Project Development Phase
Delivery of Sprint - 3

Date	01 November 2022
Team ID	PNT2022TMID18082
Project Name	Detect Parkinson's disease

Creating model and pickle section of flask

```
!pip install imutils
```

[3] Python

```
Collecting imutils
  Downloading imutils-0.5.4.tar.gz (17 kB)
Building wheels for collected packages: imutils
  Building wheel for imutils (setup.py): started
  Building wheel for imutils (setup.py): finished with status 'done'
  Created wheel for imutils: filename=imutils-0.5.4-py3-none-any.whl size=25862
sha256=4df05009385f6b58dc9eec4b4514713f4109ca05f581439f57b7d4032f2c29d0
  Stored in directory: c:\users\yogeshwaran\appdata\local\pip\cache\wheels\86\d7\0a\4923351ed1cec5d5e24c1eaf8905567b02a0343b24aa873df2
Successfully built imutils
Installing collected packages: imutils
Successfully installed imutils-0.5.4
```

```
pip install opencv-python
```

[7] Python

```
Collecting opencv-python
  Downloading opencv_python-4.6.0.66-cp36-abi3-win_amd64.whl (35.6 MB)
Requirement already satisfied: numpy>=1.14.5; python_version >= "3.7" in d:\studies\anaconda\lib\site-packages (from opencv-python) (1.18.1)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.6.0.66
Note: you may need to restart the kernel to use updated packages.
```

```
pip install pickle-mixin
```

[2] Python

```
Collecting pickle-mixin
  Downloading pickle-mixin-1.0.2.tar.gz (5.1 kB)
Building wheels for collected packages: pickle-mixin
  Building wheel for pickle-mixin (setup.py): started
  Building wheel for pickle-mixin (setup.py): finished with status 'done'
  Created wheel for pickle-mixin: filename=pickle_mixin-1.0.2-py3-none-any.whl size=6002
sha256=96ef57064bc440dee6d48fec8b329f13b1bb79fb1c516cbd5cfa44b44c10c0ca
  Stored in directory: c:\users\yogeshwaran\appdata\local\pip\cache\wheels\d0\70\0b\673e09a7ed429660d22352a1b117b4f616a8fc054bdd7eb157
Successfully built pickle-mixin
Installing collected packages: pickle-mixin
Successfully installed pickle-mixin-1.0.2
Note: you may need to restart the kernel to use updated packages.
```

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusionjnatrix
from skimage import feature
from imutils import build_montages
from imutils import paths
import numpy as np
import cv2
import os
import pickle
```

Python

```
trainingPath=r"D:\STUDIES\Project\Nalaya Thiran Dataset\dataset-20220920T082711Z-001\dataset\spiral\training"
testingPath=r"D:\STUDIES\Project\Nalaya Thiran Dataset\dataset-20220920T082711Z-001\dataset\spiral\testing"
```

```python
def loadsplit ( path ) :
    #grab the list of images in the input directory , then initialize
    # the list of data ( i.e. , images ) and class labels
    imagePaths = list ( paths.listimages ( path ) )
    data = [ ]
    labels = [ ]
   #loop over the image paths
    for imagePath in imagePaths :
        #extract the class label from the filename
        label = imagePath.split ( os.path.sep ) [ -2 ]
        print(label)
        #load the input image , convert it to grayscale , and resize
        # it to 200x200 pixels , ignoring aspect ratio
        image = cv2.imread ( imagePath )
        image = cv2.cvtColor ( image , cv2.COLORBGR2GRAY )
        image = cv2.resize ( image , ( 200 , 200 ) )
        # threshold the image such that the drawing appears as white
        # on a black background
        image = cv2.threshold(image , 0 , 255 , cv2.THRESHBINARYINV | cv2 . THRESHOTSU)[ 1 ]
        # quantify the image
        features = quantify_image(image)
        print(features)
        # update the data and labels lists , respectively
        data.append ( features )
        labels.append ( label )
    # return the data and labels
    return ( np.array ( data ) , np.array ( labels ) )
```

Python

```python
def quantifyimage(image):
    # compute the histogram
    features = feature.hog(image, orientations=9,
    pixels_per_cell=(10, 10), cells_per_block=(2, 2),
    transform_sqrt=True, block_norm="L1")
    # return the feature vector
    return features
```

```python
(X_train, y_train)=load_split(trainingPath)
(Xtest,ytest)=load_split(testingPath)
```

```
Output exceeds the size limit. Open the full output data in a text editor
healthy
[0. 0. 0. ... 0. 0. 0.]
healthy
[0. 0. 0. ... 0. 0. 0.]
iTSlirv
```

```
parkinson
[0. 0. 0. ... 0. 0. 0.]
parkinson
[0. 0. 0. ... 0. 0. 0.]
```

```
# encode the labels as integers
le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)
print(X_train.shape,y_train.shape)
```

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

```
X_train
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

```
# RANDOM FOREST
for i in idxs:
    image = cv2.imread(testPaths[i])
    print(image)
    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]
    print(image)
    features = quantify_image(image)
    print(features)
    preds = model.predict([features])
    label = le.inverse_transform(preds)[0]
    print(label)
    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)
```

```
Output exceeds the size limit. Open the full output data in a text editor
[[[239 239 239]
  [237 237 237]
  [237 237 237]
```

```
#KNN
for i in idxs:
    image = cv2.imread(testPaths[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 = knn.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)
```

```
#Decision Tree
for i in idxs:
    image = cv2.imread(testPaths[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 = dTree.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)
```