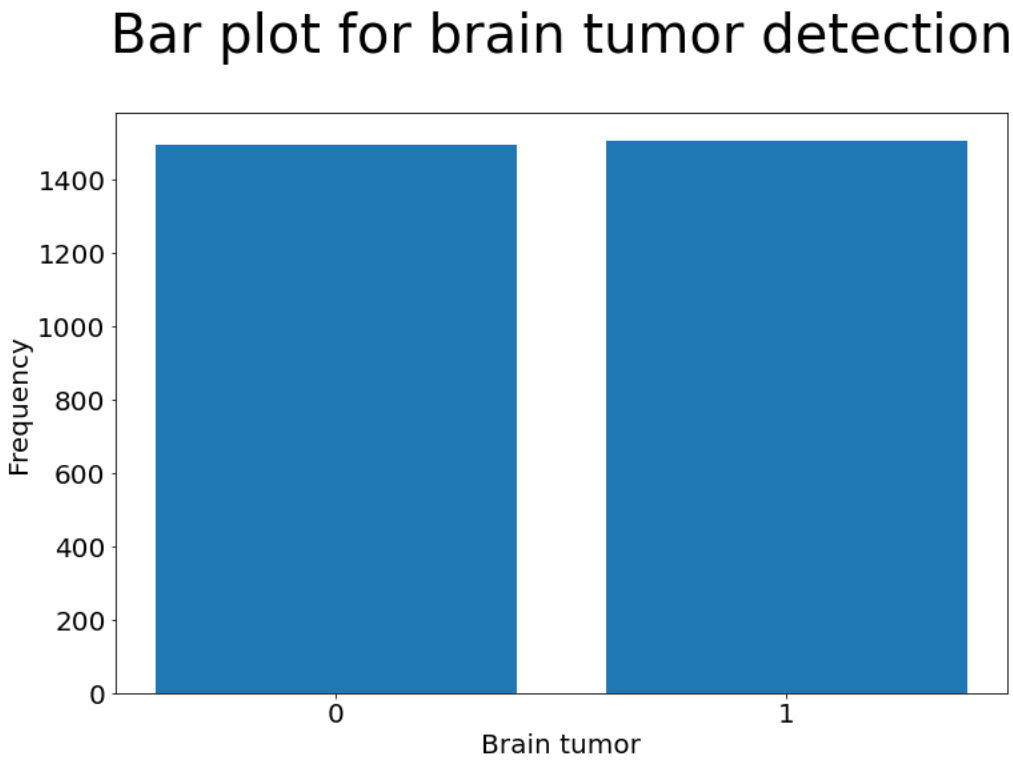
Brain Tumor Detection

Team members: Shalaka Gaidhani, Goutham Thota, Tejaswi Chintapalli

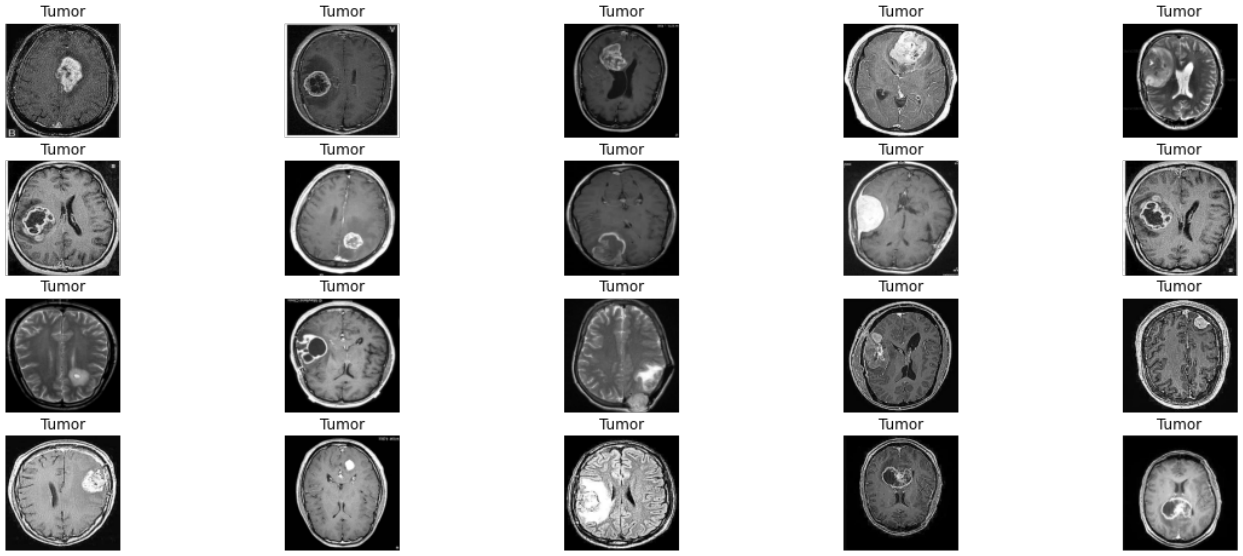
Source: https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-detection

Data set name: Br35H :: Brain Tumor Detection 2020
Number of images: 3000 images
Number of classes: 2 classes (yes and no)
Number of images based on classes: 1500 no, 1500 yes

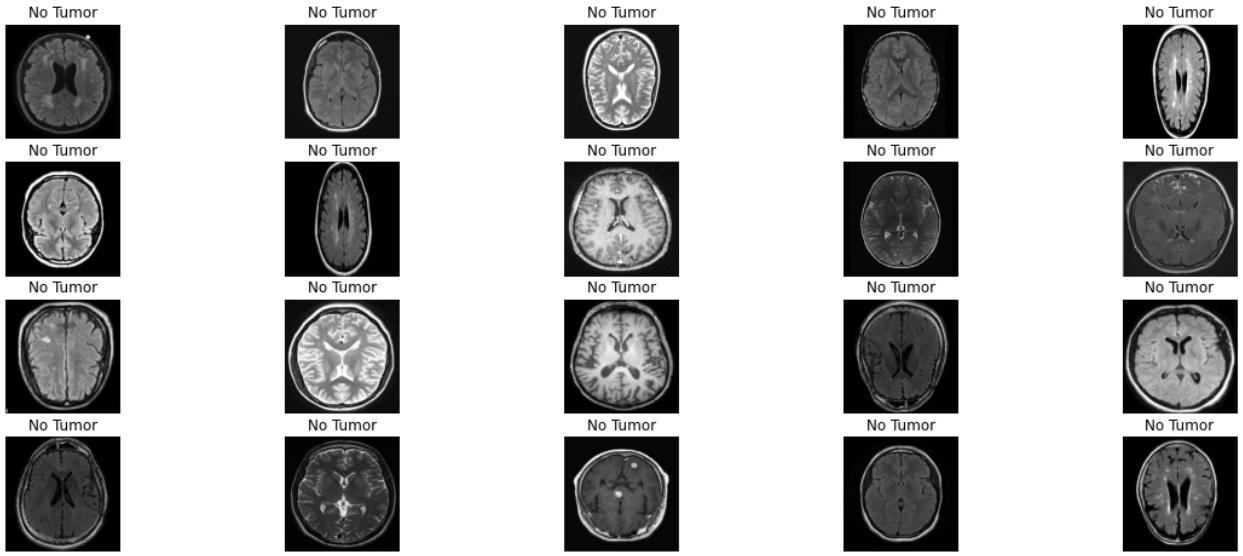
Distribution: Balanced data set



The images with tumor:



The images with no tumor:



Problem Statement: Detect a tumor in the brain from an MRI image utilizing python.

Implementation:

1. Load the dataset
    - Download the data from Kaggle
    - Load the data with help of OS module
    - Read the images using cv2

2. Convert the images into an understandable format
    - Convert the images to grayscale
    - Grayscale image helps to convert the image pixels between 0 and 255
    - 0 being black and 255 being white

3. Then preprocess images and fix the noisy data

    - Each image has a different size
    - Hence, convert them to a fixed size (256,256)
    - Then normalize the image by using min-max normalization
    - Now, the images are transformed in the range from 0 to 1

4. Data splitting:
    - Split the data into 70% train, and 30% test sets
    - Name of the train test sets are X_train, X_test, y_train, y_test

5. Apply Machine learning and Deep learning:
    - SVM
    - CNN
    - ADABoost

1) <u>SVM:</u>

    - SVM takes two-dimensional array input
    - The images in this dataset are three-dimensional arrays because they are grayscale images with a height, width, and a single channel
    - To use the SVM classifier, flatten the images into one-dimensional arrays first
    - For dimensionality reduction, PCA helps to transform 3-d to 2-d image
    - Run the classifier
    - Fit the model to the trained data
    - Make the predictions on the test data
    - Evaluate the performance on the test data and predicted outcomes
    - The considered performance measures are accuracy and recall
        - Accuracy is ~92%
        - Recall is ~92%
    - Hence, SVM give correct predicts 92% of times to detect brain tumor
    - Thus, this model best fits the data

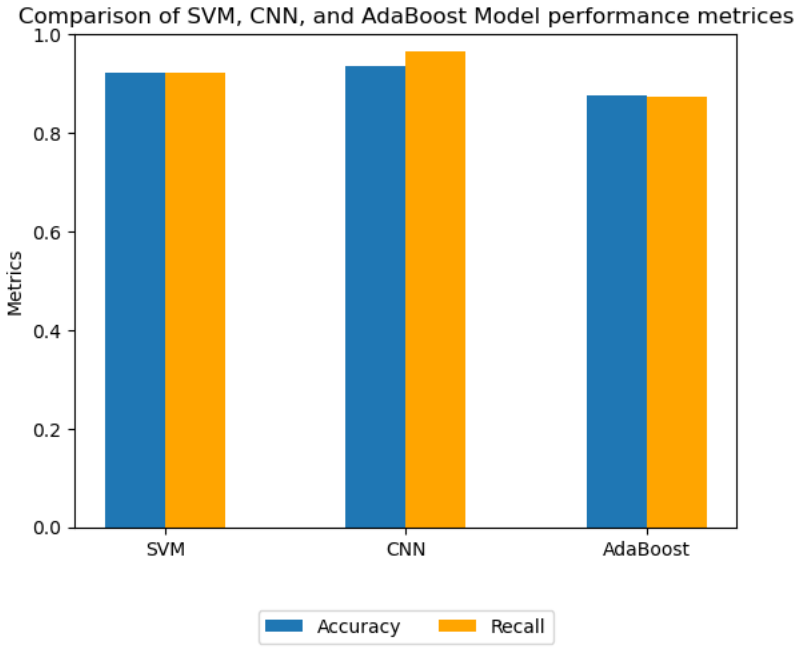
2) CNN:

- Define the model architecture using TensorFlow's Keras API to classify brain MRI images as either having a tumor or not
- compile the model with hyperparameters
- Keras has an inbuilt function to flatten out the images on its own
- The activation function taken in the hidden layer is ReLU
- The model optimizer is 'adam', loss function is 'binary_crossentropy' and metrics is accuracy
- To convert & expand the data, tf.constant function is used to transform the data to a TensorFlow constant tensor, and tf.expand_dims function is used to add an additional dimension to the data to match the input shape of the CNN model architecture
- The model is then trained on the training data for 2 epochs with a batch size of 32 and validate on the test data
- Fit the model to the trained data
- Make the predictions on the test data
- Evaluate the performance on the test data and predicted outcomes
- The considered performance measures are accuracy and recall
- Accuracy is ~95%
- Recall is ~94%
- This model's accuracy is higher than SVM
- Consequently, this model performs well on this dataset

3) ADABoost:

- Adaboost requires 2D/1D images whereas, our data has 3D images hence, we will have to flatten these images like what we did in SVM using PCA for dimensionality reduction
- AdaBoost has base_esmitors as a DecisionTree hence, we initialized a DT classifier with maximum depth 1
- We then train AdaBoost classifier using 50 decision trees as weak classifiers, learning rate of 0.5, and the SAMME.R algorithm for multi-class exponential loss function on transformed training data X_train_pca and y_train
- Fit the model to the trained data
- Make the predictions on the test data
- Evaluate the performance on the test data and predicted outcomes
- The considered performance measures are accuracy and recall
    - Accuracy is ~87%
    - Recall is ~87%
- This model's accuracy is comparatively lower than SVM and CNN
- However, this model performs well.

Performance:
Convolutional Neural networks is the best model that fits our data to predict brain Tumor.




```
table = [
    ["SVM", svm_acc, svm_rec],
    ["CNN", cnn_acc, cnn_rec],
    ["AdaBoost", ada_test_sc, ada_rec]
]

headers = ["Model", "Accuracy", "Recall"]

print(pd.DataFrame(table, headers=headers))
```

```
Model       Accuracy    Recall
--------  ----------  --------
SVM         0.922222  0.921924
CNN         0.935556  0.963719
AdaBoost    0.874444  0.873927
```