UNIVERSITÄT ZU LÜBECK
INSTITUTE OF MEDICAL INFORMATICS

Seminar

Medical Image Computing and e-Health

WS 2020/2021

Automatic Detection and Segmentation of Brain Tumor Using Random Forest Approach

Leonard Brenk
Matr.-Nr.: 697947, Computer Science

Supervisor
Marja Fleitmann

Lübeck, February 27, 2021

IM FOCUS DAS LEBEN

Contents

1 Motivation

The detection and treatment of tumors is a very complex and challenging task. Finding mutating cells and preventing them from spreading is a complicated task. Usually, tumors occur nested in non-affected tissues, which makes their discovery and treatment heavily challenging. The current state of technology operating in that field has complications filtering the negative cells out and segmenting the pure tumor completely. Due to the variety of anatomical structures and how they can differ from person to person, even for a trained professional, it takes a lot of time to detect and fully segment a tumor in MRI scans. Therefore not only are the needed financial and personal requirements greatly inefficient, but also the outcome might be flawed and incomplete based on the individual know-how of the experts.

Seeing that finding the tumor in an early stage has the highest impact and can facilitate and enhance the treatment, the detection is the key to success. The paper written by Zoltán Kapás in 2016 captures an experiment regarding this topic. It carries out a Random Forest(RDF) based Machine Learning(ML) technique using multispectral MRI volumes. Training an algorithm to find and segment tumor cells on MRI scans reliably can, due to its large resources in time and memory, possibly perform on a higher level than multiple experts, as Kapás states. Before analyzing the data, it must undergo several pre-processing steps to be suitable for the application in Binary Decision Trees(BDT). Furthermore, after the employment of RDF, the data is post-processed and then analyzed. In order to illustrate the accuracy of a decision, the authors applied a Dice Score. Thereby RDF can be compared and graded. The ML method used in this paper belongs to the supervised learning techniques, which means that the outcome of data the machine is trained with is already known and can be used to adjust and optimize the parameters.

The paper presents initial outcomes and recommendations regarding a complex brain tumor detection and segmentation system and its future implementation in a clinical context.

2 Introduction BDT and RDF

In this section, the essential basics regarding BDT and RDF and their implementation are being explained and discussed.

2.1 Binary Decision Trees

2.1.1 General

A Binary Decision Tree (BDT) is trained and employed in order to make a decision based on an input vector. It consists of multiple levels of two-way decisions until it reaches a leaf node at the end, classifying the vector into a certain class. A BDT can be used to either deploy data into classes or predict values in the future using regression. However, this paper will concentrate on the ability to assign a label to a vector.

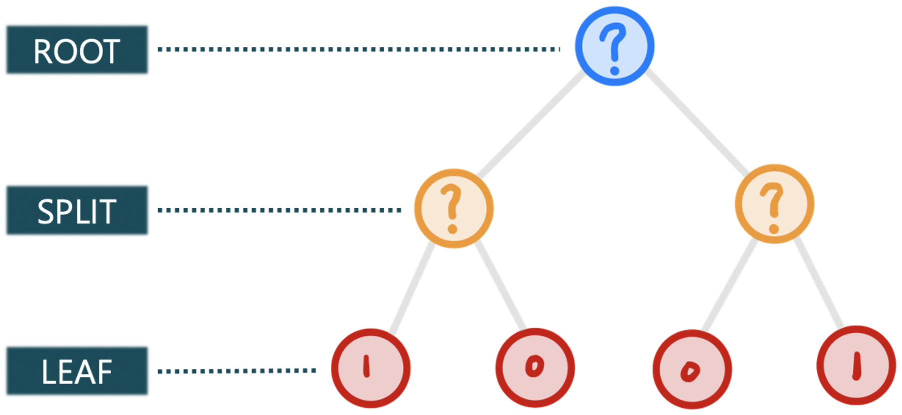


Fig. 1: The structure of a BDT. Usually there are multiple levels of split nodes.

The root node is the starting point of a BDT. In order for a BDT to reach a decision, it goes through several inner decision nodes, which divide the data into two subgroups and forwards them to the next node. At each split node, the data is divided again as long as the data subset decisions are distinguishable. If the decisions are unilateral, a leaf node is created. Such leaf nodes are then attributed to a class, also called a label. Thereby an inserted data vector can be classified.

