LINGI2262 : Assignment 2 - Decision Tree Learning

Group 8: Maxime Dimidschstein, Doriane Olewicki

March 2, 2018

1

1.1

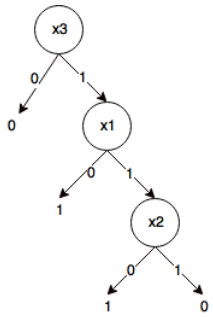

Figure 1

At state 1, we need to choose for the set y which attribute will maximise the gain.

$$Entropy(y) = \frac{1}{2} \cdot log_2(2) + \frac{1}{2} \cdot log_2(2) = 1$$
$$Gain(y, x_1) = 1 - \frac{2}{4} \cdot (\frac{1}{2} \cdot log_2(2) + \frac{1}{2} \cdot log_2(2)) - \frac{2}{4} \cdot (\frac{1}{2} \cdot log_2(2) + \frac{1}{2} \cdot log_2(2)) = 0$$
$$Gain(y, x_2) = 1 - \frac{2}{4} \cdot (\frac{1}{2} \cdot log_2(2) + \frac{1}{2} \cdot log_2(2)) - \frac{2}{4} \cdot (\frac{1}{2} \cdot log_2(2) + \frac{1}{2} \cdot log_2(2)) = 0$$
$$Gain(y, x_3) = 1 - \frac{1}{4} \cdot 0 - \frac{3}{4} \cdot (\frac{2}{3} \cdot log_2(\frac{3}{2}) + \frac{1}{3} \cdot log_2(3)) = 0.3113$$
$$Gain(y, x_4) = 1 - \frac{4}{4} \cdot 1 = 0$$

We thus choose to go with attribute x_3.
We will continue those calculation for each level until each subsets of y only contains 1 or 0.
At stage 2, we chose x_1 with a gain of 0.2516 (as x_2), while x_4 would have brought a gain of 0.
At stage 3, we chose x_2 with a gain of 1. x_4 would still have brought a gain of 0.

1.2

In figure 2, we have an example of a tree with two levels that perfectly classify the training examples. This tree is not given by ID3 because it does not maximise the gain during the construction of the tree.

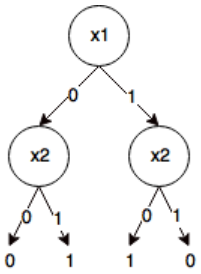

Figure 2

2

2.1

$$Entropy(S) = \frac{10}{20} \cdot log_2(2) + \frac{10}{20} \cdot log_2(2) = 1$$
$$Gain(S, x_1) = 1 - \frac{1}{2}(\frac{8}{10} \cdot log_2(\frac{10}{8}) + \frac{2}{10} \cdot log_2(\frac{10}{2})) - \frac{1}{2}(\frac{8}{10} \cdot log_2(\frac{10}{8}) + \frac{2}{10} \cdot log_2(\frac{10}{2})) = 0.2781$$
$$Gain(S, x_2) = 1 - \frac{16}{20}(\frac{10}{16} \cdot log_2(\frac{16}{10}) + \frac{6}{16} \cdot log_2(\frac{16}{6})) - \frac{4}{20} \cdot 0 = 0.2365$$

In this case, the gain is maximised by using the first split.

2.2

$$Entropy(S) = \frac{10}{20} \cdot log_2(2) + \frac{10}{20} \cdot log_2(2) = 1$$
$$Gain(S, x_1) = 1 - \frac{1}{2}(\frac{80}{82} \cdot log_2(\frac{82}{80}) + \frac{2}{82} \cdot log_2(\frac{82}{2})) - \frac{1}{2}(\frac{20}{28} \cdot log_2(\frac{28}{20}) + \frac{8}{28} \cdot log_2(\frac{28}{8})) = 0.4857$$
$$Gain(S, x_2) = 1 - \frac{16}{20}(\frac{100}{106} \cdot log_2(\frac{106}{100}) + \frac{6}{106} \cdot log_2(\frac{106}{6})) - \frac{4}{20} \cdot 0 = 0.7489$$

This time, the gain is maximised by using the second split.

2.3

Without cost imbalance, none of the class is preferred when splitting. Here in section 2.1, we take the split where both positive and negative example are as well classified instead of the split that puts all the positive examples in one place. Meanwhile, in section 2.2, we see that increasing the cost of positive examples lead us to pick the second split.
Clearly, the cost of the two miss-classified positive examples from the first split has become way too high.

