A Project report on

Prediction of Parkinson's disease and severity of the type using Machine Learning

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

Bachelor of Technology

in

Computer Science and Engineering

Submitted by

RANABOTU RAKSHITH REDDY
(19H51A0521)

RANGA SATYA RAJ
(19H51A0522)

GURIJALA GANENDHAR
(19H51A05K3)

Under the esteemed guidance of

Dr. P. Senthil
(Assistant Professor)

Department of Computer Science and Engineering

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC Autonomous)

*Approved by AICTE *Affiliated to JNTUH *NAAC Accredited with A^+ Grade

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

2019- 2023

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHALROAD, HYDERABAD–501401

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Major Project Phase-II report entitled **"Prediction of Parkinson’s disease and severity of the type using Machine Learning"** being submitted by **Ranabotu Rakshith Reddy(19H51A0521)**, **Ranga Satya Raj (19H51A0522)** and **Gurijala Ganendhar(19H51A05K3)** in partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineering is a record of bonafide work carried out his/her under my guidance and supervision.

The results embodies in this project report have not been submitted to any other University or Institute for the award of any Degree.

Dr.P. Senthil
Assistant Professor
Dept. Of CSE

Dr. S. Siva Skandha
Associate Professor and HOD
Dept. Of CSE

ACKNOWLEDGMENT

With great pleasure we want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project work a grand success.

We are grateful to **Dr. P. Senthil,** Assistant Professor, Department of Computer Science and Engineering for his valuable technical suggestions and guidance during the execution of this project work.

We would like to thank **Dr. Siva Skandha Sanagala,** Head of the Department of Computer Science and Engineering, CMR College of Engineering and Technology, who is the major driving forces to complete my project work successfully.

We are very grateful to **Dr. Vijaya Kumar Koppula**, Dean-Academics, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We are highly indebted to **Major Dr. V A Narayana,** Principal, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the **Teaching & Non- teaching** staff of Department of Computer Science and Engineering for their co-operation

We express our sincere thanks to **Shri. Ch. Gopal Reddy**, Secretary, CMR Group of Institutions, for his continuous care.

Finally, We extend thanks to our parents who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project work.

SIGNATURE

Ranabotu Rakshith Reddy(19H51A0521)

Ranga Satya Raj (19H51A0522)

Gurijala Ganendhar (19H51A05K3)

TABLE OF CONTENTS

List of Figures

List of Tables

ABSTRACT

Parkinson disease is a neurodegenerative disorder that affects nervous system and the root cause of it is falling rates of dopamine levels in the forebrain. It is a chronic degenerative disease with progressive illness, which means it develops new symptoms over time. This happens with progressive neuronal loss in the substantia nigra of brain. People with Parkinson disease cannot do their works as a normal human. Though clinical assessments considered ample amount of data that include various features, sometimes it is hard to decide whether a person is suffering from Parkinson disease or not based on the type of data, feature selection methods help to solve this issue. Various methods are developed, proposed, and analyzed to detect the Parkinson disease, given the required data. This project predicts Parkinson disease using machine learning algorithms by applying various technologies.

CHAPTER 1

INTRODUCTION

INTRODUCTION

Parkinson's disease (PD), or simply Parkinson's, is a long-term degenerative disorder of the central nervous system that mainly affects the motor system. The symptoms usually emerge slowly, and as the disease worsens, non-motor symptoms become more common. The most obvious early symptoms are tremor, rigidity, slowness of movement, and difficulty with walking. Cognitive and behavioral problems may also occur with depression, anxiety, and apathy occurring in many people with PD. Parkinson's disease dementia becomes common in the advanced stages of the disease. Those with Parkinson's can also have problems with their sleep and sensory systems. The motor symptoms of the disease result from the death of cells in the substantia nigra, a region of the midbrain, leading to a dopamine deficit. The cause of this cell death is poorly understood but involves the build-up of misfolded proteins into Lewy bodies in the neurons. Collectively, the main motor symptoms are also known as parkinsonism or a parkinsonian syndrome.

Parkinson's disease symptoms can be different for everyone. Early signs are mild that goes unnoticed. Symptoms usually begin on one side of your body and gets worsen on that side, afterwards it affects both the sides. Parkinson's symptoms may include

- Tremor
- Slowed movement
- Rigid muscles.
- Impaired posture and balance.
- Loss of automatic movements
- Speech changes
- Writing changes
- Muscle loss

The Parkinson's disease is due to a loss of neurons that produce a chemical messenger in the brain called dopamine. when there is a decrease in level of the amino acid named dopamine it leads to the abnormal brain activity, which leads to Parkinson’s disease. The cause of Parkinson's disease is still a question mark, but several factors appear to play a role, including:

- Genes
- Environmental
- Triggers

As a result, people suffer from this disease for many years before diagnosis. The estimated results have shown that there are 7-10 million people are affected by Parkinson’s disease worldwide. People with age above 50 are the ones who has the higher possibility of getting Parkinson’s disease but still an estimated 4 percentage of people who are under the age 50 are diagnosed with Parkinson’s disease. There is no cure or prevention for PD. However, the disease can be controlled in early stage. Since this disease is progressive in nature, negligence in the diagnosis of this disease in the early stage and monitoring at different stages would create a severe negative impact on the patients in terms of healthcare costs as well as the severe healthrelated disorders. To prevent the major negative impact on PD patient's it is necessary to detect the PD at the early stage.

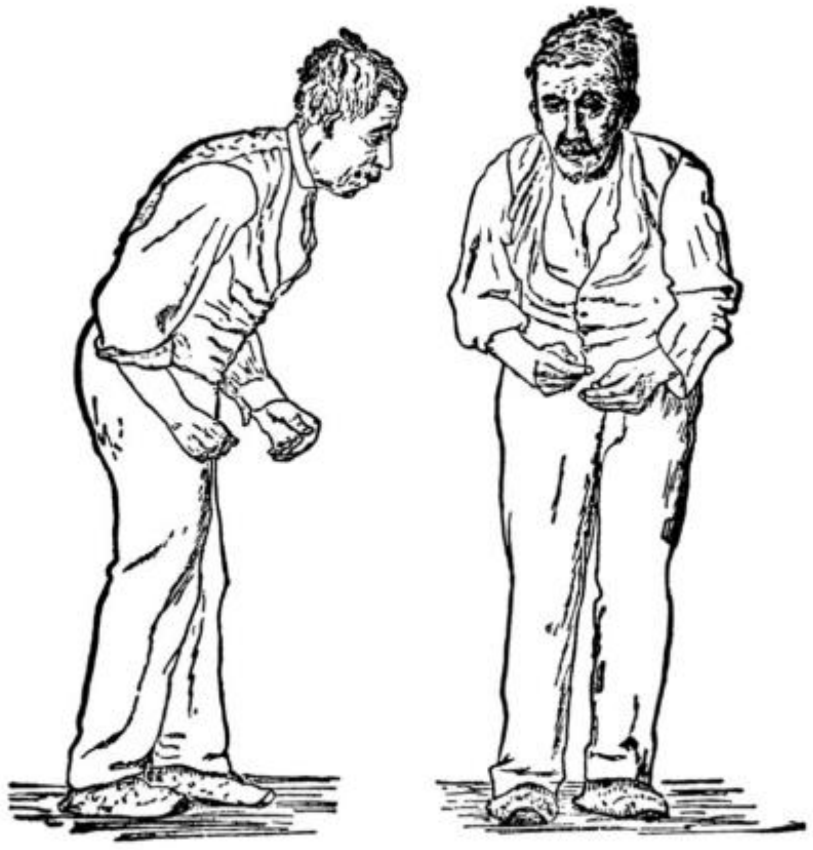


Fig. 1.1 Parkinson’s Disease Patients

ANXIETY AND DEPRESSION

The most common symptoms experienced by people who have been diagnosed with Parkinson’s disease are anxiety and depression. The patient is understandably anxious, fearful about how their lives will change in general and how functional impairment caused by the disease will manifest itself. Depression, apathy and withdrawal from things a person previously enjoyed are another frequent symptom. Anxiety and depression occur in about 40% to 50% of Parkinson’s patients at one time or another over the course of the disease. PERSONALITY CHANGES Family members, friends and caregivers may notice changes in personality brought on by neurological changes in the brain because of Parkinson’s disease. The changes can be varied. Examples include:

- A person who was always conscientious becomes careless
- A previously easy-going person becomes rigid and stubborn
- An outgoing social butterfly turns into a stay-at-home introvert

IMPULSIVE OR COMPULSIVE BEHAVIORS

Some Parkinson’s patients act impulsively, unable to control the desire to do certain things. This behavior can range from innocuous-seeming excessive internet use to:

- Hoarding
- Charity donations
- Gambling
- Excessive eating or drinking
- Compulsive sexual preoccupations

These symptoms are typically caused by certain medications used for treatment of motor symptoms of Parkinson's disease. They may occur in up to 15% of patients who receive such medications. But they are most likely to occur in people who were predisposed to these conditions before diagnosis. These behaviors can be quite destructive if, for example, a spouse is distressed by a loved one’s new obsession with pornography or neighbors become alarmed at the patient’s insistence on fixing their fences even after completing all the work.

HALLUCINATIONS

Parkinson's patients can experience hallucinations that range from mild to severe, pleasant to frightening. Sometimes, patients describe the sensation of feeling a presence near them or of seeing something passing on the periphery of their vision. Others may be delighted by a vision of little children or flowers. Sometimes the hallucinations are dark or upsetting. The patient may become alarmed at the sight of bugs on the floor, at sensing a stranger's presence in the house or believing someone is stealing from them. Parkinson's patients have reported seeing human faces in clouds or a person in a coat that is hanging on a rack.

What are the symptoms?

The best-known symptoms of Parkinson's disease involve loss of muscle control. However, experts now know that muscle control-related issues aren't the only possible symptoms of Parkinson's disease.

Motor-related symptoms

Motor symptoms — which means movement-related symptoms — of Parkinson's disease include the following:

• Slowed movements (bradykinesia): A Parkinson's disease diagnosis requires that you have this symptom. People who have this describe it as muscle weakness, but it happens because of muscle control problems, and there's no actual loss of strength.

• A tremor: while muscles are at rest. This is a rhythmic shaking of muscles even when you're not using them and happens in about 80% of Parkinson's disease cases. Resting tremors are different from essential tremors, which don't usually happen when muscles are at rest.

• Rigidity or stiffness: Lead-pipe rigidity and cogwheel stiffness are common symptoms of Parkinson's disease. Lead-pipe rigidity is a constant, unchanging stiffness when moving a body part. Cogwheel stiffness happens when you combine tremor and leadpipe rigidity. It gets its name because of the jerky, stop-and-go appearance of the movements (think of it as the second hand on a mechanical clock).

• Unstable posture or walking gait:The slowed movements and stiffness of Parkinson's disease cause a hunched over or stooped stance. This usually appears as the disease gets worse. It's visible when a person walks because they'll use shorter, shuffling strides and move their arms less. Turning while walking may take several steps.

Additional motor symptoms

- Blinking less often than usual. This is also a symptom of reduced control of facial muscles.
- Cramped or small handwriting. Known as micrographia, this happens because of muscle control problems.
- Drooling. Another symptom that happens because of loss of facial muscle control.
- Mask-like facial expression. Known as hypomimia, this means facial expressions change very little or not at all.

Non-motor symptoms

Several symptoms are possible that aren't connected to movement and muscle control. In years past, experts believed non-motor symptoms were risk factors for this disease when seen before motor symptoms. However, there’s a growing amount of evidence that these symptoms can appear in the earliest stages of the disease. That means these symptoms might be warning signs that start years or even decades before motor symptoms. Non-motor symptoms (with the potential early warning symptoms in bold) include:

- Autonomic nervous system symptoms. These include orthostatic hypotension (low blood pressure when standing up), constipation and gastrointestinal problems, urinary incontinence and sexual dysfunctions.
- Depression.
- Loss of sense of smell (anosmia).
- Sleep problems such as periodic limb movement disorder (PLMD), rapid eye movement (REM) behavior disorder and restless legs syndrome.
- Trouble thinking and focusing (Parkinson’s-related dementia).

