Final Project report

Applied Data Science

Team ID	PNT2022TMID00582
Team Members	V.l.sathwika gurram Sherin.s Suryaprabha s Sharan jothikala J
Project name	Detecting Parkinson's Disease using Machine learning

1. INTRODUCTION

1.1 Project Overview

Parkinson's disease is a progressive disorder of the central nervous system affecting movement and inducing tremors and stiffness. It has 5 stages to it and affects more than 1 million individuals every year in India. This is chronic and has no cure yet. It is a neurodegenerative disorder affecting dopamine-producing neurons in the brain. For detecting PD, various machine learning models such as logistic regression, naive Bayes, KNN, and forest decision tree were used, with the features used here being minimum-redundancy maximum-relevance and recursive feature elimination. The accuracy obtained was 95.3% using data from the UCI machine learning repository. The researchers found that the drawing speed was slower and the pen pressure is lower among Parkinson's patients. One of the indications of Parkinson's is tremors and rigidity in the muscles, making it difficult to draw smooth spirals and waves. It is possible to detect Parkinson's disease using the drawings alone instead of measuring the speed and pressure of the pen on paper. Our goal is to quantify the visual appearance (using HOG method) of these drawings and then train a machine learning model to classify them. In this project, We are using, Histogram of Oriented Gradients (HOG) image descriptor along with a Random Forest classifier to automaticallydetect Parkinson's disease in hand-drawn images of spirals and waves.

1.2 Purpose

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. Also, our proposed system provides accurate results by integrating spiral drawing inputs of normal and Parkinson's affected patients. Machine learning also allows for combining different modalities, such as magnetic resonance imaging (MRI) and single-photon emission computed tomography (SPECT) data. in the diagnosis of PD. By using machine learning approaches, we may therefore identify relevant features that are not traditionally used in the clinical diagnosis of PD and rely on these alternative measures to detect PD in preclinical stages or atypical forms. In recent years, the number of publications on the application of machine learning to the diagnosis of PD has increased.

feasibility and efficiency of different machine learning methods in the diagnosis of PD, and (c) provide machine learning practitioners interested in the diagnosis of PD with an overview of previously used models and data modalities and the associated outcomes, and recommendations on how experimental protocols and results could be reported to facilitate reproduction. As a result, the application of machine learning to clinical and non-clinical data of different modalities has often led to high diagnostic accuracies in human participants, therefore may encourage the adaptation of machine learning algorithms and novel biomarkers in clinical settings to assist more accurate and informed decision making. While Parkinson's cannot be cured, early detection along with proper medication can significantly improve symptoms and quality of life.

2. LITERATURE SURVEY

2.1 Existing problem

More than 10 million people are living with Parkinson's Disease worldwide, according to the Parkinson's Foundation. While Parkinson's cannot be cured, early detection along with proper medication can significantly improve symptoms and quality of life. The researchers found that the drawing speed was slower and the pen pressure is lower among Parkinson's patients. One of the indications of Parkinson's is tremors and rigidity in the muscles, making it difficult to draw smooth spirals and waves. It is possible to detect Parkinson's disease using the drawings alone instead of measuring the speed and pressure of the pen on paper.

2.2 References

Literature Survey

Detecting Parkinson's Disease using Machine Learning

1. Jie Mei, Christian Desrosiers, Johannes Frasnelli, "Machine Learning for the Diagnosis of Parkinson's Disease," 2021.

This paper conveys extremely about the importance of Diagnosis of Parkinson's disease (PD) is commonly based on medical observations and assessment of clinical signs, including the characterization of a variety of motor symptoms. However, traditional diagnostic approaches may suffer from subjectivity as they rely on the evaluation of movements that are sometimes subtle to human eyes and therefore difficult to classify, leading to possible misclassification. In the meantime, early non-motor symptoms of PD may be mild and can be caused by many other conditions. Therefore, these symptoms are often overlooked, making diagnosis of PD at an early stage challenging. To address these difficulties and to refine the diagnosis and assessment procedures of PD, machine learning methods have been implemented for the classification of PD and healthy controls or patients with similar clinical presentations (e.g., movement disorders).

2. C K Gomathy, "The Parkinson's Disease Detection using Machine Learning Techniques." 2021.

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 behavior 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 this

3. Iqra Nissar, Waseem Ahmad Mir, Izharuddin, Tawseef Ayoub Shaikh, "Machine Learning Approaches for Detection and Diagnosis of Parkinson's Disease," 2021.

Parkinson's disease (PD) is disabling disease that affects the quality of life. It happens due to the death of cells that produce dopamine's in the substantia nigra part of the central nervous system (CNS) which affects the human body. People who have Parkinson's disease feel difficulty in doing activities like speaking, writing, and walking. However, speech analysis is the most considered technique to be used. Researches have shown that 90% of the people who suffer from Parkinson's disease have speech disorders. With the increase in the severity of the disease, the patient's voice gets more and more deteriorated. The proper interpretation of speech signals is one of the important classification problems for Parkinson's disease diagnosis. This paper contemplates the survey work of the machine learning techniques and deep learning procedures used for Parkinson's disease classification.

4.Radouani Laila, Lagdali Salwa, Rziza Mohammed, "Detection of voice impairment for parkinson's disease using machine learning tools," 2021.

In this paper, it proposes that Parkinson's disease (PD) is disabling disease that affects the quality of life. It happens due to the death of cells that produce dopamine's in the substantia nigra part of the central nervous system (CNS) which affects the human body. People who have Parkinson's disease feel difficulty in doing activities like speaking, writing, and walking. Speech analysis is the most considered technique to be used. Researches have shown that 90% of the people who suffer from Parkinson's disease have speech disorders. With the increase in the severity of the disease, the patient's voice gets more and more deteriorated. The proper interpretation of speech signals is one of the important classification problems for Parkinson's disease diagnosis. The main purpose of this paper is to contemplate the survey work of the machine learning techniques and deep learning procedures used for Parkinson's disease classification.

5.Zehra Karapinar Senturk, "Early diagnosis of Parkinson's disease using machine learning algorithms," 2020.

Parkinson's disease is caused by the disruption of the brain cells that produce substance to allow brain cells to communicate with each other, called dopamine. The cells that produce dopamine in the brain are responsible for the control, adaptation, and fluency of movements. When 60–80% of these cells are lost, then enough dopamine is not produced and Parkinson's motor symptoms appear. It is thought that the disease begins many years before the motor (movement related) symptoms and therefore, researchers are looking for ways to recognize the non-motor symptoms that appear early in the disease as early as possible, thereby halting the progression of the disease. In this paper, machine learning based diagnosis of Parkinson's disease is presented. The proposed diagnosis method consists of feature selection and classification processes.

2.3 Problem Statement Definition

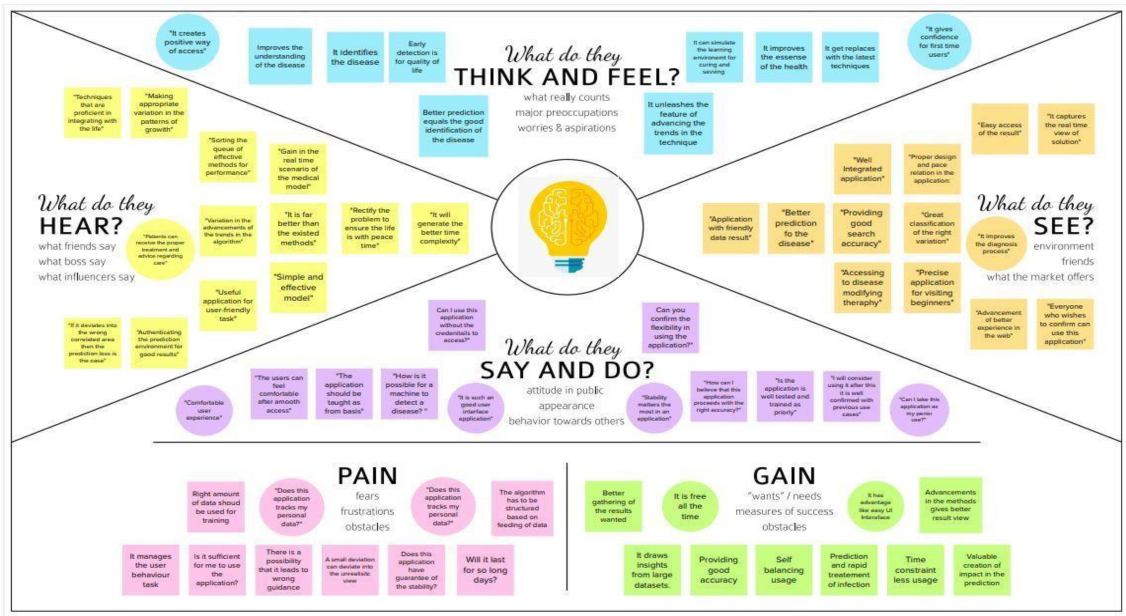
Lack of adequate knowledge poses a barrier in the provision of appropriate treatment and care for individuals with Parkinson's Disease. We had conducted a important survey between rural and urban areas in which we found that 68% of rural people from agricultural field are getting majorly affected by Parkinson's disease whereas 32% of urban people are affected by the disease with the ages over 50. We further researched and analyzed the data that was gathered from all over the network for figuring out the accurate reason for why this disease majorly affects the agricultural life.

