UiO : Department of Chemistry
University of Oslo

STK-IN4300: Statistical Learning Methods in Data Science

Second mandatory assignment

Regression and classification

Jørn Eirik Betten

Oct 15 2022

Abstract

These exercise contain comparisons of regression methods on the QASR data set and classification methods on the Pima Indian Diabetes data set. All code related to these exercises can be found at the Github repository STK-IN4300.

1 Datasets and acknowledgments

1.1 QASR data set

Consider the data from the study of (Cassotti et al., 2014) on aquatic toxicity. The goal is to predict its effect towards Daphnia Magna, a small planktonic crustacean that lives in fresh water. Aquatic toxicity is measured through the variable LC50, namely the concentration that causes death in 50% of crustacean over a test duration of 48 hours. For the prediction, 8 variables have been considered:

- **TPSA**: the topological polar surface area calculated by means of a contribution method that takes into account nitrogen, oxygen, potassium and sulphur;
- **SAacc**: the Van der Waals surface area (VSA) of atoms that are acceptors of hydrogen bonds;
- **H**050: the number of hydrogen atoms bonded to heteroatoms;
- **MLOGP**: expresses the lipophlicity of a molecule, this being the driving force of narcosis;
- **RDCHI**: a topological index that encodes information about molecular size and branching;
- **GATS1p**: information on molecular polarisability;
- **nN**: the number of nitrogen atoms present in the molecule;
- **C**040: the number of carbon atoms of a certain type, including esters, carboxylic acids, thioesters, carbamic acis, nitriles, etc.

,

The data set contains 546 observations.

1.2 Pima Indians Diabetes data set

Unfortunately, I could not find a proper citation for the Pima Indians Diabetes data set. However, the data set contains information about 768 women of a population (Pima) particularly susceptible to diabetes. The response **diabetes** indentifies which of the persons involved in the study (268 women, **diabates=‘pos’**) developed the disease. Eight continuous independent variables contain information on:

- **pregnant**: number of pregnancies;
- **glucose**: plasma glucose concentration at 2 h in an oral glucose tolerance test;
- **pressure**: diastolic blood pressure (mmHg);
- **triceps**: triceps skin fold thickness (mm);
- **insulin**: 2-h serum insulin (μU/mL);
- **mass**: body mass index (kg/m^2);
- **pedigree**: diabetes pedigree function;
- **age**: age (years)

1.3 Acknowledgements

I heavily used sci-kit learns package for solving these exercises (Pedregosa et al., 2011). I also used pygam for the generalized additive models (Servén and Brummitt, 2018).

2 Regression

In this section the QASR dataset will be analyzed using various regression methods.

2.1 Dichotomization effects with standard splits

We split the data in training and testing sets, with a relative size of approximately 2 : 1, respectively. From the training set, we create two instances; one where the independent variables H050, nN and C040 are all dichotomized (binary values; presence or non-presence of atom), and one that remain non-dichotomized. Then we make linear models for both training sets with the same targets (LC50), and compare. In Table 2.2 the mean squared errors and the corresponding r2 scores for computations on the non-dichotomized and dichotomized testing sets (otherwise identical), using a model trained, respectively, on the non-dichotomized training data and one model trained on the dichotomized training data. The corresponding training errors are in Table 2.1. The non-dichotomized versions of the data set yields better performance in this case.

Data set	Mean Squared Error	R^2 score
Non-Dichotomized	1.261	0.493
Dichotomized	1.316	0.471

Table 2.1: *Training errors measured in mean squared error and R^2 on the non-dichotomized and dichotimized versions of the data.*

Data set	Mean Squared Error	R^2 score
Non-Dichotomized	1.770	0.468
Dichotomized	1.821	0.453

Table 2.2: *Testing errors measured in mean squared error and R^2 on the non-dichotomized and dichotimized versions of the data.*

2.2 Dichotomization effects on multiple split ratios

Figure 2.1 shows that the expected error of the linear model on the test data is higher when the count variables are dichotomized, compared with when they are not. This can be due to the loss of information in the dichotomized data set. Dichotomizing the features increased the p-value of nN dramatically (see Table 2.4 and Table 2.3), which tells us that the linear relationship between the nN feature and response variable has been greatly reduced. This may be the largest contributor to the differences between the expected prediction errors on non-dichotomized and dichotomized features.

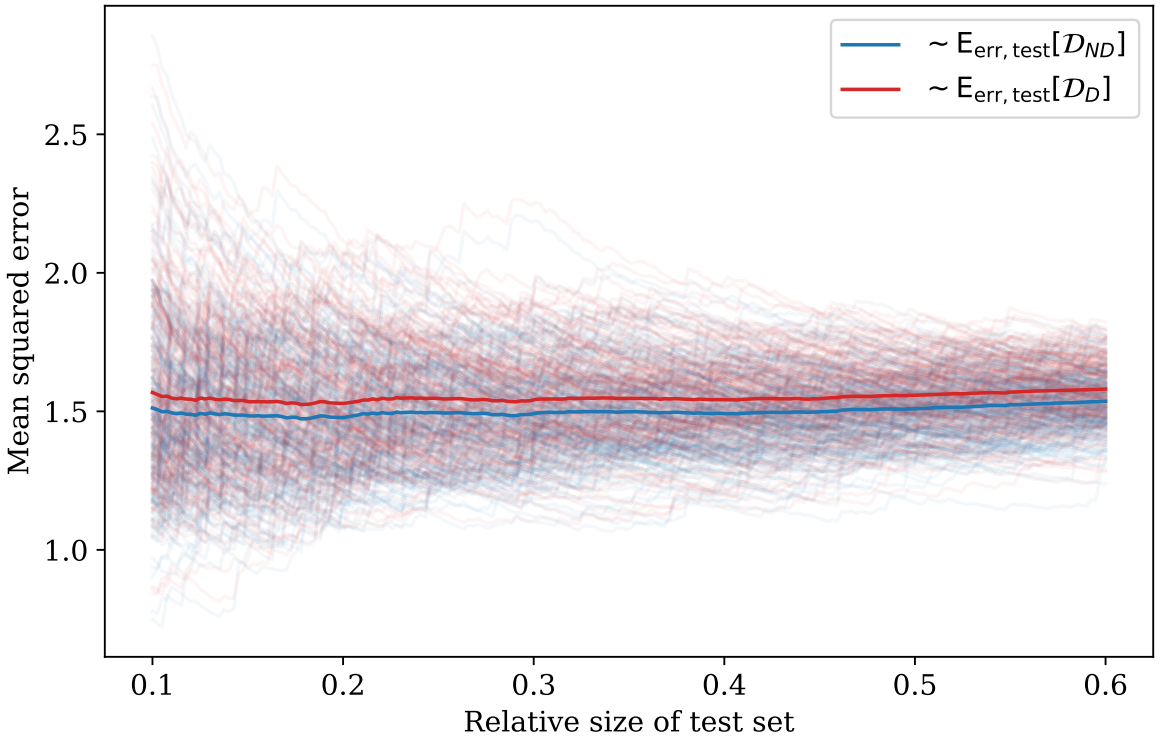


Figure 2.1: *Plot of errors on test set using 200 different split ratios on the dataset, and 200 different splits per split ratio. The highlighted line represents the average of all these (the expected prediction error). The $\mathscr{D}_{ND}$ represents the non-dichotomized version of the dataset, while $\mathscr{D}_D$ represents the dichotomized version.*

