Class_Exercises_ClassNotes6

Tobias Boggess

3/14/2022

Section 11.1 Exercises

Exercise 1: Answer the following questions without using predict().

a) What type of animal (cat or dog) would you predict a 9-inch, 10-pound pet to be?
I would predict the type of an animal to have a weight of 10 pounds and have a height of 9 inches to be a cat.

b) What type of animal (cat or dog) would you predict a 14-inch, 21-pound pet to be?
I would predict the type of an animal to have a weight of 21 pounds and have a height of 14 inches to be a dog.

Exercise 2: Answer the following.

Data frame:

```
library(rpart)
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)
```

```r
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))
newPets
```

```
##   Ht Wt
## 1  9 10
## 2 14 21
```

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

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

a) What type of animal (cat or dog) would you predict the 9-inch, 10-pound pet to be?
This animal with 9 inch height and a 10 pound weight would be predicted as a cat.

b) What type would you predict the 14-inch, 21-pound pet to be?
This animal with 14 inch height and a 21 pound weight would be predicted as a dog.

c) Are your answers using predict() consistent with your answers to Exercise 1?
My answers are consistent with the evaluations in *Exercise 1*.

Exercise 3: Fit the following decision tree for classification (predicting Species) based on Petal.Length and Petal.Width:

Data frame:

```r
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) How many terminal nodes does the tree have?
There should be 3 terminal nodes.

b) What is the tree's correct classification rate (as a percent)? What is it's misclassification rate (as a percent)?
Code:

```
preds <- predict(my.tree, type = "class")
#This places a copy of the built-in iris data set in the Workspace:
iris <- mutate(iris, predSpecies = preds)
# The conf_mat() function automatically converts Species and predSpecies to factors:
conf_mat(data = iris, truth = Species, estimate = predSpecies)
```

```
##               Truth
## Prediction    setosa versicolor virginica
##   setosa          50          0         0
##   versicolor       0         49         5
##   virginica        0          1        45
```

Correct Classification Rate:

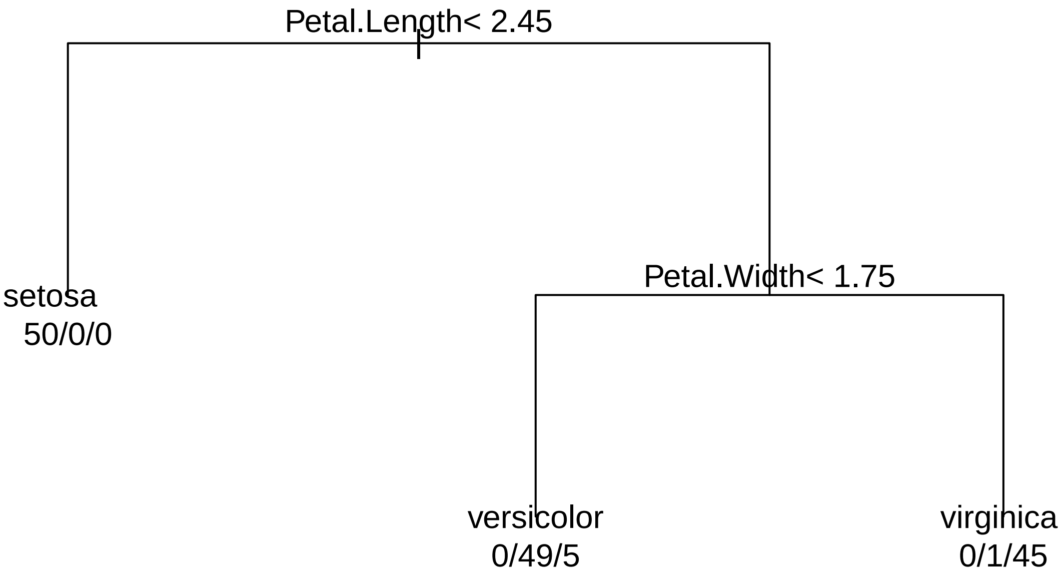
```
((50+49+45) / (50+49+5+1+45)) * 100
```

```
## [1] 96
```

There is a correct classification rate of 96%. The misclassification rate is 4%.

c) Classify the following based on the following flowers using the plots.
Code:

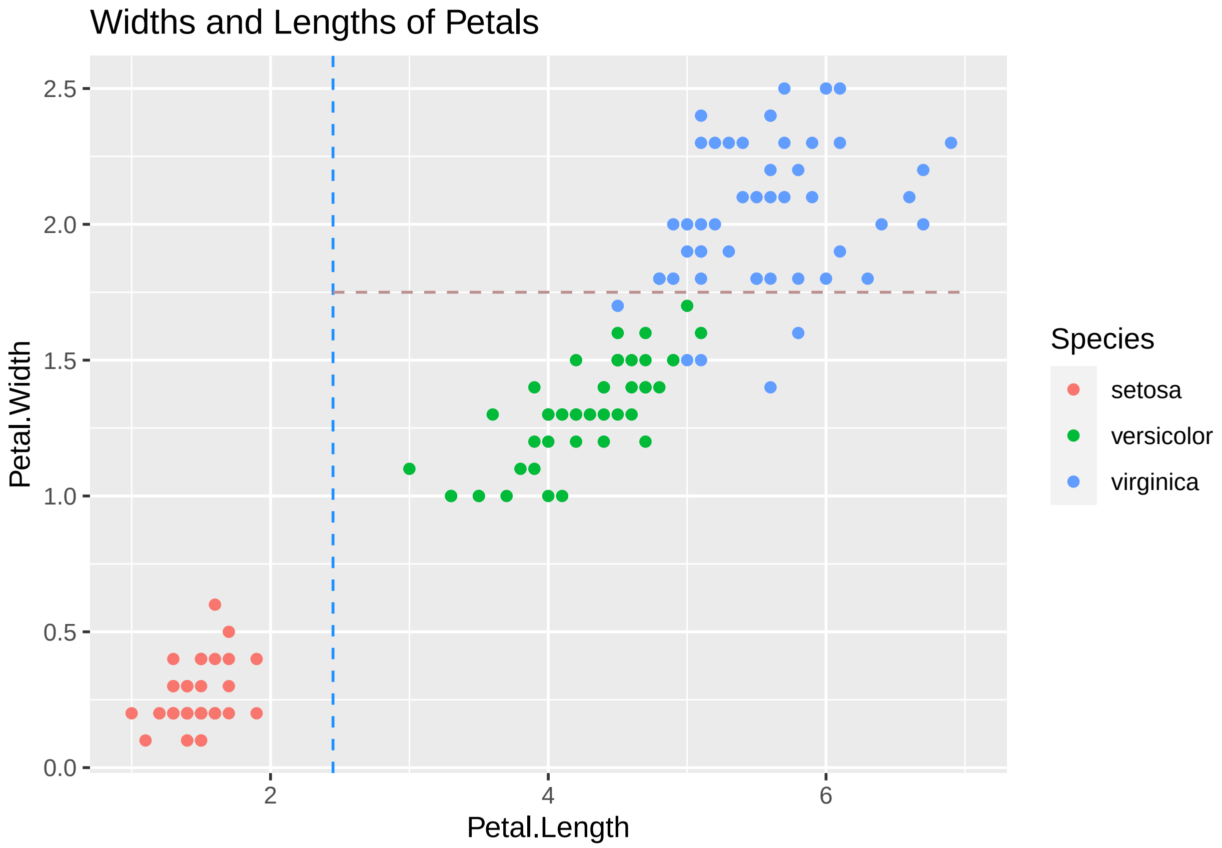
```
## First plot:
par(xpd = TRUE)
plot(my.tree, compress = TRUE)
text(my.tree, use.n = TRUE)
```



```
par(xpd = FALSE)

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

```r
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
```



- A flower whose Petal.Length is 3.0 cm and whose Petal.Width is 1.5 cm.

Based on the tree diagram, this type of flower will be labeled as versicolor.

- A flower whose Petal.Length is 4.0 cm and whose Petal.Width is 2.1 cm.

Based on the dimensions of the petal, this flower would be classified as a virginica.

d) Use predict(), with my.tree, to predict the species of the flowers in the newIris data set. Report the two predictions and compare them to those of Part c.
Code:

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

```
##   Petal.Length Petal.Width
## 1            3         1.5
## 2            4         2.1
```

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

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

The results are the same as above where the first flower with petal length 3 and petal width of 1.5 was predicted to be versicolor. The second flower with petal length 4 and petal width of 2.1 was classified as virginica.

