Decision Trees and Random Forests in R
Stan Hatko

If they are not already installed, we first need to install the R packages rpart and randomForest:

```r
install.packages("rpart")
install.packages("randomForest")
```

To load these packages into R:

```r
library(rpart)
library(randomForest)
```

Suppose that we have \(X_1, X_2 \sim \text{Uniform}(0, 1)\) and we have a response

\[
Y = \begin{cases} 
1 & \text{if } X_1 > 0.4 \text{ and } X_2 > 0.6 \\
0 & \text{otherwise}
\end{cases}
\]

We construct the dataset:

```r
n <- 5000
x <- cbind(runif(n), runif(n))
y <- factor(ifelse(x[,1] > .4, x[,2] > .6, 0))
r <- data.frame(x, y)
```

We construct a decision tree for this using rpart:

```r
tree <- rpart(y ~ X1 + X2, data = r, method = "class")
printcp(tree)
```

We can generate a simple diagram of this tree:

```r
plot(tree, compress = TRUE, mar = c(.2, .2, .2, .2))
text(tree, use.n = TRUE)
```

We can generate predictions using this tree on data using the predict function. Here we generate a testing set the same way as the training set above and we find the accuracy of our classifier:
tx <- cbind(runif(n), runif(n))
ty <- factor(ifelse(x[,1] > .4, x[,2] > .6, 0))
p <- predict(tree, data.frame(tx), type = "class")
mean(p == ty) # determine accuracy of prediction

As another example, suppose we have $X_1, X_2 \sim \text{Uniform}(0, 1)$ iid and we have a response

$$Y = \begin{cases} 
1 & \text{if } X_1 + X_2 > 1 \\
0 & \text{otherwise}
\end{cases}$$

We construct the dataset:

n <- 5000
x <- cbind(runif(n), runif(n))
y <- factor(ifelse(x[,1] + x[,2] > 1, 1, 0))
r <- data.frame(x, y)

We can again construct a decision tree using rpart:

tree <- rpart(y ~ X1 + X2, data = r, method = "class")
printcp(tree)

We cross-validate our tree classifier:

tx <- cbind(runif(n), runif(n))
ty <- factor(ifelse(tx[,1] + tx[,2] > 1, 1, 0))
p <- predict(tree, data.frame(tx), type = "class")
mean(p == ty) # determine accuracy of prediction

To control the minimum information gain threshold, we can use the cp parameter (complexity parameter) to rpart, the default being 0.01. For instance, if we use a larger value, we obtain a simpler but less accurate tree:

tree <- rpart(y ~ X1 + X2, data = r, method = "class", cp = 0.1)
printcp(tree)

If we use a smaller value, we obtain a more complex but more accurate tree:

tree <- rpart(y ~ X1 + X2, data = r, method = "class", cp = 0.001)
printcp(tree)

To generate predictions using the random forest classifier, we use the function randomForest:

rf <- randomForest(x, y)
To change the number of trees that are generated, we can use the ntree parameter, whose default is 500 trees. We can also change the number of variables that are sampled as candidates for each split using the mtry parameter, which is by default $\sqrt{m}$ where $m$ is the number of variables in the dataset. The sampsize parameter sets the size of each sample, by default it is the number of points in the data set (when the samples are picked with replacement, as is the default). The nodesize parameter sets the minimum size of each node. There are various other parameters including importance (in which the importance of predictors is assessed) and proximity (in which a proximity measure among the points is calculated). For instance, if we want 1000 trees and a minimum node size of 5, we use

\[
\text{rf} \leftarrow \text{randomForest}(x, y, \text{ntree} = 1000, \text{nodesize} = 5)
\]

To generate the predictions from the random forest object, we can use the function predict. We can use this to cross-validate our previous dataset with random forest:

\[
\text{tx} \leftarrow \text{cbind(\text{runif}(n), \text{runif}(n))}
\]
\[
\text{ty} \leftarrow \text{factor(\text{ifelse(tx[,1] + tx[,2] > 1, 1, 0)})}
\]
\[
\text{p} \leftarrow \text{predict(rf, tx)}
\]
\[
\text{mean(p == ty)} \# \text{determine accuracy of prediction}
\]

We now show an example where random forest is not consistent. We take the uniform distribution on diagonal from $(0, 0)$ to $(1, 1)$ in $\mathbb{R}^2$, and for each point we have a $2/3$ probability of that point being 0 and a $1/3$ probability of that point being 1. We see that the Bayes error is $L^* = 1/3$, which consists of predicting 0 always, which is correct for each point with probability $2/3$. We now find the accuracy of the random forest classifier on this dataset:

\[
\text{n} \leftarrow 5000
\]
\[
\text{x} \leftarrow \text{runif(n) * cbind(rep(1, n), rep(1, n))}
\]
\[
\text{y} \leftarrow \text{factor(\text{ifelse(runif(n) < 1 / 3, 1, 0)})}
\]
\[
\text{rf} \leftarrow \text{randomForest(x, y)}
\]
\[
\text{tx} \leftarrow \text{runif(n) * cbind(rep(1, n), rep(1, n))}
\]
\[
\text{ty} \leftarrow \text{factor(\text{ifelse(runif(n) < 1 / 3, 1, 0)})}
\]
\[
\text{p} \leftarrow \text{predict(rf, tx)}
\]
\[
\text{mean(p == ty)} \# \text{determine accuracy of prediction}
\]

We see that this accuracy is approximately $5/9$, which is the accuracy obtained by guessing the same value as the nearest neighbour.