Homework 4&5

Ma Jingchun, 2020111235

Homework 4

```
library(ISLR2)
library(tree)
library(gbm)
library(glmnet)
library(randomForest)
```

1.This problem involves the OJ data set which is part of the ISLR2 package.

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

```
set.seed(1)
train = sample(1:nrow(OJ), 800)
oj.train = OJ[train,]
oj.test = OJ[-train,]
oj.train.y = OJ[train,"Purchase"]
oj.test.y = OJ[-train,"Purchase"]
```

b. Fit a tree to the training data, with Purchase as the response and the other variables 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 ~ ., OJ, subset=train)
summary(tree.oj)
```

```
##
## Classification tree:
## tree(formula = Purchase ~ ., data = OJ, subset = train)
## Variables actually used in tree construction:
## [1] "LoyalCH"       "PriceDiff"     "SpecialCH"     "ListPriceDiff"
## [5] "PctDiscMM"
## Number of terminal nodes:  9
## Residual mean deviance:  0.7432 = 587.8 / 791
## Misclassification error rate: 0.1588 = 127 / 800
```

use 5 predictors to split the tree.

training error rate: Misclassification rate: 15.88%

terminal nodes: 9

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
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 800 1073.00 CH ( 0.60625 0.39375 )
##    2) LoyalCH < 0.5036 365  441.60 MM ( 0.29315 0.70685 )
##      4) LoyalCH < 0.280875 177  140.50 MM ( 0.13559 0.86441 )
##        8) LoyalCH < 0.0356415 59   10.14 MM ( 0.01695 0.98305 ) *
##        9) LoyalCH > 0.0356415 118  116.40 MM ( 0.19492 0.80508 ) *
##      5) LoyalCH > 0.280875 188  258.00 MM ( 0.44149 0.55851 )
##       10) PriceDiff < 0.05 79   84.79 MM ( 0.22785 0.77215 )
##         20) SpecialCH < 0.5 64   51.98 MM ( 0.14062 0.85938 ) *
##         21) SpecialCH > 0.5 15   20.19 CH ( 0.60000 0.40000 ) *
##       11) PriceDiff > 0.05 109  147.00 CH ( 0.59633 0.40367 ) *
##    3) LoyalCH > 0.5036 435  337.90 CH ( 0.86897 0.13103 )
##      6) LoyalCH < 0.764572 174  201.00 CH ( 0.73563 0.26437 )
##       12) ListPriceDiff < 0.235 72   99.81 MM ( 0.50000 0.50000 )
##         24) PctDiscMM < 0.196196 55   73.14 CH ( 0.61818 0.38182 ) *
##         25) PctDiscMM > 0.196196 17   12.32 MM ( 0.11765 0.88235 ) *
##       13) ListPriceDiff > 0.235 102   65.43 CH ( 0.90196 0.09804 ) *
##      7) LoyalCH > 0.764572 261   91.20 CH ( 0.95785 0.04215 ) *
```

Interpret one terminal node:

“8) LoyalCH < 0.0356415 59 10.14 MM (0.01695 0.98305) *”

“9) LoyalCH > 0.0356415 118 116.40 MM (0.19492 0.80508) *”

for branch 8:

59 - Number observations in that branch

10.14 - Deviance

MM - Predicted class

(0.01695 0.98305) - (Prob CH, Prob MM)

for branch 9:

