Name: Jyothsna Sriya M, Jainithaesh D, Rama Prasada Rao
Reg No.: 20BCR7067, 20BCE2460, 20BCE2644

SMART INTERNZ

EXTERNSHIP PROGRAM

APPLIED DATA SCIENCE

Detecting Parkinson's Disease using Machine Learning

Submitted by:

1. Jyothsna Sriya Mahabhashyam, 20BCR7067 (VIT-AP)
2. Jainithaesh Dhevaraju, 20BCE2644 (VIT Vellore)
3. Akkireddi S V S Rama Prasada Rao, 20BCE2460 (VIT Vellore)

Name: Jyothsna Sriya M, Jainithaesh D, Rama Prasada Rao
Reg No.: 20BCR7067, 20BCE2460, 20BCE2644

INDEX

Name: Jyothsna Sriya M, Jainithaesh D, Rama Prasada Rao
Reg No.: 20BCR7067, 20BCE2460, 20BCE2644

1. INTRODUCTION:

a. OVERVIEW:

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. 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 machine learning algorithms to automatically detect Parkinson's disease in hand-drawn images of spirals and waves.

b. PURPOSE:

The main aim is to predict the prediction efficiency that would be beneficial for the patients who are suffering from Parkinson and the percentage of the disease will be reduced. Generally, in the first stage, Parkinson's can be cured by the proper treatment. So, it's important to identify the PD at the early stage for the betterment of the patients. The main purpose of this research work is to find the best prediction model i.e., the best machine learning technique which will distinguish the Parkinson's patient from the healthy person. The techniques used in this problem are Decision Tree Classifier, KNN, Random Forest Classifier and Logistic Regression. The prediction is evaluated using evaluation metrics like confusion matrix, sensitivity, specificity and accuracy.

2. LITERATURE SURVEY:

a. EXISTING PROBLEM:

Some of the existing approaches or methods to solve this problem are:

- Using Microsoft Azure Machine Learning Studio tested the model and found the best suited model to be Two-Class Boosted Decision Trees, an ensemble model of boosted regression trees made in a stepwise method. Another conclusion that was derived was that Spread1, spread2, and PPE in the dataset, had the strongest weights in labeling a patient as healthy or not. The results were that we successfully created a predictive model for PD from voice analysis. The best accuracy was found using Boosted Decision Trees and the accuracy achieved was 95%

- Using supervised classification algorithms, such as deep neural networks, to learn how effective the model is in accurately diagnosing individuals with the disease. The peak accuracy was of 85% using AVEC selected feature and Gradient Boosted Decision Tree exceeding the average clinical diagnosis accuracy of non-experts (73.8%) and average accuracy of movement disorder specialists (79.6% without follow-up, 83.9% after follow-up) with pathological postmortem examination as ground truth
- Using vowels with sustained phonation and a ResNet architecture dedicated originally to image classification." The author converted the frequency-based features into images and then performed the ResNet algorithm. The testing was performed using 10-fold cross validation. The accuracy achieved was over 90%
- Using KNN and SVM and found out that SVM performs consistently than KNN and gives an accuracy of 95%. We can also infer from this paper that using motor UPDRS score as the index of disease progression instead of more specific UPDRS sub scores representing the severity of speech and other disorders is a limitation of this study
- Used the kernel support vector machine and got a classification performance of around 91%. The combination of HNR, RPDE, DFA and PPE obtained the best overall classification performance. When PPE was included, the healthy and PD classes became better separated PPE measures and contributed significantly towards a big improvement in the effectiveness of the classification.
- Used the keystroke timing information from 103 subjects, including 32 with just mild PD severity, which he captured as they typed on a computer keyboard over a 17 extended period. It showed that PD does affect various characteristics of hand and finger movement, which can be detected from keystroke features. A novel methodology was used to classify the subjects‟ disease status, by utilizing a combination of extracted features from those keystrokes which were then analyzed by an ensemble of machine learning classification models.

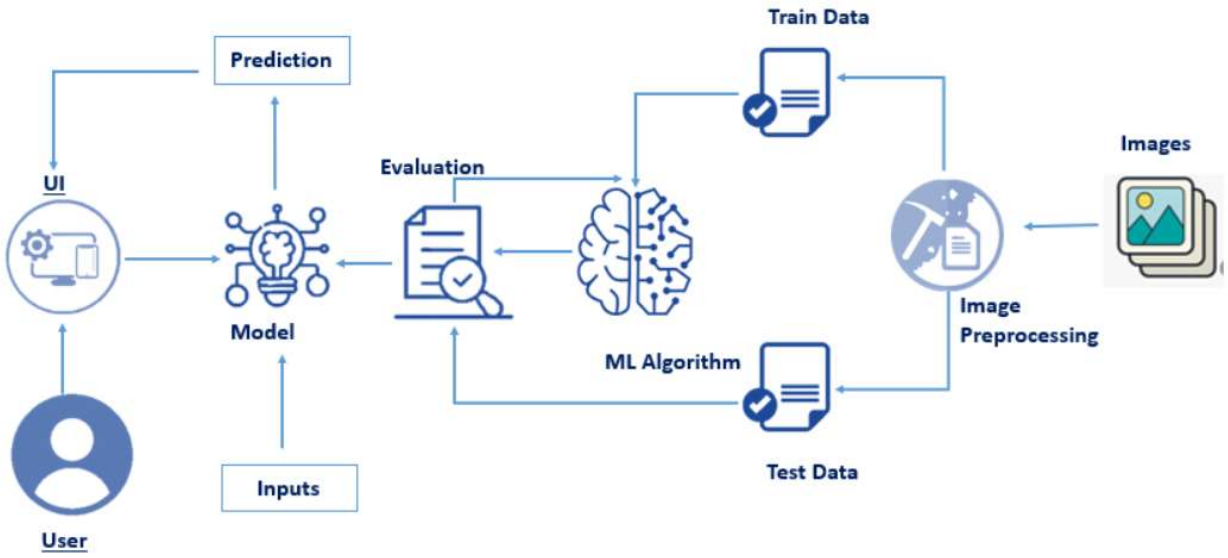
b. PROPOSED SOLUTION:

We use Deep learning and Convolution Neural Network for image data. In CNN we use data augmentation to pre-process the data. But as our data is very sensitive, improper use of data augmentation may lead to predicting a healthy person's drawing look like Parkinson's patient(drawing). Therefore, we apply computer vision to this problem. There are several Machine learning algorithms which can be used, we choose our algorithm depending on the data we are going to process such as images, sound, text, and numerical values. As the dataset which we are using is a classification so you can use the following algorithms: Random Forest Classification, Logistic Regression, KNN and Decision Tree Classification. Later we will be building a web application that is integrated to the model we built. A UI is provided for the uses where the individual has

to upload the image (spiral/wave drawing). The uploaded image is analyzed by our saved model and prediction is showcased on the UI.

3. THEORETICAL ANALYSIS:

a. BLOCK DIAGRAM:



b. HARDWARE/SOFTWARE DESIGNING:

Software Requirements:

- Operating System: Windows 10
- Framework: Flask
- Languages used: Python, Javascript, HTML, CSS
- Tools: Visual Studio Code, Jupyter Notebook, Anaconda

Hardware Requirements:

- Processor: Intel Core i5, 7th generation CPU @2.70GHz
- Hard-Disk: 250 GB or Higher
- Ram: 8 GB

4. EXPERIMENTAL INVESTIGATIONS:

Performance evaluations measures are the parameters which helps in comparative analysis of different machine learning techniques i.e., it tells the best algorithm among all other algorithms or method which can be used by medical science in the early prediction

of Parkinson's diseases. We have used several measures to evaluate the predictive results. These measures are: Confusion Matrix, Accuracy, Recall or Sensitivity, Specificity.

- Confusion Matrix:

The confusion matrix is also called as Error matrix. It is a table that is often used to describe the performance of a classification method on a set of test data for which actual value are known. Each column of the matrix represents the instances in a predicted class. the correlation matrix is represented as given:

	Positive	Negative
Positive	TP	FN
Negative	FP	TN

where,
TP: True positive,
FP: False Positive,
FN: False Negative,
TN: True Negative

- Accuracy:

Accuracy is the proportion of the total number of predictions that were correct. It can be obtained by the sum of true positive and true negative instances divided by the total number of samples. It is expressed as:

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

- Recall (or) Sensitivity:

Recall is defined as the fraction between True Positive instances and Actual yes instances or it is the ratio of correct positive results to the number of all relevant samples. It is expressed as:

$$Sensitivity = \frac{TP}{(TP + FN)}$$

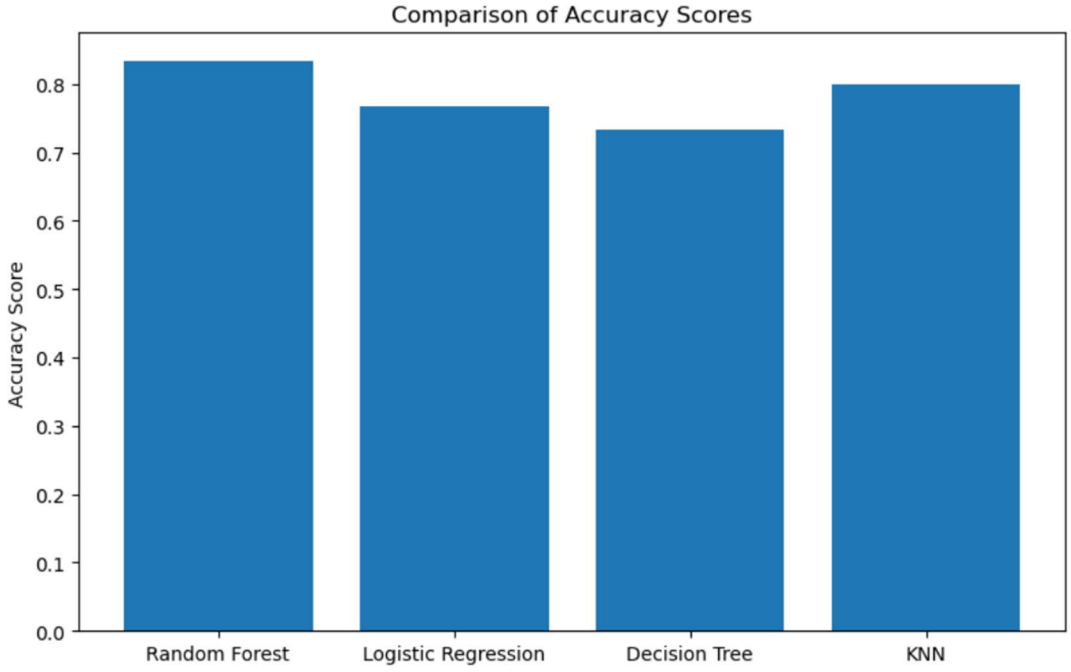
- Specificity:

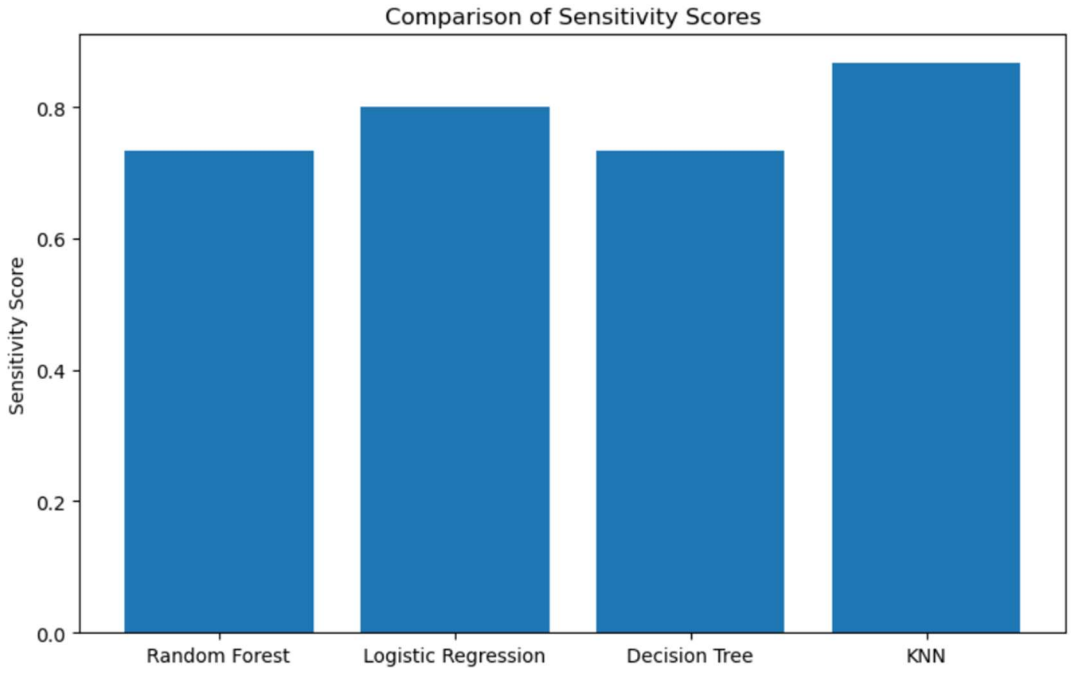
Specificity is a measure of how well a test can identify true negatives. Specificity is also referred to as selectivity or true negative rate, and it is the percentage, or proportion, of the true negatives out of all the samples that do not have the condition (true negatives and false positives). It is expressed as:

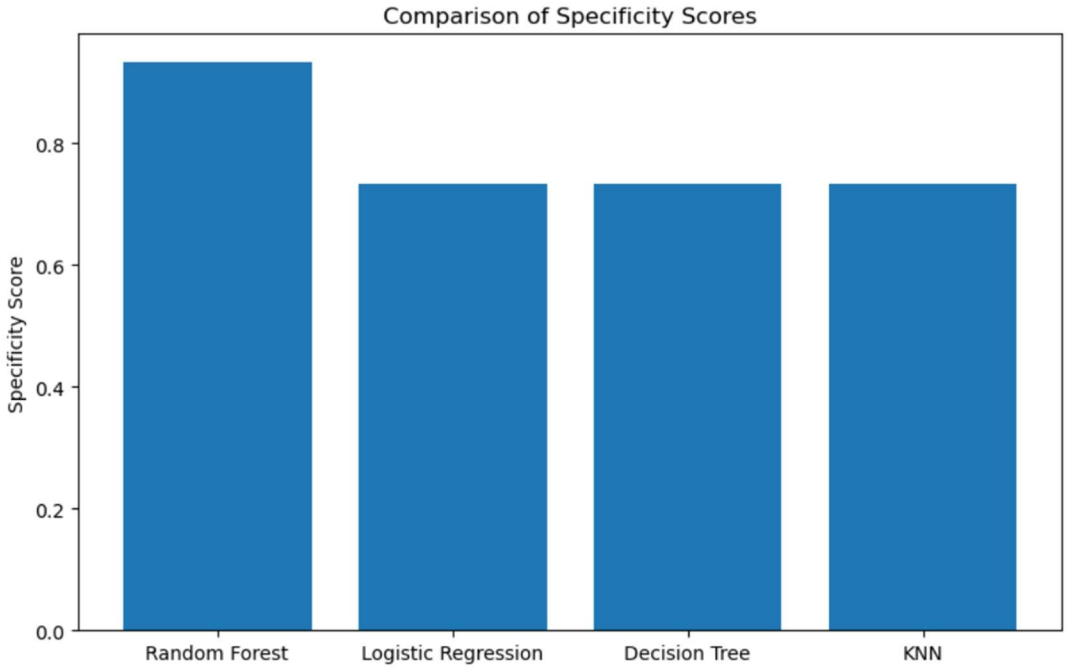
$$Specificity = \frac{TN}{TN + FP}$$

- Comparative Analysis:

The model that suites for our system has been found out through mentioned graph comparison. In this graph we shown the comparative analysis of the three models based on some of the above-mentioned evaluation metrics.



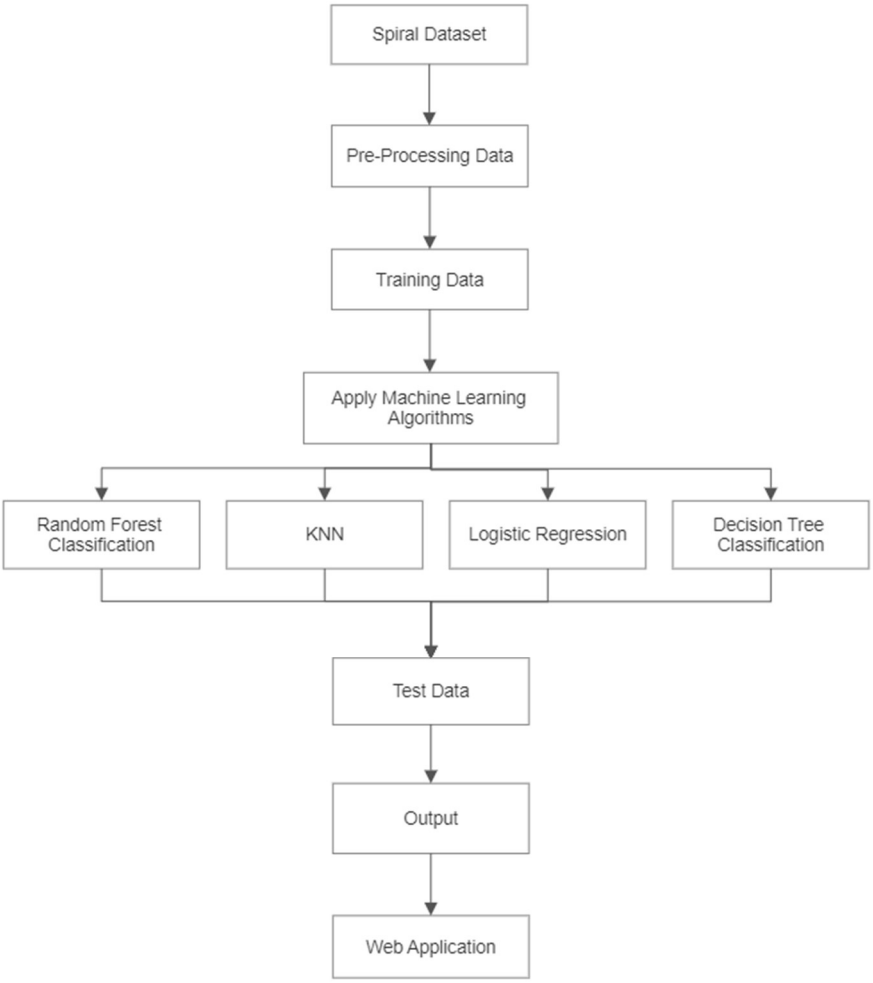




5. FLOWCHART:

The flowchart describes the high-level overview of major system components and important working relationships. It represents the flow of execution and it involves the following five major steps:

- The flowchart is defined with the flow of the process which is used to refine the raw data and used for predicting the Parkinson's data.
- The next step is preprocessing the collected raw data into an understandable format.
- Then we have to train the data.
- The Parkinson's data is evaluated with the application of a machine learning algorithm that is Random Forest, Logistic Regression, KNN, and Decision Tree algorithm, and the classification accuracy, specificity and sensitivity of this model is found.
- After training the data with these algorithms we have to test on the same algorithms.
- Finally, the result of these three algorithms is compared on the basis of classification accuracy, specificity and sensitivity.



