FYS-STK4155 - Project 3

A comparison of classification algorithms on the Wisconsin Breast Cancer data

Hannah Lovise Okstad Ekeberg & Idunn Aamnes Mostue

University of Oslo

December 2019

Abstract

"Every second, the deforestation of the Earths rain forests is said to be of the size of a football field. Contrary, in machine learning random forests are certainly growing"

— Hannah Lovise Ekeberg, Idunn Aamnes Mostue

In this project, we have analyzed the Wisconsin Breast Cancer data by application of different machine learning algorithms to see what method performed best for optimal classification of breast cancer diagnosis. We wanted to classify whether a tumor is benign or malignant based upon different features. We included the classification algorithms of Neural Network, Decision Trees, and Decision Trees with added bagging (Random Forest) and boosting (AdaBoost). Neural Network showed the best performance with an accuracy score of 0.9737 on the validation data, and the Decision Tree performed worst with an accuracy score of 0.9474 on the validation data. Random Forest and Adaboost performed equally well with an accuracy score of 0.9561 on the validation data. However, Random Forest did a better job in predicting class benign with a recall of 1.0, while AdaBoost had a recall of 0.9861. For the malignant class, Random Forest had a recall of 0.8810, while AdaBoost had a recall of 0.9048, meaning that AdaBoost did slightly better in classifying true malignant tumors. Neural Network had a recall of 0.9286 for the malignant class which means that Neural Network had the overall best performance in this classification problem.

All code and data used in this project can be found at:
https://github.com/HannahEkeberg/FYS-STK4155/tree/master/Project3

Contents

1 Introduction

One in four new cases of cancer in women is diagnosed as breast cancer, making it the most common form of cancer for the female gender. The World Health Organization predicted in 2018, approximately 2.1 million new incidents of breast cancer worldwide. It was also ranked as the fifth leading cause of death for women[1]. However, the five-year relative survival rate can be increased if discovered at an relatively early stage[2]. Screening through mammography, clinical breast exam and breast self-exam are well known methods for detecting breast cancer[3], but the use of machine learning techniques in healthcare analysis is growing progressively[4]. With increasing quality of medical imaging, such as PET and MRI, as well as data processing of the images over the recent years, the combination with machine learning techniques can help us in deciding whether a tumor should be classified as malignant or benign, based on different features such as radius, texture and smoothness[5].

Through this project we wanted to use the Wisconsin Breast Cancer Data in analysing the classification performance of different machine learning algorithms. We have evaluated how well Artificial Neural Network, Decision Trees, and Decision Trees with boosting and random forest ensemble methods, manage to predict the diagnosis of each incidence of the data set. The article by Asri et al.[6] was used as inspiration for the layout of this project.

This report proceeds in six chapters. Chapter 2 provides theory on decision trees, and optimization through ensemble methods. In chapter 3 we introduce the Wisconsin Breast Cancer Data, and explain the application of the machine learning algorithms used for our analysis. Further, in chapter 4 we present our main findings, and proceeds with a discussion of our main findings in chapter 5. Lastly, in chapter 6 we give a conclusion of the project.

2 Theory

This project aims to explore how different machine learning algorithms perform on a classification task. Theory on Artificial Neural Network has previously been discussed in project 2[1]. In this chapter we introduce some theoretical background on Decision Tree learning and growing a classification tree in section 2.1. Further, in section 2.2 we touch on how we can increase the accuracy of decision trees through various ensemble methods.

2.1 Decision trees

The following section is primarily based on the lecture notes *"Data Analysis and Machine Learning: From Decision Trees to Forests and all that"* by Morten H. Jensen[7].

Decision Tree learning are supervised machine learning methods of which can be used both for regression and classification problems. In this project we will be focusing on the latter.

With an upside down structure, the decision trees final goal is to predict some conclusion for the target features based on some descriptive features. For a set of data containing descriptive features and target features, we want to model and train the decision tree by finding and splitting the descriptive features containing the most information regarding the target feature. If a descriptive feature points towards one single target, we call the feature *pure*, otherwise *impure*. The descriptive feature with the highest purity is said to contain the most information. We define this particular target as the *root node*, and selected it to go on the top of the tree structure. When growing the tree, we split the data into *interior nodes*. Each node specifies a test of some attribute of the instance. The tree grows until a stopping criteria is accomplished, i.e once a maximum depth or maximum purity is reached. The end node is called a *leaf node*, which gives a classification of the given instance. The nodes are connected through what we call *branches*, giving possible values of an attribute.

