

MSIS 5223: Tutorial 7 – Classification in R

If you have Crawley's book *The R Book* you should read Chapter 23.

Instructions

The purpose of this tutorial is to help you become familiar with using R. This reading covers a classification using decision trees. Decision trees are very good for exploratory purposes in probing your data. This exercise will use the following data files:

- Pollute.txt
- Epilobium.txt

1. Regression Trees

The first tree this exercise looks at is a regression tree because the dependent and independent variables are continuous. The variables in the data include *Pollution*, *Temp*, *Industry*, *Population*, *Wind*, *Rain*, and *Wet.days*. *Pollution*, the dependent variable, is the SO₂ concentration.

The tree function in R is simply `tree(data)` where *data* is the name of your dataframe. The tree function uses binary recursive partitioning. By default, the tree function uses Deviance, but you can switch to the Gini Index. An important aspect of the `tree(data)` function is it assumes the first column of the dataframe is the target variable. Ensure the dataframe you use is formatted with the first column as the target variable. To use the tree function, install the tree library and load it using the following code:

```
install.packages("tree")  
library(tree)
```

During the process of installing the library, you will be asked to select a CRAN mirror. A CRAN mirror is a server that stores the files. Select the nearest one to you. Since I am in Oklahoma, I always select the one in Texas, which is the company RevolutionAnalytics (how nice of them!). Once installed, simply load the library. Using the pollution data, you can run the function and obtain results as shown in Figure 1.

```

> pollute_tree = tree(pollute_data)
> pollute_tree
node), split, n, deviance, yval
* denotes terminal node

1) root 41 22040 30.05
 2) Industry < 748 36 11260 24.92
   4) Population < 190 7 4096 43.43 *
   5) Population > 190 29 4187 20.45
     10) Wet.days < 108 11 96 12.00 *
     11) Wet.days > 108 18 2826 25.61
       22) Temp < 59.35 13 1895 29.69
         44) Wind < 9.65 8 1213 33.88 *
         45) Wind > 9.65 5 318 23.00 *
       23) Temp > 59.35 5 152 15.00 *
 3) Industry > 748 5 3002 67.00 *
>

```

Figure 1 Output of the Tree Function

The terminal nodes are designated with an asterisk, *. The top of the output contains the column headers of the output; it includes the following:

- node): the node number in the tree
- split: the split criterion for the node
- n: the sample size going into the split
- deviance: the deviance score at that node
- yval: the mean of the target variable within the node

If you sum up the sample size, n , for each of the terminal nodes, you will get the total sample size of the data. In this case, the total $n = 41$. The results of the tree can be plotted for a visual representation, as seen in the figure below.

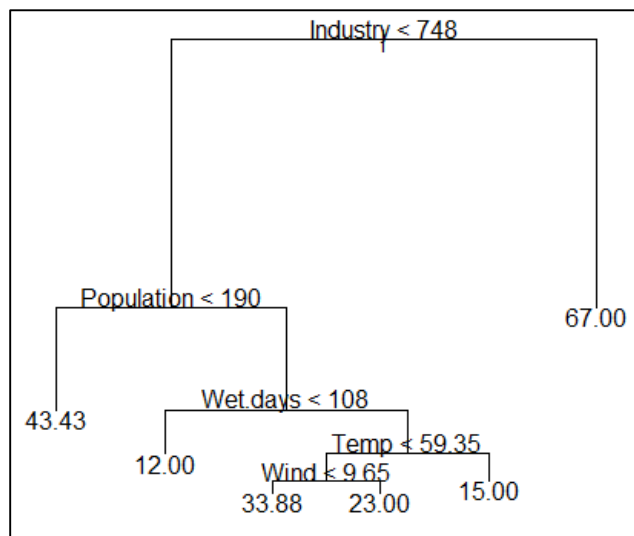


Figure 2 Tree of Pollution

The output reveals *Industry* is the first split and represents the most important variable for splitting. To help explain the process of the tree operation, the following plot is created. On the y-axis is the target, *Pollution*; on the x-axis is *Industry*.