Exercise 4: Consider again the (built-in) iris data set. This time fit the decision tree for classifying flowers into the different Species based on their Sepal.Length and Sepal.Width:

Decision Tree:

```r
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) How many terminal nodes does this tree have?
There seems to be five terminal nodes.

b) What is the tree's correct classification rate (as a percent)? What is it's misclassification rate (as a percent)?
Code:

```
preds <- predict(my.tree, type = "class")
iris <- mutate(iris, predSpecies = preds)
# The conf_mat() function automatically converts Species and predSpecies to factors:
conf_mat(data = iris, truth = Species, estimate = predSpecies)
```

```
##             Truth
## Prediction   setosa versicolor virginica
##   setosa         49          3         0
##   versicolor      1         31        11
##   virginica       0         16        39
```

Classification Rate Correct/Incorrect

```
correct <- ((49+31+39) / (49+3+1+31+11+16+39)) * 100
correct
```

```
## [1] 79.33333
```

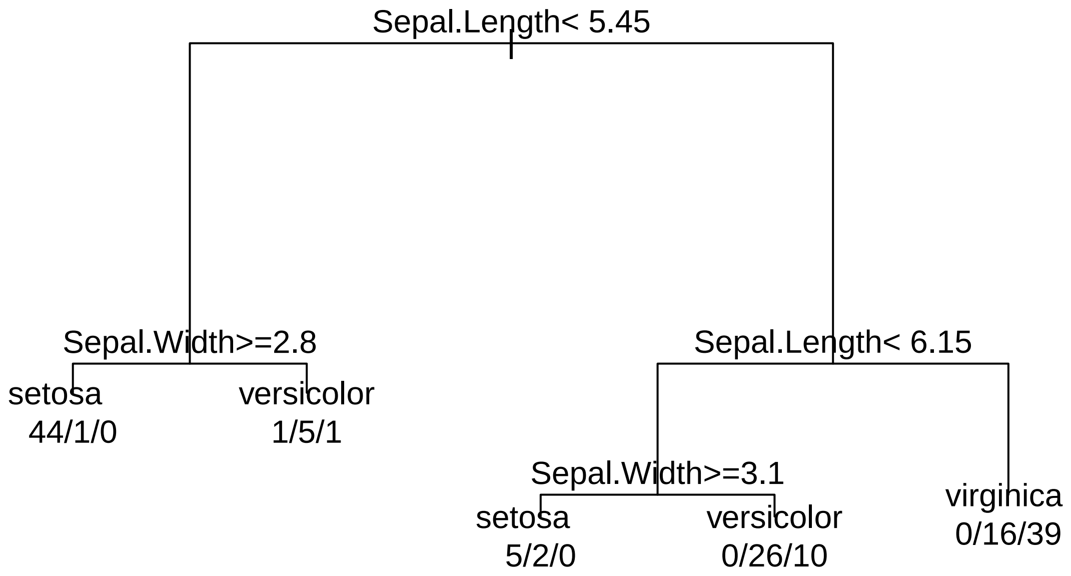
```
incorrect <- 100 - correct
incorrect
```

```
## [1] 20.66667
```

The tree's correct classification rate is now 79.33% and the incorrect rate is 20.67%.

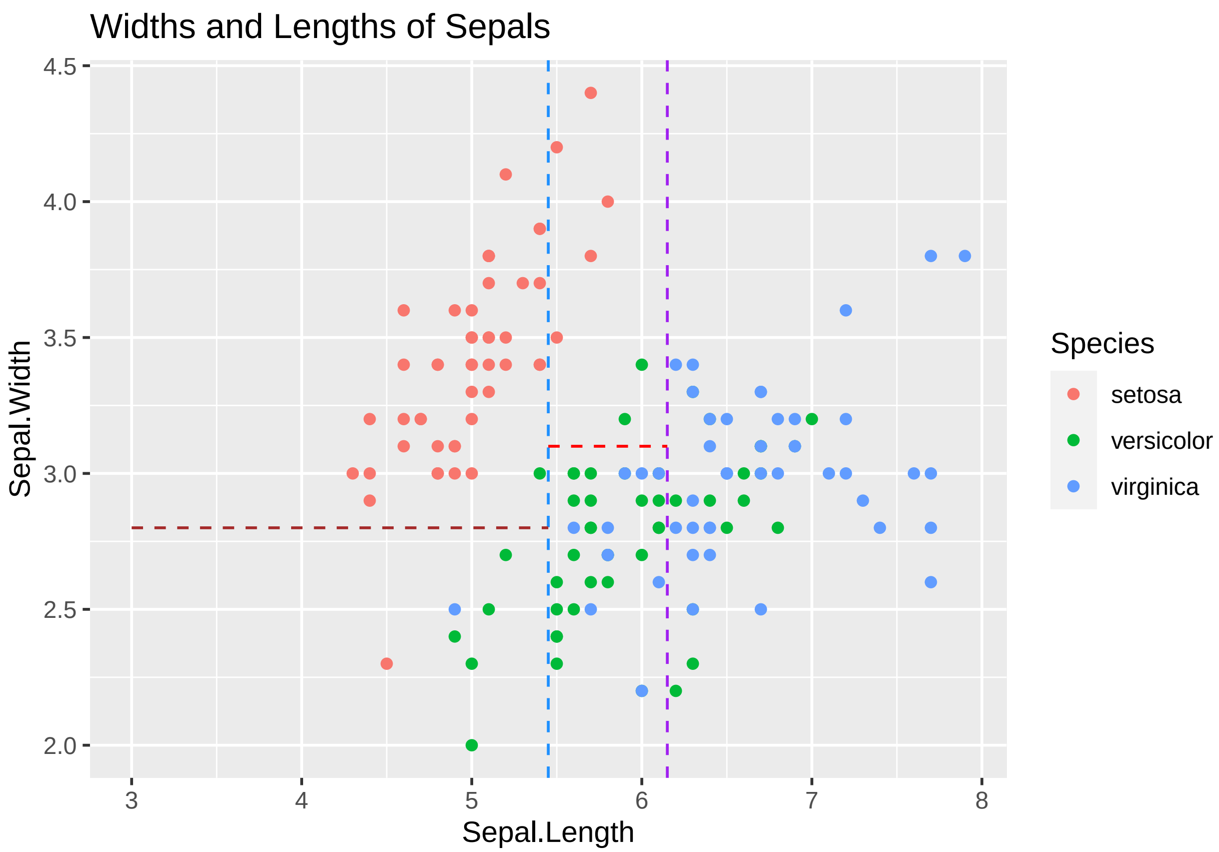
c) Use the plots (not predict()) to classify the the following flowers into Species:
Code:

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



```
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
```



- A flower whose Sepal.Length is 6.0 cm and whose Sepal.Width is 3.5 cm?

This flower should be classified as a setosa.

- A flower whose Sepal.Length is 7.0 cm and whose Sepal.Width is 3.0 cm?

This flower shoudl be classified as a virginica.

d) Use predict(), with my.tree, to predict the species of the flowers in the newIris data set. Report the two predictions and compare them to those of Part c. Code:

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

```
##   Sepal.Length Sepal.Width
## 1            6         3.5
## 2            7         3.0
```

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

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

The results are the same as *part c*. The first flower is classified as setosa and the second flower is classified as virginica.

Exercise 5: Do the following using the Gini index.

Gini Index:

```r
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
```

```r
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) Guess which of the two data sets, y1 and y2, is ”purer” according to the Gini index, then check your answer by computing G using expression 2 with the proportions p1, p2, and p3 shown below.
Based on the data sets, y1 and y2, my guess of which data is more pure is y1 because the overall sum of the squares for y1 seems better than the sum of the squares in y2.

Gini Calc:

```r
g1 <- 1 - (0.1^2 + 0.1^2 + 0.8^2)
g1
```

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

```r
g2 <- 1 - (0.3^2 + 0.3^2 + 0.3^2)
g2
```

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

b) What is the Gini index value for the following data set?
Code:

```r
y <- c("A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A")
round(prop.table(table(y)), digits = 1)
```

```
## y
## A
## 1
```

The Gini index value for the data above is 0 because the variable y contains only the value “A” in the data so it would be pure.

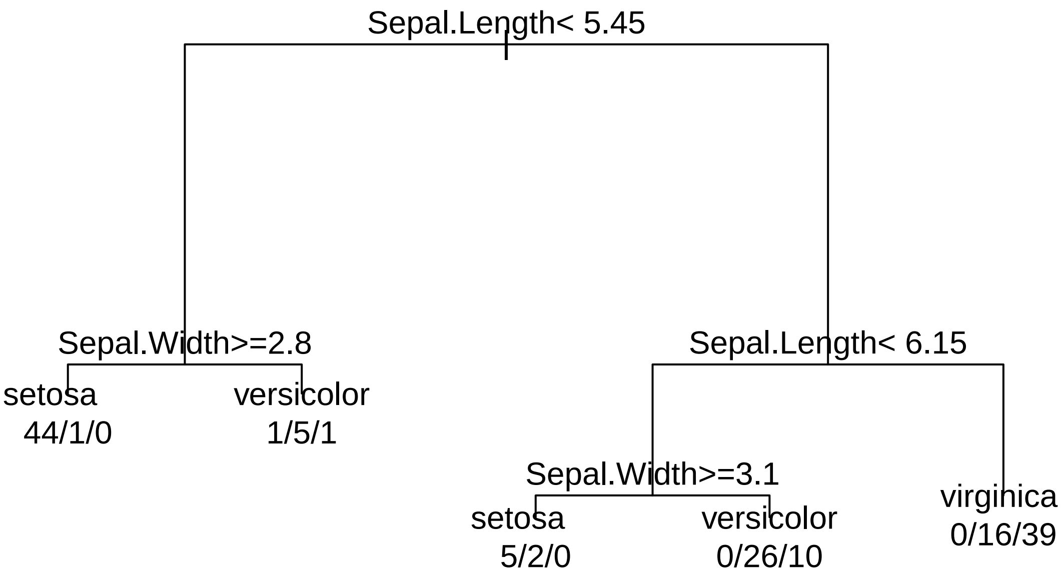
Exercise 6: Which threshold value, 0.2% or 5%, resulted in more terminal nodes (i.e. higher complexity in the tree)?

Code:

```
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) *
```

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

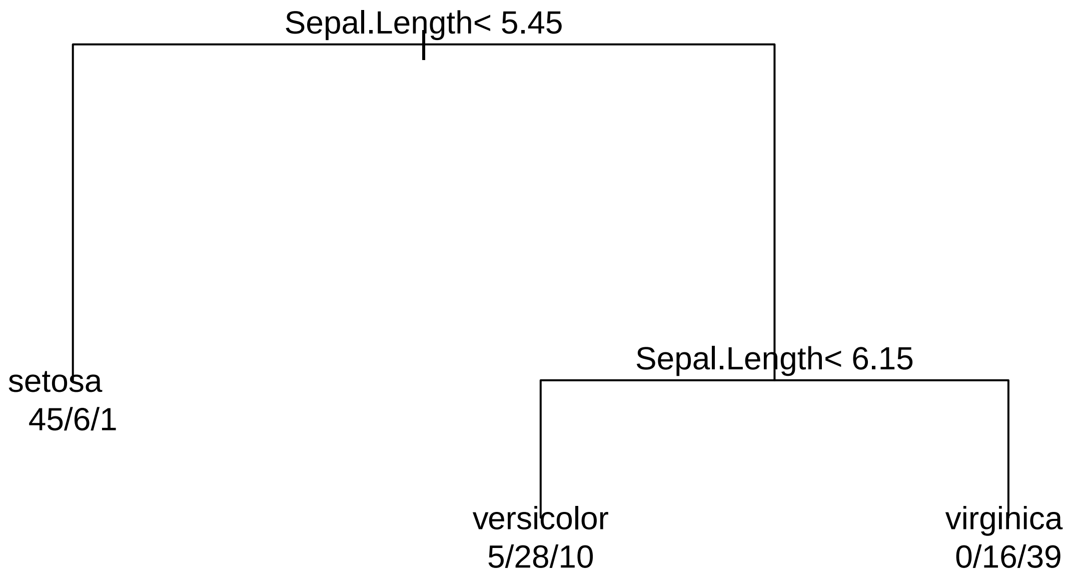


```
par(xpd = FALSE)

my.tree1 <- rpart(Species ~ Sepal.Length + Sepal.Width, data = iris,
control = rpart.control(cp = 0.05))
my.tree1
```

```
## 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.tree1, compress = TRUE)
text(my.tree1, use.n = TRUE)
```



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

The tree with 0.2% threshold resulted in more terminal nodes and higher complexity.

Exercise 7: Compute the (in-sample) correct classification rate for each of the following random forests for predicting Species using the iris data and the accuracy() function (from the "yardstick" package):

a) Using mtry = 1 in randomForest()
Code:

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

my.forest
```

```
##
## Call:
##  randomForest(formula = Species ~ Sepal.Length + Sepal.Width +      Petal.Length + Petal.Width, data
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 6%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         50          0         0        0.00
## versicolor      0         46         4        0.08
## virginica       0          5        45        0.10
```

```
100 - 5.33
```

```
## [1] 94.67
```

```
accuracy(data = iris, truth = as.factor(Species), estimate = as.factor(predSpecies))
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.793
```

The accuracy using mtry = 1 is 94.67% correct estimating species in the iris data set.

b) Using mtry = 3 in randomForest():
Code:

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

my.forest
```

```
##
## Call:
##  randomForest(formula = Species ~ Sepal.Length + Sepal.Width +      Petal.Length + Petal.Width, data
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 4%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         50          0         0        0.00
## versicolor      0         47         3        0.06
## virginica       0          3        47        0.06
```

```
accuracy(data = iris, truth = as.factor(Species), estimate = as.factor(predSpecies))
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.793
```

```
100 - 4.67
```

```
## [1] 95.33
```

The correct classification rate is 95.33% accurate.

Exercise 8: Carry out bagging for classification (predicting Species) using the iris data by setting mtry = 4 in randomForest():

Bagging Operation:

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

a) Which of the four explanatory variables is most important for predicting Species? Which is least important?
Code:

```
importance(my.forest)
```

```
##              MeanDecreaseGini
## Sepal.Length         1.255936
## Sepal.Width          1.376543
## Petal.Length        45.399012
## Petal.Width         51.260376
```

According to the output of the importance() function, the most important variable is petal.width, then petal.length followed by sepal.width and sepal.length. The least important variable is sepal.length.

b) Report the three species predictions.
Code:

```
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)
)
newIris
```

```
##   Petal.Length Petal.Width Sepal.Length Sepal.Width
## 1          3.0         1.2          5.5         3.0
## 2          2.2         2.1          5.1         2.7
## 3          2.7         1.6          5.9         3.2
```

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

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

Exercise 9: Do the following.

a) Experiment with a few different values of k by editing the code below. Report the correct classification rate (for observations in the original data set, i.e. in-sample observations) for the different values of k you chose.

Code:

```
# Change this value to try different k, then run the code
# below for each choice of k:
ktry <- 3
# k nearest neighbor classification procedure:
my.knn <- kknn(
  Species ~ Petal.Length + Petal.Width,
  train = iris,
  test = iris,
  k = ktry
)
preds <- fitted(my.knn)
iris <- mutate(iris, predSpecies = preds)

# The accuracy() function requires the arguments truth and estimate to be factors:
accuracy(data = iris, truth = as.factor(Species), estimate = as.factor(predSpecies))
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.993
```

```
ktry <- 2
# k nearest neighbor classification procedure:
my.knn <- kknn(
  Species ~ Petal.Length + Petal.Width,
  train = iris,
  test = iris,
  k = ktry
)
preds <- fitted(my.knn)
iris <- mutate(iris, predSpecies = preds)
accuracy(data = iris, truth = as.factor(Species), estimate = as.factor(predSpecies))
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.993
```

```
ktry <- 5
# k nearest neighbor classification procedure:
my.knn <- kknn(
  Species ~ Petal.Length + Petal.Width,
  train = iris,
  test = iris,
  k = ktry
)
preds <- fitted(my.knn)
iris <- mutate(iris, predSpecies = preds)
accuracy(data = iris, truth = as.factor(Species), estimate = as.factor(predSpecies))
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass      0.98
```

```
ktry <- 7
# k nearest neighbor classification procedure:
my.knn <- kknn(
  Species ~ Petal.Length + Petal.Width,
  train = iris,
  test = iris,
  k = ktry
)
preds <- fitted(my.knn)
iris <- mutate(iris, predSpecies = preds)
accuracy(data = iris, truth = as.factor(Species), estimate = as.factor(predSpecies))
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.967
```

```
ktry <- 9
# k nearest neighbor classification procedure:
my.knn <- kknn(
  Species ~ Petal.Length + Petal.Width,
  train = iris,
  test = iris,
  k = ktry
)
preds <- fitted(my.knn)
iris <- mutate(iris, predSpecies = preds)
accuracy(data = iris, truth = as.factor(Species), estimate = as.factor(predSpecies))
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.973
```

b) Do your predictions for the 4.35 cm long, 1.65 cm wide flower change with the choice of k? Edit the code below to find out. Code:

```
# Change this value to try different k, then run the code
# below for each choice of k:
ktry <- 3
# Data frame containing new flower for classification:
newIris <- data.frame(Petal.Length = 4.35,
Petal.Width = 1.65)
# k nearest neighbor classification procedure:
my.knn <- kknn(Species ~ Petal.Length + Petal.Width,
train = iris,
test = newIris,
k = ktry)
# Get the classification of the new iris flower:
pred <- fitted(my.knn)
pred
```

```
## [1] versicolor
## Levels: setosa versicolor virginica
```

```
ktry <- 2
# Data frame containing new flower for classification:
newIris <- data.frame(Petal.Length = 4.35,
Petal.Width = 1.65)
# k nearest neighbor classification procedure:
my.knn <- kknn(Species ~ Petal.Length + Petal.Width,
train = iris,
test = newIris,
k = ktry)
# Get the classification of the new iris flower:
pred <- fitted(my.knn)
pred
```

```
## [1] versicolor
## Levels: setosa versicolor virginica
```

```
ktry <- 5
# Data frame containing new flower for classification:
newIris <- data.frame(Petal.Length = 4.35,
Petal.Width = 1.65)
# k nearest neighbor classification procedure:
my.knn <- kknn(Species ~ Petal.Length + Petal.Width,
train = iris,
test = newIris,
k = ktry)
# Get the classification of the new iris flower:
pred <- fitted(my.knn)
pred
```

```
## [1] versicolor
## Levels: setosa versicolor virginica
```

```r
ktry <- 7
# Data frame containing new flower for classification:
newIris <- data.frame(Petal.Length = 4.35,
Petal.Width = 1.65)
# k nearest neighbor classification procedure:
my.knn <- kknn(Species ~ Petal.Length + Petal.Width,
train = iris,
test = newIris,
k = ktry)
# Get the classification of the new iris flower:
pred <- fitted(my.knn)
pred
```

```
## [1] versicolor
## Levels: setosa versicolor virginica
```

```r
ktry <- 9
# Data frame containing new flower for classification:
newIris <- data.frame(Petal.Length = 4.35,
Petal.Width = 1.65)
# k nearest neighbor classification procedure:
my.knn <- kknn(Species ~ Petal.Length + Petal.Width,
train = iris,
test = newIris,
k = ktry)
# Get the classification of the new iris flower:
pred <- fitted(my.knn)
pred
```

```
## [1] versicolor
## Levels: setosa versicolor virginica
```

The prediction does not change with my inclusion of more ktry points.

Exercise 10: Which value, k = 1, 3, 5, or 7, resulted in the smallest mean squared (prediction) error?

Code:

```
# Rescale the variables in rock so they're on roughly equal scales:
rockRescaled <- rock %>% mutate(area = area / 10000,
                                peri = peri / 10000,
                                perm = log(perm))
```

Code ktry 1:

```
ktry <- 1
my.nn <- nnet(
  perm ~ area + peri + shape,
  data = rockRescaled,
  size = ktry,
  linout = TRUE,
  maxit = 1000,
  trace = FALSE
)
preds <- predict(my.nn)
rockRescaled <- mutate(rockRescaled, predPerm = preds)
squaredPredErrors <- (rockRescaled$perm - rockRescaled$predPerm) ^ 2
mean(squaredPredErrors) # Mean squared (prediction) error
```

```
## [1] 0.5010857
```

Code ktry 3:

```
# Neural network procedure for prediction using k = 3 hidden units.
# Change this value to try different k, then run the code below
# for each choice of k:
ktry <- 3
my.nn <- nnet(
  perm ~ area + peri + shape,
  data = rockRescaled,
  size = ktry,
  linout = TRUE,
  maxit = 1000,
  trace = FALSE
)

# Get the predicted permeabilities:
preds <- predict(my.nn)

# Insert the predicted permeabilities as a new column in rock:
rockRescaled <- mutate(rockRescaled, predPerm = preds)

squaredPredErrors <- (rockRescaled$perm - rockRescaled$predPerm) ^ 2
mean(squaredPredErrors) # Mean squared (prediction) error
```

```
## [1] 0.214879
```

Code ktry 5:

```
ktry <- 5
my.nn <- nnet(
  perm ~ area + peri + shape,
  data = rockRescaled,
  size = ktry,
  linout = TRUE,
  maxit = 1000,
  trace = FALSE
)
preds <- predict(my.nn)
rockRescaled <- mutate(rockRescaled, predPerm = preds)
squaredPredErrors <- (rockRescaled$perm - rockRescaled$predPerm) ^ 2
mean(squaredPredErrors) # Mean squared (prediction) error
```

```
## [1] 0.1446055
```

Code ktry 7:

```
ktry <- 7
my.nn <- nnet(
  perm ~ area + peri + shape,
  data = rockRescaled,
  size = ktry,
  linout = TRUE,
  maxit = 1000,
  trace = FALSE
)
preds <- predict(my.nn)
rockRescaled <- mutate(rockRescaled, predPerm = preds)
squaredPredErrors <- (rockRescaled$perm - rockRescaled$predPerm) ^ 2
mean(squaredPredErrors) # Mean squared (prediction) error
```

```
## [1] 0.1110572
```

Code ktry 9:

```
ktry <- 9
my.nn <- nnet(
  perm ~ area + peri + shape,
  data = rockRescaled,
  size = ktry,
  linout = TRUE,
  maxit = 1000,
  trace = FALSE
)
preds <- predict(my.nn)
rockRescaled <- mutate(rockRescaled, predPerm = preds)
squaredPredErrors <- (rockRescaled$perm - rockRescaled$predPerm) ^ 2
mean(squaredPredErrors) # Mean squared (prediction) error
```

```
## [1] 0.07713901
```

The smallest mean squared(prediction) error is when ktry equals 9.

Exercise 11: Do the following.

Data:

```
# Neural network classification procedure with k = 3 hidden units:
my.nn <-
  nnet(
    Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
    data = iris,
    size = 3,
    maxit = 200,
    trace = FALSE
  )
summary(my.nn)
```

```
## a 4-3-3 network with 27 weights
## options were - softmax modelling
##  b->h1 i1->h1 i2->h1 i3->h1 i4->h1
##  30.85 111.94 109.99 -62.67 -43.07
##  b->h2 i1->h2 i2->h2 i3->h2 i4->h2
##  -3.62  -5.54 -14.74  25.33  10.37
##  b->h3 i1->h3 i2->h3 i3->h3 i4->h3
##  -6.34 -24.89  59.92 -49.88  30.54
##  b->o1 h1->o1 h2->o1 h3->o1
##  24.09  -8.81 -27.46  23.61
##  b->o2 h1->o2 h2->o2 h3->o2
##   4.14 -11.31  13.54  -0.91
##  b->o3 h1->o3 h2->o3 h3->o3
## -28.97  21.08  14.26 -23.18
```

```
# Data frame containing new flowers for classification:
newIris <- data.frame(
  Petal.Length = c(1.5, 5.2, 5.5),
  Petal.Width = c(0.2, 1.9, 2.0),
  Sepal.Length = c(5.0, 6.3, 6.5),
  Sepal.Width = c(3.4, 2.9, 2.9)
)
```

a) Use predict() (with type = "class") to classify (predict the Species of) these three new flowers. Report your R command(s) and the three Species predictions.

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

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

b) Remove type = "class" from your predict() command of part a, so that the (estimated) probabilities p1, p2, p3 for each of the three Species are returned for each of the three flowers in newIris. Report, for each of the three flowers, the Species whose probability is highest. Hint: They should be the same as those predicted in part a.

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

```
##         setosa   versicolor    virginica
## 1 1.000000e+00 5.757493e-21 8.258311e-31
## 2 4.320506e-09 5.000000e-01 5.000000e-01
## 3 4.320506e-09 5.000000e-01 5.000000e-01
```

The first one is setosa, the second is virginica/versicolor, and the last one is also virginica/versicolor. They interchange based on the time the above code is run.

Section 11.3 Exercises

Exercise 12: Do the following.

Snakes Data:

```
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)

pred0 <- predict(mod0, newdata = newSnakes)
pred1 <- predict(mod1, newdata = newSnakes)
pred2 <- predict(mod2, newdata = newSnakes)
pred3 <- predict(mod3, newdata = newSnakes)
pred4 <- predict(mod4, newdata = newSnakes)
pred5 <- predict(mod5, newdata = newSnakes)

newSnakes <- mutate(newSnakes,
predWeight0 = pred0,
predWeight1 = pred1,
predWeight2 = pred2,
predWeight3 = pred3,
predWeight4 = pred4,
predWeight5 = pred5)
```

a) Which of the six polynomial models is best according to the (out-of-sample) MSE?

MSE Calcs:

```r
mse0 <- mean((newSnakes$Weight - newSnakes$predWeight0)^2)
mse1 <- mean((newSnakes$Weight - newSnakes$predWeight1)^2)
mse2 <- mean((newSnakes$Weight - newSnakes$predWeight2)^2)
mse3 <- mean((newSnakes$Weight - newSnakes$predWeight3)^2)
mse4 <- mean((newSnakes$Weight - newSnakes$predWeight4)^2)
mse5 <- mean((newSnakes$Weight - newSnakes$predWeight5)^2)

mean_tab <- c(mse0, mse1, mse2, mse3, mse4, mse5)
mean_tab
```

```
## [1] 7783.052 1194.625 1347.830 2445.738 3049.605 2892.007
```

The mod1 model is the best for predicting according the its out-of-sample MSE.

b) Which of the six polynomial models is best according to the (out-of-sample) MAE?

MAE Calcs:

```r
mae0 <- mean(abs(newSnakes$Weight - newSnakes$predWeight0))
mae1 <- mean(abs(newSnakes$Weight - newSnakes$predWeight1))
mae2 <- mean(abs(newSnakes$Weight - newSnakes$predWeight2))
mae3 <- mean(abs(newSnakes$Weight - newSnakes$predWeight3))
mae4 <- mean(abs(newSnakes$Weight - newSnakes$predWeight4))
mae5 <- mean(abs(newSnakes$Weight - newSnakes$predWeight5))

mae_tab <- c(mae0, mae1, mae2, mae3, mae4, mae5)
mae_tab
```

```
## [1] 74.96889 29.79193 29.45119 44.99129 47.87050 45.09594
```

The best fitting model is the mod2 according to the mae values calculated.