Using Machine Learning to Classify Brain Tumors

Abhishek Mele Sydney

Abstract

For this project report we are looking into the best meth ods of machine learning for classifying brain tumors. We used data from images of MRI's to help identify whether an image had a brain tumor or not. Our main focus was on decision trees and ensembles of decision trees. In previous research, support vector machines (SVM) had been shown to have good accuracy for classifying tumors. A lot of addi tional preprocessing was used in order for the SVM to work, such as cropping the image. The main purpose of this re search is to help in clinical settings to identify brain tumors. Brain treatment is very expensive and knowing whether or not there is actually a brain tumor can have a huge im pact on the pocketbook of the patient. Gradient boosting and bagging were the ensembles that worked best for our data, with a very high accuracy. These methods use deci sion trees to help classify the data. With the help of machine learning, the next steps would be to see how these methods would work to differentiate between malignant and benign tumors. These findings can lead to life or death for indi viduals. Overall this research will help the medical field to have a starting point for directions to go in classifying brain tumors from MRI's.

1. Introduction

A brain tumor is an increase of cell growth that is ab normal to the traditional cranial development in the brain. Brain tumors are estimated to be in 700,000 people. With an additional 85,000 to be diagnosed with one this year. Brain tumors also do not have a preference on gender or ethnicity, and they affect all groups[3]. The average age to be diag nosed with a brain tumor is 60. But they affect people of all ages. There are different types of brain tumors. For ex ample: benign and malignant. Benign tumors are tumors that are non-life threatening. A majority (70%) of brain tu mors are benign and have a minimal effect on well being. Malignant tumors are tumors that are life-threatening. Cra nial tumors can also commence generation in different parts of the body. The primary type is the one that starts within the brain. The secondary type are tumors that start in non cranial portions of the body, then spread to the brain. Some symptoms of brain tumors are the following[2]:

- Seizures
- Headaches
- Sensory defectiveness (i.e. vision, smell, hearing, sen sation, etc)
- Loss of Balance

Brain tumor treatment is very expensive. To treat a brain tumor, it typically costs about $50,000 to $700,000 for treatment (benign to malignant, respectively)[1]. For misdiagnosis, a brain tumor can not only burden the indi vidual, but also the hospital. In one instance, a man was given a $59,000 compensation for a misdiagnosis of a brain tumor[10]. In another instance, a man almost lost his life due to a misdiagnosis of a brain tumor[9]. As you can see, the misclassification of patients with brain tumors are con cerning for financial and for personal health purposes. To decrease the chance of an error like that happening, using Machine Learning would be ideal. For our research, the group used Machine Learning to classify brain tumors ac curately. We used numeric data of the brain tumor matrix – given by a dataset on Kaggle [8] – to use in our algorithms. Throughout this paper, you'll see different Machine Learn ing approaches to classifying brain tumors. You will also see the efficiency potential in Machine Learning has in this field of medicine.

2. Related Work

Related works parallel with a research paper titled "A Machine Learning Approach for MRI Brain Tumor Classi fication" [11]. In this paper they found that they were able to use support vector machines, with the focus on a binary tree implementation, to classify the images. They also used wavelet transform to remove noise and extract features.

Another related work is "Brain tumor classification in MRI image using convolutional neural network" [12]. In this they use convolutional neural networks to use the im ages themselves instead of changing them into numerical data. This is a form of deep learning which is past the scope

of the class, but is still interesting to look at and consider for future work.

Dr. Umar Alqasemi, Mohammed Bamaleibd, Abdullah Al Baiti perform a study of developing a system to automat ically classify MRI brain images as normal or abnormal. Abnormal images mean a tumor is detected. The prepro cessing involved manually cropping 32x32 to focus on the area of interest or damaged area. Out of the first- order sta tistical, second-order statistical, and higher-order statistical features, 17 were extracted. Support Vector Machine clas sifier with parameters SVM-RBF, SVM-polynomial, SVM gaussian, and SVM-linear was used along with KNN with nearest neighbor values of 1, 2, 3, 4, and 5. They achieved the best accuracy of 100% with SVM-RBF[5].