6. RESULT:

```
[14  1  4 11]
```

Image 1: Random Forest Confusion Matrix

	precision	recall	f1-score	support
0	0.78	0.93	0.85	15
1	0.92	0.73	0.81	15
accuracy			0.83	30
macro avg	0.85	0.83	0.83	30
weighted avg	0.85	0.83	0.83	30

Image 2: Random Forest Classification Report

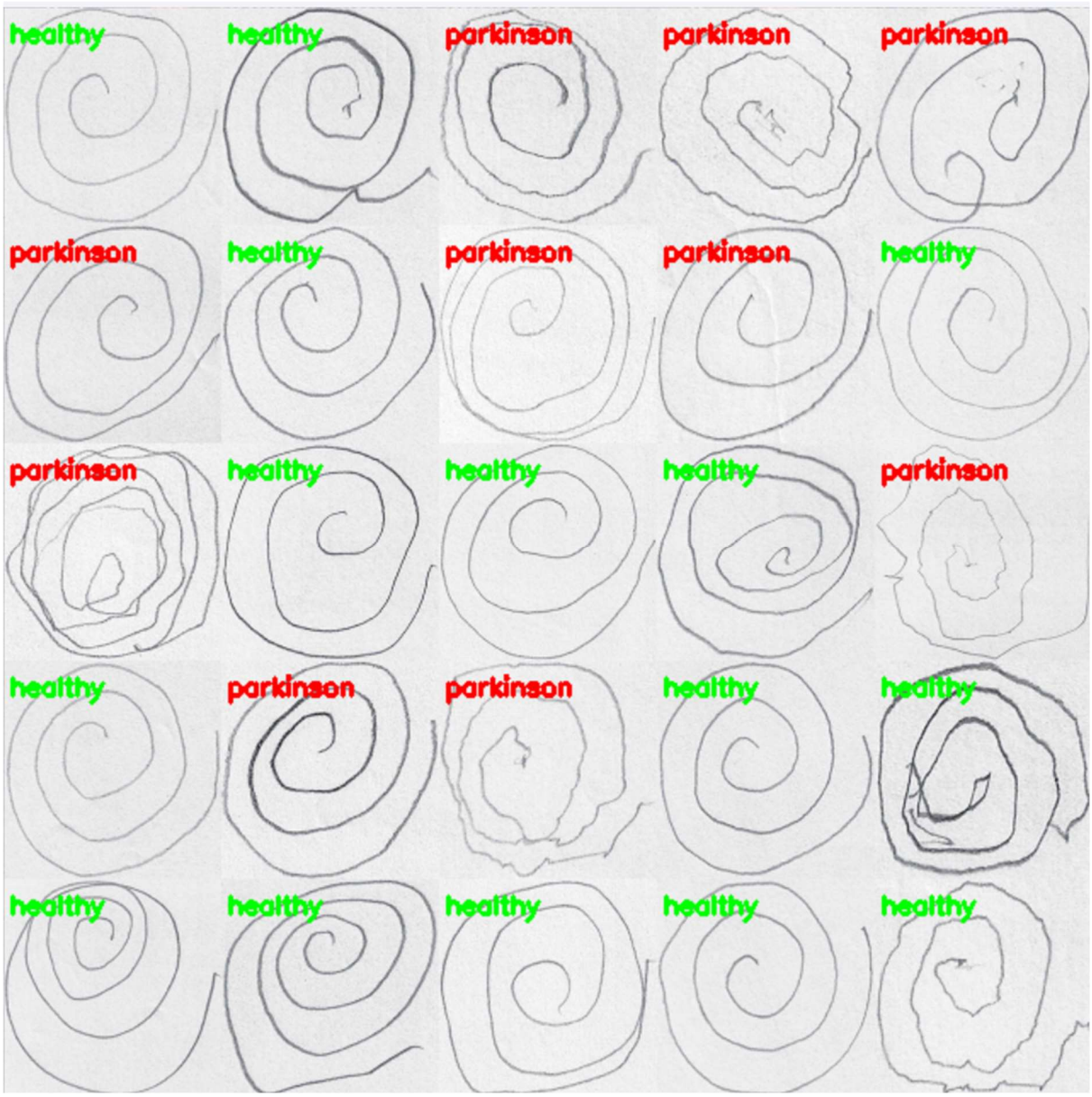


Image 3: Random Forest Classifier Output

```
[11  4  3 12]
```

Image 4: Logistic Regression Confusion Matrix

	precision	recall	f1-score	support
0	0.79	0.73	0.76	15
1	0.75	0.80	0.77	15
accuracy			0.77	30
macro avg	0.77	0.77	0.77	30
weighted avg	0.77	0.77	0.77	30

Image 5: Logistic Regression Classification Report

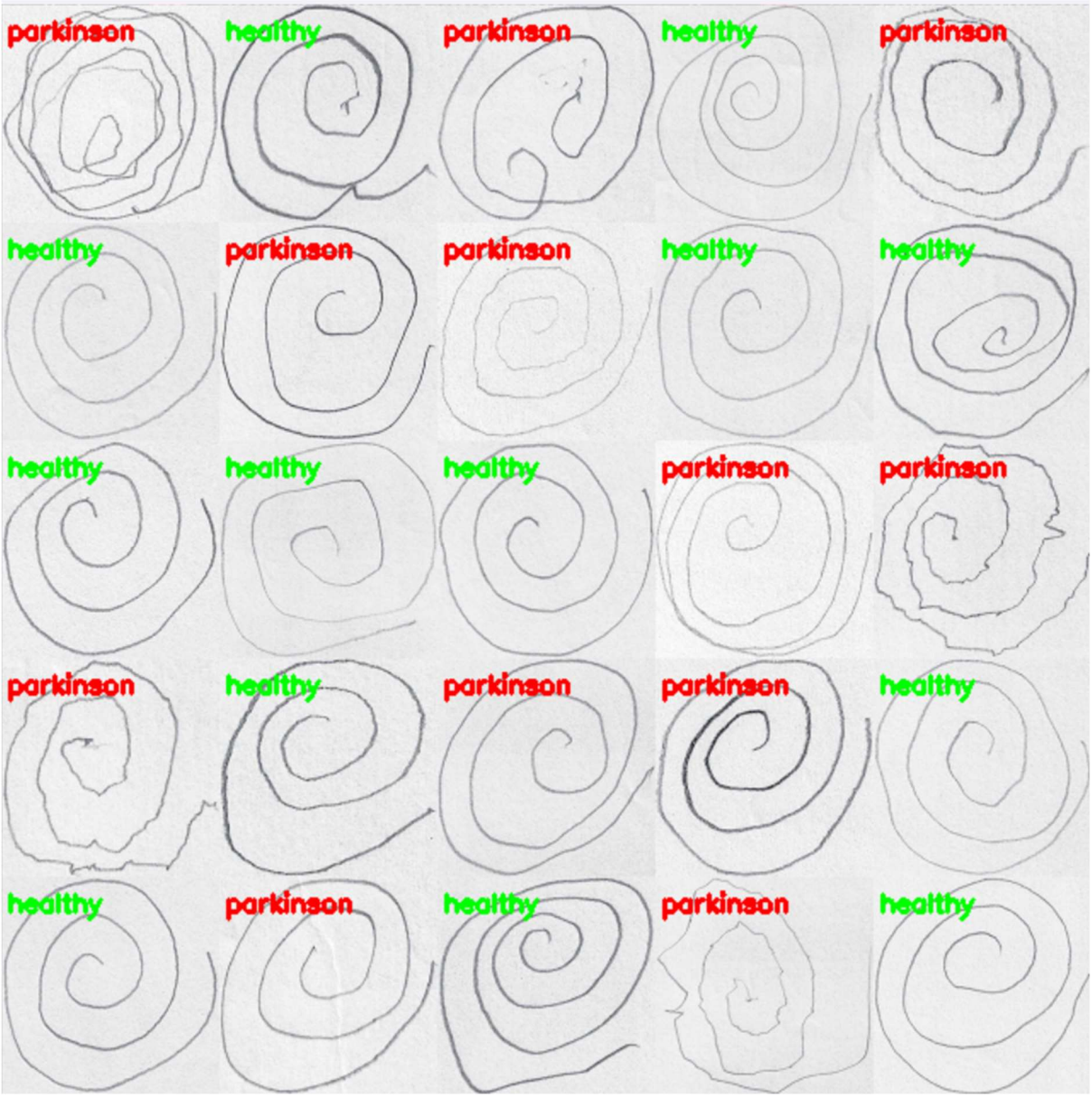


Image 6: Logistic Regression Output

```
[11  4  4 11]
```

Image 7: Decision Tree Confusion Matrix

	precision	recall	f1-score	support
0	0.73	0.73	0.73	15
1	0.73	0.73	0.73	15
accuracy			0.73	30
macro avg	0.73	0.73	0.73	30
weighted avg	0.73	0.73	0.73	30

Image 8: Decision Tree Classification Report

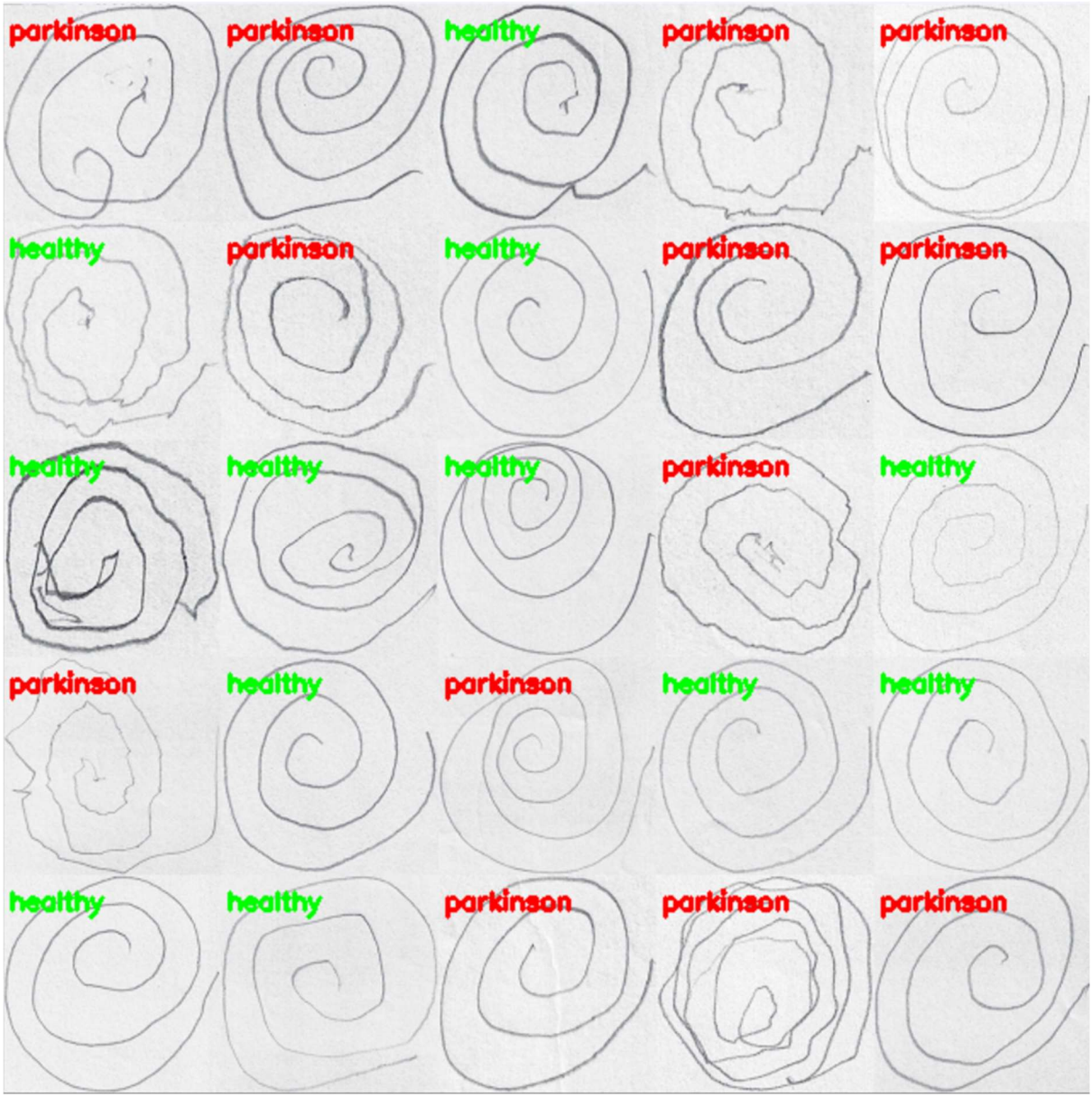


Image 9: Decision Tree Classifier Output

```
[11  4  2 13]
```

Image 10: KNN Confusion Matrix

	precision	recall	f1-score	support
0	0.85	0.73	0.79	15
1	0.76	0.87	0.81	15
accuracy			0.80	30
macro avg	0.81	0.80	0.80	30
weighted avg	0.81	0.80	0.80	30

Image 11: KNN Classification Report

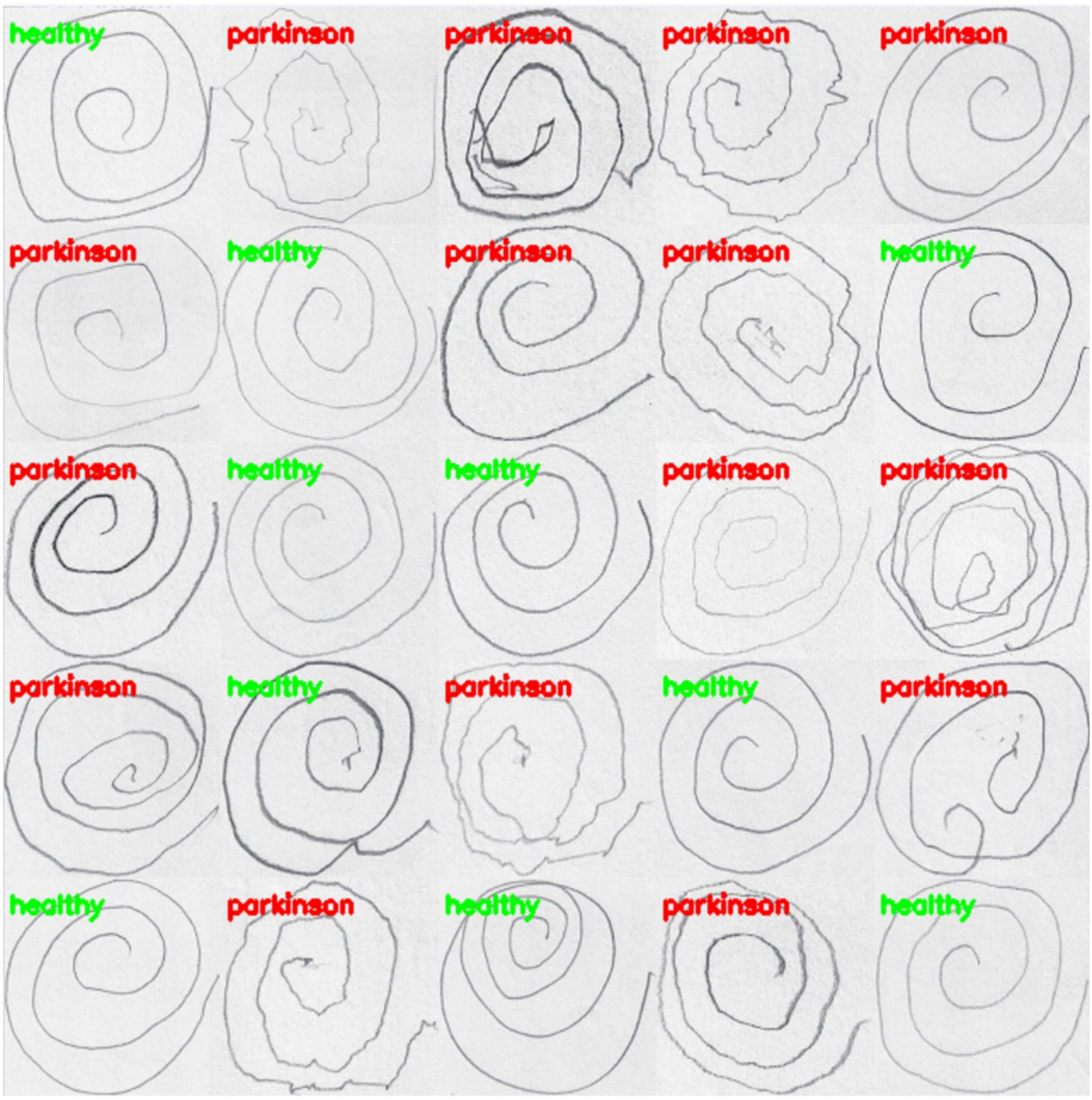


Image 12: KNN Output

```
Accuracy Scores:

Random Forest :  0.8333333333333334
Logistic Regression :  0.7666666666666667
Decision Tree :  0.7333333333333333
KNN :  0.8

Sensitivity:

Random Forest :  0.7333333333333333
Logistic Regression :  0.8
Decision Tree :  0.7333333333333333
KNN :  0.8666666666666667

Specificity:

Random Forest :  0.9333333333333333
Logistic Regression :  0.7333333333333333
Decision Tree :  0.7333333333333333
KNN :  0.7333333333333333
```

Image 13: Accuracy Outputs

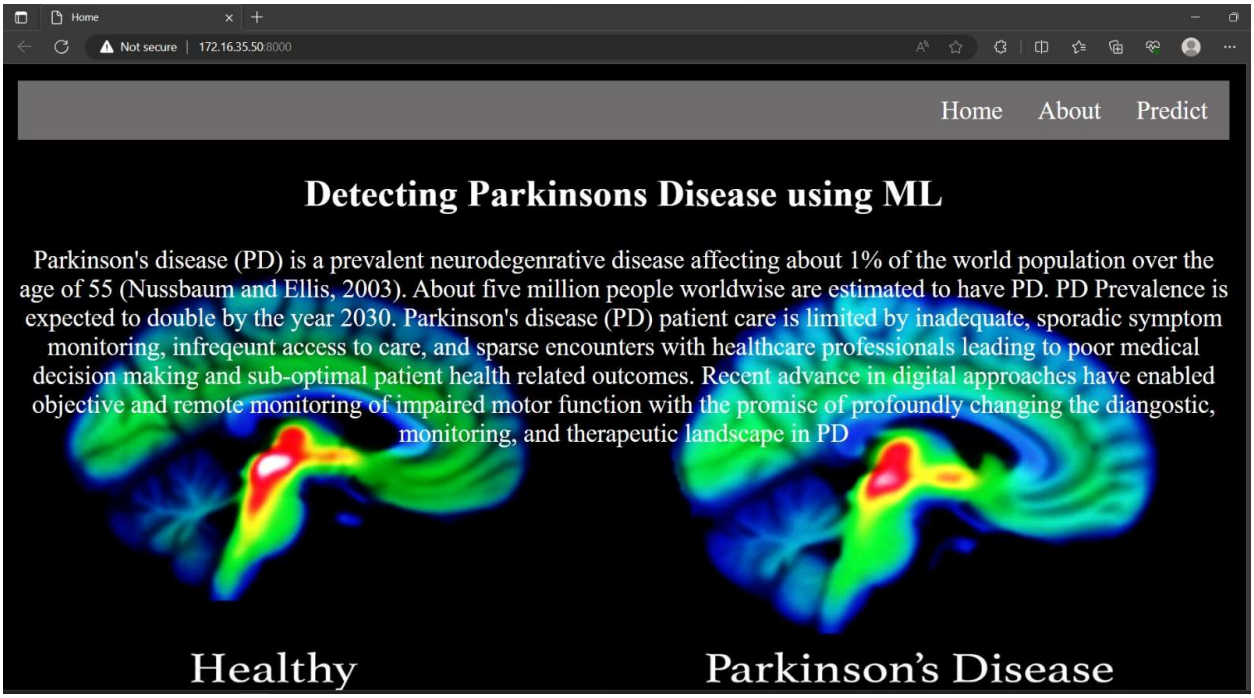


Image 14: home.html Webpage

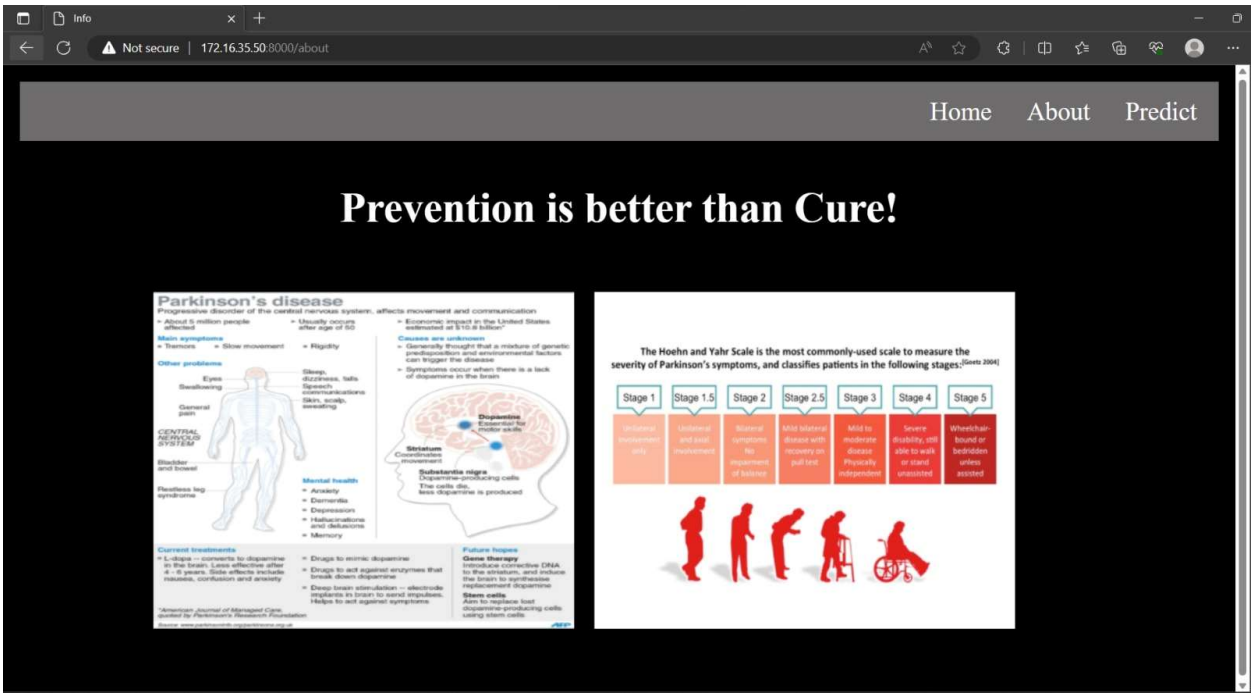


Image 15: about.html Webpage

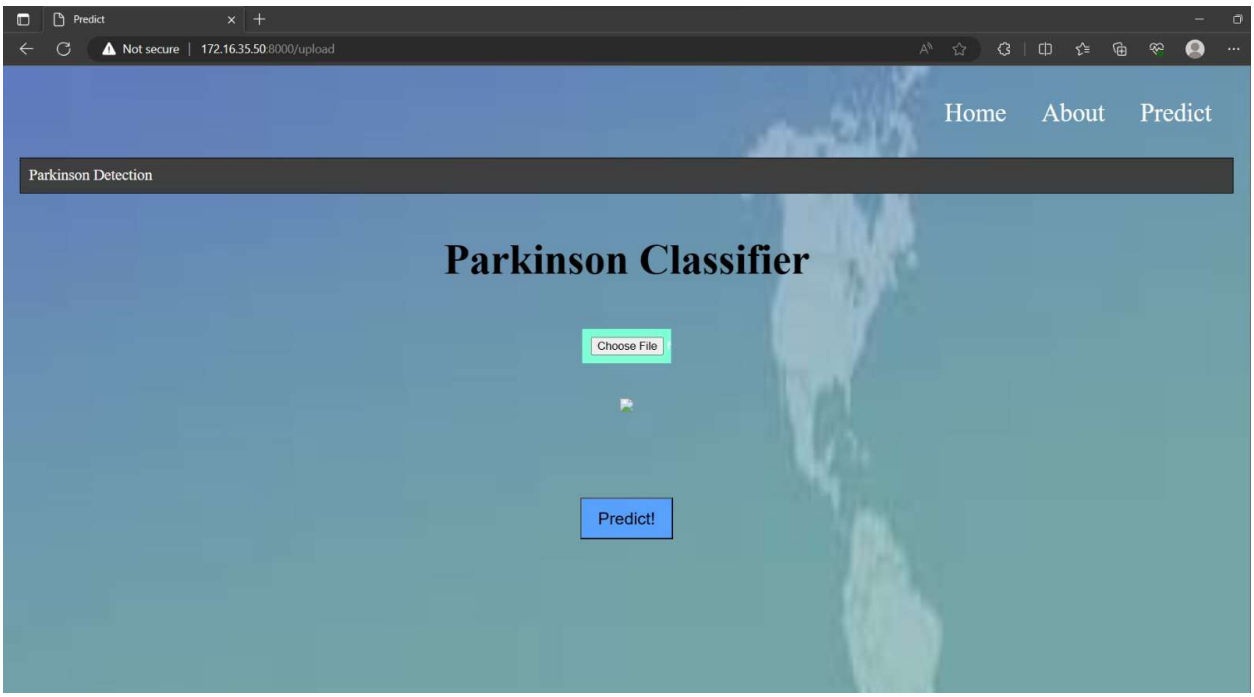


Image 16: index6.html Webpage

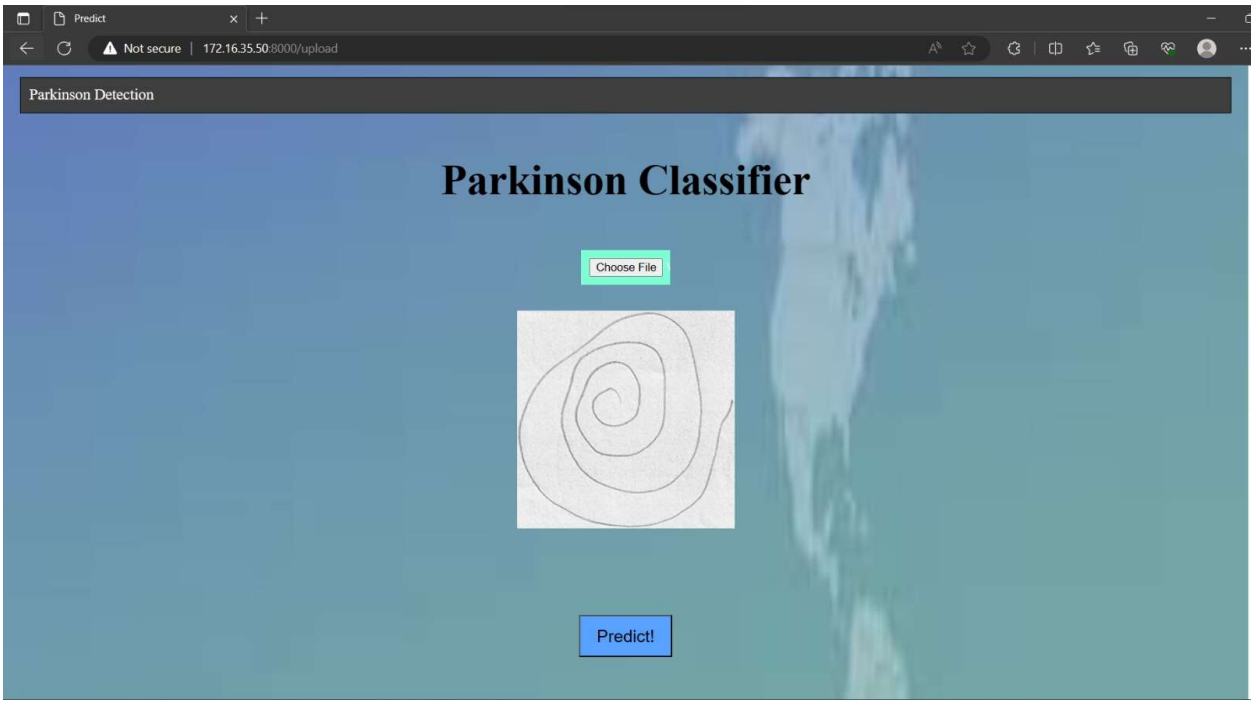


Image 17(a): Image Selected for Prediction

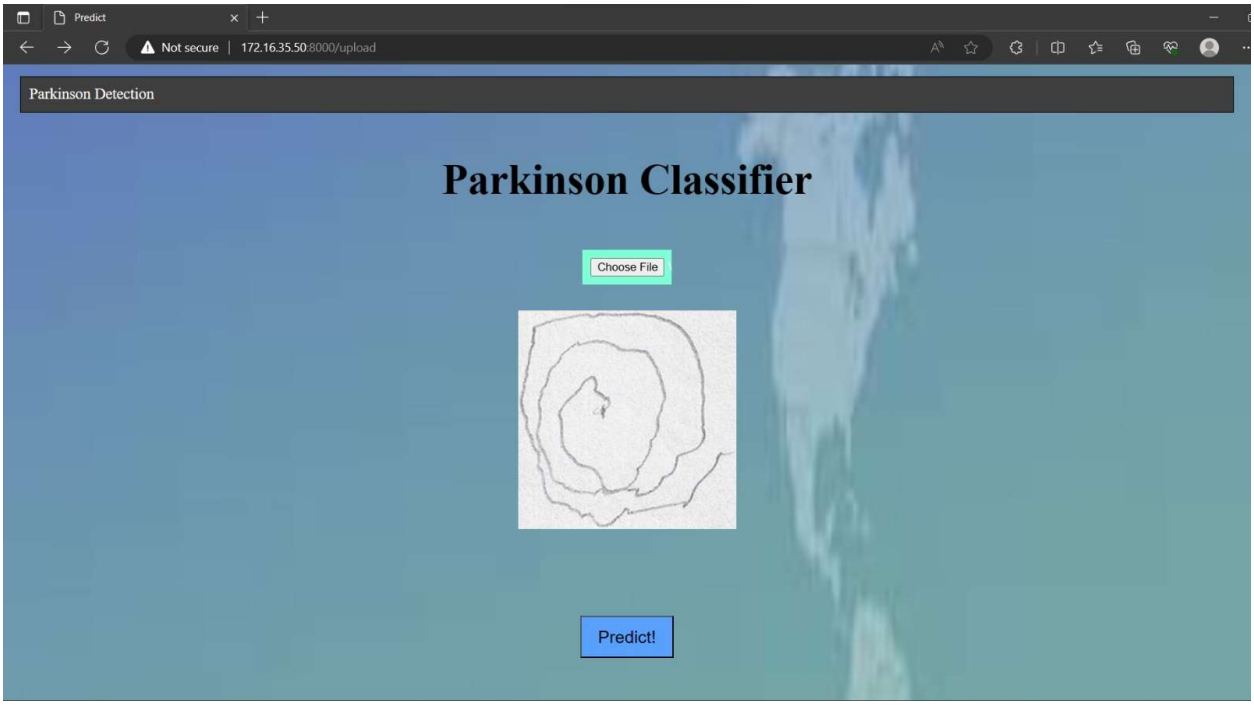


Image 17(b): Image Selected for Prediction

Image 18(a): Output Displayed Healthy

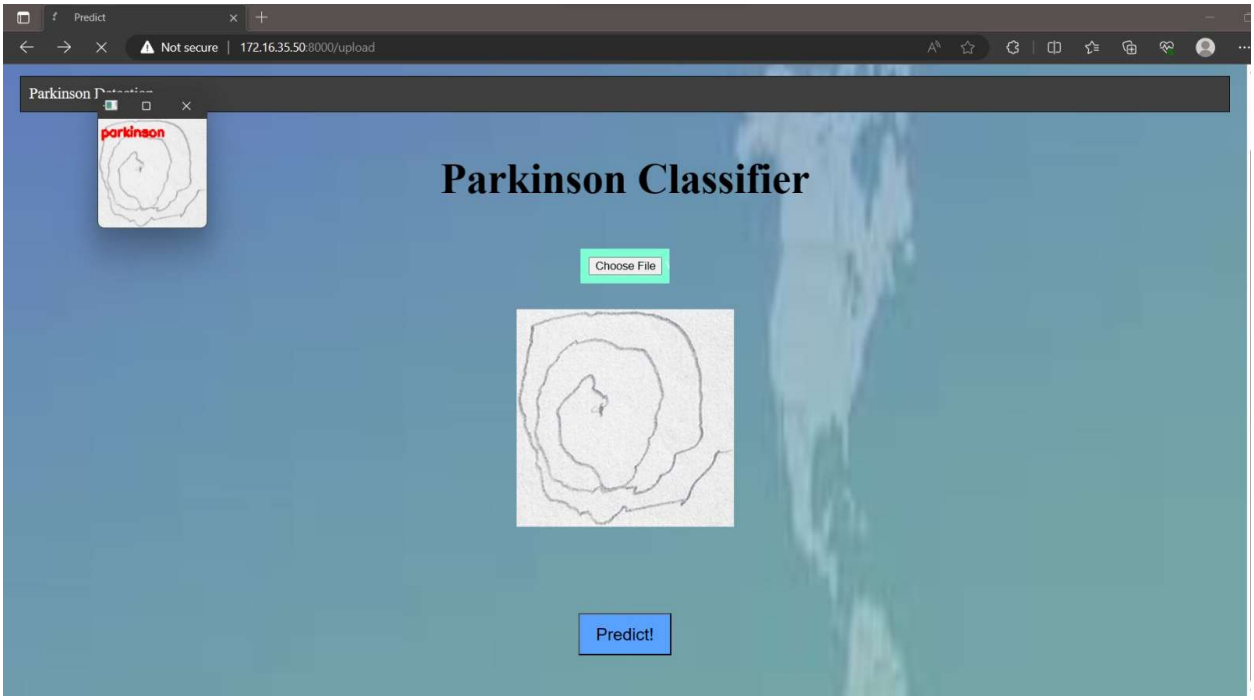


Image 18(b): Output Displayed Parkinson

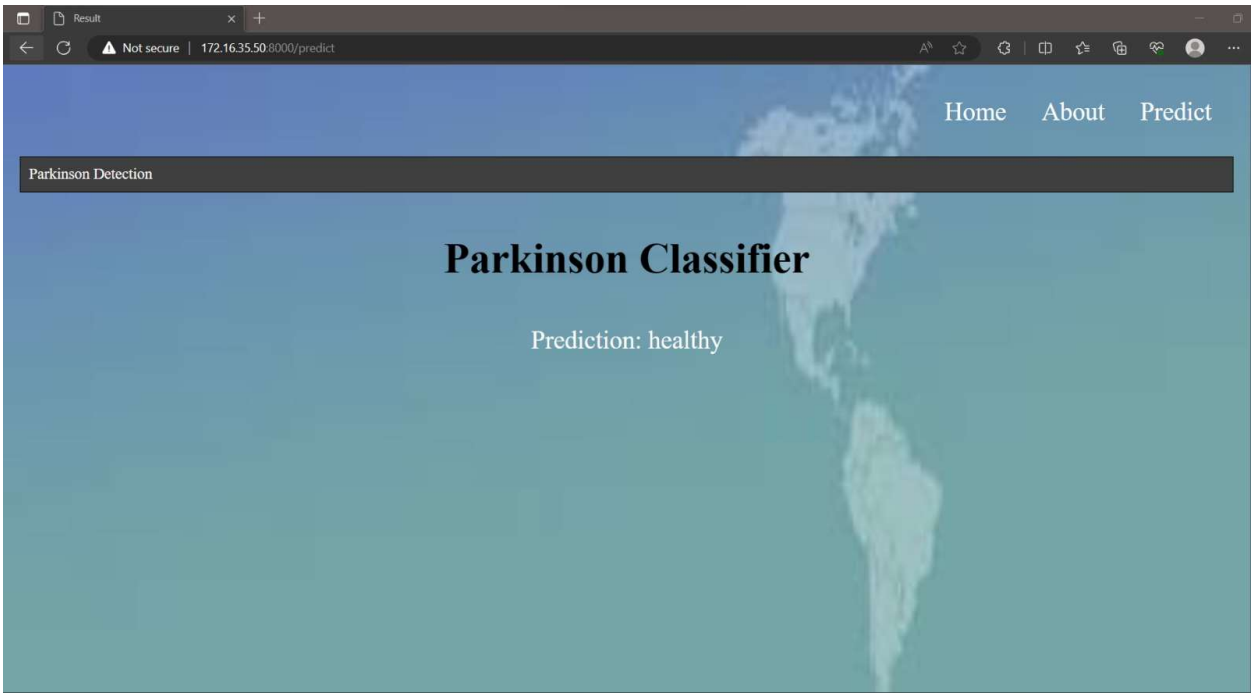


Image 19(a): result.html Webpage Healthy

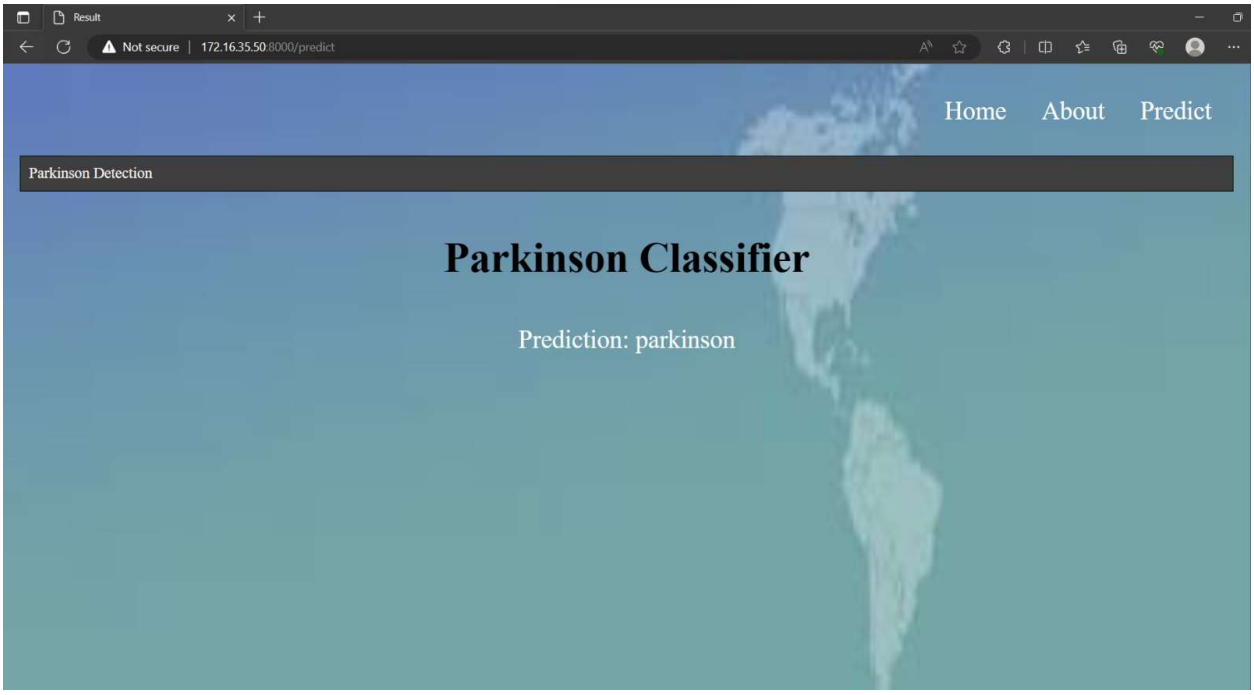


Image 19(b): result.html Webpage Parkinson

7. ADVANTAGES & DISADVANTAGES:

Detecting Parkinson's disease using machine learning software offers several advantages over traditional methods:

- The foremost advantage is improved accuracy. Machine learning algorithms can analyze large amounts of data and identify patterns that may not be apparent to human observers. This can lead to more accurate and reliable detection of Parkinson's disease, potentially allowing for early diagnosis and intervention.
- Secondly, Machine learning algorithms can quickly process and analyze data, making them well-suited for screening large populations. This efficiency can help in identifying potential cases of Parkinson's disease in a timely manner, even in resource-constrained settings.
- Moreover, they offer a Non-Invasive Approach compared to many traditional methods of Parkinson's disease diagnosis that involve invasive procedures or specialized equipment, such as brain imaging techniques. Machine learning software can leverage non-invasive data sources, such as voice recordings or motion sensor data, making the detection process more accessible and less burdensome for patients.

