Mandatory assignment 2, STK2100

Jonas Øren

April 22, 2019

Problem 1 (*a*) Only figure *A* could have been produced by a regression tree, because the process recursively divides the space into new regions, by dividing previous regions into two new ones, where the dividing line is parallel to the axis of a predictor. This produces areas that are strictly rectangles.

- Figure *C* contains a line that is not parallel to any axis.
- Figure *B* and *D* have regions that are not rectangles.

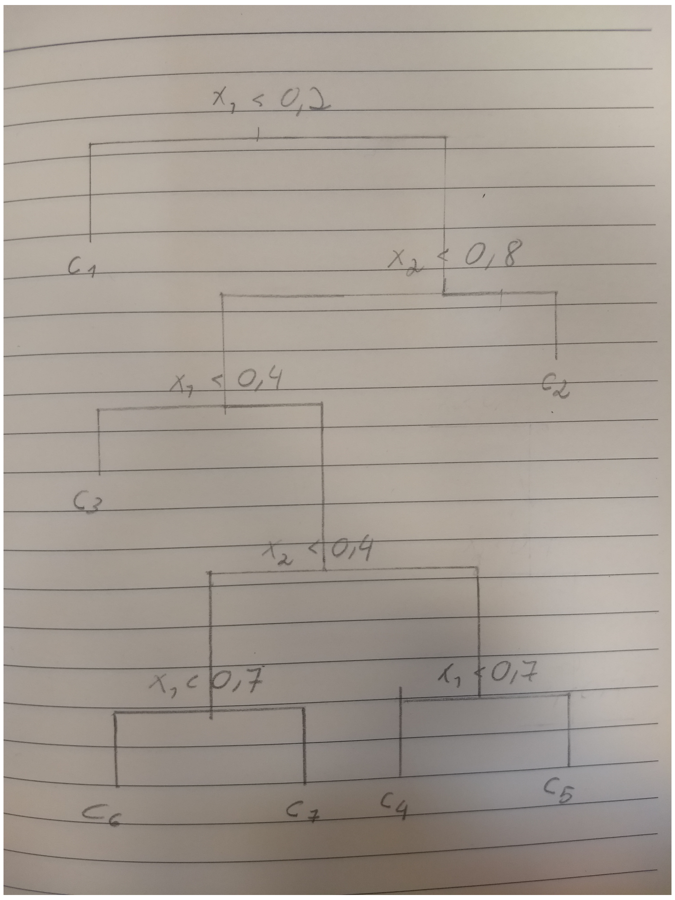

Figure 1: Sketch of the tree.

(b) The regression tree is shown in Figure 1 on the preceding page:

(c) The estimates are:

$$\begin{aligned}
(i)\ & f(x_1 = 0.5, x_2 = 0.5) &= c_4 \\
(ii)\ & f(x_1 = 0.3, x_2 = 0.5) &= c_3 \\
(iii)\ & f(x_1 = 0.5, x_2 = 0.3) &= c_6 \\
(iv)\ & f(x_1 = 0.95, x_2 = 0.05) &= c_7 \\
(v)\ & f(x_1 = 0.05, x_2 = 0.95) &= c_1 \\
(vi)\ & f(x_1 = 0.45, x_2 = 0.75) &= c_4
\end{aligned}$$

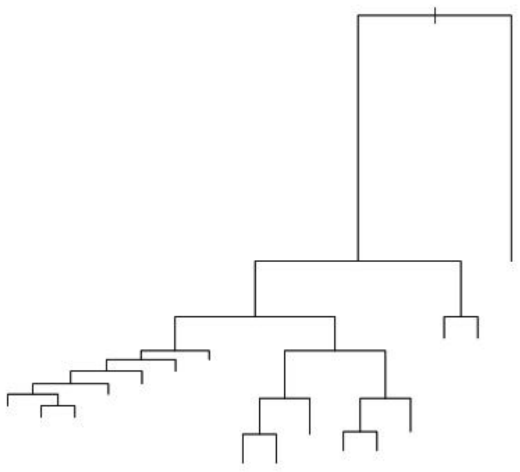

Figure 2: Tree before pruning.

