**DETECTING PARKINSON’S DISEASE USING MACHINE LEARNING**

MITSUMI NANDI

1. Introduction:
2. Overview:

Parkinson’s Disease is a progressive brain disorder that eventually leads to shaking, stiffness and can result in difficulty with walking, coordination and balance. This disease can get worse over time and there is no proper medical cure to heal this disorder completely. But detecting Parkinson’s Disease at an early stage followed by proper medical treatment can significantly enhance the quality of life and relieve the symptoms. Researchers have discovered that patients suffering from Parkinson’s Diseases have slower drawing speed and lower pen pressure. Tremors and Rigidity in the muscles are some symptoms of this disease, therefore making it difficult to draw smooth spirals and waves.

1. Purpose:

Our purpose in this project is to detect the presence of Parkinson’s Disease with the aid of the drawings alone instead of measuring the pressure on the pen and speed of drawing. We quantify the visual appearance, using HOG method, of the given drawings and then train the machine learning model to classify as Healthy or Parkinson. We use Histogram of Oriented Gradients (HOG) image descriptor and Random Forest Classifier (ML and OpenCV) to automatically detect the presence of Parkinson’s Disease and build web application for the same using FLASK framework.

1. Literature Survey
2. Existing Problem:

The major problem that lies in the medical field is that there is no cure for Parkinson’s Disease but early detection can lead to early treatment which can significantly improve the patient’s lifestyle and alleviate or reduce the symptoms. Hence, early detection can ensure that the disorder doesn’t get untreated and eventually get much worse over time.

1. Proposed Solution:

As we know from research that people suffering from Parkinson’s Disease have lower drawing speed and have lesser pressure on the pen, therefore making it difficult for them to draw smooth curves, we can detect the presence of this disorder at an early stage via the drawings made and predicting using a machine learning model and OpenCV. We use Histogram of Oriented Gradients, image descriptor and Random Forest Classifier in the program code.

1. Theoretical Analysis:
2. Block Diagram

Training the data

Outputs the prediction

ML Algorithm

(Random Forest Classifier)

Dataset of Images with drawings

User Interface

Evaluation of the model

Image pre-processing

Model

Testing the data

User

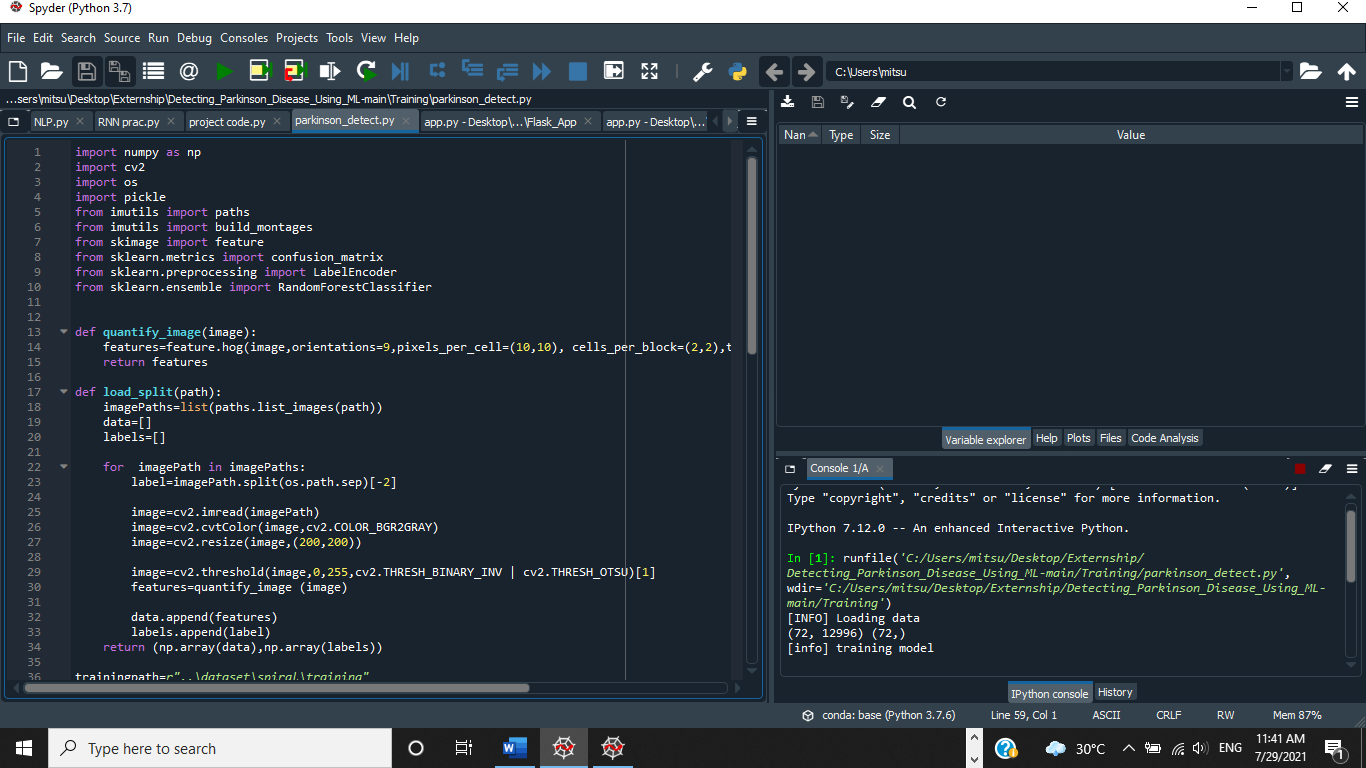
Inputting an image with drawing

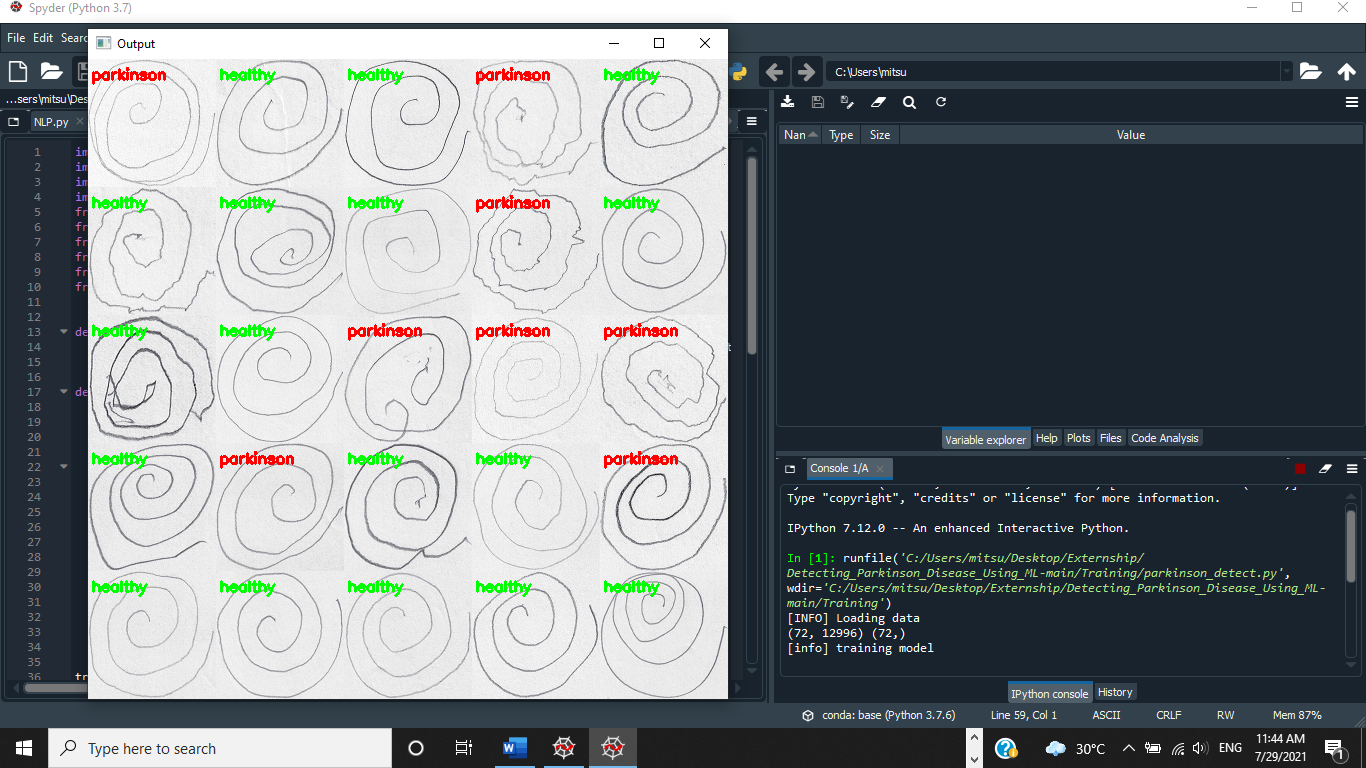
1. Software Designing:

* Importing necessary libraries
* OpenCV for rescaling, resizing and thresholding images
* Imutils package contains many convenience functions for image pre-processing. From this package, build\_monatges is used for visualization and paths import helps to extract file paths.
* Sklearn.metrics measures classification performance
* Sklearn.preprocessing is the package containing Random Forest Classifier (an ensemble learning method for classification that operates by constructing multitude of decision trees at training time)
* Scikit-image is package for image pre=processing. Histogram of Oriented Gradients will come from the feature import.
* Pickle coverts python objects into byte stream.
* Next, we load the training and testing data, For this we have used imutils to get image path. Each label is extracted from the os.path.split() method in Python which is used to split the pathname into head and tail. Here, the tail is the last pathname component and the head is everything leading up to that. Images were read via imread() and converted to grayscale and resized. Next the image was thresholded for the image to appear as white on black background. Features have been extracted via quantify\_image function. The features and label are appended to the data and labels lists respectively. Finally, data and labels are converted to NumPy arrays and returned in a tuple.
* Features from each image is extracted using the function, quantify\_image. The structural descriptor, HOG will quantify changes in local gradient of the image and will be able to quantify how the directions of the spiral change. It will be able to analyse if the drawings have more shake to them. The 3 parameters: orientations, pixels\_per\_cell and cells\_per\_block control dimensionality of resulting feature vector.
* Label Encoding is done to turn categorical values into 0s and 1s (0 for Healthy and 1 for Parkinson)
* The machine learning algorithm used in this project is Random Forest classifier. So, the model is trained using Random Forest Classifier with n\_estimator as 100
* Next, we test the model by selecting 25 random images from the test model and initialize the output images for montage. Images are randomly sampled and pre-processed like before (convert to grayscale, resize and threshold) using OpenCV. Then automatically classify using HOG and Random Forest Classifier. Images are quantified with HOG features and image is classified by passing these features to model.predict and the features are appended to images list to develop a montage. On executing, the montage gets displayed, until a key is pressed.
* The model is then evaluated using confusion matrix which gives us true negative, false positive, false negative and true positive, from which we can also calculate accuracy score using (tp+tn)/float (cm.sum())
* The model is then saved using Pickel.
* Application Building:
  + Four HTML pages are created: about.html, base.html, index6.html, info.html
  + Next in app.py, flask module is imported, then required libraries are imported and html pages are rendered as desired.
  + The upload function retrieves all the values from HTML using post request and we request user to upload an image using request function. Input from user is taken and pre-processed (convert to grayscale, resize and threshold). Then the data is given to model to predict the output. Output is displayed on OpenCV window and HTML page.
  + To run the app, we open Anaconda Navigator and locate to where our app.py is located and enter command python app.py and we receive the URL of where the application is running and paste it on local browser.

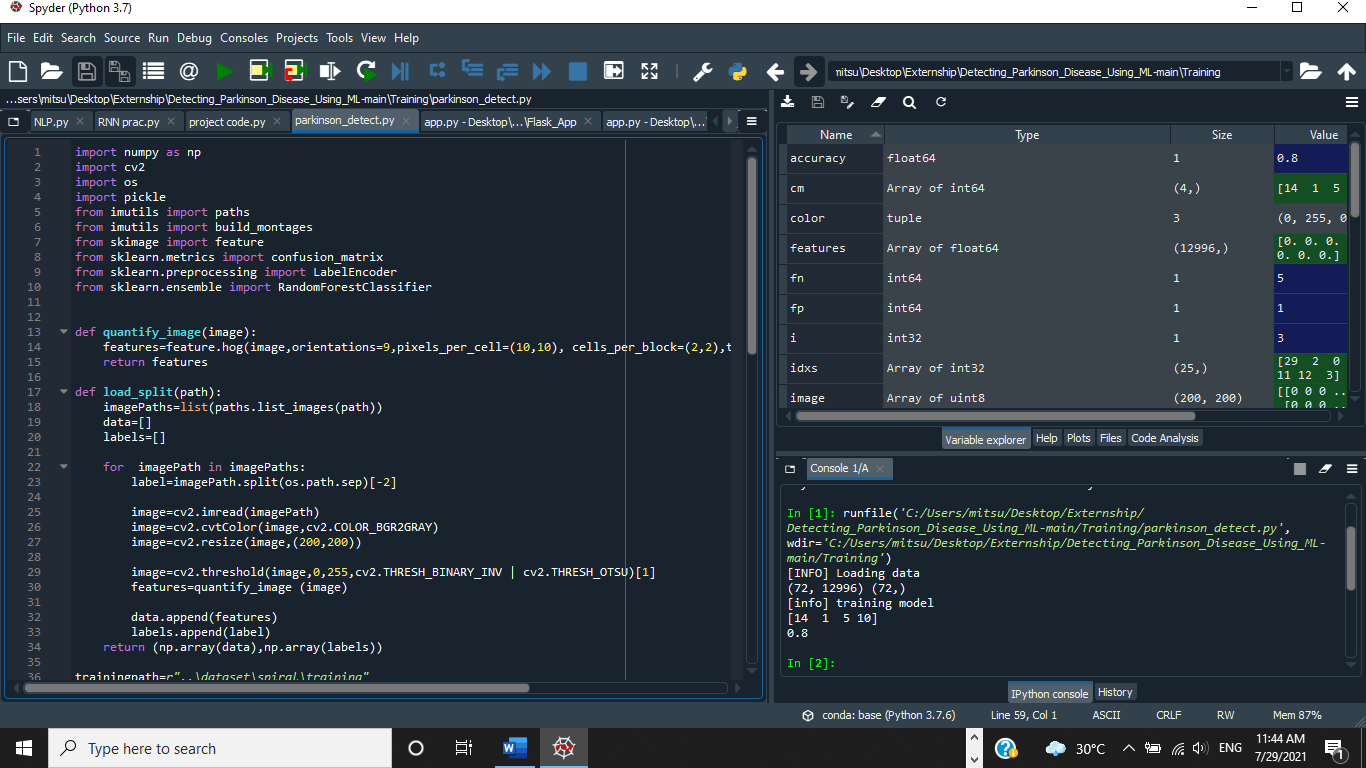
1. Experimental Investigations:

* After executing the python program, Parkinson\_detect, we observe that after training the model and testing, a montage is displayed until a key is pressed. It shows 50 images and they are labeled as Parkinson(red) or Healthy(green) depending on the geometric pattern.

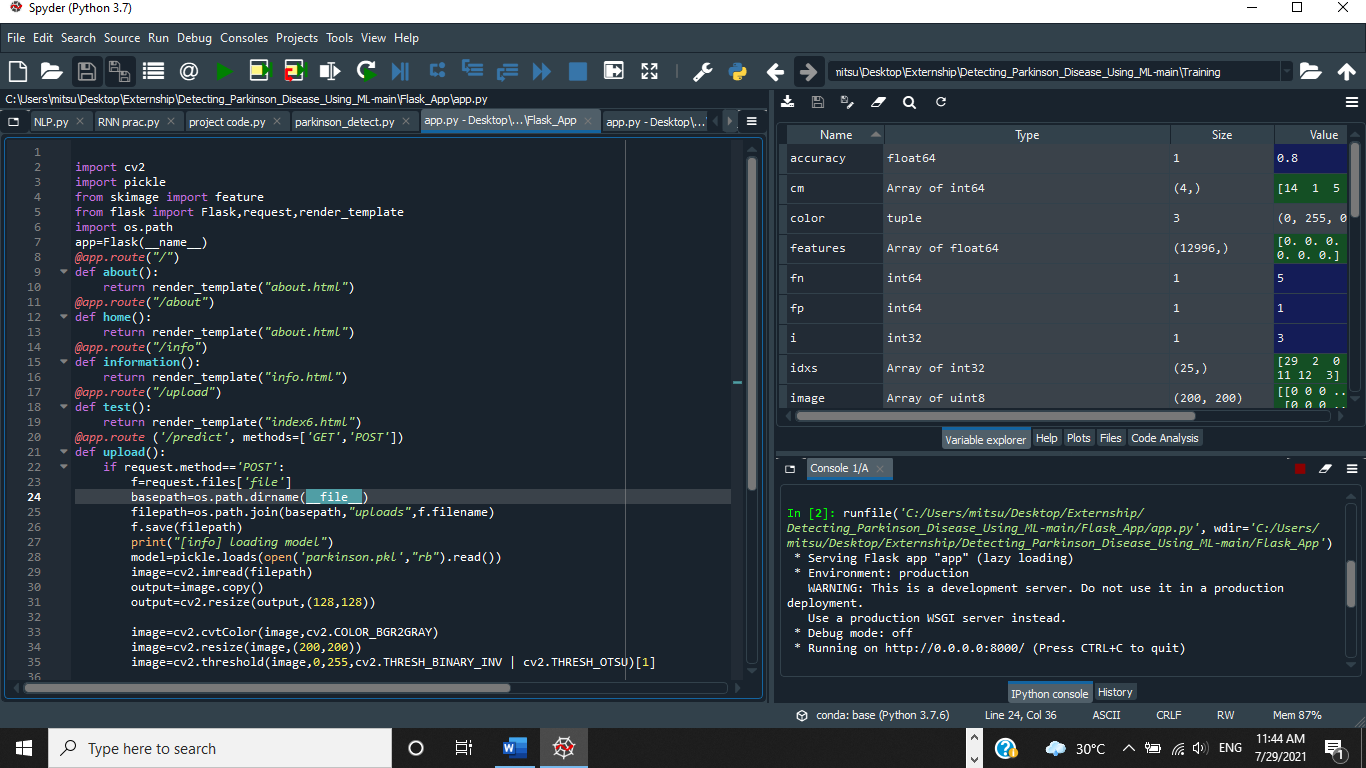


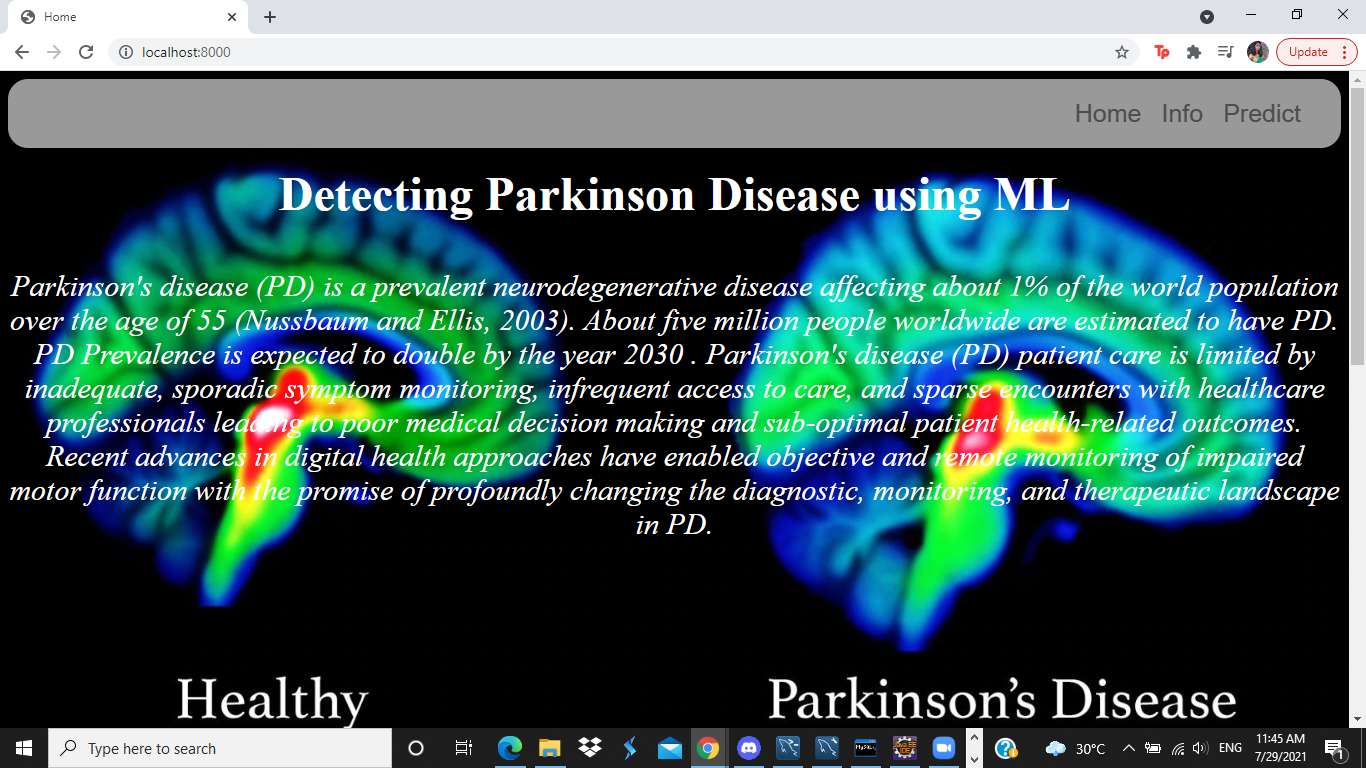


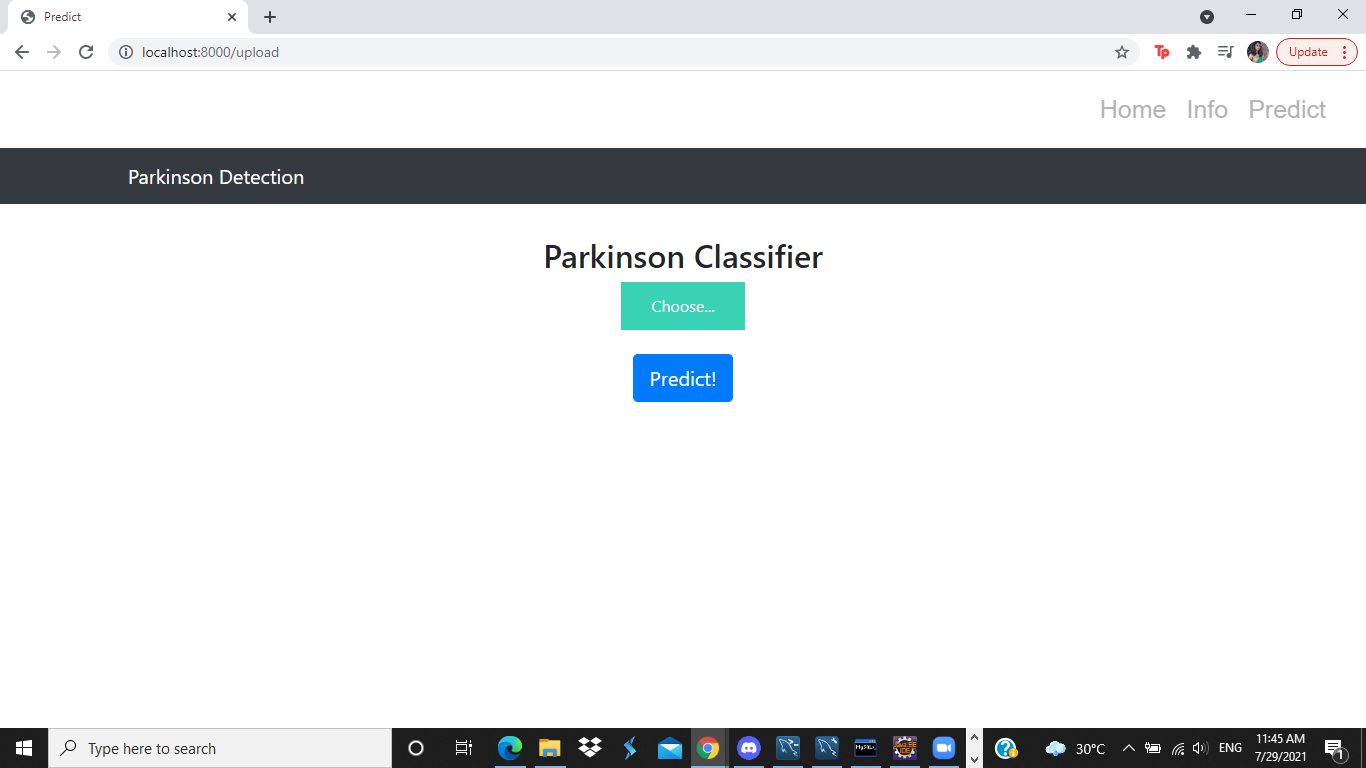
- Then on pressing a key, we can observe the confusion matrix as well as the accuracy score of the model.

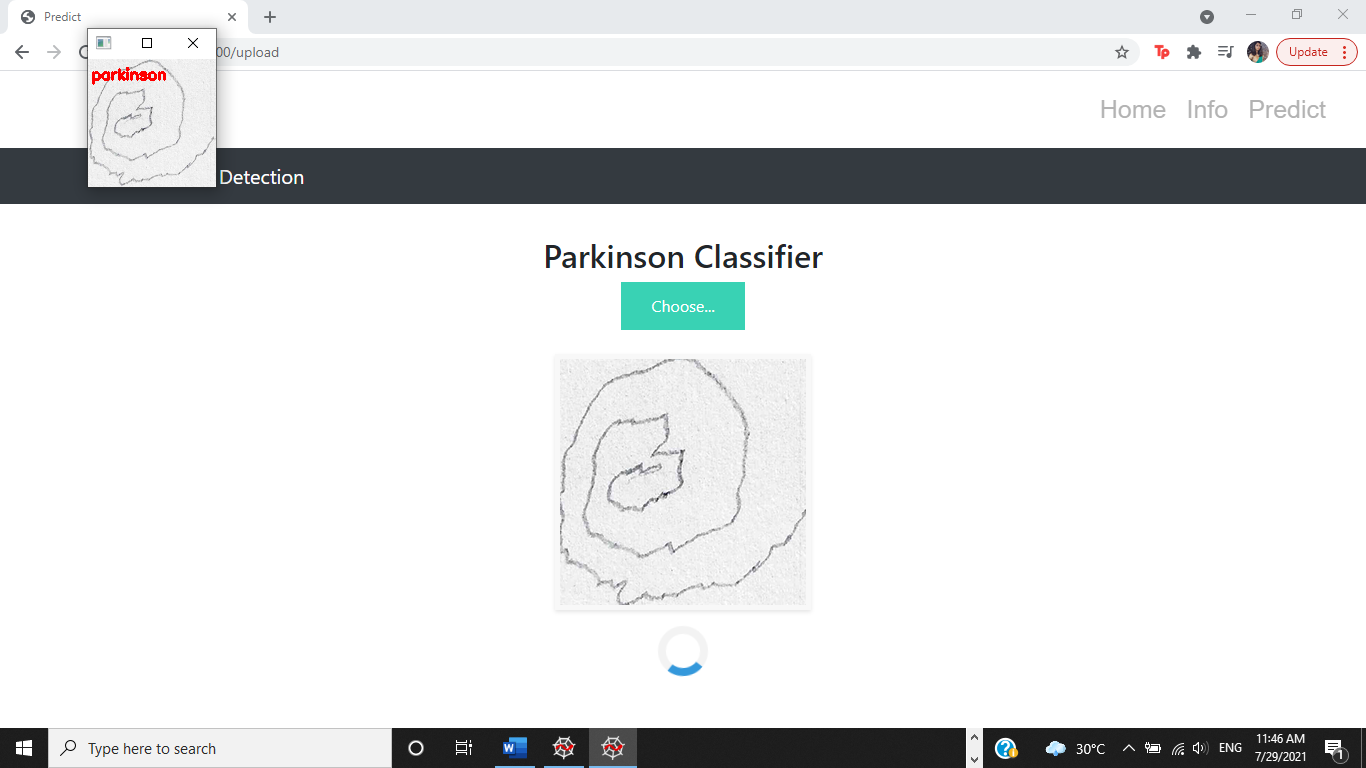


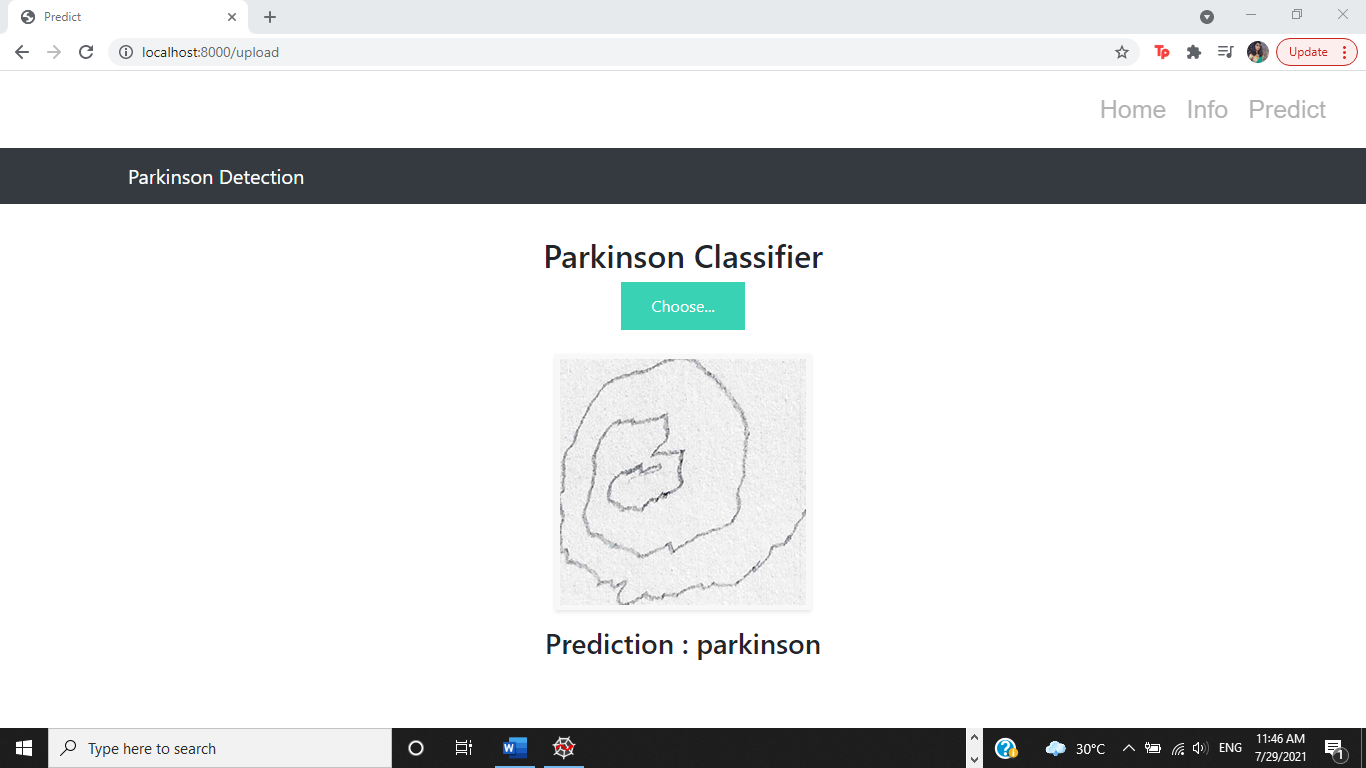
* After executing the python program, app.py, we then open <http://localhost:8000/> on local browser and we observe the home page and we click on predict and upload an image and select the predict image and a new tab, predict.html will display the output. On pressing any key, the output gets displayed on the webpage screen.











1. Flowchart:

Collecting and downloading the dataset

Importing all required libraries

Image pre-processing

Load training and testing dataset

Quantify Images using structural descriptor, HOG

Label Encoding

Testing and Training the Data with Random Forest Classifier.

Model Building

Model Evaluation using cm matrix and accuracy score

Saving the model using pickle

Building HTML pages and saving to template

Application Building

Building python code with flask framework.

Running the web application

1. Result:

A machine learning model was successfully built to detect Parkinson’s Disease using drawings of spirals. Various concepts like image pre-processing, image quantifying using HOG (Histogram of Oriented Gradients), Label Encoder, Random Forest Classification, OpenCV, accuracy and flask framework were applied and executed correctly to build the model. Finally, a model was made and a web application was also built for the same to upload pictures of drawings and to predict the presence of Parkinson’s Disease.

7. Advantages and Disadvantages:

=> Advantages- This method will help to a great extent in medical field with diagnosis of Parkinson’s disease and starting early treatment after early detection. This procedure is:

-simple and convenient

-quick and time effective

-affordable and cost-effective.

=> Disadvantages- A major disadvantage can be that the model is not 100% accurate and this can lead to some errors at times. The model can lead to predictions that can be categorized as False Positive or False Negative. So, we cannot completely rely on this ML model and will probably have to do further medical testing to confirm the presence of Parkinson’s Disease.

8. Applications:

This project can play an indispensable role in the medical field and provide a very simple and quick technique to doctors and health professionals for diagnosis of Parkinson’s Disease. This way the project can also promote early detection and early treatment to prevent the disease from getting progressively worse.

9. Conclusion:

* Successfully created a Machine Learning model using ML and OpenCV for a simple and relatively quick method to detect Parkinson’s Disease using hand drawn pictures of spirals.
* Learned to pre-process images and also evaluate the model by finding the accuracy.
* Successfully built a web application for diagnosis of Parkinson’s

Disease using the flask framework.

10. Future Scope:

This project can be expanded to predict the presence of other diseases such as Alzheimer’s Disease as well. In Alzheimer’s Disease, the structure of the brain degenerates progressively and we can use ML model to detect the brain structures that have been degenerated and predict the presence of Alzheimer’s Disease.

11. Bibliography:

Yadav, D. (2019, December 9). *Categorical encoding using Label-Encoding And one-hot-encoder*. Medium. https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd.

12. Appendix:

*Source code of Parkinson\_detect.py*

import numpy as np

import cv2

import os

import pickle

from imutils import paths

from imutils import build\_montages

from skimage import feature

from sklearn.metrics import confusion\_matrix

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier

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"..\dataset\spiral\training"

testingpath=r"..\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)

cm=confusion\_matrix(y\_test, predictions).flatten()

print(cm)

(tp,fp,fn,tn)=cm

accuracy=(tp+tn)/float(cm.sum())

print(accuracy)

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

*Source Code of app.py*

*import cv2*

*import pickle*

*from skimage import feature*

*from flask import Flask,request,render\_template*

*import os.path*

*app=Flask(\_\_name\_\_)*

*@app.route("/")*

*def about():*

*return render\_template("about.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=['GET','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)*

*return result*

*return None*

*if \_\_name\_\_=="\_\_main\_\_":*

*app.run(host='0.0.0.0', port=8000, debug=False)*

