Assignment 3 - Statistical Modelling

Mughundhan Chandrasekar

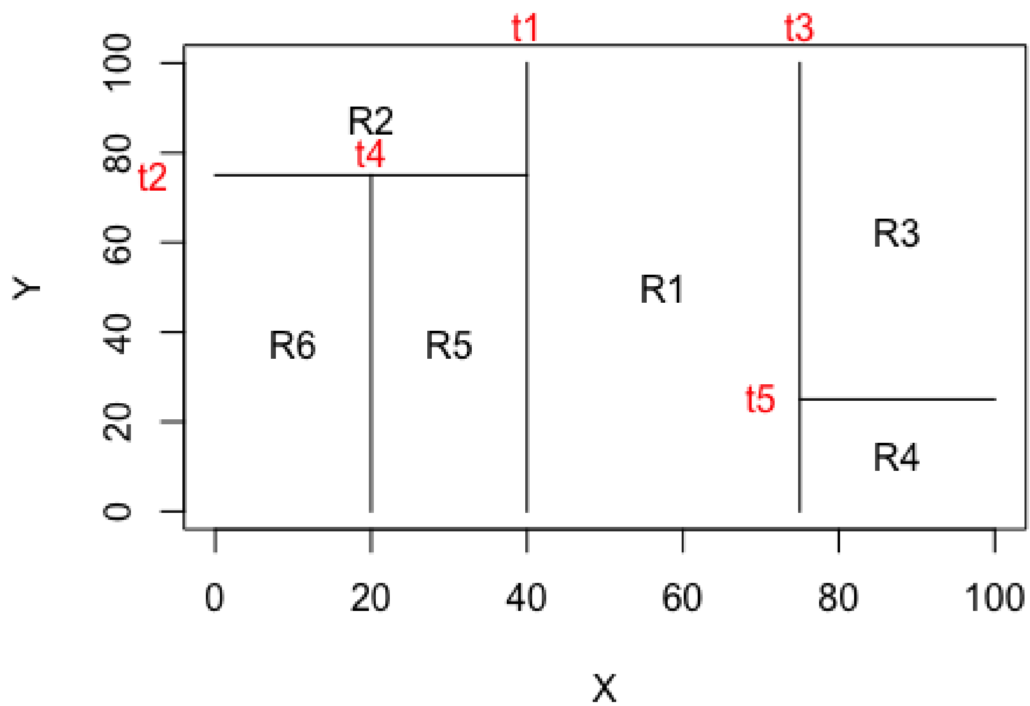
11/6/2017

Question 1.

Draw an example (of your own invention) of a partition of two-dimensional feature space that could result from recursive binary splitting. Your example should contain at least six regions. Draw a decision tree corresponding to this partition. Be sure to label all aspects of your figures, including the regions R1, R2,..., the cutpoints t1, t2,..., and so forth.

Solution: - t1, t2 ..... t5 -> Cut points - R1, R2 ..... R6 -> Regions

```
rm(list=ls())
par(xpd = NA)
plot(NA, NA, type = "n", xlim = c(0,100), ylim = c(0,100), xlab = "X", ylab =
"Y")
# t1: x = 40; (40, 0) (40, 100)
lines(x = c(40,40), y = c(0,100))
text(x = 40, y = 108, labels = c("t1"), col = "red")
# t2: y = 75; (0, 75) (40, 75)
lines(x = c(0,40), y = c(75,75))
text(x = -8, y = 75, labels = c("t2"), col = "red")
# t3: x = 75; (75,0) (75, 100)
lines(x = c(75,75), y = c(0,100))
text(x = 75, y = 108, labels = c("t3"), col = "red")
# t4: x = 20; (20,0) (20, 75)
lines(x = c(20,20), y = c(0,75))
text(x = 20, y = 80, labels = c("t4"), col = "red")
# t5: y=25; (75,25) (100,25)
lines(x = c(75,100), y = c(25,25))
text(x = 70, y = 25, labels = c("t5"), col = "red")

text(x = (40+75)/2, y = 50, labels = c("R1"))
text(x = 20, y = (100+75)/2, labels = c("R2"))
text(x = (75+100)/2, y = (100+25)/2, labels = c("R3"))
text(x = (75+100)/2, y = 25/2, labels = c("R4"))
text(x = 30, y = 75/2, labels = c("R5"))
text(x = 10, y = 75/2, labels = c("R6"))
```