2.1.2 Building a BDT

A BDT can, amongst other options, be built from a table that contains multiple attributes and a column for the individual decision of that data row. When building a BDT, a cycle is implemented recursively. The data shall be divided into two subgroups at a time. Then each subgroup is again divided into two subgroups. In the end, there are as many subgroups as it takes to classify every single data vector correctly. The cycle starts with a given data set. For any first split, an attribute needs to be picked, which the data is split with. Having decided upon that attribute, a threshold is set to decide whether a vector is forwarded to the left or right child node - depending on whether its value is higher or smaller than the threshold. Using the threshold, the complete data set can then be divided into two subgroups. Now each subgroup is considered individually, and the cycle begins again. The process ends when the decisions of a subgroup are all identical. Then the leaf node is created labeled with that decision.

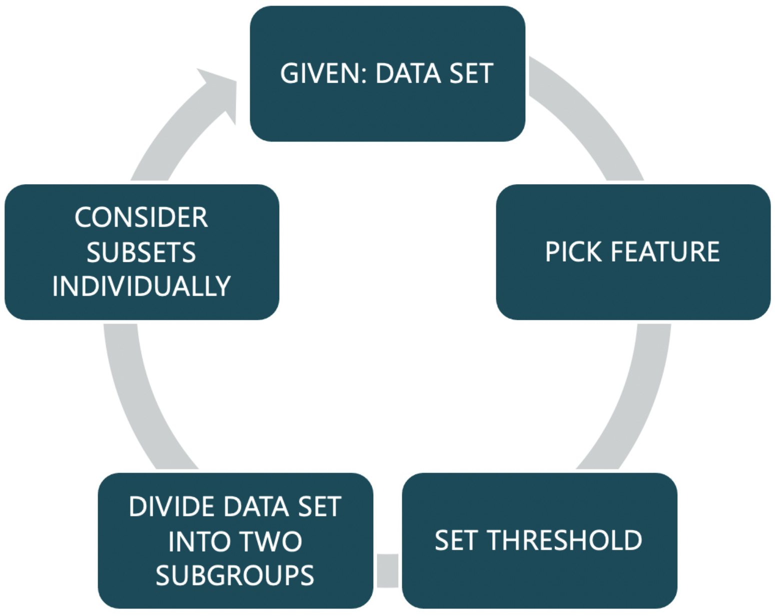

GIVEN: DATA SET
PICK FEATURE
SET THRESHOLD
DIVIDE DATA SET INTO TWO SUBGROUPS
CONSIDER SUBSETS INDIVIDUALLY

Fig. 2: The process of buidling a BDT.

The decision for a specific feature while building a BDT can greatly influence the outcome since the generated subgroups will be completely different. That is why there is a need for a way to assess the most efficient way of splitting the data. For that, the information entropy and information gain are used. The entropy describes the level of information of a variable (1). The information gain compares two entropies and calculates the difference(2). The idea is to test every attribute as a possible option for a split node and calculate which attribute would yield the highest gain in information after the split.

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i) \tag{1}$$

$$InformationGain = Entropy(Parent) - AverageEntropy(Children) \tag{2}$$

Temperature	Rain	Windy	Humidity	Go outside?
High	Yes	True	High	No
Low	No	False	Low	Yes
Low	No	True	Low	No
High	No	False	High	Yes
High	Yes	False	Low	No
Low	Yes	True	Low	No
High	No	False	Low	Yes
Low	Yes	True	High	No
Low	No	True	High	Yes
Low	No	False	Low	Yes
High	Yes	True	High	No
High	Yes	False	Low	No
Low	No	True	High	Yes
High	Yes	False	High	No

Table 1: A BDT can be built from a table like such.

2.1.3 Example BDT

Given a table of sample data:

In order to decide which attribute to pick for the root node the information gain must be calculated for all of them (Temperature, Rain, Windy, and Humidity). For example splitting with Temperature would look like this:

First, we need to calculate the entropy of the complete table.

$$H(X) = -\left[\frac{6}{14}\log_2(\frac{6}{14} + \frac{8}{14}\log_2(\frac{8}{14})\right] = 0,9852 \tag{3}$$

If we were to split at Temperature, the table would be divided into two groups, one with high temperature and one with low temperature. As it has a binary result there is no need to calculate a threshold. Now the entropy for both of these subgroups needs to be calculated individually. The information gain for Temperature is the difference resulting from the complete entropy minus the average entropy of the newly generated subgroups. In this case, the information gain is $0,01615$.

