In [1]:
```
# Importing Libraries
```

In [2]:
```
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
```

In [3]:
```
# Loading and preparing the Data(Folder containing the images).
```

In [5]:
```
import os

path = os.listdir('archive/Training')
classes = {'no_tumor':0, 'pituitary_tumor':1}
```

In [7]:
```
import cv2
X = []
Y = []
for cls in classes:
    pth = 'archive/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 [8]:
```
X = np.array(X)
Y = np.array(Y)

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

In [9]:
```
# Analyzing the data..
```

In [10]:
```
np.unique(Y)
```

Out[10]:
```
array([0, 1])
```

In [11]:
```
pd.Series(Y).value_counts()
```

Out[11]:
```
1    827
0    395
dtype: int64
```

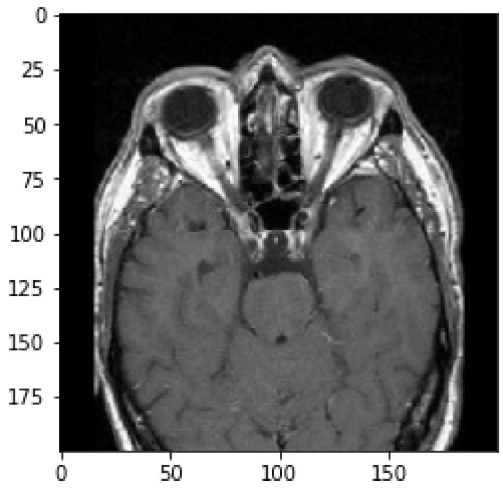
In [12]:
```
X.shape, X_updated.shape
```

Out[12]:
```
((1222, 200, 200), (1222, 40000))
```

In [13]:
```
# Visualizing the data
```

In [14]:
```
plt.imshow(X[0], cmap='gray')
```

Out[14]:
```
<matplotlib.image.AxesImage at 0x1d174ac22f0>
```



In [15]:
```
X_updated = X.reshape(len(X), -1)
X_updated.shape
```

Out[15]:
```
(1222, 40000)
```

In [16]:
```
#SPlitting the data

#In this step, we are going to split data in two parts (training and testing),
#so that we can train our model on training dataset and test its accuracy on unseen (t
```

In [17]:
```
xtrain, xtest, ytrain, ytest = train_test_split(X_updated, Y, random_state=10,
                                                test_size=.20)
```

In [18]:
```
xtrain.shape, xtest.shape
```

Out[18]:
```
((977, 40000), (245, 40000))
```

In [19]:
```
#Feature Scaling

#In this step, we are going to use minmax scaling technique to bring all the feature v
#In order to do so, we have divided the training data by its maximum value.
```

In [20]:
```
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
```

In [21]:
```
# Training the Model.

#As we have done with preprocessing part, it is time to train our model.
#I am going to train model using SVM (Support Vector Machine) and
#Logistic Regression algorithms and then we will compare the performance of these two
```

In [22]:
```
from sklearn.decomposition import PCA
```

In [23]:
```
print(xtrain.shape, xtest.shape)

pca = PCA(.98)
pca_train = xtrain
pca_test = xtest
```

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

In [24]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
```

In [25]:
```
import warnings
warnings.filterwarnings('ignore')

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

Out[25]:
```
LogisticRegression(C=0.1)
```

In [26]:
```
sv = SVC()
sv.fit(xtrain, ytrain)
```

Out[26]:
```
SVC()
```

In [27]:
```
# Evaluation
# we will compare the scores of above two models.
```

In [28]:
```
print("Training Score:", lg.score(xtrain, ytrain))
print("Testing Score:", lg.score(xtest, ytest))
```

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

In [29]:
```
print("Training Score:", sv.score(xtrain, ytrain))
print("Testing Score:", sv.score(xtest, ytest))
```

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

In [30]:
```
#As we can observe, SVM showed a great balance among training an testing score as comp
#So we can reach to the conclusion that it is ideal model for this particular dataset
```

In [31]:
```
# Predicting

#In this step we are going to predict test dataset. Afterwards, I have checked the tot
```

