Parkinson's Disease Detection

Major project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

By

ULLAS KUMAR BHERAV (181205)

GROUP NO. 01

UNDER THE SUPERVISION OF

DR. JAGPREET SIDHU

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, 173234, Himachal Pradesh, IND

Certificate

Candidate's Declaration

I hereby declare that the work presented in this report entitled "Parkinson's Disease Detect" is in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering/Information Technology submitted in the Department of Computer Science &amp; Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2015 to December 2015 under the supervision of Dr. Jagpreet Sidhu Assistant Professor (SG) Department of Computer Science and Engineering.
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

ULLAS KUMAR BHERAV
Ullas Kumar Bherav, 181205

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Dr. Jagpreet Sidhu
Assistant Professor (SG)
Computer Science and Engineering
Dated:

ACKNOWLEDGMENT

First, I express my deepest gratitude and gratitude to Almighty God for His divine blessing enabling us to complete the project successfully.
Thank you very much and I wish my deepest thanks to Supervisor Dr. JAGREET SIDHU, Assistant Professor (SG), Department of CSE Jaypee University of Information Technology, Waknaghat. His enduring patience, expert guidance, constant encouragement, constant watchfulness and enthusiasm, constructive criticism, valuable advice, reading a lot of low-level writing, and editing at all stages made it possible to complete the project.
I would like to extend my deepest gratitude to Dr. JAGPREET SIDHU, Department of CSE, for his kind assistance in completing my work.
I would also kindly welcome each of those people who have helped me directly or indirectly in making this project a success. In this unique situation, I would like to thank the various staff, educators, and non-instructors, who have helped me in my work.
Finally, I must admit that I have a lot of respect for the support of my parents, teachers, and fellow classmates.

Ullas Kumar Bherav

Contents

LIST OF ABBREVIATIONS

- **PD:** Parkinson’s Disease.
- **LR:** Linear Regression.
- **LOR:** Logistic Regression.
- **DT:** Decision Tree.
- **RF:** Random Forest.
- **XGB:** XGBoost.
- **NN:** Neural Network.
- **CM:** Confusion Matrix.
- **TP:** True Positives.
- **TN:** True Negatives.
- **FP:** False Positives.
- **FN:** False Negatives.
- **TPR:** True Positive Rate.
- **TNR:** True Negative Rate.
- **PPV:** Positive Predictive Rate.
- **NPV:** Negative Predicted Rate.
- **FPR:** Fall-Out/ False Positivity Rate.

List Of Figures

List Of Graphs

ABSTRACT

Voice-based biomarkers can help diagnose symptoms of dementia such as Parkinson's disease, PD is a modern neurodegenerative disease affecting about 7 million people worldwide (usually adults), with about 150 thousand new scientific diagnoses performed each year. Historically, PD has been difficult to find and documents tend to focus on a few symptoms and even ignore some, depending on the scores of independent points. Due to the decline in motor manipulation which is a sign of illness, the term can be used as a means of detecting and diagnosing PD. Common sense has meant that physicians often focus on the symptoms of PD while ignoring the other. By using independent measurement scales, the term can be used to diagnose and diagnose the disease. This paper presents evidence to support the concept of supervised classification, which can be used to diagnose individuals with diseases such as diabetes and pulmonary fibrosis. Through Linear Regression, Logistic Regression, Decision Trees, Support Vector Machine, Random Forest, XGBoost, Neural Network and Adaboost we were able to achieve a peak accuracy of 100% for diagnosing pathological conditions. The project also uses various Evaluation Methods and Metrics such as Confusion Matrix, Classification Report, F1 - Score, Accuracy, Precision, Recall.

CHAPTER 01

1. Introduction

Parkinson's disease (PD) shows loss of life of dopaminergic neurons in substantia nigra pars compacta in the midbrain. These neurodegeneration modifications end up with numbers and symptoms that include communication problems, voice adjustment, and allergies. Dysarthria is also seen in PD sufferers; its miles are characterized by weakness, paralysis, and a lack of communication within the motor vehicle: affecting breathing, crying, speech, and prosody. The exact reason for Parkinson's disease is unknown, but it is a concept that involves complex interactions between genetics, biology, and the environment, which not only leads to the heterogeneity of PD signs and symptoms, but also to their charge of progression through the years. This brings uncertainty to sufferers' destiny first-class of existence, and it also brings challenges to treatment trials in determining successful endpoints. More and more reports of changes in peripheral immune machine function in sufferers of Parkinson's sickness. In recent years, there's evidence that a protein referred to as α-synuclein, the enteric fearful machine, and the intestinal brain axis are associated with the etiology of Parkinson's disorder, which suggests that Parkinson's ailment can also even start from the outer edge. These studies can also provide a possibility to pick out markers that are expecting the progression of Parkinson's sickness. Since signs and the disorder path vary, PD is usually now not available for decades. Therefore, there may be a need for more sophisticated diagnostic equipment to diagnose PD because, as the disease progresses, additional symptoms appear that make PD harder to deal with. The main shortcomings of PD speech are loss of depth, tone of voice and voice, reduction of pressure, hallucinations, short speech speeds, dynamic charging, blurred consonants, and a violent and breathing voice (dysphonia). The wide range of voice-related features promises a powerful tool because voice recording is non-invasive and can be done without difficulty with mobile gadgets. This business is evaluating the effectiveness of the use of controlled type algorithms, as well as Logistic Regression, Vector Support Machines, Decision Trees, Random Forest, XGBoost, Neural Network for better identification of people with the disease.

2. Design of Problem Statement

Parkinson’s disease is a neurological disorder that affects the motion of the body, the disease can lead to bad posture, shaky hands, stiffness in muscles, and hallucinations. The detection of disease is difficult in young people and disease is common in elderly groups. Medical statistics has developed a huge scale of quantity from distinct clinical regions such as fitness care services. To handle these statistics and accomplish insights from these statistics there is a want of Big Data evaluation through Machine Learning knowledge of that goal to solve various medical and clinical hassle. Already, many of the studies show that ML knowledge of algorithms has received meaningfully excessive overall performance in classification-primarily based medical troubles. However, supervised ML strategies are one of the handiest techniques for the studies community and actual-life programs in clinical fields. This work's important objective is to enhance the detection and analysis strategies of Parkinson's disease treatment. Parkinson's disease cannot be cured; however, medicinal drugs can help manipulate your signs and symptoms, regularly dramatically. So, if it is detected within the early stage, the cost of drugs will reduce. Therefore, our observation may be playing a vital position in detecting Parkinson's disease with Machine Learning algorithms [1]

. The detection of disease is important therefore the purpose of our major project is to develop a prediction model of Parkinson’s disease. We are going to achieve our goal with the help of the following parameters.

- Working on a Parkinson’s dataset that will contain voice data parameters of different people.
- Working on the dataset and making it suitable for data analysis.
- Using different machine learning libraries like NumPy matplotlib, seaborn, pandas, XG boost, Scikit Learn.
- Visualizing the results by confusion matrix, classification report, F1 score, accuracy, recall, and precision

3. Objective of the Minor Project

PD is difficult to detect early because of the first hidden symptoms. There is a huge burden on patients and the resilience system due to delays in diagnosis. The problem in early PD analysis has encouraged researchers to expand the test gear calculations in automated algorithms to separate healthy controls in people with PD. This binary diagnosis specializes in one-step verification of visible biomarkers in disease identification and control; now it does not offer the kind of alternative analysis where the version may also distinguish PD between a host of symptoms that manifest symptoms such as PD (e.g., Lewy-Body Dementia, Essential Tremor). Modern studies are a promising first step toward the long-term goal of introducing a preferred set of physician guidelines in screening patients with PD.

4. Methodology of the Major Project

PD is difficult to detect, there are many methods to detect the disease but voice sampling is one of the most progressive and best methods for disease prediction. In order to make this project, first, I took a dataset (which one gathered from 188 patients consisting of both men and women in this data set 64 were healthy individuals while others as having PD. various algorithms were applied to data such as Time-Frequency Features, MFCC, and TWQT. The data set was developed by Sankar, C.O., et al. [2]. This data afterward went through data cleaning and data standardization and a train test split was performed on the data set. After this series of supervised, unsupervised, and deep learning techniques were applied to the data set and the accuracies were compared to find the best model to predict PD. The algorithm used are mentioned below:

- Linear regression.
- Logistic regression.
- Decision tree.
- Support vector machine.
- Random forest.
- XG boost.
- Neural networks.
- Adaboost

Fitness score, accuracy score and confusion metrics are used as evaluation parameters during the entire project.

CHAPTER 2

1. Literature Review

In general PD is defined as progressive and chronic neurodegenerative disorder and effects humans with average high age [3,4]. According to a report published in 2015 by Parkinson's diseases foundation there are around 10 million people globally are suffering with PD out of which around 10 million are in United States and according to another report from Parkinson's Disease Society there is one in every 500 British people suffering with PD. Over time the disease worsens and affects people of age 50-70. PD is named after James Parkinson a British physician in 1817 [5,6].

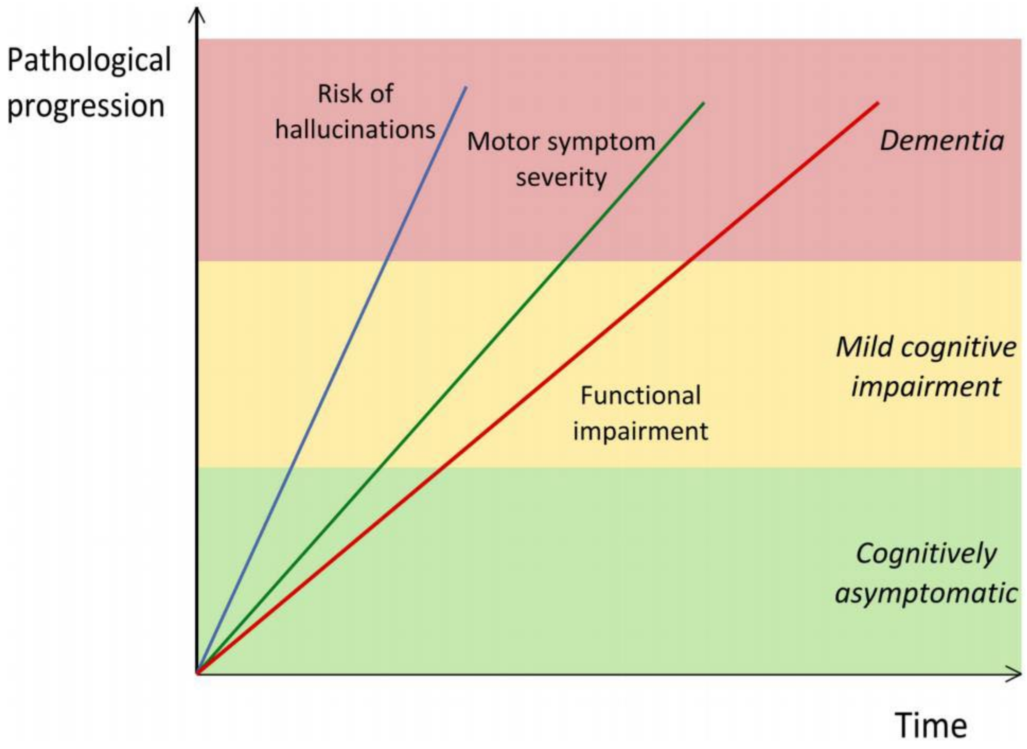
Types of PD and their Symptoms

There are mainly 2 types of PD;

1. Movement disorder (motor symptoms)
2. Cognitive dysfunction (non-motor symptoms) [7].

Motor symptoms: symptoms include cardinal symptoms, bradykinesia, rigidity, and postural instability. 70% of patients suffers with resting tremor with 3-5 HZ and is called as asymmetrical tremor. Another symptom of PD includes a feeling of resistance while joint movements called as cogwheel rigidity [8]. Third type of symptom include bradykinesia in which the handwriting of the patients is affected. In postural instability in which the patient loses control over the body balance and becomes unstable [8;9].

Non-motor symptoms: symptoms include hyposmia, sleep behavior disorder, rapid eye movement (REM), constipation, and depression [7]. Many patients suffer cognitive dysfunction which range from mild cognitive impairment (PD-MCI) to PD dementia (PDD) [10]. In many cases PD-MCI occurs at early stage whereas PDD occur approximately after 20 years of PD. In PD-MCI patients' memory and thinking is becomes abnormal from what is expected in normal aging and its diagnosis is importance as it will lead to PDD [7]. On the other hand, PDD symptoms include short-term memory, attention impairment, executive dysfunction, visual-spatial deficit, neuropsychiatric symptoms such as hallucinations, personality, and mood change [7].



PD Diagnoses

For diagnosing the PD, it is important to differentiate between Parkinsonism and PD. Parkinsonism is the clinical representation of the symptoms that are attributed to PD like rigidity, tremors, and bradykinesias. Whereas, PD is a form of parkinsonism and hence, not all the patients with parkinsonism will have PD as it might be due to other health issue.

It is difficult to differentiate PD from other form of parkinsonism based on motor symptoms and hence it becomes necessary to monitor other symptoms as well, to diagnose the disease [11].

Diagnostic criteria for the same faces many obstacles:

1. The focus of this criteria is movement problems. Non-motor symptoms are also associated with PD and this do not respond to levodopa treatment [11].
2. Measurement of Dopamine replacement is an important criterion which also occurs in other form of parkinsonism [11].

Machine learning methods have been proven to be decent for early detection of PD. These algorithms are used since many years for the detection of disease. For instance, in the year 1997 Lafunte and his team applied ANN to segregate the healthy patients and the patients with lower limb arthritis with 80% accuracy[12]. In the same year Barton and Lee used the

Kinematic analysis to distinguish the gait patterns for three conditions and their study was successful in assigning the gait patterns in right category with accuracy of 83.33% and had also proven that ANN is well capable to be used in clinical practice for Disease diagnose [13]. In the year 2006 Begg and Kamruzzaman had obtained 83.3% accuracy with ANN in identifying and classifying the moving patterns changes with respect to aging [14].

In the year 2014 Indira R. and her team had anticipated an automatic machine learning method to detect PD based on speech of the person by applying f-uzzy C-mean clustering and pattern recognition in order to identify healthy people and the people with PD with accuracy of 68.04%, specificity 45.83%, and sensitivity 75.34% [15]. Geeta R and her team in 2012 had explored and executed feature relevance analysis in order to calculate scores categorize the Tele-monitoring datasets of Parkinson diseases and dataset comparison modules Motor-UPDRS and Total-UPDRS (Unified Parkinson Disease Rating scale) [16]. Rubén A. et al. (2013) had used a wrapper feature selection scheme to anticipate five different paradigms classes which could predict each class variables with an accuracy of 72-92% [17].

A. Tsanas et al. in 2011 had proposed support vector machine, random forest, and feature selection to single out PD from healthy sample and were successful in achieving 99% accuracy with only ten dysphonia features[18].

In the year 2014 A. Sharma et al. gave pattern recognition, neural network, and support vector machine to diagnose the PD based on the dataset with biomedical voice signals of healthy and Parkinson sample with 85.294% accuracy[19]. With a prediction of achieving 100% accuracy a team of researchers presented SVM and Genetic Algorithm (GA) to classify the healthy and PD people based on the voice signals with 14 features like jitter, shimmer etc., and had achieved an accuracy of 94.50, 93.66 and 94.22 [20]. Bocklet et al. (2011) gave correlation and SVM based classification for diagnosing PD patients based on voice, articulations, and prosodic evaluation and achieved 90.5 % accuracy [21]. Data mining, neural network analysis, and regression analysis, and decision trees were compared by a team and presented an accuracy of 84.3%, 92.9%, 88.6% and 84.3% respectively [22]. Another machine learning algorithm PNN (probabilistic neural network) was used to classify the PD patients which included three PNN namely Monte Carlo Search (MCS), incremental search (IS), and hybrid search (HS) and gained accuracy between 79-81% [23]

In another study 2 type of ANN multilayer Perceptron (MLP), Adaptive Neuro-Fuzzy Classifier (ANFC), and Radial Basis Function RBF networks with Linguistic hedges to identify PD patients. Among these ANFC gave the best accuracy of 95.38% and classification performance of 94.72% [24] . Cho, C. at al. in 2009 presented an algorithm that combined principal component analysis (PCA) with linear discriminant analysis (LDA) to conduct gait analysis which will detect the gait pattern of PD by utilizing computer vision and their results depicted that LDC had a recognition rate of PD gait had 95.49% accuracy [25]. Recently in the year 2020 used Recursive feature elimination and feature importance methods wherein, artificial neural network, regression trees, and support vector machine were used to identify PD patient and 93.84% accuracy in voice feature was obtained. A study conducted in 2019 introduced VGFR Spectrogram Detector and Voice Impairment Classifier which are neural networks which would help the doctor to detect PD at early stage. CNN (Convolutional Neural Networks) and deep dense ANN were implemented to classify gait signals and voice recordings respectively. The classification accuracy with VGFR was 88.1% and with voice impairment was 89.15% [26].

In 2020 another study wherein, PD detection System were developed by using features like RMS, chrome STFT, spectral centroid, etc., herein, CNN, ANN, and Hidden Markov Model (HMM) were used to differ PD patients from healthy ones at early stages of disease with accuracies 93.5%, 96%, and 95.2% respectively [27]

CHAPTER 3

System Requirements:

3.1 Requirement

3.1.1 Language Used

The programming language used is **PYTHON**

3.1.2. Technical Requirements (Hardware)

As we will be working with Google Colab

The minimum system requirements are:

Memory: 4 GB
Free Storage: 2 GB
Screen Resolution: 1200 x 800
OS: Windows 7/8/8.1/10 (64-bit)

The recommended system requirements are

Memory: 8 GB RAM
Free Storage: 4 GB (SSD Recommended)
Screen Resolution: 1920 x 800
OS: Windows 10 (64-bit)
CPU: Intel Core i5-8400 3.0 GHz or better.

Requirement for doing Analysis. The Algorithm, Libraries used:

NumPy, Matplotlib, Seaborn.
Pandas, Scikit Learn, XGBoost.
Linear Regression, Logistic Regression, Decision Trees.
Support Vector Machine, Random Forest.

3.2 Deliverables of the Minor Project

The project will predict whether the person is or is not having PD. Using the given data set we will analyze data using Machine Learning Algorithms (Linear_Regression, Logistic_Regression, Decision_Trees, Support_Vector_Machine, Random_Forest, XGBoost, Adaboost.) Our goal is to attain a 100% accuracy Model. We will also provide a Confusion Matrix, Classification Report, F1 - Score, Accuracy, Precision, Recall. For a better conclusion of our models.

3. 3 Feasibility Study on Major Project:

This project evaluates the effectiveness of using controlled classification algorithms, such as Logistic Regression, Vector Support Machines, Decision Trees, Random Forest, XGBoost, Neural Network and AdaBoost to accurately identify people with the disease. Our 100% accuracy (in the database we used) provided by machine learning models exceeds the accuracy of clinical diagnostic tests for non-specialists (73.8%) and the accuracy among movement therapists (79.6% without follow-up, 83.9% after follow-up) with autopsy as a basic fact.

3.3.1. Requirements on Major Project

_Functional Requirements:

- Anaconda Distribution's (Jupiter Notebook) Or Google Colab
- Programming language – Python, Microsoft Windows 10, Machine Learning Libraries: NumPy, Matplotlib, Seaborn Pandas, Scikit Learn and XGBoost
- Machine Learning Algorithms Used: Linear Regression, Logistic Regression, Decision Trees, Support Vector Machine, Random Forest, XGBooster, Adaboost.
- Evaluation Methods and Metrics: Confusion Matrix, Classification, Accuracy, Precision, and Recall.

Non-Functional Requirements

- 2GB memory,
- Intel 8th generation higher CPU's
- Nvidia 1650 min GPU

3.4 Use Case Diagram of the Minor Project

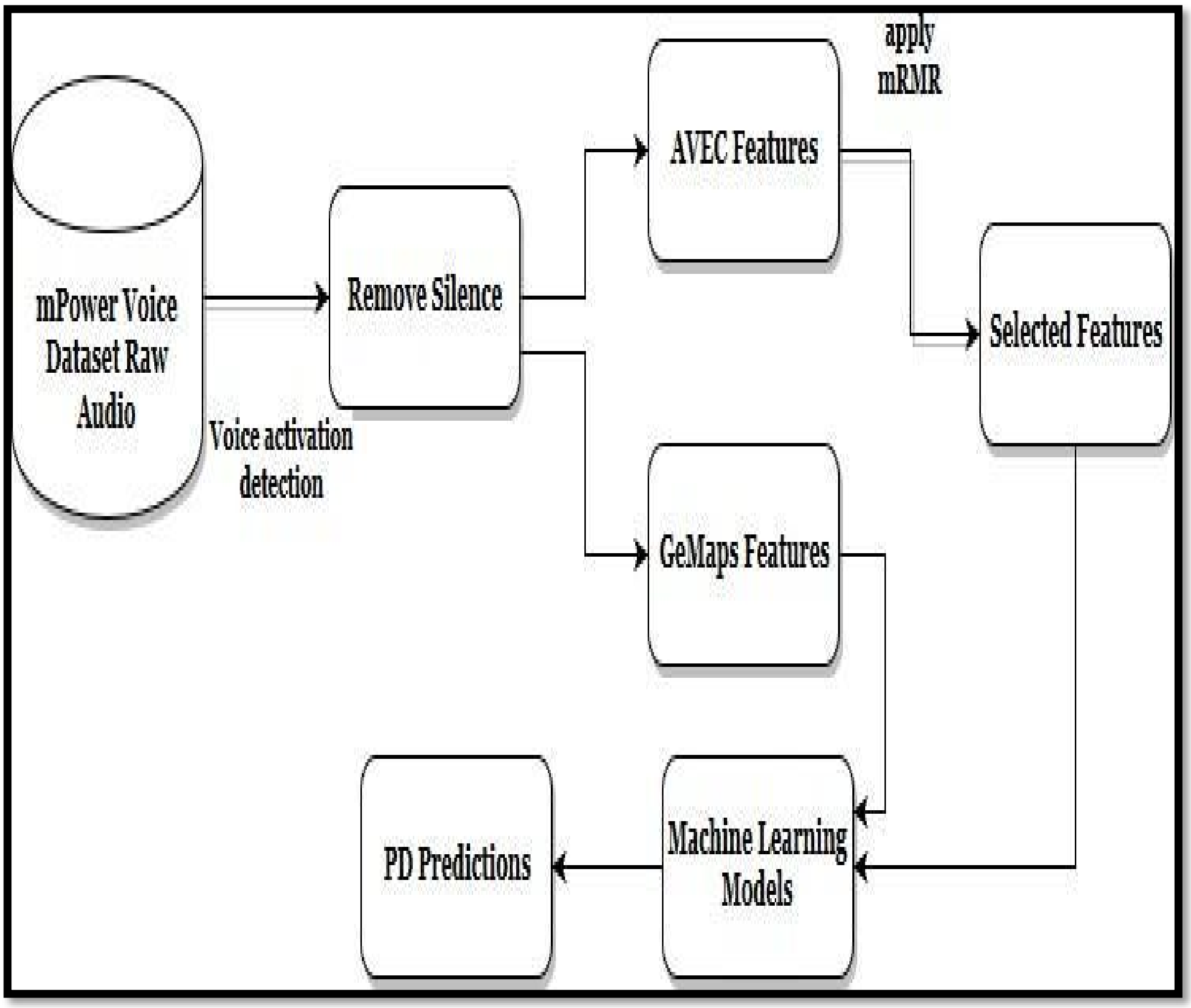


Figure 1 Use Case Diagram

3.5 DFD Diagram of the Major Project

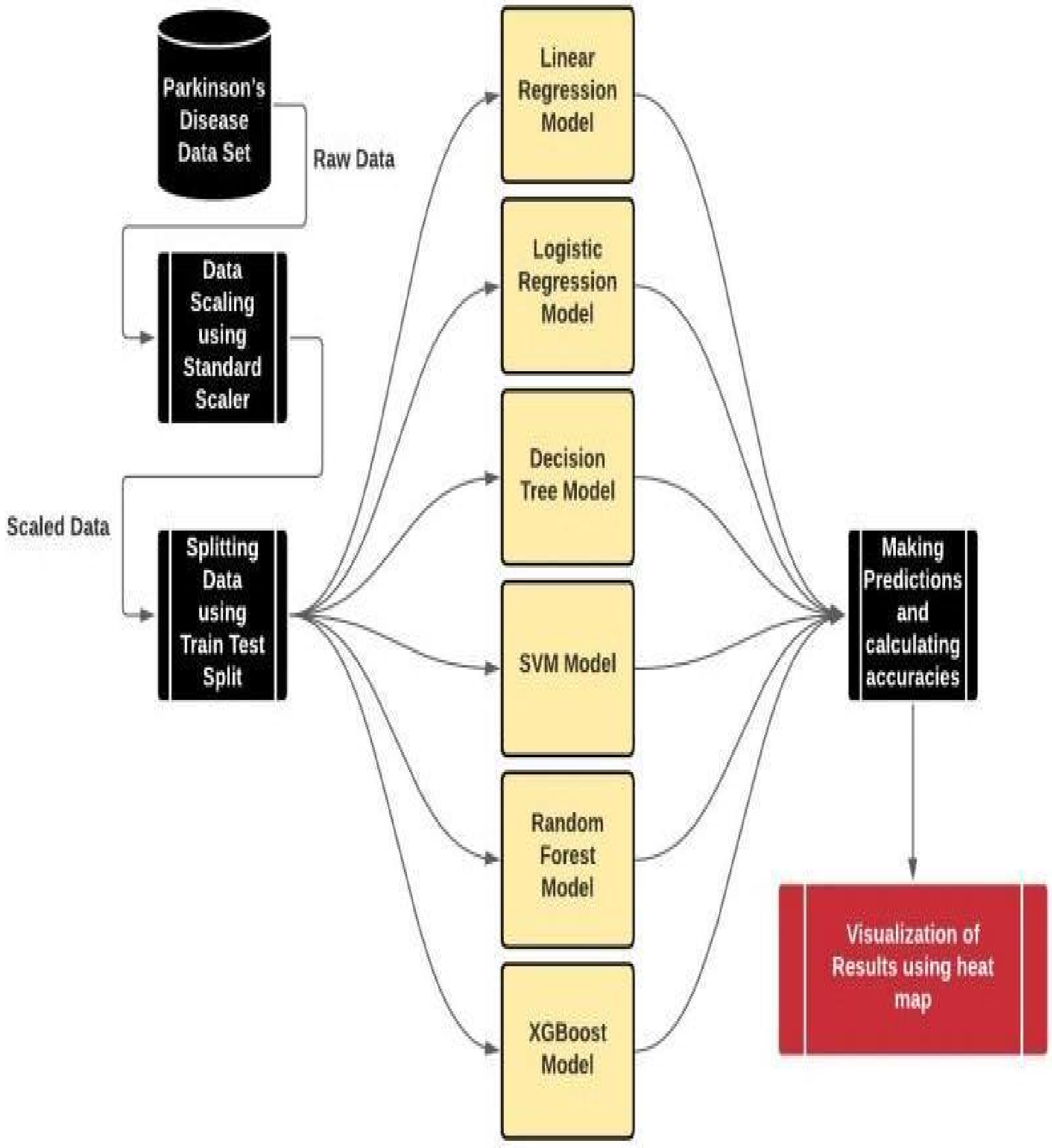

Parkinson's Disease Data Set
Raw Data
Data Scaling using Standard Scaler
Scaled Data
Splitting Data using Train Test Split
Linear Regression Model
Logistic Regression Model
Decision Tree Model
SVM Model
Random Forest Model
XGBoost Model
Making Predictions and calculating accuracies
Visualization of Results using heat map

Figure 2 DFD Diagram

3.6 Analysis/ Design/ Development/ Algorithm

The PD dataset contains data of 188 patients with 64 true negatives. The Data set contains information of both men and women. The task was to prepare or find a model which is giving higher accuracies and is better in prediction PD. For these Several supervised and unsupervised techniques were taken into consideration. The analysis was done in a Collab with python language used. The Projects aims to also compare results of these several algorithms. The dataset in general has voice attributes collected after passing through some algorithms which are collected by 188 men/women. After the dataset was selected the following components were used.

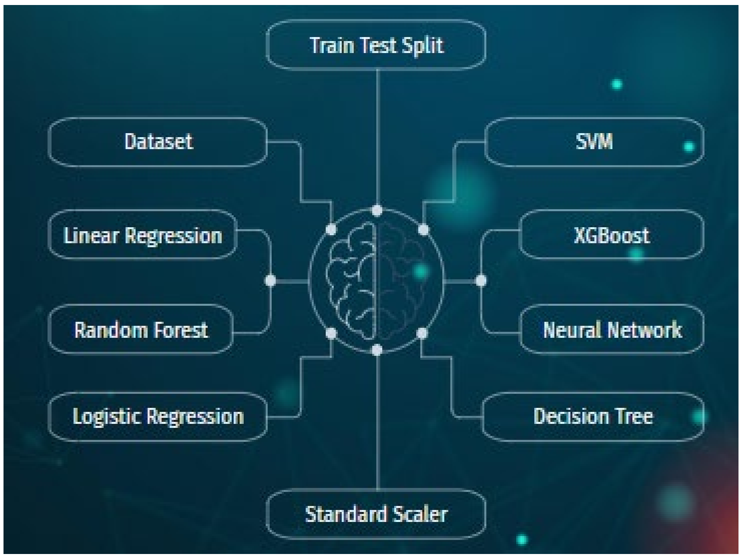


Figure 3 Roadmap Of Project

The roadmaps of the project are:

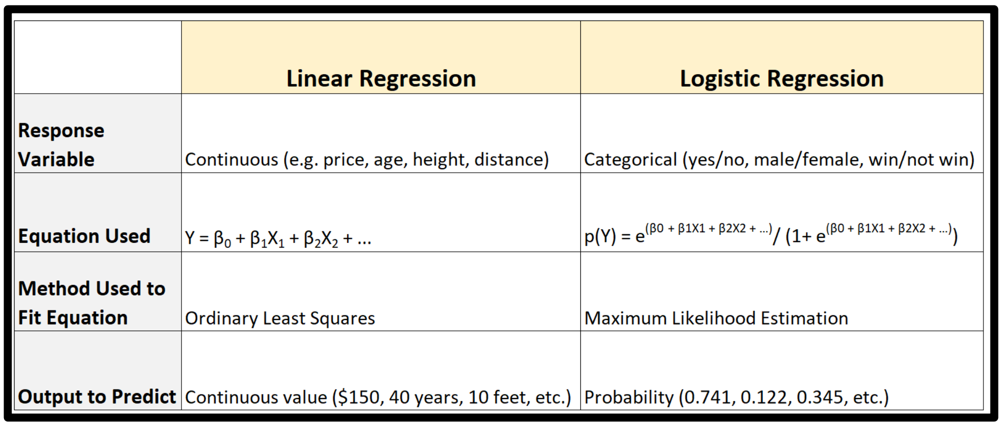
- Importing the required libraries.
- Exploratory data analysis on the dataset.
- Performing basic data cleaning.
- Using a standard scaler for data standardization.
- Splitting the dataset into training and testing samples.
- Implementing Linear Regression and Evaluating Linear Regression.
- Implementing Logistic Regression and Evaluating Logistic Regression.
- Implementation Decision Tree and evaluating Decision Tree.
- Implementing SVM and Evaluating SVM.
- Implementing XGB and Evaluating XGB.
- Implementing RT and Evaluating RT.
- Implementing NN and Evaluating NN.

- implementing ADAboost and evaluating Adaboost.

3.6 Computational/Experimental/Mathematical/ Statistical Approach.

<u>Algorithms of modeling techniques used:</u>

- Linear Regression: (Start with random weights, do the Hypothesis, compute cost/error function, minimize use gradient descent and update the weights.
- Logistic Regression:(Train and Test data, compute the regression coefficients of training data, use sigmoid function, find the relationship between the training data and the testing data, and output the object's position.
- Decision Tree: (import data, Doing EDA, Splitting Dataset, Create the DT Classifier, train model and predict data)
- SVM: (Setting Parameters, find initial value of C and E by cuckoo search, generate initial particles, evaluating the fitness of each particle, comparing the fitness value, and determining the local best and global best particle, updating the fitness values, and determining the local best particle, select best value of C and E for SVM.
- XGBoost: read in monitoring data, choose hyper parameters of XGBoost, train model, factor importance computation, lag process identification, evaluation of identification effects, validation, and model assessment.
- Neural Network: Input and target data, data normalization, selection of network structure, initialization of weights and biases, training, and testing, freeze the network, weight and biases, blind prediction.
- Adaboost: Input the data, fit it into the classifier, compute scores, perform hyper parameter tuning and compute and evaluate results.



	Linear Regression	Logistic Regression
Response Variable	Continuous (e.g. price, age, height, distance)	Categorical (yes/no, male/female, win/not win)
Equation Used	$Y = \beta_0 + \beta_1X_1 + \beta_2X_2 + \ldots$	$p(Y) = e^{(\beta0 + \beta1X1 + \beta2X2 + \ldots)} / (1 + e^{(\beta0 + \beta1X1 + \beta2X2 + \ldots)})$
Method Used to Fit Equation	Ordinary Least Squares	Maximum Likelihood Estimation
Output to Predict	Continuous value ($150, 40 years, 10 feet, etc.)	Probability (0.741, 0.122, 0.345, etc.)

Figure 4 LR VS LOR

<u>Computational analysis:</u>

Confusion matrix:

Visualizing the working of the algorithm in the field of machine learning is given by a confusion matrix. it is a nXn matrix which shows the following 4 important value:

- True Positive (TP): correct indication
- True Negative (TN): correct indication of absence of character.
- False Positive (FP): wrong prediction of present character.
- False Negative (FN): wrong prediction of absent character.
- TPR (True Positive Rate) = TP/TP+FN
- TNR (True Negative Rate) = TN/TN+FP
- PPV (Positive Predictive Value) =TP/TP+FP
- NPV (negative predictive value) = TN/TN+FN
- FPR (Fall-Out/ False Positivity rate) = FP/FP+TP
- precision= TP/PP
- recall= TP/AP
- fiscore=2 * (precision * recall) / (precision + recall)

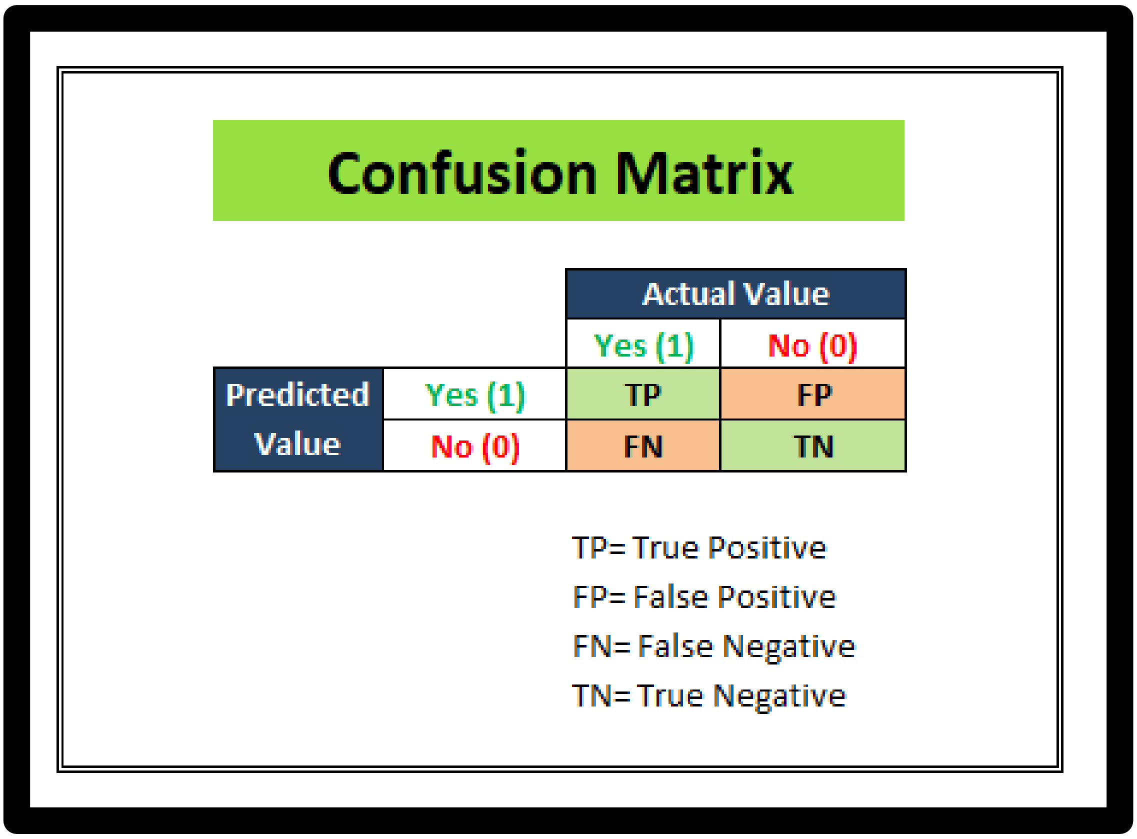
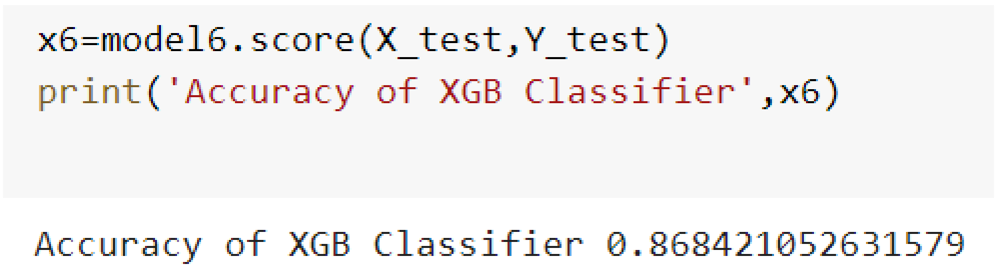


Figure 5 Confusion Matrix

Statistical:

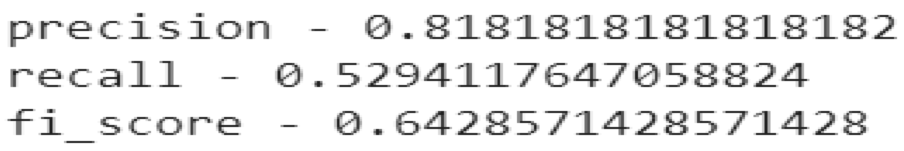
While using seven models the model provided higher accuracies and better prediction. XGBOOST gave 0.86842 or 86% of accuracy. giving 110 true positive, 18 true negatives, 16 false positive and 4 false negative. The model also gave Precision of 81%, recall of 52% and fi_score of 64%.



```
x6=model6.score(X_test,Y_test)
print('Accuracy of XGB Classifier',x6)
```

```
Accuracy of XGB Classifier 0.868421052631579
```

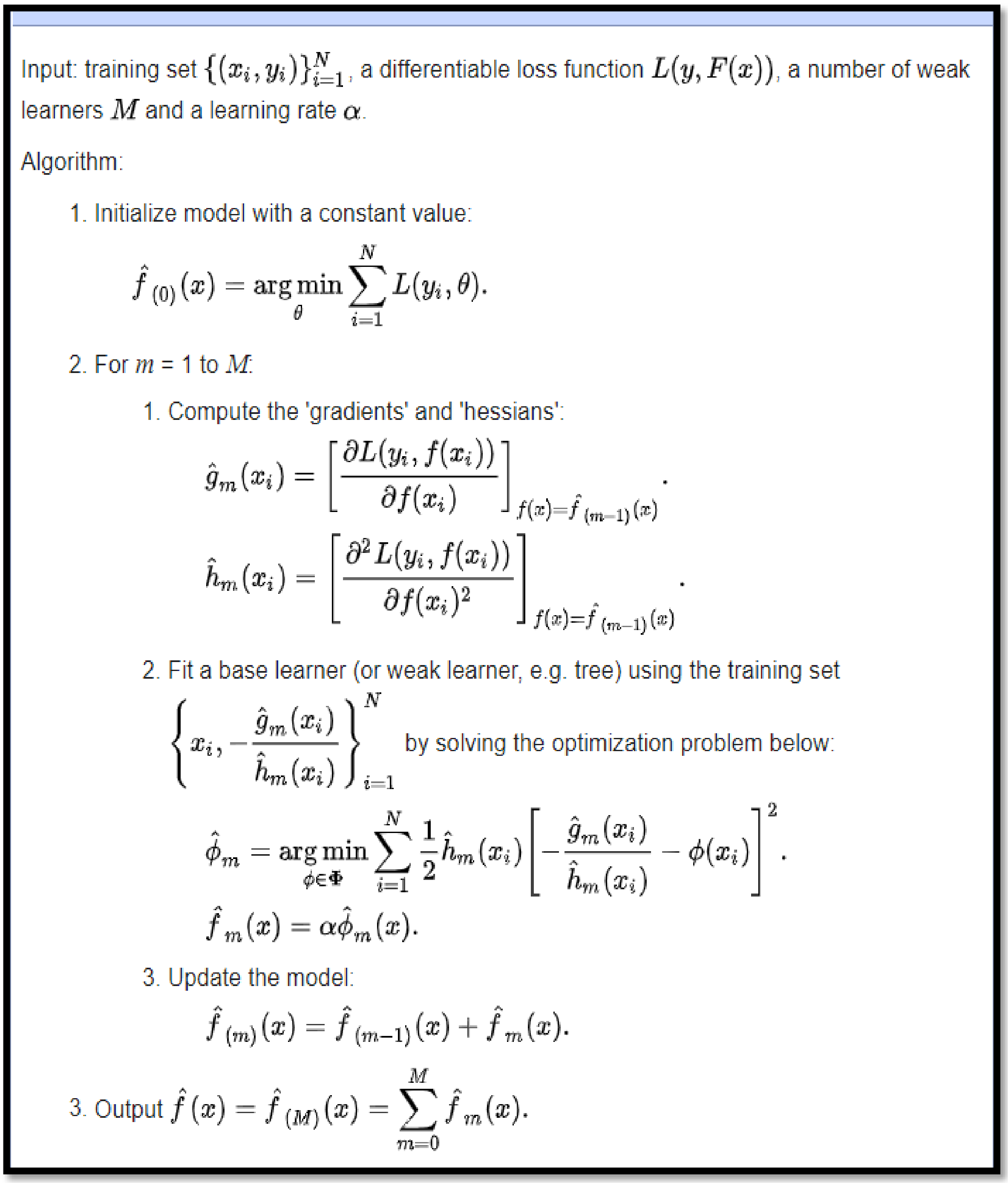
Figure 6 XGB Accuracy



```
precision - 0.8181818181818182
recall - 0.5294117647058824
fi_score - 0.6428571428571428
```

Figure 7 XGB Precision

XGBoost or extreme gradient boosting is an unsupervised classifier the functioning of the algorithm is mentioned below:



Input: training set $\{(x_i, y_i)\}_{i=1}^{N}$, a differentiable loss function $L(y, F(x))$, a number of weak learners M and a learning rate α.

Algorithm:

1. Initialize model with a constant value:

$$\hat{f}_{(0)}(x) = \arg\min_{\theta} \sum_{i=1}^{N} L(y_i, \theta).$$

2. For m = 1 to M:

    1. Compute the 'gradients' and 'hessians':

    $$\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x)=\hat{f}_{(m-1)}(x)}.$$

    $$\hat{h}_m(x_i) = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2} \right]_{f(x)=\hat{f}_{(m-1)}(x)}.$$

    2. Fit a base learner (or weak learner, e.g. tree) using the training set $\left\{ x_i, -\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)} \right\}_{i=1}^{N}$ by solving the optimization problem below:

    $$\hat{\phi}_m = \arg\min_{\phi \in \mathbf{\Phi}} \sum_{i=1}^{N} \frac{1}{2} \hat{h}_m(x_i) \left[-\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)} - \phi(x_i) \right]^2.$$

    $$\hat{f}_m(x) = \alpha \hat{\phi}_m(x).$$

    3. Update the model:

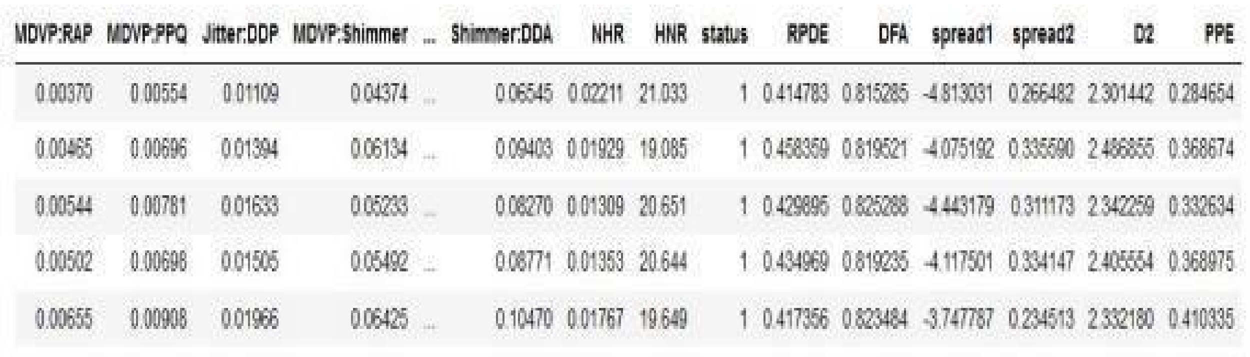
    $$\hat{f}_{(m)}(x) = \hat{f}_{(m-1)}(x) + \hat{f}_m(x).$$

3. Output $\hat{f}(x) = \hat{f}_{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x).$

Figure 8 XGB Algo.

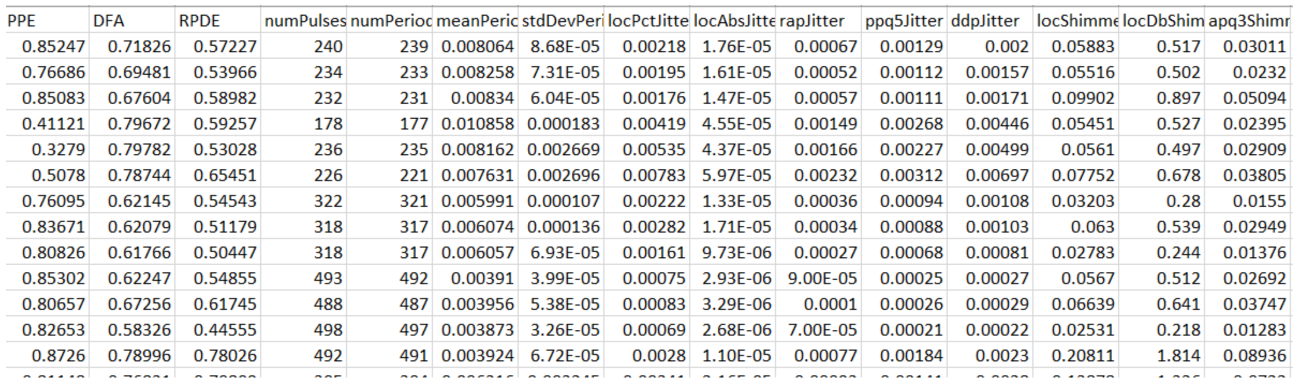
3.7 Dataset Used in the Major Project

This database contains many biomedical voice measurements (Acoustic Analysis of voice) from 31, 23 people with Parkinson's disorder (PD). Each column on the desk is a measure of a certain word, and each line corresponds exactly to one of the 195 words recorded for those people (column "calls"). An important mathematical goal is to differentiate healthy people from people with PD, according to the "status" column of almost zero for good and 1 for PD. Information is in ASCII CSV format. CSV document lines include an example corresponding to a single voice recording. There are about six recordings of the affected person, the affected person's call is found in the first column. The dataset is created by means of Max Little of the University of Oxford, in collaboration with the National Center for Voice and Speech, Denver, Colorado, who recorded the speech signals [19]. The unique have a look at published characteristic extraction techniques for trendy voice problems.



MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shimmer:DDA	NHR	HNR	status	RPDE	DFA	spread1	spread2	D2	PPE
0.00370	0.00554	0.01109	0.04374	...	0.06545	0.02211	21.033	1	0.414783	0.815285	-4.813031	0.266482	2.301442	0.284654
0.00465	0.00696	0.01394	0.06134	...	0.09403	0.01929	19.085	1	0.458359	0.819521	-4.075192	0.335590	2.486855	0.368674
0.00544	0.00781	0.01633	0.05233	...	0.08270	0.01309	20.651	1	0.429895	0.825288	-4.443179	0.311173	2.342259	0.332634
0.00502	0.00698	0.01505	0.05492	...	0.08771	0.01353	20.644	1	0.434969	0.819235	-4.117501	0.334147	2.405554	0.368975
0.00655	0.00908	0.01966	0.06425	...	0.10470	0.01767	19.649	1	0.417356	0.823484	-3.747787	0.234513	2.332180	0.410335

Figure 9 Data Attributes



PPE	DFA	RPDE	numPulses	numPerioc	meanPeric	stdDevPeri	locPctJitte	locAbsJitte	rapJitter	ppq5Jitter	ddpJitter	locShimme	locDbShim	apq3Shimr
0.85247	0.71826	0.57227	240	239	0.008064	8.68E-05	0.00218	1.76E-05	0.00067	0.00129	0.002	0.05883	0.517	0.03011
0.76686	0.69481	0.53966	234	233	0.008258	7.31E-05	0.00195	1.61E-05	0.00052	0.00112	0.00157	0.05516	0.502	0.0232
0.85083	0.67604	0.58982	232	231	0.00834	6.04E-05	0.00176	1.47E-05	0.00057	0.00111	0.00171	0.09902	0.897	0.05094
0.41121	0.79672	0.59257	178	177	0.010858	0.000183	0.00419	4.55E-05	0.00149	0.00268	0.00446	0.05451	0.527	0.02395
0.3279	0.79782	0.53028	236	235	0.008162	0.002669	0.00535	4.37E-05	0.00166	0.00227	0.00499	0.0561	0.497	0.02909
0.5078	0.78744	0.65451	226	221	0.007631	0.002696	0.00783	5.97E-05	0.00232	0.00312	0.00697	0.07752	0.678	0.03805
0.76095	0.62145	0.54543	322	321	0.005991	0.000107	0.00222	1.33E-05	0.00036	0.00094	0.00108	0.03203	0.28	0.0155
0.83671	0.62079	0.51179	318	317	0.006074	0.000136	0.00282	1.71E-05	0.00034	0.00088	0.00103	0.063	0.539	0.02949
0.80826	0.61766	0.50447	318	317	0.006057	6.93E-05	0.00161	9.73E-06	0.00027	0.00068	0.00081	0.02783	0.244	0.01376
0.85302	0.62247	0.54855	493	492	0.00391	3.99E-05	0.00075	2.93E-06	9.00E-05	0.00025	0.00027	0.0567	0.512	0.02692
0.80657	0.67256	0.61745	488	487	0.003956	5.38E-05	0.00083	3.29E-06	0.0001	0.00026	0.00029	0.06639	0.641	0.03747
0.82653	0.58326	0.44555	498	497	0.003873	3.26E-05	0.00069	2.68E-06	7.00E-05	0.00021	0.00022	0.02531	0.218	0.01283
0.8726	0.78996	0.78026	492	491	0.003924	6.72E-05	0.0028	1.10E-05	0.00077	0.00184	0.0023	0.20811	1.814	0.08936

Figure 10 Data Cleaning

3.8 Date Set Features

3.4.1. Types of Data Set

The data used in this study were gathered from 188 patients with PD (107 men and 81 women) with ages ranging from 33 to 87 (65.1Â±10.9) at the Department of Neurology in CerrahpaÅŸa Faculty of Medicine, Istanbul University. The control group consists of 64 healthy individuals (23 men and 41 women) with ages varying between 41 and 82 (61.1Â±8.9). During the data collection process, the microphone is set to 44.1 KHz and following the physicianâ€™s examination, the sustained phonation of the vowel /a/ was collected from each subject with three repetitions.

3.4.2 Number of Attributes, fields, description of the data set

The Parkinson's disease database will help us to figure out whether the respective target is or is not having the disease; it's a multivariate data set. The database is the voice samples which have been accumulated from 31 people out of which 23 are having the disease. This data set is composed of the range of biomedical voice measurements and each column describes a particular voice measure and each row corresponds to one of the 195 voice recordings from these individuals. With the help of machine learning techniques, Machine Learning Techniques Will be creating a model which can be 100% accurate in the description of the patient. The model will analyze the data in the given data set and will protect whether the patient is or is not having Parkinson disease.