Stages of parkinson’s disease

Parkinson’s disease can take years or even decades to cause severe effects. In 1967, two experts, Margaret Hoehn and Melvin Yahr, created the staging system for Parkinson’s disease. That staging system is no longer in widespread use because staging this condition is less helpful than determining how it affects each person’s life individually and then treating them accordingly. Today, the Movement Disorder Society-Unified Parkinson's Disease Rating Scale (MDSUPDRS) is healthcare providers' main tool to classify this disease. The MDS-UPDRS examines four different areas of how Parkinson’s disease affects you:

- **Part 1: Non-motor aspects of experiences of daily living.** This section deals with nonmotor (non-movement) symptoms like dementia, depression, anxiety and other mental ability- and mental health-related issues. It also asks questions about pain, constipation, incontinence, fatigue, etc. Detecting Parkinson's Disease using ML

- **Part 2: Motor aspects of experiences of daily living.** This section covers the effects on movement-related tasks and abilities. It includes your ability to speak, eat, chew and swallow, dress and bathe yourself if you have tremors and more.

- **Part 3: Motor examination**. A healthcare provider uses this section to determine the movement-related effects of Parkinson's disease. The criteria measure effects based on how you speak, facial expressions, stiffness and rigidity, walking gait and speed, balance, movement speed, tremors, etc.

- **Part 4: Motor complications.** This section involves a provider determining how much of an impact the symptoms of Parkinson's disease are affecting your life. That includes both the amount of time you have certain symptoms each day, and whether or not those symptoms affect how you spend your time

1.1 PROBLEM STATEMENT

To build a robust model capable of predicting the Parkinson's disease at early stages with higher accuracy. In general, Parkinson's disease cannot be detected at the early stages using CT/MRI scans. There is no cure for Parkinson's disease at severe stages. This disease is commonly seen in old age people. Predicting the disease at early stage will increase the chance of curing Parkinson's disease.

1.2 RESEARCH OBJECTIVES

The study shed some vital light on prior and ongoing research to predict Parkinson’s disease in an efficient and effective manner. Since most of the medical centers, hospitals, or diagnosis centers are not equipped with modern computer-based machines for testing and diagnosis, early detection of Parkinson’s disease is not possible. Using machine learning algorithms on the lab data, a model can be generated for a much more efficient diagnosis. Analysis based on the input and classification algorithms may give various accuracy rates.

In study it is observed that until now, in the majority of cases full features have been taken into consideration. As many datasets have imbalanced class, class balancing is needed for increasing the performance of classifier model. It is important that apart from accuracy it is important to make sure that the model has low false positive rates, as it could affect the mental health of the person, if the model predicts the person of having a disease while the person does not actually have any kind of disease.

The main aim of this project is to automate the Parkinson's disease 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 life of the patients and their families
The main objective of this project is to understand what is Parkinson's disease and build a robust model that would predict Parkinson’s disease in early stage with minimal error rate. To improve accuracy of the model used for detecting and predicting Parkinson’s disease. To build supervised machine learning algorithms to train the models on cleaned UCI Parkinson's dataset.

1.3 PROJECT SCOPE AND LIMITATIONS

The brain is the main controller of our body. Therefore, any damage to this sensitive part of the human body will affect badly on the other organs. One of these negative effects is Parkinson’s disease. Parkinson’s disease (PD) is a chronic, progressive, neurodegenerative disorder which begins when a certain area of the brain has been damaged. Parkinson’s disease (PD) is a progressive neurodegenerative condition leading to the death of the dopamine containing cells of the substantia nigra. There is no consistently reliable test that can distinguish Parkinson’s disease from other conditions with similar clinical presentations. This disease cannot be accurately identified at early stages with MRI, CT scans. Parkinson disease is one of the most serious diseases. Hence diagnosing it at an earlier stage could help prevent or reduce the effects.

CHAPTER 2

BACKGROUND WORK

2.1 Voice data analysis using decision tree

2.1.1 Introduction

- Training Accuracy describes the accuracy achieved on the training set.
- Validation Accuracy describes the accuracy achieved on the Test Set.
- Confusion matrix is the easiest way to measure the performance of a classification problem where the output can be of two or more type of classes. A confusion matrix is nothing but a table with two dimensions viz. "Actual" and "Predicted" and furthermore, both the dimensions have "True Positives (TP)", "True Negatives (TN)", "False Positive (FP)", "False Negatives (FN)" as shown below FIG: 1 Confusion Matrix
- Precision is defined as the ratio of correctly predicted positive observations to the total predicted positive observations.
- The formula for Precision is Precision =TP/TP +FP
- The Sensitivity or Recall is defined as the proportion of correctly identified positives.
- The formula for Recall is Recall =TP/TP+FN
- F1-Score is the Harmonic Mean of Precision and Recall.
- A correlation matrix is simply a table which displays the correlation. The measure is best used in variables that demonstrate a linear relationship between each other.

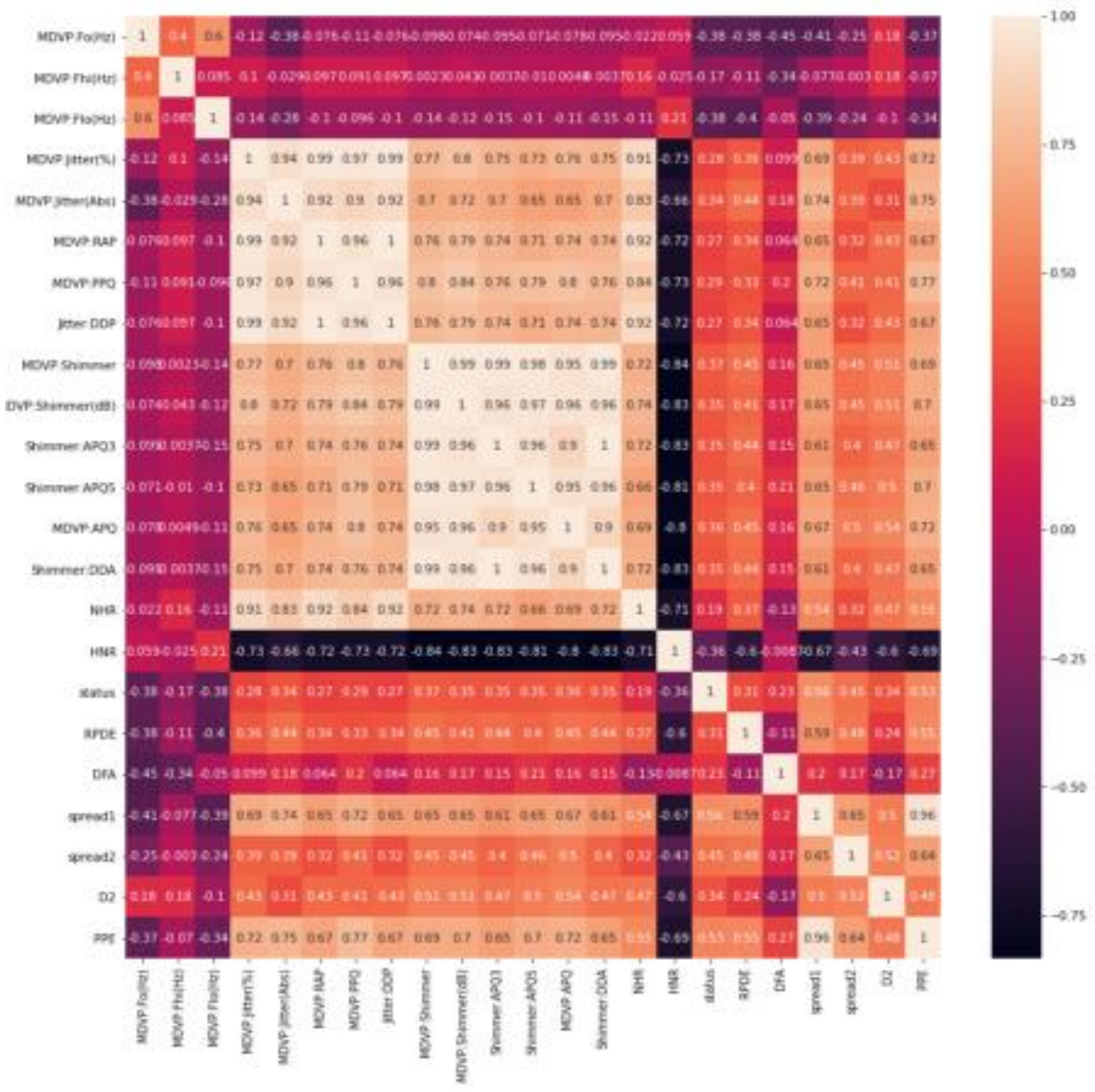


Fig. 2.1.1 Correlation Matrix

2.1.2 Merits and Demerits

Merits

Simple to understand and to interpret. Trees can be visualized.

Requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.

Demerits

Decision-tree learners can create over-complex trees that do not generalize the data well. This is called overfitting. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.

Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.

2.1.3 Implementation of Voice Data Analysis using Decision Tree

By using machine learning techniques, the problem can be solved with minimal error rate. The voice dataset of Parkinson's disease from the UCI Machine learning library is used as input. PD voice dataset is collected from UCI machine learning repository and these are stored into the RStudio environment as Testing and Training datasets. These are stored into the RStudio environment as Testing and Training datasets. R is a programming language and software environment for statistical analysis, graphics representation, data analysis and as well as machine learning. It involves the following steps and procedures

1. **Importing data to RStudio** - organize the data in an Excel worksheet to include column names in the first row (i.e., person's voice collected at various time zones) and each subsequent row contains all the information (i.e. set of 22 parameter is taken into consideration and the person's voice range for those parameters is tested and then noted), finally the status column shows two values 0 (healthy) and 1(affected). Import data into RStudio, using the "Import data..." feature.

2. **Classification** - It is also called a prediction tree. It uses a structure to specify sequences of decisions and consequences, the goal is to predict a response or output. The forecast can be accomplished by creating a decision tree with test points and branches. At each check point, a decision is made to pick a particular branch and cross the trees and can be used in a variety

of disciplines, on the basis of individual characteristics

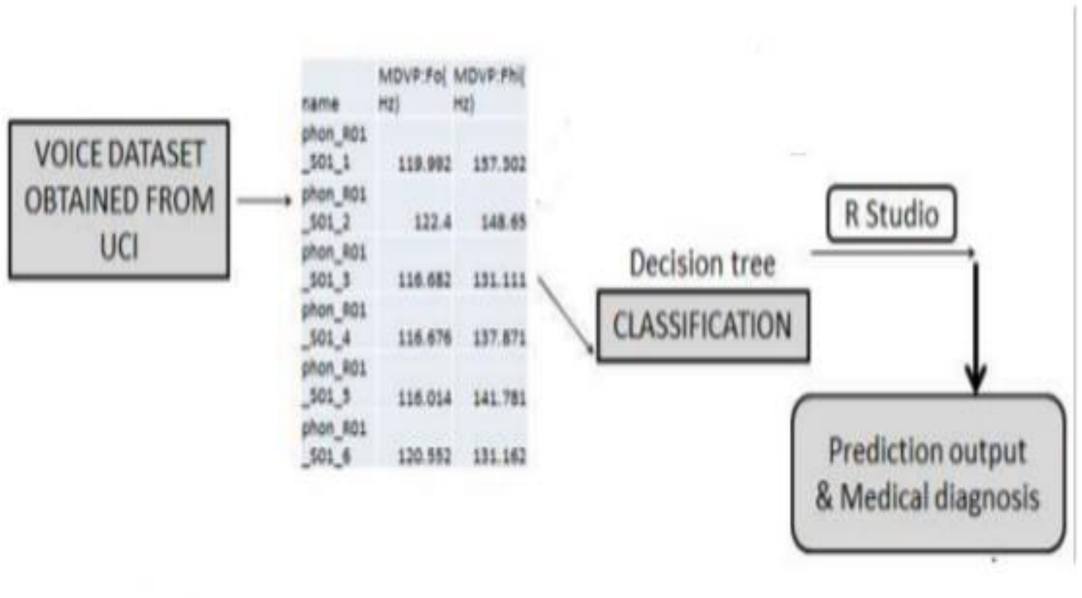

VOICE DATASET OBTAINED FROM UCI
name
MDVP:Fo(Hz)
MDVP:Fhi(Hz)
phon_R01_S01_1 119.992 157.302
phon_R01_S01_2 122.4 148.65
phon_R01_S01_3 116.682 131.111
phon_R01_S01_4 116.676 137.871
phon_R01_S01_5 116.014 141.781
phon_R01_S01_6 120.552 131.162
Decision tree
CLASSIFICATION
R Studio
Prediction output & Medical diagnosis

Fig. 2.1.2 Architecture Diagram of Voice Data Analysis using Decision Tree

The predicted output for voice data analysis based on classification is with an accuracy of 88%.

