Identification of Glioma Grade Using Machine Learning

Fabio Blom (4556089), Sterre de Jonge (4477464), Esther van Marrewijk (4531450), Janno Schouten (4476565)
April 11th 2020

Link GitHub: `https://github.com/JSSchouten/TM10007_Group_10.git`

1 Introduction

Gliomas are the most frequent type of intracranial tumor, comprising 81 % of the diagnosed brain malignancies[1]. The location of gliomas varies and the malignant tissue differs in cell type and grade. The Glioma Grade is considered to be an important prognostic factor and ranges from I to IV. Grades I and II are considered low grade glioma (LGG) and have a relatively high survival rate (approximately 47 % over 10 years). A glioma with a III-IV Grade is defined as high grade glioma, or glioblastoma multiforme (GBM) and is much more aggressive, which results in poorer prognosis (median survival of 15 months). Furthermore, Glioma Grade is also an important factor for decision to proceed with surgery. Surgical removal of the glioma is commonly used as treatment, but the surgical procedure depends, among other things, on the Glioma Grade.[2]

The grade may be determined with Magnetic Resonance Imaging (MRI). Varying results have been reported concerning diagnosis of gliomas. I.e. a high classification has been found with MRI for differentiating GBM from LGG, resulting in a sensitivity of 0.81 and 0.87 [2]. However, MRI is still not considered as the optimal 'method' for diagnoses of gliomas. Research has shown that biological specificity of the MRI signal is poor and even expert radiologists have difficulty recognizing the tumor tissue.[3] In this case biopsy of the Glioma tissue is needed for histologic examination. It would however be preferable that biopsy is avoided since this procedure is invasive for the patient. For this reason, a Machine Learning Glioma classifier is developed.

The aim of this research is to differentiate between the tumor grade types LGG and GMB based on features extracted from multiple MRI scans (T2-weighted, T2-weighted FLAIR and T1-weighted before and after administration of a contrast agent) by means of a machine learning classifier with a performance above 80 % mean accuracy.

2 Methods

The data obtained from the multi-centre acquired MRI scans contain two data sets: labeled as LGG and GBM. The GBM dataset contains 103 samples with 725 features and the LGG dataset 66 samples with 725 features. The features mainly provide information about volumes of different tissue. The relatively high amount of features compared to samples causes a dimensionality problem.

2.1 Experimental and evaluation set-up

The data was split into a test set and a training set with a distribution of 20-80%. The training set was consequently used for model optimization using cross-validation. Within cross-validation a distribution of 20-80% for the validation set and the training set was used, as well, resulting in a five fold cross-validation. The decision for these distributions was based on the relatively low number of samples, fewer samples in the validation or test set would negatively influence reliability of the classifier. A full run of the script splits the data 100 times into a training and test set. Within each iteration, the best performing classifier is selected using the validation data set, after which the performance of this optimized model is calculated using the test set. The output of one run is a ranking of the following: prevalence of best picked classifier, prevalence of selected features, overall mean accuracy, mean sensitivity and mean specificity. Performance is measured with correctly classified GBM termed as true positive and correctly classified LGG termed as true negative. The most frequent classifier is chosen to be most suitable for classification of the Glioma Grade.

For adequate performance of the model, preprocessing and model optimization was necessary. This include data adjustment, scaling, feature extraction and/ or selection and settings of classifiers. All choices for these hyperparameters were made within cross-validation, using the training set. The methods with the chosen hyperparameters were then applied on the test set. All values for these hyperparameters can be found in Appendix A.1.