[1]For theory on Artificial Neural Network, please see project 2 at https://github.com/HannahEkeberg/FYS-STK4155/tree/master/Project2

Recursive Binary Splits

Decision trees uses a top-down approach in splitting the nodes, starting at the root node working its way down to the leaf nodes. Using a multiway split is not a good idea, as this will separate the data too quickly, and we are left with insufficient data at the next step down [8]. Therefore, recursive binary splits are used. This is a greedy approach where each node performs the *best* split at each step, rather than looking at what may be the best split in future steps.

For a classification problems with $k = 1, 2, ..., K$ target values, we need to define some splitting criteria for each node. The goal is for each leaf node to represent a set of data points that are of the same class, making the purity of the node as high as possible. For a the majority class k of N_m observations in a region R_m, we define a probability density function, P_{mk}, representing how many target variables equals the majority class k, $I(y_i = k)$.

$$P_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k) \tag{1}$$

There are multiple ways of evaluating the quality of each split, but for this project we will be using two of the most common ways of splitting nodes, namely the *Gini index, g,* and *Entrophy, s.*

$$g = \sum_{k=1}^{K} P_{mk}(1 - P_{mk}) \tag{2}$$

$$s = -\sum_{k=1}^{K} P_{mk} \log P_{mk} \tag{3}$$

For the set-up of our decision trees, we will be exploring two of the most used algorithms, being the *Classification and Regression Trees-* algorithm (CART), and the *ID3*-algorithm.

CART-algorithm

The CART-algorithm can be used for both regression and classification trees. For this algorithm the data is split into two subsets using a single feature k and a threshold t_k. Here, we wish to find the pair of k and t_k giving the purest subsets. In order to find these optimal values, a cost function (4) is constructed so that the purest subset is achieved,

$$C(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}} \tag{4}$$

where m is the number of instances, G is the *Gini Factor* measuring the impurity of the *left/right* subset. Once the training set is split, the subsets are split using the same method.

ID3-algorithm

The ID3-algorithm is only used for classification purposes. From top-down, the tree is grown through selecting what attributes to use at each node in the tree. Information gain, calculated through the entropy equation (3), measures how well a given attribute separates descriptive data according to their target classification. The best fitted attribute is then selected, and by repeating the process new subsets occur moving down the tree. If the subset is pure within a threshold, it is left as a leaf node.

2.2 Ensemble Methods

Decision Trees are beneficial methods in that they can be applied on both continuous and categorical problems. They are also very intuitive in their interpretation. However, implementation of trees also calls for some problems. Firstly, they usually generalize the data poorly as they tend to overfit to the training data. This often cause in a high variance[7]. Secondly, this high variance also results in high instability in the tree, where small changes in the data will result in very different sequence of splits[8].

The performance of the Decision Trees can be improved by applying bagging and boosting.

Bagging and Random Forest

Bagging (Bootstrap aggregation) is a method which uses the bootstrap resampling method[2]. The method divides the data into subsets (bootstraps), where the data is random and can be listed several times in each bootstrap. Each subset is divided into train and test data, and fitted towards a model. Using bagging on a decision tree, we get different trees that perform slightly different. This method typically results in improved accuracy, as well as reduction in variance.

Random Forest - it is said that *"if a tree is good, then a forest should be better, provided that there is enough variety in them"* [9]. Random Forest is a method which uses bagging to train the trees, but also limits the choices at the splits by restricting the number of predictors available. The sample of predictors typically contains the square root of the total number of predictors, p:

$$m \approx \sqrt{p} \tag{5}$$

Random Forest allows us to reduce variance without affecting the bias. This is a clever method as there might be a very strong predictor, causing that almost every tree will use this specific predictor as its top split. In such a case, the trees will look similar and have high correlation, which again does not give a reduction in variance.

Boosting and AdaBoost

The principle of boosting is to take a collection of poor classifiers where the error rate is slightly better than a random guess, and then putting them together to make a well-performing classifier. While bagging uses independent classifiers to find the best tree, boosting uses sequential classifiers. The principal algorithm is called *AdaBoost* (adaptive boosting). It is an iterative technique, where each datapoint is assigned with a weight. Initially, all weights are set to set same value, 1/N, where N is the number of datapoints. At each iteration, a new classifier is trained on the training set, and the weight applied for the datapoint is modified dependent on how well the classification have been previously, given a lower weight for low correct classification rate. For each iterations, the error, ϵ, is computed, and the cases where classification fails, the weight are updated, for instance by multiplying with $\alpha = (1-\epsilon)/\epsilon$ ([9], p. 269).
Given a set of data $\{x_i, y_i\}$, where $i = 1, 2, ..., N$, and y_i has a binary outcome. The error rate of the training sample is then given by the following equation

$$\epsilon = \frac{1}{n}\sum_{i=1}^{N} I(y_i \neq G(x_i)) \tag{6}$$

where $G(x)$ is the classification function. The boosting method is used to create a weak classification algorithm $G_m(x)$, where $m = 1, 2, ..., M$ for each iteration, for slightly different samples of the data. Our final classifier will be a combination of the weak classification algorithms along with the weights:

$$G_M(x) = sign\sum_{i=1}^{M} \alpha_m G_m(x) \tag{7}$$

where α_m is the weight of classification G_m.