Model Name	**Accuracy**
Decision Tree	88%

Table 2.1 Accuracy of Decision Tree

2.2 Spiral drawing analysis using random forest classifier

2.2.1 Introduction

A random forest is a machine learning technique that’s used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems. A random forest algorithm consists of many decision trees. The ‘forest’ generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.

2.2.2 Merits and Demerits

Merits

It reduces overfitting in decision trees and helps to improve the accuracy

It is flexible to both classification and regression problems

Demerits

It requires much computational power as well as resources as it builds numerous trees to combine their outputs.

It also requires much time for training as it combines a lot of decision trees to determine the class.

Due to the ensemble of decision trees, it also suffers interpret ability and fails to determine the significance of each variable.

2.2.3 Implementation of Spiral Drawing Analysis using Random Forest Classifier

Evaluation metrics were used to evaluate the performance of the random forest classifier. One of these measures is through the confusion matrix, from which the accuracy is extracted by computing the correctly classified samples (TP and TN) and the incorrectly classified samples (FP and FN), as shown in the following equation.

$$\text{accuracy} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{TP} + \text{FN} + \text{FP}} * 100\%,$$

In this method, input is spiral drawings from the machine learning repository. These drawings are subjected to resizing and histogram equalization. The images from the spiral drawings were resized to 256px width and 256px height. The spiral drawing images that are collected were lack in contrast and brightness. Therefore, contrast enhancement and adjustment performed over all the images using histogram equalization.

Random Forest Classifier is applied to these enhanced images and Parkinson's disease prediction is made.

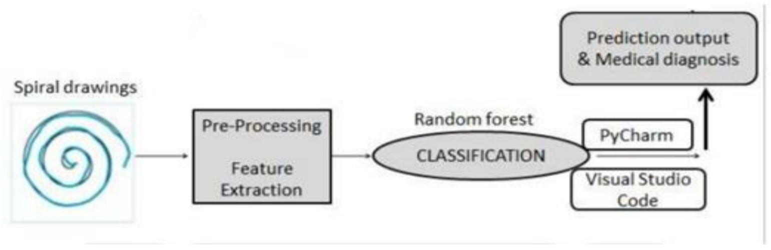


Fig. 2.2.1 Architecture Diagram of Spiral Drawing Analysis using Random Forest Classifier

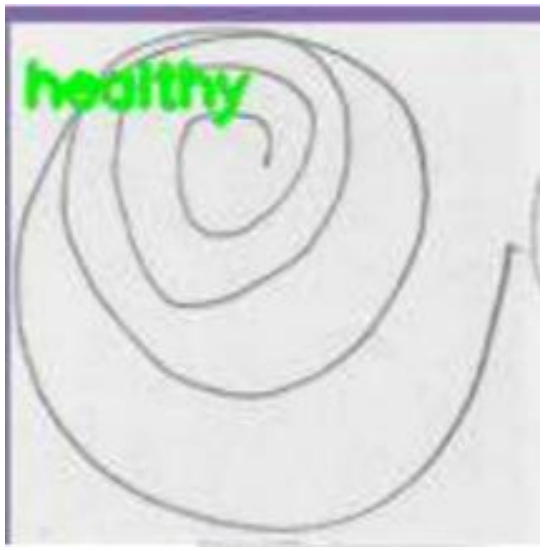


Fig. 2.2.2 Spiral Drawing of Healthy Person

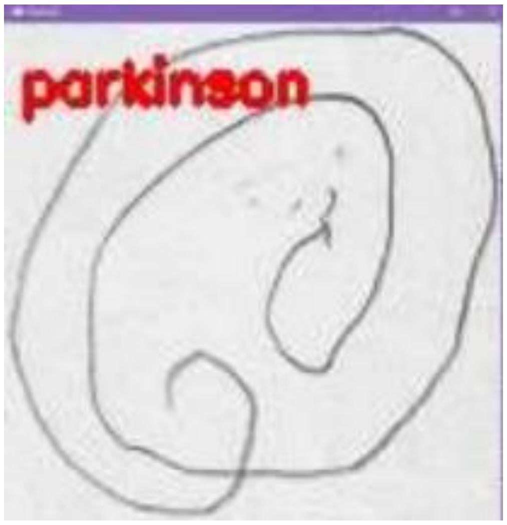


Fig. 2.2.3 Spiral Drawing of Parkinson's Patient

Model Name	**Accuracy**
Random Forest using spiral drawings	83%

Table 2.2 Accuracy of Random Forest Classifier

2.3 Parkinson's Disease Prediction using KNN

2.3.1 Introduction

The K-Nearest Neighbours (KNN) family of classification algorithms and regression algorithms is often referred to as memory-based learning or instance-based learning. These terms correspond to the main concept of KNN. The concept is to replace model creation by memorizing the training data set and then use this data to make predictions. The KNN algorithm uses a majority voting mechanism. It collects data from a training data set and uses this data later to make predictions for new records**.**

2.3.2 Merits and Demerits

Merits

No Training Period: KNN is called Lazy Learner (Instance based learning). It does not learn anything in the training period. It does not derive any discriminative function from the training data. In other words, there is no training period for it. It stores the training dataset and learns from it only at the time of making real time predictions. This makes the KNN algorithm much faster than other algorithms that require training e.g., SVM, etc.

Since the KNN algorithm requires no training before making predictions, new data can be added seamlessly which will not impact the accuracy of the algorithm.

Demerits

Does not work well with large dataset: In large datasets, the cost of calculating the distance between the new point and each existing points is huge which degrades the performance of the algorithm.

Does not work well with high dimensions: The KNN algorithm doesn't work well with high dimensional data because with large number of dimensions, it becomes difficult for the algorithm to calculate the distance in each dimension.

2.3.3 Implementation of Parkinson's Disease Prediction using KNN

Evaluation metrics were used to evaluate the performance of four classifiers. One of these measures is through the confusion matrix, from which the accuracy is extracted by computing the correctly classified samples (TP and TN) and the incorrectly classified samples (FP and FN), as shown in the following equation.

$$\text{accuracy} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{TP} + \text{FN} + \text{FP}} * 100\%,$$

The dataset used consists of a range of biomedical voice measurement with 195 samples of features from 31 people, 23 with Parkinson's disease (PD) and 8 of them are the control group. The data set has about 75% of cases suffering from Parkinson disease and 25% of cases which are healthy. Feature importance analysis is done which can provide insight into the model. Most important scores are calculated by a predictive model that has been fit on the dataset. This paper implemented four different algorithms out of which the one with higher accuracy is determined. These algorithms are:

- Logistic Regression
- Decision Tree
- SVM (Support Vector Machine)
- KNN (K-Nearest Neighbour)

CLASSIFICATION TECHNIQUES	**TRAINING ACCURACY RATE**	**TEST ACCURACY RATE**
Logistic Regression	0.88	0.79
Decision Tree	1.0	0.9
SVM	0.89	0.92
KNN	0.94	0.95

Table 2.3 Machine Learning Models and their accuracies

CHAPTER 3
PROPOSED SYSTEM

3.1 Objective of Proposed Model

XGBOOST ALGORITHM

XGBoost stands for eXtreme Gradient Boosting and it’s an open-source implementation of the gradient boosted trees algorithm designed for speed and performance. It has been one of the most popular machine learning techniques in Kaggle competitions, due to its prediction power and ease of use. It is a supervised learning algorithm that can be used for regression or classification tasks.

XGBoost classifier is a Machine learning algorithm that is applied for structured and tabular data. XGBoost is an extreme gradient boost algorithm. And that means it’s a big Machine learning algorithm with lots of parts. XGBoost works with large, complicated datasets. XGBoost is an ensemble modelling technique.

How XGBoost works?

To understand XGBoost first, a clear understanding of decision trees and ensemble learning algorithms is needed.

Gradient boosting

Boosting is an ensemble method, meaning it’s a way of combining predictions from several models into one. It does that by taking each predictor sequentially and modelling it based on its predecessor’s error (giving more weight to predictors that perform better):

Fit a first model using the original data

Fit a second model using the residuals of the first model

Create a third model using the sum of models 1 and 2

Gradient boosting is a specific type of boosting, called like that because it minimizes the loss function using a gradient descent algorithm.

Decision trees

Decision trees are arguably the most easily interpretable ML algorithms you can find and, if used in combination with the right techniques, can be quite powerful.

A decision tree has this name because of its visual shape, which looks like a tree, with a root and many nodes and leaves. Imagine you take a list of titanic survivors with some information such as their age and gender, and a binary variable telling who survived the disaster and who didn’t. You now want to create a classification model, to predict who will survive, based on this data. A very simple one would look like this:

As you can see, decision trees are just a sequence of simple decision rules that, combined, produce a prediction of the desired variable.

.

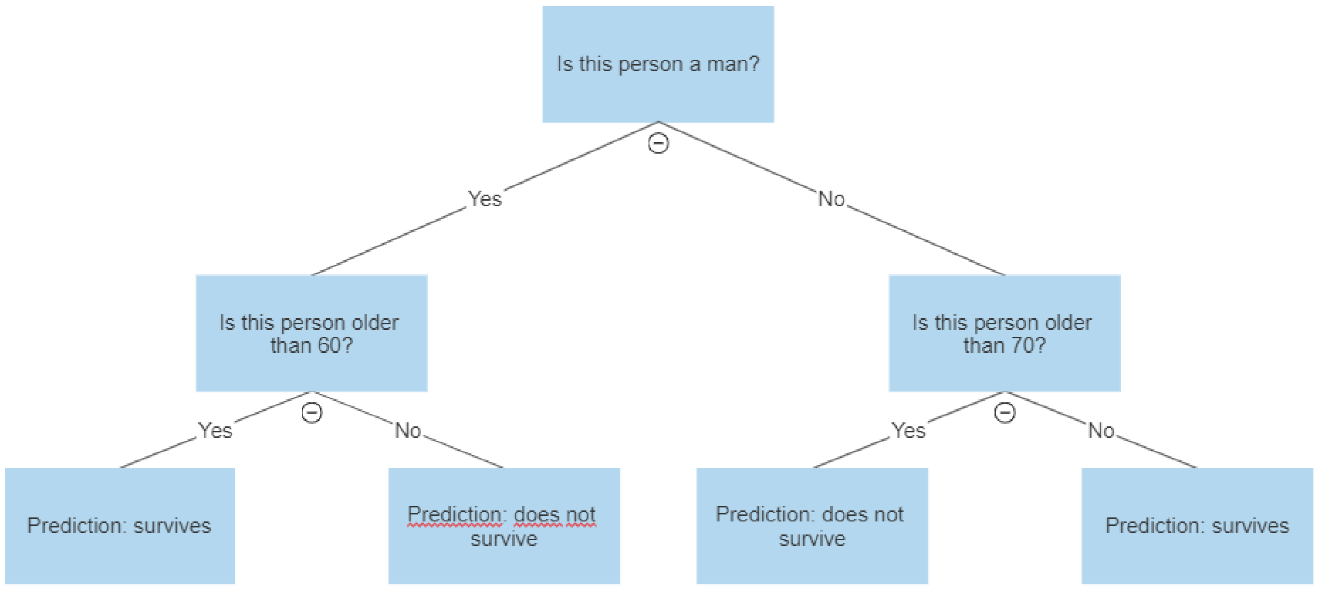

Is this person a man?
Yes
No
Is this person older than 60?
Is this person older than 70?
Yes
No
Yes
No
Prediction: survives
Prediction: does not survive
Prediction: does not survive
Prediction: survives

Fig. 3.1 Decision Tree Classifier

Advantages of XGBoost Algorithm in Machine Learning

XGBoost is an efficient and easy to use algorithm which delivers high performance and accuracy as compared to other algorithms. XGBoost is also known as regularized version of GBM. Let see some of the advantages of XGBoost algorithm:

- **Regularization:** XGBoost has in-built L1 (Lasso Regression) and L2 (Ridge Regression) regularization which prevents the model from overfitting. That is why, XGBoost is also called regularized form of GBM (Gradient Boosting Machine). While using Scikit Learn library, we pass two hyper-parameters (alpha and lambda) to XGBoost related to regularization. alpha is used for L1 regularization and lambda is used for L2 regularization.

- **Parallel Processing:** XGBoost utilizes the power of parallel processing and that is why it is much faster than GBM. It uses multiple CPU cores to execute the model. While using Scikit Learn library, thread hyper-parameter is used for parallel processing. Thread represents number of CPU cores to be used. If you want to use all the available cores, don't mention any value for thread and the algorithm will detect automatically.

- **Handling Missing Values:** XGBoost has an in-built capability to handle missing values. When XGBoost encounters a missing value at a node, it tries both the left- and right-hand split and learns the way leading to higher loss for each node. It then does the same when working on the testing data.

- **Cross Validation:** XGBoost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run. This is unlike GBM where we have to run a grid-search and only a limited values can be tested.

- **Effective Tree Pruning:** A GBM would stop splitting a node when it encounters a negative loss in the split. Thus, it is more of a greedy algorithm. XGBoost on the other hand make splits up to the max depth specified and then start pruning the tree backwards and remove splits beyond which there is no positive gain.

Building prediction model

The following libraries are required for building a robust model that would predict Parkinson’s disease:

• **NumPy:** NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

• **Pandas:** Pandas is an open-source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named NumPy, which provides support for multi-dimensional arrays.

• **Sci-Kit Learn:** Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python.

• **Matplotlib:** Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

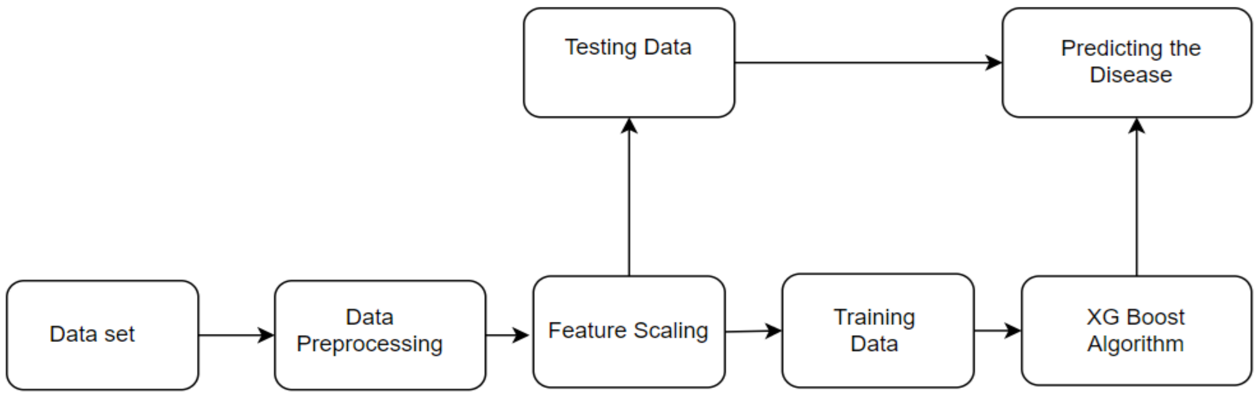
• **Seaborn:** Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

3.2 Algorithms Used for Proposed Model

- **XGBOOST Algorithm**
- **Decision Tree Classifier**

3.3 Designing

3.3.1 UML Diagram



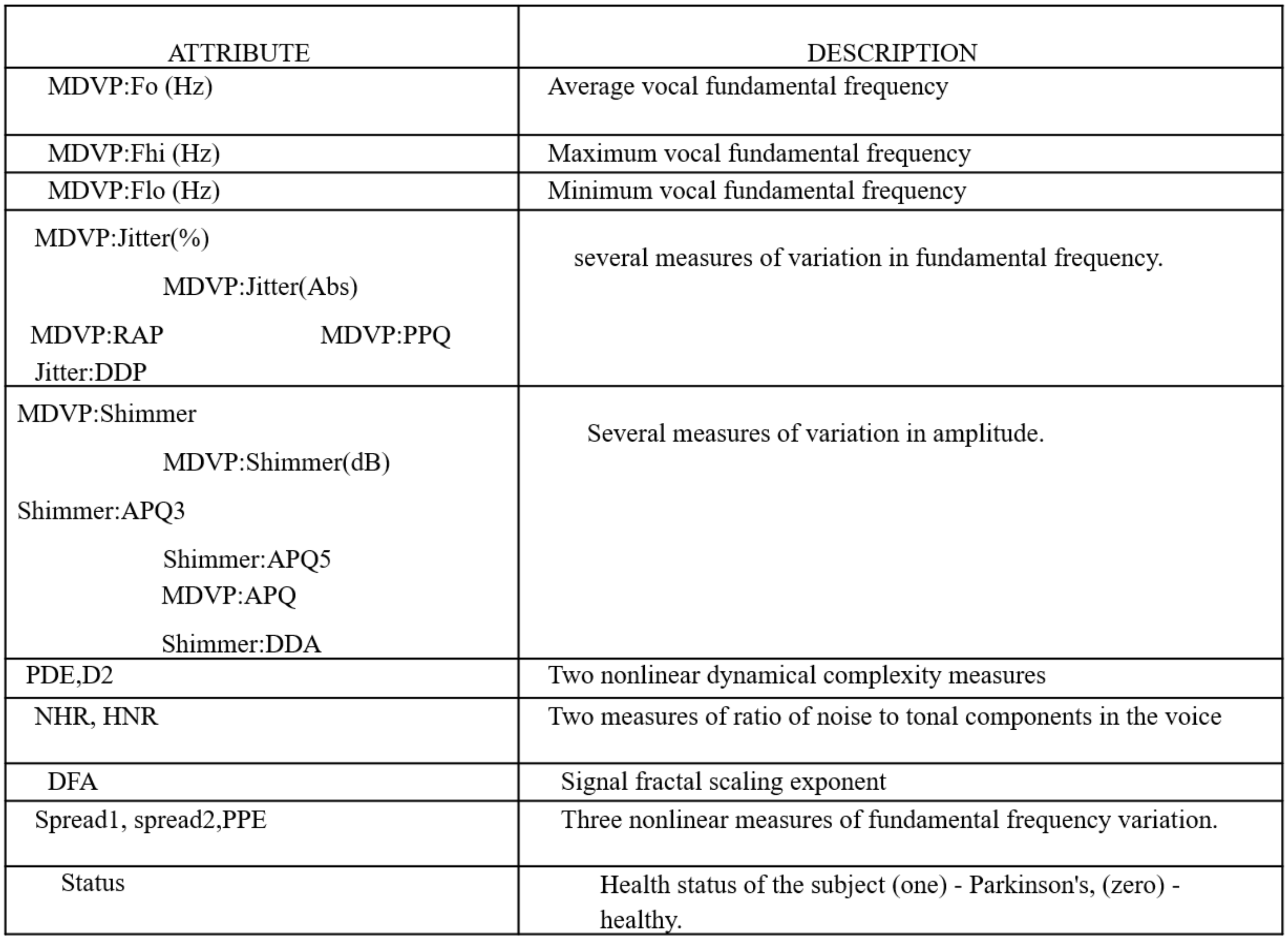
3.4 Stepwise Implementation and Code

Step 01 : Data Set Collection

For the proposed project datasets are taken from kaggle (a website that provides various datasets) for Parkinson's disease

+

3.4 Description of Dataset:



ATTRIBUTE	DESCRIPTION
MDVP:Fo (Hz)	Average vocal fundamental frequency
MDVP:Fhi (Hz)	Maximum vocal fundamental frequency
MDVP:Flo (Hz)	Minimum vocal fundamental frequency
MDVP:Jitter(%) MDVP:Jitter(Abs) MDVP:RAP MDVP:PPQ Jitter:DDP	several measures of variation in fundamental frequency.
MDVP:Shimmer MDVP:Shimmer(dB) Shimmer:APQ3 Shimmer:APQ5 MDVP:APQ Shimmer:DDA	Several measures of variation in amplitude.
PDE,D2	Two nonlinear dynamical complexity measures
NHR, HNR	Two measures of ratio of noise to tonal components in the voice
DFA	Signal fractal scaling exponent
Spread1, spread2,PPE	Three nonlinear measures of fundamental frequency variation.
Status	Health status of the subject (one) - Parkinson's, (zero) - healthy.

Step 02 : Importing Libraries and Dataset

- We import libraries such as Pandas, Numpy, Sklearn, Matplotlib and Seaborn. We import the dataset as.csv (It stands for Comma-Separated Values) file. Basically, CSV is a simple file format which is used to store tabular data (number and text) such as a spreadsheet in plain text.

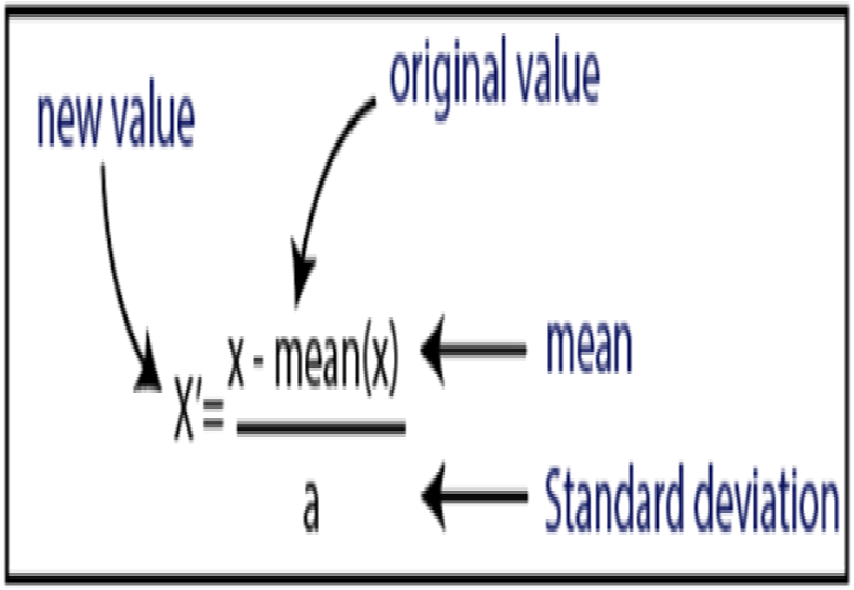
Step 03 : Preprocessing of Data

- Removing Missing Values: To remove the missing values by replacing them with the mean of that particular attribute. The other way is by dropping the rows or columns that contain null values.
- Extraction of Dependent and Independent Values: For every Machine Learning model, it is necessary to separate the independent variables (matrix of features) and dependent variables in a dataset. Pandas provides functions such as iloc() that can be used.
- Handling Categorical Variables: Categorical variables are basically the variables that are discrete and not continuous. Ex — color of an item is a discrete variable whereas its price is a continuous variable.

