Chapter 8 HW

Ulices Gonzalez

May 15th, 2025

## Conceptual Questions

### 3. This question relates to the plots in Figure 8.14.

#### (a) Sketch the tree corresponding to the partition of the predictor space illustrated in the left-hand panel of Figure 8.14. The numbers inside the boxes indicate the mean of Y within each region.

#### (b) Create a diagram similar to the left-hand panel of Figure 8.14, using the tree illustrated in the right-hand panel of the same figure. You should divide up the predictor space into the correct regions, and indicate the mean for each region.

## Applied Questions

### 8. In the lab, a classifcation tree was applied to the Carseats data set after converting Sales into a qualitative response variable. Now we will seek to predict Sales using regression trees and related approaches, treating the response as a quantitative variable.

#### (a) Split the data set into a training set and a test set.

library(tree)  
library(randomForest)

## randomForest 4.7-1.2

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

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

## The following object is masked from 'package:dplyr':  
##   
## combine

carseats <- read.csv("DataSets/Carseats.csv")  
set.seed(1)  
carSeatsTrain = sample(1:nrow(carseats), nrow(carseats) / 2)  
carTrainSet = carseats[carSeatsTrain, ]  
carTestSet = carseats[-carSeatsTrain,]

#### (b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

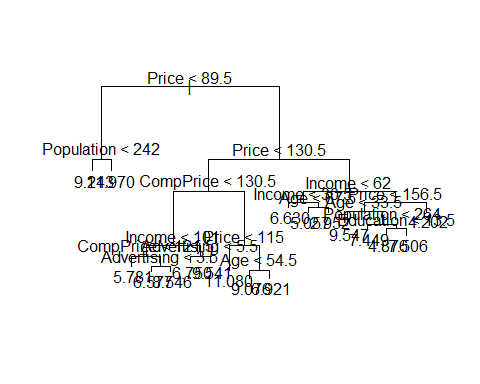
carTree = tree(Sales~.,data = carTrainSet)

## Warning in tree(Sales ~ ., data = carTrainSet): NAs introduced by coercion

summary(carTree)

##   
## Regression tree:  
## tree(formula = Sales ~ ., data = carTrainSet)  
## Variables actually used in tree construction:  
## [1] "Price" "Population" "CompPrice" "Income" "Advertising"  
## [6] "Age" "Education"   
## Number of terminal nodes: 18   
## Residual mean deviance: 2.726 = 496.2 / 182   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -4.361000 -1.092000 0.006545 0.000000 1.125000 4.989000

plot(carTree)  
text(carTree ,pretty =0)



mean((predict(carTree,newdata = carTestSet) - carTestSet$Sales)^2)

## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion

## [1] 7.871697

The MSE is roughly 7.871, due to the output of the tree it is difficult to analyze but it is clear that price leads to various different branches being created.

#### (c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

altCarSeats = cv.tree(carTree)

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

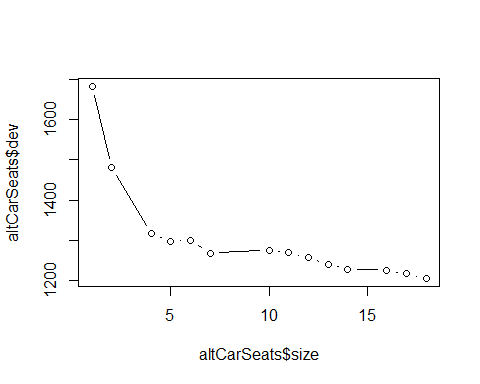
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

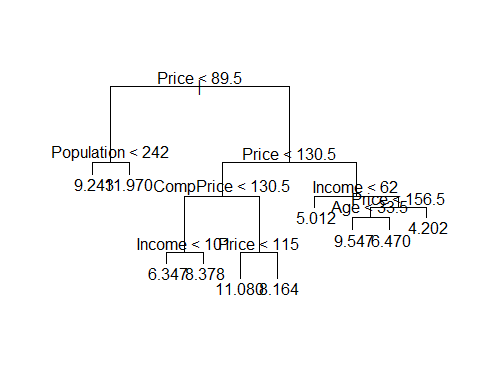
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by  
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

plot(altCarSeats$size, altCarSeats$dev, type = "b")



carTreePruned = prune.tree(carTree, best = 8)  
plot(carTreePruned)  
text(carTreePruned,pretty=0)



mean((predict(carTreePruned,newdata = carTestSet) - carTestSet$Sales)^2)

## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by coercion

## [1] 7.607696

The MSE is roughly after pruning show a slight decrease which is an improvement overall but very minimal.

#### (d) Use the bagging approach in order to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important.

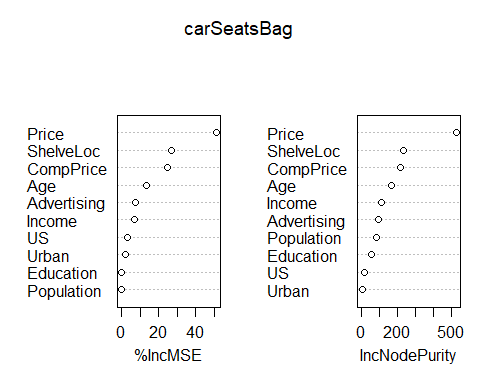
carSeatsBag = randomForest(Sales~.,data=carTrainSet,mtry = 10, importance = TRUE)  
mean((predict(carSeatsBag,newdata = carTestSet) - carTestSet$Sales)^2)

## [1] 3.090809

importance(carSeatsBag)

## %IncMSE IncNodePurity  
## CompPrice 24.7575854 215.43043  
## Income 6.8191334 109.95161  
## Advertising 7.4901742 97.03162  
## Population -0.5077432 85.77853  
## Price 51.6002288 527.70536  
## ShelveLoc 26.8710880 235.46208  
## Age 13.5783971 166.99652  
## Education -0.1412874 55.14556  
## Urban 1.9902069 10.53419  
## US 2.8038741 17.36050

varImpPlot(carSeatsBag)



The bagging approach yielded a much better MSE with a significant drop from 7.6 to 3.09 and my previous assumption of price being a big influence being confirmed with it being shown to have high importance followed by ShelveLoc and the CompPrice.

#### (e) Use random forests to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important. Describe the efect of m, the number of variables considered at each split, on the error rate obtained

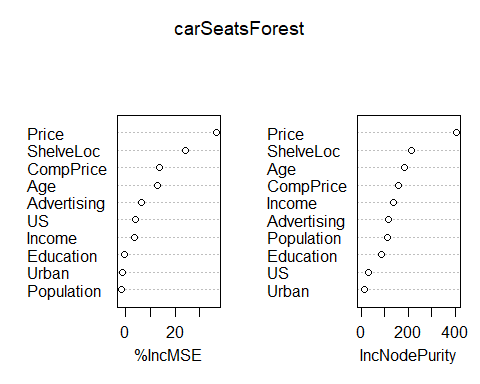
carSeatsForest = randomForest(Sales~.,data=carTrainSet,mtry = 3, importance = TRUE)  
mean((predict(carSeatsForest,newdata = carTestSet) - carTestSet$Sales)^2)

## [1] 3.309537

importance(carSeatsForest)

## %IncMSE IncNodePurity  
## CompPrice 13.7655035 159.33251  
## Income 3.6678665 136.95752  
## Advertising 6.2055659 117.30502  
## Population -2.0136414 114.04367  
## Price 37.0896583 406.89139  
## ShelveLoc 24.3620962 215.89157  
## Age 12.9539510 184.16209  
## Education -0.7762981 86.64465  
## Urban -1.3538311 16.05842  
## US 3.8410277 31.79657

varImpPlot(carSeatsForest)



The test MSE is slightly higher than the bagging approach but not by much with only a slight increase being shown as well as the importance remaining relatively the same.

### 10. We now use boosting to predict Salary in the Hitters data set.

#### (a) Remove the observations for whom the salary information is unknown, and then log-transform the salaries.

library(gbm)

## Loaded gbm 2.2.2

## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.com/gbm-developers/gbm3

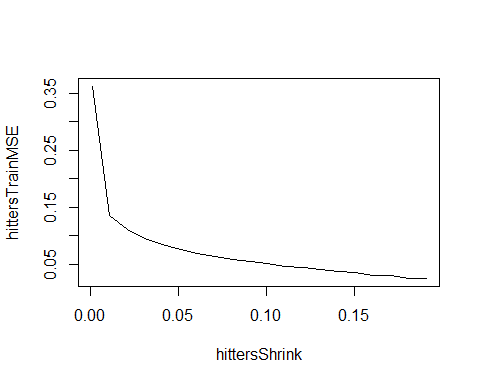
hitters <- read.csv("DataSets/Hitters.csv")  
hitters = na.omit(hitters)  
hitters$Salary = log(hitters$Salary)

#### (b) Create a training set consisting of the frst 200 observations, and a test set consisting of the remaining observations.

hittersTrain = 1:200  
hittersTrainSet = hitters[hittersTrain, ]  
hittersTestSet = hitters[-hittersTrain,]  
  
hittersTrainSet$League = as.factor(hittersTrainSet$League)  
hittersTrainSet$Division = as.factor(hittersTrainSet$Division)  
hittersTestSet$League = as.factor(hittersTestSet$League)  
hittersTestSet$Division = as.factor(hittersTestSet$Division)  
hittersTestSet$NewLeague = as.factor(hittersTestSet$NewLeague)   
hittersTrainSet$NewLeague = as.factor(hittersTrainSet$NewLeague)

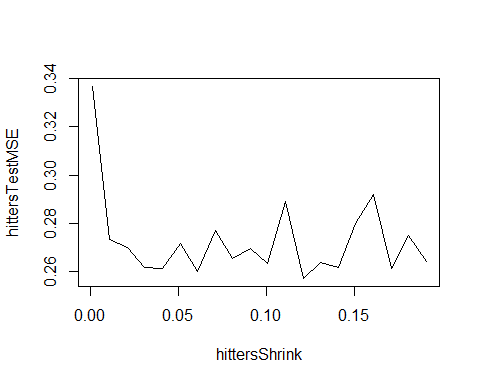
#### (c) Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter λ. Produce a plot with diferent shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

hittersShrink = seq(0.001, 0.2, by = 0.01)  
hittersTrainMSE = numeric(length(hittersShrink))  
  
for (i in seq\_along(hittersShrink))   
{  
 hittersBoost = gbm(Salary ~ ., data = hittersTrainSet, n.trees = 1000, shrinkage = hittersShrink[i], distribution = "gaussian", verbose = FALSE)  
 hittersPredTrain = predict(hittersBoost, newdata = hittersTrainSet, n.trees = 1000)  
 hittersTrainMSE[i] = mean((hittersPredTrain - hittersTrainSet$Salary)^2)  
}  
  
plot(hittersShrink, hittersTrainMSE, type = "l")



#### (d) Produce a plot with diferent shrinkage values on the x-axis and the corresponding test set MSE on the y-axis

hittersTestMSE = numeric(length(hittersShrink))  
  
for (i in seq\_along(hittersShrink))   
{  
 hittersBoost = gbm(Salary ~ ., data = hittersTrainSet, n.trees = 1000, shrinkage = hittersShrink[i], distribution = "gaussian", verbose = FALSE)  
 hittersTestPred = predict(hittersBoost, newdata = hittersTestSet, n.trees = 1000)  
 hittersTestMSE[i] = mean((hittersTestPred - hittersTestSet$Salary)^2)  
}  
  
plot(hittersShrink, hittersTestMSE, type = "l")



#### (e) Compare the test MSE of boosting to the test MSE that results from applying two of the regression approaches seen in Chapters 3 and 6.

hittersLinear = lm(Salary ~ ., data = hittersTrainSet)  
hittersLinearPred = predict(hittersLinear, newdata = hittersTestSet)  
mean((hittersLinearPred - hittersTestSet$Salary)^2)

## [1] 0.4917959

hittersColumns = sapply(hittersTrainSet, is.numeric)  
hittersColNames = names(hittersTrainSet)[hittersColumns & names(hittersTrainSet) != "Salary"]  
hittersFormula = as.formula(paste("Salary ~ . +", paste(paste0("I(", hittersColNames, "^2)"), collapse = " + ")))  
  
hittersPolynomial = lm(hittersFormula, data = hittersTrainSet)  
hittersPolynomialPred = predict(hittersPolynomial, newdata = hittersTestSet)  
mean((hittersPolynomialPred - hittersTestSet$Salary)^2)

## [1] 0.3068952

In my comparison I used a multiple linear approach as well as a polynomial approach with the two being successful but between the two the polynomial performed better with it also having a lower MSE however of the two the multiple linear was closest to the original MSE.

#### (f) Which variables appear to be the most important predictors in the boosted model?

#### (g) Now apply bagging to the training set. What is the test set MSE for this approach?

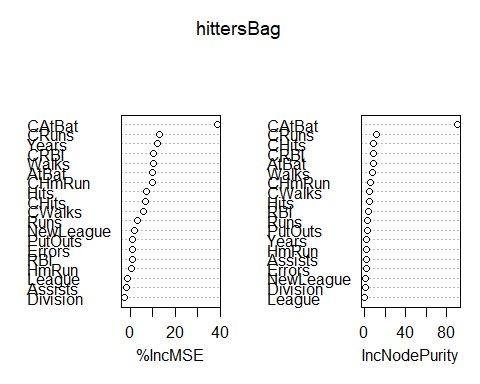
hittersTrainSet$League = as.factor(hittersTrainSet$League)  
hittersTrainSet$Division = as.factor(hittersTrainSet$Division)  
hittersTestSet$League = as.factor(hittersTestSet$League)  
hittersTestSet$Division = as.factor(hittersTestSet$Division)  
  
hittersBag = randomForest(Salary ~ ., data = hittersTrainSet, mtry = ncol(hittersTrainSet) - 1, importance = TRUE)  
hittersBagPred = predict(hittersBag, newdata = hittersTestSet)  
mean((hittersBagPred - hittersTestSet$Salary)^2)

## [1] 0.2341505

importance(hittersBag)

## %IncMSE IncNodePurity  
## AtBat 9.7710550 8.0746521  
## Hits 7.2226820 4.2689653  
## HmRun 0.6751159 1.4739921  
## Runs 3.3350497 2.9480589  
## RBI 0.9657612 3.1163014  
## Walks 10.1286148 7.4741245  
## Years 12.0723480 1.8746970  
## CAtBat 38.6447217 90.2170687  
## CHits 6.8894065 8.8843409  
## CHmRun 9.6762154 5.0275865  
## CRuns 12.7996487 11.6931864  
## CRBI 10.1287761 8.4971919  
## CWalks 5.8639563 4.9265005  
## League -1.3438470 0.0835882  
## Division -2.4667203 0.1763950  
## PutOuts 1.1063708 2.6868971  
## Assists -1.6146041 1.3397763  
## Errors 1.0089679 1.2480821  
## NewLeague 1.9408373 0.1884629

varImpPlot(hittersBag)



The test MSE for the bagging model is roughly 0.2341