[2]For theory on bootstrap, please see project 1 - *Regression Analysis and Resampling Methods*

3 Methods

This chapter give a description of the breast cancer data used and explains the updates made for the analysis in section 3.1. In section 3.2 we introduce the algorithms and associated parameters used in the analysis. Section 3.3 show the tuning of the parameters. Lastly, section 3.4 give an overview and description of our source code.

3.1 Wisconsin Breast Cancer Data

In this project we study the Wisconsin Breast Cancer Data[3] produced by Dr. Holberg et al. 1995. The data consist of 569 instances with features computed from digitalized images of fine needle aspirate (FNA) of breast masses. The dataset provides one column of id-numbers, one column of diagnosis (marked M for malignant or B for benign), and 30 columns of real-valued attributes such as texture, perimeter, area and smoothness. The data included no missing values.

We used the 30 real-valued variables as our independent descriptive variables X, and considered the variables of diagnosis as our target variable Y. We checked for high correlation within the independent variables, and detected and deleted seven of the 30 variables showing correlation greater than 0.95. This included *mean perimiter*, *mean area*, *perimiter error*, *area error*, *worst radius*, *worst perimiter* and *worst area*, leaving us with 23 independent variables of X, listed in table 1. Thereafter, we explored the balance of the target data presented in figure 1. Here we see an imbalance in the target data with 212 instances of malignant and 357 instances of benign, giving a ratio of 37:63. Working with unbalanced data may cause the model to overfit for the target value in majority, thus should be further tested.

Table 1: Independent variables, X, with correlation less than 95%, used in the analysis.

1. mean radius	9. radius error	17. worst texture
2. mean texture	10. texture error	18. worst smoothness
3. mean smoothnes	11. smoothnes error	19. worst compactness
4. mean compactness	12. compactness error	20. worst concavity
5. mean concavity	13. concavity error	21. worst concave points
6. mean concave points	14. concave points error	22. symmetry
7. mean symmetry	15. symmetry	23. worst fractial dimension
8. mean fractial dimension	16. fractial dimension error	

The target values were converted from strings to binary number, making 0=malignant and 1=beningn, using Pandas *get_dummies*[4] function. As our data may contain features highly varying in magnitude, we standardized our independent features using *StandardScaler*[5] form ScikitLearn, as most machine learning models require standardization of the data.

3.2 Algorithms

ScikitLearn's DecisionTreeClassifier[6] was used to make a decision tree. The depth of the tree, criterion of the splitting mechanism, the minimum number of samples to split an internal node, and the minimum number of samples required to be at the leaf node, were tuned to find the values giving the highest accuracy score. The other variables were set to default.

ScikitLearn's RandomForestClassifier[7] was used to make a random forest. The number of trees, criterion of

[3]The Wisconsin Breast Cancer data can be extracted from https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

[4]Description of Pandas get$_{d}ummiesinbuiltfunctioncanberevisedathttps : //pandas.pydata.org/pandas - docs/stable/refer$

[5]Description of ScikitLearns inbuilt class of StandardScaler can be revised at https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

[6]Description of ScikitLearns inbuilt class of DecisionTreeClassifier can be revised at https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[7]Description of ScikitLearns inbuilt class of RandomForestClassifier can be revised at https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

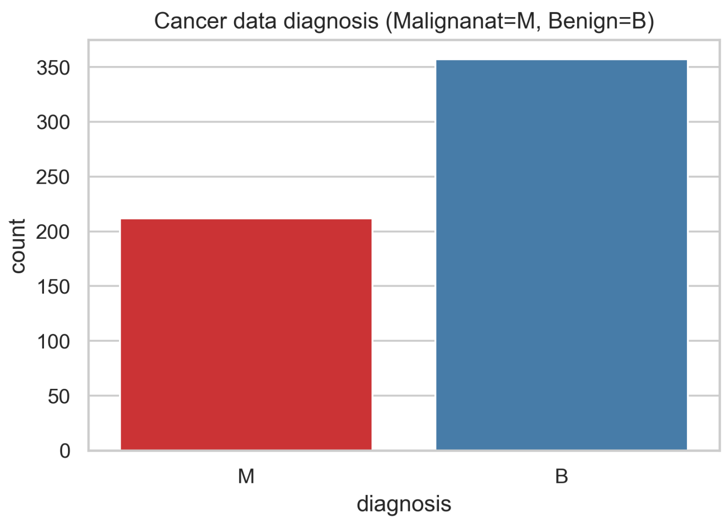


Figure 1: Counter plot of the balance of diagnosis in the target classes. The figure shows an imbalance between the number of M = Malignant, and B = benign, with an 37:63 ratio.

the splitting mechanism, the depth of the trees, the minimum number of samples to split an internal node, and the minimum number of samples required to be at the leaf node, were tuned to find the values giving the highest accuracy score. The other values were set to default.

