

ABSTRACT

The Parkinson's disease is progressive neuro degenerative disorder that affects a lot only people significantly affecting their quality of life. It mostly affects the motor functions of human. The main motor symptoms are called "parkinsonism" or "parkinsonian syndrome". The symptoms of Parkinson's disease will occur slowly, the symptoms include shaking, rigidity, slowness of movement and difficulty with walking, Thinking and behaviour change, Depression and anxiety are also common. There is a model for detecting Parkinson's using voice. The deflections in the voice will confirm the symptoms of Parkinson's disease. This project showed 73.8% efficiency. In our model, a huge amount of data is collected from the normal person and also previously affected person by Parkinson's disease. these data is trained using machine learning algorithms. From the whole data 60% is used for training and 40% is used for testing. The data of any person can be entered in db to check whether the person is affected by Parkinsons disease or not. There are 24 columns in the data set each column will indicate the symptom values of a patient except the status column. The status column has 0's and 1's. Those values will decide the person is affected with Parkinsons disease. 1's indicate person is affected, 0's indicate normal conditions.

Key Words: Parkinson's disease; machine learning (ML), XGBoost, Decision tree.

1. INTRODUCTION

The recent report of the World Health Organization shows a visible increase in the number and health burden of Parkinson's disease patients increases rapidly. In China, this disease is spreading so fast and estimated that it reaches half of the population in the next 10 years. Classification algorithms are mainly used in the medical field for classifying data into different categories according to the number of characteristics. Parkinson's disease is the second most dangerous neurological disorder that can lead to shaking, shivering, stiffness, and difficulty walking and balance. It caused mainly due by the breaking down of cells in the nervous system. Parkinson's can have both motor and non-motor symptoms. The motor symptoms include slowness of movement, rigidity, balance problems, and tremors. If this disease continues, the patients may have difficulty walking and talking.

The non-motor symptoms include anxiety, breathing problems, depression, loss of smell, and change in speech. If the above-mentioned symptoms are present in the person, then the details are stored in the records. In this paper, the author considers the speech features of the patient, and this data is used for predicting whether the patient has Parkinson's disease or not. Neurodegenerative disorders are the results of progressive tearing and neuron loss in different areas of the nervous system. Neurons are functional units of the brain. They are contiguous rather than continuous.

A good healthy-looking neuron as shown in fig 1 has extensions called dendrites or axons, a cell body, and a nucleus that contains our DNA. DNA is our genome and a hundred billion neurons contain our entire genome which is packaged into it. When a neuron gets sick, it loses its extension and hence its ability to communicate which is not good for it and its metabolism becomes low so it starts to accumulate junk and it tries to contain the junk in the little packages in little pockets. When things become worse and if the neuron is a cell culture it completely loses its extension, becomes round and full of vacuoles.

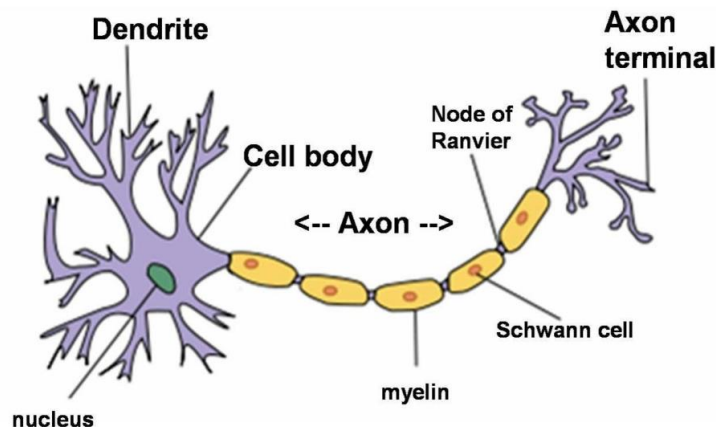


Fig:1.1 Structure of Neuron

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

- **Environmental factors:** Environment is defined as the surroundings or the place in which an individual lives. So, the environment is the major factor that will not only affects the human's brain but also affects all the living organism who lives in the vicinity of it. Many types of research and evidence have proved that the environment has a big hand in the development of neurodegenerative disorders mainly Alzheimer's and Parkinson's. There are certain environmental factors that are influencing neurodegenerative disorder with high pace are:
 - Exposure to heavy metals (like lead and aluminium) and pesticides.
 - Air Quality: Pollution results in respiratory diseases.
 - Water quality: Biotic and Abiotic contaminants present in water lead to water pollution.
 - Unhealthy lifestyle: It leads to obesity and a sedentary lifestyle.

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

1.1 Parkinson's Disease Symptoms

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

- **Motor symptoms:** This is a symptom where any voluntary action involved. It indicates the movement-related disorders like tremors, rigidity, freezing, Bradykinesia or any voluntary muscle movement.
- **Non-Motor symptoms:** Non motor symptoms include disorders of mood and affect with apathy, cognitive dysfunction as well as complex behavioural

disorders. There are two other categories of PD which are divided by doctors: Primary symptom and Secondary symptom.

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

1.2 Introduction to Machine Learning

Machine Learning may be a sub-area of AI, whereby the term refers to the power of IT systems to independently find solutions to problems by recognizing patterns in databases. In other words: Machine Learning enables IT systems to acknowledge patterns in the idea of existing algorithms and data sets and to develop adequate solution concepts. Therefore, in Machine Learning, artificial knowledge is generated on the idea of experience. In order to enable the software to independently generate solutions, the prior action of people is important. For example, the required algorithms and data must be fed into the systems in advance and the respective analysis rules for the recognition of patterns in the data stock must be defined. Once these two steps have been completed, the system can perform the following tasks by Machine Learning:

- Finding, extracting and summarizing relevant data
- Making predictions based on the analysis data
- Calculating probabilities for specific results Basically, algorithms play a crucial role in Machine Learning: On the one hand, they're liable for recognizing patterns and on the opposite hand, they will generate solutions. Algorithms can be divided into different categories:

1.2.1 Supervised learning

In the course of monitored learning, example models are defined beforehand. So as to make sure an adequate allocation of the knowledge to the respective model groups of the algorithms, these then need to be specified. In other words, the system learns on the idea of given input and output pairs. within the course of monitored learning, a programmer, who acts as a sort of teacher, provides the acceptable values for specific input. The aim is to coach the system within the context of successive calculations with different inputs and outputs to determine connections. Supervised learning is where you've got input variables (X) and an output variable (Y) and you employ an algorithm to find out the mapping function from the input to the output. $Y = f(X)$ The goal is to approximate the mapping function so well that once you have a new input file (X) that you simply can predict the output variables (Y) for that data. It's called supervised learning because the method of an algorithm learning from the training dataset is often thought of as an educator supervising the training process. We all know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected. Learning stops when the algorithm achieves a suitable level of performance.

Techniques of Supervised Machine Learning algorithms include linear and logistic regression, multi-class classification, Decision Tree, and Support Vector Machine. Supervised Learning problems are a kind of machine learning technique often further grouped into Regression and Classification problems. The difference between these two is that the dependent attribute is numerical for regression and categorical for classification:

- **Regression:** Linear regression could also be a linear model, e.g., a model that assumes a linear relationship between the input variables (x) and thus the only output variable (y). More specifically, that y is usually calculated from a linear combination of the input variables (x). When there's one input variable (x), the tactic is mentioned as simple linear regression. When there are multiple input variables, literature from statistics often refers to the tactic as multiple linear regression.
- **Classification:** Classification could also be a process of categorizing a given set of data into classes, it is often performed on both structured or unstructured data. the tactic starts with predicting the category of given data points. The classes are

often mentioned as target, label, or categories. In short, classification either predicts categorical class labels or classification data supported the training set and thus the values (class labels) in classifying attributes and uses it in classifying new data. There is a variety of classification models. Classification models include Logistic Regression, Decision Tree, Random Forest, Gradient Boosted Tree, One-vs.-One, and Naïve Bayes.

1.2.2 Unsupervised learning

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

Unsupervised Learning is accessed into two categories of algorithms:

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

1.2.3 Applications of Machine Learning

Virtual Personal Assistants: Siri, Alexa, Google Now are a number of the favored samples of virtual personal assistants. As the name suggests, they assist find information, when asked over voice. Machine learning is a crucial a part of these personal assistants as they collect and refine the knowledge on the idea of your previous involvement with them.

Later, this set of knowledge is employed to render results that are tailored to your preferences.

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

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

Videos Surveillance

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

Social Media Services

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

- People You May Know
- Face Recognition

Search Engine Result Refining

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

requirement. This way, the algorithms performing at the backend improve the search results.

1.3 Contributions

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

1.4 Problem Statement

The main aim is to detect the efficiency that would be beneficial for the patients who are suffering from Parkinson and the percentage of the disease will be reduced. Generally, in the first stage, Parkinson's can be cured by the proper treatment. So, it's important to identify the PD at the early stage for the betterment of the patients. The main purpose of this research work is to find the best prediction model i.e., the best machine learning technique which will distinguish the Parkinson's patient from the healthy person. The techniques used in this problem are KNN, Naïve Bayes, and Logistic Regression. The experimental study is performed on the voice dataset of Parkinson's patients which is downloaded from the Kaggle. The prediction is evaluated using evaluation metrics like confusion matrix, precision, recall accuracy, and f1-score. The author used feature selection where the important features are taken into consideration to detect Parkinson's.