Split at Temperature:

$$\text{Temperature} = \text{high:}\ H(2,5) = -\left[\frac{2}{7}\log_2\left(\frac{2}{7}\right) + \frac{5}{7}\log_2\left(\frac{5}{7}\right)\right] = 0,8631$$

$$\text{Temperature} = \text{low:}\ H(4,3) = -\left[\frac{4}{7}\log_2\left(\frac{4}{7}\right) + \frac{3}{7}\log_2\left(\frac{3}{7}\right)\right] = 0,9852$$

$$\text{Gain:}\ H(6,8) - H\left[\frac{7}{14}H(2,5) + \frac{7}{14}H(4,3)\right] = 0{,}01615$$

Since the information gain is the value that needs to be maximized, the information gain is calculated for the remaining attributes as well. For the root node, the attribute is selected that yields the highest information gain. In this case: Rain.

Split at Temperature:

$$\text{Temperature} = \text{high:}\ H(2,5) = -\left[\frac{2}{7}\log_2\left(\frac{2}{7}\right) + \frac{5}{7}\log_2\left(\frac{5}{7}\right)\right] = 0,8631$$

$$\text{Temperature} = \text{low:}\ H(4,3) = -\left[\frac{4}{7}\log_2\left(\frac{4}{7}\right) + \frac{3}{7}\log_2\left(\frac{3}{7}\right)\right] = 0,9852$$

$$\text{Gain:}\ H(6,8) - H\left[\frac{7}{14}H(2,5) + \frac{7}{14}H(4,3)\right] = 0{,}01615$$

Split at Windy:

$$\text{Windy} = \text{True:}\ H(2,5) = -\left[\frac{2}{7}\log_2\left(\frac{2}{7}\right) + \frac{5}{7}\log_2\left(\frac{5}{7}\right)\right] = 0,8631$$

$$\text{Windy} = \text{False:}\ H(4,3) = -\left[\frac{4}{7}\log_2\left(\frac{4}{7}\right) + \frac{3}{7}\log_2\left(\frac{3}{7}\right)\right] = 1,3781$$

$$\text{Gain:}\ H(6,8) - H\left[\frac{7}{14}H(2,5) + \frac{7}{14}H(4,3)\right] = 0{,}1354$$

Split at Rain:

$$\text{Rain} = \text{yes:}\ H(0,7) = -\left[\frac{0}{7}\log_2\left(\frac{0}{7}\right) + \frac{7}{7}\log_2\left(\frac{7}{7}\right)\right] = 0$$

$$\text{Rain} = \text{no:}\ H(6,1) = -\left[\frac{6}{7}\log_2\left(\frac{6}{7}\right) + \frac{1}{7}\log_2\left(\frac{1}{7}\right)\right] = 0,5916$$

$$\text{Gain:}\ H(6,8) - H\left[\frac{7}{14}H(0,7) + \frac{7}{14}H(5,2)\right] = 0{,}6894$$

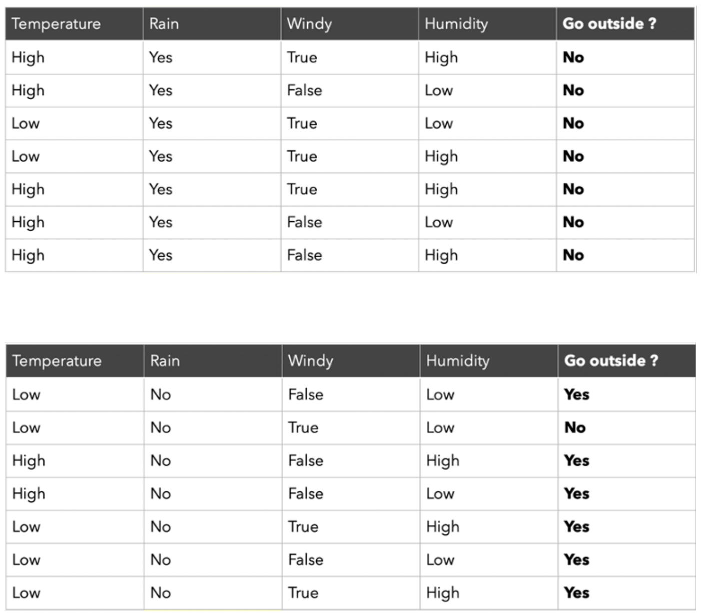
Split at Humidity:

$$\text{Humidity} = \text{High:}\ H(3,4) = -\left[\frac{3}{7}\log_2\left(\frac{3}{7}\right) + \frac{4}{7}\log_2\left(\frac{4}{7}\right)\right] = 0,9852$$

$$\text{Humidity} = \text{Low:}\ H(3,4) = -\left[\frac{3}{7}\log_2\left(\frac{3}{7}\right) + \frac{4}{7}\log_2\left(\frac{4}{7}\right)\right] = 0,9852$$

$$\text{Gain:}\ H(6,8) - H\left[\frac{7}{14}H(3,4) + \frac{7}{14}H(3,4)\right] = 0$$

