Load Modules

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('brain_tumor/Training/')
classes = {'no_tumor':0, 'pituitary_tumor':1}
```

In [3]:
```
import cv2
X = []
Y = []
for cls in classes:
    pth = 'brain_tumor/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]:
```
X = np.array(X)
Y = np.array(Y)

X_updated = X.reshape(len(X), -1)
```

In [5]:
```
np.unique(Y)
```

Out[5]:
```
array([0, 1])
```

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

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

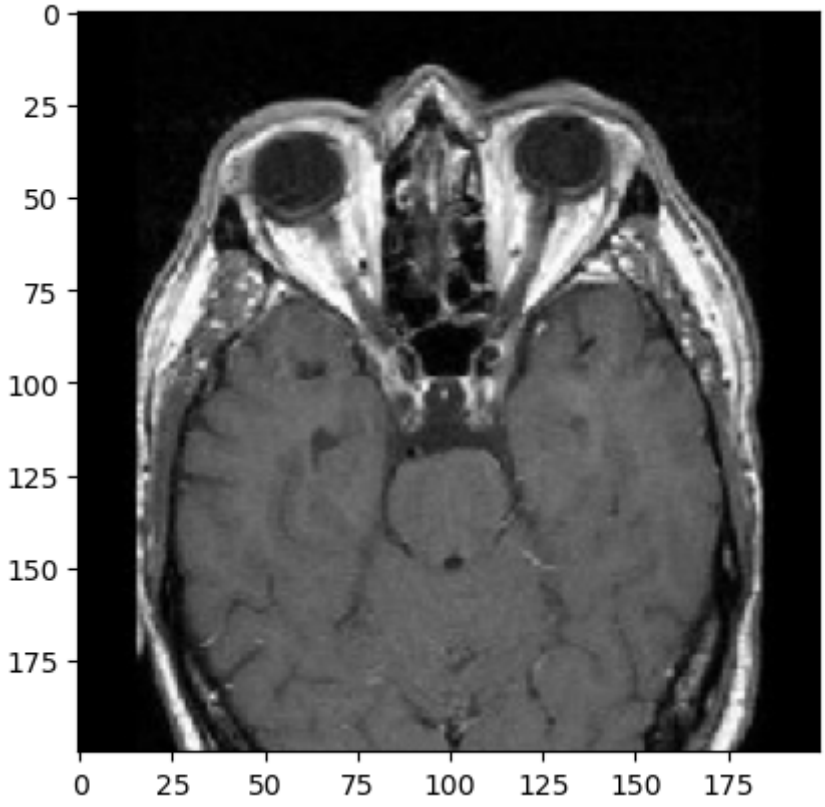
In [7]:
```
X.shape, X_updated.shape
```

Out[7]:
```
((1222, 200, 200), (1222, 40000))
```

Visualize data

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

Out[8]:
```
<matplotlib.image.AxesImage at 0x17d09313190>
```



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=.20)
```

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 = pca.fit_transform(xtrain)
# pca_test = pca.transform(xtest)
pca_train = xtrain
pca_test = xtest
```

```
(977, 40000) (245, 40000)
```

In [15]:
```
# print(pca_train.shape, pca_test.shape)
# print(pca.n_components_)
# print(pca.n_features_)
```

Train Model

In [16]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
```

In [17]:
```
import warnings
warnings.filterwarnings('ignore')

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

Out[17]:
```
LogisticRegression(C=0.1)
```

In [18]:
```
sv = SVC()
sv.fit(xtrain, ytrain)
```

Out[18]:
```
SVC()
```

Evaluation

In [19]:
```
print("Training Score:", lg.score(xtrain, ytrain))
print("Testing Score:", lg.score(xtest, ytest))
```

```
Training Score: 1.0
Testing Score: 0.9591836734693877
```

In [20]:
```
print("Training Score:", sv.score(xtrain, ytrain))
print("Testing Score:", sv.score(xtest, ytest))
```

```
Training Score: 0.9938587512794268
Testing Score: 0.963265306122449
```

Prediction

In [21]:
```
pred = sv.predict(xtest)
```

In [22]:
```
misclassified=np.where(ytest!=pred)
misclassified
```

Out[22]:
```
(array([ 36,  51,  68, 120, 212, 214, 220, 227, 239], dtype=int64),)
```