2. LITERATURE SURVEY

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

Anila M and Dr G Pradeepini proposed the paper titled “Diagnosis of Parkinson’s disease using Artificial Neural network”. The main objective of this paper is that the detection of the disease is performed by using the voice analysis of the people affected with Parkinson's disease. For this purpose, various machine learning techniques like ANN, Random Forest, KNN, SVM, XG Boost are used to classify the best model, error rates are calculated, and the performance metrics are evaluated for all the models used. The main drawback of this paper is that it is limited to ANN with only two hidden layers. And this type of neural networks with two hidden layers are sufficient and efficient for simple datasets. They used only one technique for feature selection which reduces the number of features.

Arvind Kumar Tiwari Proposed the paper titled “Machine Learning-based Approaches for Prediction of Parkinson’s Disease”. In this paper, minimum redundancy maximum relevance feature selection algorithms were used to select the most important feature among all the features to predict Parkinson diseases. Here, it was observed that the random forest with 20 number of features selected by minimum redundancy maximum relevance feature selection algorithms provide the overall accuracy 90.3%, precision 90.2%, Mathews correlation coefficient values of 0.73 and ROC values 0.96 which is better in comparison to all other machine learning based approaches such as bagging, boosting, random forest, rotation forest, random subspace, support vector machine, multilayer perceptron, and decision tree based methods.

Mohamad Alissa Proposed the paper titled “Parkinson’s Disease Diagnosis Using Deep Learning”. This project mainly aims to automate the PD diagnosis process using deep learning, Recursive Neural Networks (RNN) and Convolutional Neural Networks (CNN), to differentiate between healthy and PD patients. Besides that, since different datasets may capture different aspects of this disease, this project aims to explore which PD test is more effective in the discrimination process by analysing different imaging and movement datasets (notably cube and spiral pentagon datasets). In general, the main aim of this paper is to automate the PD diagnosis process in order to discover this disease as early as possible. If we discover this disease earlier, then the treatments are more likely to improve the quality of life of the patients and their families.

There are some limitations to this paper namely:

- They used the validation set only to investigate the model performance during the training and this reduced the number of samples in the training set.
- RNN training is too slow and this is not flexible in practice work.
- Disconnecting and resource exhaustion: working with cloud services like Google Collaboratory causes many problems like disconnecting suddenly. And because it is shareable service by the world zones, this leads to resource exhaustion error many times.

Afzal Hussain Shahid and Maheshwari Prasad Singh proposed the paper titled “A deep learning approach for prediction of Parkinson’s disease progression”. This paper proposed a deep neural network (DNN) model using the reduced input feature space of Parkinson’s telemonitoring dataset to predict Parkinson’s disease (PD) progression and also proposed a PCA based DNN model for the prediction of Motor-UPDRS and Total-UPDRS in Parkinson's Disease progression. The DNN model was evaluated on a real-world PD dataset taken from UCI. Being a DNN model, the performance of the proposed model may improve with the addition of more data points in the datasets.

T. J. Wroge, Y. Özkanca, C. Demiroglu, D. Si, D. C. Atkins and R. H. Ghomi, proposed the paper titled “Parkinson’s Disease Diagnosis Using Machine Learning and Voice” is that it explores the effectiveness of using supervised classification algorithms, such as deep neural networks, to accurately diagnose individuals with the disease. Historically, PD has been difficult to quantify and doctors have tended to focus on some symptoms

while ignoring others, relying primarily on subjective rating scales. The analysis of this paper provides a comparison of the effectiveness of various machine learning classifiers in disease diagnosis with noisy and high dimensional data. Their peak accuracy of 85% provided by the machine learning models exceeds the average clinical diagnosis accuracy of non-experts (73.8%) and average accuracy of movement disorder specialists (79.6%) without follow-up, 83.9% after follow-up) with pathological post-mortem examination as ground truth.

Siva Sankara Reddy Donthi Reddy and Udaya Kumar Ramanadham proposed the paper “Prediction of Parkinson’s Disease at Early Stage using Big Data Analytics”. This paper describes mainly various Big Data Analytical techniques that may be used in diagnosing of right disease in the right time. The main intention is to verify the accuracy of prediction algorithms. Their future study aims to propose an efficient method to diagnose this type of neurological disorder by some symptoms at the early stage with better accuracy using different Big Data Analytical techniques like Hadoop, Hive, R Programming, MapReduce, PIG, Zookeeper, HBase, Cassandra, Mahout etc...

Daiga Heisters proposed the paper titled “Parkinson’s: symptoms, treatments and research”. This paper initially says that Current treatments can help to ease the symptoms but none can repair the damage in the brain or slow the progress of the condition; now, Parkinson’s UK researchers are working to develop new treatments that can and finally worked together to build on existing discoveries and explore these innovative areas of research, it is hoped that a cure for Parkinson’s will be found. Parkinson’s UK offers support for everyone affected, including people with the condition, their family, friends and careers, researcherand professionals working in this area.

T. Swapna, Y. Sravani Devi proposed a paper and titled “Performance Analysis of Classification algorithms on Parkinson’s Dataset with Voice Attributes”. This paper deals with the application of seven classification algorithms on the acquired data set and then drawing out a comparison of the results to one another and also predicting the outcome whether the person is healthy or Parkinson disease effected from the given data. The results of the selected algorithms namely Naïve Bayes, Random Forest, Neural Networks, Decision Trees, AdaBoost, SVM, KNN were compared and tabulated. According to the outputs derived with the help of python, implementing Scikit Libraries. Final accuracy was calculated using these parameters. Random Forest algorithm gives with optimum

accuracy of 78.56% which is closely followed by Decision Tree Algorithm with the optimal accuracy of 77.63%. Following the Decision Tree Algorithm is the MLP Classifier with an optimal accuracy of 76.72%, and lastly the Naïve Bayes Algorithm which has the optimal accuracy of 70.82%. Finally, these algorithms can help in classifying whether a person get effected with Parkinson's disease or not.

M. Abdar and M. Zomorodi-Moghadam proposed a paper "Impact of Patients' Gender on Parkinson's disease using Classification Algorithms". In this paper, the author chooses the UCI PD dataset for finding the accuracy of Parkinson's using SVM and Bayesian Network algorithms. The author chooses the most ten important features in the dataset to predict PD. The output variable is Sex and other factors are input, the author provides an approach for finding relationships between genders. The result obtained is SVM algorithm gives better performance than Bayesian Network with 90.98% accuracy.

Dragana Miljkovic, et al, proposed a paper "Machine Learning and Data Mining Methods for Managing Parkinson's Disease". In this paper, the author concluded that based on the medical tests taken by the patients the Predictor part was able to predict the 15 different Parkinson's symptoms separately. The machine learning and data mining techniques are applied on different symptoms separately and gives an accuracy range between 57.1% and 77.4% where tremor detection has the highest accuracy.

Sriram, T. V., et al. proposed a paper "Intelligent Parkinson Disease Prediction Using Machine Learning Algorithms". In this paper, the author used voice measures of the patients to check whether the patient has Parkinson's or not. The author applied the dataset to various machine learning algorithms and find the maximum accuracy. To analyse the models the author used the ROC curve and sieve graph. The random forest results with more accuracy i.e., are 90.26%.

Dr. R. GeethaRamani, G. Sivagami, and Shomona Graciajacob proposed a paper "Feature Relevance Analysis and Classification of Parkinson's Disease Tele Monitoring Data Through Data Mining". In this paper, the author used thirteen classification algorithms to diagnose the disease. The author used the Tele-monitoring dataset which contains 16 biomedical voice features for evaluating the system. The aim of this paper is to predict motor UPDRS and total UPDRS from the voice measures.

A.Ozcift, proposed a paper "SVM feature selection-based rotation forest ensemble classifiers to improve computer-aided diagnosis of Parkinson disease". In this paper, the

author summarizes that improve the PD diagnosis accuracy with the use of support vector machine feature selection. To evaluate the performances the author used accuracy, kappa statistics, and area under the curve of the classification algorithms. The rotation Forest ensemble of these classifiers used to increase the performance of the system.

3. SYSTEM ANALYSIS

3.1 Existing System

In existing system, we use unsupervised learning, AI learns without predefined target values and without rewards. It's mainly used for learning segmentation (clustering). The machine tries to structure and type the info entered consistent with certain characteristics. For instance, a machine could (very simply) learn that coins of various colors are often sorted consistent with the characteristic "color" to structure them. Unsupervised Machine Learning algorithms are used when the knowledge used to train is neither classified nor labeled. The system doesn't determine the right output but it explores the data and should draw inferences from datasets to elucidate hidden structures from unlabeled data. Unsupervised Learning is that the training of Machines using information that's neither classified nor labeled and allowing the algorithm to act thereon information without guidance. Existing system handles only structured data. In current past, countless disease estimate classifications have been advanced and in procedure. The standing organizations arrange a blend of machine learning algorithms which are judiciously exact in envisaging diseases. First, the prevailing systems are dearer only rich people could pay for to such calculation systems. And also, when it comes to folks, it becomes even higher. Second, the guess systems are non-specific and indefinite so far. So that, a machine can envisage a positive disease but cannot expect the sub types of the diseases and diseases caused by the existence of one bug.