Disease identification application for disease prediction

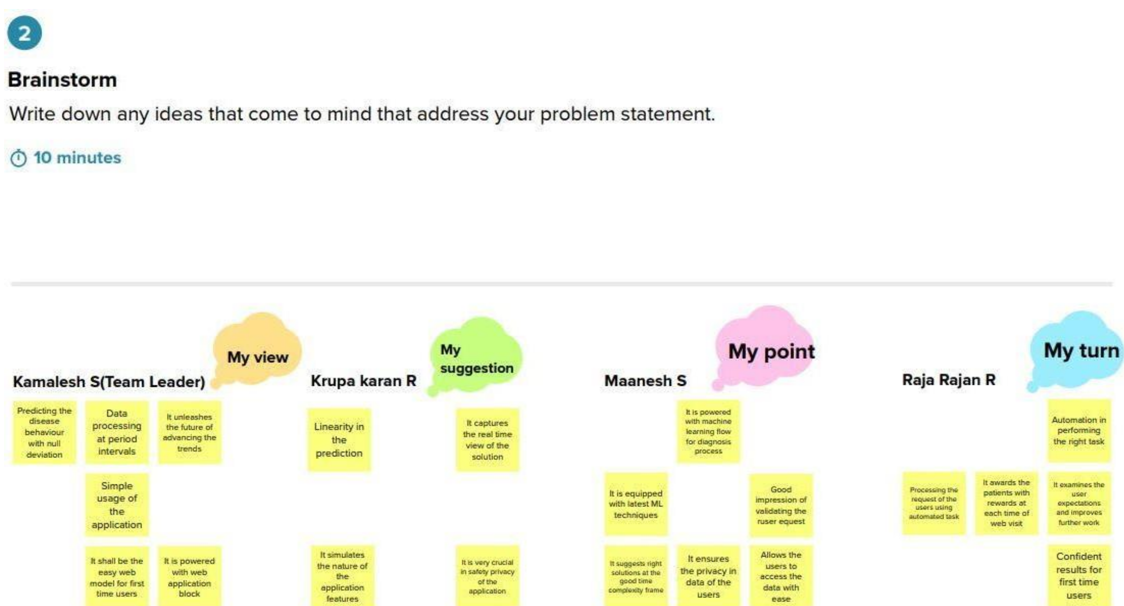
Who does the problem affect?	People who are men with minimization of nerve cells in primarily of village areas.
What are the boundaries of the problem?	People who are men with weak nerve cells and age over 50
What is the issue?	In real time life of human, if the person is affected by Parkinson disease then it produces the side effect problems like dry skin and dandruff which majorly affects the quality of the life. As the age gets progresses, it causes the people to face major problem with the nerve cells in the brain.
When does the issue occur?	During the age excess of over 50 as they will affect the people with loss of nerve cells in the brain.
Where is the issue coming?	It majorly occurs due to the age getting over 50 and as maximum in village areas.
Why is it important that we fix the problem?	It is very crucial to develop a application that detects the disease at good prediction rate so that it helps to get a clear line of disease symptoms during the times.
Which solution can be used to address this issue?	An machine learning powered web application model with the strong building of algorithm that helps to identify and predicts the disease with the identification of symptoms. It processes the breathing signals using a neural network that infers whether the person has Parkinson's disease, and if they are identified then it assesses the severity of their disease in accordance with the Movement Disorder Society Unified Parkinson's Disease using ML algorithms.
What methodology used to solve the issue?	Supervised and Un-supervised machine learning, Data mining , Computer vision with OpenCV, Python web application interface – Flask , IBM Cloud.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

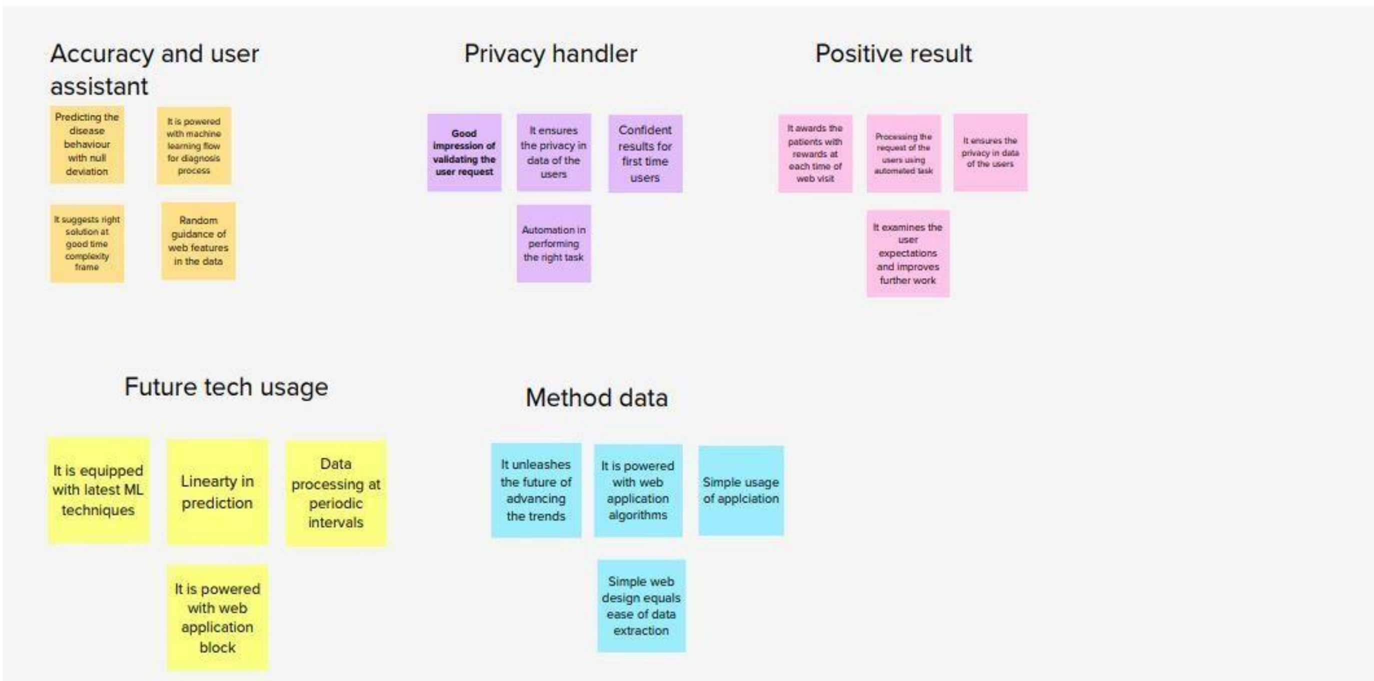


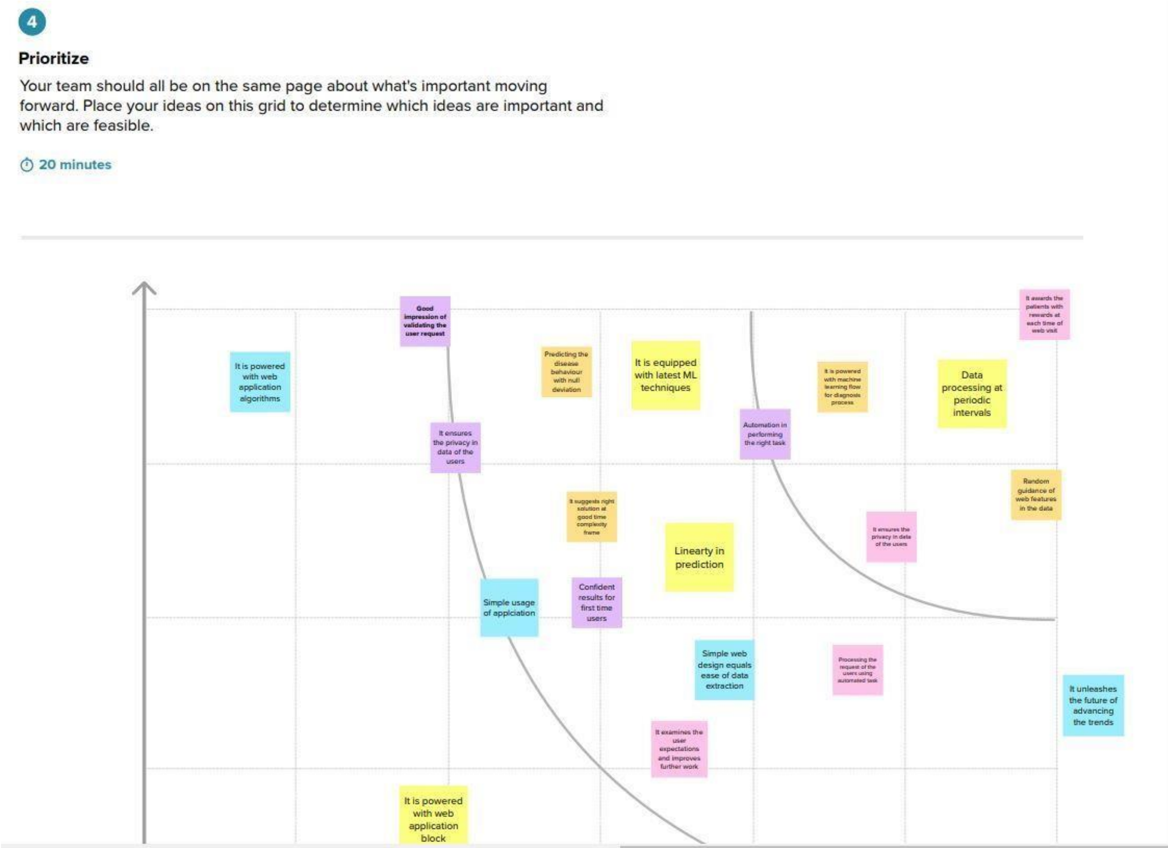
3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes





3.3 Proposed Solution

Proposed Solution:

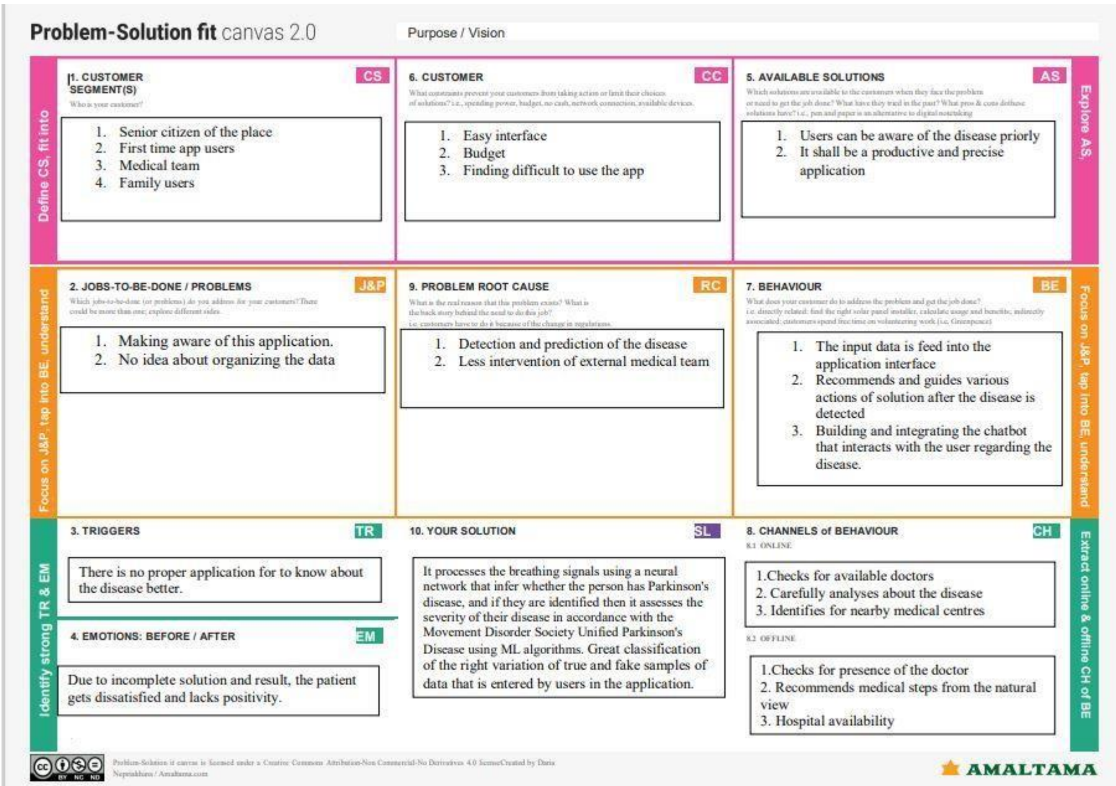
S.No.	Parameter	Description
1	Problem Statement (Problem to be solved)	Parkinson's disease (PD) is a neurodegenerative movement disease where the symptoms gradually develop start with a slight tremor in one hand and a feeling of stiffness in the body and it became worse over time. It affects over 6 million people worldwide. At present there is no conclusive result for this disease by non-specialist clinicians, particularly in the early stage of the disease where identification of the symptoms is very difficult in its earlier stages. The disease is majorly is said to be affecting the individuals who are living in village areas with their respective ages over 40 and 50 which outcomes itself as a reason for Parkinson's disease to occur at unexpected times. Lack of adequate knowledge poses a barrier in the provision of appropriate treatment and care for individuals with Parkinson's Disease. We had conducted a important survey between rural and urban areas in which we found that 68% of rural people from agricultural field are getting majorly affected by Parkinson's disease whereas 32% of urban people are affected by the disease with the ages over 50. We further researched and analyzed the data that was gathered from all over the network for figuring out the accurate reason for why this disease majorly affects the agricultural life. So, we found that as Parkinson's disease is believed to be caused by a combination of environmental risk factors and genetic susceptibility. As use of pesticides and Parkinson's disease have

2	Idea / Solution description	• It processes the breathing signals using a neural network that infer whether the person has Parkinson's disease, and if they are identified then it assesses the severity of their disease in accordance with the Movement Disorder Society Unified Parkinson's Disease using ML algorithms. • User can place their values and interact with the friendly user assistance bot which guides the person in using the application. • Great classification of the right variation of true and fake samples of data that is entered by users in the application.

3	Novelty / Uniqueness	Parkinson's Disease is detected at the secondary stage only (Dopamine deficiency) which leads to medical challenges. Also, doctor must manually examine and suggest medical diagnosis in which the symptoms might vary from person to person so suggesting medicine is also a challenge. So hence the disease examination varies at different instances of the medical operations. Here by using machine learning methods,

4	Social Impact / Customer Satisfaction	• Increases interaction with the human and application • Personalize the UI experience • Improves accurate result as expected • An automated chatbot controls the user interaction environment • Accurate prediction at good time complexity.
5	Business Model (Revenue Model)	• Solutions prospects of improvement • Suits for better saving of involvements • Economical Development • Easy interface
6	Scalability of the Solution	• Good conversation with ethnicity people. • Saves enough time for performing internal operations. • It does not require for the users to spend some money in offering their basic data into the model. • On the spot result for the users.

3.4 Problem Solution fit



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User account registration	Registration through Google account and forms
FR-2	Input data	Application received the data and processes its roles
FR-2	User Authorization	Verifying the user’s account
FR-3	Data classification	Classification of the real data for the user
FR-4	Accuracy verification	Accuracy is determined in the application
FR-5	Time efficient usage	Interaction with the chatbot till the result gets generated for the user
FR-6	Medical recommendations	User receives the medical suggestions and assistance for to offer speed
FR-7	Data extraction	User gets their personal disease report data from the application

4.2 Non-Functional requirements

Non-functional Requirements:

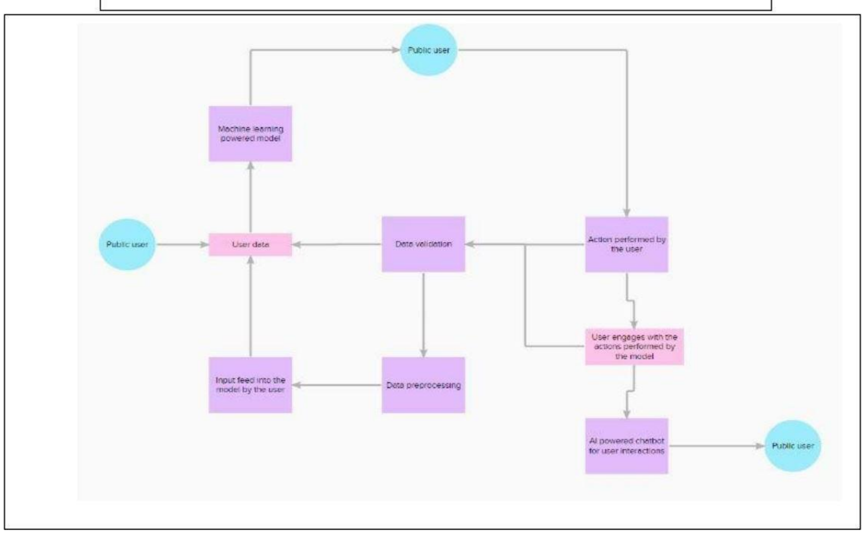
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The application can be used for accurate prediction and classifier of the true and fake input data sample
NFR-2	**Security**	User's data is well encrypted using stable machine learning algorithms
NFR-3	**Reliability**	The application is monitored periodically in terms of its constant prediction ability, quality, and availability towards the user
NFR-4	**Performance**	It classifies the images and predicts the disease with careful accuracy output
NFR-5	**Availability**	The application is active throughout the day. While awaiting the prediction result, User can interact with the chatbot for to spend time in knowing important details. If the application doesn't respond for the user, then the automated chatbot will forward the issue to our server then it can be resolved at that instance.
NFR-6	**Scalability**	It does not request money or bank details to setup their account and download their final medical result from the application

5. PROJECT DESIGN

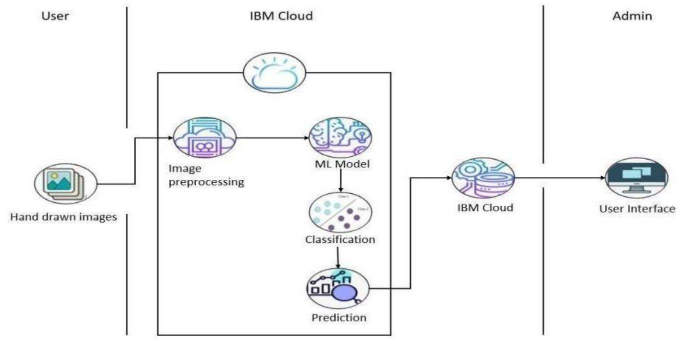
5.1 Data Flow Diagrams

Data flow diagram – Detecting Parkinson's Disease using Machine Learning



5.2 Solution & Technical Architecture

Technical Architecture:



5.3 User Stories

User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Public user)	Account creation	USN-1	As a user, I can connect my google into the application	I can access my account / application dashboard	High	Sprint-1
Input data	Adding data	USN-2	As a user, I can feed my data as the input into the application for it to classify the true fake data	I can cross verify the data that entered in the initial step	High	Sprint-1
Data validation	Checking accuracy	USN-3	As a user, I can check the ability and accuracy of the model in obtaining the required information	I can log into my account and check the capability of the model	Medium	Sprint-2
Classification	Data classification	USN-4	As a user, I can view the real data	I can verify my data with the real data	Medium	Sprint-2
App work	Work flow	USN-5	As a user, I can examine the working action of the application model	I can view how the application works and responds to the actions imposed	High	Sprint-2
Image classification	Checking for the disease	USN-6	As a user, I can verify with the application that the image is identified with the actual disease with the help of the trained and tested data's	I can confirm that the data shows the accurate result	Low	Sprint-3
User interaction	AI-powered chatbot	USN-7	As a user, I can interact with the automated chatbot to engage my time till the application processed the accurate result in a meanwhile	I can see the results from the interaction with the chatbot	Low	Sprint-3
Medical assistance	Medical suggestions	USN-8	As a user, I can get medical advises and recommendations for to boost the action of curing the disease	I can get enough assistance by getting the suggestions for curing the disease	High	Sprint-3
Data extraction	Obtaining the data	USN-9	As a user, I can retrieve the result data from the application for data storage for further medical research uses.	I can download the result in the form of data as a proof to show to medical teams	Medium	Sprint-4

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

User Stories

Use the below template to list all the user stories for the product.

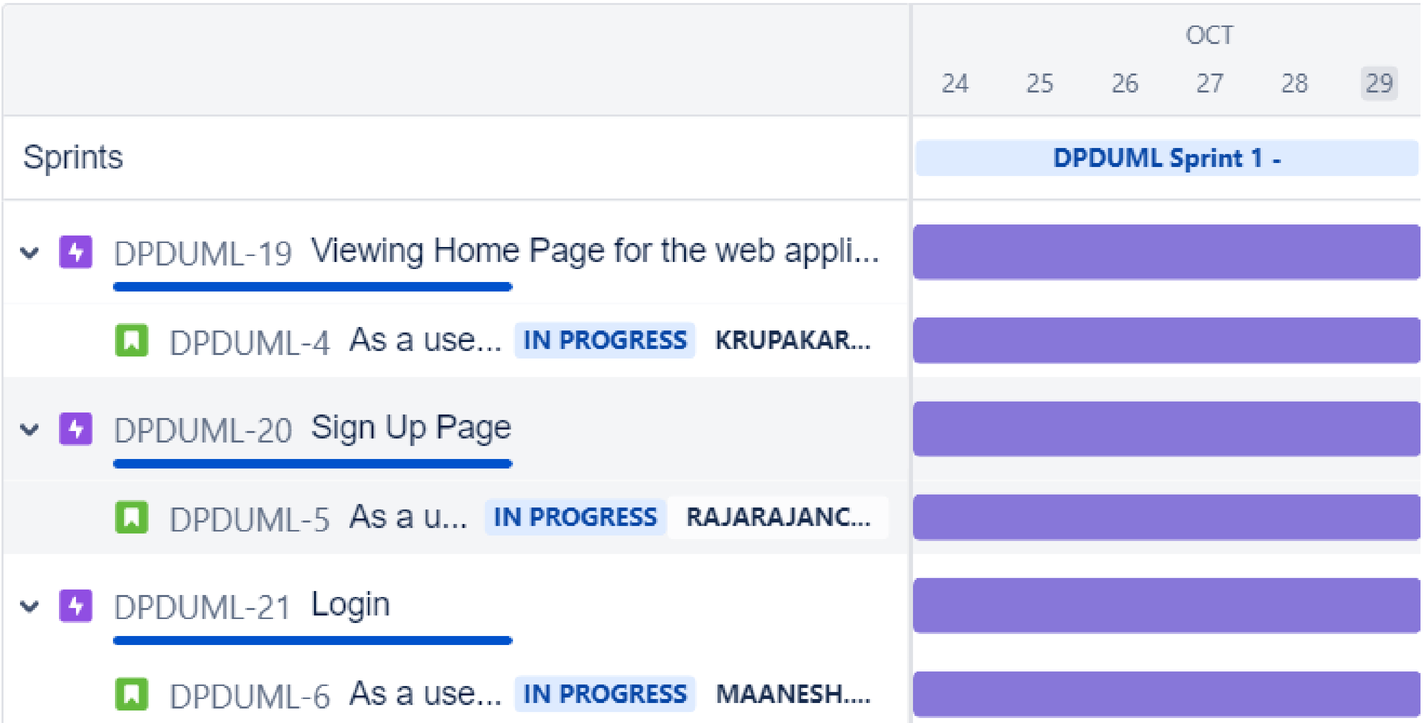
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Public user)	Account creation	USN-1	As a user, I can connect my google into the application	I can access my account / application dashboard	High	Sprint-1
Input data	Adding data	USN-2	As a user, I can feed my data as the input into the application for it to classify the true fake data	I can cross verify the data that entered in the initial step	High	Sprint-1
Data validation	Checking accuracy	USN-3	As a user, I can check the ability and accuracy of the model in obtaining the required information	I can log into my account and check the capability of the model	Medium	Sprint-2
Classification	Data classification	USN-4	As a user, I can view the real data	I can verify my data with the real data	Medium	Sprint-2
App work	Work flow	USN-5	As a user, I can examine the working action of the application model	I can view how the application works and responds to the actions imposed	High	Sprint-2
Image classification	Checking for the disease	USN-6	As a user, I can verify with the application that the image is identified with the actual disease with the help of the trained and tested data's	I can confirm that the data shows the accurate result	Low	Sprint-3
User interaction	AI-powered chatbot	USN-7	As a user, I can interact with the automated chatbot to engage my time till the application processed the accurate result in a meanwhile	I can see the results from the interaction with the chatbot	Low	Sprint-3
Medical assistance	Medical suggestions	USN-8	As a user, I can get medical advises and recommendations for to boost the action of curing the disease	I can get enough assistance by getting the suggestions for curing the disease	High	Sprint-3
Data extraction	Obtaining the data	USN-9	As a user, I can retrieve the result data from the application for data storage for further medical research uses.	I can download the result in the form of data as a proof to show to medical teams	Medium	Sprint-4

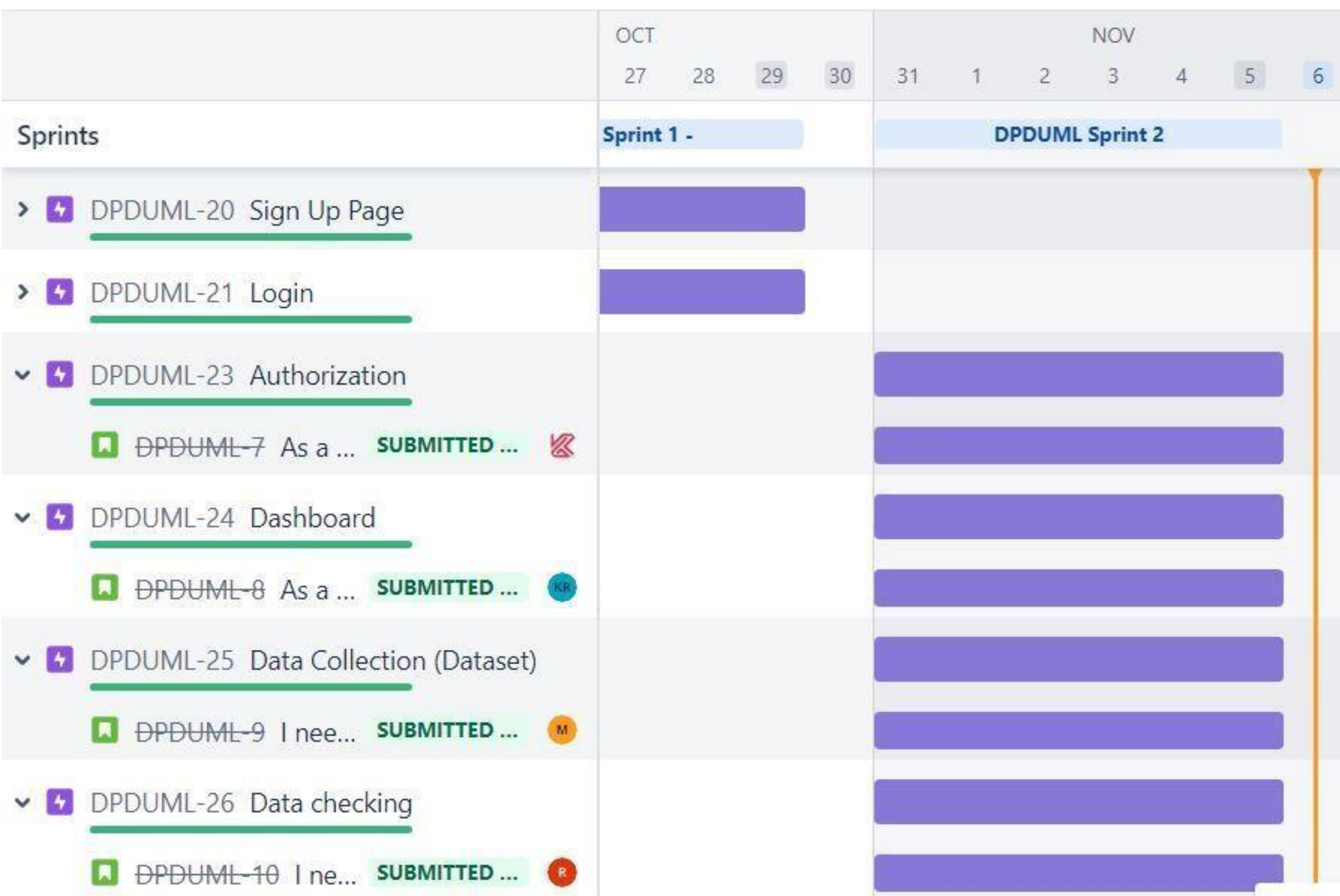
6.2 Sprint Delivery Schedule

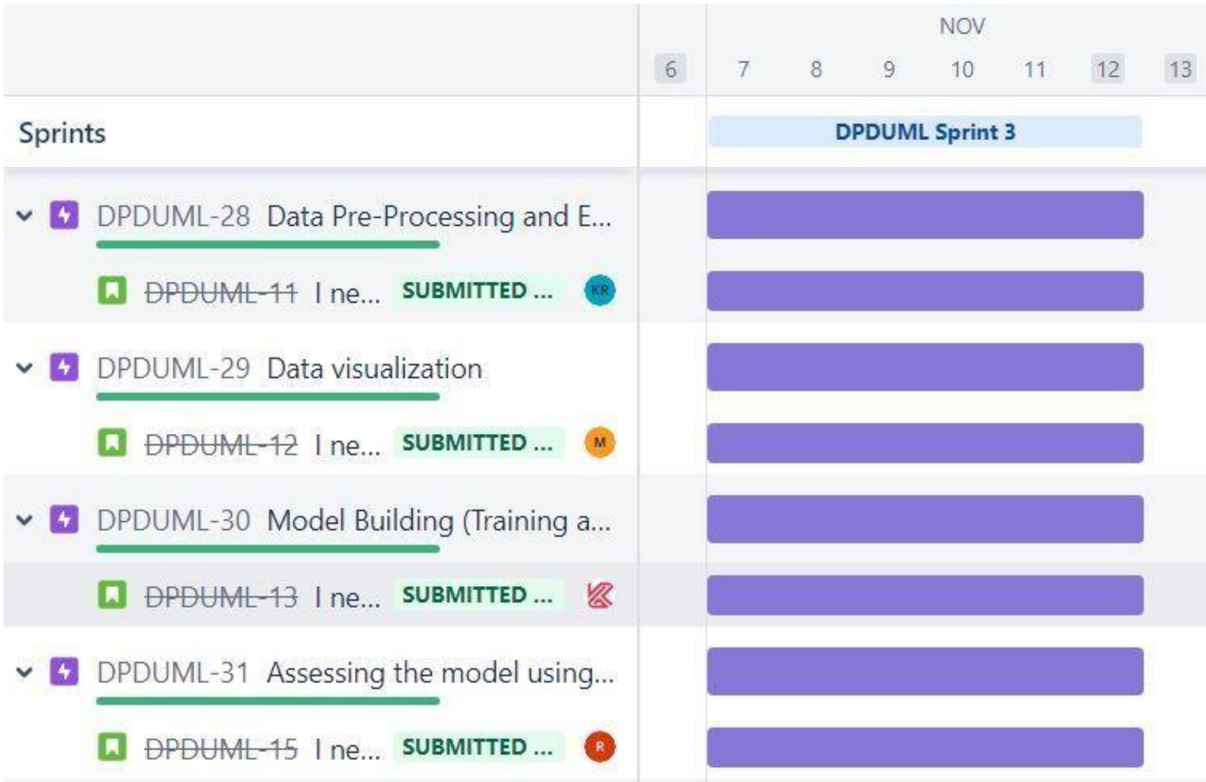
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	10	6 Days	24 Oct 2022	29 Oct 2022	10	29 Oct 2022

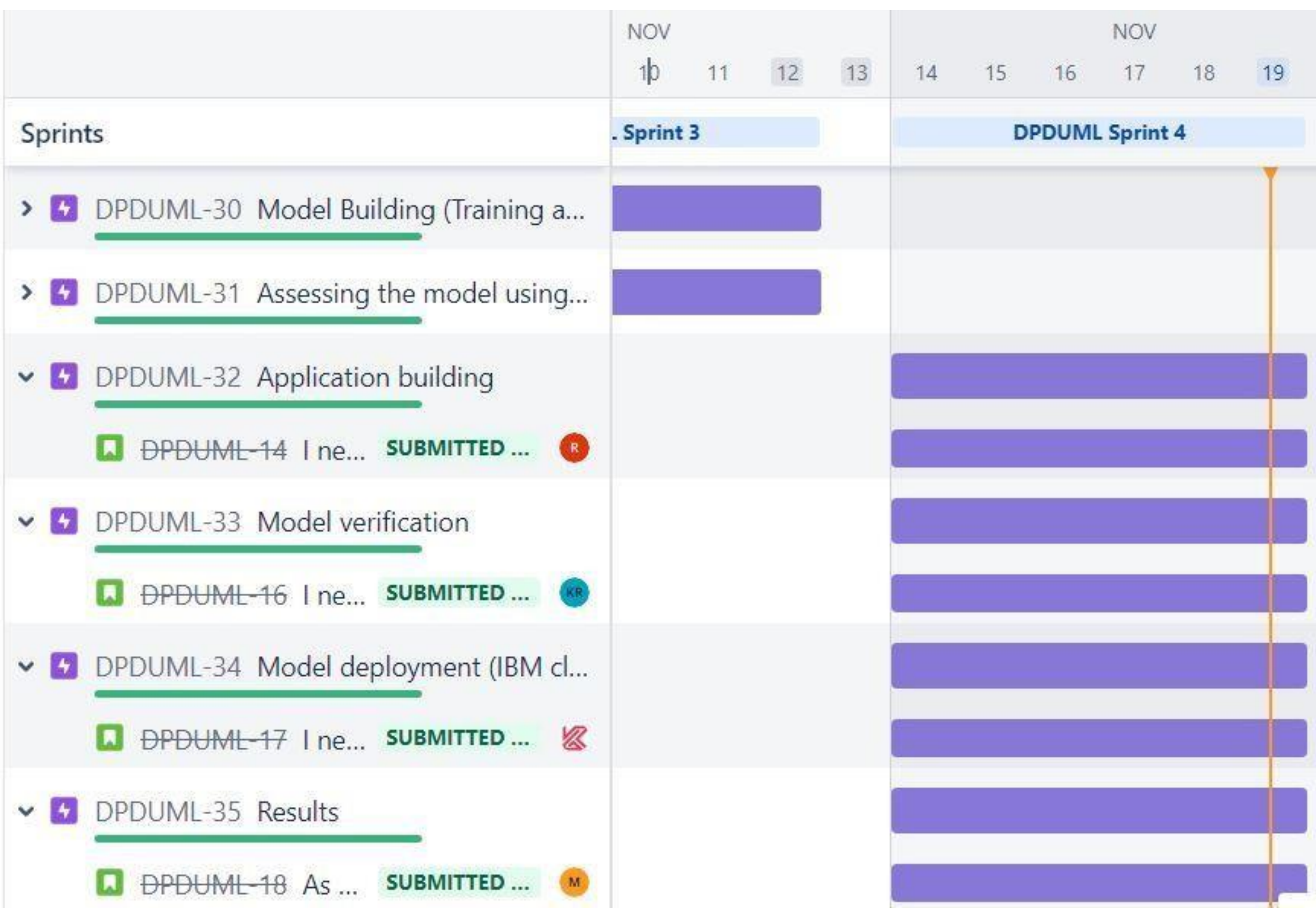
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 Reports from JIRA


OCT
24 25 26 27 28 29
Sprints
DPDUML Sprint 1 -
DPDUML-19 Viewing Home Page for the web appli...
DPDUML-4 As a use... IN PROGRESS KRUPAKAR...
DPDUML-20 Sign Up Page
DPDUML-5 As a u... IN PROGRESS RAJARAJANC...
DPDUML-21 Login
DPDUML-6 As a use... IN PROGRESS MAANESH....


OCT
27 28 29 30
NOV
31 1 2 3 4 5 6
Sprints
Sprint 1 -
DPDUML Sprint 2
DPDUML-20 Sign Up Page
DPDUML-21 Login
DPDUML-23 Authorization
~~DPDUML-7~~ As a ... SUBMITTED ...
DPDUML-24 Dashboard
~~DPDUML-8~~ As a ... SUBMITTED ...
DPDUML-25 Data Collection (Dataset)
~~DPDUML-9~~ I nee... SUBMITTED ...
DPDUML-26 Data checking
~~DPDUML-10~~ I ne... SUBMITTED ...


NOV
6
7
8
9
10
11
12
13
Sprints
DPDUML Sprint 3
DPDUML-28 Data Pre-Processing and E...
~~DPDUML-11~~ I ne... SUBMITTED ...
DPDUML-29 Data visualization
~~DPDUML-12~~ I ne... SUBMITTED ...
DPDUML-30 Model Building (Training a...
~~DPDUML-13~~ I ne... SUBMITTED ...
DPDUML-31 Assessing the model using...
~~DPDUML-15~~ I ne... SUBMITTED ...


NOV
10
11
12
13
NOV
14
15
16
17
18
19
Sprints
. Sprint 3
DPDUML Sprint 4
DPDUML-30 Model Building (Training a...
DPDUML-31 Assessing the model using...
DPDUML-32 Application building
~~DPDUML-14~~ I ne... SUBMITTED ...
DPDUML-33 Model verification
~~DPDUML-16~~ I ne... SUBMITTED ...
DPDUML-34 Model deployment (IBM cl...
~~DPDUML-17~~ I ne... SUBMITTED ...
DPDUML-35 Results
~~DPDUML-18~~ As ... SUBMITTED ...

