STAT 551

Regression Tree Assignment 1

Yuchi Hu

Contents

[**1.** **Introduction** 2](#_Toc1854672)

[**2.** **Exploratory Analysis** 2](#_Toc1854673)

[**2.1** **Numerical Summary** 2](#_Toc1854674)

[**2.2** **Histograms** 3](#_Toc1854675)

[**2.3** **Boxplots** 4](#_Toc1854676)

[**3.** **Classification Trees** 5](#_Toc1854677)

[**3.1** **Creating a Classification Tree Using *rpart*** 5](#_Toc1854678)

[**3.2** **Creating a Classification Tree Using *party*** 6](#_Toc1854679)

[**3.3** **Creating a Classification Tree Using *tree*** 7](#_Toc1854680)

[**3.4** **Comparison of Tree Models** 8](#_Toc1854681)

[**4.** **Conclusion** 8](#_Toc1854682)

**1. Introduction**

In this assignment, we analyze the iris dataset using three different tree libraries: *rpart*, *party*, and *tree*. The dataset contains 50 samples from each of three species of *Iris* (*setosa*, *versicolor*, and *virginica*). Each sample was measured in terms of four characteristics: petal length, petal width, sepal length, and sepal width (all in centimeters). The response variable is species, and the predictor variables are the four measurements. Since the response is categorical, our task is to construct a classification tree that can be used to predict the species of an *Iris* flower based on its petal length, petal width, sepal length, and sepal width.

We start our analysis with a numerical summary and descriptive plots. Then we move on to creating a classification tree with each tree library (*rpart*, *party*, and *tree*). And finally, we compare the three tree models.

**2. Exploratory Analysis**

**2.1 Numerical Summary**

We begin our analysis with a numerical summary of the iris data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Sepal.Length** | **Sepal.Width** | **Petal.Length** | **Petal.Width** |
| mean | 5.84 | 3.06 | 3.76 | 1.2 |
| sd | 0.83 | 0.44 | 1.77 | 0.76 |
| median | 5.8 | 3 | 4.35 | 1.3 |
| min | 4.3 | 2 | 1 | 0.1 |
| max | 7.9 | 4.4 | 6.9 | 2.5 |

**Table 1.** *Summary statistics (in centimeters) of the predictors (sepal length/width and petal length/width)*.

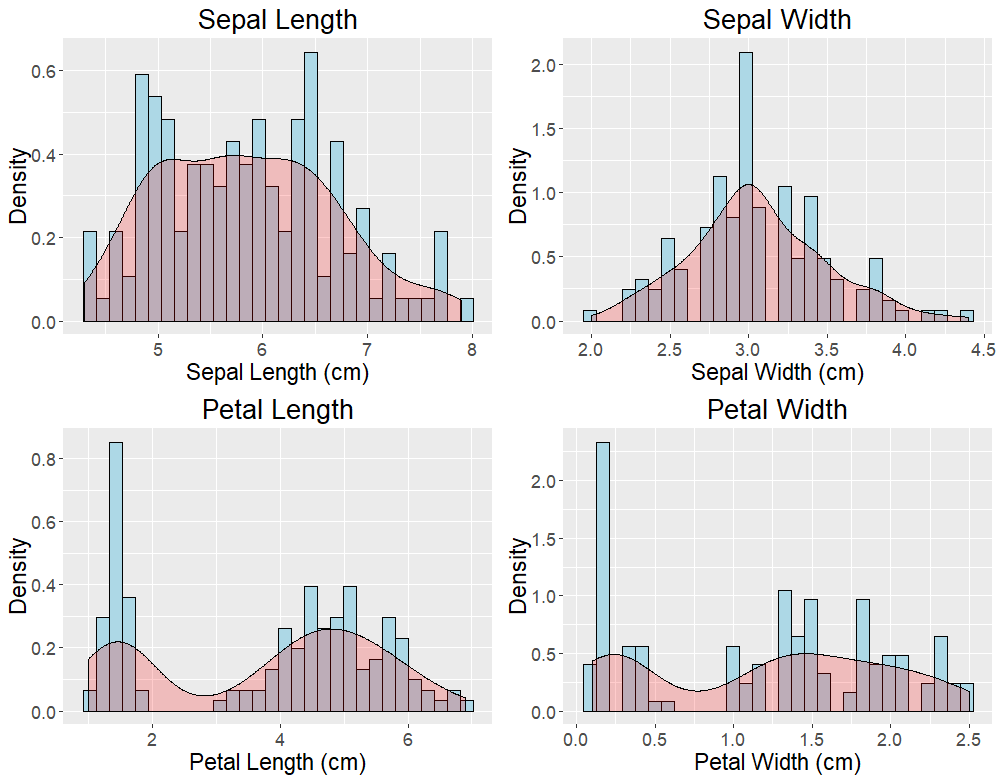
|  |  |
| --- | --- |
| **Species** | **Frequency** |
| setosa | 50 |
| versicolor | 50 |
| virginica | 50 |

**Table 2.** *Frequency distribution of the response (species)*.

From Table 1, we can see that sepal length is generally largest in size, whereas petal width is generally smallest in size. From Table 2, we can see that each species has 50 samples, so the classes of the response are balanced.

**2.2 Histograms**

We continue our analysis with histograms of each predictor, overlaid with density plots:

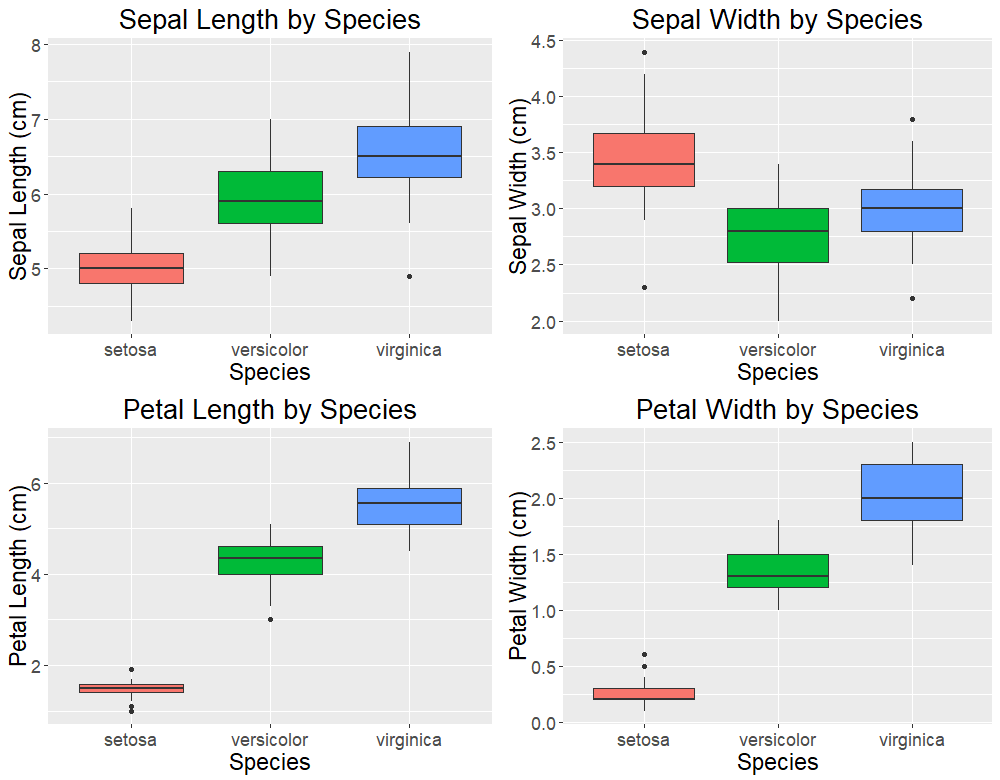


**FIGURE 1.** *Histograms of each predictor, overlaid with density plots*.

From Figure 1, we can see that the distributions of sepal length and sepal width are both approximately normal, whereas the distributions of petal length and petal width appear to be bimodal.

**2.3 Boxplots**

Next, we look at boxplots of each predictor by species:



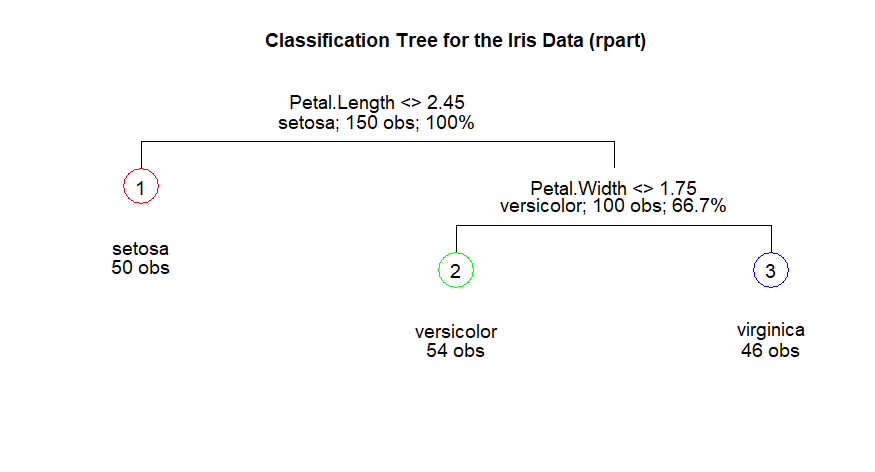
**FIGURE 2.** *Boxplots of each predictor by species*.

From Figure 2, we can see that the three species are generally distinguishable from each other by any of the four predictors. In fact, *Iris setosa* and *Iris virginica* tend to be the smallest and largest, respectively, in terms of all predictors except sepal width.

**3. Classification Trees**

**3.1 Creating a Classification Tree Using *rpart***

We construct a classification tree of the irisdata using the *rpart* library. We graph the tree using the draw.tree function from the *maptree* library:



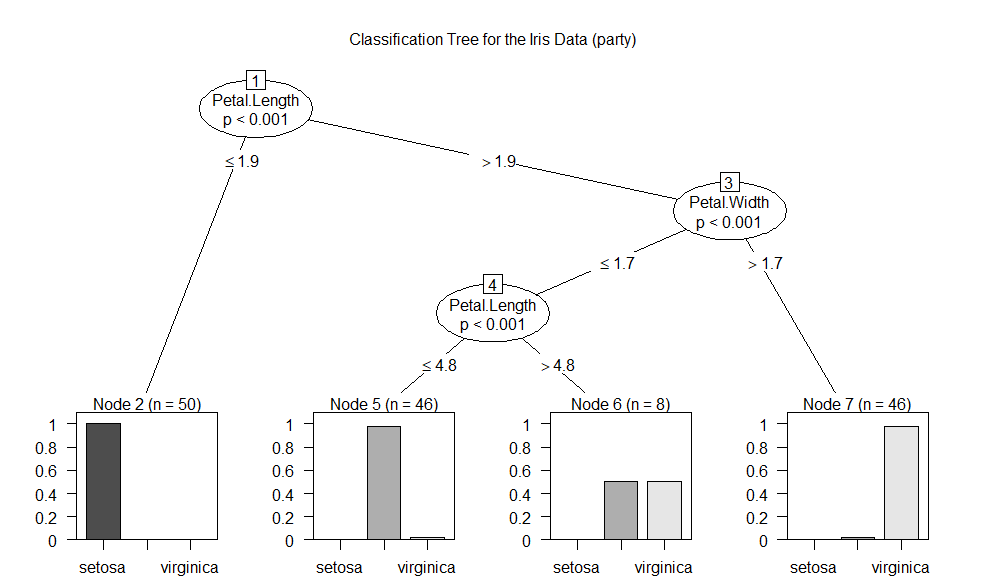
**FIGURE 3.** *Classification tree for the Iris data, created using rpart.*

From Figure 3, we can see that there are two splits, or three terminal nodes, in the classification tree. The top split assigns flowers with petal length < 2.45 cm to the left branch; these flowers are categorized as *setosa*. Flowers with petal length ≥ 2.45 cm are assigned to the right branch and then further split based on petal width. At the bottom split, flowers with petal width < 1.75 cm are categorized as *versicolor*; otherwise, they are categorized as *virginica*. After examining the cross-validated error results, we find that pruning the tree is not necessary since the lowest cross-validation error occurs at two splits or three terminal nodes.

The misclassification error rate is 6/150 = 0.04.

**3.2 Creating a Classification Tree Using *party***

Next, we construct a classification tree using the *party* library. We graph the tree using the plot function:



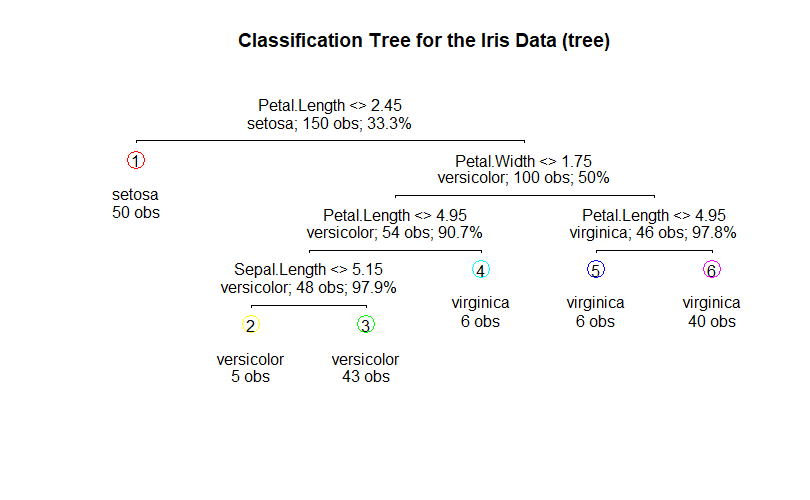
**FIGURE 4.** *Classification tree for the Iris data, created using party.*

From Figure 4, we can see that there are three splits, or four terminal nodes, in the classification tree. In comparison, the *rpart* tree only had two splits. The extra split occurs on petal length at a value of 4.8 cm. The first split still occurs on petal length; however, the value of the split is 1.9 cm compared to 2.45 cm from the *rpart* tree. The tree is grown using conditional inference procedures, or permutation tests, that do not require additional pruning or cross-validation.

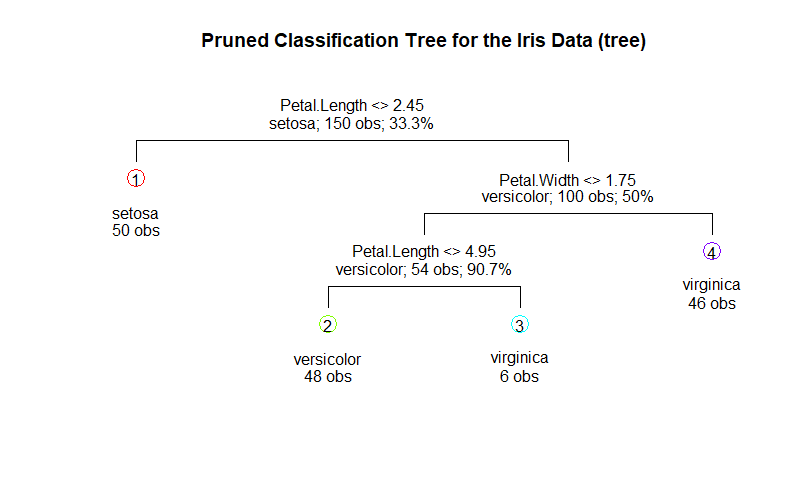
The misclassification error rate is 6/150 = 0.04.

**3.3 Creating a Classification Tree Using *tree***

Finally, we construct a classification tree using the *tree* library. We graph the tree using the draw.tree function from the *maptree* library:



**FIGURE 5.** *Classification tree for the Iris data, created using tree.*



**FIGURE 6.** *Pruned classification tree for the Iris data, created using tree.*

From Figure 5, we can see that there are five splits, or six terminal nodes, in the classification tree, which is greater than the number of splits or terminal nodes in the other two trees. After performing cross-validation, we find that the lowest CV error occurs at three splits or four terminal nodes; thus, pruning the tree improves the results. The pruned tree is shown in Figure 6. We can see that this tree is essentially the same as the *rpart* tree except that flowers with petal width < 1.75 cm are further split on petal length at a value of 4.95 cm.

The misclassification error rate is 4/150 = 0.02667.

The *tree* library differs from the *rpart* library in that they set different default values for the minimum number of observations that must exist in any terminal node. However, both *tree* and *rpart* use information measures (Gini index) to grow the trees.

The *tree* library differs from the *party* library in that *party* uses recursive partitioning but in a conditional inference framework: the splits are based on permutation tests.

**3.4 Comparison of Tree Models**

The *rpart* tree has three terminal nodes and does not require pruning since examination of the cross-validated error results show that the initial tree already had the lowest CV error. The *party* tree has four terminal nodes (one more than the rpart tree) and also does not require pruning since it is grown using a statistical approach that does not require additional pruning or cross-validation. On the other hand, the initial six-node *tree* tree did require pruning since performing cross-validation shows that the lowest CV error occurs with four terminal nodes. All three trees’ first two splits are on petal length and petal width. The initial *tree* tree also has a final split on sepal length. Sepal width is the only variable that was not split in any of the trees.

**4. Conclusion**

In summary, the *rpart* and *tree* libraries utilize similar algorithms to grow their trees, which only differ in the number of terminal nodes. They can be made identical by tuning various control parameters. On the other hand, the *party* library can grow a tree quite different from those of the other two libraries due to a conditional inference tree-growing procedure. I would recommend the *party* library since additional pruning or cross-validation is not necessary, whereas in the other two libraries, additional cross-validation is needed to check whether the tree requires pruning.