ScikitLearn's AdaBoostClassifier[8] was used to make the AdaBoost classification algorithm. The number of trees, learning rate, and base estimator were tuned to find the values giving the highest accuracy score. For the base estimator, where the boost ensemble is built, we used DecisionTreeClassifier with different tree depth as values. The rest of the AdaBoostClassifier algorithm was set as default, using SAMME.R-algorithm.

ScikitLearn's MLPClassifier[9] was used to make a Neural Network classification. Hidden layer sizes, activation function and learning rate were tuned to find the values giving the highest accuracy score. Solver was set as default using *adam*, referring to a stochastic gradient-based optimizer.

3.3 Tuning of hyperparameters

For each of the methods analyzed, parameters were tuned using ScikitLearn's GridSearchCV[10] function. The function evaluates parameters, which are sorted in a parametric grid (typically a dictionary with variables as key and a list of values as element). It returns the values which gives the highest accuracy score. The following parameters where used in the search for the highest accuracy for each algorithm.

· Decision tree:

  max_depth: 2, 3, 5, 10, 15, 20, 30
  criterion: 'gini', 'entropy'
  min_samples_leaf: 0.25, 0.5, 1, 2, 3
  min_samples_split: 0.1, 0.5, 2, 4, 6

· Random forest:

  n_estimators: 10, 30, 70, 150
  max_depth: 2, 3, 5, 10, 15, 20, 30

[8] Description of ScikitLearns inbuilt class of AnaBoostClassifier can be revised at https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

[9] Description of ScikitLearns inbuilt class of MLPClassifier can be revised at https://scikit-learn.org/stable/modules/generated/sklearn.neural$_n$*etwork.MLPClassifier.html*

[10] Description of ScikitLearns inbuilt class of GridSearchCV can be revised at https://scikit-learn.org/stable/modules/generated/sklearn.model$_s$*election.GridSearchCV.html*

criterion: 'gini', 'entropy'
min_samples_leaf: 0.25, 0.5, 1, 2, 3 min_samples_split: 0.1, 0.5, 2, 4, 6

· AdaBoost:
  n_estimators: 10, 30, 70, 150
  learning_rate: 0.0001, 0.001, 0.01, 0.1, 1.0
  base_estimator: DecisionTreeClassifier with max_depth = 1, 2, 5, 10, 20, 30

· Neural network:
  max_iter: 10, 20, 50, 100, 200, 400
  hidden_layer_sizes: 1, 4, 10, 20, 50, 100, 250
  learning_rate_init: 0.001, 0.01, 0.1, 1.0
  activation: 'logistic', 'tanh', 'relu'

3.4 Source code

1. main.py - *All code is ran through this program.*
2. get_data.py - *Reads data, scales (using Sklearn's StandardScaler(), splits data into training and test data and drops highly correlated features, if desired.*
3. sklearn_methods.py - *Contains Sklearn's methods for decision tree, random forest, bagging and neural network.*
4. analyse.py - *Generate scores to see how well model performs compared to data. Contains accuracy, false positive and true positive rates (and area under curve of plotted rates), and confusion matrix. Does also contain an importance evaluation of predictors.*
5. grid_searchCV.py - *Finds hyperparameters for each method that return the highest accuracy score.*
6. tuning.py - *Hyperparameters for different sklearn methods from sklearn_methods.py are valued based upon accuracy scores. Calculates accuracy score for multiple hyperparameters. Not used directly in analysis, but good for visualization.*

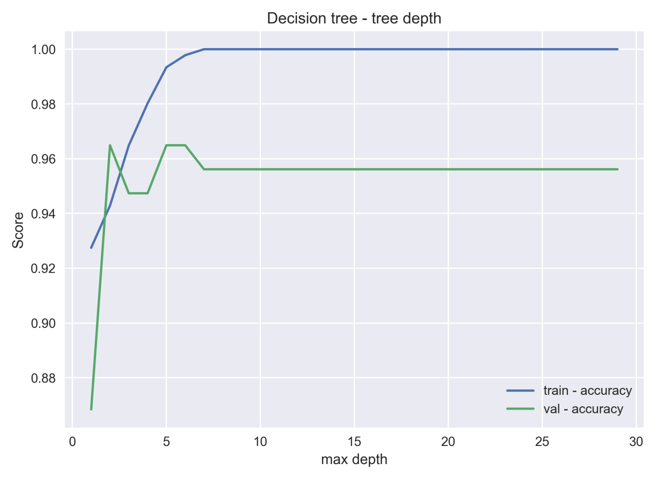
4 Results

Through the analysis of the different classification algorithms we obtained the following results.