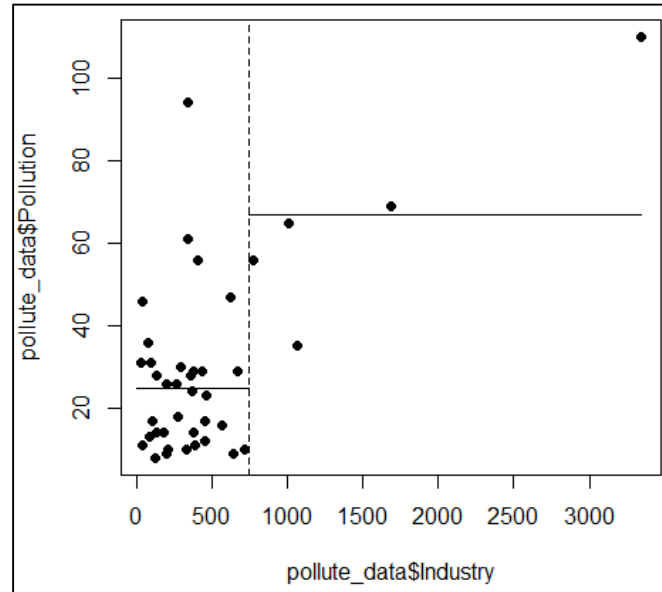


Figure 3 *Industry* Plotted Against *Pollution*

Each independent variable is assessed by determining how much deviance in the dependent variable is explained. In this case, *Industry* explains the most for *Pollution*. The Deviance is based on a threshold value for each predictor; it produces two mean values: 1 above the threshold, and one below. The value 748 (the dashed, vertical line in Figure 3) is the chosen threshold value of *Industry* based on deviance.

The mean values for the two groups are shown via the two horizontal lines. Both of these mean values are used to calculate the deviance. The algorithm iterates through all x-values of *Industry* as the threshold (i.e. the vertical line); a mean below and above the threshold is calculated (i.e. the horizontal lines). For each threshold, the x-value with the lowest deviance is chosen. The data set is then split based on this threshold value. The program then runs through the other independent variables for each of the new data subsets. This continues until further deviance is not possible or there are too few data points (fewer than 6 cases is default).

Returning to the pollution data, the interpretation of the tree is as follows. At the first split, *Industry* values 748 or greater lead to pollution problems with a mean of 67.00; only a sample of 5 is in this split. For *Industry* values less than 748, *Population* explains

what occurs; this branch contains the other 36 data points. Low populations have a high mean of 43.43! For high levels of population, the number of wet days is a key factor, with a mean value of 12.00. Note, all these values come from terminal nodes. See the output in Figure 1 or Figure 2.

One of the most valuable characteristics of trees is the simplicity of their solutions. Often, trees will produce a large number of terminal nodes with too many levels. For the Pollution Tree, it contains four levels, or splits, after the root. The tree also has six terminal nodes. Some researchers recommend a tree with no more than three levels. This creates a very simple tree. The tree library in R comes with a function that helps with pruning a tree. This function is `prune.tree(data)`.

```
> pollute_prune_tree = prune.tree(pollute_tree)
> pollute_prune_tree
$size
[1] 6 5 4 3 2 1
$dev
[1] 8876.589 9240.484 10019.992 11284.887 14262.750 22037.902
$k
[1] -Inf 363.8942 779.5085 1264.8946 2977.8633 7775.1524
$method
[1] "deviance"
attr(,"class")
[1] "prune" "tree.sequence"
>
```