Qualification Information:

Matrix column entries (attributes):

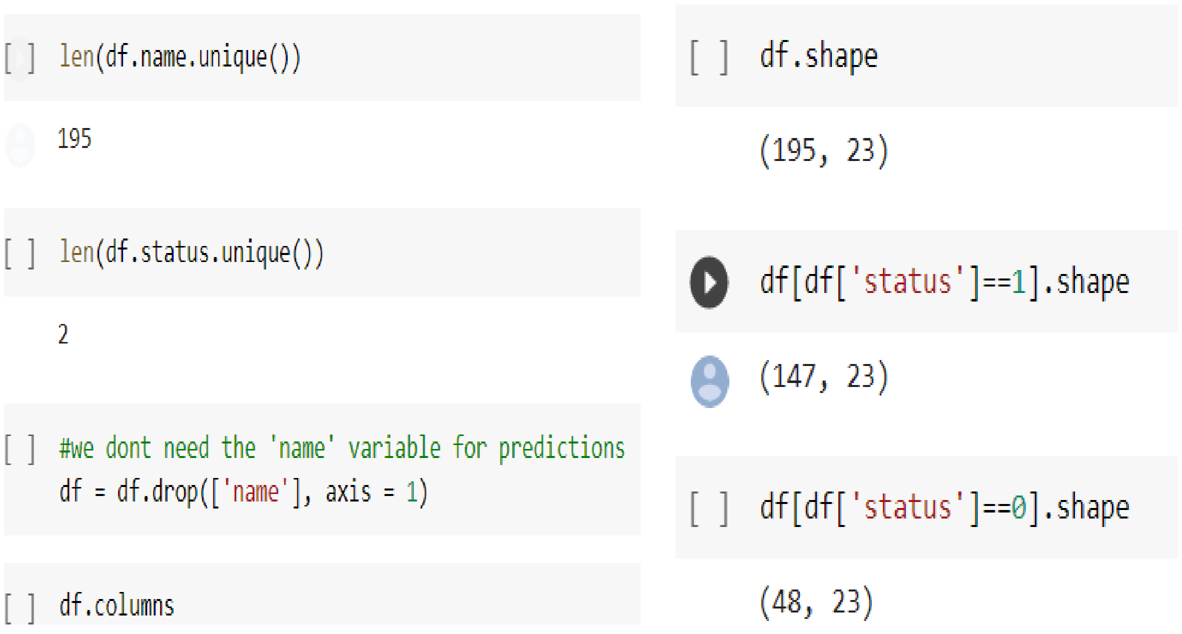
- Name - ASCII title name and recording number
- MDVP: Fo (Hz) - Basic voice frequency
- MDVP: Fhi (Hz) - The frequency of the basic voice
- MDVP: Flo (Hz) - Basic voice frequency
- MDVP: Jitter (%), MDVP: Jitter (Abs), MDVP: RAP, MDVP: PPQ, Jitter: DDP Several estimates of basic frequency variability
- MDVP: Shimmer, MDVP: Shimmer (dB), Shimmer: APQ3, Shimmer: APQ5, MDVP: APQ,
- Shimmer: DDA - A few steps for size variation
- NHR, HNR - Two levels of sound measurement and tone components invoice
- Condition - The health condition of the subject (one) - Parkinson's (zero)-is healthy

3.9 MODEL DEVELOPMENT

1. DATA CLEANING:

Before starting with the analysis part, we will modify our data for proper and correct analysis so that we will check the unique values in the name and status column. After that we will remove the name variable from our data set because the name variable is not important for making predictions. After that, we see the shape of our data set when the value of status is one and 0. After seeing the shape we will see that whether our data is having missing values or not, if the data is having missing values, we will have two do appropriate calculations to remove it but, in our case, data didn't have any missing values therefore it was ready for further analysis.

Some screenshots of the process have been attached below:



```
[ ] len(df.name.unique())

    195

[ ] len(df.status.unique())

    2

[ ] #we dont need the 'name' variable for predictions
    df = df.drop(['name'], axis = 1)

[ ] df.columns
```

```
[ ] df.shape

    (195, 23)

    df[df['status']==1].shape

    (147, 23)

[ ] df[df['status']==0].shape

    (48, 23)
```

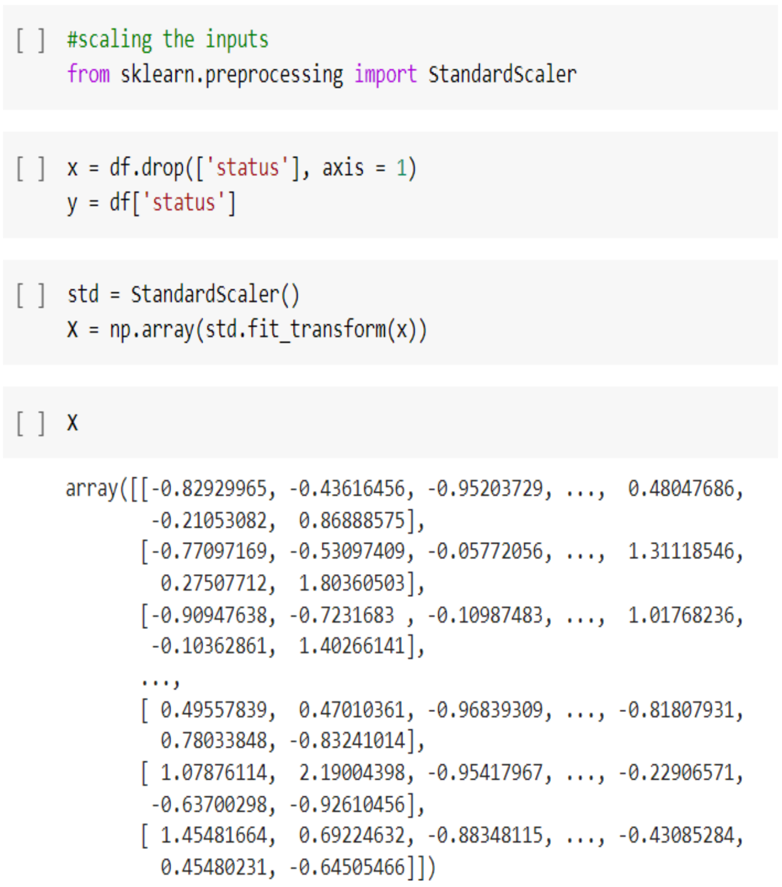
Figure 11 Standard Scaling

Standard Scaling:
After the data cleaning process is completed, our next step is 2 standard scale our data. For making a model we will have to do a train test split and for that, it is important that data has been successfully transformed into X and Y arrays. We will drop the status column from the

X array and transform X using a standard scaler meanwhile we will create another array that will only contain the value of status. Now the data is completely ready for train test split and further analysis. It measures features by subtracting meaning and measuring unit variations. A typical sample school x is calculated as:

z = (x - u) / s

where u is the method of training samples, and s the standard deviation of training sample Two of the most popular ways to measure numerical data before modeling are customization and configuration. The practice of measuring different inputs separately at a range of 0-1, which is a floating-point value range when we have high accuracy. The measurement is different for each input by subtracting the meaning (called centering) and dividing it by standard deviation to change the distribution so that it has zero meaning and standard deviation for each. MinMaxScaler (feature_range = (0, 1)) will convert each value into a column equally within the range [0-1]. Use this as the first scale option to change the feature, as it will maintain the database status (no distortion).



```
[ ] #scaling the inputs
    from sklearn.preprocessing import StandardScaler
```

```
[ ] x = df.drop(['status'], axis = 1)
    y = df['status']
```

```
[ ] std = StandardScaler()
    X = np.array(std.fit_transform(x))
```

```
[ ] X
```

```
array([[-0.82929965, -0.43616456, -0.95203729, ...,  0.48047686,
        -0.21053082,  0.86888575],
       [-0.77097169, -0.53097409, -0.05772056, ...,  1.31118546,
         0.27507712,  1.80360503],
       [-0.90947638, -0.7231683 , -0.10987483, ...,  1.01768236,
        -0.10362861,  1.40266141],
       ...,
       [ 0.49557839,  0.47010361, -0.96839309, ..., -0.81807931,
         0.78033848, -0.83241014],
       [ 1.07876114,  2.19004398, -0.95417967, ..., -0.22906571,
        -0.63700298, -0.92610456],
       [ 1.45481664,  0.69224632, -0.88348115, ..., -0.43085284,
         0.45480231, -0.64505466]])
```

Figure 12 Standard Scaling

StandardScaler () will convert each value in a column to a range of 0 with a standard deviation of 1, that is, each value will be made normal by subtracting and dividing by standard deviation. Use StandardScaler if you know that data distribution is common.

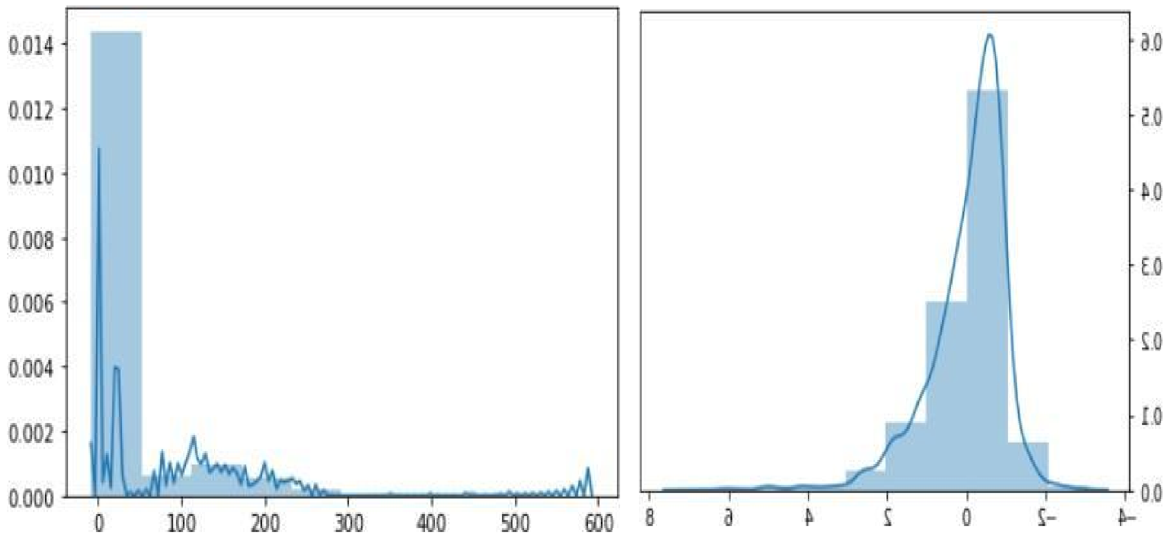


Figure 13 Standard Scaling 3

3.10 Train test split:

In order to obtain effective model calculation in machine learning, it is important to train and build an algorithm that can assist in data prediction. The data provided is usually categorized into data sets and reused for training and testing purposes which are usually training, validation, and test sets. The method is used to measure the overall performance of ML algorithms while it may be used to speculate on unspecified facts to teach the model. It is a fast and easy way to do it, the results of which will allow you to test the performance of ML algorithms to your predictive modeling complexity. Although easy to use and translate, there are instances when the process should not be used now, including when you have a small database and situations where additional configuration is required, including when used in class and the database is uneven. The model was first included in the training data after the model was trained using a supervised learning method. The current model or model we are developing is used with a set of training data and will produce a result based on the result we can predict whether the model successfully predicts prices or not.

The embedded model is useful for predicting a confirmation data set that provides an unbiased evaluation of the model at the end of the data set and provides an unbiased evaluation of the final model of the training data set. The separation of the train test will result in two trained databases and the test train data will be used to match the machine learning model and the test data set will be used for testing purposes. The average train ride to the test is 80% train and 30% inspection.

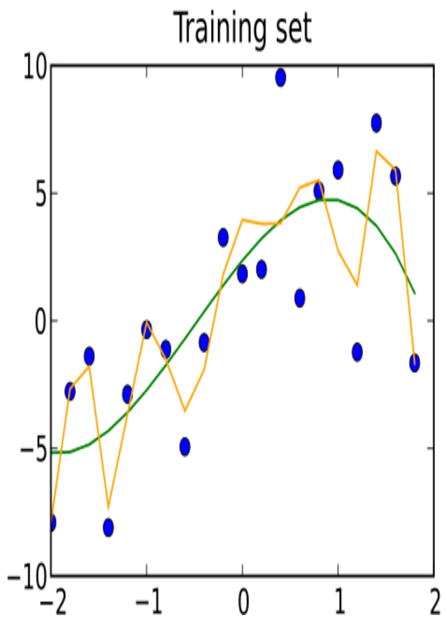
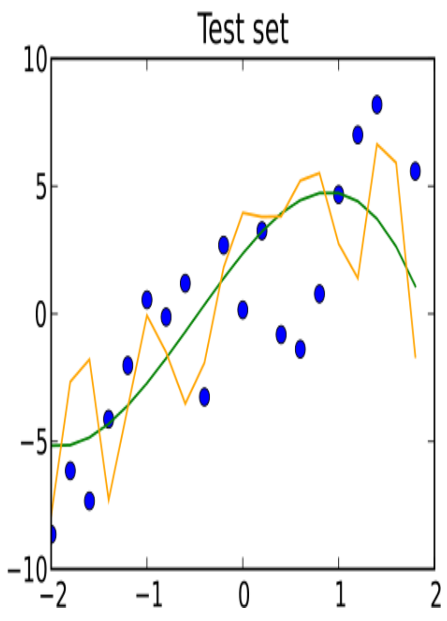


Figure 14 Train test split 1

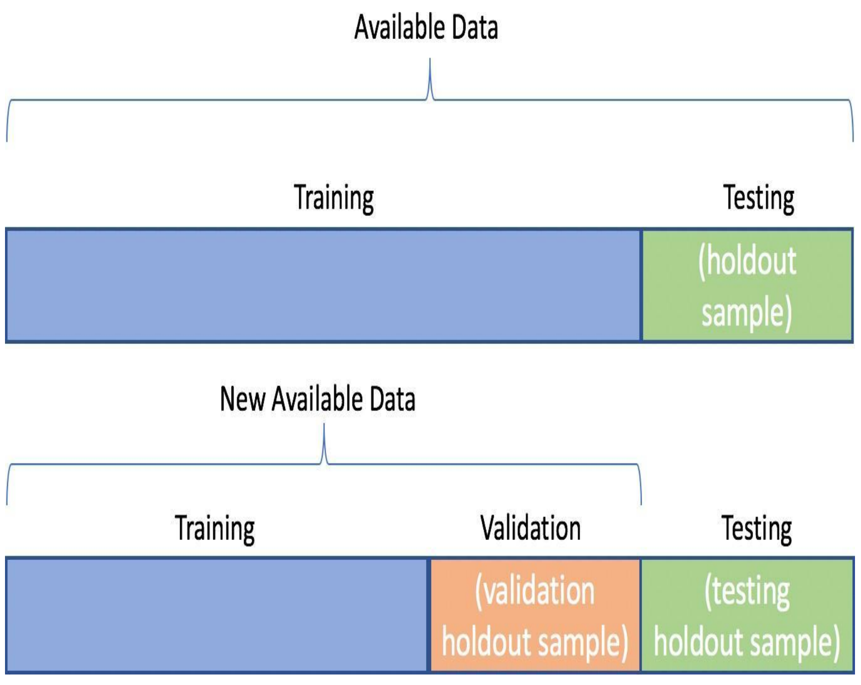
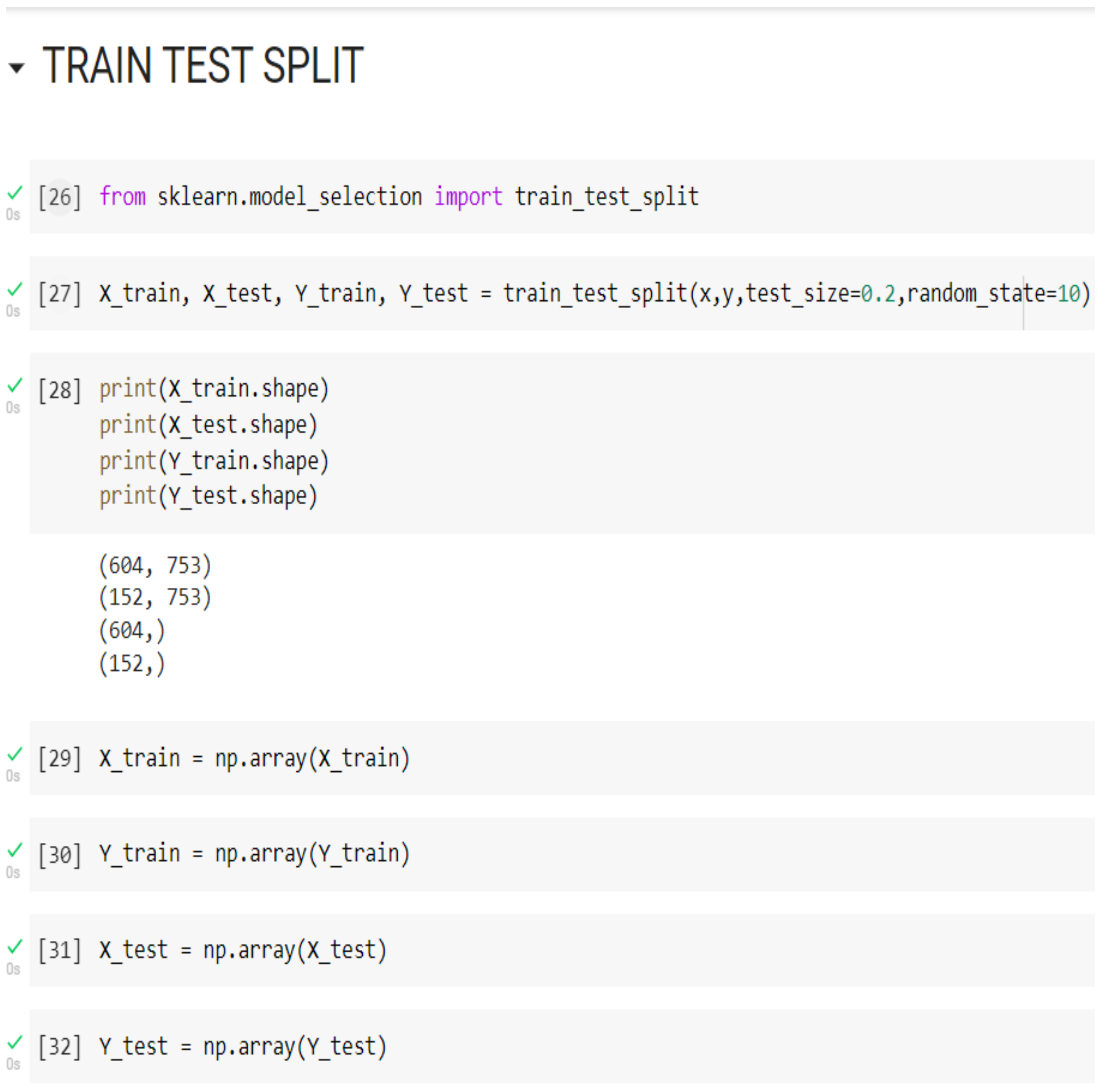


Figure 15 Train test Split

In our model, we will use a trains test split with a test size of 0.2 the method will result in two X_train, X_test, Y_train, and Y_test these values would be then stored in an array the value stored now would be used for further analysis.



TRAIN TEST SPLIT

```
[26] from sklearn.model_selection import train_test_split
```

```
[27] X_train, X_test, Y_train, Y_test = train_test_split(x,y,test_size=0.2,random_state=10)
```

```
[28] print(X_train.shape)
     print(X_test.shape)
     print(Y_train.shape)
     print(Y_test.shape)

     (604, 753)
     (152, 753)
     (604,)
     (152,)
```

```
[29] X_train = np.array(X_train)
```

```
[30] Y_train = np.array(Y_train)
```

```
[31] X_test = np.array(X_test)
```

```
[32] Y_test = np.array(Y_test)
```

Figure 16 Train test code

CHAPTER 4

4.1 Detail Implementation and computational analysis.

4.1.1. Linear Regression Model:

The linear regression is helping to model the relationship between one or more than one variable B1 is an independent variable while the other is a dependent variable. This model is used to find relationships between two or more continuous variables.

The General equation is y = mx + c where x = independent variable y = dependent variable and m = slope between them and c = constant. The model will fit a straight line that minimizes the difference between predicted and original output. The implementation of linear regression is much easier than any other model and generally provides good accuracy scores. The most common and the best linear regression method is the least square method. The prediction made by the linear model can be easily fitted into the best model using gradient descent algo. Gradient descent algo is an optimization algorithm which after changing the value of coefficients provides the optimal solution. Linear Regression is easy to use and not so difficult to translate the output coefficients. If the connection between independent and dependent variants has a dating line, this algorithm is ideal for use due to its very small complexity compared to different algorithms. If the connection between the independent and dependent variables has a linear relationship, this set of rules is special to apply due to its very small complexity compared to other algorithms. On the other hand, within the line regression paths exits can have significant effects on retrospect, and the limits are straightforward in this process. In contrast, linear regression assumes a linear relationship between dependent variations and bias. Thus, it is thought that there is an immediate courtship between them. It assumes independence between attributes, but the regression of the line is also reflected in the relationship between the definition of dependent variation and independent variation. Just as the definition is not always the complete definition of a single variance, the linear regression is not always the full definition of the relationship between variables.

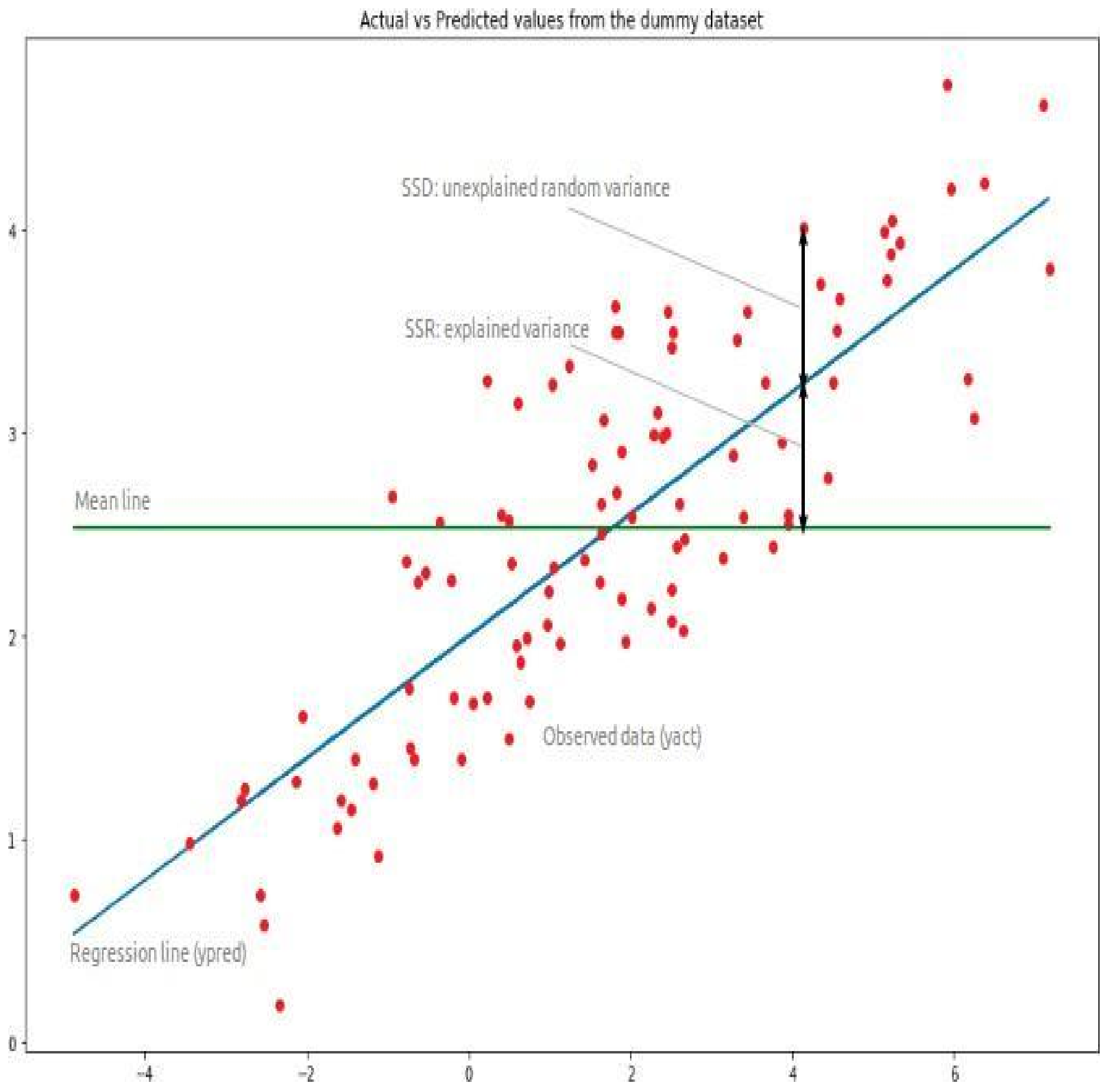


Figure 17 Linear Regression

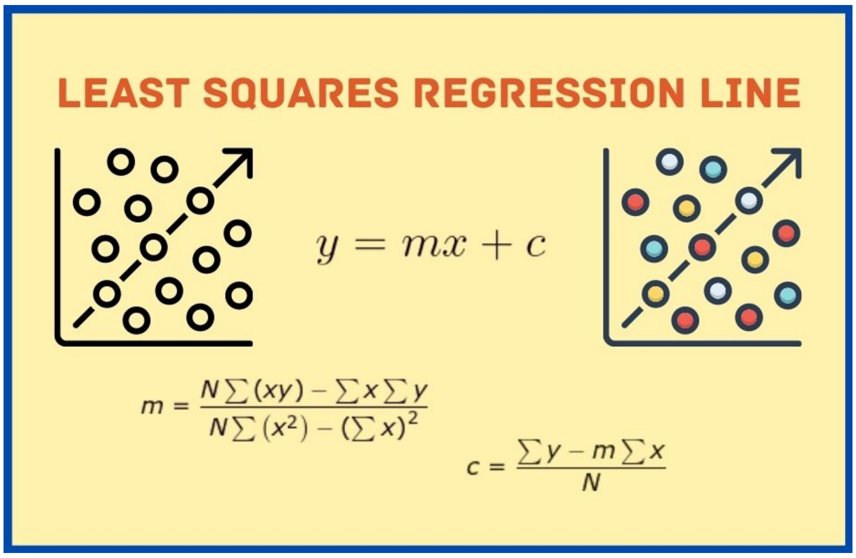

LEAST SQUARES REGRESSION LINE
$y = mx + c$
$m = \frac{N\sum(xy) - \sum x \sum y}{N\sum(x^2) - (\sum x)^2}$
$c = \frac{\sum y - m\sum x}{N}$

