HIGH SCHOOL EDITION
Journal of Student Research
Volume 12 Issue 1 (2023)

Comparing Machine Learning Models to Determine Which is Most Effective at Detecting Brain Tumors

Samya Chauhan[1], Shreya Parchure[#] and Jenna Scott[#]

[1]Westlake High School, Broomfield, CO, USA
[#]Advisor

ABSTRACT

In this project, by using machine learning techniques to analyze various brain tumor scans, the goal was to determine which techniques are the most efficient and accurate in determining the presence of brain tumors. Specifically, the goal was to adequately process a dataset to determine if brain tumors can be detected with more than 75% accuracy using machine learning. The chosen dataset includes just over 250 axial MRI brain scans, thus providing a sufficient dataset to properly analyze. This research is most applicable to the healthcare field, specifically relating to the work done by neurologists and radiologists. If the most efficient and accurate way to detect brain tumors is not determined soon, individuals may have to continue waiting longer than necessary for their results. In this project, K-Nearest Neighbors, Decision Trees, and Muti-Layer Perceptron Models were compared to determine which machine learning algorithm is most effective at determining the presence of a brain tumor in an axial brain scan.

Introduction

Brain tumors are known to be one of the deadliest tumors. However, almost all tumor detection is done through radiologists and neurologists. For MRI scans to be read and a diagnosis to be made, the service of both a radiologist and neurologist are likely required. However, if machine learning were able to rapidly and accurately come to a conclusion, an initial diagnosis could be made, and a radiologist or neurologist would only have to confirm the diagnosis. This process could save the time of both patients and healthcare professionals. In addition, machine learning detection can be used in areas without access to many radiologists, providing patients with reliable and accurate diagnoses. In addition, this research is applicable to individuals living in areas with access to many neurologists or radiologists. An effective machine learning technique can reduce the workload on these medical professionals while still giving patients accurate diagnoses.

The first step in setting up this research was to acquire a trusted dataset of brain scans, some of which had brain tumors. After acquiring the dataset, it had to be processed and normalized to make it uniform across all brain scans. After the dataset had been preprocessed, it had to be split into training data and testing data. Following this split, different machine learning methods were applied to determine the most efficient method.

Background

A brain tumor is an abnormal growth of cells in the brain. These tumors can develop in various parts of the brain, making them difficult to detect.[1] Seeing as the number of individuals suffering with a brain tumor is increasing by around 1.50% every year, complications resulting from the presence of brain tumors are not forecasted to decrease any time soon.[2] Hence, researching machine learning's ability to detect brain tumors can eventually aid in efficient and accurate tumor detection.

In this research project, there are three main used models: K-Nearest Neighbors (KNN), Decision Tree, and Muti-Layer Perceptron Model (MLP). These three models were chosen because they are particularly compatible with image data for a classification problem. All models are supervised learning algorithms, meaning that they use training data to produce test data results.

K-Nearest Neighbors is a model that compares the similarity between a specified number of neighbors, already classified data, and the image that is to be classified.[3] Essentially, this model classifies data points based on similarity.

Decision Trees is a model that separates data based on features before classifying it.[4] The number of branches in a Decision Tree algorithm determines possible outcomes that will be tested. This is a binary algorithm, so each branch corresponds to an additional binary split.

Multi-Layer Perceptron Models have input and output layers with a varying number of hidden layers. This supervised neural network trains data on input data and learns the correlation between features and output.[5] The hidden layers apply different weights to transform the input data or data from the previous layer into something usable by the next layer or output layer. The neurons within the hidden layers apply the actual functions used in transforming the data.

Figure 1, from *TensorFlow Playground*, highlights the output of a neural network with 1 hidden layer and 3 neurons.[6]

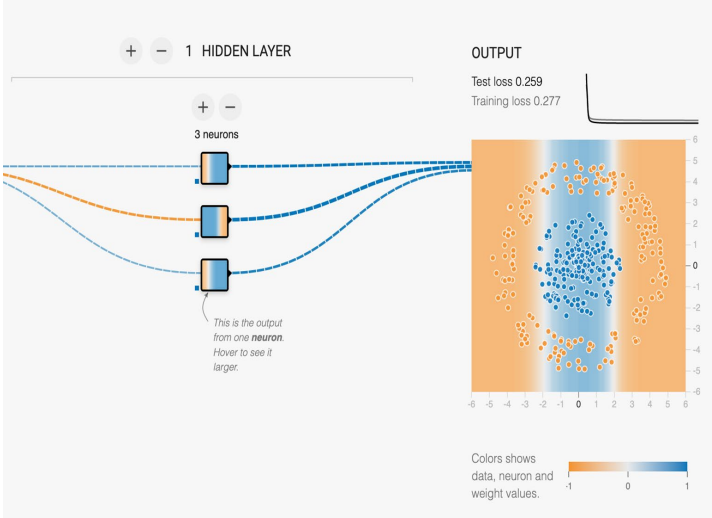


Figure 1. Neural Network with 1 Hidden Layer and 3 Neurons

Currently, most research done on using machine learning to detect brain tumors focuses only on neural networks. While these models have proved to be effective, research must still be done with other machine learning algorithms to determine if neural networks are truly the most effective.[7] While it is possible that this method is the most efficient, without considering other methods, the most accurate and efficient method could remain unemployed and untested. With this research, the objective was to compare multiple different machine learning algorithms to determine which algorithm is truly the most accurate and efficient for the purpose of brain tumor detection.

Dataset

The dataset used in this research project is titled “Brain Tumor Detection”. This dataset is entirely composed of axial MRI brain scans. These scans include both brains with a brain tumor and brains without a brain tumor. This dataset has 253 axial brain scans. In addition, the dataset labels scans with a brain tumor “1” and scans without a brain tumor “0”. 38.74% of the 253 scans had no brain tumor and 61.26% of the 253 scans had a brain tumor. As seen in Figure 2, this breaks down to 98 healthy scans and 155 tumor scans. Figures 3 and 4 depict how a brain scan without a brain tumor looks in comparison to a brain scan with a brain tumor.

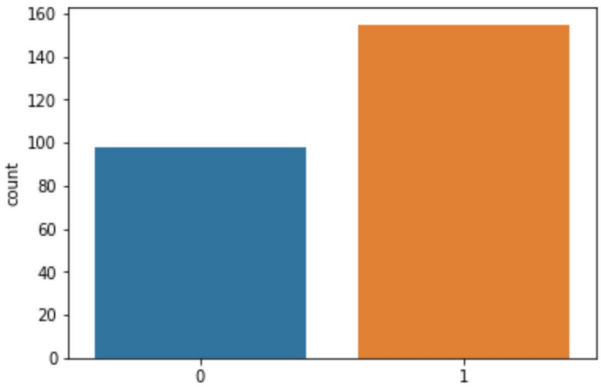


Figure 2. Bar Graph of Scans in Dataset with Brain Tumors vs Without Brain Tumors

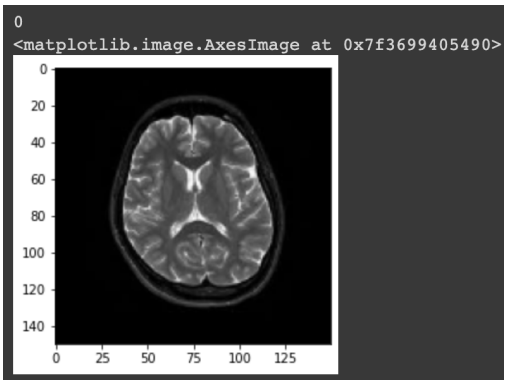


Figure 3. Brain Scan with Label 0: Healthy Scan Present

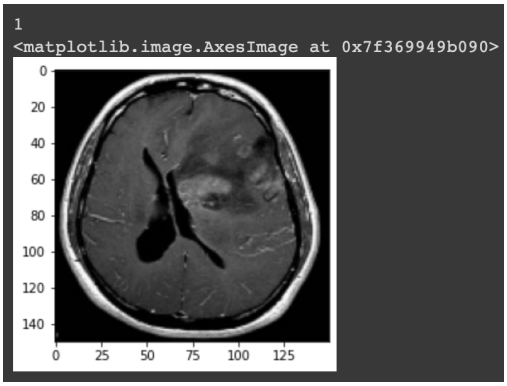


Figure 4. Brain Scan with Label 1: Tumor

After the dataset was acquired and the breakdown of brain scans with brain tumors versus brain scans without brain tumors was determined, the data was then preprocessed. The dataset before preprocessing had a shape of *(250, 150, 150, 3)*. In preprocessing the data, the goal was to normalize the dataset. Normalization ensured that all images acquired from the dataset were uniform and that there would be no variation in images that would cause variable outputs in the machine learning algorithms. To preprocess the data, first, all images were converted to greyscale.

Equation 1: Image Size Normalization

$$normalized = (image - np.min(image)) / (np.max(image) - np.min(image))$$

In this equation, *normalized* refers to a variable that stores the new, normalized image, while *image* refers to the image before it was normalized in respect to size. Lastly, the normalized images were made into an array so that they could be split before being used in the algorithms. The normalized dataset had a shape of *(253, 22500)*. This two-dimensional shape is usable by the machine learning algorithms employed in this research project.

Following preprocessing, the dataset was split into training and testing sets. The *normalizedImages* dataset was split into two datasets: 80% of the data went into the training set and 20% went into the testing set. A larger training set than testing set was used so that the models could discover and learn different patterns that will result in more accurate testing results. Alongside this dataset split, a random state of 35 was pre-assigned. Random state controls the shuffling of data when splitting the data into training and testing subsets.

Methodology

When evaluating the algorithm most effective at determining the presence of a brain tumor, both speed and accuracy were considered for all algorithms.

Equation 2: Elapsed Time Recorder

$$elapsedTime = time.time() - t$$

In the above equation, *time.time()* is a function that measures the total time, in seconds, the model has been running for. The variable *t* adds the total amount of time, in seconds, past runs of the model took to run. The variable *elapsedTime* quantifies the number of seconds the current algorithm took to run.

Classification Using K-Nearest Neighbors

When applying a K-Nearest Neighbors model, the number of neighbors was variable. The algorithm started with 3 neighbors, and after each run, the algorithm would increment the number of neighbors by 2 until the algorithm reached 31 neighbors. Essentially, the algorithm ran the model from 3 neighbors to 31 neighbors. Both accuracy and speed were graphed for this algorithm. To determine if the random state influenced the accuracy or efficiency, the model was again run, but this time it split the data with a variable random state. The random state used in splitting the data began at 35 and was incremented by 30 five times. In other words, each neighbor, from 3 to 31, was tested with random states from 35 to 185.

Classification Using Decision Trees

When applying a Decision Tree model, the number of branches was variable. The algorithm began with 2 branches, and after each run, the algorithm incremented the number of branches by 1 until the algorithm reached 22; branches 2 to 22 were tested. Both accuracy and speed were measured for this model. Only random state 35 was tested with this model.

Classification Using Multi-Layer Perceptrons

The multi-layer perceptron model was only tested with random state 35. This model only tested two neural networks. One neural network had 1 hidden layer of 3 neurons and the other neural network had 2 hidden layers, one layer with 3 neurons and another layer with 4 neurons.

Results

For this research, an effective model is one that is both accurate and efficient. If a model achieves an accuracy of over 70% and a fast run time, it is an acceptable model.[8]

When applying a K-Nearest Neighbors with a random state of 35 to the dataset, the relative speed in relation to the number of neighbors was analyzed. As seen in Figure 5, it was determined that the number of neighbors had a direct correlation to the speed of the model – as the number of neighbors increased, so did the speed. The lowest number of neighbors, 3, correlated to the fastest runtime, 0.06 seconds.

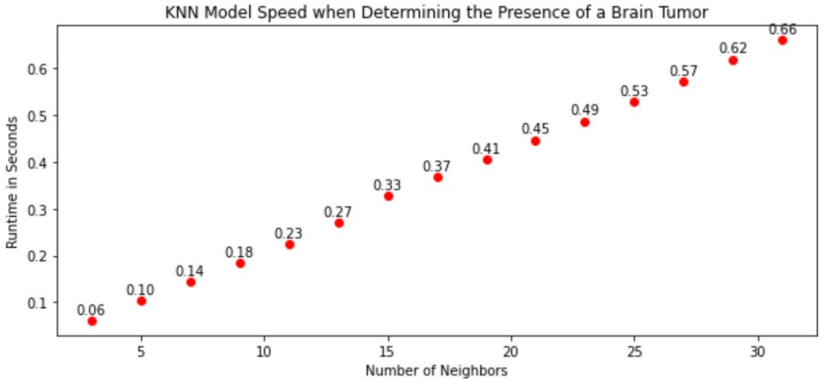


Figure 5. Scatter Plot with Relationship Between Number of Neighbors and K-Nearest Neighbors Model Runtime

However, the model's accuracy did not have a similar correlation. As shown in Figure 6, the model with 9 neighbors achieved the highest accuracy at 82%. Conversely, the fastest model, with 3 neighbors, had only a 73% accuracy.

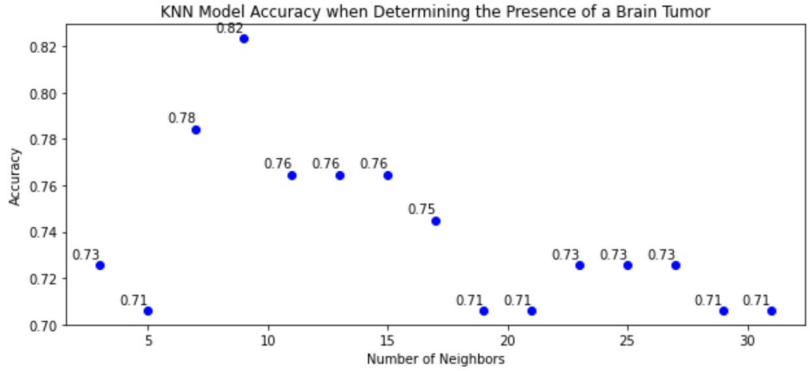
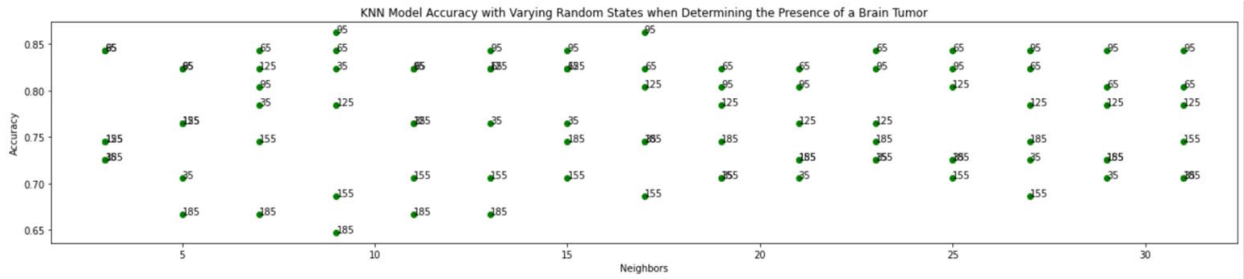


Figure 6. Scatter Plot with Relationship Between Number of Neighbors and K-Nearest Neighbors Model Accuracy with Random State 35

To determine whether the random state influenced the K-Nearest Neighbors model, varying random states from 35 to 185 were next applied when splitting the dataset into training and testing subsets. The accuracy of the model with varying random states is shown in Figure 7.

Figure 7. Scatter Plot Comparing the Number of Neighbors and Accuracy with Varying Random States in a



KNN Model

The x-axis is the number of neighbors tested in that algorithm, the y-axis is the accuracy represented as a decimal, and each point has its associated random state next to it. Despite these higher accuracy rates, the associated times were also higher. For instance, as shown in Figure 8, though 9 neighbors with a random state of 95 yielded one of the highest accuracy rates, 86.27%, it also yielded a higher runtime at 2.71 seconds.

```
Number of Neighbors: 9
Random State: 95
Accuracy: 0.8627450980392157
Time: 2.712177276611328 seconds
```

Figure 8. K-Nearest Neighbor Model Runtime and Accuracy with Random State 95 and 9 Neighbors

When applying a Decision Tree model with a random state of 35 to the dataset, once again the relative speed was compared to the accuracy. As seen in Figure 9, the Decision Tree model had a clear correlation between the number of branches and the runtime – as the number of branches increased, so did the runtime. Similarly, as seen in Figure 10, the Decision Tree model with the least number of branches had the highest accuracy.

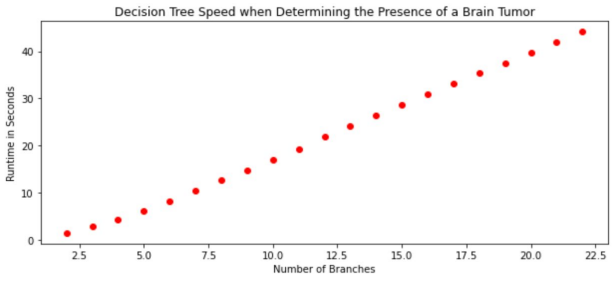


Figure 9. Scatter Plot with Relationship Between the Number of Branches and Decision Tree Model's Runtime

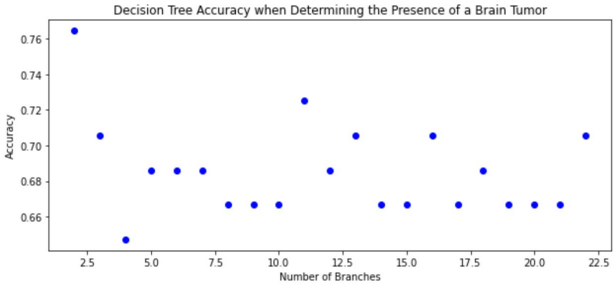


Figure 10. Scatter Plot with Relationship Between the Number of Branches and Decision Tree Model's Accuracy

Evidently, for the Decision Tree model, 2 branches yields both a low runtime, 1.40 seconds, and a relatively high accuracy in comparison to other branch amounts, 76.47%. Thus, it can be deduced that for the Decision Tree model, 2 branches produce the most effective algorithm.

The use of the Multi-Layer Perceptron Model in this research project was to determine whether a neural network would produce accuracy and efficiency comparable to the other models. As seen in Figure 11, a neural network with 1 hidden layer of 3 neurons produces a 78.43% accuracy. Furthermore, as seen in Figure 12, a neural network with 2 hidden layers of 3 neurons and 4 neutrons respectively has 70.59% accuracy.

```
nnet = MLPClassifier(hidden_layer_sizes=(3), random_state=1, max_iter=100000)
nnet.fit(XNN_train, yNN_train)
predictions = nnet.predict(XNN_test)
print("MLP Testing Set Score:")
print(accuracy_score(yNN_test, predictions)*100)

MLP Testing Set Score:
78.43137254901961
```

Figure 11. MLP Accuracy with 1 Hidden Layer of 3 Neurons

```
nnet = MLPClassifier(hidden_layer_sizes=(3, 4), random_state=1, max_iter=100000)
nnet.fit(XNN_train, yNN_train)
predictions = nnet.predict(XNN_test)
print("MLP Testing Set Score:")
print(accuracy_score(yNN_test, predictions)*100)

MLP Testing Set Score:
70.58823529411765
```

Figure 12. MLP Accuracy with 2 Hidden Layers of 3 Neurons and 4 Neurons Respectively

Though these accuracy scores are not as high as those of the K-Nearest Neighbors model, they are still viable models that could be further explored as potential models to be used in brain tumor detection.