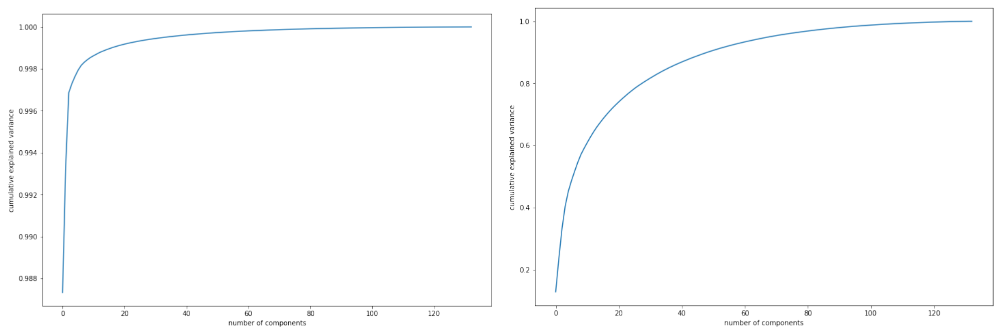
2.2 Preprocessing

2.2.1 Data Adjustments

To begin with, the data is divided into a training and test set. The data contains non numeric values, but numeric values are required for classifiers to be operable. Possible non numeric values that cause errors are NaNs (not a number), strings (words), and infinite numbers. To prevent errors, these values need to be replaced. Firstly, strings and infinite numbers are replaced by NaN. Then, features for which over 50 % of the samples contain NaN, are excluded from further classification. Every other NaN is replaced by the median value of that specific feature. These median values are calculated for the training set and test set separately, to ensure the test set is independent of the training set.

2.2.2 Scaling

The training data is consequently scaled with a standard scaler. Features are scaled to have a normalized range, so they are comparable. A standard scaler was chosen, in which features are scaled with range matching (minimum at 0 and maximum at 1), instead of a robust scaler. With a robust scaler a certain percentage (e.g. 95 %) of the data is scaled on the range. This method would be preferred because it is less sensitive to outliers. However, it was experienced that with this method outliers will play a definitive role in accounting for the variance, which was unwanted. This is shown in Figure 1(a). In this figure the accumulated variance, obtained after applying principal component analysis (PCA), is visualized. To prevent this sensitivity to outliers in the data, standard scaling was applied to the data. The accumulated variance for standard scaling is shown in Figure 1(b). For some features, standard scaling might not work well, since with this method outliers will dominate in the minimum or maximum end of the range and this will highly influence the distribution of the rest of the data. However, with this data set that is not thought to be a problem, since there are many more features than samples and these features will be discarded after feature extraction or selection.



(a) Robust Scaler (b) Standard Scaler

Figure 1: Accumulated variance of PCA (Principal Component Analysis) explained variance after different scaling methods

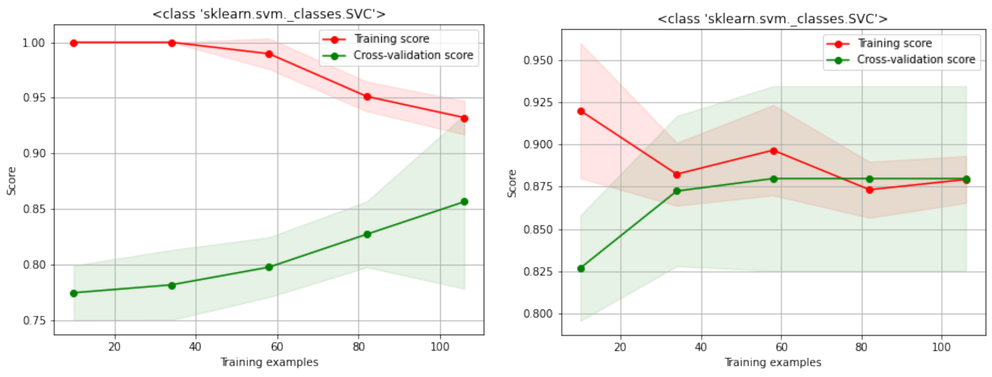
2.2.3 Feature Extraction/Selection

As mentioned before, the data consists of many more features than samples. Too many features is unwanted because that will lead to overfitting. A feature extraction or selection method has to be applied in order to derive a small amount of features that are chosen as such that they can differentiate between the classes the best. First, PCA was applied to extract a certain amount of components that consist of the most variance. The accumulated variance was visualized in Figure 1(b) to determine the amount of components that was thought to be sufficient to classify with. Based on Figure 1(b) it was decided that with 75 % enough variance is covered, which equals $+/-21$ principal components for the training sample set shown here. This hyperparameter (amount of components in PCA) was therefore chosen after inspection of Figure 1(b). A higher percentage level could not be chosen since that would result in too many components for classification.

The second method that was applied is a feature selection method, to be exact: a feature selection method based on univariate statistical testing. This method tests all features separately and selects the features that statistically obtain the highest score for predicting a class. The statistical test that was used for this method is: ANOVA F-value. This is a test that compares numeric data for multiple samples. The variability between group means is compared to the variability within the groups to determine if the means are statistically equal. The hyperparameter of the amount of features to select is optimized using learning curves, which will be discussed in the following section. After optimization the choice was made to use five features.