We don't need to physically replicate the positive examples to get the desired effect. Indeed, what we truly want to influence is the computation of the entropy. Replicating the examples is just an easy and visual way to manipulate the proportions that are used in the entropy equation. We can thus directly change those by setting :

$$p'_+ = \frac{10\ p_+}{10\ p_+ + p_-}$$
$$p'_- = \frac{p_-}{10\ p_+ + p_-}$$

The end result will then be exactly the same.

3

For d attributes having each k possible values, we can choose between d different roots. Once the root is picked, we still have to build trees with $d-1$ attributes having each k possible values. And that is for each of the k values of the root.
With this reasoning, we get a recurrence relation :

$$Trees(d,k) = d\ (Trees(d-1,k))^k$$

With $Trees(d,k)$ the number of trees with parameters d, k as described, and with base case $Trees(1,k) = 1$.
Solving this equation yields

$$Trees(d,k) = \prod_{i=0}^{d-1} (d-i)^{k^i}$$

ID3 will consider only a fraction of the trees. First, it will try all d attributes as root node, and then pick one. The trees with other roots won't be developed any further. Then, the same is done with $d-1$ attributes for all k children. This is repeated until the end of the tree.
In the end, if we take the unfinished trees into account, we get $\sum_{i=1}^{d} i.k^{d-i}$. If we only consider the full trees, we have k^{d-1} of them.

4

As we have two attributes, we are working in the plane $\mathbb{R}^2$. Splitting is thus always done along one of the main axis. That means those splits define rectangles in the instance plane.
The number of regions depends both on the attribute values and on the number of classes. If the attribute values are well gathered, we will need less splits than if they are completely mixed. On the other hand, the more classes there are, the more classes could be represented in the training set, and the more splittings we might need.

5

For this question, we had to build a decision tree on the data "playTennis". We obtained the figure 3.
We were not able to reproduce the decision tree presented during the lecture because `rpart` CART implementation builds binary trees. It still tries to maximise the gain at each step, so it may not be the most efficient to build a tree testing first all the values of the outlook (which would look more like the decision tree from the slide). That's why the tree is here different.
To use `rpart` CART implementation, we needs to modify some control parameters : the minsplit is now at 1, so that there is no pre-pruning, minbucket was set to 1 also to have a coherent value regarding minsplit and the value of cp was put to 0.0 so there is no pruning.

6

When building a learning tree for the BostonHouseTrain data without pruning, we got 70 leaves. As we can observe it while predicting the output for the training examples, the training set accuracy is guaranteed to be 100% with CART without pruning. In fact, the tree fits perfectly the data. But it may over-fit the data, that's why we have only 71.7% accuracy on the prediction for the test examples.

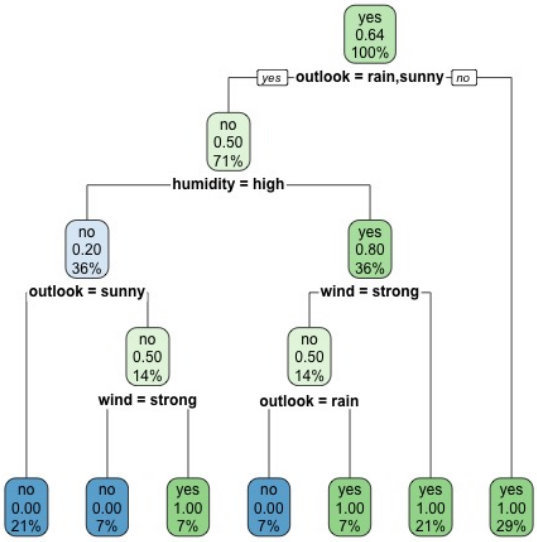

Figure 3: Decision Tree on the data PlayTennis

7

We sampled 25% of the training that and observed the variations over the number of leaves, the accuracy when predicting the training examples and the accuracy when prediction the testing examples.

Number of leaves	Accuracy for training	Accuracy for testing
19.0	1	0.5943396
22.0	1	0.7264151
21.0	1	0.6509434
19.0	1	0.8018868
26.0	1	0.7452830

We can observe that the number of leaves varies a little. The mean number of leaves is 21.4. The accuracy for the training examples is 1 each time, because we still have a perfect fit at the moment, not using pruning. The accuracy for the testing examples varies a lot between 60% and 80% : this shows that the accuracy obtains depends a lot on the training examples given. Luck can bring us a really good sample of training examples that creates a really good model but it may also do the opposite.

8

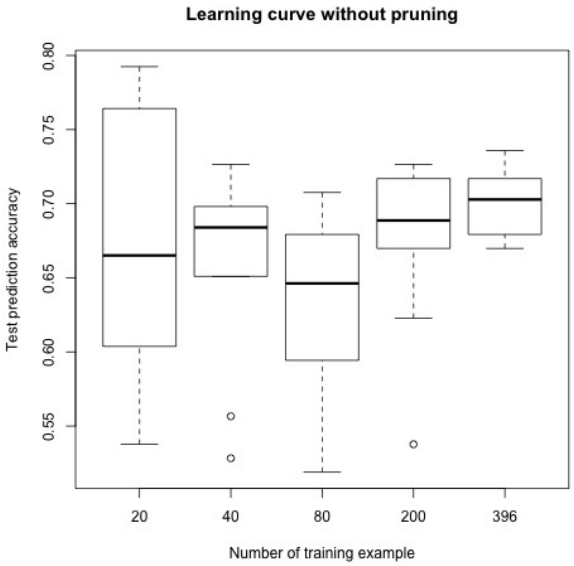

Figure 4: Learning curve without prunning

In the figure 4, it can be observed that the learning curve acts exponentially, increasing a lot at the beginning then slowing down. The number of training example does not need to be really big to get a better model and it is almost impossible to get a 100% accuracy on the test prediction.

9

In the following section, we will analyse graphs regarding the impact of pruning on the learning curve and on the number of leaves in the tree. This second information will allow us to speak about the size of the tree : the most leaves there are, the larger the tree is. For all those graphs, we made 30 measurements to have the best view on the results.
For pre-pruning, we observe that the accuracy stays mostly the same as when we are not pruning at all. But the interesting part is the amount of leaves. We get around twice less leaves thanks to pre-pruning, which is a huge improvement considering the steady accuracy.
Post-pruning gives similar accuracy for lower numbers of training examples. But once this number starts to grow, the accuracy is significantly improved. The total number of leaves is also drastically reduced, even when we handle many training examples.
Finally, combining the two gives us the best of both worlds. On one side, the accuracy is increased when the number of training sets is not too low. On the other side, we get less leaves than with any other technique.
We can conclude by saying the right balance between pre- and post-pruning is clearly what gives the best results, both in terms of accuracy and in terms of tree size.

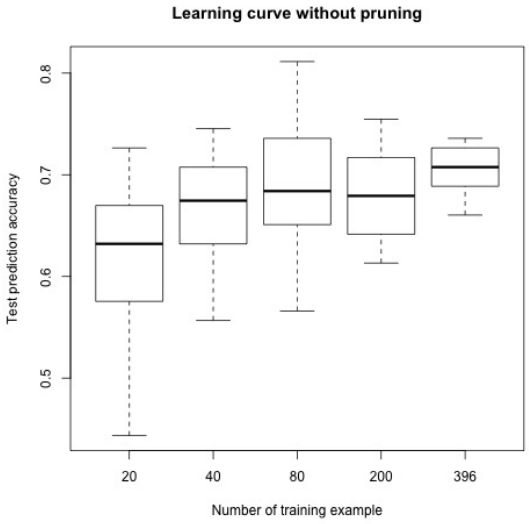

Figure 5: Accuracy when not using pruning

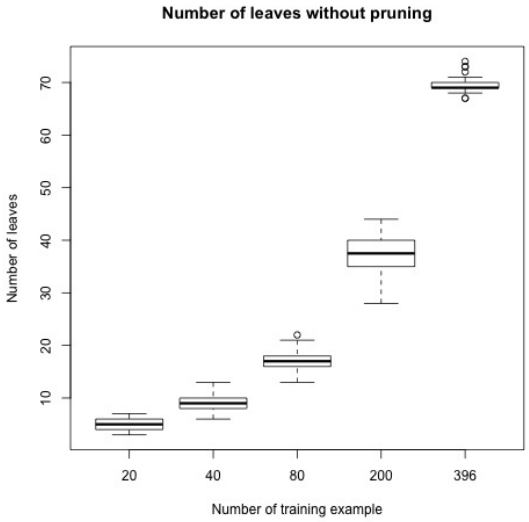

Figure 6: Number of leaves when not using pruning

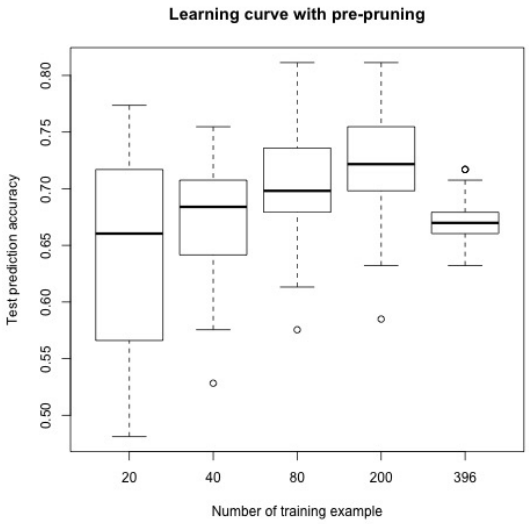

Figure 7: Accuracy when using pre-pruning

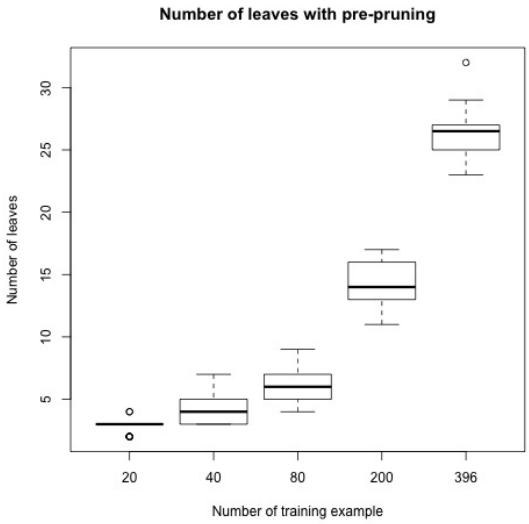

Figure 8: Number of leaves when using pre-pruning

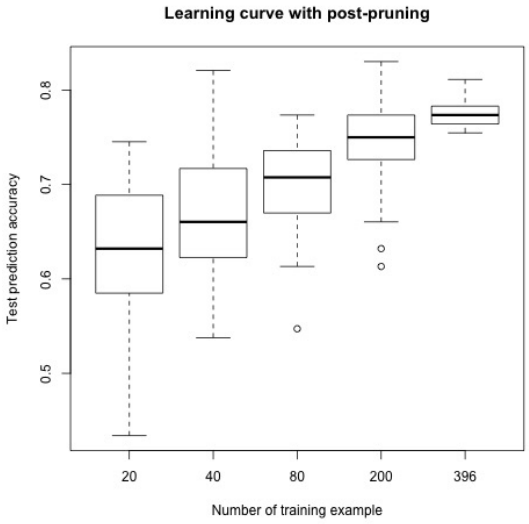

Figure 9: Accuracy when using post-pruning

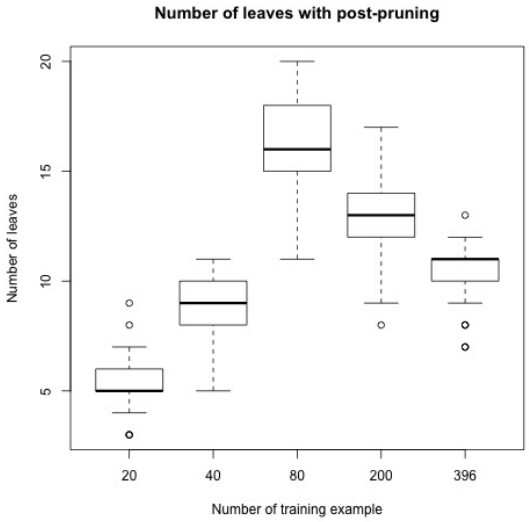

Figure 10: Number of leaves when using post-pruning

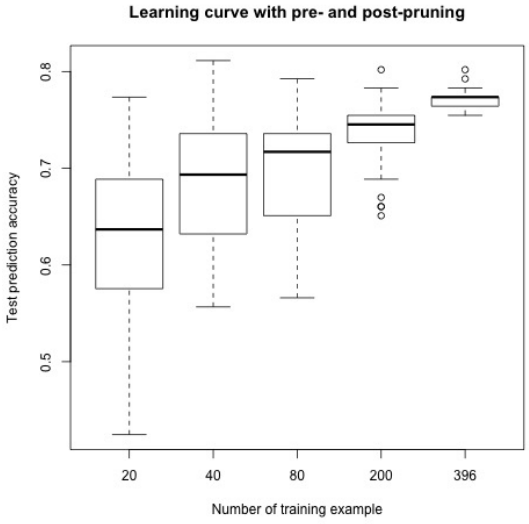

Figure 11: Accuracy when using pre- and post-pruning

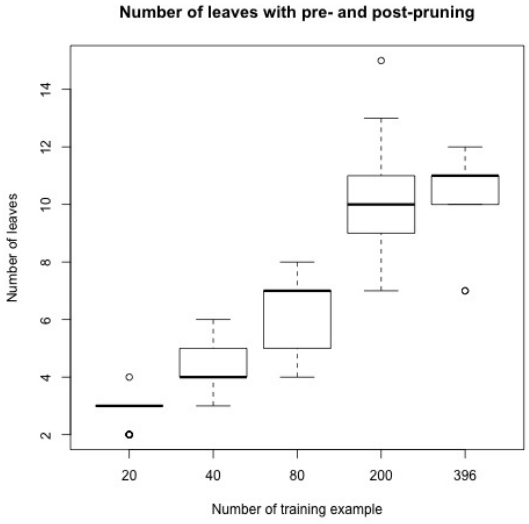

Figure 12: Number of leaves when using pre- and post-pruning

10

To check which features were the most relevant in our model, we checked which of them gave the highest accuracy when building a model only on them. We thus saw that a model based only on the data "lstat" gave an test prediction accuracy of 70%. An other model based on the data "nox" gave a result of 63% accuracy. Finally, a model based on the data "rm" returned an accuracy of 62%. All the other models returned results between 40% and 60%, except for the model based on "chas". In fact, for this one, the value was almost each time zero, while the "class" changed a lot. We could thus not get any information from it.

