Example stat 562

Sagar Kalauni

2023-11-07

library(tidyverse)

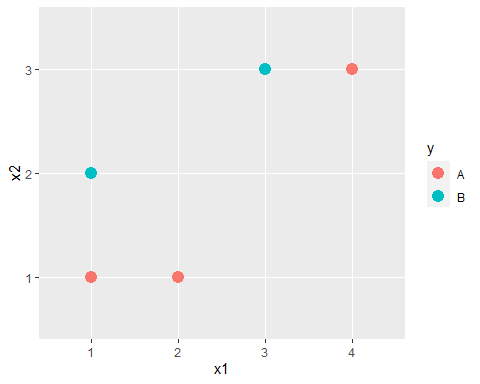
## Warning: package 'tidyverse' was built under R version 4.3.1

## Warning: package 'ggplot2' was built under R version 4.3.1

## Warning: package 'lubridate' was built under R version 4.3.1

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.2 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ ggplot2 3.4.2 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.1   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

y=c("A","B","A","B","A")  
x1=c(1,1,4,3,2)  
x2=c(1,2,3,3,1)  
data=as.data.frame(cbind(y,x1,x2))  
ggplot(data,aes(x=x1,y=x2,col=y))+geom\_point(size=4)



#originally: # – Originally we are checking how much impurity does the data set have in the very starting. #– Since this is the very small data set so we are doing it manually, but it is not possible to do like this #– manually in a huge data set.

g=2/5\*3/5\*2

#0.48

#choosing 1st node split: #– Start spliting through x-axis # – Now here we are trying different place to split the data set and according to each place we are tying to #– to calculate the impurity, and we get minimum impurity in g1.3 this split (means x1 less then3.5 and greater then 3.5)

g1.1=1/2\*1/2\*2\*(0.4)+1/3\*2/3\*2\*(0.6) #0.4667  
g1.2=1/3\*2/3\*2\*(0.6)+1/2\*1/2\*2\*(0.4) #0.4667  
g1.3=1/2\*1/2\*2\*(0.8)+0\*0.2 #0.4

# Now we will similarly split through y-axis

# check the impurity in the each split and choose the one having the minimum impurity, and mimimum impurity

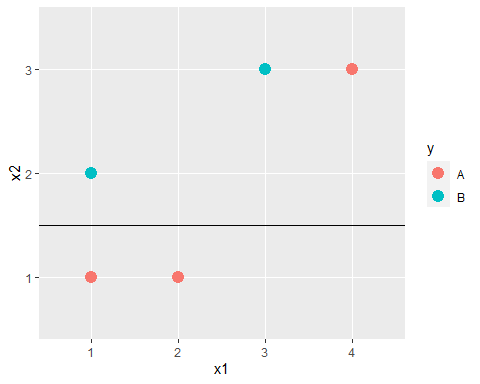
# will be for the one having the pure node.

# The minimum impurity is along g2.2 so we will split along that, means x2<1.5 or x2>1.5

g2.1=0\*(0.4)+1/3\*2/3\*2\*(0.6) #0.2667  
g2.2=1/3\*2/3\*2\*(0.6)+1/2\*1/2\*2\*(0.4) #0.4667

#pick x2< 1.5 v.s x2 > 1.5 as 1st split

ggplot(data,aes(x=x1,y=x2,col=y))+geom\_point(size=4)+geom\_hline(yintercept=1.5)

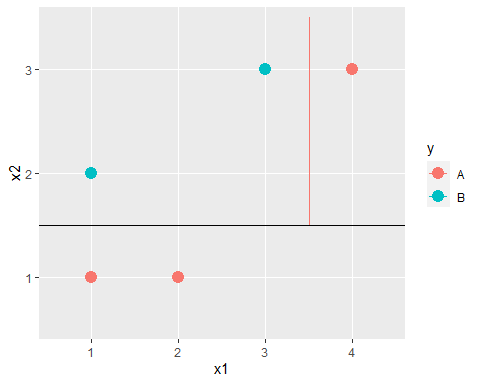


#choosing 2nd split

g1.1=0\*1/3+1/2\*1/2\*2\*2/3 #0.333  
g1.3=0   
g2.2=0\*1/3+1/2\*1/2\*2\*2/3 #0.333

#pick x1< 3.5 v.s x1 > 3.5 as 2nd split

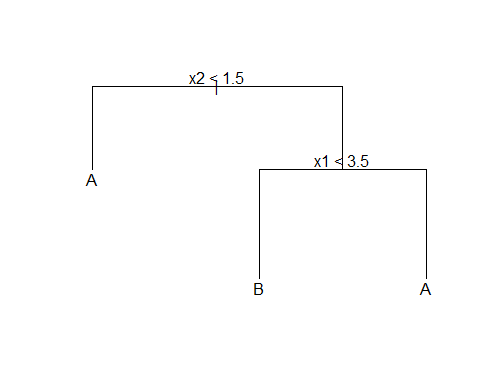
ggplot(data,aes(x=x1,y=x2,col=y))+geom\_point(size=4)+  
 geom\_hline(yintercept=1.5)+geom\_segment(aes(x = 3.5, y = 1.5, xend = 3.5, yend = 3.5))



library(tree)

## Warning: package 'tree' was built under R version 4.3.2

out=tree(as.factor(y)~.,data,control=tree.control(nobs=5,mincut = 0, minsize=0, mindev = 0))  
plot(out)  
text(out)



##Classification Tree Example, Default data

library(tree)  
library(ISLR2)

## Warning: package 'ISLR2' was built under R version 4.3.2

train=sample(1:10000,7000) # We take 7000 for training and 3000 for test  
test=Default[-train,]  
tree.d=tree(default~.,Default,split="gini",subset=train)

# – default is only one column of the table which need to be predicted

# – the name of the data set is Default which is in the ISLR2 library

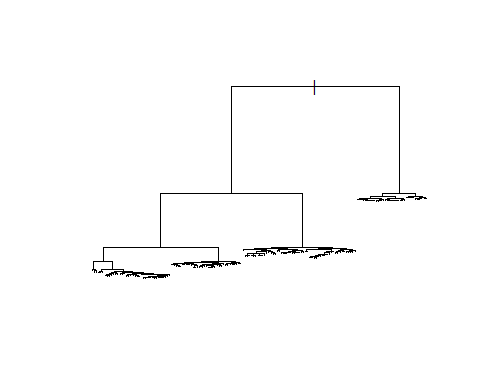
# –code: default is the variable I need to predict and I want to predict it crossponidng to all predictior (~.)

# – my data set name is Default and spliting criteria is gini and I will only make tree using tarining data set.

summary(tree.d)

##   
## Classification tree:  
## tree(formula = default ~ ., data = Default, subset = train, split = "gini")  
## Number of terminal nodes: 156   
## Residual mean deviance: 0.0955 = 653.6 / 6844   
## Misclassification error rate: 0.024 = 168 / 7000

plot(tree.d)



#predict class on test data

pred.d=predict(tree.d,test,type="class")  
table(pred.d,test$default)

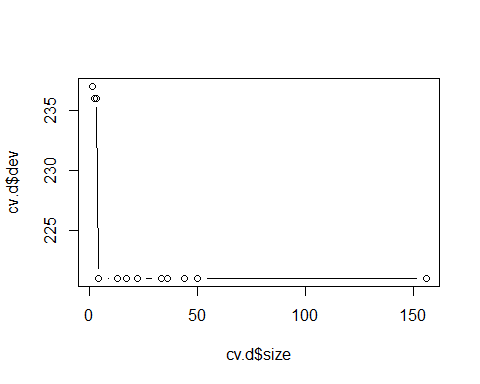
##   
## pred.d No Yes  
## No 2854 43  
## Yes 50 53

#pruning

cv.d=cv.tree(tree.d)

#or if you want to use misclassification rate for the CV instead of the default deviance,

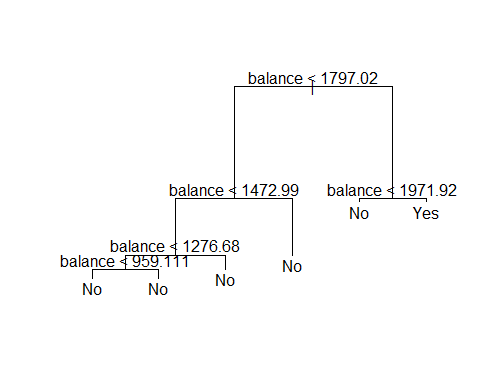
cv.d=cv.tree(tree.d,FUN = prune.misclass)  
plot(cv.d$size, cv.d$dev, type="b")



prune.d=prune.tree(tree.d,best=6)   
summary(prune.d)

##   
## Classification tree:  
## snip.tree(tree = tree.d, nodes = c(9L, 7L, 16L, 5L, 17L, 6L))  
## Variables actually used in tree construction:  
## [1] "balance"  
## Number of terminal nodes: 6   
## Residual mean deviance: 0.161 = 1126 / 6994   
## Misclassification error rate: 0.02886 = 202 / 7000

plot(prune.d)  
text(prune.d)



#predict class on test data

pred.d.prune=predict(prune.d,test,type="class")  
table(pred.d.prune,test$default)

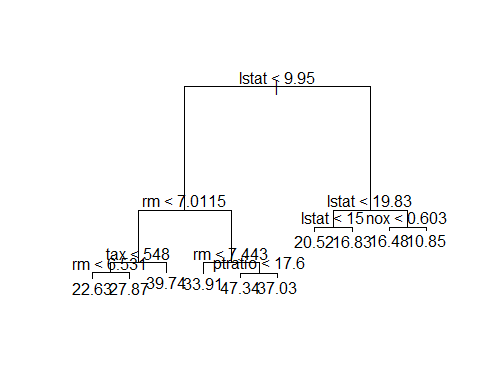
##   
## pred.d.prune No Yes  
## No 2893 58  
## Yes 11 38

#Regression Tree Example: Boston House Price

library(ISLR2)  
train=sample(1:506,350) # We take 350 for training and for test  
test=Boston[-train,]  
tree.b=tree(medv~.,Boston,subset=train)  
summary(tree.b)

##   
## Regression tree:  
## tree(formula = medv ~ ., data = Boston, subset = train)  
## Variables actually used in tree construction:  
## [1] "lstat" "rm" "tax" "ptratio" "nox"   
## Number of terminal nodes: 10   
## Residual mean deviance: 13.28 = 4515 / 340   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -16.04000 -2.04600 0.06297 0.00000 2.17300 16.09000

plot(tree.b)  
text(tree.b,pretty = 0)

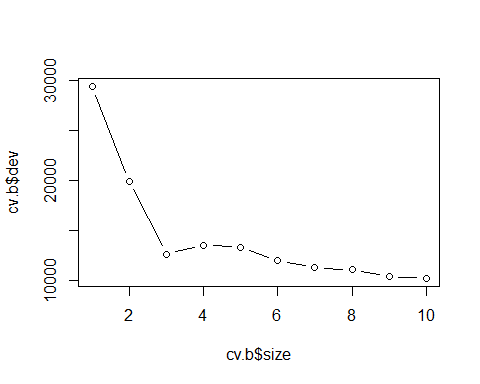


test.mse=mean((test$medv-predict(tree.b,test))^2)  
test.mse

## [1] 17.33287

#pruning if needed

cv.b=cv.tree(tree.b)  
plot(cv.b$size, cv.b$dev, type="b")



prune.b=prune.tree(tree.b,best=8)  
plot(prune.b)  
text(prune.b)