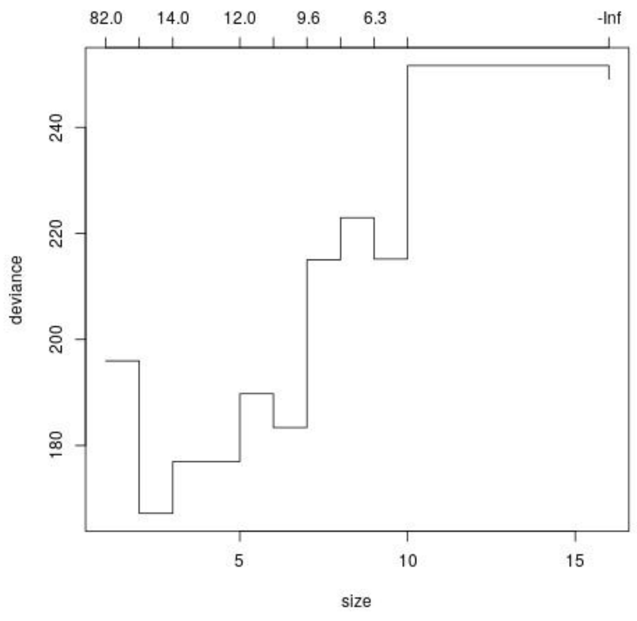

Figure 3: Deviance vs complexity, found with cross-validation.

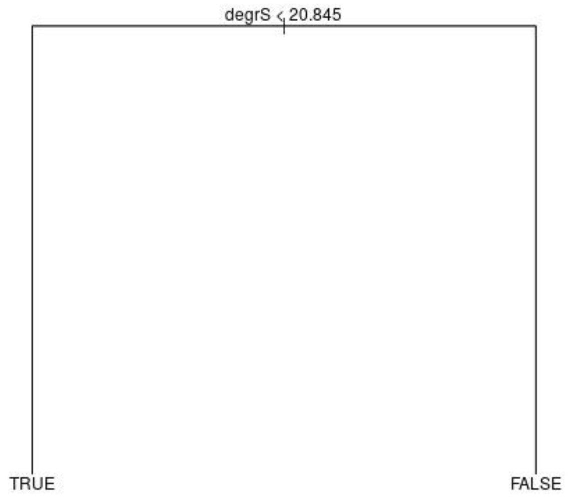

Figure 4: Tree after pruning.

Problem 2 (*a*) I set healthy patients as ”TRUE” and sick patients sa ”FALSE”. I then divide the data into healthy and unhealthy patients, then divide both into test and training data, and the training data into a grow and pruning set, and finally combine the sets of healthy and unhealthy patients. I first grow a tree on the grow set without any restrictions. Figure 2 on page 2 shows the result.

(*b*) Using cross validation i find the relationship between deviance and tree size shown in Figure 3 on the preceding page.

I then prune the tree (now based on misclassification error, and using the pruning data) and get the decision tree shown in Figure 4 on the previous page (*c*) I compute the misclassification errors to be

```
> err.original
[1] 0.2427184
> err.pruned
[1] 0.2330097
```

The pruned tree now does slightly better than the original, but still the improvement is not great, and the model still misclassified almost every fourth observation.

The dataset is not very large, and I did not expect classification of disease based on risk factors to be very precise but I did expect the difference in performance between the simpler model and the complex model to be bigger, especially since such a big difference in complexity was chosen.

(*d*) Applying LDA and QDA gives the following prediction errors:

```
> err.lda
[1] 0.1650485
> err.qda
[1] 0.184466
```

In this case the less complex LDA model gives the best predictions on the training set. And LDA also performs better than classification trees.

(*e*) With three classes i get the unpruned tree shown in Figure 5 on the following page, deviance found with cross validation shown in Figure 6 on the next page and the pruned tree shown in Figure 7 on page 7. The cross-selection only chooses one node, and never predicts the third class. Computed errors using all four methods are:

```
> err.original
[1] 0.2038835
> err.pruned
[1] 0.1941748
> err.lda
[1] 0.1165049
> err.qda
[1] 0.1359223
```

With more classes the performance is better in this case, but subsequent runs show that the errors vary widely from run to run, and I should not infer too much from one run. But LDA still consistently performs best of all the models.