Having split the data with rain, two subgroups are created (Fig 5). One with all Rain values 'Yes' and one with 'No' Rain. Selecting the attribute for the next split node for a subgroup proceeds exactly like with the root node, only with a smaller set of data. The two subgroups now look like this:



Temperature	Rain	Windy	Humidity	**Go outside ?**
High	Yes	True	High	**No**
High	Yes	False	Low	**No**
Low	Yes	True	Low	**No**
Low	Yes	True	High	**No**
High	Yes	True	High	**No**
High	Yes	False	Low	**No**
High	Yes	False	High	**No**

Temperature	Rain	Windy	Humidity	**Go outside ?**
Low	No	False	Low	**Yes**
Low	No	True	Low	**No**
High	No	False	High	**Yes**
High	No	False	Low	**Yes**
Low	No	True	High	**Yes**
Low	No	False	Low	**Yes**
Low	No	True	High	**Yes**

Fig. 3: The two generated subgroups when splitting with Rain at the root node.

The subgroups are recursively divided as long as there are differences in the subgroups decisions. When this is the case leaf nodes are created to label the inserted vector. If all leaf nodes have been created, the final tree is completely trained. In this case the Temperature is not essential for the decision making process. The tree looks like this:

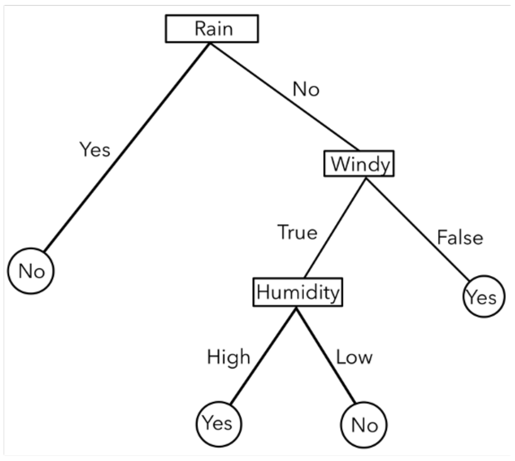


Fig. 4: The final BDT.

2.1.4 Testing a BDT

After building a BDT, it has to be ensured that it is working correctly and classifies properly. Therefore not all data is used for building the tree, but a part is saved for testing purposes. Now that the BDT is built, we can pick samples of the test data and insert it into the BDT. As this is a supervised learning ensemble, the result is already known and can thereby be compared with the tree's yielded outcome. If it is correct, the tree is working fine. The accuracy characterizing Dice Score is based on testing cases and describes their outcome in a numerical, understandable way.

2.1.5 Pro and Cons of BDT

A BDT can be used to classify and label data vectors, which can help many problems. As it is not limited to categorical values for classification but can also work with numerical values, a BDT can also predict future values through regression curves. Furthermore, the technique and structure of a BDT and its decision finding process is intuitive and visualizable. Additionally, the data used only requires little data pre-processing TODO WHY.

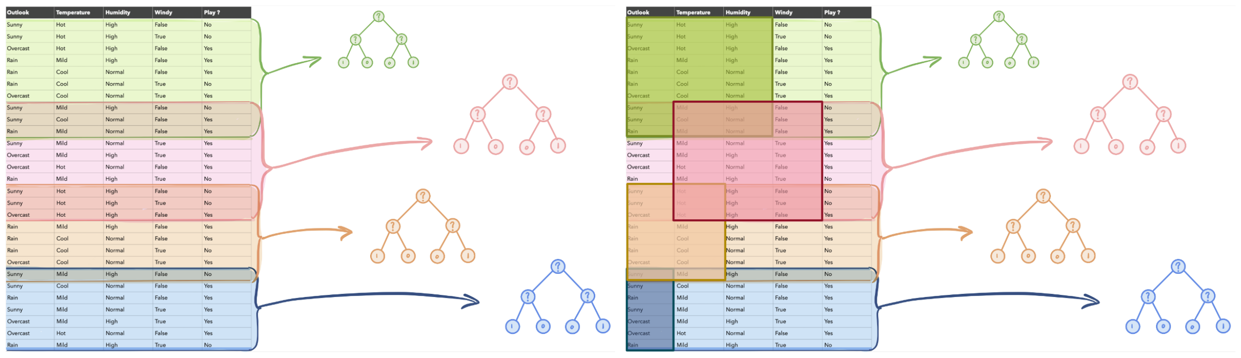
The problem with BDTs is that small changes in data can heavily affect the outcome of the decision. If the tree internalizes every detail of the training data, it can overfit. That means that the BDT does not reflect the general input data but has adapted to the training data. For instance: If an algorithm was given scans of tumors that are compared to all other existing tumor scans rather dark, then the BDT will set this as the general intensity of a tumor scan. Thereby brighter parts of tumor scans that the BDT has not seen will not be classified as a tumor. In order to face that issue, Random Forests have been introduced.

2.2 Random Forest(RDF)

Random Forest (RDF) overcome the obstacle of overfitting through the use of multiple BDTs. A tested vector is inserted into multiple trees, which all yield a decision for that vector. One way to determine the overall final decision is to decide by a majority vote. An individual tree might be overfitted at some points in the forest; however, this does not influence the final decision too much.

2.2.1 Bagging

While building a random forest, one must build several trees. When for each tree, only a subset of the training data is being used, it is called Bagging. Thereby randomness is introduced in the procedure and improves the reliability of the decision.



Outlook	Temperature	Humidity	Windy	Play ?
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rain	Mild	High	False	Yes
Rain	Cool	Normal	False	Yes
Rain	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rain	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rain	Mild	High	True	No
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rain	Mild	High	False	Yes
Rain	Cool	Normal	False	Yes
Rain	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rain	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rain	Mild	High	True	No

Fig. 5: Bagging(left): Each tree is trained by individual data set, Baggin +XY (right) Each tree is trained by an individual set of data and features

3 Application

3.1 Goal of the Paper

The paper aims to develop a reliable procedure based on machine learning ensembles to detect tumors on MRI scans in an early stage and segment them. For the execution of the experiment, 12 records from the BRATS Data Set were used. Each record belongs to one patient and contains four different MRI scans (T1, T2, T1C, FLAIR), which display the same brain in different ways. Each record consists of approximately 1.5 Mio feature vectors. There is a truth image containing expert annotations for either an active tumor or edema cells for every volume. The algorithm is supposed to separate tumor cells, edema cells, and negative cells.

3.2 Data & Pre-Processing

In order to address a three-dimensional pixel, also called voxel, a feature vector is generated. It looks like this:

$$T[x, y, z] = T1[x, y, z], T2[x, y, z], T1C[x, y, z], FLAIR[x, y, z]$$

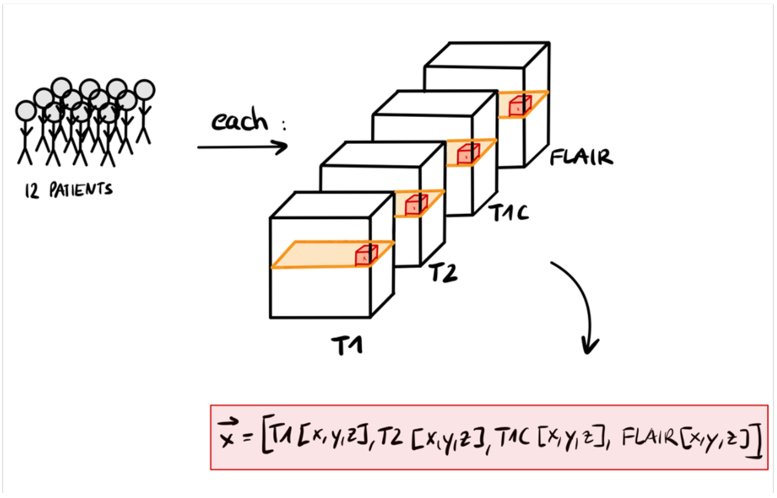

12 PATIENTS
each :
FLAIR
T1C
T2
T1
$\vec{x}$ = [T1 [x,y,z], T2 [x,y,z], T1C [x,y,z], FLAIR[x,y,z]]

Fig. 6: The Feature vector is generated to address a specific cell in all four cells

Before the data is used in training and testing BDTs and RDFs, it needs to be pre-processed. The first step is the histogram normalization. A histogram displays the number of intensity values of a scan. In this case, the middle 50% of the data shall be located between the intensity values of 600 and 800. Also, the minimum value of 200 and the maximum value of 1200 are set as boundaries. The second step involves the location of a considered voxel. Since the feature vector does not include any information regarding the voxel's location, two more features have to be computed for each scan. The new feature vector will thereby consist of 12 elements. For the computation of the first added feature, all direct neighbor voxels in a $3 \times 3 \times 3$-sized Cube are considered. The average intensity value of them is the first new feature value of the vector. The second feature is computed based on the average intensity value of the eight closest voxels of that slice TODO WHAT SLICE and two closest ones from the neighboring slices. That is what the new feature vector now looks like:

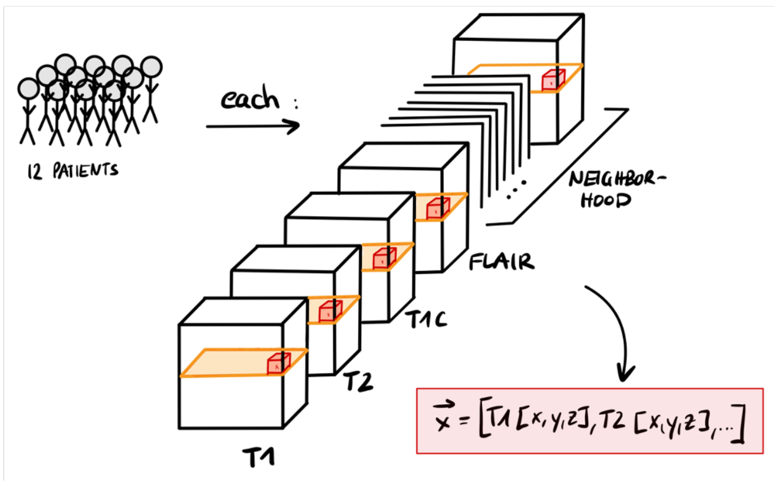


Fig. 7: The updated Feature vector contains twelve elements

The last step of pre-processing excludes the missing voxels from the average calculations for the neighborhood. Null values would distort the result and must thereby be ignored for the average intensity value.

3.3 Training and Testing

The data is now ready to be used for training BDTs. As explained in section 2.1 the tree is built, while the attributes like Rain or Temperature are now T1, T2, etc. As 1.5 Mio vectors are too many to use on training, the tree samples have to be selected. When selecting, e.g., 300 samples, then it means that 300 feature vectors representing tumor cells, 300 feature vectors representing edema, and 300 feature vectors representing negative cells are used.

3.4 post-Processing

Even though the BDT and RDF classify most of the voxels reliably, there are usually some negative cells labeled as tumor or edema due to the great variety of normal tissues they can belong to. In order to correct falsely labeled positives for each tumor and edema voxel, a 250-voxel neighborhood is defined. The average of these is then compared with a pre-defined threshold, which value is debatable. If the voxel intensity in question is less than the average intensities, it is relabeled into a negative voxel; it remains the same if it is higher.

4 Results

The first achievement of this paper's experiment is that the size of a sample used for training a BDT influences the accuracy of the decision. In Figure 10 the increase of the Dice Score is shown per rising number of test cases.