2.3 Problem Complexity

Assessing problem complexity can be achieved with different methods, there is not one method to measure complexity. Therefore, to begin with, the data was observed. The supplied data is high-dimensional: $Nfeatures >> Nsamples$ and thus, the data is complex. The number of features was therefore reduced to a smaller number. Whether the data can be classified with a complex decision boundary is still unknown and this will be assessed through estimating the complexity with learning curves. A complex decision boundary would generally need more data to learn. Ultimately, different classifiers were chosen to use within model optimization. Learning curves were used within model optimization for several choices. With the learning curves it was tried to achieve generalization, which occurs when the training and the validation curve have converged. The first choice was the selection of the best feature extraction/selection method. In Figure 2, the learning curves are shown for the support vector machine classifier after applying PCA and applying univariate feature selection. With PCA it can be seen that there are not enough training samples to fully learn the classifier. Overfitting takes place for the training set and the validation score remains low. This method does not generalize on the validation set. With univariate feature selection, however, a much better learning curve is obtained because the test and validation set have converged. Based on the learning curves for other classifiers it was chosen to use univariate feature selection method instead of PCA. Other choices that have been made based on the learning curves were choices in the optimization of the hyperparameters of different classifiers, this will be explained in the following section.



(a) PCA (b) Univeriate feature selection

Figure 2: Learning curves of support vector machine classifier after different feature selection methods

2.4 Classification

Finally, for the classification of the data into either LGG or GBM, different algorithms were used in training and validation. The hyperparameters of these algorithms were optimized with the validation set, using both visual evaluation of the previously described learning curve and calculation of accuracy. The classifiers Support Vector Machine (SVM), k Nearest Neighbors (kNN) and Random Forest (RF) were used to train the classifier part of this algorithm. The Bayesian classifier and a linear regression classifier were also considered, but rejected based on the learning curve performance and properties of the data these classifiers require, such as normal distributed data and the existence of a nonlinear relationship. The SVM classifier was initially set with no slack and a linear kernel. Since the accuracy on the training data set was higher than on the validation data set, different combinations of kernels and values of the slack parameter were used. A slack parameter of 0.05 in combination with a linear kernel appeared to be the most suitable on the validation data. The most important kNN classifier hyperparameter is the number of neighbors. The accuracy on the validation set was highest when 9 neighbors were used. The RF classifier with 5 trees, a random state of 42 and a maximum depth of 4 appeared to be the most suitable. All other hyperparameters of the SVM, kNN or RF algorithms were the default settings.

3 Results

In Table 1 the results are shown for a particular run of the model. It can be seen that for 100 iterations the kNN classifier is picked most often (49 %), followed by the SVM classifier (36 %). The least picked classifier is RF (15 %). The mean accuracy of all the best classifiers together combine to 88 %. The specificity is equal to 95 % and the specificity is equal to 76 %. Furthermore, it can be seen that there are four features that have been chosen most often. Within every iteration, five features are used to classify with, of which four features have been chosen (almost) every iteration. The result for each separate iteration of the best classifier and the accuracy, sensitivity and specificity for this classifier is shown in Appendix A.2. A pair plot illustrating the distribution of frequently selected features is shown in Appendix A.3.

Table 1: Results

Outcome	Result
Prevalence of classifiers	Support Vector Machine: 36 % K Nearest Neighbors: 49 % Random Forrest:15 %
Prevalence of selected features	Volume ET over WT: 100 % Volume NET over TC: 100 % Volume ET over TC: 100 % Volume NET over WT: 97 % Solidity NET: 55 % Volume ET: 23 % Volume NET over Brain: 12 % Volume ET over Brain: 11 % Else: 2 %
Results over all iterations	Mean accuracy 87.9 % Sensitivity: 95.1 % Specificity: 76.5 %

Abbreviations: ET: the enhancing part of the tumor core, WT: whole tumor, NET: non-enhancing part of the tumor core, TC: tumor core [4]

4 Discussion

4.1 Model Interpretation and Improvements

4.1.1 Preprocessing

The decision was made to use standard scaling instead of robust scaling, since robust scaling did not function properly using PCA. However, it was later decided to use univariate feature selection instead of PCA, since this method showed better performance on the learning curves. Robust scaling was never applied on univariate feature selection. It may well be possible that robust scaling can be applied before the univariate feature selection, and that this would improve performance.

Univariate feature selection showed better performance than PCA. While assessing cross-validation accuracy and learning curves, it was concluded that selecting five features resulted in the highest accuracy. However, after 100 iterations, it became clear that four features were (almost) always selected. The fifth feature varied. This indicates that selecting four features would be sufficient for identifying the tumor grade. This has to be further investigated.

4.1.2 Problem complexity

With a fairly simple classifier a good performance is achieved. The data seemed complex, at first, because many features were available to describe the samples. After feature selection and classification it is concluded that only five features are needed to classify the data. In Appendix A.3 the distribution of the data for five features is visualized which shows that the data is quite well separated.

4.1.3 Classifier Performance

Based on the results, the kNN classifier is chosen to be the most promising classifier to identify tumor grades of low grade gliomas. This decision is based on the prevalence of the classifier being the best performing of the three. However, in an iteration, classifiers which did not perform best in the training set are neither evaluated in the test set in this method. Meaning that it may well be possible that, on average, another classifier would perform better. This is a disadvantage of this method that has to be kept in mind. In future research i.e. this could be tested by only using the kNN classifier for the 100 iterations in the test set. Also, the hyperparameter optimization was done by hand, while a randomized search could give a better result. Since the chosen outcome was to select the best classifier, the hyperparameter optimization was not performed in the loop but set and validated beforehand. Furthermore, currently, conclusions are drawn without statistical testing. Therefore, in future research statistical testing is necessary to assure reliable results.

4.2 Clinical Relevance

The mean accuracy of 88 % indicates that machine learning is a promising possibility for identifying the tumor grade of low grade gliomas. However, sensitivity and specificity show contradictory results when compared to current practice. The current practice sensitivity and specificity of glioma grade prediction are 81 % and 87 %, respectively. The mean sensitivity of the algorithm is higher (95 %) while the mean specificity is lower(76 %). Meaning that the algorithm over-diagnoses patients having LGG with GBM more often than in current practice.

The feature selection indicated that only five features are necessary to reliably predict tumor grade. Even if the machine learning algorithm would not be implemented in practice, this finding could ease workload for radiologists, since only these five features have to be examined to predict tumor grade.

4.3 Recommendations

The machine learning algorithm is not yet suited for clinical implementation. To achieve this, multiple steps need to be taken first. The results indicate that the kNN classifier is most promising. Therefore, it is recommended to further develop this classifier. More data is needed to further train and test the classifier. Hyperparameters will probably have to be adjusted to increase the classifier's accuracy. When the classifier has been tested sufficiently, clinical trials will have to be performed.

4.4 Conclusion

Since the kNN classifier was the best performing classifier in 49 % of iterations, it can be concluded that this classifier is most promising in correctly predicting the tumor grade of gliomas. Compared to current clinical practice, the algorithm shows conflicting results. The sensitivity of the algorithm is higher (95 % compared to 81 %) while the specificity is lower (76 % compared to 87 %). The classifier may be improved by further training and testing using more data. However, since multiple following steps, which take time and have to meet strict requirements, have to take place. It can be concluded that the clinical implementation of a machine learning algorithm to solely identify the tumor grade of gliomas, is still far away.

References

[1] Ostrom QT, Bauchet L, Davis FG, Deltour I, Fisher JL, Langer CE, et al. The epidemiology of glioma in adults: a "state of the science" review. Neuro-oncology. 2014 Jul;16(7):896–913.

[2] Wang QP, Lei DQ, Yuan Y, Xiong NX. Accuracy of ADC derived from DWI for differentiating high-grade from low-grade gliomas: Systematic review and meta-analysis. Medicine (Baltimore). 2020 Feb;99(8):e19254.

[3] Upadhyay N, Waldman AD. Conventional MRI evaluation of gliomas. Br J Radiol. 2011 Dec;84 Spec No 2:S107–111.

[4] Bakas S, Akbari H, Sotiras A, Bilello M, Rozycki M, Kirby JS, et al. Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features. Sci Data. 2017 09;4:170117.

A Appendix

A.1 Hyperparameters

	Method	**Hyperparameter**	**Value**
Pre-processing	Replace infinites	Replace by	NaN
	Replace strings	Replace by	NaN
	Drop NaN values	Axis to delete	Columns (feature)
	Drop NaN values	Threshold	0.5 * number of samples
	Impute NaN values	Impute with	Median
Scaling	Standard scaler	-	-
Feature selection	Univariate statistical testing	Statistical test	F-ANOVA
	Univariate statistical testing	Number of features	5
Cross validation	Cross validation	Number of cross validations	5
Problem complexity	Learning curve	-	-
Classifier	Support Vector Machine	Kernel	Linear
	Support Vector Machine	Slack parameter	0.05
	K Nearest Neighbors	Number of neighbors	9
	Random Forest	Number of trees	5
	Random Forest	Random state	42
	Random Forest	Maximum depth	4
Evaluation	Split data with random selection	Test size	20%
	With test set	Number of iterations	100
	Performance	Score	Mean accuracy