However, there are also some disadvantages to consider:

- Machine learning algorithms heavily rely on high-quality and diverse datasets for training and validation. Availability of such data, especially in the case of rare subtypes or specific populations, can be a challenge.
- Insufficient or biased data may lead to inaccurate predictions or limited generalizability of the models.
- Some machine learning algorithms, such as deep learning models, can be seen as black boxes, meaning their decision-making process is not easily explainable.
- This lack of interpretability can raise concerns, particularly in the medical field, where it is essential to understand the rationale behind diagnostic decisions.

8. APPLICATIONS:

- **Remote Monitoring:** Machine learning models can be incorporated into mobile devices or mobile applications, enabling remote monitoring of Parkinson's patients. These models can continuously analyze data such as tremor intensity, gait patterns or speech characteristics and provide real-time feedback to both patients and healthcare professionals. Remote monitoring facilitates regular assessment and timely intervention, especially for those individuals who have difficulty accessing regular health services.
- **Treatment Optimization:** Machine learning can help optimize treatment plans for patients with Parkinson's disease. Machine learning models can identify optimal drug doses and schedules by analyzing patient data such as medication history, symptom severity and response to therapy. It can help reduce side effects, improve symptom control and improve quality of life for people with Parkinson's disease.

- **Predictive Analytics:** Machine learning algorithms can analyze longitudinal data from patients with Parkinson's disease and identify patterns or markers that can predict disease progression, complications or response to treatment. This predictive ability can help clinicians make informed decisions about future interventions, facilitating early intervention and personalized treatment plans.
- **Drug Research:** Machine learning algorithms can analyze large data sets, including genetic information, clinical data, and imaging data, to identify potential biomarkers, genetic risk factors, or new therapeutic targets for Parkinson's disease. By accelerating research and drug development, machine learning is contributing to the understanding and treatment of Parkinson's disease.

9. CONCLUSION:

In summary, the application of machine learning in Parkinson's disease detection represents a major advance in medical diagnosis. Machine learning algorithms offer multiple advantages such as increased accuracy, efficient screening, non-invasive approaches, continuous monitoring, and potential for personalized medicine. By leveraging a variety of data sources, including voice recordings, motion sensor data, and genetic information, machine learning models can detect early signs of Parkinson's disease, provide objective assessments of symptoms, classify subtypes, and optimize your treatment plan.

Furthermore, future applications of machine learning in Parkinson's disease detection are expected. Advances in multimodal data integration, wearable technology, longitudinal data analysis, explainable AI, integration with clinical decision support systems, collaborative and personalized treatment strategies will enable precision, efficiency and individualization in Parkinson's disease treatment It is expected that the care provided will be further improved.

However, to use machine learning effectively and responsibly in clinical settings, it is critical to address challenges related to data limitations, interpretability, expertise, and ethical considerations. Collaboration between clinicians, researchers, and data scientists is essential to advancing the field, establishing data-sharing initiatives, and refining machine learning models to increase their reliability and versatility.

Overall, Parkinson's disease detection using machine learning has great potential to transform healthcare delivery, enabling early diagnosis, personalized treatment strategies and continuous monitoring. Through continued research and advancements, machine learning can improve the lives of people living with Parkinson's disease and make a significant contribution to expanding our understanding of this complex neurodegenerative disease.

10.FUTURE SCOPE:

Currently, machine learning models for Parkinson's disease detection are primarily based on single data sources such as voice recordings, motion sensor data, and image scans. In the future, we may integrate multiple data modalities, such as combining speech and gait analysis, or incorporating genetic and imaging data. Leveraging a variety of data sources will improve the accuracy of machine learning algorithms and provide a more comprehensive understanding of disease. Wearable devices such as smartwatches and activity trackers are increasingly used for data collection to detect and monitor Parkinson's disease. Future advances in wearable technology may include the development of more specialized sensors that can detect a wide range of motor symptoms and physiological changes associated with Parkinson's disease. This will enable more accurate and real-time monitoring of symptoms, facilitating early detection and intervention. With the prevalence of machine learning algorithms in clinical settings, the need for explainable AI models is growing. Research is currently underway to develop techniques to ensure interpretability and transparency in the decision-making process of machine learning models. This will increase trust between medical professionals and patients and enable broader adoption of machine learning-based Parkinson's disease detection systems. Integrating machine learning algorithms into clinical decision support systems empowers healthcare professionals to make accurate and timely diagnostic decisions. By providing clinicians with evidence-based recommendations and helping them interpret complex data, machine learning can serve as a valuable tool in the clinical setting, improving the efficiency and accuracy of Parkinson's disease detection.

11.BIBLIOGRAPHY:

- https://pyimagesearch.com/2019/04/29/detecting-parkinsons-disease-with-opencv-computer-vision-and-the-spiral-wave-test/
- https://www.frontiersin.org/articles/10.3389/fnagi.2021.633752/full
- https://www.kaggle.com/code/vikasukani/detecting-parkinson-s-disease-machine-learning/comments
- https://www.analyticsvidhya.com/blog/2021/07/parkinson-disease-onset-detection-using-machine-learning/
- https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd
- https://www.analyticsvidhya.com/blog/2021/09/a-beginners-guide-to-image-processing-with-opencv-and-python/

12.APPENDIX:

- **Source Code:**

a) ADS_Project.ipynb

```
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 argparse
import cv2
import os
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
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
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))
trainingPath=r"C:\Users\Sriya\Downloads\Dataset\spiral\training"
testingPath=r"C:\Users\Sriya\Downloads\Dataset\spiral\testing"
print("[INFO] loading data...")
```

```
(X_train,y_train)=load_split(trainingPath)
(X_test,y_test)=load_split(testingPath)
le = LabelEncoder()
y_train= le.fit_transform(y_train)
y_test = le.transform(y_test)
print(X_train.shape,y_train.shape)
print("[INFO] training model...")
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train,y_train)
testingPaths = list(paths.list_images(testingPath))
idxs = np.arange(0, len(testingPaths))
idxs = np.random.choice(idxs, size=(25,), replace=False)
images = []
for i in idxs:
    image = cv2.imread(testingPaths[i])
    output = image.copy()
    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]
    features = quantify_image(image)
    preds = model.predict([features])
    label = le.inverse_transform(preds)[0]
    color = (0, 255, 0) if label == "healthy" else (0, 0, 255)
    cv2.putText(output, label, (3, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5,color, 2)
    images.append(output)
montage = build_montages(images, (128, 128), (5, 5))[0]
cv2.imshow("Output", montage)
cv2.waitKey(0)
predictions = model.predict(X_test)
accuracy = {}
sensitivity= {}
specificity={}
cm = confusion_matrix(y_test, predictions).flatten()
print(cm)
(tn, fp, fn, tp) = cm
accuracy["Random Forest"]= (tp + tn) / float(cm.sum())
sensitivity["Random Forest"] = tp / float(tp + fn)
specificity["Random Forest"] = tn / float(tn + fp)
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test,predictions))
print("[INFO] training model...")
from sklearn.linear_model import LogisticRegression
model =  LogisticRegression(random_state=0)
model.fit(X_train,y_train)
testingPaths = list(paths.list_images(testingPath))
idxs = np.arange(0, len(testingPaths))
idxs = np.random.choice(idxs, size=(25,), replace=False)
images = []
for i in idxs:
    image = cv2.imread(testingPaths[i])
    output = image.copy()
    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]
    features = quantify_image(image)
    preds = model.predict([features])
    label = le.inverse_transform(preds)[0]
    color = (0, 255, 0) if label == "healthy" else (0, 0, 255)
    cv2.putText(output, label, (3, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5,color, 2)
    images.append(output)
montage = build_montages(images, (128, 128), (5, 5))[0]
cv2.imshow("Output", montage)
cv2.waitKey(0)
predictions = model.predict(X_test)
cm = confusion_matrix(y_test, predictions).flatten()
print(cm)
(tn, fp, fn, tp) = cm
accuracy["Logistic Regression"]= (tp + tn) / float(cm.sum())
sensitivity["Logistic Regression"] = tp / float(tp + fn)
specificity["Logistic Regression"]= tn / float(tn + fp)
print(classification_report(y_test,predictions))
print("[INFO] training model...")
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier(criterion='entropy',random_state=0)
model.fit(X_train,y_train)
testingPaths = list(paths.list_images(testingPath))
```

```
idxs = np.arange(0, len(testingPaths))
idxs = np.random.choice(idxs, size=(25,), replace=False)
images = []
for i in idxs:
    image = cv2.imread(testingPaths[i])
    output = image.copy()
    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]
    features = quantify_image(image)
    preds = model.predict([features])
    label = le.inverse_transform(preds)[0]
    color = (0, 255, 0) if label == "healthy" else (0, 0, 255)
    cv2.putText(output, label, (3, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5,color, 2)
    images.append(output)
montage = build_montages(images, (128, 128), (5, 5))[0]
cv2.imshow("Output", montage)
cv2.waitKey(0)
predictions = model.predict(X_test)
cm = confusion_matrix(y_test, predictions).flatten()
print(cm)
(tn, fp, fn, tp) = cm
accuracy["Decision Tree"]= (tp + tn) / float(cm.sum())
sensitivity["Decision Tree"] = tp / float(tp + fn)
specificity["Decision Tree"]= tn / float(tn + fp)
print(classification_report(y_test,predictions))
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier()
model.fit(X_train,y_train)
testingPaths = list(paths.list_images(testingPath))
idxs = np.arange(0, len(testingPaths))
idxs = np.random.choice(idxs, size=(25,), replace=False)
images = []
for i in idxs:
    image = cv2.imread(testingPaths[i])
    output = image.copy()
    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]
    features = quantify_image(image)
    preds = model.predict([features])
    label = le.inverse_transform(preds)[0]
    color = (0, 255, 0) if label == "healthy" else (0, 0, 255)
    cv2.putText(output, label, (3, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5,color, 2)
    images.append(output)
montage = build_montages(images, (128, 128), (5, 5))[0]
cv2.imshow("Output", montage)
cv2.waitKey(0)
predictions = model.predict(X_test)
cm = confusion_matrix(y_test, predictions).flatten()
print(cm)
(tn, fp, fn, tp) = cm
accuracy["KNN"]= (tp + tn) / float(cm.sum())
sensitivity["KNN"] = tp / float(tp + fn)
specificity["KNN"]= tn / float(tn + fp)
print(classification_report(y_test,predictions))
print("Accuracy Scores: \n")
for key,value in accuracy.items():
    print(key, ": ", value)
print("\nSensitivity: \n")
for key,value in sensitivity.items():
    print(key, ": ", value)
print("\nSpecificity: \n")
for key,value in specificity.items():
    print(key, ": ", value)
models = accuracy.keys()
results = accuracy.values()
accuracy_scores = accuracy.values()
sensitivity_scores = sensitivity.values()
specificity_scores = specificity.values()
fig, axes = plt.subplots(3, 1, figsize=(8, 15))
axes[0].bar(models, accuracy_scores)
axes[0].set_ylabel('Accuracy Score')
axes[0].set_title('Comparison of Accuracy Scores')
axes[1].bar(models, sensitivity_scores)
axes[1].set_ylabel('Sensitivity Score')
```

```
axes[1].set_title('Comparison of Sensitivity Scores')
axes[2].bar(models,specificity_scores)
axes[2].set_ylabel('Specificity Score')
axes[2].set_title('Comparison of Specificity Scores')
plt.tight_layout()
plt.show()
final_model = RandomForestClassifier(n_estimators=100)
final_model.fit(X_train,y_train)
import pickle
pickle.dump(final_model, open("parkinson.pkl", "wb"))
```