3.1.1 Disadvantages

- More man power.
- Time Consuming.
- Consumes large volume of paper work.

3.2 Proposed system

Machine learning has given computer systems the ability to automatically learn without being explicitly programmed. In this we used supervised learning algorithm (Navie Bayes). Our application will be at affordable cost. Map Reduce Algorithm is implemented to increase operational efficiency. Accuracy is improved using Machine

Learning algorithm. The proposed system begins with the thought that was not executed by the ancestors. Its gadget Decision Tree machine learning procedure for calculating diseases as well as calculating all the other thinkable sub diseases. Its member Map Reduce algorithm for subdividing the data such that a request would be scrutinized only in the explicit partition, which will increase effective proficiency but cut query rescue time. In tally to that, it provides definite rations for specific clients to pattern his/her condition. Thus, making our presentation broadly open by all at cheap cost.

3.2.1 Advantages

- They judge the software supported Responsiveness, Usability, Security, Portability, and other non-functional standards that are critical to the success of the software.
- Accuracy
- Reliability
- Flexibility

3.3 Modules

Let us discuss about the various modules in our proposed system and what each module contributes in achieving our goal.

3.3.1 Speech Dataset

The main aim of this step is to spot and acquire all data-related problems. during this step, we'd like to spot the various data sources, as data are often collected from various sources like files and databases. The number and quality of the collected data will determine the efficiency of the output. The more are going to be the info, the more accurate are going to be the prediction. We've collected our data from the Kaggle website.

FileHOMEINSERTPAGE LAYOUTFORMULASDATAVIEWVIEW

Calibri11A⁺

</

Fig:1.2 Sample of acquired speech dataset from Kaggle

In the above Fig 1.2, we can see the speech dataset that has collected from kaggle website. This acquired dataset has around 756 patient's data and each row has 755 different voice features. But in this paper, we chosen 10 main features that required to find the prediction. The features are listed below:

- Id
- Gender
- PPE
- (Pitch Period Entropy)
- DFA (Detrended Fluctuation Analysis)
- RPDE (Recurrent Period Density Entropy)
- Bradykinesia
- Tremor
- Rigidity
- Postural deformities
- Postural instability
- Cognitive and neuro-behavioural abnormalities
- Sleep disorders

```

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score

# loading the data from csv file to a Pandas DataFrame
parkinsons_data = pd.read_csv('/archive.csv (1).zip')

# printing the first 5 rows of the dataframe
parkinsons_data.head()

```

	name	MDVP:F0(Hz)	MDVP:Fh1(Hz)	MDVP:F1o(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shimmer:DDA	NHR	HNR	status
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374	...	0.06545	0.02211	21.033	1 0.4
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134	...	0.09403	0.01929	19.085	1 0.4
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233	...	0.08270	0.01309	20.651	1 0.4
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492	...	0.08771	0.01353	20.644	1 0.4
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425	...	0.10470	0.01767	19.649	1 0.4

Fig:1.3 Reading the dataset from the CSV file into notebook

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

3.3.2 Data Pre-Processing

The main aim of this step is to study and understand the nature of data that was acquired in the previous step and also to know the quality of data. A real-world data generally contains noises, missing values, and maybe in an unusable format that cannot be directly used for machine learning models. Data pre-processing is a required task for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model. Identifying duplicates in the dataset and removing them is also done in this step. Actually, in this dataset, we have 755 features out of which some may not be useful in building our model. So, we must leave out all those unnecessary features which are not responsible to produce the output. If we take more features in this model the accuracy, we got is less. When we check the correlation of the features, some of them are the same. As our data is now stored as a data frame in a python notebook, we can easily drop those unnecessary features using the inbuilt functions.

```
[ ] # printing the first 5 rows of the dataframe
parkinsons_data.head()
```

	name	MDVP:F0(Hz)	MDVP:F1(Hz)	MDVP:F2(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	...	Shimmer:DDA	NHR	HNR	status
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374	...	0.06545	0.02211	21.033	1 0.4
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134	...	0.09403	0.01929	19.085	1 0.4
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233	...	0.08270	0.01309	20.651	1 0.4
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492	...	0.08771	0.01353	20.644	1 0.4
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425	...	0.10470	0.01767	19.649	1 0.4

5 rows x 24 columns

```
# getting more information about the dataset
parkinsons_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 24 columns):
#   Column              Non-Null Count  Dtype
---  -
0   name                 195 non-null    object
1   MDVP:F0(Hz)          195 non-null    float64
2   MDVP:F1(Hz)          195 non-null    float64
3   MDVP:F2(Hz)          195 non-null    float64
4   MDVP:Jitter(%)       195 non-null    float64
5   MDVP:Jitter(Abs)     195 non-null    float64
6   MDVP:RAP             195 non-null    float64
7   MDVP:PPQ             195 non-null    float64
8   Jitter:DDP           195 non-null    float64
9   MDVP:Shimmer         195 non-null    float64
10  MDVP:Shimmer(dB)     195 non-null    float64
11  Shimmer:APQ3         195 non-null    float64
12  Shimmer:APQ5         195 non-null    float64
13  MDVP:APQ             195 non-null    float64
14  Shimmer:DDA          195 non-null    float64
15  NHR                  195 non-null    float64
16  HNR                  195 non-null    float64
17  status               195 non-null    int64
18  RPDE                 195 non-null    float64
19  DFA                  195 non-null    float64
20  spread1              195 non-null    float64
21  spread2              195 non-null    float64
22  D2                   195 non-null    float64
23  PPE                  195 non-null    float64
dtypes: float64(22), int64(1), object(1)
memory usage: 36.7+ KB
```

Fig:1.4 Dropping unnecessary features from data frame

After identifying and dropping some features, the initial 755 features that we have are reduced to 10 features. The features are listed below:

- Id
- Gender
- PPE (Pitch Period Entropy)
- DFA (Detrended Fluctuation Analysis)
- RPDE (Recurrent Period Density Entropy)
- Bradykinesia
- Tremor
- Rigidity
- Postural deformities
- Cognitive and neuro-behavioural abnormalities
- Sleep disorders

After pre-processing the acquired data, the next step is to identify the best features. The identified best features should be able to give high efficiency. The classes within the sklearn Feature selection module are often used for feature selection/dimensionality

reduction on sample sets, either to enhance estimators’ accuracy scores or to spice up their performance on very high-dimensional datasets.

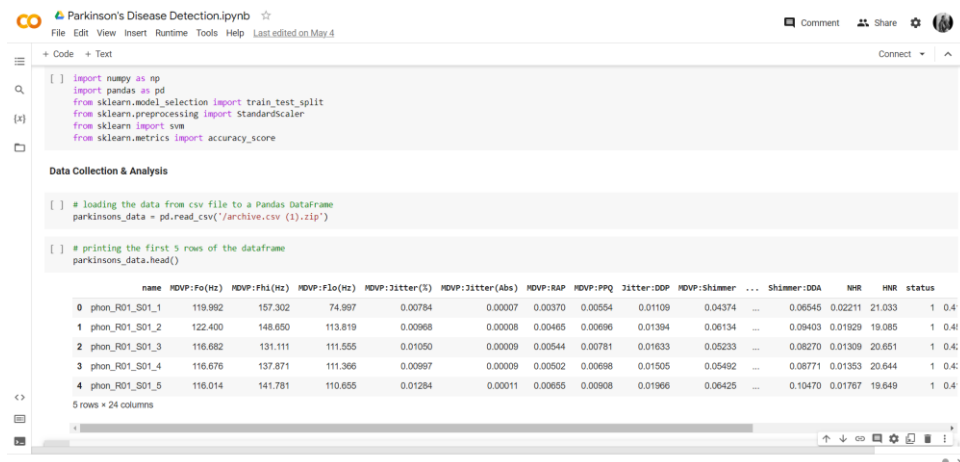


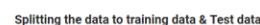
Fig:1.5 Process of Feature selection and sample data

3.3.3 Training data

Splitting the dataset into Training set and testing set:

- In machine learning data pre-processing, we must break our dataset into both training set and test set. This is often one among the crucial steps of knowledge pre-processing as by doing this, we will enhance the performance of our machine learning model.
- Suppose, if we've given training to our machine learning model by a dataset and that we test it by a totally different dataset. Then, it'll create difficulties for our model to know the correlations between the models.