A.2 All Results

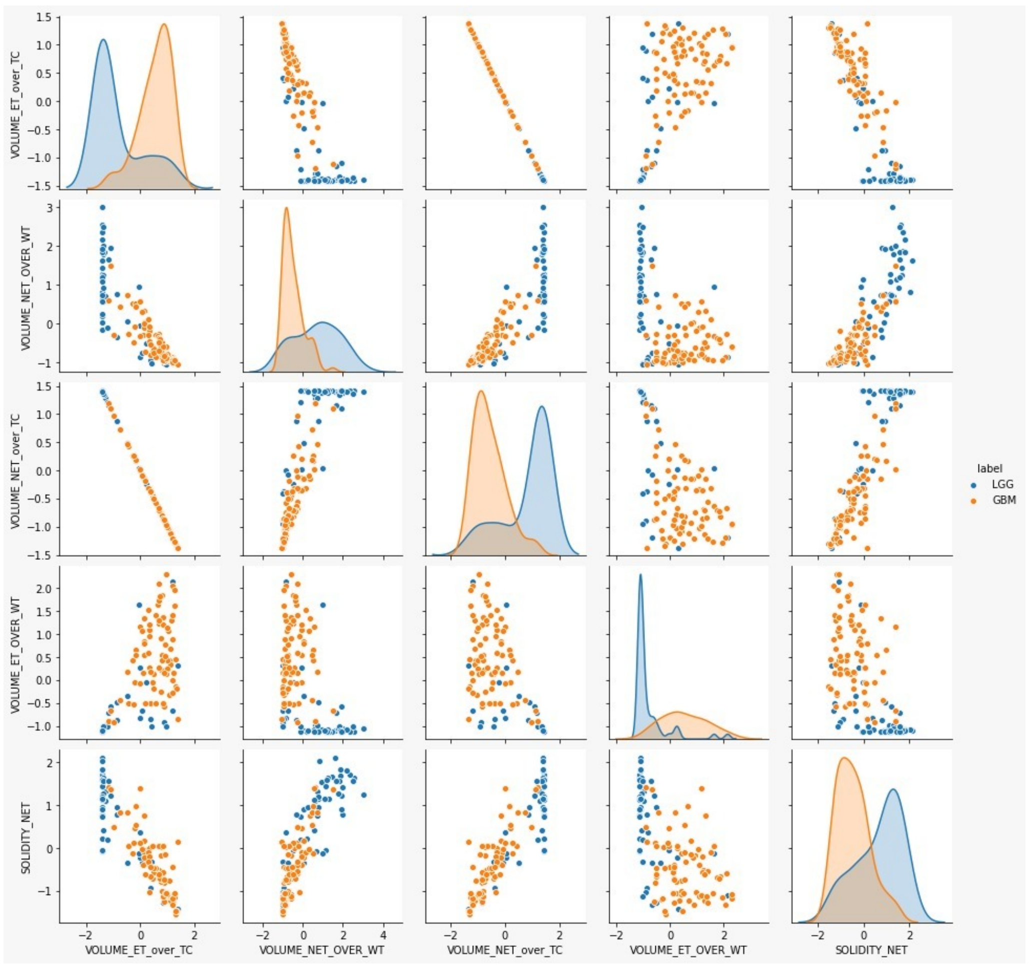
	Classifier	Accuracy	Sensitivity	Specificity
0	K Nearest Neighbors	0.8235294117647058	0.8823529411764706	0.7647058823529411
1	Support Vector Machine	0.8823529411764706	0.9523809523809523	0.7692307692307693
2	Random Forest	0.9117647058823529	1.0	0.7692307692307693
3	Support Vector Machine	0.9117647058823529	0.95	0.8571428571428571
4	Random Forest	0.7941176470588235	0.8636363636363636	0.6666666666666666
5	Support Vector Machine	0.8823529411764706	0.9130434782608695	0.8181818181818182
6	Support Vector Machine	0.8235294117647058	0.9	0.7142857142857143
7	Random Forest	0.8823529411764706	0.9	0.8571428571428571
8	Support Vector Machine	0.9411764705882353	0.9545454545454546	0.9166666666666666
9	Support Vector Machine	0.8823529411764706	1.0	0.6
10	K Nearest Neighbors	0.7058823529411765	1.0	0.4444444444444444
11	K Nearest Neighbors	0.8823529411764706	1.0	0.7142857142857143
12	Support Vector Machine	0.9117647058823529	0.9130434782608695	0.9090909090909091
13	K Nearest Neighbors	0.7941176470588235	0.95	0.5714285714285714
14	K Nearest Neighbors	0.8529411764705882	0.9545454545454546	0.6666666666666666
15	K Nearest Neighbors	0.8235294117647058	0.95	0.6428571428571429
16	K Nearest Neighbors	0.8529411764705882	0.9090909090909091	0.75
17	K Nearest Neighbors	0.8823529411764706	0.9047619047619048	0.8461538461538461
18	K Nearest Neighbors	0.9117647058823529	0.9523809523809523	0.8461538461538461
19	Support Vector Machine	0.8529411764705882	0.9545454545454546	0.6666666666666666
20	Random Forest	0.8529411764705882	0.9	0.7857142857142857
21	K Nearest Neighbors	0.8823529411764706	0.9583333333333334	0.7
22	K Nearest Neighbors	0.8235294117647058	0.9523809523809523	0.6153846153846154
23	Support Vector Machine	0.8823529411764706	0.95	0.7857142857142857

24	K Nearest Neighbors	0.9411764705882353	1.0	0.8333333333333334
25	Support Vector Machine	0.8529411764705882	0.9090909090909091	0.75
26	K Nearest Neighbors	0.9117647058823529	0.9444444444444444	875
27	K Nearest Neighbors	0.9117647058823529	1.0	0.75
28	Support Vector Machine	0.9117647058823529	1.0	0.8
29	Random Forest	0.9117647058823529	0.8846153846153846	1.0
30	K Nearest Neighbors	0.9117647058823529	0.95	0.8571428571428571
31	K Nearest Neighbors	0.9411764705882353	0.9583333333333334	0.9
32	Random Forest	0.8235294117647058	0.8333333333333334	0.8125
33	K Nearest Neighbors	0.9117647058823529	1.0	0.75
34	Support Vector Machine	0.9117647058823529	1.0	0.7857142857142857
35	Support Vector Machine	0.8529411764705882	0.9130434782608695	0.7272727272727273
36	K Nearest Neighbors	0.9117647058823529	0.9523809523809523	0.8461538461538461
37	K Nearest Neighbors	0.8823529411764706	0.9523809523809523	0.7692307692307693
38	Support Vector Machine	0.8235294117647058	0.8571428571428571	0.7692307692307693
39	Support Vector Machine	0.9411764705882353	0.9583333333333334	0.9
40	K Nearest Neighbors	0.7647058823529411	0.9375	0.6111111111111112
41	K Nearest Neighbors	0.8823529411764706	1.0	0.6
42	Support Vector Machine	0.9117647058823529	1.0	0.7
43	K Nearest Neighbors	0.9705882352941176	1.0	0.9333333333333333
44	Support Vector Machine	0.8823529411764706	1.0	0.7142857142857143
45	K Nearest Neighbors	0.9411764705882353	0.9545454545454546	0.9166666666666666
46	K Nearest Neighbors	0.9117647058823529	1.0	0.7692307692307693
47	Support Vector Machine	0.9411764705882353	0.9545454545454546	0.9166666666666666
48	Support Vector Machine	0.9411764705882353	0.9090909090909091	1.0
49	Random Forest	0.8823529411764706	0.9	0.8571428571428571

