We would like to express my sincere gratefulness to our Guide **Dr. V. Rama Chandran, Professor & HOD**, CSE for his insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this project.

We would like to express our sincere heartfelt thanks to our Project Coordinator **J. Madhu Babu**, Associate Professor, CSE for his valuable advices, motivating suggestions, moral support, help and coordination among us in successful completion of this project.

We would like to take this opportunity to express my thanks to the **Teaching and Non-Teaching** Staff in the Department of Computer Science & Engineering, VVIT for their invaluable help and support.

**G. HEMANTH (19BQ1A0553)**

**E. JAHAVI (19BQ1A0547)**

**A. VASUNDHARA DEVI (19BQ1A0505)**

**B. RAJ KUMAR (20BQ5A0505)**



## **VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY**

Permanently Affiliated to JNTU Kakinada, Approved by AICTE

Accredited by NAAC with 'A' Grade, ISO 9001:2008 Certified

Nambur, Pedakakani (M), Guntur (Dt) - 522508

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

B.Tech Program is Accredited by NBA

### **CERTIFICATE**

This is to certify that this **Project Report** is the bonafide work of **G.HEMANTH, E. JAHNAVI, A.VASUNDHARA DEVI, B. RAJ KUMAR** bearing Reg. No. **19BQ1A0553, 19BQ1A0547, 19BQ1A0505. 20BQ5A0505** respectively who had carried out the project entitled **"A SYSTEMATIC APPROACH TO DETECT PARKINSON'S DISEASE USING TRADITIONAL AND ENSEMBLE MACHINE LEARNING TECHNIQUES"** under our supervision.

**Project Guide**

(Dr. V. Rama Chandran, Professor)

**Head of the Department**

(Dr. V. Rama Chandran, Professor)

---

Submitted for Viva voce Examination held on \_\_\_\_\_

**Internal Examiner**

**External Examiner**

## TABLE OF CONTENTS

CH No	Title	Page No
	Contents	i
	List of Figures	iv
	Nomenclature	vi
	List of Tables	vii
	Abstract	viii
<b>1</b>	<b>INTRODUCTION</b>	
	1.1 Introduction to Parkinson's Disease	1
	1.2 Symptoms of Parkinson's Disease	3
	1.3 Introduction to Machine Learning	4
	1.3.1 Supervised Learning	5
	1.3.2 Unsupervised Learning	6
	1.3.3 Applications	7
	1.4 Motivation of Work	8
	1.5 Problem Statement	8
<b>2</b>	<b>REVIEW OF LITERATURE</b>	10
<b>3</b>	<b>PROPOSED SOLUTION</b>	
	3.1 System Architecture	12
	3.2 Dataset Description	13
	3.3 System Configuration	14
	3.3.1 Software Requirements	14
	3.3.2 Hardware Requirements	14
	3.4 Feasibility Study	14
	3.4.1 Economical Feasibility	14
	3.4.2 Technical Feasibility	15
	3.4.3 Operational Feasibility	15
<b>4</b>	<b>IMPLEMENTATION</b>	

4.1 Tools Used	17
4.1.1 Anaconda	17
4.1.2 Jupyter Notebook	18
4.1.2.1 Jupyter Kernel	19
4.1.2.2 Jupyter Hub	20
4.1.2.3 Jupyter Lab	20
4.1.2.4 Local Machine	20
4.2 Libraries Used	22
4.2.1 Numpy	22
4.2.2 Pandas	22
4.2.3 Matplotlib	23
4.2.4 Seaborn	23
4.2.5 Sklearn	24
4.3 Execution	25
4.3.1 Loading Dataset	25
4.3.2 Data Pre-Processing	25
4.3.3 Training Data	26
4.3.4 Apply ML algorithms	27
4.3.4.1 KNN	27
4.3.4.2 Logistic Regression	29
4.3.4.3 SVM	32
4.3.4.4 Random Forest	34
4.3.4.5 Stacking	36
4.3.4.6 AdaBoost	38
4.3.4.7 XGBoost	39
4.3.5 Testing Data	41
<b>5 RESULTS</b>	42
5.1 Performance Metrics	42

	5.1.1 Confusion Matrix	42
	5.1.2 Accuracy	42
	5.1.3 Precision	43
	5.1.4 Recall	43
	5.1.5 F1 Score	44
	5.2 Algorithm Outputs	45
	5.3 Comparative Analysis	49
<b>6</b>	<b>CONCLUTION AND FUTURE SCOPE</b>	51
	6.1 Conclusion	51
	6.2 Future Scope	51
<b>7</b>	<b>REFERENCES</b>	53
	<b>APPENDIX</b>	
	Conference Presentation Certificate	
	Published Article in the Journal	

## LIST OF FIGURES

Figure No	Figure Name	Page No
1.1	Structure of Neuron	2
1.2	Symptoms	4
3.1	System Architecture	12
3.2	Dataset Analysis	13
4.1	Anaconda Prompt	20
4.2	Jupyter Dashboard	21
4.3	Files Execution using Jupyter Notebook	21
4.4	Reading the dataset from the CSV file into notebook	25
4.5	Renaming Columns	26
4.6	Splitting Dataset into training data and test data	27
4.7	KNN Graph-1	28
4.8	KNN Graph-2	28
4.9	Finding K Value	29
4.10	KNN Classifier	29
4.11	S-Shaped Curve	30
4.12	Logistic Regression Model	32
4.13	Hyper Plane	33
4.14	SVM Classifier	34
4.15	Working of Random Forest	35
4.16	Random Forest Classifier	36
4.17	Architecture of Stacking Model	36
4.18	Stacking Classifier	37
4.19	Working of AdaBoost Model	38
4.20	AdaBoost Classifier	39
4.21	XGBoost Classifier	41



5.1	Performance metrics of KNN model	45
5.2	Performance metrics of Logistic Regression	46
5.3	Performance metrics of SVM model	46
5.4	Performance metrics of Random Forest Model	47
5.5	Performance metrics of Stacking Model	48
5.6	Performance metrics of AdaBoost Model	48
5.7	Performance metrics of XGBoost Model	49
5.8	Comparative Analysis of all Models	50

## NOMENCLATURE

UCI	University of California Irvine
KNN	K Nearest Neighbor
SVM	Support Vector Machine
ML	Machine Learning
PD	Parkinson's Disease
DNA	Deoxyribonucleic acid
CSV	Comma Separated Value
AI	Artificial Intelligence
IT	Information Technology
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
REPL	Read Eval Print Loops

## **List of Tables**

<b>Table No</b>	<b>Table Name</b>	<b>Page No</b>
3.1	Dataset Description	13
5.1	Confusion Matrix	42

## ABSTRACT

Parkinson's disease (PD) is a neurodegenerative condition that worsens with time and affects both the neurological system and body components under the control of the nervous system. This condition causes slow movements, tremors, balance problems, and more. Currently, we have no proper cure or treatment available, but it can sometimes be cured with medication if it is diagnosed in its initial stages. Voice deterioration is also a common symptom, which often presents in the initial stages of the disease. As a result, the project 'A Systematic Approach to Detect Parkinson's Disease Using Traditional and Ensemble Machine Learning Techniques' is used to detect PD using voice data. In order to create a model that is capable of accurately identifying the disease's existence in a person's body, this project makes use of a variety of machine learning techniques, ensemble learning approaches, and Python libraries. This work aims to compare various machine learning models in the successful prediction of PD and develop an effective and accurate model to help detect the disease at an earlier stage, which could help doctors assist in the cure and recovery of PD patients. This project showed 97% efficiency. For this purpose, we plan to use the Parkinson's disease dataset in [5], which is acquired from the UCIML repository.

**Keywords:** Parkinson's disease, PD, Machine Learning, Ensemble Learning, Logistic Regression, K-Nearest Neighbour, Support Vector Machine, Random Forest, Adaboost, Stacking, XGBoost.



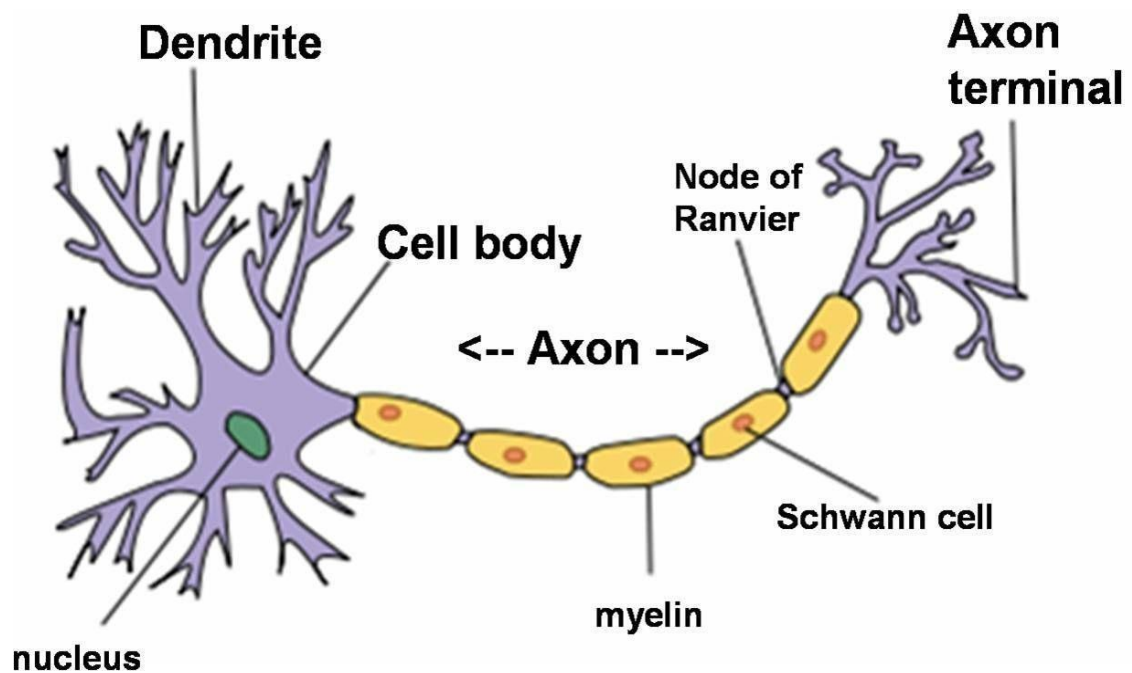
# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 INTRODUCTION TO PARKINSON'S DISEASE**

The recent report of the World Health Organization shows a visible increase in the number and health burden of Parkinson's disease patients increases rapidly. In China, this disease is spreading so fast and estimated that it reaches half of the population in the next 10 years. Classification algorithms are mainly used in the medical field for classifying data into different categories according to the number of characteristics. Parkinson's disease is the second most dangerous neurological disorder that can lead to shaking, shivering, stiffness, and difficulty walking and balance. It caused mainly due by the breaking down of cells in the nervous system. Parkinson's can have both motor and non-motor symptoms. The motor symptoms include slowness of movement, rigidity, balance problems, and tremors. If this disease continues, the patients may have difficulty walking and talking. The non-motor symptoms include anxiety, breathing problems, depression, loss of smell, and change in speech. If the above-mentioned symptoms are present in the person then the details are stored in the records. In this paper, the author considers the speech features of the patient, and this data is used for predicting whether the patient has Parkinson's disease or not.

Neurodegenerative disorders are the results of progressive tearing and neuron loss in different areas of the nervous system. Neurons are functional units of the brain. They are contiguous rather than continuous. A good healthy looking neuron as shown in fig 1 has extensions called dendrites or axons, a cell body, and a nucleus that contains our DNA. DNA is our genome and a hundred billion neurons contain our entire genome which is packaged into it. When a neuron gets sick, it loses its extension and hence its ability to communicate which is not good for it and its metabolism becomes low so it starts to accumulate junk and it tries to contain the junk in the little packages in little pockets. When things become worse and if the neuron is a cell culture it completely loses its extension, becomes round and full of vacuoles.



*Figure 1.1 : Structure of Neuron*

This work deals with the prediction of Parkinson's disorder which is now a day is tremendously increasing incurable disease. Parkinson's disease is a most spreading disease which gets its name from James Parkinson who earlier described it as a paralysis agitans and later gave his surname was known as PD. It generally affects the neurons which are responsible for overall body movements. The main chemicals are dopamine and acetylcholine which affect the human brain. There is a various environmental factor which has been implicated in PD below are the listed factor which caused Parkinson's disease in an individual.

➤ **Environmental factors:** Environment is defined as the surroundings or the place in which an individual lives. So the environment is the major factor that will not only affects the human's brain but also affects all the living organism who lives in the vicinity of it. Many types of research and evidence have proved that the environment has a big hand in the development of neurodegenerative disorders mainly Alzheimer's and Parkinson's. There are certain environmental factors that are influencing neurodegenerative disorder with high pace are:-

- Exposure to heavy metals (like lead and aluminum) and pesticides.
- Air Quality: Pollution results in respiratory diseases.
- Water quality: Biotic and Abiotic contaminants present in water lead to water pollution.
- Unhealthy lifestyle: It leads to obesity and a sedentary lifestyle.

- **Psychological stress:** It increases the level of stress hormone that depletes the functions of neurons.
- **Brain injuries or Biochemical Factors:** The brain is the control center of our complete body. Due to certain trauma, people have brain injuries which leads some biochemical enzymes to come into the picture which provides neurons stability and provides support to some chromosomes and genes in maintenance.
  - **Aging Factor:** Aging is one of the reasons for the development of Parkinson's disease. According to the author in India, 11,747,102 people out of 1, 065, 070, 6072 are affected by Parkinson's disease.
  - **Genetic Factors:** Genetic factor is considered as the main molecular physiological cause which leads to neurodegenerative disorders. The size, depth, and effect of actions of different genes define the status or level of neurodegenerative disease which increases itself gradually over time. Mainly the genetic factors which lead to Neurodegenerative disorders are categorized into pharmacodynamics and pharmacokinetics.
  - **Speech Articulation factors:** Due to the condition associated with Parkinson's disease (rigidity and bradykinesia), some speech-language pathology such as voice, articulation and swallowing alterations are found. There are various ways in which Parkinson's disease (PD) might affect the individual.
    - The voice get breathy and softer.
    - Speech may be smeared.
    - The person finds difficulty in finding the right words due to which speech becomes slower.

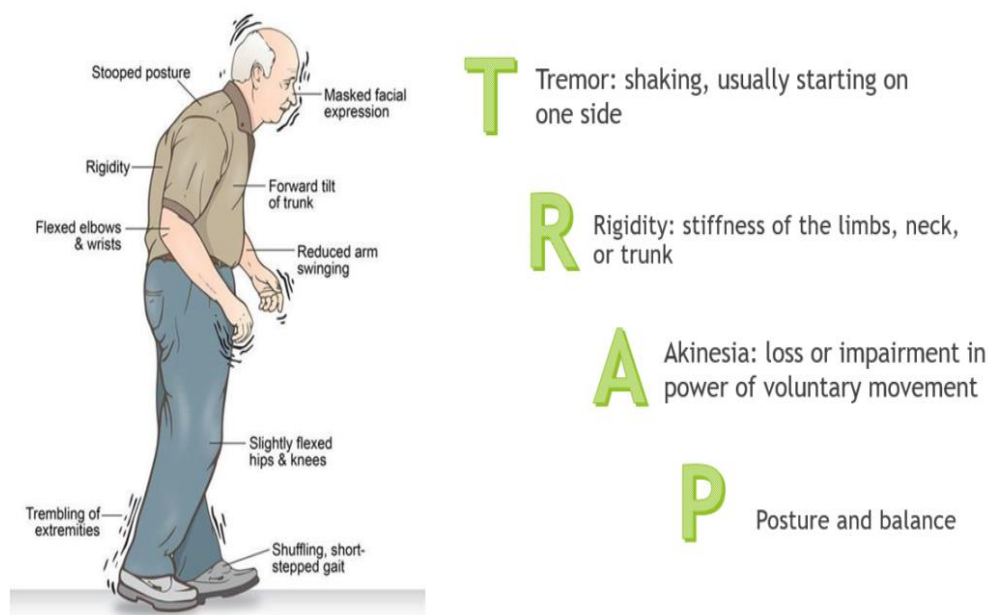
## 1.2 PARKINSON'S DISEASE SYMPTOMS

The symptoms of Parkinson's disease broadly divided into two categories.

- **Motor symptoms:** This is a symptom where any voluntary action involved. It indicates the movement-related disorders like tremors, rigidity, freezing, Bradykinesia or any voluntary muscle movement.



- **Non-Motor symptoms:** Non motor symptoms include disorders of mood and affect with apathy, cognitive dysfunction as well as complex behavioral disorders. There are two other categories of PD which are divided by doctors: Primary symptom and Secondary symptom.
- **Primary symptoms:** It is the most important symptom. Primary symptoms are rigidity, tremor and slowness of movement.
- **Secondary symptoms:** It is a symptom that directly impacts the life of an individual. These can be either motor or non-motor. Its effect depends on person to person. A very wide range of symptoms is associated with Parkinson's,. Besides these symptoms, there are some other symptoms found that lead to Parkinson's disease. These symptoms are micrographic, decreased olfaction & postural instability, slowing of the digestive system, constipation, fatigue, weakness, and Hypotension. Speech difficulties i.e. dysphonia (impaired speech production) and dysarthria (speech articulation difficulties) are found in patients with Parkinson's.



**Figure 1.2: Symptoms**

### 1.3 INTRODUCTION TO MACHINE LEARNING:

Machine Learning may be a sub-area of AI, whereby the term refers to the power of IT systems to independently find solutions to problems by recognizing patterns in databases. In other words: Machine Learning enables IT systems to acknowledge patterns in the idea of existing algorithms and data sets and to develop adequate solution concepts. Therefore, in

Machine Learning, artificial knowledge is generated on the idea of experience. In order to enable the software to independently generate solutions, the prior action of people is important. For example, the required algorithms and data must be fed into the systems in advance and the respective analysis rules for the recognition of patterns in the data stock must be defined. Once these two steps have been completed, the system can perform the following tasks by Machine Learning:

- Finding, extracting and summarizing relevant data
- Making predictions based on the analysis data
- Calculating probabilities for specific results

Basically, algorithms play a crucial role in Machine Learning: On the one hand, they're liable for recognizing patterns and on the opposite hand, they will generate solutions. Algorithms can be divided into different categories:

### **1.3.1 SUPERVISED LEARNING:**

In the course of monitored learning, example models are defined beforehand. So as to make sure an adequate allocation of the knowledge to the respective model groups of the algorithms, these then need to be specified. In other words, the system learns on the idea of given input and output pairs. within the course of monitored learning, a programmer, who acts as a sort of teacher, provides the acceptable values for specific input. The aim is to coach the system within the context of successive calculations with different inputs and outputs to determine connections.

Supervised learning is where you've got input variables (X) and an output variable (Y) and you employ an algorithm to find out the mapping function from the input to the output.  $Y = f(X)$  The goal is to approximate the mapping function so well that once you have a new input file (X) that you simply can predict the output variables (Y) for that data. It's called supervised learning because the method of an algorithm learning from the training dataset is often thought of as an educator supervising the training process. We all know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected. Learning stops when the algorithm achieves a suitable level of performance.

Techniques of Supervised Machine Learning algorithms include linear and logistic regression, multi-class classification, Decision Tree, and Support Vector Machine. Supervised

Learning problems are a kind of machine learning technique often further grouped into Regression and Classification problems. The difference between these two is that the dependent attribute is numerical for regression and categorical for classification:

➤ **Regression:**

Linear regression could also be a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and thus the only output variable (y). More specifically, that y is usually calculated from a linear combination of the input variables (x). When there's one input variable (x), the tactic is mentioned as simple linear regression. When there are multiple input variables, literature from statistics often refers to the tactic as multiple linear regression.

➤ **Classification:**

Classification could also be a process of categorizing a given set of data into classes, It is often performed on both structured or unstructured data. the tactic starts with predicting the category of given data points. The classes are often mentioned as target, label, or categories. In short, classification either predicts categorical class labels or classification data supported the training set and thus the values (class labels) in classifying attributes and uses it in classifying new data. There is a variety of classification models. Classification models include Logistic Regression, Decision Tree, Random Forest, Gradient Boosted Tree, One-vs.-One, and Naïve Bayes.

### **1.3.2 UNSUPERVISED LEARNING:**

In unsupervised learning, AI learns without predefined target values and without rewards. It's mainly used for learning segmentation (clustering). The machine tries to structure and type the info entered consistent with certain characteristics. For instance, a machine could (very simply) learn that coins of various colors are often sorted consistent with the characteristic "color" so as to structure them. Unsupervised Machine Learning algorithms are used when the knowledge used to train is neither classified nor labeled. The system doesn't determine the right output but it explores the data and should draw inferences from datasets to elucidate hidden structures from unlabeled data. Unsupervised Learning is that the training of Machines using information that's neither classified nor labeled and allowing the algorithm to act thereon information without guidance.

Unsupervised Learning is accessed into two categories of algorithms:

- Clustering: A clustering problem is where you would like to get the inherent grouping in the data such as grouping customers by purchasing behavior.
- Association: An Association rule learning problem is where you would wish to get rules that describe large portions of your data such as folks that buy X also tend to shop for Y.

### **1.3.3 Applications of Machine Learning:**

#### **Virtual Personal Assistants:**

Siri, Alexa, Google Now are a number of the favored samples of virtual personal assistants. As the name suggests, they assist find information, when asked over voice. Machine learning is a crucial a part of these personal assistants as they collect and refine the knowledge on the idea of your previous involvement with them. Later, this set of knowledge is employed to render results that are tailored to your preferences.

Virtual Assistants are integrated to a spread of platforms. For example:

- Smart Speakers : Amazon Echo and Google Home
- Smartphone : Samsung Bixby on Samsung S8
- Mobile Apps : Google Allo

#### **Videos Surveillance:**

- Imagine one person monitoring multiple video cameras! Certainly, a difficult job to try to do and boring also. This is why the thought of coaching computers to try to do this job is sensible.
- The video closed-circuit television nowadays is powered by AI that creates it possible to detect crimes before they happen. They track unusual behavior of individuals like standing motionless for an extended time, stumbling, or napping on benches, etc. The system can thus give an awareness of human attendants, which may ultimately help to avoid mishaps. And when such activities are reported and counted to be true, they assist to enhance the surveillance services. This happens with machine learning doing its job at the backend.

**Social Media Services:**

From personalizing your news feed to raised ads targeting, social media platforms are utilizing machine learning for his or her own and user benefits.

- People you may know
- Face Recognition

**Search Engine Result Refining:**

Google and other search engines use machine learning to enhance the search results for you. Every time you execute an inquiry, the algorithms at the backend keep a watch on how you answer the results. If you open the highest results and stay on the online page for long, the program assumes that the results it displayed were in accordance with the query. Similarly, if you reach the second or third page of the search results but don't open any of the results, the program estimates that the results served did not match the requirement. This way, the algorithms performing at the backend improve the search results

**1.4 Motivation of the work:**

Many of the people aged 65 or more do have a neurodegenerative disease, which has no cure. If we detect the disease in the early stages, then we can control it. Almost 30% of the patients are facing this incurable disease. Current treatment is available for patients who have minor symptoms. If these symptoms cannot be found at the early stages, it leads to death. The main cause for Parkinson's disease is the accumulation of protein molecules in the neuron which gets misfolded and hence causing Parkinson's disease. So till now, researchers got the symptoms and the root causes i.e. from where this disease had evolved. But very few symptoms have come to their cure and there are many symptoms that have no solution. So in this era where Parkinson's disease is increasing, it is very important to find the solution which can predict it in its early stages.

**1.5 PROBLEM STATEMENT:**

The main aim is to predict the prediction efficiency that would be beneficial for the patients who are suffering from Parkinson and the percentage of the disease will be reduced.

Generally in the first stage, Parkinson's can be cured by the proper treatment. So it's important to identify the PD at the early stage for the betterment of the patients. The main purpose of this research work is to find the best prediction model i.e. the best machine learning technique which will distinguish the Parkinson's patient from the healthy person. The techniques used in this problem are KNN, SVM, Logistic Regression, Random Forest, AdaBoost, Stacking and XGBoost. The experimental study is performed on the voice dataset of Parkinson's patients which is downloaded from the UCI Machine Learning Repository. The prediction is evaluated using evaluation metrics like confusion matrix, precision, recall accuracy, and f1-score.

## **CHAPTER 2**

### **REVIEW OF LIERATURE**

Speech or voice data is assumed to be 90% helpful to diagnose a person for identifying the presence of disease. It is one of the most important problems that have to be detected in the early stages so that the progression rate of the disease is reduced. Many of the researchers work on different datasets to predict the disease more efficiently. In general, Persons with PD suffer from speech problems, which can be categorized into two: hypophonia and dysarthria. Hypophonia indicates a very soft and weak voice from a person and dysarthria indicates slow speech or voice, that can hardly be understood at one time and this causes damage to the central nervous system. So, most of the clinicians who treat PD patients observe dysarthria and check out to rehabilitate with specific treatments to improvise vocal intensity. Lots of researchers did work on the pre-processing data and feature selection in the past.

Rahul R. Zaveri and Pramila M. Chawan are the authors in [1]. The datamining techniques used in this paper are Decision Tree, SVM, Naive Bayes, K-NN, and Logistic Regression. All the classifiers are compared with each other to get the highest accuracy. Using Decision Tree classifier, we achieve highest accuracy of 94%.

Raunak Sulekh and Anupam Bhatia are the researchers in [2]. The Naive Bayesian classification model was used in the current paper. A recorded speech signals dataset is analyzed in this paper whose features are voice measures. This model classifies a person with PD as 1 and healthy as 0. In this paper, the authors used the Rapid miner tool to analyze the data. An accuracy of 98.5% is recorded.

Marianna Amboni et al. proposed the paper "Using Gait Analysis' Parameters to Classify Parkinsonism: A Data Mining Approach" [3]. In this research, the authors compared the performance of two algorithms: Random Forest and Gradient Boosted Trees. Using a data mining approach, PD patients at various stages were taken into consideration and identified as typical or atypical based on gait analysis. Random Forest obtained the highest accuracy of 86.4%.

Marimuthu et.al. proposed [4] to detect PD. The proposed model used XGBClassifier as the traditional machine learning techniques failed to detect PD accurately because of its complex features. The dataset Parkinson's disease is taken from the UCI ML repository to implement the model. XGBClassifier is compared with other classifiers such as Logistic Regression, Decision Tree, and SVC. The XGBClassifier achieved an accuracy of 94.87% proving that it is better than traditional algorithms.



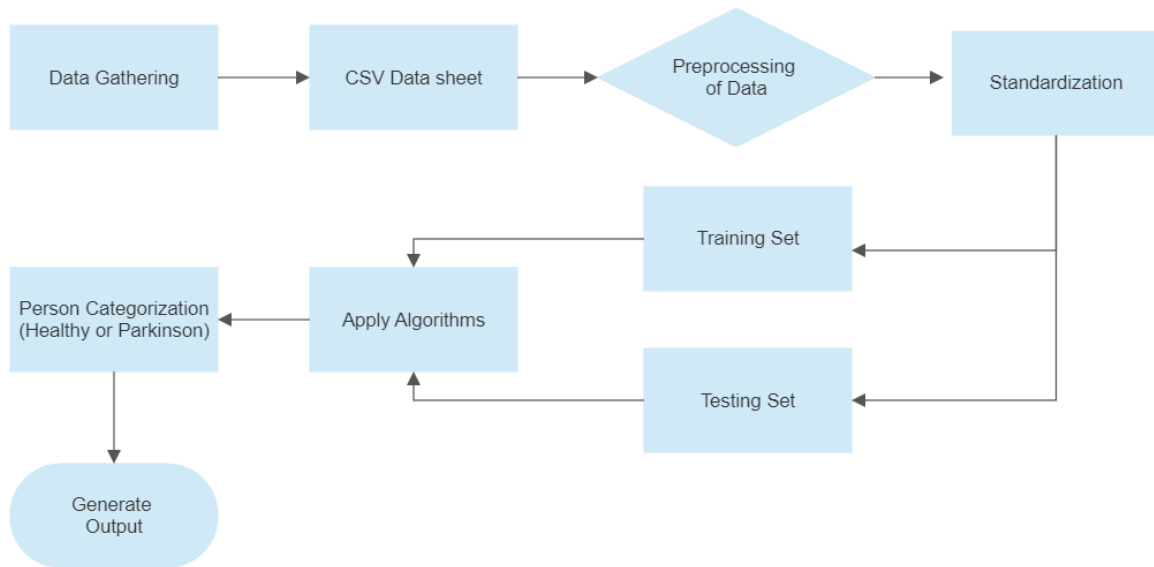
## CHAPTER 3

### PROPOSED SOLUTION

#### 3.1 SYSTEM ARCHITECTURE:

Machine learning has given computer systems the ability to automatically learn without being explicitly programmed. In this, the author has used three machine learning algorithms (Logistic Regression, KNN, and Naïve Bayes). The architecture diagram describes the high-level overview of major system components and important working relationships. It represents the flow of execution and it involves the following five major steps:

1. The architecture diagram is defined with the flow of the process which is used to refine the raw data and used for predicting the Parkinson's data.
2. The next step is preprocessing the collected raw data into an understandable format.
3. Then we have to train the data by splitting the dataset into train data and test data.
4. The Parkinson's data is evaluated with the application of a machine learning algorithms and the classification accuracy of this model is found.
5. After training the data with these algorithms we have to test on the same algorithms.
6. Finally, the result of these three algorithms is compared on the basis of classification accuracy.



*Figure 3.1: System Architecture*

### 3.2 DATASET DESCRIPTION:

The project's data is sourced from the UCI machine learning repository. Max Little of the University of Oxford created the dataset in collaboration with the National Centre for Voice and Speech in Denver, Colorado, which recorded the speech signals. This dataset is made up of a variety of biomedical voice measurements collected from a large demographic with the goal to detect Parkinson's disease (PD). Each column in the table represents a specific voice measure, and each row corresponds to one of 195 voice recordings from these people ("name" column). The primary goal of the data is to distinguish between healthy and PD people using the "status" column, which is set to 0 for healthy and 1 for PD. The data are in the CSV (ASCII) format. Each instance in the CSV file corresponds to a single voice recording. Each patient has approximately six recordings, and their name appears in the first column.

*Table-3.1: Dataset description*

Dataset	No.of.Attributes	No.of.Instances
Parkinson's Data Set	24	195

The mdvp\_fo\_hz, mdvp\_fhi\_hz and mdvp\_flo\_hz characteristics give information about the

Dataset Analysis	
Data Set Characteristics	Multi Variate
Attribute Characteristics	Real
Associated Tasks	Classification
No : of instances	195
No : of Attributes	24

*Figure 3.2 : Dataset Analysis*

voice fundamental frequency of the data. Various measurements of fluctuation in fundamental frequency are described by the variables mdvp\_jitter\_in\_percent, mdvp\_jitter\_abs, mdvp\_rap, mdvp\_ppq, jitter\_ddp. Various metrics of change in amplitude are described by the variables mdvp\_shimmer, mdvp\_shimmer\_db, shimmer\_apq3, shimmer\_apq5, mdvp\_apq, shimmer\_dda. Two measurements of the ratio of noise to tonal components in the voice are nhr and hnr. Two nonlinear dynamical complexity measures are rpde and d2. dfa is the exponent of signal fractal scaling. Three nonlinear metrics of fundamental frequency change are spread1, spread2, and ppe.

### **3.3 SYSTEM CONFIGURATION:**

#### **3.3.1 SOFTWARE REQUIREMENTS**

1. Software:
  - a. Jupyter Notebook
  - b. Google Colab
2. Operating System: Windows 10
3. Tools: Web Browser
4. Python Libraries: numpy, pandas, matplotlib, seaborn, sklearn.

#### **3.3.2 HARDWARE REQUIREMENTS**

1. RAM: 4 GB or above
2. Storage: 30 to 50 GB
3. Processor: Any Processor above 500MHz

### **3.4 FEASIBILITY STUDY:**

The preliminary investigation examines project feasibility, the likelihood the system are going to be useful to the organization. The main objective of the feasibility study is to check the Technical, Operational, and Economical feasibility for adding new modules and debugging old running systems. All systems are possible if they have unlimited resources and infinite time to do a task. There are aspects within the feasibility study portion of the preliminary investigation:

- Economical Feasibility
- Technical Feasibility
- Operational Feasibility

#### **3.4.1 ECONOMICAL FEASIBILITY STUDY:**

As system are often developed technically which are going to be used if installed must still be an honest investment for the organization. In the economic feasibility, the event cost in

creating the system is evaluated against the last word benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It doesn't require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies java1.6 open source, there is nominal expenditure and economic feasibility for certain.

### **3.4.2 TECHNICAL FEASIBILITY:**

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves evaluation of the hardware, software, and other technology requirements of the proposed system. This assessment is predicated on an overview design of system requirements, to work out whether the corporate has the technical expertise to handle completion of the project. When writing a feasibility report, the subsequent should be taken to consideration:

- A brief description of the business to assess more possible factors which could affect the study
- The part of the business being examined
- The human and economic factor
- The possible solutions to the problem At this level, the concern is whether the proposal is both technically and legally feasible (assuming moderate cost). The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system

### **3.4.3 OPERATIONAL FEASIBILITY:**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as a crucial a part of the project implementation. Some of the important issues raised are to check the operational feasibility of a project includes the following:

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits? This system is targeted to be in accordance with the above- mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

## CHAPTER 4

### IMPLEMENTATION

#### 4.1 TOOLS USED:

##### 4.1.1 ANACONDA:

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS.

**Anaconda distribution** comes with more than 1,400 packages as well as the Conda package and virtual environment manager, called **Anaconda Navigator**, so it eliminates the need to learn to install each library independently.

The open source packages can be individually installed from the Anaconda repository with the `conda install` command or using the `pip install` command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together.

Custom packages can be made using the `conda build` command and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder
- Glueviz
- Orange
- Rstudio
- Visual Studio Code

### ***Conda:***

Conda is an open source, that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The Conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

### **4.1.2 JUPYTER NOTEBOOK:**

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebooks documents. The "notebook" term can colloquially refer to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to several open standard output formats (HTML, presentation slides, LaTeX, PDF, Restructured Text, Markdown, Python) through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command line interface in a shell.

To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Jupyter Notebook provides a browser-based REPL built upon several popular open-source libraries:

- IPython
- ØMQ
- Tornado (web server)
- jQuery
- Bootstrap (front-end framework)
- MathJax

Jupyter Notebook can connect to many kernels to allow programming in many languages. By default, Jupyter Notebook ships with the IPython kernel. As of the 2.3 release (October 2014), there are currently 49 Jupyter-compatible kernels for as many programming languages, including Python, R, Julia and Haskell.

The Notebook interface was added to IPython in the 0.12 release (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is like the notebook interface of other programs such as Maple, Mathematica, and SageMath, a computational interface style that originated with Mathematica in the 1980s. According to The Atlantic, Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

#### ***4.1.2.1 Jupyter kernel:***

A Jupyter kernel is a program responsible for handling various types of request (code execution, code completions, inspection), and providing a reply. Kernels talk to the other components of Jupyter using ZeroMQ over the network, and thus can be on the same or remote machines. Unlike many other Notebook-like interfaces, in Jupyter, kernels are not aware that they are attached to a specific document and can be connected to many clients at once. Usually kernels allow execution of only a single language, but there are a couple of exceptions.



By default, Jupyter ships with IPython as a default kernel and a reference implementation via the ipykernel wrapper. Kernels for many languages having varying quality and features are available.

#### ***4.1.2.2 JupyterHub:***

JupyterHub is a multi-user server for Jupyter Notebooks. It is designed to support many users by spawning, managing, and proxying many singular Jupyter Notebook servers. While JupyterHub requires managing servers, third-party services like Jupyter provide an alternative to JupyterHub by hosting and managing multi-user Jupyter notebooks in the cloud.

#### ***4.1.2.3 JupyterLab:***

JupyterLab is the next-generation user interface for Project Jupyter. It offers all the familiar building blocks of the classic Jupyter Notebook (notebook, terminal, text editor, file browser, rich outputs, etc.) in a flexible and powerful user interface. The first stable release was announced on February 2018, and in December 2018 it was adopted as the primary interface for the cloud-based Jupyter service Jupyter.

#### **4.1.2.4 Local Machine**

In local machine the software we use to implement our project in a normal usage and by utilizing the computational power are as follows Anaconda is an local machine python environment set up software in which we can execute programs and code for the strong things in the daily tasks. Jupyter is an local machine python environment set up software in which we can execute programs and code for the strong things in the daily tasks.

Step 1: Open Anaconda prompt And execute the command “ jupyter Notebook”

```
Anaconda Prompt (Anaconda3) - jupyter notebook

(base) C:\Users\Personal>jupyter notebook
[I 07:31:24.376 NotebookApp] JupyterLab extension loaded from C:\Users\Personal\Anaconda3\lib\site-packages\jupyterlab
[I 07:31:24.376 NotebookApp] JupyterLab application directory is C:\Users\Personal\Anaconda3\share\jupyter\lab
[I 07:31:24.492 NotebookApp] Serving notebooks from local directory: C:\Users\Personal
[I 07:31:24.492 NotebookApp] The Jupyter Notebook is running at:
[I 07:31:24.493 NotebookApp] http://localhost:8888/?token=f33e47bc83107fd07ceed2d78c7c34622f63437754ce52a
[I 07:31:24.493 NotebookApp] or http://127.0.0.1:8888/?token=f33e47bc83107fd07ceed2d78c7c34622f63437754ce52a
[I 07:31:24.494 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 07:31:24.764 NotebookApp]

To access the notebook, open this file in a browser:
file:///C:/Users/Personal/AppData/Roaming/jupyter/runtime/nbserver-8368-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=f33e47bc83107fd07ceed2d78c7c34622f63437754ce52a
or http://127.0.0.1:8888/?token=f33e47bc83107fd07ceed2d78c7c34622f63437754ce52a
```

Fig 4.1 Anaconda prompt

Step 2: Goto Jupyter dashboard and navigate to the “.ipynb ” file which we need to execute.

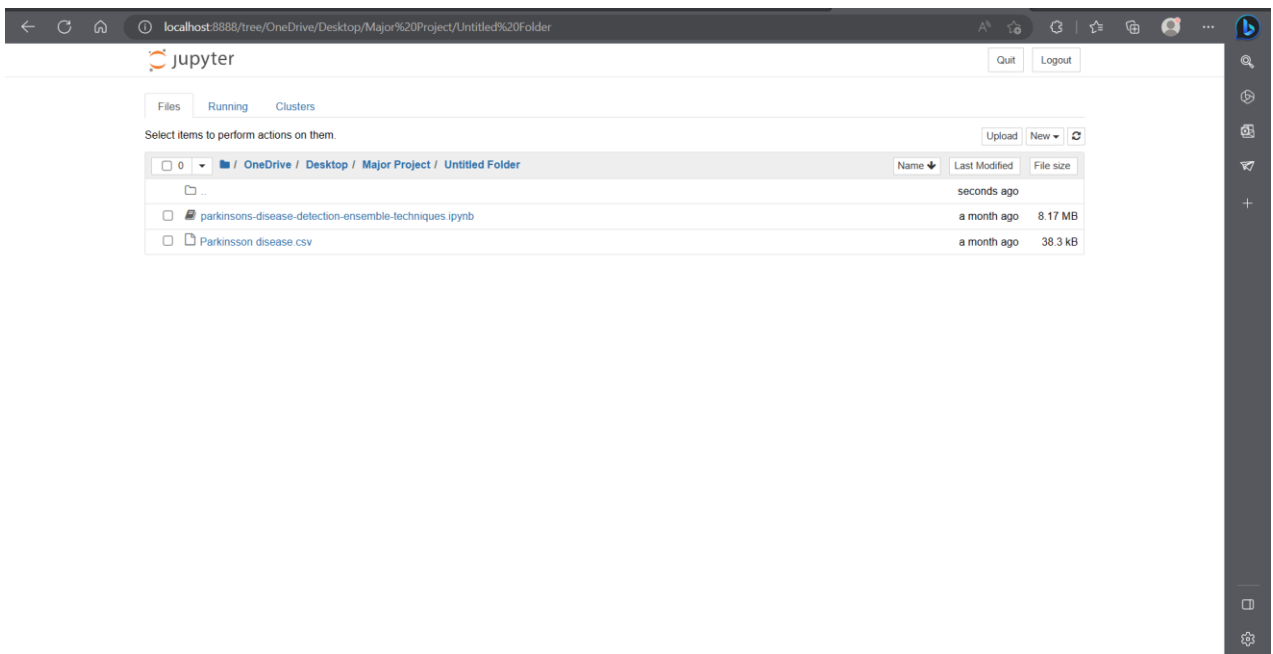


Fig 4.2: Jupyter Dashboard

Step 3: Now execute the script in the cells that are provided in Jupyter notebook interface and wait until everything is executed.



5. Integration with other libraries: NumPy integrates seamlessly with other Python libraries, such as SciPy, Pandas, and Matplotlib, making it easy to work with data.

#### **4.2.2 PANDAS:**

Pandas is a Python library for data manipulation and analysis. It provides a set of data structures and functions for working with structured data, including tables, time series, and matrices. Some of its key features include:

1. Data structures: Pandas provides two main data structures, Series and DataFrame. A Series is a one-dimensional array-like object that can hold any data type. A DataFrame is a two-dimensional table-like data structure that consists of rows and columns, similar to a spreadsheet.
2. Data manipulation: Pandas provides a variety of functions for manipulating data, including filtering, merging, sorting, and grouping data.
3. Missing data handling: Pandas provides functions for handling missing data, including filling in missing values and dropping missing values.
4. Data visualization: Pandas provides built-in visualization tools for exploring data, including line charts, scatter plots, and histograms.
5. Input/output: Pandas can read and write data from a variety of sources, including CSV files, Excel files, SQL databases, and more.

#### **4.2.3 MATPLOTLIB:**

Matplotlib is a Python library for data visualization. It provides a wide range of high-quality, customizable 2D and 3D plots for scientific and engineering applications. Some of its key features include:

1. Ease of use: Matplotlib is easy to use and has a simple interface for creating complex visualizations.
2. Wide range of plots: Matplotlib provides a wide range of plots, including line plots, scatter plots, bar plots, histograms, heatmaps, and many more.
3. Customization: Matplotlib allows for extensive customization of visualizations, including control over colors, fonts, line styles, markers, and more.

4. Integration with other libraries: Matplotlib integrates seamlessly with other Python libraries, such as NumPy and Pandas, making it easy to work with data.
5. Interactive visualization: Matplotlib can be used to create interactive visualizations using tools like Matplotlib's interactive mode, Jupyter Notebooks, and other web-based tools.

#### **4.2.4 SEABORN:**

Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for creating informative and attractive statistical graphics. Some of its key features include:

1. Built-in datasets: Seaborn comes with several built-in datasets that are ready to use, which can help users to quickly explore and visualize their data.
2. Easy customization: Seaborn allows for easy customization of plots using built-in themes, color palettes, and other styling options.
3. Statistical plots: Seaborn provides a wide range of statistical plots, including scatter plots, line plots, bar plots, heatmaps, violin plots, and more. These plots are designed to highlight patterns and relationships in data.
4. Visualizing distributions: Seaborn provides functions for visualizing and comparing data distributions, including kernel density plots, histograms, and box plots.
5. Regression analysis: Seaborn provides functions for visualizing and analyzing regression models, including scatter plots with linear regression lines, residual plots, and more.

#### **4.2.5 SKLEARN:**

Scikit-learn, or sklearn for short, is a popular machine learning library for Python. It provides a wide range of tools for machine learning, including classification, regression, clustering, and dimensionality reduction. Some of its key features include:

1. Simple and efficient tools: Scikit-learn provides simple and efficient tools for data mining and data analysis, making it easy to implement machine learning algorithms.
2. Preprocessing: Scikit-learn provides functions for preprocessing data, including scaling, normalization, and feature selection.

3. Supervised learning: Scikit-learn provides a variety of supervised learning algorithms, including decision trees, random forests, logistic regression, and support vector machines.
4. Unsupervised learning: Scikit-learn provides a variety of unsupervised learning algorithms, including clustering, dimensionality reduction, and density estimation.
5. Cross-validation: Scikit-learn provides functions for performing cross-validation, which is a technique for evaluating the performance of machine learning models.
6. Model selection: Scikit-learn provides functions for model selection, including grid search and randomized search, which are useful for finding the best hyperparameters for machine learning models.
7. Integration with other Python libraries: Scikit-learn integrates with other Python libraries, including NumPy, Pandas, and Matplotlib, making it easy to work with data.

#### **4.3 EXECUTION:**

Let us discuss about the various modules in the execution of proposed system.

##### **4.3.1 LOADING THE DATASET:**

The dataset we chose is in the form of CSV (Comma Separated Value) file. After acquiring the data our next step is to read the data from the CSV file into the Google colab also called a Python notebook. Python notebook is used in our project for data pre- processing, features selection, and for model comparison. In the figure-4.4, we have shown how to read data from CSV files using the inbuilt python functions that are part of the pandas library.

we have to import the dataset by using pandas library for easily loading and manipulating dataset.

To install pandas, use the following pip command: `pip install pandas`  
Then we have to import it by using `import pandas as pd`, where “as pd” is alias name for pandas which we choosen. In which it has the `read_csv()` function which is used to load the dataset i.e.,

```
ds = pd.read_csv('diabetes.csv')
```

where ds is variable which stores the dataset as data frames.

#### 1. Load the dataset

```
[ ] pdDataOrg = pd.read_csv("Parkinson disease.csv") # using pandas read_csv function to load dataset into pdData variable
pdDataOrg.head() # fetching and showing top 5 rows of the pdData variable
```

	name	MDVP:F0(Hz)	MDVP:F1(Hz)	MDVP:F1o(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shimmer:DDA	NHR	HNR	status
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374	...	0.06545	0.02211	21.033	1 0.4
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134	...	0.09403	0.01929	19.085	1 0.4
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233	...	0.08270	0.01309	20.651	1 0.4
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492	...	0.08771	0.01353	20.644	1 0.4
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425	...	0.10470	0.01767	19.649	1 0.4

5 rows × 24 columns

**Figure 4.4: Reading the dataset from the CSV file into notebook**

### 4.3.2 DATA PRE-PROCESSING:

The main aim of this step is to study and understand the nature of data that was acquired in the previous step and also to know the quality of data. A real-world data generally contains noises, missing values, and maybe in an unusable format that cannot be directly used for machine learning models. Data pre-processing is a required task for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model. Identifying duplicates in the dataset and removing them is also done in this step.

The modifications made as shown in the Figure-4.5 make it easier to use the data set's columns.

```
a. pushing target column i.e 'status' to last column
b. converting all column names in lower case
c. replacing spaces in column names with '_'
d. replacing ':' in column names with '_'
e. replacing '(' in column names with '_'
f. replacing ')' in column names with '' i.e blank
g. replacing '%' in column names with 'in_percent'
...

pdData = pdDataOrg.copy() # creating a copy of loanDataOrg into loanData

targetCol = 'status' # defining target column
targetColDf = pdData.pop(targetCol) # popping target column from loanData df
pdData.insert(len(pdData.columns),targetCol, targetColDf) # inserting target column to last column

# deleting variables that were used for changing column position of target column
del targetCol
del targetColDf

# converting column names into lower case
pdData.columns = [c.lower() for c in pdData.columns]
# replacing spaces in column names with '_'
pdData.columns = [c.replace(' ', '_') for c in pdData.columns]
# replacing ':' in column names with '_'
pdData.columns = [c.replace(':', '_') for c in pdData.columns]
# replacing '(' in column names with '_'
pdData.columns = [c.replace('(', '_') for c in pdData.columns]
# replacing ')' in column names with '' i.e blank
pdData.columns = [c.replace(')', '') for c in pdData.columns]
# replacing '%' in column names with 'in_percent'
pdData.columns = [c.replace('%', 'in_percent') for c in pdData.columns]

# to check the above printing top 5 rows
pdData.head()
```

**Figure 4.5: Renaming Columns**

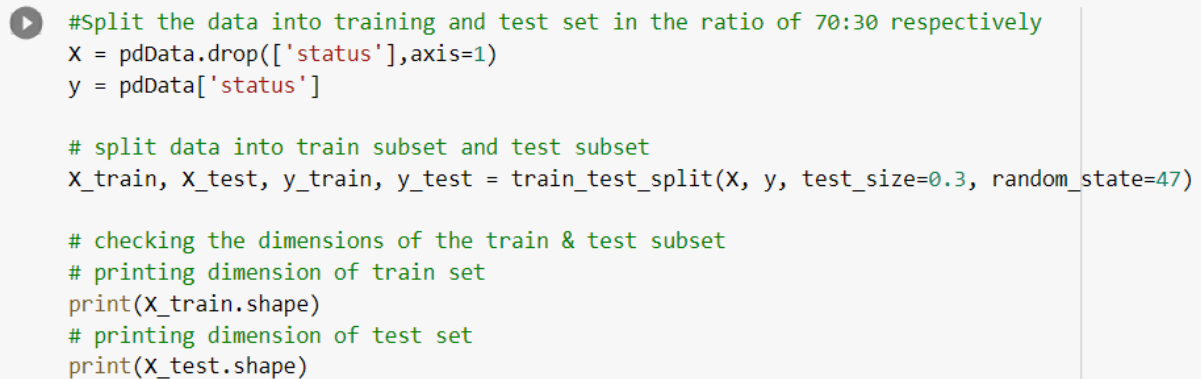
### 4.3.3 Training Data:

Splitting the dataset into Training set and testing set:

In machine learning data preprocessing, we have to break our dataset into both training set and test set. This is often one among the crucial steps of knowledge preprocessing as by doing this, we will enhance the performance of our machine learning model.

Suppose, if we've given training to our machine learning model by a dataset and that we test it by a totally different dataset. Then, it'll create difficulties for our model to know the correlations between the models.

If we train our model alright and its training accuracy is additionally very high, but we offer a replacement dataset there to, then it'll decrease the performance. So we always attempt to make a machine learning model which performs well with the training set and also with the test dataset.



```
#Split the data into training and test set in the ratio of 70:30 respectively
X = pdData.drop(['status'],axis=1)
y = pdData['status']

# split data into train subset and test subset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=47)

# checking the dimensions of the train & test subset
# printing dimension of train set
print(X_train.shape)
# printing dimension of test set
print(X_test.shape)
```

***Figure 4.6: Splitting dataset into training data and test data***

Usually, we split the dataset into train and test in the ratio of 7:3 i.e., 70 percent of data is used for training and 30 percent of data is used for testing the model. We have done it in the same way and it has been shown in the above Figure-4.6.

### 4.3.4 APPLY MACHINE LEARNING ALGORITHMS:

Now, we've both the train and test data. The subsequent step is to spot the possible training methods and train our models. As this is often a classification problem, we've used three different classification methods KNN, SVM, Logistic Regression, Random Forest, Stacking, AdaBoost, XGBoost. Each algorithm has been run over the Training dataset and their performance in terms of accuracy is evaluated alongside the prediction wiped out the testing data set.



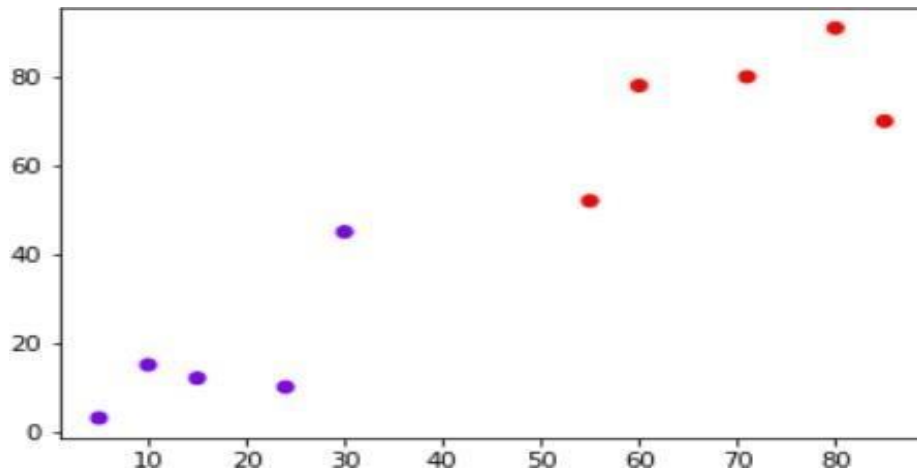
#### 4.3.4.1 K-NEAREST NEIGHBOR:

The k-nearest neighbors (KNN) algorithm may be a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand. It belongs to the supervised learning domain.

Let  $m$  be the amount of training data samples. Let  $p$  be an unknown point.

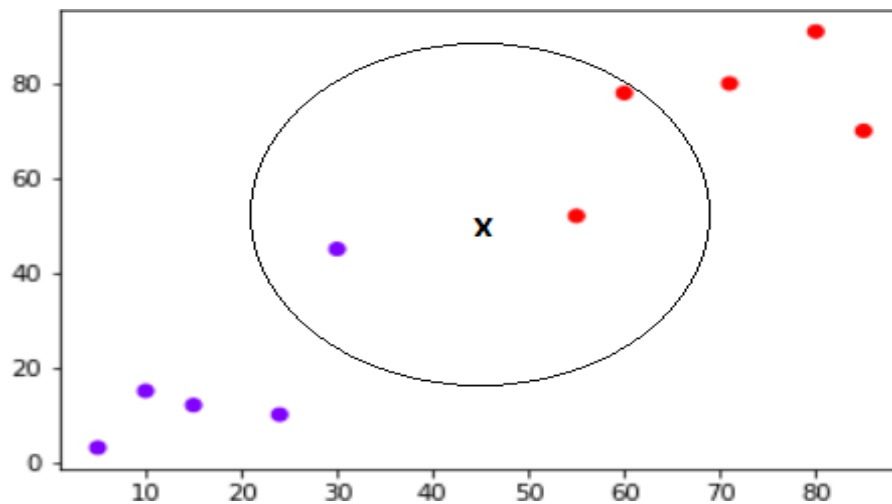
- Store the training samples in an array of data points  $arr[]$ . This means each element of this array represents a tuple  $(x, y)$ .
- for  $i=0$  to  $m$ :
- Calculate Euclidean distance  $d(arr[i], p)$
- Make set  $S$  of  $K$  smallest distances obtained. Each of those distances corresponds to an already classified datum.
- Return the majority label among  $S$ .

Let's see this algorithm can be seen with the help of a simple example. Suppose the dataset have two variables, which are plotted and shown in fig 3.9.



**Figure 4.7 : KNN Graph-1**

Your task is to classify a replacement datum with 'X' into "Blue" class or "Red" class. The coordinate values of the info point are  $x=45$  and  $y=50$ . If the  $K$  value is of 3 then the KNN algorithm starts by calculating the space of point  $X$  from all the points. Then it finds the nearest three points with least distance to point  $X$ . This process can be shown in the fig 3.10. The three nearest points in the results have been encircled.



**Figure 4.8 : KNN Graph-2**

The final step of the KNN algorithm is to assign a replacement point to the category to which the bulk of the three nearest points belong. From the figure above we can see that two of the three nearest points belong to the class "Red" while one belongs to the class "Blue". Therefore the new datum are going to be classified as "Red".

In KNN, finding the value of K is not so easy. So we used an optimal way to identify the k value through error rate. We will find the error value at each k value and from that we identify the value which gives minimal error. It is show in Figure 10.

```
# creating odd list of K for KNN
myList = list(range(3,40,2))

# creating empty list for F1 scores od different value of K
f1ScoreList = []

# perform accuracy metrics for values from 3,5....29
for k in myList:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_scaled, y_train)
    # predict the response
    y_pred = knn.predict(X_test_scaled)
    # evaluate F1 Score
    f1Score = f1_score(y_test, y_pred)
    f1ScoreList.append(f1Score)

# changing to misclassification error
MSE = [1 - x for x in f1ScoreList]

# determining best k
bestk = myList[MSE.index(min(MSE))]
print("The optimal number of neighbors is %d" % bestk)
```

**Figure 4.9: Finding K value**

In scikit-learn python library, from sklearn.neighbors import KNeighborsClassifier Module is used for carrying out the K Nearest Neighbor. We have to specify the value of K from the

above and assign an object to the classifier. We will use our training dataset to fit the model. Fig 3.12 shows the sample code for training model using K-Nearest Neighbor.

```
# instantiate learning model (k = 29)
knn = KNeighborsClassifier(n_neighbors = 29, weights = 'uniform', metric='euclidean')

# fitting the model
knn.fit(X_train_scaled, y_train)

# predict the response
knn_y_pred = knn.predict(X_test_scaled)

# Confusion Matrix for the K-nearest neighbors Model
print("Confusion Matrix : K-nearest neighbors")
print(confusion_matrix(y_test, knn_y_pred))

# Classification Report for the K-nearest neighbors Model
classRep = classification_report(y_test, knn_y_pred, digits=2)
print(classRep)
```

*Figure 4.10: KNN Classifier*

#### 4.3.4.2 LOGISTIC REGRESSION:

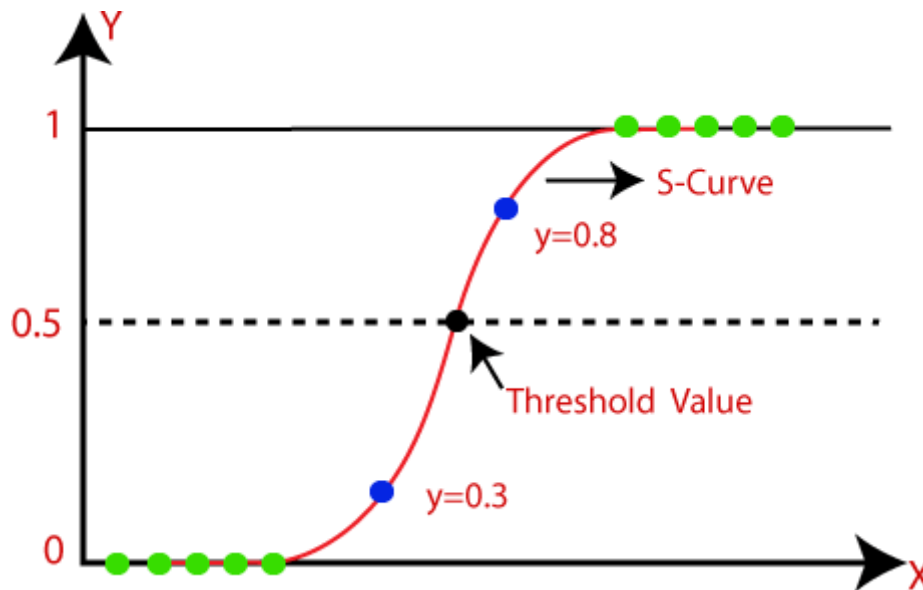
Logistic regression is additionally one among the foremost popular Machine Learning algorithms, which comes under the Supervised Learning technique. It's used for predicting the specific variable employing a given set of independent variables. It becomes a classification technique only a choice threshold is brought into the image. The setting of the edge value may be a vital aspect of Logistic regression and depends on the classification problem itself.

The decision for the worth of the edge value is majorly suffering from the values of precision and recall. Ideally, we would like both precision and recall to be 1, but this seldom is that the case. within the case of Precision-Recall tradeoffs we use the subsequent arguments to make a decision upon the threshold:

- **Low Precision/High Recall:** In applications where we would like to scale back the amount of false negatives without necessarily reducing the amount of false positives, we elect a choice value that features a low value of Precision or a high value of Recall.
- **High Precision/Low Recall:** In applications where we would like to scale back the amount of false positives without necessarily reducing the amount of false negatives, we elect a choice value that features a high value of Precision or a low value of Recall.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1). The curve from the logistic function

indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.



*Figure 4.11 : S-shaped curve*

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 variety of 0 and 1. The value of the logistic regression must be between 0 and 1, which cannot transcend this limit, so it forms a curve just like the "S" form. The S-form curve is called the sigmoid function or the logistic function which is shown above in Fig

3.14. In logistic regression, we use the concept of the edge value, which defines the probability of either 0 or 1. Such values above the threshold value tend to 1, and a value below the threshold value tends to 0.

$$F(X) = \frac{1}{1+e^x}$$

$F(x)$  = Output between the 0 and 1 value.

$x$  = input to the function

$e$  = base of natural logarithm.

On the idea of the categories, Logistic Regression are often classified into three types:

- **Binomial:** The target variable can have only 2 possibilities either "0" or "1" which may represent "win" or "loss", "pass" or "fail", "dead" or "alive", etc.

- **Multinomial:** In multinomial Logistic regression, the target variable can have 3 or more possibilities which are not ordered that means it has no measure in quantity like “disease A” or “disease B” or “disease C”.
- **Ordinal:** In ordinal Logistic regression, the target variables deals with ordered categories. For example, a test score can be categorized as: “very poor”, “poor”, “good”, and “very good”. Here, each category can be given a score like 0, 1, 2, and 3.