Figure 18 Least Square

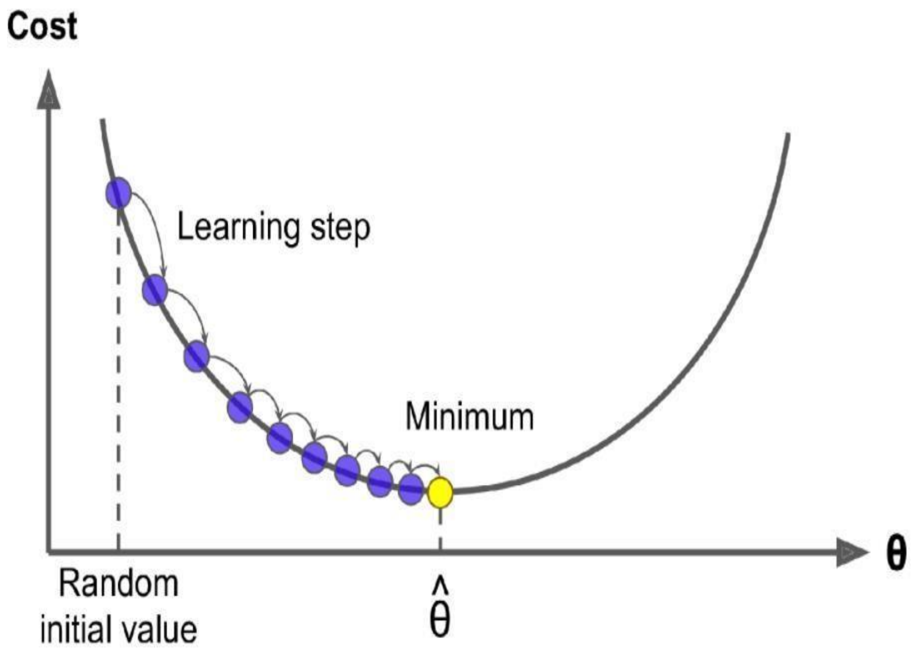
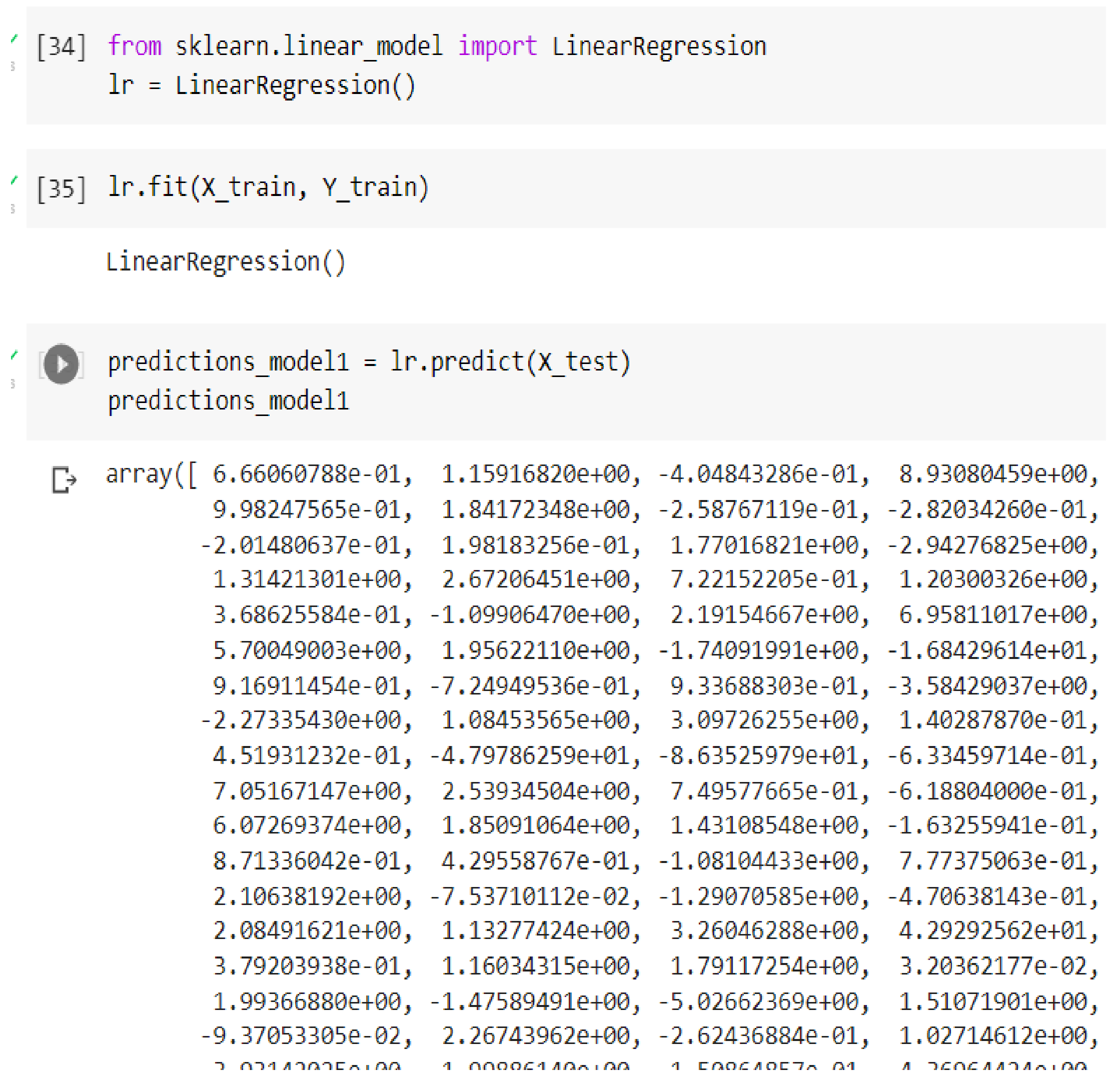


Figure 19 Linear Regression 3 Gradient

In our model we have fitted X_train and Y_train in a linear regression model. After that we applied gradient descent in order to find the best fit. followed, we have predicted the accuracy, and the accuracy we caught was 66%.

The implementation figures with the confusion matrix and accuracy score are been mentioned below:



```
[34] from sklearn.linear_model import LinearRegression
     lr = LinearRegression()
```

```
[35] lr.fit(X_train, Y_train)
```

```
LinearRegression()
```

```
predictions_model1 = lr.predict(X_test)
predictions_model1
```

```
array([ 6.66060788e-01,  1.15916820e+00, -4.04843286e-01,  8.93080459e+00,
        9.98247565e-01,  1.84172348e+00, -2.58767119e-01, -2.82034260e-01,
       -2.01480637e-01,  1.98183256e-01,  1.77016821e+00, -2.94276825e+00,
        1.31421301e+00,  2.67206451e+00,  7.22152205e-01,  1.20300326e+00,
        3.68625584e-01, -1.09906470e+00,  2.19154667e+00,  6.95811017e+00,
        5.70049003e+00,  1.95622110e+00, -1.74091991e+00, -1.68429614e+01,
        9.16911454e-01, -7.24949536e-01,  9.33688303e-01, -3.58429037e+00,
       -2.27335430e+00,  1.08453565e+00,  3.09726255e+00,  1.40287870e-01,
        4.51931232e-01, -4.79786259e+01, -8.63525979e+01, -6.33459714e-01,
        7.05167147e+00,  2.53934504e+00,  7.49577665e-01, -6.18804000e-01,
        6.07269374e+00,  1.85091064e+00,  1.43108548e+00, -1.63255941e-01,
        8.71336042e-01,  4.29558767e-01, -1.08104433e+00,  7.77375063e-01,
        2.10638192e+00, -7.53710112e-02, -1.29070585e+00, -4.70638143e-01,
        2.08491621e+00,  1.13277424e+00,  3.26046288e+00,  4.29292562e+01,
        3.79203938e-01,  1.16034315e+00,  1.79117254e+00,  3.20362177e-02,
        1.99366880e+00, -1.47589491e+00, -5.02662369e+00,  1.51071901e+00,
       -9.37053305e-02,  2.26743962e+00, -2.62436884e-01,  1.02714612e+00,
```

Figure 20 Linear Regression Code

4.1.2 Logistic Regression Model:

The next model that we're going to use is the logistic regression model which is like linear regression model is another supervised learning technique. The logistic regression model generally gives the solution in a binary yes or no or true or false situation. The main difference between both models is that linear regression provides a continuous prediction whereas logistic regression provides predictions in categories. Logistic regression is simpler to put into effect, interpret, and very efficient to teach, It makes no assumptions approximately distributions of classes in characteristic space, It can without problems increase to multiple training(multinomial regression) and a herbal probabilistic view of sophistication predictions, It, not handiest presents a measure of the way appropriate a predictor(coefficient size)is, however additionally its path of association (positive or negative), The good accuracy of many mathematical sets is simple and playable when the database is subdivided along the line and the Logistic decline is slightly inclined to over-equilibrium but may extend to large-scale datasets.. One can also not forget Regularization (L1 and L2) strategies to avoid over-fitting in those scenarios. The logistic regression ML model is used for prediction of various diseases including heart disease, neurological disorders prediction of GDP and economy.

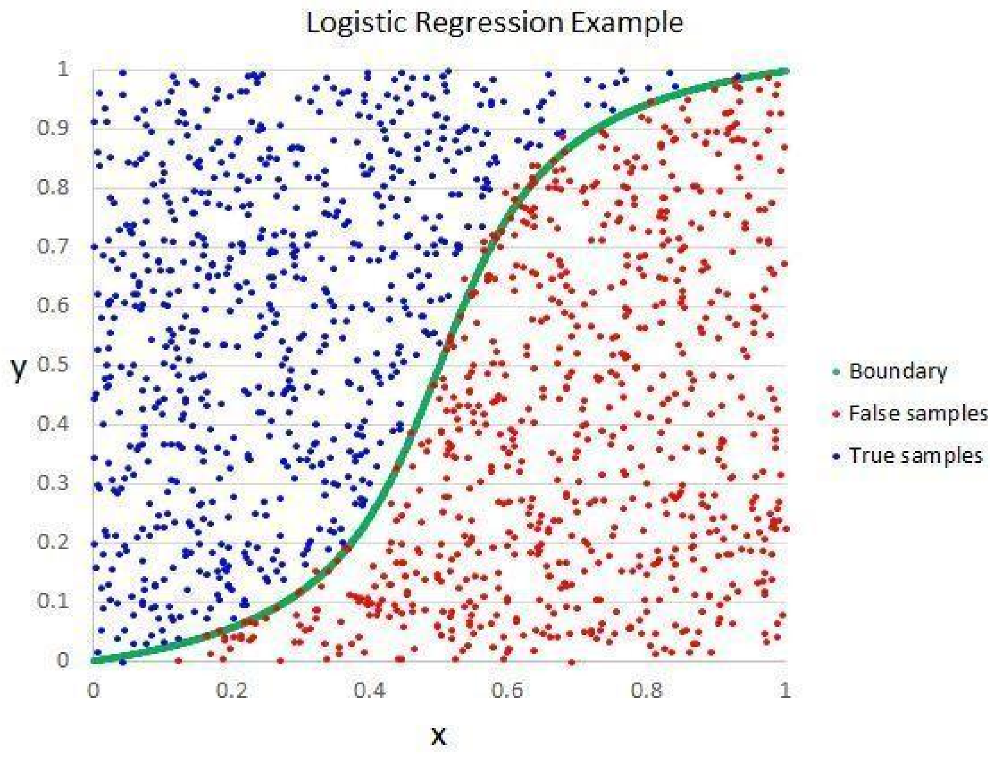


Figure 21 LoR Sigmiod Fuction

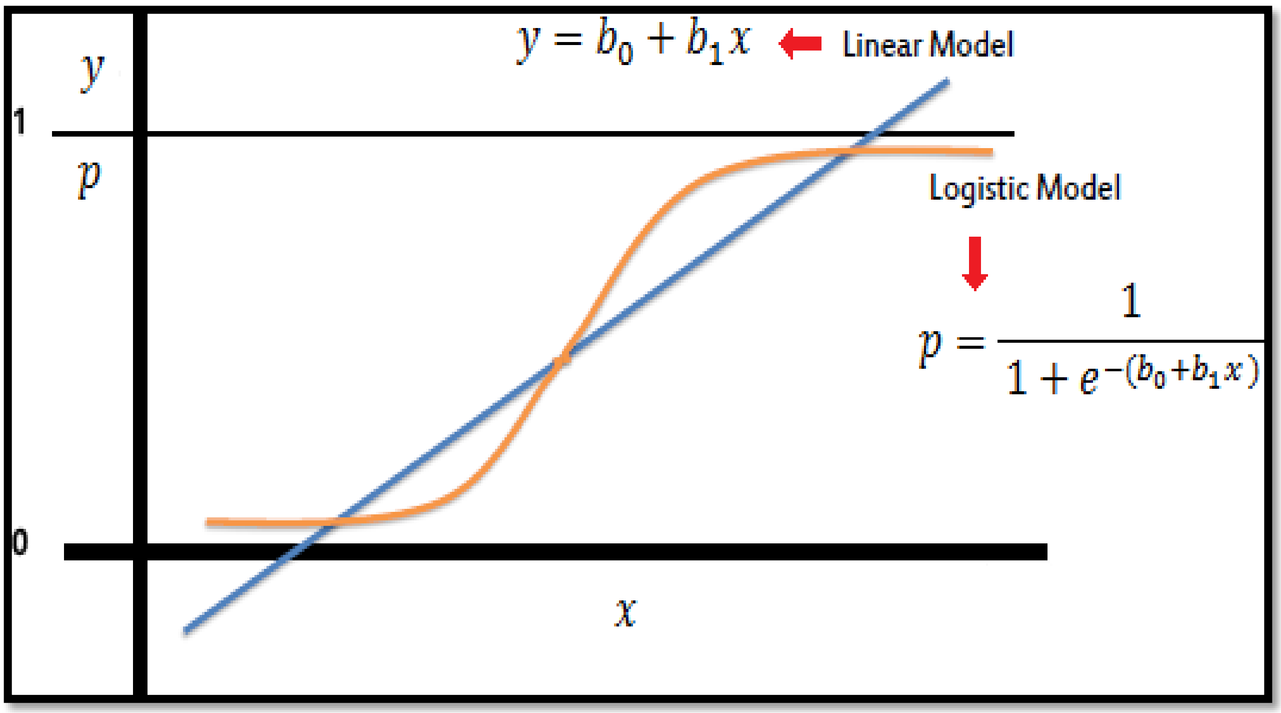
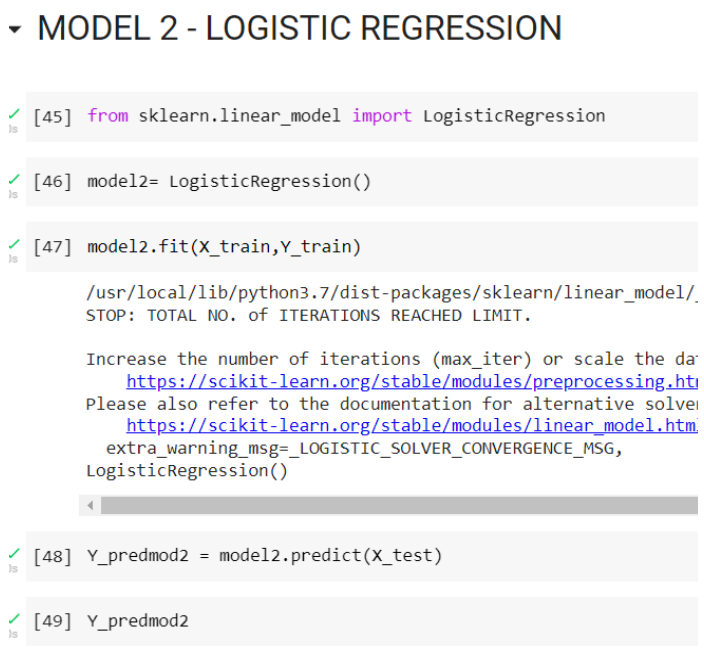


Figure 22 LOR Mathematical approach

Like a linear model that uses gradient descent for the best fit, logistic regression uses a maximum likelihood estimation method for best fit.

Our second model in this project is being made with the help of logistic regression we have fitted the values of X train and Y train in logistic regression model after death we have predicted the result and calculated the accuracy in this case, we got a 97% accuracy a confusion matrix based on the same model is also constructed.



MODEL 2 - LOGISTIC REGRESSION

```
[45] from sklearn.linear_model import LogisticRegression
```

```
[46] model2= LogisticRegression()
```

```
[47] model2.fit(X_train,Y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the da
    https://scikit-learn.org/stable/modules/preprocessing.ht
Please also refer to the documentation for alternative solve
    https://scikit-learn.org/stable/modules/linear_model.htm
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
LogisticRegression()
```

```
[48] Y_predmod2 = model2.predict(X_test)
```

```
[49] Y_predmod2
```

Figure 23 LOR Code

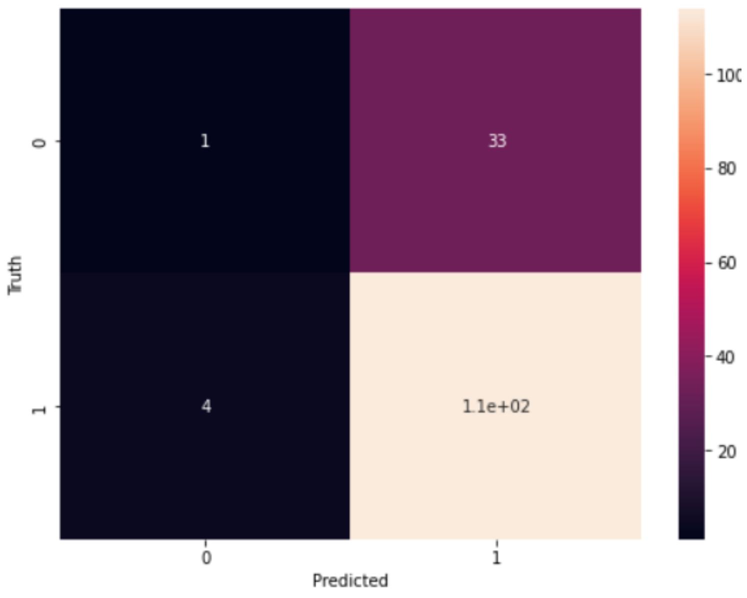


Figure 24 LOR Confusion Matrix

4.1.3 Decision Tree:

A decision tree organizes results in a flowchart manner with the help of conditional statements its classifieds data and then branches the data into two or more directions. Branches direct us to two different possible outcomes: the decision tree provides an optimal visual solution to the user and is extremely useful in data analysis. It is responsible for giving us solutions. Hey now in a loose fashion it breaks complex data and divides it into different parts. A decision tree is used for various predictions including disease prediction, vaccination predictions, economical predictions, electronics, and communication predictions with computer science predictions. A decision tree comprises three parts. Decision notes, chance notes, and end nodes are depicted in squares, circles, and triangles respectively and all these nodes relate to branches.

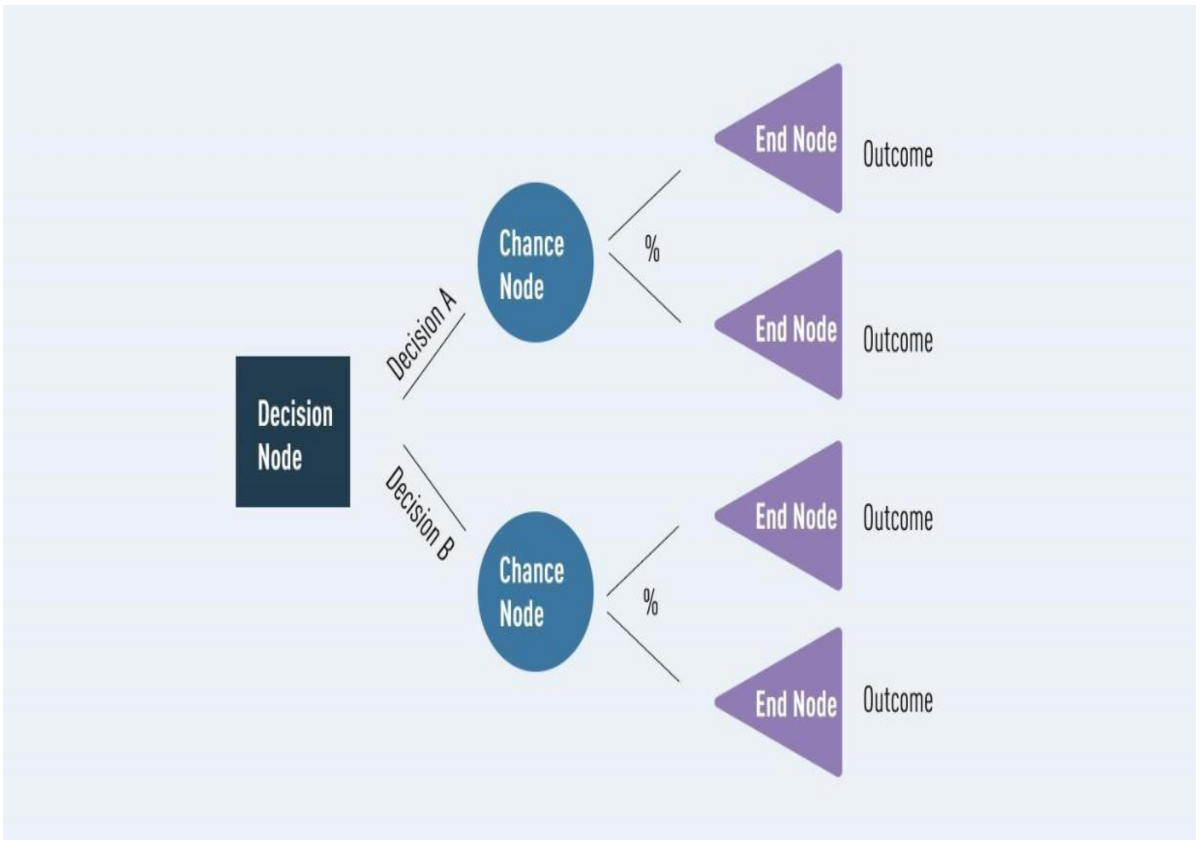


Figure 25 DT 1

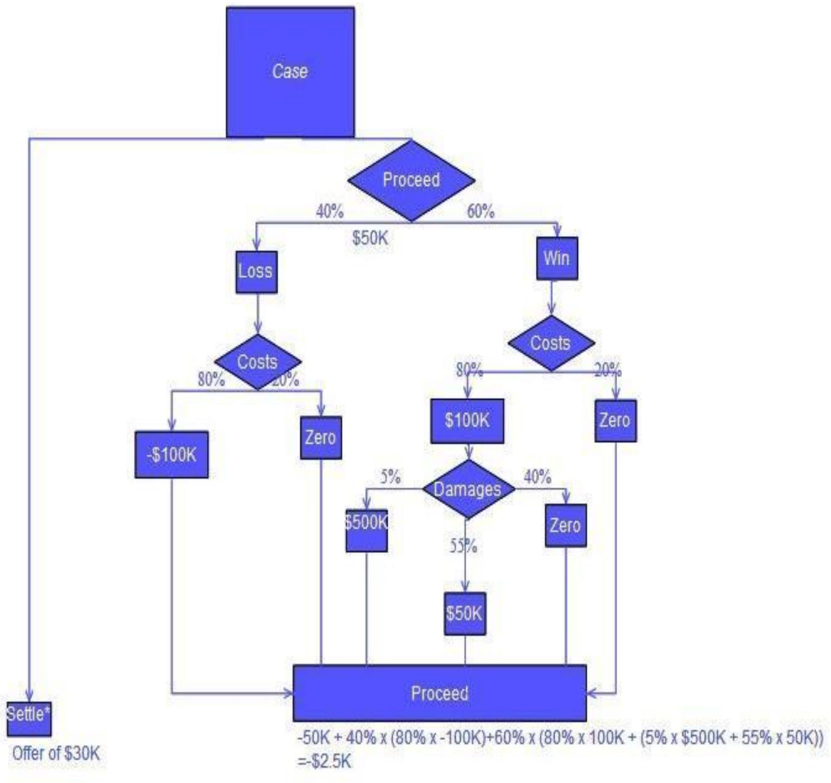
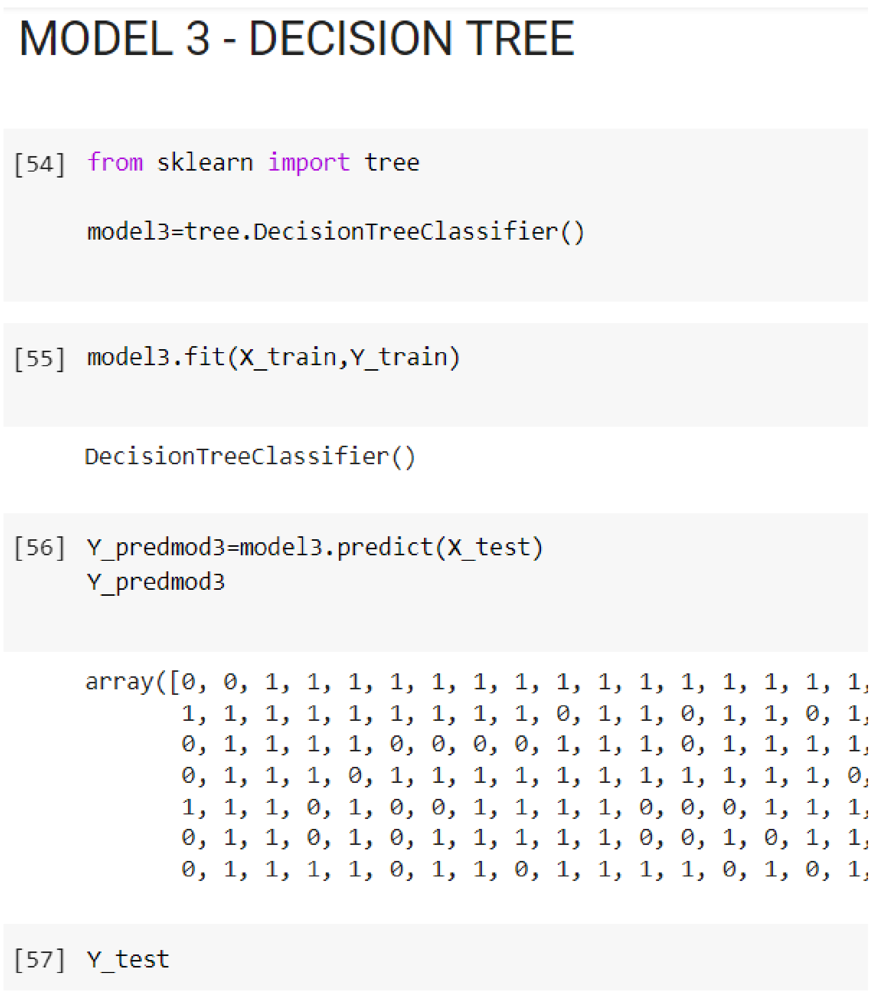


Figure 26 DT Tree Structure

It also contains a root note which is the first node on the path, a leaf node that shows the end of the decision path, and internal nodes that are present between the root and the leaf nodes. The decision tree starts at a single control statement to classify data which then splits into two or more directions.

The third model in our project is the pics they use of the decision tree. We have fitted the train data sets into the decision tree model. After making predictions we have calculated the accuracy score of 92%. A confusion matrix is also made for the same.



