Blossoming with R

In this lab, you will be learning the basics of building a Classification And Regression Tree (CART) with the R programming language. Before you can begin this exercise, you will need to obtain some data. We will be using a classic machine learning data set known as the "iris data" in this lab. You can download this data set from http://archive.ics.uci.edu/ml/machine-learning-databases/iris/, by clicking on the "iris.data" link. This data set consists of measurements from 150 different irises of three different species. Each data point represents one measurement in centimeters. The final element in each line of the data file is the name of the species of iris for that instance. See the below example:

```
5.1,3.5,1.4,0.2,Iris-setosa
```

This line can be divided up into the following components:

5.1	3.5	1.4	0.2	Iris-setosa
V1	V2	V3	V4	V5

V1-V4 all represent measurements taken from this particular iris. V5 indicates that this iris is of the species Iris-setosa. In this exercise, V1-V4 is our prediction variables, and V5 is the classification variable, or the variable we are trying to predict.

Once you save the data file, move it to the directory you'll be working in for this lab.

Most machine learning tasks require you to perform some pre-processing on your data set. However, using CART in R is very user friendly and you will not be required to do any pre-processing on the data for this task. R will accept the data in the Comma Separated Value (CSV) format of the iris data.

The first step in this exercise is to install and import the library containing CART, known as "rpart." To install rpart, simply begin an R session in your command line (type R). Next, enter

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

This should install the rpart package to your machine. Next, open up a new file, which will contain your R script. Include the rpart package in this script by typing

```
library(rpart)
```

in your R file.

The next step is to read in the data from the `iris.data` file. To do so, you may use a line similar to the following:

```
inData <- read.csv("iris.data", header=FALSE)
```

This line uses the read.csv function to read in a series of comma separated values from the file "iris.data." The second parameter indicates that there is no header line present on the initial row

of the file. The result of this line is that the data from the iris.data file is now stored in a data frame.

Data frames are the essential data structure in the R language. You can think of a data frame as you would a table. It contains rows of data, in this case each representing an instance of an iris. Each column represents the different values for one variable.

Next we want to establish some parameters for the CART algorithm. The first parameter is the minimum split value. This value will be used to determine how many observations must be present at a node before a split is created on that node. In this example, we'll use 10.

```
minimumSplit <- 10
```

The next parameter we'll specify is the minimum bucket value. This value determines the minimum amount of observations that can be found in a leaf node. The default and usually optimal value for this is the minimum split divided by three. We'll do so explicitly in this exercise.

```
minimumBucket <- minimumSplit/3
```

Now that we have our parameters set up for CART, we will build our model. The line to form the tree is as below:

```
model <- rpart(V5~V1+V2+V3+V4, method="class", data=inData,
        control=rpart.control(minsplit=minimumSplit,
                minBucket=minimumBucket) )
```

In the first portion of this line, you are instructing the algorithm how to perform the learning. Essentially, you are saying to predict column V5, the column containing the iris species. The algorithm will use the data in columns V1, V2, V3, and V4 to predict the value of V5. In the second portion of the line, you are simply telling the algorithm to build a classification tree rather than a regression tree. The data parameter tells the algorithm which data frame to draw data from. The control parameter allows you to provide more specific instructions to the algorithm. In this case, we are setting the minimum split and minimum bucket values that we established prior to calling the algorithm.

Once this is done, CART has built your tree, but it does not plot it for you. To do this, add these three lines to your R file.

```
plot(model, uniform=TRUE, main="CART for Iris Data")
text(model, use.n=TRUE, all=TRUE, cex=.8)
post(model, file="tree.ps", title="CART for Iris Data")
```

These three lines simply plot the data, set some text to make the file version more readable, and then output a tree graph to the file `tree.ps`. If you open this file, you will see that it is nicer looking than the simple tree that your R code displays immediately upon running.

Task:

Using CART to build a tree is an essential step in using R for supervised machine learning. However, we should apply a set of tests to determine how good our model is.

Cross-validation is the process by which a programmer can test the integrity of a machine learning model. In one method, known as *k-folds* cross validation, the data is separated into *k* different groups. Then the model is built (trained) using all but one of these groups, or folds. The remaining fold (the test set) is then classified using the model and the accuracy is noted. This is then repeated for each of the folds as the test set, using the remainder to train the model. The overall accuracy is then tallied and can be used as a measure of accuracy for the algorithm on this particular data set.

Edit the file you built over the course of this exercise to perform k-fold cross-validation on CART.