While most of the chosen models are acceptable, it is evident that a K-Nearest Neighbor model with 9 neighbors and a random state of 95 is the most accurate, achieving an accuracy of 86.27%. This model took 2.71 seconds to run. Conversely, the fastest model, K-Nearest Neighbors with Random State 35 and 3 neighbors, corresponded to a runtime of 0.06 seconds. However, this model only achieved a 73% accuracy. When detecting brain tumors, accuracy is more important than runtime.

Thus, from the tested models, the K-Nearest Neighbor model with 9 neighbors and a random state of 95 is the best model for detecting brain tumors.

Discussion

By measuring different metrics, such as runtime and random states, this research paper demonstrates that there are multiple factors to be considered when determining the most effective machine learning model for determining the presence of brain tumors.

Conclusion

It was deduced that, from the tested models, a K-Nearest Neighbor model with 9 neighbors and a random state of 95 is the most accurate. Though it is not the fastest model, considering the importance of brain tumor detection to be accurate, this model would be the best to employ. Moreover, K-Nearest Neighbors are relatively transparent, and its results would not be difficult for radiologists and neurologists to decipher.

Using this research, neurologists and radiologists can reduce the amount of time they spend determining whether a brain tumor exists in their patient's MRI scan. In addition, healthcare officials located in regions without access to neurologists and radiologists can use these accurate models to efficiently diagnose patients and provide them with any medical support needed.

Limitations

The primary limitation in these models is the computational power required to run the different machine learning models. Since the chosen dataset has 253 images, this was not an enormous problem for this dataset. However, with larger datasets, this can prove to be an issue. Models that took under 5 seconds to run with this dataset size could potentially multiply in run time with larger datasets. To combat this issue, the models can be parallelized. This way, the models can be run in parallel and most likely reduce run time while maintaining relative accuracy.

Acknowledgments

I would like to thank my advisor for the valuable insight provided to me on this topic.

References

[1] John Hopkins Medicine, "Brain tumors and brain cancer," *Johns Hopkins Medicine*, 2022. [Online]. Available: https://www.hopkinsmedicine.org/health/conditions-and-diseases/brain-tumor. [Accessed: 04-Sep-2022].

[2] G. D. Healthcare, "Burden of brain cancer will continue to rise in the US aging population," *Clinical Trials Arena*, 14-Jan-2022. [Online]. Available: https://www.clinicaltrialsarena.com/comment/brain-cancer-us-ageing-population/#:~:text=GlobalData%20 epidemiologists%20forecast%20an%20increase,(AGR)%20of%201.50%25. [Accessed: 04-Sep-2022].

[3] O. Harrison, "Machine learning basics with the K-nearest neighbors algorithm," *Medium*, 14-Jul-2019. [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-

algorithm-6a6e71d01761#:~:text=KNN%20works%20by%20finding%20the,in%20the%20case%20of%20regression. [Accessed: 04-Sep-2022].

[4] P. Gupta, "Decision trees in machine learning," *Medium*, 12-Nov-2017. [Online]. Available: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052. [Accessed: 04-Sep-2022].

[5] G. Boesch, "Deep Neural Network: The 3 popular types (MLP, CNN and RNN)," *viso.ai*, 10-Jul-2022. [Online]. Available: https://viso.ai/deep-learning/deep-neural-network-three-popular-types/. [Accessed: 03-Sep-2022].

[6] D. Smilkov and S. Carter, "Tensorflow - Neural Network Playground," *A Neural Network Playground.* [Online]. Available: https://playground.tensorflow.org [Accessed: 01-Sep-2022].

[7] J. Amin, M. Sharif, A. Haldorai, M. Yasmin, and R. S. Nayak, " Brain tumor detection and classification using machine learning: a comprehensive survey," *Complex Intell. Syst.*, 08-Nov-2021. [Online]. Available: https://doi.org/10.1007/s40747-021-00563-y. [Accessed: 24-Jun-2022].

[8] K. Barkved, "How to Know If Your Machine Learning Model Has Good Performance." *Obviously AI*, 9 Mar. 2022. [Online]. Available: https://www.obviously.ai/post/machine-learning-model-performance. [Accessed: 07-Jul-2022].