Study performed by Nilesh Bhaskarrao Bahadure, Arun Kumar Ray, and Har Pal Thethi aims to improve per formance and reduce complexity of the medical image segmentation process, which depends on the experience of radiologists. The steps in the algorithm they use in volve preprocessing which enhances the image, removal of skull from image, and segmentation of tissues and tumor from each other based on Berkley Wavelet Transformation (BWT) and classification (into normal and abnormal tissue) using Support Vector Machines (SVM). Their SVM model got an accuracy of 96.51 which increased from 90.54 when not using feature extraction. The Accuracy of the area com pared to the area calculated by an expert radiologist was close to 100%[6].

Ravikumar Gurusamy and Dr Vijayan Subramaniam also use BWT based segmentation and SVM to classify images with malignant and benign tumors. They involved a pre processing technique of removing unwanted noise from the MRI images and achieved 98% performance in positive and negative predicted values[15].

3. Proposed Method

In this section we discuss all of the methods we will be using to classify brain tumors. We describe what they are and how they work.

3.1. KNN

KNN is a supervised ML algorithm mostly used for clas sification. Data points are assigned values bases on how closely they match the points in the training set. The dis tance is calculated between the test data and each row of the training data using methods such as Euclidean, Man hattan or Hamming distance. For our model we use Eu clidean distance. We sort this distance in ascending order and choose the top K rows. The value of K is the near est neighbor who we determine. Larger K value leads to smoother decision boundaries with low variance, but high bias. We use grid search hyper parameter tuning with dif ferent k values to find k value which returns the lowest error value. Grid search iteratively examines all combinations of the parameters for fitting the model. For each combination of hyper-parameters, the model is evaluated using the k-fold cross validation.

3.2. Decision Tree

A decision tree is a hierarchy of binary decisions which in turn divide a dataset into many subspaces. In decision trees there are four types of nodes that are important to know. The first one is a root node. A root node is the start ing point; it has no incoming edges and has zero or more outgoing edges. The next type of node is an internal node. Internal nodes have one incoming edge and one or more outgoing nodes. Another type of node is a leaf node. Leaf nodes are the branches off of the previous node. Finally, nodes that are split are referred to as parent nodes and all of the resulting nodes are child nodes. Decision trees have 2^m potential splits given the data set has m features.

3.3. Random Forest

Random forest is a combination of many individual de cision trees that operate together as an ensemble. Each tree has its own class prediction and the class with the most votes becomes the model's prediction. It is important that the cor relation between the decision trees is low to ensure that they are not relying on each other.

3.4. Majority Voting

Majority voting is the voting that combines predictions from a variety of models. Each model in the choosing makes a prediction. These predictions are considered as votes. The model that receives more than 50% of the votes wins. From this, we know that that model works best on the given dataset. We use that model to predict the best accuracy[13].

3.5. Bagging

Bagging is a method of the ensemble group. This method was created to improve accuracy and reduce variance. With bagging, the focus is to avoid overfitting the data and is used with high dimension data. We have a few steps in order to complete this bagging approach. This includes the follow ing:

- Selecting a random sample from the training dataset without replacement.
- Having a subset of feature that are randomly chosen
- The best split of the feature is used to

do other splits • The cycle is repeated

We can use a variety of different numbers of the training dataset to be selected at random. This way, we can find the best number of samples that gives the best accuracy[7].

2

3.6. Gradient Boosting

Boosting involves training a bunch of models sequen tially with each model trying to predict and explain the er ror of the previous one. We start by training a base tree and then training the next tree based on the errors of the previous one. In Gradient boosting we are optimizing a dif ferentiable loss function which in the case of classification is negative log likelihood for classification. The final model adds up the results of every step and we end up with a strong learner by combining the weak learners. The two important parameters for gradient boosting are learning rate and num ber of estimators. Learning rate controls how fast the model learns, and a slow learning rate has the advantage of being more robust, efficient and avoid overfitting, but is slow to train. The number of estimators controls the number trees, and more trees are required with slower learning rate.

3.7. Stacking CV

Stacking CV Classifier is essentially the same method as the regular Stacking Classifier but using cross-validation. A regular Stacking Classifier is one that uses a variety of ensemble methods – hence where the “stacking” concepts come in. StackingCV comes into play because it is used to combat the overfitting that regular Stacking brings. Stack ingCV uses the idea of “leave-one-out”, where you leave a certain section of the dataset for testing (say 1/5th) and the rest for training. You repeat this until each 1/5th of the dataset has been a part of the test set. From then, the train ing and test accuracy are computed[14].

