



**NAALAIYA THIRAN PROJECT - 2022
19ECI01-PROFESSIONAL READINESS FOR
INNOVATION, EMPLOYABILITY AND
ENTREPRENEURSHIP**



Detecting Parkinson's Disease using Machine Learning

A PROJECT REPORT

Submitted by

AAKASHVARMA A 1905001

HANSIKA V 1905017

NARASIMAN C V 1905032

VISHNU RAM R 1905061

**DEPARTMENT OF ELECTRONICS AND
COMMUNICATION ENGINEERING**

**COIMBATORE INSTITUTE OF TECHNOLOGY, COIMBATORE – 641
014**

(Government Aided Autonomous Institution affiliated to Anna University)

ANNA UNIVERSITY: CHENNAI 600025

NOVEMBER 2022

COIMBATORE INSTITUTE OF TECHNOLOGY

(Government aided Autonomous Institution Affiliated to Anna University)

COIMBATORE – 641014

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this report **“DETECTING PARKINSON’S DISEASE USING MACHINE LEARNING”** is the Bonafide work of **AAKASHVARMA A (1905001), HANSIKA V (1905017), NARASIMAN C V (1905032), VISHNU RAM R (1905061)** who carried out **19ECI01 Professional Readiness for Innovation, Employability and Entrepreneurship** project offered by IBM and Anna University ,Chennai.

SIGNATURE

DR.A.N.SENTHILVEL

FACULTY MENTOR

Professor

Department of Computer

Science

Engineering,

SIGNATURE

Dr. KOUSALYA G, M.E., Ph.D.,

PROFESSOR AND HEAD

Department of Computer Science

Engineering

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

Most studies in the current decade of accelerated advances in medical sciences fail to focus on ageing diseases. These are diseases that show symptoms at a much later stage, making a full recovery almost impossible. Parkinson's disease (PD) is the second most common neurodegenerative brain disorder. One could argue that it is nearly incurable and causes significant pain in the patients. All of this points to a growing need for accurate, dependable, and expandable Parkinson's disease diagnosis. The goal of this work is to compare various machine learning models in the successful prediction of the severity of Parkinson's disease and develop an effective and accurate model to help diagnose the disease accurately at an earlier stage, which could help doctors assist in the cure and recovery of PD patients. For the aforementioned purpose, we intend to use the Parkinson's spiral dataset obtained from the UCIML repository.

1.2 PURPOSE

The goal of this project is to compare various machine learning models in the successful prediction of the severity of Parkinson's disease and develop an effective and accurate model to help diagnose the disease accurately at an earlier stage, which could help doctors aid in the cure and recovery. This project had a 90% efficiency rate. A large amount of data is collected in our model from both normal people and people who have previously been affected by Parkinson's disease.

CHAPTER - 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM

S.No.	Author Name	Title	Methods	Description
1	Timothy J, Wroge	Parkinson's Disease Diagnosis Using Machine Learning and Voice.	Decision support algorithm	It is difficult to detect early due to the subtle initial symptoms.
2	Johannes Frasnelli	Machine Learning for the Diagnosis of Parkinson's Disease.	Machine learning algorithm	Difficulties and to refine the diagnosis and assessment procedures of machine learning methods have been implemented for the classification.
3	Chirag Mittal	Parkinson's Disease Detection Using Different Machine Learning Algorithms.	K Nearest Neighbors algorithm	Late detection leads to no treatment and loss of life by using this algorithm.

Table 2.1 - Literature Survey

2.1 REFERENCES

1. Chirag Mittal, Parkinson's Disease Detection Using Different Machine Learning Algorithms, International Journal of Scientific and Research Publications, Volume 12, Issue 2, February 2022 23 ISSN 2250-3153.
2. Johannes Frasnelli, Machine Learning for the Diagnosis of Parkinson's Disease, Front. Aging Neurosci, 06 May 2021Sec. Parkinson's Disease and Aging related Movement Disorders.
3. Timothy J, Wroge, Parkinson's Disease Diagnosis Using Machine Learning and Voice. 2018 IEEE Signal Processing in medicine and Biology symposium (SPMB).

2.2 PROBLEM STATEMENT DEFINITION

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Patient	consult a doctor	I can't consult a doctor	There is more crowd.	Restless
PS-2	Person	Check whether I am a PD patient or not.	I don't know how to recognize	I don't know the method of recognition	I am not with enough knowledge.

Table 2.2 - Problem Statement Definition

CHAPTER 3

IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

EMPATHY MAP

Identifying stakeholder behaviour



Project name:

Stakeholder:

Day: Month: Year:

Designed for:

Designed by:

Version:

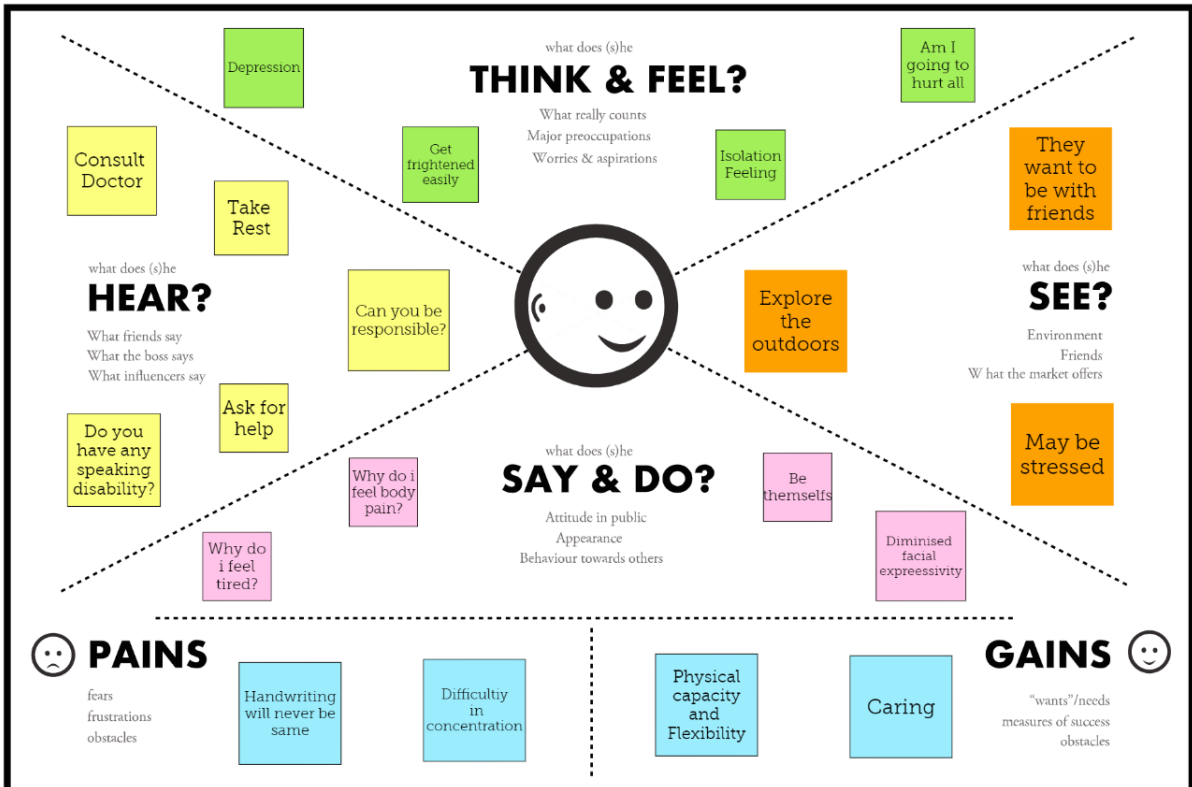


Figure 3.1 - Empathy Map Canvas

3.2 IDEATION AND BRAINSTORMING

3 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 can break it up into smaller sub-groups.