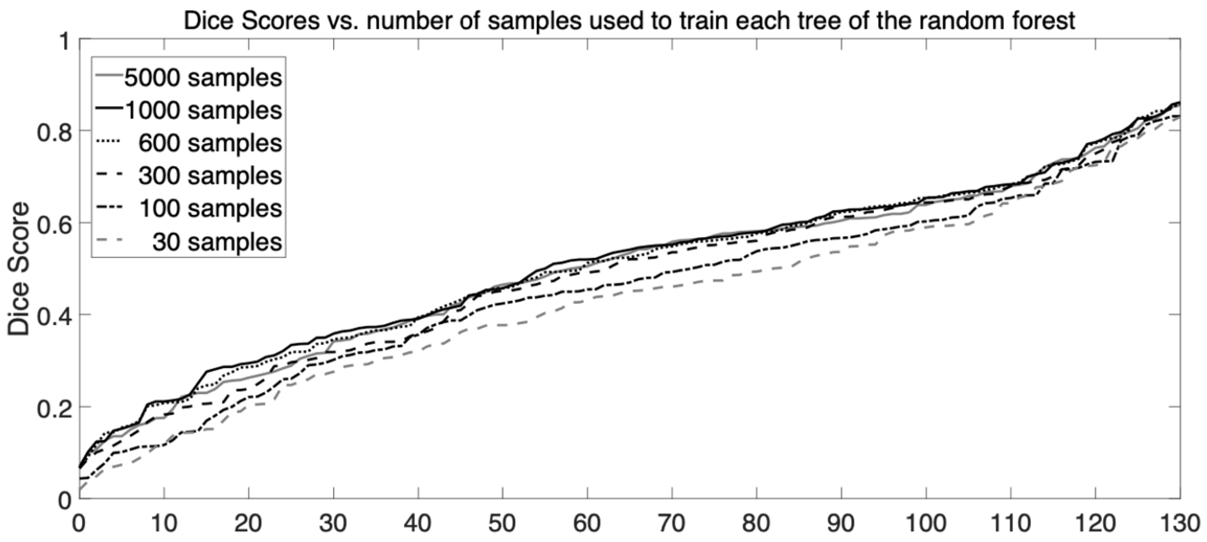


Fig. 8: The sample size can influence the Dice Score

The second result shows that training a RDF with volume *a* and testing with volume *b* does not yield the same outcome concerning accuracy as training with volume *b* and testing with *a*. That means that there is no symmetry regarding training and testing with volumes. Figure 11 shows the table's left side shows Dice Scores after training with just one volume and testing with all others, while the right site is the other way around. It should be emphasized that the right site draws a higher overall score. This shows that training a whole RDF with just one volume does not yield acceptable results, while training with all volumes but one can be possible.

Volume i	Train data selected from volume i Testing on each volume $j \neq i$				Testing on volume j When trained on each volume $i \neq j$			
	Average	SD	Maximum	Minimum	Average	SD	Maximum	Minimum
HG01	0.5216	0.1722	0.7822	0.2133	0.5857	0.1627	0.8304	0.2508
HG02	0.6191	0.1280	0.8262	0.3500	0.4637	0.2117	0.7416	0.1013
HG03	0.5721	0.1767	0.8615	0.2945	0.4532	0.2309	0.8082	0.1579
HG04	0.5107	0.1983	0.8304	0.2043	0.3902	0.1502	0.6377	0.1474
HG05	0.3872	0.2188	0.6652	0.0552	0.3848	0.1235	0.5279	0.1519
HG06	0.5648	0.1537	0.8386	0.3113	0.5918	0.0955	0.7273	0.6980
HG07	0.5815	0.1537	0.8270	0.3368	0.4796	0.1666	0.7052	0.2110
HG09	0.2662	0.1626	0.5126	0.0684	0.4659	0.1230	0.5789	0.2043
HG11	0.4939	0.2064	0.8564	0.2111	0.5718	0.1320	0.7090	0.2863
HG13	0.5630	0.1992	0.8082	0.2758	0.3368	0.2351	0.6995	0.0552
HG14	0.4721	0.1804	0.7299	0.1248	0.5628	0.1621	0.8270	0.3340
HG15	0.4837	0.1359	0.6426	0.2255	0.7493	0.1144	0.8615	0.5000

Fig. 9: Training and testing is not symmetric

The following charts concern the effect of post-processing. Looking at the blue diagrams, it becomes clear that Post-Processing can be highly beneficial for increasing the Dice Score. In the lower graphs, the Dice Score is drawn depending on how the threshold for the 250-voxel neighborhood. One can say that a threshold around 200 yields the most promising results.