Figure 2 shows the accuracy score for decision tree depth (left) and neural network iterations (right). For the accuracy score of the decision tree algorithm, we see the training data increases with maximum tree depth, before it stabilizes at a perfect accuracy score of 1.0 at about seven steps, where as the validation data stabilizes at an accuracy score of about 0.95 at seven steps. For the accuracy score of the neural network algorithm, we see that the train and validation data follow each other closely increasing with increasing maximum iterations, before the training data stabilizes at an accuracy score of 1.0, just above the validation data stabilizing around 0.97, after about 100 iterations.

For each algorithm, the tuned parameters are listed in table 2 for Decision Tree, Random Forest and AdaBoost and in table 3 for Neural Network.

Accuracy and area under ROC curve (ROC curve can be seen in figure 3) are listed in table 4.



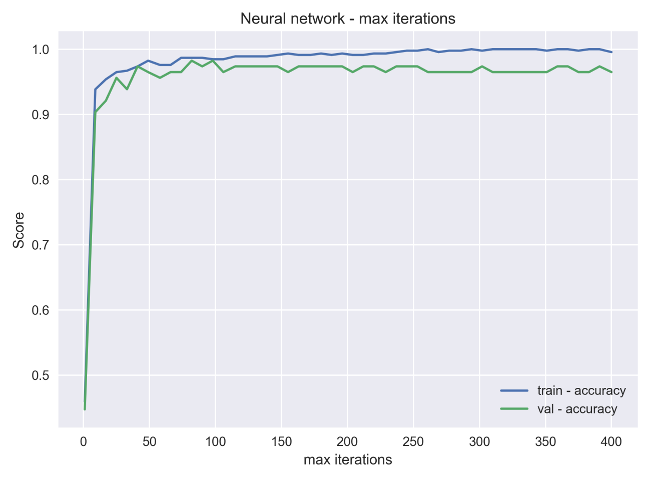


Figure 2: Plots of the accuracy score versus the maximum tree depth for Decision tree (left), and versus the maximum interations for Neural network (right).

Table 2: Tuned parameters of tree classifiers
* indicates default values.

Method	Algorithm	Estimators	Depth	Min samples split	Min samples leaf	Learning rate
Decision tree	Gini	1	3	2	2	-
Random forest	Entropy	150	10	2	2	-
AdaBoost	Gini*	200	1	2*	1*	0.1

Table 3: Tuned parameters of Neural Network classifiers.

Method	Activation function	max iterations	learning rate	hidden layer sizes
Neural network	relu	100	0.1	20

Table 4: Accuracy and AUC

	accuracy		area under curve	
Classifier	Train	Val	Train	Val
Decision Tree	0.9648	0.9474	0.9577	0.9335
Random Forest	0.9978	0.9561	0.9971	0.9405
AdaBoost	0.9978	0.9561	0.9971	0.9454
Neural Network	1.0	0.9737	1.0	0.9573

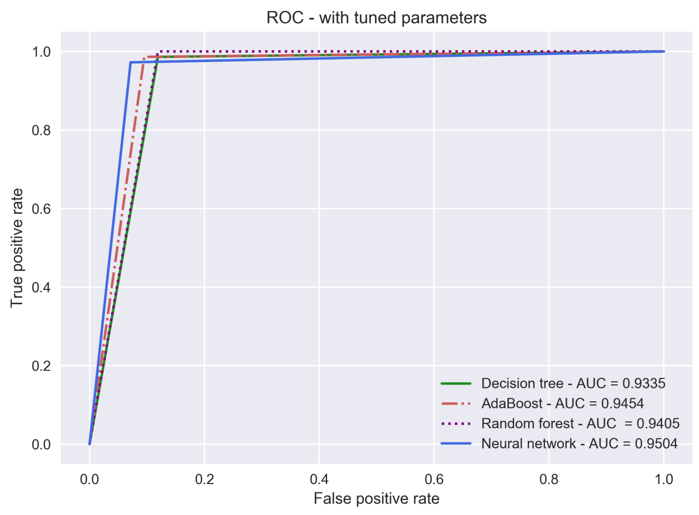


Figure 3: ROC-curve with tuned parameters, showing that the neural network code has the highest area under curve score with 0.9504.