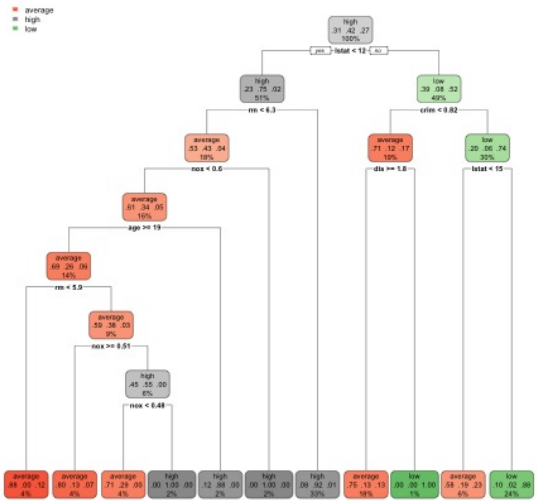

Figure 13: Decision tree with pruning

In the figure 13, we can see that in fact, the features "lstat", "nox" and "rm" are mostly used.

11

For the last part of the assignment, we analysed the effect of bagging on decision trees. We first applied the bagging on the all training set and checked the accuracy of the test prediction. Previously, using only one decision tree, we had 71.7% accuracy. Now, using a number of bootstrap rounds of 10, we obtain the result of 80.2%. The bagging worked and we got a better accuracy.
We now needed to test the right number of bootstrap rounds to use. As it can be seen on the figure 14, having only a few bootstrap rounds gives already quiet a good accuracy. However, once we reach a certain number of bootstrap rounds (around 5 or 10), the accuracy improves and stays around 75% to 85%. The curve seems to follow an exponential behaviour. It is does not necessary to do much more bootstrap rounds, knowing it can be costly. We recommend to use to use around 10 bootstrap round to have a better accuracy.

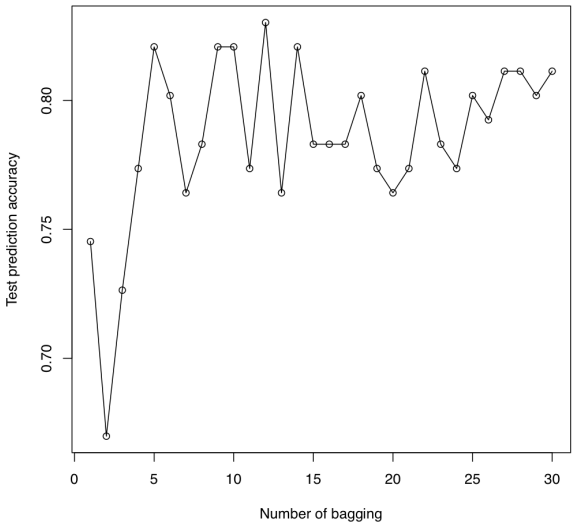

Figure 14: Study of the effect of bagging on decision trees