50	Support Vector Machine	0.8529411764705882	0.9090909090909091	0.75
51	Support Vector Machine	0.8235294117647058	0.9047619047619048	0.6923076923076923
52	Support Vector Machine	0.9117647058823529	1.0	0.8
53	Random Forest	0.9117647058823529	0.9473684210526315	0.8666666666666667
54	K Nearest Neighbors	0.9411764705882353	1.0	0.8181818181818182
55	K Nearest Neighbors	0.8235294117647058	0.9565217391304348	0.5454545454545454
56	Support Vector Machine	0.9705882352941176	1.0	0.8888888888888888
57	K Nearest Neighbors	0.8823529411764706	0.8888888888888888	875
58	Support Vector Machine	0.8529411764705882	875	0.8
59	Support Vector Machine	0.8823529411764706	0.9565217391304348	0.7272727272727273
60	K Nearest Neighbors	0.8235294117647058	1.0	0.6470588235294118
61	Random Forest	0.8529411764705882	0.9130434782608695	0.7272727272727273
62	K Nearest Neighbors	0.9411764705882353	0.9523809523809523	0.9230769230769231
63	K Nearest Neighbors	1.0	1.0	1.0
64	K Nearest Neighbors	0.9705882352941176	1.0	0.9090909090909091
65	Support Vector Machine	0.8235294117647058	0.9523809523809523	0.6153846153846154
66	K Nearest Neighbors	0.9117647058823529	1.0	0.8333333333333334
67	Support Vector Machine	0.7941176470588235	0.9565217391304348	0.45454545454545453
68	Support Vector Machine	0.8823529411764706	1.0	0.6666666666666666
69	Support Vector Machine	0.8529411764705882	0.96	0.5555555555555556
70	Support Vector Machine	0.9117647058823529	1.0	0.7692307692307693
71	Support Vector Machine	0.8823529411764706	1.0	0.6363636363636364
72	Random Forest	0.8529411764705882	0.9047619047619048	0.7692307692307693
73	K Nearest Neighbors	0.8529411764705882	0.9130434782608695	0.7272727272727273
74	K Nearest Neighbors	0.9117647058823529	0.95	0.8571428571428571

75	Support Vector Machine	0.9117647058823529	0.9411764705882353	0.8823529411764706
76	Support Vector Machine	0.8529411764705882	1.0	0.6666666666666666
77	K Nearest Neighbors	0.7352941176470589	0.95	0.42857142857142855
78	K Nearest Neighbors	0.9411764705882353	1.0	0.8666666666666667
79	K Nearest Neighbors	0.9411764705882353	1.0	0.8461538461538461
80	K Nearest Neighbors	0.9411764705882353	1.0	0.8571428571428571
81	K Nearest Neighbors	0.7941176470588235	1.0	0.5333333333333333
82	K Nearest Neighbors	0.7941176470588235	0.9285714285714286	0.7
83	Support Vector Machine	0.8235294117647058	0.85	0.7857142857142857
84	K Nearest Neighbors	0.9411764705882353	1.0	0.8333333333333334
85	Support Vector Machine	0.8823529411764706	0.9523809523809523	0.7692307692307693
86	K Nearest Neighbors	0.8529411764705882	1.0	0.6153846153846154
87	K Nearest Neighbors	0.9705882352941176	1.0	0.9333333333333333
88	K Nearest Neighbors	0.9117647058823529	1.0	0.7272727272727273
89	K Nearest Neighbors	0.8529411764705882	0.9473684210526315	0.7333333333333333
90	Random Forest	0.9411764705882353	0.9545454545454546	0.9166666666666666
91	K Nearest Neighbors	0.8529411764705882	0.84	0.8888888888888888
92	Random Forest	0.8235294117647058	0.9473684210526315	0.6666666666666666
93	Random Forest	0.7941176470588235	875	0.6
94	Random Forest	0.8823529411764706	0.9523809523809523	0.7692307692307693
95	K Nearest Neighbors	0.8529411764705882	0.9523809523809523	0.6923076923076923
96	K Nearest Neighbors	0.8235294117647058	0.9523809523809523	0.6153846153846154
97	Random Forest	0.9117647058823529	0.9	0.9285714285714286
98	K Nearest Neighbors	0.8823529411764706	0.9523809523809523	0.7692307692307693
99	Support Vector Machine	0.8823529411764706	0.9523809523809523	0.7692307692307693

A.3 Pairplot


VOLUME_ET_over_TC
VOLUME_NET_OVER_WT
VOLUME_NET_over_TC
VOLUME_ET_OVER_WT
SOLIDITY_NET
label
LGG
GBM

A.4 Reflection

A.4.1 General planning, communication strategy, division of roles

From the start we frequently met up to discuss the lectures and the general assignments. Because of the current situation we did not meet up in personal, but via Teams. Communication took place mainly during Microfoft Teams sessions but also via WhatsApp. We did not start with our group assignment immediately, since we had to await certain lectures before we could effectively start. We started in the third week on the assignment. We mainly got together and worked on the code together. One person shared his/ her screen and we could all watch the changes that the person made to the code. The other team members simultaneously discussed the plan of action and the choices that had to be made. After a session together we discussed things that could be prepared for the next session, for instance parts that had to be written for the paper or some things of the code that could be investigated.