In [32]:
```
pred = sv.predict(xtest)
```

In [33]:
```
misclassified=np.where(ytest!=pred)
misclassified
```

Out[33]:
```
(array([ 36,  51,  68, 120, 212, 214, 220, 227, 239], dtype=int64),)
```

In [34]:
```
print("Total Misclassified Samples: ",len(misclassified[0]))
print(pred[36],ytest[36])
```

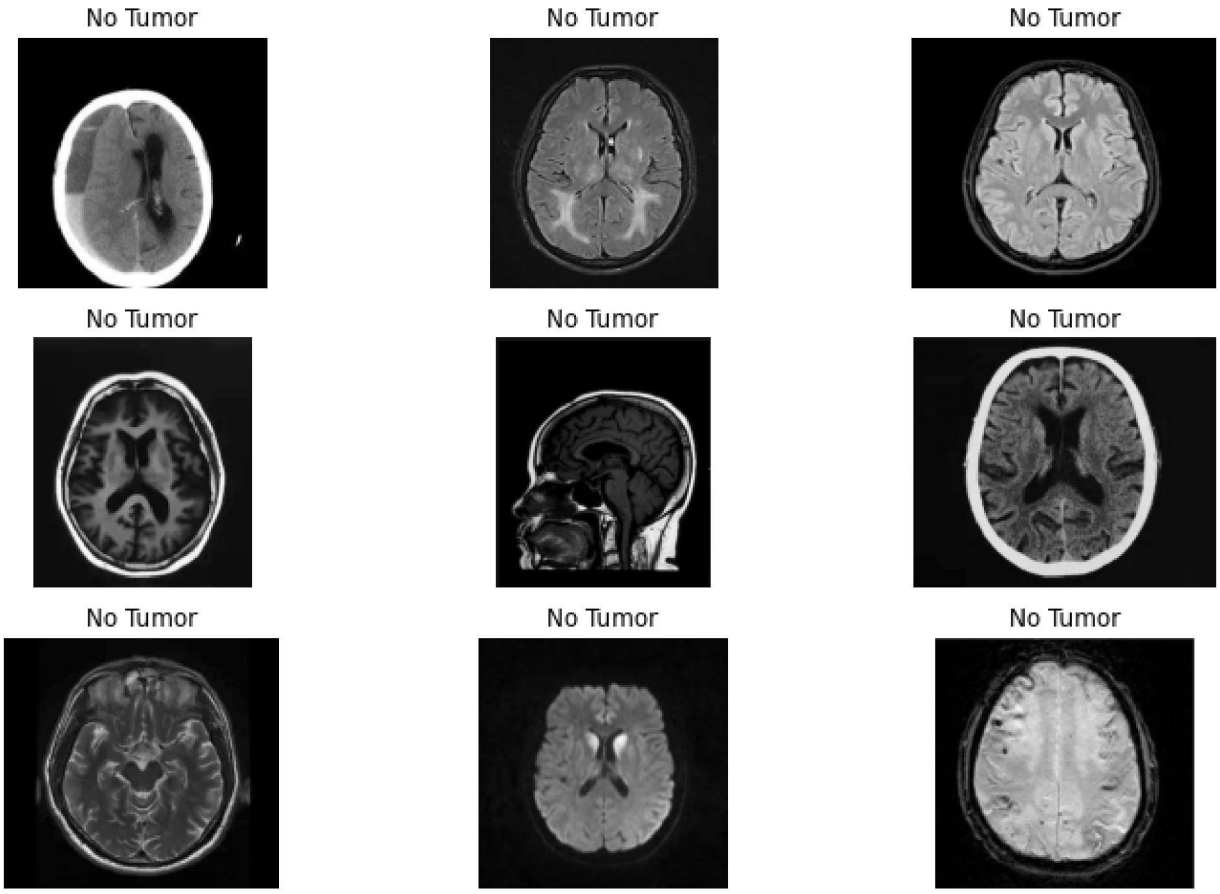
```
Total Misclassified Samples:  9
0 1
```