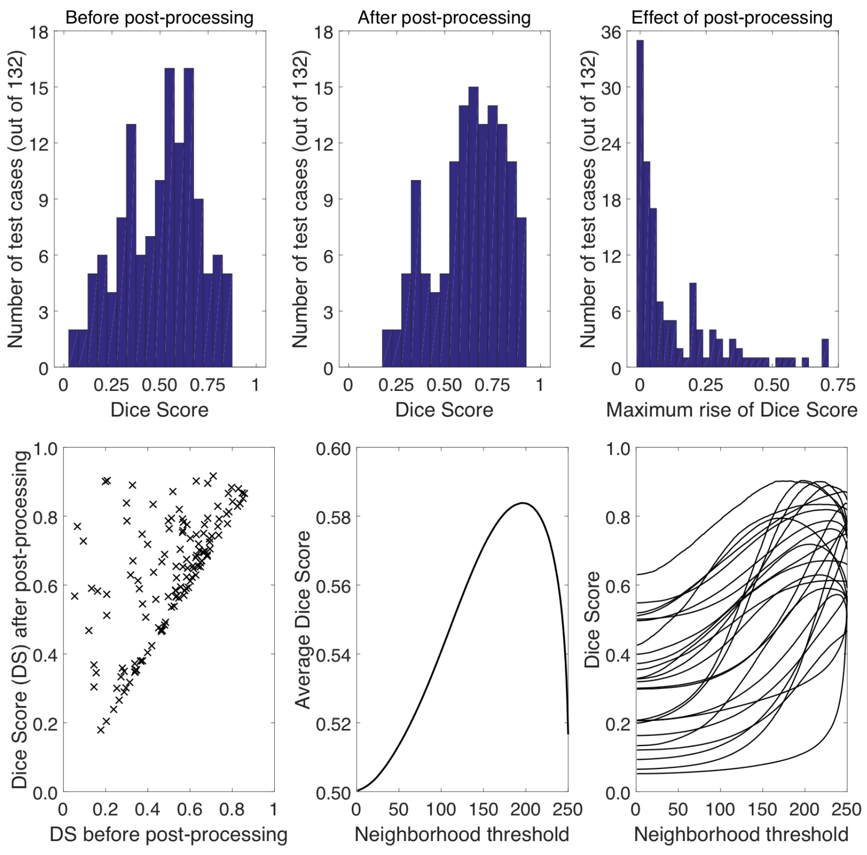


Fig. 10: The effect of Post-Processing

In Figure 13, two MRI scans are shown, one without Post-processing (left) and one with Post-Processing (right). What stands out is that Post-Processing finds non-affected cells in between tumor cells that would have been labeled positive. That gives a more detailed and concise scan, which can facilitate and enhance the tumor's treatment.

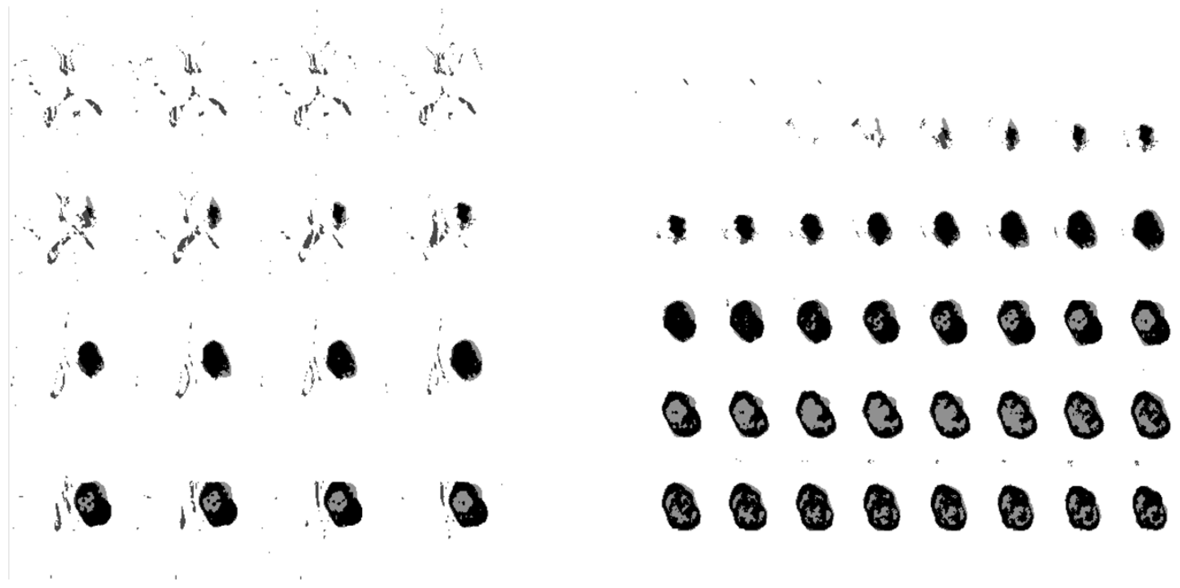


Fig. 11: MRI scan without Post-Processing (left), MRI scan with Post-Processing (right)

5 Conclusion

In conclusion, the automatic detection and segmentation of Brain tumors using the Random Forest Approach yields acceptable results and can well be envisioned in future clinical applications. However the Random Forest must be applied under the right conditions like the right number of volumes used to train and test it. Also other parameters like the size of samples can affect the accuracy of the Forests decision deeply. The future development and application will probably consist of multiple clusters of trees, each trained with individually dedicated volumes. This papers experiment showed that it will be possible to apply Random Forests in the future treatment of tumors.

6 References

Author	**Literature**	**Release**
Zoltán Kapás	Automatic Detection and Segmentation of Brain Tumor Using Random Forest Approach	2016
Ralf Moeller	Vorlesung Einführung in Web und Data Science	2018
A. Criminisi	Decision Forests for Classification, Regression, Density, Esimation, Manifold, Learning and Semi-Supervised Learning	2011
Kahild Usman	Brain tumor classification from multi-modality MRI using wavelets and machine learning	2017
Wikipedia	Random Forest	2020