officehrsHW7.R

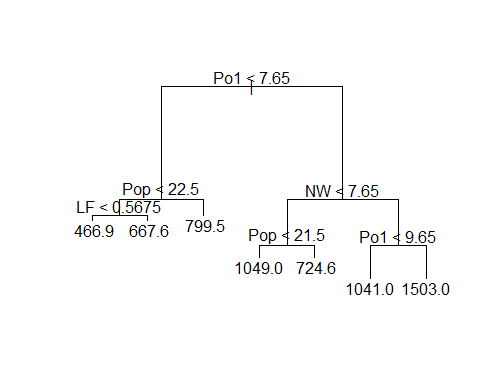
harit

2020-10-07

rm(list = ls())  
uscrime <- read.delim("uscrime.txt")  
   
#Fit a regression tree function to the crime data. Note that deviance   
# is a quality of fit statistic that is a generalization of the sum of  
#squared residuals. Also,check out the function documentation to learn  
#more about how the function makes splits in the tree.  
  
# Regression Tree Model  
library(tree)  
uscrime\_tree <- tree(Crime~., data = uscrime)  
summary(uscrime\_tree)

##   
## Regression tree:  
## tree(formula = Crime ~ ., data = uscrime)  
## Variables actually used in tree construction:  
## [1] "Po1" "Pop" "LF" "NW"   
## Number of terminal nodes: 7   
## Residual mean deviance: 47390 = 1896000 / 40   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -573.900 -98.300 -1.545 0.000 110.600 490.100

#visualize the regression tree, see how many points are in each leaf,  
#and see which leaf each point is in.  
  
plot(uscrime\_tree)  
text(uscrime\_tree)



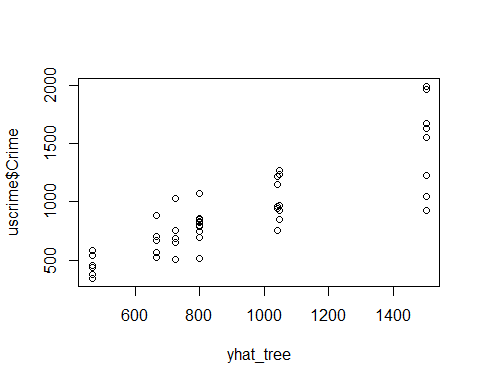
uscrime\_tree$frame

## var n dev yval splits.cutleft splits.cutright  
## 1 Po1 47 6880927.66 905.0851 <7.65 >7.65  
## 2 Pop 23 779243.48 669.6087 <22.5 >22.5  
## 4 LF 12 243811.00 550.5000 <0.5675 >0.5675  
## 8 <leaf> 7 48518.86 466.8571   
## 9 <leaf> 5 77757.20 667.6000   
## 5 <leaf> 11 179470.73 799.5455   
## 3 NW 24 3604162.50 1130.7500 <7.65 >7.65  
## 6 Pop 10 557574.90 886.9000 <21.5 >21.5  
## 12 <leaf> 5 146390.80 1049.2000   
## 13 <leaf> 5 147771.20 724.6000   
## 7 Po1 14 2027224.93 1304.9286 <9.65 >9.65  
## 14 <leaf> 6 170828.00 1041.0000   
## 15 <leaf> 8 1124984.88 1502.8750

uscrime\_tree$where

## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26   
## 6 13 4 13 9 10 12 13 6 5 13 6 6 5 6 12 4 13 10 13 6 4 12 9 5 13   
## 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47   
## 4 12 13 6 4 12 6 9 10 9 6 5 6 12 5 4 6 10 4 10 9

#Manually compute R2. Is this a good measure of the quality of fit?  
#Notice that we can only use averages of each leaf to make   
#predictions.  
  
yhat\_tree <- predict(uscrime\_tree)  
plot(yhat\_tree,uscrime$Crime)



#Examine training and cv deviance for different tree sizes.  
#what does this indicate about the quality of fit of our model?  
#Should we prune some branches?  
  
prune.tree(uscrime\_tree)$size

## [1] 7 6 5 4 3 2 1

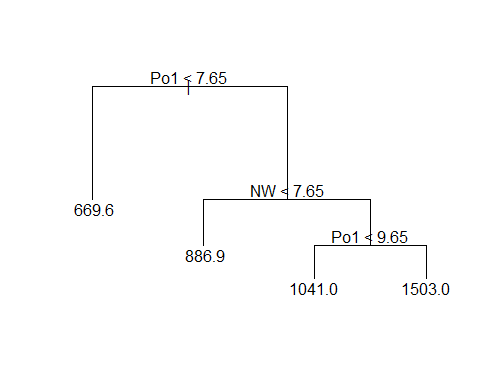
prune.tree(uscrime\_tree)$dev

## [1] 1895722 2013257 2276670 2632631 3364043 4383406 6880928

set.seed(42)  
cv.tree(object = uscrime\_tree, FUN = prune.tree)

## $size  
## [1] 7 6 5 4 3 2 1  
##   
## $dev  
## [1] 7986688 7986688 7885099 7920849 7707536 7035494 8536032  
##   
## $k  
## [1] -Inf 117534.9 263412.9 355961.8 731412.1 1019362.7 2497521.7  
##   
## $method  
## [1] "deviance"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

# we can do cross validation.  
  
#Example of manaully pruning a tree in which we choose to onlu have 4 leaves  
  
uscrime\_tree\_prune4 <- prune.tree(uscrime\_tree,best = 4)  
plot(uscrime\_tree\_prune4)  
text(uscrime\_tree\_prune4)



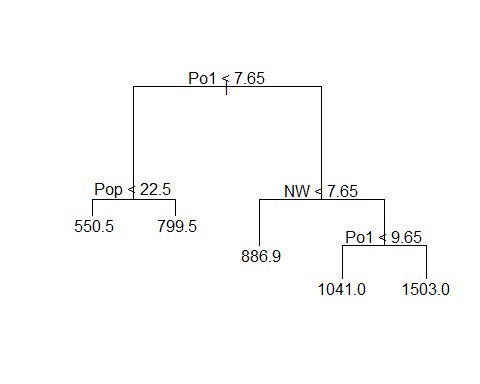
summary(uscrime\_tree\_prune4)

##   
## Regression tree:  
## snip.tree(tree = uscrime\_tree, nodes = c(6L, 2L))  
## Variables actually used in tree construction:  
## [1] "Po1" "NW"   
## Number of terminal nodes: 4   
## Residual mean deviance: 61220 = 2633000 / 43   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -573.90 -152.60 35.39 0.00 158.90 490.10

crimeTree4Predict <- predict(uscrime\_tree\_prune4, data = uscrime[,1:15])  
RSSofTree4 <- sum((crimeTree4Predict - uscrime[,16])^2)  
TSS <- sum((uscrime[,16] - mean(uscrime[,16]))^2)  
R2ofTree4 <- 1 - RSSofTree4/TSS  
R2ofTree4

## [1] 0.6174017

uscrime\_tree\_prune5 <- prune.tree(uscrime\_tree,best = 5)  
plot(uscrime\_tree\_prune5)  
text(uscrime\_tree\_prune5)



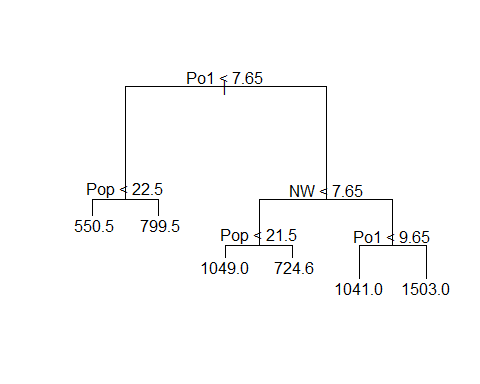
summary(uscrime\_tree\_prune5)

##   
## Regression tree:  
## snip.tree(tree = uscrime\_tree, nodes = c(4L, 6L))  
## Variables actually used in tree construction:  
## [1] "Po1" "Pop" "NW"   
## Number of terminal nodes: 5   
## Residual mean deviance: 54210 = 2277000 / 42   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -573.9 -107.5 15.5 0.0 122.8 490.1

crimeTree5Predict <- predict(uscrime\_tree\_prune5, data = uscrime[,1:15])  
RSSofTree5 <- sum((crimeTree5Predict - uscrime[,16])^2)  
R2ofTree5 <- 1 - RSSofTree5 /TSS  
R2ofTree5

## [1] 0.6691333

uscrime\_tree\_prune6 <- prune.tree(uscrime\_tree,best = 6)  
plot(uscrime\_tree\_prune6)  
text(uscrime\_tree\_prune6)



summary(uscrime\_tree\_prune6)

##   
## Regression tree:  
## snip.tree(tree = uscrime\_tree, nodes = 4L)  
## Variables actually used in tree construction:  
## [1] "Po1" "Pop" "NW"   
## Number of terminal nodes: 6   
## Residual mean deviance: 49100 = 2013000 / 41   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -573.900 -99.520 -1.545 0.000 122.800 490.100

crimeTree6Predict <- predict(uscrime\_tree\_prune6, data = uscrime[,1:15])  
RSSofTree6 <- sum((crimeTree6Predict - uscrime[,16])^2)  
R2ofTree6 <- 1 - RSSofTree6 /TSS  
R2ofTree6

## [1] 0.7074149

#since you are being asked to build a regression tree, you should  
#at least attempt to build a regression model on one of the   
#leaves instead of just taking the average among the points in   
#the leaf for predictions. Will there be overfitting if we try to   
# build regression models for each of the original 7 leaves?  
  
#Might want to consider looking at different numbers of branches   
# and/or different splitting criteria (look at the split argument).  
#Can use the rpart() function from the rpart library very similarly.  
  
#### Q10.1b###  
# Random Forest  
  
#rm(list = ls())  
#uscrime <- read.table("uscrime.txt", stringsAsFactors = FALSE, header = TRUE)  
  
#Grow the random trees and set the number of predictors that you want  
#to consider at each split of the tree(numpred). A good recommendation   
#for numpred is 1+log(n) or n/3 where n is the number of predictors.  
  
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

set.seed(42)  
num\_pred <- 4  
uscrime\_rf <- randomForest(Crime~.,data = uscrime,mtry = num\_pred,importance = TRUE , ntree = 500)  
uscrime\_rf

##   
## Call:  
## randomForest(formula = Crime ~ ., data = uscrime, mtry = num\_pred, importance = TRUE, ntree = 500)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 4  
##   
## Mean of squared residuals: 83636.77  
## % Var explained: 42.87

crime\_rf\_predict <- predict(uscrime\_rf, data=uscrime[,-16])  
RSS <- sum((crime\_rf\_predict - uscrime[,16])^2)  
R2 <- 1 - RSS/TSS  
R2

## [1] 0.4287212

num\_pred5 <- 5  
uscrime\_rf5 <- randomForest(Crime~.,data = uscrime,mtry = num\_pred5,importance = TRUE , ntree = 500)  
uscrime\_rf5

##   
## Call:  
## randomForest(formula = Crime ~ ., data = uscrime, mtry = num\_pred5, importance = TRUE, ntree = 500)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 5  
##   
## Mean of squared residuals: 88034.11  
## % Var explained: 39.87

crime\_rf\_predict5 <- predict(uscrime\_rf5, data=uscrime[,-16])  
RSS5 <- sum((crime\_rf\_predict5 - uscrime[,16])^2)  
R2\_pre5 <- 1 - RSS5/TSS  
R2\_pre5

## [1] 0.3986853

#consider computing CV R^2 of this model. If you compare it to the   
#training R^2, what can we conclude about this model?  
  
  
#remember that random forest can give you a model with better   
#of quality of fit, but it is a black model. we can use the importance  
#function to see what are the most important factors for the   
#prediction are . %IncMSE is the amount that the MSE of predictions increases  
# if the variable is randomly chosen instead of using its actual value.  
#IncNodePurity measures how much splitting on it improves the similarity   
# of the data points in each leaf.  
  
importance(uscrime\_rf)

## %IncMSE IncNodePurity  
## M 2.4984854 200566.40  
## So 1.3802135 33881.59  
## Ed 4.8378328 198601.72  
## Po1 9.7354718 1076933.25  
## Po2 10.6715396 1268930.03  
## LF 0.6449124 311872.13  
## M.F 1.1555044 239897.22  
## Pop 2.1893155 379760.15  
## NW 8.7310286 542658.76  
## U1 2.6422460 145760.60  
## U2 1.6754487 190587.49  
## Wealth 3.2683848 626353.30  
## Ineq 2.1162044 238557.90  
## Prob 8.6884908 812217.29  
## Time 1.6622726 202467.06

##########Q10.3########  
set.seed(10)  
  
rm(list = ls())  
germancredit <- read.table("germancredit.txt",header = FALSE)  
str(germancredit)

## 'data.frame': 1000 obs. of 21 variables:  
## $ V1 : chr "A11" "A12" "A14" "A11" ...  
## $ V2 : int 6 48 12 42 24 36 24 36 12 30 ...  
## $ V3 : chr "A34" "A32" "A34" "A32" ...  
## $ V4 : chr "A43" "A43" "A46" "A42" ...  
## $ V5 : int 1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...  
## $ V6 : chr "A65" "A61" "A61" "A61" ...  
## $ V7 : chr "A75" "A73" "A74" "A74" ...  
## $ V8 : int 4 2 2 2 3 2 3 2 2 4 ...  
## $ V9 : chr "A93" "A92" "A93" "A93" ...  
## $ V10: chr "A101" "A101" "A101" "A103" ...  
## $ V11: int 4 2 3 4 4 4 4 2 4 2 ...  
## $ V12: chr "A121" "A121" "A121" "A122" ...  
## $ V13: int 67 22 49 45 53 35 53 35 61 28 ...  
## $ V14: chr "A143" "A143" "A143" "A143" ...  
## $ V15: chr "A152" "A152" "A152" "A153" ...  
## $ V16: int 2 1 1 1 2 1 1 1 1 2 ...  
## $ V17: chr "A173" "A173" "A172" "A173" ...  
## $ V18: int 1 1 2 2 2 2 1 1 1 1 ...  
## $ V19: chr "A192" "A191" "A191" "A191" ...  
## $ V20: chr "A201" "A201" "A201" "A201" ...  
## $ V21: int 1 2 1 1 2 1 1 1 1 2 ...

# Notice that we have many categorical predictors.  
  
#Make the response variable binary in terms of 0 and 1.  
germancredit$V21[germancredit$V21==1] <- 0  
germancredit$V21[germancredit$V21==2] <- 1  
  
head(germancredit)

## V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18  
## 1 A11 6 A34 A43 1169 A65 A75 4 A93 A101 4 A121 67 A143 A152 2 A173 1  
## 2 A12 48 A32 A43 5951 A61 A73 2 A92 A101 2 A121 22 A143 A152 1 A173 1  
## 3 A14 12 A34 A46 2096 A61 A74 2 A93 A101 3 A121 49 A143 A152 1 A172 2  
## 4 A11 42 A32 A42 7882 A61 A74 2 A93 A103 4 A122 45 A143 A153 1 A173 2  
## 5 A11 24 A33 A40 4870 A61 A73 3 A93 A101 4 A124 53 A143 A153 2 A173 2  
## 6 A14 36 A32 A46 9055 A65 A73 2 A93 A101 4 A124 35 A143 A153 1 A172 2  
## V19 V20 V21  
## 1 A192 A201 0  
## 2 A191 A201 1  
## 3 A191 A201 0  
## 4 A191 A201 0  
## 5 A191 A201 1  
## 6 A192 A201 0

#split the data into training and testing sets.  
germancredit\_train <- germancredit[1:800,]  
germancredit\_test <- germancredit[801:1000,]  
  
#create a logistic regression model  
germancredit\_model = glm(V21~., family=binomial(link = "logit"),  
 data=germancredit\_train)  
  
summary(germancredit\_model)

##   
## Call:  
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = germancredit\_train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.7373 -0.6979 -0.3604 0.6663 2.5591   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 3.978e-01 1.249e+00 0.318 0.750139   
## V1A12 -2.701e-01 2.437e-01 -1.108 0.267784   
## V1A13 -9.306e-01 4.018e-01 -2.316 0.020567 \*   
## V1A14 -1.737e+00 2.686e-01 -6.465 1.02e-10 \*\*\*  
## V2 2.893e-02 1.042e-02 2.777 0.005479 \*\*   
## V3A31 2.255e-01 6.289e-01 0.359 0.719936   
## V3A32 -7.639e-01 4.753e-01 -1.607 0.108022   
## V3A33 -9.172e-01 5.233e-01 -1.753 0.079627 .   
## V3A34 -1.487e+00 4.907e-01 -3.031 0.002440 \*\*   
## V4A41 -1.832e+00 4.425e-01 -4.141 3.46e-05 \*\*\*  
## V4A410 -1.413e+00 8.263e-01 -1.710 0.087326 .   
## V4A42 -9.368e-01 2.990e-01 -3.134 0.001727 \*\*   
## V4A43 -9.044e-01 2.799e-01 -3.230 0.001236 \*\*   
## V4A44 -8.312e-01 8.946e-01 -0.929 0.352807   
## V4A45 -3.222e-01 6.092e-01 -0.529 0.596843   
## V4A46 1.688e-02 4.255e-01 0.040 0.968354   
## V4A48 -2.213e+00 1.219e+00 -1.816 0.069365 .   
## V4A49 -8.368e-01 3.850e-01 -2.173 0.029760 \*   
## V5 1.138e-04 5.166e-05 2.202 0.027682 \*   
## V6A62 -3.991e-01 3.182e-01 -1.254 0.209771   
## V6A63 -4.615e-01 4.762e-01 -0.969 0.332404   
## V6A64 -1.222e+00 5.473e-01 -2.232 0.025592 \*   
## V6A65 -7.093e-01 2.929e-01 -2.421 0.015462 \*   
## V7A72 -2.017e-01 4.948e-01 -0.408 0.683485   
## V7A73 -3.028e-01 4.706e-01 -0.643 0.519975   
## V7A74 -1.105e+00 5.113e-01 -2.162 0.030623 \*   
## V7A75 -4.092e-01 4.712e-01 -0.869 0.385102   
## V8 3.602e-01 9.933e-02 3.626 0.000287 \*\*\*  
## V9A92 -4.434e-01 4.300e-01 -1.031 0.302374   
## V9A93 -1.230e+00 4.245e-01 -2.897 0.003769 \*\*   
## V9A94 -4.630e-01 5.119e-01 -0.905 0.365705   
## V10A102 7.521e-01 4.771e-01 1.576 0.114917   
## V10A103 -9.329e-01 4.830e-01 -1.931 0.053423 .   
## V11 3.282e-03 9.850e-02 0.033 0.973420   
## V12A122 4.101e-01 2.897e-01 1.415 0.156969   
## V12A123 1.536e-01 2.649e-01 0.580 0.562115   
## V12A124 7.122e-01 4.714e-01 1.511 0.130827   
## V13 -1.868e-02 1.055e-02 -1.770 0.076682 .   
## V14A142 -1.442e-02 4.733e-01 -0.030 0.975695   
## V14A143 -4.354e-01 2.724e-01 -1.599 0.109919   
## V15A152 -3.967e-01 2.739e-01 -1.448 0.147576   
## V15A153 -5.576e-01 5.303e-01 -1.051 0.293071   
## V16 3.297e-01 2.124e-01 1.552 0.120602   
## V17A172 5.151e-01 7.807e-01 0.660 0.509351   
## V17A173 5.655e-01 7.507e-01 0.753 0.451267   
## V17A174 8.202e-01 7.597e-01 1.080 0.280307   
## V18 5.065e-01 2.854e-01 1.775 0.075972 .   
## V19A192 -3.739e-01 2.323e-01 -1.610 0.107489   
## V20A202 -1.498e+00 8.079e-01 -1.854 0.063779 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 975.68 on 799 degrees of freedom  
## Residual deviance: 705.07 on 751 degrees of freedom  
## AIC: 803.07  
##   
## Number of Fisher Scoring iterations: 5

#consider doing some type of variable selection even though this has not  
#been covered in the lectures yet. Also, notice how glm() implicitly  
#creates dummy binary variables for each of the categorical variables.  
#This is the correct way to do regression with categorical variables.  
#however, if you want to do variable selection with these many dummy  
#variables, you must re-define your categorical variables either manually  
# or with an R function.  
  
yhat<-predict(germancredit\_model,germancredit\_test[,-21],type= "response")  
table(germancredit\_test$V21, round(yhat))

##   
## 0 1  
## 0 115 24  
## 1 29 32

#Important to use type = "response" here because without this  
# we are given predictions of log-odds in the default case.  
  
#"round" your the yhat to get binary predictions from which   
#you can compute an accuracy (classification rate). You may want  
#to try out differnt thresholds for rounding. You can also use AUC to   
#estimate the quality of fit.  
  
library(pROC)

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

roc(germancredit\_test$V21,round(yhat))

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

##   
## Call:  
## roc.default(response = germancredit\_test$V21, predictor = round(yhat))  
##   
## Data: round(yhat) in 139 controls (germancredit\_test$V21 0) < 61 cases (germancredit\_test$V21 1).  
## Area under the curve: 0.676

#Look at different threshold probability values and then compute the   
#cost that corresponds to each threshold.  
  
thresh <- 0.8  
yhat\_thresh <- as.integer(yhat > thresh)  
conf\_matrix <- as.matrix(table(yhat\_thresh,germancredit\_test$V21))  
conf\_matrix

##   
## yhat\_thresh 0 1  
## 0 134 53  
## 1 5 8

accuracy <- (conf\_matrix[1,1]+conf\_matrix[2,2])/(conf\_matrix[1,1]+conf\_matrix[1,2]+conf\_matrix[2,1]+conf\_matrix[2,2])  
accuracy

## [1] 0.71

specificity <- (conf\_matrix[1,1])/(conf\_matrix[1,1]+conf\_matrix[2,1])  
specificity

## [1] 0.9640288

thresh <- 0.7  
yhat\_thresh <- as.integer(yhat > thresh)  
conf\_matrix <- as.matrix(table(yhat\_thresh,germancredit\_test$V21))  
conf\_matrix

##   
## yhat\_thresh 0 1  
## 0 132 43  
## 1 7 18

accuracy <- (conf\_matrix[1,1]+conf\_matrix[2,2])/(conf\_matrix[1,1]+conf\_matrix[1,2]+conf\_matrix[2,1]+conf\_matrix[2,2])  
accuracy

## [1] 0.75

specificity <- (conf\_matrix[1,1])/(conf\_matrix[1,1]+conf\_matrix[2,1])  
specificity

## [1] 0.9496403

#(Cost computation from the confusion matrix goes here)