MODEL 3 - DECISION TREE

```
[54]  from sklearn import tree

      model3=tree.DecisionTreeClassifier()
```

```
[55]  model3.fit(X_train,Y_train)
```

```
      DecisionTreeClassifier()
```

```
[56]  Y_predmod3=model3.predict(X_test)
      Y_predmod3
```

```
      array([0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1,
             0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1,
             0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
             1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1,
             0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1,
             0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1,
```

```
[57]  Y_test
```

Figure 27 DT Code

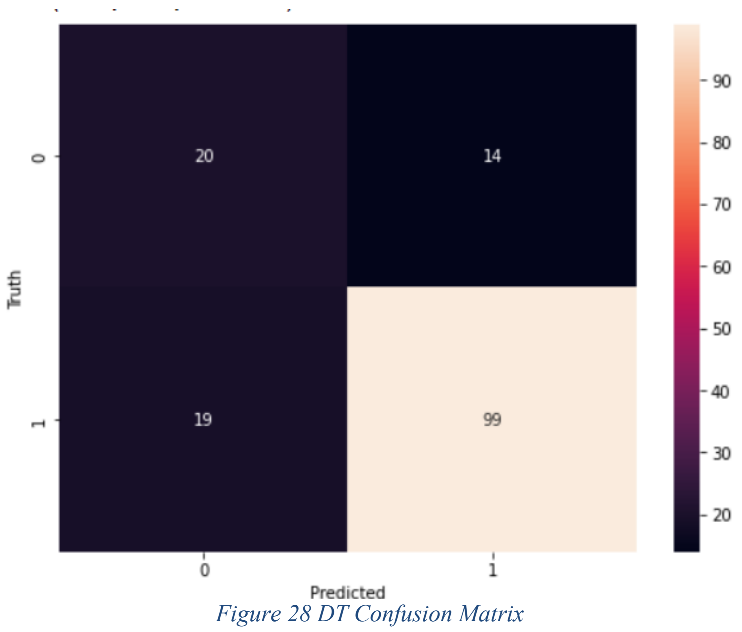


Figure 28 DT Confusion Matrix

4.1.4 Support Vector Machine Model:

The vector support machine is a supervised learning algorithm for regression and classification questions and scenarios. Creates a decision boundary that can divide N-dimensional spaces into classes; this decision boundary is called a hyperplane. It selects excess points to create hyperplanes and is called support vectors. We have two types of vertical and indirect vector learning machines. Is Kim creating multiple decision boundaries to segregate data but we need to find out the best decision boundary to classify our data The best decision boundary is known as the hyperplane supporting vectors are the data points that are in the most proximity to the hyperplane.

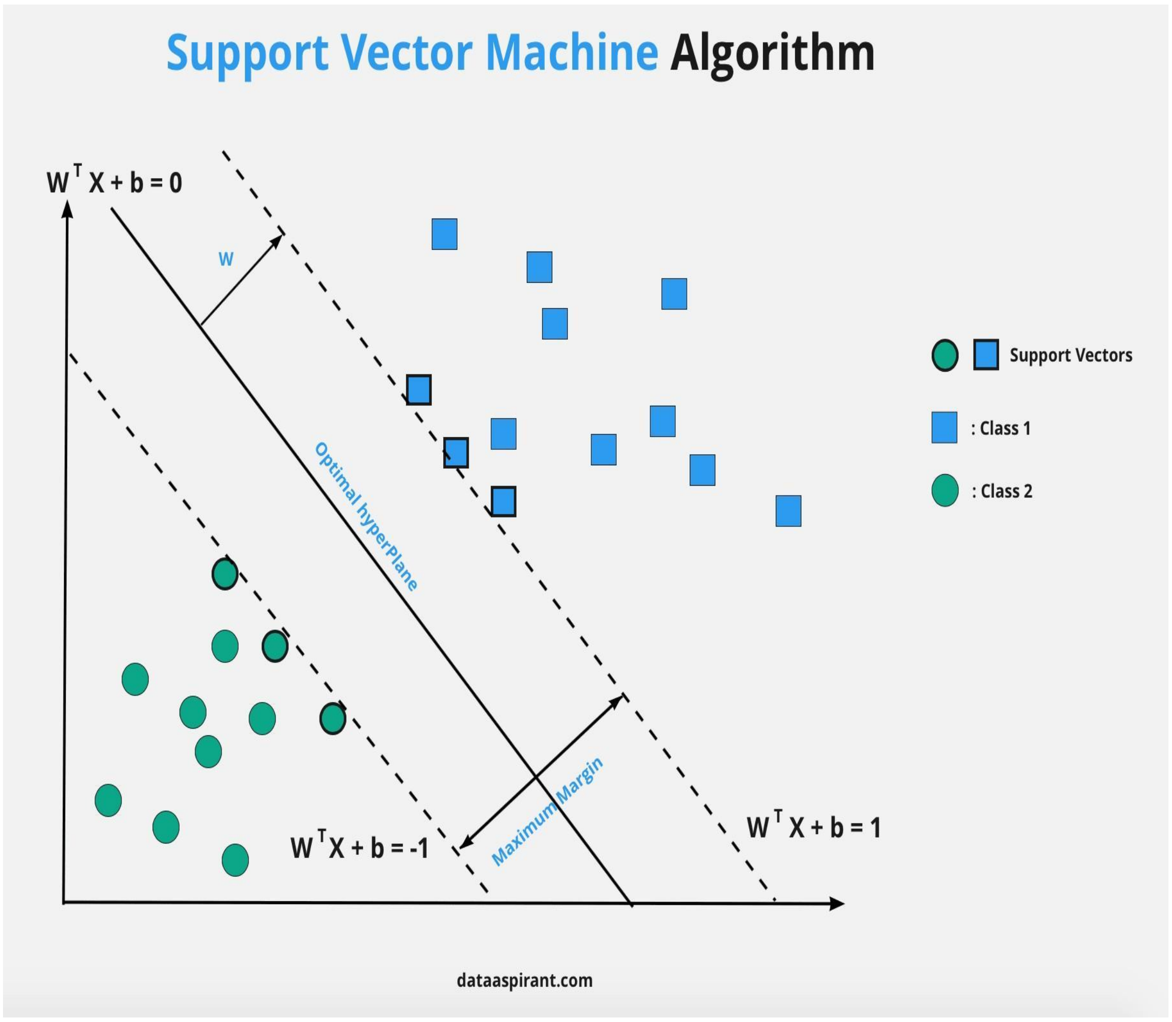
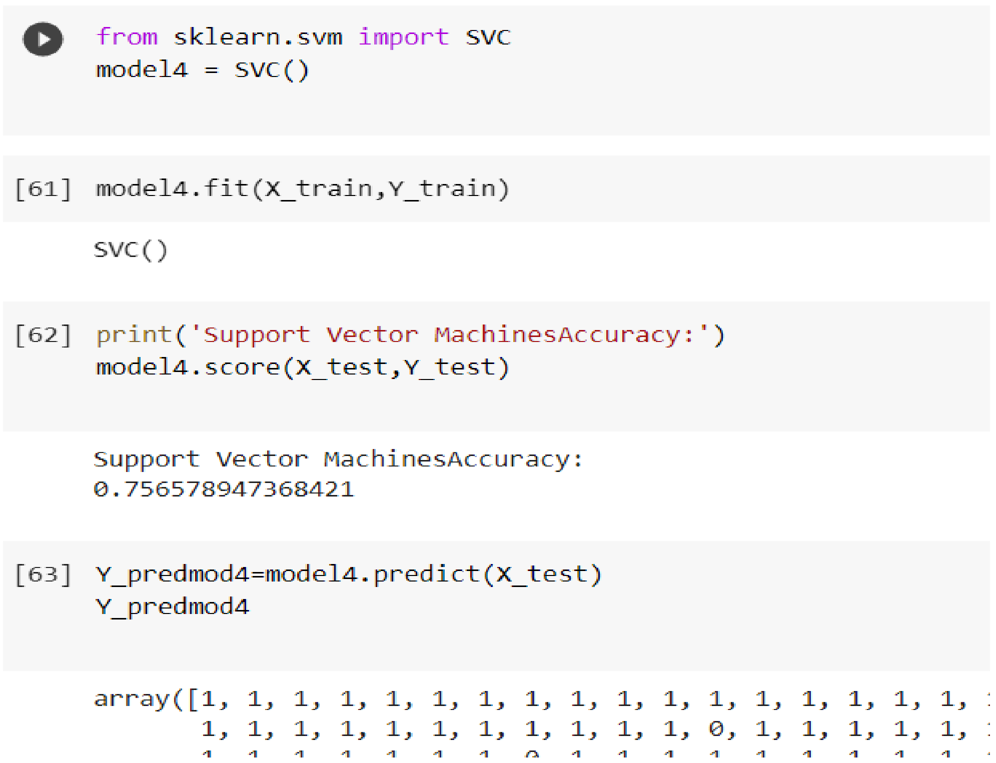


Figure 29 SVM 1

In our 4th model, we have applied the support vector machine method for which we have imported SVM from SK learn then fitted it with our X train and Y train values. After that we have predicted the accuracy of 84%, we have also constructed a confusion matrix for the same model.

MODEL 4 - SUPPORT VECTOR MACHINE



```
from sklearn.svm import SVC
model4 = SVC()
```

```
[61] model4.fit(X_train,Y_train)
```

```
SVC()
```

```
[62] print('Support Vector MachinesAccuracy:')
     model4.score(X_test,Y_test)
```

```
Support Vector MachinesAccuracy:
0.756578947368421
```

```
[63] Y_predmod4=model4.predict(X_test)
     Y_predmod4
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
```

Figure 30 SVM Code

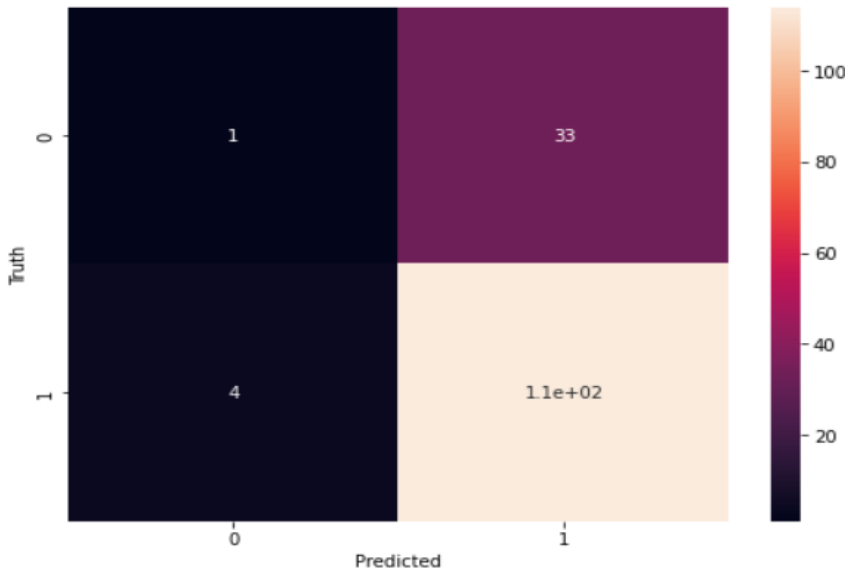


Figure 31 SVM Confusion Matrix

4.1.5 Random Forest Model:

Another supervised learning technique that we have used in our model is random forest classifier it is also among the most popular algos used for classification or regression problems random forest classifier predicts data by merging multiple decision trees with the help of randomness it enhances the prediction accuracy and provides the better result as compared to the decision tree random forest classifier can be used in various disease prediction, various economical prediction, and prediction in electronics and communication sector. If a data set will have N features it will select some random features which are less than the value of total features and will calculate the root node among those fewer selected features by picking the node with the highest information then the algorithm splits the node into child node and repeats the procedure. The random forest algorithm is more reliable than the decision tree divider and will not deal with any overfill problems and can be used for both regression and classification problems. Missing values also does not cause any issue while using random forest. I realize this is slower than other algorithms because of multiple decision tree formation and some time is unfit for fast predictions.

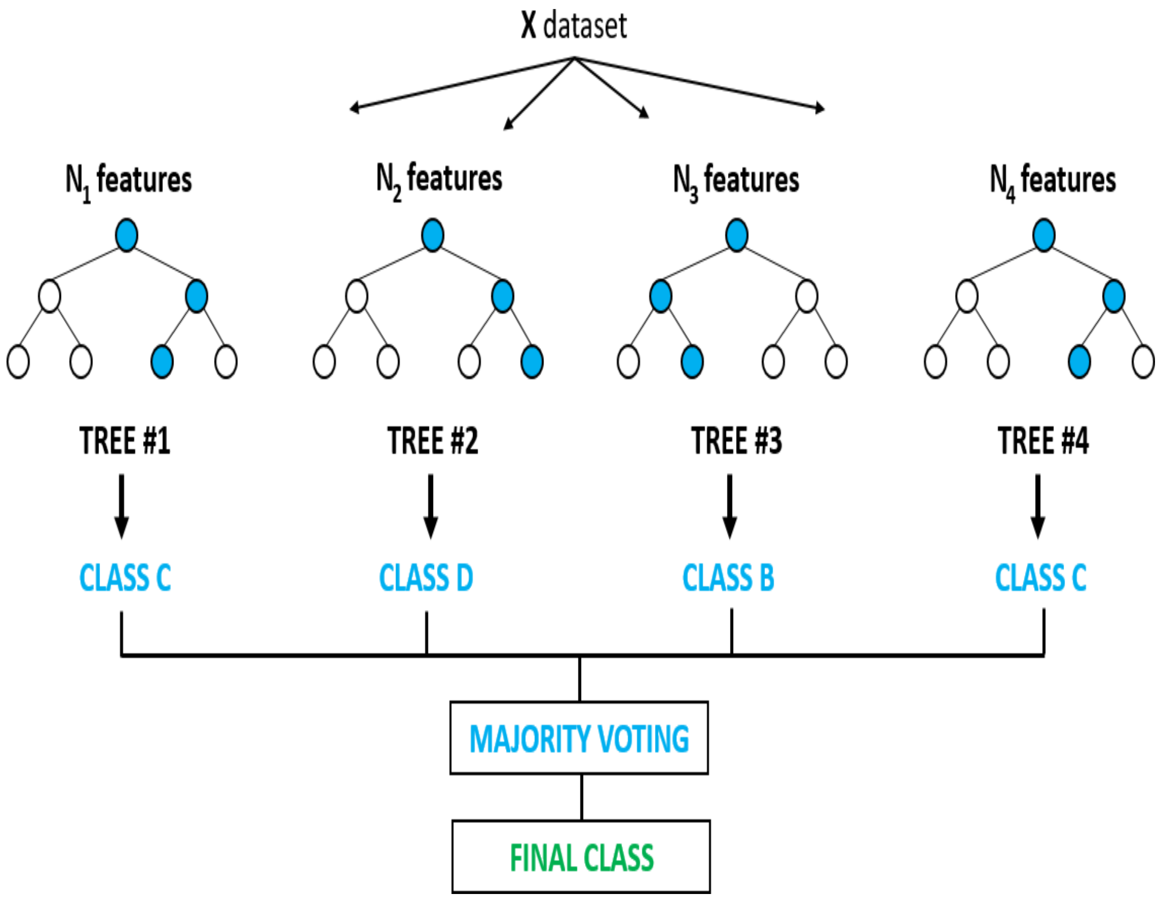


Figure 32 RF 1

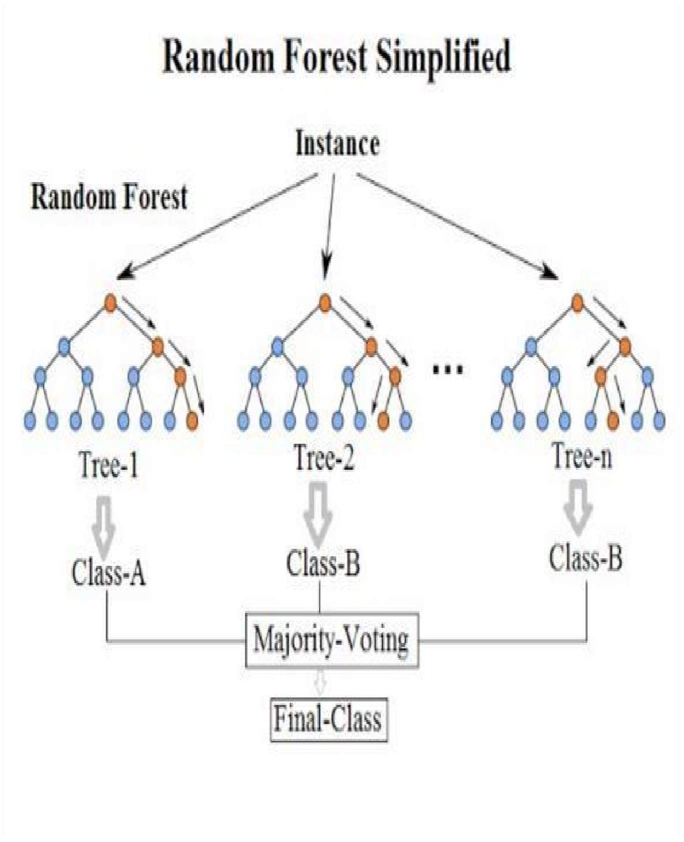


Figure 33 RF 2

In our 5th model, we have applied the Random Forest Classifier for that we have imported Random Forest Classifier from SK learn then fitted it with our X train and Y train values after that we have predicted the accuracy of 97%, we have also constructed a confusion matrix for the same

```
Text(69.0, 0.5, 'Truth')
```

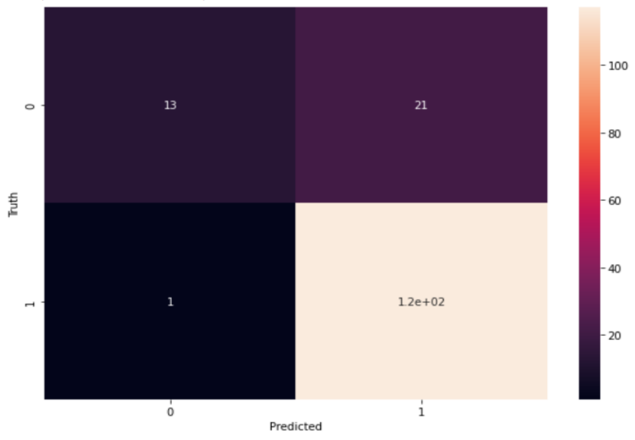
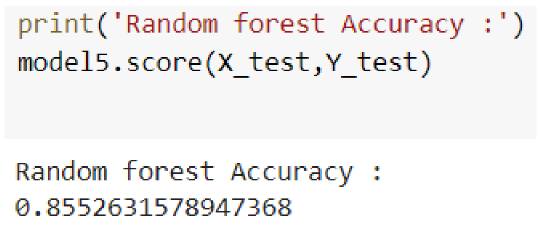


Figure 34 RF Confusion Matrix



```
print('Random forest Accuracy :')
model5.score(X_test,Y_test)
```

```
Random forest Accuracy :
0.8552631578947368
```

Figure 35 RF Accuracy

4.1.6 XGBOOST:

XG boost or extreme gradient boost is designed for being highly portable and uses machine learning algorithms under the framework of gradient solving data science problems in a fast and accurate way. It is the extension of the gradient boosted decision tree and it's much faster than the GBM. It supports regularized learning, gradient tree boosting and shrinkage, and column subsampling. It has a remarkably fast speed as compared to other algorithms and the model performance is also far better than any other model used for supervision and prediction. The algorithm works on a decision tree and constructs a graph after examining various if statements this algorithm will add more and more if to the tree for stronger model construction. XGBoost is a boosting-based ensemble learning method. In Boosting, the trees are arranged in such a way that each subsequent tree aims to reduce the flaws of the previous tree. Each tree learns from its predecessors and reviews the remaining errors. Therefore, the tree that grows next in succession will learn from the revised version of the fossils. The boosting ensemble strategy consists of three simple steps:

- The first model F0 is defined by the predictable variable prediction y. This model will be associated with the remainder (y - F0)
- The new h1 model is equal to the remnants of the previous step
- Now, F0 and h1 are combined to offer F1, an improved version of F0. The average square error from F1 will be lower than that from F0.

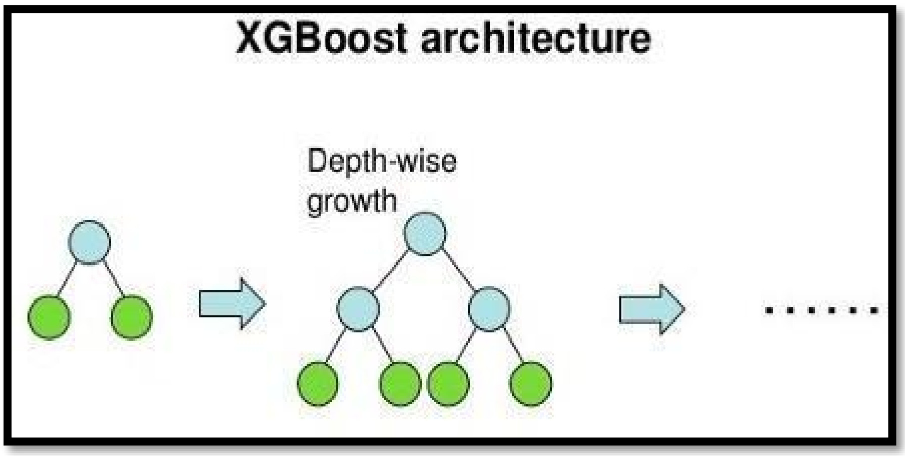


Figure 37 XGB 1

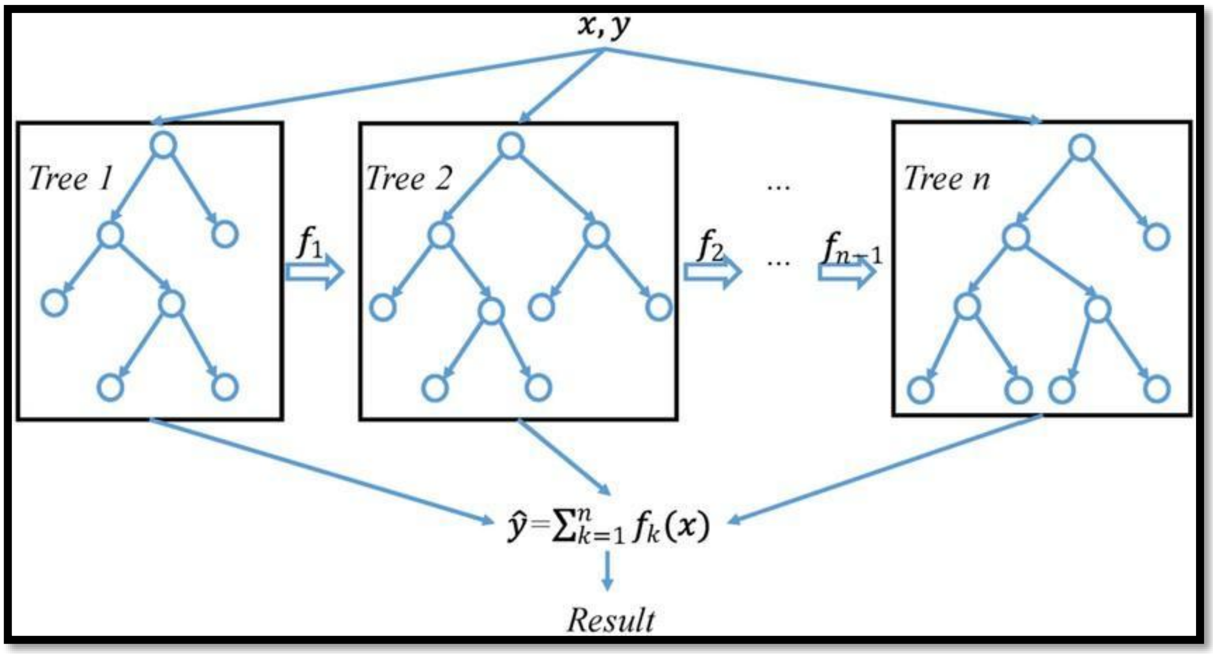
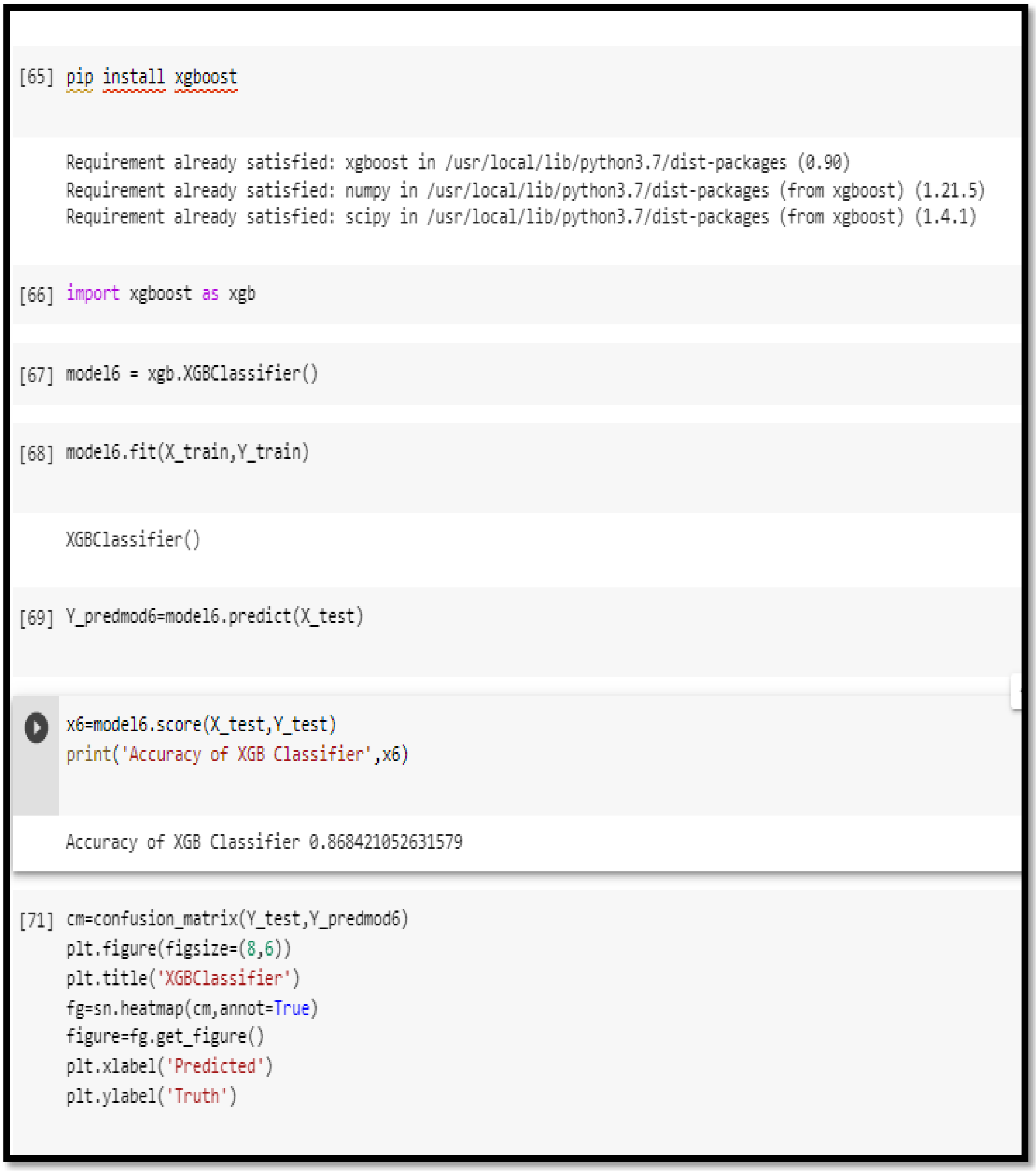


Figure 36 XGB Multiple Tree

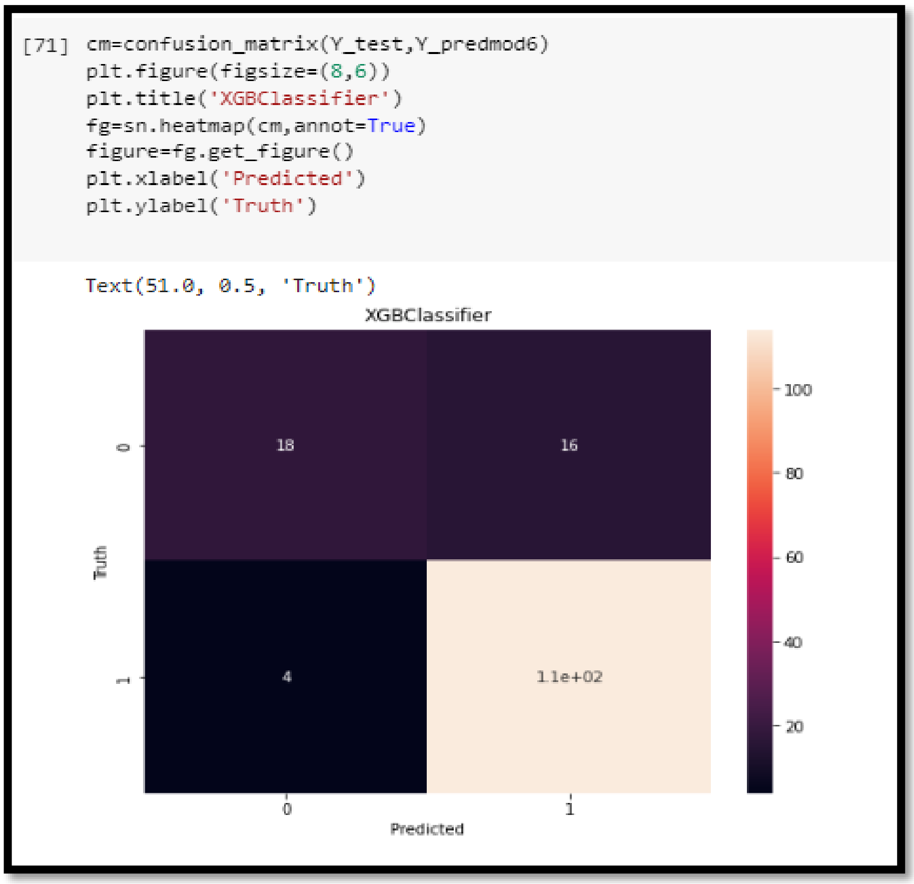
We have constructed our 6th model with the help of XG booster for that first we have installed XG booster then imported it then we have fitted X train and Y train values in our model after that we have predicted values and calculated need accuracy with XG booster we have got an accuracy of 100% after that we have constructed a confusion matrix for the same model mentioned above.