7. CODING & SOLUTIONING

7.1 Feature 1

We have performed Data preprocessing & Exploratory Data Analysis (EDA), Data visualization, Data mining (model building) and Performance metrics. Finally, we have saved the model

Machine Learning Algorithm for Parkinson Disease

Importing libaries

In [5]:

```
import warnings
warnings.filterwarnings("ignore") #Not to display the warnings

import numpy as np
import pandas as pd
import os, sys
from sklearn.preprocessing import MinMaxScaler
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score #Modelmetrics
```

Data preprocessing and Exploratory Data Analysis(EDA)

In [10]:

```
parkinson_data = pd.read_csv('parkinsons.data')
print(parkinson_data)
```

```
               name  MDVP:Fo(Hz)  MDVP:Fhi(Hz)  MDVP:Flo(Hz)  MDVP:Jitter(%)  \
0    phon_R01_S01_1      119.992       157.302        74.997         0.00784
1    phon_R01_S01_2      122.400       148.650       113.819         0.00968
2    phon_R01_S01_3      116.682       131.111       111.555         0.01050
3    phon_R01_S01_4      116.676       137.871       111.366         0.00997
4    phon_R01_S01_5      116.014       141.781       110.655         0.01284
..              ...          ...           ...           ...             ...
190  phon_R01_S50_2      174.188       230.978        94.261         0.00459
191  phon_R01_S50_3      209.516       253.017        89.488         0.00564
192  phon_R01_S50_4      174.688       240.005        74.287         0.01360
193  phon_R01_S50_5      198.764       396.961        74.904         0.00740
194  phon_R01_S50_6      214.289       260.277        77.973         0.00567

     MDVP:Jitter(Abs)  MDVP:RAP  MDVP:PPQ  Jitter:DDP  MDVP:Shimmer  ...  \
0             0.00007   0.00370   0.00554     0.01109       0.04374  ...
1             0.00008   0.00465   0.00696     0.01394       0.06134  ...
2             0.00009   0.00544   0.00781     0.01633       0.05233  ...
3             0.00009   0.00502   0.00698     0.01505       0.05492  ...
4             0.00011   0.00655   0.00908     0.01966       0.06425  ...
..                ...       ...       ...         ...           ...  ...
190           0.00003   0.00263   0.00259     0.00790       0.04087  ...
191           0.00003   0.00331   0.00292     0.00994       0.02751  ...
192           0.00008   0.00624   0.00564     0.01873       0.02308  ...
193           0.00004   0.00370   0.00390     0.01109       0.02296  ...
194           0.00003   0.00295   0.00317     0.00885       0.01884  ...
```

MDVH denotes Maximum or Minimum Vocal Fundamental Frequency

In [11]:
```
parkinson_data
```

```
Button(description='Toggle Pandas/Lux', layout=Layout(top='5px', width='140px'), style=ButtonStyle())
Output()
```

In [12]:
```
parkinson_data.head(n=20)
```

```
Button(description='Toggle Pandas/Lux', layout=Layout(top='5px', width='140px'), style=ButtonStyle())
Output()
```

In [14]:
```
parkinson_data.tail(50)
```

```
Button(description='Toggle Pandas/Lux', layout=Layout(top='5px', width='140px'), style=ButtonStyle())
Output()
```

In [15]:
```
parkinson_data.shape
#(rows,columns)
```

Out[15]: (195, 24)

In [17]:
```
#Capturing for null values if any of it is available
parkinson_data.isnull().sum()
```

```
Button(description='Toggle Pandas/Lux', layout=Layout(top='5px', width='140px'), style=ButtonStyle())
Output()
```

No null values are present in the data

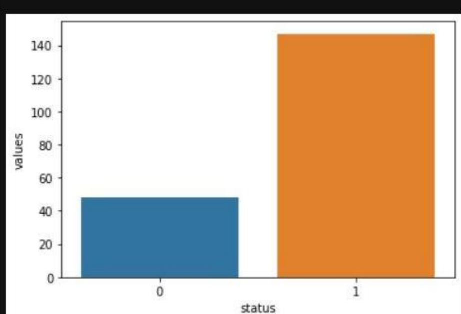
In [23]:
```
variable=parkinson_data['status'].value_counts()
variable_data=pd.DataFrame({'status':variable.index,'values':variable.values})
variable_data
```

```
Button(description='Toggle Pandas/Lux', layout=Layout(top='5px', width='140px'), style=ButtonStyle())
Output()
```

Data visualization

In [24]:
```
#Data visualization
import seaborn as sns
import matplotlib.pyplot as plt
variable = parkinson_data["status"].value_counts()
variable_data = pd.DataFrame({'status':variable.index,'values':variable.values})
sns.barplot(x='status',y='values',data=variable_data)
```

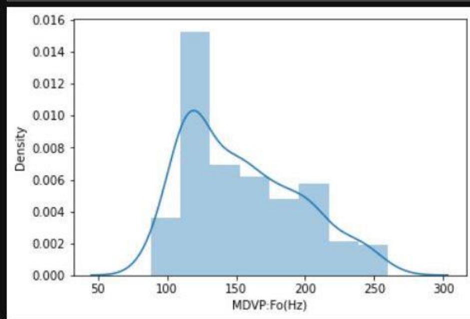
Out[24]:

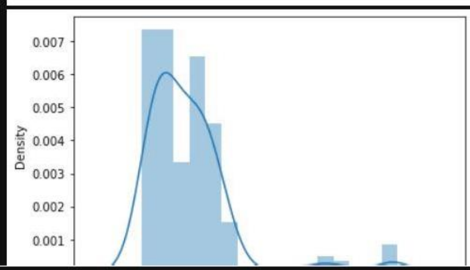


In [25]:

```
#Analyzing the distribution of the data using distplot
def distplots(col):
  sns.distplot(parkinson_data[col])
  plt.show()

for i in list(parkinson_data.columns)[1:]:
  distplots(i)
```



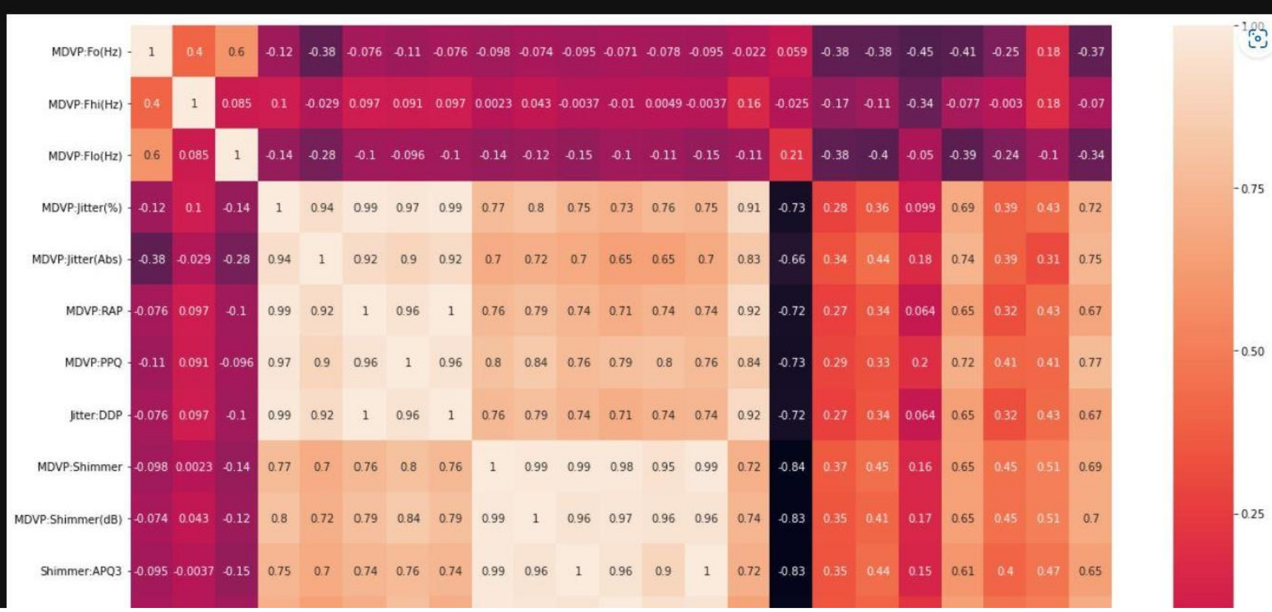


In [29]:

```
#Figuring out the correlations using heatmap to visualize between the features and patterns in the data used for this project

plt.figure(figsize=(20,20))
correlation_data=parkinson_data.corr()
sns.heatmap(correlation_data,annot=True)
```

Out[29]:



In [30]:
```python
#We are making the final changes in the data by dividing the data into independent as x and dependent variables as y and removing the ID column
x = parkinson_data.drop(["status","name"],axis=1)
y = parkinson_data["status"]
#It is done to integrate the two x and y variables into the model building steps
```

In [31]:
```python
#After the changes,let's detect the label balance
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter #For priortizing the importance to store elements as dictionary keys, and their counts as values.
print(Counter(y))
```

```
Counter({1: 147, 0: 48})
```

In [32]:
```python
#Now,we are balancing the labels
ROS = RandomOverSampler() #To compensate the imbalance part present in the data
x_ROS,y_ROS = ROS.fit_resample(x, y)
print(Counter(y_ROS))
```

```
Counter({1: 147, 0: 147})
```

Scaling the data

In [33]:
```python
#It is very much important to scale the data for the betterment of the model using such as Support Vector Machine and K Nearest Neighbor Algorithms
Scaler_data = MinMaxScaler((-1,1))
x = Scaler_data.fit_transform(x_ROS)
y = y_ROS
```