Table 5: Confusion matrices, precision and recall score for each algorithm. Class benign is shortened to B, and class malignant is shortened to M

Classifier	Confusion Matrix $\begin{pmatrix} TrueMalignant & FalseBenign \\ FalseMalignang & TrueBenign \end{pmatrix}$	Precision B	Recall class B	Precision M	Recall M
Decision Tree	$\begin{pmatrix} 37 & 5 \\ 1 & 71 \end{pmatrix}$	0.9342	0.9861	0.9737	0.8810
Random Forest	$\begin{pmatrix} 37 & 5 \\ 0 & 72 \end{pmatrix}$	0.9351	1.0	1.0	0.8810
AdaBoost	$\begin{pmatrix} 38 & 4 \\ 1 & 71 \end{pmatrix}$	0.9467	0.9861	0.9744	0.9048
Neural Network	$\begin{pmatrix} 39 & 3 \\ 0 & 72 \end{pmatrix}$	0.9600	1.0	1.0	0.9286

Table 5 shows the confusion matrix with precision and recall score for each target class.

Figure 4 shows how important the different features are in each method. For Decision Tree, most important features was *worst concave point* (21) with importance of 0.8312, for Random Forest was also *worst concave point* (21) with importance 0.2241, and for AdaBoost, the most important feature was *mean radius* with importance 0.1950 (1).

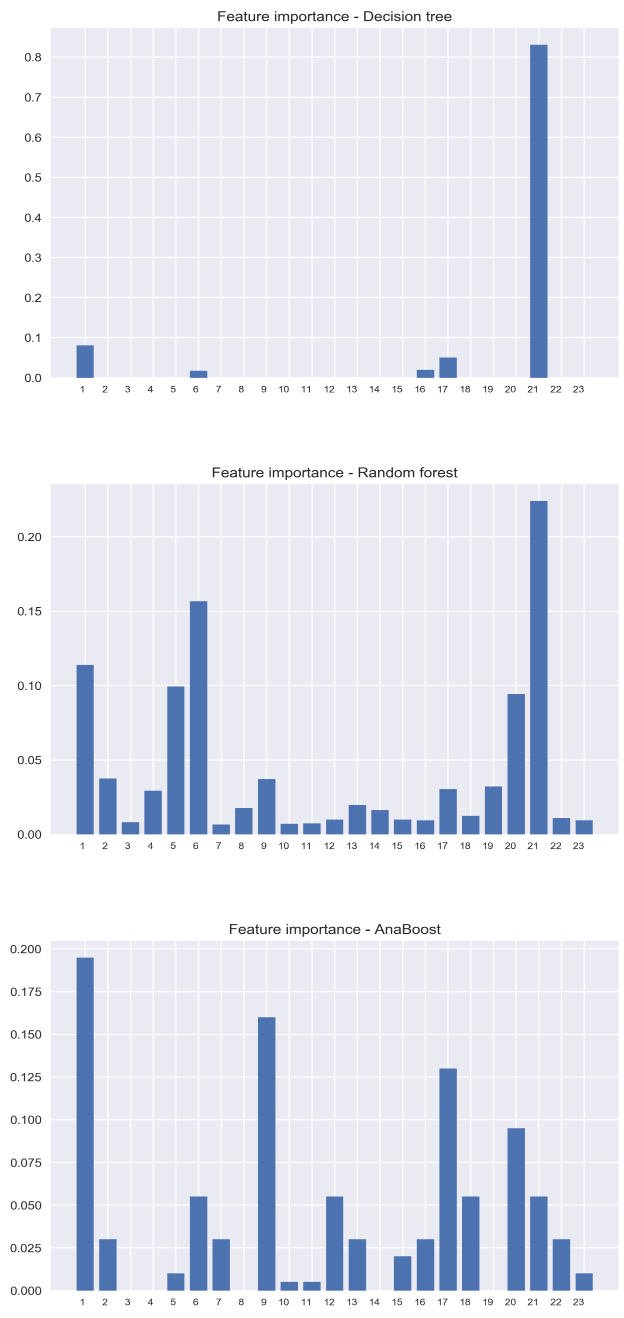


Figure 4: Feature importance for the different trees. The numbers along the x-axis have corresponding feature names listed in table 1, due to lack of space. As we can see in the Decision Tree (upper figure), there are only 5 parameters which even is considered in the tree; *mean radius* (1), *mean concave points* (6), *fractial dimension error* (16), *concave points error* (17) and *worst concave points* (21). For Random Forest and AdaBoost the importance of the other predictors are much more evenly spread. For Decision Tree, most important parameter is worst concave point (21) with importance of 0.8312, for Random Forest is worst concave point (21) with importance 0.2241, and for AdaBoost, the most important parameter was mean radius with importance 0.1950 (1).