The implementation and the confusion matrix for the model mentioned above is been presented:



```
[65] pip install xgboost

     Requirement already satisfied: xgboost in /usr/local/lib/python3.7/dist-packages (0.90)
     Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from xgboost) (1.21.5)
     Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from xgboost) (1.4.1)

[66] import xgboost as xgb

[67] model6 = xgb.XGBClassifier()

[68] model6.fit(X_train,Y_train)

     XGBClassifier()

[69] Y_predmod6=model6.predict(X_test)

     x6=model6.score(X_test,Y_test)
     print('Accuracy of XGB Classifier',x6)

     Accuracy of XGB Classifier 0.868421052631579

[71] cm=confusion_matrix(Y_test,Y_predmod6)
     plt.figure(figsize=(8,6))
     plt.title('XGBClassifier')
     fg=sn.heatmap(cm,annot=True)
     figure=fg.get_figure()
     plt.xlabel('Predicted')
     plt.ylabel('Truth')
```

Figure 38 XGB Code



```
[71] cm=confusion_matrix(Y_test,Y_predmod6)
     plt.figure(figsize=(8,6))
     plt.title('XGBClassifier')
     fg=sn.heatmap(cm,annot=True)
     figure=fg.get_figure()
     plt.xlabel('Predicted')
     plt.ylabel('Truth')
```

```
Text(51.0, 0.5, 'Truth')
```

Figure 39 XGB Confusion Matrix

4.1.7 Neural Network:

With Keras and TensorFlow.

Neural networks use a series of algorithms that establish relationships between data and work like a human brain system where each note acts as a neuron. It is an unsupervised learning algorithm so it relies on training data to improve accuracy overtime. In general terms neural networks are used to find patterns in each data set and form nodes (works like a human brain neurons). There are three important layers for proper functioning of a neural network. (*Input Layer, Hidden Layer & Output Layer.)

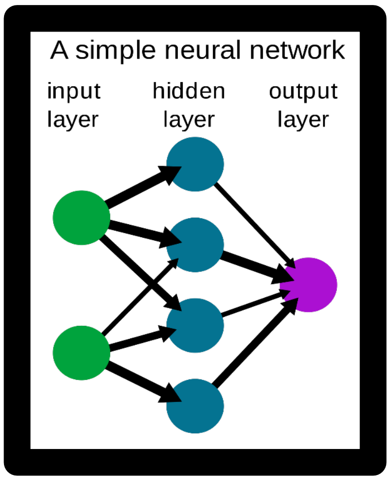


Figure 40 NN 1

The Input Layer is providing nodes with data provided by the user to the model. This is the original data which is trained and passed to the next layer, THE HIDDEN LAYER.

The Hidden Layers will contain a series of sequences minimum one and one sequence will have a series of nodes/neurons. Model will train and learn from previous errors and will eventually pass to Output Layers.

The Output Layer, this layer will consist of conclusions. There will be a single or multiple nodes in the Output Layer.

Inputs or an Input put layer with initial data is provided to the model where some weights are assigned to these nodes. These nodes are then provided to hidden layers, which can be multiple or single. These hidden layers will do some computations on the data. These inputs are multiplied by their respective Random, Unique weights after the assigning of weights Bias is added which will help model's fitting.

$$Z_1 = W_1{}^{*}In_1 + W_2{}^{*}In_2 + W_3{}^{*}In_3 + W_4{}^{*}In_4 + W_5{}^{*}In_5 + b$$

Figure 41 NN Equation

Here Z1 is the single sequence, In1…. Inn is the input, W1……. Wn is the weights and b are a Bias constant. An Activation function is applied to the equation; this entire weight assignment and activation function is applied to each hidden layer and then we move to our output layer. After predictions errors are calculated, if huge errors are found, we must use certain methods to minimize these errors. Such as the Back Propagation method, here optimum values of weights are found which help in minimizing the errors. For this purpose, a gradient descent method is used. In this method random weights are assigned to data and intercepts are assigned to the model. Gradient is calculated and which is differentiating error with weights. The new weights are calculated and the process is repeated until we reach a global minimum.

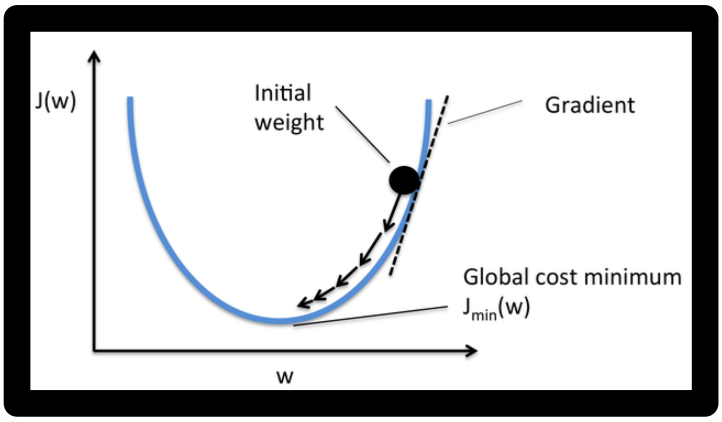


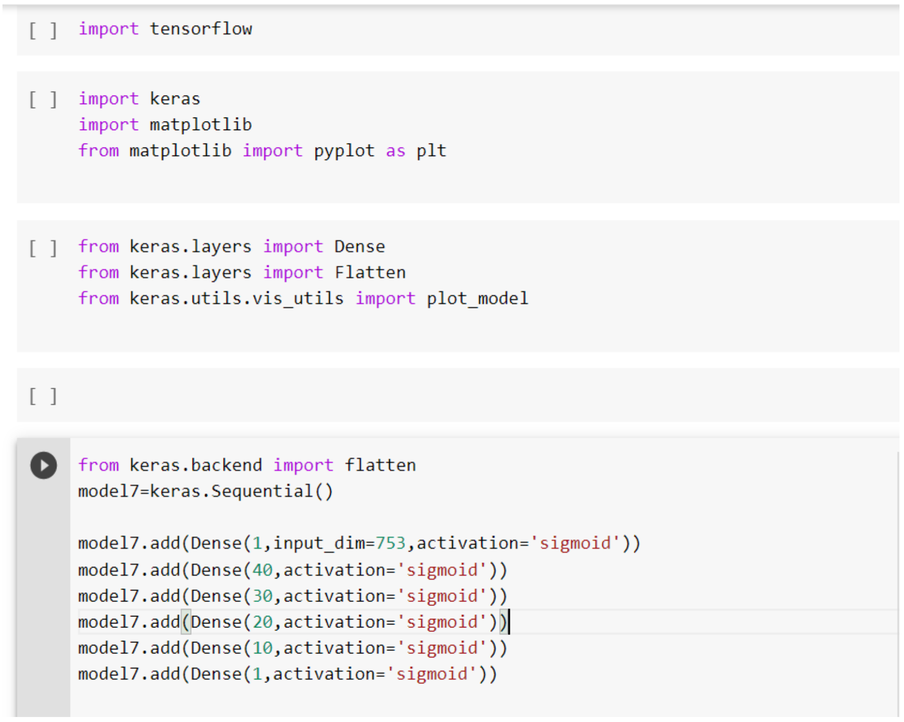
Figure 42 nn 3

The Activation function is applied to layers which determine whether a neuron should be or not be a part of models. Few Activation functions are:

- Sigmoid
- TanH
- Rectified
- Leaky ReLU
- Softmax

TensorFlow: Tensor flow is an open-source software used for machine learning and artificial language. First introduced in 2015, the platform is written in Python, C++, and CUDA. TensorFlow can be used for Auto Differentiation, Eager Execution, Distribute, Losses and Metrics.
It has applications in the Medical, social media, Search Engine, Education, Retail, and Research sector.

Keras: Keras is open-source software and acts as an interface for TensorFlow. Keras contains numerous implementations communally used for building layers in neural networks. It also has support for convolutional and recurrent neural networks.
The 7th model is based on neural networks. In this model, I first Installed important libraries like TensorFlow and Keras. After these libraries, I made layers using Keras.Sequential(). In this model I created 5 hidden layers with sigmoid as the activation function. I trained the model with epochs=1000 the model upon evaluation gave an accuracy of 77% and a loss of 53%. The confusion matrix gave 120 true positives, 0 false negatives, 34 false positives, and 0 true negatives. Codes Section



```
[ ] import tensorflow
```

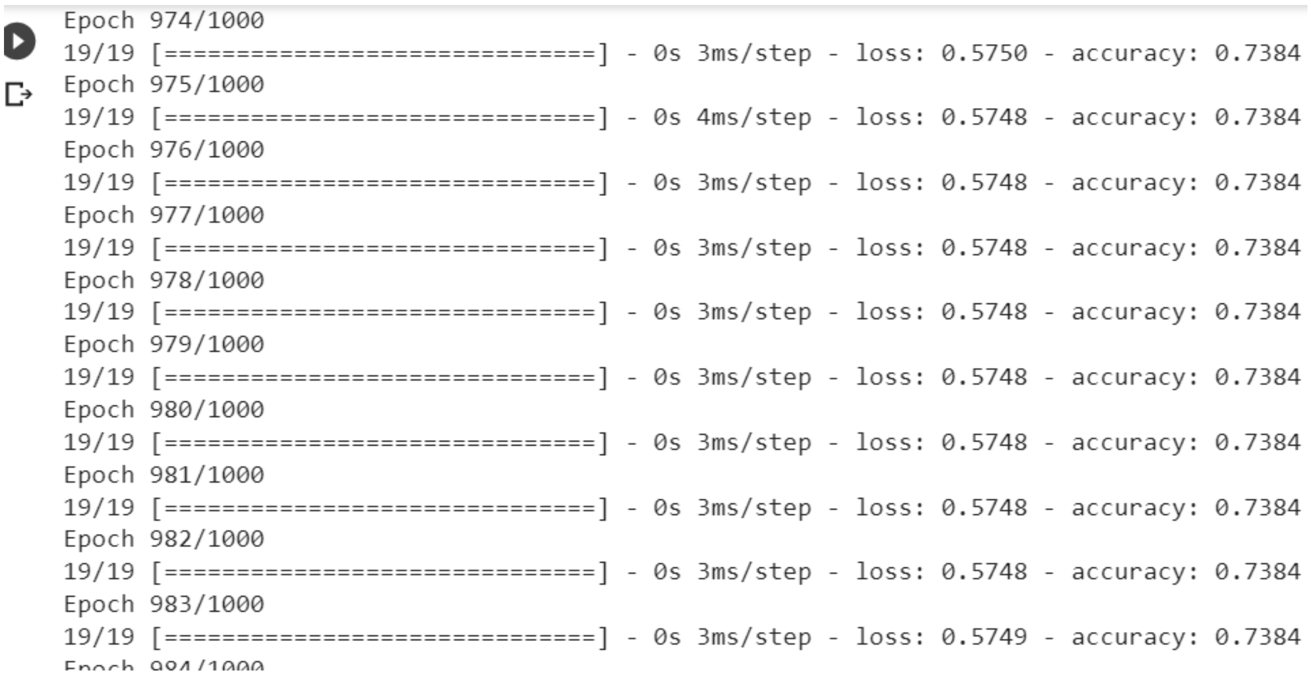
```
[ ] import keras
    import matplotlib
    from matplotlib import pyplot as plt
```

```
[ ] from keras.layers import Dense
    from keras.layers import Flatten
    from keras.utils.vis_utils import plot_model
```

```
[ ]
```

```
from keras.backend import flatten
model7=keras.Sequential()

model7.add(Dense(1,input_dim=753,activation='sigmoid'))
model7.add(Dense(40,activation='sigmoid'))
model7.add(Dense(30,activation='sigmoid'))
model7.add(Dense(20,activation='sigmoid'))
model7.add(Dense(10,activation='sigmoid'))
model7.add(Dense(1,activation='sigmoid'))
```

Figure 43 NN Code



```
Epoch 974/1000
19/19 [==============================] - 0s 3ms/step - loss: 0.5750 - accuracy: 0.7384
Epoch 975/1000
19/19 [==============================] - 0s 4ms/step - loss: 0.5748 - accuracy: 0.7384
Epoch 976/1000
19/19 [==============================] - 0s 3ms/step - loss: 0.5748 - accuracy: 0.7384
Epoch 977/1000
19/19 [==============================] - 0s 3ms/step - loss: 0.5748 - accuracy: 0.7384
Epoch 978/1000
19/19 [==============================] - 0s 3ms/step - loss: 0.5748 - accuracy: 0.7384
Epoch 979/1000
19/19 [==============================] - 0s 3ms/step - loss: 0.5748 - accuracy: 0.7384
Epoch 980/1000
19/19 [==============================] - 0s 3ms/step - loss: 0.5748 - accuracy: 0.7384
Epoch 981/1000
19/19 [==============================] - 0s 3ms/step - loss: 0.5748 - accuracy: 0.7384
Epoch 982/1000
19/19 [==============================] - 0s 3ms/step - loss: 0.5748 - accuracy: 0.7384
Epoch 983/1000
19/19 [==============================] - 0s 3ms/step - loss: 0.5749 - accuracy: 0.7384
Epoch 984/1000
```

Fitting model

```
[60] model7.evaluate(X_test,Y_test)

     5/5 [==============================] - 0s 2ms/step - loss: 0.5352 - accuracy: 0.7763
     [0.5352029204368591, 0.7763158082962036]
```

Evaluation

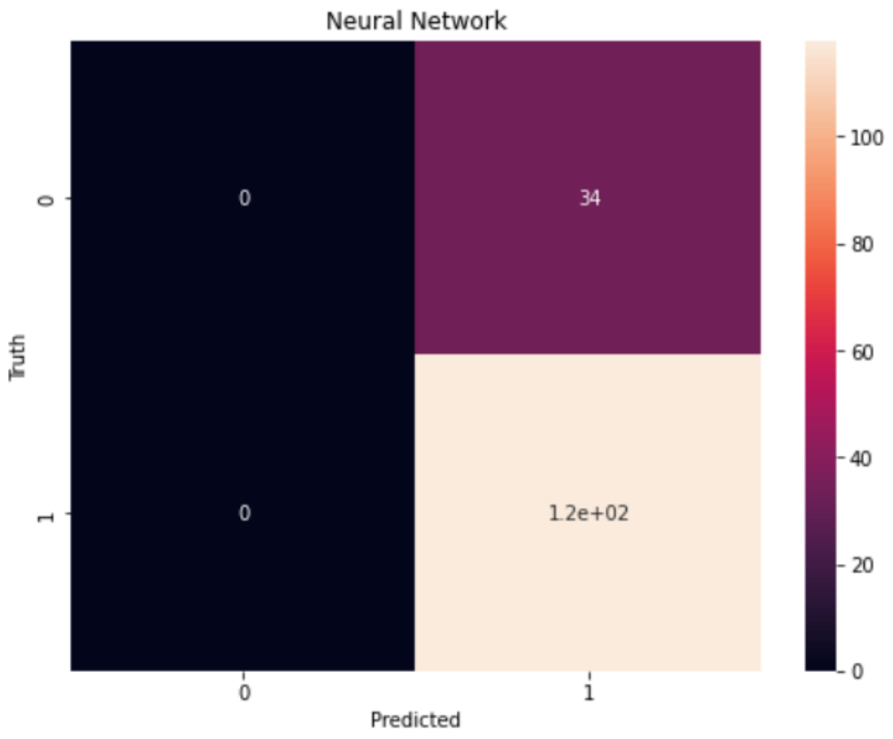


Figure 44 NN Confusion Matrix

4.1.8 AdaBoost Algorithms:

AdaBoost or Adaptive boost is an unsupervised boosting algorithm. In the boosting model a model is made from the initial data Data set and the model is rectified and is considered as new training data for the successor. Unlike Random Forest which works in parallel implementation, adaboost algorithm works in sequential implementation where the model learns from the previous model's mistake. There are mainly3 boosting algorithms Adaboost, the gradient descent algorithm and the extreme gradient descent algorithm. The boosting algorithms are different as compared to the bagging ones, in the bagging algorithms apparel processing is initiated as in boosting algorithms model learns from errors or simply retrains itself until better accuracy scores are achieved boosting algorithms promise higher accuracy, decision, recall, fiscore, precision as compared to bagging algorithms. Adaboost will start with initial data and will make a model after making the first model Adaboost will look for wrong predictions and will assign them highest weights. After the weights are assigned the amount of say is calculated on the basis of weights Like this a new model is made and again weights are assigned after this process all these models are combined and eventually a model with better accuracy is formed. Adaboost is an unsupervised machine learning model therefore requires less or no interaction of humans fingers up everyone the way in which algorithm assigns weight is completely human. Adaboost makes stumps and assigns weight to the wrong predictions.

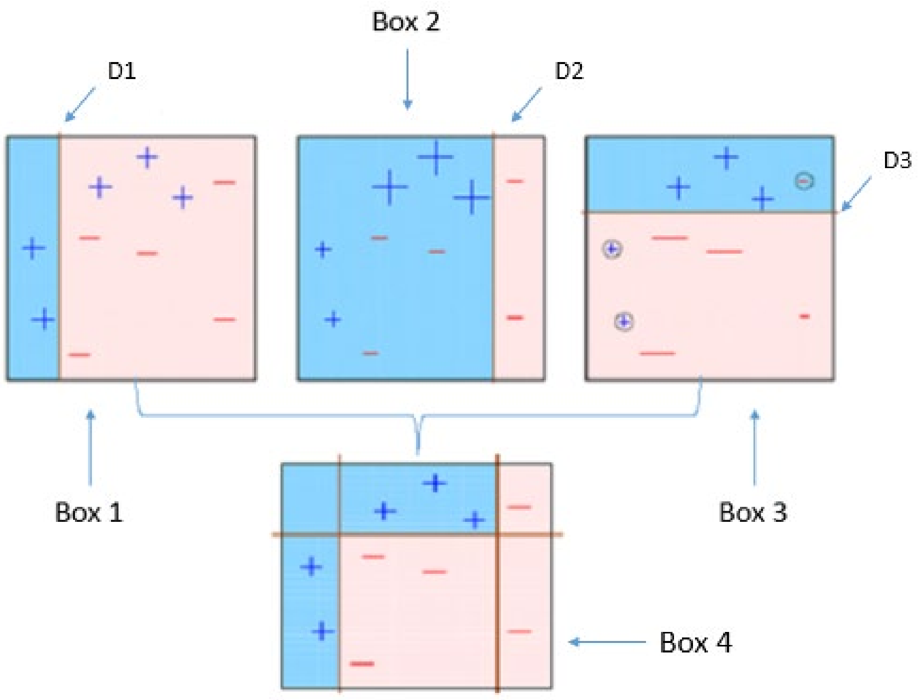
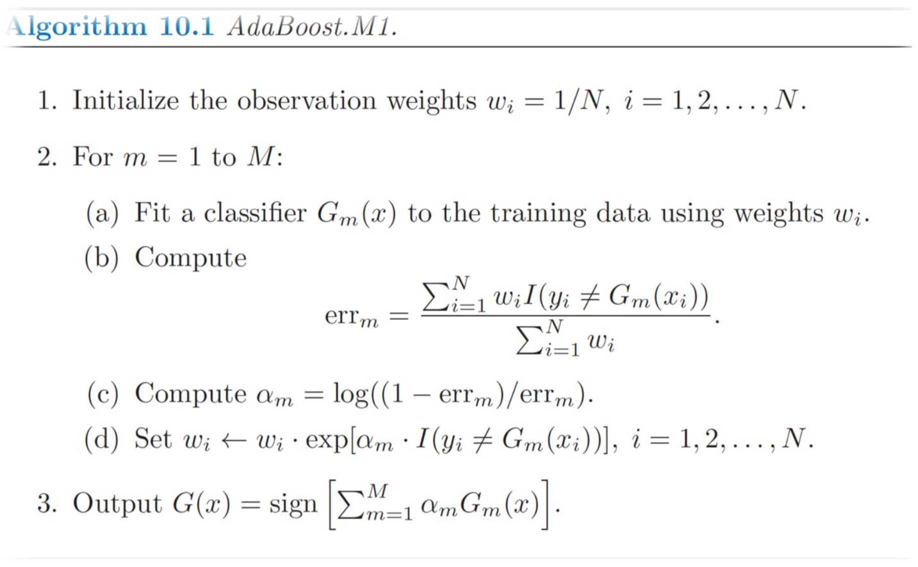


Figure 45 AfaBoost 1

In order to understand the working of AdaBoost, let's take classes + and -. The 1st model is based on initial data and it correctly predicts two positives and all negatives but incorrectly predicts three positives. For the second model the three wrong positives are given higher weight and as a result all positives are predicted correctly, 2 negatives are predicted correctly but 3 negatives are predicted incorrectly. After this for the next model, three incorrect negatives are given higher weight and the resultant model gives three positives correctly predicted, four negatives correctly predicted. After these all models also known as weak learners are combined and evaluated to form a strong learner. This Is How AdaBoost Works. Adaboost can be considered as a better replacement to Random Forest and an earlier boosting algorithm. The Adaboost have a default ensemble "decision tree" which can be changed into other's using hyperparameter tuning. I have considered two models one using hyperparameter tuning and the other without any tuning with the default settings.

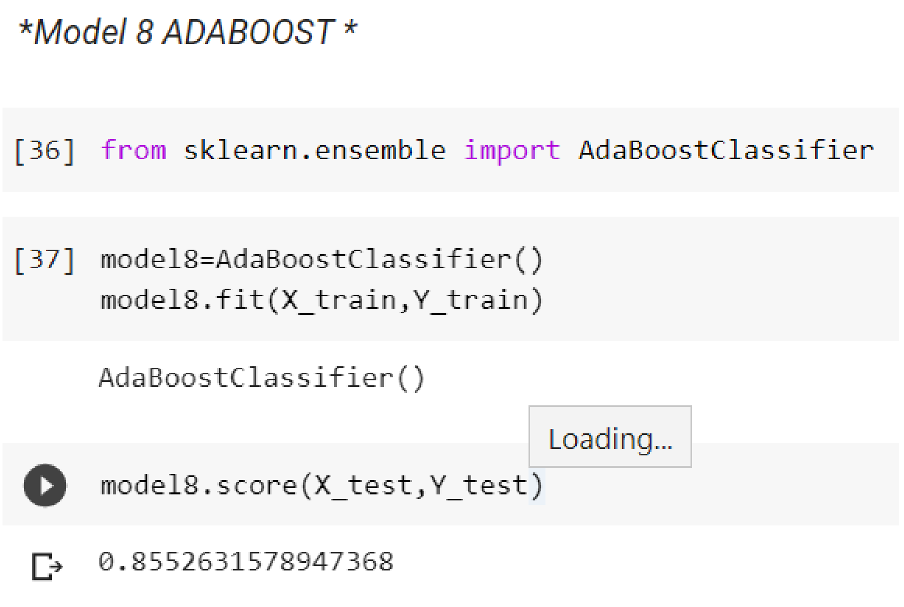


```
Algorithm 10.1 AdaBoost.M1.
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$:
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
    (b) Compute
        $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
    (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
    (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.
```

Figure 46 AdaBoost Algorithm

Using default settings:

The model 8 uses Adaboost, using sklearn. ensemble I have imported Adaboost classifiers then simply X_train and Y_train data were fitted to the classifier and the scores were predicted which were around 85%.



