Data Analysis

Instructor: prof. Panagiotidou Sofia

3rd assignment

Aristotle University of Thessaloniki
Department of Mechanical Engineering

Dimitris Nentidis

Contents

1 Introduction

In this assignment, the continuation of the previous two, the task is to classify the data into high or low. Initially, the Classification Tree method is examined, and then ensemble methods based on it, namely Bagging, Random Forest, and Boosting. Finally, a comparison is made regarding various parameters.

2 Classification Trees

To build the classification trees, the Gini Index criterion is used for the selection of each predictive variable.

$$\text{Gini Index} = 1 - \sum_{i=1}^{n} p_i^2$$

where p_i is the probability of a specific class i, and n is the total number of classes.

2.1 Depth 3

For depth equal to 3, the following tree is produced. The root of the tree is adult mortality. Then, in the left branch, the criteria "Income Composition of Resources" appear, which are examined at two different comparison values, and GDP. On the other side, "Income Composition of Resources", "Adult Mortality", and "Schooling" appear.

It is worth noting that the same predictive variables are used more than once at different comparison values. This matches the results from the simple linear regression of the first assignment, where 52.7% of variability was attributed to this predictor with a very small p-value. Similarly, the simple linear regression for Adult Mortality had an R^2 of 0.505 with an equally small p-value.

The training error, 0.08, and the testing error, 0.10, indicate moderate bias, while variance is also moderate. The model as it stands shows higher sensitivity, which is about 0.04 above specificity.

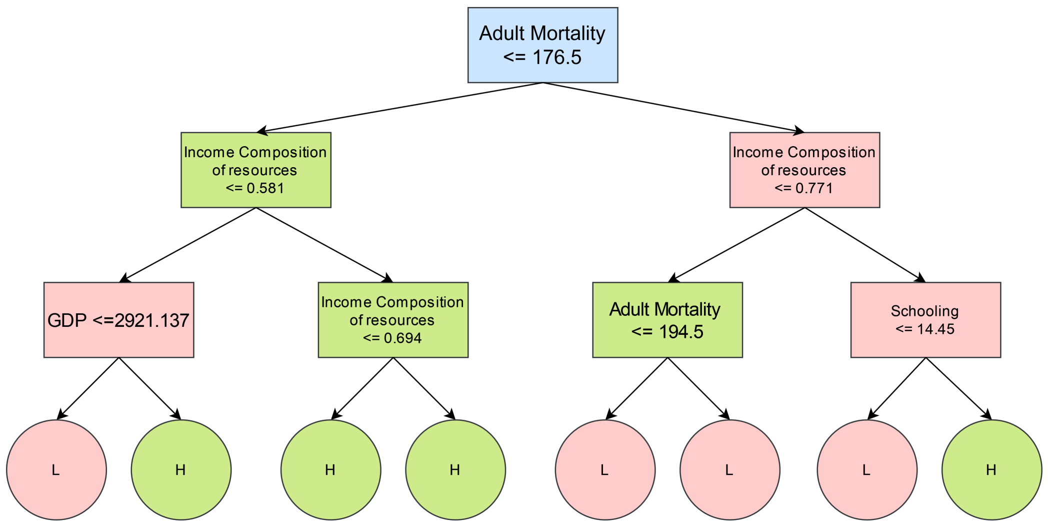
Adult Mortality <= 176.5
Income Composition of resources <= 0.581
Income Composition of resources <= 0.771
GDP <=2921.137
Income Composition of resources <= 0.694
Adult Mortality <= 194.5
Schooling <= 14.45
L
H
H
H
L
L
L
H

Figure 1: Tree of depth 3.

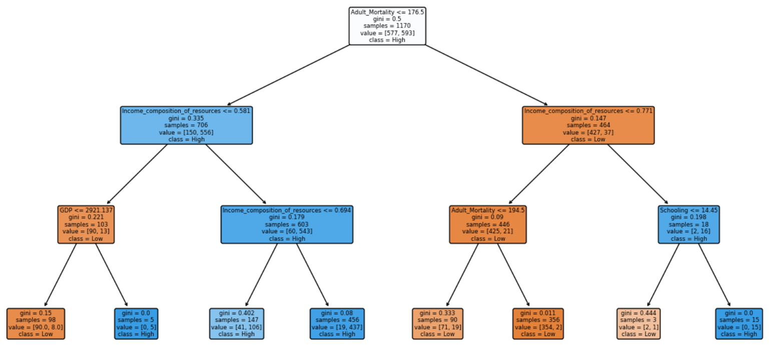

Figure 2: Tree of depth 3.

	Predicted 0	Predicted 1	Support
Training Data			
Actual 0	517	60	577
Actual 1	30	563	593
Testing Data			
Actual 0	134	19	153
Actual 1	11	129	140

Table 1: Confusion Matrix for a Tree of depth 3.

	Precision	Recall	F1-score
Training Data			
0	0.95	0.90	0.92
1	0.90	0.95	0.93
Testing Data			
0	0.92	0.88	0.90
1	0.87	0.92	0.90

Table 2: Classification Report for a tree of depth 3.

It is interesting that some splits lead to leaves with the same output. This happens because such splits reduce the impurity of the leaves, thereby increasing the confidence in each prediction.

2.2 Depth Comparison

Looking at the tree depth values, it can be observed that the testing error generally decreases as depth increases. Having written that, however, the decrease is not significant from depth 3 to 7, with a minimum appearing at depth 8. From there, the error increases. The training error continues to decrease. In fact, after depth 8, the difference between the testing and training error increases.

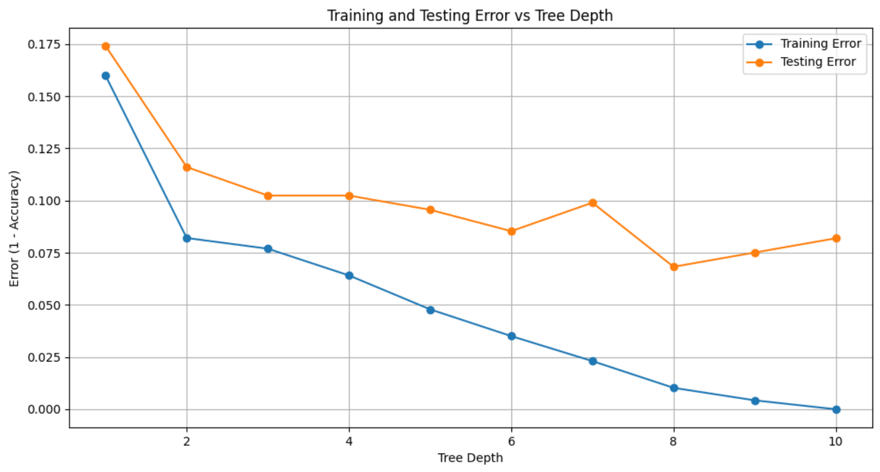

Figure 3: Test error comparison at different depths.

3 Bagging Method

The central idea of the Bagging or Bootstrap Aggregating method is to train multiple base estimators for random subsets of the training data. In this case, a tree model with depth of 8 is used as it previously had the best performance. Each model is trained separately and then through a "majority vote" function, the final model is produced. The number of estimators is essentially the number of models, or trees in this case, to be used. The goal is to reduce overfitting by having each model capture a different part of the dataset.

	Predicted 0	**Predicted 1**	**Support**
Training Data			
Actual 0	576	1	577
Actual 1	0	593	593
Testing Data			
Actual 0	144	9	153
Actual 1	11	129	140

Table 3: Confusion Matrix for Bagging with Decision Trees

	Precision	Recall	F1-score
Training Data			
0	1.00	1.00	1.00
1	1.00	1.00	1.00
Testing Data			
0	0.93	0.94	0.94
1	0.93	0.92	0.93

Table 4: Classification Report for Bagging with Decision Trees

Bagging eliminates the training error while the testing error hovers around 0.07. The bias is very low while the model generalizes well and variance is lower compared to a single tree. With zero training error, both sensitivity and specificity are 1 in the training set, while in the testing set they are 0.92 and 0.94 respectively—also improved compared to one tree.

4 Random Forest Method

The Random Forest method takes the idea of Bagging further by adding an extra layer of randomness. As before, each tree is trained on a different part of the data, but here, at each split of the tree, only a random subset of candidate features is considered. Here, the number of features considered at each split is given as $\sqrt{p}$, where p is the total number of predictors.

	Predicted 0	Predicted 1	Support
Training Data			
Actual 0	577	0	577
Actual 1	0	593	593
Testing Data			
Actual 0	146	7	153
Actual 1	8	132	140

Table 5: Confusion Matrix for Random Forest

	Precision	Recall	F1-score
Training Data			
0	1.00	1.00	1.00
1	1.00	1.00	1.00
Testing Data			
0	0.95	0.95	0.95
1	0.95	0.94	0.95

Table 6: Classification Report for Random Forest

The Random Forest model achieved the best results, with training error of 0.00 and testing error of 0.05. It has very low bias as it fits the training data perfectly, and low variance as indicated by the lowest testing error among all methods. Random Forest effectively reduces variance by averaging multiple trees, resulting in strong generalization and robustness to new data. Sensitivity in the training set is 1 and in the testing set 0.94, while specificity is 1 and 0.95 respectively.

5 Boosting Method

In contrast to the previous two methods where the models are built independently, here the models are built sequentially, with each trying to improve the previous one. In this assignment, the AdaBoost algorithm is used to obtain the result. With the learning rate set to 1, it ensures that each time only the previous tree is taken into account.

	Predicted 0	Predicted 1	Support
Training Data			
Actual 0	557	20	577
Actual 1	15	578	593
Testing Data			
Actual 0	144	9	153
Actual 1	13	127	140

Table 7: Confusion Matrix for AdaBoost

	Precision	**Recall**	**F1-score**
Training Data			
0	0.97	0.97	0.97
1	0.97	0.97	0.97
Testing Data			
0	0.92	0.94	0.93
1	0.93	0.91	0.92

Table 8: Classification Report for AdaBoost

The model developed with AdaBoost showed a training error of 0.03 and testing error of 0.08. Bias is low, but slightly higher than that of Random Forest, indicating it does not perfectly fit the training data. Variance is higher than that of Random Forest. The slight increase in testing error compared to Random Forest suggests it is more sensitive to training data and more prone to overfitting. Sensitivity on the training set is 0.97 and on the testing set 0.91, while specificity is 0.97 and 0.94 respectively.

6 Comparison of n estimators

In all methods tested, changing the number of estimators showed similar behavior. For Bagging, the test error fluctuates around 0.06 with a minimum appearing at 75 estimators. The train error becomes zero for $n > 50$. Similar behavior is observed for the Random Forest models, though the test error remains stable for $n > 100$. While the test error also follows a similar oscillatory behavior, the training error continuously decreases for the Boosting method without becoming zero, even at $n = 400$.

Notably, the models are not optimal for the n given in the instructions. The optimal test error for Bagging is slightly below 0.06, for Random Forest below 0.05, and for Boosting below 0.07.

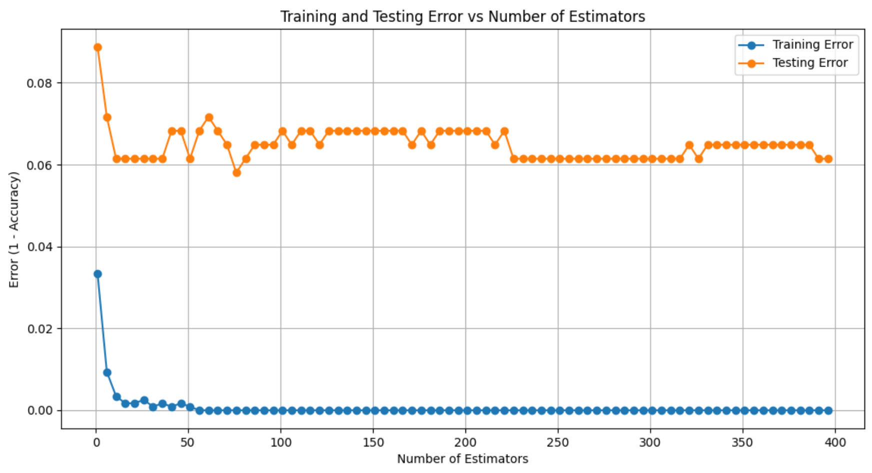

Figure 4: Test error comparison for Bagging with different n.

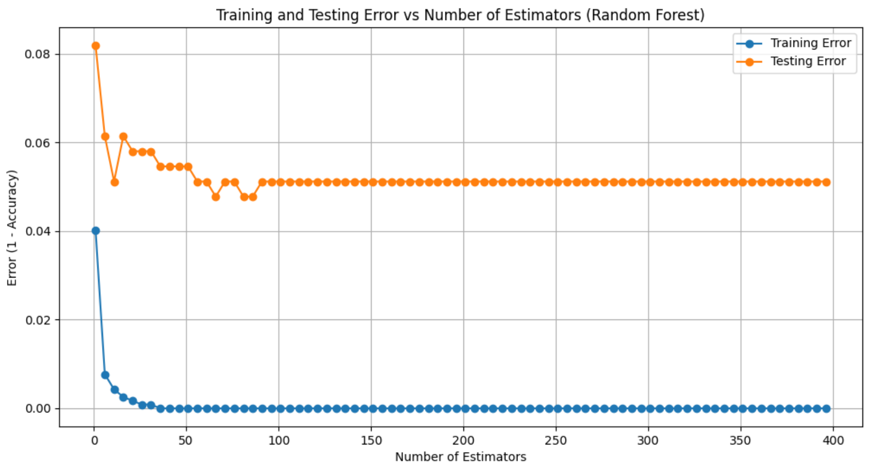

Figure 5: Test error comparison for Random Forest with different n.

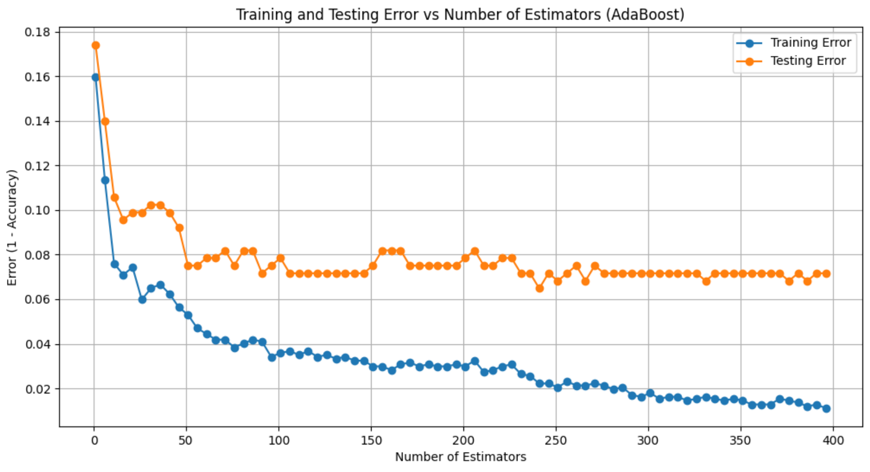

Figure 6: Test error comparison for Boosting with different n.

7 Method Comparison

In evaluating the performance of the various models, it is appropriate to compare the bias and variance of the different methods.

The model resulting from the Classification Tree method has a training error of 0.08 and a testing error of 0.10, the highest among all tested methods, thus showing the highest bias. The training error is not much higher, which indicates that variance is relatively small.

Both the Bagging with Decision Trees method and Random Forest eliminated the training error with testing errors at 0.07 and 0.05 respectively. The bias is minimal for both. Random Forest generalizes slightly better, avoiding overfitting more effectively and having the lowest variance of all the methods examined. Boosting showed a training error of 0.03 and testing error of 0.08.

It is clear that Random Forest achieves the highest accuracy, which is expected, as it is one of the ensemble methods.

Method	**Training Error**	**Testing Error**
Decision Tree	0.08	0.10
Bagging with Decision Trees	0.00	0.07
Random Forest	0.00	0.05
AdaBoost	0.03	0.08

Table 9: Summary of Training and Testing Errors for Different Methods

Finally, comparing these values with the results from the previous assignment, it becomes clear that these methods are far more effective for classifying values. Even in the best configuration, with threshold = 0.4, LDA had training error equal to 0.1145 and testing error of 0.088–comparable to the results of a single tree.