⌚ 20 minutes

TIP

Add color-coding to sticky notes to speed up the process. Assign colors to different categories of ideas.

4 Prioritize

Your team should all be on the same page about what's important, moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

⌚ 20 minutes

1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

⌚ 5 minutes

PROBLEM

How might we detect parkinson's disease in humans effectively?

Key rules of brainstorming

To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

2 Brainstorm

Write down any ideas that come to mind that address your problem statement.

⌚ 10 minutes

TIP

You can select a sticky note and hit the pencil icon to start drawing!

AAKASH

- Web based
- Ease of use
- Error free
- Fast
- Machine learning

Hansika

- CNN based
- Image spiral classification
- Website to upload images then classify
- Output in screen fast

Narasiman

- Image clarity and requirements of software and hardware
- Camera phones to upload easily
- Use Anywhere
- Accessibility

Vishnu ram

- Remote usage
- Feasibility and accessibility
- Abstraction
- UI design
- All age group access

Figure 3.2 - Ideation and Brainstorming

3.3 PROPOSED SOLUTION

S.No.	PARAMETER	DESCRIPTION
1	Problem Statement (Problem to be solved)	Parkinson's disease disorder is a brain disorder that causes unintended or uncontrollable movements, such as shaking, stiffness, and difficulty with balance.
2	Idea / Solution description	Studies investigates signals from sustained phonation and text dependent speech modalities for Parkinson's disease screening. Phonation corresponds to the vowel voicing task and speech.
3	Novelty / Uniqueness	Testing 25 non impulsive patients with Parkinson's disease(PD) and 27 PD Patients.
4	Social Impact / Customer Satisfaction	Since it is based on the voice based detection it is very convenient to use. As it helps the people to detect the Parkinson's disease in early stage.
5	Business Model (Revenue Model)	A free platform on the voice based detection it is very convenient to use. As it helps the people to detect the Parkinson's disease in early stage the loss of life is prevented.
6	Scalability of the Solution	Additional features can be added anytime anywhere. Any number of users can access it all at once.

Table 3.1 - Proposed Solution

3.4 PROBLEM SOLUTION FIT

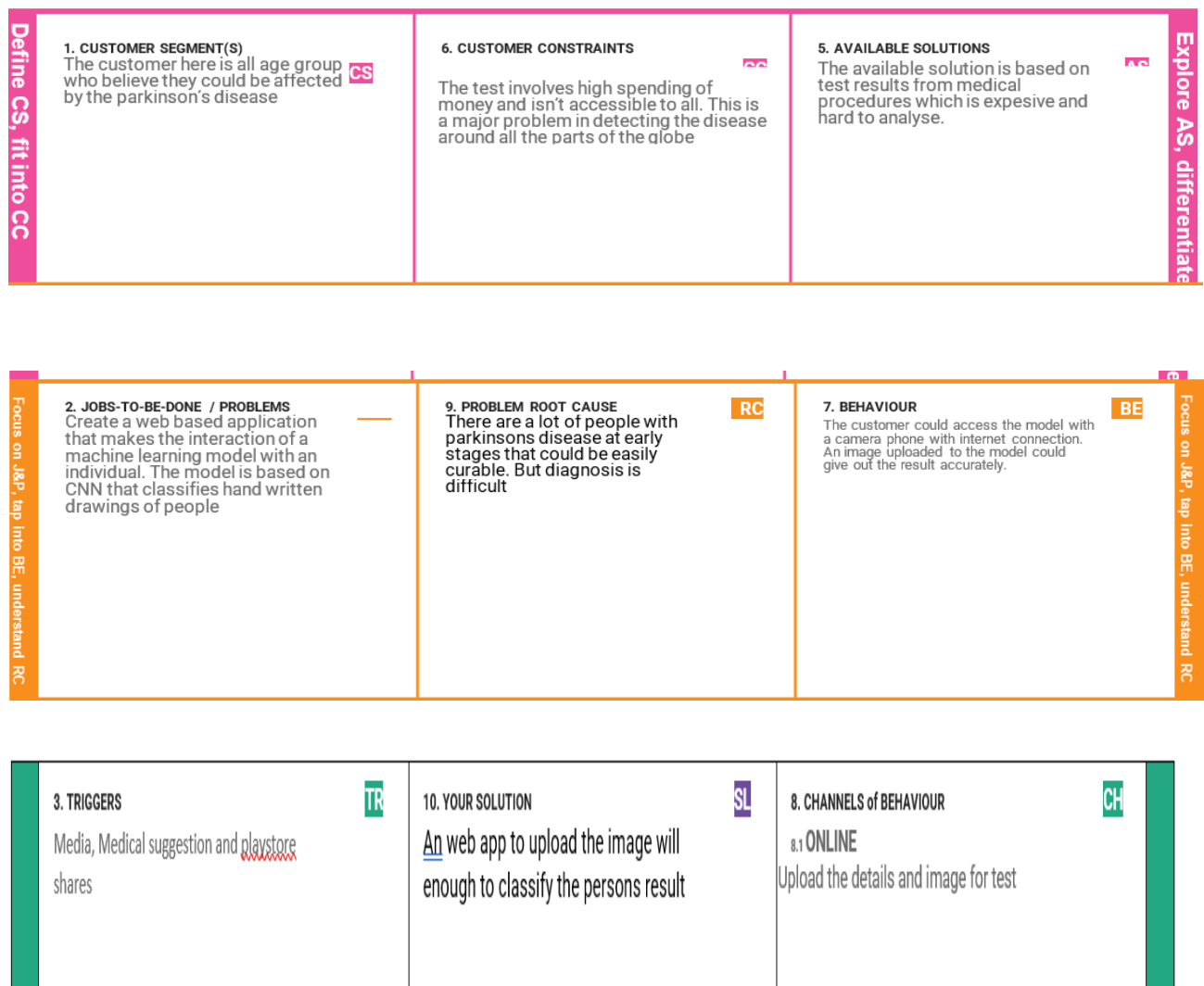


Figure 3.3 - Problem Solution Fit

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

FR No.	FUNCTIONAL REQUIREMENT	SUB REQUIREMENT
FR-1	User Registration	Registration through Gmail
FR-2	User Confirmation	Confirmation via Email
FR-3	Uploading Dataset	Spiral and wave images are to be uploaded.
FR-4	Requesting Solution	Uploaded images are compared with the pre-defined Model and solution is generated.
FR-5	Downloading Solution	The Output can be downloaded in the pdf format.

