In [1]:

```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.metrics import accuracy_score
import os
import cv2
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn import preprocessing
from sklearn import utils
# from sklearn.model_selection import GridSearchCV
```

In [2]:

```
path=os.listdir('C:/Users/admin/Desktop/Untitled Folder 1/archive (8)/brain_tumor_datas
et/')
classes={'no':0,'yes':1}
```

In [3]:

```
X=[]
Y=[]
for cls in classes:
    pth='C:/Users/admin/Desktop/Untitled Folder 1/archive (8)/brain_tumor_dataset/'+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    155
0     98
dtype: int64
```

In [7]:

```
X.shape#index 0 shows no. of images and rest shows the dimension of a single image
```

Out[7]:

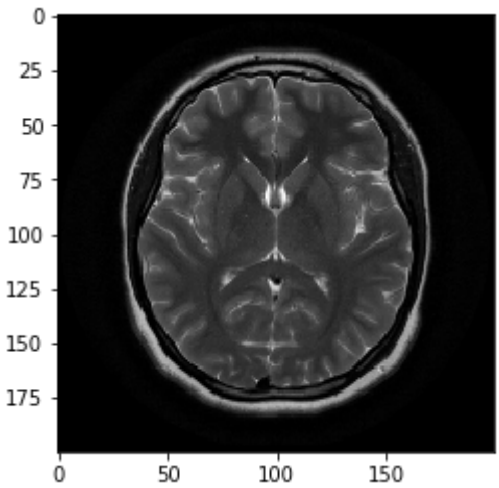
```
(253, 200, 200)
```

In [8]:

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

Out[8]:

```
<matplotlib.image.AxesImage at 0x1f3620a77c8>
```



In [9]:

```
#sklearn works on 2-D array, so reshape X
X_updated=X.reshape(len(X),-1)
X_updated.shape
```

Out[9]:

```
(253, 40000)
```

In [10]:

```
#now split the data for training in tts in sklearn
xtrain, xtest, ytrain,ytest=train_test_split(X_updated,Y,random_state=10,test_size=0.20
)
```

In [11]:

```
xtrain.shape,xtest.shape
```

Out[11]:

```
((202, 40000), (51, 40000))
```

In [12]:

```
#to bring all in same scale, used feature scalling
xtrain=xtrain/255
ytrain=ytrain/255
#as max pixel value will be 255 according to rgb standards
```

In [13]:

```
from sklearn.decomposition import PCA
```

In [14]:

```
print(xtrain.shape,xtest.shape)
```

```
(202, 40000) (51, 40000)
```

In [15]:

```
# pca=PCA(0.98)
# xtrain=pca.fit_transform(xtrain)
# xtest=pca.transform(xtest)
```

In [16]:

```
#Train Model
lab_enc = preprocessing.LabelEncoder()
encoded = lab_enc.fit_transform(ytrain)
```

In [17]:

```
# log_reg_grid = {
#     "C": np.logspace(-4, 4, 20),
#     "solver": ["liblinear","saga"]
# }

# lg = GridSearchCV(LogisticRegression(),
#                           param_grid=log_reg_grid,
#                           cv=5,
#                           verbose=True)
# lg.fit(xtrain,encoded)
lg = LogisticRegression(C=0.0012689610031679222,
                          solver='liblinear')
lg.fit(xtrain,encoded)
```

Out[17]:

```
LogisticRegression(C=0.0012689610031679222, class_weight=None, dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=100, multi_class='warn', n_jobs=None, penalty
='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verb
ose=0,
                   warm_start=False)
```

In [18]:

```
# sv=SVC()
# sv.fit(xtrain,encoded)
```

In [19]:

```
print(lg.score(xtest,ytest))
# print(sv.score(xtest,ytest))
```

```
0.6666666666666666
```

In [20]:

```
from sklearn.model_selection import cross_val_score
cv_acc = cross_val_score(lg,
                         X_updated,
                         Y,
                         cv=5,
                         scoring="accuracy")

pred=lg.predict(xtest)
np.where(ytest!=pred)#to check results which are wrongly predicted
```

Out[20]:

```
(array([ 2, 10, 14, 15, 16, 17, 20, 22, 31, 33, 34, 36, 41, 42, 43, 46, 4
9],
       dtype=int64),)
```

In [21]:

```
print(pred[10])
print(ytest[10])
```

```
1
0
```

In [22]:

```
#Test Model
dec={0:'no tumor',1:'yes tumor'}
```

In [23]:

```
# import ipywidgets as widgets
# widgets.interact(test_result,t=[1,3,6,9,16],title=True);
```

In [24]:

