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 penguins data set found in the package datasets in R. To access this data set, simply type:

library(palmerpenguins)

## Warning: package 'palmerpenguins' was built under R version 4.2.3

penguins <- as.data.frame(penguins)

To learn more about the data set:

`?`(penguins)

You will also use the “ozone.csv” dataset available on our shared Google Drive for regression trees.

oz <- read.csv("G:/Shared drives/MultivariateStatistics/Data/LabData/Lab9/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 two-thirds of the samples from the penguins data set. Use “set.seed” so we are all working with the same training set:

set.seed(51)  
# Create a vector whose length is one half of the number of rows in the  
# penguins data set called train that will indicate the rows randomly selected  
# for the training data.  
train <- sample(seq(1:dim(penguins)[1]), round(dim(penguins)[1] \* 0.66))

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(penguins$species[train])  
freq

##   
## Adelie Chinstrap Gentoo   
## 101 49 77

## Specifying the model

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

model <- species ~ .

**“Species ~ .” is the model and the “dot” stands for all of the explanatory 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)

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

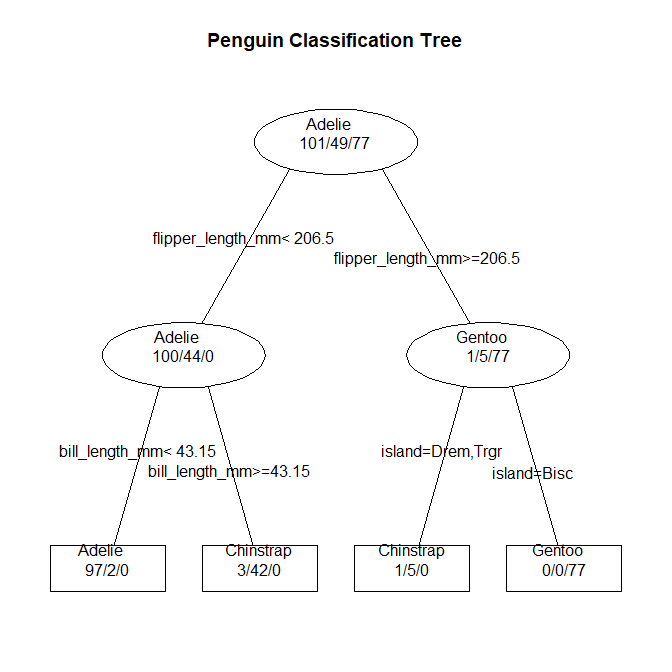
**Look up what rpart.control and it’s parameters do and play with them!**

## Plotting Cart Tree and viewing summary

You will plot your tree (made by the code I provided) using the function post.rpart:

`?`(post.rpart)

post(penguins\_rpart, file = "", title = "Penguin Classification Tree")



## Question 1: What do the three numbers separated by slashes refer to in each square or circle? (10 pts)

## Question 2: What are the splitting criteria (i.e., the questions asked as the data are sorted into groups by the tree)? (10 pts)

## Question 3: What island are Gentoos most associated with? (10 pts)

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

summary(penguins\_rpart)

There is a lot in this output. Below we will sort through some of this output and take a closer look at it.

## Cost-complexity pruning

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

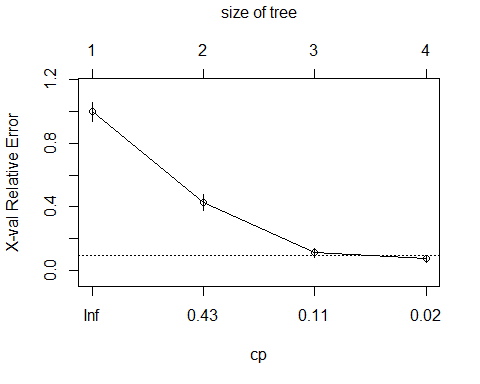
printcp(penguins\_rpart)

##   
## Classification tree:  
## rpart(formula = model, data = penguins[train, ], method = "class",   
## control = rpart.control(minsplit = 10))  
##   
## Variables actually used in tree construction:  
## [1] bill\_length\_mm flipper\_length\_mm island   
##   
## Root node error: 126/227 = 0.55507  
##   
## n= 227   
##   
## CP nsplit rel error xerror xstd  
## 1 0.603175 0 1.000000 1.000000 0.059424  
## 2 0.309524 1 0.396825 0.428571 0.050914  
## 3 0.039683 2 0.087302 0.111111 0.028765  
## 4 0.010000 3 0.047619 0.071429 0.023333

* 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(penguins\_rpart)



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

## Question 4: How big is the optimal tree? (10 pts)

## 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 row, column designations for other examples. Remember that matrix notation in R is [rows,columns]. The below code selects the cp values in the first column for the first three rows that correspond to the first three nodes of the dendrogram*.

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

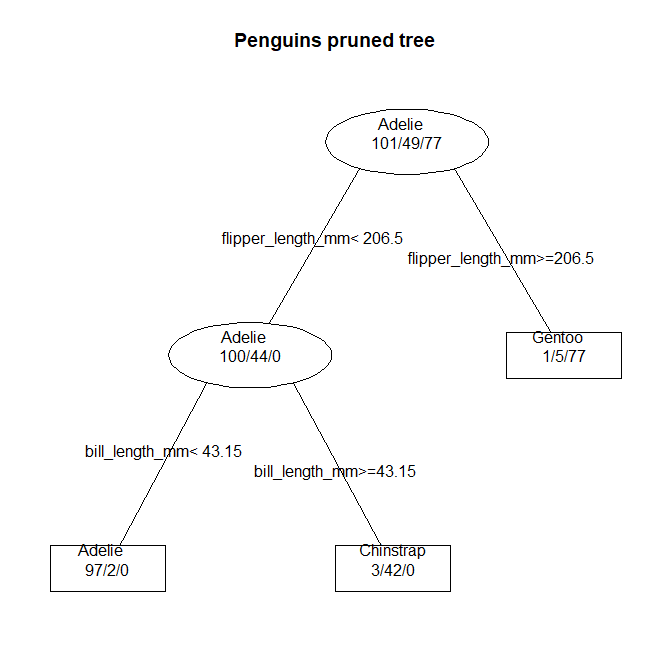
##   
## Classification tree:  
## rpart(formula = model, data = penguins[train, ], method = "class",   
## control = rpart.control(minsplit = 10))  
##   
## Variables actually used in tree construction:  
## [1] bill\_length\_mm flipper\_length\_mm island   
##   
## Root node error: 126/227 = 0.55507  
##   
## n= 227   
##   
## CP nsplit rel error xerror xstd  
## 1 0.603175 0 1.000000 1.000000 0.059424  
## 2 0.309524 1 0.396825 0.428571 0.050914  
## 3 0.039683 2 0.087302 0.111111 0.028765  
## 4 0.010000 3 0.047619 0.071429 0.023333

Now, get out the pruning shears:

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

## n= 227   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 227 126 Adelie (0.44493392 0.21585903 0.33920705)   
## 2) flipper\_length\_mm< 206.5 144 44 Adelie (0.69444444 0.30555556 0.00000000)   
## 4) bill\_length\_mm< 43.15 99 2 Adelie (0.97979798 0.02020202 0.00000000) \*  
## 5) bill\_length\_mm>=43.15 45 3 Chinstrap (0.06666667 0.93333333 0.00000000) \*  
## 3) flipper\_length\_mm>=206.5 83 6 Gentoo (0.01204819 0.06024096 0.92771084) \*

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

 ## Question 5: How does this tree differ from the original tree? (10 pts)

## Classification accuracy

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

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

##   
## Adelie Chinstrap Gentoo  
## Adelie 47 2 0  
## Chinstrap 4 17 1  
## Gentoo 0 0 46

Ct\_prune

##   
## Adelie Chinstrap Gentoo  
## Adelie 47 2 0  
## Chinstrap 3 17 1  
## Gentoo 1 0 46

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

## [1] 0.9401709

class\_prune

## [1] 0.9401709

## Question 6: Did pruning help much? (10 pts)

# Compare with DA

Pull up your code from last week and run a DA for the penguin data.

## Question 7: How does the CART classification rate for the training data compare to the DA classification rate? (10 pts)

# Regression Tree With a Continuous response

In the penguins example, you were predicting samples to species (classes). In this example, there is a continuous response variable. I don’t ask any questions for this example as it is mainly to show you how the coding works for a continuous response variable. Here we will use regression trees on the “ozone.csv” available on our shared Google Drive. 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 predictors (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))

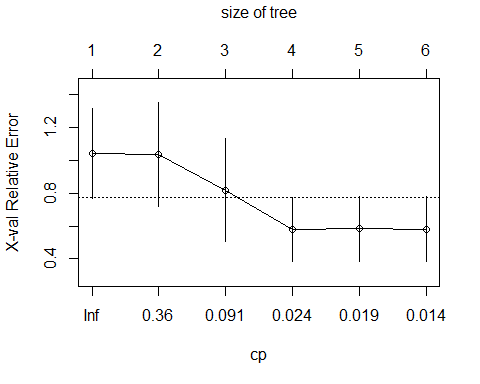
Here we denote the type of tree using method = “anova”, which is used for continues variables.

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: 70278/60 = 1171.3  
##   
## n= 60   
##   
## CP nsplit rel error xerror xstd  
## 1 0.469766 0 1.00000 1.04211 0.27157  
## 2 0.273137 1 0.53023 1.03626 0.31515  
## 3 0.030210 2 0.25710 0.81920 0.31166  
## 4 0.018762 3 0.22689 0.57904 0.19807  
## 5 0.018399 4 0.20813 0.58234 0.19794  
## 6 0.010000 5 0.18973 0.58043 0.19814

plotcp(oz\_rpart)



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

##   
## 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: 70278/60 = 1171.3  
##   
## n= 60   
##   
## CP nsplit rel error xerror xstd  
## 1 0.469766 0 1.00000 1.04211 0.27157  
## 2 0.273137 1 0.53023 1.03626 0.31515  
## 3 0.030210 2 0.25710 0.81920 0.31166  
## 4 0.018762 3 0.22689 0.57904 0.19807  
## 5 0.018399 4 0.20813 0.58234 0.19794  
## 6 0.010000 5 0.18973 0.58043 0.19814

cp

## [1] 0.03020962

Now, get out the pruning shears:

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

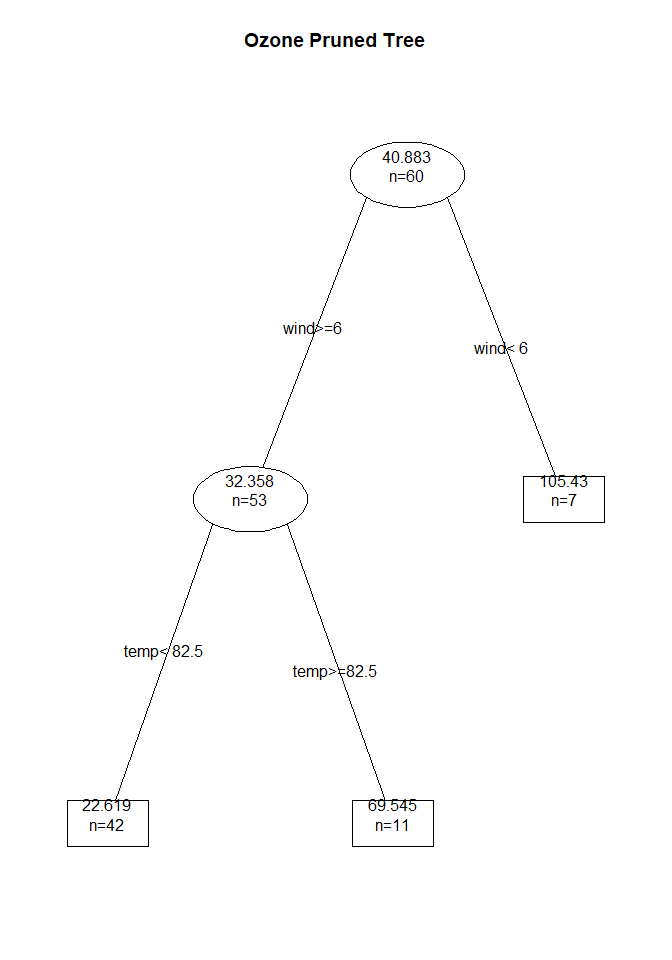
## n= 60   
##   
## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 60 70278.180 40.88333   
## 2) wind>=6 53 29304.190 32.35849   
## 4) temp< 82.5 42 7541.905 22.61905 \*  
## 5) temp>=82.5 11 2566.727 69.54545 \*  
## 3) wind< 6 7 7959.714 105.42860 \*

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 0.46976571 0 1.0000000 1.0421052 0.2715742  
## 2 0.27313678 1 0.5302343 1.0362577 0.3151466  
## 3 0.03020962 2 0.2570975 0.8192004 0.3116571  
##   
## Variable importance  
## wind temp   
## 58 42   
##   
## Node number 1: 60 observations, complexity param=0.4697657  
## mean=40.88333, MSE=1171.303   
## left son=2 (53 obs) right son=3 (7 obs)  
## Primary splits:  
## wind < 6 to the right, improve=0.4697657, (0 missing)  
## temp < 77.5 to the left, improve=0.4685858, (0 missing)  
## radiation < 161.5 to the left, improve=0.2526936, (0 missing)  
## Surrogate splits:  
## temp < 91 to the left, agree=0.9, adj=0.143, (0 split)  
##   
## Node number 2: 53 observations, complexity param=0.2731368  
## mean=32.35849, MSE=552.9092   
## left son=4 (42 obs) right son=5 (11 obs)  
## Primary splits:  
## temp < 82.5 to the left, improve=0.6550448, (0 missing)  
## radiation < 168 to the left, improve=0.2546642, (0 missing)  
## wind < 8.9 to the right, improve=0.1447507, (0 missing)  
##   
## Node number 3: 7 observations  
## mean=105.4286, MSE=1137.102   
##   
## Node number 4: 42 observations  
## mean=22.61905, MSE=179.5692   
##   
## Node number 5: 11 observations  
## mean=69.54545, MSE=233.3388

Plot pruned tree:

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



sum(residuals(oz\_prune)^2)

## [1] 18068.35

Predict with both trees:

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

## Question 8: Consider your own dataset, would a CART analysis be appropriate to perform on it? (30 pts)