118 - Number observations in that branch

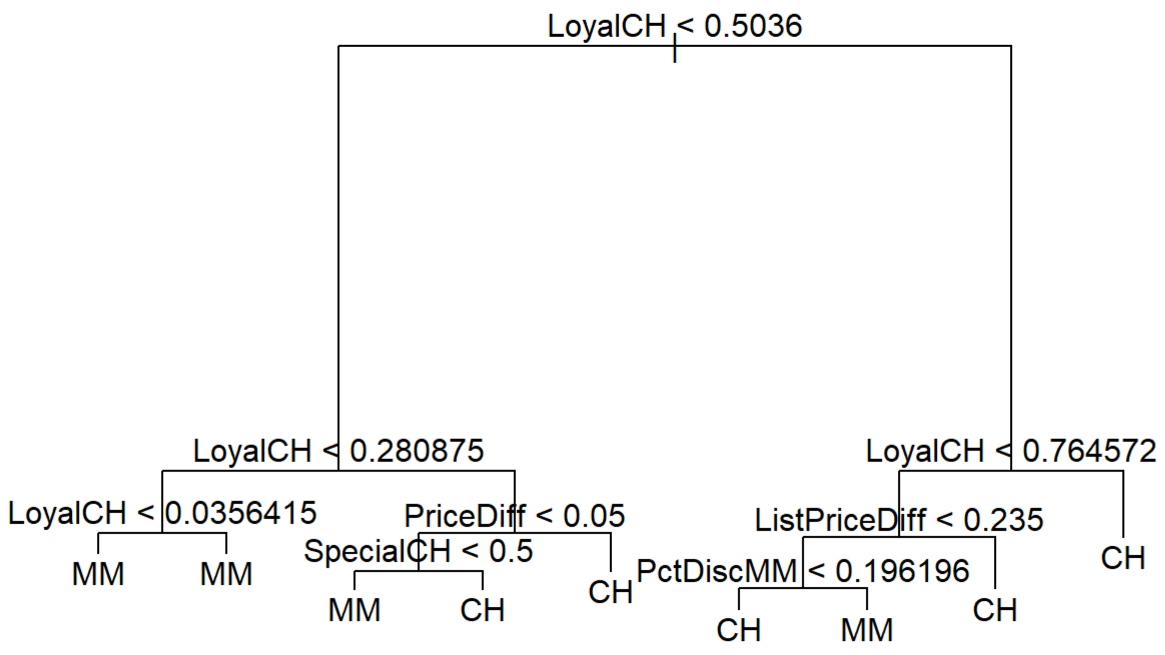
116.40 - Deviance

MM - Predicted class

(0.19492 0.80508) - (Prob CH, Prob MM)

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

```
{plot(tree.oj)
text(tree.oj, pretty=0)
}
```



There are 9 terminal nodes. And at least 1 redundant node.

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")

yhat = predict(tree.oj, newdata = OJ[-train ,], type = "class")
oj.test.y = OJ[-train, "Purchase"]
table(yhat, oj.test.y)
```

```
##      oj.test.y
## yhat   CH  MM
##    CH 160  38
##    MM   8  64
```

test error rate :(160+64)/270 = 0.8296296

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

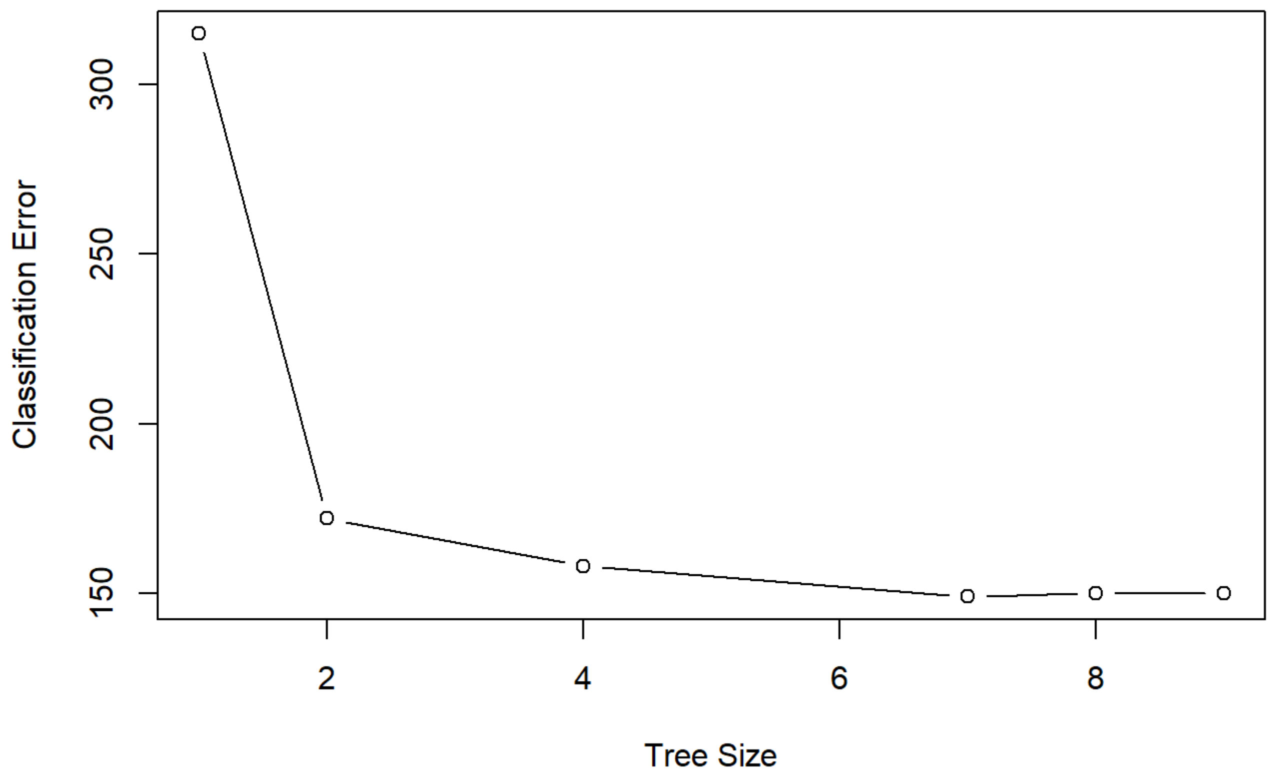
```
cv.oj=cv.tree(tree.oj, FUN=prune.misclass)
cv.oj
```

