**Brain Tumor Detection**

Casey Patel

1. **Introduction**

What is Brain Tumor?

A brain tumor is characterized by the growth of abnormal cells within the brain. The brain's complex anatomy encompasses various regions, each responsible for distinct nervous system functions. Brain tumors can emerge in any part of the brain or skull, including the protective lining, the skull base, the brainstem, the sinuses, the nasal cavity, and numerous other areas.

How does it impact human lives?

* A brain tumor is considered one of the most aggressive diseases, affecting both children and adults. Brain tumors account for 85 to 90 percent of all primary Central Nervous System (CNS) tumors.
* Each year, approximately 11,700 people are diagnosed with a brain tumor.
* The 5-year survival rate for individuals with a cancerous brain or CNS tumor is about 34 percent for men and 36 percent for women.
* Brain tumors are classified into several types, including benign tumors, malignant tumors, and pituitary tumors.
* Proper treatment, planning, and accurate diagnostics are essential to improving patients' life expectancy.

1. **Problem Statement**

The detection and classification of brain tumors, which account for a significant portion of primary Central Nervous System (CNS) tumors, present a critical challenge due to their complexity and the substantial volume of MRI image data generated. Current manual examination methods are prone to errors, underscoring the need for advanced diagnostic tools to enhance accuracy and improve patient outcomes. The goal of this project is to analyze MRI images of the brain to accurately determine the presence of tumors and minimize human error in diagnosis.

1. **Stakeholder**

The client for a brain tumor detection problem could be medical researchers, hospitals and clinics, regulatory bodies, healthcare technology companies and healthcare providers.

1. **Dataset**

The dataset is obtained from Kaggle called [Brain Tumor Detection.](https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-detection/data) The dataset contains 3 folders: yes, no and pred which contains 3060 Brain MRI Images.

|  |  |
| --- | --- |
| Folder | Description |
| yes | The folder yes contains 1500 Brain MRI Images that are tumorous |
| no | The folder no contains 1500 Brain MRI Images that are non-tumorous |

1. **Data Wrangling:**

To being with I preprocessed MRI brain images for training a Convolutional Neural Network (CNN) model to detect the presence of brain tumors. The main goals of the preprocessing steps were to ensure that the images are in a consistent format and size, and to label them appropriately for use in the model.

* **Standardization:** Resized all images to the same dimensions (224x224) which ensures that the input to the CNN is consistent. This standardization is critical for the model to learn effectively.
* **Labeling:** Properly labeled the images with 0 for no tumor and 1 for tumor which provides the necessary supervised learning framework. The CNN model can use these labels to learn and distinguish between the two classes.
* **Data Preparation:** Converted images to numpy arrays which is an essential preprocessing step. CNN models require numerical data, and numpy arrays which provide a suitable format for efficient computation and model training.
* **Error Handling:** Included checks to handle cases where an image might not be read properly. This error handling ensured that the process is robust and can continue even if some images are missing or corrupted.

1. **Exploratory Data Analysis (EDA):**

Next step is to prepare the preprocessed MRI brain images and their corresponding labels for training a Convolutional Neural Network (CNN) model. It involves shuffling the dataset, creating data generators with augmentation, and generating batches of data for training and validation.

Utilized **ImageDataGenerator** to create two data generators: **train\_datagen** for training and **test\_datagen** for validation.

* **Normalization:** Rescaled the pixel values by 1/255 which normalizes the image data to the range [0, 1]. This is crucial for ensuring that the inputs to the CNN are on a similar scale, which helps in faster convergence during training.
* **Shuffling:** Shuffled the dataset to ensure that the training process does not follow any specific order, which can help in preventing overfitting and making the model more robust.
* **Batch Generation:** Generated data in batches which is efficient and necessary for training large datasets that cannot be loaded into memory all at once. It allowed the model to be trained incrementally, which is especially useful for deep learning models like CNNs.

Then, I splited the preprocessed dataset and corresponding labels into training and testing sets using an 80-20 split ratio. By specifying a **random\_state,** the split is made reproducible. The training set **(X\_train, y\_train)** was used to train the CNN model, while the test set **(X\_test, y\_test)** was used to evaluate the model's performance.

Normalized the image data by scaling pixel values to the range [0, 1] which ensures that the CNN model receives inputs that are consistent in scale and numerically stable.

converted binary class labels into one-hot encoded vectors, which is essential for training a CNN model with categorical cross-entropy loss.

1. **Machine Learning Model:**

The model is compiled with the categorical crossentropy loss function and the Adam optimizer, which are standard choices for classification tasks. The architecture and layers used in this model help to automatically learn and extract relevant features from brain tumor images, making it suitable for detecting the presence of tumors with reduced human error.

Convolutional neural networks (CNNs)

**Model Initialization:** I began by initializing sequential model.

**Input Layer:** Then, the input layer is added with the specified input shape, which is (224, 224, 3). This shape represents 224x224 pixels with 3 color channels (RGB).

**First Convolutional Layer:** Next, Added a 2D convolutional layer with 32 filters of size 3x3 and ReLU activation function. Convolutional layers are used to detect local patterns.

**First Max Pooling Layer:** Added a max pooling layer with a pool size of 2x2 which reduces the spatial dimensions (height and width) of the input volume.

**Second Convolutional Layer:** Added another 2D convolutional layer with 32 filters of size 3x3, ReLU activation function, and He uniform kernel initializer. This layer also helps in detecting local patterns with a different initialization technique for weights.

**Second Max Pooling Layer:** Added a max pooling layer with a pool size of 2x2, similar to the first max pooling layer, to further reduce the spatial dimensions.

**Third Convolutional Layer:** Added a 2D convolutional layer with 64 filters of size 3x3, ReLU activation function, and He uniform kernel initializer. This layer captures more complex features.

**Third Max Pooling Layer:** Added a max pooling layer with a pool size of 2x2, further reducing the spatial dimensions.

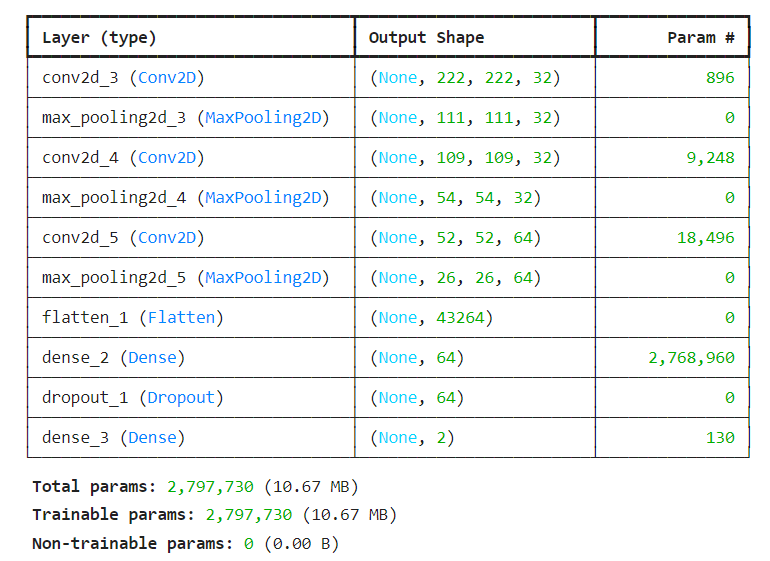
**Flatten Layer:** Flattened the 2D matrix from the convolutional and pooling layers into a 1D vector, making it suitable for the fully connected (dense) layers.

**Dense Layer:** Added a fully connected layer with 64 neurons and ReLU activation function. Dense layers are used to perform high-level reasoning in the network.

**Dropout Layer:** Added a dropout layer with a 50% dropout rate to reduce overfitting by randomly setting half of the input units to 0 at each update during training time.

**Output Layer:** Added the output layer with 2 neurons (one for each class in binary classification) and softmax activation function. The softmax function outputs a probability distribution over the two classes.

Lastly, Compiled the model by specifying the loss function **(categorical\_crossentropy),** optimizer **(adam)**, and evaluation metric (accuracy).

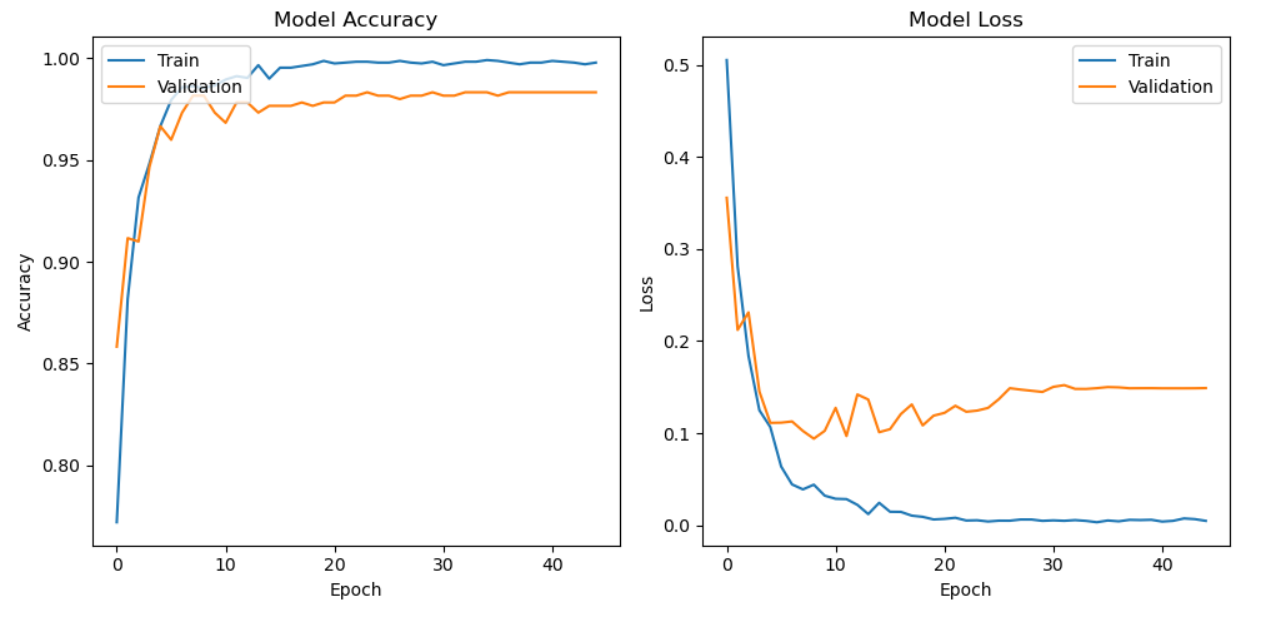
Next, employed **EarlyStopping** to prevent overfitting by stopping the training when no further improvement in the training loss is observed, thereby saving computational resources and time.

Also, utilized **ReduceLROnPlateau** dynamically to adjust the learning rate when the validation loss plateaus, ensuring more refined weight updates and potentially leading to better model performance.

These mechanisms together help in creating a robust and efficient CNN model for brain tumor detection.

**Visualizing accuracy and loss**

Next**,** visualized the training history of a Convolutional Neural Network (CNN) model by plotting the accuracy and loss for both the training and validation datasets over the epochs.

 The high accuracy of the model, around 98%, indicates a robust and well-trained AI system that can significantly improve brain tumor detection.

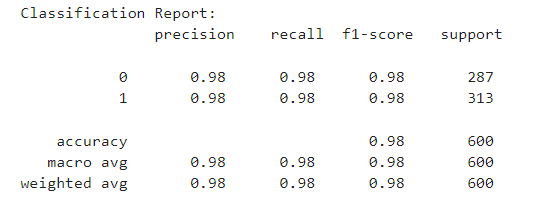
**Model Prediction:**

Generated predictions for the test dataset X\_test. Since the output layer of the model uses the softmax activation function, the predictions are probabilities for each class (in this case, two classes: no tumor and tumor).

Next, I converted these probabilities to class labels (0 or 1) by taking the index of the maximum probability value along the second axis (which corresponds to the class dimension).

By predicting the class labels for the test data and comparing them to the true labels, the model achieves a high validation accuracy of 98.33%. This high accuracy indicates that the model is well-trained and effective at distinguishing between MRI images of brain tumors and non-tumor cases, thus reinforcing the reliability of the model for medical diagnostics.

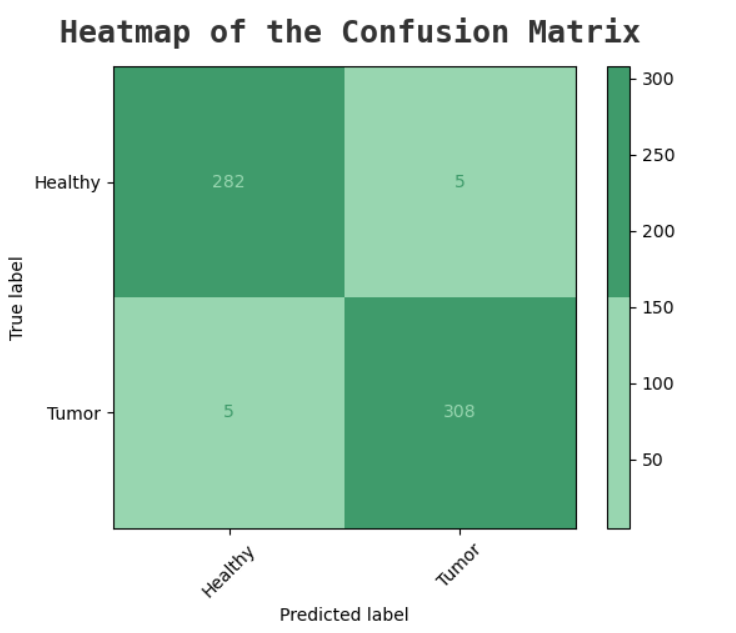
**Classification Report:**



The classification report indicated that the model is highly effective at distinguishing between tumor and no tumor cases in MRI images, with an accuracy of 98%. Both classes have high precision, recall, and F1-scores, demonstrated the model's strong predictive performance and reliability. This suggests that the model is well-trained and can be used effectively for medical diagnostics to assist in identifying brain tumors.

**Heatmap of the Confusion Matrix:**

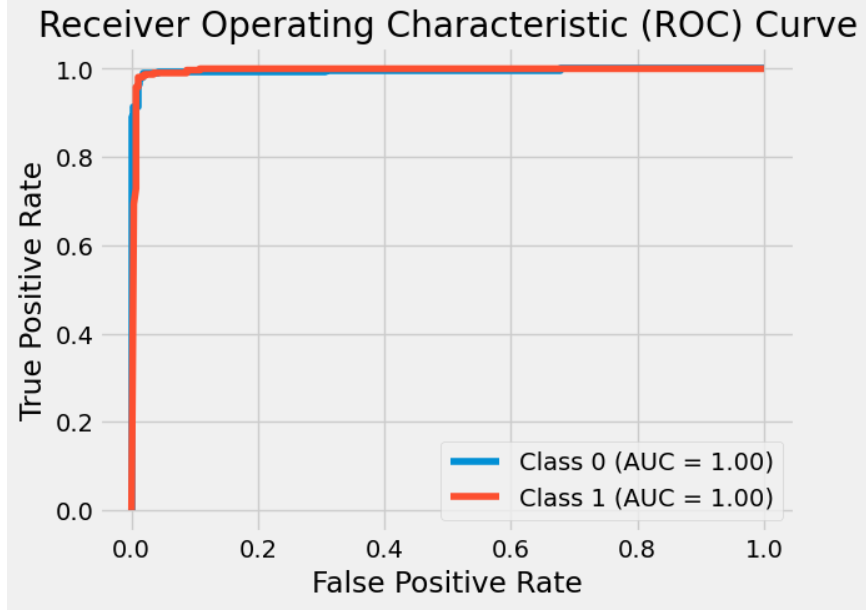
visualized the performance of a machine learning model(CNNs) using a confusion matrix plotted as a heatmap.



As we can see in the above figure 98% of the time model predicts correct image.

**Receiver Operating Characteristic (ROC) curves**

Next computed and plotted the Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC) score for a multi-class classification model.



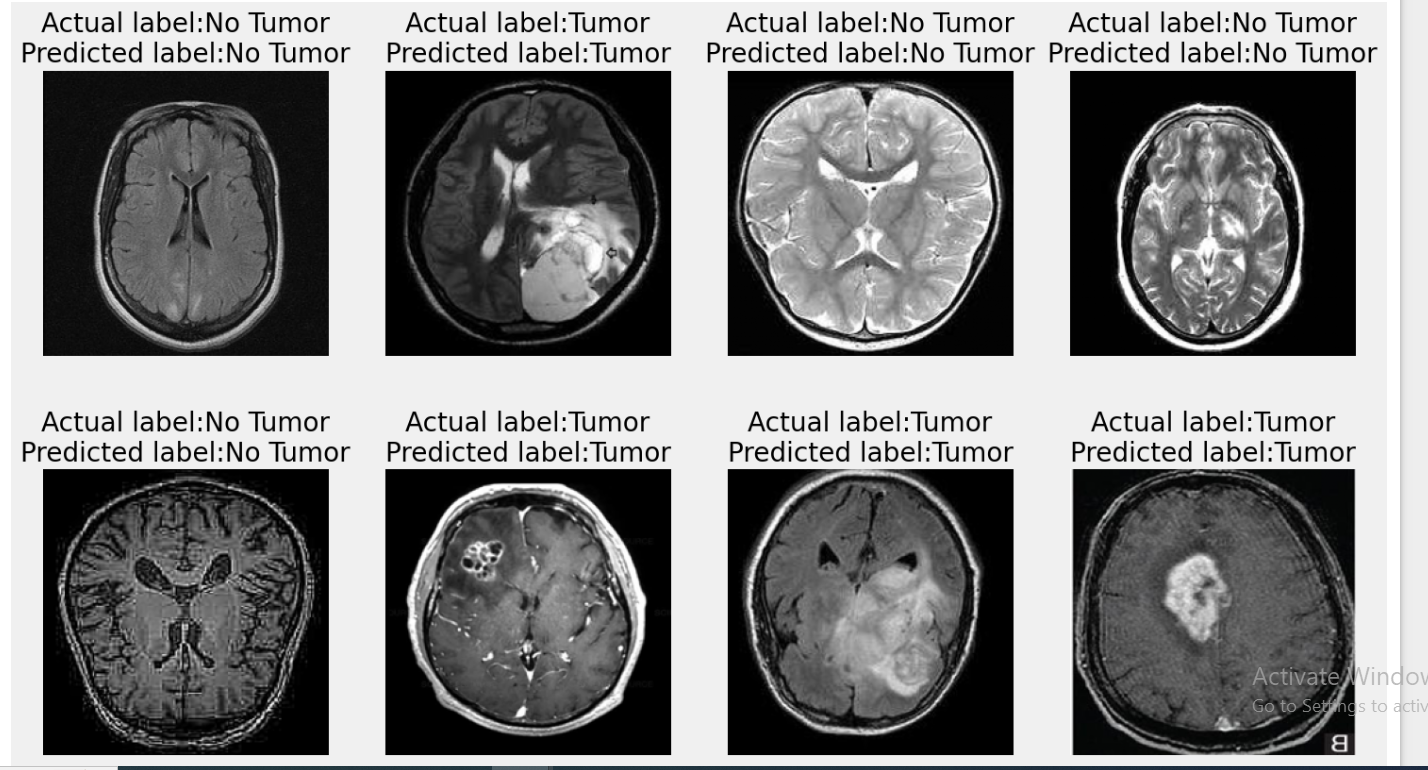
The ROC curve is curving near the point (0,1), it signifies that the model has excellent discriminatory power, making it a robust and accurate classifier for the given task.

**Visualizing test images**

I created a function which iterates through the first 16 test images (X\_test) of a tumor classification model. For each image, it displays the image itself, along with its actual and predicted labels as titles. This visualization helps in understanding how well the model is performing by comparing its predictions against the ground truth labels.

In order to create a function defined the class labels corresponding to the categories "No Tumor" (class 0) and "Tumor" (class 1). Then, createed a subplot in a 4x4 grid (16 subplots in total) and displayed the i-th test image X\_test[i].

The images are shown below:



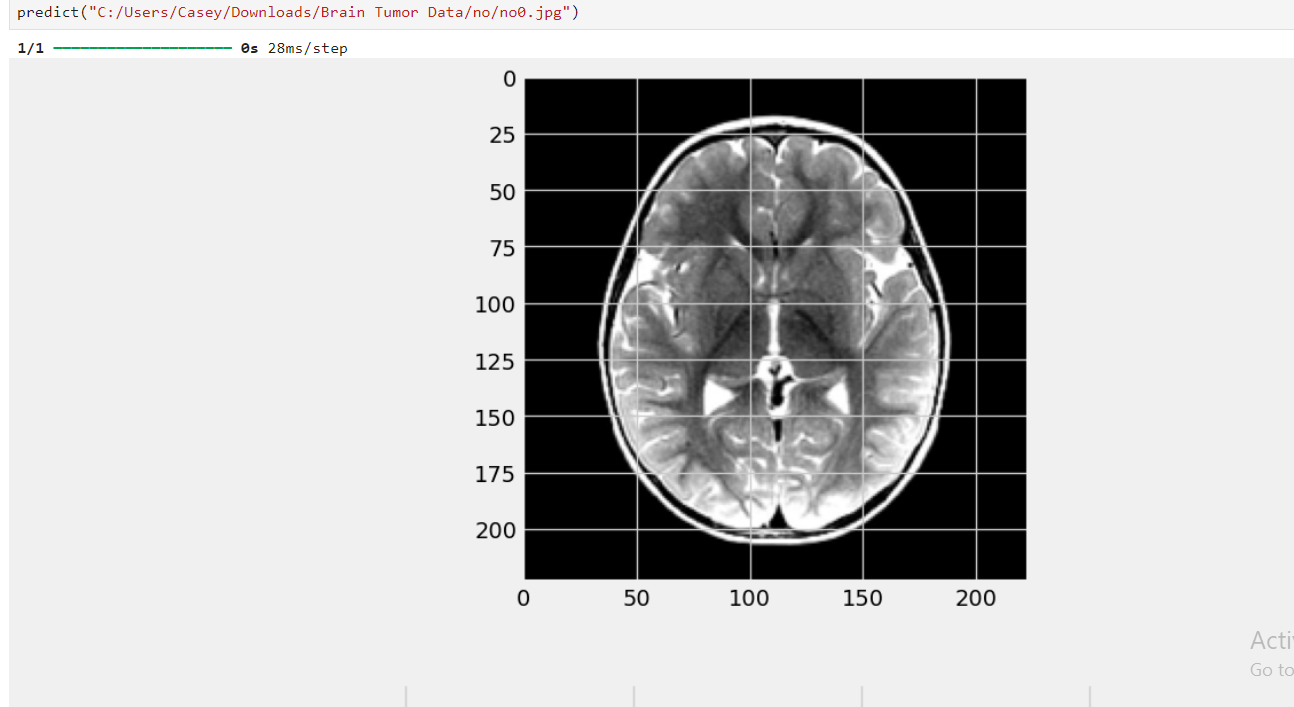
**“Predict” Function**

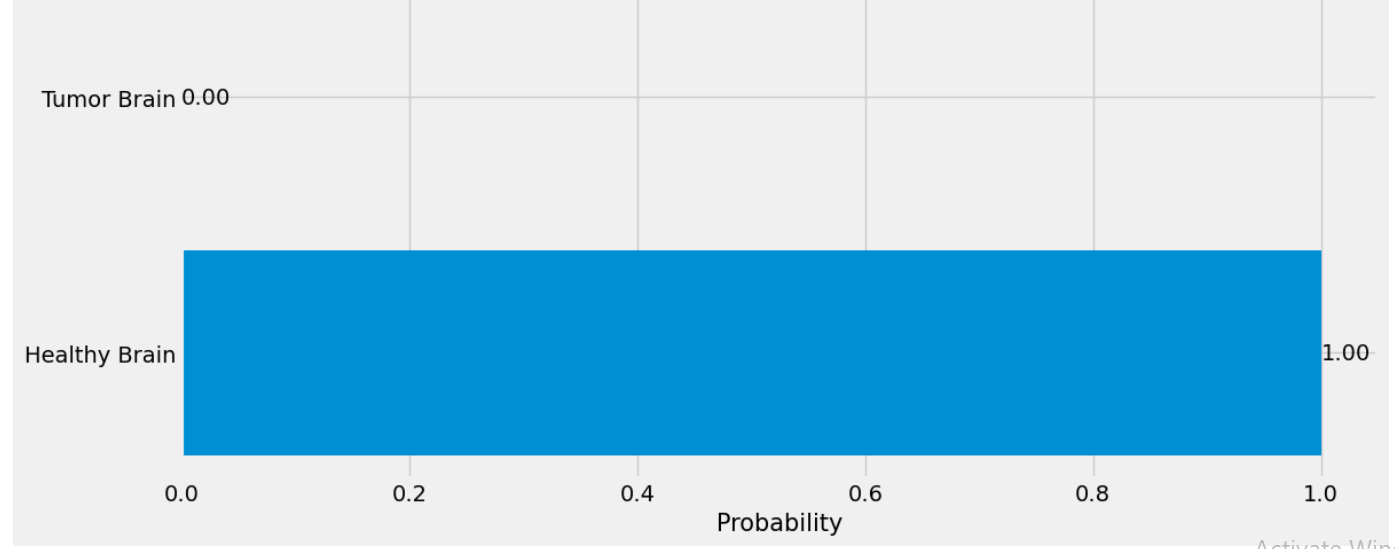
I created a function Predict to predict the images.

Defines the class labels that the model will predict. In this case, it distinguishes between a "Healthy Brain" and a "Tumor Brain".

* Loaded the image from the specified img\_path.
* Resizeed the image to a standard size of 224x224 pixels, which is a common input size for many deep learning models.
* Converted the image into a numpy array (img), expands its dimensions to match the expected input shape for model prediction (adds a batch dimension), and normalized the pixel values to the range [0, 1].
* Useed a pre-loaded model (not explicitly defined here) to predict the class probabilities for the input image (img).
* Retrieved the predicted probabilities from the model output and converts them into a list (probs).
* Ploted the resized image in the upper subplot (subplot(2, 1, 1)).
* Createed a horizontal bar chart in the lower subplot (subplot(2, 1, 2)) to visualize the predicted probabilities (probs) for each class (labels).
* Added labels to the bars showing the predicted probabilities (ax.bar\_label(bars, fmt='%.2f')).
* Displayed the entire plot with the image and the probability chart.

Finally, I used “Predict” function to predict whether an image is tumorous or non-tumorous.





Again, model predicted the image correctly, it was a non-tumor image

**Conclusion**

The predict function effectively demonstrates the capability of a pre-trained model to classify brain images as either "Healthy Brain" or "Tumor Brain". By preprocessing the input image and generating a visual representation of the model's predictions, the function provides a user-friendly way to interpret the results. The use of horizontal bar charts helps to clearly communicate the confidence levels of the predictions.

**Next Steps:**

#### Model Evaluation: Evaluate the model on a larger test dataset to measure its accuracy, precision, recall, and F1 score. This will help in understanding the model's performance and identifying any areas of improvement.

#### Data Augmentation: Implement data augmentation techniques to increase the diversity of the training dataset. This can help improve the model's generalization capabilities and robustness.

#### Fine-Tuning the Model: Fine-tune the pre-trained model using a specific dataset of brain images to improve its accuracy. This can be achieved by adjusting hyperparameters, using a different architecture, or employing techniques like transfer learning.

#### Error Analysis: Conduct a thorough error analysis to identify common misclassifications and their potential causes. This will provide insights into how the model can be further improved.

#### Cross-Validation: Use cross-validation techniques to ensure that the model performs consistently across different subsets of the data. This can help in validating the model's robustness.

#### User Interface Development: Develop a user-friendly interface for the predict function, allowing users to upload images and receive predictions easily. This can be in the form of a web application or a standalone software tool.

#### Integration with Clinical Workflow: Explore opportunities to integrate the model into clinical workflows for real-time brain tumor detection. This would involve collaborating with healthcare professionals to ensure the tool meets clinical standards and provides actionable insights.

#### Model Interpretability: Implement methods to increase the interpretability of the model, such as Grad-CAM or other visualization techniques. This can help clinicians understand why the model is making certain predictions, thereby increasing trust and adoption.

#### Regular Updates and Maintenance: Keep the model updated with new data and improvements to maintain its accuracy and relevance. Regular maintenance will ensure that the model continues to perform well as new brain imaging data becomes available.

By following these next steps, the predictive model can be continuously improved, making it a more reliable and valuable tool for brain tumor detection in clinical settings.