Table 4.1 - Functional Requirements

4.2 NON-FUNCTIONAL REQUIREMENTS

NFR.No.	NON-FUNCTIONAL REQUIREMENT	DESCRIPTION
NFR-1	Usability	The user interface screen will be very much user friendly.
NFR-2	Security	The data given by the user will be very secure.
NFR-3	Reliability	Users can access the website all time without any failure

NFR-4	Performance	Load time for the user interface screen
NFR-5	Availability	Maximum downtime will be about 4 hours
NFR-6	Scalability	System can handle

Table 4.2 - Non Functional Requirements

CHAPTER 5

PROJECT DESIGN

5.1 DATAFLOW DIAGRAM

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

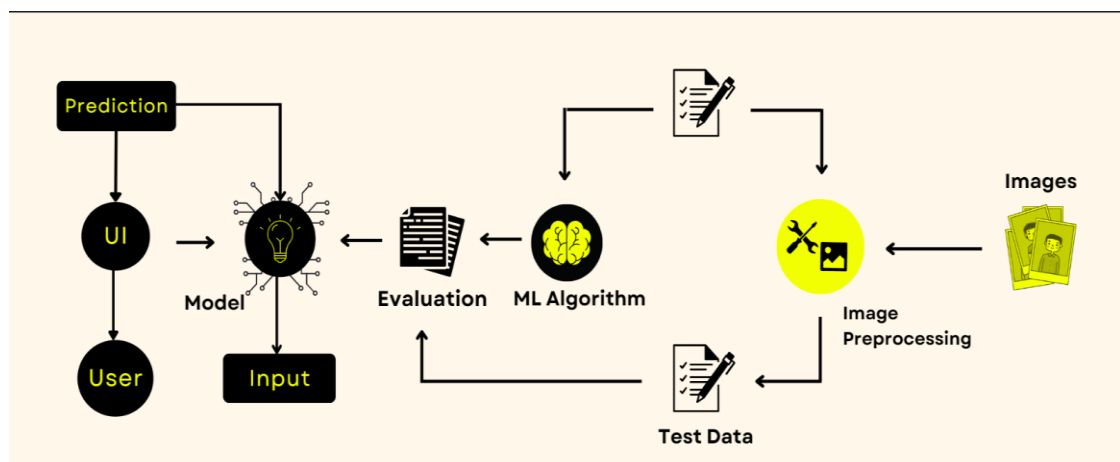


Figure 5.1 - Dataflow Diagram

5.2 SOLUTION AND TECHNICAL ARCHITECTURE

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions.

Its goals are to:

- Create and login to the IBM Credentials.
- Link the GitHub account with the IBM.
- Notebook downloads from the dataset and imports data to analyses the patients.
- After analyzing the affected patients we have to capture the images of them.
- By using Machine Learning Algorithm, we have train and test the data for the further evaluation process.
- After getting out the evaluation process we have to predict the given model by using Machine Learning.

