

Brain Tumor Detection Classification

Load Module

```
In [1]: 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
```

Prepare/Collect Data

```
In [2]: import os

path = os.listdir('F:/A DATA SCIENCE 2023/Brain Tumor Detection Classification/Training/')
classes = {'no_tumor':0, 'pituitary_tumor':1}
```

```
In [3]: import cv2
X = []
Y = []
for cls in classes:
    pth = 'F:/A DATA SCIENCE 2023/Brain Tumor Detection Classification/Training/'+cls
    for j in os.listdir(pth):
        img = cv2.imread(pth+'/'+j,0)
        img = cv2.resize(img, (200, 200))
        X.append(img)
        Y.append(classes[cls])
```

```
In [4]: np.unique(Y)
```

```
Out[4]: array([0, 1])
```

```
In [5]: X = np.array(X)
Y = np.array(Y)
```

```
In [6]: pd.Series(Y).value_counts()
```

```
Out[6]: 1    827
0    395
dtype: int64
```

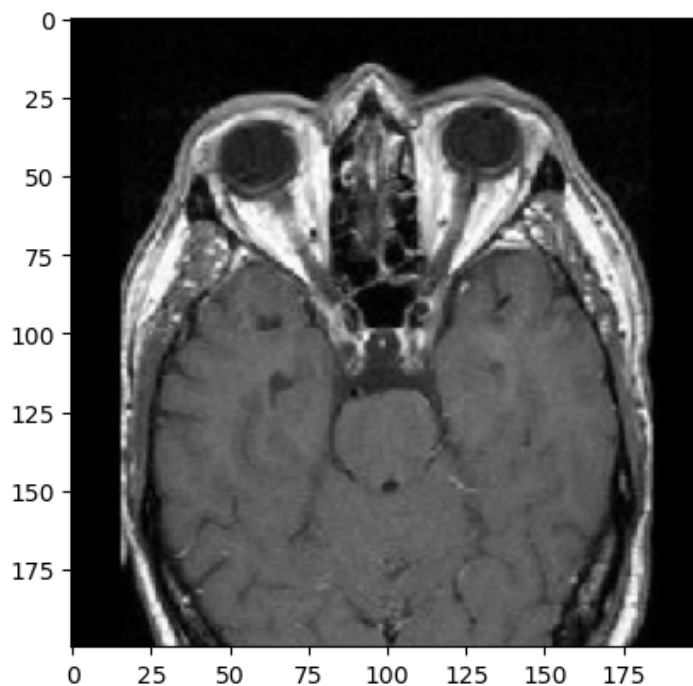
```
In [7]: X.shape
```

```
Out[7]: (1222, 200, 200)
```

Visualize Data

```
In [8]: plt.imshow(X[0], cmap='gray')
```

```
Out[8]: <matplotlib.image.AxesImage at 0x224fc080550>
```



Prepare Data

```
In [9]: X_updated = X.reshape(len(X), -1)
X_updated.shape
```

```
Out[9]: (1222, 40000)
```

Split Data

```
In [10]: xtrain, xtest, ytrain, ytest = train_test_split(X_updated, Y, random_state=10, test_size=.2)
```

```
In [11]: xtrain.shape, xtest.shape
```

```
Out[11]: ((977, 40000), (245, 40000))
```

Feature Scaling

```
In [12]: print(xtrain.max(), xtrain.min())
print(xtest.max(), xtest.min())
xtrain = xtrain/255
xtest = xtest/255
print(xtrain.max(), xtrain.min())
print(xtest.max(), xtest.min())
```

```
255 0
255 0
1.0 0.0
1.0 0.0
```

Feature Selection : PCA

```
In [13]: from sklearn.decomposition import PCA
```

```
In [14]: print(xtrain.shape, xtest.shape)
pca = PCA(.98)
pca_train = xtrain
pca_test = xtest
```

(977, 40000) (245, 40000)

Train Model

```
In [15]: from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
```

```
In [16]: import warnings
warnings.filterwarnings('ignore')

lg = LogisticRegression(C=0.1)
lg.fit(pca_train, ytrain)
```

```
Out[16]: LogisticRegression
LogisticRegression(C=0.1)
```

```
In [17]: sv = SVC()
sv.fit(pca_train, ytrain)
```

```
Out[17]: SVC
SVC()
```

Evaluation

```
In [18]: print("Training Score:", lg.score(pca_train, ytrain))
print("Testing Score:", lg.score(pca_test, ytest))
```

Training Score: 1.0
Testing Score: 0.9591836734693877

```
In [19]: print("Training Score:", sv.score(pca_train, ytrain))
print("Testing Score:", sv.score(pca_test, ytest))
```

Training Score: 0.9938587512794268
Testing Score: 0.963265306122449

Prediction

```
In [20]: pred = sv.predict(pca_test)
np.where(ytest != pred)
```

```
Out[20]: (array([ 36,  51,  68, 120, 212, 214, 220, 227, 239], dtype=int64),)
```

```
In [22]: pred[36]
```

```
Out[22]: 0
```

```
In [23]: ytest[36]
```

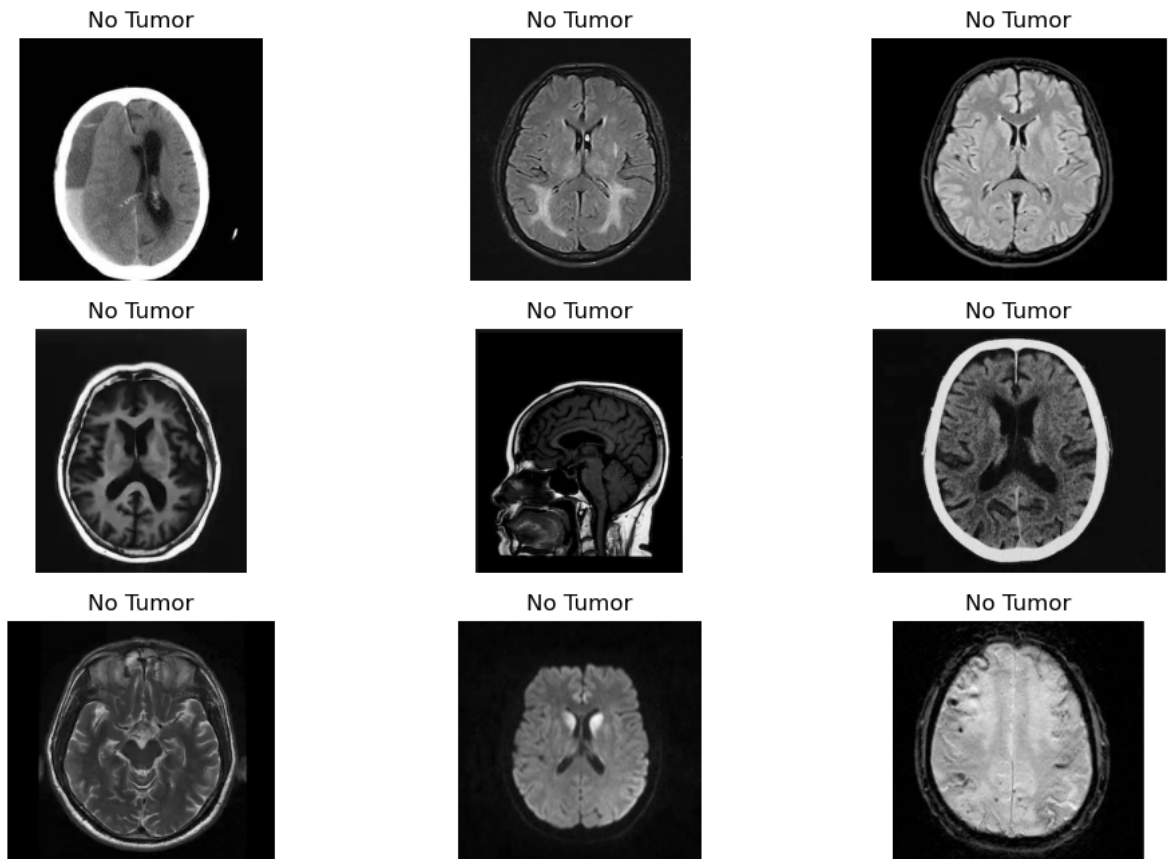
```
Out[23]: 1
```

Test Model

```
In [28]: dec = {0:'No Tumor', 1:'Positive Tumor'}
```

```
In [31]: plt.figure(figsize=(12,8))
p = os.listdir('F:/A DATA SCIENCE 2023/Brain Tumor Detection Classification/Testing/')
c=1
for i in os.listdir('F:/A DATA SCIENCE 2023/Brain Tumor Detection Classification/Testing/no'):
    plt.subplot(3,3,c)

    img = cv2.imread('F:/A DATA SCIENCE 2023/Brain Tumor Detection Classification/Testing/no/' + i)
    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
```



```

In [32]: plt.figure(figsize=(12,8))
p = os.listdir('F:/A DATA SCIENCE 2023/Brain Tumor Detection Classification/Testing/')
c=1
for i in os.listdir('F:/A DATA SCIENCE 2023/Brain Tumor Detection Classification/Testing/p:
plt.subplot(4,4,c)

img = cv2.imread('F:/A DATA SCIENCE 2023/Brain Tumor Detection Classification/Testing/p:
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

```