In scikit-learn python library, from `sklearn.linear_model` import `LogisticRegression` Module is used for carrying out the Logistic Regression. We have to specify the iterations to the function parameter and assign an object to the classifier. We will use our training dataset to fit the model. Fig 3.15 shows the sample code for training model using Logistic Regression.

```
# Train and Fit model
lr = LogisticRegression(random_state=0)
lr.fit(X_train_scaled, y_train)

#predict status for X_test_scaled dataset
lr_y_pred = lr.predict(X_test_scaled)

# Confusion Matrix for the Logistic Regression Model
print("Confusion Matrix : Logistic Regression")
print(confusion_matrix(y_test,lr_y_pred))

# Classification Report for the Logistic Regression Model
classRep = classification_report(y_test, lr_y_pred, digits=2)
print(classRep)
```

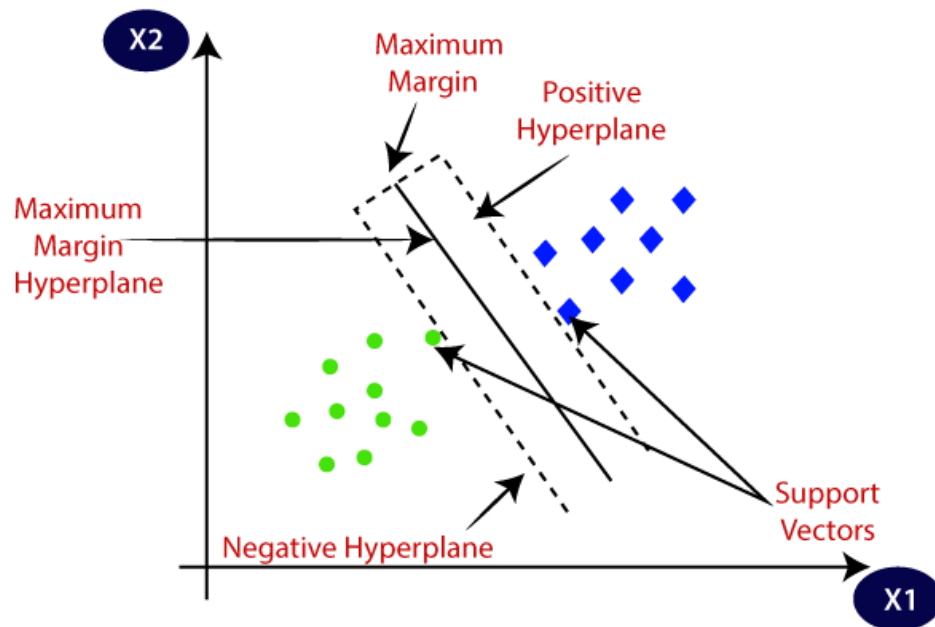
**Figure 4.12: Logistic Regression model**

#### 4.3.4.3 SUPPORT VECTOR MACHINE:

Support Vector Machine or SVM 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. The goal of the 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:



*Figure 4.13: Hyperplane*

SVM can be of two types:

1. 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.
2. 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, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Some of the advantages of SVM include:

- Effective in high-dimensional spaces
- Effective when the number of dimensions is greater than the number of samples
- Works well with both linearly separable and non-linearly separable data
- Can handle a large number of input variables
- Robust against over fitting

Some of the disadvantages of SVM include:

- Computationally intensive, especially in the case of non-linear problems

- Difficult to interpret the results of SVM
- Requires careful selection of kernel function and its parameters to achieve optimal performance
- Can be sensitive to the choice of hyper parameters

SVM has a wide range of applications, including image classification, text classification, bioinformatics, and finance.

```
[ ] svm = SVC(gamma=0.05, C=70, random_state=47)
    svm.fit(X_train_scaled, y_train)

    # predict the response
    svm_y_pred = svm.predict(X_test_scaled)

    # Confusion Matrix for the Support Vector Machine Model
    print("Confusion Matrix : Support Vector Machine")
    print(confusion_matrix(y_test, svm_y_pred))

    # Classification Report for the Support Vector Machine Model
    classRep = classification_report(y_test, svm_y_pred, digits=2)
    print(classRep)
```

*Figure 4.14: SVM Classifier*

#### 4.3.4.4 RANDOM FOREST:

Random Forest is a popular machine learning algorithm used for both classification and regression problems. It is an ensemble learning method that combines multiple decision trees to make better predictions.

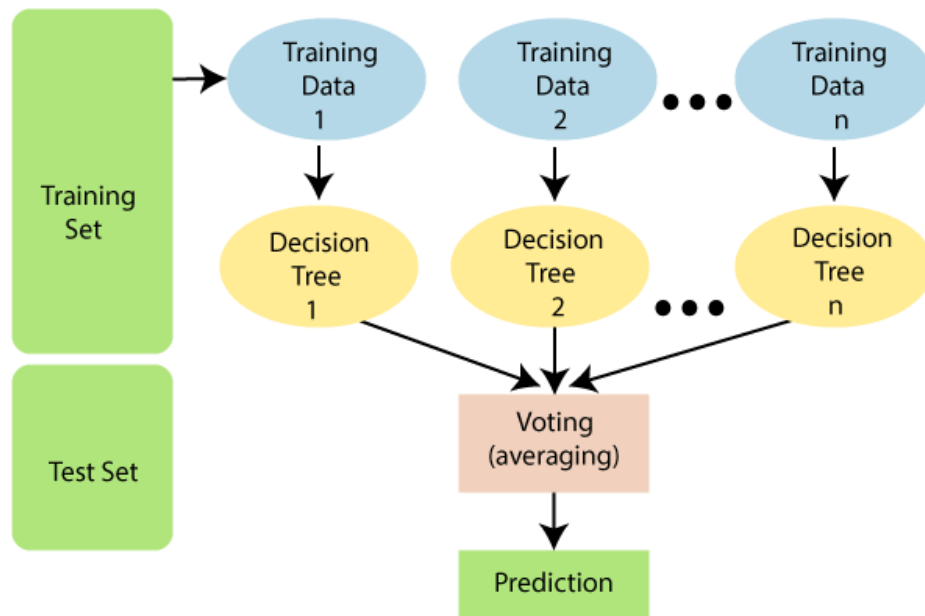
The basic idea behind Random Forest is to create multiple decision trees using randomly selected subsets of the training data and features. Each decision tree is trained on a subset of the data and features, and then the predictions of all the trees are combined to make the final prediction.

Random Forest is a highly flexible algorithm and can handle a wide range of data types and problems. Some of the advantages of Random Forest include:

- Robust against overfitting
- Can handle high-dimensional data with a large number of features
- Can handle missing data and imbalanced datasets
- Can handle both categorical and continuous data
- Provides feature importance scores that can help with feature selection

The following steps are involved in building a Random Forest model:

1. Randomly select a subset of the training data with replacement (bootstrapping)
2. Randomly select a subset of features for each tree
3. Build a decision tree on the selected subset of data and features
4. Repeat steps 1-3 for a specified number of trees
5. Combine the predictions of all the trees to make the final prediction



**Figure 4.15: Working of Random Forest**

One of the key benefits of Random Forest is that it can provide an estimate of feature importance, which can help with feature selection. The algorithm calculates the reduction in the impurity of the data (measured by Gini index or entropy) when a particular feature is used in the decision tree. The feature importance scores can be used to identify the most important features for the prediction task.

Some of the disadvantages of Random Forest include:

- Computationally intensive, especially for large datasets and a large number of trees
- Difficult to interpret the results of the model compared to a single decision tree
- Less effective for problems where the relationships between features and target variable are highly non-linear

Random Forest has a wide range of applications, including image and speech recognition, fraud detection, and credit risk analysis.



```
[ ] #creating model of Random Forest
RandomForest = RandomForestClassifier(n_estimators = 100,criterion='entropy',max_features=10,random_state=47)
RandomForest = RandomForest.fit(X_train_scaled, y_train)

# predict the response
RandomForest_pred = RandomForest.predict(X_test_scaled)

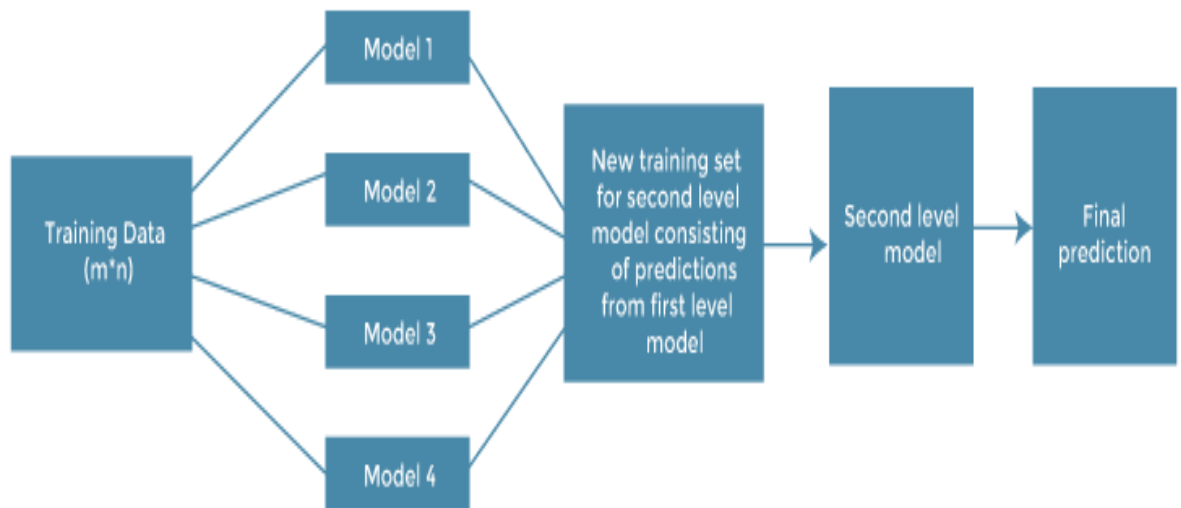
# Confusion Matrix for the Random Forest Model
print("Confusion Matrix : Random Forest")
print(confusion_matrix(y_test,RandomForest_pred))

# Classification Report for the Random Forest Model
print(classification_report(y_test, RandomForest_pred, digits=2))
```

**Figure 4.16: Random Forest Classifier**

#### 4.3.4.5 STACKING:

Stacking (short for stacked generalization) is an ensemble learning method that combines multiple predictive models to improve the overall prediction accuracy. It is a meta-learning technique that involves training a new model to learn how to best combine the predictions of individual models.



**Figure 4.17: Architecture of Stacking Model**

The basic idea behind stacking is to use the predictions of multiple base models as input to a higher-level model, which then makes the final prediction. The base models can be any type of predictive models, such as decision trees, random forests, support vector machines, neural networks, etc.

The following steps are involved in building a stacking model:

1. Split the training data into two or more subsets
2. Train several base models on one subset of the data

3. Use the other subset of the data to make predictions using the trained base models
4. Combine the predictions of the base models as input to a higher-level model
5. Train the higher-level model on the combined predictions and the target variable
6. Use the trained stacking model to make predictions on new data

The key advantage of stacking is that it can improve the overall prediction accuracy by combining the strengths of different base models. The stacking model can learn to weigh the predictions of individual models based on their performance on the training data. Stacking can also be used to handle different types of data or different types of models, allowing for more flexibility in the modeling process.

However, there are some potential drawbacks to using stacking. It can be computationally expensive, especially if the base models are complex or require a large amount of training data. Stacking also requires careful tuning of the parameters to avoid overfitting to the training data. In summary, stacking is a powerful ensemble learning technique that can improve prediction accuracy by combining the strengths of multiple base models. It is useful when dealing with complex prediction problems where no single model is able to provide satisfactory results.

```
[ ] # defining level heterogeneous model
level0 = list()
level0.append(('lr', LogisticRegression(random_state=47)))
level0.append(('knn', KNeighborsClassifier(n_neighbors = 29, weights = 'uniform', metric='euclidean')))
level0.append(('cart', DecisionTreeClassifier()))
level0.append(('svm', SVC(gamma=0.05, C=70, random_state=47)))
level0.append(('bayes', GaussianNB()))

# define meta learner model
level1 = SVC(gamma=0.05, C=3, random_state=47)

# define the stacking ensemble with cross validation of 5
Stack_model = StackingClassifier(estimators=level0, final_estimator=level1, cv=5)

# predict the response
Stack_model.fit(X_train_scaled, y_train)
prediction_stack = Stack_model.predict(X_test_scaled)

# Confusion Matrix for the Stacking Model
print("Confusion Matrix : Stacking")
print(confusion_matrix(y_test, prediction_stack))

# Classification Report for the Stacking Model
print(classification_report(y_test, prediction_stack, digits=2))
```

***Figure 4.18: Stacking Classifier***

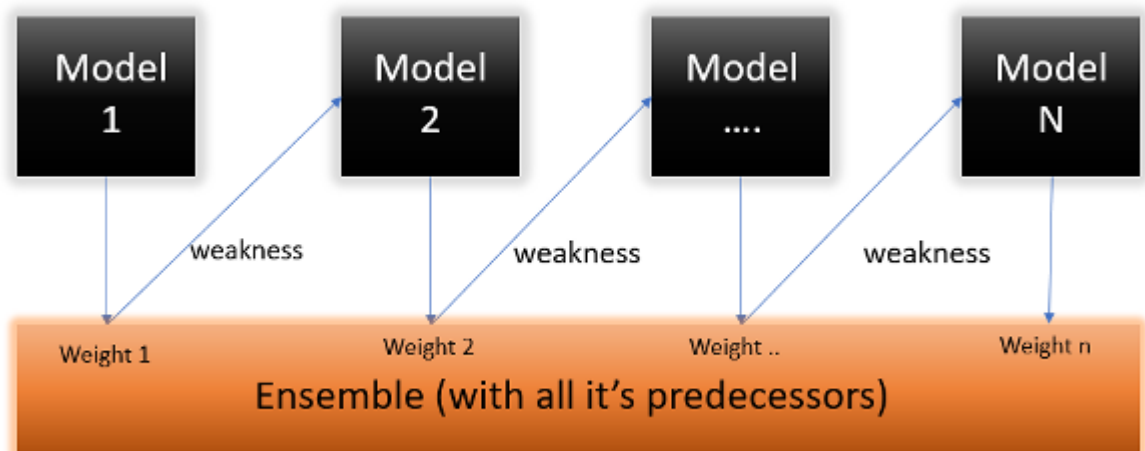
#### 4.3.4.6 ADABOOST:

AdaBoost (Adaptive Boosting) is a popular ensemble learning algorithm used for classification problems. It is a type of boosting algorithm that combines multiple weak classifiers to create a strong classifier that can make accurate predictions.

The basic idea behind AdaBoost is to iteratively train weak classifiers on different subsets of the training data, where each subset is selected based on the weights of the samples. The weights of the training samples are initially set to be equal, and are updated after each iteration to emphasize the samples that were misclassified by the previous weak classifiers. The final classifier is created by combining the weak classifiers using a weighted majority vote.

The following steps are involved in building an AdaBoost model:

1. Initialize the weights of the training data to be equal.
2. Train a weak classifier on the training data using the current weights.
3. Calculate the weighted error of the classifier, which is the sum of the weights of the misclassified samples.
4. Calculate the importance of the classifier as a function of the weighted error.
5. Update the weights of the training data based on the importance of the classifier.
6. Repeat steps 2-5 for a specified number of iterations or until the training error reaches a threshold.
7. Combine the weak classifiers to create a strong classifier using a weighted majority vote.



*Figure 4.19: Working of AdaBoost Model*

The key advantage of AdaBoost is that it can improve the performance of weak classifiers by emphasizing the samples that were misclassified by previous weak classifiers. The weights of the samples are adjusted after each iteration to give more weight to the misclassified samples, which allows the weak classifiers to focus on the samples that are difficult to classify.

Another advantage of AdaBoost is that it can handle high-dimensional data with a large number of features, which is common in many real-world applications. AdaBoost can also handle both categorical and continuous data, and is less prone to overfitting than some other machine learning algorithms.

However, AdaBoost can be sensitive to noisy data and outliers, which can affect the performance of the weak classifiers. AdaBoost can also be computationally expensive, especially for large datasets and a large number of weak classifiers. In summary, AdaBoost is a powerful ensemble learning algorithm that combines multiple weak classifiers to create a strong classifier. It is useful when dealing with complex prediction problems where no single classifier is able to provide satisfactory results.

```
[ ] #creating model of Adaptive Boosting
    AdBs = AdaBoostClassifier( n_estimators= 50)
    AdBs = AdBs.fit(X_train_scaled, y_train)

    # predict the response
    AdBs_y_pred = AdBs.predict(X_test_scaled)

    # Confusion Matrix for the Adaptive Boosting Model
    print("Confusion Matrix : Adaptive Boosting")
    print(confusion_matrix(y_test,AdBs_y_pred))

    # Classification Report for the Adaptive Boosting Model
    print(classification_report(y_test, AdBs_y_pred, digits=2))
```

*Figure 4.20: AdaBoost Classifier*

#### **4.3.4.7 XGBOOST:**

XGBoost (Extreme Gradient Boosting) is a powerful and efficient machine learning algorithm used for both regression and classification problems. It is based on the gradient

boosting framework, and is designed to improve upon the performance of traditional gradient boosting algorithms by addressing some of their limitations.

The key features of XGBoost include:

1. **Regularization:** XGBoost has a built-in regularization parameter that helps to prevent overfitting by penalizing complex models.
2. **Parallel Processing:** XGBoost can take advantage of parallel processing on a single machine or across a cluster of machines to speed up training and prediction.
3. **Tree Pruning:** XGBoost includes a mechanism for pruning trees during the model training process, which can improve the accuracy and efficiency of the algorithm.
4. **Support for Multiple Objective Functions:** XGBoost supports multiple objective functions, including regression, classification, and ranking, which makes it a versatile algorithm that can be used for a wide range of problems.
5. **Built-in Cross Validation:** XGBoost includes built-in cross-validation functionality that allows for easy tuning of hyperparameters and helps to prevent overfitting.

The XGBoost algorithm works by iteratively adding decision trees to the model, with each new tree attempting to correct the errors of the previous trees. During each iteration, the algorithm calculates the gradients of the loss function with respect to the predictions made by the current ensemble of trees. It then fits a new decision tree to the negative gradients of the loss function, which helps to correct the errors made by the previous trees.

The key difference between XGBoost and traditional gradient boosting algorithms is that XGBoost uses a more efficient and accurate approximation of the second-order gradient (the Hessian) to update the weights of the decision trees. This allows XGBoost to learn more complex and accurate models with less computational resources than traditional gradient boosting algorithms.

In summary, XGBoost is a powerful and efficient machine learning algorithm that is widely used in both academia and industry for a wide range of applications. Its ability to handle complex and high-dimensional data, its built-in regularization, and its support for

multiple objective functions make it a versatile and effective algorithm for a wide range of machine learning problems.

```
model=XGBClassifier()  
model.fit(X_train,y_train)  
predict=model.predict(X_test)  
  
print(accuracy_score(y_test,predict)*100)  
  
# Confusion Matrix for the XGBoosting Model  
print("Confusion Matrix : XGBoosting")  
print(confusion_matrix(y_test,predict))  
  
# Classification Report for the XGBoosting Model  
print(classification_report(y_test, predict, digits=2))
```

***Figure 4.21: XGBoost Classifier***

#### **4.3.5 Testing Data**

Once Parkinson's disease Prediction model has been trained on the pre-processed dataset, then the model is tested using different data points. In this testing step, the model is checked for correctness and accuracy by providing a test dataset to it. All the training methods need to be verified for finding out the best model to be used. In figures 4.10, 4.12, 4.14, 4.16, 4.18, 4.20, 4.21 after fitting our model with training data, we used this model to predict values for the test dataset. These predicted values on testing data are used for model comparison and accurate calculation.

## CHAPTER 5

### RESULTS

#### 5.1 Performance Metrics:

Performance evaluations measures are the parameters which helps in comparative analysis of different machine learning techniques i.e. it tells the best algorithm among all other algorithms or method which can be used by medical science in the early prediction of Parkinson's diseases. We have used several measures to evaluate the predictive results. These measures are Confusion Matrix, Accuracy, Precision, Recall or Sensitivity, Specificity, F1-score.

##### 5.1.1 Confusion Matrix:

The confusion matrix is also called as Error matrix. It is a table that is often used to describe the performance of a classification method on a set of test data for which actual value are known. Each column of the matrix represents the instances in a predicted class. the correlation matrix is represented as given:

	Positive	Negative
Positive	TP	FN
Negative	FP	TN

***Table 5.1 : Confusion Matrix***

Where TP: True positive

FP: False Positive

FN: False Negative

TN: True Negative

##### 5.1.2 Accuracy:

Accuracy is a common performance metric used to evaluate the performance of a classification model. It measures the percentage of correct predictions made by the model on a given set of data. The accuracy formula is:

$$\text{Accuracy} = (\text{Number of correct predictions} / \text{Total number of predictions}) * 100\%$$

For example, if a model correctly classifies 90 out of 100 test samples, the accuracy would be:

$$\text{Accuracy} = (90 / 100) * 100\% = 90\%$$

It is important to note that accuracy may not always be the best metric to use for evaluating a model's performance, especially when dealing with imbalanced datasets. In such cases, metrics like precision, recall, and F1-score may be more appropriate.

### **5.1.3 Precision:**

Precision is a common performance metric used to evaluate the performance of a classification model, particularly in cases where the focus is on minimizing false positives. It measures the proportion of positive predictions that are actually correct. The precision formula is:

$$\text{Precision} = (\text{True Positives} / (\text{True Positives} + \text{False Positives}))$$

True positives are the number of correctly predicted positive examples, while false positives are the number of incorrectly predicted positive examples.

For example, if a model predicts that 100 samples are positive, out of which 80 are actually positive and 20 are negative, and out of the 80 samples predicted as positive, 70 are actually positive and 10 are negative, then the precision would be:

$$\text{Precision} = (\text{True Positives} / (\text{True Positives} + \text{False Positives})) = (70 / (70 + 10)) = 0.875 \text{ or } 87.5\%$$

This means that out of all the samples predicted as positive, 87.5% were actually positive. It is important to note that precision should not be used in isolation, but rather in combination with other metrics like recall, F1-score, and accuracy to get a comprehensive understanding of the model's performance.

### **5.1.4 Recall (or) Sensitivity:**

Recall is a common performance metric used to evaluate the performance of a classification model, particularly in cases where the focus is on minimizing false negatives. It measures the proportion of actual positive samples that are correctly predicted as positive by the model. The recall formula is:

$$\text{Recall} = (\text{True Positives} / (\text{True Positives} + \text{False Negatives}))$$

True positives are the number of correctly predicted positive examples, while false negatives are the number of incorrectly predicted negative examples.



For example, if a model predicts that 100 samples are positive, out of which 80 are actually positive and 20 are negative, and out of the 80 actual positive samples, 70 are predicted as positive and 10 are predicted as negative, then the recall would be:

$$\text{Recall} = (\text{True Positives} / (\text{True Positives} + \text{False Negatives})) = (70 / (70 + 10)) = 0.875 \text{ or } 87.5\%$$

This means that out of all the actual positive samples, 87.5% were correctly predicted as positive by the model. It is important to note that recall should not be used in isolation, but rather in combination with other metrics like precision, F1-score, and accuracy to get a comprehensive understanding of the model's performance.

### 5.1.5 F1 Score:

The F1 score is a common performance metric used to evaluate the performance of a classification model, taking into account both precision and recall. It is the harmonic mean of precision and recall, and it provides a balanced measure of the model's performance. The F1 score formula is:

$$\text{F1 score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

Precision and recall are given by the following formulas:

$$\text{Precision} = (\text{True Positives} / (\text{True Positives} + \text{False Positives}))$$

$$\text{Recall} = (\text{True Positives} / (\text{True Positives} + \text{False Negatives}))$$

True positives are the number of correctly predicted positive examples, false positives are the number of incorrectly predicted positive examples, and false negatives are the number of incorrectly predicted negative examples.

For example, if a model predicts that 100 samples are positive, out of which 80 are actually positive and 20 are negative, and out of the 80 actual positive samples, 70 are predicted as positive and 10 are predicted as negative, then the F1 score would be:

$$\text{Precision} = (\text{True Positives} / (\text{True Positives} + \text{False Positives})) = (70 / (70 + 10)) = 0.875 \text{ or } 87.5\%$$

$$\text{Recall} = (\text{True Positives} / (\text{True Positives} + \text{False Negatives})) = (70 / (70 + 10)) = 0.875 \text{ or } 87.5\%$$

$$\text{F1 score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}) = 2 * (0.875 * 0.875) / (0.875 + 0.875) = 0.875$$

The F1 score ranges from 0 to 1, with a higher score indicating better model performance. It is a commonly used metric in applications where both precision and recall are important, such as medical diagnosis and fraud detection.

## 5.2 Algorithm Outputs:

To demonstrate the results of our project, we take the remaining test data and it is tested using three algorithms. After that our trained model is ready to predict the disease is present or not. The test accuracy is done in the Google colab which is our python notebook.

Below we described how the algorithms are processed. First, KNN algorithm is trained with the training dataset and later it was tested with the remaining test data. In Figure 5.1, a screenshot of our notebook is showing that how the process of KNN algorithms is done and the accuracy the model returns and it is of 92%.

```
Confusion Matrix : K-nearest neighbors
[[ 8  5]
 [ 0 46]]
```

	precision	recall	f1-score	support
0	1.00	0.62	0.76	13
1	0.90	1.00	0.95	46
accuracy			0.92	59
macro avg	0.95	0.81	0.86	59
weighted avg	0.92	0.92	0.91	59

**Figure 5.1: Performance metrics of KNN model**

From the above K-nearest neighbors Model, we can find out the following details:

- Accuracy of the model:- 92%
- Re-call of the model:- 100%
- Precision of the model:- 90%
- F1-Score of the model:- 95%

Second, the Logistic Regression algorithm is trained with the training dataset and later it was tested with the remaining test data. In Figure 5.2, a screenshot of our notebook is

showing that how the process of Logistic Regression algorithm is done and the accuracy the model returns and it is 86%.

Confusion Matrix : Logistic Regression

```
[[ 9  4]
 [ 4 42]]
```

	precision	recall	f1-score	support
0	0.69	0.69	0.69	13
1	0.91	0.91	0.91	46
accuracy			0.86	59
macro avg	0.80	0.80	0.80	59
weighted avg	0.86	0.86	0.86	59

**Figure 5.2: Performance metrics of Logistic Regression**

From the above Logistic Regression Model, we can find out the following details:

- Accuracy of the model:- 86%
- Re-call of the model:- 91%
- Precision of the model:- 91%
- F1-Score of the model:- 91%

Third, SVM algorithm is trained with the training dataset and later it was tested with the remaining test data. In Figure 5.3, a screenshot of our notebook is showing that how the process of SVM algorithms is done and the accuracy the model returns and it is of 95%.

Confusion Matrix : Support Vector Machine

```
[[10  3]
 [ 0 46]]
```

	precision	recall	f1-score	support
0	1.00	0.77	0.87	13
1	0.94	1.00	0.97	46
accuracy			0.95	59
macro avg	0.97	0.88	0.92	59
weighted avg	0.95	0.95	0.95	59

**Figure 2.3: Performance metrics of SVM Model**

From the above Support Vector Machine Model, we can find out the following details:

- Accuracy of the model:- 95%
- Re-call of the model:- 100%
- Precision of the model:- 94%
- F1-Score of the model:- 97%

Fourth, Random Forest algorithm is trained with the training dataset and later it was tested with the remaining test data. In Figure 5.4, a screenshot of our notebook is showing that how the process of Random Forest algorithm is done and the accuracy the model returns and it is of 95%.

```
Confusion Matrix : Random Forest
[[ 9  4]
 [ 1 45]]
```

	precision	recall	f1-score	support
0	0.90	0.69	0.78	13
1	0.92	0.98	0.95	46
accuracy			0.92	59
macro avg	0.91	0.84	0.86	59
weighted avg	0.91	0.92	0.91	59

***Figure 5.4: Performance metrics of Random Forest Model***

From the above Random Forest Model, we can find out the following details:

- Accuracy of the model:- 92%
- Re-call of the model:- 98%
- Precision of the model:- 92%
- F1-Score of the model:- 95%
- ROC-AUC : 84%

Fifth, Stacking algorithm is trained with the training dataset and later it was tested with the remaining test data. In Figure 5.5, a screenshot of our notebook is showing that how the process of Stacking algorithm is done and the accuracy the model returns and it is of 95%.

```

Confusion Matrix : Stacking
[[10  3]
 [ 0 46]]

```

	precision	recall	f1-score	support
0	1.00	0.77	0.87	13
1	0.94	1.00	0.97	46
accuracy			0.95	59
macro avg	0.97	0.88	0.92	59
weighted avg	0.95	0.95	0.95	59

***Figure 5.5: Performance metrics of Stacking Model***

From the above Stacked meta classifier Model, we can find out the following details:

- Accuracy of the model:- 95%
- Re-call of the model:- 100%
- Precision of the model:- 94%
- F1-Score of the model:- 97%
- ROC-AUC : 88%

Sixth, AdaBoost algorithm is trained with the training dataset and later it was tested with the remaining test data. In Figure 5.6, a screenshot of our notebook is showing that how the process of AdaBoost algorithm is done and the accuracy the model returns and it is of 90%.

```

Confusion Matrix : Adaptive Boosting
[[ 9  4]
 [ 2 44]]

```

	precision	recall	f1-score	support
0	0.82	0.69	0.75	13
1	0.92	0.96	0.94	46
accuracy			0.90	59
macro avg	0.87	0.82	0.84	59
weighted avg	0.89	0.90	0.90	59

***Figure 5.6: Performance metrics of AdaBoost Model***

From the above Adaptive Boosting Model, we can find out the following details:

- Accuracy of the model:- 90%
- Re-call of the model:- 96%
- Precision of the model:- 92%
- F1-Score of the model:- 94%
- ROC-AUC : 82%

Finally, XGBoost algorithm is trained with the training dataset and later it was tested with the remaining test data. In Figure 5.7, a screenshot of our notebook is showing that how the process of XGBoost algorithm is done and the accuracy the model returns and it is of 97%.

```
Confusion Matrix : XGBoosting
[[ 7  1]
 [ 0 31]]
```

	precision	recall	f1-score	support
0	1.00	0.88	0.93	8
1	0.97	1.00	0.98	31
accuracy			0.97	39
macro avg	0.98	0.94	0.96	39
weighted avg	0.98	0.97	0.97	39

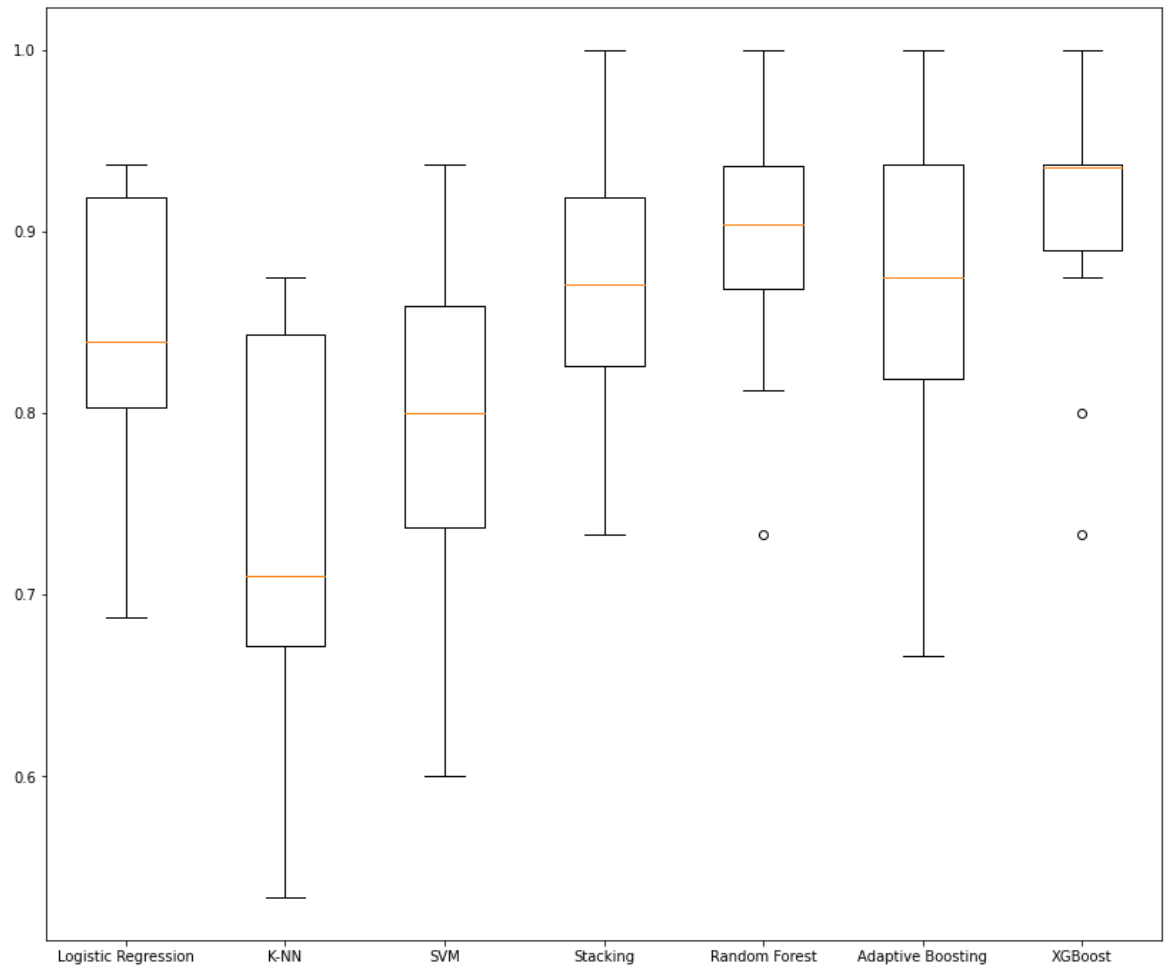
***Figure 5.7: Performance metrics of XGBoost Model***

From the above XGBoost Model, we can find out the following details:

- Accuracy of the model:- 97%
- Re-call of the model:- 100%
- Precision of the model:- 97%
- F1-Score of the model:- 98%

### **5.3 Comparative Analysis:**

The model that suites for our system has been found out through mentioned bar plot comparison. In this bar plot we shown the comparative analysis of all the models based on some of the above mentioned evaluation metrics.



***Figure 5.8: Comparative analysis of all models***

In Fig 5.8, we observed that results after applying all machine learning techniques. And it is shown that XGBosst algorithm gives the more accuracy of 97%. The XGBoost algorithm is the predicted model with more accuracy and taken into consideration.

## **CHAPTER 6**

### **CONCLUSION AND FUTURE SCOPE**

#### **6.1 Conclusion:**

In conclusion, Parkinson's disease is a neurodegenerative disorder that affects the motor system of the human body. It is characterized by symptoms such as tremors, stiffness, and slowness of movement. While there is no cure for Parkinson's disease, early detection and management can help to improve the patient's quality of life. However, the development of machine learning models for Parkinson's disease detection requires careful data collection, preprocessing, feature selection, and model selection. Moreover, the models need to be validated and fine-tuned to ensure their accuracy and reliability. Overall, machine learning has the potential to revolutionize the early detection and management of Parkinson's disease, leading to better patient outcomes and quality of life.

In this project, we have used various prediction models to predict the Parkinson's disease which are Machine Learning Techniques i.e. KNN, SVM, Logistic Regression, Random Forest, Stacking, AdaBoost, XGBoost. The dataset is trained using these models and we also compared these different models built using different methods and identifies the best model that fits.

The aim is to use various evaluation metrics such as Accuracy, Precision, Recall, Specificity and F1-score that produce the predicts the disease efficiently. We have used the Speech dataset that contains voice features of the patients which is available in the UCI Machine Learning Repository. The dataset consists of 24 features and 195 patient details. From the results, XGBoost Model outstands from the other machine learning algorithms with an accuracy of 97%.

#### **6.2 Future Scope:**

In future, these models can be trained with different datasets that have best features and can be predicted more accurately. If the accuracy rate increases, it can be used by the laboratories and hospitals so that it is easy to predict in early stages. This models can be



also used with different medical and disease datasets. In future the work can be extended by building a hybrid model that can find more than one disease with an accurate dataset and that dataset has common features of two diseases. In future the work can be extended to build a model that may extract more important features among all features in the dataset so that it produces more accuracy.

## CHAPTER 7

### REFERENCES

- [1] Rahul R. Zaveri and Pramila M. Chawan, “Prediction of Parkinson's Disease using Data Mining: A Survey”, 2021/02/21.
- [2] Dr. Anupam Bhatia and Raunak Sulekh, “Predictive Model for Parkinson’s Disease through Naive Bayes Classification”.
- [3] Marianna Amboni, et al, “Using gait analysis’ parameters to classify Parkinsonism: A data mining approach” Computer Methods and Programs in Biomedicine vol. 180, Oct. 2019.
- [4] Marimuthu M., Vidhya G., Dhaynithi J., Mohanraj G., Basker N., Theetchenya S., Vidyabharathi D. (2021). Detection of Parkinson’s disease using Machine Learning Approach. Annals of the Romanian Society for Cell Biology, 2544–2550.
- [5] Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection', Little MA, McSharpy PE, Roberts SJ, Costello DAE, Moroz IM. BioMedical Engineering OnLine 2007.
- [6] <https://numpy.org/devdocs/reference/index.html>
- [7] <https://pandas.pydata.org/docs/reference/index.html>
- [8] <https://matplotlib.org/stable/api/index.html>
- [9] <https://scikit-learn.org/stable/>
- [10] <https://www.ibm.com/in-en/topics/knn>
- [11] <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>
- [12] <https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc>
- [13] <https://www.ibm.com/in-en/topics/random-forest>
- [14] <https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/>
- [15] <https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/>
- [16] <https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/>

## APPENDIX

### Conference Presentation Certificate

