Lab 9 Classification and Regression Trees (CART)

The goal of this lab is to become familiar with Classification and Regression (CART). CART allows you to determine what variables are important in separating groups. It differs from DA in that it does not have the assumptions of a parametric test and thus provides a more flexible approach to discrimination.

# Set up R session

## Data

Today you will be using the iris data set found in the package datasets in R. To access this data set, simply type:

iris

To learn more about the data set:

`?`(iris)

You will use the “ozone.csv” avaliable in Module 9 in CANVAS for regression trees.

oz <- read.csv("Data/lab\_9/ozone.csv", header = TRUE)

## Download packages

We will be using the following packages:

library(MASS)  
library(rpart)  
library(ade4)  
library(vegan)

# CART

## Setting the training and testing data

Like you did for DA, you are going to split our data set into “training” data and “testing” data. This will allow you to test the predictions of the CART model on a “new” data set.

Randomly select 75 samples from the iris data set. Use “set.seed” so we are all working with the same training set:

set.seed(51)  
train <- sample(1:150, 75)

Check the frequency of each species in the training data, to make sure they are relatively proportional to the frequency in the complete data set.

freq <- table(iris$Sp[train])  
freq

##   
## setosa versicolor virginica   
## 26 26 23

## Specifying the model

Next specify the CART model. In the iris data set, Species is the categorical response variables and the four measures of flower morphology are the explanatory variables:

model <- Species ~ .

**“Species ~ .” is the model and the “dot” stands for all of the variables (so you don’t have to type them all)**

## Running the CART algorithm

You will use the rpart function in the *rpart* package to develop CART models.

`?`(rpart)

iris\_rpart <- rpart(model, data = iris[train, ], method = "class", control = rpart.control(minsplit = 10))

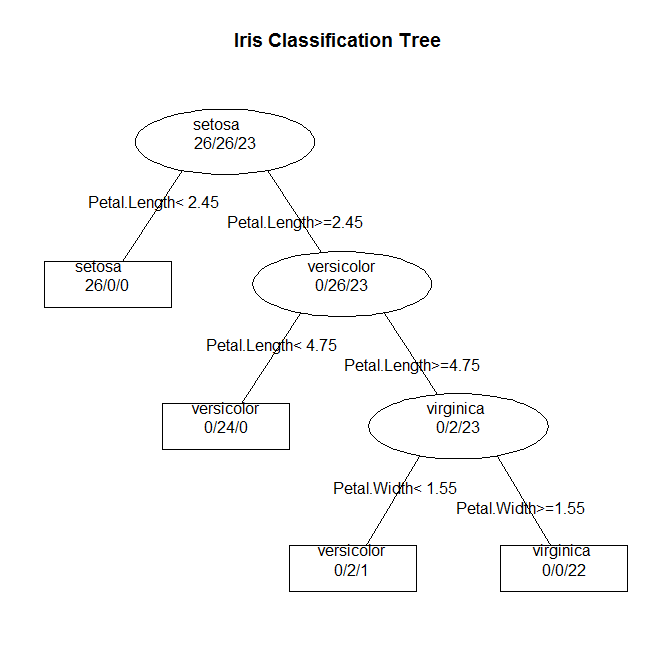
**Look up what rpart.control and it’s parameters do and play with them!**

## Plotting Cart Tree and viewing summary

You will plot your tree using the function post.rpart:

`?`(post.rpart)

post(iris\_rpart, file = "", title = " Iris Classification Tree")



Now, look at a node by node summary of the tree and the variable importance:

summary(iris\_rpart)

## Cost-complexity pruning

Now let’s look at the cross-validation results:

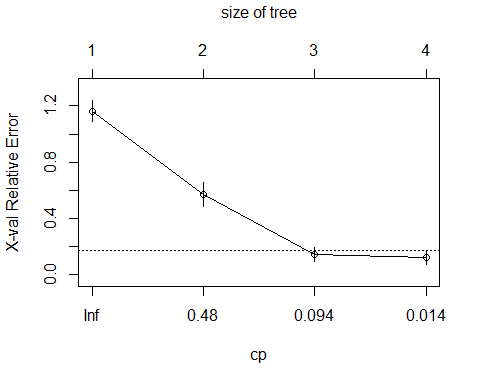
printcp(iris\_rpart)

##   
## Classification tree:  
## rpart(formula = model, data = iris[train, ], method = "class",   
## control = rpart.control(minsplit = 10))  
##   
## Variables actually used in tree construction:  
## [1] Petal.Length Petal.Width   
##   
## Root node error: 49/75 = 0.65333  
##   
## n= 75   
##   
## CP nsplit rel error xerror xstd  
## 1 0.530612 0 1.000000 1.16327 0.075483  
## 2 0.428571 1 0.469388 0.57143 0.085487  
## 3 0.020408 2 0.040816 0.14286 0.051413  
## 4 0.010000 3 0.020408 0.12245 0.047948

* The first column is the complexity parameter (CP) which shows how much each node contributes to the classification rate and thus how misclassification would increase with the removal of a given node.
* The second column shows the split.
* The third column shows the relative error (misclassification rate). This is the proportion of samples misclassified in the “root node” (first node) that are misclassified at any subsequent node. To get the absolute error, multiply these values times the misclassification rate of the first node.
* The fourth column is the estimated error from the cross-validation procedure.
* The fifth column is the standard error from the cross-validation procedure.

Next, plot the results to determine the optimal tree:

plotcp(iris\_rpart)



Remember, you are going to use the 1SE rule of thumb. Select the tree size furthers to the left (i.e. fewest leaves) that is within 1 SE of the minimum estimated error.

How big is you optimal tree?

## Pruning the tree

You will prune the tree using prune.rpart function.

`?`(prune.rpart)

First let’s set the cp for the optimal tree size. This extracts the cp value according to the 1-SE rule. *Note you will have to change the column, row designations for other examples*.

cp <- printcp(iris\_rpart)[3, 1]

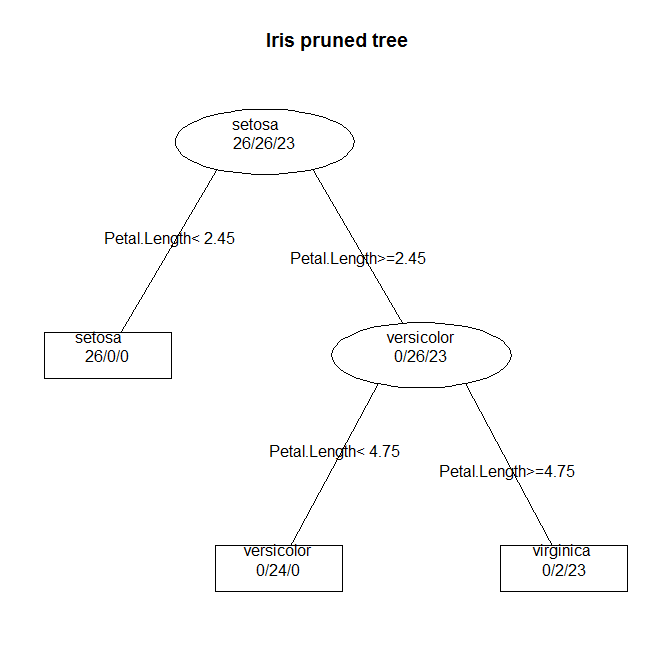
##   
## Classification tree:  
## rpart(formula = model, data = iris[train, ], method = "class",   
## control = rpart.control(minsplit = 10))  
##   
## Variables actually used in tree construction:  
## [1] Petal.Length Petal.Width   
##   
## Root node error: 49/75 = 0.65333  
##   
## n= 75   
##   
## CP nsplit rel error xerror xstd  
## 1 0.530612 0 1.000000 1.16327 0.075483  
## 2 0.428571 1 0.469388 0.57143 0.085487  
## 3 0.020408 2 0.040816 0.14286 0.051413  
## 4 0.010000 3 0.020408 0.12245 0.047948

Now, get out the pruning shears:

iris\_prune <- prune(iris\_rpart, cp = cp)  
print(iris\_prune)

## n= 75   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 75 49 setosa (0.3466667 0.3466667 0.3066667)   
## 2) Petal.Length< 2.45 26 0 setosa (1.0000000 0.0000000 0.0000000) \*  
## 3) Petal.Length>=2.45 49 23 versicolor (0.0000000 0.5306122 0.4693878)   
## 6) Petal.Length< 4.75 24 0 versicolor (0.0000000 1.0000000 0.0000000) \*  
## 7) Petal.Length>=4.75 25 2 virginica (0.0000000 0.0800000 0.9200000) \*

# And plot the pruned tree:  
post(iris\_prune, file = "", title = " Iris pruned tree")



## Classification accuracy

Now we have our optimum tree. Let’s see how accurately both the unpruned and pruned tree classify the testing data:

# classification matrix  
Ct\_unprune <- table(predict(iris\_rpart, iris[-train, ], type = "class"), iris[-train,   
 "Species"])  
Ct\_prune <- table(predict(iris\_prune, iris[-train, ], type = "class"), iris[-train,   
 "Species"])  
  
Ct\_unprune

##   
## setosa versicolor virginica  
## setosa 24 0 0  
## versicolor 0 21 3  
## virginica 0 3 24

Ct\_prune

##   
## setosa versicolor virginica  
## setosa 24 0 0  
## versicolor 0 20 1  
## virginica 0 4 26

# classification accuracy  
class\_unprune <- sum(diag(prop.table(Ct\_unprune)))  
class\_prune <- sum(diag(prop.table(Ct\_prune)))  
  
class\_unprune

## [1] 0.92

class\_prune

## [1] 0.9333333

**Did pruning help much?**

# Compare with DA

Pull up your coed from last week and run a DA for the iris data. Compare classification rates and variable importance.

# Regression Tree

Here we will use regression trees on the “ozone.csv” avaliable in Module 9 in CANVAS. This data set contains information on ozone, radiation, temperature, and wind on 110 days from May to September 1973 in New York. Measurements of daily ozone con- centration (ppb), wind speed (mph), daily maximum temperature (degrees F), and solar radiation (langleys). Here, you will use ozone and the response variable and the remaining variable as predicators (i.e., used to make splits in the tree).

Create a training data set with 60 samples:

train <- sample(1:110, 60)

Specify the model:

model <- ozone ~ .

Utilize rpart to construct the tree:

oz\_rpart <- rpart(model, data = oz[train, ], method = "anova", control = rpart.control(minsplit = 10))

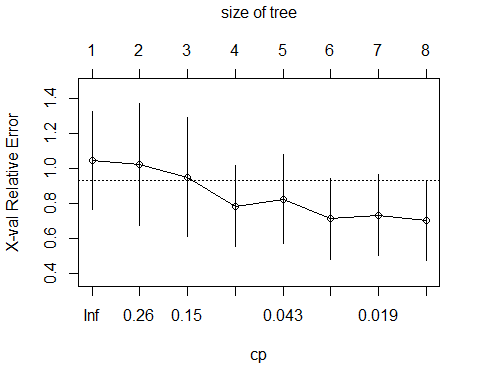
Here we denote the type of tree using method = “anova”, which is used for continues variables.

Now let’s look at the cross-validation results:

printcp(oz\_rpart)

##   
## Regression tree:  
## rpart(formula = model, data = oz[train, ], method = "anova",   
## control = rpart.control(minsplit = 10))  
##   
## Variables actually used in tree construction:  
## [1] radiation temp wind   
##   
## Root node error: 60827/60 = 1013.8  
##   
## n= 60   
##   
## CP nsplit rel error xerror xstd  
## 1 0.361654 0 1.00000 1.04632 0.27807  
## 2 0.189552 1 0.63835 1.02229 0.34518  
## 3 0.126207 2 0.44879 0.95021 0.34005  
## 4 0.048735 3 0.32259 0.78528 0.23238  
## 5 0.038384 4 0.27385 0.82410 0.25300  
## 6 0.020983 5 0.23547 0.71290 0.23098  
## 7 0.017536 6 0.21449 0.73296 0.23052  
## 8 0.010000 7 0.19695 0.70222 0.22929

plotcp(oz\_rpart)



cp <- printcp(oz\_rpart)[3, 1]

cp

Now, get out the pruning shears:

oz\_prune <- prune(oz\_rpart, cp = cp)

print(oz\_prune)

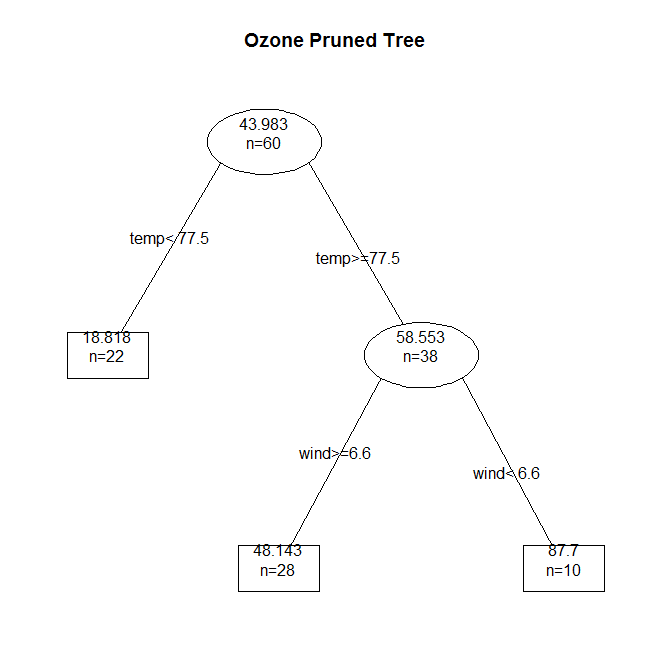
## n= 60   
##   
## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 60 60826.980 43.98333   
## 2) temp< 77.5 22 1773.273 18.81818 \*  
## 3) temp>=77.5 38 37055.390 58.55263   
## 6) wind>=6.6 28 15727.430 48.14286 \*  
## 7) wind< 6.6 10 9798.100 87.70000 \*

summary(oz\_prune)

## Call:  
## rpart(formula = model, data = oz[train, ], method = "anova",   
## control = rpart.control(minsplit = 10))  
## n= 60   
##   
## CP nsplit rel error xerror xstd  
## 1 0.3616539 0 1.0000000 1.0463162 0.2780705  
## 2 0.1895518 1 0.6383461 1.0222880 0.3451815  
## 3 0.1262068 2 0.4487943 0.9502084 0.3400496  
##   
## Variable importance  
## temp wind radiation   
## 55 34 11   
##   
## Node number 1: 60 observations, complexity param=0.3616539  
## mean=43.98333, MSE=1013.783   
## left son=2 (22 obs) right son=3 (38 obs)  
## Primary splits:  
## temp < 77.5 to the left, improve=0.3616539, (0 missing)  
## wind < 6.6 to the right, improve=0.3246449, (0 missing)  
## radiation < 152.5 to the left, improve=0.2002591, (0 missing)  
## Surrogate splits:  
## radiation < 36.5 to the left, agree=0.717, adj=0.227, (0 split)  
## wind < 11.75 to the right, agree=0.700, adj=0.182, (0 split)  
##   
## Node number 2: 22 observations  
## mean=18.81818, MSE=80.60331   
##   
## Node number 3: 38 observations, complexity param=0.1895518  
## mean=58.55263, MSE=975.142   
## left son=6 (28 obs) right son=7 (10 obs)  
## Primary splits:  
## wind < 6.6 to the right, improve=0.3111522, (0 missing)  
## temp < 87.5 to the left, improve=0.1967411, (0 missing)  
## radiation < 152.5 to the left, improve=0.1768124, (0 missing)  
## Surrogate splits:  
## temp < 91 to the left, agree=0.816, adj=0.3, (0 split)  
##   
## Node number 6: 28 observations  
## mean=48.14286, MSE=561.6939   
##   
## Node number 7: 10 observations  
## mean=87.7, MSE=979.81

Plot pruned tree:

post(oz\_prune, file = "", title = "Ozone Pruned Tree")



sum(residuals(oz\_prune)^2)

## [1] 27298.8

Predict with both trees:

pruned <- predict(oz\_prune, oz[-train, ], type = "vector")  
  
full <- predict(oz\_rpart, oz[-train, ], type = "vector")