Image Pre-Processing and Import the necessary Libraries

In [1]:
```
pwd
```

Out[1]: '/home/wsuser/work'

In [2]:
```
import json
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='27mbnX1oZUsDIwLoyE6wBrNx0tptF6uQ-jxKF67vgvjH',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'detectionofparkinson39sdisease-donotdelete-pr-obdkgohtssns2d'
object_key = 'dataset.zip'

streaming_body_1 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']

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

In [3]:
```
from io import BytesIO
import zipfile
unzip = zipfile.ZipFile(BytesIO(streaming_body_1.read()),'r')
file_paths = unzip.namelist()
for path in file_paths:
    unzip.extract(path)
```

In [4]:
```
import os
filenames = os.listdir('/home/wsuser/work/dataset/training')
filenames = os.listdir('/home/wsuser/work/dataset/testing')
```

In [5]:
```
pip install opencv-python-headless
```

```
Collecting opencv-python-headless
  Downloading opencv_python_headless-4.6.0.66-cp36-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (48.3 MB)
     |████████████████████████████████| 48.3 MB 10.6 MB/s eta 0:00:01
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from opencv-python-headless) (1.20.3)
Installing collected packages: opencv-python-headless
Successfully installed opencv-python-headless-4.6.0.66
Note: you may need to restart the kernel to use updated packages.
```

In [6]:
```
pip install numpy
```

```
Requirement already satisfied: numpy in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.20.3)
Note: you may need to restart the kernel to use updated packages.
```

In [7]:
```
pip install imutils
```

```
Collecting imutils
  Downloading imutils-0.5.4.tar.gz (17 kB)
Building wheels for collected packages: imutils
  Building wheel for imutils (setup.py) ... done
  Created wheel for imutils: filename=imutils-0.5.4-py3-none-any.whl size=25860 sha256=214c945761152d8ab7ab521860207c7f495b8719c869c0ab5e1ec4d88c1ef062
  Stored in directory: /tmp/wsuser/.cache/pip/wheels/4b/a5/2d/4a070a801d3a3d93f033d3ee9728f470f514826e89952df3ea
Successfully built imutils
Installing collected packages: imutils
Successfully installed imutils-0.5.4
Note: you may need to restart the kernel to use updated packages.
```

In [8]:
```
pip install scikit-image
```

```
Requirement already satisfied: scikit-image in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (0.18.3)
Requirement already satisfied: numpy>=1.16.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from scikit-image) (1.20.3)
Requirement already satisfied: scipy>=1.0.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from scikit-image) (1.7.3)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from scikit-image) (3.5.0)
Requirement already satisfied: networkx>=2.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from scikit-image) (2.6.3)
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,>=4.3.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from scikit-image) (9.0.1)
Requirement already satisfied: imageio>=2.3.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from scikit-image) (2.9.0)
Requirement already satisfied: tifffile>=2019.7.26 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from scikit-image) (2021.7.2)
Requirement already satisfied: PyWavelets>=1.1.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from scikit-image) (1.1.1)
Requirement already satisfied: cycler>=0.10 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image) (0.11.0)
Requirement already satisfied: packaging>=20.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image) (21.3)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image) (4.25.0)
Requirement already satisfied: pyparsing>=2.2.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image) (3.0.4)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image) (1.3.1)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from python-dateutil>=2.7->matplotlib!=3.0.0,>=2.0.0->scikit-image) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

In [9]:
```
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
```

Path for train and test data

In [10]:
```
trainingpath=r"/home/wsuser/work/dataset/training"
testingpath=r"/home/wsuser/work/dataset/testing"
```

Quantifying Images

In [11]:
```
def quantify_image(image):
    features = feature.hog(image, orientations=9,
                           pixels_per_cell=(10, 10),
                           cells_per_block=(2, 2),
                           transform_sqrt=True,
                           block_norm="L1")
    return features
```

Loading Train Data and Test Data

In [12]:
```
def load_split(path):
    imagePaths = list(paths.list_images(path))
    data = []
    labels = []

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

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

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

        features = quantify_image(image)

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

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

Load the train and test data

In [13]:
```
print("[INFO] loading data...")
(X_train, y_train) = load_split(trainingpath)
(X_test, y_test) = load_split(testingpath)
```

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

Label Encoding

In [14]:
```
le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)
print(X_train.shape,y_train.shape)
```

```
(144, 12996) (144,)
```

Model Building and Training The Model

In [15]:
```
print("[INFO] training model")
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)
```

```
[INFO] training model
```

Out[15]:
```
RandomForestClassifier()
```

Testing The Model

In [16]:
```
testingpath=list(paths.list_images(testingpath))
idxs=np.arange(0,len(testingpath))
idxs=np.random.choice(idxs,size=(25,),replace=False)
images=[]
```

In [17]:
```
for i in idxs:
    image=cv2.imread(testingpath[i])
    output=image.copy()

    # load the input image,convert to grayscale and resize

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

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

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

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