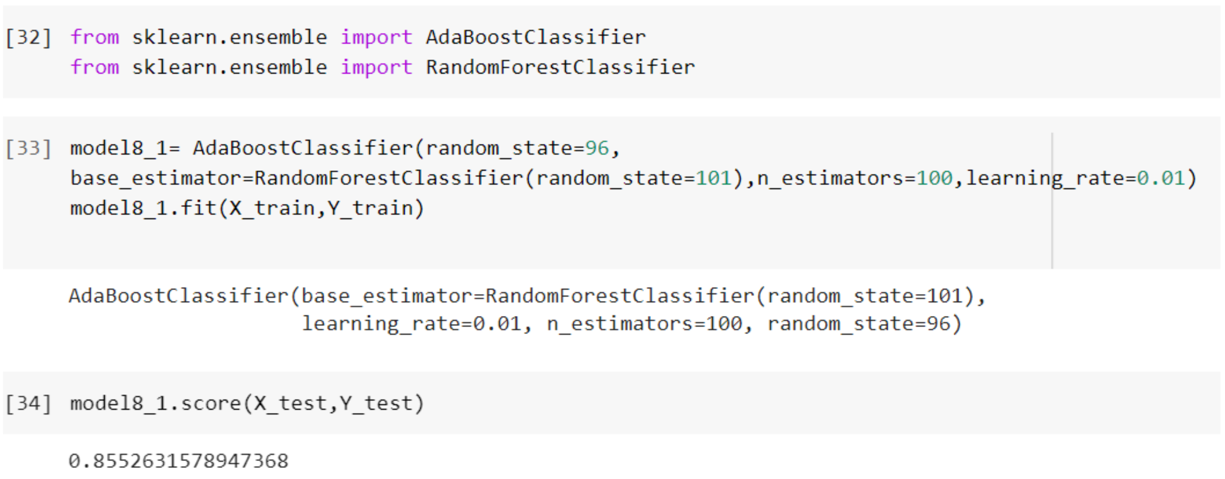
*Model 8 ADABOOST *

```
[36] from sklearn.ensemble import AdaBoostClassifier

[37] model8=AdaBoostClassifier()
     model8.fit(X_train,Y_train)

     AdaBoostClassifier()

     model8.score(X_test,Y_test)

     0.8552631578947368
```

Figure 47 AdaBoost Code

Using Hyper Parameter Tunning:

For using hyper parameter tuning I changed the base estimator to random forest classifier, n_estimator to 100, learning rate to 0.1 and random state to 101. Model 8_1 was made using Adaboost with hyperparameter tuning which gave accuracy of 85%.



```
[32] from sklearn.ensemble import AdaBoostClassifier
     from sklearn.ensemble import RandomForestClassifier
```

```
[33] model8_1= AdaBoostClassifier(random_state=96,
     base_estimator=RandomForestClassifier(random_state=101),n_estimators=100,learning_rate=0.01)
     model8_1.fit(X_train,Y_train)
```

```
     AdaBoostClassifier(base_estimator=RandomForestClassifier(random_state=101),
                        learning_rate=0.01, n_estimators=100, random_state=96)
```

```
[34] model8_1.score(X_test,Y_test)
```

```
     0.8552631578947368
```

Figure 48 Adaboost Code Hypertunning

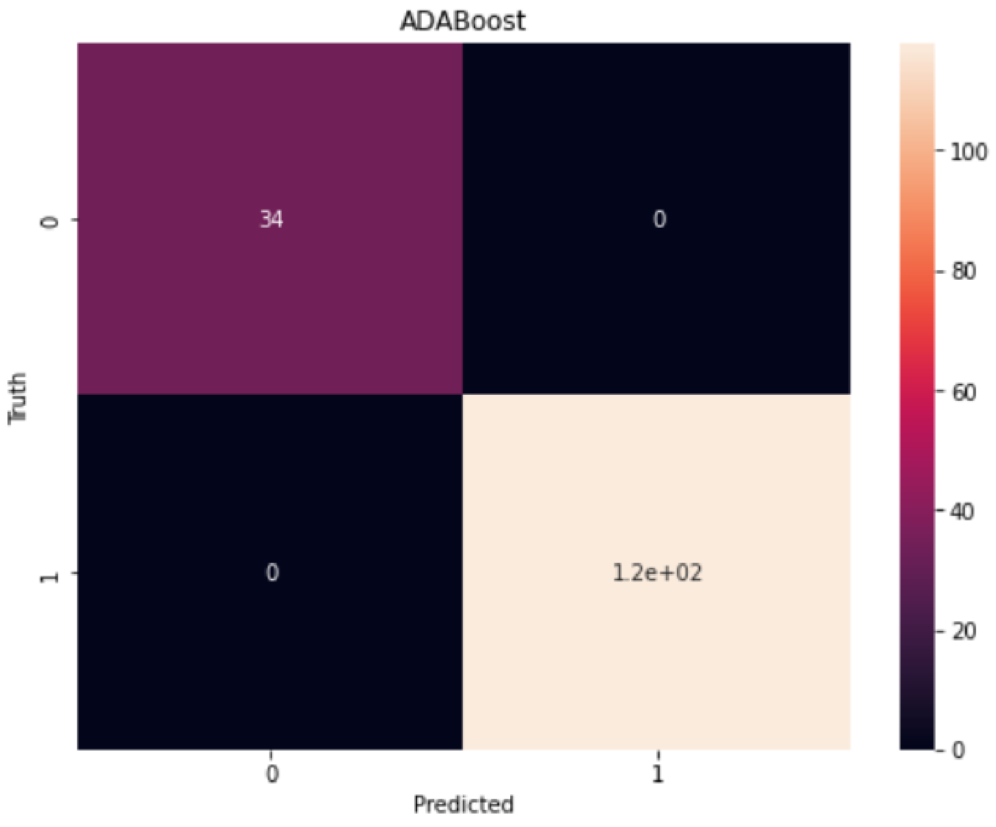


Figure 49 Adaboost Confusion Matrix

The confusion matrix gave 120 TP, 34TN, 0 FP, 0 FN.

4.1.9 Two Best Working Algorithms:

The Two Models Which are working best for the Predictor Are the XG boost and random forests.

The XGBclassifier.Give me the accuracy of 86%. Whereas the random forest gave me the accuracy of.84%. The confusion matrix. For XGBclassifier give me 122 positives with 18 true negatives, whereas the. Confusion matrix. For the random forest give me 120 true positives and 17 true negatives. The precision. For random forest.Is.89%. The recall is 0.5. And fi_ score is 64%. For the case of XG boost. The precision is 81%. Recall is 0.52. And fi_ score is 0.64. The reason why these two algorithms are the one giving accuracies in 80s and why not other algorithms are working as efficiently as these are is:

XGboost uses accurate approximations to find the best model because it is a boosting algorithm a new training data set would be created from the old original data set which will have certain improvements and error fixation that will result in greater higher accuracies. XGboost boost is also an unsupervised machine learning model because of which there is lessor no human interference which results in less errors which can be caused by human interference therefore the accuracy results or higher. The new data set which is formed is formed using random sampling and random weight assessment therefore each data set’s Value gets a say in new data formation though value with highest weight is given priority whereas the values with low weight can be discarded will stop the model also supports sequential learning model development techniques were the model for a tree is grown while converting weak learners into strong Learners a new tree will always have strong parameters Ask computer do it predecessor and will have weak parameters as compared to his successor the algorithm eventually stop when there is no difference or less difference between the predecessor and The successor. For a data set like Parkinson's disease Detector which is having larger and greater attributes and instances with approximately 750 instance and patient records of 188people the data set is huge and will have requirement for multiple model building in order to present the model with higher accuracies and less error The XG boost classifier took the initial data given to it and made new datasets and new models up until there were no or less errors and a good accuracy score was a achieved therefore that's why the model XG boost was able to give the highest accuracy486 percent.

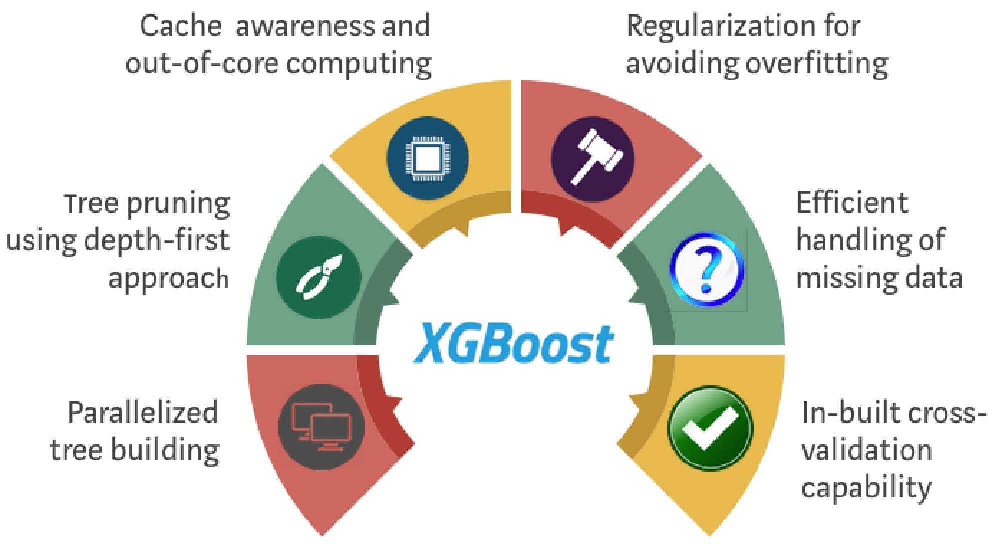

Cache awareness and out-of-core computing
Regularization for avoiding overfitting
Tree pruning using depth-first approach
Efficient handling of missing data
XGBoost
Parallelized tree building
In-built cross-validation capability

Figure 50 XGBoost Features

The Model XGBoost Also have features like Parallelized tree building, Tree pruning using depth first approach, Regularization for avoiding overfitting and an in-built Cross-Validation capability, all these result in an effective new training dataset buildup.

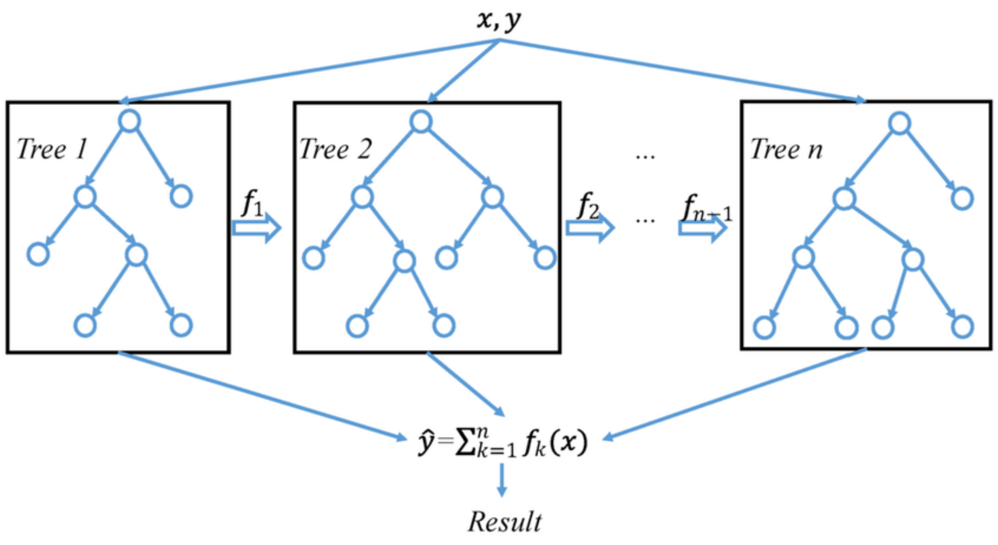
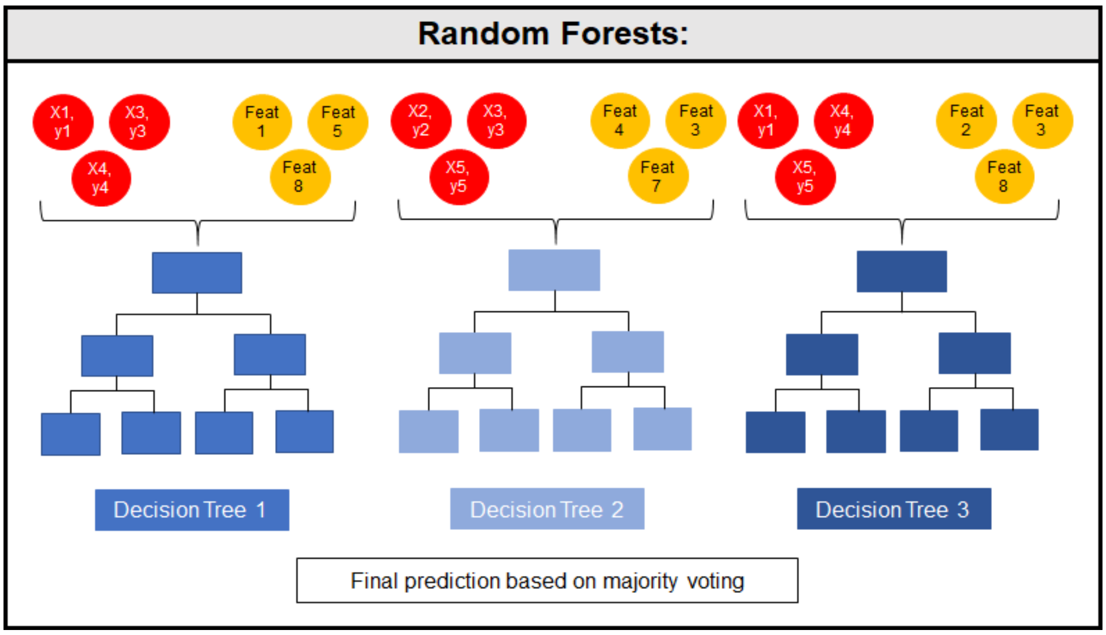
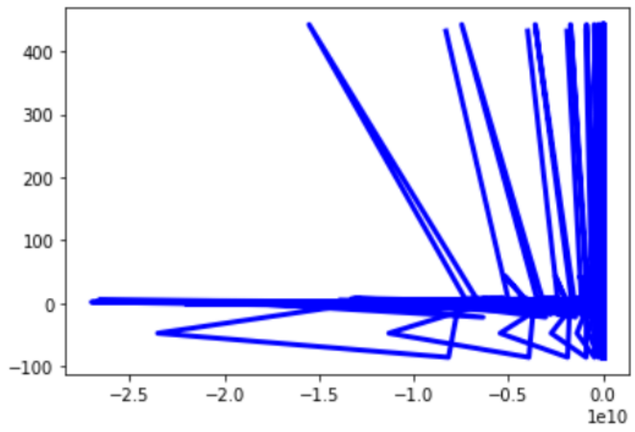


Figure Shows general working of XGBoost .

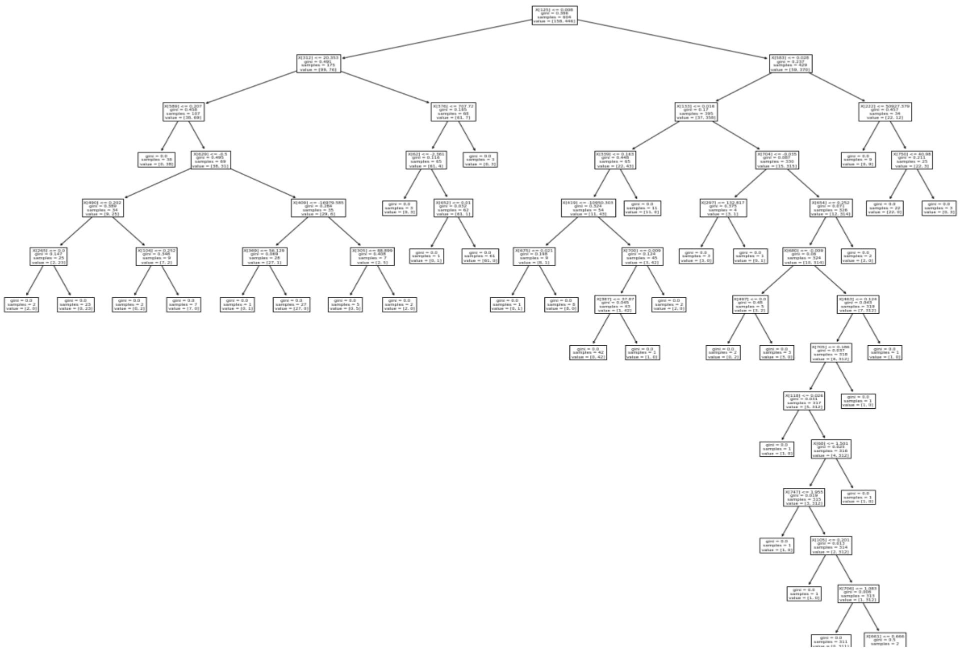
On the other hand, the next model, the random forest model gives the accuracy of 87 percent. The random forest is an upgrade to the decision tree model which gave the accuracy of 76% in my project. Random forest creates multiple decision trees segregating their information and does the prediction. The Random Forest will use multiple combinations on the dataset. Random forest is not a boosting algorithm but kind of works like XGBoost, which is one the reasons for the same accuracy. Instead of training a new dataset and creating a tree out of it (like in Random Forest), Random Forest creates multiple decision trees from the given initial datasets without modifying the data set, by just using combinations. The Random Forest because it is creating multiple decision trees, can allocate multiple data attributes and thus can use larger datasets and give good accuracies.



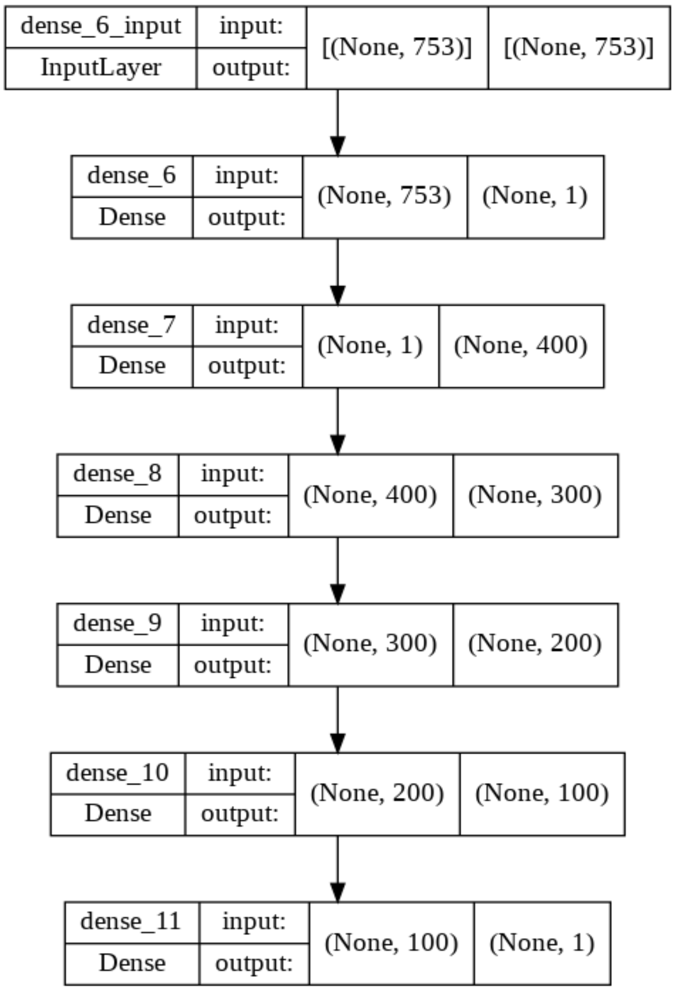
Algorithms such as linear regression gave the worst score or accuracy around -15012 The reason for this is unlike re-training and or multiple parallel model computation nothing of that sort happens in linear regression. linear regression tends to form or always assumes that there is a linear relationship between the dependent and the independent value which is not always the case especially if we are dealing with larger data. LR always looks at the mean of the dependent value which does not give the complete description of the variable, LR is sensitive to outliers and the data needs to be independent.



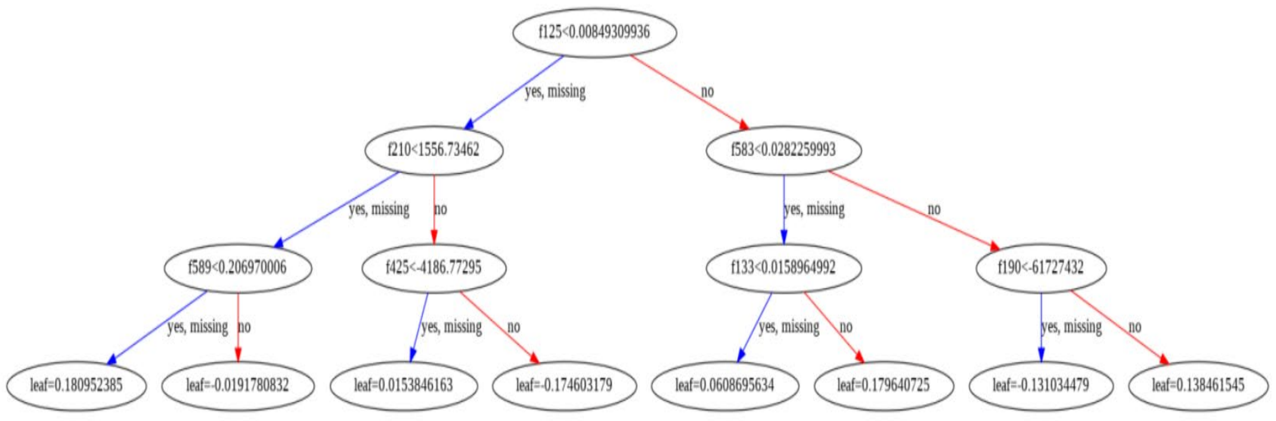
Graph 1 Linear Regression



Graph 2 DT Tree



Graph 3 NN Layers



Graph 4 XGB Boost Tree

CHAPTER 5

5.1 Conclusion:

We, in this Major Project, Developed a Parkinson's Predictor. We Initially started with a database of human voice parameters of 31 different people, 23 of which were actually suffering from the disease. we Initially wanted to develop a model with the highest accuracy and with greater true Positives and true negatives. We used several Supervised and Unsupervised algorithms including (Linear Regression, Logistic Regression, Decision Trees, Support Vector Machine, Random Forest, XGBooster Adaboost. From our calculation and models, we were able to: Achieve 87% Accuracy using Random Forest.

Accuracies:

- Achieved 75% Accuracy using SVM.
- Achieved 78% Accuracy using the Decision Tree Model.
- Achieved 75% Accuracy using the Logistic Regression Model.
- Achieved -15012 Accuracy using Linear Regression Model.
- Achieved 86% Accuracy using XGBooster Model.
- Achieved 87% Accuracy using Random Forest.
- Achieved 85% Accuracy using AdaBoost
- Achieved 77% Accuracy using Neural Networks.

Confusion Matrix:

- Linear Regression:

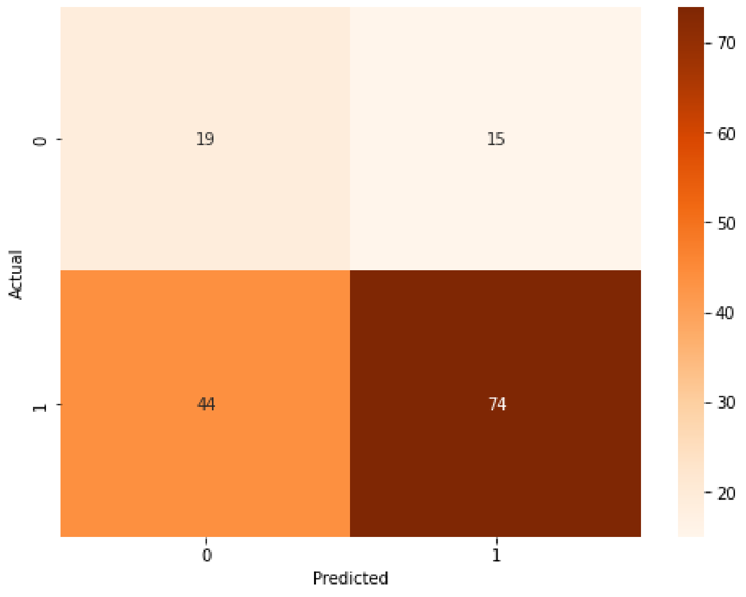


Figure 51 CM LR

TP=74, TN=19, FN=44, FP=15.

- Logistic Regression:

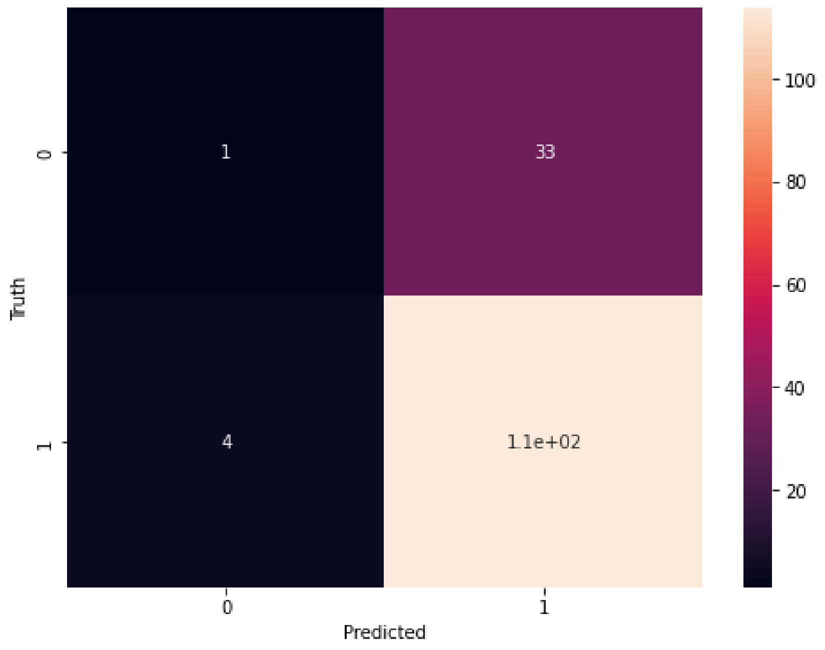
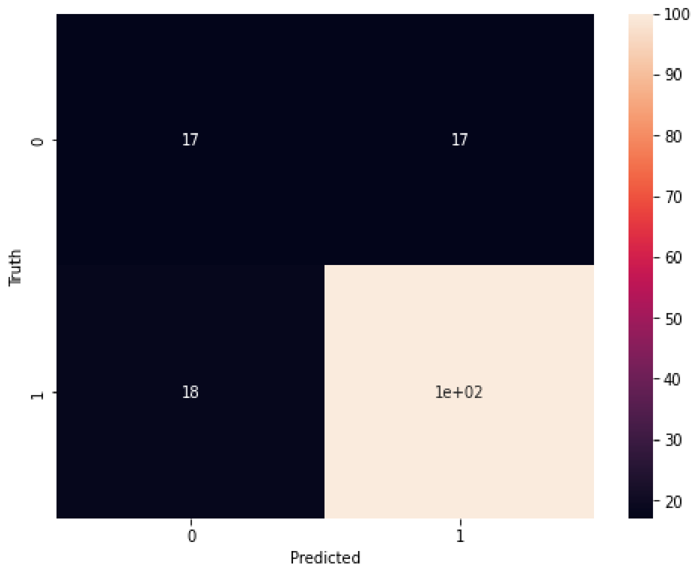


Figure 52 CM LOR

TP=110, TN=1, FN=4, FP=33



- Decision Tree:

Figure 53 CM DT

TP=110, TN=17, FN=18, FP=17

- SVM:

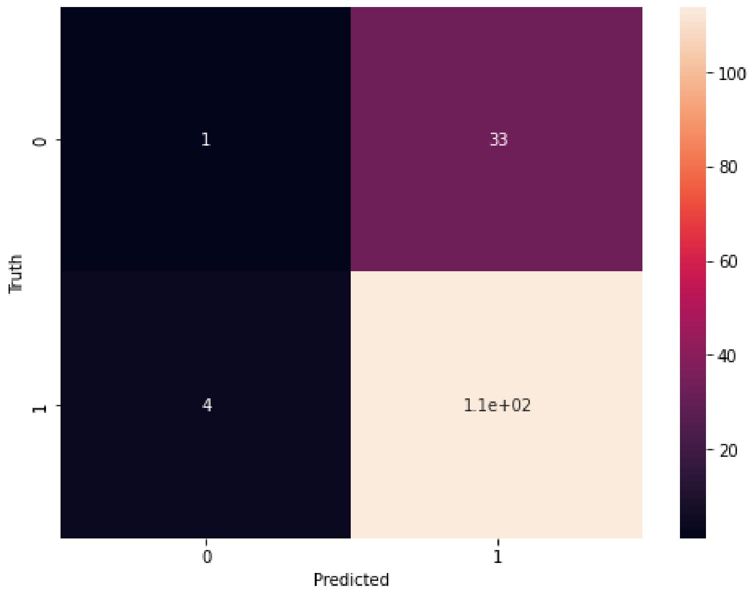


Figure 54 CM SVM

TP=110, TN=1, FN=4, FP=33

- Random Forest:

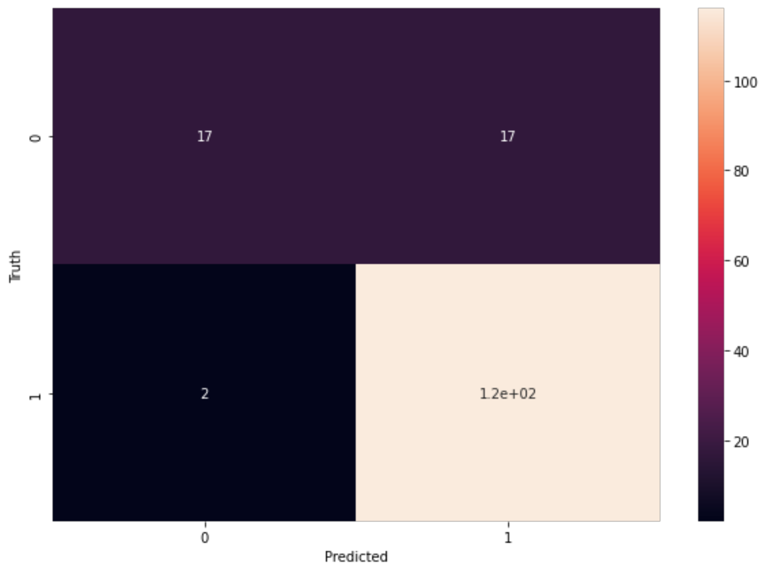


Figure 55 CM RF

TP=120, TN=17, FN=2, FP=17

- XGBoost:

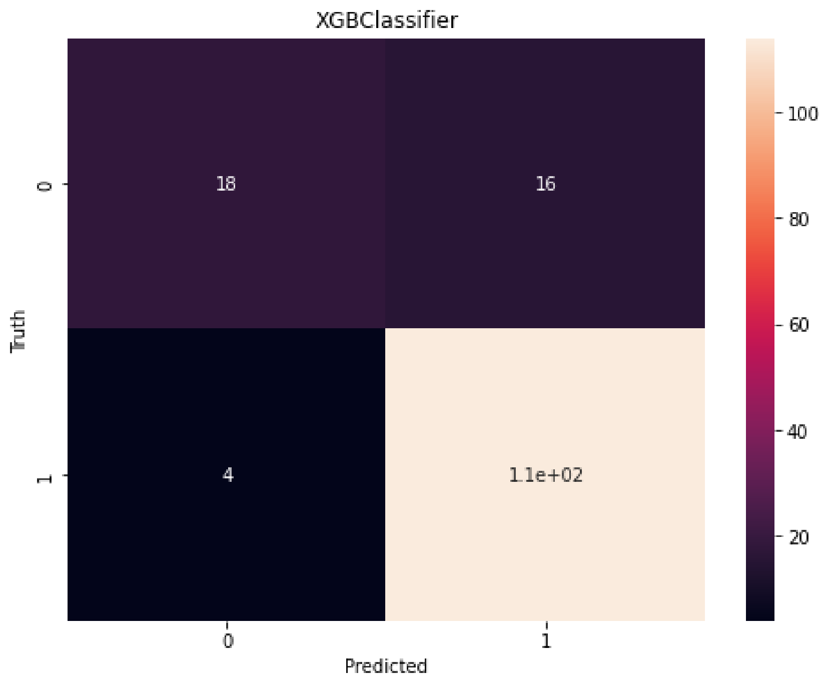


Figure 56 CM XGB

- Neural Network:

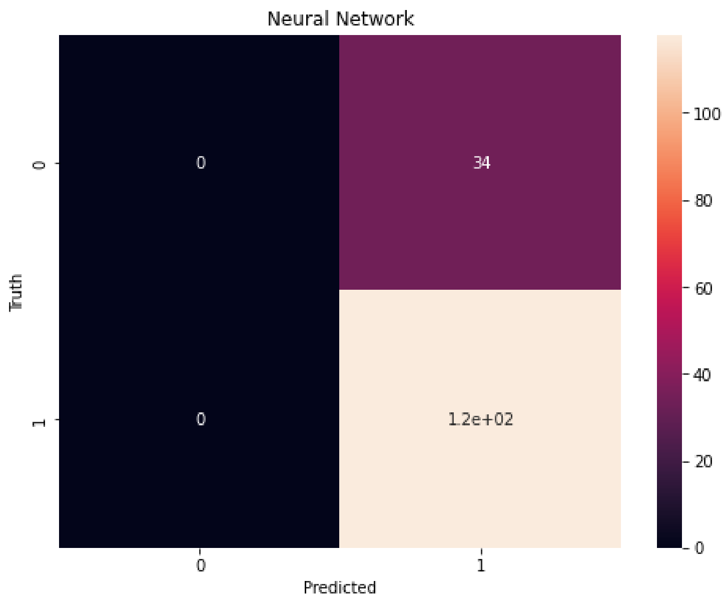


Figure 57 CM NN

TP=120, TN=0, FN=0, FP=34

- AdaBoost

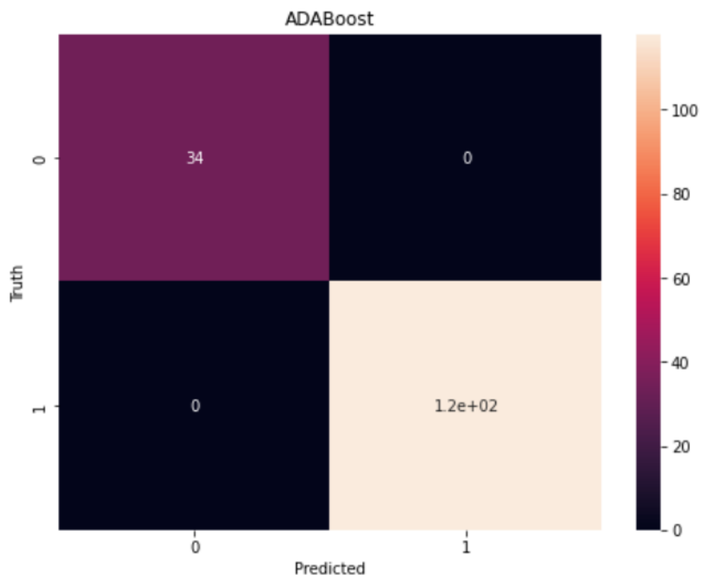
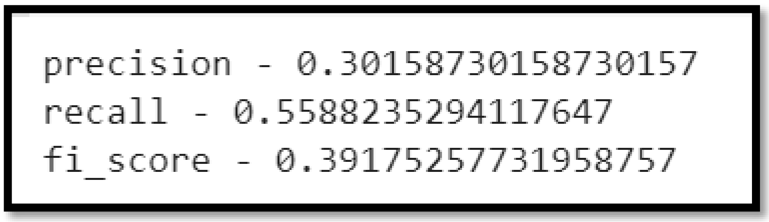


Figure 58 CM AdaBoost

TP=120, TN=34, FP=0, FN=0

Recall, Precision and fi_score.

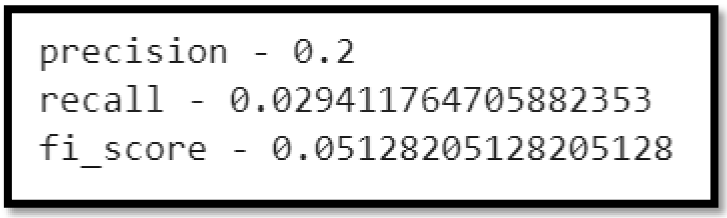
- Linear Regression:



```
precision - 0.30158730158730157
recall - 0.5588235294117647
fi_score - 0.39175257731958757
```

Figure 59 Scores LR

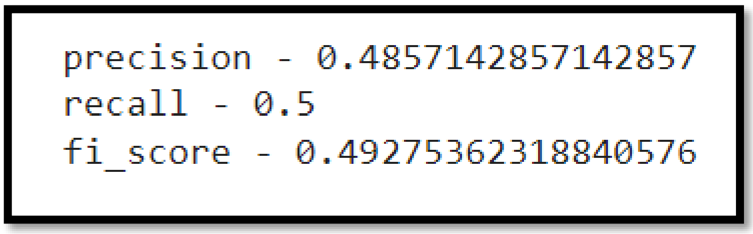
- Logistic Regression:



```
precision - 0.2
recall - 0.029411764705882353
fi_score - 0.05128205128205128
```

Figure 60 Scores LOR

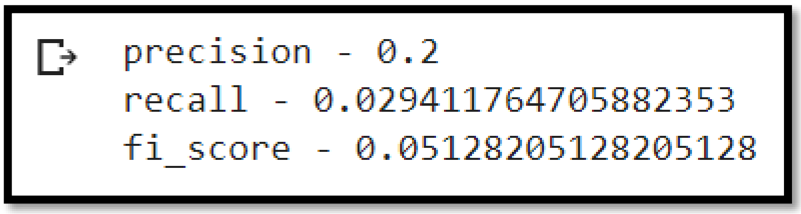
- Decision Tree:



```
precision - 0.4857142857142857
recall - 0.5
fi_score - 0.49275362318840576
```

Figure 61 Scores DT

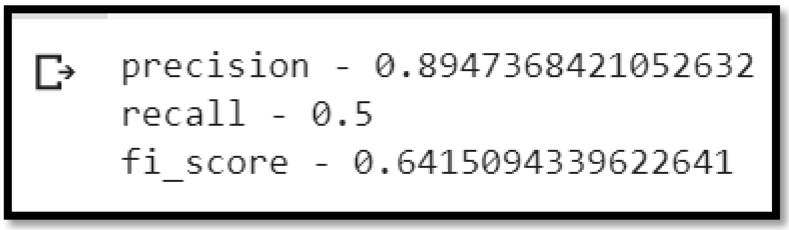
- SVM:



```
precision - 0.2
recall - 0.029411764705882353
fi_score - 0.05128205128205128
```

Figure 62 Scores SVM

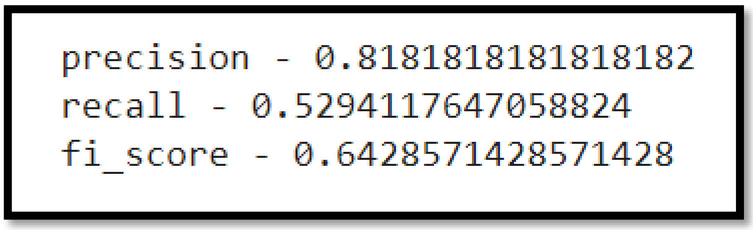
- Random Forest:



```
precision - 0.8947368421052632
recall - 0.5
fi_score - 0.6415094339622641
```

Figure 63 Sources Rf

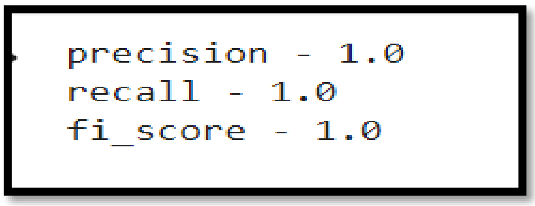
- XGboost:



```
precision - 0.8181818181818182
recall - 0.5294117647058824
fi_score - 0.6428571428571428
```

Figure 64 Scores XGB

- AdaBoost:



```
precision - 1.0
recall - 1.0
fi_score - 1.0
```

Figure 65 Sources AdaBoost

The Models efficiency in decreasing order is mentioned below:

- Random Forest
- XGBoost
- Adaboost
- Neural Network
- Decision Tree
- Support Vector machine
- Logistic Regression
- Linear Regression

5.2 Applications of the Major Project

- Biomarkers derived from the human voice can provide insight into neurological problems, such as Parkinson's disease (PD), due to their subtle cognitive function and neuromuscular function.
- PD is an ongoing neurological disease that affects about 7 million people worldwide (most of them adults), with approximately 150 thousand new clinical diagnoses each year. Historically, PD has been difficult to diagnose and physicians tend to focus on some symptoms while becoming others, relying heavily on the rating scale. Due to the declining motor control that is a symptom of the disease, the term can be used as a means of detecting and diagnosing PD.
- With the advancement of technology and the proliferation of audio collection devices in everyday life, reliable models can translate this audio data into a diagnostic tool for health care professionals who can provide a cheaper and more accurate diagnosis. We provide evidence to substantiate this concept here using a set of voice data collected from people with and without PD.
- PD is a degenerative disease which affects movement of a person. The average age for Pd is typically 55 to 65. The diagnosis of the disease is important yet difficult because of its hard diagnosis the project is working on one of the best techniques to detect the disease.
- In the project we have used Deep learning methods for PD diagnosis, clustering and classification algorithms, effective methods for analysis, scalability, and accuracies.

- The Project also sheds light on various beginners and advanced Machine Learning Algorithms, their comparison and effectiveness.
- Project also contains visualization aids such as graphs, trees and confusion matrixes for better understanding of the project.

5.3 Limitations of the Major Project:

Note in my Project I have got an 86 % prediction accuracy, I am still researching deep learning corresponding to this project as they can work better and more efficiently if the data set in hand was more complex and bigger. The methods that I have used were not able to give me 100% accuracy which was one of the goals for my project. For further enhancements, I will research other methods and will try to achieve a 100% accuracy in my project. Despite using many advanced algorithms, the accuracy was not as expected. Therefore, more study and research are required in the field to find good accuracies.

5.4 Contributions:

<u>Ullas Kumar Bherav:</u>

Linear Regression Implementation, Logistic Regression Implementation, Decision Tree Implementation, Support Vector Machine implementation, XGBoost, Neural Network.

5.5 Future Work

- Note that although in our Project we have got a 86 % prediction accuracy, we are still researching deep learning methods corresponding to this project as they can work better and more efficiently if the data set in hand was more complex and bigger.
- First, I am thinking of increasing the size of the current data set by a huge amount by gathering data off the internet or by putting some random manual data into it, then I will try to implement several other Deep Learning Advanced Techniques in order to get 100% accuracy.
- Studying about new strategies in handling data or creating model is essential in order to generate high accuracies. understanding how the data set works and what are the liabilities, for better predictions.

Reference:

1. Max A. Little, Patrick E. McSharry, Eric J. Hunter, Lorraine O. Ramig (2008),'Suitability of dysphonia measurements for telemonitoring of Parkinson's disease',IEEE Transactions on Biomedical Engineering https://dx.doi.org/10.1109%2FTBME.2008.2005954
2. Sankar, C.O., Serbes, G., Gunduz, A., Tunc, H.C., Nizam, H., Sakar, B.E., Tutuncu, M., Aydin, T., Isenkul, M.E. and Apaydin, H., 2018.)
3. Pereira, C.R., Pereira, D.R., Papa, J.P., Rosa, G.H. and Yang, X.S., 2017. Convolutional neural networks applied for Parkinson’s disease identification. In *Machine Learning for Health Informatics* (pp. 377-390). Springer, Cham.
4. Vallejo, M., Jamieson, S., Cosgrove, J., Smith, S.L., Lones, M.A., Alty, J.E. and Corne, D.W., 2016, December. Exploring diagnostic models of Parkinson's disease with multi-objective regression. In *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on* (pp. 1-8). IEEE.
5. Lones, M.A., Alty, J.E., Cosgrove, J., Duggan-Carter, P., Jamieson, S., Naylor, R.F., Turner, A.J. and Smith, S.L., 2017. A New Evolutionary Algorithm-Based Home Monitoring Device for Parkinson’s Dyskinesia. *Journal of medical systems*, *41*(11), p.176.
6. Pereira, C.R., Pereira, D.R., da Silva, F.A., Hook, C., Weber, S.A., Pereira, L.A. and Papa, J.P., 2015, June. A step towards the automated diagnosis of Parkinson’s disease: Analysing handwriting movements. In *Computer-Based Medical Systems (CBMS), 2015 IEEE 28th International Symposium on* (pp. 171-176). IEEE.
7. Meireles, J. and Massano, J., 2012. Cognitive impairment and dementia in Parkinson’s disease: clinical features, diagnosis, and management. *Frontiers in neurology*, *3*, p.88.
8. Samii A., J.G. Nutt, B.R. Ransom. Parkinson’s disease. *Lancet*, 363 (2004), pp. 1783-1793.
9. Khan Academy. (2015). *Movement signs and symptoms of Parkinson's disease | NCLEX-RN | Khan Academy*. [online] Available at: https://www.youtube.com/watch?v=4qdD4Ny34cc [Accessed 11 Apr. 2018].
10. Litvan, I., Goldman, J.G., Tröster, A.I., Schmand, B.A., Weintraub, D., Petersen, R.C., Mollenhauer, B., Adler, C.H., Marder, K., Williams-Gray, C.H. and Aarsland, D., 2012. Diagnostic criteria for mild cognitive impairment in Parkinson's disease:

Movement Disorder Society Task Force guidelines. *Movement disorders*, *27*(3), pp.349-356.

11. Berg, D., Lang, A.E., Postuma, R.B., Maetzler, W., Deuschl, G., Gasser, T., Siderowf, A., Schapira, A.H., Oertel, W., Obeso, J.A. and Olanow, C.W., 2013. Changing the research criteria for the diagnosis of Parkinson's disease: obstacles and opportunities. *The Lancet Neurology*, *12*(5), pp.514-524.
12. R. Lafuente, J. M. Belda, J. Sanchez-Lacuesta, C. Soler, and J. Prat, "Design and test of neural networks and statistical classifiers in computer-aided movement analysis: a case study on gait analysis," 1997.
13. J. G. Barton and A. Lees, "An application of neural networks for distinguishing gait patterns on the basis of hip-knee joint angle diagrams," 1997.
14. R. Begg and J. Kamruzzaman, "Neural networks for detection and classification of walking pattern changes due to ageing," 2006.
15. I. Rustempasic and M. Can, "Diagnosis of Parkinson's Disease using Fuzzy C-Means Clustering and Pattern Recognition," *Southeast Europe Journal of Soft Computing*, vol. 2, no. 1, Mar. 2013, doi: 10.21533/scjournal. v2i1.44.
16. R. Geetha, R. Professor, & Head, and G. Sivagami, "Parkinson Disease Classification using Data Mining Algorithms," 2011.
17. R. Armañanzas, C. Bielza, K. R. Chaudhuri, P. Martinez-Martin, and P. Larrañaga, "Unveiling relevant non-motor Parkinson's disease severity symptoms using a machine learning approach," *Artificial Intelligence in Medicine*, vol. 58, no. 3, pp. 195–202, Jul. 2013, doi: 10.1016/j.artmed.2013.04.002.
18. A. Tsanas, M. A. Little, P. E. McSharry, J. Spielman, and L. O. Ramig, "Novel speech signal processing algorithms for high-accuracy classification of Parkinsons disease," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 5, pp. 1264–1271, May 2012, doi: 10.1109/TBME.2012.2183367.
19. J. Mei, C. Desrosiers, and J. Frasnelli, "Machine Learning for the Diagnosis of Parkinson's Disease: A Review of Literature," *Frontiers in Aging Neuroscience*, vol. 13. Frontiers Media S.A., May 06, 2021. doi: 10.3389/fnagi.2021.633752.
20. M. Shahbakhi, D. T. Far, and E. Tahami, "Speech Analysis for Diagnosis of Parkinson's Disease Using Genetic Algorithm and Support Vector Machine," *Journal of Biomedical Science and Engineering*, vol. 07, no. 04, pp. 147–156, 2014, doi: 10.4236/jbise.2014.74019.
21. Z. Karapinar Senturk, "Early diagnosis of Parkinson's disease using machine

learning algorithms," *Medical Hypotheses*, vol. 138, May 2020, doi: 10.1016/j.mehy.2020.109603.

22. R. Das, "A comparison of multiple classification methods for diagnosis of Parkinson disease," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1568–1572, Mar. 2010, doi: 10.1016/j.eswa.2009.06.040.
23. S. Bind, A. K. Tiwari, and A. K. Sahani, "A Survey of Machine Learning Based Approaches for Parkinson Disease Prediction." [Online]. Available: www.ijcsit.com
24. M. F. Caglar, B. Cetisli, and I. B. Toprak, "Automatic Recognition of Parkinson's Disease from Sustained Phonation Tests Using ANN and Adaptive Neuro-Fuzzy Classifier," 2010.
25. C. W. Cho, W. H. Chao, S. H. Lin, and Y. Y. Chen, "A vision-based analysis system for gait recognition in patients with Parkinson's disease," *Expert Systems with Applications*, vol. 36, no. 3 PART 2, pp. 7033–7039, 2009, doi: 10.1016/j.eswa.2008.08.076.
26. S. S. Iyengar, V. Saxena, IEEE Computer Society. Technical Committee on Parallel Processing, Institute of Electrical and Electronics Engineers, Jaypee Institute of Information Technology University, and University of Florida. College of Engineering, *2019 Twelfth International Conference on Contemporary Computing (IC3-2019): 8-10 August 2019, Jaypee Institute of Information Technology, Noida, India*.
27. S. R. Madhavan and S. S. holy, "Parkinson's Disease Detection using Machine Learning Techniques," vol. XXX, pp. 543–552, doi: 10.24205/03276716.2020.4055.
28. https://www.kaggle.com/datasets
29. Little, M.A., McSharry, P.E., Roberts, S.J. et al. Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection. BioMed Eng OnLine 6, 23 (2007). https://doi.org/10.1186/1475-925X-6-23
30. Diagnosis of Parkinson's Disease Using SVM Classifier G. wiselin Jiji, A. Rajesh and P. Johnson Durai Raj https://doi.org/10.1142/S021946782150011X
31. LR, LOR for the Diagnosis of Parkinson's Disease Jie Mei, Christian Desrosiers and Johannes Frasnelli. https://doi.org/10.3389/fnagi.2021.63375
32. Abiyev R. H., Abizade S. (2016). Diagnosing Parkinson's diseases using a fuzzy neural system. Comput. Mathe. Methods Med. 2016:1267919. 10.1155/2016/1267919 [PMC free article] [PubMed] [CrossRef] [Google Scholar]
33. Abos A., Baggio H. C., Segura B., Campabadal A., Uribe C., Giraldo D. M., et al.

(2019). Differentiation of multiple system atrophy from Parkinson's disease by structural connectivity derived from probabilistic tractography. Sci. Rep. 9:16488. 10.1038/s41598-019- 52829-8 [PMC free article] [PubMed] [CrossRef] [Google Scholar]

34. Abujrida H., Agu E., Pahlavan K. (2017). Smartphone-based gait assessment to infer Parkinson's disease severity using crowdsourced data, in 2017 IEEE Healthcare Innovations and Point of Care Technologies (HI-POCT) (Bethesda, MD:), 208–211. 10.1109/HIC.2017.8227621 [CrossRef] [Google Scholar]
35. Adams W. R. (2017). High-accuracy detection of early Parkinson's Disease using multiple characteristics of finger movement while typing. PLoS ONE 12: e0188226. 10.1371/journal.pone.0188226 [PMC free article] [PubMed] [CrossRef] [Google Scholar]
36. Adeli E., Shi F., An L., Wee C.-Y., Wu G., Wang T., et al. (2016). Joint feature-sample selection and robust diagnosis of Parkinson's disease from MRI data. NeuroImage 141, 206–219. 10.1016/j.neuroimage.2016.05.054 [PMC free article] [PubMed] [CrossRef] [Google Scholar]
37. Adeli E., Thung K.-H., An L., Wu G., Shi F., Wang T., et al. (2019). Semi-supervised discriminative classification robust to sample-outliers and feature-noises. IEEE Trans. Pattern Anal. Mach. Intell. 41, 515–522. 10.1109/TPAMI.2018.2794470 [PMC free article] [PubMed] [CrossRef] [Google Scholar]
38. Agarwal A., Chandrayan S., Sahu S. S. (2016). Prediction of Parkinson's disease using speech signal with Extreme Learning Machine, in 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) (Chennai:), 3776–3779. 10.1109/ICEEOT.2016.7755419 [CrossRef] [Google Scholar]
39. Ahlrichs C., Lawo M. (2013). Parkinson's disease motor symptoms in machine learning: a review. arXiv preprint arXiv:1312.3825. 10.5121/hiij.2013.2401 [CrossRef] [Google Scholar]
40. Ahmadi S. A., Vivar G., Frei J., Nowoshilow S., Bardins S., Brandt T., et al. (2019). Towards computerized diagnosis of neurological stance disorders: data mining and machine learning of posturography and sway. J. Neurol. 266(Suppl 1), 108–117. 10.1007/s00415-019-09458-y [PubMed] [CrossRef] [Google Scholar]
41. Aich S., Kim H., younga K., Hui K. L., Al-Absi A. A., Sain M. (2019). A supervised machine learning approach using different feature selection techniques on voice

datasets for prediction of Parkinson's disease, in 2019 21st International Conference on Advanced Communication Technology (ICACT) (PyeongChang:), 1116–1121. 10.23919/ICACT.2019.8701961 [CrossRef] [Google Scholar]

42. Alam M. N., Garg A., Munia T. T. K., Fazel-Rezai R., Tavakolian K. (2017). Vertical ground reaction force marker for Parkinson's disease. PLoS ONE 12: e0175951. 10.1371/journal.pone.0175951 [PMC free article] [PubMed] [CrossRef] [Google Scholar]
43. Alaskar H., Hussain A. (2018). Prediction of Parkinson disease using gait signals, in 2018 11th International Conference on Developments in eSystems Engineering (DeSE) (Cambridge:), 23–26. 10.1109/DeSE.2018.00011 [CrossRef] [Google Scholar]
44. Al-Fatlawi A. H., Jabardi M. H., Ling S. H. (2016). Efficient diagnosis system for Parkinson's disease using deep belief network, in 2016 IEEE Congress on Evolutionary Computation (CEC) (Vancouver, BC:), 1324–1330. 10.1109/CEC.2016.7743941 [CrossRef] [Google Scholar]
45. Alharthi A. S., Ozanyan K. B. (2019). Deep learning for ground reaction force data analysis: application to wide-area floor sensing, in 2019 IEEE 28th International Symposium on Industrial Electronics (ISIE) (Vancouver, BC:),, 1401–1406. 10.1109/ISIE.2019.8781511 [CrossRef] [Google Scholar]
46. Ali L., Khan S. U., Arshad M., Ali S., Anwar M. (2019a). A multi-model framework for evaluating type of speech samples having complementary information about Parkinson's disease, in 2019 International Conference on Electrical, Communication, and Computer Engineering (ICECCE) (Swat:), 1–5. 10.1109/ICECCE47252.2019.8940696 [CrossRef] [Google Scholar]