Table 2.3: *Table containing information about the significance of the regression coefficients, using the non-dichotomized features.*

labels	w	se	t	.025	.975	df	P > |t|
Intercept	2.786	0.349	7.975	2.436	3.135	355	0.000
TPSA	0.029	0.004	6.800	0.025	0.033	355	0.000
SAacc	-0.017	0.003	-5.578	-0.020	-0.014	355	0.000
H050	0.012	0.077	0.161	-0.064	0.089	355	0.436
MLOGP	0.380	0.090	4.228	0.290	0.470	355	0.000
RDCHI	0.609	0.192	3.172	0.417	0.801	355	0.001
GATS1p	-0.683	0.208	-3.286	-0.891	-0.475	355	0.001
nN	-0.230	0.074	-3.125	-0.304	-0.156	355	0.001
C040	-0.014	0.121	-0.118	-0.136	0.107	355	0.453

Table 2.4: *Table containing information about the significance of the regression coefficients, using the dichotomized features.*

labels	w	se	t	.025	.975	df	P > |t|
Intercept	2.864	0.277	10.322	2.586	3.141	355	0.000
TPSA	0.024	0.004	5.518	0.019	0.028	355	0.000
SAacc	-0.015	0.003	-6.040	-0.018	-0.013	355	0.000
H050	-0.123	0.154	-0.797	-0.277	0.031	355	0.213
MLOGP	0.368	0.075	4.896	0.293	0.443	355	0.000
RDCHI	0.602	0.183	3.296	0.419	0.785	355	0.001
GATS1p	-0.676	0.188	-3.590	-0.864	-0.488	355	0.000
nN	-0.047	0.150	-0.315	-0.197	0.103	355	0.377
C040	-0.103	0.170	-0.604	-0.273	0.067	355	0.273

2.3 Variable selection procedures

We go back to the split where the train data size and test data size ratio is $2:1$, respectively, and we will use this split for the remaining regression models. I ran sequential forward variable selection and backward elimination variable selection, and they produced the same features. In Figure 2.2 the Akaike Information Criteria and Bayesian Information Criteria with all possible numbers of features with the chosen features from the forward selection and backward elimination procedures (they yield the same values). I have also added the full data, with all features.

With the AIC stopping criteria, we see that we minimize the AIC when the number of features is four, and the features yielded by both forward selection and backward elimination were TPSA, SAacc, MLOGP and nN. With BIC as the stopping criteria, we found that three features minimized the score, and also here the same features were found in the forward selection and backward elimination: TPSA, SAacc and MLOGP. In Table 2.5 the training and testing errors, together with their respective R^2 scores are tabulated.

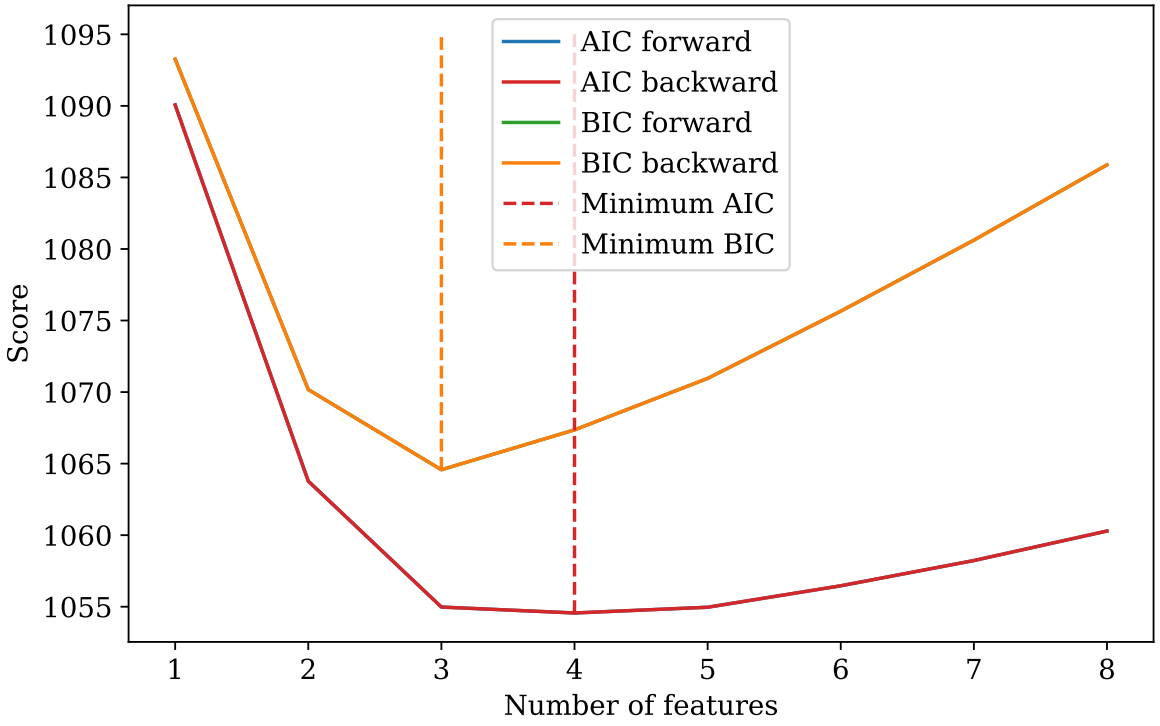


Figure 2.2: *AIC and BIC scores as a function of the number of parameters. In this case the forward selection and the backward elimination procedures produce the same features.*

Stopping criteria	Test MSE	Test R^2	Train MSE	Train R^2
AIC	1.7925	0.4618	1.3427	0.4605
BIC	1.8166	0.4545	1.3938	0.4400

Table 2.5: *Metrics (R^2 and MSE) on the test and training set using the best subset yielded with the AIC and BIC stopping criteria, for both forward selection and backward elimination methods.*

2.4 Ridge regression with CV and Bootstrap

Here we will explore ridge regression, and we will search for the best model using two approaches; 10-fold cross-validation and Bootstrap resampling (1000 samples). We know that ridge regression is a smooth shrinkage method, as is visualized in Figure 2.3, where the expected coefficient values over the bootstrap method are plotted against tuning parameter. These figures show how the coefficients change with the tuning parameter, α, (they are shrinked, and the lower the p-value of the coefficient, the more they are shrinked). Figure 2.4(a) shows a plot of the expected training errors for the Bootstrap and 10-fold CV procedures against the shrinkage parameter. Figure 2.4(b) displays the expected prediction errors from the two approaches, where the minima are marked.
The minima are quite close for both approaches. As $\alpha \to 0$, we would gain the ordinary least squares model. In Figure 2.4(b) we see that the minima are found just before the shrinkage of the feature space becomes too large, but the expected prediction errors does not become noticeably smaller, compared with the the lowest α values in the plot. We therefore test the

ridge models on the test set, and the results are displayed in Table 2.6. We find that the mean-squared errors is considerably lower compared with the OLS model (Table 2.2).

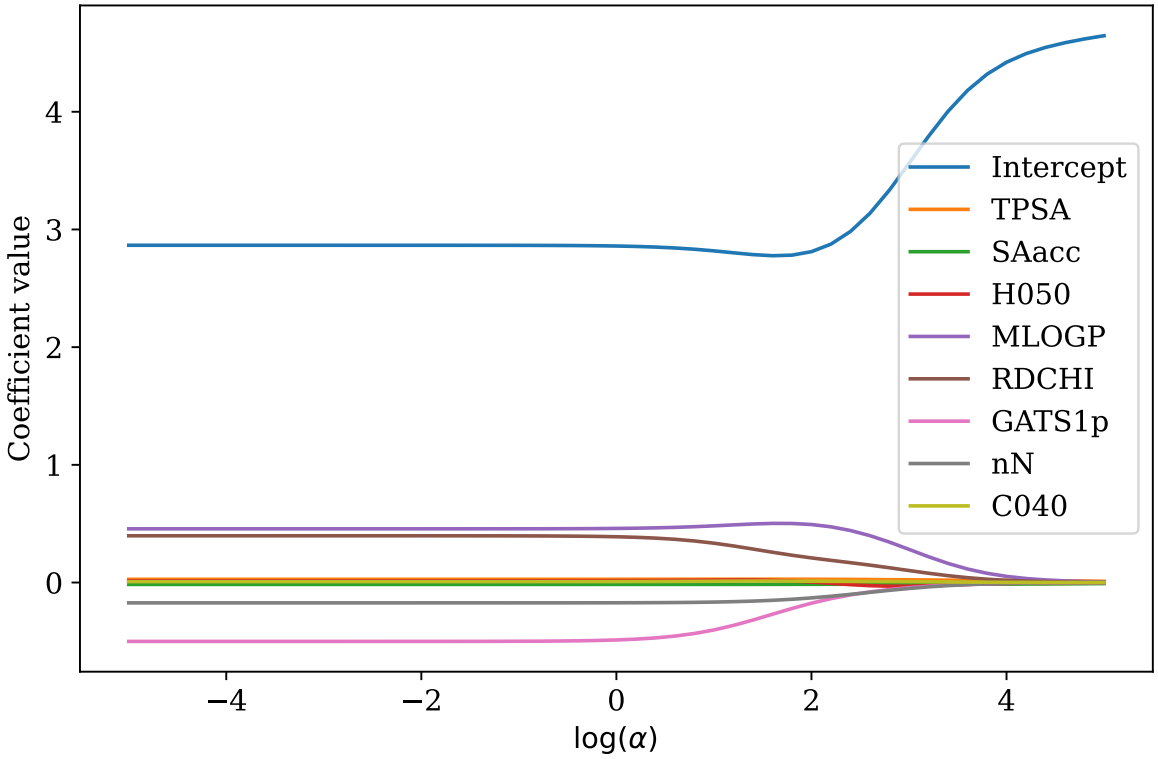
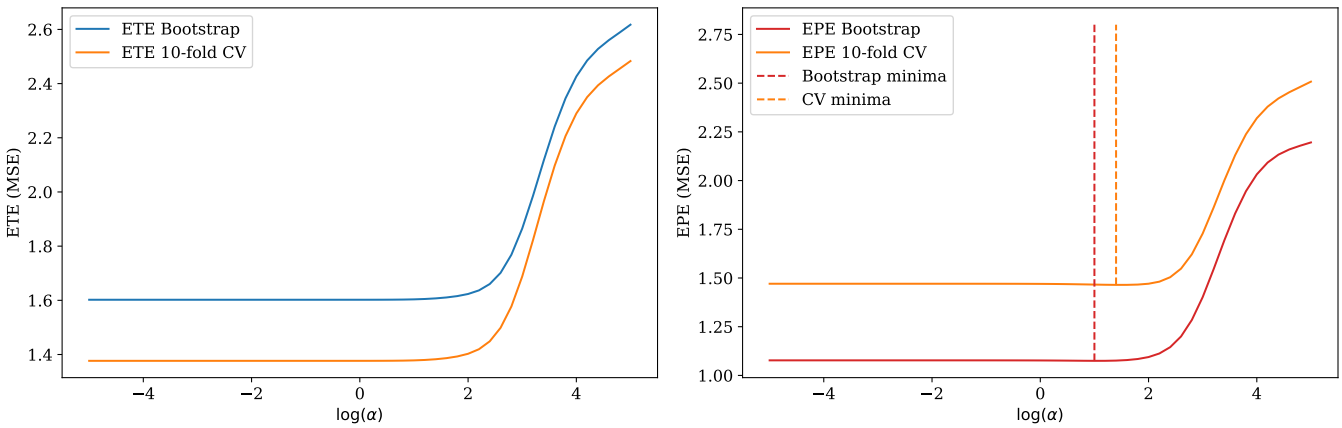


Figure 2.3: *Ridge coefficients found via the Bootstrap resampling method as a function of the α shrinkage parameter.*



(a) (b)

Figure 2.4: *In (a) the expected training errors yielded by Bootstrap resampling and 10-fold cross-validation against the tuning parameter α are displayed. In (b) the expected prediction errors in for the two procedures are plotted against α, and the minima for the approaches are marked with dashed vertical lines.*

Method for finding optimal α	Optimal α	Test MSE	Test R^2	Train MSE	Train R^2
Bootstrap resampling	10	1.5799	0.4597	1.3826	0.4861
10-fold CV	25.1	1.5960	0.4542	1.3862	0.4847

Table 2.6: *Metrics (R^2 and MSE) on the test and training sets using the hyperparameters found via the Bootstrap and 10-fold CV approaches.*

2.5 Generalized additive model

I implemented a generalized additive model with p-splines as basis functions, using pygam. I tested three models, one where the basis functions were of first order, where the number of basis functions was 10, another where the basis functions were of quadratic order, with 11 basis functions, and the a third model, where the basis functions were of cubic order, with 12 basis functions.
Figure 2.5 displays the expected prediction and training errors found using 100 bootstrap resampling sets using the training set. The spline basis of cubic order yields the lowest expected prediction error, with quadratic quite close, and linear not too far from those either. Table 2.7 provides tabulated MSE and R^2 scores for both training and testing sets, and shows very low mean squared errors compared with the other methods.

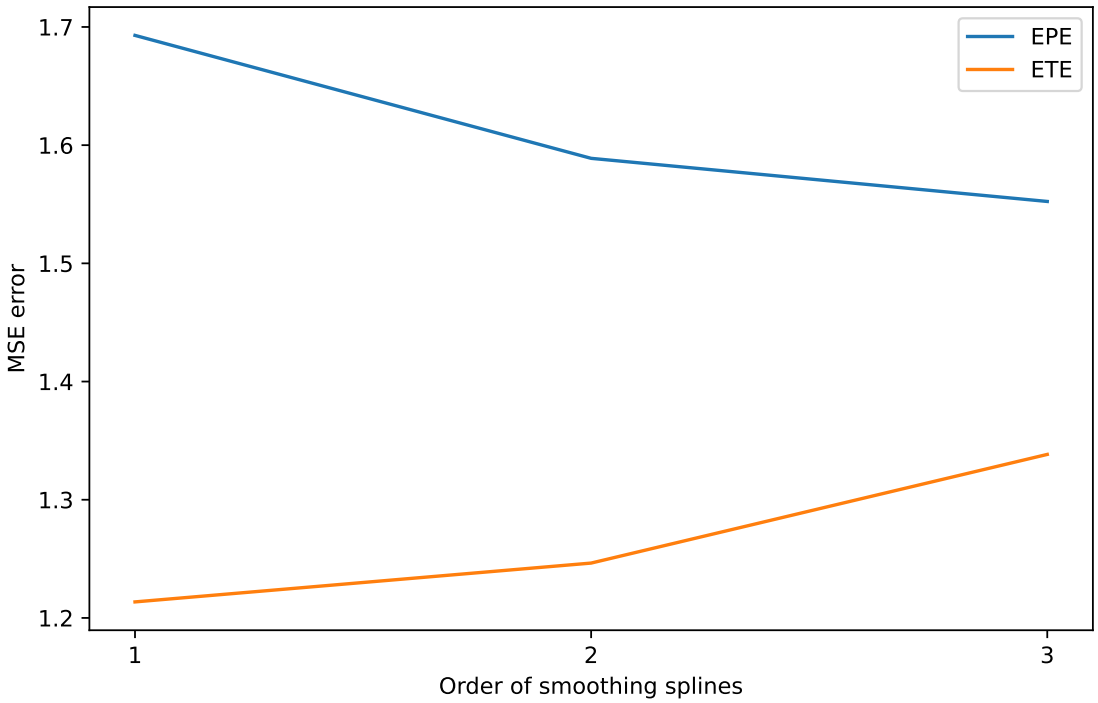


Figure 2.5: *Expected training and testing error against the order of the spline-basis.*

Order of spline basis	Test MSE	Test R^2	Train MSE	Train R^2
Linear	1.2922	0.4721	1.2835	0.5607
Quadratic	1.3007	0.4686	1.3085	0.5521
Cubic	1.3270	0.4579	1.3595	0.5347

Table 2.7: *Metrics (R^2 score and MSE) from generalized additive models with a spline basis of linear, quadratic and cubic orders.*

2.6 Regression tree

I implemented the tree using scikit-learn. My implementations started with finding the cost complexity pruning path, and then pruning the tree, while making an estimation of the expected prediction error (using 5-fold cross-validation to tune the cost complexity parameter α). Then I choose the α that minimizes expected prediction error, and I build the tree. Figure 2.6(a) shows how the number of nodes the decision tree contains is related to the cost complexity parameter, α. The number of nodes is rapidly decreasing with increasing cost in the complexity of the tree, analogous to how the depth of the tree is reduced, as can be seen in Figure 2.6(b). We found the optimal $\alpha = 0.044$, visualized in Figure 2.7, and the final regression decision tree model is drawn in Figure 2.8. The training and testing errors are displayed in Table 2.8.

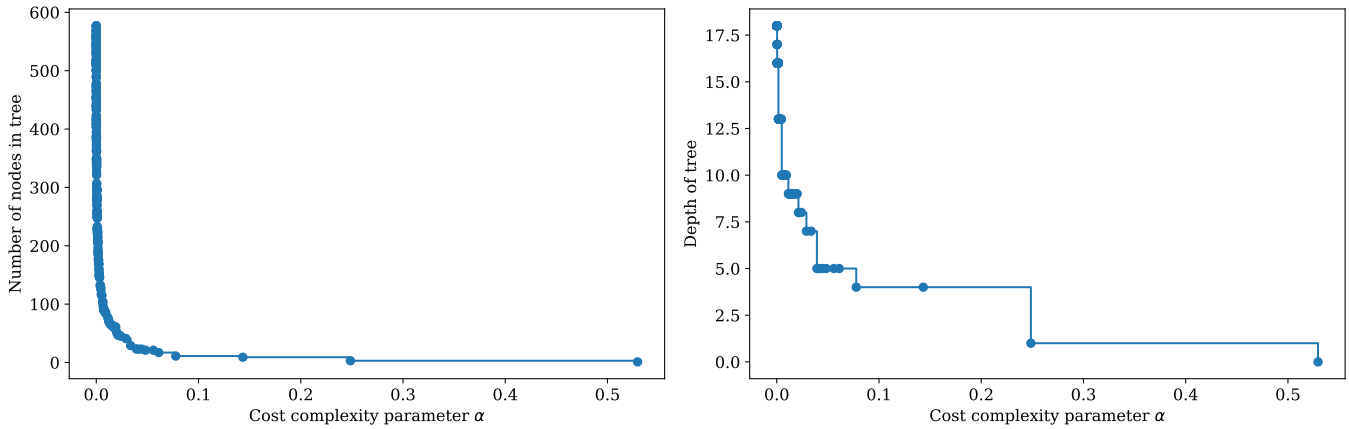


Figure 2.6: *(a) Number of nodes in tree against the cost complexity parameter α. (b) Depth of tree against the cost complexity parameter α.*

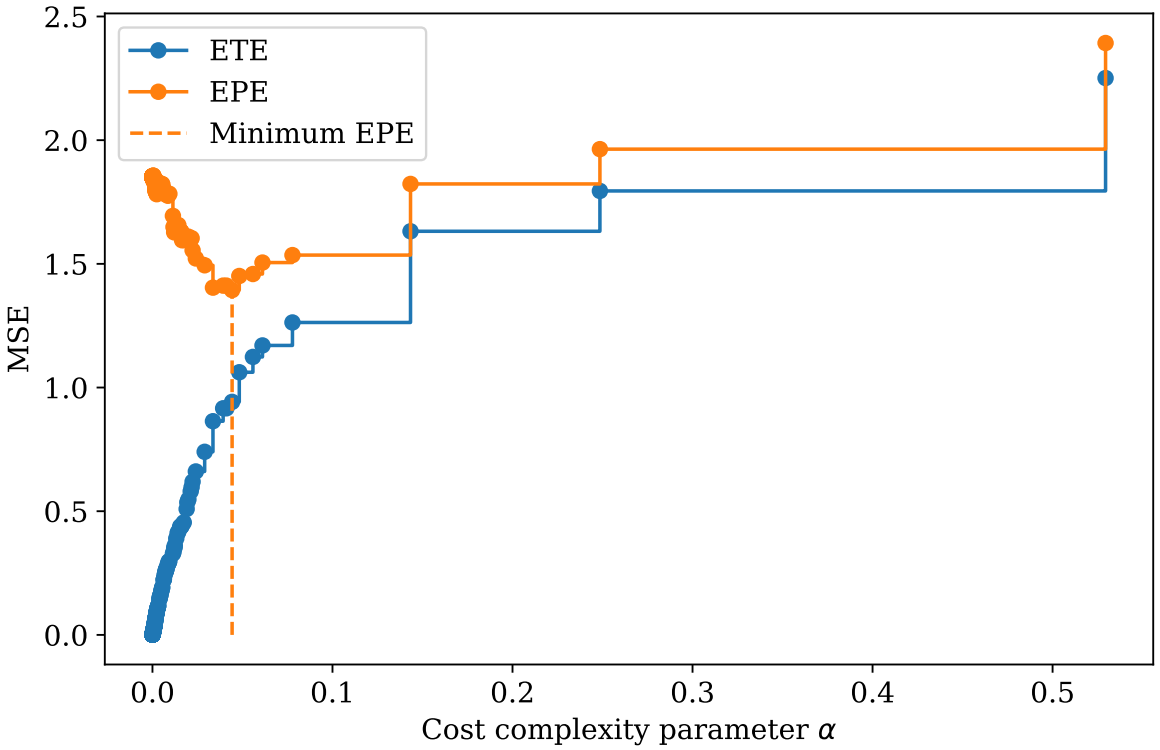


Figure 2.7: *Expected training and test errors using 5-fold cross validation against the cost complexity parameter, α.*

print(preds)

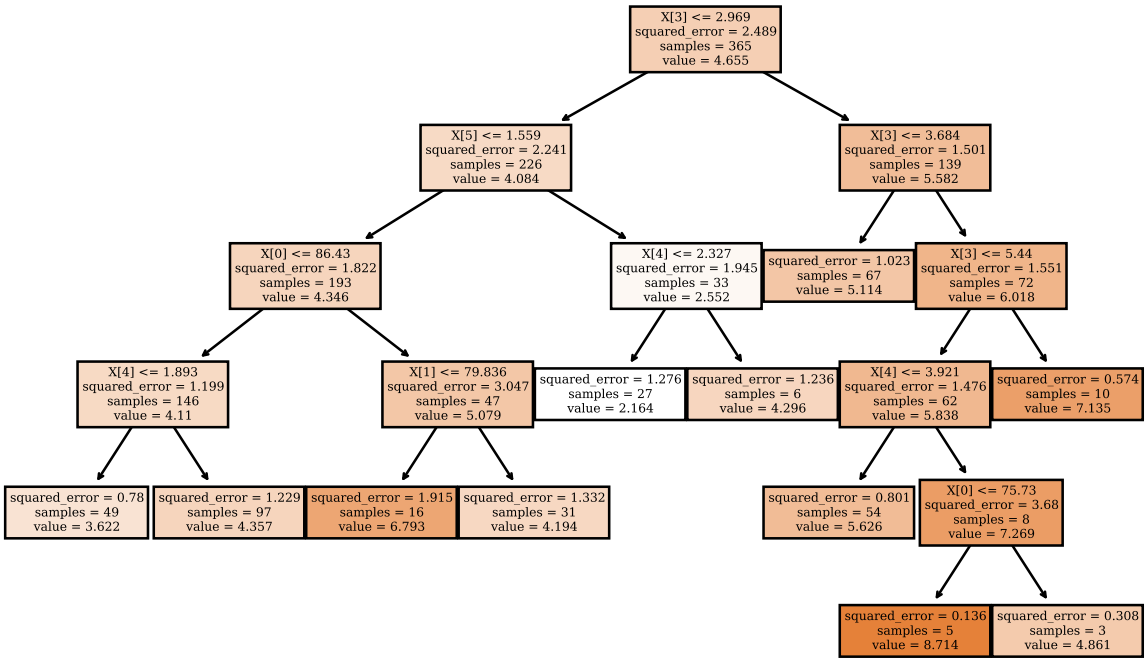


Figure 2.8: *Tree architecture produced on the full training set, with the found optimal complexity parameter $\alpha = 0.044$ yielded by the cross-validation (Figure 2.7).*

α	Test MSE	Test R^2	Train MSE	Train R^2
0.4423	1.8603	0.4414	1.0697	0.5702

Table 2.8: *Regression decision tree errors after pruning the tree according to Figure 2.4.*

2.7 Results

The results from all the different models given on the same training-test split ratio of the data, with the MSE and R^2 metrics, are given in Table 2.9. The top performers are the generalized additive models, with ridge at a solid second place. The GAM models has the advantage of being able to model non-linearity atop the linear correlations within the dataset, and is therefore, not surpisingly perhaps, the best model. The subset models performs almost as well as the full ordinary least squares. The bootstrap and cross-validation methods for hyperparameter tuning is performing similarly, although the cross-validation approach has less computational costs. The regression decision tree performs worst of all models.

Model	Test MSE	Test R^2	Train MSE	Train R^2
OLS (ND)	1.7701	0.4685	1.2614	0.4932
OLS (D)	1.8216	0.4530	1.3155	0.4714
Subset (AIC)	1.7925	0.4618	1.3427	0.4605
Subset (BIC)	1.8166	0.4545	1.3938	0.4400
Ridge (Bootstrap)	1.5799	0.4597	1.3826	0.4861
Ridge (10-fold CV)	1.5960	0.4542	1.3862	0.4847
GAM (linear)	1.2922	0.4721	1.2835	0.5607
GAM (quadratic)	1.3007	0.4686	1.3085	0.5521
GAM (cubic)	1.3270	0.4579	1.3595	0.5347
Regression tree	1.8603	0.4414	1.0697	0.5702

Table 2.9: *Comparison of regression methods using the mean-squared error and R^2 score on the test set.*

3 Classification

In this section the Pima Dataset will be analyzed using various classifiers. We split the data in test size/train size ratio of $1:2$, as before.

3.1 K-Nearest Neighbors

Figure 3.1 displays the results of the 5-fold (a) and Leave-One-Out (b) cross-validation, where the maximum expected prediction accuracy score has been marked with a dashed line. They yield the same optimal k-value, 16, and they are yielding similar expected prediction accuracy scores overall. The 5-fold CV seems to like less general models (low k-values). This makes sense as the leave-one-out cross-validation makes should make for a better approximation of the true expectation value for the accuracy score, as it test every single point in the training set on a model that is very similar to the original training set.

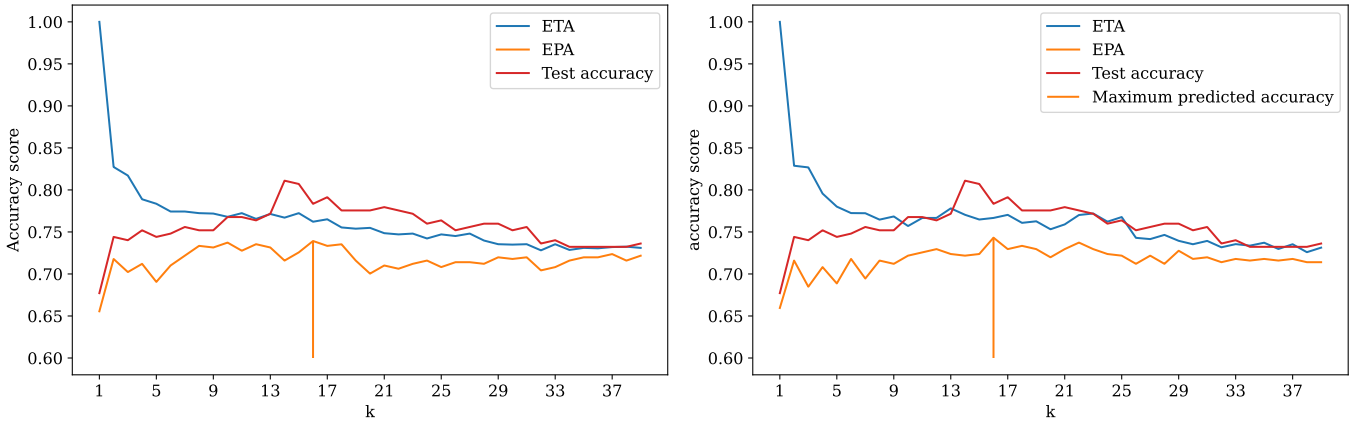


Figure 3.1: *(a) Displays the expected prediction and training accuracy scores from the 5-fold cross validation together with the test accuracy score. (b) Displays the expected prediction and training accuracy scores from the Leave-One-Out cross validation approach together with the test accuracy.*

Approach	Optimal k	Accuracy Test	Accuracy Train
5-fold CV	16	0.7835	0.7665
LOO CV	16	0.7835	0.7665

Table 3.1: *Optimal k-values for the 5-fold and LOO cross-validation approaches for the k-nearest neighbor method.*

3.2 Generalized additive models

The GAM model chosen was of the type **LogisticGAM** found in pygam documentation. I only investigated the splines of cubic order, with 12 basis functions, and I performed the subset search via the forward selection with AIC as criteria.
The results of the forward selection with AIC as criteria, can be displayed in Figure 3.2. We found the best subset of features to be **glucose**, **age**, **mass**, **pedigree** and **pressure**, and the results, given in accuracy scores on training and test sets, are tabulated in Table 3.2.

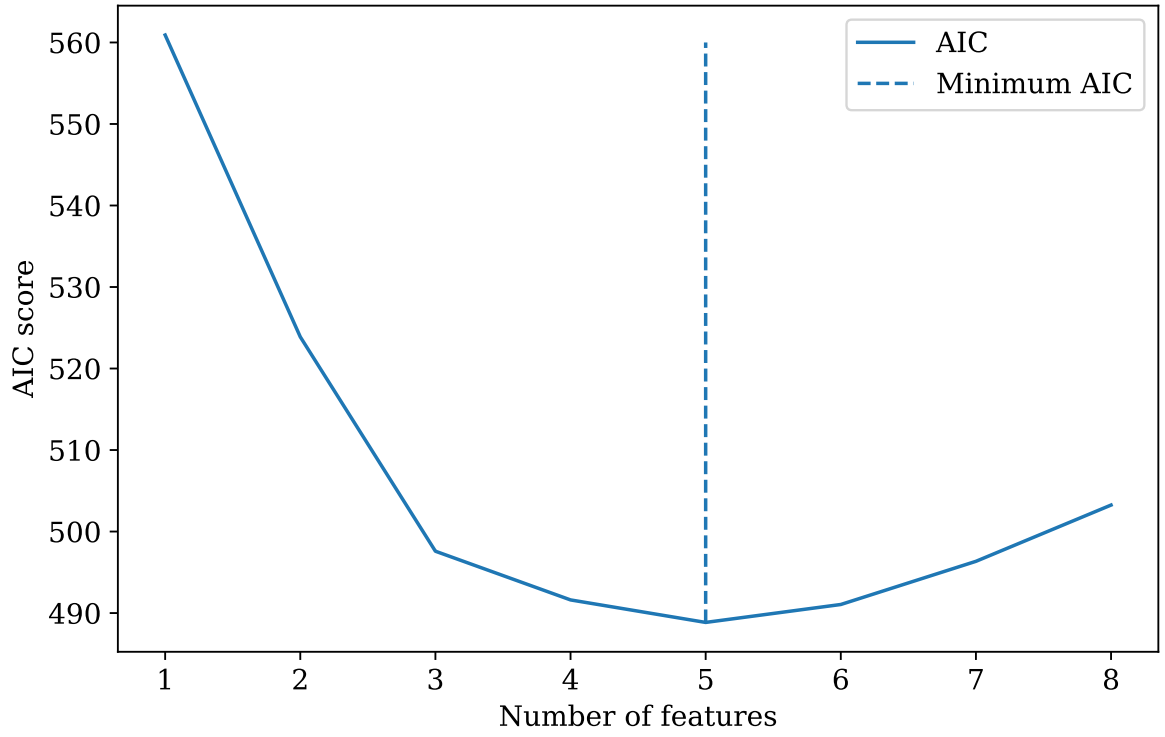


Figure 3.2: *AIC scores as function of the number of features used in the forward selection procedure for the Logistic GAM model.*

Model	Accuracy Test	Accuracy Train
Best subset	0.7953	0.7860
Full model	0.7913	0.7840

Table 3.2: *Accuracy scores on test and training sets on the model found via the best subset selection, and the full model.*

3.3 Decision trees, Bagging, Random Forest and AdaBoost

3.3.1 Decision tree classifier

Figure 3.3 displays the estimated predicted and training accuracy scores, found via 5-fold cross validation. The results of the optimized model is listed in Table 3.3. The architecture of the final tree is displayed in Figure 3.4.

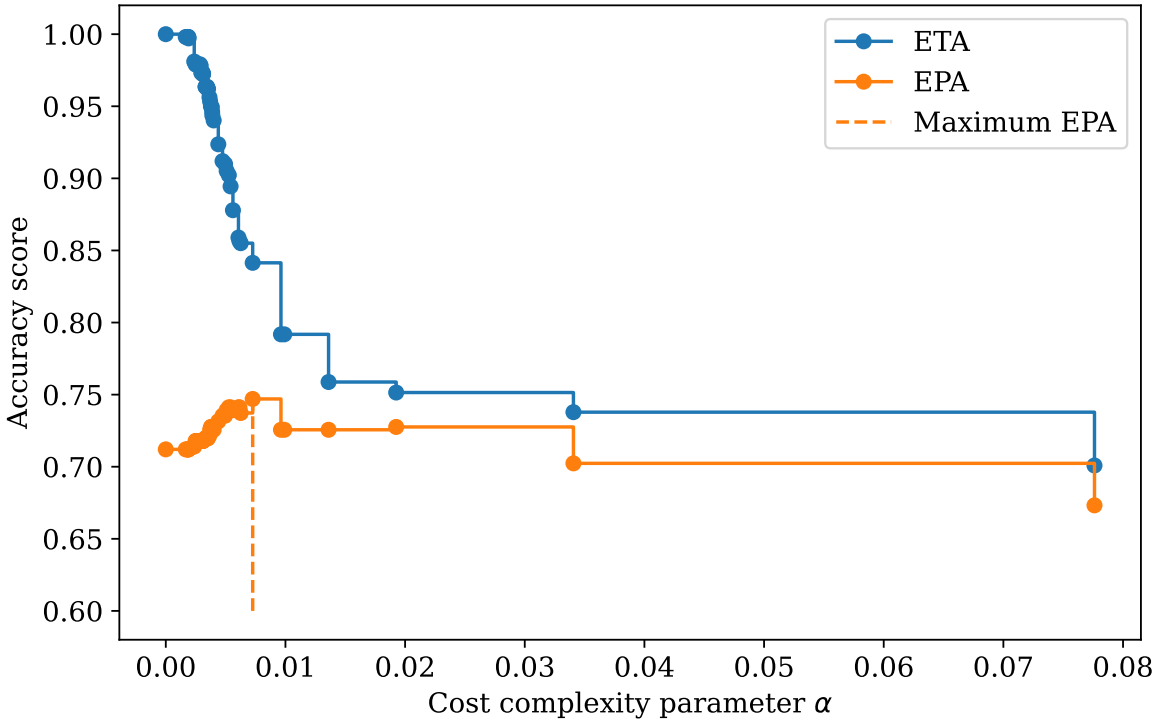


Figure 3.3: *Expected prediction and training accuracy found via 5-fold cross-validation of classifier tree, using the cost complexity pruning path. The dashed line represent the best expected prediction accuracy score.*

7

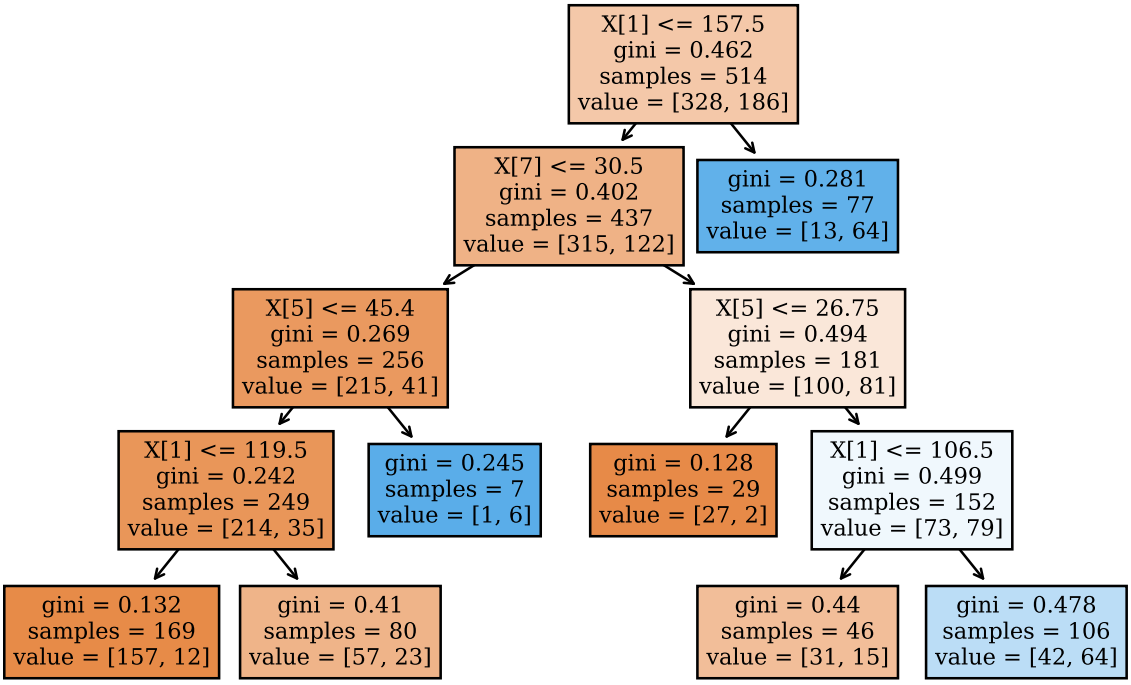


Figure 3.4: *Architecture of best tree given the expected best cost complexity parameter α.*

Approach	Optimal α	Accuracy Test	Accuracy Train
5-fold CV CCP	0.007262	0.7952	0.7899

Table 3.3: *Results of the tree classifier after being pruned according to expected prediction accuracy score given by 5-fold cross validation of the cost-complexity path.*

3.3.2 Bagging

I initiated a bagging ensemble method with 100 decision tree classifiers, where each base estimator draws three features to train on, for the probability votes, while for the consensus votes I chose the k-nearest neighbor base estimator, with k= 10 where each estimator also here drew three features to train on. The results are given in Table 3.4.

Method	Accuracy Test	Accuracy Train
Bagging (Tree)	0.8031	1.0
Baggin (kNN)	0.7992	0.8171

Table 3.4: *The results of a bagging of* 100 *decision tree classifiers.*

3.3.3 Random Forest

The Random Forest ensemble method was initiated with 100 trees of a maximum depth of 4 (one lower than depth of best tree in cost complexity path). The results yielded from the Random Forest method is tabulated in Table 3.5.

Method	Accuracy Test	Accuracy Train
Random Forest	0.8189	0.8288

Table 3.5: *Accuracy scores on training and test sets using the Random Forest ensemble method.*

3.3.4 AdaBoost

The AdaBoost ensemble method was initiated with 100 decision trees of maximum depth 1, and I did a coarse grid search over learning rates to find that the best estimate I could get was when the learning rate was 0.1. The accuracy scores on the training and test sets are given in Table 3.6.

Method	Accuracy Test	Accuracy Train
AdaBoost	0.7992	0.7626

Table 3.6: *Accuracy scores on training and test sets using the AdaBoost ensemble method.*

3.4 Choice of method

Table 3.7 displays a summary of the results (evaluation accuracy scores) on the PimaIndians-Diabetes.csv data set. The top performing model seems to be the Random Forest model,

which is a weighted mean of 100 decision trees. The decision path of the Random Forest can easily be extracted, and the data can be analyzed thoroughly, although I'd prefer to use the GAM with the best subset, where the interpretability of the model is easily visualized with 95% confidence intervals for the dependence of the response on each input. In danger of exposing my laziness, finding a clean interpretation of the model swiftly is of importance to me, and the accuracy scores are of similar magnitudes. Also, the entire process of building the best subset GAM yielded insights into the nature of the data set. My choice of methods would therefore be the GAM and Random Forest approaches, and try to combine the insights I get from both models. We must also remember that test accuracy scores given here may have a large stochastic character, as the data set is quite small (only 763 observations).

Model	Hyperparameter	Accuracy Test	Accuracy Train
k-Nearest Neighbors	k=16	0.7835	0.7665
GAM (full model)	order basis = 3	0.7913	0.7860
GAM (best subset)	order basis = 3	0.7953	0.7860
Decision Tree (ccp)	$\alpha = 0.007262$	0.7952	0.7899
Bagging (Decision Trees)	max features= 3	0.8031	1.0
Bagging (10-nearest neighbors)	max features= 3	0.7992	0.8171
Random Forest	tree(2 features)	**0.8189**	0.8288
AdaBoost	lr=0.1	0.7992	0.7626

Table 3.7: *Accuracy scores on the test and training sets given model.*

3.5 Results on corrected dataset

Table 3.8 displays the results using the same procedures as in Section 3.1, Section 3.2 and Section 3.3, only now dropping the values of the data set that are missing (identifying the missing values by loading the PimaIndiansDiabetes2.csv file). We observe a general decline of performance in the data set, suggesting that some of the correlations found in Section 3.4 may have been due to errors in the dataset, which is of concern. The performance of the decision tree has gravely worsened. The bagging methods perform worse aswell, but not to the same degree. The only performance increase is for the AdaBoost ensemble method. This performance decrease may be of stochastic character here aswell, due to the small size of the data set.

Another interesting observation is that the best subset using the GAM model of cubic order was changed when the missing values were removed. The best subset features given this new data set become **glucose**, **age**, **mass** and **pedigree**.

Model	Hyperparameter	Accuracy Test	Accuracy Train
k-Nearest Neighbors	k=16	0.7692	0.7634
GAM (full model)	order basis = 3	0.7692	0.8435
GAM (best subset)	order basis = 3	0.7769	0.8244
Decision Tree (ccp)	$\alpha = 0.01264$	0.7154	0.8664
Bagging (Decision Trees)	max features= 3	0.7692	1.0
Bagging (10-nearest neighbors)	max features= 3	0.7769	0.8168
Random Forest	tree(2 features)	**0.800**	0.8969
AdaBoost	lr=0.1	**0.800**	0.8397

Table 3.8: *Results from all classification methods presented in this assignment on the data set where the missing values have been dropped.*

References

Cassotti, Matteo, Davide Ballabio, Viviana Consonni, Andrea Mauri, Igor Tetko, and Roberto Todeschini (Mar. 2014). "Prediction of Acute Aquatic Toxicity Toward Daphnia magna by using the GA-kNN Method". In: *Alternatives to laboratory animals : ATLA* 42, pp. 31–41. DOI: 10.1177/026119291404200106 (cit. on p. 1).

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830 (cit. on p. 2).

Servén, Daniel and Charlie Brummitt (Mar. 2018). *pyGAM: Generalized Additive Models in Python*. DOI: 10.5281/zenodo.1208723. URL: https://doi.org/10.5281/zenodo.1208723 (cit. on p. 2).