4. Experiments

This section starts by describing the data and all of its variables. Followed by how we split the data to run our experiments. Before finally explaining what different pa rameters we used for all of our methods.

4.1. Dataset

For this project we are using a dataset from Kaggle called Brain Tumor [8]. The data began as over 3000 images and the author of the dataset compiled them all into 8 columns of numbers. This dataset used both first order and second order features of images to help describe and define the dif ferences in images. These numbers can then be used to clas sify the data. Figure 1 shows one of the images where a tumor is present, whereas Figure 2 is an image of a brain without a tumor.

To create the data the author started by creating a his togram with the likelihood of observing an intensity value at a random location in the image. From this he created 13 features as listed below. To help and define these features we used and article by Aggarwal [4]. Each point of the his togram can be found by using $P(I)$ as seen below.

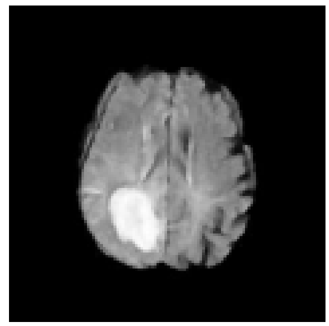

Figure 1. A sample image data of a brain scan where a tumor is present

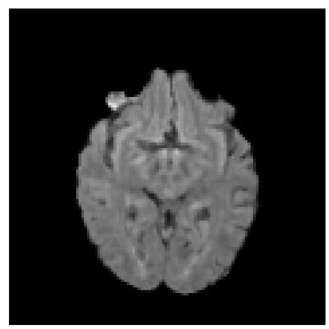

Figure 2. A sample image data of a brain scan where a tumor is not present.

$$P(I) = \frac{\text{number of pixels with gray level}}{\text{total number of pixels in the region}}$$

For the features we included a description and an equa tion for how the feature is calculated.

4.1.1 First Order Features

- Mean: average (central value) of the distribution $\mu = \sum I^1 P(I)$
- Variance: dispersion of the distribution $\sum (I - \mu)^2 P(I)$

- Standard Deviation: square root of the variance

  $\sqrt{\sum (I-\mu)^2 P(I)}$

- Skewness: describes asymmetry of the distribution $(I-\mu)^3 P(I)$

- Kurtosis: describes peakedness of the distribution $(I-\mu)^4 P(I)$

4.1.2 Second Order Features

- Contrast: measure of local level variation $\sum_{I_1, I_2} |I_1 - I_2|^2 log P(I_1, I_2)$

- Energy: mean level of steadiness and unifor mity in pixel intensity (square root of ASM)

  $\sqrt{\sum_{i,j} P(I_1, I_2)^2}$

- ASM (Angular second moment): measures the smoothness of an image (Uniformity of an image)

  $\sum_{i,j} P(I_1, I_2)^2$

- Entropy: measure of randomness and takes low values for smooth images. As its value increases, it becomes harder to predict events

  $-\sum_{I_1, I_2} P(I_1, I_2) log P(I_1, I_2)$

- Homogeneity: measure that takes high values for low contrast images $\sum_{I_1, I_2} \frac{P(I_1, I_2)}{1+|I_1-I_2|^2}$

- Dissimilarity: a measure of distance between pairs of pixels in the region of interest $\sum_{i,j} P(I_1, I_2)|I_1 - I_2|$

- Correlation: measure of correlation between pixels in two different directions $\sum_{I_1, I_2} \frac{(I_1-\mu_1)(I_2-\mu_2)P(I_1, I_2)}{\sigma_1 \sigma_2}$

- Coarseness: identify the largest size at which a texture exists (larger value of all texture values)

4.1.3 Splitting Data

To keep the experiments consistent we split the data the same for all methods. Our Y column was just the "class" column which included 1's for brain tumors and 0's for no brain tumor. Our X values included all of the columns for the 13 features listed above. This can be seen in Figure 3

3

4.2. KNN

We ran a KNN classifier with grid search and had the grid search try the values [1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 15, 17, 20] for the n neighbors parameter. Running the grid search looks through all the possibilities and outputs the best training accuracy.

4.3. Decision Tree

```
#X, y = dataset.data, dataset.target
tumors = dataset
y = tumors["Class"]
X = tumors.drop("Class", axis = 1)
X = tumors.drop("Image", axis = 1)
```

Figure 3. Data split into X Y variables

When using the test train split our test size was 0.3 for the initial split and 0.2 for the second split. We used ran dom state 123 for both splits and used the respective y value to stratify the data. This can all be seen in Figure 4.

```
X_temp, X_test, y_temp, y_test = \
    train_test_split(X,y, test_size = 0.3, random_state = 123, stratify = y)
X_train, X_valid, y_train, y_valid = \
    train_test_split(X_temp, y_temp, test_size = 0.2, random_state=123, stratify = y_temp)
```

Figure 4. Splits the Data Into Test, Validation, and Training Sub sets

For the decision tree we tried a variety of combinations changing the parameters criterion, max depth, and splitter. We then ran a grid search over the following parameters:

- max depth = [1, 2, 3, 4, 5, 15, 20, None]
- criterion = ['gini', 'entropy']
- splitter = ['random', 'best']

This grid search tries all the combinations and then out puts the parameters that give the best training accuracy to be used on the test data.

We also looked at the decision tree for the first and sec ond order features separately to see how they compared in accuracy to when they were used together. We performed the same grid search as above.

4.4. Random Forest

For random forest methods we also tried a variety of combinations. The parameters that were varied for this were criterion, max depth, and n estimators. After testing differ ent combinations we decided to do a grid search over the following parameters...

- max depth = [1, 3, 5, 10, 15, 20]
- criterion = ['gini', 'entropy']
- n estimators = [10, 50, 100]

This allowed us to allow the computer to search for us and give us the combination of parameters with the best training accuracy.

We also looked at the random forest method for the first and second order features separately to see how they com pared in accuracy to when they were used together. We per formed the same grid search as above.

4.5. Majority Voting

For Majority Voting, the first trial of parameters that were used produced the desired accuracy – as shown in sec tion 5.4. Because of this, there was no need to change pa rameters to increase accuracy. Refer to section 4.9 where a Majority Voting was used as one of the compare-baselines in a Dummy Classifier approach.

4.6. Bagging

For Bagging, the first trial of parameters that were used produced the desired accuracy – as shown in section 5.5. Because of this, there was no need to change parameters to increase accuracy. Refer to section 4.9 where a Bagging was used as one of the compare-baselines in a Dummy Classifier approach.

4.7. Gradient Boosting

Tried a few different variations of parameters, but kept the learning rate constant at 0.2. We did this to keep it more efficient and robust, along with avoiding overfitting. Max depth is the property that determines how deep the tree should go. Both the max depth and n estimators variables were changed to see their affect on the accuracy of the clas sification.

4.8. Stacking CV

For StackingCV, the first trial of parameters that were used produced the desired accuracy – as shown in section 5.7. Because of this, there was no need to change param eters to increase accuracy. Refer to section 4.9 where a StackingCV was used as one of the compare-baselines in a Dummy Classifier approach.

4.9. Dummy Classifier

During the experiment, the same theme was shown across the Bagging, Majority Voting, and StackingCV ap proaches. As a result, a Dummy Classifier from scikit-learn was used to compare accuracies. Originally this was used to compare the Bagging results; however, since Majority Vot ing and StackingCV followed the same trends as the Bag ging classifier, all three approaches were used as a compare baseline with the Dummy Classifier . The parameter used in the Dummy Classifier is strategy="most frequent".

4.10. Software

We used Jupyter Notebook to complete the experiments. Along with the sklearn python library to help us with the machine learning.

4.11. Hardware

We all used our respective laptops to complete this project. Our data was from Kaggle created by Jakesh Bohaju[8].

5. Results and Discussion

In this section we will be discussing the results of each of the methods and experiments we have mentioned in the report above.

5.1. KNN

The grid search for KNN gave the best accuracy for n neighbors = 3. This gave a best accuracy of 80.53%. and the test accuracy was 80.78%. After seeing this accuracy we wanted to see what else we could try to improve our accuracy.

5.2. Decision Tree

The grid search for decision tree gave us a best training accuracy of 98.37% with the parameters criterion = 'gini', max depth = 4, and splitter = 'best'. This gave us a test accuracy of 97.87%.

For the first order features the parameters that gave the best training accuracy at 90.68% were max depth = 10, cri terion = 'gini', and splitter = 'random'. When these param eters were used on the test data the accuracy was 89.39%.

For the second order features the parameters that gave the best training accuracy at 98.31% were max depth = 10, criterion = 'gini', and splitter = 'best'. When these parame ters were used on the test data the accuracy was 97.35%.

We can then see that using just the second order features gives us more better accuracy than just the first order fea tures. But when compared to the overall accuracy the sec ond order accuracy is slightly lower.

5.3. Random Forest

The random tree grid search gave us a best training ac curacy of 98.71% with the parameters: max depth = 10, criterion = 'gini', and n estimators = 50. The test accuracy with these parameters was 98.94%.

For the first order features the parameters that gave the best training accuracy at 92.68% were max depth = None, criterion = 'entropy', and n estimators = 100. When these parameters were used on the test data the accuracy was 91.51%.

For the second order features the parameters that gave the best training accuracy at 98.19% were max depth = 15, criterion = 'gini', and n estimators = 50. When these parameters were used on the test data the accuracy was 98.14%.

We can then see that using just the second order features gives us more better accuracy than just the first order fea tures. But when compared to the overall accuracy the sec ond order accuracy is slightly lower.

5.4. Majority Voting

With this approach, we had a Train size of 2106, a Vali dation size of 527, and a Test size of 1129. For classifier 1 of a Decision Tree Classifier, we used a random-state 1 and a max-depth of None. For classifier 2 of a Decision Tree Classifier, we used a random-state of 1 and a max-depth of 1. For classifier 3, we used a Decision Tree Classifier with a

random-state of 1 and a max-depth of 2. For the Ensemble, we used the Ensemble Vote Classifier with parameters of the previous trio along with weights of [1,1,1]. With these parameters, we yield 100% accuracy for Train, Validation, and Test for each classifier and ensemble.

5.5. Bagging

With this approach, a random-state of "1" and a max depth of "None" was used to build the tree. To build the bag, 500 estimators were used, the score and bootstrap were set to True, the bootstrap-features were set to False, and 1 job was used. With these parameters, we yield 100% accuracy for OOB, Train, Validation, and Test.

5.6. Gradient Boosting

Gradient boosting appeared to be a very good model for this data. With all combinations of n estimators and max depth the test accuracy was 100%.

5.7. Stacking CV

With this approach, we used the 5 neighbors, a random state of 123 for the Random Forest, Gradient Boosting, Ad aBoosting, Logistic Regression, and Decision Tree parame ters. For the StackingCVClassifier, we used the above clas sifiers as parameters for the classifier argument. We used Logistic Regression for the meta-classifier. Use probas was set to True. drop proba col was set to 'last' and CV was set to 10. With these parameters, we yield 100% accuracy for Train, Validation, and Test.

5.8. Dummy Classifier

There was a discrepancy between the compare-baselines (Bagging, Majority Voting, and StackingCV) and the Dummy Classifier. The Dummy Classifier produced a 55% accuracy whereas the trio produced 100% all-around. We came to the conclusion the accuracy given by the first three approaches were more adequate to the representation of the dataset and test accuracies principle as a whole.

6. Conclusions

We will begin by showing a table with all the methods we tried along with their accuracies when testing the data in Table 1.

Overall all the methods we tried did a nice job of clas sifying whether there was a brain tumor or not. Our meth ods and their results can be seen in table (reference) below. Our best methods were bagging, gradient boosting, major ity voting, and stacking cv all with test accuracies of 100%. With these high accuracies we could be worried about over fitting, but since both our train and test accuracies were this high we are not worried this is the case. We also used a validation set to ensure that our training accuracies were

Method	Accuracy
Dummy Classifier	55%
KNN	80.73%
Decision Tree	97.87%
(combined) Decision	89.39%
Tree (first order)	97.35%
Decision Tree (second	98.94%
order) Random Forest	91.51%
(combined) Random	98.14%
Forest (first order)	100%
Random Forest (second	100%
order) Majority Voting	100%
Bagging	100%
Gradient Boosting	
Stacking CV	

Table 1. This is a table of our results form all methods.

comparable to those of the validation set. With these high accuracies we also looked at a dummy classifier which gave us 55% test accuracies. We came to the conclusion that our data was just really well represented by these models and did not take into account the dummy classifier.

