Decision Tree with the Iris Data

Karen Mazidi

Using rpart

The **rpart** package is a popular package for making decision trees. The code below builds a tree on the iris data set.

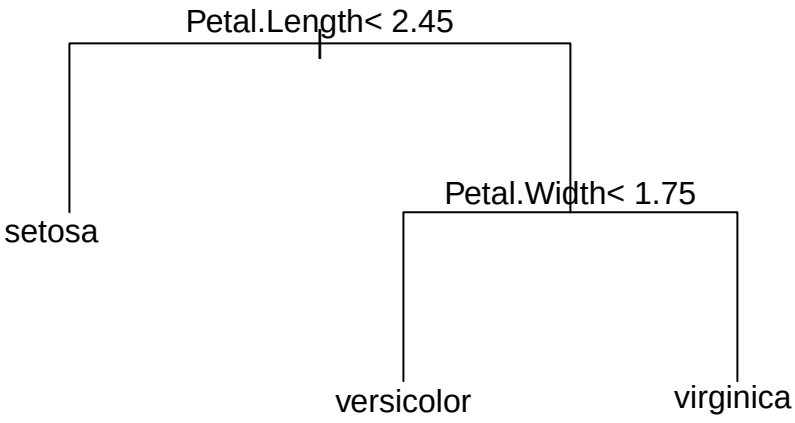
The variable **tree_iris** contains the tree. When output, the indents show the tree branches. On each split the information given is the node, the variable/value of the split, the number of observations, the loss (in number of observations), the class, and the probabilities for each class in parenthesis.

```
library(rpart)
tree_iris <- rpart(Species~., data=iris, method="class")
tree_iris
```

```
## 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) *
```

The rpart tree can be plotted as follows.

```
plot(tree_iris, uniform=TRUE,margin=0.2)
text(tree_iris)
```



Using tree

The tree() package is also popular for decision trees. Notice that it made more splits than rpart. Both packages have many parameters to adjust to give different results. The tree display is similar to the display of rpart, but it gives a deviance metric instead of loss.

```
library(tree)
tree_iris2 <- tree(Species~., data=iris)
tree_iris2
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 150 329.600 setosa ( 0.33333 0.33333 0.33333 )
##    2) Petal.Length < 2.45 50   0.000 setosa ( 1.00000 0.00000 0.00000 ) *
##    3) Petal.Length > 2.45 100 138.600 versicolor ( 0.00000 0.50000 0.50000 )
##      6) Petal.Width < 1.75 54  33.320 versicolor ( 0.00000 0.90741 0.09259 )
##       12) Petal.Length < 4.95 48   9.721 versicolor ( 0.00000 0.97917 0.02083 )
##         24) Sepal.Length < 5.15 5   5.004 versicolor ( 0.00000 0.80000 0.20000 ) *
##         25) Sepal.Length > 5.15 43   0.000 versicolor ( 0.00000 1.00000 0.00000 ) *
##       13) Petal.Length > 4.95 6   7.638 virginica ( 0.00000 0.33333 0.66667 ) *
##      7) Petal.Width > 1.75 46   9.635 virginica ( 0.00000 0.02174 0.97826 )
##       14) Petal.Length < 4.95 6   5.407 virginica ( 0.00000 0.16667 0.83333 ) *
##       15) Petal.Length > 4.95 40   0.000 virginica ( 0.00000 0.00000 1.00000 ) *
```

The summary() function gives an overview of the model, its deviance, and error rate.

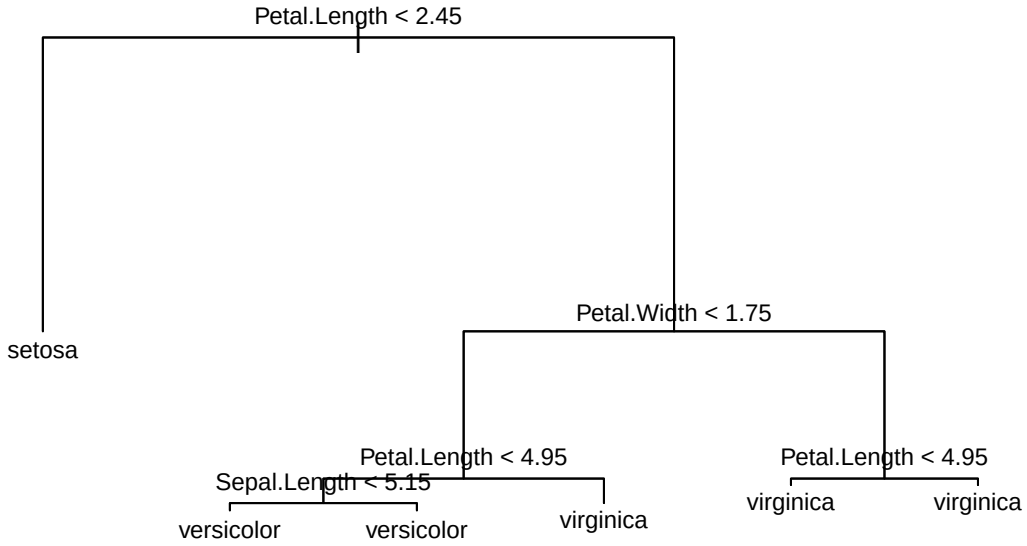
```
summary(tree_iris2)
```

```
##
## Classification tree:
## tree(formula = Species ~ ., data = iris)
## Variables actually used in tree construction:
## [1] "Petal.Length" "Petal.Width"  "Sepal.Length"
## Number of terminal nodes:  6
## Residual mean deviance:  0.1253 = 18.05 / 144
## Misclassification error rate: 0.02667 = 4 / 150
```

The plot is shown below. The tree starts out the same as the rpart tree, but makes additional splits.

Notice that some leaves have the same class. For example, at the bottom right, both leaf nodes are of class virginica. The split was made because one class may be more pure than the other.

```
plot(tree_iris2)
text(tree_iris2, cex=0.75, pretty=0)
```



train and test

The following makes another tree using only training data, and evaluates on test data. The accuracy was 94%.

```
set.seed(1958)
i <- sample(150, 100, replace=FALSE)
train <- iris[i,]
test <- iris[-i,]
tree_iris3 <- tree(Species~., data=train)
pred <- predict(tree_iris3, newdata=test, type="class")
table(pred, test$Species)
```

```
##
## pred         setosa versicolor virginica
##   setosa         18          0         0
##   versicolor      0         14         0
##   virginica       0          3        15
```

```
mean(pred==test$Species)
```

```
## [1] 0.94
```