R Notebook

***8. In the lab, a classification 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.***

1. Split the data set into a training set and a test set.

library(ISLR)  
attach(Carseats)  
set.seed(1)  
  
train = sample(dim(Carseats)[1], dim(Carseats)[1]/2)  
Carseats.train = Carseats[train, ]  
Carseats.test = Carseats[-train, ]

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

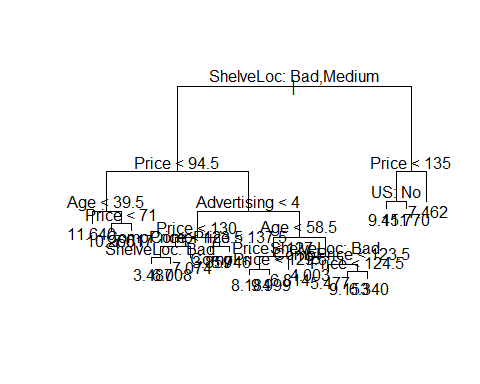
library(tree)

## Warning: package 'tree' was built under R version 4.2.1

tree.carseats = tree(Sales ~ ., data = Carseats.train)  
summary(tree.carseats)

##   
## Regression tree:  
## tree(formula = Sales ~ ., data = Carseats.train)  
## Variables actually used in tree construction:  
## [1] "ShelveLoc" "Price" "Age" "Advertising" "CompPrice"   
## [6] "US"   
## Number of terminal nodes: 18   
## Residual mean deviance: 2.167 = 394.3 / 182   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -3.88200 -0.88200 -0.08712 0.00000 0.89590 4.09900

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



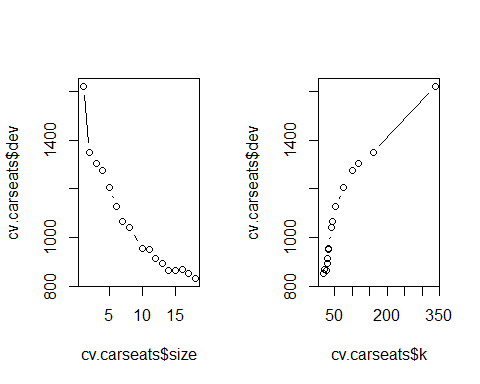
pred.carseats = predict(tree.carseats, Carseats.test)  
mean((Carseats.test$Sales - pred.carseats)^2)

## [1] 4.922039

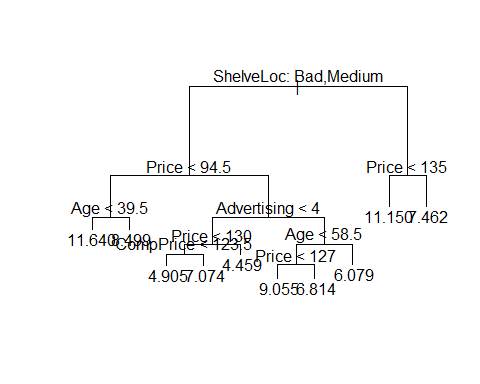
The test MSE is about 4.92.

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

cv.carseats = cv.tree(tree.carseats, FUN = prune.tree)  
par(mfrow = c(1, 2))  
plot(cv.carseats$size, cv.carseats$dev, type = "b")  
plot(cv.carseats$k, cv.carseats$dev, type = "b")



# Best size = 9  
pruned.carseats = prune.tree(tree.carseats, best = 9)  
par(mfrow = c(1, 1))  
plot(pruned.carseats)  
text(pruned.carseats, pretty = 0)



pred.pruned = predict(pruned.carseats, Carseats.test)  
mean((Carseats.test$Sales - pred.pruned)^2)

## [1] 4.918134

Pruning the tree in this case increases the test MSE to 4.91.

1. 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.

library(randomForest)

## Warning: package 'randomForest' was built under R version 4.2.1

## randomForest 4.7-1.1

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

bag.carseats = randomForest(Sales ~ ., data = Carseats.train, mtry = 10, ntree = 500,   
 importance = T)  
bag.pred = predict(bag.carseats, Carseats.test)  
mean((Carseats.test$Sales - bag.pred)^2)

## [1] 2.657296

importance(bag.carseats)

## %IncMSE IncNodePurity  
## CompPrice 23.07909904 171.185734  
## Income 2.82081527 94.079825  
## Advertising 11.43295625 99.098941  
## Population -3.92119532 59.818905  
## Price 54.24314632 505.887016  
## ShelveLoc 46.26912996 361.962753  
## Age 14.24992212 159.740422  
## Education -0.07662320 46.738585  
## Urban 0.08530119 8.453749  
## US 4.34349223 15.157608

Bagging improves the test MSE to 2.65. We also see that Price, ShelveLoc and Age are three most important predictors of Sale.

1. 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 effect of m, the number of variables considered at each split, on the error rate obtained.

rf.carseats = randomForest(Sales ~ ., data = Carseats.train, mtry = 5, ntree = 500,   
 importance = T)  
rf.pred = predict(rf.carseats, Carseats.test)  
mean((Carseats.test$Sales - rf.pred)^2)

## [1] 2.701665

importance(rf.carseats)

## %IncMSE IncNodePurity  
## CompPrice 19.8160444 162.73603  
## Income 2.8940268 106.96093  
## Advertising 11.6799573 106.30923  
## Population -1.6998805 79.04937  
## Price 46.3454015 448.33554  
## ShelveLoc 40.4412189 334.33610  
## Age 12.5440659 169.06125  
## Education 1.0762096 55.87510  
## Urban 0.5703583 13.21963  
## US 5.8799999 25.59797

In this case, random forest worsens the MSE on test set to 2.70. Changing m varies test MSE between 2.65 to 3. We again see that Price, ShelveLoc and Age are three most important predictors of Sale.

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

1. Remove the observations for whom the salary information is unknown, and then log-transform the salaries.

library(ISLR)  
sum(is.na(Hitters$Salary))

## [1] 59

Hitters = Hitters[-which(is.na(Hitters$Salary)), ]  
sum(is.na(Hitters$Salary))

## [1] 0

Hitters$Salary = log(Hitters$Salary)

1. Create a training set consisting of the first 200 observations, and a test set consisting of the remaining observations.

train = 1:200  
Hitters.train = Hitters[train, ]  
Hitters.test = Hitters[-train, ]

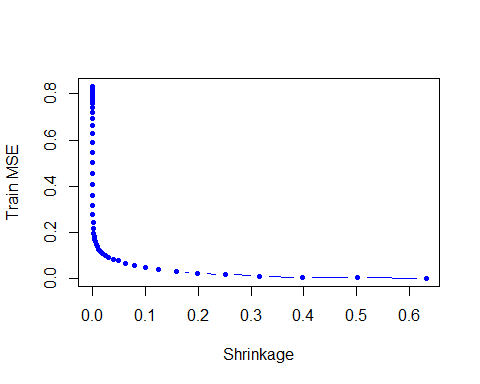
1. Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter λ. Produce a plot with different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

library(gbm)

## Warning: package 'gbm' was built under R version 4.2.1

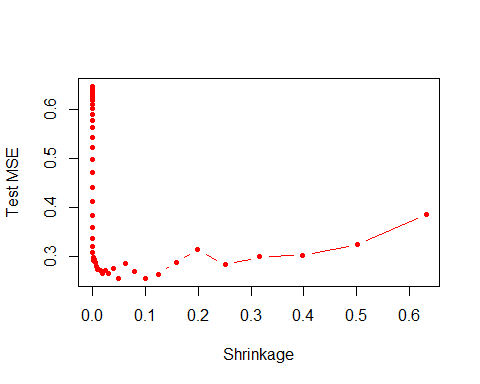
## Loaded gbm 2.1.8

set.seed(103)  
pows = seq(-10, -0.2, by = 0.1)  
lambdas = 10^pows  
length.lambdas = length(lambdas)  
train.errors = rep(NA, length.lambdas)  
test.errors = rep(NA, length.lambdas)  
for (i in 1:length.lambdas) {  
 boost.hitters = gbm(Salary ~ ., data = Hitters.train, distribution = "gaussian",   
 n.trees = 1000, shrinkage = lambdas[i])  
 train.pred = predict(boost.hitters, Hitters.train, n.trees = 1000)  
 test.pred = predict(boost.hitters, Hitters.test, n.trees = 1000)  
 train.errors[i] = mean((Hitters.train$Salary - train.pred)^2)  
 test.errors[i] = mean((Hitters.test$Salary - test.pred)^2)  
}  
  
plot(lambdas, train.errors, type = "b", xlab = "Shrinkage", ylab = "Train MSE",   
 col = "blue", pch = 20)



1. Produce a plot with different shrinkage values on the x-axis and the corresponding test set MSE on the y-axis.

plot(lambdas, test.errors, type = "b", xlab = "Shrinkage", ylab = "Test MSE", col = "red", pch = 20)



min(test.errors)

## [1] 0.2560507

lambdas[which.min(test.errors)]

## [1] 0.05011872

Minimum test error is obtained at λ=0.05.

1. 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.

lm.fit = lm(Salary ~ ., data = Hitters.train)  
lm.pred = predict(lm.fit, Hitters.test)  
mean((Hitters.test$Salary - lm.pred)^2)

## [1] 0.4917959

library(glmnet)

## Warning: package 'glmnet' was built under R version 4.2.1

## Loading required package: Matrix

## Loaded glmnet 4.1-4

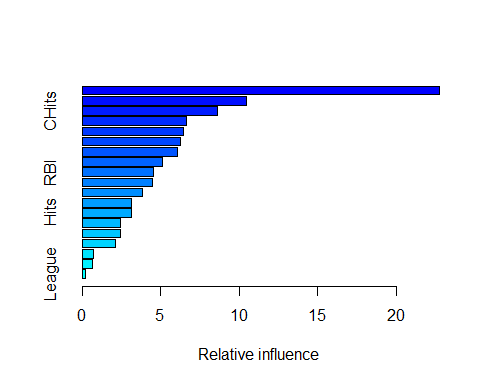
set.seed(134)  
x = model.matrix(Salary ~ ., data = Hitters.train)  
y = Hitters.train$Salary  
x.test = model.matrix(Salary ~ ., data = Hitters.test)  
lasso.fit = glmnet(x, y, alpha = 1)  
lasso.pred = predict(lasso.fit, s = 0.01, newx = x.test)  
mean((Hitters.test$Salary - lasso.pred)^2)

## [1] 0.4700537

Both linear model and regularization like Lasso have higher test MSE than boosting.

1. Which variables appear to be the most important predictors in the boosted model?

boost.best = gbm(Salary ~ ., data = Hitters.train, distribution = "gaussian",   
 n.trees = 1000, shrinkage = lambdas[which.min(test.errors)])  
summary(boost.best)



## var rel.inf  
## CAtBat CAtBat 22.7562681  
## CWalks CWalks 10.4279674  
## CHits CHits 8.6198109  
## PutOuts PutOuts 6.6159325  
## Years Years 6.4611683  
## Walks Walks 6.2331148  
## CRBI CRBI 6.0926744  
## CHmRun CHmRun 5.1076104  
## RBI RBI 4.5321678  
## CRuns CRuns 4.4728132  
## Assists Assists 3.8366575  
## HmRun HmRun 3.1554038  
## Hits Hits 3.1229284  
## AtBat AtBat 2.4338530  
## Errors Errors 2.4324185  
## Runs Runs 2.1425481  
## Division Division 0.7041949  
## NewLeague NewLeague 0.6675446  
## League League 0.1849234

CAtBat, CRBI and CWalks are three most important variables in that order.

1. Now apply bagging to the training set. What is the test set MSE for this approach?

library(randomForest)

set.seed(21)  
rf.hitters = randomForest(Salary ~ ., data = Hitters.train, ntree = 500, mtry = 19)  
rf.pred = predict(rf.hitters, Hitters.test)  
mean((Hitters.test$Salary - rf.pred)^2)

## [1] 0.2303919

Test MSE for bagging is about 0.23, which is slightly lower than the best test MSE for boosting.