```
## $size
## [1] 9 8 7 4 2 1
##
## $dev
## [1] 150 150 149 158 172 315
##
## $k
## [1]       -Inf   0.000000   3.000000   4.333333  10.500000 151.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

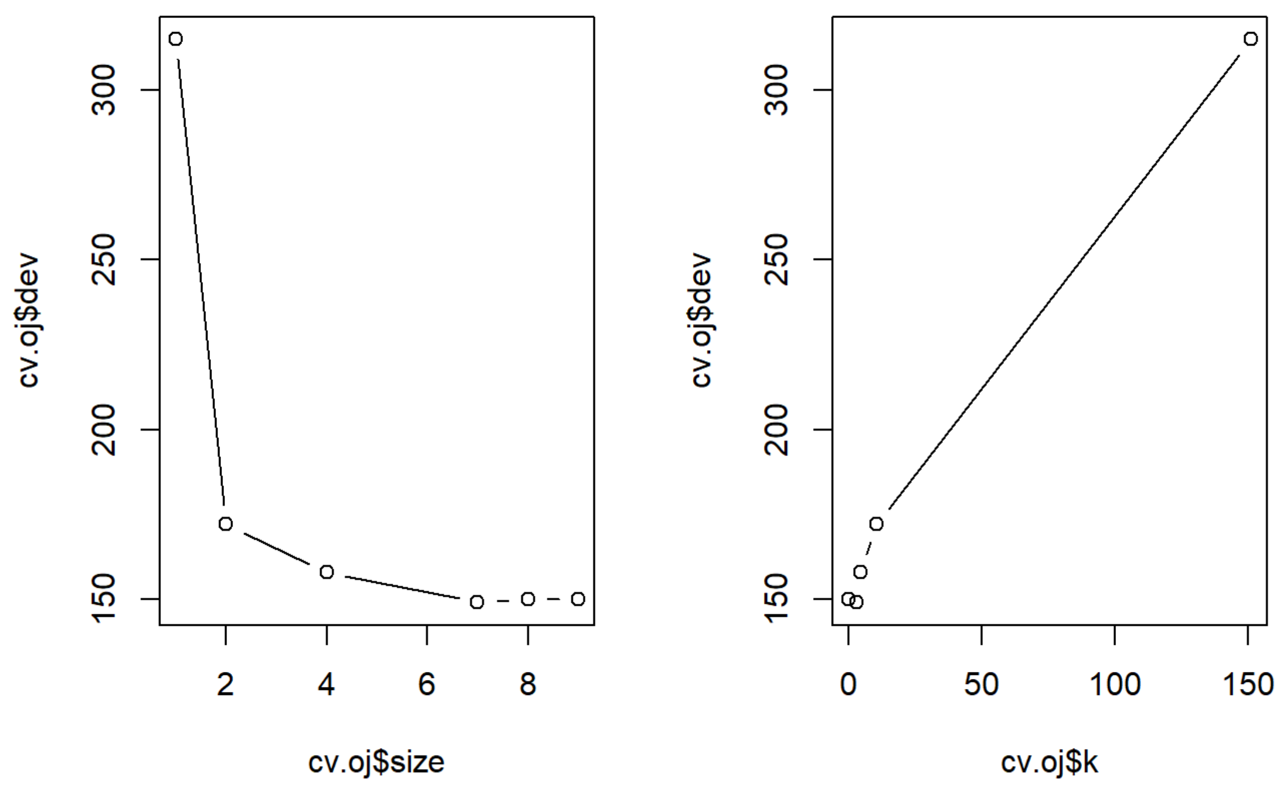
The optimal number of terminal nodes is 7.

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, type='b', xlab="Tree Size", ylab="Classification Error")
```



```
par(mfrow=c(1,2))
plot(cv.oj$size, cv.oj$dev, type="b")
plot(cv.oj$k,cv.oj$dev, type="b")
```

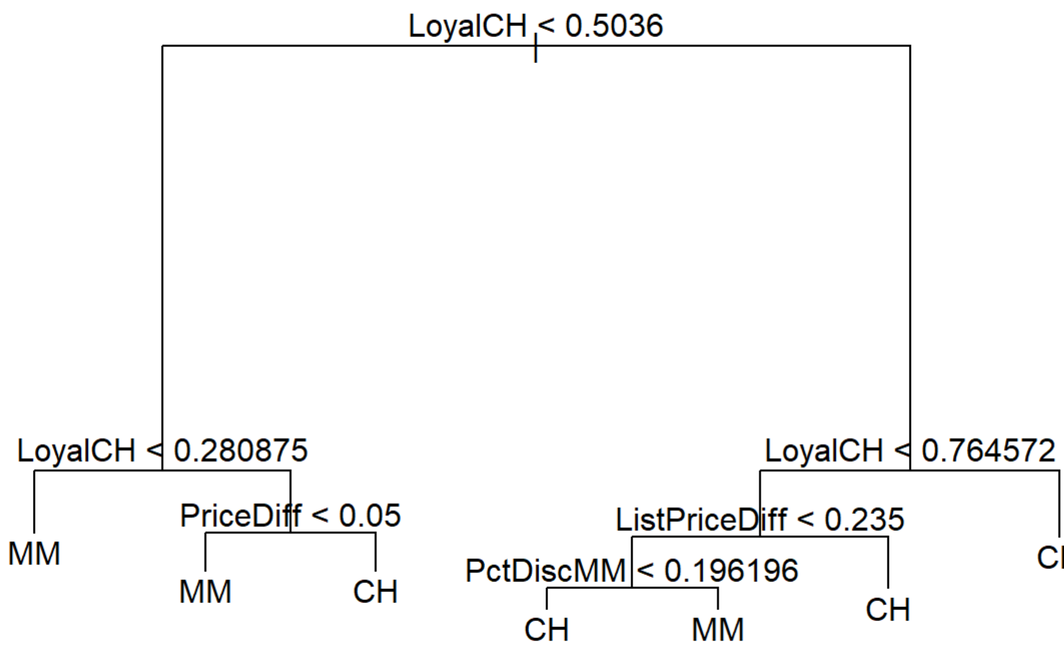


h. Which tree size corresponds to the lowest cross-validated classification error rate?

size 7 corresponds to the lowest cross-validated 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 with five terminal nodes.

```
prune.oj=prune.tree(tree.oj, best=7)
{plot(prune.oj)
text(prune.oj, pretty=0)
}
```



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

```
train.predict = predict(tree.oj, newdata = oj.train, type="class")

table(oj.train$Purchase, train.predict)
```

```
##     train.predict
##       CH  MM
##   CH 450  35
##   MM  92 223
```

training error rates = (450+223)/800 = 0.84125

```
train.pruned.predict = predict(prune.oj, newdata = oj.train, type="class")
table(oj.train$Purchase, train.pruned.predict)
```

```
##     train.pruned.predict
##       CH  MM
##   CH 441  44
##   MM  86 229
```

training error rates = (441+229)/800 = 0.8375

unpruned is better because of overfitting

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

```
test.pruned.predict = predict(prune.oj, newdata = oj.test, type="class")
table(oj.test$Purchase, test.pruned.predict)
```

```
##     test.pruned.predict
##       CH  MM
##   CH 160   8
##   MM  36  66
```

test error rates = (160 + 66)/270 = 0.837037 > 0.8296296

pruned gives a better test error rate.

2. 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.

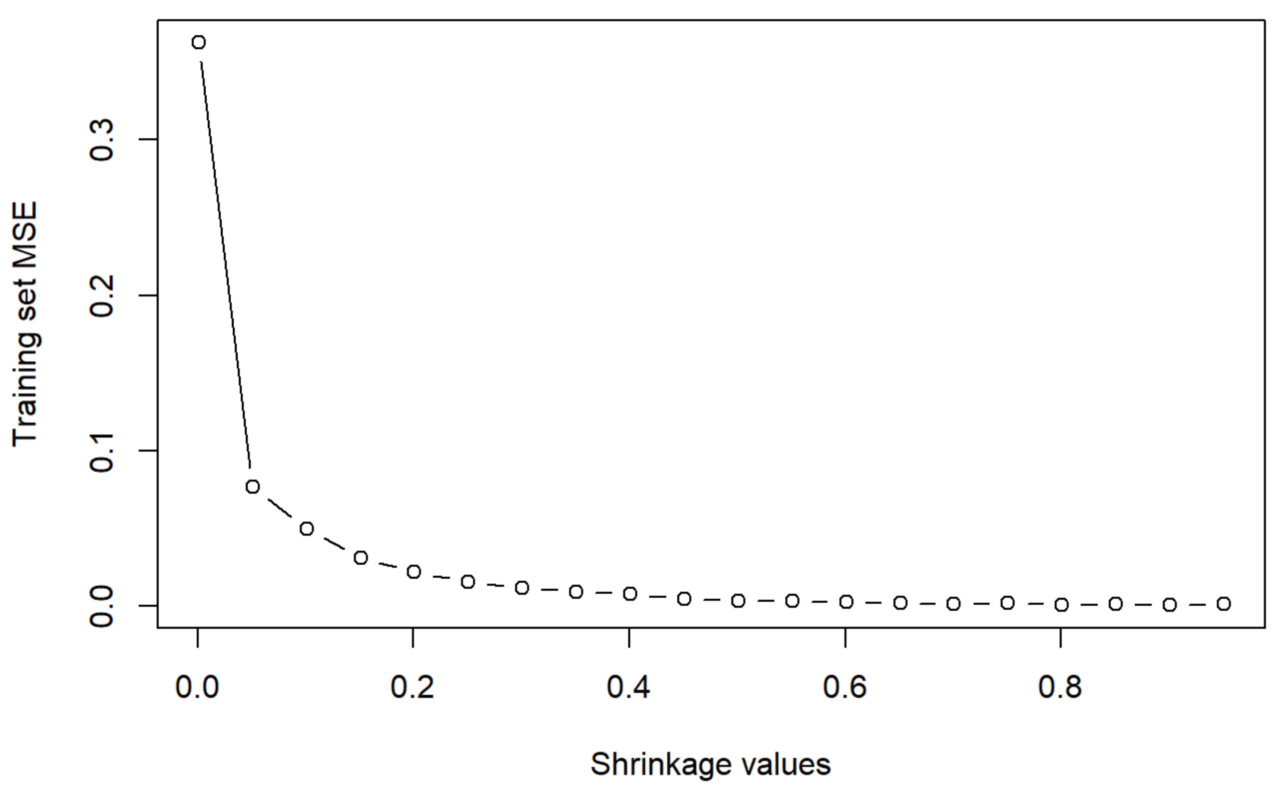
```
Hitters = na.omit(Hitters)
Hitters$Salary <- log(Hitters$Salary)
```

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