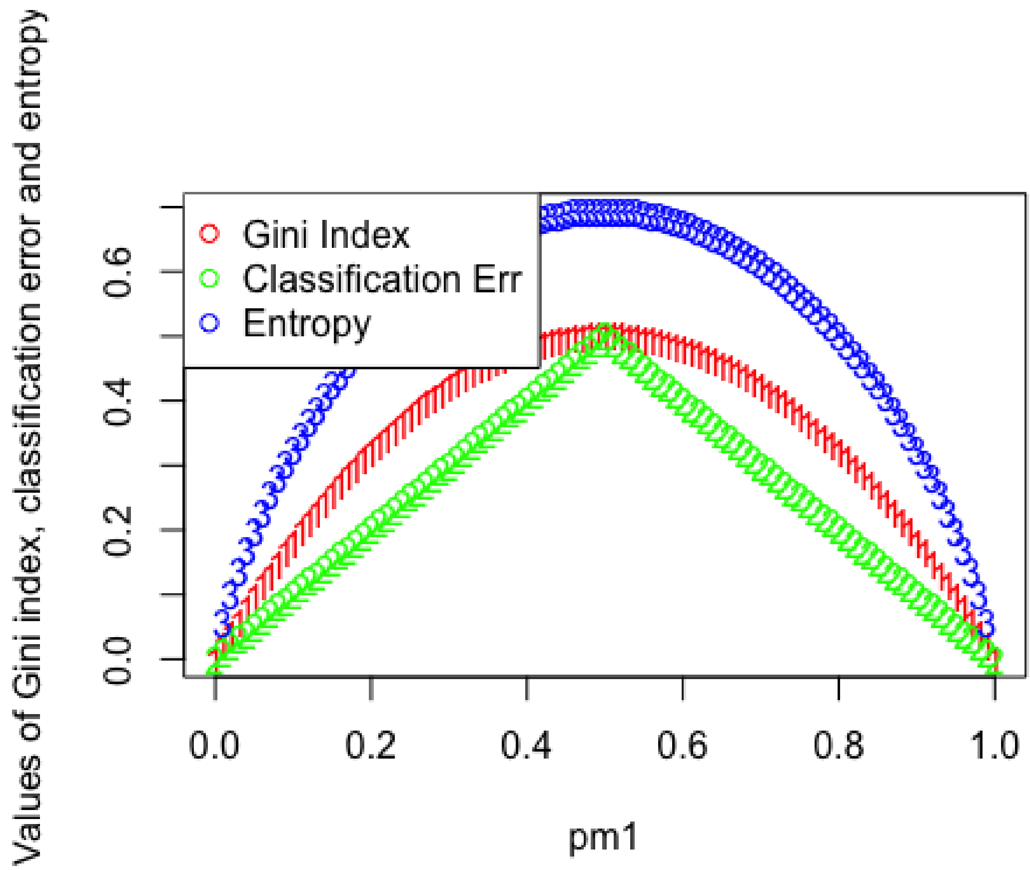


Question 2.

Consider the Gini index, classification error, and entropy in a simple classification setting with two classes. Create a single plot that displays each of these quantities as a function of ??pm1. The xaxis should display ??pm1, ranging from 0 to 1, and the y-axis should display the value of the ***Gini index***, ***classification error***, and ***entropy***.

Solution:

```
pm1 <- seq(0, 1, 0.01)
gini.index <- 2 * pm1 * (1 - pm1)
class.error <- 1 - pmax(pm1, 1 - pm1)
cross.entropy <- - (pm1 * log(pm1) + (1 - pm1) * log(1 - pm1))
Values <- cbind(gini.index, class.error, cross.entropy)
matplot(pm1, Values, col = c("red", "green", "blue"), ylab = "Values of Gini
index, classification error and entropy")

legend("topleft",
       inset=-.01,
       legend=c("Gini Index", "Classification Err", "Entropy"),
       pch=1,
       col = c("red", "green", "blue"),
       horiz=FALSE)
```



Question 3.

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.

Question 3.a.

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

Solution:

```
rm(list=ls())
library(ISLR)
library(plotrix)
set.seed(1)
train <- sample(1:nrow(Carseats), nrow(Carseats) / 2)
Carseats.train <- Carseats[train, ]
Carseats.test <- Carseats[-train, ]
head(Carseats)
```

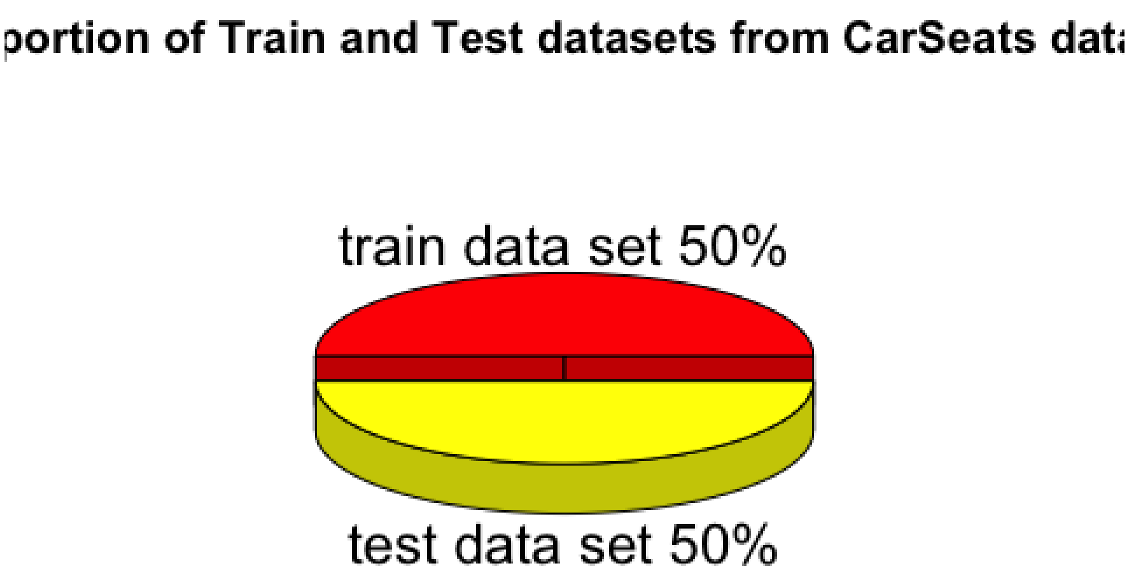
```
##   Sales CompPrice Income Advertising Population Price ShelveLoc Age
## 1  9.50       138     73          11        276   120       Bad  42
## 2 11.22       111     48          16        260    83      Good  65
```

```
## 3 10.06       113     35           10         269     80     Medium   59
## 4  7.40       117    100            4         466     97     Medium   55
## 5  4.15       141     64            3         340    128        Bad   38
## 6 10.81       124    113           13         501     72        Bad   78
##   Education Urban  US
## 1        17   Yes Yes
## 2        10   Yes Yes
## 3        12   Yes Yes
## 4        14   Yes Yes
## 5        13   Yes  No
## 6        16    No Yes
```

```
prop_fdata <- cbind(nrow(Carseats.train), nrow(Carseats.test))
pie_label <- cbind("train data set 50%", "test data set 50%")
prop_fdata
```

```
##      [,1] [,2]
## [1,]  200  200
```

```
pie3D(prop_fdata, labels = pie_label,
      explode = 0.1,
    main="Proportion of Train and Test datasets from CarSeats dataset",
    col = c("Red", "Yellow"))
```



```r
test.size <- nrow(Carseats.test)
train.size <- nrow(Carseats.train)
cbind(test.size, train.size)
```

```
##      test.size train.size
## [1,]       200        200
```

```r
rm(test.size, train.size)
```

Observation:

1. Number of observations in original data set is 400
2. Number of observations in train data set is 200
3. Number of observations in test data set is 200

Question 3.b.

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

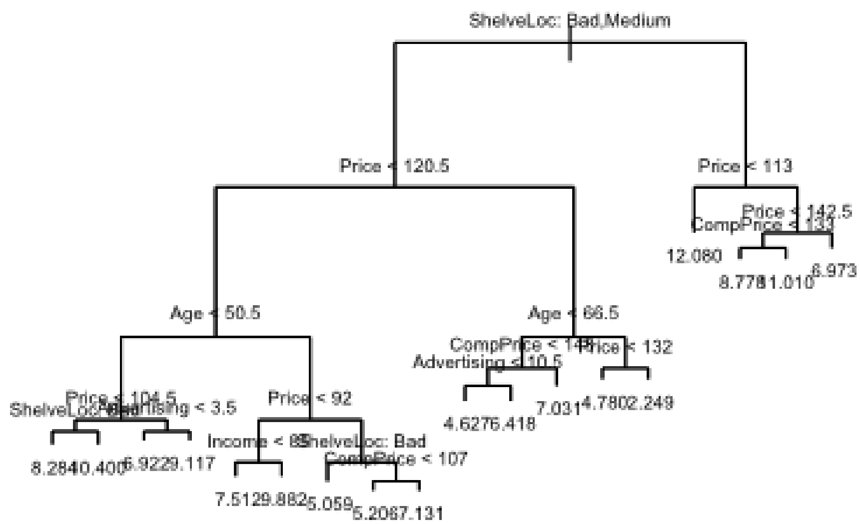
Solution: The tree plot is shown below followed by the interpretation

```r
library(tree)
library(readr)
library(dplyr)
library(party)
library(rpart)
library(rpart.plot)
library(ROCR)


# tree.carseats <- rpart(Sales ~ .,
#             data = Carseats.train,
#             method = "anova")
# rpart.plot(tree.carseats)
#
# rm(tree.carseats)


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" "Income"
## [6] "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
```

```r
plot(tree.carseats)
text(tree.carseats, pretty = 0, cex = 0.5)
```



Solution: The test MSE can be computed as shown below.

```r
yhat <- predict(tree.carseats, newdata = Carseats.test)
round(mean((yhat - Carseats.test$Sales)^2),4)

## [1] 4.1489
```

Observation: The Test MSE is found to be 4.1489, which is approximately 4.15.

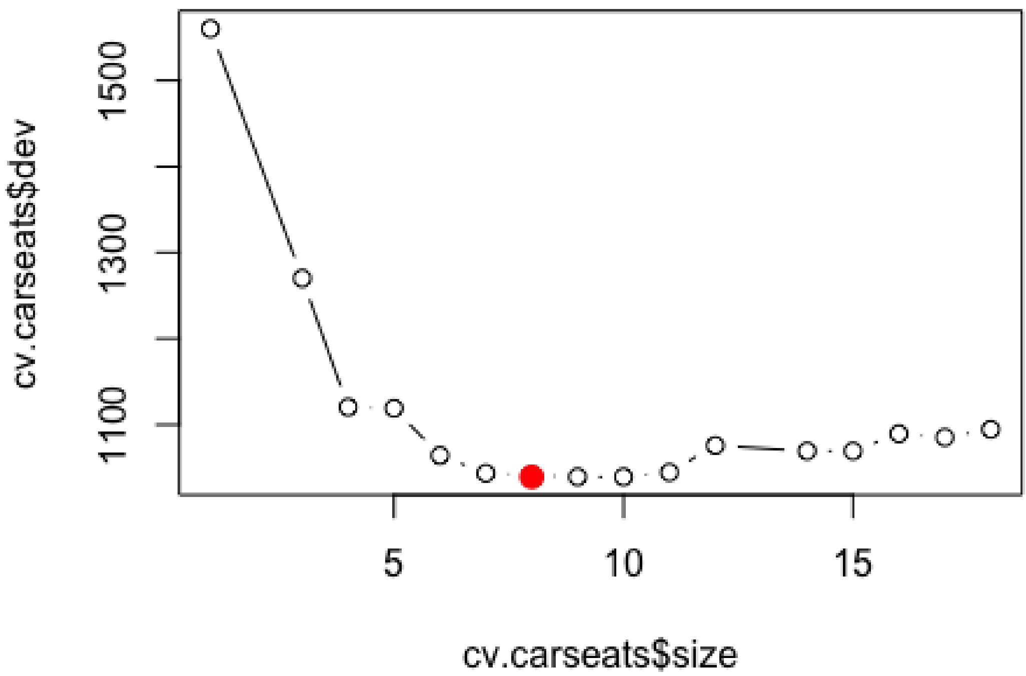
Question 3.c.

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

Solution:

```r
cv.carseats <- cv.tree(tree.carseats)
plot(cv.carseats$size, cv.carseats$dev, type = "b")
```

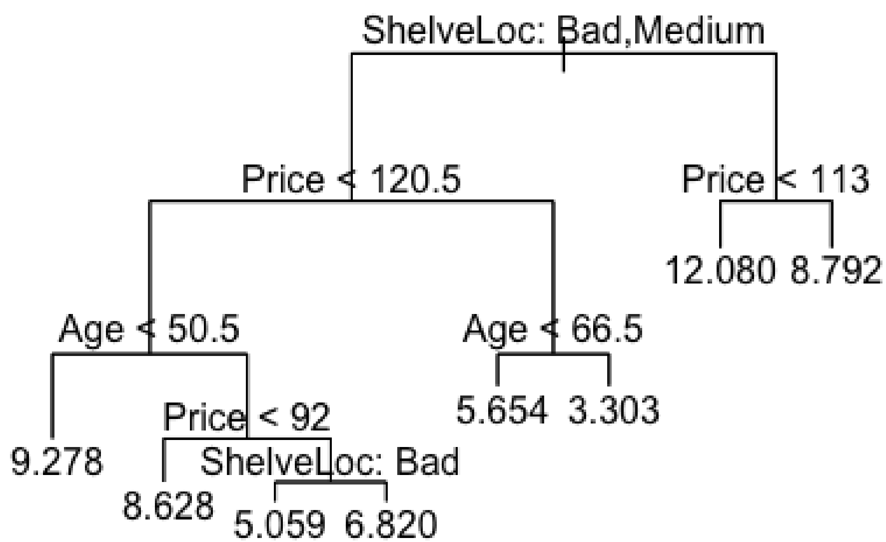
```r
tree.min <- which.min(cv.carseats$dev)
points(tree.min, cv.carseats$dev[tree.min], col = "red", cex = 2, pch = 20)
```




```r
tree.min #Size of tree selected by cross-validation

## [1] 8
```

Observation: The size of the tree selected by cross-validation for the pruning purpose is found to be 8.

```r
tree.carseats <- tree(Sales ~ ., data = Carseats.train)

prune.carseats <- prune.tree(tree.carseats, best = 8)
plot(prune.carseats)
text(prune.carseats, pretty = 0)
```




```
yhat <- predict(prune.carseats, newdata = Carseats.test)
round(mean((yhat - Carseats.test$Sales)^2), 4)

## [1] 5.0909
```

Observation: On pruning it is found that the Test MSE increases. On pruning the Test MSE is found to be 5.0909, which is approximately 5.10.

Question 3.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.

Solution:

```
library(randomForest)
bag.carseats <- randomForest(Sales ~ ., data = Carseats.train, mtry = 10,
ntree = 500, importance = TRUE)
yhat.bag <- predict(bag.carseats, newdata = Carseats.test)
round(mean((yhat.bag - Carseats.test$Sales)^2), 4)

## [1] 2.6044

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
```

Observation:

1. We can see that ***Bagging reduces the Test MSE***. The Test MSE is found to be 2.5799, which is approximately 2.58.
2. With the help of the ***importance function***, we can determine that ***Price*** and ***ShelveLoc*** are the two most important variables.

Question 3.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.

Solution:

```
rf.carseats <- randomForest(Sales ~ ., data = Carseats.train, mtry = 3, ntree
= 500, importance = TRUE)
yhat.rf <- predict(rf.carseats, newdata = Carseats.test)
round(mean((yhat.rf - Carseats.test$Sales)^2), 4)
```

```
## [1] 3.2961
```

```
importance(rf.carseats)
```

```
##                %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
```

Observation:

1. We can see that ***Test MSE on performing Random Forest is higher than when we perform Bagging***. The Test MSE is found to be 3.3298, which is approximately 3.33.
2. With the help of the ***importance function***, we can determine that ***Price*** and ***ShelveLoc*** are the two most important variables.

Question 4

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

Question 4.a.

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

```
str(Hitters$Salary)

##  num [1:322] NA 475 480 500 91.5 750 70 100 75 1100 ...

Hitters <- na.omit(Hitters)
Hitters$Salary <- log(Hitters$Salary)
str(Hitters$Salary)

##  num [1:263] 6.16 6.17 6.21 4.52 6.62 ...
```

Observation:

1. The observations for whom the salary information is unknown is removed using ***na.omit function***
2. log-transform the salaries using ***log function***

Question 4.b.

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

Solution:

```
train <- 1:200
Hitters.train <- Hitters[train, ]
Hitters.test <- Hitters[-train, ]

nrow(Hitters)

## [1] 263

nrow(Hitters.train)

## [1] 200

nrow(Hitters.test)

## [1] 63
```

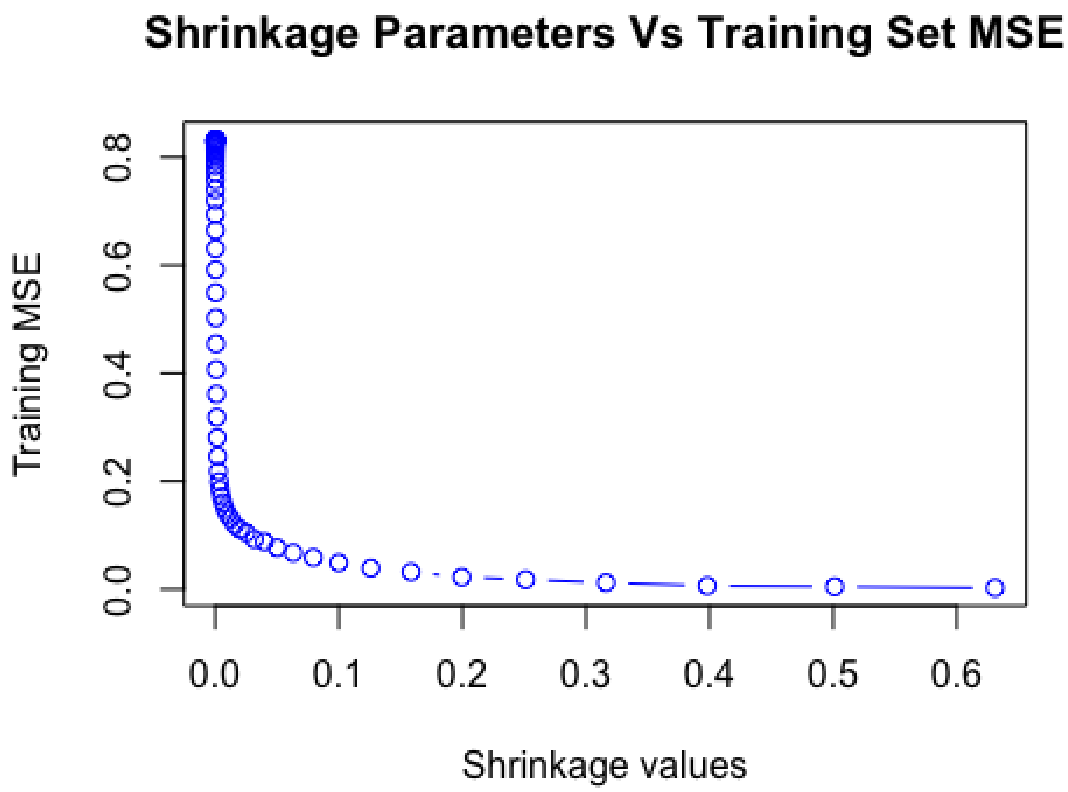
Observation:

1. Number of observations in Hitters dataset: 263
2. Number of observations in train dataset: 200
3. Number of observations in test dataset: 63

Question 4.c.

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

```
library(gbm)
set.seed(1)
pows <- seq(-10, -0.2, by = 0.1)
lambdas <- 10^pows
train.err <- 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])
    pred.train <- predict(boost.hitters, Hitters.train, n.trees = 1000)
    train.err[i] <- mean((pred.train - Hitters.train$Salary)^2)
}
plot(lambdas, train.err, type = "b", xlab = "Shrinkage values", ylab =
"Training MSE", col="blue", main = "Shrinkage Parameters Vs Training Set
MSE")
```




```
round(min(train.err), 4)

## [1] 0.0023

round(lambdas[which.min(train.err)], 4)

## [1] 0.631
```

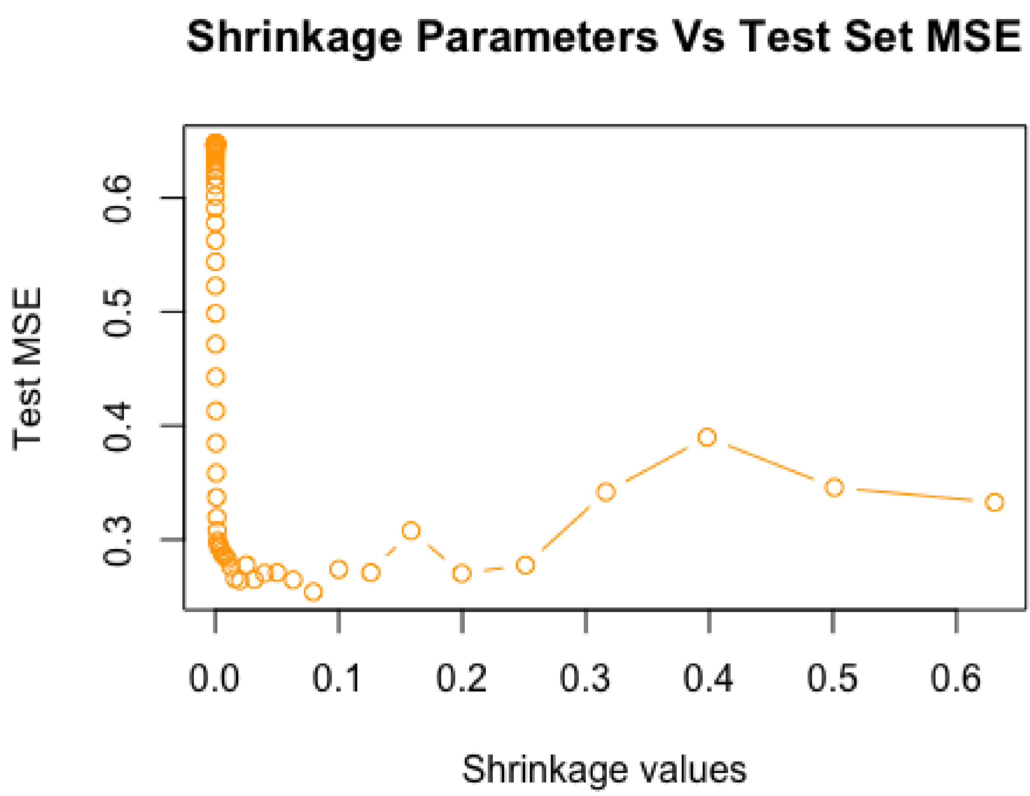
Observation: We can infer that the minimum train MSE is 0.0023, and is obtained for ??=0.6309

Question 4.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)
test.err <- 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])
    yhat <- predict(boost.hitters, Hitters.test, n.trees = 1000)
    test.err[i] <- mean((yhat - Hitters.test$Salary)^2)
}
```

```r
plot(lambdas, test.err, type = "b", xlab = "Shrinkage values", ylab = "Test
MSE", col="orange", main = "Shrinkage Parameters Vs Test Set MSE")
```




```r
round(min(test.err), 4)

## [1] 0.254

round(lambdas[which.min(test.err)], 4)

## [1] 0.0794
```

Observation: We can infer that the minimum test MSE is 0.25, and is obtained for ??=0.079

Question 4.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.

```r
library(glmnet)

fit1 <- lm(Salary ~ ., data = Hitters.train)
pred1 <- predict(fit1, Hitters.test)
round(mean((pred1 - Hitters.test$Salary)^2), 4)

## [1] 0.4918
```

```
x <- model.matrix(Salary ~ ., data = Hitters.train)
x.test <- model.matrix(Salary ~ ., data = Hitters.test)
y <- Hitters.train$Salary
fit2 <- glmnet(x, y, alpha = 0)
pred2 <- predict(fit2, s = 0.01, newx = x.test)
round(mean((pred2 - Hitters.test$Salary)^2), 4)

## [1] 0.457
```

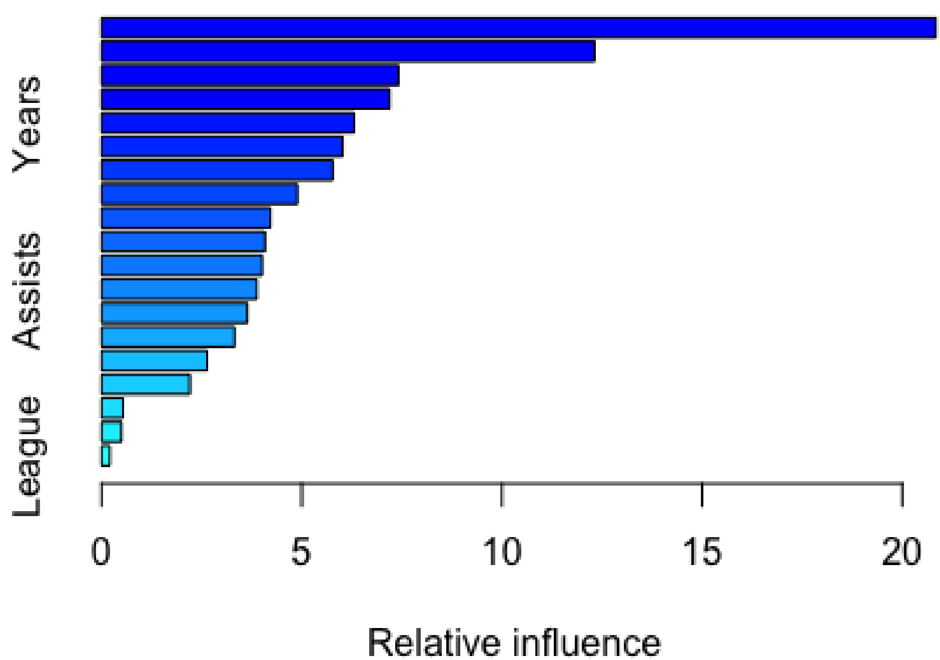
Observation:

1. The test MSE for boosting is lower than Linear Regression [0.4918].
2. The test MSE for boosting is lower than Ridge Regression [0.457].

Question 4.f.

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

```
library(gbm)
boost.hitters <- gbm(Salary ~ ., data = Hitters.train, distribution =
"gaussian", n.trees = 1000, shrinkage = lambdas[which.min(test.err)])
summary(boost.hitters)
```



```
##                   var    rel.inf
## CAtBat       CAtBat 20.8404970
## CRBI           CRBI 12.3158959
## Walks         Walks  7.4186037
## PutOuts     PutOuts  7.1958539
## Years         Years  6.3104535
## CWalks       CWalks  6.0221656
## CHmRun       CHmRun  5.7759763
## CHits         CHits  4.8914360
## AtBat         AtBat  4.2187460
## RBI             RBI  4.0812410
## Hits           Hits  4.0117255
## Assists     Assists  3.8786634
## HmRun         HmRun  3.6386178
## CRuns         CRuns  3.3230296
## Errors       Errors  2.6369128
## Runs           Runs  2.2048386
## Division   Division  0.5347342
## NewLeague NewLeague  0.4943540
## League       League  0.2062551
```

Observation: We can infer that ***CAtBat*** is the most important predictor in the boosted model.

Question 4.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 = 500)
yhat.bag <- predict(bag.hitters, newdata = Hitters.test)
round(mean((yhat.bag - Hitters.test$Salary)^2), 4)

## [1] 0.2314
```

Observation: The test MSE for bagging [0.2314] is slightly lower than test MSE for boosting.