In [35]:
```
#Testing on The Dataset..
```

In [36]:
```
dec = {0:'No Tumor', 1:'Positive Tumor'}
```

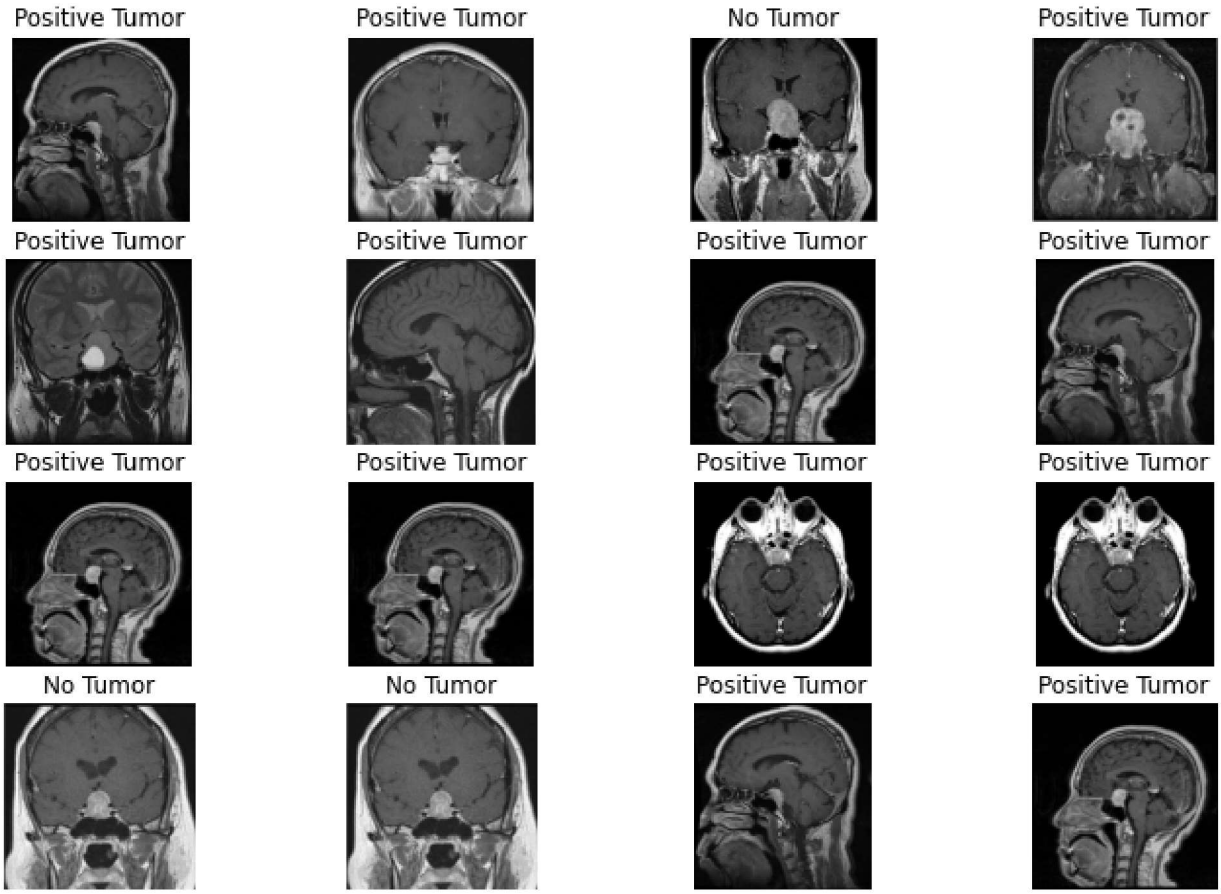
In [42]:
```
plt.figure(figsize=(12,8))
p = os.listdir('archive/Testing')
c=1
for i in os.listdir('archive/Testing/no_tumor/')[:9]:
    plt.subplot(3,3,c)

    img = cv2.imread('archive/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 [44]:
```
plt.figure(figsize=(12,8))
p = os.listdir('archive/Testing')
c=1
for i in os.listdir('archive/Testing/pituitary_tumor/')[:16]:
    plt.subplot(4,4,c)

    img = cv2.imread('archive/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
```


Positive Tumor
Positive Tumor
No Tumor
Positive Tumor
Positive Tumor
Positive Tumor
Positive Tumor
Positive Tumor
Positive Tumor
Positive Tumor
Positive Tumor
Positive Tumor
No Tumor
No Tumor
Positive Tumor
Positive Tumor

In [ ]: