Exercises 6

Brady Lamson

2022-03-28

11: Supervised Learning

11.1 Classifiers

Exercise 1

Based on the decision tree I would make the following predictions:

- A 9-inch, 10-pound pet would likely be a **cat** and be pretty confident.
- A 14-inch, 21-pound pet would likely be a **dog** and be very confident.

Exercise 2

```r
type <- c("dog", "dog", "cat", "dog", "cat", "dog", "cat", "dog", "cat",
          "dog", "cat", "dog", "dog", "cat", "dog", "cat", "dog", "cat", "dog")

wt <- c(8, 17, 8, 18, 7, 22, 6, 16, 7, 20, 10, 15, 14, 11, 13, 13, 15, 17, 10)
ht <- c(7.5, 10, 8, 15, 7, 15, 7, 13, 11, 16, 7, 10.5, 9, 9.5, 9, 8, 9, 8, 12)

pets <- data.frame(Type = type, Ht = ht, Wt = wt)

my.tree <- rpart(
    Type ~ Wt + Ht, data = pets,
    control = rpart.control(minsplit = 7)
)

newPets <- data.frame(Ht = c(9, 14), Wt = c(10, 21))

predict(my.tree, newdata = newPets, type = "class")
```

```
##   1   2
## cat dog
## Levels: cat dog
```

First pet is predicted to be a cat and the second pet is predicted to be a dog. This prediction makes the same predictions I made in exercise 1.

Exercise 3

```
my.tree <- rpart(Species ~ Petal.Length + Petal.Width, data = iris)
my.tree
```

```
## n= 150
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
##   2) Petal.Length< 2.45 50   0 setosa (1.00000000 0.00000000 0.00000000) *
##   3) Petal.Length>=2.45 100  50 versicolor (0.00000000 0.50000000 0.50000000)
##     6) Petal.Width< 1.75 54   5 versicolor (0.00000000 0.90740741 0.09259259) *
##     7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173913 0.97826087) *
```

a) This tree has 3 different terminal nodes.

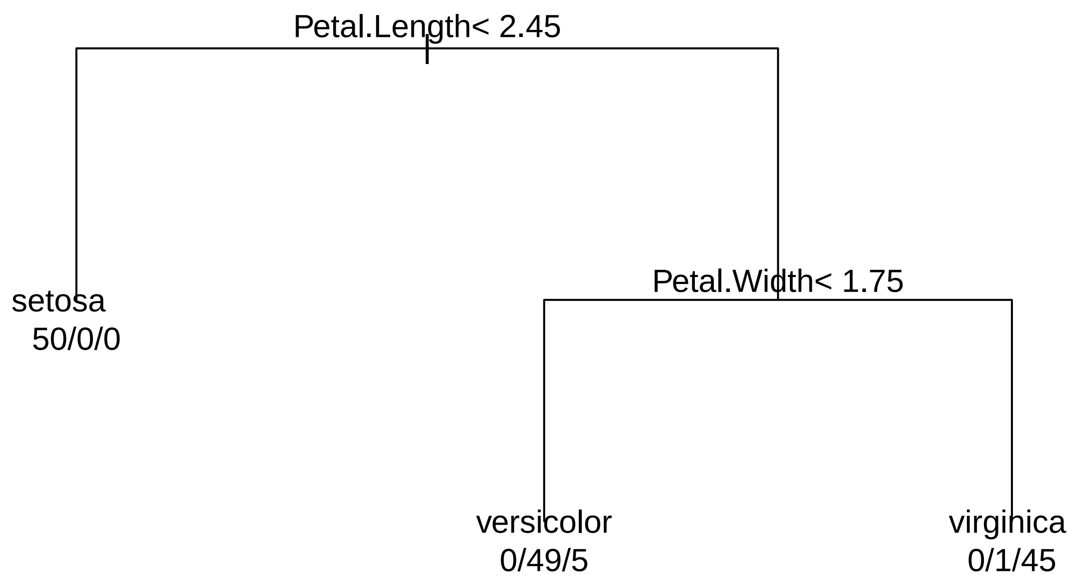
```
predSpecies <- predict(my.tree, type = "class")
species <- data.frame(Actual = iris$Species, Predicted = predSpecies)

confusion <- table(species)
confusion
```

```
##              Predicted
## Actual        setosa versicolor virginica
##   setosa          50          0         0
##   versicolor       0         49         1
##   virginica        0          5        45
```

b) The trees correct classification rate is **96%** and the mis-classification rate is, by extension, **4%**.

