#using svm model

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

import os

path = os.listdir('/kaggle/input/gcet-minor-brain-tumor/brain-tumor-detection-master/brain\_tumor/Training/')

classes = {'no\_tumor':0, 'pituitary\_tumor':1} #crreating a class,

# as machine learning works only on numerical data, so labelled as no tumor for class and pituar tumor for class 1

import cv2

X = [] # creating list with name x , in x we are to append all images

Y = [] # in y we append the targets of the particular images

for cls in classes: # first cls will be no tumor, then cls will be pitutary tumor

pth = '/kaggle/input/gcet-minor-brain-tumor/brain-tumor-detection-master/brain\_tumor/Training/'+cls

for j in os.listdir(pth): #to read all images, creating another nested loop

img = cv2.imread(pth+'/'+j, 0) # for reading image in greyscale or 2D, we are using 0bit pth

#resizing each and every image in 200\*200 pixels

img = cv2.resize(img, (200,200)) # must have same dimensions for all images in machine learning

X.append(img) # then append images in x

Y.append(classes[cls]) # append classed in y

X = np.array(X) # converting this into numpy array

Y = np.array(Y)

X\_updated = X.reshape(len(X), -1)

np.unique(Y) # to check the number of classes which is and

pd.Series(Y).value\_counts() #to check the number of samples in 0 class and 1 classnn... 0 points in no tumor

X.shape, X\_updated.shape # to check the shape ... 1222 are total number of samples,, 200, 2000 are dimesions

# this is three dimesion...1220, 200 and 200)

plt.imshow(X[0], cmap='gray') # plot any particular image

#as sklearn works on 2D only , converting 3d into 3d by flatening each and every image

# now coloums should be 200\*200=40000

#no of coloums is -1, takes all colums

X\_updated = X.reshape(len(X), -1)

X\_updated.shape

#now shape is 40000

# no of rows would be len(x) which is 1222

#from 1222 samples will we use 20% of data as testing remaing 80% for training

#using train, test and split

xtrain, xtest, ytrain, ytest = train\_test\_split(X\_updated, Y, random\_state=10,

test\_size=.20)

# using 977 samples for training and 245 for testing

xtrain.shape, xtest.shape

#using feature scaling , in order to bring all the features in the same scale

#as rgb value ranges from 0 to 255 ,the maximum pixel value is 255

#so simply divide these samples in to two 255

#so final values , the xtrain values would be from 0 to 1

print(xtrain.max(), xtrain.min())

print(xtest.max(), xtest.min())

xtrain = xtrain/255 # dividing into

xtest = xtest/255

print(xtrain.max(), xtrain.min())

print(xtest.max(), xtest.min())

"""feature selection: pca"""

from sklearn.decomposition import PCA

#principle component analysis, used for reducing the diminision

# suppose if we take a date

#

print(xtrain.shape, xtest.shape)

pca = PCA(.98)

pca\_train = pca.fit\_transform(xtrain)

pca\_test = pca.transform(xtest)

#pca\_train = xtrain

#pca\_test = xtest

# print(pca\_train.shape, pca\_test.shape)

print(pca.n\_components\_)

print(pca.n\_features\_)

# for training model we will compare two models logistic regression and svc

#svc is support vector machine

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

# to ignore warnings

import warnings

warnings.filterwarnings('ignore')

#for logistic regression we have set penality parameter c to 0.1

lg = LogisticRegression(C=0.1)

lg.fit(xtrain, ytrain)

sv = SVC()

sv.fit(xtrain, ytrain)

print("Training Score:", lg.score(xtrain, ytrain))

print("Testing Score:", lg.score(xtest, ytest))

print("Training Score:", sv.score(xtrain, ytrain))

print("Testing Score:", sv.score(xtest, ytest))

pred = sv.predict(xtest)

misclassified=np.where(ytest!=pred)

misclassified

print("Total Misclassified Samples: ",len(misclassified[0]))

print(pred[36],ytest[36])

dec = {0:'No Tumor', 1:'Positive Tumor'}

plt.figure(figsize=(12,8))

p = os.listdir('/kaggle/input/gcet-minor-brain-tumor/brain-tumor-detection-master/brain\_tumor/Testing/')

c=1

for i in os.listdir('/kaggle/input/gcet-minor-brain-tumor/brain-tumor-detection-master/brain\_tumor/Testing/no\_tumor/')[:9]:

plt.subplot(3,3,c)

img = cv2.imread('/kaggle/input/gcet-minor-brain-tumor/brain-tumor-detection-master/brain\_tumor/Testing/no\_tumor/'+i,0)

img1 = cv2.resize(img, (200,200))

img1 = img1.reshape(1,-1)/255

p = sv.predict(img1)

plt.title(dec[p[0]])

plt.imshow(img, cmap='gray')

plt.axis('off')

c+=1

plt.figure(figsize=(12,8))

p = os.listdir('/kaggle/input/gcet-minor-brain-tumor/brain-tumor-detection-master/brain\_tumor/Testing/')

c=1

for i in os.listdir('/kaggle/input/gcet-minor-brain-tumor/brain-tumor-detection-master/brain\_tumor/Testing/pituitary\_tumor/')[:16]:

plt.subplot(4,4,c)

img = cv2.imread('/kaggle/input/gcet-minor-brain-tumor/brain-tumor-detection-master/brain\_tumor/Testing/pituitary\_tumor/'+i,0)

img1 = cv2.resize(img, (200,200))

img1 = img1.reshape(1,-1)/255

p = sv.predict(img1)

plt.title(dec[p[0]])

plt.imshow(img, cmap='gray')

plt.axis('off')

c+=1