Model Building (Training and Testing)

Data mining and performance metrics

In [36]:
```python
#We are going to import and use it for assessing the model using performance metrics from Classification process
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score
List_metrics = []
List_accuracy = []

#Logistic Regression
from sklearn.linear_model import LogisticRegression
Classification_model = LogisticRegression(C=0.4,max_iter=1000,solver='liblinear')
Log_Regression = Classification_model.fit(x_train, y_train)
y_pred = Classification_model.predict(x_test)  #Prediction
Log_Regression_accuracy = accuracy_score(y_test, y_pred)  #Accuracy
print("The accuracy score with Logistic regression is:",Log_Regression_accuracy)

#Decision Tree Classificaton using supervised machine learning for classifiying the data with confident accuracy
from sklearn.tree import DecisionTreeClassifier
Classification_tree = DecisionTreeClassifier(random_state=14)
Decision_tree = Classification_tree.fit(x_train, y_train)
y_pred2 = Classification_tree.predict(x_test) #Prediction
Dec_tree_accuracy = accuracy_score(y_test, y_pred2) #Accuracy
print("The accuracy score with Decision Tree Classifier is:",Dec_tree_accuracy)

#Random Forest Classifier is used for its high dimensionality and accuracy capabilities, here information gain is priortized
from sklearn.ensemble import RandomForestClassifier
Classification_random = RandomForestClassifier(random_state=14)
RFE = Classification_random.fit(x_train, y_train)
y_pred3 = Classification_random.predict(x_test) #Prediction
Ran_For_accuracy = accuracy_score(y_test, y_pred3) #Accuracy
print("The accuracy score with Random Forest Classifier(Information gain) is:",Ran_For_accuracy)

#Random Forest Classifier with entropy condition
from sklearn.ensemble import RandomForestClassifier
Classification_entropy = RandomForestClassifier(criterion='entropy')
RFE = Classification_entropy.fit(x_train,y_train)
```

Converging the above classification algorithms and performance metric using Voting Classifier.

In [37]:
```
from sklearn.ensemble import VotingClassifier
VC = VotingClassifier(estimators=[('Classification_model',Classification_model),('Classification_tree',Classification_tree),('Classification_random',C
Model_VC = VC.fit(x_train, y_train)
Model_prediction = VC.predict(x_test)
Model_accuracy = accuracy_score(y_test,pred_gnb)
print(Model_accuracy)
```

```
0.8813559322033898
```

XGBClassification - Supervised Machine Learning

In [38]:
```
Model_XG = XGBClassifier(random_state=0)
Model_XG.fit(x_train,y_train)
```

Out[38]:
```
XGBClassifier()
```

7.2 Feature 2

*** We have created an Application with Home Page (After logging in by the user), Layout and Predict Page.**

```html
<!DOCTYPE html>
<!--
This is a starter template page. Use this page to start your new project from
scratch. This page gets rid of all links and provides the needed markup only.
-->
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Parkinson Detection</title>

    <!-- Google Font: Source Sans Pro -->
    <link
      rel="stylesheet"
      href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback"
    />
    <!-- Font Awesome Icons -->
    <link
      rel="stylesheet"
      href="../static/plugins/fontawesome-free/css/all.min.css"
    />
    <!-- Theme style -->
    <link rel="stylesheet" href="../static/dist/css/adminlte.min.css" />
    <link
      rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/admin-lte@3.1/dist/css/adminlte.min.css"
    />
  </head>
  <body
    class="hold-transition layout-top-nav layout-footer-fixed layout-navbar-fixed"
  >
    <div class="wrapper">
      <!-- Navbar -->
      <nav
```

```
<script src="../static/plugins/jquery/jquery.min.js"></script>
<!-- Bootstrap 4 -->
<script src="../static/plugins/bootstrap/js/bootstrap.bundle.min.js"></script>
<!-- AdminLTE App -->
<script src="../static/dist/js/adminlte.min.js"></script>
<!-- AdminLTE for demo purposes -->
<script src="../static/dist/js/demo.js"></script>
<script src="https://cdn.jsdelivr.net/npm/admin-lte@3.1/dist/js/adminlte.min.js"></script>
<script>
  var currentTheme = sessionStorage.getItem("theme");
  var mainHeader = document.querySelector(".main-header");

  if (currentTheme) {
    if (currentTheme === "dark") {
      if (!document.body.classList.contains("dark-mode")) {
        document.body.classList.add("dark-mode");
      }
      if (mainHeader.classList.contains("navbar-light")) {
        mainHeader.classList.add("navbar-dark");
        mainHeader.classList.remove("navbar-light");
      }
      toggleSwitch.checked = true;
    }
  }
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet">
    <link href="https://getbootstrap.com/docs/5.2/assets/css/docs.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body style="background-color:rgb(205, 205, 205)">
    <div class="container-fluid" style=
                                    "background-color:rgb(41, 41, 41);
                                    border-radius: 1px;">
        <ul class="nav justify-content-end">
            <li class="nav-item">
              <a class="nav-link" href="{{url_for('Home_page')}}"><b>Home</b></a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="{{url_for('info_page')}}"><b>Info</b></a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="{{url_for('Predict_page')}}"><b>Predict</b></a>
            </li>
          </ul>
    </div>
Value : <input type=”radio” name=”parkinsons.data” value=”MDVP:Fo(Hz)” /> MDVP:Fo(Hz)
Value : <input type=”radio” name=”parkinsons.data” value=”MDVP:Fhi(Hz)” /> MDVP:Fhi(Hz)
Value : <input type=”radio” name=”parkinsons.data” value=”MDVP:Flo(Hz)” /> MDVP:Flo(Hz)
Value : <input type=”radio” name=”parkinsons.data” value=”MDVP:Jitter(%)” /> MDVP:Jitter(%)
Value : <input type=”radio” name=”parkinsons.data” value=”MDVP:Jitter(Abs)” /> MDVP:Jitter(Abs)
Value : <input type=”radio” name=”parkinsons.data” value=”MDVP:RAP” /> MDVP:RAP
Value : <input type=”radio” name=”parkinsons.data” value=”MDVP:PPQ” /> MDVP:PPQ
```

```
            <li class="nav-item">
              <a class="nav-link" href="{{url_for('Home_page')}}"><b>Home</b></a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="{{url_for('info_page')}}"><b>Info</b></a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="{{url_for('Predict_page')}}"><b>Predict</b></a>
            </li>
          </ul>
    </div>
Value : <input type="radio" name="parkinsons.data" value="MDVP:Fo(Hz)" /> MDVP:Fo(Hz)
Value : <input type="radio" name="parkinsons.data" value="MDVP:Fhi(Hz)" /> MDVP:Fhi(Hz)
Value : <input type="radio" name="parkinsons.data" value="MDVP:Flo(Hz)" /> MDVP:Flo(Hz)
Value : <input type="radio" name="parkinsons.data" value="MDVP:Jitter(%)" /> MDVP:Jitter(%)
Value : <input type="radio" name="parkinsons.data" value="MDVP:Jitter(Abs)" /> MDVP:Jitter(Abs)
Value : <input type="radio" name="parkinsons.data" value="MDVP:RAP" /> MDVP:RAP
Value : <input type="radio" name="parkinsons.data" value="MDVP:PPQ" /> MDVP:PPQ
Value : <input type="radio" name="parkinsons.data" value="Jitter:DDP" /> Jitter:DDP
Value : <input type="radio" name="parkinsons.data" value="MDVP:Shimmer" /> MDVP:Shimmer
Value : <input type="radio" name="parkinsons.data" value="MDVP:Shimmer(dB)" /> "MDVP:Shimmer(dB)
Value : <input type="radio" name="parkinsons.data" value="Shimmer:APQ3" /> Shimmer:APQ3
Value : <input type="radio" name="parkinsons.data" value="Shimmer:APQ5" /> Shimmer:APQ5
Value : <input type="radio" name="parkinsons.data" value="MDVP:APQ" /> MDVP:APQ
Value : <input type="radio" name="parkinsons.data" value="Shimmer:DDA" /> Shimmer:DDA
Value : <input type="radio" name="parkinsons.data" value="NHR" /> NHR
Value : <input type="radio" name="parkinsons.data" value="HNR" /> HNR
Value : <input type="radio" name="parkinsons.data" value="status" /> status
Value : <input type="radio" name="parkinsons.data" value="RPDE" /> RPDE
Value : <input type="radio" name="parkinsons.data" value="MDVP:Fo(Hz) /> MDVP:Fo(Hz)
Value : <input type="radio" name="parkinsons.data" value="DFA" /> DFA
Value : <input type="radio" name="parkinsons.data" value="spread1" /> spread1
Value : <input type="radio" name="parkinsons.data" value="spread2" /> spread2
Value : <input type="radio" name="parkinsons.data" value="D2" /> D2
Value : <input type="radio" name="parkinsons.data" value="PPE" /> PPE
<button type="PREDICT">Send your prediction data</button>
```

```
<!DOCTYPE html>
<!--
This is a starter template page. Use this page to start your new project from
scratch. This page gets rid of all links and provides the needed markup only.
-->
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Parkinson Detection</title>

    <!-- Google Font: Source Sans Pro -->
    <link
      rel="stylesheet"
      href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback"
    />
    <!-- Font Awesome Icons -->
    <link
      rel="stylesheet"
      href="../static/plugins/fontawesome-free/css/all.min.css"
    />
    <!-- Theme style -->
    <link rel="stylesheet" href="../static/dist/css/adminlte.min.css" />
    <!-- dropzonejs -->
    <link
      rel="stylesheet"
      href="../static/plugins/dropzone/min/dropzone.min.css"
    />
  </head>
  <body
    class="hold-transition layout-top-nav layout-footer-fixed layout-navbar-fixed"
  >
    <div class="wrapper">
```

```html
<div class="collapse navbar-collapse order-3" id="navbarCollapse">
  <!-- Left navbar links -->
  <ul class="navbar-nav">
    <li class="nav-item">
      <a href="/" class="nav-link">Home</a>
    </li>
    <li class="nav-item">
      <a href="/info" class="nav-link">Info</a>
    </li>
    <li class="nav-item">
      <a href="/test" class="nav-link">Predict</a>
    </li>
  </ul>
</div>

<!-- Right navbar links -->
<ul class="order-1 order-md-3 navbar-nav navbar-no-expand ml-auto">
  <li class="nav-item">
    <button
      type="button"
      onclick="switchTheme()"
      class="btn btn-primary btn-block btn-sm"
    >
      <i class="fa fa-bell"></i> Switch Theme
```

```
var previewNode = document.querySelector("#template");
previewNode.id = "";
var previewTemplate = previewNode.parentNode.innerHTML;
previewNode.parentNode.removeChild(previewNode);

var myDropzone = new Dropzone(document.body, {
  // Make the whole body a dropzone
  url: "/predict", // Set the url
  thumbnailWidth: 80,
  thumbnailHeight: 80,
  parallelUploads: 20,
  previewTemplate: previewTemplate,
  autoQueue: false, // Make sure the files aren't queued until manually added
  previewsContainer: "#previews", // Define the container to display the previews
  clickable: ".fileinput-button", // Define the element that should be used as click trigger to select files.
  success: function (file, response) {
    if (response === "healthy") {
      $("#successModel").click();
    } else {
      $("#dangerModel").click();
    }
  },
});

myDropzone.on("addedfile", function (file) {
  // Hookup the start button
  file.previewElement.querySelector(".start").onclick = function () {
    myDropzone.enqueueFile(file);
  };
});
```

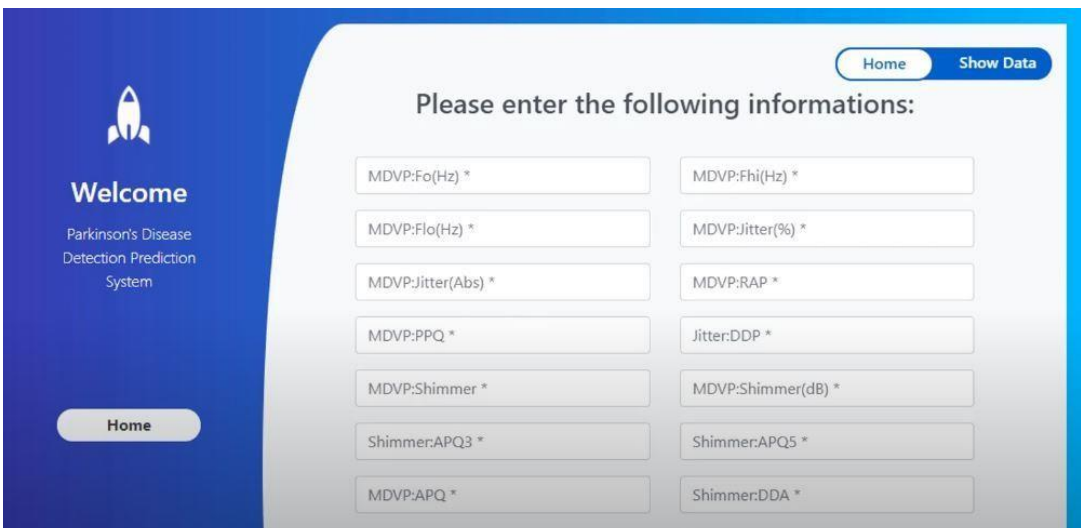
```
    var predict = function(input) {
      if (window.model) {
        window.model.predict([tf.tensor(input).reshape([1, 28, 28, 1])]).array().then(function(scores){
          scores = scores[0];
          predicted = scores.indexOf(Math.max(...scores));
          $('#number').html(predicted);
        });
      } else {
        // The model takes a bit to load, if we are too fast, wait
        setTimeout(function(){predict(input)}, 50);
      }
    }

    $('#clear').click(function(){
      context.clearRect(0, 0, canvas.width, canvas.height);
      $('#number').html('');
    });
    </script>
  </body>
</html>
```

```
!pip install tensorflowjs
!tensorflowjs_converter --input_format keras
    '/content/Parkinson_MLmodel.sav' '/content/standardScaler.sav'
```

Login Page:

Disease input data by registering in this Page:



Predict result side:

Predict Page:

8. TESTING

8.1 Test Cases

Date	17-Nov-22
Team ID	PNT2022TMID28255
Project Name	Project - Detecting Parkinson's
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute
TC_001	Functional	Home Page	Verify user is able to visit home	PC or Laptop & URL	1. Login and enter the input data
TC_002	Functional	Home Page	Verify user is able to enter the input	PC or Laptop, URL & Hand-	1. Enter the input data and click
TC_003	Functional	Home page	Verify user is able to get the result	PC or Laptop, URL & Hand-	1. Enter input data 2. Click the get
TC_004	UI	Home page	Verify user is able to identify	PC or Laptop & URL	1.Enter input data and click go
TC_005	UI	Home page	Verify user is able to see the get the	PC or Laptop, URL & Hand-	1. Know about the disease in the

Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	Executed By
User able to visit home page	Working as	Pass	Easy to access	N	Kamalesh S
User is able to enter the input data	Working as	Pass	Less time taken	N	RajaRajan R
Verify user is able to get the result	Working as	Pass	Accurate result	N	Maanesh S
User is able to identify the correct	Working as	Pass	Easy to identify the upload	N	KrupaKaran R
User is able to see the get the correct	Working as	Pass	Easy to identify the get result	N	All team members

8.2 User Acceptance Testing

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Detecting Parkinson's Disease using Machine Learning project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	0	1	1	0	2
Duplicate	0	0	0	0	0
External	2	2	0	1	5
Fixed	1	0	0	0	1
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	3	3	1	1	8

3. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Login/Register Page	8	0	0	8
Home Page	1	0	0	1
Logout Page	2	0	1	1
Prediction	10	0	0	10
Version Control	2	0	0	2

9. RESULTS

9.1 Performance Metrics

* **Classification Model: Confusion Matrix, Accuracy Score & Classification Report**

Assessing the model using metrics

In [39]:
```
y_predict = Model_XG.predict(x_test)
print(accuracy_score(y_test,y_predict)*100)
```

```
96.61016949152543
```

Hence by reducing the overfitting using XGBoost Classifier, we are getting accuracy_score of **98.30%** for the model

Confusion metrics

In [40]:
```
from sklearn.metrics import confusion_matrix
ypre = Classification_model.predict(x_test)
ypre = (ypre>0.5)
confusion_matrix(y_test,ypre)
```

Out[40]:
```
array([[20,  4],
       [ 7, 28]])
```

F1 score

In [41]:
```
from sklearn.metrics import f1_score
Variation_score = f1_score(y_test, Model_XG.predict(x_test), average='binary')
print(Variation_score/0.01)
```

```
97.14285714285714
```

Classification report

In [42]:
```
from sklearn import metrics
from sklearn.metrics import classification_report
print("\n Classification report for Model  %s:\n%s\n" % (Model_XG, metrics.classification_report(y_test, y_pred)))
```

10. ADVANTAGES & DISADVANTAGES

10.1 Advantages

- We developed a model using the XG Boost Classifier using sklearn module of python to detect if an individual has Parkinson’s Disease or not. We got the machine learning model with 96.61% accuracy, which is good as our dataset contains good labels and values.
- More accuracy in the model
- The data of any person can be entered in db to check whether the person is affected by Parkinson’s disease or not.

10.2 Disadvantages

- Packages to be installed
- It produces fake results if the input data is entered wrong

11. CONCLUSION

It is possible to detect Parkinson`s disease using the drawings alone instead of measuring the speed and pressure of the pen on paper. Here, we presented included studies in a high-level summary, providing access to information including machine learning methods that have been used in the diagnosis of PD and associated outcomes, types of clinical, behavioral, and biometric data that could be used for rendering more accurate diagnoses, potential biomarkers for assisting clinical decision making, and other highly relevant information, including databases that could be used to enlarge and enrich smaller datasets. In summary, realization of machine learning-assisted diagnosis of PD yields high potential for a more systematic clinical decision-making system, while adaptation of novel biomarkers may give rise to easier access to PD diagnosis at an earlier stage.

.

12. FUTURE SCOPE

Following years of minimal progress in the treatment of Parkinson's disease, pioneering pipeline therapies such as those previously discussed offer hope to those affected by this devastating condition.

13. APPENDIX

13.1 Source Code

Machine Learning code : IBM-Project-27034-1660044417/Project Development Phase/Sprint 3/Machine Learning Algorithm at main · IBM-EPBL/IBM-Project-27034-1660044417 (github.com)

Web development code : IBM-Project-27034-1660044417/Project Development Phase/Sprint 4/Web application (Application building) at main · IBM-EPBL/IBM-Project-27034-1660044417 (github.com)

13.2 Github Link

Repository link: https://github.com/IBM-EPBL/IBM-Project-27034-1660044417

13.3 Project Demo Link

Demonstration video link: https://drive.google.com/file/d/1QimXSIX3-NDfVHbUNObs3tyG60UzCtfi/view?usp=sharing