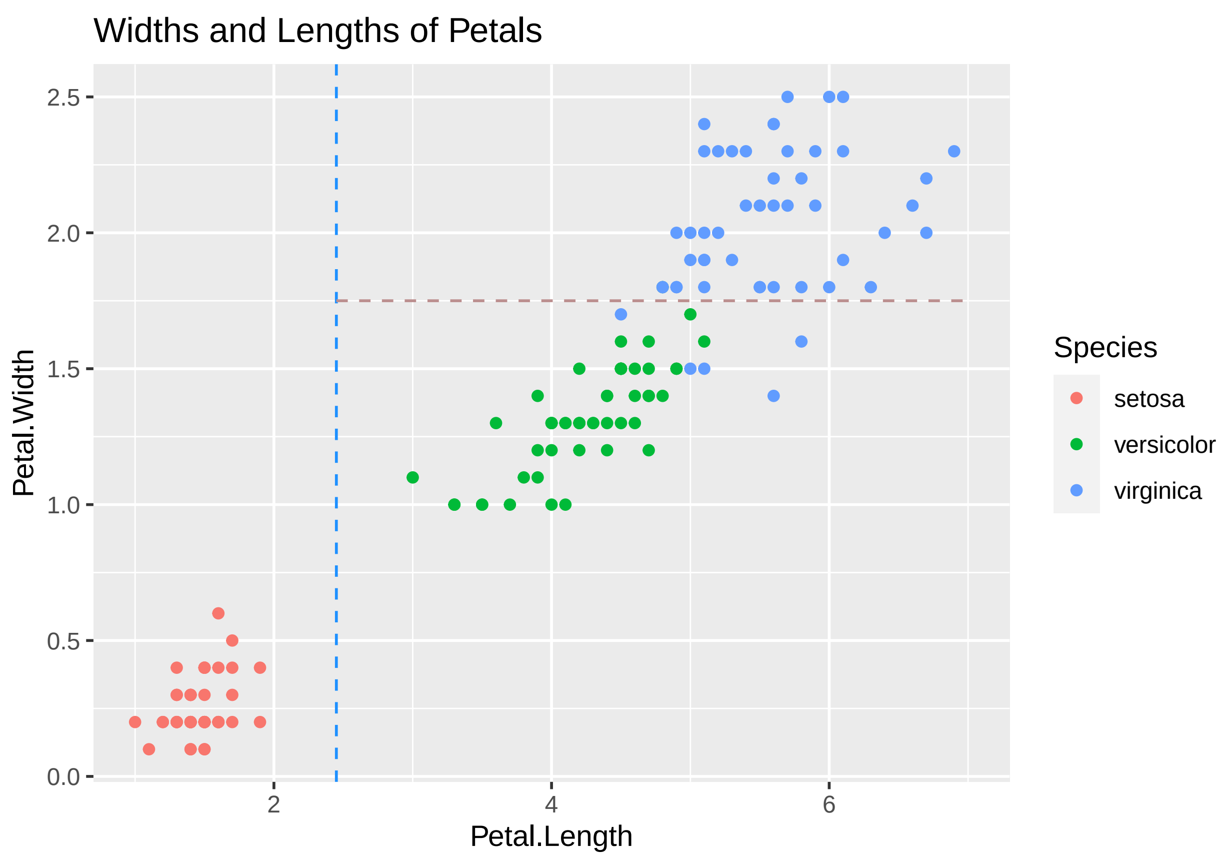
```
## First plot:
par(xpd = TRUE)
plot(my.tree, compress = TRUE)
text(my.tree, use.n = TRUE)
```



```r
par(xpd = FALSE)
```

```r
## Second plot:
splitPetal.Length <- 2.45
splitPetal.Width <- 1.75
splitLines <- data.frame(x1 = splitPetal.Length, x2 = 7,
                         y1 = splitPetal.Width, y2 = splitPetal.Width)

g <- ggplot(data = iris,
            mapping = aes(x = Petal.Length, y = Petal.Width,
                          color = Species)) +
    geom_point() +
    labs(title = "Widths and Lengths of Petals") +
    geom_vline(xintercept = splitPetal.Length,
               color = "dodgerblue",
               linetype = 2) +
    geom_segment(data = splitLines,
                 mapping = aes(x = x1, y = y1, xend = x2, yend = y2),
                 color = "rosybrown", linetype = 2)
g
```



c) Using the above plots I make the following predictions.

- A flower with `Petal.Length` of 3.0cm and `Petal.Width` of 1.5cm would be **veriscolor**.

A flower with `Petal.Length` of 4.0cm and `Petal.Width` of 2.1cm would be **virginica**. There aren't any virginica flowers with that length, but the cutoffs on the plot show that's the best prediction to make.

```
newIris <-
    data.frame(Petal.Length = c(3.0, 4.0),
               Petal.Width = c(1.5, 2.1)
    )
```

```
predict(my.tree, newdata = newIris, type = "class")
```

```
##          1          2
## versicolor  virginica
## Levels: setosa versicolor virginica
```

The predictions made by the function match that of the predictions I made in part c.

Exercise 4

```
my.tree <- rpart(Species ~ Sepal.Length + Sepal.Width, data = iris)
my.tree
```

```
## n= 150
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
##    2) Sepal.Length< 5.45 52   7 setosa (0.86538462 0.11538462 0.01923077)
##      4) Sepal.Width>=2.8 45   1 setosa (0.97777778 0.02222222 0.00000000) *
##      5) Sepal.Width< 2.8 7   2 versicolor (0.14285714 0.71428571 0.14285714) *
##    3) Sepal.Length>=5.45 98  49 virginica (0.05102041 0.44897959 0.50000000)
##      6) Sepal.Length< 6.15 43  15 versicolor (0.11627907 0.65116279 0.23255814)
##       12) Sepal.Width>=3.1 7   2 setosa (0.71428571 0.28571429 0.00000000) *
##       13) Sepal.Width< 3.1 36  10 versicolor (0.00000000 0.72222222 0.27777778) *
##      7) Sepal.Length>=6.15 55  16 virginica (0.00000000 0.29090909 0.70909091) *
```

a) This tree has **5** terminal nodes

```
predSpecies <- predict(my.tree, type = "class")
species <- data.frame(Actual = iris$Species, Predicted = predSpecies)
confusion <- table(species)
confusion
```

```
##             Predicted
## Actual        setosa versicolor virginica
##   setosa          49          1         0
##   versicolor       3         31        16
##   virginica        0         11        39
```

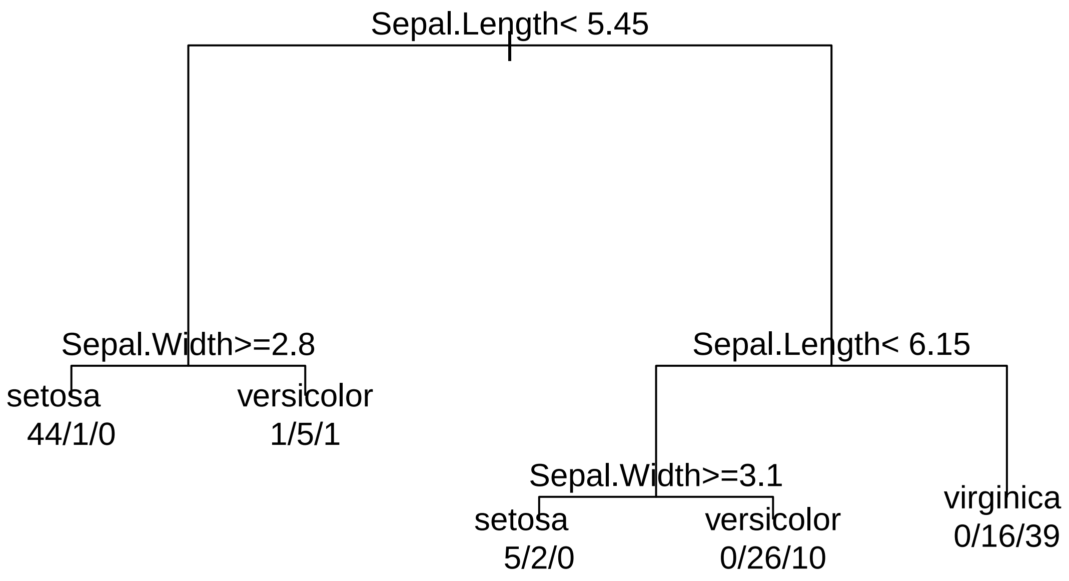
```
# B

classification_rate <- sum(diag(confusion) / nrow(species))

glue::glue("
    The correct classification rate is approximately {(classification_rate * 100) %>% round(3)}%,
    The misclassification rate is approximately {((1 - classification_rate) * 100) %>% round(3)}%
")
```

```
## The correct classification rate is approximately 79.333%,
## The misclassification rate is approximately 20.667%
```

```
## First plot:
par(xpd = TRUE)
plot(my.tree, compress = TRUE)
text(my.tree, use.n = TRUE)
```

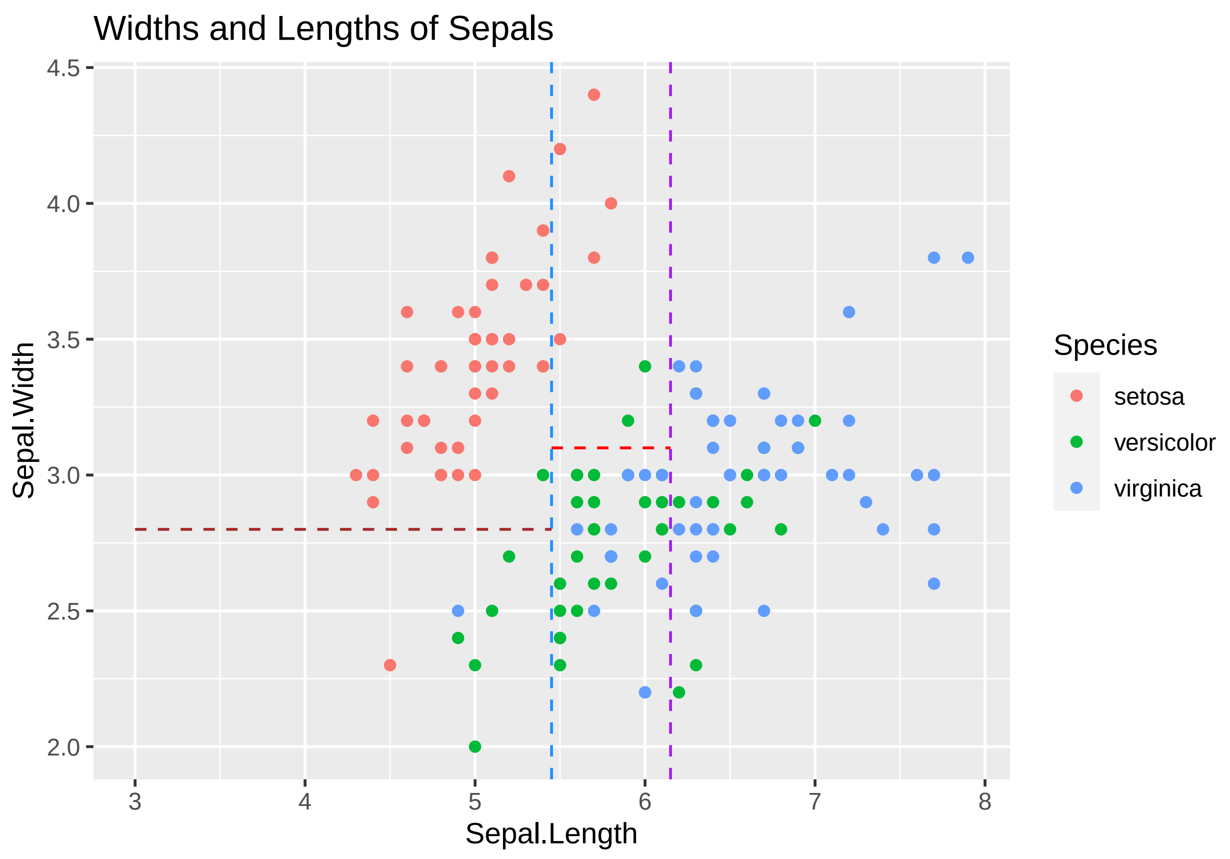


```r
par(xpd = FALSE)
```

```r
## Second plot:
split1Sepal.Length <- 5.45
split1Sepal.Width <- 2.8
splitLines1 <- data.frame(
    x1 = 3, x2 = split1Sepal.Length,
    y1 = split1Sepal.Width, y2 = split1Sepal.Width
)

split2Sepal.Length <- 6.15
split2Sepal.Width <- 3.1
splitLines2 <- data.frame(
    x1 = split1Sepal.Length, x2 = split2Sepal.Length,
    y1 = split2Sepal.Width, y2 = split2Sepal.Width
)

g <- ggplot(data = iris,
            mapping = aes(x = Sepal.Length, y = Sepal.Width,
                          color = Species)) +
    geom_point() +
    labs(title = "Widths and Lengths of Sepals") +
    geom_vline(xintercept = split1Sepal.Length,
               color = "dodgerblue",
               linetype = 2) +
    geom_vline(xintercept = split2Sepal.Length,
               color = "purple",
               linetype = 2) +
    geom_segment(data = splitLines1,
                 mapping = aes(x = x1, y = y1, xend = x2, yend = y2),
                 color = "brown", linetype = 2) +
    geom_segment(data = splitLines2,
                 mapping = aes(x = x1, y = y1, xend = x2, yend = y2),
                 color = "red", linetype = 2)
g
```



c) Using the above plots I make the following predictions.

- A flower with `Petal.Length` of 6.0cm and `Petal.Width` of 3.5cm would be **setosa**. I used the decision tree over the scatterplot for this decision.

A flower with `Petal.Length` of 7.0cm and `Petal.Width` of 3.0cm would be **virginica**.

```
newIris <- data.frame(Sepal.Length = c(6.0, 7.0),
Sepal.Width = c(3.5, 3.0))

predict(my.tree, newdata = newIris, type = "class")
```

```
##          1          2
##     setosa virginica
## Levels: setosa versicolor virginica
```

The same decisions as part c were made, the prediction simply appears to match the decision tree cases.

Exercise 5

```
y1 <- c("A", "B", "C", "C", "C", "C", "C", "C", "C", "C", "C", "C")
round(prop.table(table(y1)), digits = 1)
```

```
## y1
##   A   B   C
## 0.1 0.1 0.8
```

```
y2 <- c("A", "B", "C", "C", "A", "C", "B", "A", "A", "B", "C", "B")
round(prop.table(table(y2)), digits = 1)
```

```
## y2
##   A   B   C
## 0.3 0.3 0.3
```

a)

- Based just on the fact that more homogeneous data scores higher on the Gini index, I would guess `y1` would be considered more *pure* and thus have a lower score. There are far more C's than anything else. `y2` is too evenly split to be considered homogeneous and as such would be further from 0 than `y1`.

```
1 - sum(
    c(
        .1^2, .1^2, .8^2
    )
)
```

```
## [1] 0.34
```

.34 is the gini index for `y1` assuming I did the calculation correctly.

```
1 - sum(
    c(
        .3^2, .3^2, .3^2
    )
)
```

```
## [1] 0.73
```

.73 is the gini index for `y2` assuming I did the calculation correctly. This would, at the very least, match my earlier assumptions that this would be higher than the gini index for `y1`.

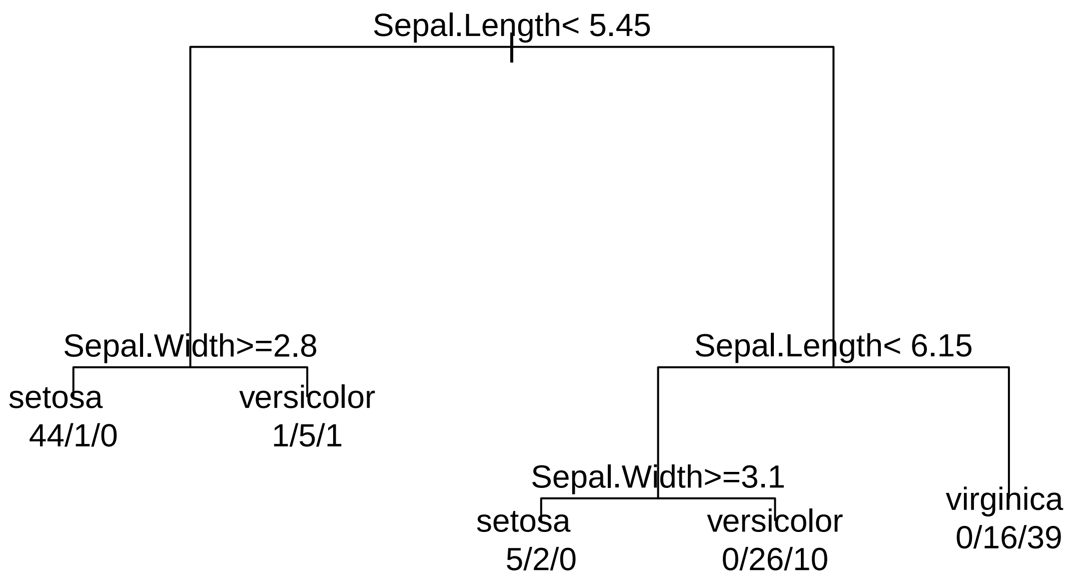
b) The gini index for a vector with all of the same values in them is 0, because the vector is completely pure. $1 - 1^2 = 0$.

Exercise 6

```r
my.tree <- rpart(Species ~ Sepal.Length + Sepal.Width, data = iris,
                 control = rpart.control(cp = 0.002))
my.tree
```

```
## n= 150
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
##    2) Sepal.Length< 5.45 52   7 setosa (0.86538462 0.11538462 0.01923077)
##      4) Sepal.Width>=2.8 45   1 setosa (0.97777778 0.02222222 0.00000000) *
##      5) Sepal.Width< 2.8 7   2 versicolor (0.14285714 0.71428571 0.14285714) *
##    3) Sepal.Length>=5.45 98  49 virginica (0.05102041 0.44897959 0.50000000)
##      6) Sepal.Length< 6.15 43  15 versicolor (0.11627907 0.65116279 0.23255814)
##       12) Sepal.Width>=3.1 7   2 setosa (0.71428571 0.28571429 0.00000000) *
##       13) Sepal.Width< 3.1 36  10 versicolor (0.00000000 0.72222222 0.27777778) *
##      7) Sepal.Length>=6.15 55  16 virginica (0.00000000 0.29090909 0.70909091) *
```

```r
par(xpd = TRUE)
plot(my.tree, compress = TRUE)
text(my.tree, use.n = TRUE)
```

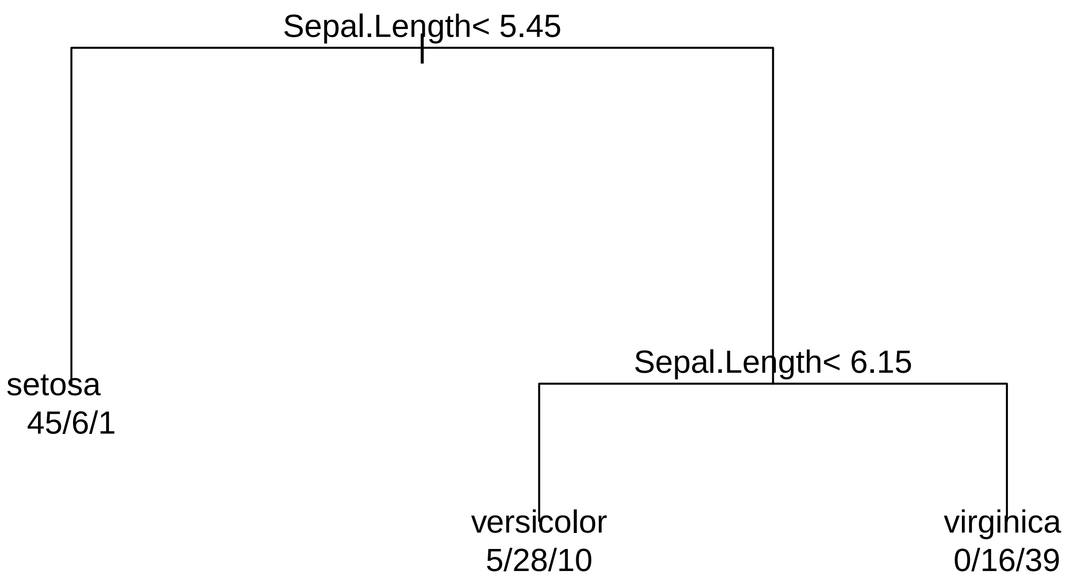


```
par(xpd = FALSE)
```

```
my.tree <- rpart(Species ~ Sepal.Length + Sepal.Width, data = iris,
                 control = rpart.control(cp = 0.05))
my.tree
```

```
## n= 150
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
##   2) Sepal.Length< 5.45 52   7 setosa (0.86538462 0.11538462 0.01923077) *
##   3) Sepal.Length>=5.45 98  49 virginica (0.05102041 0.44897959 0.50000000)
##     6) Sepal.Length< 6.15 43  15 versicolor (0.11627907 0.65116279 0.23255814) *
##     7) Sepal.Length>=6.15 55  16 virginica (0.00000000 0.29090909 0.70909091) *
```

```
par(xpd = TRUE)
plot(my.tree, compress = TRUE)
text(my.tree, use.n = TRUE)
```



```
par(xpd = FALSE)
```

0.2% Resulted in far more terminal nodes.