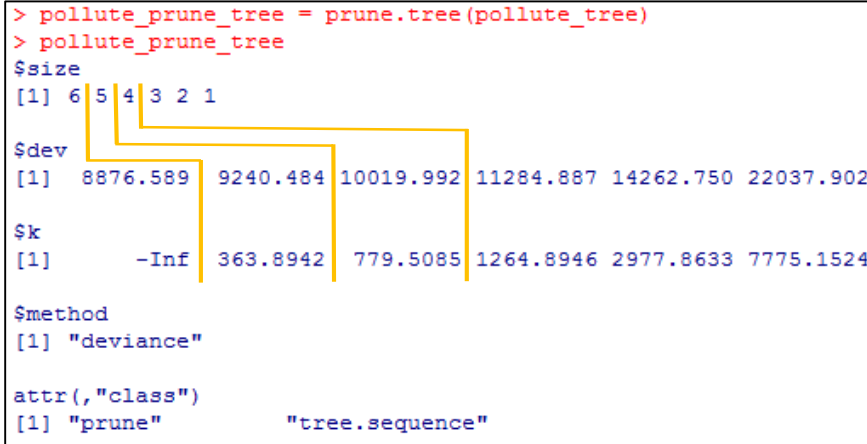


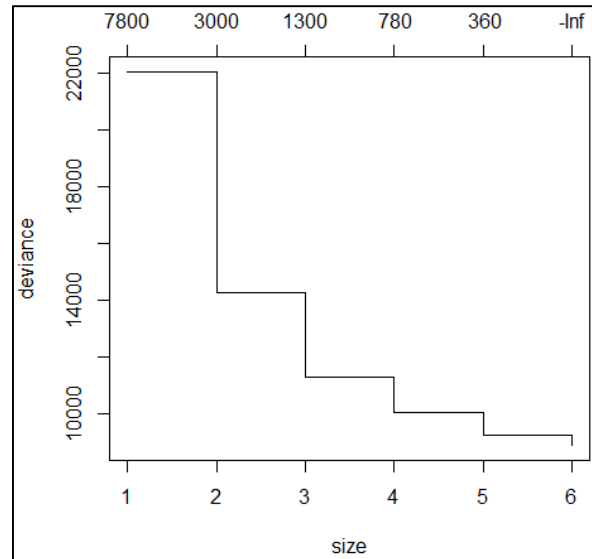
Figure 4 Using the Pruning Function in R

Figure 4 presents the results of the pruned tree. I have drawn some lines to help you read the output. The first part of the output indicates that six different trees were created with the value representing the number of terminal nodes within each. The first value, 6, is a tree with six terminal nodes; the second value, 5, is a tree with five terminal nodes.

The next part of the output is denoted with `$dev`, which provides the total deviance of each tree. The 6-terminal node tree has a total deviance of 8876.59; the 5-terminal node tree has a total deviance of 9240.48. Note, as the number of nodes decreases, the total deviance decreases.

The last part of the output, `$k`, provides the cost-complexity. The first tree, the one with 6-terminal nodes, has a negative, infinite value. That is, it has the lowest

possible bound. As the number of nodes decreases, the cost-complexity decreases (yes, a higher value of k is a lower complexity value). You can plot the pruned tree object to compare the cost-complexity and deviance. In my case, since the pruned tree is `pollute_prune_tree`, I simply type `plot(pollute_prune_tree)`.



The tree with the best values for cost-complexity and deviance is a tree with size 3 or 4. Remember, deviance can be thought of as the measurement of explanatory power of a tree. After 3, the loss in deviance is minimal. Run the tree again with a limitation of 3 terminal nodes. This is done using the pruning function with the `best=""` argument.

```
> #### Use the attribute best=3 to select the best tree
> #### using 3 terminal nodes
> pollute_prune_tree2 = prune.tree(pollute_tree, best=3)
> pollute_prune_tree2
node), split, n, deviance, yval
* denotes terminal node

1) root 41 22040 30.05
 2) Industry < 748 36 11260 24.92
   4) Population < 190 7 4096 43.43 *
   5) Population > 190 29 4187 20.45 *
 3) Industry > 748 5 3002 67.00 *
```

Figure 5 Tree with 3 Nodes

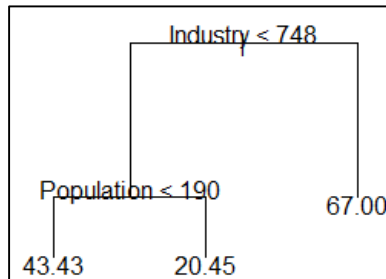


Figure 6 Pollution Tree with 3 Nodes

2. Classification Trees

The data for the classification tree contains various plant flora and fauna. It contains 9 columns of data with 9 rows of data. There is only one entry per species for a total of 9. Thus, the goal is to create a key to classify these plants with each row given its own node. The data is presented below in Figure 7. All the data is categorical, so we are dealing with a classification tree.

```
> epi_data
```

	species	stigma	stem.hairs	glandular.hairs	seeds	pappilose	stolons	petals	base
1	hirsutum	lobed	spreading	absent	none	uniform	absent	>9mm	rounded
2	parviflorum	lobed	spreading	absent	none	uniform	absent	<10mm	rounded
3	montanum	lobed	spreading	present	none	uniform	absent	<10mm	rounded
4	lanceolatum	lobed	spreading	present	none	uniform	absent	<10mm	cuneate
5	tetragonum	clavate	appressed	present	none	uniform	absent	<10mm	rounded
6	obscurum	clavate	appressed	present	none	uniform	stolons	<10mm	rounded
7	roseum	clavate	spreading	present	none	uniform	absent	<10mm	cuneate
8	palustre	clavate	spreading	present	appendage	uniform	absent	<10mm	rounded
9	ciliatum	clavate	spreading	present	appendage	ridged	absent	<10mm	rounded

Figure 7 Lowland British Species in Genus *Epilobium*

To produce a tree that fits the data perfectly, set `mindev=0` and `minsize=2`, if the limit on tree depth allows such a tree. Ideally, the minimum deviance should be 0, but that leads to problems; give it a value as close to 0 as possible, such as 10^{-6} . Create a regression within the function.

```
> epi_tree = tree(species~., epi_data, mindev=1e-6, minsize=2)
```

Figure 8 Genus *Epilobium* Tree

The output in R's console is fairly large, so I won't place it in this document. The resultant tree is plotted below. It may help to have the data in front of you to compare.

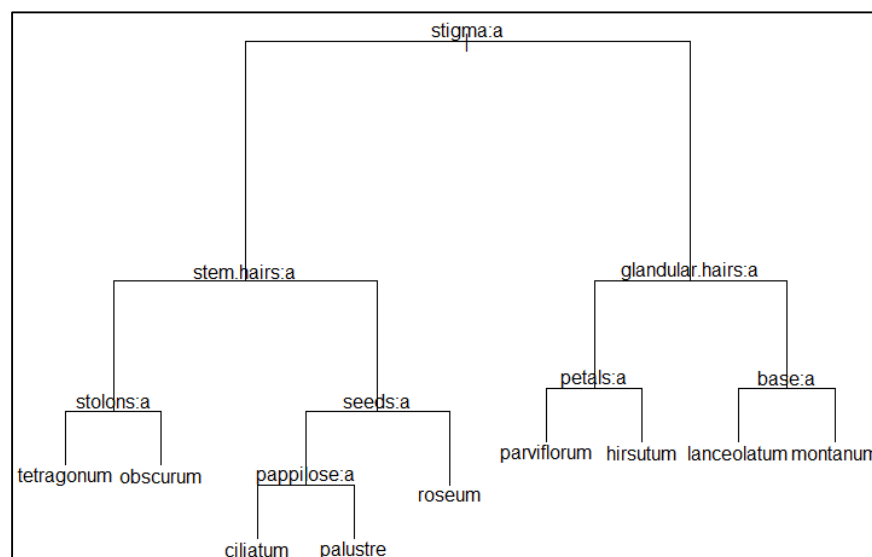


Figure 9 Tree Structure of *Epilobium* Data

Using the output in R's console helps with reading the plot. The first split is based on stigma; a clavate stigma results in the left-hand side (node 2) while a lobed stigma results in the right-hand side (node 3). Follow the right side of stigma. The next split is based on glandular hairs: left split has a value of absent (node 6) while the right split has values of present (node 7). Following the right side of the split, the next criteria is based on base. The left-split has a base of cuneate while the right-split has a base of rounded.