A.4.2 Fabio Blom

There was a very good collaboration between the group members. We often had Microsoft Teams meetings to discuss and make a plan. During these meetings we often worked on google colab together to make the python code working, by sharing screen. In this way everyone was sharing ideas about implementation of the code, while all group members were immediately informed about new adjustments. This way of working gave everyone space to give there input in the python code. However, I personally did not consider myself the best in python coding, although I do understand everything that was eventually implemented. I focused more on how several concepts had to be implemented in the general code. For example I gave my input about how the cross-validation should be elaborated and implemented in the model, while other group members, like Janno, were more focused on writing this concept in hardcore Python code. Furthermore I was involved in debating on certain important decisions in the model. In line with the elaboration on google colab, division in tasks the report of the research was well defined. Every person wrote out certain parts of the text. Thereafter, I focused on checking and rewriting the text together with Sterre and Esther. Janno and Esther made sure all the lay out of text and results were properly elaborated in LaTeX.

A.4.3 Sterre de Jonge

I am really positive about the teamwork within this group. I am satisfied with the result that we managed to achieve, and I enjoyed the teamwork since we worked very efficient with each other. I was a bit anxious at first that we did not start right away on the assignment, but in the end, it perfectly worked out. We mainly worked together on the code, which I found to be effective. Since we could not meet up in real-life, we communicated via Teams and WhatsApp. One team member shared his/ her computer screen and we could all watch the changes that this person made to the code. The other team members discussed the plan of action and the choices that had to be made. I am myself not the best in programming, but I am critical and precise, which I used in my contribution to the code. I spent individually quite some time on the lectures, to fully understand the principles of machine learning, this understanding I used to contribute in discussion on choices that had to be made. I enjoyed the discussions that we had, because that helped me in understanding the topics better. My contribution in this project was, furthermore, in writing on the report. I have written a part of the methods, the results and I spent time on checking and improving other peoples' work. For me this cooperation worked really well because I am very fond of effectively working together. However, a negative aspect might be that for some this cooperation does not work well because of personal preferences. For a future collaboration I think it would be better to pay more attention to this.

A.4.4 Esther van Marrewijk

The communication between the group members was quite good and efficient. At the beginning of the project I created a link to a Microsoft Teams meeting, so our group meetings could be held using that platform. Next, my contribution in this project consisted of creating an overview of possible ways to approach the different steps of the machine learning algorithm, based on the lectures and interpretations from the other group members. When we had this overview, we argued amongst ourselves which strategies would be most suitable for our problem. This contributed to an overall better-founded strategy and understanding of the lectures and theory.

Naturally, this initial strategy had to be adjusted several times during the project, but always in good agreement from all group members and with clear reasoning. The coding itself was done during our Microsoft Teams meetings, in which one person shared his/her screen and ran the code. The other group members sought for ways to implement our strategy using available functions on the internet or from the lecture exercises. Most of the times, my contribution was in the latter, since I enjoy troubleshooting and I intuitively understand the hyperparameters of some functions after reading short explanations on the internet. Finally, the report was made in LaTeX, in which Janno and I provided the formatting of the document. Moreover, some parts of the method section and some parts of the discussion were written by me and in the end I read and improved other peoples' writings. In conclusion, a completely digital collaboration would not be my method of choice but it worked surprisingly well.

A.4.5 Janno Schouten

I believe that my role during this project was focused on writing the python code. This was the part of the project that I enjoy doing most, which may have led to me working on the code more than others. Writing this, I need to emphasize that I do not mean that I wrote the majority of the code by myself. Most of the code was written during video call sessions. I did spend more time than others on tidying up the code and implementing different parts of the code into functions. This was especially important for the script to be able to iterate a chosen amount of times. I spent less time in writing the report. Other group members started focusing on this part of the project earlier than I did. I did write part of the discussion together with Esther, with adjustments and additions of Fabio and Sterre. Also, I started implementing the code into LaTeX. Esther contributed with finalizing the formatting of the report in LaTeX. To finalize the report, we all read each other's writings and commented on them. In my opinion, collaboration during this project went well. Decisions were taken together, and writing of the code was performed together for the majority of the time as well. Group members which had more difficulty writing code started working on the report earlier. I enjoyed working together with this group of people and would like to do it again for another project.

A.4.6 Conclusion

As stated stated in all personal reflections, everyone was satisfied with the collaboration during this project. Even with the limitations caused by the corona virus, we managed to communicate well and work together as a team. We even stated that programming together might actually work better sharing screens compared to everyone working on their own screen. We did not divide different tasks with clear agreements, but this process occurred naturally during the project. In this particular project that worked well, however in future projects it might be better to divide tasks more clearly. We are all proud of the results we managed to achieve and would like to work together with this group sometime in the future.