Exercise 7

```r
my.forest <- randomForest(Species ~ Sepal.Length + Sepal.Width +
                              Petal.Length + Petal.Width,
                          data = iris,
                          ntree = 500,
                          mtry = 1)

sum(diag(my.forest$confusion)) / nrow(iris)
```

```
## [1] 0.9466667
```

a) An `mtry` of 1 gives a correct classification rate of approximately 94.67%.

```r
my.forest <- randomForest(Species ~ Sepal.Length + Sepal.Width +
                              Petal.Length + Petal.Width,
                          data = iris,
                          ntree = 500,
                          mtry = 3)

sum(diag(my.forest$confusion)) / nrow(iris)
```

```
## [1] 0.9533333
```

b) An `mtry` of 3 gives a correct classification rate of 96%.

Exercise 8

```
my.forest <- randomForest(Species ~ Sepal.Length + Sepal.Width +
                              Petal.Length + Petal.Width,
                          data = iris,
                          ntree = 500,
                          mtry = 4)

importance(my.forest)
```

```
##              MeanDecreaseGini
## Sepal.Length         1.180067
## Sepal.Width          1.296986
## Petal.Length        44.495274
## Petal.Width         52.378472
```

a) `Petal.Width` is the most important, `Sepal.Length` is the least important.

```
newIris <-
    data.frame(Petal.Length = c(3.0, 2.2, 2.7),
               Petal.Width = c(1.2, 2.1, 1.6),
               Sepal.Length = c(5.5, 5.1, 5.9),
               Sepal.Width = c(3.0, 2.7, 3.2))

predict(my.forest, newdata = newIris, type = "class")
```

```
##          1          2          3
## versicolor     setosa versicolor
## Levels: setosa versicolor virginica
```

b) The three predictions were **veriscolor**, **setosa**, **veriscolor**.

Exercise 9

```r
# A

train_iris <- select(iris, -c(Species, Sepal.Length, Sepal.Width))
ret_vec <- c()

for (ktry in 3:8) {

    # k nearest neighbor classification procedure:
    my.knn <- class::knn(train = train_iris,
                         test = train_iris,
                         cl = iris$Species,
                         k = ktry)

    # Save actual and predicted species:
    species <- data.frame(Actual = iris$Species, Predicted = my.knn)

    # Obtain correct classification rate:
    confusion <- table(species)

    ret_vec <- c(ret_vec, sum(diag(confusion)) / nrow(iris))
}

ret_vec
```

```
## [1] 0.9800000 0.9600000 0.9600000 0.9666667 0.9666667 0.9600000
```

```r
# B

# Data frame containing new flower for classification:
newIris <- data.frame(Petal.Length = 4.35,
                      Petal.Width = 1.65)
predict_vec <- c()

for (ktry in seq(from = 3, to = 21, by = 3)) {
    # k nearest neighbor classification procedure:
    my.knn <- class::knn(train = train_iris,
                         test = newIris,
                         cl = iris$Species,
                         k = ktry)

    predict_vec <- c(predict_vec, my.knn)
}
predict_vec
```

```
## [1] 2 2 2 2 2 2 2
```

The prediction does not change for any of the provided values of `ktry` I used.

Exercise 10

```
rock2 <-
    rock %>%
    dplyr::mutate(
        area = area / 10000,
        peri = peri / 10000,
        perm = log(perm)
    )

ret_vec <- c()

for (ktry in c(1,3,5,7)) {
    my.nn <-
        nnet::nnet(
            perm ~ area + peri + shape, data = rock2,
            size = ktry, linout = TRUE, maxit = 1000, trace = FALSE
        )

    rss <- sum((rock2$perm - predict(my.nn))^2)

    ret_vec <- c(ret_vec, rss)
}

ret_vec
```

```
## [1] 24.052120 10.310556  8.269845  3.806577
```

7 resulted in the smallest residual sum of squares.

Exercise 11

```
# Neural network classification procedure with k = 2 hidden units:
my.nn <-
    nnet::nnet(
        Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
        data = iris, size = 2, maxit = 200, trace = FALSE
    )

newIris <-
    data.frame(
        Petal.Length = c(3.5, 4.7, 1.3),
        Petal.Width = c(1.0, 1.5, 0.5),
        Sepal.Length = c(5.2, 6.2, 4.8),
        Sepal.Width = c(3.3, 3.6, 2.3)
    )

predict(my.nn, newdata = newIris, type = "class")
```

```
## [1] "setosa"     "versicolor" "setosa"
```

```
print("--------------------------------------")
```

```
## [1] "--------------------------------------"
```

```
predict(my.nn, newdata = newIris)
```

```
##        setosa   versicolor     virginica
## 1 1.0000e+00 9.232407e-78 1.428532e-218
## 2 4.9877e-64 1.000000e+00 6.771769e-106
## 3 1.0000e+00 9.232407e-78 1.428532e-218
```

Based on the given probabilities, we can see that **veriscolor** has the largest probabilities for the first 2 new flowers. **Setosa** is the highest for the final new flower.

Exercise 12

```r
Ln <- c(85.7, 64.5, 84.1, 82.5, 78.0, 81.3, 71.0, 86.7, 78.7)
Wt <- c(331.9, 121.5, 382.2, 287.3, 224.3, 245.2, 208.2, 393.4, 228.3)
snakes <- data.frame(Length = Ln, Weight = Wt)

newSnakes <-
    data.frame(
        Length = c(67, 72, 77, 81, 86),
        Weight = c(127.9, 153.7, 204.7, 300.6, 291.4)
    )

mod0 <- lm(Weight ~ 1, data = snakes)
mod1 <- lm(Weight ~ Length, data = snakes)
mod2 <- lm(Weight ~ poly(Length, 2, raw = TRUE), data = snakes)
mod3 <- lm(Weight ~ poly(Length, 3, raw = TRUE), data = snakes)
mod4 <- lm(Weight ~ poly(Length, 4, raw = TRUE), data = snakes)
mod5 <- lm(Weight ~ poly(Length, 5, raw = TRUE), data = snakes)
```

```r
# A

rmse0 <- sqrt(
    mean(
        (newSnakes$Weight - predict(mod0, newdata = newSnakes))^2
    )
)

rmse1 <- sqrt(
    mean(
        (newSnakes$Weight - predict(mod1, newdata = newSnakes))^2
    )
)

rmse2 <- sqrt(
    mean(
        (newSnakes$Weight - predict(mod2, newdata = newSnakes))^2
    )
)

rmse3 <- sqrt(
    mean(
        (newSnakes$Weight - predict(mod3, newdata = newSnakes))^2
    )
)

rmse4 <- sqrt(
    mean(
        (newSnakes$Weight - predict(mod4, newdata = newSnakes))^2
    )
)

rmse5 <- sqrt(
    mean(
        (newSnakes$Weight - predict(mod5, newdata = newSnakes))^2
    )
)

rmse0
```

```
## [1] 88.22161
```

```
rmse1
```

```
## [1] 34.56335
```

```
rmse2
```

```
## [1] 36.7128
```

```
rmse3
```

```
## [1] 49.4544
```

```
rmse4
```

```
## [1] 55.22323
```

```
rmse5
```

```
## [1] 53.77739
```

a) The best model according to RMSE is the 2nd. It has the lowest value.

```
mae0 <- mean(abs(newSnakes$Weight - predict(mod0, newdata = newSnakes)))
mae1 <- mean(abs(newSnakes$Weight - predict(mod1, newdata = newSnakes)))
mae2 <- mean(abs(newSnakes$Weight - predict(mod2, newdata = newSnakes)))
mae3 <- mean(abs(newSnakes$Weight - predict(mod3, newdata = newSnakes)))
mae4 <- mean(abs(newSnakes$Weight - predict(mod4, newdata = newSnakes)))
mae5 <- mean(abs(newSnakes$Weight - predict(mod5, newdata = newSnakes)))

mae0
```

```
## [1] 74.96889
```

```
mae1
```

```
## [1] 29.79193
```

```
mae2
```

```
## [1] 29.45119
```

```
mae3
```

```
## [1] 44.99129
```

```
mae4
```

```
## [1] 47.8705
```

```
mae5
```

```
## [1] 45.09594
```

The best model according to MAE is the 3rd model with the 2nd trailing close behind.