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



```
[ ] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

```
[ ] print(X.shape, X_train.shape, X_test.shape)
```

(195, 22) (156, 22) (39, 22)

Fig:1.6 Splitting dataset into training data and test data

Usually, we split the dataset into train and test in the ratio of 7:3 i.e., 70 percent of data is used for training and 30 percent of data is used for testing the model. We have done it in the same way and it has been shown in the above Fig 1.6.

3.4.4 Testing Data Once Parkinson's disease

Detection model has been trained on the pre-processed dataset, then the model is tested using different data points. In this testing step, the model is checked for correctness and accuracy by providing a test dataset to it. All the training methods need to be verified for finding out the best model to be used. In above figures, after fitting our model with training data, we used this model to predict values for the test dataset. These predicted values on testing data are used for model comparison and accurate calculation.

4. FEASIBILITY STUDY

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

There are aspects within the feasibility study portion of the preliminary investigation:

- Economic Feasibility
- Technical Feasibility
- Operational Feasibility

4.3.1 Economic Feasibility

As system are often developed technically which are going to be used if installed must still be an honest investment for the organization. In the economic feasibility, the event cost in creating the system is evaluated against the last word benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It doesn't require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies java1.6 open source, there is nominal expenditure and economic feasibility for certain.

4.3.2 Technical Feasibility

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team can convert the ideas into working systems. Technical feasibility also involves evaluation of the hardware, software, and other technology requirements of the proposed system. This assessment is predicated on an overview design of system requirements, to work out whether the corporate has the technical expertise to handle completion of the project.

When writing a feasibility report, the subsequent should be taken to consideration: ·

- A brief description of the business to assess more possible factors which could affect the study

- The part of the business being examined
- The human and economic factor
- The possible solutions to the problem at this level, the concern is whether the proposal is both technically and legally feasible (assuming moderate cost).

The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system.

4.3.3 Operational Feasibility

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

Some of the important issues raised are to check the operational feasibility of a project includes the following:

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So, there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

5. SOFTWARE REQUIRIMENT SPECIFICATIONS

5.1 Hardware Requirements

Laptop / PC with 4 GB RAM
Processor – i3 or higher version, AMD 3 or higher version
Storage – 5 GB max
Internet Connection
1 GHz speed

Table:1.1 Hardware Requirements

5.2 Software Requirements

Programming Language	Python
Operating System	Windows 7, Windows 8 and higher versions, Linux, MacOS.
Interpreter	Web Application

Table:1.2 Software Requirements

5.3 Hardware interfaces

The solution involves extensive use of several hardware devices.

These devices include:

1. Internet modem
2. LAN
3. Windows/Linux/MacOS

5.4 Software interfaces

We are using python programming language for backend programming. Python uses NLTK library for natural language processing to perform the text summarization.

6. SYSTEM DESIGN

6.1 UML Concept

As UML describes the real-time systems, it is very important to make a conceptual model and then proceed gradually. The conceptual model of UML can be mastered by learning the following three major elements

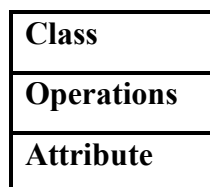
- UML building blocks
- Rules to connect the building blocks
- Common mechanisms of UML

6.2 Building blocks of UML

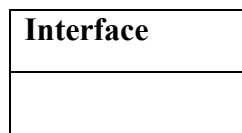
6.2.1 Things in the UML

Structural Things - Structural things define the static part of the model. They represent the physical and conceptual elements. Following are the brief descriptions of the structural things.

Class – Class represents a set of objects having similar responsibilities.



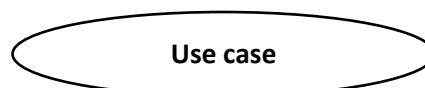
Interface – Interface defines a set of operations, which specify the responsibility of a class.



Collaboration – Collaboration defines an interaction between elements.



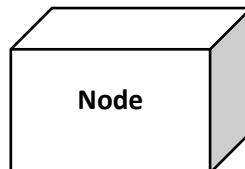
Use case – Use case represents a set of actions performed by a system for a specific goal.



Component –Component describes the physical part of a system.



Node – A node can be defined as a physical element that exists at run time.

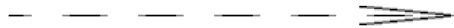


6.2.2 Relationships in UML

Relationship is another most important building block of UML. It shows how the elements are associated with each other and this association describes the functionality of an application. There are four kinds of relationships available.

Dependency

Dependency is a relationship between two things in which change in one element also affects the other.



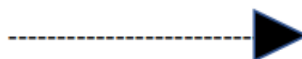
Association

Association is basically a set of links that connects the elements of a UML model. It also describes how many objects are taking part in that relationship



Generalization

Generalization can be defined as a relationship which connects a specialized element with a generalized element.



Realization

Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility, which is not implemented and the other one implements them. This relationship exists in case of interfaces.

6.2.3 UML diagrams