Figure 5 shows the decision tree that we got with tree depth set to 3, and using the gini index.

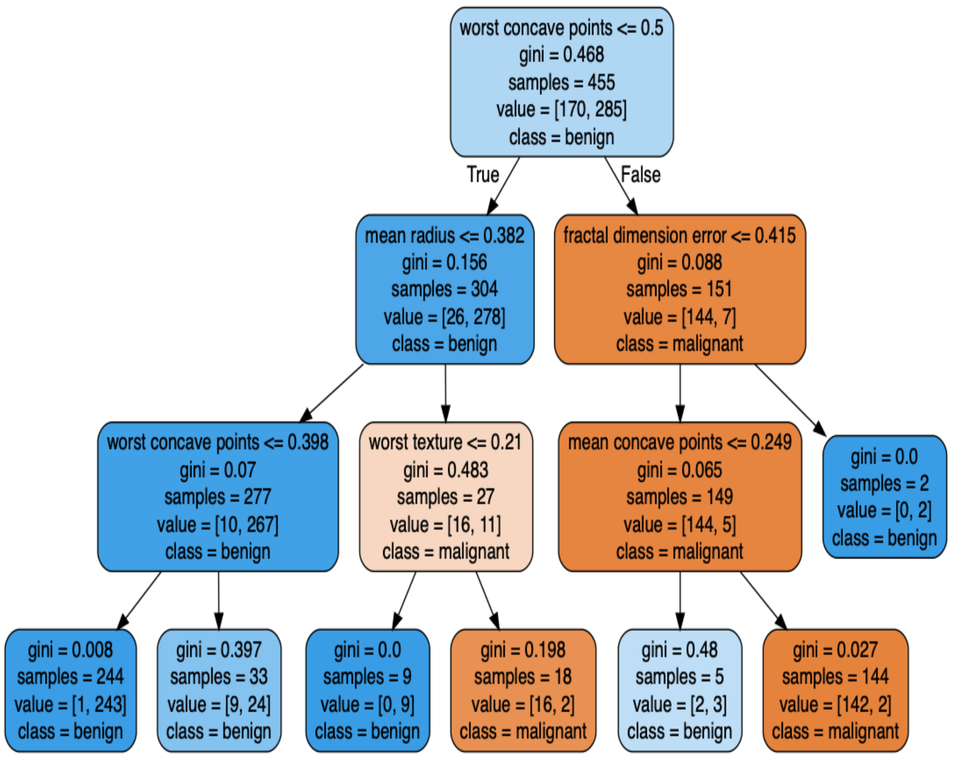


Figure 5: Decision Tree with depth=3, and with settings as in table 2 for Decision Tree.

5 Discussion

This chapter provide a discussion of our main findings of the analysis on the machine learning algorithms. This includes a discussion on our handling of the data in section 5.1, the tuning of hyperparameters in section 5.2, and thoughts around the different methods and algorithms in section 5.3.

5.1 Data handling

In this analysis we tested the accuracy score for a full set of predictors. However, this led to overfitting of the training data for all of the models[11]. The reasoning for the overfitting may be due to high dimensionality of the predictors together with only two classes of the target values. Also, the biased distribution of incidents in each class of the target values, may lead to overfitting of the majority class. One could try to handle this by balancing the data, but with a small set of only 569 data points, we decided not to cut down on data points. However, this is a thought for further investigation.

5.2 Hyperparameters and tuning

From the results of the accuracy score produced in figure 2, we clearly see the decision tree and neural network models overfitting for the training data[12]. The models also seem to do a poor job on classifying new data, consequently making it hard to decide upon the tuning parameters. Therefore, we chose to use *GridSearch*[10]. From table 2, for decision tree, the tree depth was set to 3 according to the grid search. If we look on figure 2, depth 3 seems to be a peak in the validation data. Whether this is a random peak or not, we do not know. As a thought for further investigation one should test for more values in this region, for instance, 1,2,3,4,5,6 in order to be certain that this is a good value for the tree depth. The same accounts for neural network, where max iterations was set to 100 (table 3). We can clearly see on figure 2 that the validation data peaks at 100 iterations. Here, we could also have tested more, especially around 100, with steps of 10. The same tendency seems to be relevant for the other parameters as well.

[11]The training accuracy can be tested in *main.py* by setting correlation_drop=False [line 19,20], and change the tuned
parameters [line 91,92] to tunes_full_predictors

[12]For more examples regarding accuracy score versus tuning parameters, please see the folder /Figures on GitHub

The Decision Tree with tree depth set to 3 is illustrated in figure 5. For each node, we have feature, and a criterion which decides the threshold for the splits, the gini index, the number of samples in each node, the number of malignant and number of benign tumors. The class which is overrepresented is deciding the *class* paramater. There is only one pure leaf node (gini=0.0), where we had 9 true benign classifications. This means that we could probably have gone for a deeper tree depth.

5.3 Discussion on methods

Looking at the accuracy score in table 4, the overall scores are high, which means that the models does a pretty good job in the classification. However, it is clear that the Decision Tree performs worst with an accuracy of 0.9474 for the validation data, while Neural Network performs best with an accuracy score of 0.9737 for the validation data. For all methods, we see that the difference between training and validation data is rather high, with a difference of 0.0174 for Decision Tree, 0.0417 for Random Forest and AdaBoost, and 0.0263 for Neural Network, which again is a sign of overfitting to the training data (especially in the case of AdaBoost and Random Forest).
Looking at the area under curve for table 4 (which the area under ROC curve in figure 3), the numbers agree well with the accuracy, but there is a slight difference in the area under curve for Random Forest and AdaBoost, where AdaBoost is slightly higher. On figure 3, we can see that the ROC curves for AdaBoost and Random Forest differs. These values imply that AdaBoost is doing a slightly better job than Random Forest.
Looking at the confusion matrices in table 5, we can get a clearer picture of how well the different methods classifies data to the correct class. The validation data contains 114 datapoints, where we have 72 cases of benign tumors and 42 cases of malignant tumors. In general, the models does a very good job in predicting the benign tumors, with the incidences of false malignant tumors being 0 or 1 for each model. However, there are more incidences of predicting malignant tumors as benign, which is problematic for our model, since we are more interested in classifying malignant tumors as malignant, as it would be catastrophic to classify a malignant tumor as benign.
Figure 4 shows a barplot of the importance of the different features. As we can see for decision tree, only *worst concave points*, *worst texture*, *mean concave points* and *mean radius* are the predictors that were used to classify. For the other two cases, we see that the trees are based upon other features as well, which is good, and it makes a better model. For decision tree, only 5 parameters are used (due to tree depth=3), while for the other two methods, the importance of the parameters are much more evenly spread out, which means that the models can reduce the variance of the models, by allowing other features to test for the accuracy. For AdaBoost, the most important feature is not *worst concave points*, but rather *mean radius.*

6 Conclusion

When handling medical data like the Wisconsin Breast Cancer Data, we believe that it is more important that the chosen machine learning method truly classifies malignant tumors as malignant, than falsely classifying benign as malignant, as the consequence can be catastrophic unless the tumor is further followed up with other medical methods. Our models does a very good job in predicting true benign tumors, and does slightly worse in predicting true malignant. However, the overall classification is very good, with only a few cases of misclassifications.
It is clear that boosting and bagging the Decision Tree slightly improves the performance, but overall, the neural network performs best in all measurements. However, it seems like we are dealing with some overfitting of the data, which proposedly could have been avoided if we balanced the data. For improvements, we would have tried different feature testing and balancing of the data.

References

[1] World Health Organization, (2018, 12.dec), *"Latest global cancer data: Cancer burden rises to 18.1 million new cases and 9.6 million cancer deaths in 2018"*, Press release No. 263, https://www.who.int/cancer/PRGlobocanFinal.pdf (accessed 17.12.19)

[2] American Cancer Society, (2019, 20.sep), *"Survival Rates for Breast Cancer"*, https://www.cancer.org/cancer/breast-cancer/understanding-a-breast-cancer-diagnosis/breast-cancer-survival-rates.html (accessed 17.12.19)

[3] World Health Organization, *"Breast Cancer"*, https://www.who.int/cancer/prevention/diagnosis-screening/breast-cancer/en/ (accessed 17.12.19)

[4] Kumari, M. & Singh, V., *"Breast Cancer Prediction system"*, Procedia Computer Science, 2018, Vol.132, p.371-376

[5] Street, Holberg & Mangasarian, *"Nuclear Feature Extraction For Breast Tumor Diagnosis"*, Science and Technology, Vol.1905, p.861-870.

[6] Asri, Mousannif, Al Moatassime & Noel, *"Using Machine Learning Algorithms for Breast Cancer Risk Prediction and Diagnosis"*, Procedia Computer Science, 2016, Vol.83, p.1064-1069

[7] *"Data Analysis and Machine Learning: From Decision Trees to Forests and all that"*. https://compphysics.github.io/MachineLearning/doc/pub/DecisionTrees/html/DecisionTrees.html?fbclid=IwAR3 (accessed 04.12.19)

[8] Hastie, Tibshirani, & Friedman(2009) *"The Elements of Statistical Learning"* (p.305-334). Standford University: Springer. Second edition.

[9] Marsland, Stephen. "Machine Learning- *An Algorithmic Perspective*", second edition. Chapman & Hall/CRC.