b) index6.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Predict</title>
    <style>
      /* CSS for the navbar */
      body {
        background-image: url("../static/index6bg.jpeg");
        background-size: 100% 170%;
        background-repeat: no-repeat;
        text-align: center;
        color: white;
        font-size: 30px;
        padding: 10px;
        margin: 10px;
      }
      nav {
        color: #fff;
        display: flex;
        justify-content: flex-end;
        align-items: center;
        padding: 10px;
      }

      ul {
        list-style-type: none;
        margin: 0;
        padding: 0;
```

```
      display: flex;
    }
    li {
      margin-left: 10px;
    }
    li a {
      display: block;
      color: #fff;
      text-align: center;
      text-decoration: none;
      padding: 8px 16px;
    }
    .buttons {
      display: flex;
    }
    .buttons li {
      margin-left: 10px;
    }
  </style>
 </head>
 <body>
  <nav>
    <ul class="buttons">
      <li><a href="/">Home</a></li>
      <li><a href="about">About</a></li>
      <li><a href="upload">Predict</a></li>
    </ul>
  </nav>
  <!-- Content of the page -->
  <p
    style="
      border: solid 1px black;
      background-color: rgb(63, 63, 63);
      text-align: left;
      padding: 10px;
      font-size: 18px;
    "
  >
    Parkinson Detection
  </p>
```

```
  <p style="font-size: 50px; color: black"><b>Parkinson Classifier</b></p>
  <form action="/predict" method="POST" enctype="multipart/form-data">
    <p>
      <input
        type="file"
        name="file"
        id="file"
        value="Choose..."
        style="width: 85px; padding: 10px; background-color: aquamarine"
      />
    </p>
    <p><img id="previewImage" src="#" /></p>
    <img src="{{filepath}}" />
    <p>
      <input
        type="submit"
        value="Predict!"
        style="
          width: 110px;
          height: 50px;
          padding: 5px;
          background-color: rgb(89, 161, 255);
          font-size: 20px;
        "
      />
    </p>
  </form>
  <p>{{result}}</p>
</body>
<script>
  const imageInput = document.getElementById("file");
  const previewImage = document.getElementById("previewImage");

  imageInput.addEventListener("change", function (event) {
    const file = event.target.files[0];
    const reader = new FileReader();

    reader.onload = function (e) {
      previewImage.src = e.target.result;
    };
```

```
    reader.readAsDataURL(file);
  });
 </script>
</html>
```

c) home.html

```
<!DOCTYPE html>
<html>
 <head>
  <title>Home</title>
  <style>
   /* CSS for the navbar */
   body {
    background-image: url("../static/homebg.jpg");
    background-repeat: no-repeat;
    background-size: 100% 150%;
    text-align: center;
    color: white;
    font-size: 30px;
    padding: 10px;
    margin: 10px;
   }
   nav {
    background-color: #6e6c6c;
    color: #fff;
    display: flex;
    justify-content: flex-end;
    align-items: center;
    padding: 10px;
   }
   ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    display: flex;
   }
   li {
    margin-left: 10px;
   }
```

```
    li a {
      display: block;
      color: #fff;
      text-align: center;
      text-decoration: none;
      padding: 8px 16px;
    }
    .buttons {
      display: flex;
    }
    .buttons li {
      margin-left: 10px;
    }
  </style>
 </head>
 <body>
  <nav>
   <ul class="buttons">
    <li><a href="">Home</a></li>
    <li><a href="about">About</a></li>
    <li><a href="upload">Predict</a></li>
   </ul>
  </nav>

  <!-- Content of the page -->
  <h2>Detecting Parkinsons Disease using ML</h2>
  <p>
   Parkinson's disease (PD) is a prevalent neurodegenrative disease affecting
   about 1% of the world population over the age of 55 (Nussbaum and Ellis,
   2003). About five million people worldwise are estimated to have PD. PD
   Prevalence is expected to double by the year 2030. Parkinson's disease
   (PD) patient care is limited by inadequate, sporadic symptom monitoring,
   infreqeunt access to care, and sparse encounters with healthcare
   professionals leading to poor medical decision making and sub-optimal
   patient health related outcomes. Recent advance in digital approaches have
   enabled objective and remote monitoring of impaired motor function with
   the promise of profoundly changing the diangostic, monitoring, and
   therapeutic landscape in PD
  </p>
 </body>
```

```
</html>
```

d) about.html

```
<!DOCTYPE html>
<html>
 <head>
  <title>Info</title>
  <style>
   /* CSS for the navbar */
   body {
    background-repeat: no-repeat;
    background-color: black;
    text-align: center;
    color: white;
    font-size: 30px;
    padding: 10px;
    margin: 10px;
   }
   nav {
    background-color: #6e6c6c;
    color: #fff;
    display: flex;
    justify-content: flex-end;
    align-items: center;
    padding: 10px;
   }
   ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    display: flex;
   }
   li {
    margin-left: 10px;
   }
   li a {
    display: block;
    color: #fff;
    text-align: center;
    text-decoration: none;
```

```
      padding: 8px 16px;
    }
    .buttons {
      display: flex;
    }
    .buttons li {
      margin-left: 10px;
    }
  </style>
 </head>
 <body>
  <nav>
   <ul class="buttons">
    <li><a href="/">Home</a></li>
    <li><a href="about">About</a></li>
    <li><a href="upload">Predict</a></li>
   </ul>
  </nav>
  <!-- Content of the page -->
  <p style="font-size: 50px"><b>Prevention is better than Cure!</b></p>
  <p><img src="../static/aboutbg.png" /></p>
 </body>
</html>
```

e) result.html

```
<!DOCTYPE html>
<html>
 <head>
  <title>Result</title>
  <style>
   /* CSS for the navbar */
   body {
    background-image: url("../static/index6bg.jpeg");
    background-size: 100% 200%;
    background-repeat: no-repeat;
    text-align: center;
    color: white;
    font-size: 30px;
    padding: 10px;
    margin: 10px;
```

```
    }
    nav {
      color: #fff;
      display: flex;
      justify-content: flex-end;
      align-items: center;
      padding: 10px;
    }
    ul {
      list-style-type: none;
      margin: 0;
      padding: 0;
      display: flex;
    }
    li {
      margin-left: 10px;
    }
    li a {
      display: block;
      color: #fff;
      text-align: center;
      text-decoration: none;
      padding: 8px 16px;
    }
    .buttons {
      display: flex;
    }
    .buttons li {
      margin-left: 10px;
    }
  </style>
 </head>
 <body>
  <nav>
    <ul class="buttons">
      <li><a href="/">Home</a></li>
      <li><a href="about">About</a></li>
      <li><a href="upload">Predict</a></li>
    </ul>
  </nav>
```

```
    <!-- Content of the page -->
    <p
      style="
        border: solid 1px black;
        background-color: rgb(63, 63, 63);
        text-align: left;
        padding: 10px;
        font-size: 18px;
      "
    >
      Parkinson Detection
    </p>
    <p style="font-size: 50px; color: black"><b>Parkinson Classifier</b></p>
    <p>Prediction: {{result}}</p>
  </body>
</html>
```

f) app.py

```
import pickle
import cv2
from skimage import feature
from flask import Flask, request, render_template
import os.path
app = Flask(__name__)
@app.route("/")
def about():
    return render_template("home.html")
@app.route("/about")
def home():
    return render_template("about.html")
@app.route("/info")
def information():
    return render_template("info.html")
@app.route("/upload")
def test():
    return render_template("index6.html")
@app.route('/predict', methods=['POST'])
def upload():
    if request.method == 'POST':
        f = request.files['file']
```

```
        basepath = os.path.dirname(__file__)
        filepath = os.path.join(basepath, "uploads", f.filename)
        f.save(filepath)
        print("[INFO] Loading model...")
        model = pickle.loads(open('parkinson.pkl', "rb").read())
        image = cv2.imread(filepath)
        output = image.copy()
        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]
        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]]
        color = (0, 255, 0) if result == "healthy" else (0, 0, 255)
        cv2.putText(output, result, (3, 20),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
        cv2.imshow("Output", output)
        cv2.waitKey(0)
        print(filepath)
        return render_template("result.html", result=result, filename=f.filename)
    return None
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=8000, debug=True)
```