UML diagram is a diagram that is designed based on Unified Modeling language with the aim to visually represent the system with roles, actors, anchors etc to understand and maintain the system easily. By using this we can better understand flaws or errors in the system so that we can maintain or alter the system properly. Different types of UML diagrams include

1.7 Data Flow diagram

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.



Fig:1.7 Data Flow diagram

1.8 Use case diagram

It shows the set of use cases, actors & their relationships. In our project we have 2 actors as the user and admin interacting with the chatbot system. The use cases shown are asking query, which could be course related, admissions related, training the model and getting response.

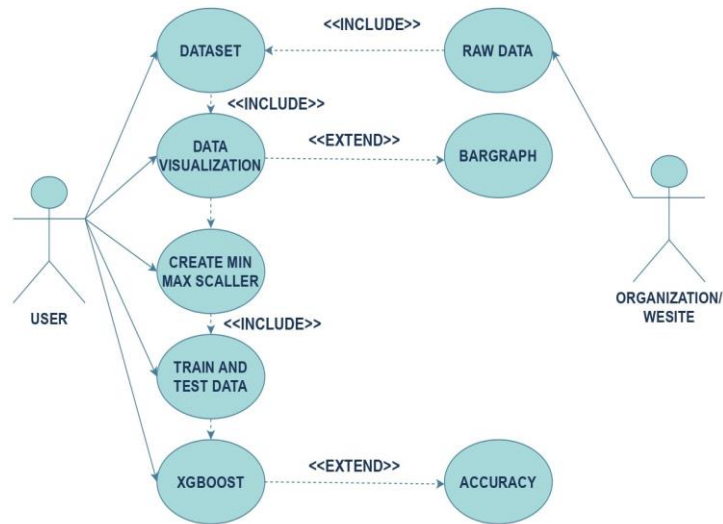


Fig:1.8 Use Case Diagram.

1.9 Class diagram

A class diagram shows a set of classes, interfaces, and collaborations and their relationships. These diagrams are the most common diagram found in modeling object-oriented systems. Class diagrams address the static design view of a system. Class diagrams that include active classes address the static process view of a system.

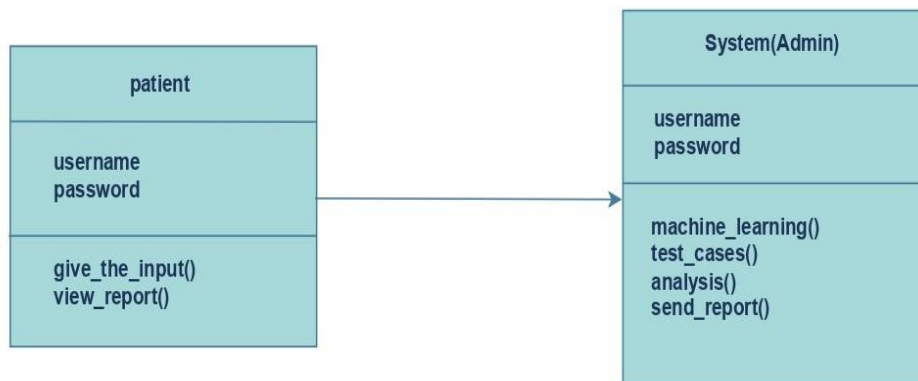


Fig:1.9 Class Diagram.

1.10 Sequence diagram

A sequence diagram is an interaction diagram that emphasizes the time-ordering of messages; a collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages.

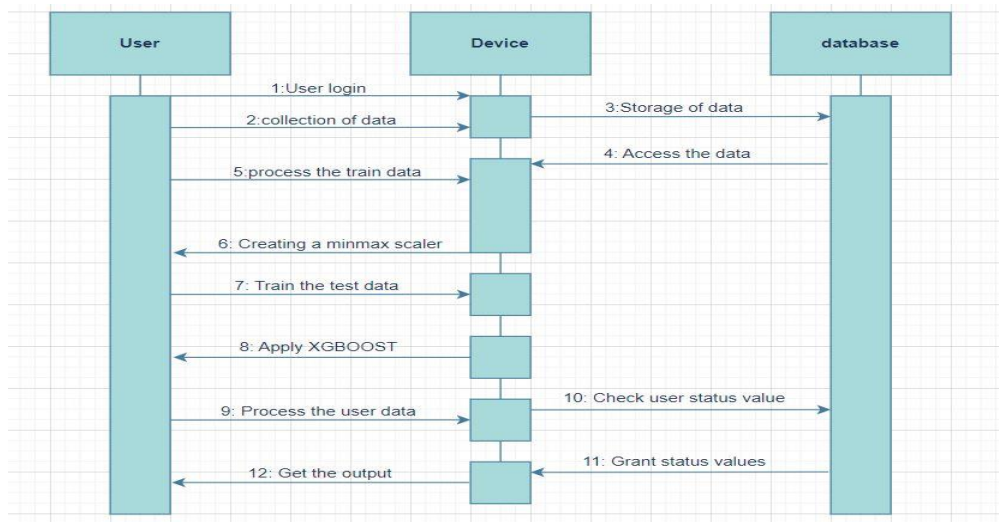


Fig:1.10 Sequence Diagram.

1.11 Collaboration diagram

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML).

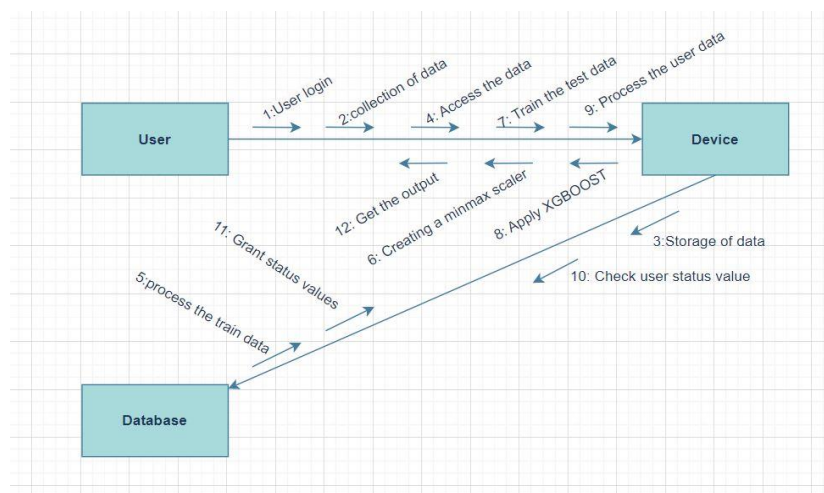


Fig:1.11 Collaboration Diagram.

1.12 Activity diagram

An activity diagram is a special kind of a state chart diagram that shows the flow from activity to activity within a system. Activity diagrams address the dynamic view of a system.

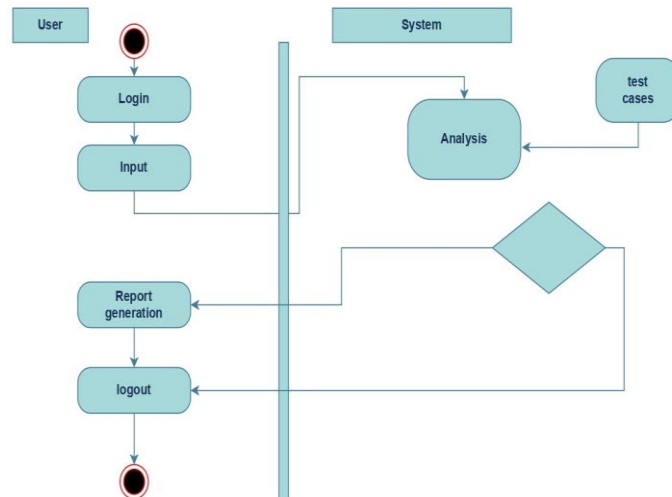


Fig:1.12 Activity Diagram.

1.13 State Chart diagram

A state diagram is an illustration of all the possible behavioral states a software system component may exhibit and the various state changes it's predicted to undergo over the course of its operations.

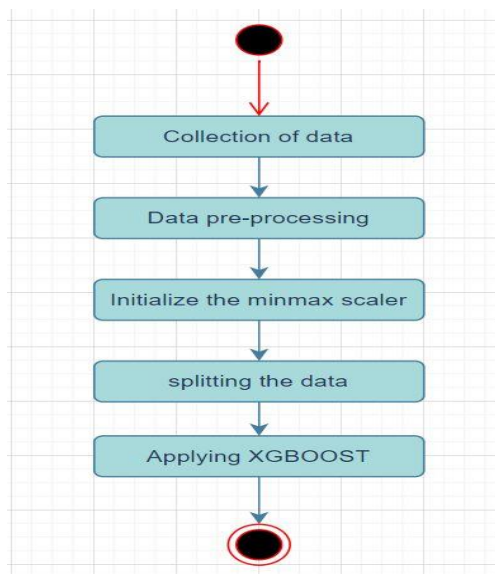


Fig:1.13 State Chart Diagram.

1.14 Component diagram

The purpose of a component diagram is to show the relationship between different components in a system.

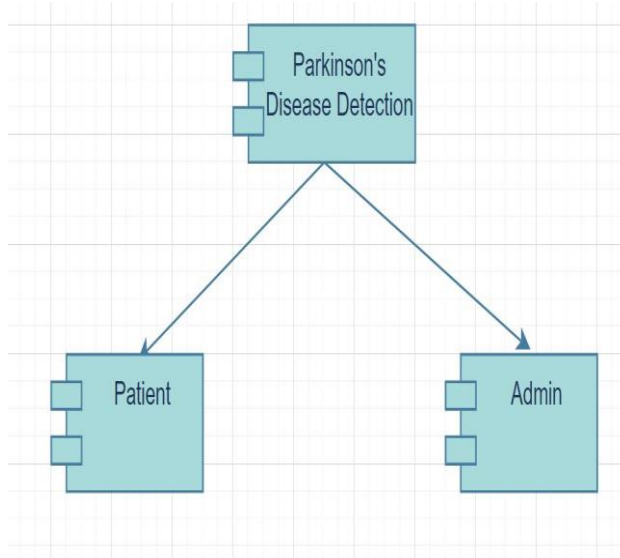


Fig:1.14 Component Diagram.

1.15 Deployment diagram

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them.

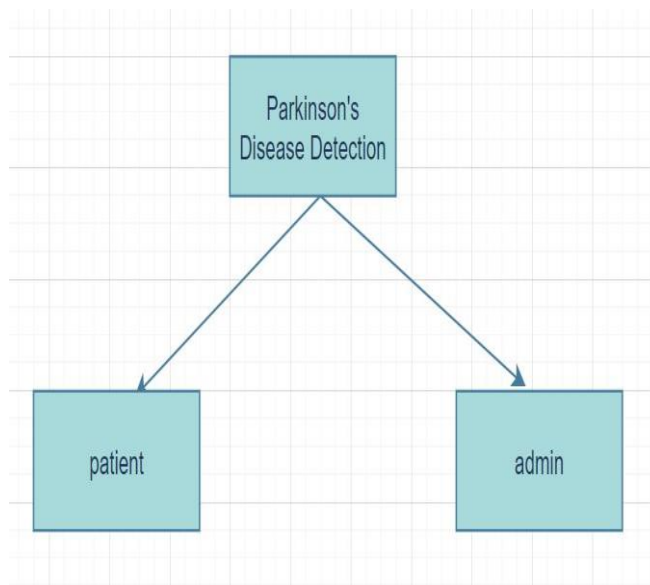


Fig:1.15 Component Diagram

7. SYSTEM IMPLEMENTATION

7.1 System Architecture

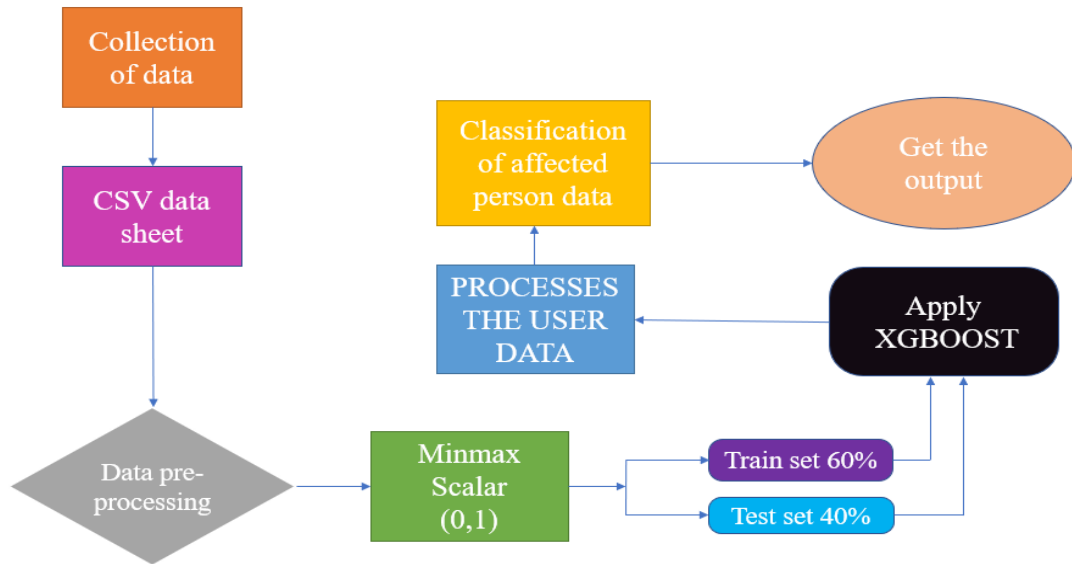


Fig:1.16 Parkinson's Disease Detection Architecture

7.2 Algorithm

7.2.1 Naïve Bayes

Naive Bayes may be a statistical classification technique supported Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier may be a fast, accurate, and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets. Naive Bayes classifier assumes that the effect of a specific feature during a class is independent of other features. For example, a loan applicant is desirable or not counting on his/her income, previous loan and transaction history, age, and site. Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered naive. This assumption is called class conditional independence.

Naive Bayes classifier makes an assumption that every feature in the dataset is independent of all other features. For example, a patient is having Parkinson's or not depends on the speech features of the patient.

$$P\left(\frac{h}{D}\right) = \frac{P\left(\frac{D}{h}\right) * P(h)}{P(D)} \longrightarrow (1)$$

$P(h)$: the probability of hypothesis h being true (regardless of the data). This is known as the prior probability of h .

$P(D)$: the probability of the data (regardless of the hypothesis). This is known as the prior probability.

$P(h|D)$: the probability of hypothesis h given the data D . This is known as posterior probability.

$P(D|h)$: the probability of data d given that the hypothesis h was true. This is known as posterior probability.

We can frame classification as a conditional classification problem with Bayes Theorem as follows: $P(y_i | x_1, x_2, \dots, x_n) = P(x_1, x_2, \dots, x_n | y_i) * P(y_i) / P(x_1, x_2, \dots, x_n)$.

The prior $P(y_i)$ is easy to estimate from a dataset, but the conditional probability of the observation based on the class $P(x_1, x_2, \dots, x_n | y_i)$ is not feasible unless the number of examples is extraordinarily large, e.g., large enough to effectively estimate the probability distribution for all different possible combinations of values. As such, the direct application of Bayes Theorem also becomes intractable, especially as the number of variables or features (n) increases.

Naive Bayes classifier calculates the probability of an event in the following steps:

Step 1: Calculate the prior probability for given class labels

Step 2: Find Likelihood probability with each attribute for each class

Step 3: Put this value in Bayes Formula and calculate posterior probability.

Step 4: See which class has a higher probability, given the input belongs to the higher probability class.

Types of Naive Bayes Algorithms:

- **Gaussian Naïve Bayes:** When the feature values are continuous in the nature then there is an assumption to be made that the values linked with each category are dispersed according to Gaussian that is Normal Distribution.
- **Multinomial Naïve Bayes:** Multinomial Naive Bayes is mostly favoured to be used on the data that is multinomial distributed. It is widely utilized in text classification in NLP. Each event in text classification constitutes the presence of a word in a document.

- **Bernoulli Naïve Bayes:** When data is deleted according to the multivariate Bernoulli distributions then came the Bernoulli Naïve Bayes. That means there can be exist a different number of features but each one is assumed to contain a binary value. So, it requires features to be binary-valued.
- In scikit-learn python library, from `sklearn.naive_bayes` import `GaussianNB` Module is used for carrying out the Naïve Bayes classifier. We will use our training dataset to fit the model. Fig 3.13 shows the sample code for training model using Naïve Bayes.

```

from sklearn.naive_bayes import GaussianNB
nb=GaussianNB()
nb.fit(x_train,y_train)
y_head=nb.predict(x_test)
print("Naive Bayes Algorithm test accuracy",nb.score(x_test,y_test))

```

Naive Bayes Algorithm test accuracy 0.8105726872246696

```

classid,tn,fp,fn,tp=perf_measure(y_test,y_head)
auc_scor.append(roc_auc_score(y_test,y_head))
score_list.append(accuracy(classid,tn,fp,fn,tp))
precision_scor.append(precision(classid,tn,fp,fn,tp))
recall_scor.append(recall(classid,tn,fp,fn,tp))
f1_scor.append(f1_score(y_test,y_head,average='macro'))
NPV_scor.append(NPV(classid,tn,fp,fn,tp))
specificity_scor.append(specificity(classid,tn,fp,fn,tp))
TPR=recall(classid,tn,fp,fn,tp)
TNR=specificity(classid,tn,fp,fn,tp)

print("Naive Bayes algorithm report: \n",classification_report(y_test,y_head))

```

Naive Bayes algorithm report:

	precision	recall	f1-score	support
0	0.54	0.15	0.23	48
1	0.81	0.97	0.88	179
accuracy			0.79	227
macro avg	0.67	0.56	0.55	227
weighted avg	0.75	0.79	0.74	227

Fig:1.17 Naïve Bayes Classifier Model

7.2.2 XGBoost

XGBoost is a new Machine Learning algorithm designed with speed and performance in mind. XGBoost stands for Extreme Gradient Boosting and is based on decision trees. XGBoost, which stands for Extreme Gradient Boosting, is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It's vital to an understanding of XGBoost to first grasp the machine learning concepts and algorithms that XGBoost builds upon: supervised machine learning, decision trees, ensemble learning, and gradient boosting. Supervised machine learning uses algorithms to train a

model to find patterns in a dataset with labels and features and then uses the trained model to predict the labels on a new dataset's features.

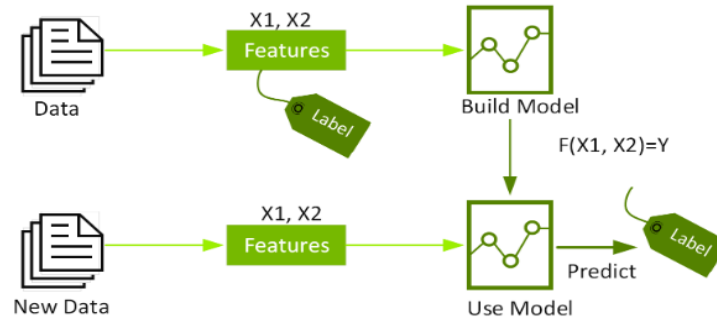


Fig:1.18 XGboost data

Decision trees create a model that predicts the label by evaluating a tree of if-then-else true/false feature questions, and estimating the minimum number of questions needed to assess the probability of making a correct decision. Decision trees can be used for classification to predict a category, or regression to predict a continuous numeric value. In the simple example below, a decision tree is used to estimate a house price (the label) based on the size and number of bedrooms (the features).

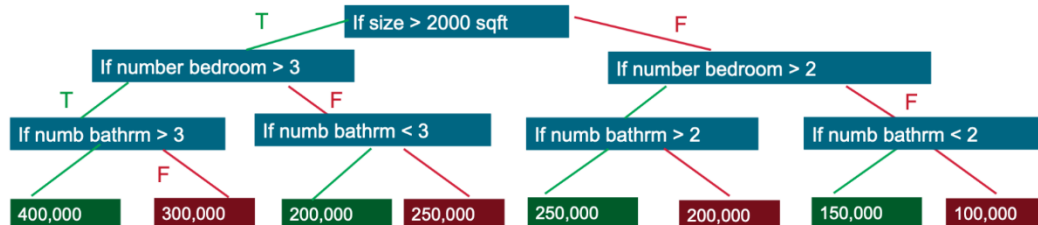


Fig:1.19 Decision tree

A Gradient Boosting Decision Trees (GBDT) is a decision tree ensemble learning algorithm like random forest, for classification and regression. Ensemble learning algorithms combine multiple machine learning algorithms to obtain a better model. Both random forest and GBDT build a model consisting of multiple decision trees. The difference is in how the trees are built and combined.

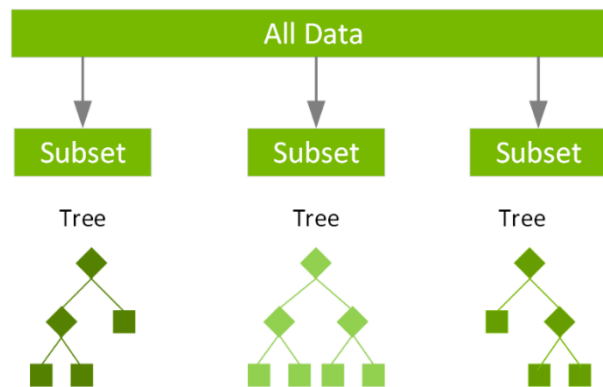


Fig:1.20 Gradient Boosting Decision Trees (GBDT)

The term “gradient boosting” comes from the idea of “boosting” or improving a single weak model by combining it with several other weak models to generate a collectively strong model. Gradient boosting is an extension of boosting where the process of additively generating weak models is formalized as a gradient descent algorithm over an objective function. Gradient boosting sets targeted outcomes for the next model to minimize errors. Targeted outcomes for each case are based on the gradient of the error (hence the name gradient boosting) with respect to the prediction.

GBDTs iteratively train an ensemble of shallow decision trees, with each iteration using the error residuals of the previous model to fit the next model. The final prediction is a weighted sum of all the tree predictions. Random forest “bagging” minimizes the variance and overfitting, while GBDT “boosting” minimizes the bias and underfitting.

XGBoost is a scalable and highly accurate implementation of gradient boosting that pushes the limits of computing power for boosted tree algorithms, being built largely for energizing machine learning model performance and computational speed. With XGBoost, trees are built in parallel, instead of sequentially like GBDT. It follows a level-wise strategy, scanning across gradient values and using these partial sums to evaluate the quality of splits at every possible split in the training set.

Why XGBoost?

XGBoost gained significant favor in the last few years as a result of helping individuals and teams win virtually every Kaggle structured data competition. In these competitions, companies and researchers post data after which statisticians and data miners compete to produce the best models for predicting and describing the data.

Initially both Python and R implementations of XGBoost were built. Owing to its popularity, today XGBoost has package implementations for Java, Scala, Julia, Perl, and other languages. These implementations have opened the XGBoost library to even more developers and improved its appeal throughout the Kaggle community.

XGBoost has been integrated with a wide variety of other tools and packages such as scikit-learn for Python enthusiasts and caret for R users. In addition, XGBoost is integrated with distributed processing frameworks like Apache Spark and Dask.

In 2019 XGBoost was named among InfoWorld's coveted Technology of the Year award winners.

XGBoost Benefits and Attributes

The list of benefits and attributes of XGBoost is extensive, and includes the following:

- A large and growing list of data scientists globally that are actively contributing to XGBoost open source development
- Usage on a wide range of applications, including solving problems in regression, classification, ranking, and user-defined prediction challenges
- A library that's highly portable and currently runs on OS X, Windows, and Linux platforms
- Cloud integration that supports AWS, Azure, Yarn clusters, and other ecosystems
- Active production use in multiple organizations across various vertical market areas
- A library that was built from the ground up to be efficient, flexible, and portable

XGBoost and Data Scientists

It's noteworthy for data scientists that XGBoost and XGBoost machine learning models have the premier combination of prediction performance and processing time compared with other algorithms. This has been borne out by various benchmarking studies and further explains its appeal to data scientists.

7.3 Sample Code

Importing the Dependencies

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score
```

Data Collection & Analysis

```
# loading the data from csv file to a Pandas DataFrame
parkinsons_data = pd.read_csv('/archive.csv (1).zip')
# printing the first 5 rows of the dataframe
parkinsons_data.head()
# number of rows and columns in the dataframe
parkinsons_data.shape
# getting more information about the dataset
parkinsons_data.info()
# checking for missing values in each column
parkinsons_data.isnull().sum()
# getting some statistical measures about the data
parkinsons_data.describe()
# distribution of target Variable
parkinsons_data['status'].value_counts()
```

1 --> Parkinson's Positive

0 --> Healthy

```
# grouping the data based on the target variable
parkinsons_data.groupby('status').mean()
```

Data Pre-Processing

Separating the features & Target

```
X = parkinsons_data.drop(columns=['name','status'], axis=1)
```

```
Y = parkinsons_data['status']
print(Y)
```

Splitting the data to training data & Test data

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
print(X.shape, X_train.shape, X_test.shape)
```

Data Standardization

```
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
print(X_train)
```

Model Training

Support Vector Machine Model

```
model = svm.SVC(kernel='linear')
# training the SVM model with training data
model.fit(X_train, Y_train)
```

Model Evaluation

Accuracy Score

```
# accuracy score on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
print('Accuracy score of training data : ', training_data_accuracy)
# accuracy score on training data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
print('Accuracy score of test data : ', test_data_accuracy)
```

Building a Predictive System

```
input_data =
(197.07600,206.89600,192.05500,0.00289,0.00001,0.00166,0.00168,0.00498,0.01098,0
```

```

.09700,0.00563,0.00680,0.00802,0.01689,0.00339,26.77500,0.422229,0.741367,-
7.348300,0.177551,1.743867,0.085569)
# changing input data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)
# reshape the numpy array
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
# standardize the data
std_data = scaler.transform(input_data_reshaped)
prediction = model.predict(std_data)
print(prediction)
if (prediction[0] == 0):
    print("The Person does not have Parkinsons Disease")
else;
    print("The Person has Parkinsons")

```

8. TESTING

8.1 System Testing

System testing, also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application. System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic. System testing is important because of the following reasons: System testing is the first step in the Software Development Life Cycle, where the application is tested. The application is tested in an environment that is very close to the production environment where the application will be deployed. System testing enables us to test, verify, and validate both the business requirements as well as the application architecture.

❖ System test case

SI# Test Case: ↯	STC-1
Name of Test: ↯	System testing in various versions of OS
Items being tested: ↯	OS compatibility
Sample Input: ↯	Execute the program in windows 8 and higher
Expected Output: ↯	Performance is better in windows-8
Actual Output: ↯	Performance is better in windows-10
Remarks: ↯	Pass

Table:1.3 Test case of STC

8.2 Unit Testing

Unit testing incorporates the arrangement of analysis that within program basis is working properly, and that program information sources produce significant yields. It checks whether little segments are working appropriately or not. Every single decision branch and inside code stream should be endorsed. It is the attempting of individual

programming units of the application. It is done after the completion of a unit before fuse. This is an auxiliary attempting, that relies upon learning of its improvement.

Unit tests perform fundamental tests at section level and test a specific business system, application, or possibly structure plan. Unit tests ensure that all a thoughtful method for a business technique performs unequivocally to the recorded points of interest and contains obviously portrayed data sources and foreseen results. A unit test encourages you to discover which part is broken in your application and fixes it quicker.

❖ Unit test cases

SI# Test Case: ↯	UTC-1
Name of Test: ↯	Admin accepts the user login.
Items being tested: ↯	Variables updated to the db.
Sample Input: ↯	Users Variables
Expected Output: ↯	Details stored in DataBase.
Actual Output: ↯	Stored and viewed by Admin.
Remarks: ↯	Pass

Table:1.4 Test case of UTC-1

SI# Test Case: ↯	UTC-1
Name of Test: ↯	User login.
Items being tested: ↯	Verifies variables
Sample Input: ↯	Characteristics
Expected Output: ↯	Matches for input variables
Actual Output: ↯	Detected variables
Remarks: ↯	Pass

Table:1.5 Test case of UTC-2

8.3 Integration Testing

Integration tests are expected to test joined programming modules to choose whether they everything considered continue running as one program. Testing is an event driven and is dynamically stressed over the crucial after effect of screens or fields. Combination tests show that in spite of the way that the sections were autonomously satisfied, as showed up by successfully unit testing, the gathering of portions are correct and unsurprising. Combination testing is expressly away for revealing the issues that rise up out of the gathering of these portions. Integration testing permits to discover blunders because of unexpected communication between the framework and the sub-framework segments. We test the product in order to test and to identify all the potential mistakes in our undertaking once we complete the source code and before conveying it to the clients. The techniques for performing tests.

These techniques provide guidance for testing:

- To test the internal logic of the software components.
- To test the input and output domains of a programs and to uncover the errors in program function, behaviour and performance. We can test the software by using two methods:

8.3.1 Black-box

Testing Black box testing is a software testing technique in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software.

- This type of testing is based entirely on the software requirements and specifications.
- In Black Box Testing we just focus on inputs and output of the software system. Types of Black Box Testing There are many types of Black Box Testing but following are the prominent ones -
- Functional testing – This black box testing type is related to functional requirements of a system; it is done by software testers.
- Non-functional testing – This type of black box testing is not related to testing of a specific functionality, but non-functional requirements such as performance, scalability, usability.

- Regression testing – Regression testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

8.3.2 White-Box Testing

These techniques analyse the internal structures the used data structures, internal design, code structure, and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing or clear box testing or structural testing. White Box Testing is also known as transparent testing, open box testing.

Working process of white box testing –

Input: Requirements, Functional specifications, design documents, source code.

Processing: Performing risk analysis for guiding through the entire process.

Proper test planning: Designing test cases to cover the entire code. Execute rinse repeat until error-free software is reached. Also, the results are communicated.

Output: Preparing final report of the entire testing process.

❖ Integration Test cases

SI# Test Case: ▯	ITC-1
Name of Test: ▯	Receives variables.
Items being tested: ▯	Variables of user.
Sample Input: ▯	Variables
Expected Output: ▯	Details stored in db
Actual Output: ▯	Variables Stored in db
Remarks: ▯	Pass

Table:1.6 Test case of ITC-1

SI# Test Case: ↯	ITC-2
Name of Test: ↯	User variables
Items being tested: ↯	Verifies variables
Sample Input: ↯	Variables & characteristics
Expected Output: ↯	Details stored in db
Actual Output: ↯	Variables Stored in db
Remarks: ↯	Pass

Table:1.7 Test case of ITC-2

8.4 Functional Testing

Functional testing may be a sort of software testing that validates the software against the functional requirements/specifications. This testing is detecting Parkinson's will based on machine learning algorithm. ML algorithm will boost up the speed. Typically, functional testing involves the following steps:

- Identifying the functions of that the software is expected to perform.
- Create input-data based on the function's specifications.
- It Determines the output based up on the function's specifications.
- Execute the test case.
- Compare the actual and expected outputs.

8.5 Verification and Validation

Testing procedure is a piece of subject alluding to checking and approval of our task. We must find the framework determinations and we should attempt to meet the details of the client and to fulfil the client, for this reason, we need to check and approve the item and we must ensure that everything is working appropriately. Check and approval are the two unique things. One is performed to guarantee that the product is working accurately and to implement a particular usefulness and the other is done to guarantee if the client prerequisites are appropriately met or not by the finished result.

9. OUTPUT SCREENSHORTS

9.1 Importing the Dependencies

```
[1] import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.metrics import accuracy_score
```

9.2 Data Collection & Analysis

Data Collection & Analysis

```
[2] # loading the data from csv file to a Pandas DataFrame
parkinsons_data = pd.read_csv('/content/archive.csv.zip')

# printing the first 5 rows of the dataframe
parkinsons_data.head()
```

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

5 rows x 24 columns

```
[4] # number of rows and columns in the dataframe
parkinsons_data.shape

(195, 24)
```

9.3 Grouping the data based on status

```
[9] # grouping the data based on the target variable
parkinsons_data.groupby('status').mean()
```

<ipython-input-9-fe279e5566c>:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Either pass numeric_only=False or define only the columns to aggregate on.

```
parkinsons_data.groupby('status').mean()
```

	MDVP:F0(Hz)	MDVP:F1(Hz)	MDVP:F1o(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer	MDVP:Shimmer(db)	...	MDVP:APQ	Shimmer:DDA	NHR
status														
0	181.937771	223.636750	145.207292	0.003866	0.000023	0.001925	0.002056	0.005776	0.017615	0.162958	...	0.013305	0.028511	0.011483
1	145.180762	188.441463	106.893558	0.006989	0.000051	0.003757	0.003900	0.011273	0.033658	0.321204	...	0.027600	0.053027	0.029211

2 rows x 22 columns

9.4 Data Pre-Processing & Separating the features & Target

```
✓ [10] X = parkinsons_data.drop(columns=['name', 'status'], axis=1)
0s      Y = parkinsons_data['status']
```

```
✓ [11] print(X)
0s
```

	MDVP:F0(Hz)	MDVP:F1(Hz)	MDVP:F2(Hz)	MDVP:Jitter(%)	\
0	119.992	157.302	74.997	0.00784	
1	122.400	148.650	113.819	0.00968	
2	116.682	131.111	111.555	0.01050	
3	116.676	137.871	111.366	0.00997	
4	116.014	141.781	110.655	0.01284	
..	
190	174.188	230.978	94.261	0.00459	
191	209.516	253.017	89.488	0.00564	
192	174.688	240.005	74.287	0.01360	
193	198.764	396.961	74.904	0.00740	
194	214.289	260.277	77.973	0.00567	

9.5 Splitting the data to training data & Test data

```
✓ [13] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
0s
```

```
✓ [14] print(X.shape, X_train.shape, X_test.shape)
0s
```

(195, 22) (156, 22) (39, 22)

9.6 Data Standardization

```
✓ [15] scaler = StandardScaler()
0s
```

```
✓ [16] scaler.fit(X_train)
0s
```

```
▼ StandardScaler
StandardScaler()
```

```
✓ [17] X_train = scaler.transform(X_train)
0s
```

```
X_test = scaler.transform(X_test)
```

```
✓ [18] print(X_train)
0s
```

```
[[ 0.63239631 -0.02731081 -0.87985049 ... -0.97586547 -0.55160318
  0.07769494]
 [-1.05512719 -0.83337041 -0.9284778 ... 0.3981808 -0.61014073
  0.39291782]
 [ 0.02996187 -0.29531068 -1.12211107 ... -0.43937044 -0.62849605
 -0.50948408]
 ...
 [-0.9096785 -0.6637302 -0.160638 ... 1.22001022 -0.47404629
 -0.2159482 ]
 [-0.35977689 0.19731822 -0.79063679 ... -0.17896029 -0.47272835
  0.28181221]
 [ 1.01957066 0.19922317 -0.61914972 ... -0.716232 1.23632066
 -0.05829386]]
```

9.7 Model Training

```
✓ 0s model = svm.SVC(kernel='linear')
```

```
✓ 0s [20] # training the SVM model with training data
      model.fit(X_train, Y_train)
```

```
▼ SVC
SVC(kernel='linear')
```

9.7.1 Accuracy Score

```
✓ 0s [21] # accuracy score on training data
      X_train_prediction = model.predict(X_train)
      training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
```

```
✓ 0s [22] print('Accuracy score of training data : ', training_data_accuracy)

      Accuracy score of training data :  0.8846153846153846
```

```
✓ 0s [23] # accuracy score on training data
      X_test_prediction = model.predict(X_test)
      test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
```

```
✓ 0s [24] print('Accuracy score of test data : ', test_data_accuracy)

      Accuracy score of test data :  0.8717948717948718
```

9.8 Building a Predictive System

```
✓ 0s [25] input_data = (197.07600,206.89600,192.05500,0.00289,0.00001,0.00166,0.00168,0.00498,0.01098,0.09700,0.00563,0.00680,0.00802,0.01689,0.00339,26.77500,0.422229,0.741367,-7.348300,0.1775

      # changing input data to a numpy array
      input_data_as_numpy_array = np.asarray(input_data)

      # reshape the numpy array
      input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

      # standardize the data
      std_data = scaler.transform(input_data_reshaped)

      prediction = model.predict(std_data)
      print(prediction)

      if (prediction[0] == 0):
          print("The Person does not have Parkinsons Disease")
      else:
          print("The Person has Parkinsons")

[0]
The Person does not have Parkinsons Disease
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
warnings.warn(
```

10. CONCLUSION

10.1 Conclusion

Parkinson's disease is the second most dangerous neurodegenerative disease which has no cure till now and to make it reduce detection is important. In this project, we have used three various detection models to predict the Parkinson's disease which are Machine Learning Techniques i.e., KNN, Naïve Bayes and Logistic Regression. The dataset is trained using these models and we also compared these different models built using different methods and identifies the best model that fits. The aim is to use various evaluation metrics such as Accuracy, Precision, Recall, Specificity score that produce the detects the disease efficiently. We have used the Speech dataset that contains voice features of the patients which is available in the Kaggle website. The dataset consists of more than 700 features and 750 patient details. The models are built using the five best features which were identified by feature selection. From this results, Naïve Bayes outstands from the other two machine learning algorithms with an accuracy of 81%. This system we designed can make the predictions of the Parkinson's disease.

10.2 Future Work

In future, these models can be trained with different datasets that have best features and can be predicted more accurately. If the accuracy rate increases, it can be used by the laboratories and hospitals so that it is easy to predict in early stages. This model can be also used with different medical and disease datasets. In future the work can be extended by building a hybrid model that can find more than one disease with an accurate dataset and that dataset has common features of two diseases. In future the work can extended to build a model that may extract more important features among all features in the dataset so that it produces more accuracy.

11. REFERENCES

1. Ozcift, “SVM feature selection-based rotation forest ensemble classifiers to improve computer-aided diagnosis of Parkinson disease” *Journal of medical systems*, vol-36, no. 4, pp. 2141-2147, 2012.
2. Carlo Ricciardi, et al, “Using gait analysis’ parameters to classify Parkinsonism: A data mining approach” *Computer Methods and Programs in Biomedicine* vol. 180, Oct. 2019.
3. Dr. R.GeethaRamani, G.Sivagami, ShomonaGraciaJacob “Feature Relevance Analysis and Classification of Parkinson’s Disease TeleMonitoring data Through Data Mining” *International Journal of Advanced Research in Computer Science and Software Engineering*, vol-2, Issue 3, March 2012.
4. Dragana Miljkovic et al, “Machine Learning and Data Mining Methods for Managing Parkinson’s Disease” *LNAI 9605*, pp. 209-220, 2016.
5. Heisters. D, “Parkinson’s: symptoms, treatments and research”. *British Journal of Nursing*, 20(9), 548–554. doi:10.12968/bjon.2011.20.9.548, 2011.
6. M. Abdar and M. Zomorodi-Moghadam, “Impact of Patients’ Gender on Parkinson’s disease using Classification Algorithms” *Journal of AI and Data Mining*, vol-6, 2018.
7. M. A. E. Van Stiphout, J. Marinus, J. J. Van Hilten, F. Lobbezoo, and C. De Baat, “Oral health of Parkinson’s disease patients: a case-control study” *Parkinson’s disease*, vol- 2018, Article ID 9315285, 8 pages, 2018.
8. Md. Redone Hassan, et al, “A Knowledge Base Data Mining based on Parkinson’s Disease” *International Conference on System Modelling & Advancement in Research Trends*, 2019.
9. R. P. Duncan, A. L. Leddy, J. T. Cavanaugh et al., “Detecting balance decline in Parkinson disease: a prospective cohort study” *Journal of Parkinson’s Disease*, vol-5, no. 1, pp. 131–139, 2015.
10. Ramzi M. Sadek et al., “Parkinson’s Disease Prediction using Artificial Neural Network” *International Journal of Academic Health and Medical Research*, vol- 3, Issue 1, January 2019.

11. Shahid, A.H., Singh, M.P. A deep learning approach for prediction of Parkinson's disease progression, <https://doi.org/10.1007/s13534-020-00156-7>, Biomed. Eng. Lett. 10, 227–239, 2020.
12. Sriram, T. V., et al. "Intelligent Parkinson Disease Detection Using Machine Learning Algorithms" International Journal of Engineering and Innovative Technology, vol-3, Issue 3, September 2013.
13. T. Swapna, Y. Sravani Devi, "Performance Analysis of Classification algorithms on Parkinson's Dataset with Voice Attributes". International Journal of Applied Engineering Research ISSN 0973-4562 Volume 14, Number 2 pp. 452-458, 2019.
14. T. J. Wroge, Y. Özkanca, C. Demiroglu, D. Si, D. C. Atkins and R. H. Ghomi, "Parkinson's Disease Diagnosis Using Machine Learning and Voice," IEEE Signal Processing in Medicine and Biology Symposium (SPMB), pp.1-7, doi: 10.1109/SPMB.2018.8615607, 2018.
15. <https://www.researchgate.net/publication/357448942>
16. Blauwendraa, C., Bandres-Ciga, S. Singleton, A. B. Predicting progression in patients with Parkinsons disease using voice. Lancet Neurol. 16, 2017.
17. Chaudhuri, K. R., Healy, D. G. Schapira, A. H. Non-motor symptoms of Parkinson's disease: diagnosis and management. Lancet Neurol. 5,2006.
18. Das R. "A comparison of multiple classification methods for diagnosis of Parkinson disease". Expert Systems With Applications"; 2010.
19. Hon, V., Lokaj, J. Rektor, I. Interleukin-6 may contribute to mortality in Parkinson's disease patients: a 4-year prospective study. Park. Dis. 2015.
20. Leclair-Visonneau, L. et al. REM sleep behavior disorder is related to enteric neuro-pathology in Parkinson disease. Neurology 89, 2017.