# Data Mining HW4 Applied

Yaniv Bronshtein

5/7/2021

**Import the necessary libraries**

```
library(ISLR)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.5     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(leaps)
library(gam)
```

```
## Warning: package 'gam' was built under R version 4.0.5

## Loading required package: splines

## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loaded gam 1.20
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.0.5
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.0.5
```

```
## Loaded gbm 2.1.8
```

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.5
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

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

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

#Question 2: GAM Problem 10 Pg 300 of ISL a). Split the data into a training set and a test set. Using out-of-state tuition as the response and the other variables as the predictors, perform forward stepwise selection on the training set in order to identify a satisfactory model that uses just a subset of the predictors

```
college_t = College %>% as_tibble()
set.seed(1)
train <- sample(nrow(college_t) * .7)
test <- -train
college_train <- college_t[train, ]
college_test <- college_t[test, ]
```

**Fit the forward stepwise selection model**

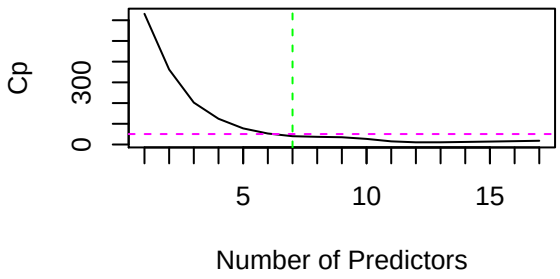
```
reg_fit_fwd <- leaps::regsubsets(Outstate ~ ., data = college_train,
                                 nvmax=17, method = "forward")
```

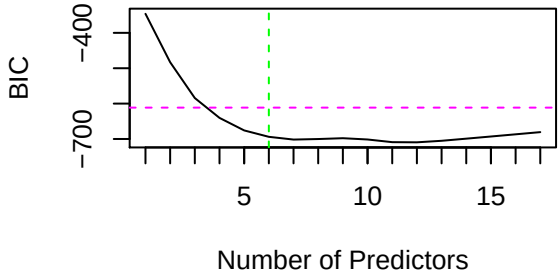
**Extract the metrics from the summary object to perform analysis**

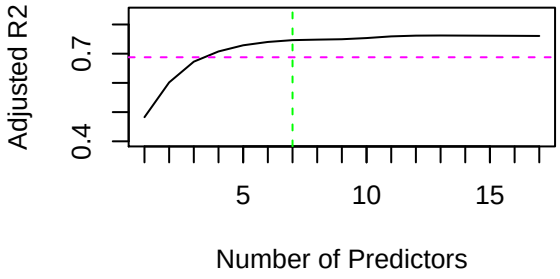
```
reg_summary <- summary(reg_fit_fwd)
cp <- reg_summary$cp
bic <- reg_summary$bic
adjr2 <- reg_summary$adjr2
```

**Create plots to determine the number of predictors**

```
par(mfrow=c(2, 2))
xtick <- seq(1, 17, by = 1)

# Cp plot
plot(cp, xlab="Number of Predictors",ylab="Cp",type='l')
axis(side = 1, at = xtick, labels = FALSE)
min_cp <-  min(cp)
std_cp <-  sd(cp)
abline(h=min_cp + std_cp/sqrt(length(cp)), col="magenta", lty=2)
abline(v=which(cp < min_cp + std_cp/sqrt(length(cp)))[1], col="green", lty=2)


# BIC plot
plot(bic, xlab="Number of Predictors",ylab="BIC",type='l')
axis(side = 1, at = xtick, labels = FALSE)
min_bic <- min(bic)
std_bic <-  sd(bic)
abline(h = min_bic + std_bic, col="magenta", lty=2)
abline(v=which(bic < min_bic + std_bic/sqrt(length(bic)))[1], col="green", lty=2)



# Adjusted R^2 plot
plot(adjr2, xlab="Number of Predictors",
     ylab="Adjusted R2",type='l', ylim=c(0.4, 0.84))
axis(side = 1, at = xtick, labels = FALSE)
max_adjr2 <-  max(reg_summary$adjr2)
std_adjr2 <-  sd(reg_summary$adjr2)
abline(h=max_adjr2 - std_adjr2, col="magenta", lty=2)
abline(v=which(max_adjr2 - std_adjr2/sqrt(length(adjr2)) < adjr2)[1], col="green", lty=2)
```



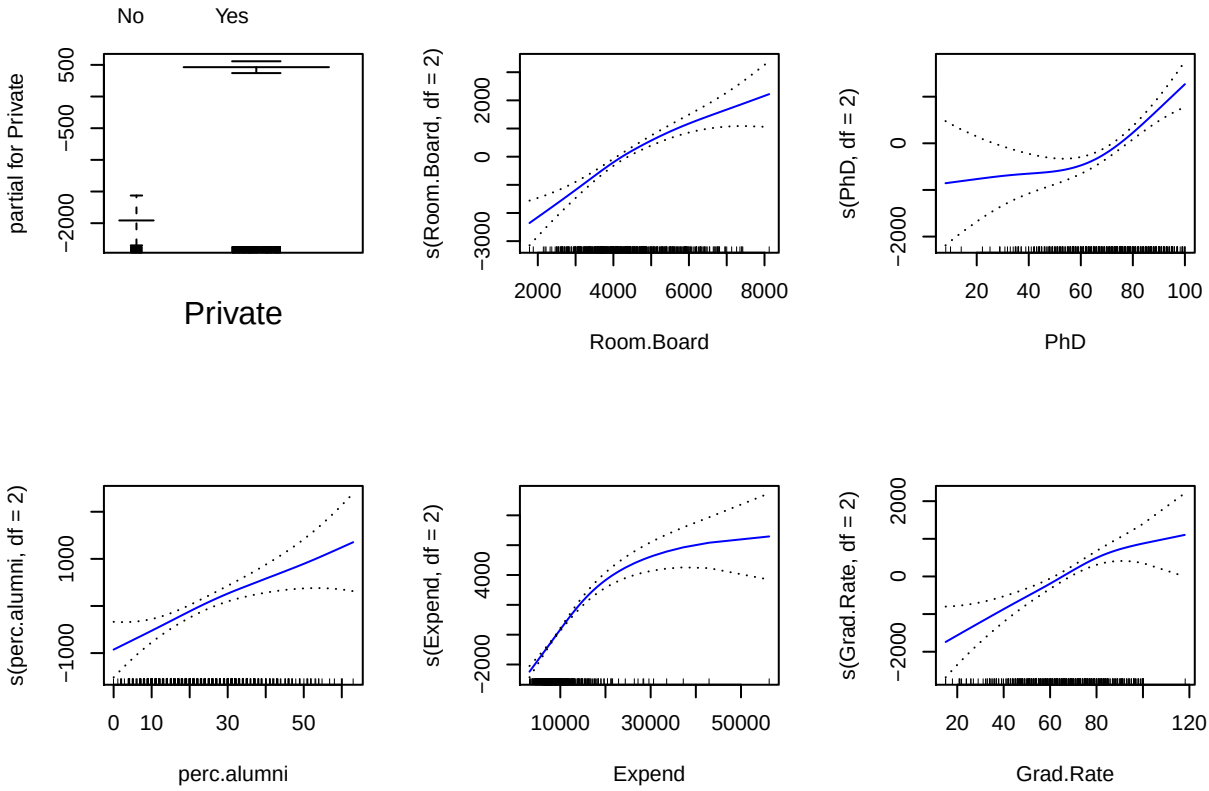

**Based on the plots, we will take the conservative BIC estimate of 6 predictors**

```
coef(reg_fit_fwd, 6)
```

```
##   (Intercept)    PrivateYes    Room.Board           PhD   perc.alumni
## -3769.0587788  2748.6944010     0.8999634    38.5143460    44.4889713
##        Expend     Grad.Rate
##     0.2543900    31.2043096
```

b). Fit a GAM on the training data, using out-of-state tuition as the response

```
gam_model <- gam(Outstate ~ Private + s(Room.Board, df=2) + s(PhD, df=2) +
                   s(perc.alumni, df=2) + s(Expend, df=2) + s(Grad.Rate, df=2),
                 data=college_train)

par(mfrow=c(2,3))
plot(gam_model, se=TRUE, col="blue")
```




*Based on the plots, holding all other variables constant, Private increases the cost of Out of state tuition. The other variables Room.Board, PhD, perc.alumni, Expend, and Grad.Rate show the same trend*

c). Evaluate the model obtained on the test set, and explain the results obtained

**Determine the gam RMSE**

```
gam_pred <- predict(gam_model, college_test)
gam_rmse <- sqrt(mean((college_test$Outstate - gam_pred)^2))
gam_rmse
```

```
## [1] 1984.385
```

**Determine the gam r^2**

```
gam_r2 <- 1 - (sum((college_test$Outstate - gam_pred)^2) /
                 sum((college_test$Outstate - mean(college_test$Outstate))^2))
gam_r2
```

```
## [1] 0.7614328
```

*We get an RMSE of 1984.385 and an R^2 of 0.7614328 or around 76%. This is a very strong result* **Let us print the summary object for the GAM model**

```
summary(gam_model)
```

```
##
## Call: gam(formula = Outstate ~ Private + s(Room.Board, df = 2) + s(PhD,
##     df = 2) + s(perc.alumni, df = 2) + s(Expend, df = 2) + s(Grad.Rate,
##     df = 2), data = college_train)
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
## -7164.185 -1192.389     9.746  1195.918  8668.434
##
## (Dispersion Parameter for gaussian family taken to be 3515849)
##
##     Null Deviance: 8614032615 on 542 degrees of freedom
## Residual Deviance: 1866916248 on 531.0002 degrees of freedom
## AIC: 9739.358
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                          Df     Sum Sq    Mean Sq F value    Pr(>F)
## Private                   1 1968096367 1968096367 559.779 < 2.2e-16 ***
## s(Room.Board, df = 2)     1 1852547030 1852547030 526.913 < 2.2e-16 ***
## s(PhD, df = 2)            1  771964114  771964114 219.567 < 2.2e-16 ***
## s(perc.alumni, df = 2)    1  376380024  376380024 107.052 < 2.2e-16 ***
## s(Expend, df = 2)         1  669471730  669471730 190.415 < 2.2e-16 ***
## s(Grad.Rate, df = 2)      1  105813599  105813599  30.096 6.372e-08 ***
## Residuals               531 1866916248    3515849
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                          Npar Df Npar F     Pr(F)
## (Intercept)
## Private
## s(Room.Board, df = 2)         1  5.624 0.0180708 *
## s(PhD, df = 2)                1 11.780 0.0006455 ***
## s(perc.alumni, df = 2)        1  0.517 0.4725227
## s(Expend, df = 2)             1 70.804 4.441e-16 ***
## s(Grad.Rate, df = 2)          1  3.159 0.0761030 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

*Based on the plots and summary, holding all other variables constant, Private increases the cost of Out of state tuition. The other variables Room.Board, PhD, perc.alumni, Expend, and Grad.Rate show the same trend*

d). For which variables, if any, is there evidence of a non-linear relationship with the response? *Based on the summary, we can see a non-linear relationship between out-of-state tuition and instructional expenditure per student and a non-linear relationship between out-of-state tuition and percent of faculty with PhD's. To a lesser degree, we see a non-linear relationship between out-of-state tuition and room and board costs*

## Question 3 Decision Tree. Proble 9 at Page 334 of ISL

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

```
data(OJ)
set.seed(1)
train <- sample(800)
test <- -train
oj_train <- OJ[train, ]
oj_test <- OJ[test, ]
```

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_model <- tree(Purchase ~., data = oj_train)
summary(tree_model)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = oj_train)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  7
## Residual mean deviance:  0.7342 = 582.3 / 793
## Misclassification error rate: 0.165 = 132 / 800
```

*The tree obtained has 7 terminal nodes with an error rate of 0.165 or 16.5% Two variables were used in the tree construction: LoyalCH and PriceDiff, suggesting that these were the only features influencing customer purchases*

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.

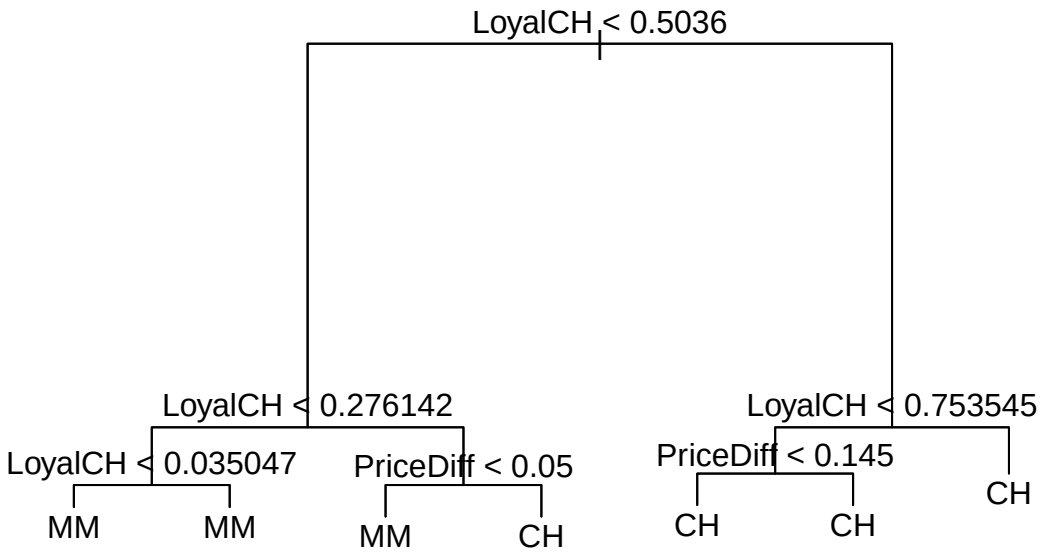
```
tree_model
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1062.00 CH ( 0.62125 0.37875 )
##    2) LoyalCH < 0.5036 348  413.70 MM ( 0.28161 0.71839 )
##      4) LoyalCH < 0.276142 164  113.50 MM ( 0.10976 0.89024 )
##        8) LoyalCH < 0.035047 62    0.00 MM ( 0.00000 1.00000 ) *
##        9) LoyalCH > 0.035047 102   95.06 MM ( 0.17647 0.82353 ) *
##      5) LoyalCH > 0.276142 184  251.90 MM ( 0.43478 0.56522 )
##       10) PriceDiff < 0.05 75   77.75 MM ( 0.21333 0.78667 ) *
##       11) PriceDiff > 0.05 109  147.80 CH ( 0.58716 0.41284 ) *
##    3) LoyalCH > 0.5036 452  326.70 CH ( 0.88274 0.11726 )
##      6) LoyalCH < 0.753545 172  188.90 CH ( 0.76163 0.23837 )
##       12) PriceDiff < 0.145 67   92.15 CH ( 0.55224 0.44776 ) *
##       13) PriceDiff > 0.145 105   70.44 CH ( 0.89524 0.10476 ) *
##      7) LoyalCH > 0.753545 280   99.08 CH ( 0.95714 0.04286 ) *
```

*Suppose a customer scored LoyalCH >=0.51. They will be predicted to be of class CH. Thus, we expect them to purchase Citrus Hill instead of Minute Maid*

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

```
par(mfrow=c(1,1))
plot(tree_model)
text(tree_model, pretty = 0)
```




*The plot gives the same results as the printed model. Given information about our customer, we can use the visualization to predict which orange juice brand they will purchase*

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_model, oj_test, type = "class")
table(tree_pred, oj_test$Purchase)
```

```
##
## tree_pred  CH  MM
##         CH 141  44
##         MM  15  70
```

**Determine the test error for the tree prediction**

```
test_error <- mean(tree_pred != oj_test$Purchase)
test_error
```

```
## [1] 0.2185185
```

*The test error is 0.2185185 or around 21.85%*

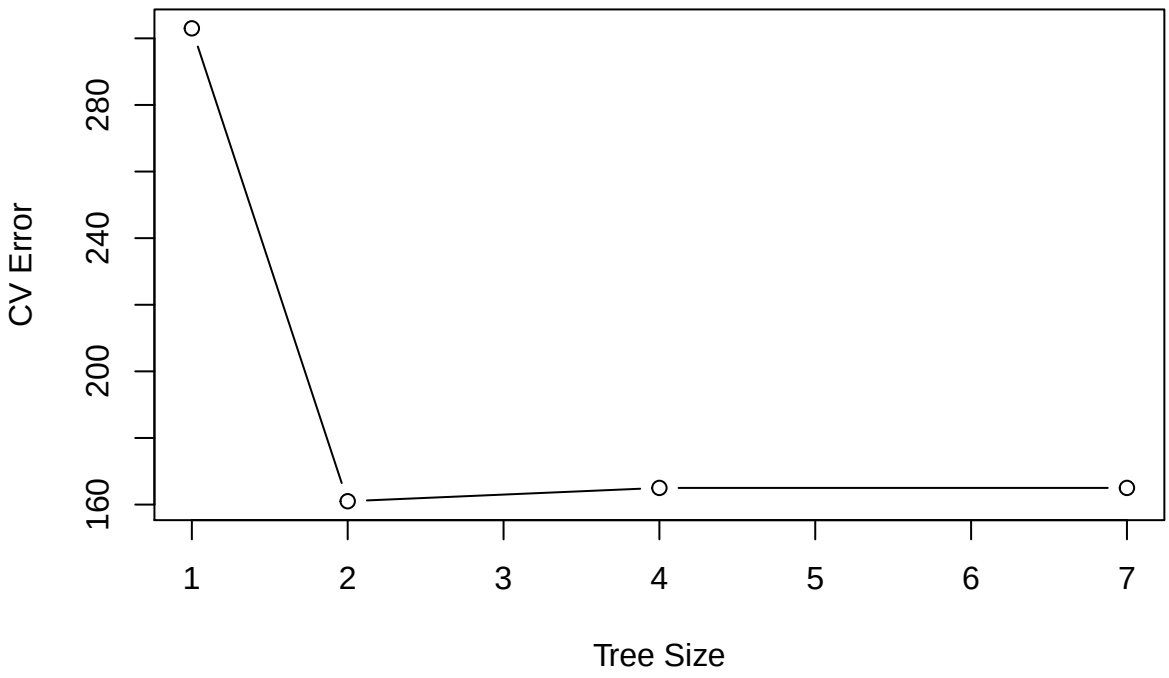
f).Apply the cv.tree() function to the training set in order to determine the optimal tree size.

```
cv_tree <- cv.tree(tree_model, FUN = prune.misclass)
cv_tree
```

```
## $size
## [1] 7 4 2 1
##
## $dev
## [1] 165 165 161 303
##
## $k
## [1]  -Inf   0.0   9.5 152.0
##
## $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

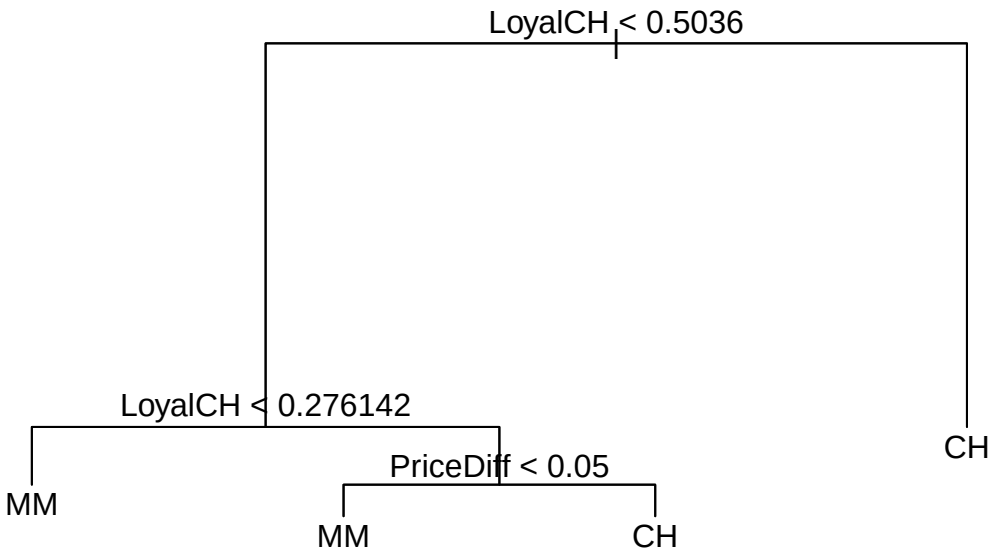
```
plot(cv_tree$size, cv_tree$dev, type = "b", xlab = "Tree Size", ylab = "CV Error")
```




h). Which tree size corresponds to the lowest cross-validated classification error rate on the y-axis? *The tree of size four has the lowest classification error rate*

i).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

```
pruned_tree <- prune.misclass(tree_model, best = 4)
plot(pruned_tree)
text(pruned_tree, pretty = 0)
```




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

```
summary(pruned_tree)
```

```
##
## Classification tree:
## snip.tree(tree = tree_model, nodes = 3:4)
## Variables actually used in tree construction:
## [1] "LoyalCH"   "PriceDiff"
## Number of terminal nodes:  4
## Residual mean deviance:  0.8364 = 665.7 / 796
## Misclassification error rate: 0.165 = 132 / 800
```

*According to the summary, the pruned tree error rate is 16.5% This is the same error rate as for the unpruned tree*

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

```
pruned_tree_pred = predict(pruned_tree,oj_test, type = "class")
pruned_test_error <- mean(oj_test$Purchase != pruned_tree_pred)
pruned_test_error
```

```
## [1] 0.2185185
```

```
table(tree_pred, oj_test$Purchase)
```

```
##
## tree_pred  CH  MM
##        CH 141  44
##        MM  15  70
```

*As we see from the pruned_test_error value and confusion matrix, the two test errors are the same*

#Problem 4 Bagging and Boosting. Problem 10 at page 334-335 of ISL. 10. We now use boosting to predict Salary in the Hitters data set.

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

```
data("Hitters")

full_dataset <- Hitters[!is.na(Hitters$Salary), ]
full_dataset$Salary <- log10(full_dataset$Salary)
```

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

```
train <- 1:200
test <- -train
hitters_train <- full_dataset[train,]
hitters_test <- full_dataset[test,]
```

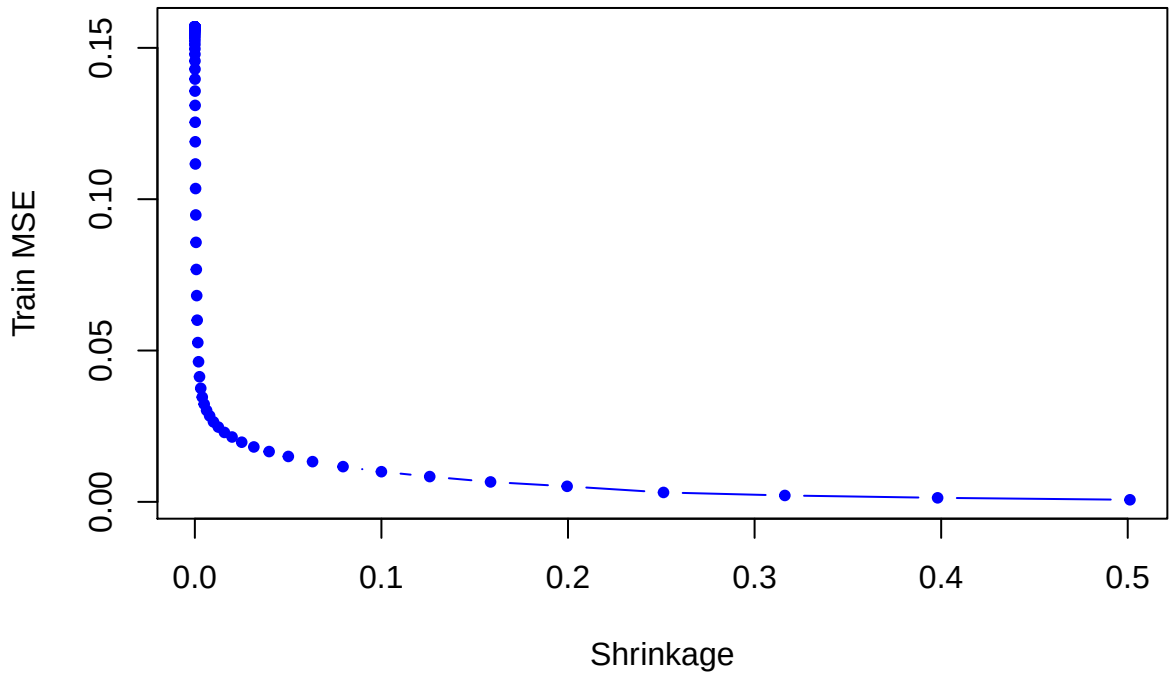
c).Performing boosting on the training set with 1000 trees for a range of values of the shrinkage parameter lambda. Produce a plot with different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

**Perform boosting**

```
pows <-  seq(-9, -0.3, by=0.1)
shrinkage <-  10 ^ pows
len_shrinkage <- length(shrinkage)
train_mse <- rep(NA, len_shrinkage)
test_mse <- rep(NA, len_shrinkage)
for (i in seq(len_shrinkage)) {
  set.seed(1)
  boost <- gbm(Salary ~., data = hitters_train, distribution = "gaussian",
               n.trees = 1000, shrinkage = shrinkage[i], verbose = FALSE)
  boost_pred_train <- predict(boost, hitters_train, n.trees = 1000)
  boost_pred_test <- predict(boost, hitters_test, n.trees = 1000)
  train_mse[i] <- mean((boost_pred_train - hitters_train$Salary)^2)
  test_mse[i] <- mean((boost_pred_test - hitters_test$Salary)^2)

}
```

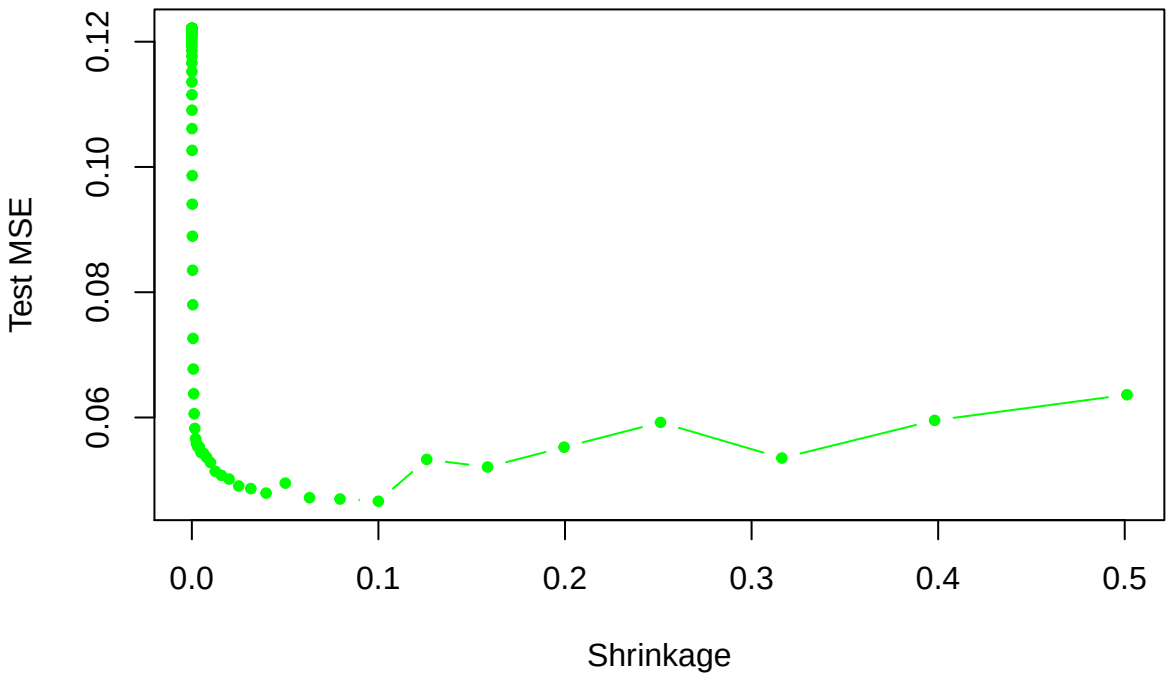
**Create the plot**

```
plot(shrinkage, train_mse, type="b",
     xlab="Shrinkage", ylab="Train MSE",
     col="blue", pch=20)
```




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

```
plot(shrinkage, test_mse, type="b",
     xlab="Shrinkage", ylab="Test MSE",
     col="green", pch=20)
```




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

**Calculate the boosting minimal test mse**

```
min_test_mse <- min(test_mse)
min_test_mse
```

```
## [1] 0.04662511
```

*The minimum test mse for boosting is 0.04662511*

**Determine the best shrinkage lambda from the vector of lambdas to train the final gbm**

```
best_shrinkage <- shrinkage[which.min(test_mse)]
best_shrinkage
```

```
## [1] 0.1
```

*Min test error obtained for lambda=0.1*

**Now let us train a linear model**

```
lm_model <- lm(Salary ~. , data = hitters_train)
lm_pred <- predict(lm_model, hitters_test)
mean_lm_test_mse <- mean((hitters_test$Salary - lm_pred)^2)
mean_lm_test_mse
```

```
## [1] 0.09275847
```

*The minimum test mse for linear regression is 0.09275847*

**Let's create the matrices necessary to fit ridge and lasso models**

```
x <- model.matrix(Salary~., hitters_train)
x_test <- model.matrix(Salary ~ . , hitters_test)
y <- hitters_train$Salary
```

**Now let us train a Ridge Model**

```
set.seed(1)
cv_ridge <- cv.glmnet(x = x, y = y, alpha = 0)
best_lambda_ridge <- cv_ridge$lambda.min
ridge_model <- glmnet(x, y, alpha = 0, lambda = best_lambda_ridge)
ridge_pred <- predict(ridge_model, s = best_lambda_ridge, x_test)
ridge_mse <- mean((ridge_pred - hitters_test$Salary)^2)
ridge_mse
```

```
## [1] 0.08617622
```

*The test mse for ridge regression is 0.08617622*

**Now let us train a Lasso Model**

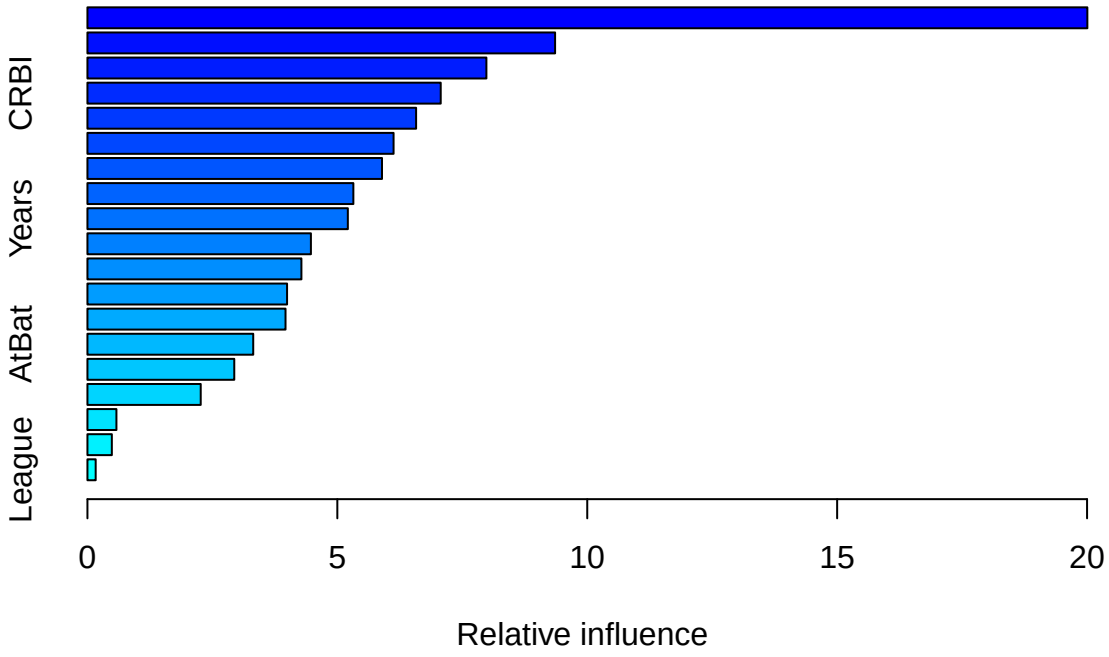
```
set.seed(1)
cv_lasso <- cv.glmnet(x = x, y = y, alpha = 1)
best_lambda_lasso <- cv_lasso$lambda.min
lasso_model <- glmnet(x, y, alpha = 1, lambda = best_lambda_lasso)
lasso_pred <- predict(lasso_model, s = best_lambda_lasso, x_test)
lasso_mse <- mean((lasso_pred - hitters_test$Salary)^2)
lasso_mse
```

```
## [1] 0.08882712
```

*The lasso MSE is 0.08882712 In conclusion, the test MSE obtained using gradient boosting is around half of the best two regression methods from the previous chapters (lasso and ridge.) Surprisingly, lasso performs worse than ridge. The Linear Model is predictably last*

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

```
set.seed(1)
best_boost <- gbm(Salary ~., data = hitters_train, distribution = "gaussian",
                  n.trees = 1000, shrinkage=best_shrinkage)
summary(best_boost)
```




```
##                 var    rel.inf
## CAtBat       CAtBat 20.0059611
## PutOuts     PutOuts  9.3563411
## CRuns         CRuns  7.9820668
## CRBI           CRBI  7.0681649
## Walks         Walks  6.5760083
## CHits         CHits  6.1247660
## CHmRun       CHmRun  5.8935313
## Assists     Assists  5.3203916
## Years         Years  5.2100551
## CWalks       CWalks  4.4709356
## RBI             RBI  4.2812231
## Hits           Hits  3.9941017
## Runs           Runs  3.9628594
## AtBat         AtBat  3.3167653
## HmRun         HmRun  2.9373769
## Errors       Errors  2.2664392
## Division   Division  0.5802059
## NewLeague NewLeague  0.4879580
## League       League  0.1648489
```

*Based on the summary, CAtBat is by far the most important variable.*

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

```
set.seed(1)
bag_model <- randomForest(Salary ~.,
                          data = hitters_train, mtry = 19, importance=TRUE)

bag_predict <- predict(bag_model, hitters_test)
bag_test_error <- mean((bag_predict - hitters_test$Salary)^2)
bag_test_error
```

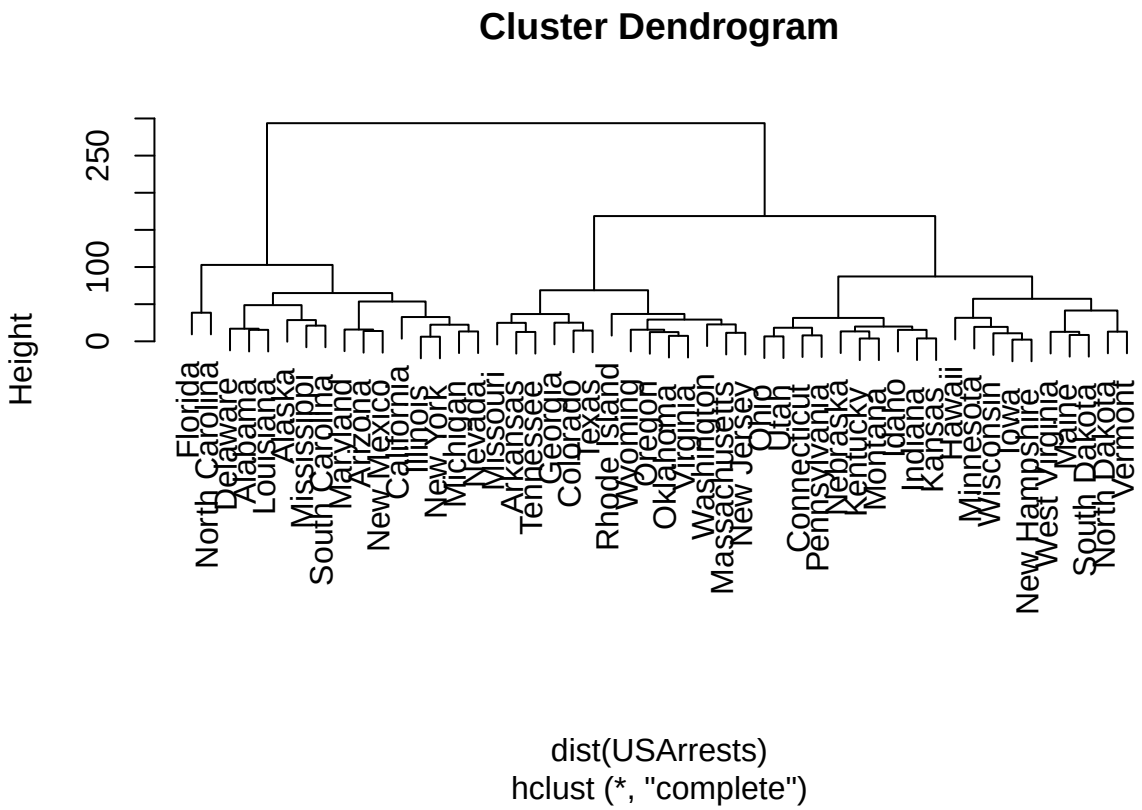
```
## [1] 0.04342996
```

*The bag test error is 0.04342996*

## Problem 5 Hierarchical Clustering. Problem 9 at page 416-417 of ISL

a).Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states

```
set.seed(1)
hc_complete = hclust(dist(USArrests), method="complete")
plot(hc_complete)
```




b).Cut the dendrogram at a height that results in three distinct clusters. Which states belong to which clusters?

```
table(cutree(hc_complete, 3))
```

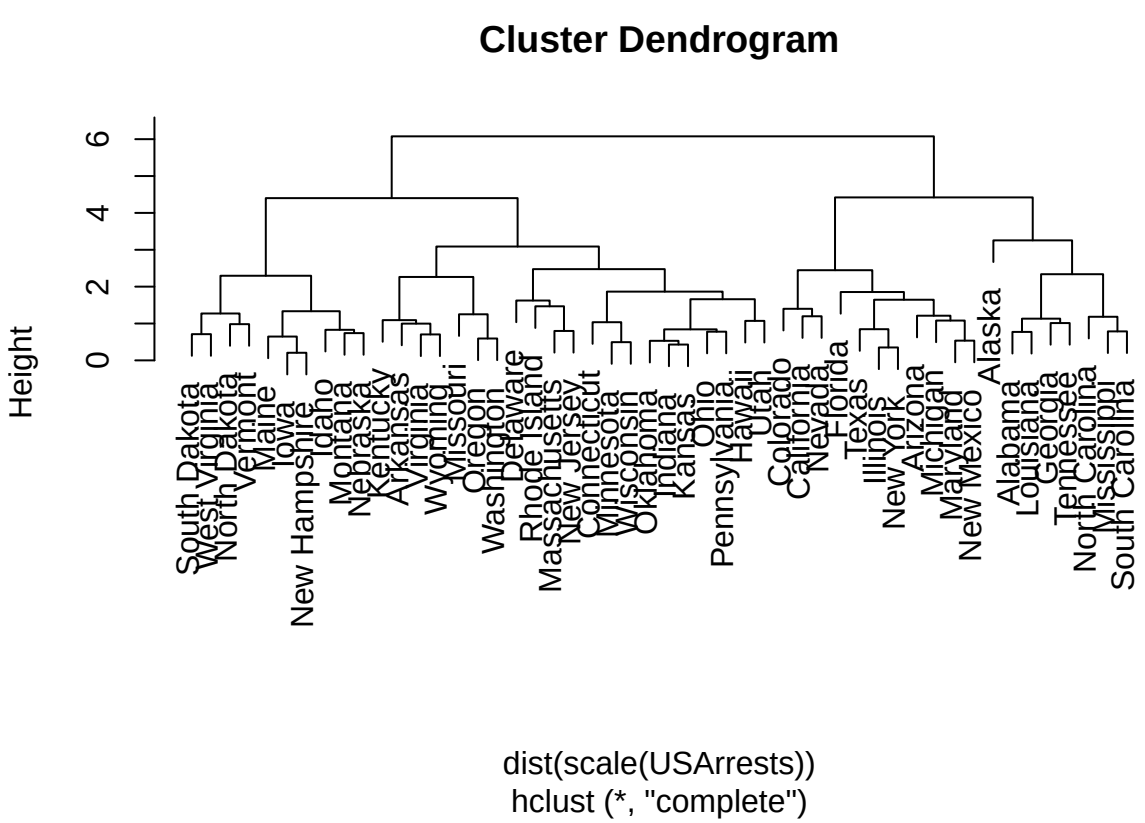
```
##
##  1  2  3
## 16 14 20
```

```
cutree(hc_complete, 3)
```

```
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              1              2              1
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              3              1              1              2
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              3              3              1              3              3
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              3              3              1              3              1
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              2              1              3              1              2
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              3              3              1              3              2
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              1              1              1              3              3
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##              2              2              3              2              1
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              3              2              2              3              3
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              2              2              3              3              2
```

c).Hierarchically cluster the states using complete linkage and Euclidean distance, after scaling the variables to have standard deviation one.

```
hc_scale <- hclust(dist(scale(USArrests)), method = "complete")
plot(hc_scale)
```




*d). What effect does scaling the variables have on the hierarchical clustering obtained? In your opinion, should the variables be scaled before the inter-observation dissimilarities are computed? Provide a justification for your answer. Scaling the data causes the size of the third clster to increase a lot to include around one third of all the data. In my opinion, variables in clustering algorithms should always be pre-scaled. This is because if each variable has a different scale, certain data points would gravitate towards a cluster.*