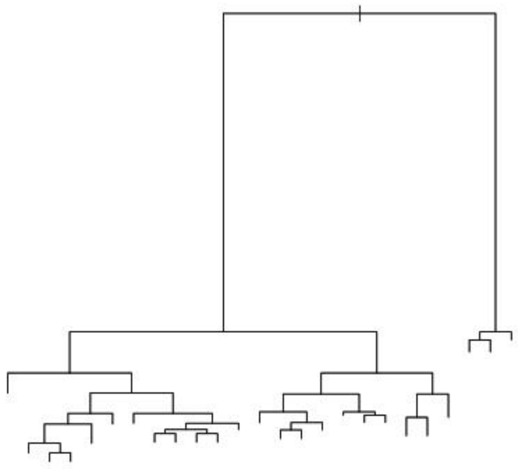

Figure 5: Tree before pruning.

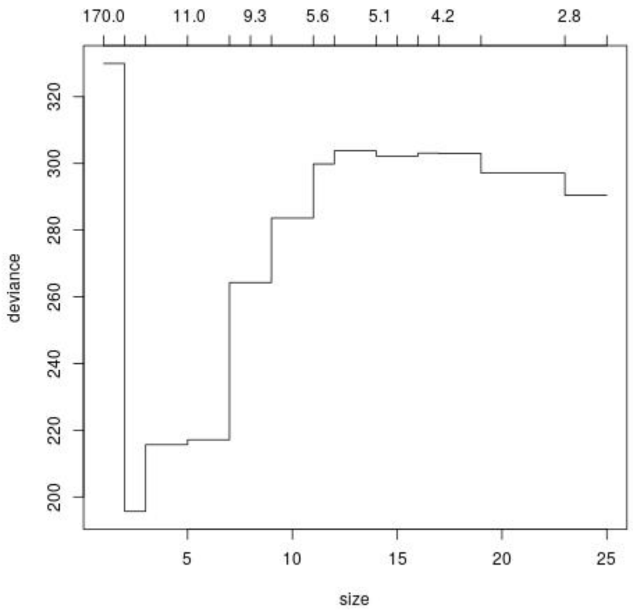

Figure 6: Deviance vs tree-complexity with three classes, found with cross-validation.

(*f*) Report:
You can use the tree found in Figure 4 on page 3 to classify if a patient is sick or healthy by starting at the root of the tree and following the nodes downwards until you reach a prediction for the patients health. At each note you go left if the statement is true, e.g. at the root node go left if "degrS $<$ 15.155". The prediction "TRUE" means the patient is healthy.

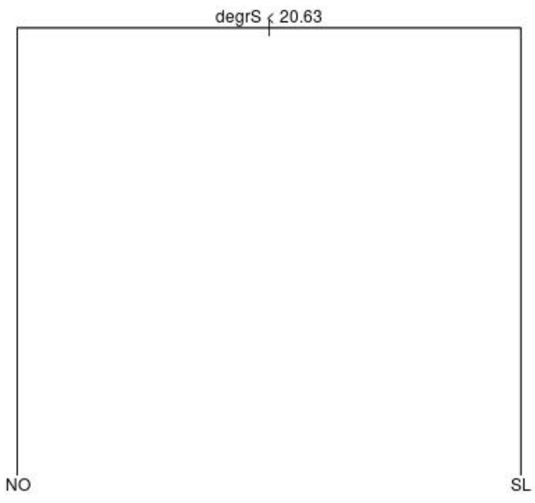

Figure 7: Tree after pruning.

Problem 3 A plot of all fitted models are shown in Figure 10 on page 9.

(*a*) I randomly split the data into a training set, and a test set of equal size. I then randomly split the training set into 5 equal parts and find the optimal number of neighbors for the kNN algorithm using 5-fold cross validation.

I find the optimal k to be 6. Figure 8 on the next page shows the cross validation mean squared error for each k. Because the data follows a very strong pattern, the algorithm chose a small number for k, meaning each point is chosen according to the 6 points nearest the chosen point, resulting in a funciton that follows the pattern in the data closely.

(*b*) Again using 5-fold cross-validation, I narrow down an interval for lambda, and find an optimal lambda close to 5e-5, meaning almost no smoothing is applied to the function. Again this is because there is such a strong pattern apparent in the data, that the estimated function wants to follow it closely.

Figure 9 on page 9 shows the cross validation mean squared error for each lambda in the narrowed down interval.

