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

```
In [3]: import cv2
        X = []
        Y = []
        for cls in classes:
            pth = '../data/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 [5]: X = np.array(X)
        Y = np.array(Y)
```

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

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

```
Out[6]: 1    827
        0    395
        dtype: int64
```
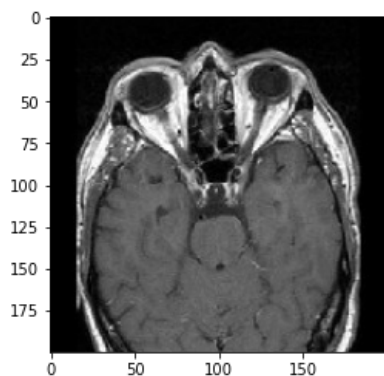
```
In [7]: X.shape
```

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

Visualize data

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

```
Out[8]: <matplotlib.image.AxesImage at 0x1d2466d4400>
```


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

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 [14]: from sklearn.decomposition import PCA
```

```
In [15]: 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 [ ]: # 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 [20]: import warnings
         warnings.filterwarnings('ignore')

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

Out[20]: LogisticRegression(C=0.1)
```

```
In [21]: sv = SVC()
         sv.fit(pca_train, ytrain)

Out[21]: SVC()
```

Evaluation

```
In [22]: 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 [23]: 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 [24]: pred = sv.predict(pca_test)
         np.where(ytest!=pred)

Out[24]: (array([ 36,  51,  68, 120, 212, 214, 220, 227, 239], dtype=int64),)
```

```
In [27]: pred[36]

Out[27]: 0
```
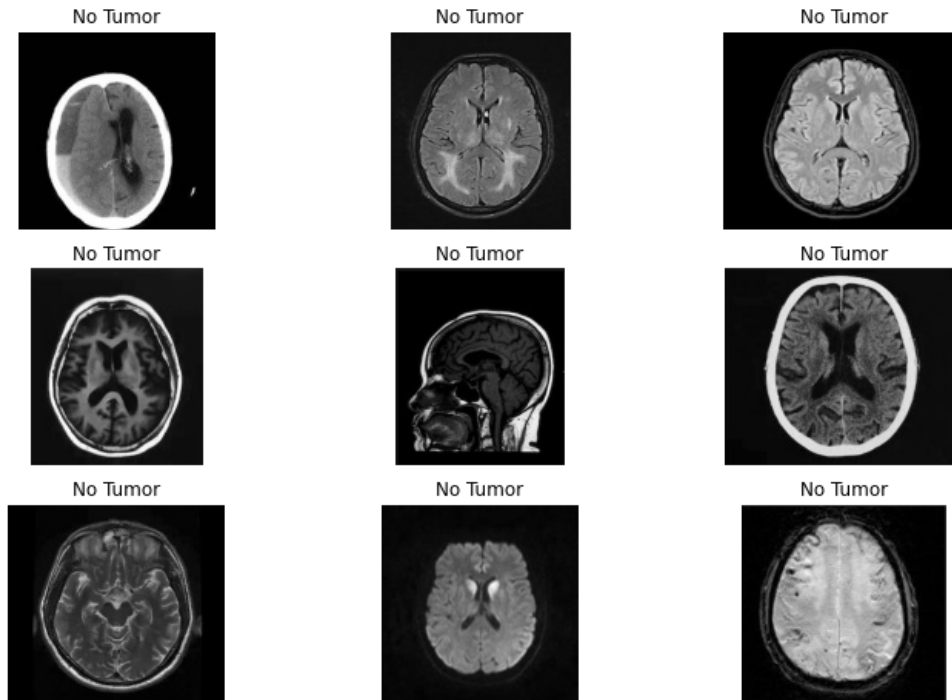
```
In [28]: ytest[36]

Out[28]: 1
```

TEST MODEL

```
In [29]:  dec = {0:'No Tumor', 1:'Positive Tumor'}
```

```
In [30]:  plt.figure(figsize=(12,8))
          p = os.listdir('../data/brain_tumor/Testing/')
          c=1
          for i in os.listdir('../data/brain_tumor/Testing/no_tumor/')[:9]:
              plt.subplot(3,3,c)

              img = cv2.imread('../data/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
```
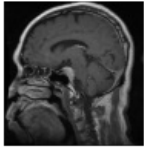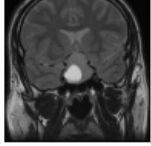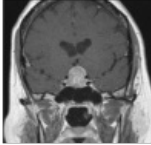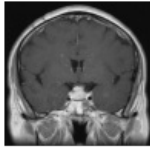
No Tumor · No Tumor · No Tumor
No Tumor · No Tumor · No Tumor
No Tumor · No Tumor · No Tumor


```
In [31]:  plt.figure(figsize=(12,8))
          p = os.listdir('../data/brain_tumor/Testing/')
          c=1
          for i in os.listdir('../data/brain_tumor/Testing/pituitary_tumor/')[:16]:
              plt.subplot(4,4,c)

              img = cv2.imread('../data/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
```
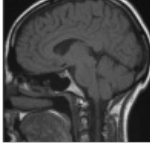
Positive Tumor | Positive Tumor | No Tumor | Positive Tumor
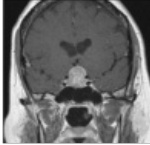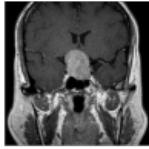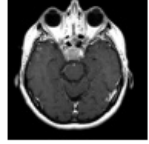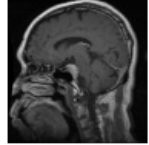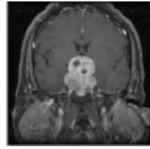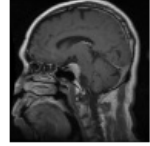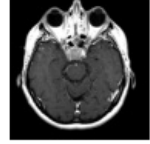Positive Tumor | Positive Tumor | Positive Tumor | Positive Tumor
Positive Tumor | Positive Tumor | Positive Tumor | Positive Tumor
No Tumor | No Tumor | Positive Tumor | Positive Tumor

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js