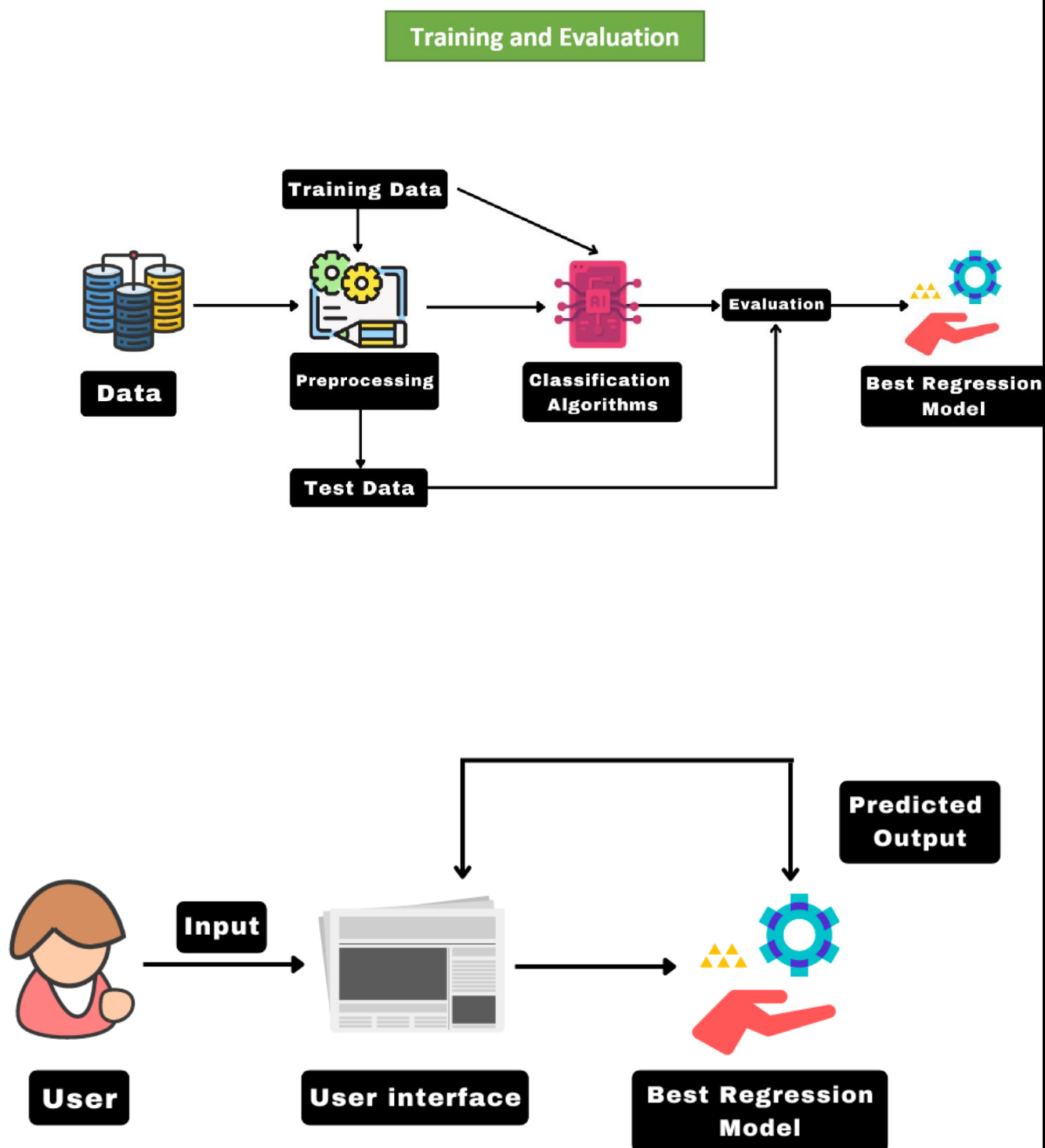


Figure 5.2 - Solution and Technical Architecture

S.No	Component	Description	Technology
1.	User Interface	Website designed for users to use the disease prediction system	HTML, CSS, JavaScript
2.	User registration	Users can register and receive confirmation for the process.	Python, HTML, CSS, Javascript
3.	Disease prediction	User enters the input to predict the disease	Machine learning
4.	Updating the results	Result of the disease prediction is displayed to user	Python, HTML, CSS, Javascript
5.	Database	Relational database to store user details	MySQL
6.	Cloud Database	Database Service on Cloud	IBM DB2
7.	File Storage	File storage requirements	Local Filesystem
8.	External API-1	To allow the system to use google API features like google account login, translate	Gmail API, Google Translate.
9.	Machine LearningModel	To predict whether the user input has Parkinson disease	Random Forest, Decision Tree, SVM
10.	Infrastructure (Server / Cloud)	Application Deployment on Cloud	IBM Cloud

Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Open source frameworks used to build web application and machine learning models.	Tensorflow, Flask, Sklearn, Keras, OpenCV etc.
2.	Scalable Architecture	3 tier architecture is used which contains user interface, application tier, data tier.	IBM Watson Studio
3.	Availability	Web application is highly available and it is deployed in cloud.	IBM Cloud
4.	Performance	The website performance is improved with caching mechanisms and model with best performance is selected for the system.	IBM Cloud InternetServices.

5.3 USER STORIES

6

User Type	Functional Requirement	User Story Number	User Story/Task	Acceptance Criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application before entering my email, password, and confirming my password	I can access my account/ dashboard	High	Sprint-1
	Login	USN-2	As a user, I can log into the application by entering email & password	I can login using my E-mail ID accounts or user credentials	High	Sprint-1
	Dashboard	USN-3	As a user I can view the page of the application where I can upload my images of spiral and wave	I can access my account / dashboard	High	Sprint-2
	Login	USN-5	As a user, I can login to my website dashboard with the login credentials	I can login using my user credentials	High	Sprint-3
Administrator	Login	USN-7	As a admin,I can login to the website using my login credentials	I can login to the website using my login credentials	High	Sprint-1
	Dashboard	USN-8	As a admin, I can view the dashboard if the application	I can access my Dashboard	High	Sprint-2

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	10	High	Aakash Varma A Hansika V Narasiman CV Vishnu Ram R
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	8	High	Aakash Varma A Hansika V Narasiman CV Vishnu Ram R
Sprint-1		USN-3	As a user, I can register for the application through Facebook	2	Low	Aakash Varma A Hansika V Narasiman CV Vishnu Ram R
Sprint-1		USN-4	As a user, I can register for the application through Gmail	6	Medium	Aakash Varma A Hansika V Narasiman CV Vishnu Ram R
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	10	High	Aakash Varma A Hansika V Narasiman CV Vishnu Ram R
Sprint-2	Data Collection	USN-6	Once Logged in, Image data is Collected and preform pre-processing.	4	Medium	Aakash Varma A Hansika V Narasiman CV Vishnu Ram R
Sprint-3	Implementation	USN-7	As an admin,split the Dataset for training and testing in 80:20 ratio	6	Medium	Aakash Varma A Hansika V Narasiman CV Vishnu Ram R
Sprint-3	Implementation	USN-8	Application uses to find the hyperplane.	10	High	Aakash Varma A Hansika V Narasiman CV Vishnu Ram R
Sprint-4	Deployment	USN-9	Predict the results.	10	High	Aakash Varma A Hansika V Narasiman CV Vishnu Ram R
Sprint-4	Cloud Deployment	USN-10	Deploy the model on IBM cloud	6	Medium	Aakash Varma A Hansika V Narasiman CV Vishnu Ram R

Table 6.1 Sprint planning and estimation

6.2 SPRINT DELIVERY SCHEDULE

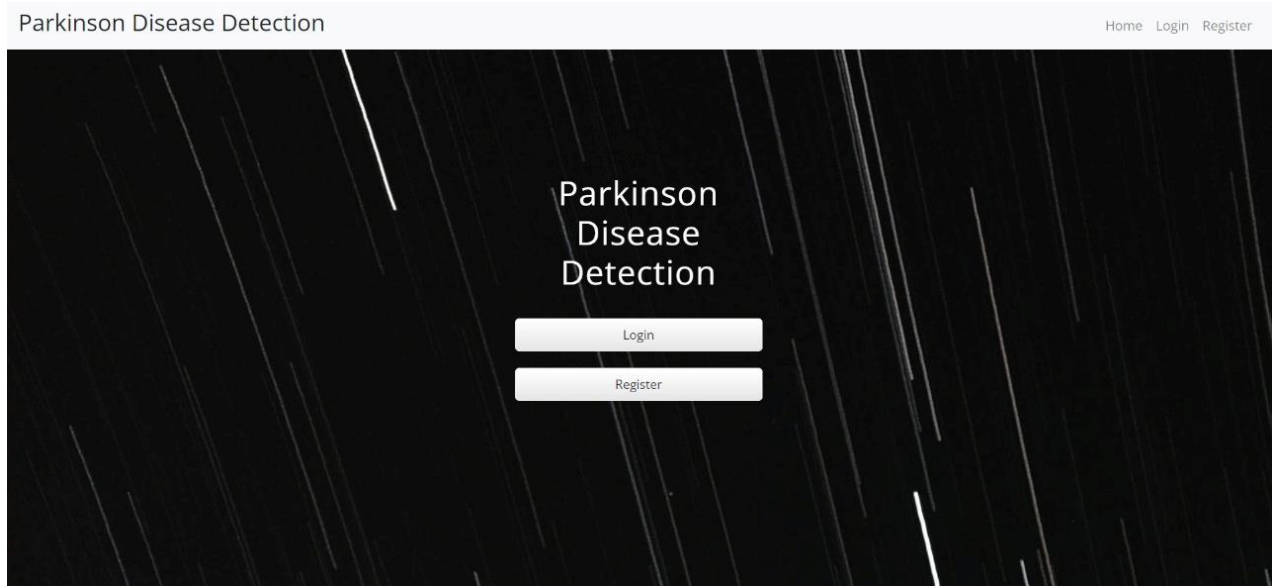
Sprint	Total Story Point s	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	10	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	10	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	10	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	10	19 Nov 2022

Table 6.2 Sprint delivery schedule

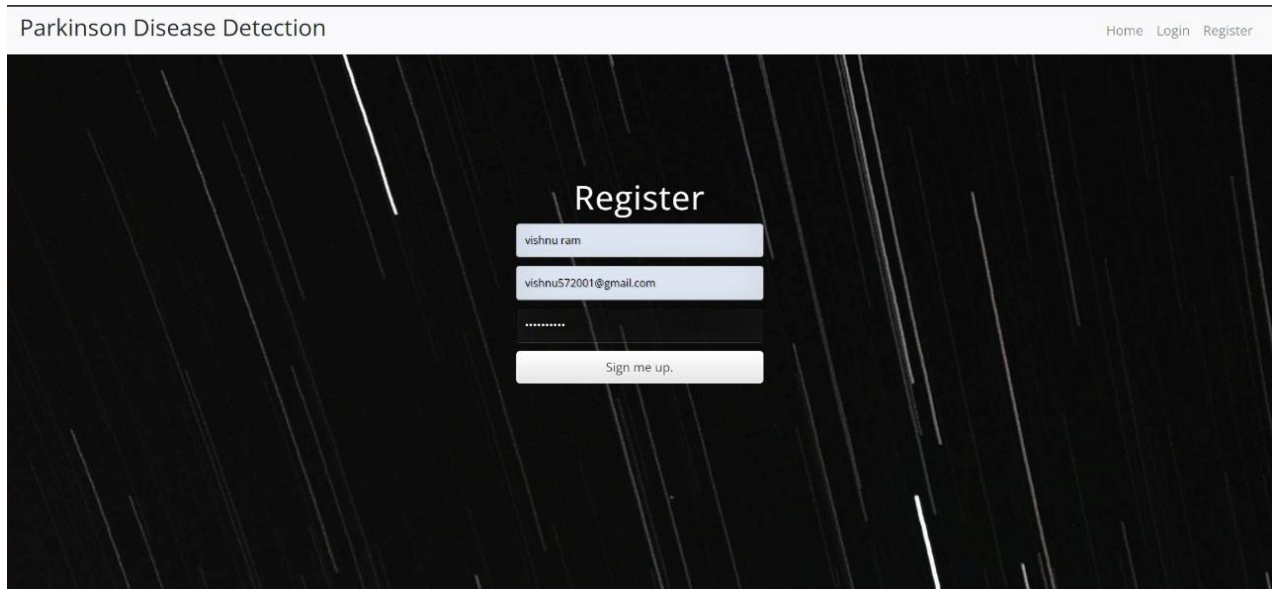
CHAPTER 7

CODING AND SOLUTION

Home page Design



Register Page Design



Login Page Design

Parkinson Disease Detection

Home Login Register

Login

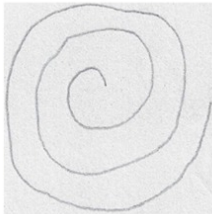
Let me in.

Output of Model Prediction: Not Parkinson Disease Detected

Parkinson Disease Detection

Parkinson Disease Detection

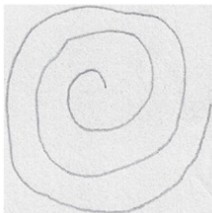
Upload image



Parkinson Disease Detection

Parkinson Disease Detection

Upload image

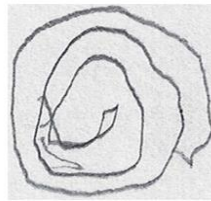
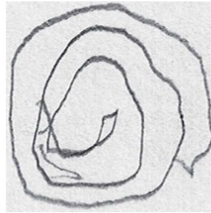


Parkinson Disease Detected

Parkinson Disease Detection

Parkinson Disease Detection

Upload image



Result: parkinson Parkinson disease detected!!

Database for Registration

New Database	Open Database	Write Changes	Revert Changes	Open Project	Save Project
Database Structure	Browse Data	Edit Pragmas	Execute SQL		
Table: users					Filter in any column
password	name	id	email		
Filter	Filter	Filter	Filter		
1 pbkdf2:sha256:260000\$4qflAUMh\$77...	vishnu ram	5	vishnu572001@gmail.com		
2 pbkdf2:sha256:260000\$49Bs56qa\$94...	Narasiman	7	narasimanmadhav@gmail.com		
3 pbkdf2:sha256:260000\$10IWzTIA\$de...	Hansika	8	hansika1010@gmail.com		
4 pbkdf2:sha256:260000\$bOyJRALV\$8f...	Aakash	9	aakash_raze@gmail.com		

CHAPTER 8

PERFORMANCE METRICS

Performance metrics are defined as figures and data representative of an organization's actions, abilities, and overall quality. There are many different forms of performance metrics, including sales, profit, return on investment, customer happiness, customer reviews, personal reviews, overall quality, and reputation in a marketplace. Performance metrics can vary considerably when viewed through different industries.

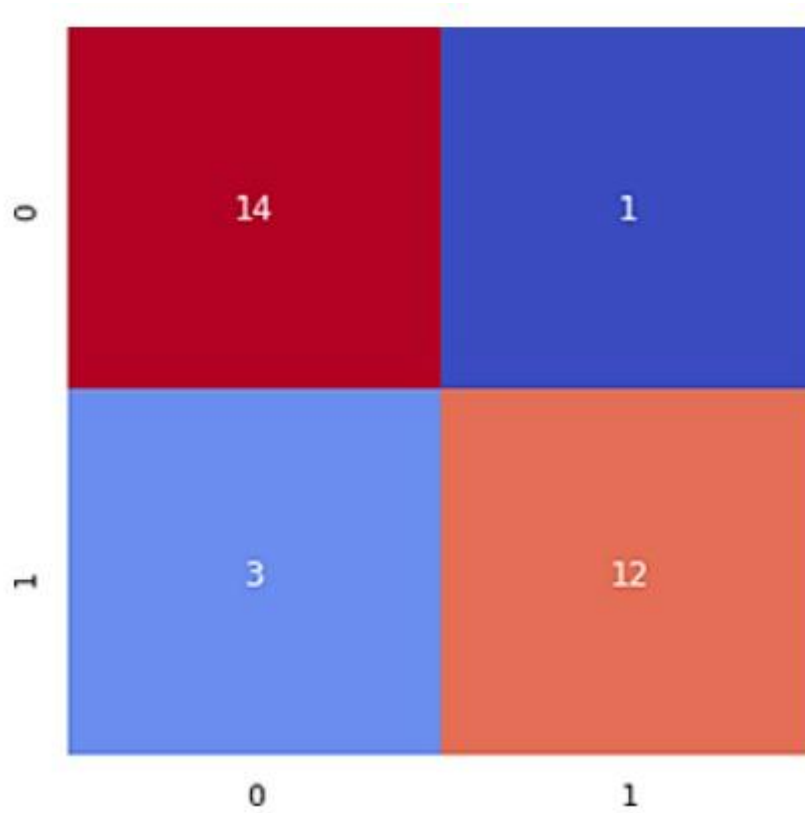


Figure 9.1 Performance metrics

CHAPTER 10

ADVANTAGES AND DISADVANTAGES

ADVANTAGES

- a. Reduces manual work
- b. More accurate than average human
- c. Capable of handling a lot of data
- d. Can be used anywhere from any device.

DISADVANTAGES

- a. Cannot handle complex data
- b. All the data must be in image format
- c. Requires a high performance server for faster predictions
- d. Prone to occasional errors

CHAPTER 11

CONCLUSION

Parkinson's Disease is a totally grave disease and has no cure till date. since it impacts the actions of the parts of the body, the speech additionally stands affected. here, the gadget tries to offer a way of detecting Parkinson's ailment so one can bring about a quick action to reduce or even put off it from affecting the whole body. This gadget aims to make this method of expertise a case of Parkinson's on the earliest via each, the affected person as well as scientific experts. hence, the goal is to apply numerous machine getting to know strategies like Random Forest Classifier , CNN,for buying the maximum accurate result. Here using Decision Tree and building a classifier results in an accuracy of 86%.

CHAPTER 12

FUTURE SCOPE

This project is far from complete and there is a lot of room for improvement. Some of the improvements that can be made to this project are as follows:

- i. Add support to detect from multiple images and save the results
- ii. Add support to detect multiple images
- iii. Improve model to detect from complex images

This project has endless potential and can always be enhanced to become better. Implementing this concept in the real world will benefit several industries and reduce the workload on many workers, enhancing overall work efficiency.

13. APPENDIX

13.1 Model Development

```
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

def quantify_image(img):

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


def load_split(path):
    imgp=list(paths.list_images(path))
    data=[]
    labels=[]
    for ip in imgp:
        label=ip.split(os.path.sep)[-2]
        img=cv2.imread(ip)
        img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
        img=cv2.resize(img,(200,200))
        img=cv2.threshold(img,0,255,cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
        features=quantify_image(img)
        data.append(features)
        labels.append(label)
    return (np.array(data),np.array(labels))

trp=r"spiral\spiral\training"
tep=r"spiral\spiral\testing"
(xtrain,ytrain)=load_split(trp)
(xtest,ytest)=load_split(tep)
[10:48 pm, 19/11/2022] Vishnu Ram: le=LabelEncoder()
ytrain=le.fit_transform(ytrain)
ytest=le.transform(ytest)
print(xtrain.shape,ytrain.shape)
model=RandomForestClassifier(n_estimators=100)
model.fit(xtrain,ytrain)
```

```

testp=list(paths.list_images(tep))
idxs=np.arange(0,len(testp))
idxs=np.random.choice(idxs,size=(25,),replace=False)
images=[]
for i in idxs:
    image=cv2.imread(testp[i])
    op=image.copy()
    op=cv2.resize(op,(128,128))
    img=image
    img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    img=cv2.resize(img,(200,200))
    img=cv2.threshold(img,0,255,cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
    features=quantify_image(img)
    preds=model.predict([features])
    label=le.inverse_transform(preds)[0]
    color=(0,255,0) if label=="healthy" else (0,0,255)
    cv2.putText(op,label,(3,20),cv2.FONT_HERSHEY_SIMPLEX,0.5,color,2)
    images.append(op)
montage=build_montages(images,(128,128),(5,5))[0]
cv2.imshow("ahahaa",montage)
cv2.waitKey(0)

predictions=model.predict(xtest)
cm=confusion_matrix(ytest,predictions).flatten()
print(cm)
(tn,fp,fn,tp)=cm
acc=(tp+tn)/float(cm.sum())
print(acc)

pickle.dump(model,open('modelRF.pkl','wb'))

```

13.2 Testing the Model

```

image=cv2.imread('test1.jpg')
op=image.copy()
op=cv2.resize(op,(128,128))
img=image
img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
img=cv2.resize(img,(200,200))
img=cv2.threshold(img,0,255,cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]
features=quantify_image(img)
preds=model.predict([features])
label=le.inverse_transform(preds)[0]
color=(0,255,0) if label=="healthy" else (0,0,255)
cv2.putText(op,label,(3,20),cv2.FONT_HERSHEY_SIMPLEX,0.5,color,2)
print(label)
cv2.imshow("hello",op)

```

13.3 Flask Code

```
import pickle
import sklearn
from flask import Flask, render_template, request, redirect, url_for, flash
from flask_bootstrap import Bootstrap
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy.orm import relationship
from flask_wtf import FlaskForm
from werkzeug.utils import secure_filename
from wtforms import StringField, SubmitField, FloatField, IntegerField
from wtforms.validators import DataRequired
from werkzeug.security import generate_password_hash, check_password_hash
import os
import cv2
from skimage import feature
from flask_login import
login_user, logout_user, LoginManager, UserMixin, current_user, login_required
app = Flask(__name__)
app.config['SECRET_KEY'] = '8BYkEfBA6O6donzWlSihBXox7C0sVR6b'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
Bootstrap(app)
login_manager = LoginManager()
login_manager.init_app(app)
class users(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(200), nullable=False)
    password = db.Column(db.String(300), nullable=False)
    name = db.Column(db.String(100), nullable=False)
@login_manager.user_loader
def user_load(id):
    return users.query.get(int(id))
@app.route("/")
def home():
    return render_template("index1.html")
@app.route("/register", methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        if users.query.filter_by(email=request.form['email']).first():
            flash('User already registered')
            return redirect(url_for('login'))
        else:
            password =
generate_password_hash(request.form['password'], method="pbkdf2:sha256", salt_length=👁️)
            user = users(
                email = request.form['email'],
                password = password,
                name = request.form['name']
            )
            db.session.add(user)
            db.session.commit()
            return redirect(url_for('home'))
    return render_template('register.html')
```

```

@app.route("/login",methods=['GET','POST'])
def login():
    if request.method == 'POST':
        email= request.form['email']
        password = request.form['password']
        k=users.query.filter_by(email=email).first()
        if not k:
            flash('User not registered')
            return redirect(url_for('login'))
        elif check_password_hash(k.password,password):
            login_user(k)
            return redirect(url_for('model'))
        else:
            flash('Wrong password')
            return redirect(url_for('login'))

    return render_template('login.html')
@app.route("/logout")
def logout():
    logout_user()
    return redirect(url_for('home'))
@app.route("/parkinson")
def model():
    return render_template('index.html')
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
@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)
        image = cv2.imread(filepath)
        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)
        model = pickle.loads(open("modelRF.pkl", "rb").read())
        preds= model.predict([features])
        ls = ["healthy", "parkinson"]
        result = ls[preds[0]]
        if(result=="healthy"):
            result+": You are healthy!!"
        elif(result=="parkinson"):
            result+=" Parkinson disease detected!! "
        return result

    return None
admin=[1]
if __name__ == '__main__':
    app.run(debug=True)

```

13.4 Home page HTML

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

<div class="box">
    <h1>Parkinson Disease Detection</h1><br>

    <a href="{ { url_for('login') } } " class="btn btn-default btn-block btn-large">Login</a>
    <a href="{ { url_for('register') } } " class="btn btn-secondary btn-block btn-large">Register</a>

</div>

{% endblock %}
```

13.5 Home page (Index.html)

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Parkinson Disease Detection</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <link rel="stylesheet" href="{ { url_for('static', filename='css/styles1.css') } }">
</head>

<body>
<nav class="navbar navbar-expand-lg navbar-light bg-light">

    <div class="collapse navbar-collapse">
        <h3>Parkinson Disease Detection</h3>
        <ul class="navbar-nav ml-auto">
            <li class="nav-item">
                <a class="nav-link" href="{ { url_for('home') } }">Home</a>
            </li>
            {% if not logged_in: %}
            <li class="nav-item">
                <a class="nav-link" href="{ { url_for('login') } }">Login</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{ { url_for('register') } }">Register</a>
            </li>
            {% endif %}
        </ul>
```

```

</div>
</nav>
    {% block content %}
    {% endblock %}
</body>

</html>

```

13.6 Register page HTML Code

```

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

<div class="box">
    <h1>Register</h1>

    <form action="{{ url_for('register') }}" method="post">

        <input type="text" name="name" placeholder="Name" required="required" />
        <input type="email" name="email" placeholder="Email" required="required" />
        <input type="password" name="password" placeholder="Password" required="required" />
        <button type="submit" class="btn btn-default btn-block btn-large">Sign me up.</button>
    </form>
</div>

{% endblock %}

```

13.7 Login Page HTML Code

```

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

<div class="box">
    <h1>Login</h1>
    {% with messages = get_flashed_messages() %}
        {% if messages %}
            {% for message in messages %}
                <p>{{ message }}</p>
            {% endfor %}
        {% endif %}
    {% endwith %}
    <form action="{{ url_for('login') }}" method="post">
        <input type="text" name="email" placeholder="Email" required="required"/>
        <input type="password" name="password" placeholder="Password" required="required"/>
        <button type="submit" class="btn btn-default btn-block btn-large">Let me in.</button>
    </form>
</div>

{% endblock %}

```

13.8 CSS Codes:

Main.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: #541690;  
  color: #fff;  
  font-size: 1em;  
  transition: all .4s;  
  cursor: pointer;  
}
```

```
.upload-label:hover{  
  background: #34495E;  
  color: #541690;  
}
```

```
.loader {  
  border: 8px solid #34495E; /* Light grey */  
  border-top: 8px solid #34495E; /* 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

```
*, *:before, *:after {
    box-sizing: border-box;
}
html {
    font-size: 18px;
    line-height: 1.5;
    font-weight: 300;
    color: black;
    font-family: Helvetica, Arial, sans-serif;
}
body {
    margin: 0;
    padding: 0;
    height: 100vh;
    background-color: cyan;
    background-attachment: fixed;
}
.large {
    font-size: 3rem;
}

.content {
    display: flex;
    margin: 0 auto;
    justify-content: center;
    align-items: center;
    flex-wrap: wrap;
    max-width: 1500px;
}
p.overview {
    font-size: 12px;
    height: 200px;
    width: 100%;
    overflow: hidden;
    text-overflow: ellipsis;
}
.heading {
    width: 100%;
    margin-left: 1rem;
    font-weight: 900;
    font-size: 1.618rem;
    text-transform: uppercase;
    letter-spacing: 0.1ch;
    line-height: 1;
    padding-bottom: 0.5em;
    margin-bottom: 1rem;
    position: relative;
}
.heading:after {
    display: block;
    content: ";
    position: absolute;
    width: 60px;
```

```

        height: 8px;
        background: linear-gradient(135deg, red, green);
        bottom: 0;
    }
    .description {
        width: 100%;
        margin-top: 0;
        margin-left: 1rem;
        margin-bottom: 3rem;
    }
    .card {
        color: inherit;
        cursor: pointer;
        width: calc(33% - 3rem);
        min-width: calc(33% - 3rem);
        height: 400px;
        min-height: 400px;
        perspective: 1000px;
        margin: 1rem auto;
        position: relative;
    }

    @media screen and (max-width: 800px) {
        .card {
            width: calc(50% - 3rem);
        }
    }
    @media screen and (max-width: 500px) {
        .card {
            width: 100%;
        }
    }
    .front, .back {
        display: flex;
        border-radius: 6px;
        background-position: center;
        background-size: cover;
        text-align: center;
        justify-content: center;
        align-items: center;
        position: absolute;
        height: 100%;
        width: 100%;
        -webkit-backface-visibility: hidden;
        backface-visibility: hidden;
        transform-style: preserve-3d;
        transition: ease-in-out 600ms;
    }
    .front {
        background-size: cover;
        padding: 2rem;
        font-size: 1.618rem;
        font-weight: 600;
        color: #fff;
        overflow: hidden;
        font-family: Helvetica, Arial, sans-serif;
    }

```

```
}
.front:before {
    position: absolute;
    display: block;
    content: "";
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background: linear-gradient(135deg, #1a9be6, #1a57e6);
    opacity: 0.25;
    z-index: -1;
}
.card:hover .front {
    transform: rotateY(180deg);
}
.card:nth-child(even):hover .front {
    transform: rotateY(-180deg);
}

.back {
    background: #fff;
    transform: rotateY(-180deg);
    padding: 0 2em;
}
.card:hover .back {
    transform: rotateY(0deg);
}
.card:nth-child(even) .back {
    transform: rotateY(180deg);
}
.card:nth-child(even):hover .back {
    transform: rotateY(0deg);
}

.button {
    transform: translateZ(40px);
    cursor: pointer;
    -webkit-backface-visibility: hidden;
    backface-visibility: hidden;
    font-weight: bold;
    color: #fff;
    padding: 0.5em 1em;
    border-radius: 100px;
    font: inherit;
    background: linear-gradient(135deg, red, green);
    border: none;
    position: relative;
    transform-style: preserve-3d;
    transition: 300ms ease;
}
.button:before {
    transition: 300ms ease;
    position: absolute;
    display: block;
    content: "";
    transform: translateZ(-40px);
```

```

        -webkit-backface-visibility: hidden;
        backface-visibility: hidden;
        height: calc(100% - 20px);
        width: calc(100% - 20px);
        border-radius: 100px;
        left: 10px;
        top: 16px;
        box-shadow: 0 0 10px 10px rgba(26, 87, 230, 0.25);
        background-color: rgba(26, 87, 230, 0.25);
    }

.button.delete-button {
    background-color: rgba(230, 87, 230, 0.25);
    background: linear-gradient(135deg, #e61a46, #e61a1a);
}

.button.delete-button:before {
    background-color: rgba(230, 87, 230, 0.25);
    box-shadow: 0 0 10px 10px rgba(230, 87, 230, 0.25);
}

.button:hover {
    transform: translateZ(55px);
}

.button:hover:before {
    transform: translateZ(-55px);
}

.button:active {
    transform: translateZ(20px);
}

.button:active:before {
    transform: translateZ(-20px);
    top: 12px;
    top: 12px;
}

.container.add {
    margin-top: 40px;
    margin-bottom: 20px;
}

.rating {
    color: #E4BB23;
}

.review {
    font-style: italic;
}

.movie_gens {
    font-size: 11.5px;
}

.title {
    font-weight: bold;
}

.release_date {
    font-weight: normal;
}

```

Main CSS FILE (style1.css)

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
.btn { display: inline-block; *display: inline; *zoom: 1; padding: 4px 10px 4px; margin-bottom: 0;
font-size: 13px; line-height: 18px; color: #333333; text-align: center; text-shadow: 0 1px 1px
rgba(255, 255, 255, 0.75); vertical-align: middle; background-color: #f5f5f5; background-image: -
moz-linear-gradient(top, #ffffff, #e6e6e6); background-image: -ms-linear-gradient(top, #ffffff,
#e6e6e6); background-image: -webkit-gradient(linear, 0 0, 0 100%, from(#ffffff), to(#e6e6e6));
background-image: -webkit-linear-gradient(top, #ffffff, #e6e6e6); background-image: -o-linear-
gradient(top, #ffffff, #e6e6e6); background-image: linear-gradient(top, #ffffff, #e6e6e6);
background-repeat: repeat-x; filter:
progid:dximagetransform.microsoft.gradient(startColorstr=#ffffff, endColorstr=#e6e6e6,
GradientType=0); border-color: #e6e6e6 #e6e6e6 #e6e6e6; border-color: rgba(0, 0, 0, 0.1) rgba(0,
0, 0, 0.1) rgba(0, 0, 0, 0.25); border: 1px solid #e6e6e6; -webkit-border-radius: 4px; -moz-border-
radius: 4px; border-radius: 4px; -webkit-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0
1px 2px rgba(0, 0, 0, 0.05); -moz-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px
rgba(0, 0, 0, 0.05); box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0,
0.05); cursor: pointer; *margin-left: .3em; }

.btn:hover, .btn:active, .btn.active, .btn.disabled, .btn[disabled] { background-color: #e6e6e6; }
.btn-large { padding: 9px 14px; font-size: 15px; line-height: normal; -webkit-border-radius: 5px; -
moz-border-radius: 5px; border-radius: 5px; }
.btn:hover { color: #333333; text-decoration: none; background-color: #e6e6e6; background-
position: 0 -15px; -webkit-transition: background-position 0.1s linear; -moz-transition:
background-position 0.1s linear; -ms-transition: background-position 0.1s linear; -o-transition:
background-position 0.1s linear; transition: background-position 0.1s linear; }
.btn-primary, .btn-primary:hover { text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25); color: #ffffff; }
.btn-primary.active { color: rgba(255, 255, 255, 0.75); }
.btn-primary { background-color: #4a77d4; background-image: -moz-linear-gradient(top,
#6eb6de, #4a77d4); background-image: -ms-linear-gradient(top, #6eb6de, #4a77d4); background-
image: -webkit-gradient(linear, 0 0, 0 100%, from(#6eb6de), to(#4a77d4)); background-image: -
webkit-linear-gradient(top, #6eb6de, #4a77d4); background-image: -o-linear-gradient(top,
#6eb6de, #4a77d4); background-image: linear-gradient(top, #6eb6de, #4a77d4); background-
repeat: repeat-x; filter: progid:dximagetransform.microsoft.gradient(startColorstr=#6eb6de,
endColorstr=#4a77d4, GradientType=0); border: 1px solid #3762bc; text-shadow: 1px 1px 1px
rgba(0,0,0,0.4); box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.5); }
.btn-primary:hover, .btn-primary:active, .btn-primary.active, .btn-primary.disabled, .btn-
primary[disabled] { filter: none; background-color: #4a77d4; }
.btn-block { width: 100%; display: block; margin-bottom: 20px; }

* { -webkit-box-sizing: border-box; -moz-box-sizing: border-box; -ms-box-sizing: border-box; -o-
box-sizing: border-box; box-sizing: border-box; }

html { width: 100%; height: 100%; overflow: hidden; }

body {
width: 100%;
height: 100%;
font-family: 'Open Sans', sans-serif;
background: black;
background-image:
```

```
url("https://static.vecteezy.com/system/resources/previews/001/231/051/large_2x/motion-blur-
light-streaks-free-photo.jpeg");
}
```

```
a {
  text-decoration: none;
  text-align: center;
}
```

```
p {
  color: #ee6f57;
  text-align: center;
}
```

```
.box {
  position: absolute;
  top: 50%;
  left: 50%;
  margin: -150px 0 0 -150px;
  width: 300px;
  height: 300px;
}
```

```
.box h1 { color: #fff; text-shadow: 0 0 10px rgba(0,0,0,0.3); letter-spacing: 1px; text-align: center;
}
```

```
.container {
  margin-top: 10rem;
  text-align: center;
}
```

```
.container h1 { color: #fff; text-shadow: 0 0 10px rgba(0,0,0,0.3); letter-spacing: 1px; text-
align: center; }
```

```
input {
  width: 100%;
  margin-bottom: 10px;
  background: rgba(0,0,0,0.3);
  border: none;
  outline: none;
  padding: 10px;
  font-size: 13px;
  color: #fff;
  text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
  border: 1px solid rgba(0,0,0,0.3);
  border-radius: 4px;
  box-shadow: inset 0 -5px 45px rgba(100,100,100,0.2), 0 1px 1px rgba(255,255,255,0.2);
  -webkit-transition: box-shadow .5s ease;
  -moz-transition: box-shadow .5s ease;
  -o-transition: box-shadow .5s ease;
  -ms-transition: box-shadow .5s ease;
  transition: box-shadow .5s ease;
}
```

```
input:focus { box-shadow: inset 0 -5px 45px rgba(100,100,100,0.4), 0 1px 1px
rgba(255,255,255,0.2); }
```

JavaScript Code

```
$(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(' Result: ' + data);  
                console.log('Success!');  
            },  
        });  
    });  
});  
})
```

GITHUB:

<https://github.com/IBM-EPBL/IBM-Project-48384-1660807114>