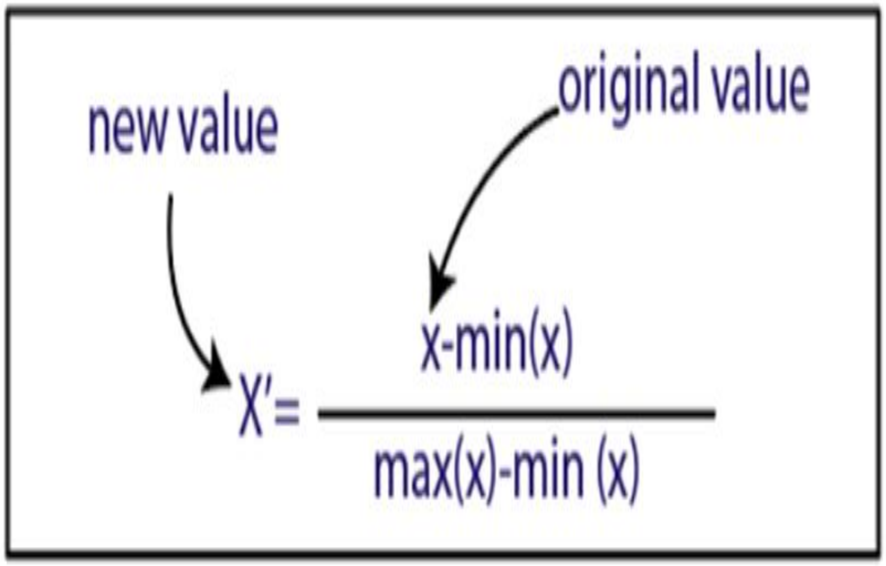
Step 04 : Feature Scaling

- Feature scaling marks the end of the data preprocessing in Machine Learning.
- It is a method to standardize the independent variables of a dataset within a specific range.
- In other words, feature scaling limits the range of variables so that you can compare them on common grounds.
- Feature scaling in Machine Learning can be done in two ways:

1. Standardization:



2.Normalization:

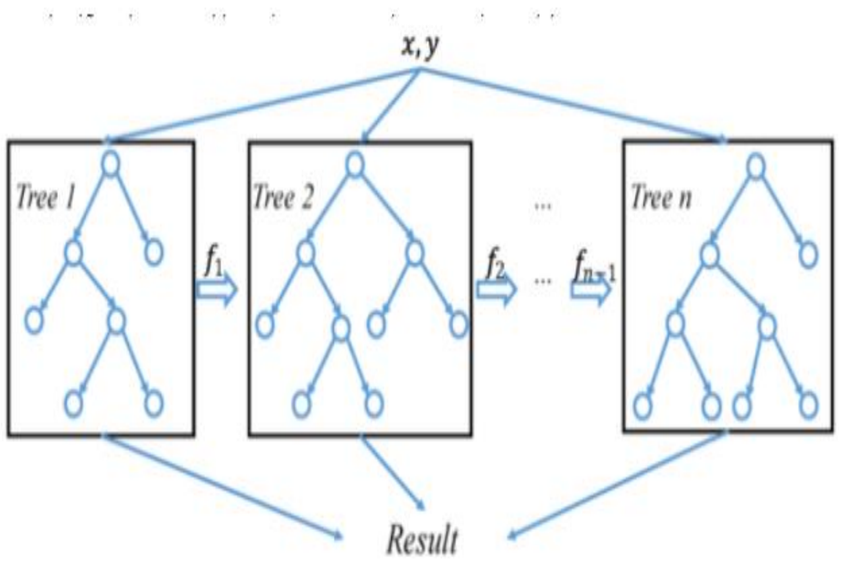


Step 05 : Splitting the Dataset

- Machine Learning model must be split into two separate sets – training set and test set.
- Training set denotes the subset of a dataset that is used for training the machine learning model. Here, you are already aware of the output.
- A test set, on the other hand, is the subset of the dataset that is used for testing the machine learning model. The ML model uses the test set to predict outcomes.
- Usually, the dataset is split into 70:30 ratio or 80:20 ratio.
- Sklearn Library provides a function called train_test_split function that can be used to perform splitting.

Step 06: Building XGBoost Model

- In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers then ensemble to give a strong and more precise model.



Step 07: Predicting Test Set Results

- Since the model is fitted to the training set, so now we can predict the test result. Accuracy is also calculated for the predicted values.

Step 08: Saving The Trained Model using Pickle

- Training a machine learning model may take from few seconds to several days even months. Hence it’s not practical to train a machine learning model again and again. Hence it’s required to save trained model and reuse it.
- Pickle is the standard way of serializing objects in Python. It provides an operation called pickle to serialize the machine learning models and save the serialized format to a file.
- Once the model is built and trained it is exported to a file using pickle.
- The model can be simply loaded to predict when required

SOURCE CODE

User side views.py

```
from django.contrib import messages
from django.shortcuts import render, HttpResponse
from django.conf import settings
import os
from .forms import UserRegistrationForm
from .models import UserRegistrationModel
# Create your views here.
def UserRegisterActions(request):
    if request.method == 'POST':
        form = UserRegistrationForm(request.POST)
        if form.is_valid():
            print('Data is Valid')
            form.save()
            messages.success(request, 'You have been successfully registered')
            form = UserRegistrationForm()
            return render(request, 'UserRegistrations.html', {'form': form})
        else:
            messages.success(request, 'Email or Mobile Already Existed')
            print("Invalid form")
    else:
        form = UserRegistrationForm()
    return render(request, 'UserRegistrations.html', {'form': form})
def UserLoginCheck(request):
    if request.method == "POST":
        loginid = request.POST.get('loginname')
        pswd = request.POST.get('pswd')
        print("Login ID = ", loginid, ' Password = ', pswd)
        try:
            check = UserRegistrationModel.objects.get(loginid=loginid, password=pswd)
            status = check.status
            print('Status is = ', status)
            if status == "activated":
                request.session['id'] = check.id
                request.session['loggeduser'] = check.name
                request.session['loginid'] = loginid
                request.session['email'] = check.email
                print("User id At", check.id, status)
                return render(request, 'users/UserHome.html', {})
            else:
                messages.success(request, 'Your Account Not at activated')
                return render(request, 'UserLogin.html')
        except Exception as e:
            print('Exception is ', str(e))
            pass
        messages.success(request, 'Invalid Login id and password')
    return render(request, 'UserLogin.html', {})
def UserHome(request):
    return render(request, 'users/UserHome.html', {})
def user_view_dataset(request):
    path = os.path.join(settings.MEDIA_ROOT, 'parkinsons.csv')
    import pandas as pd
    df = pd.read_csv(path)
    df = df.drop(['name',
'MDVP:Fo(Hz)','MDVP:Fhi(Hz)','MDVP:Flo(Hz)','MDVP:Jitter(%)','MDVP:Jitter(Abs)','MDVP:
RAP','MDVP:PPQ','Jitter:DDP','spread1','spread2','MDVP:Shimmer'], axis=1)
    df = df.to_html
    return render(request, 'users/view_data.html', {'df': df})
def user_model_evaluations(request):
    from .utility.parkinson_utility import start_models
    result = start_models()
    print(result)
    return render(request, 'users/model_results.html',result)
def user_predict_form(request):
    if request.method == 'POST':
        nhr = float(request.POST.get('NHR'))
        hnr = float(request.POST.get('HNR'))
        rpde = float(request.POST.get('RPDE'))
        dfa = float(request.POST.get('DFA'))
        ppe = float(request.POST.get('PPE'))
        test_data = [nhr, hnr,rpde,dfa, ppe]
        from .utility import process_user_input
        test_pred = process_user_input.get_result(test_data)
        print("Test Result is:", test_pred)
        if test_pred[0] == 0:
            rslt = False
        else:
            rslt = True
        return render(request, "users/parkinson_form.html", {"test_data": test_data, "result": rslt})
    else:
        return render(request, 'users/parkinson_form.html', {})
from django.contrib import messages
from django.shortcuts import render, HttpResponse
from django.conf import settings
import os
from .forms import UserRegistrationForm
from .models import UserRegistrationModel
# Create your views here.
def UserRegisterActions(request):
    if request.method == 'POST':
        form = UserRegistrationForm(request.POST)
        if form.is_valid():
            print('Data is Valid')
            form.save()
            messages.success(request, 'You have been successfully registered')
            form = UserRegistrationForm()
            return render(request, 'UserRegistrations.html', {'form': form})
        else:
            messages.success(request, 'Email or Mobile Already Existed')
            print("Invalid form")
    else:
        form = UserRegistrationForm()
    return render(request, 'UserRegistrations.html', {'form': form})
def UserLoginCheck(request):
    if request.method == "POST":
        loginid = request.POST.get('loginname')
        pswd = request.POST.get('pswd')
        print("Login ID = ", loginid, ' Password = ', pswd)
        try:
            check = UserRegistrationModel.objects.get(loginid=loginid, password=pswd)
            status = check.status
            print('Status is = ', status)
            if status == "activated":
                request.session['id'] = check.id
                request.session['loggeduser'] = check.name
                request.session['loginid'] = loginid
                request.session['email'] = check.email
                print("User id At", check.id, status)
                return render(request, 'users/UserHome.html', {})
            else:
                messages.success(request, 'Your Account Not at activated')
                return render(request, 'UserLogin.html')
        except Exception as e:
            print('Exception is ', str(e))
            pass
        messages.success(request, 'Invalid Login id and password')
    return render(request, 'UserLogin.html', {})
def UserHome(request):
    return render(request, 'users/UserHome.html', {})
def user_view_dataset(request):
    path = os.path.join(settings.MEDIA_ROOT, 'parkinsons.csv')
    import pandas as pd
    df = pd.read_csv(path)
    df = df.drop(['name',
'MDVP:Fo(Hz)','MDVP:Fhi(Hz)','MDVP:Flo(Hz)','MDVP:Jitter(%)','MDVP:Jitter(Abs)','MDVP:
RAP','MDVP:PPQ','Jitter:DDP','spread1','spread2','MDVP:Shimmer'], axis=1)
```

```
        df = df.to_html
        return render(request, 'users/view_data.html', {'df': df})
    def user_model_evaluations(request):
        from .utility.parkinson_utility import start_models
        result = start_models()
        print(result)
        return render(request, 'users/model_results.html',result)
    def user_predict_form(request):
        if request.method == 'POST':
            nhr = float(request.POST.get('NHR'))
            hnr = float(request.POST.get('HNR'))
            rpde = float(request.POST.get('RPDE'))
            dfa = float(request.POST.get('DFA'))
            ppe = float(request.POST.get('PPE'))
            test_data = [nhr, hnr,rpde,dfa, ppe]
            from .utility import process_user_input
            test_pred = process_user_input.get_result(test_data)
            print("Test Result is:", test_pred)
            if test_pred[0] == 0:
                rslt = False
            else:
                rslt = True
            return render(request, "users/parkinson_form.html", {"test_data": test_data, "result": rslt})
        else:
            return render(request, 'users/parkinson_form.html', {})
```

forms.py

```
from django import forms
from .models import UserRegistrationModel


class UserRegistrationForm(forms.ModelForm):
    name = forms.CharField(widget=forms.TextInput(attrs={'pattern': '[a-zA-Z]+'}),
required=True, max_length=100)
    loginid = forms.CharField(widget=forms.TextInput(attrs={'pattern': '[a-zA-Z]+'}),
required=True, max_length=100)
    password = forms.CharField(widget=forms.PasswordInput(attrs={'pattern': '(?=.*\d)(?=.*[a-
z])(?=.*[A-Z]).{8,}',
 'title': 'Must contain at least one number and one uppercase and lowercase letter, and at least 8 or
more characters'}),
                               required=True, max_length=100)
    mobile = forms.CharField(widget=forms.TextInput(attrs={'pattern': '[56789][0-9]{9}'}),
required=True,
                             max_length=100)
    email = forms.CharField(widget=forms.TextInput(attrs={'pattern': '[a-z0-9._%+-]+@[a-z0-9.-
]+\.[a-z]{2,}$'}),
                            required=True, max_length=100)
    locality = forms.CharField(widget=forms.TextInput(), required=True, max_length=100)
    address = forms.CharField(widget=forms.Textarea(attrs={'rows': 4, 'cols': 22}), required=True,
max_length=250)
    city = forms.CharField(widget=forms.TextInput(
        attrs={'autocomplete': 'off', 'pattern': '[A-Za-z ]+', 'title': 'Enter Characters Only '}),
required=True,
        max_length=100)
    state = forms.CharField(widget=forms.TextInput(
        attrs={'autocomplete': 'off', 'pattern': '[A-Za-z ]+', 'title': 'Enter Characters Only '}),
required=True,
        max_length=100)
    status = forms.CharField(widget=forms.HiddenInput(), initial='waiting', max_length=100)

    class Meta():
        model = UserRegistrationModel
        fields = '__all__'
from django import forms
from .models import UserRegistrationModel


class UserRegistrationForm(forms.ModelForm):
    name = forms.CharField(widget=forms.TextInput(attrs={'pattern': '[a-zA-Z]+'}),
required=True, max_length=100)
    loginid = forms.CharField(widget=forms.TextInput(attrs={'pattern': '[a-zA-Z]+'}),
required=True, max_length=100)
```

```
    password = forms.CharField(widget=forms.PasswordInput(attrs={'pattern': '(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}',
 'title': 'Must contain at least one number and one uppercase and lowercase letter, and at least 8 or more characters'}),
                               required=True, max_length=100)
    mobile = forms.CharField(widget=forms.TextInput(attrs={'pattern': '[56789][0-9]{9}'}), required=True,
                             max_length=100)
    email = forms.CharField(widget=forms.TextInput(attrs={'pattern': '[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$'}),
                            required=True, max_length=100)
    locality = forms.CharField(widget=forms.TextInput(), required=True, max_length=100)
    address = forms.CharField(widget=forms.Textarea(attrs={'rows': 4, 'cols': 22}), required=True, max_length=250)
    city = forms.CharField(widget=forms.TextInput(
        attrs={'autocomplete': 'off', 'pattern': '[A-Za-z ]+', 'title': 'Enter Characters Only '}), required=True,
        max_length=100)
    state = forms.CharField(widget=forms.TextInput(
        attrs={'autocomplete': 'off', 'pattern': '[A-Za-z ]+', 'title': 'Enter Characters Only '}), required=True,
        max_length=100)
    status = forms.CharField(widget=forms.HiddenInput(), initial='waiting', max_length=100)

    class Meta():
        model = UserRegistrationModel
        fields = '__all__'
```

Model Building and results

```
import os

import pandas as pd
from django.conf import settings


path = os.path.join(settings.MEDIA_ROOT, 'parkinsons.csv')
a = pd.read_csv(path)
a.head()
a.shape
a.dtypes
# print(a.head())

# sns.catplot(x='status', kind='count', data=a)
# for i in a:
# if i != 'status' and i != 'name':
# sns.catplot(x='status', y=i, kind='box', data=a)
b = a.drop(['name'], axis=1)
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score

features = a.drop(['status', 'name'], axis=1)
labels = a['status']
scaler = MinMaxScaler((-1, 1))
x = scaler.fit_transform(features)
y = labels
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=5)


def start_models():
    from sklearn.linear_model import LogisticRegression
    from xgboost import XGBRFClassifier, XGBClassifier
    from sklearn.svm import SVC
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import AdaBoostClassifier, BaggingClassifier, ExtraTreesClassifier, \
        GradientBoostingClassifier, RandomForestClassifier
    lr = cross_val_score(LogisticRegression(), x_train, y_train)
    xgbc = cross_val_score(XGBRFClassifier(), x_train, y_train)
    xgb = cross_val_score(XGBClassifier(), x_train, y_train)
    svm = cross_val_score(SVC(), x_train, y_train)
    dtc = cross_val_score(DecisionTreeClassifier(), x_train, y_train)
    adb = cross_val_score(AdaBoostClassifier(), x_train, y_train)
    bbc = cross_val_score(BaggingClassifier(), x_train, y_train)
    etc = cross_val_score(ExtraTreesClassifier(), x_train, y_train)
```

```
gbc = cross_val_score(GradientBoostingClassifier(), x_train, y_train)
rfc = cross_val_score(RandomForestClassifier(), x_train, y_train)

print('log reg', lr, lr.mean())
print('xgbd', xgbc, xgbc.mean())
print('xgb', xgb, xgb.mean())
print('svm', svm, svm.mean())
print('dtc', dtc, dtc.mean())
print('adb', adb, adb.mean())
print('bbc', bbc, bbc.mean())
print('etc', etc, etc.mean())
print('gbc', gbc, gbc.mean())
print('rfc', rfc, rfc.mean())

model = XGBClassifier()
model.fit(x_train, y_train)

classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
print(classifier.summary())
classifier.fit(x_train, y_train, batch_size=10, nb_epoch=100)

y_pred = classifier.predict(x_test)
y_pred = (y_pred > 0.5)
ann_accuracy = accuracy_score(y_test, y_pred) * 100
accuracy_dict.update({'ann_accuracy': ann_accuracy})
return accuracy_dict
```

Predictions_user_input.py

```
import os

import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from django.conf import settings

path = os.path.join(settings.MEDIA_ROOT, 'parkinsons.csv')
a = pd.read_csv(path)
a = a[['NHR', 'HNR', 'RPDE', 'DFA', 'PPE', 'status']]
a.head()
a.shape
a.dtypes
# print(a.head())

sns.catplot(x='status', kind='count', data=a)
plt.show()
for i in a:
    if i != 'status' and i != 'PPE':
        sns.catplot(x='status', y=i, kind='box', data=a)
        plt.show()
# b = a.drop(['name'], axis=1)
data = a.drop('status', axis=1)
sns.heatmap(data.corr(), annot=True)
plt.show()
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

features = a.drop(['status'], axis=1)
labels = a['status']
scaler = MinMaxScaler((-1, 1))
x = scaler.fit_transform(features)
y = labels
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=5)
def get_result(test_set):
    from xgboost import XGBClassifier
    from sklearn.ensemble import RandomForestClassifier
    model = RandomForestClassifier()
    model.fit(x_train, y_train)
    scaler = MinMaxScaler((-1, 1))
    x = scaler.fit_transform([test_set])
    y_pred = model.predict(x)
    # y_pred = model.predict(x_test)
    print(y_pred)
    return y_pred
```

Base.html

```
{%load static%}
<!doctype html>
<html class="no-js" lang="en">
<head>
   <!-- META DATA -->
   <meta charset="utf-8">
   <meta content="IE=edge" http-equiv="X-UA-Compatible">
   <meta content="width=device-width, initial-scale=1" name="viewport">
   <link
href="https://fonts.googleapis.com/css?family=Playfair+Display:400,400i,700,700i,900,900i"
rel="stylesheet">
   <link
href="https://fonts.googleapis.com/css?family=Poppins:100,200,300,400,500,600,700,800,900"
rel="stylesheet">
   <link href="https://fonts.googleapis.com/css?family=Lato:100,300,400,700,900"
rel="stylesheet">
   <title>Parkinson</title>
   <link href="{%static 'images/logo/favicon.png'%}" rel="shortcut icon" type="image/icon"/>
   <link href="{%static 'css/font-awesome.min.css'%}" rel="stylesheet">
   <link href="https://cdn.linearicons.com/free/1.0.0/icon-font.min.css" rel="stylesheet">
   <link href="{%static 'css/animate.css'%}" rel="stylesheet">
   <link href="{%static 'css/hover-min.css'%}" rel="stylesheet">
   <link href="{%static 'css/magnific-popup.css'%}" rel="stylesheet">
   <link href="{%static 'css/owl.carousel.min.css'%}" rel="stylesheet">
   <link href="{%static 'css/owl.theme.default.min.css'%}" rel="stylesheet"/>
   <link href="{%static 'css/bootstrap.min.css'%}" rel="stylesheet">
   <link href="{%static 'css/bootsnav.css'%}" rel="stylesheet"/>
   <link href="{%static 'css/style.css'%}" rel="stylesheet">
   <link href="{%static 'css/responsive.css'%}" rel="stylesheet">
   <!--[if lt IE 9]>
   <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>
   <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
   <![endif]-->

</head>

<body>
<!--[if lte IE 9]>
<p class="browserupgrade">You are using an <strong>outdated</strong> browser. Please <a
href="https://browsehappy.com/">upgrade
   your browser</a> to improve your experience and security.</p>
<![endif]-->
```

```
<!--menu start-->
<section id="menu">
   <div class="container">
      <div class="menubar">
         <nav class="navbar navbar-default">
            <div class="navbar-header">
               <a class="navbar-brand" href="{%url 'index'%}">
                  <h2><font style="color:WHITE">Parkinson’s Disease</font></h2>
                  <!--<img src="{%static 'images/logo/logo.png'%}" alt="logo">-->
               </a>
            </div><!--/.navbar-header -->

            <!-- Collect the nav links, forms, and other content for toggling -->
            <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
               <ul class="nav navbar-nav navbar-right">
                  <li><a href="{%url 'index'%}">Home</a></li>
                  <li><a href="{%url 'AdminLogin'%}">Admin</a></li>
                  <li><a href="{%url 'UserLogin'%}">User</a></li>
                  <li><a href="{%url 'UserRegister'%}">Registrations</a></li>
               </ul><!-- / ul -->
            </div><!-- /.navbar-collapse -->
         </nav><!--/nav -->
      </div><!--/.menubar -->
   </div>

</section>
{%block contents%}
{%endblock%}

<footer class="footer-copyright">
   <div class="container">
      <div class="row">
         <div class="col-sm-7">
            <div class="foot-copyright pull-left">
               <p>
                  &copy; All Rights Reserved. Designed and Developed by
                  <a href="#">Alex Corp</a>
               </p>
            </div><!--/.foot-copyright-->
         </div><!--/.col-->
      </div><!--/.row-->
      <div id="scroll-Top">
         <i aria-hidden="true" class="fa fa-angle-double-up return-to-top" data-original-
title="Back to Top" data-placement="top"
              data-toggle="tooltip" id="scroll-top" title=""></i>
      </div><!--/.scroll-Top-->
```

```
    </div><!-- /.container-->

  </footer><!-- /.footer-copyright-->
  <!-- footer-copyright end -->

  <script src="{%static 'js/jquery.js'%}"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/modernizr/2.8.3/modernizr.min.js"
  type="text/javascript"></script>
  <script src="{%static 'js/bootstrap.min.js'%}" type="text/javascript"></script>
  <script src="{%static 'js/bootsnav.js'%}"></script>
  <script src="{%static 'js/jquery.hc-sticky.min.js'%}" type="text/javascript"></script>
  <script src="{%static 'js/jquery.magnific-popup.min.js'%}"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-easing/1.4.1/jquery.easing.min.js"
        type="text/javascript"></script>
  <script src="{%static 'js/owl.carousel.min.js'%}" type="text/javascript"></script>
  <script src="{%static 'js/jquery.counterup.min.js'%}"></script>
  <script src="{%static 'js/waypoints.min.js'%}"></script>
  <script src="{%static 'js/jak-menusearch.js'%}" type="text/javascript"></script>
  <script src="{%static 'js/custom.js'%}" type="text/javascript"></script>
  </body>
  </html>
```

CHAPTER 4

RESULTS & DISCUSSION

4.1 Performance Metrics

Confusion matrices represent counts from predicted and actual values. The output “TN” stands for True Negative which shows the number of negative examples classified accurately. Similarly, “TP” stands for True Positive which indicates the number of positive examples classified accurately. The term “FP” shows False Positive value, i.e., the number of actual negative examples classified as positive; and “FN” means a False Negative value which is the number of actual positive examples classified as negative.

		Predicted	
Actual		Negative	Positive
	Negative	TN	FP
	Positive	FN	TP

The accuracy of a model (through a confusion matrix) is calculated using the given formula below.

$$\text{Accuracy} = \frac{TN + TP}{TN + FP + FN + TP}$$

Accuracy can be misleading if used with imbalanced datasets, and therefore there are other metrics based on confusion matrix which can be useful for evaluating performance. Precision is a good measure to determine when the costs of False Positive is high. For instance, email spam detection. In email spam detection, a false positive means that an email that is nonspam (actual negative) has been identified as spam (predicted spam). The email user might lose important emails if the precision is not high for the spam detection model.

$$\text{Precision} = \frac{\text{True positive}}{\text{True Positive + False Positive}}$$

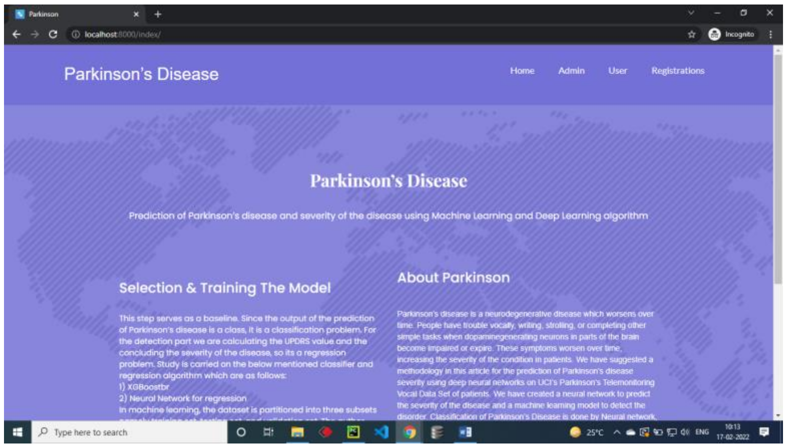
$$= \frac{\text{True Positive}}{\text{Total Prediction Positive}}$$

Recall calculates how many of the Actual Positives our model capture through labelling it as Positive (True Positive). Applying the same understanding, we know that Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative.

$$F1 = 2 \times \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1 Score is needed when you want to seek a balance between Precision and Recall. Right…so what is the difference between F1 Score and Accuracy then? We have previously seen that accuracy can be largely contributed by a large number of True Negatives which in most business circumstances, we do not focus on much whereas False Negative and False Positive usually has business costs (tangible & intangible) thus F1 Score might be a better measure to use if we need to seek a balance between Precision and Recall AND there is an uneven class distribution (large number of Actual Negatives).

RESULTS



Parkinson’s Disease

Prediction of Parkinson’s disease and severity of the disease using Machine Learning and Deep Learning algorithm

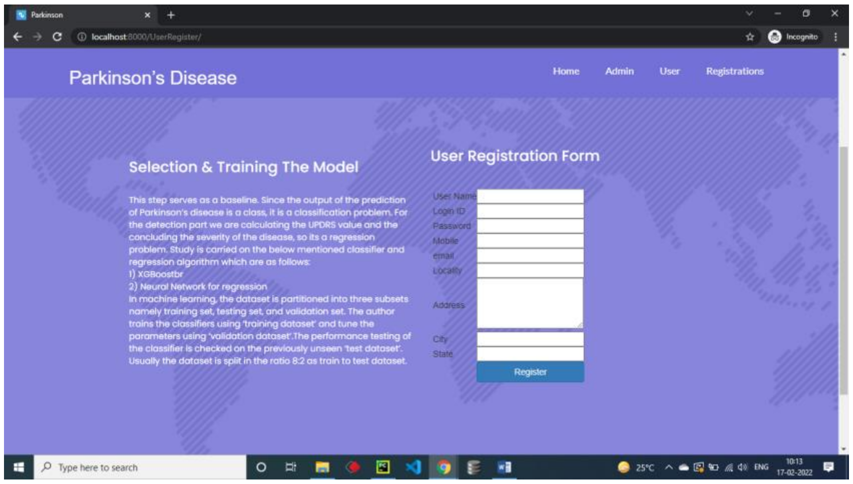
Selection & Training The Model

This step serves as a baseline. Since the output of the prediction of Parkinson’s disease is a class, it is a classification problem. For the detection part we are calculating the UPDRS value and the concluding the severity of the disease, so its a regression problem. Study is carried on the below mentioned classifier and regression algorithm which are as follows:
1) XGBoostbr
2) Neural Network for regression
In machine learning, the dataset is partitioned into three subsets

About Parkinson

Parkinson’s disease is a neurodegenerative disease which worsens over time. People have trouble vocally, writing, strolling, or completing other simple tasks when dopaminegenerating neurons in parts of the brain become impaired or expire. These symptoms worsen over time, increasing the severity of the condition in patients. We have suggested a methodology in this article for the prediction of Parkinson’s disease severity using deep neural networks on UCI’s Parkinson’s Telemonitoring Vocal Data Set of patients. We have created a neural network to predict the severity of the disease and a machine learning model to detect the disorder. Classification of Parkinson’s Disease is done by Neural network,

4.1.1 Home Page



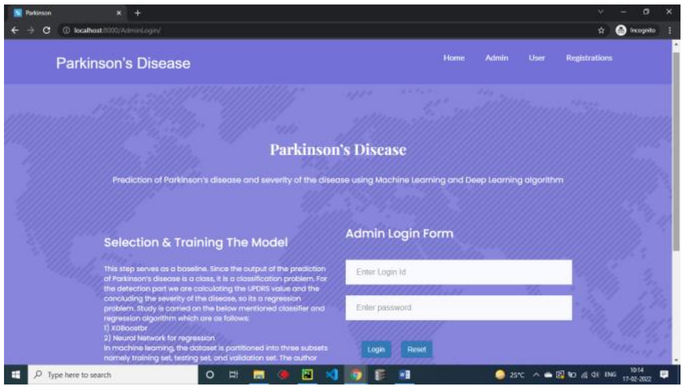
Selection & Training The Model

This step serves as a baseline. Since the output of the prediction of Parkinson’s disease is a class, it is a classification problem. For the detection part we are calculating the UPDRS value and the concluding the severity of the disease, so its a regression problem. Study is carried on the below mentioned classifier and regression algorithm which are as follows:
1) XGBoostbr
2) Neural Network for regression
In machine learning, the dataset is partitioned into three subsets namely training set, testing set, and validation set. The author trains the classifiers using ‘training dataset’ and tune the parameters using ‘validation dataset’.The performance testing of the classifier is checked on the previously unseen ‘test dataset’. Usually the dataset is split in the ratio 8:2 as train to test dataset.

User Registration Form

User Name	
Login ID	
Password	
Mobile	
email	
Locality	
Address	
City	
State	

4.1.2 User Registration Form


Parkinson's Disease

Home Admin User Registrations

Parkinson's Disease

Prediction of Parkinson's disease and severity of the disease using Machine Learning and Deep Learning algorithm

Selection & Training The Model

This step serves as a baseline. Since the output of the prediction of Parkinson's disease is a class, it is a classification problem. For the detection part we are calculating the UPDRS value and the concluding the severity of the disease, so its a regression problem. Study is carried on the below mentioned classifier and regression algorithm which are as follows:
1) XGBoostbr
2) Neural Network for regression
In machine learning, the dataset is partitioned into three subsets namely training set, testing set, and validation set. The author

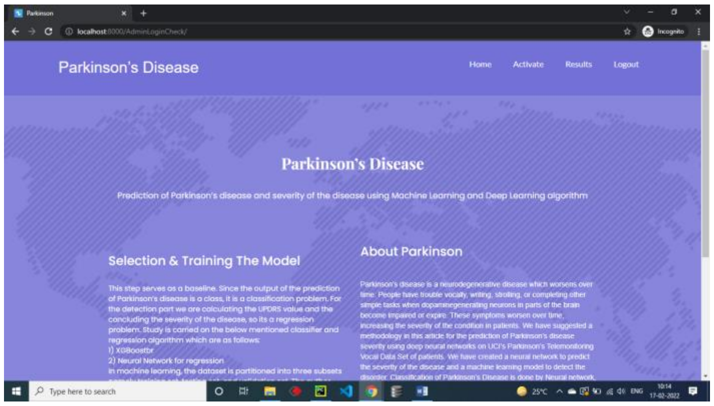
Admin Login Form

Enter Login Id

Enter password

Login Reset

4.1.3 Admin Login Form


Parkinson's Disease

Home Activate Results Logout

Parkinson's Disease

Prediction of Parkinson's disease and severity of the disease using Machine Learning and Deep Learning algorithm

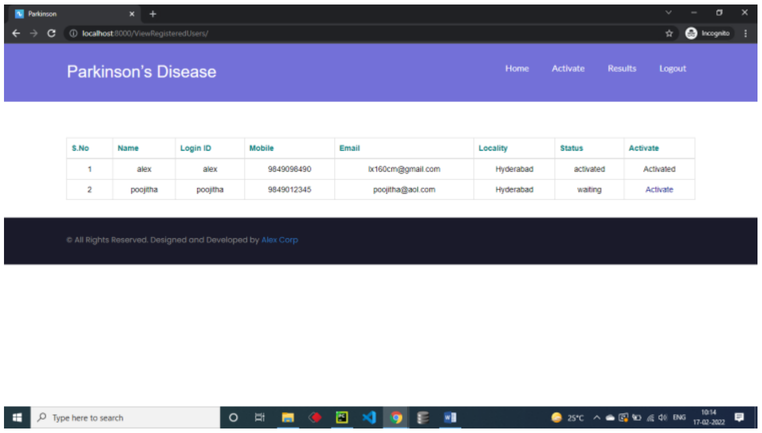
Selection & Training The Model

This step serves as a baseline. Since the output of the prediction of Parkinson's disease is a class, it is a classification problem. For the detection part we are calculating the UPDRS value and the concluding the severity of the disease, so its a regression problem. Study is carried on the below mentioned classifier and regression algorithm which are as follows:
1) XGBoostbr
2) Neural Network for regression
In machine learning, the dataset is partitioned into three subsets

About Parkinson

Parkinson's disease is a neurodegenerative disease which worsens over time. People have trouble vocally, writing, strolling, or completing other simple tasks when dopaminegenerating neurons in parts of the brain become impaired or expire. These symptoms worsen over time, increasing the severity of the condition in patients. We have suggested a methodology in this article for the prediction of Parkinson's disease severity using deep neural networks on UCI's Parkinson's Telemonitoring Vocal Data Set of patients. We have created a neural network to predict the severity of the disease and a machine learning model to detect the disorder. Classification of Parkinson's Disease is done by Neural network.

4.1.4 Admin Home Page

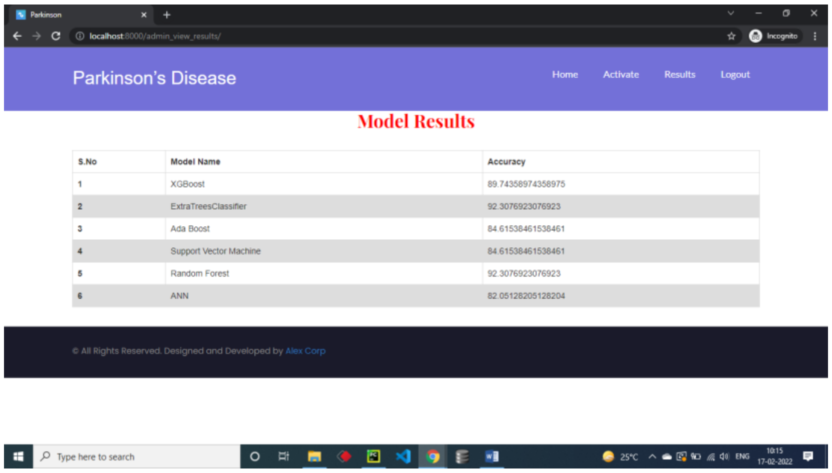

Parkinson's Disease

Home Activate Results Logout

S.No	Name	Login ID	Mobile	Email	Locality	Status	Activate
1	alex	alex	9849098490	lx160cm@gmail.com	Hyderabad	activated	Activated
2	poojitha	poojitha	9849012345	poojitha@aol.com	Hyderabad	waiting	Activate

© All Rights Reserved. Designed and Developed by Alex Corp

4.1.5 Admin View User & Activate

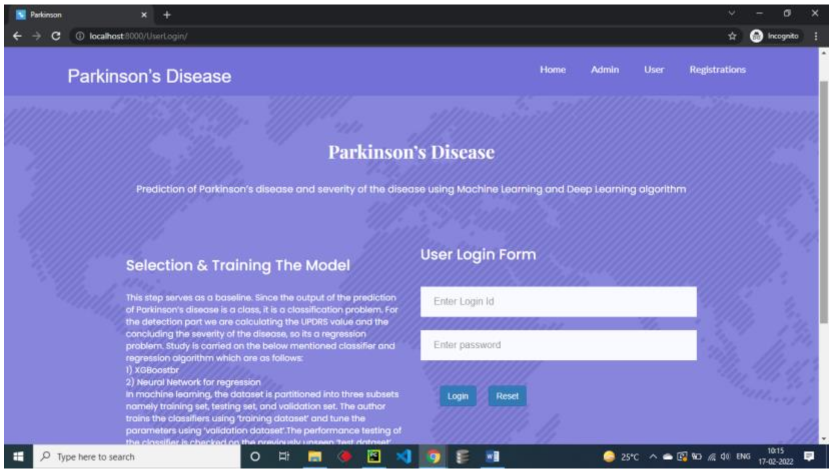


Model Results

S.No	Model Name	Accuracy
1	XGBoost	89.74358974358975
2	ExtraTreesClassifier	92.3076923076923
3	Ada Boost	84.61538461538461
4	Support Vector Machine	84.61538461538461
5	Random Forest	92.3076923076923
6	ANN	82.05128205128204

© All Rights Reserved. Designed and Developed by Alex Corp

4.1.6 Admin View Results



Parkinson's Disease

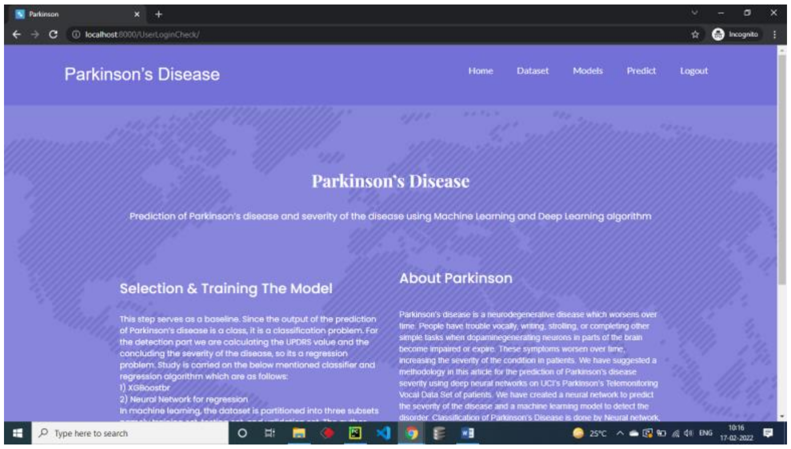
Prediction of Parkinson's disease and severity of the disease using Machine Learning and Deep Learning algorithm

Selection & Training The Model

This step serves as a baseline. Since the output of the prediction of Parkinson's disease is a class, it is a classification problem. For the detection part we are calculating the UPDRS value and the concluding the severity of the disease, so its a regression problem. Study is carried on the below mentioned classifier and regression algorithm which are as follows:
1) XGBoostbr
2) Neural Network for regression
In machine learning, the dataset is partitioned into three subsets namely training set, testing set, and validation set. The author trains the classifiers using 'training dataset' and tune the parameters using 'validation dataset'.The performance testing of the classifier is checked on the previously unseen 'test dataset'

User Login Form

4.1.7 User Login Form



Parkinson's Disease

Prediction of Parkinson's disease and severity of the disease using Machine Learning and Deep Learning algorithm

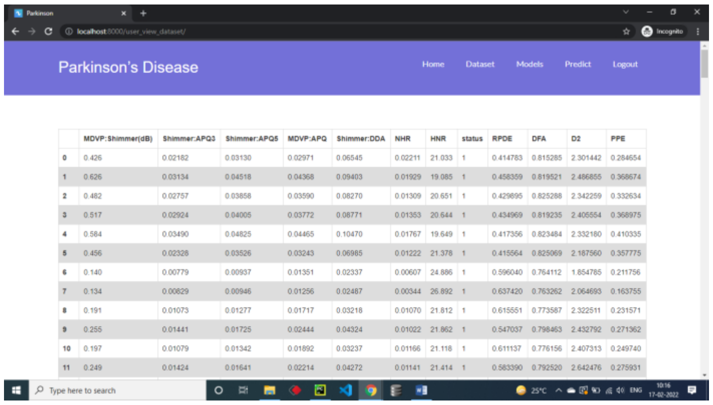
Selection & Training The Model

This step serves as a baseline. Since the output of the prediction of Parkinson's disease is a class, it is a classification problem. For the detection part we are calculating the UPDRS value and the concluding the severity of the disease, so its a regression problem. Study is carried on the below mentioned classifier and regression algorithm which are as follows:
1) XGBoostbr
2) Neural Network for regression
In machine learning, the dataset is partitioned into three subsets

About Parkinson

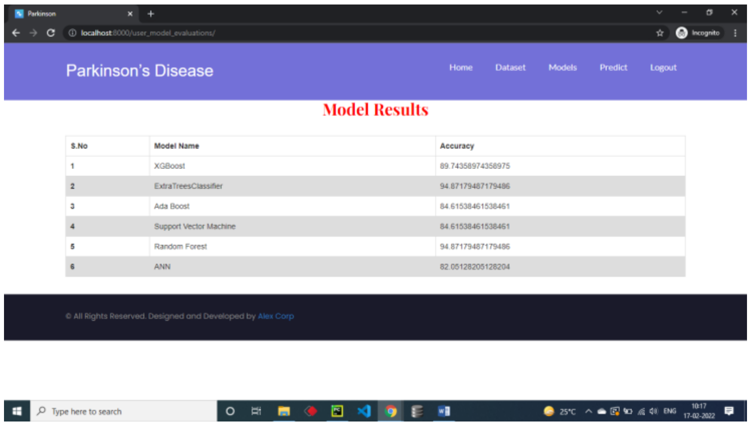
Parkinson's disease is a neurodegenerative disease which worsens over time. People have trouble vocally, writing, strolling, or completing other simple tasks when dopaminegenerating neurons in parts of the brain become impaired or expire. These symptoms worsen over time, increasing the severity of the condition in patients. We have suggested a methodology in this article for the prediction of Parkinson's disease severity using deep neural networks on UCI's Parkinson's Telemonitoring Vocal Data Set of patients. We have created a neural network to predict the severity of the disease and a machine learning model to detect the disorder. Classification of Parkinson's Disease is done by Neural network.

4.1.8 User Home Page



	MDVP:Shimmer(dB)	Shimmer:APQ3	Shimmer:APQ5	MDVP:APQ	Shimmer:DDA	NHR	HNR	status	RPDE	DFA	D2	PPE
0	0.426	0.02182	0.03130	0.02971	0.06545	0.02211	21.033	1	0.414783	0.815285	2.301442	0.284654
1	0.626	0.03134	0.04518	0.04368	0.09403	0.01929	19.085	1	0.458359	0.819521	2.486855	0.368674
2	0.482	0.02757	0.03858	0.03590	0.08270	0.01309	20.651	1	0.429895	0.825288	2.342259	0.332634
3	0.517	0.02924	0.04005	0.03772	0.08771	0.01353	20.644	1	0.434969	0.819235	2.405554	0.368975
4	0.584	0.03490	0.04825	0.04465	0.10470	0.01767	19.649	1	0.417356	0.823484	2.332180	0.410335
5	0.456	0.02328	0.03526	0.03243	0.06985	0.01222	21.378	1	0.415564	0.825069	2.187560	0.357775
6	0.140	0.00779	0.00937	0.01351	0.02337	0.00607	24.886	1	0.596040	0.764112	1.854785	0.211756
7	0.134	0.00829	0.00946	0.01256	0.02487	0.00344	26.892	1	0.637420	0.763262	2.064693	0.163755
8	0.191	0.01073	0.01277	0.01717	0.03218	0.01070	21.812	1	0.615551	0.773587	2.322511	0.231571
9	0.255	0.01441	0.01725	0.02444	0.04324	0.01022	21.862	1	0.547037	0.798463	2.432792	0.271362
10	0.197	0.01079	0.01342	0.01892	0.03237	0.01166	21.118	1	0.611137	0.776156	2.407313	0.249740
11	0.249	0.01424	0.01641	0.02214	0.04272	0.01141	21.414	1	0.583390	0.792520	2.642476	0.275931

4.1.9 User View Dataset

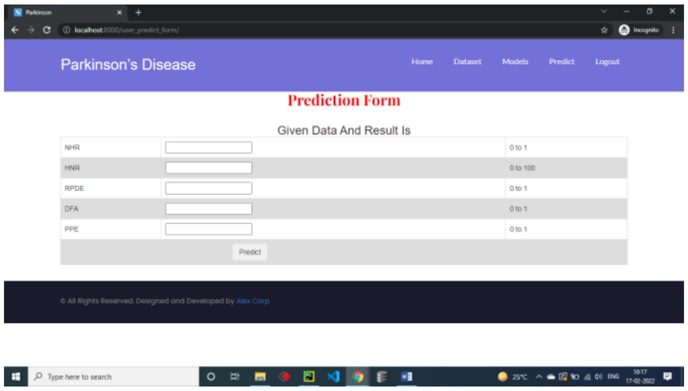


Model Results

S.No	Model Name	Accuracy
1	XGBoost	89.74358974358975
2	ExtraTreesClassifier	94.87179487179486
3	Ada Boost	84.61538461538461
4	Support Vector Machine	84.61538461538461
5	Random Forest	94.87179487179486
6	ANN	82.05128205128204

© All Rights Reserved. Designed and Developed by Alex Corp

4.1.10 User View Model Results



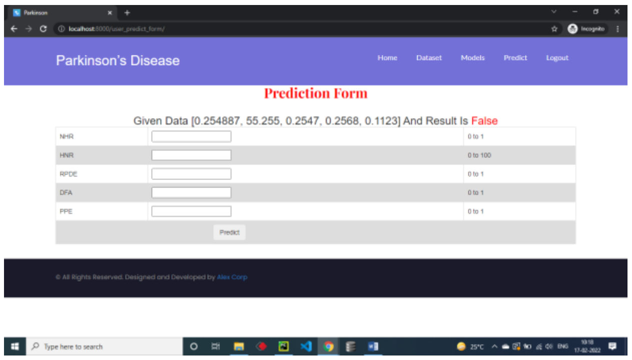
Prediction Form

Given Data And Result Is

NHR		0 to 1
HNR		0 to 100
RPDE		0 to 1
DFA		0 to 1
PPE		0 to 1

© All Rights Reserved. Designed and Developed by Alex Corp

4.1.11 Prediction Form



Prediction Form

Given Data [0.254887, 55.255, 0.2547, 0.2568, 0.1123] And Result Is False

NHR		0 to 1
HNR		0 to 100
RPDE		0 to 1
DFA		0 to 1
PPE		0 to 1

4.1.12 Prediction Results

PARKINSON’S DISEASE PREDICTION RESULTS:

The metrics module implements functions assessing prediction errors for specific purposes. By using metrics module from Sklearn library on the trained model, the performance of the model can be computed by calculating performance metrics such as confusion matrix, precision, recall, f1 score, etc.

Confusion matrix of the Parkinson’s Disease prediction model is computed and following are the attained results:

Precision is the ability of a classifier not to label an instance positive that is negative. For each class it is defined as the ratio of true positives to the sum of true and false positives. Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives. The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0.

CHAPTER 5
CONCLUSION

5.1 CONCLUSION

Due to lack of robust prognosis models results in difficulty for doctors to identify a patient is suffering from Parkinson’s disease. So, in such a scenario, our project will be useful to predict whether a person is suffering from Parkinson’s disease. With the Parkinson dataset, we got higher accuracy for XGBoost model. From this project’s results, we can conclude that we can predict the Parkinson’s diseases with accuracy of 90 % or more.

5.2 FUTURE ENHANCEMENT

To predict the different stages and severity of Parkinson’s disease in a human.

CHAPTER 6
REFERENCES

6.1 REFERENCES

[1] National Institute for Health and Care Excellence (NICE). Parkinson’s disease: diagnosis and management in primary and secondary care. NICE clinical guidelines 35. June2006.Available http://www.nice.org.uk/guidance/cg35/resources/guidanceparkinsons-disease-pdf.Accessed April 28, 2015.

[2] Parkinson J. An Essay on the Shaking Palsy. London: Sherwood, Neely, and Jones;1817: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4517533/pdf/ptj4008504.pdf

[3] https://easychair.org/publications/preprint_open/NQlD

[4] Early diagnosis of Parkinson’s disease using machine learning algorithms Zehra Karapinar Senturk Duzce University, Engineering Faculty, Department of Computer Engineering, 81620 Duzce, Turkey

[5] A. Tsanas, et al., "Novel speech signal processing algorithms for high-accuracy classification of Parkinson's disease," IEEE Transactions on biomedical engineering, vol. 59, pp. 1264-1271, January 2012.

[6] http://ijariie.com/AdminUploadPdf/EARLY_DETECTION_OF_PARKINSON_S_DISEASE_USING_MACHINE_LEARNING_ijariie11591.pdf

GitHub Link:

https://github.com/Ganendhar/parkinson-s-disease-prediction.git