Both decision trees and random forests had high accura cies as well, around 98%. These were more accurate when using the second order features compared to the first order features, but were most accurate when using both. KNN fell short, with only a 80% accuracy and would not be the method of choice when classifying brain tumors.

The accuracy of these results are very serious and we do not want to be wrong. Knowing if there is a brain tumor or not is detrimental to diagnosis and future tests for a patient. Since our accuracy was so high across the board there is lit tle chance of false positives or negatives. A false positive in this case would be if we predicted there was a brain tumor if there actually was not. This could lead to stress and mental health struggles as anyone would feel with the diagnosis of a brain tumor. On the other hand, a false negative would be predicting no brain tumor and there actually being one. This would also cause issues as it is often important to find a brain tumor early, and find the best course of action for the patient may be delayed. As important as it is to reduce stress and mental health struggles, in this case it would be more important to optimize a lesser false negative. Overall we are pleased with the accuracy of our results and would not worry too much about these outcomes.

The next steps for this project would be to find data that says whether the tumor is malignant or benign. It would be interesting to see if the image data for these types of tumors would be classified with the same accuracy as we saw in this report. This would have a huge impact on the medi cal industry and would help to find cancerous brain tumors sooner. Along with limiting unnecessary brain surgeries for those with benign tumors that are not having an affect on the patient.

7. Acknowledgements

We would like to thank Professor Sebastian Raschka for teaching us about Machine Learning this semester. With out him we would have had no idea where to go with this project. We would also like to thank Jakesh Bohaju for post ing the dataset on Kaggle[8].

8. Contributions

For this project we split up the sections and we each took a few methods to test. Abhishek did KNN and gra dient boosting along with helping to find related works and write the report. Mele looking into bagging, majority vot ing, and stacking. She also helped to write the introduction and other parts of the report. Finally, Sydney looked into the methods of decision trees and random forests. She also formatted the report and wrote the conclusion.

9. Code

Our code can be found in a zip file named ”braintu mors.zip” that was turned in with the report.

References

[1] How much does brain tumor treatment cost?

[2] Brain tumor, Aug 2021.

[3] Quick brain tumor facts, Mar 2021.

[4] N. Aggarwal and R. Agrawal. First and second order statis tics features for classification of magnetic resonance brain images. 2012.

[5] D. U. Alqasemi, M. Bamaleibd, and A. A. Baiti. Classifi cation of brain mri tumor images. *International Journal of Engineering Researcg Technology*, 10, 2021.

[6] N. B. Bahadure, A. K. Ray, and H. P. Thethi. Image analysis for mri based brain tumor detection and feature extraction us ing biologically inspired bwt and svm. *International journal of biomedical imaging*, 2017, 2017.

[7] A. Biswal. What is bagging in machine learning and how to perform bagging, Sep 2021.

[8] J. Bohaju. Brain tumor, 2020.

[9] R. Cahill. Misdiagnosed brain tumor could have cost man his life.

[10] S. Donaldson James. Montana man gets $59,000 for brain cancer misdiagnosis, May 2013.

[11] R. Gurusamy and V. Subramaniam. A machine learning ap proach for mri brain tumor classification. *Computers, Mate rials and Continua*, 53(2):91–109, 2017.

[12] H. A. Khan, W. Jue, M. Mushtaq, and M. U. Mushtaq. Brain tumor classification in mri image using convolutional neural network. *Math. Biosci. Eng*, 17:6203, 2020.

[13] P. Necati Demir. Ensemble methods: Elegant techniques to produce improved machine learning results, Feb 2016. [14] S. Raschka. Stackingcvclassifier.

[15] D. V. S. Ravikumar Gurusamy. A machine learning approach for mri brain tumor classification. *Computers, Materials & Continua*, 53(2):91–108, 2017.