DETECTION OF 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 automatically detect 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. Although previous studies have reviewed the use of machine learning in the diagnosis and assessment of PD, they were limited to the analysis of motor symptoms, kinematics, and wearable sensor data. Moreover, some of these reviews only included studies published between 2015 and 2016. In this study, we aim to comprehensively summarize all published studies that applied machine learning models to the diagnosis of PD for an exhaustive overview of data sources, data types, machine learning models, and associated outcomes, (b) assess and compare the

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

S.NO	AUTHOR & YEAR	TITLE	DESCRIPTION	ADVANTAGE	DISADVANTAGE
1.	Deepa Shenoy, Vibhudendra simha. G. G, P L Rrashmi, K.R. Venugopal, Sandhya Joshi, L.M Patnaik. (2010)	Classification of Alzheimer's Disease and Parkinson's Disease by Using Machine Learning and Neural Network Methods.	The objective of this paper was to classify the Alzheimer's disease and Parkinson's disease based on the most influencing risk factors using different classifier techniques.	The classification model was validated with the test cases and the model achieved a high classification accuracy of 99.25% with Random Forest tree and the Multilayer Perceptron.	The classification accuracy varies greatly with the change in the identification of the important risk factor with another and hence the model needs to be trained again.
2.	Liaqat Ali, Ce Zhu, Zhonghao Zhang, Yipeng Liu. (2019)	Automated Detection of Parkinson's Disease Based on Multiple Types of Sustained Phonation's Using Linear Discriminant Analysis and Genetically Optimized Neural Network.	Developed a hybrid intelligent system for PD detection based on multiple types of sustained phonation's data. The developed system uses LDA model for dimensionality reduction and neural network model for classification. The architecture of the neural network model was optimized using genetic algorithm. Experimental results showed that the developed intelligent system could discriminate between PD patients and healthy subjects with an accuracy of 95% on training database and 100% on testing database using all the collected features of the dataset. However, the limitation of gender imbalance in the dataset was highlighted and hence the gender dependent features were eliminated, and the proposed system was again simulated. Consequently, we obtained 80% accuracy on training database and 82.14% on testing database.	The proposed LDA-NN-GA method shows better performance and lower complexity.	The first limitation is in the independent dataset (testing dataset) which was only collected from PD patients and is highly imbalanced *i.e.*, the healthy class has no representation in the testing database. Additionally, no information about UPDRS is provided for the subjects of the testing database. The second limitation is missing information about the feature extraction process *e.g.*, was the extraction of features corrected for pitch halving/doubling? Third, information about speech severity and whether the PD patients were investigated in the OFF or ON state, are also missing.

3.	Saurabhchand Bhati, Laureano Moro Velazquez, Jesus villalba. (2019)	LSTM Siamese Network for Parkinson's Disease Detection from Speech.	They proposed a two-step strategy to use machine learning methods for PD detection. In the first step, we use Long Short-Term Memory (LSTM)-based Siamese networks to learn feature representations that highlight the information related to speech articulation and prosody relevant for PD detection. Siamese networks are trained on data pairs employing a Spanish corpus containing 52 patients and 56 control subjects. In the second step, we train a classifier to make decisions about the presence or absence of PD employing the features provided by the LSTM networks.	They achieved an EER of 1.9% in the detection by combining the scores of different text-dependent models. Preliminary experiments show the efficacy of the proposed method and prove the usefulness of LSTM for PD detection from speech.	Long short-term memory (LSTM) Siamese networks are used for dysarthric speech detection. Networks with Siamese architectures are trained on pairs of input data with the same phonetic content.
4.	Ishan Vatsaraj, Dr. Gajanan Nagare. (2021)	Early Detection of Parkinson's Disease using Contrast Enhancement Techniques and CNN.	Augmentation methods like rotation, vertical and horizontal flipping along with Support vector machines and HOG methods are applied. The significant improvement in the accuracy can be attributed to the optimal contrast enhancement technique used and that two different CNN models were used for predicting the spiral and wave patterns respectively.	The proposed model showed an accuracy of 96.67% with a precision of 93.33% and recall of 100%.	This model does not provide the probability of the percentage that is affected in a person by the Parkinson's disease.

2.3 Problem Statement Definition

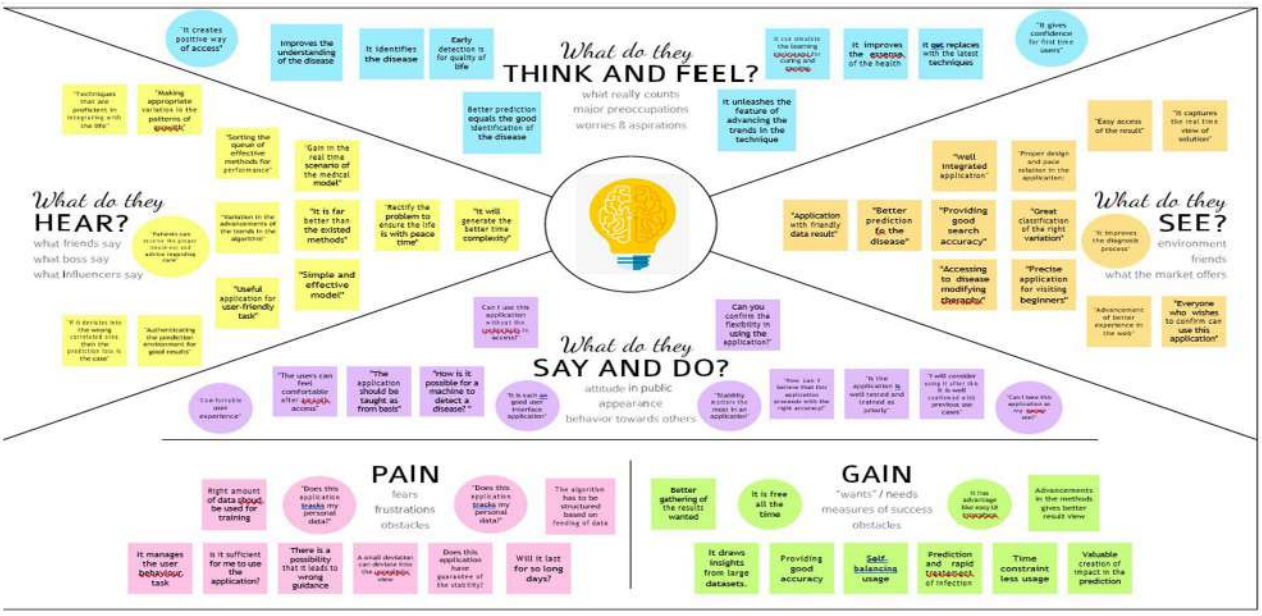
Parkinson's disease is a chronic progressive disease of the nervous system characterised by the cardinal features of rigidity, bradykinesia, tremor and postural instability. However there is no recognized test for PD for patients, particularly in the early stages. This results in increased mortality rate. Thus detection system of Parkinson's disease with easy steps and feasible one to detect parkinson's disease at the early stage is essential.

INSIGHTS FROM PROBLEM STATEMENTS:

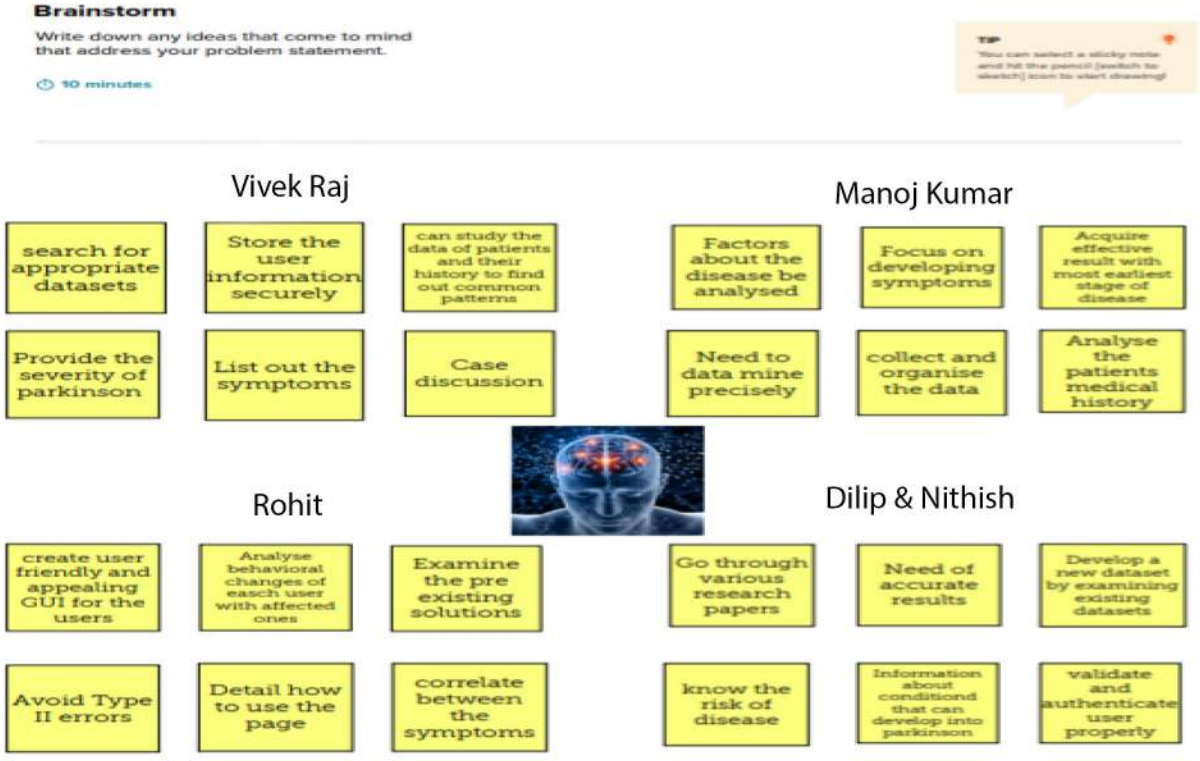
Who does the problem affect?	**Men are slightly more likely to get affected by Parkinson's disease compared to women.**
What are the boundaries of the problem?	**People usually develop the disease around age 60 or older.**
What is the impact of the issue?	**Parkinson's disease is caused by a loss of nerve cells in part of the brain called the substantia nigra. This leads to a reduction in a chemical called dopamine in the brain.**
What impact is the issue causing?	**Motor symptoms: slow movement, tremor, rigidity, walking and imbalance. Non-motor complications: cognitive impairment, mental health disorders, sleep disorders and pain and other sensory disturbances.**
When does it need to be fixed?	**It needs to be fixed at the earliest when the suspected symptoms like soft or low voice, tremors, lack of facial expression and so on occur.**
What would happen if we didn't solve the problem?	**Does not directly cause people to die, but the condition can place great strain on the body and can make some people more vulnerable to serious and life-threatening infections.**
Where is the issue is occurring?	**The most prominent signs and symptoms occur when nerve cells in the basal ganglia that control movement become impaired or die.**
Why is it important that we fix the problem?	**By early detection of disease makes the people to take proper diagnosis on time to improve the quality of life of patients.**

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



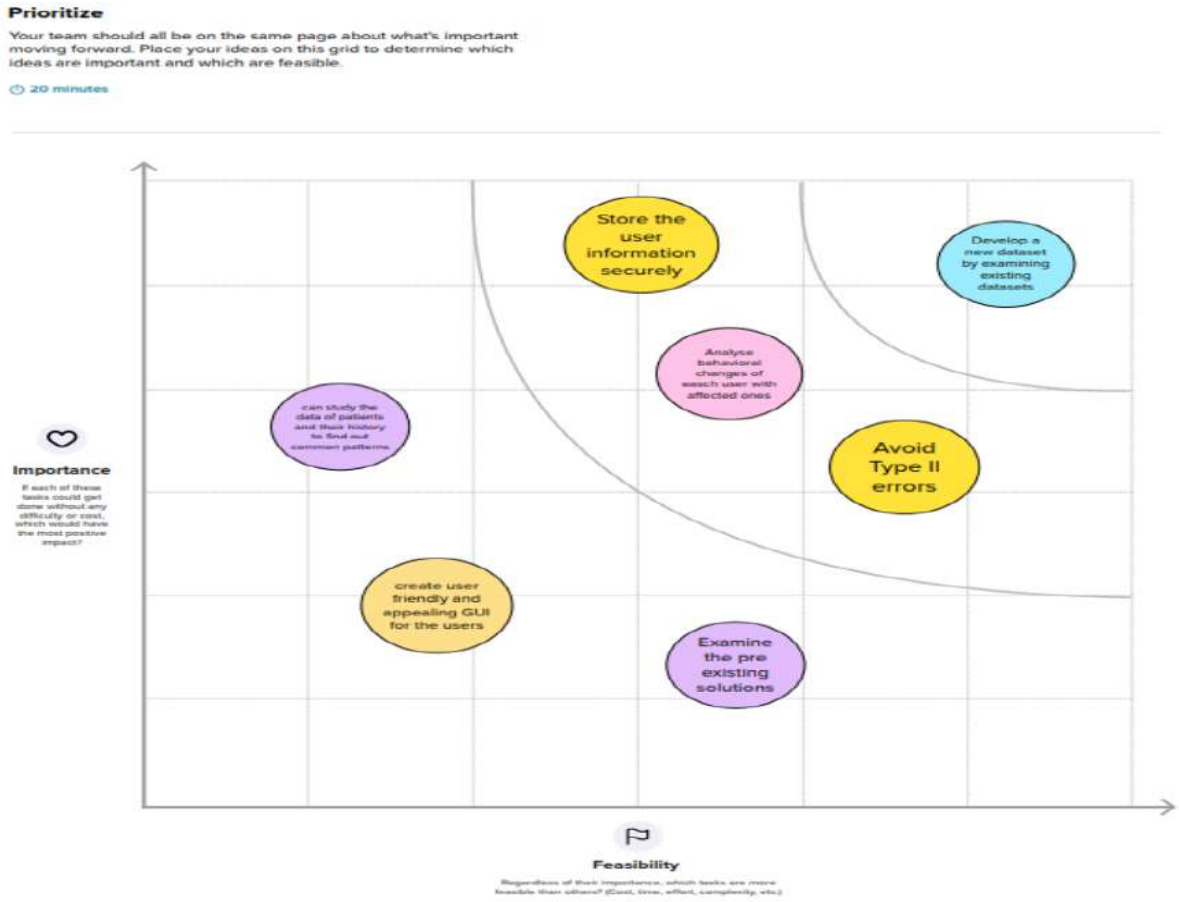
3.2 Ideation & Brainstorm



Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you end break it up into smaller sub-groups.

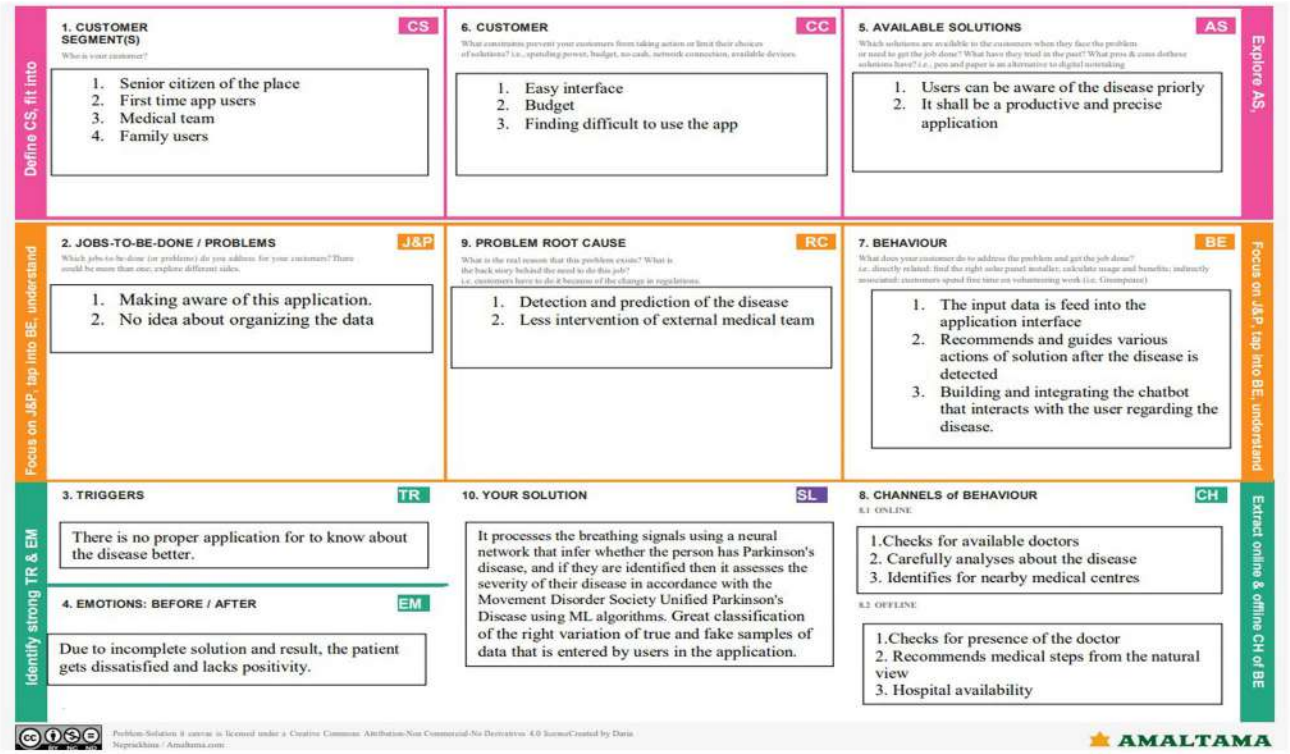
20 minutes



3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Parkinson's disease is a chronic, progressive neurodegenerative ailment due to the loss of dopamine in nerve cells. The disease can't be cured but early detection makes people take proper medication to improve their quality of life.
2.	Idea / Solution Description	Our goal is to quantify the visual appearance(using the HOG method) of these drawings and then train a machine learning model to classify them. We use a Histogram of Oriented Gradients (HOG) image descriptor along with a Random Forest classifier to automatically detect Parkinson's disease in hand-drawn images of spirals and waves.
3.	Novelty / Uniqueness	The accuracy of our project will be high. The OpenCV techniques are also used to eliminate even the use of paper for drawings contributing to the novelty factor.
4.	Social Impact / Customer Satisfaction	The user-friendly web app is created. Early detection of Parkinson's disease with high accuracy.
5.	Business Model (Revenue Model)	The application is free of cost. Any person can use it from anywhere.
6.	Scalability of the Solution	The dimensionality reduction process can be adjusted to produce precise predictions with an increase in the features taken into account.

3.4 Problem Solution fit



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

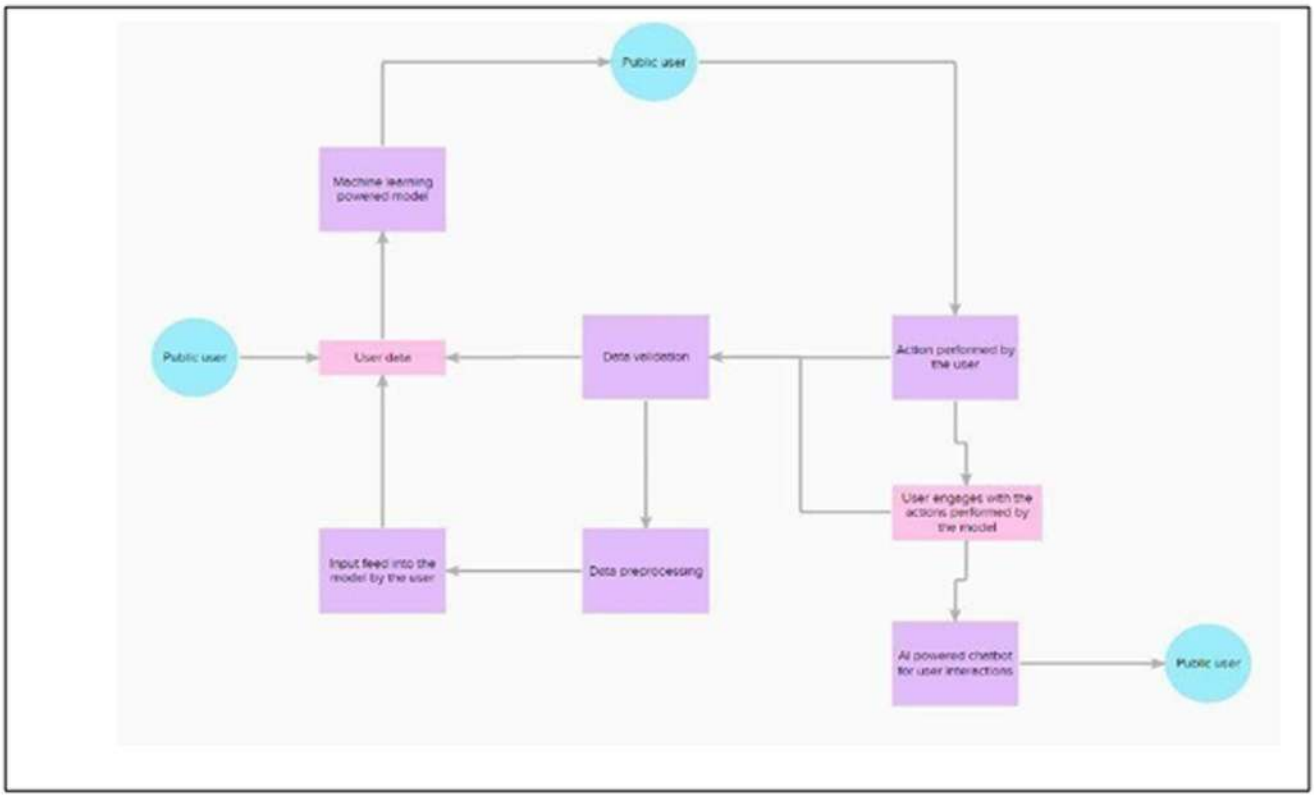
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User account registration	Registration through Google account and forms
FR-2	Input data	Application received the data and processes its roles
FR-2	User Authorization	Verifying the user's account
FR-3	Data classification	Classification 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

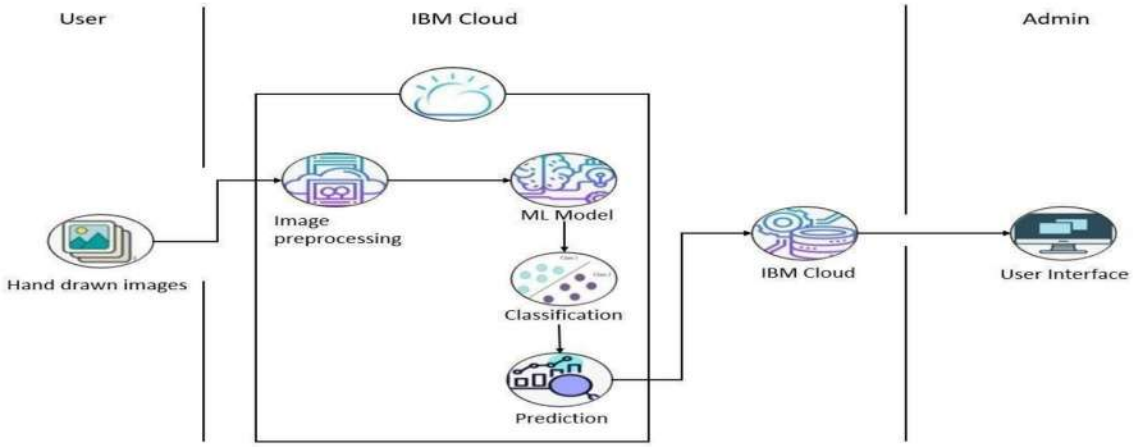
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The application can be used for accurate prediction and classifier of the true and fake input data sample
NFR-2	**Security**	User's data is well encrypted using stable machine learning algorithms
NFR-3	**Reliability**	The application is 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



5.2 Solution & Technical Architecture



5.3 User Stories

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint1	Sign Up	USN-1	As a user, I want to register for the application with my Gmail and get verified.	2	Medium	Vivek S Nithish M Dilip T Manoj S Rohit R
Sprint1	Login	USN-2	As a user, I can able to login with my credentials.	4	Low	Vivek S Nithish M Dilip T Manoj S Rohit R
Sprint1	Dash Board	USN-3	As a user, I need to know about the Parkinson disease and symptoms involved.	2	High	Vivek S Nithish M Dilip T Manoj S Rohit R
Sprint2	Data Collection (Dataset)	USN-4	I should collect data, images of spirals and waves drawn by patients for processing.	6	High	Vivek S Nithish M Dilip T Manoj S Rohit R

6. PROJECT PLANNING & SCHEDULING

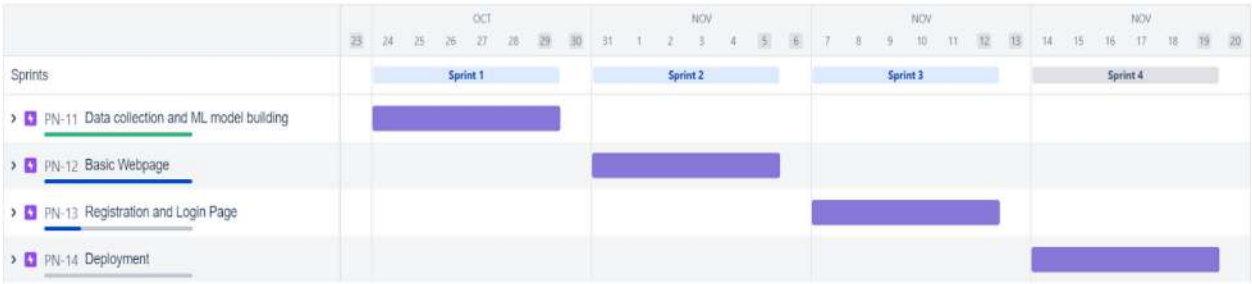
6.1 Sprint Planning & Estimation

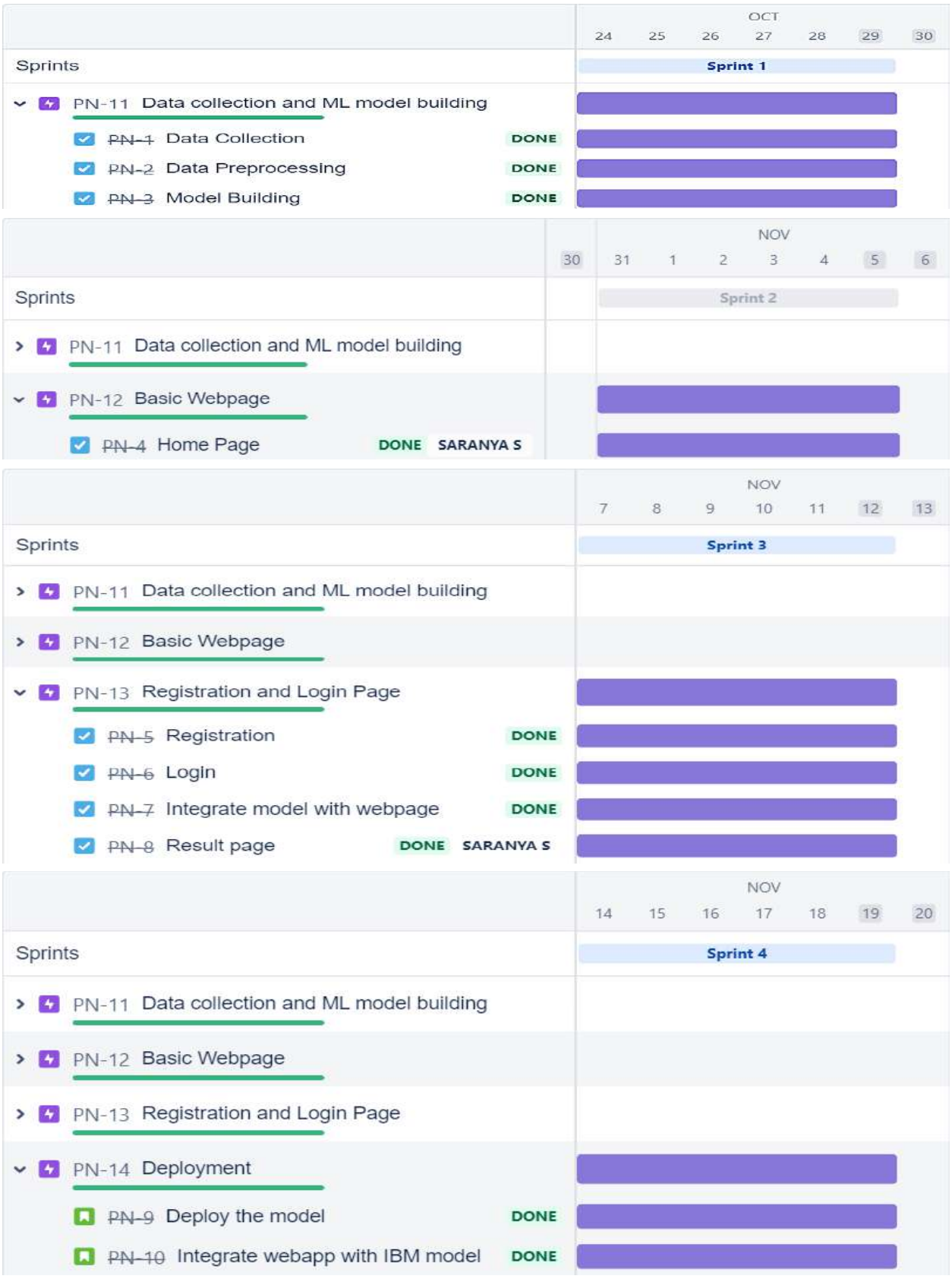
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint2	Data Pre-processing and EDA	USN-5	I need to prepare and process the data for model building and do pre-processing activities like data visualization.	6	High	Vivek S Nithish M Dilip T Manoj S Rohit R
Sprint3	Model Building (Training and Testing)	USN-6	Building the model using Data mining processes like Random Forest Classifier, K Nearest Neighbour Form Regression, classification, and clustering techniques.	4	Medium	Vivek S Nithish M DilipT Manoj S Rohit R
Sprint3	Application Building	USN-7	Building website for model application using HTML, CSS and JavaScript.	7	High	Vivek S Nithish M Dilip T Manoj S Rohit R
Sprint4	Model Verification	USN-8	Need to verify, whether developed model works with application.	4	High	Vivek S Nithish M Dilip T Manoj S Rohit R
Sprint4	Results	USN-9	As a user, I can get to know the results in addition to recommendations.	6	High	Vivek S Nithish M Dilip T Manoj S Rohit R

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	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
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	07 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	14 Nov 2022

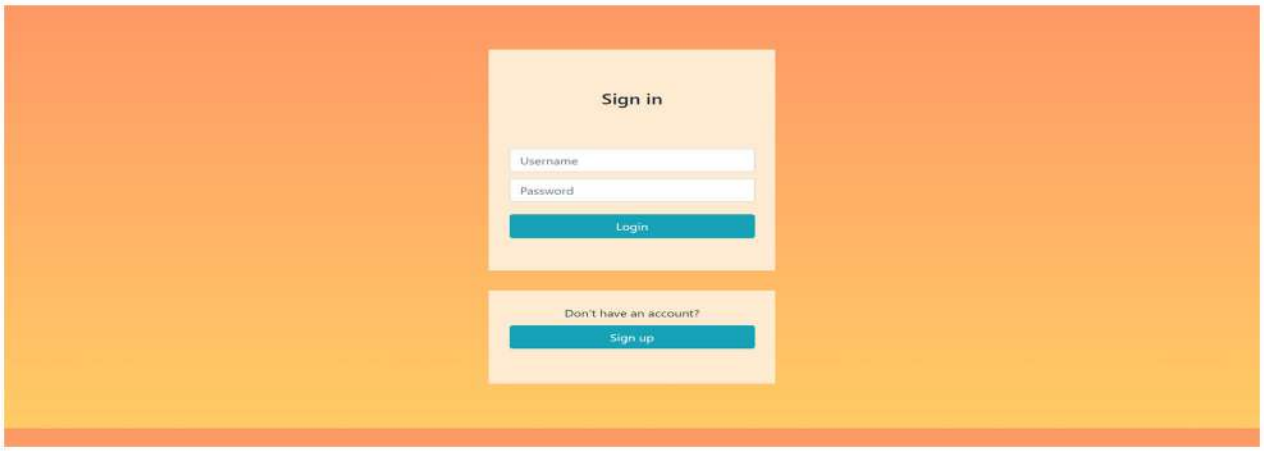
6.3 Reports from JIRA

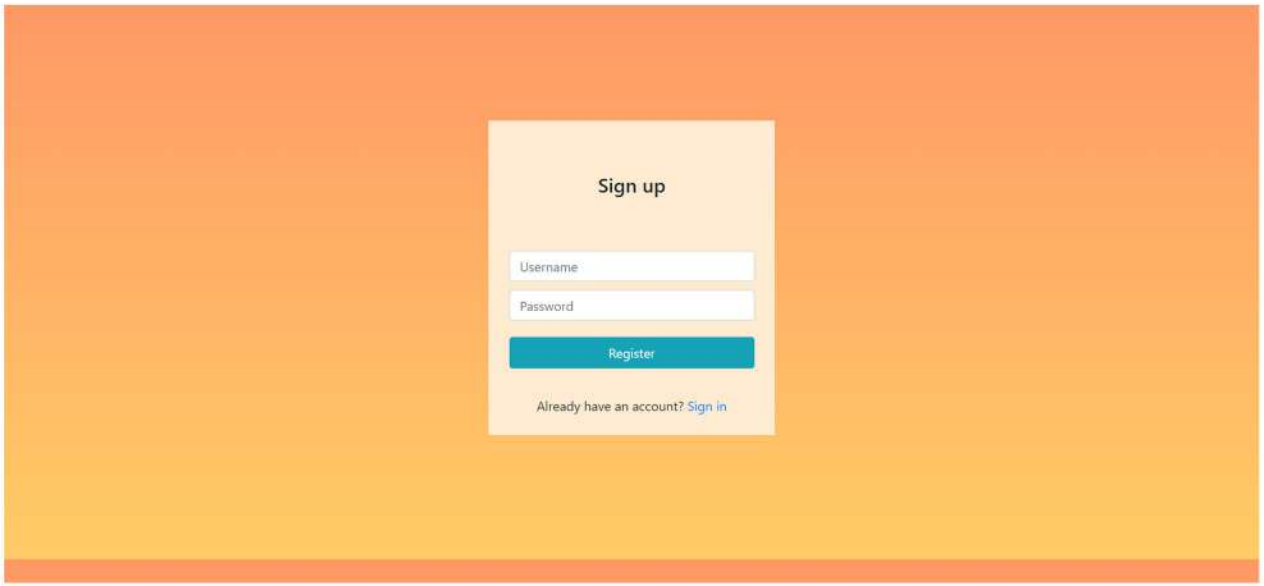


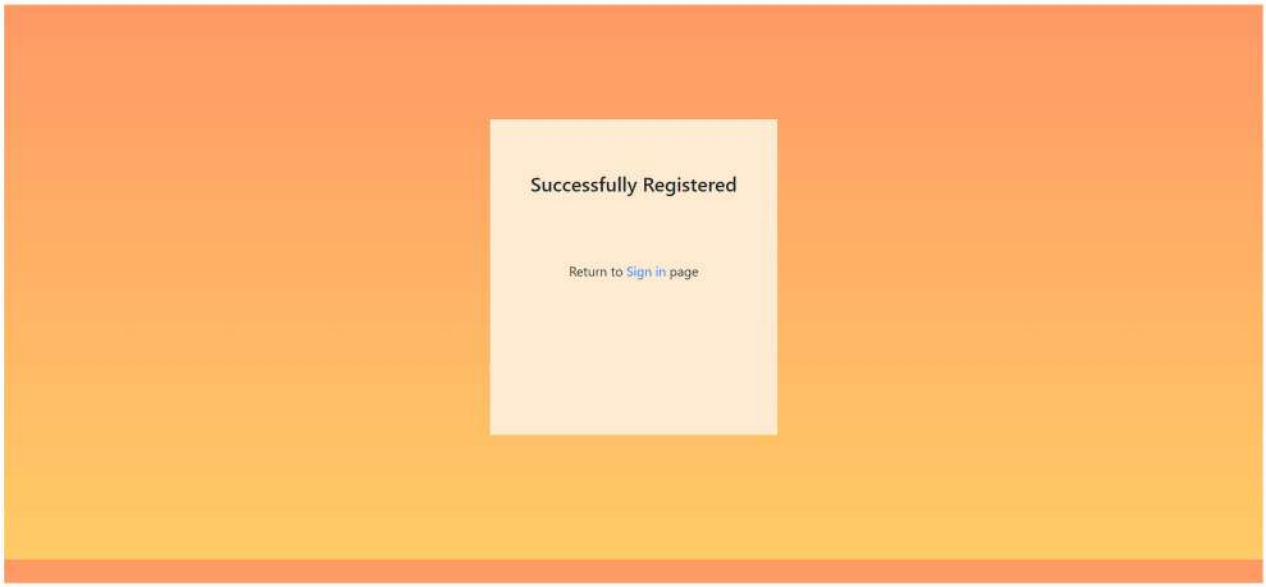

OCT
24 25 26 27 28 29 30
Sprints
Sprint 1
PN-11 Data collection and ML model building
PN-1 Data Collection DONE
PN-2 Data Preprocessing DONE
PN-3 Model Building DONE
NOV
30 31 1 2 3 4 5 6
Sprints
Sprint 2
PN-11 Data collection and ML model building
PN-12 Basic Webpage
PN-4 Home Page DONE SARANYA S
NOV
7 8 9 10 11 12 13
Sprints
Sprint 3
PN-11 Data collection and ML model building
PN-12 Basic Webpage
PN-13 Registration and Login Page
PN-5 Registration DONE
PN-6 Login DONE
PN-7 Integrate model with webpage DONE
PN-8 Result page DONE SARANYA S
NOV
14 15 16 17 18 19 20
Sprints
Sprint 4
PN-11 Data collection and ML model building
PN-12 Basic Webpage
PN-13 Registration and Login Page
PN-14 Deployment
PN-9 Deploy the model DONE
PN-10 Integrate webapp with IBM model DONE

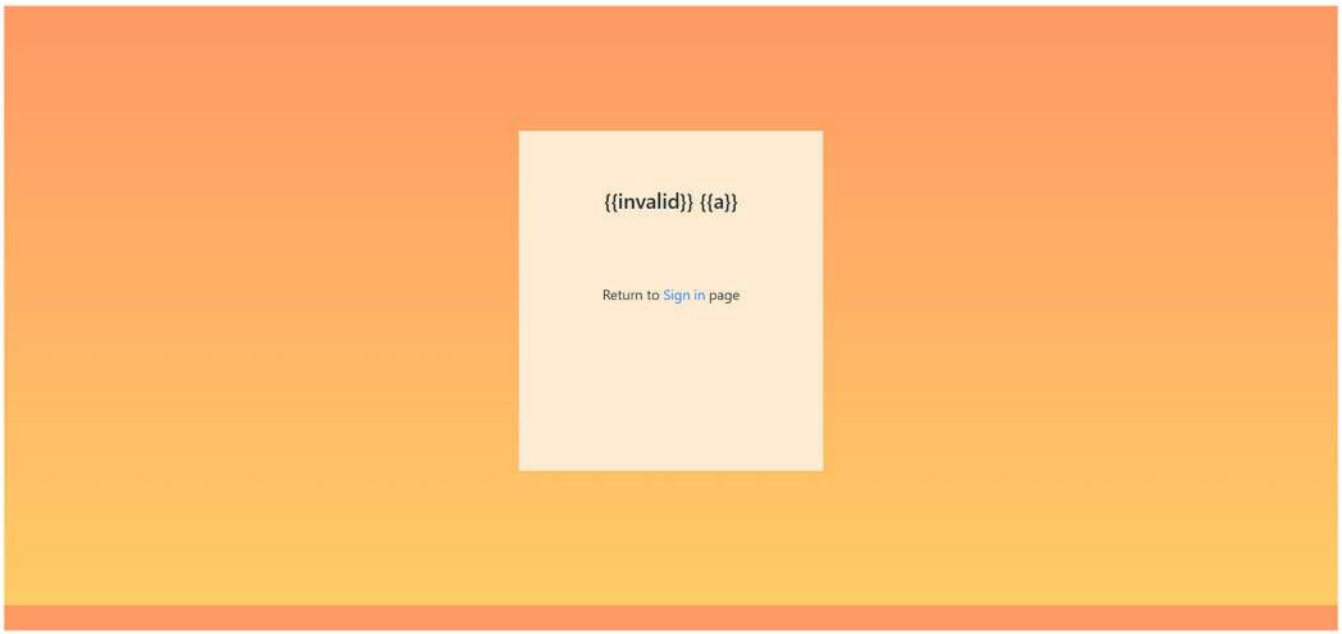
7. CODING & SOLUTIONING

7.1 Feature 1









Processing....

Home Information Predict

Add Image
Choose File No file chosen

Drag and drop a file or select add Image

your image
Remove Uploaded Image

predict

7.2 Feature 2

```python
from flask import Flask,redirect, request, url_for, render_template
import sqlite3 as sql
import os.path
import pickle
import cv2
from skimage import feature

app=Flask(__name__)

class DB:
    def _init_(self, name = 0):
        self._name = name
        self._lid = 0
        self._tot=0

    # getter method
    def get_name(self):
        return self._name

    # setter method
    def set_name(self, x):
        self._name = x
    def get_lid(self):
        return self._lid

    # setter method
    def set_lid(self, x):
        self._lid = x

    def get_tot(self):
        return self._tot
```

```python

    # setter method
    def set_tot(self, x):
        self._tot = x
obj=DB()
@app.route('/')
def home():
    try:
        lid=obj.get_lid()
        print(lid)
        return render_template('login.html',data=lid)
    except Exception as e:
        return render_template('login.html')


@app.route('/login')
def login():
    # obj._lid=0
    return render_template('login.html')
@app.route('/predict',methods=["GET","POST"])
def predict():
    if request.method == 'POST':
        f=request.files['file'] #requesting the file
        f.save(f.filename)
        # basepath=os.path.dirname(__file__)#storing the file directory
        # filepath=os.path.join(basepath, "uploads", f.filename)#storing the
        # f.save(filepath)#saving the file #Loading the saved model
        print("[INFO] loading model...")
        model = pickle.loads (open('model.pkl', "rb").read()) #pre-process the image in the same manner we did earlier
        image=cv2.imread(f.filename)
        output = image.copy() # load the input image, convert it to grayscale, and resize
        output = cv2.resize(output, (128, 128))
        image= cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        image = cv2.resize(image, (200, 200))
```

```python
        image= cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        image = cv2.resize(image, (200, 200))
        image = cv2.threshold(image, 0, 255,cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU) [1]

        # quantify the image and make predictions based on the extracted #features using the last trained Random Forest

        features = feature.hog(image, orientations=9,  pixels_per_cell=(10, 10), cells_per_block=(2, 2), transform_sqrt=True, block_norm="L1")

        preds=model.predict([features])

        print(preds)
        ls=["healthy","parkinson"]

        result=ls[preds[0]]
        print(result)
    # obj._lid=0
    return render_template('predict.html',d=result)
@app.route("/login", methods = ["GET","POST"])
def Login():

    l_id = request.form["logname"]
    l_pass  = request.form["logpass"]

    tab=l_id+l_pass
    #print("name")
    if request.method == 'POST':
        print("name")
        l_id = request.form["logname"]
        l_pass  = request.form["logpass"]

        tab=l_id+l_pass
        print(tab)
        obj.set_name(tab)
        obj.set_lid(l_id)
        return redirect(f"/check")
```

8. TESTING

8.1 Test Cases

TEST CASE ID	FEATURE TYPE	COMPONENT	TEST SCENARIO	PRE-REQUISITE	STEPS TO EXECUTE	TEST DATA	EXPECTED RESULT	ACTUAL RESULT	STATUS	COMMENTS	BUG ID	EXECUTED BY
BasicWebPageDesign_TC_001	UI	Home Page	Verify user is able to see the webpage with basic components	Install and setup visual studio code.	1.Enter Url and click go 2.Verify navbar with components like Home,Predict,Login is available	http://127.0.0.1:5000/	Basic navbar in the webpage should be displayed	As expected but redirection is not possible	PASS	Decent webpage is created.		SARANYA.S SRIMATHI.S SWATHI.A ROOBASHREE.S
BasicWebPageDesign_TC_002	Functional	Home Page	Verify redirection is possible when any components in navbar is clicked	Install python,flask and necessary packages	1.Enter Url and click go 2.Click the components in navbar	http://127.0.0.1:5000/	Redirect to the respective pages when user click the component in the navbar	Logout button is not redirected instead it throws an error	FAIL	Page not found error is thrown	BUG - 1	SARANYA.S SRIMATHI.S SWATHI.A ROOBASHREE.S
LoginPage_TC_003	UI	Login/Register Page	Verify whether login and register page is visible with required fields when clicked the URL	Integrate webpage with flask	1.Enter URL and click enter to go. 2.Application should display login and register tabs with respective fields.	http://127.0.0.1:5000/	Application should show below UI elements: 1.LOGIN tab a.User Id b.Password 2.REGISTER tab a.Email b.User Id(required) c.Password(required)	Working as expected	PASS	Successfully created login/register page.		SARANYA.S SRIMATHI.S SWATHI.A ROOBASHREE.S
LoginPage_TC_004	Functional	Login/Register Page	Verify whether it is possible to enter the valid details.		1.Enter URL(http://127.0.0.1:5000/) and click enter to go. 2.Click register tab 3.Enter invalid email id 4.click register button	Email : vivek1rajbhar@gmail.com	Should show an error as '@' is missing in email id.	Working as expected	PASS			SARANYA.S SRIMATHI.S SWATHI.A ROOBASHREE.S
LoginPage_TC_005	Functional	Login/Register Page	Verify when existing user try to register it throws an error message "User already exists".		1.Enter URL(http://127.0.0.1:5000/) and click enter to go. 2.Click Register tab. 3.Enter already registered username. 4.click Register button	Username : vivek Password : Test123456	Should show an validation error as user already exists.	Working as expected	PASS			SARANYA.S SRIMATHI.S SWATHI.A ROOBASHREE.S
LoginPage_TC_006	Functional	Login/Register Page	Verify user is able to log into application with invalid credentials.		1.Enter URL(http://127.0.0.1:5000/) and click enter to go. 2.Enter invalid/valid username in respective field. 3.Enter invalid/valid password. 4.click login button	Username : vivek Password : Test123456	Application should show 'Invalid user or password ' validation message.	Working as expected	PASS			SARANYA.S SRIMATHI.S SWATHI.A ROOBASHREE.S
LoginPage_TC_007	Functional	Login/Register Page	Verify user is able to log into application with valid credentials.		1.Enter URL(http://127.0.0.1:5000/) and click enter to go. 2.Enter valid username in respective field. 3.Enter valid password. 4.click login button	Username : vivek Password : Test123456	User should navigate to homepage.	Working as expected	PASS	Successfully logged in .		SARANYA.S SRIMATHI.S SWATHI.A ROOBASHREE.S
HomePage_TC_008	UI	Home Page	Verify user is able to see information on parkinsons disease.		1.Enter URL(http://127.0.0.1:5000/) and click enter to go. 2.Login with valid credentials. 3.Click Home button in navbar	Username : Nithish Password : Malli123456	User should be able to see information on parkinsons disease such as symptoms, cause,treatment.	Working as expected	PASS			SARANYA.S SRIMATHI.S SWATHI.A ROOBASHREE.S
PredictPage_TC_009	UI	Predict Page	Verify user is able to redirect to predict page .		1.Enter URL(http://127.0.0.1:5000/) and click enter to go. 2.Login with valid credentials. 3.Click Predict button in navbar	Username : Nithish Password : Malli123456	User is able to see choose and predict button in predict page with an NOTE message	Working as expected	PASS			SARANYA.S SRIMATHI.S SWATHI.A ROOBASHREE.S
PredictPage_TC_010	Functional	Predict Page	verify user is able to upload the image and get the result	Build an ML model for parkinsons disease prediction	1.Enter URL(http://127.0.0.1:5000/) and click enter to go. 2.Login with valid credentials. 3.Click Predict button in navbar 4.Click Choose button to upload an essential image. 5.Click Predict button to get the result.	Images : https://drive.google.com/drive/folders/1Y3m9E98m33v6L9AqWWgHIzvMA_TryQ8l?usp=sharing	User is able to upload pic from the computer and review the output	Working as expected	PASS	Predicted accurately.		SARANYA.S SRIMATHI.S SWATHI.A ROOBASHREE.S
Logout_TC_011	Functional	Logout	Verify user is able to logout	Login page is needed.	1.Enter URL(http://127.0.0.1:5000/) and click enter to go. 2.Login with valid credentials . 3.Click Logout button in navbar.	Username : vivek Password : Malli123456	User is able to click Logout button and redirect to Login/Regster page	Working as expected	PASS	BUG 1 is resolved		SARANYA.S SRIMATHI.S SWATHI.A ROOBASHREE.S

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**

Model Evaluation

In [12]:
```
predictions = model.predict(X_test)
predictions
```

Out[12]:
```
array([0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 0, 1, 1, 1], dtype=int64)
```

In [13]:
```
cm = confusion_matrix(y_test, predictions)
cm
```

Out[13]:
```
array([[14,  1],
       [ 3, 12]], dtype=int64)
```

In [14]:
```
accuracy = accuracy_score(y_test, predictions)
accuracy
```

Out[14]:
```
0.8666666666666667
```

In [15]:
```
cr = classification_report(y_test, predictions)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.82      0.93      0.87        15
           1       0.92      0.80      0.86        15

    accuracy                           0.87        30
   macro avg       0.87      0.87      0.87        30
weighted avg       0.87      0.87      0.87        30
```

9.2 Hyperparameter Tuning

In [7]:
```
from sklearn.model_selection import GridSearchCV
```

In [22]:
```
model = RandomForestClassifier()
```

In [23]:
```
parameters = {
    'max_depth' : [5,10,20,30,35],
    'random_state' : [0,1,2,3,4],
    'n_estimators' : [70,100,80,85,110]
}
```

In [24]:
```
grid = GridSearchCV(model,parameters,cv=5)
```

In [25]:
```
grid.fit(X_train, y_train)
```

Out[25]:
```
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [5, 10, 20, 30, 35],
                         'n_estimators': [70, 100, 80, 85, 110],
                         'random_state': [0, 1, 2, 3, 4]})
```

In [26]:
```
grid.best_params_
```

Out[26]:
```
{'max_depth': 5, 'n_estimators': 100, 'random_state': 2}
```

In [27]:
```
grid.best_estimator_
```

Out[27]:
```
RandomForestClassifier(max_depth=5, random_state=2)
```

In [28]:
```
grid.best_score_
```

Out[28]:
```
0.7923809523809524
```

```
In [29]: from sklearn.model_selection import RandomizedSearchCV
         rs = RandomizedSearchCV(model,parameters,cv=5)
         rs.fit(X_train, y_train)

Out[29]: RandomizedSearchCV(cv=5, estimator=RandomForestClassifier(),
                            param_distributions={'max_depth': [5, 10, 20, 30, 35],
                                                 'n_estimators': [70, 100, 80, 85, 110],
                                                 'random_state': [0, 1, 2, 3, 4]})

In [30]: rs.best_params_

Out[30]: {'random_state': 2, 'n_estimators': 100, 'max_depth': 20}

In [31]: rs.best_score_

Out[31]: 0.7780952380952382
```

10. ADVANTAGES & DISADVANTAGES

10.1 Advantages

- Less time consuming
- More accuracy in the model
- Easily implemented

10.2 Disadvantages

- Packages to be installed
- Data collection is difficult

11. CONCLUSION

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 automatically detect Parkinson's disease in hand-drawn images of spirals and waves. Here, we presented included studies in a high-level summary, providing access to information including (a) machine learning methods that have been used in the diagnosis of PD and associated outcomes, (b) types of clinical, behavioral and biometric data that could be used for rendering more accurate diagnoses, (c) potential biomarkers for assisting clinical decision making, and (d) 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. Machine learning approaches therefore have the potential to provide clinicians with additional tools to screen, detect or diagnose PD.

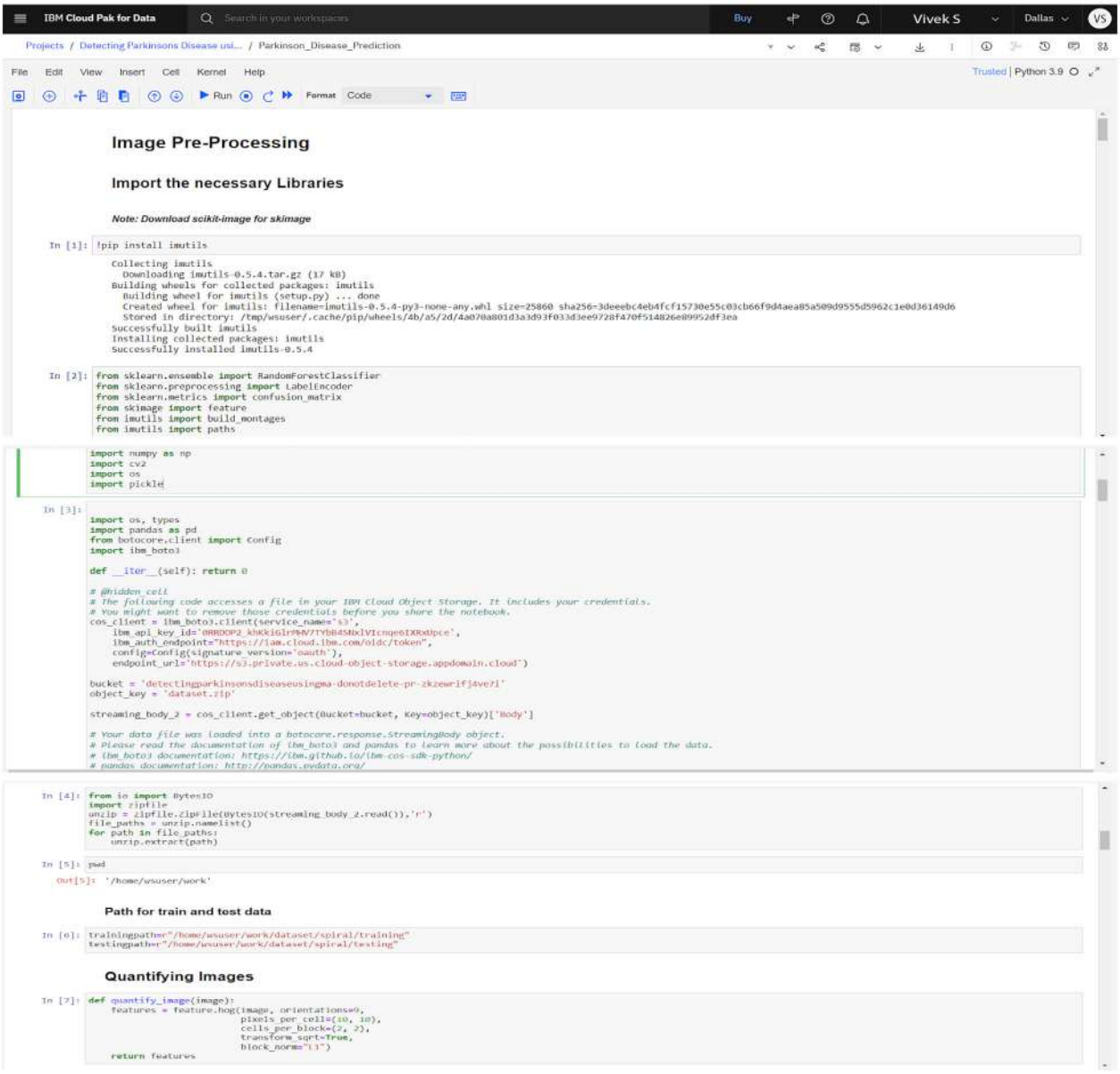
12. FUTURE SCOPE

The model can be trained with enormous amount of data to improve the accuracy. We can also merge the voice dataset and train the model accordingly for higher productivity.

13. APPENDIX

13.1 Source Code

Parkinson's_Disease_Prediction.ipynb：



IBM Cloud Pak for Data

Projects / Detecting Parkinsons Disease usi... / Parkinson_Disease_Prediction

File Edit View Insert Cell Kernel Help

Trusted | Python 3.9

Run Format Code

Image Pre-Processing

Import the necessary Libraries

Note: Download scikit-image for skimage

```
In [1]: !pip install imutils
```

```
Collecting imutils
  Downloading imutils-0.5.4.tar.gz (17 kB)
Building wheels for collected packages: imutils
  Building wheel for imutils (setup.py) ... done
  Created wheel for imutils: filename=imutils-0.5.4-py3-none-any.whl size=25860 sha256=3deeebc4eb4fcf15730e55c03cb66f9d4aea85a509d9555d5962c1e0d36149d6
  Stored in directory: /tmp/wsuser/.cache/pip/wheels/4b/a5/2d/4a070a801d3a3d93f033d3ee9728f470f514826e80952df3ea
Successfully built imutils
Installing collected packages: imutils
Successfully installed imutils-0.5.4
```

```
In [2]: from sklearn.ensemble import RandomForestClassifier
        from sklearn.preprocessing import LabelEncoder
        from sklearn.metrics import confusion_matrix
        from skimage import feature
        from imutils import build_montages
        from imutils import paths
```

```
        import numpy as np
        import cv2
        import os
        import pickle
```

```
In [3]:
        import os, types
        import pandas as pd
        from botocore.client import Config
        import ibm_boto3

        def __iter__(self): return 0

        # @hidden_cell
        # The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
        # You might want to remove those credentials before you share the notebook.
        cos_client = ibm_boto3.client(service_name='s3',
            ibm_api_key_id='0RRDOP2_khKkiGlr9kN7TYbB4SNxlVIcnqeeOIXRxUpce',
            ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
            config=Config(signature_version='oauth'),
            endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

        bucket = 'detectingparkinsonsdiseaseusingma-donotdelete-pr-zkzewrlfj4ve7l'
        object_key = 'dataset.zip'

        streaming_body_2 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']

        # Your data file was loaded into a botocore.response.StreamingBody object.
        # Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
        # ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
        # pandas documentation: http://pandas.pydata.org/
```

```
In [4]: from io import BytesIO
        import zipfile
        unzip = zipfile.ZipFile(BytesIO(streaming_body_2.read()),'r')
        file_paths = unzip.namelist()
        for path in file_paths:
            unzip.extract(path)
```

```
In [5]: pwd
```

```
Out[5]: '/home/wsuser/work'
```

Path for train and test data

```
In [6]: trainingpath=r"/home/wsuser/work/dataset/spiral/training"
        testingpath=r"/home/wsuser/work/dataset/spiral/testing"
```

Quantifying Images

```
In [7]: def quantify_image(image):
            features = feature.hog(image, orientations=9,
                                   pixels_per_cell=(10, 10),
                                   cells_per_block=(2, 2),
                                   transform_sqrt=True,
                                   block_norm="L1")
            return features
```

Loading Train Data and Test Data

In [8]:
```
def load_split(path):
    imagePaths = list(paths.list_images(path))
    data = []
    labels = []

    for imagePath in imagePaths:
        label = imagePath.split(os.path.sep)[-2]

        image = cv2.imread(imagePath)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        image = cv2.resize(image, (200, 200))

        image=cv2.threshold(image,0,255,cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]

        features = quantify_image(image)

        data.append(features)
        labels.append(label)

    return (np.array(data), np.array(labels))
```

Load the train and test data

In [9]:
```
print("[INFO] loading data...")
(X_train, y_train) = load_split(trainingpath)
(X_test, y_test) = load_split(testingpath)
```

```
[INFO] loading data...
```

Label Encoding

In [10]:
```
le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)
print(X_train.shape,y_train.shape)
```

```
(72, 12996) (72,)
```

Model Building

Training The Model

In [11]:
```
print("[INFO] training model")
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)
```

```
[INFO] training model
```

Out[11]:
```
RandomForestClassifier()
```

Testing The Model

In [12]:
```
testingpathslist(paths.list_images(testingpath))
idxs=np.arange(0,len(testingpath))
idxs=np.random.choice(idxs,size=(25,),replace=False)
images=[]
```

In [13]:
```
for i in idxs:
    image=cv2.imread(testingpath[i])
    output=image.copy()

    # load the input image,convert to grayscale and resize
```

In [13]:
```
for i in idxs:
    image=cv2.imread(testingpath[i])
    output=image.copy()

    # load the input image,convert to grayscale and resize

    output=cv2.resize(output,(128,128))
    image=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    image=cv2.resize(image,(200,200))
    image=cv2.threshold(image,0,255,cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]

    #quantify the image and make predictions based on the  extracted feature using last trained random forest
    features=quantify_image(image)
    preds=model.predict([features])
    label=le.inverse_transform(preds)[0]
    #the set of output images
    if label=="healthy":
        color=(0,255,0)
    else:
        color=(0,0,255)

    cv2.putText(output,label,(3,20),cv2.FONT_HERSHEY_SIMPLEX,0.5,color,2)
    images.append(output)

#creating a montage
montage=build_montages(images,(128,128),(5,5))[0]
cv2.imshow("Output",montage)
cv2.waitKey(0)
```

Model Evaluation

In [57]:
```
predictions = model.predict(X_test)
predictions
```

Out[57]:
```
array([0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1,
       1, 0, 1, 1, 0, 1, 0, 1], dtype=int64)
```

In [58]:
```
cm = confusion_matrix(y_test, predictions)
cm
```

Out[58]:
```
array([[11,  4],
       [ 6,  9]], dtype=int64)
```

In [59]:
```
accuracy = accuracy_score(y_test, predictions)
accuracy
```

Out[59]:
```
0.6666666666666666
```

In [60]:
```
cr = classification_report(y_test, predictions)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.65      0.73      0.69        15
           1       0.69      0.60      0.64        15

    accuracy                           0.67        30
   macro avg       0.67      0.67      0.67        30
weighted avg       0.67      0.67      0.67        30
```

Save The Model

In [62]:
```
pickle.dump(model,open('parkinson.pkl','wb'))
```

Deployment

In [16]:
```
!pip install -U ibm-watson-machine-learning
```

```
Requirement already satisfied: ibm-watson-machine-learning in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.257)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (0.8.9)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (1.26.7)
Requirement already satisfied: packaging in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (21.3)
Requirement already satisfied: importlib-metadata in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (4.8.2)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2.26.0)
Requirement already satisfied: pandas<1.5.0,>=0.24.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (1.3.4)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2022.9.24)
Requirement already satisfied: ibm-cos-sdk==2.11.* in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2.11.0)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (0.3.3)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learning)
(2.11.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learni
ng) (2.11.0)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (0.1
0.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk-core==2.11.0->ibm-cos-sdk==2.11.*->ib
m-watson-machine-learning) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas<1.5.0,>=0.24.2->ibm-watson-machine-learning) (2021.3)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas<1.5.0,>=0.24.2->ibm-watson-machine-learning) (1.20.3)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->ibm-cos-sdk-core==2.11.0->ibm-cos-sdk==
2.11.*->ibm-watson-machine-learning) (1.15.0)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->ibm-watson-machine-learning) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->ibm-watson-machine-learning) (3.3)
Requirement already satisfied: zipp>=0.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from importlib-metadata->ibm-watson-machine-learning) (3.6.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from packaging->ibm-watson-machine-learning) (3.0.4)
```

In [17]:
```
# Now connect notebook ml service with api key and url

from ibm_watson_machine_learning import APIClient
import json
import numpy as np
```

Authenticate and Set Space

In [18]:
```
wml_credentials = {
    "apikey" : "RYa2JTvIsfgzBUbvFxnCYVUxLBOntmTWzc9KGStjRtC5",
    "url" : "https://us-south.ml.cloud.ibm.com" #For Dallas region
}
```

In [19]:
```
wml_client =APIClient(wml_credentials)
```

In [20]:
```
# Check the available deployments

wml_client.spaces.list()
```

```
Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50
------------------------------------  ------------------------  -------------------------
ID                                    NAME                      CREATED
efa48345-def9-4aa5-b19f-4dd7d5f766ce  ParkinsonDiseaseDetection  2022-11-06T10:09:49.894Z
------------------------------------  ------------------------  -------------------------
```

In [21]:
```
SPACE_ID = "efa48345-def9-4aa5-b19f-4dd7d5f766ce"
```

Show desktop

In [22]:
```
# Space id created default one

wml_client.set.default_space(SPACE_ID)
```

Out[22]: 'SUCCESS'

In [23]:
```
# To check the environment

wml_client.software_specifications.list()
```

```
runtime-22.1-py3.9                12b83a17-24d8-5082-900f-0ab31fbfd3cb  base
scikit-learn_0.22-py3.6           154010fa-5b3b-4ac1-82af-4d5ee5abbc85  base
default_r3.6                      1b70aec3-ab34-4b87-8aa0-a4a3c8296a36  base
pytorch-onnx_1.3-py3.6            1bc6029a-cc97-56da-b8e0-39c3880dbbe7  base
kernel-spark3.3-r3.6              1c9e5454-f216-59dd-a20e-474a5cdf5988  base
pytorch-onnx_rt22.1-py3.9-edt     1d362186-7ad5-5b59-8b6c-9d0880bde37f  base
tensorflow_2.1-py3.6              1eb25b84-d6ed-5dde-b6a5-3fbdf1665666  base
spark-mllib_3.2                   20047f72-0a98-58c7-9ff5-a77b012eb8f5  base
tensorflow_2.4-py3.8-horovod      217c16f6-178f-56bf-824a-b19f20564c49  base
runtime-22.1-py3.9-cuda           26215f05-08c3-5a41-a1b0-da66306ce658  base
do_py3.8                          295addb5-9ef9-547e-9bf4-92ae3563e720  base
autoai-ts_3.8-py3.8               2aa0c932-798f-5ae9-abd6-15e0c2402fb5  base
tensorflow_1.15-py3.6             2b73a275-7cbf-420b-a912-eae7f436e0bc  base
kernel-spark3.3-py3.9             2b7961e2-e3b1-5a8c-a491-482c8368839a  base
pytorch_1.2-py3.6                 2c8ef57d-2687-4b7d-acce-01f94976dac1  base
spark-mllib_2.3                   2e51f700-bca0-4b0d-88dc-5c6791338875  base
pytorch-onnx_1.1-py3.6-edt        32083cea-3f32-4400-8965-dde874a8d67e  base
spark-mllib_3.0-py37              36507ebe-8770-55ba-ab2a-eafe787600e9  base
spark-mllib_2.4                   390d21f8-e58b-4fac-9c55-d7ceda621326  base
```

Save and Deploy the Model

In [24]:
```
import sklearn
sklearn.__version__
```

Out[24]: '1.0.2'

In [25]:
```
MODEL_NAME = "ParkinsonDiseaseDetection_DeployedModel"
DEPLOYMENT_NAME = "ParkinsonDiseaseDetection"
```

In [26]:
```
# Set Python default version

software_spec_uid = wml_client.software_specifications.get_id_by_name('runtime-22.1-py3.9')
```

Create Model Properties to deploy the model

In [27]:
```
# Setup Model Meta

model_props = {
    wml_client.repository.ModelMetaNames.NAME: MODEL_NAME,
    wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0',
    wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
}
```

In [28]:
```
# Save Model

model_details = wml_client.repository.store_model(
    model = model,
    meta_props = model_props,
    training_data = X_train,
    training_target = y_train
)
```

In [29]:
```
model_details
```

```
'schemas': {'input': [{'fields': [{'name': 'f0', 'type': 'float'},
    {'name': 'f1', 'type': 'float'},
    {'name': 'f2', 'type': 'float'},
    {'name': 'f3', 'type': 'float'},
    {'name': 'f4', 'type': 'float'},
    {'name': 'f5', 'type': 'float'},
    {'name': 'f6', 'type': 'float'},
    {'name': 'f7', 'type': 'float'},
    {'name': 'f8', 'type': 'float'},
    {'name': 'f9', 'type': 'float'},
    {'name': 'f10', 'type': 'float'},
    {'name': 'f11', 'type': 'float'},
    {'name': 'f12', 'type': 'float'},
    {'name': 'f13', 'type': 'float'},
    {'name': 'f14', 'type': 'float'},
    {'name': 'f15', 'type': 'float'},
    {'name': 'f16', 'type': 'float'},
    {'name': 'f17', 'type': 'float'},
    {'name': 'f18', 'type': 'float'},
    {'name': 'f19', 'type': 'float'},
```

In [30]:
```
model_id = wml_client.repository.get_model_id(model_details)
model_id
```

Out[30]: '7d936b97-a55f-403a-9624-5ad86e18e6b0'

Deploy in props

In [31]:
```
# Set meta

deployment_props = {
    wml_client.deployments.ConfigurationMetaNames.NAME : DEPLOYMENT_NAME,
    wml_client.deployments.ConfigurationMetaNames.ONLINE : {}
}
```

In [32]:
```
# Deploy

deployment = wml_client.deployments.create(
    artifact_uid = model_id,
    meta_props = deployment_props
)
```

```
#######################################################################################

Synchronous deployment creation for uid: '7d936b97-a55f-403a-9624-5ad86e18e6b0' started

#######################################################################################


initializing
Note: online_url is deprecated and will be removed in a future release. Use serving_urls instead.

ready


------------------------------------------------------------------------------------------------
Successfully finished deployment creation, deployment_uid='cbe26807-da09-4ca5-919f-3b00aa88f433'
------------------------------------------------------------------------------------------------
```

App.py：

In [1]:

```
from flask import Flask,request,render_template
import pickle
import cv2
from skimage import feature
import os.path

app = Flask(__name__)


@app.route('/')
def hello_world():
    return render_template("index.html")
class my_dictionary(dict):
  def __init__(self):
    self = dict()
  def add(self, key, value):
    self[key] = value
database=my_dictionary()

@app.route('/form_reg',methods=['POST','GET'])
def reg():
    name2=request.form['userid']
    pwd1=request.form['pwd']
    database.add(name2,pwd1)
    return render_template("index.html")
@app.route('/form_login',methods=['POST','GET'])
def login():
    name1=request.form['userid']
    pwd=request.form['pwd']
    if name1 not in database:
            return render_template('index.html',info='Invalid User!!')
    else:
        if database[name1]!=pwd:
            return render_template('index.html',info='Invalid Password!!')
        else:
                return render_template('home.html',name=name1)
@app.route("/")
def about():
    return render_template("home.html")#rendering html page
```

```
@app.route("/home")
def home():
    return render_template("home.html")

@app.route("/upload")
def test():
    return render_template("pred.html")

@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        f=request.files['file'] #requesting the file
        basepath=os.path.dirname(os.path.realpath('__file__'))#storing the file directory
        filepath=os.path.join(basepath,"uploads",f.filename)#storing the file in uploads folder
        f.save(filepath)#saving the file

        #Load the saved model
        print("[INFO] loading model...")
        model = pickle.loads(open('parkinson_Deploy.pkl', "rb").read())

        # pre-process the image in the same manner we did earlier
        image = cv2.imread(filepath)
        output = image.copy()

        # load the input image, convert it to grayscale, and resize
        output = cv2.resize(output, (128, 128))
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        image = cv2.resize(image, (200, 200))
        image = cv2.threshold(image, 0, 255,
        cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]

        # Quantify the image and make predictions based on the extracted features using the last trained Random Forest.
        features = feature.hog(image, orientations=9,
        pixels_per_cell=(10, 10), cells_per_block=(2, 2),
        transform_sqrt=True, block_norm="L1")
        preds = model.predict([features])
        print(preds)
        ls=["healthy","parkinson"]
        result = ls[preds[0]]
        return result
    return None
```

```
if __name__ == '__main__':
    app.run()
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [12/Nov/2022 20:32:46] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [12/Nov/2022 20:32:46] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [12/Nov/2022 20:32:46] "GET /static/css/bg.jpg HTTP/1.1" 304 -
127.0.0.1 - - [12/Nov/2022 20:32:55] "POST /form_reg HTTP/1.1" 200 -
127.0.0.1 - - [12/Nov/2022 20:32:55] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [12/Nov/2022 20:32:55] "GET /static/css/bg.jpg HTTP/1.1" 304 -
127.0.0.1 - - [12/Nov/2022 20:33:01] "POST /form_login HTTP/1.1" 200 -
127.0.0.1 - - [12/Nov/2022 20:33:04] "GET /upload HTTP/1.1" 200 -
127.0.0.1 - - [12/Nov/2022 20:33:04] "GET /static/js/main.js HTTP/1.1" 304 -
127.0.0.1 - - [12/Nov/2022 20:33:04] "GET /static/css/main.css HTTP/1.1" 304 -
[INFO] loading model...
127.0.0.1 - - [12/Nov/2022 20:33:15] "POST /predict HTTP/1.1" 200 -
[1]
```

In STATIC Folder

main.css：

```css
.img-preview {
    width: 256px;
    height: 256px;
    position: relative;
    border: 5px solid #F8F8F8;
    box-shadow: 0px 2px 4px 0px rgba(0, 0, 0, 0.1);
    margin-top: 1em;
    margin-bottom: 1em;
}

.img-preview>div {
    width: 100%;
    height: 100%;
    background-size: 256px 256px;
    background-repeat: no-repeat;
    background-position: center;
}

input[type="file"] {
    display: none;
}

.upload-label{
    display: inline-block;
    padding: 12px 30px;
    background: #39D2B4;
    color: #fff;
    font-size: 1em;
    transition: all .4s;
    cursor: pointer;
}

```

```css

.upload-label:hover{
    background: #34495E;
    color: #39D2B4;
}

.loader {
    border: 8px solid #f3f3f3; /* Light grey */
    border-top: 8px solid #3498db; /* Blue */
    border-radius: 50%;
    width: 50px;
    height: 50px;
    animation: spin 1s linear infinite;
}

@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}
```

style.css：

```css
*{
    margin: 0;
    padding: 0;
    font-family: sans-serif;
}

.hero{
    height: 100%;
    width: 100%;
    background-image: linear-gradient(rgba(0,0,0,0.4), rgba(0,0,0,0.4)),url(bg.jpg);
    background-position: center;
    background-size: cover;
    position: absolute;
}
.form-box{
    height: 380px;
    width: 360px;
    position: relative;
    margin: 6% auto;
    background: #fff;
    padding: 5px;
    overflow: hidden;
}
.button-box{
    width: 220px;
    margin: 35px auto;
    position: relative;
    box-shadow: 0 0 20px 9px #5f97e51f;
    border-radius: 40px;
}
```

```css
.toggle-btn{
    padding: 10px 30px;
    cursor: pointer;
    background: transparent;
    border: 0;
    outline: none;
    position: relative;
}
#btn{
    top: 0;
    left: 0;
    position: absolute;
    width: 110px;
    height: 100%;
    background: linear-gradient(to right,  #7369ca,#11b1c3);
    border-radius: 30px;
    transition: 0.5s;
}
.input-group{
    top: 120px;
    position: absolute;
    width: 280px;
    transition: .5s;

}
.input-field{
    width: 100%;
    padding: 10px 0;
    margin: 5px 0;
    border-left: 0;
    border-top: 0;
    border-right: 0;
    border-bottom: 1px solid #999;
    outline: none;
    background: transparent;
}
```

```css
.submit-btn{
    width: 85%;
    padding: 10px 30px;
    cursor: pointer;
    display: block;
    margin: auto;
    background: linear-gradient(to right, #4e4888,#7bc0c8);
    border: 0;
    outline: none;
    border-radius: 30px;
}
.check-box{
    margin: 30px 10px 30px 0;
}
span{
    color: #777;
    font-size: 12px;
    bottom: 68px;
    position: absolute;

}
#login{
    left: 50px;
}
#register{
    left: 450px;
}
.err{
    color:rgb(198, 156, 243);
    margin: 265px 0 0 145px;
}
```

main.js：

```
$(document).ready(function () {
    // Init
    $('.image-section').hide();
    $('.loader').hide();
    $('#result').hide();

    // Upload Preview
    function readURL(input) {
        if (input.files && input.files[0]) {
            var reader = new FileReader();
            reader.onload = function (e) {
                $('#imagePreview').css('background-image', 'url(' + e.target.result + ')');
                $('#imagePreview').hide();
                $('#imagePreview').fadeIn(650);
            }
            reader.readAsDataURL(input.files[0]);
        }
    }
    $("#imageUpload").change(function () {
        $('.image-section').show();
        $('#btn-predict').show();
        $('#result').text('');
        $('#result').hide();
        readURL(this);
    });

    // Predict
    $('#btn-predict').click(function () {
        var form_data = new FormData($('#upload-file')[0]);

        // Show loading animation
        $(this).hide();
        $('.loader').show();

```

```

            // Make prediction by calling api /predict
            $.ajax({
                type: 'POST',
                url: '/predict',
                data: form_data,
                contentType: false,
                cache: false,
                processData: false,
                async: true,
                success: function (data) {
                    // Get and display the result
                    $('.loader').hide();
                    $('#result').fadeIn(600);
                    $('#result').text('Prediction : '+data);
                    console.log('Success!');
                },
            });
        });

    });
```

In TEMPLATE folder

base.html：

templates > <> base.html > ...

```
<html lang="en">

    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title>Predict</title>
        <link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">
        <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
        <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
        <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
        <link href="{{ url_for('static', filename='css/main.css') }}" rel="stylesheet">
    <style>
    .bar
    {
    margin: 0px;
    padding:20px;
    background-color: rgb(160, 223, 241);
    opacity:0.6;
    color: black;
    font-family:'Roboto',sans-serif;
    font-style: italic;
    border-radius:20px;
    font-size:25px;
    }
    a
    {
    color: grey;
    float:right;
    text-decoration:none;
    font-style:normal;
    padding-right:20px;
    }
    a:hover{
    background-color: black;
    color: white;
    border-radius:15px;
    font-size:30px;
    padding-left:10px;
    }
    body{
            background-image: url("https://img.freepik.com/free-vector/clean-medical-background_53876-97927.jpg?w=2000");
            position: relative;
            background-size: cover;
            background-repeat: no-repeat;
            height: 100%;
            width: 100%;
        }
        h1{
            font-size:40px;
            text-align:center;
            color: rgb(20, 176, 204);
            font-style:italic;
            font-weight:bolder;
        }
        h2{
            font-size:35px;
            text-align:center;
            color: rgb(17, 196, 227);
            font-style:italic;
            font-weight:bolder;
        }
        h5{
            font-size:25px;
            text-align:center;
            color: rgb(53, 134, 152);
            font-weight:bolder;
        }
    </style>
    </head>
```

```
    <body>

        <div class="bar">
            <a href="/logout" >Logout</a>
            <a href="/upload" >Predict</a>
            <a href="/home">Home</a>
            <br>
        </div>
        <h1>Prevention is better than cure!</h1>
        <h2><center>Parkinson Classifier</center></h2>
        <h5>NOTE: Upload an spiral or wave page drawn by the user in a white sheet</h5>
        <div class="container">
            <center> <div id="content" style="margin-top:2em">{% block content %}{% endblock %}</div></center>
        </div>
    </body>

    <footer>
        <script src="{{ url_for('static', filename='js/main.js') }}" type="text/javascript"></script>
    </footer>

    </html>
```

home.html：

templates > home.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>HomePage</title>
    <style>
        body{
            background-image: linear-gradient(rgba(218, 185, 231, 0.9), rgba(0,0,0,0.4)),url("https://cdn-prod.medicalnewstoday.com/cont
            position: relative;
            background-size: cover;
            background-repeat: no-repeat;
            height: 100%;
            width: 100%;
        }
        h3{
            text-align:center;
            color:white;
        }
        .main{
            margin-top:100px;
        }
        p{
            color:white;
            text-indent:10px;
            margin:10px;
            font-size:20px;
        }
        .navbar{
        margin: 0px;
        padding:20px;
        background-color:rgb(169, 120, 159);
        opacity:0.6;
        color:black;
        font-family:'Roboto',sans-serif;
        font-style: italic;
```

```
            border-radius:20px;
            font-size:25px;
            }
            a{
            color: rgb(11, 3, 21);
            float:right;
            text-decoration:none;
            font-style:normal;
            padding-right:20px;
            }
            a:hover{
            background-color: black;
            color: white;
            border-radius:15px;
            font-size:30px;
            padding-left:10px;
            }
            img{
            width:450px;
            height:400px;
            padding:25px;

            }
            img:hover{
            border-radius:100px;
            border-color: grey;
            }
            #im{
            width:1450px;
            height:700px;
            padding:25px;
            }
        </style>
    </head>
```

```
<body>
    <div class="navbar">
        <a href="/logout" >Logout</a>
        <a href="/upload" >Predict</a>
        <a href="/home">Home</a>
        <a>Welcome {{name}} !</a>
        <br>
    </div>
    <br>
    <center><b class="pd"><font color="white" size="15" font-family="Comic Sans MS" >Detecting Parkinson Disease using ML</font></b></center>
    <div>
    <br>
    <br>
    <center>
    <p>Parkinson's disease (PD) is a prevalent neurodegenerative disease affecting about 1% of the world population over the age of 55 (Nussbaum and
    </center>
    <span><img src="https://www.narayanahealth.org/blog/wp-content/uploads/2015/04/parkinson.png" title="Disease"></span>
    <span><img src="https://stanfordmedicine25.stanford.edu/the25/parkinsondisease/_jcr_content/main/panel_builder_0/panel_0/panel_builder_0/panel_0
    <span><img src="https://www.verywellhealth.com/thmb/AaqoBqM3QDHSNHCt_DlKCNeWoUk=/1500x0/filters:no_upscale():max_bytes(150000):strip_icc()/zhans
    <span><img src="https://www.gutmicrobiotaforhealth.com/wp-content/uploads/2016/12/parkinson.jpg" title="Effect"></span>
    <span><img src="https://i.pinimg.com/originals/02/16/e4/0216e4b8a5db4d6e2a3f7043eaf7dc32.jpg" title="Cause"></span>
    <span><img src="https://jnnp.bmj.com/content/jnnp/91/8/795/F4.large.jpg" title="diagnosis"></span>
    <span><img id="im" src="https://www.genengnews.com/wp-content/uploads/2019/06/203938_web.jpg" title="Stage"></span>
    <br><br><br><br><br>
    </div>
</body>
</html>
```

index.html :

templates > index.html > html

```
<html>
    <head>
        <title>PARKINSON'S DISEASE </title>
        <link rel = "stylesheet" href="{{url_for('static',filename='css/style.css')}}">
    </head>
    <body>
        <div class="hero">
            <div class="form-box">
                <div class="button-box">
                    <div id="btn"></div>
                    <button type="button" class="toggle-btn" onclick="login()">Log In</button>
                    <button type="button" class="toggle-btn" onclick="register()">Register</button>
                </div>
                <form id="login" class="input-group" action="/form_login" method="post">
                    <input type="text" class="input-field" placeholder="User Id" name ="userid" required>
                    <input type="password" class="input-field" placeholder="Password" name="pwd" required>
                    <input type="checkbox" class="check-box"><span>Remember Password</span>
                    <button type="submit" class="submit-btn" value="Login">Login</button>
                </form>
                <h6 class="err">{{info}}</h6>
                <form id="register" class="input-group" action="/form_reg" method="post">
                    <input type="email" class="input-field" placeholder="Email Id">
                    <input type="text" class="input-field" placeholder="User Id" name ="userid" required>
                    <input type="password" class="input-field" placeholder="Password" name="pwd" required>
                    <button type="submit" id = "sub" class="submit-btn" >Register</button>
                </form>
                <h6 class="err">{{info}}</h6>
            </div>
        </div>
        <script>
            var x = document.getElementById("login")
            var y = document.getElementById("register")
            var z = document.getElementById("btn")
            function register(){
                x.style.left = "-400px";
                y.style.left = "50px";
                z.style.left = "110px";
            }
            function login(){
                x.style.left = "50px";
                y.style.left = "450px";
                z.style.left = "0px";
            }
        </script>
    </body>
</html>
```

pred.html :

templates > pred.html > div > form#upload-file

```
{% extends "base.html" %} {% block content %}

<h2><center>Parkinson Classifier</center></h2>

<div>
    <form id="upload-file" method="post" enctype="multipart/form-data">
    <center>    <label for="imageUpload" class="upload-label">
            Choose...
        </label>
        <input type="file" name="file" id="imageUpload" accept=".png, .jpg, .jpeg">
    </center></form>

   <center> <div class="image-section" style="display:none;">
        <div class="img-preview">
            <div id="imagePreview">
            </div></center>
        </div>
        <center><div>
            <button type="button" class="btn btn-primary btn-lg " id="btn-predict">Predict!</button>
        </div></center>
    </div>

    <div class="loader" style="display:none;"></div>

    <h3 id="result">
        <span> </span>
    </h3>

</div>

</div>

{% endblock %}
```

13.2 Github Link

https://github.com/IBM-EPBL/IBM-Project-7288-1658852107