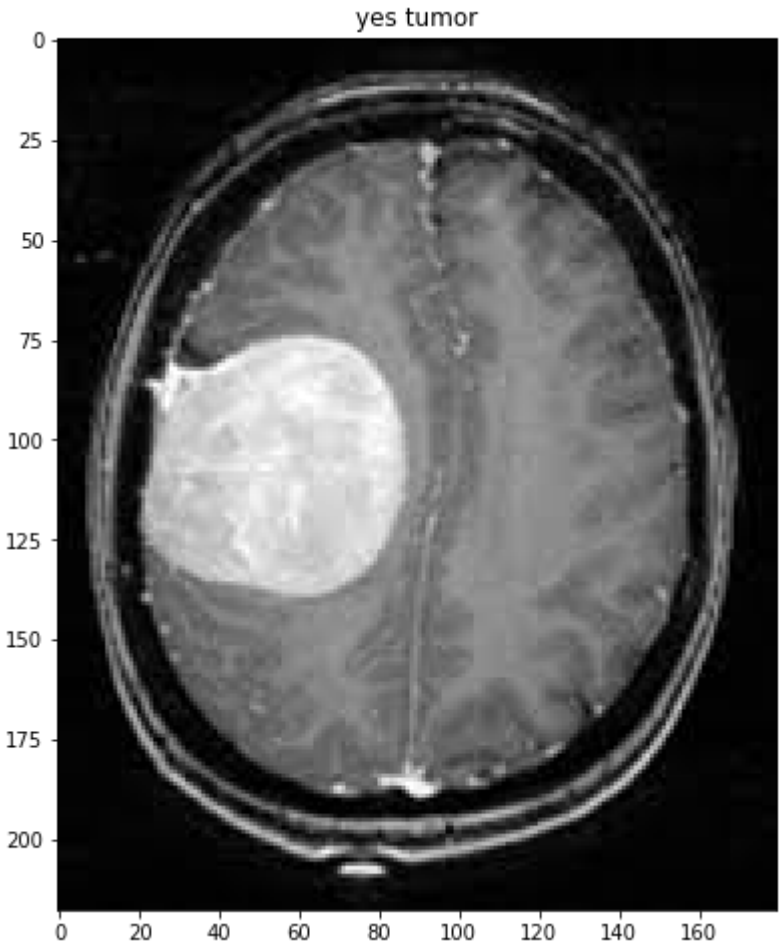
```
plt.figure(figsize=(12,8))
# def test_result(t):
p=os.listdir('C:/Users/admin/Desktop/Untitled Folder 1/archive (8)/brain_tumor_dataset/')
c=1
# for i in os.listdir('C:/Users/admin/Desktop/Untitled Folder 1/archive (8)/brain_tumor_dataset/yes/')[:9]:
#     plt.subplot(5,5,c)
#     img=cv2.imread('C:/Users/admin/Desktop/Untitled Folder 1/archive (8)/brain_tumor_dataset/yes/'+i,0)
#     img1=cv2.resize(img,(200,200))
#     img1=img1.reshape(1,-1)/255
#     p=lg.predict(img1)
#     plt.title(dec[p[0]])
#     plt.imshow(img,cmap='gray')
#     plt.axis('off')
#     print(i)
#     c+=1



img=cv2.imread('C:/Users/admin/Desktop/Untitled Folder 1/archive (8)/brain_tumor_dataset/yes/Y1.jpg',0)
img1=cv2.resize(img,(200,200))
img1=img1.reshape(1,-1)/255
p=lg.predict(img1)
plt.title(dec[p[0]])
plt.imshow(img,cmap='gray')
```

Out[24]:

```
<matplotlib.image.AxesImage at 0x1f362400a08>
```



In [31]:

```
import pickle
model_name=open("tumor_detection.pkl","wb")
pickle.dump(lg, model_name)
```

In [35]:

```
f = open('tumor_detection.pkl',"rb")
mydict = pickle.load(f)
```

In [37]:

```
mydict.score(xtest,ytest)
```

Out[37]:

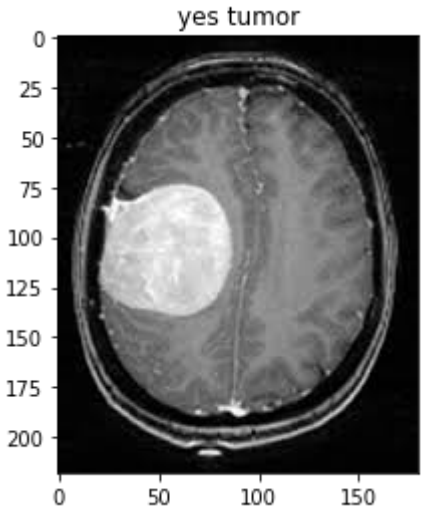
```
0.6666666666666666
```

In [38]:

```
img=cv2.imread('C:/Users/admin/Desktop/Untitled Folder 1/archive (8)/brain_tumor_dataset/yes/Y1.jpg',0)
img1=cv2.resize(img,(200,200))
img1=img1.reshape(1,-1)/255
p=mydict.predict(img1)
plt.title(dec[p[0]])
plt.imshow(img,cmap='gray')
```

Out[38]:

```
<matplotlib.image.AxesImage at 0x1f362117348>
```



In [ ]:

In [ ]: