**Statistical Learning 6B**

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.

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

**library(randomForest)**

**library(leaps)**

**library(glmnet)**

**library(gbm)**

**library(ISLR)**

**library(tree)**

**attach(Carseats)**

**set.seed(1)**

**train <- sample(1:nrow(Carseats), nrow(Carseats) / 2)**

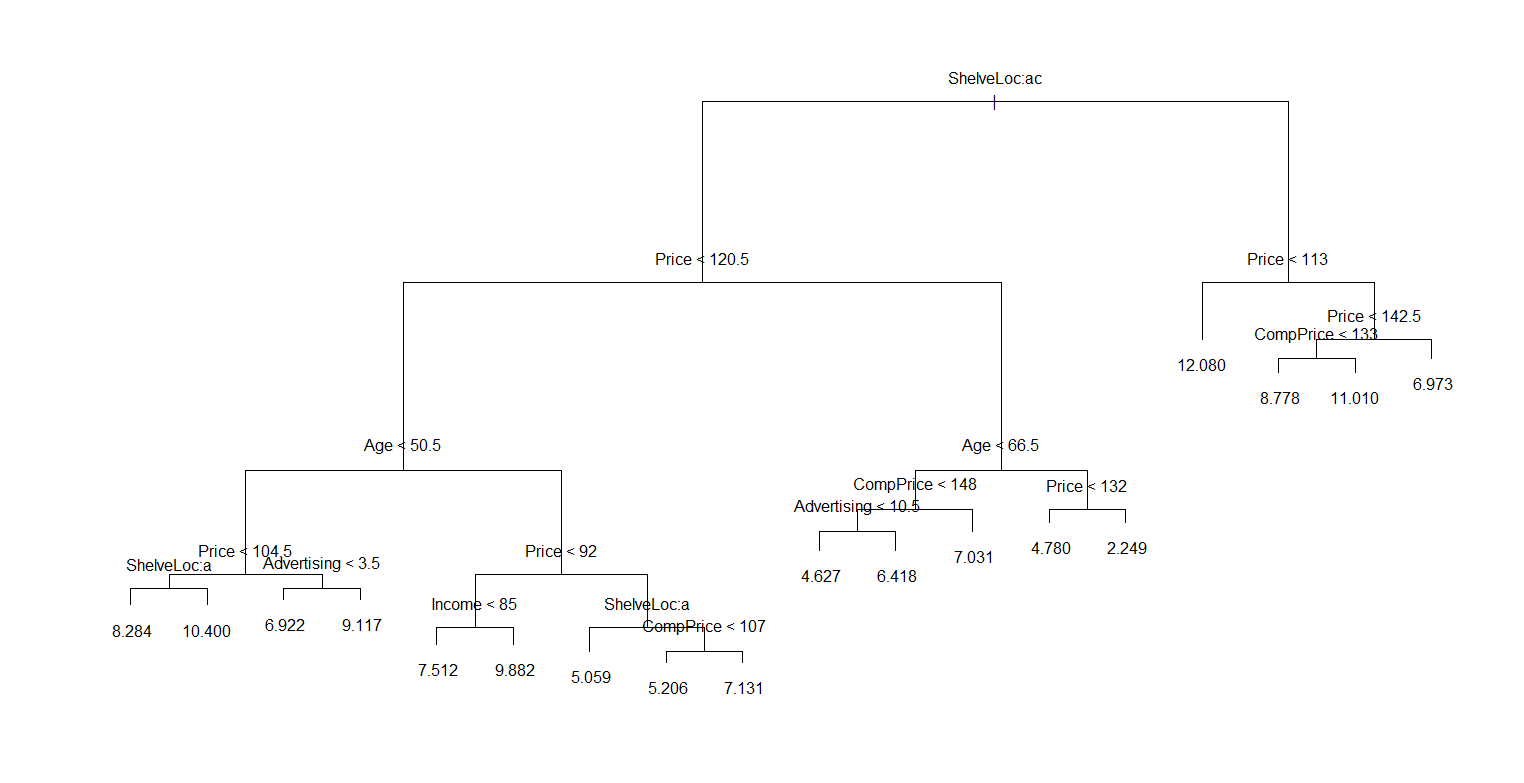
**data.train <- Carseats[train, ]**

**data.test <- Carseats[-train, ]**

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

**tree.fit <- tree(Sales ~ ., data = data.train)**

**plot(tree.fit)**



**text(tree.fit)**

**summary(tree.fit)**

**Regression tree:**

**tree(formula = Sales ~ ., data = data.train)**

**Variables actually used in tree construction:**

**[1] "ShelveLoc" "Price" "Age" "Advertising" "Income" "CompPrice"**

**Number of terminal nodes: 18**

**Residual mean deviance: 2.36 = 429.5 / 182**

**Distribution of residuals:**

**Min. 1st Qu. Median Mean 3rd Qu. Max.**

**-4.2570 -1.0360 0.1024 0.0000 0.9301 3.9130**

**yhat <- predict(tree.fit, newdata = data.test)**

**mean((yhat - data.test$Sales)^2)**

**4.148897**

**The tree test MSE is 4.15.**

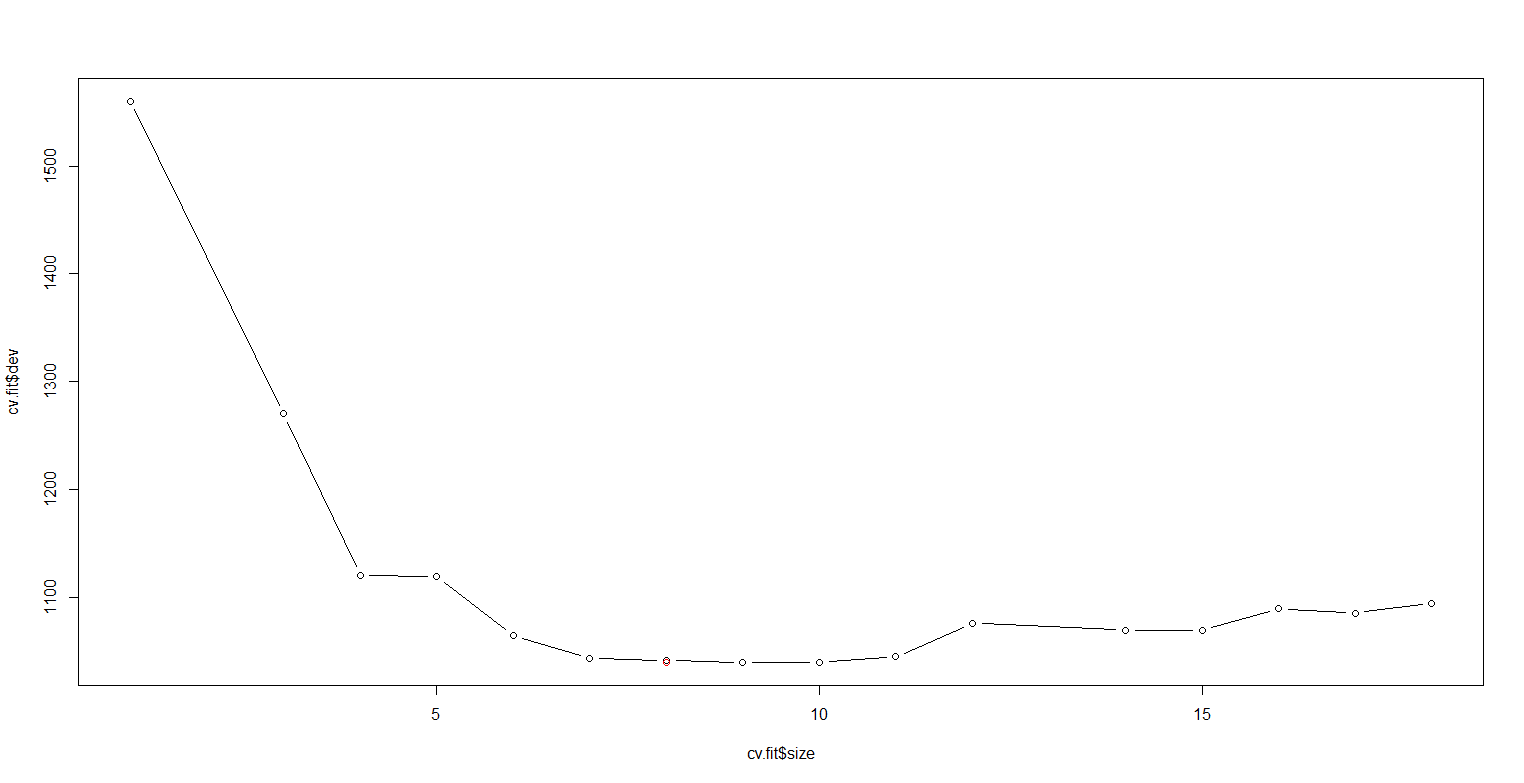
(c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

**cv.fit <- cv.tree(tree.fit)**

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

**tree.min <- which.min(cv.fit$dev)**

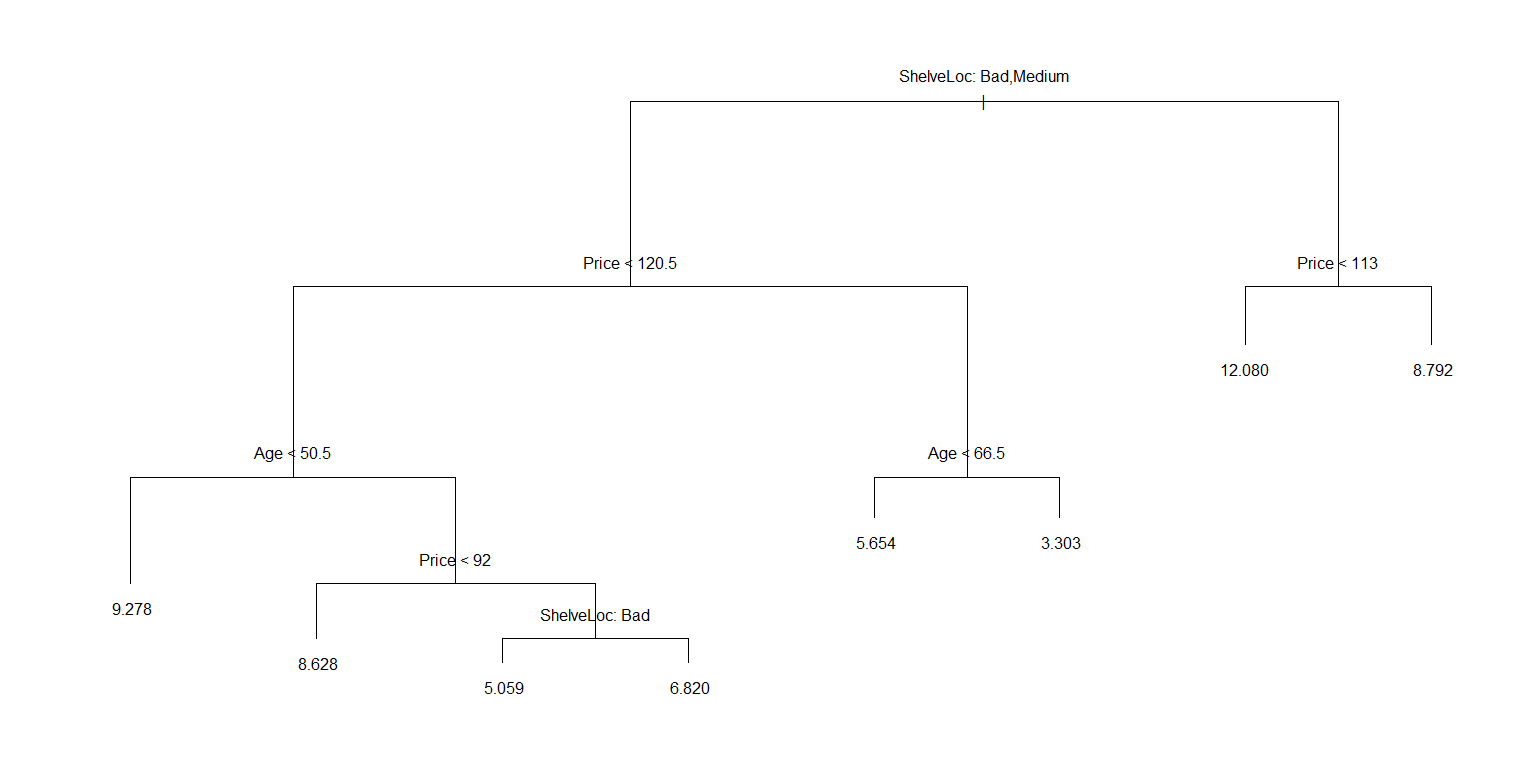
**points(tree.min, cv.fit$dev[tree.min], col = "red")**



**prune.tree.fit <- prune.tree(tree.fit, best = 8)**

**plot(prune.tree.fit)**

**text(prune.tree.fit, pretty = 0)**



**yhat2 <- predict(prune.tree.fit, newdata = data.test)**

**mean((yhat2 - data.test$Sales)^2)**

**5.09085**

**No, pruning the tree increased the MSE to 5.09.**

(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.

**bag.fit <- randomForest(Sales ~ ., data = data.train, mtry = 10, ntree = 500, importance = TRUE)**

**yhat.bag <- predict(bag.fit, newdata = data.test)**

**mean((yhat.bag - data.test$Sales)^2)**

**2.604369**

**The MSE decreased to 2.60.**

**importance(bag.carseats)**

**%IncMSE IncNodePurity**

**CompPrice 14.4124562 133.731797**

**Income 6.5147532 74.346961**

**Advertising 15.7607104 117.822651**

**Population 0.6031237 60.227867**

**Price 57.8206926 514.802084**

**ShelveLoc 43.0486065 319.117972**

**Age 19.8789659 192.880596**

**Education 2.9319161 39.490093**

**Urban -3.1300102 8.695529**

**US 7.6298722 15.723975**

**We can see from here, Price and ShelveLoc are the most important.**

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

**rf.fit <- randomForest(Sales ~ ., data = data.train, mtry = 3, ntree = 500, importance = TRUE)**

**yhat.rf <- predict(rf.fit, newdata = data.test)**

**mean((yhat.rf - data.test$Sales)^2)**

**3.296078**

**MSE increased compare to bagging.**

**importance(rf.fit)**

**%IncMSE IncNodePurity**

**CompPrice 7.5233429 127.36625**

**Income 4.3612064 119.19152**

**Advertising 12.5799388 138.13567**

**Population -0.2974474 100.28836**

**Price 37.1612032 383.12126**

**ShelveLoc 30.3751253 246.19930**

**Age 16.0261885 197.44865**

**Education 1.7855151 63.87939**

**Urban -1.3928225 16.01173**

**US 5.6393475 32.85850**

**Price and ShelveLoc remains the most important variables, but level of importance dropped. m=sqrt(p)=3.**

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.

**attach(Hitters)**

**Hitters <- na.omit(Hitters)**

**Hitters$Salary <- log(Hitters$Salary)**

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

**Hitters.train <- Hitters[1:200,]**

**Hitters.test <- Hitters[201:nrow(Hitters),]**

(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 different shrinkage values on the *x*-axis and the corresponding training set MSE on the *y*-axis.

**set.seed(1)**

**shrinkage<-c(0.00001,0.0001,0.001,0.01,0.1,1)**

**errs<-rep(NA,length(shrinkage))**

**for (i in 1:length(shrinkage)){**

**s<-shrinkage[i]**

**boost.Hitters<-gbm(Salary~., data=Hitters.train,**

**distribution="gaussian",**

**n.trees=1000,**

**shrinkage = s,**

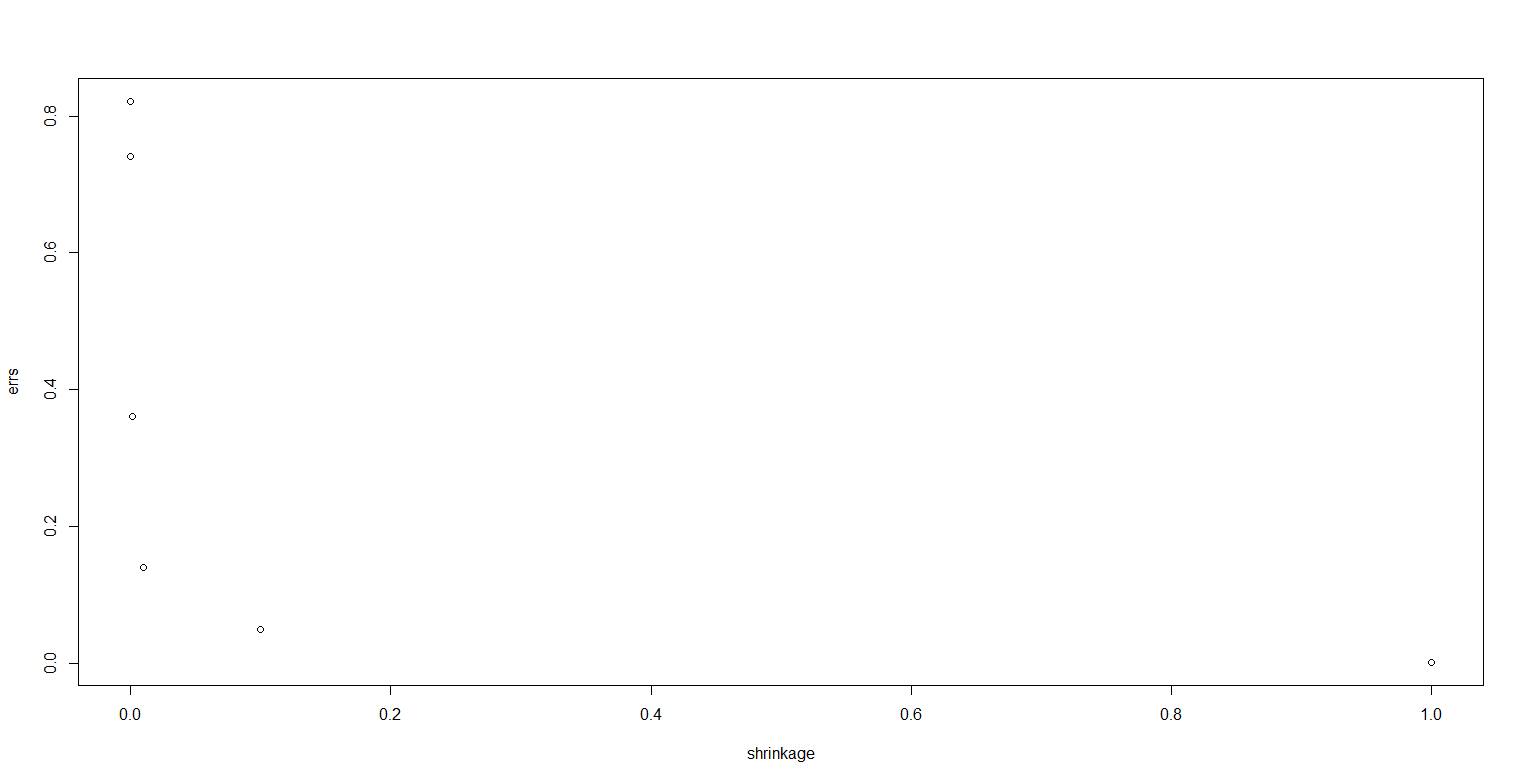
**interaction.depth=1,**

**n.cores=10)**

**yhat.boost<-predict(boost.Hitters,newdata=Hitters.train, n.trees=1000)**

**errs[i]<-mean((yhat.boost-Hitters.train$Salary)^2)**

**}**

**plot(shrinkage,errs)**

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

**set.seed(1)**

**errs.test<-rep(NA,length(shrinkage))**

**for (i in 1:length(shrinkage)){**

**s<-shrinkage[i]**

**boost.Hitters<-gbm(Salary~., data=Hitters.train,**

**distribution="gaussian",**

**n.trees=1000,**

**shrinkage = s,**

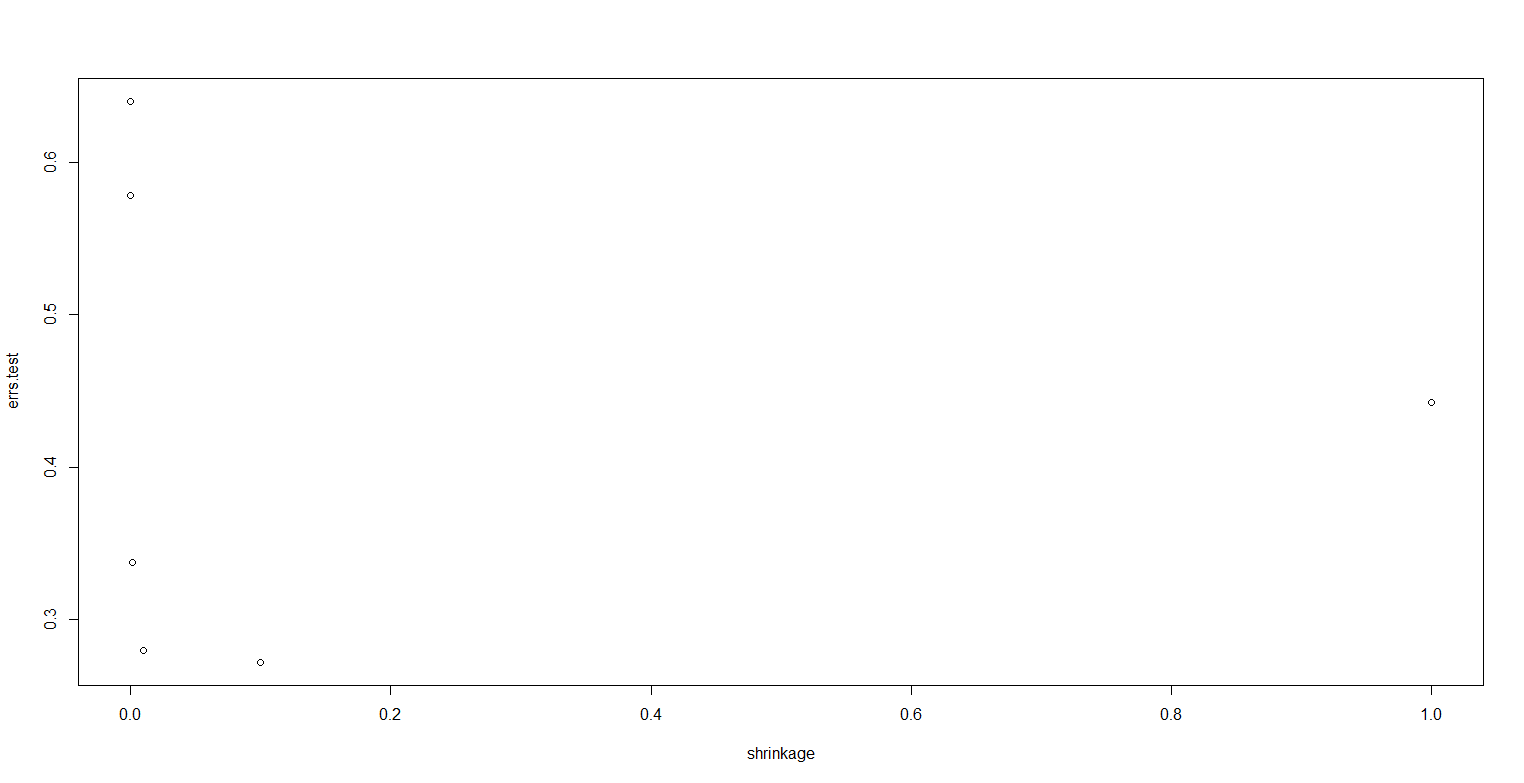
**interaction.depth=1,**

**n.cores=10)**

**yhat.boost<-predict(boost.Hitters,newdata=Hitters.test, n.trees=1000)**

**errs.test[i]<-mean((yhat.boost-Hitters.test$Salary)^2)**

**}**

**plot(shrinkage,errs.test)**

(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.

**fit1 <- lm(Salary ~ ., data = Hitters.train)**

**pred1 <- predict(fit1, Hitters.test)**

**mean((pred1 - Hitters.test$Salary)^2)**

**0.4917959**

**x.train <- model.matrix(Salary ~ ., data = Hitters.train)**

**x.test <- model.matrix(Salary ~ ., data = Hitters.test)**

**y.train <- Hitters.train$Salary**

**fit2 <- glmnet(x.train, y.train, alpha = 0)**

**pred2 <- predict(fit2, s = 0.01, newx = x.test)**

**mean((pred2 - Hitters.test$Salary)^2)**

**0.4570283**

**Test error in boosting is the lowest.**

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

**boost.hitters <- gbm(Salary ~ ., data = Hitters.train, distribution = "gaussian", n.trees = 1000)**

**summary(boost.hitters)**

**var rel.inf**

**CAtBat CAtBat 44.13402587**

**CHits CHits 14.72265906**

**CWalks CWalks 13.02408635**

**CRuns CRuns 11.67293660**

**CRBI CRBI 11.41338777**

**CHmRun CHmRun 1.60919801**

**Hits Hits 1.25363281**

**Years Years 0.99206059**

**RBI RBI 0.61019110**

**AtBat AtBat 0.44395189**

**Walks Walks 0.07937822**

**Runs Runs 0.04449174**

**HmRun HmRun 0.00000000**

**League League 0.00000000**

**Division Division 0.00000000**

**PutOuts PutOuts 0.00000000**

**Assists Assists 0.00000000**

**Errors Errors 0.00000000**

**NewLeague NewLeague 0.00000000**

**As we can see CAtBat is far more significant than other variables.**

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

**set.seed(1)**

**bag.hitters <- randomForest(Salary ~ ., data = Hitters.train, mtry = 19, ntree = 1000)**

**yhat.bag <- predict(bag.hitters, newdata = Hitters.test)**

**mean((yhat.bag - Hitters.test$Salary)^2)**

**0.2291616**

9. This problem involves the OJ data set which is part of the ISLR package.

(a) Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

**set.seed(1)**

**attach(OJ)**

**train <- sample(1:nrow(OJ), 800)**

**OJ.train <- OJ[train, ]**

**OJ.test <- OJ[-train, ]**

(b) Fit a tree to the training data, with Purchase as the response and the other variables except for Buy as predictors. Use the summary() function to produce summary statistics about the tree, and describe the results obtained. What is the training error rate? How many terminal nodes does the tree have?

**tree.oj <- tree(Purchase ~ ., data = OJ.train)**

**summary(tree.oj)**

**Classification tree:**

**tree(formula = Purchase ~ ., data = OJ.train)**

**Variables actually used in tree construction:**

**[1] "LoyalCH" "PriceDiff" "SpecialCH" "ListPriceDiff"**

**Number of terminal nodes: 8**

**Residual mean deviance: 0.7305 = 578.6 / 792**

**Misclassification error rate: 0.165 = 132 / 800**

**There is no such predictor as Buy exists in the dataset. The training error is 0.165, and the tree has 8 terminals.**

(c) Type in the name of the tree object in order to get a detailed text output. Pick one of the terminal nodes, and interpret the information displayed.

**tree.oj**

**1) root 800 1064.00 CH ( 0.61750 0.38250 )**

**2) LoyalCH < 0.508643 350 409.30 MM ( 0.27143 0.72857 )**

**4) LoyalCH < 0.264232 166 122.10 MM ( 0.12048 0.87952 )**

**8) LoyalCH < 0.0356415 57 10.07 MM ( 0.01754 0.98246 ) \***

**9) LoyalCH > 0.0356415 109 100.90 MM ( 0.17431 0.82569 ) \***

**5) LoyalCH > 0.264232 184 248.80 MM ( 0.40761 0.59239 )**

**10) PriceDiff < 0.195 83 91.66 MM ( 0.24096 0.75904 )**

**20) SpecialCH < 0.5 70 60.89 MM ( 0.15714 0.84286 ) \***

**21) SpecialCH > 0.5 13 16.05 CH ( 0.69231 0.30769 ) \***

**11) PriceDiff > 0.195 101 139.20 CH ( 0.54455 0.45545 ) \***

**3) LoyalCH > 0.508643 450 318.10 CH ( 0.88667 0.11333 )**

**6) LoyalCH < 0.764572 172 188.90 CH ( 0.76163 0.23837 )**

**12) ListPriceDiff < 0.235 70 95.61 CH ( 0.57143 0.42857 ) \***

**13) ListPriceDiff > 0.235 102 69.76 CH ( 0.89216 0.10784 ) \***

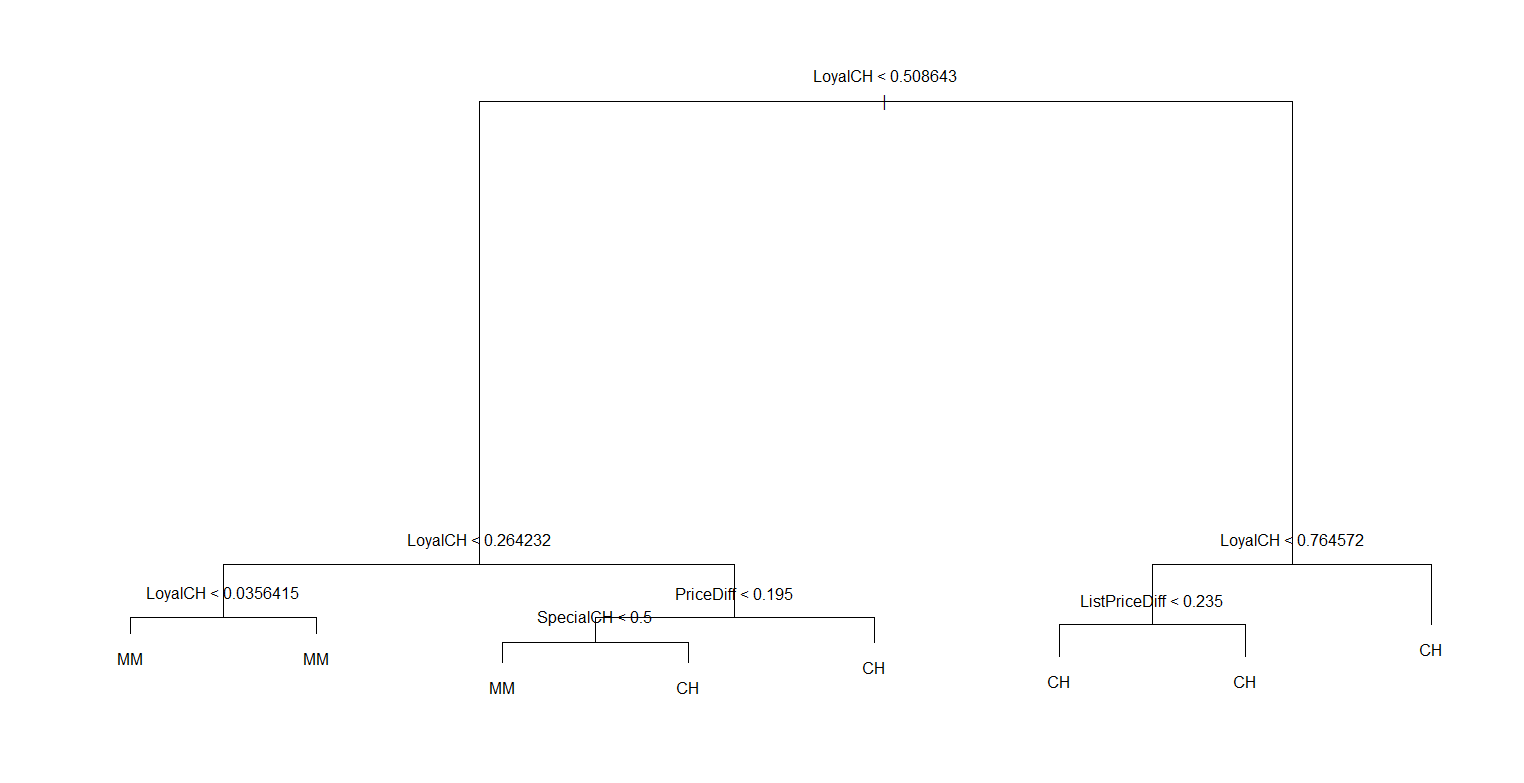
**7) LoyalCH > 0.764572 278 86.14 CH ( 0.96403 0.03597 ) \***

**Terminal 20, SpecialCH. It has 70 observations categorized as MM, with deviance of 60.89. Error rate is about 15.714%.**

(d) Create a plot of the tree, and interpret the results.

**plot(tree.oj)**

**text(tree.oj)**



**It appears that LoyalCH is a very important predictor as it shows up in more than one layers/branches and causes a wide spread.**

(e) Predict the response on the test data, and produce a confusion matrix comparing the test labels to the predicted test labels. What is the test error rate?

**tree.pred <- predict(tree.oj, OJ.test, type = "class")**

**table(tree.pred, OJ.test$Purchase)**

**tree.pred CH MM**

**CH 147 49**

**MM 12 62**

**(49+12)/270**

**0.2259259**

**Among 270 predictions, 49+12 are wrong, so the error rate is 22.59%**

(f) Apply the cv.tree() function to the training set in order to determine the optimal tree size.

**cv.oj <- cv.tree(tree.oj, FUN = prune.misclass)**

**cv.oj**

**$size**

**[1] 8 5 2 1**

**$dev**

**[1] 139 139 153 306**

**$k**

**[1] -Inf 0.000000 4.666667 160.000000**

**$method**

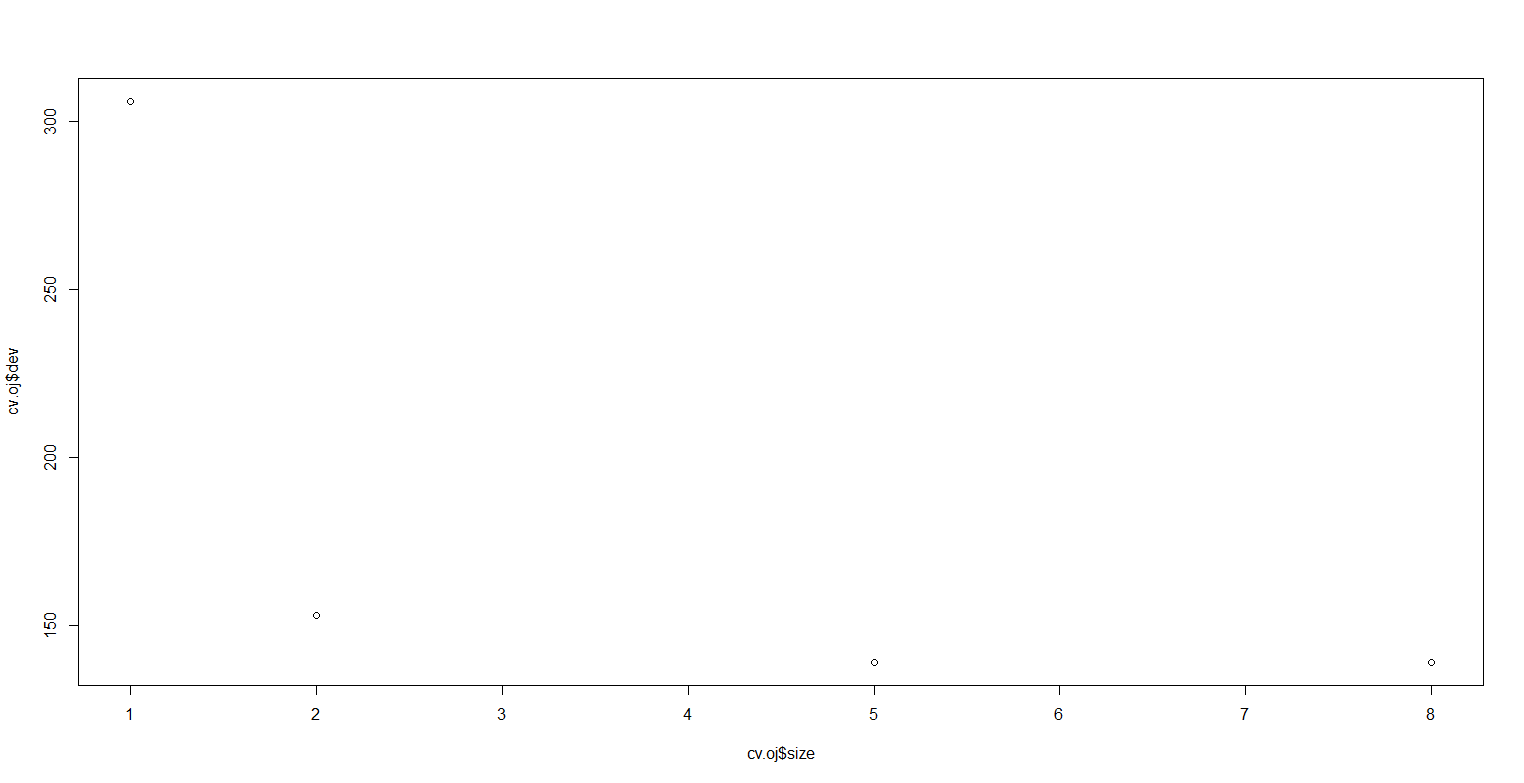
**[1] "misclass"**

**attr(,"class")**

**[1] "prune" "tree.sequence"**

(g) Produce a plot with tree size on the *x*-axis and cross-validated classification error rate on the *y*-axis.

**plot(cv.oj$size, cv.oj$dev)**



(h) Which tree size corresponds to the lowest cross-validated classification error rate?

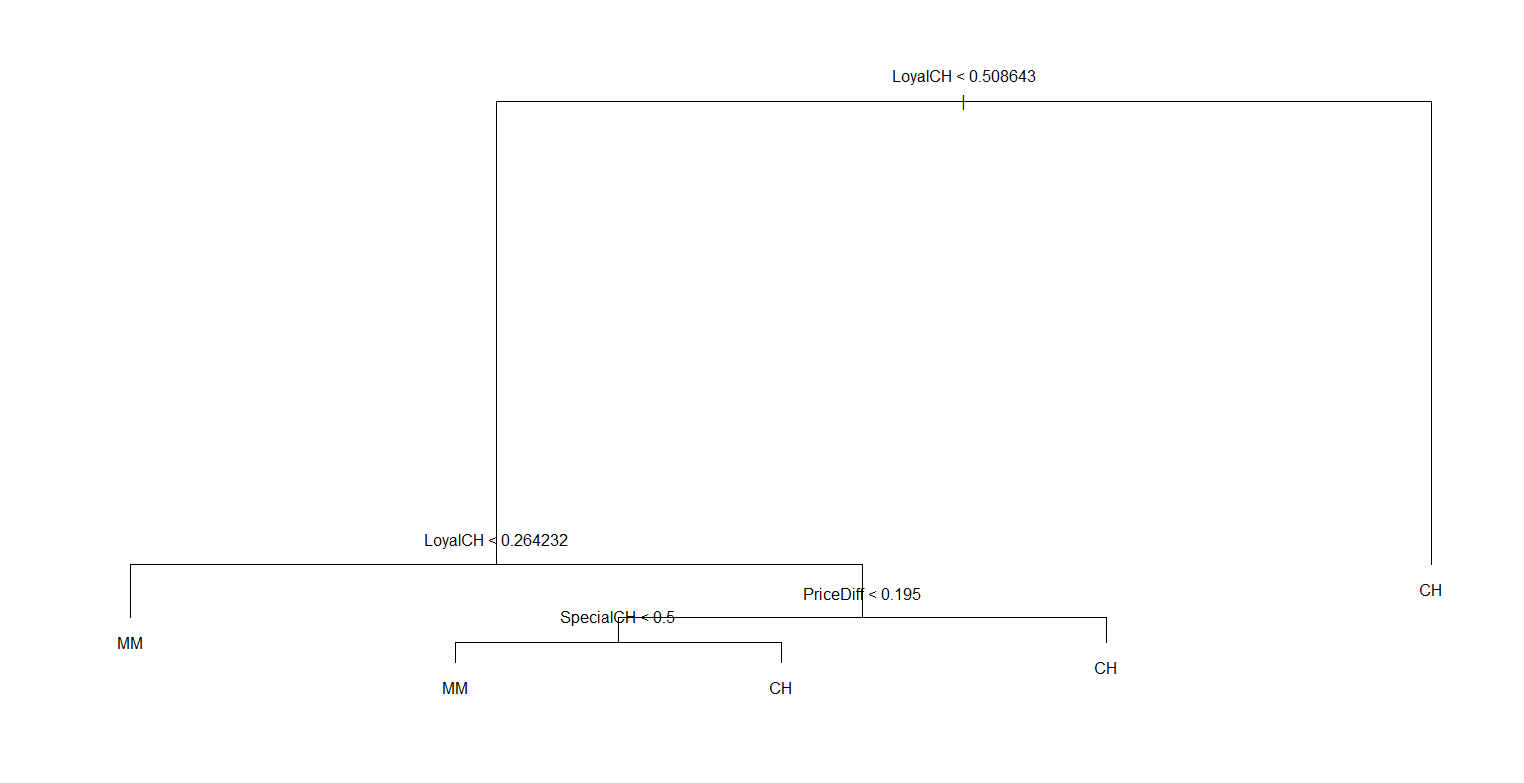
**We can see that 5 knots offers the lowest error rate.**

1. Produce a pruned tree corresponding to the optimal tree size obtained using cross-validation. If cross-validation does not lead to selection of a pruned tree, then create a pruned tree with five terminal nodes.

**prune.oj <- prune.misclass(tree.oj, best = 5)**

**plot(prune.oj)**

**text(prune.oj)**



(j) Compare the training error rates between the pruned and unpruned trees. Which is higher?

**summary(prune.oj)**

**Classification tree:**

**snip.tree(tree = tree.oj, nodes = 3:4)**

**Variables actually used in tree construction:**

**[1] "LoyalCH" "PriceDiff" "SpecialCH"**

**Number of terminal nodes: 5**

**Residual mean deviance: 0.8256 = 656.4 / 795**

**Misclassification error rate: 0.165 = 132 / 800**

**It has the same error rate with 8 knots.**

(k) Compare the test error rates between the pruned and unpruned trees. Which is higher?

**prune.pred <- predict(prune.oj, OJ.test, type = "class")**

**table(prune.pred, OJ.test$Purchase)**

**prune.pred CH MM**

**CH 147 49**

**MM 12 62**

**(12+49)/270**

**0.2259259**

**It is the same as the unpruned tree.**