In [108…
```
print('trained model : https://drive.google.com/file/d/1-p68ThAd087xti0PzqmcpgbeN
print('dataset link : https://www.kaggle.com/datasets/babaraliuser/brain-mri-imag
```

```
trained model : https://drive.google.com/file/d/1-p68ThAd087xti0PzqmcpgbeNSpL6rx
S/view?usp=drive_link
dataset link : https://www.kaggle.com/datasets/babaraliuser/brain-mri-images-dat
aset
```

In [1]:
```
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import pywt
```

In [2]:
```
#listing directories
directories = []
for dr in os.scandir('mri/'):
    directories.append(dr.path)
directories
```

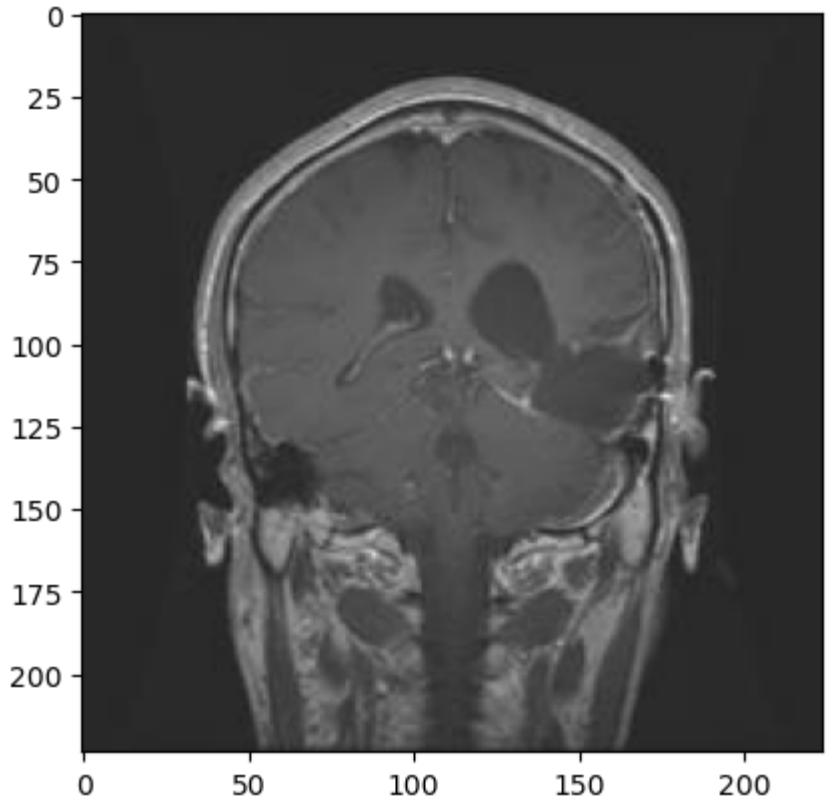
Out[2]:
```
['mri/glioma_tumor',
 'mri/meningioma_tumor',
 'mri/normal',
 'mri/pituitary_tumor']
```

In [3]:
```
#classifiying different conditions
class_dic={}
counter = 0
for dr in os.scandir('mri/'):
    class_dic[dr.path.split('/')[-1]]=counter
    counter+=1
class_dic
```

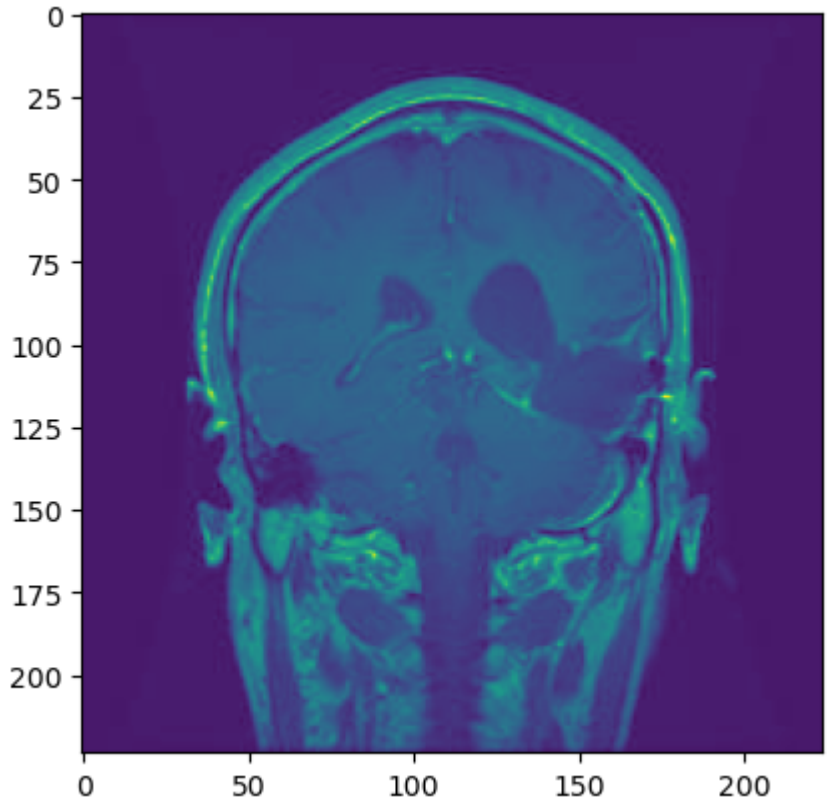
Out[3]:
```
{'glioma_tumor': 0, 'meningioma_tumor': 1, 'normal': 2, 'pituitary_tumor': 3}
```

In [ ]:
```
#genering list of files for every class
classes_files_list={}
for dr in os.scandir('mri/'):
    classes_files_list[dr.path.split('/')[-1]]=[]
    for file in os.scandir(dr.path+'/'):
        classes_files_list[dr.path.split('/')[-1]].append(file.path)
classes_files_list
```

In [5]:
```
img = cv2.imread('mri/glioma_tumor/G_165_BR_.jpg')
plt.imshow(img)
plt.show()
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
plt.imshow(gray)
```



Out[5]: <matplotlib.image.AxesImage at 0x86d0365ad0>



In [53]:
```
def w_transform(img  , mode='haar',level = 1):
    gray = img
    gray = cv2.cvtColor(gray,cv2.COLOR_BGR2GRAY)
    gray = np.float32(gray)
    gray /=255

    #coefficients
    coef = pywt.wavedec2(gray,mode,level = level)
    #listing
    coef_l = list(coef)
    coef_l[0] *=0

    #rebuilding
    coef_l=pywt.waverec2(coef_l,mode)
    coef_l *=255
    coef_l = np.uint8(coef_l)
    return coef_l
```

In [80]:
```
#generating features
x = []
y=[]
for dr in directories:
    sample_taking=1
    for file in classes_files_list[dr.split('/')[-1]]:
        img = cv2.imread(file)
        resize_img = cv2.resize(img,(32,32))
        w_img = w_transform(img,'db1',5)
        resize_w = cv2.resize(w_img,(32,32))
        combined = np.vstack((resize_img.reshape(32*32*3,1),resize_w.reshape(32*3
        x.append(combined)
        y.append(dr.split('/')[-1])
        sample_taking =sample_taking+1
        #if sample_taking==4000:
         #   break;
```

In [81]:
```
len(x)
```

Out[81]:
```
21672
```

In [82]:
```
x = np.array(x).reshape(len(x),4096).astype(float)
x.shape
```

Out[82]:
```
(21672, 4096)
```

In [24]:
```
len(y)
```

Out[24]:
```
1996
```

In [83]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=0)
```

In [71]:
```
from sklearn import svm
from sklearn.linear_model import LogisticRegression
```

In [37]:
```
svm = svm.SVC(gamma='auto',probability=True)
svm.fit(x_train,y_train)
svm.score(x_test,y_test)
```

Out[37]:
```
0.24048096192384769
```

In [84]:
```
lr = LogisticRegression()
lr.fit(x_train,y_train)
```

```
C:\Users\Sico\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressio
n
  n_iter_i = _check_optimize_result(
```

Out[84]:
```
▾ LogisticRegression

LogisticRegression()
```

In [85]:
```
lr.score(x_test,y_test)
```

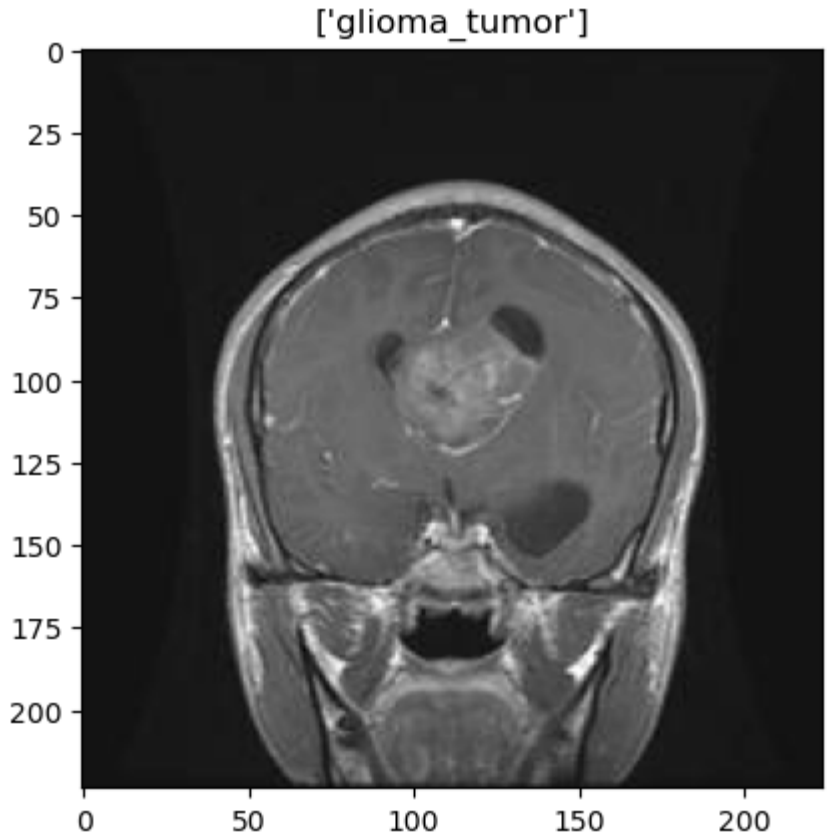
Out[85]:
```
0.7454780361757106
```

In [86]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
rf.score(x_test,y_test)
```

Out[86]:
```
0.8207825765965301
```

In [98]:
```
img = cv2.imread('mri/glioma_tumor/G_100_BR_.jpg')
scalled_raw_img = cv2.resize(img, (32, 32))
img_har = w_transform(img,'db1',5)
scalled_img_har = cv2.resize(img_har, (32, 32))
combined_img = np.vstack((scalled_raw_img.reshape(32*32*3,1),scalled_img_har.resh
xx = np.array(combined_img).reshape(1,4096).astype(float)
value = rf.predict(xx)
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
print("RandomForest classifier performed best with score of 82.this might be impr
plt.imshow(gray,cmap='gray')
plt.title(str(value))
plt.show()
```

```
RandomForest classifier performed best with score of 82.this might be improved
```



In [100…

```
import pickle
pickle.dump(rf,open('randomForest.pickle','wb'))
```