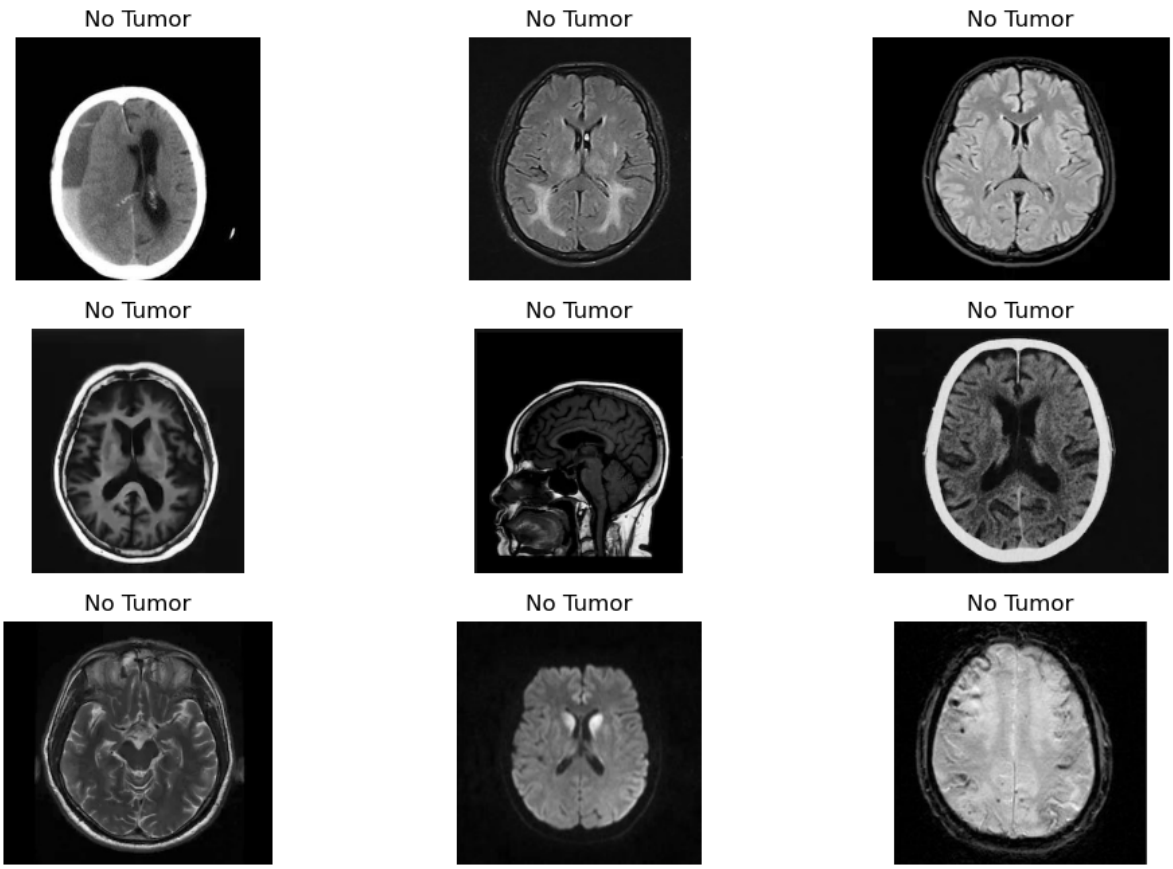
In [23]:
```
print("Total Misclassified Samples: ",len(misclassified[0]))
print(pred[36],ytest[36])
```

```
Total Misclassified Samples:  9
0 1
```

TEST MODEL

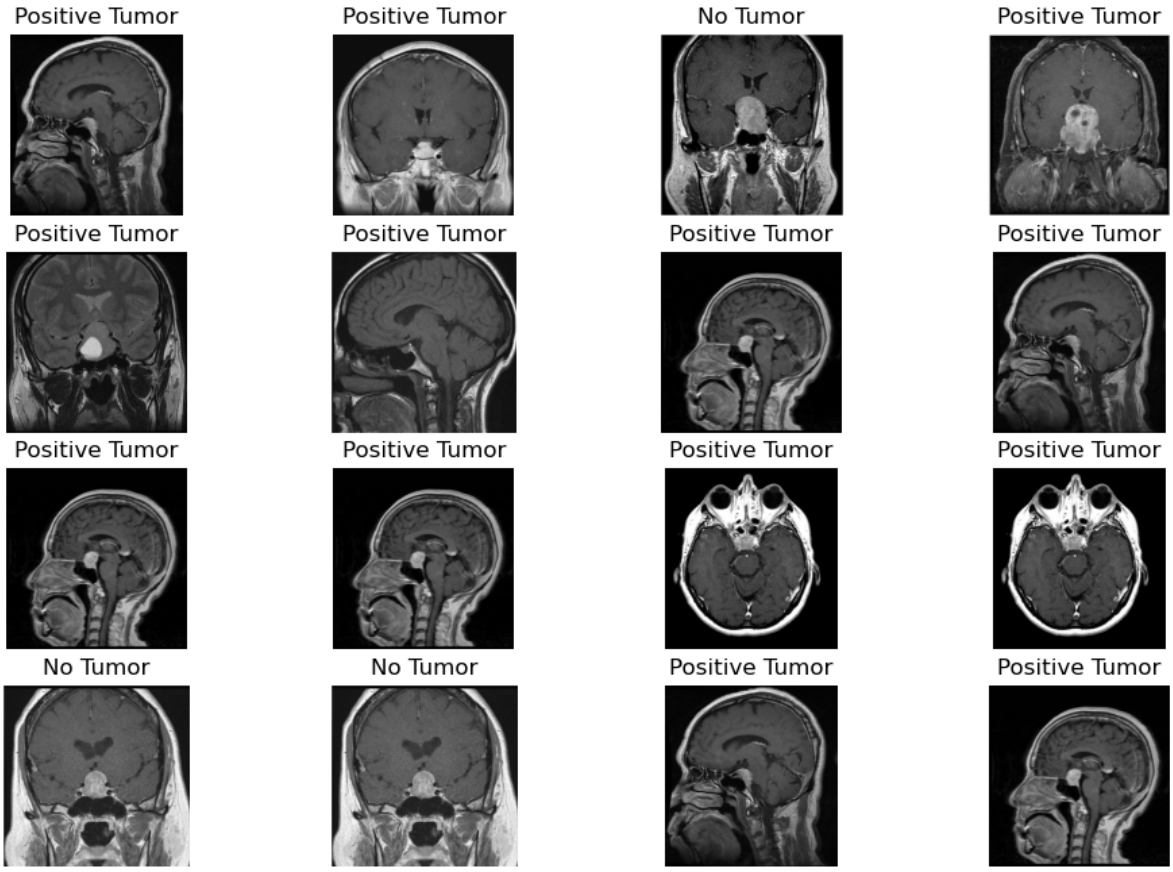
In [24]:
```
dec = {0:'No Tumor', 1:'Positive Tumor'}
```

In [25]:
```
plt.figure(figsize=(12,8))
p = os.listdir('brain_tumor/Testing/')
c=1
for i in os.listdir('brain_tumor/Testing/no_tumor/')[:9]:
    plt.subplot(3,3,c)

    img = cv2.imread('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
```



In [26]:
```
plt.figure(figsize=(12,8))
p = os.listdir('brain_tumor/Testing/')
c=1
for i in os.listdir('brain_tumor/Testing/pituitary_tumor/')[:16]:
    plt.subplot(4,4,c)

    img = cv2.imread('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
```



In [27]:
```
from sklearn.neighbors import KNeighborsClassifier

model1 = KNeighborsClassifier(n_neighbors=1)
model1.fit(xtrain, ytrain)
pred1 = model1.predict(xtest)

print(pred1)
```

```
[1 0 1 0 0 1 0 1 1 1 1 1 0 1 0 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 1 1 0 0 1 0 1
 0 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 1 1 0 0 1 1 1 1 0
 1 0 0 1 1 1 1 0 1 0 1 1 1 0 1 0 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 1 1 1 1 0 0 1 1 1 0 1 1 1 1 0
 1 0 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 0 1 0 0 1 1 1 0 1 1 0 1 1 1 1 0
 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 0 1
 0 1 1 1 1 1 1 1 0 1 1 0 1 1 0 0 1 0 1 1 1 1 1 1]
```

In [28]:
```
print(xtest)
```

```
[[0.         0.         0.         ... 0.         0.         0.        ]
 [0.00392157 0.00392157 0.00392157 ... 0.00392157 0.00392157 0.00392157]
 [0.         0.         0.         ... 0.         0.         0.        ]
 ...
 [0.         0.         0.         ... 0.00392157 0.         0.00392157]
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.         ... 0.         0.         0.        ]]
```

In [29]:
```
print(accuracy_score(ytest, pred1))
```

```
0.9673469387755103
```

In [ ]: