Lab 6: Classification Trees

Sarah Wright & Tyusha

Introduction

In this lab, we will again be interested in classifying observations in the `iris` data set as belonging to one of three species.

We will make our classification using *classification trees* which are a type of *decision tree*. A decision tree works by using *splitting rules* to divide up the predictor space. For numerical predictors, these rules take the form of *linear separators*. Linear separators are just linear inequalities of the predictors. For example, consider the problem of trying to predict whether a student will pass their first college mathematics course. The scatterplot called `passfailex.png` in Moodle shows the mathematics ACT score and GPA for 57 students. Each student dot is colored by a black dot (0) if they failed their first college mathematics course and a red dot (1) if they passed.

The node at the top of the tree represents the linear separator $ACT < 18.65$. If a student in the training data has ACT score less than 18.65, then they are sent to the left side of the tree. If a student has ACT score greater than 18.65, they are sent to the right side of the tree. Students on the right side of the tree are then further divided by the linear separator $GPA < 2.86111$. Thus, every student is placed at one of the three *leaves* of the tree. The leaf labels represent the majority class at each leaf. For example, more than half of the students with $ACT < 18.65$ failed the class, so the leaf at the far left of the tree is labeled by 0. To classify a new student, we apply the linear separators starting at the top of the tree to determine which leaf they belong to. The new student is then classified according to the label of that leaf.

Classification Tree Lab

1) The `iris` data set is contained in the `datasets` package in R. Import the data set and just as in the KNN lab, create `train` consisting of 60% of the data and `test` consisting of the remaining 40% of the data.

Again, be sure that each species is represented proportionally in the test and train sets.

```r
data("iris")

#iris <- select(iris, Sepal.Width, Petal.Width, Species)

rows = 150
intTrain = 0.60 * rows #60% of the data without replacement
train <- sample_n(iris,intTrain, replace = FALSE)
table(train$Species)
```

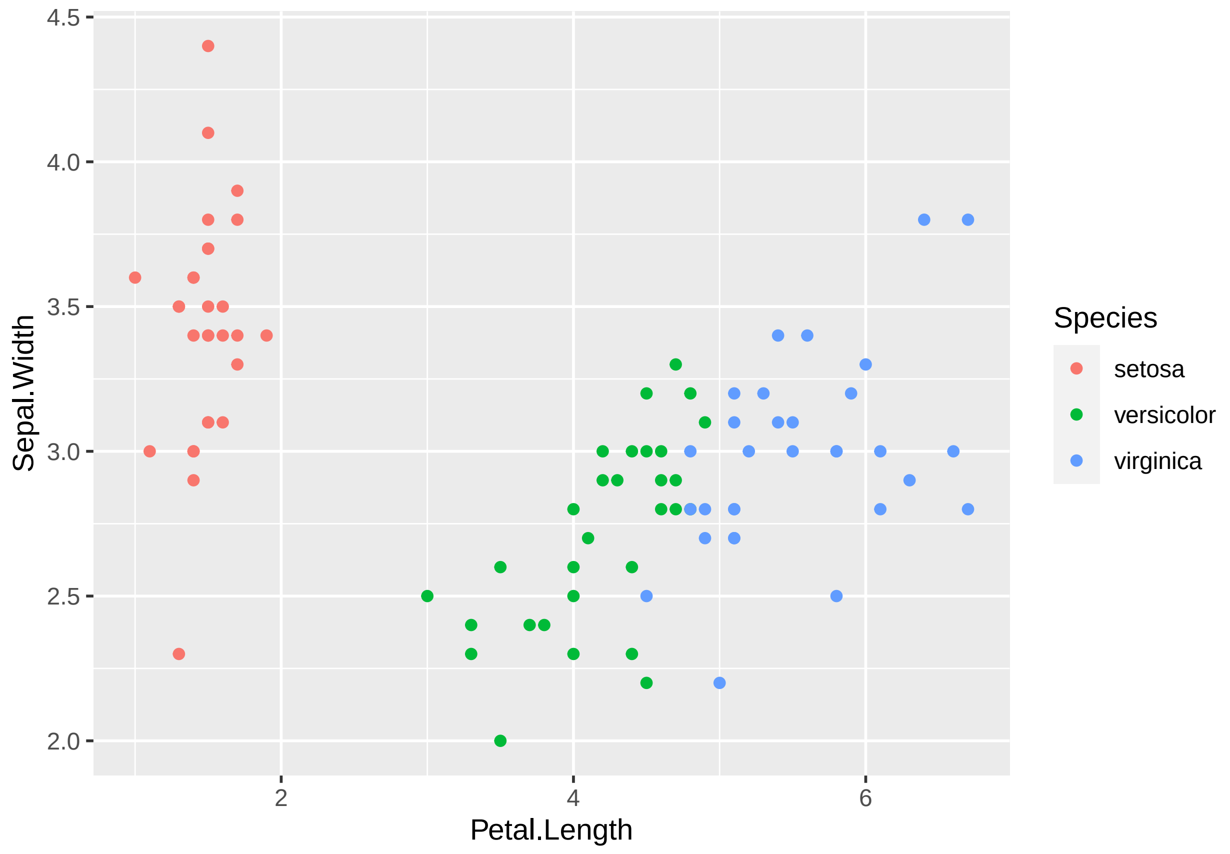
```
##
##     setosa versicolor  virginica
##         29         30         31
```

```
intTest = 0.40 * rows #40% of the data without replacement
test <- sample_n(iris, intTest, replace = FALSE)
table(test$Species)
```

```
##
##     setosa versicolor  virginica
##         18         23         19
```

2) Create a scatterplot of `Petal.Length` versus `Sepal.Width` colored by `Species` using the data in `train`. Notice that these are different traits than we used in the KNN lab. Use the `ggplot2` package to create this scatterplot.

```
ggplot(data = train, mapping = aes(x = Petal.Length, y = Sepal.Width)) + geom_point(aes(color = Species)
```



3) Based on your scatterplot above, determine some simple linear separators that can be used to classify the observations in `test`. Write these out as sentences using a comment. You should be able to label the observations in test with one of the three species.

```
# All species that have a length >2 are setosa
# All species that have a length of <=5 are virfinica
# Species that are neither are more than likely versicolor
```

4) Create a vector that consists of predictions based on your separators. That is, if you believe the species `versicolor` takes on `Sepal.Width` values between 2 and 3.4, you can write the following code:

predictions[test$Sepal.Width > 2 & test$Sepal.Length < 3.4] <- "versicolor", where you will have to first initialize a predictions vector that will contain your predictions on the test set. (Remember, you can use the logical operators <, >, & (and), | (or) to subset vectors in R.)

```
predictions <- factor(ifelse(test$Petal.Length <= 2.5, "setosa", ifelse(test$Petal.Length > 5 & test$Sep
#predictions
```

5) To get a sense of how good your simple model predictions are, use the table function in R to create a confusion matrix. As a reminder, the arguments in table should be the vector of your predictions and the vector of the actual categories to which the test observations belong.

```
predictedCorrectness <- table(predictions, test$Species)
```

Hopefully, your simple linear separators were fairly successful at making predictions. Again, this is because the iris data set just isn't that challenging. For more challenging data sets, we will want R to search for the best linear separators and to build us a *classification tree*. To implement this technique here, we will use the tree package with the syntax shown in the R chunk below.

6) Modify the code below to train and *display* a classification tree that predicts Species as a function of the Sepal.Width and Petal.Length. To display the classification tree, you need to use the plot() function. Make sure to add text to this tree that shows the separators (HINT: Check the help file).

```
library("tree")
```

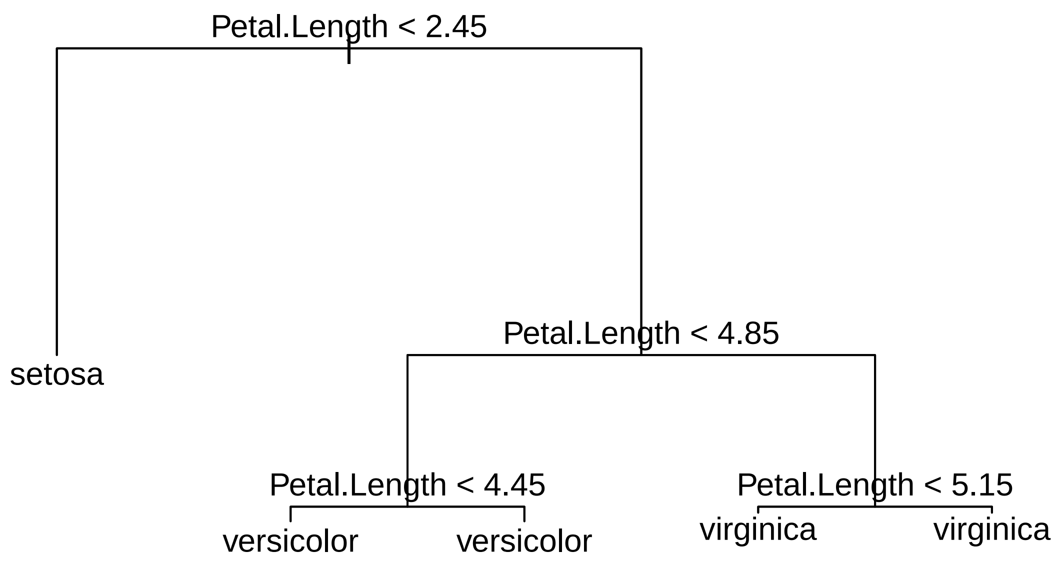
```
## Warning: package 'tree' was built under R version 4.0.2
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
#library ("ISLR2")

#class_tree <- tree(your_formula, your_data)

class_tree <- tree(Species~ Sepal.Width + Petal.Length, train)
plot(class_tree)
text(class_tree)
```



7) Use this classification tree model and `predict` (with `type = "class"`) to classify the species in the test set. Print the confusion matrix and the accuracy.

```
TreePrediction <- predict(class_tree, newdata = test, type = "class")

table(TreePrediction, test$Species)
```

```
##
## TreePrediction setosa versicolor virginica
##     setosa         18          0         0
##     versicolor      0         20         0
##     virginica       0          3        19
```

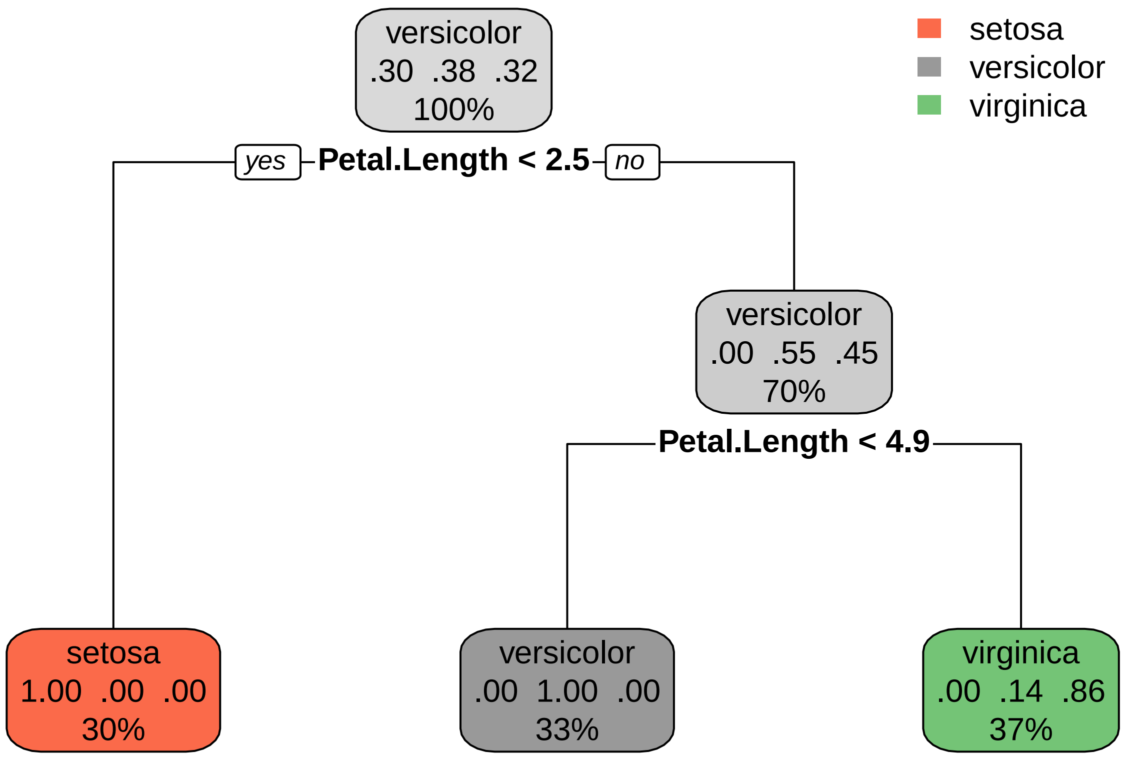
```
# All but one, 59/60 predictions are made correctly with this model
```

We use the package `tree` because that is what your book uses. However, there are other packages that build decision trees so beautiful you will want to cry. One of these is in the package `rpart`.

8) Replace *your.formula* and *your.dataset* to perform the same classification you did in Problem 6 and plot the resulting tree. The function `rpart()` also has extra parameters that you can tune (such as *minsplit* and *minbucket*) that will change how your tree is built (you can read about these in the documentation if you are curious.)

```
#install.packages("rpart.plot")

fit <- rpart(Species ~ Sepal.Width + Petal.Length, data = test, method = "class")
rpart.plot(fit)
```



Wow. That is a beautiful decision tree.