There is very little difference between the smoothing spline and the non-penalized spline model, but I had expected there to be even less difference as the smoothing parameter is so close to zero that its effect is negligible. I would have expected the smoothing spline to essentially take the same form as a non-penalized cubic spline.

(*c*) The mean squared prediction error for the three models on the test set is:

```
$`kNN model`
[1] 86.09558

$`Non penalized cubic spline`
[1] 78.44606

$`Smoothing spline`
[1] 83.95698
```

The non penalized spline performed the best, but the difference between the prediction errors are small, and I would expect some variability in prediction errors from dataset to dataset, so the models seem to predict about equally well.

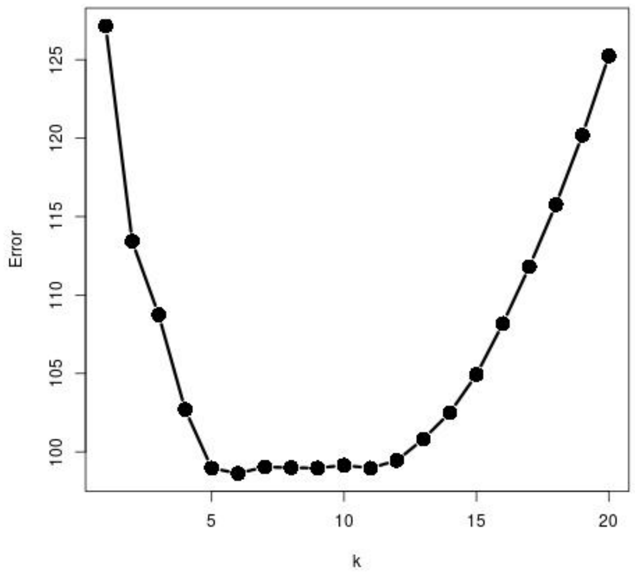

Figure 8: Cross validation error for each k.

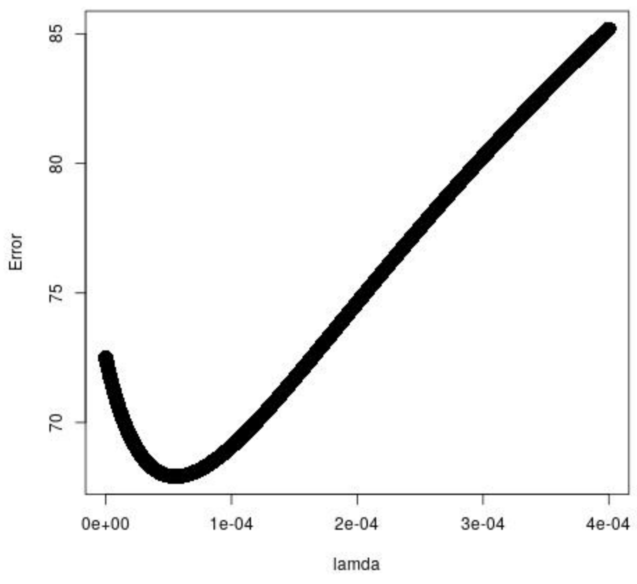

Figure 9: Cross validation error for each lambda.

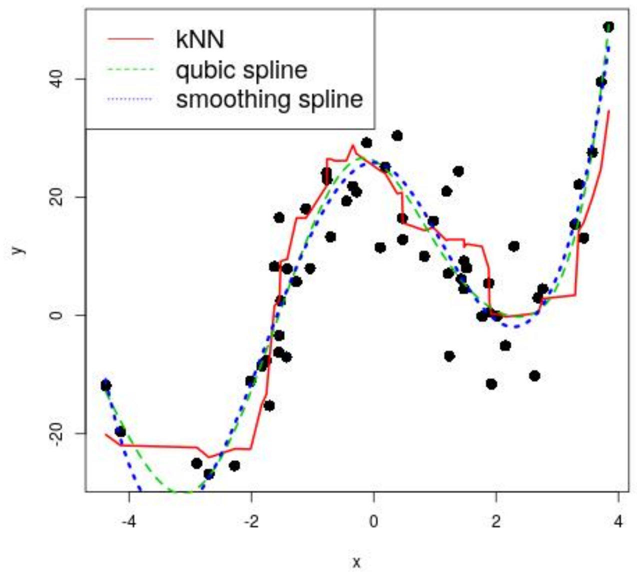

Figure 10: Training points and fitted models.