```
n = nrow(Hitters)
train = 1:200
test = 201:n
train_Hit <- Hitters[train, ]
test_Hit <- Hitters[test, ]
```

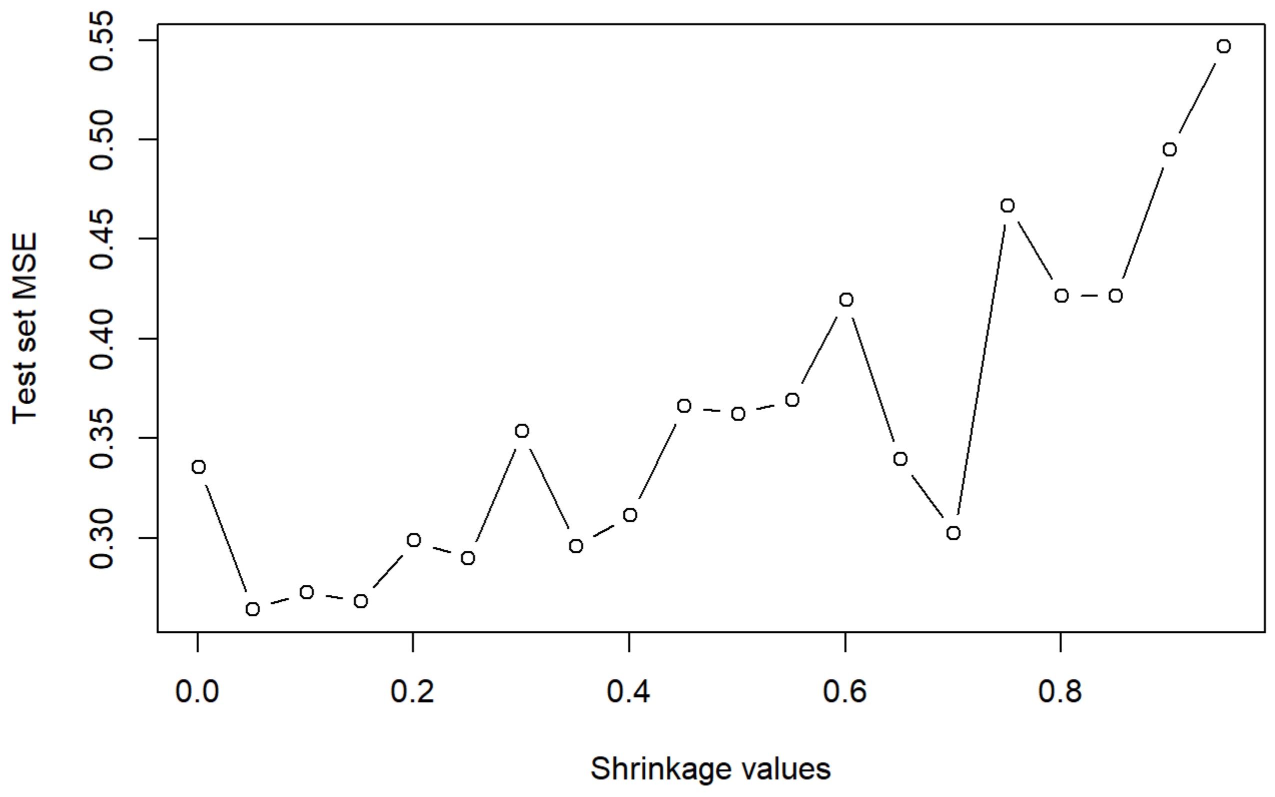
c. Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter λ. Produce a plot with different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

```
lambda_vals <- seq(from=0.001, to=1, by=0.05)
train_error <- rep(NA, length(lambda_vals))
for (l in 1:length(lambda_vals)) {
  boost_Hitt <- gbm(Salary~.,
                  data=train_Hit,
                  distribution = "gaussian",
                  n.trees = 1000,
                  shrinkage=lambda_vals[l])
  train_pred <- predict(boost_Hitt, train_Hit, n.trees=1000)
  train_error[l] <- mean((train_pred- train_Hit$Salary)^2)
}

plot(lambda_vals, train_error, type="b", xlab="Shrinkage values", ylab="Training set MSE")
```



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

```
lambda_vals <- seq(from=0.001, to=1, by=0.05)
test_error <- rep(NA, length(lambda_vals))
for (l in 1:length(lambda_vals)) {
  boost_Hitt <- gbm(Salary~.,
                  data=train_Hit,
                  distribution = "gaussian",
                  n.trees = 1000,
                  shrinkage=lambda_vals[l])
  test_pred <- predict(boost_Hitt, test_Hit, n.trees=1000)
  test_error[l] <- mean((test_pred - test_Hit$Salary)^2)
}


plot(lambda_vals, test_error, type="b", xlab="Shrinkage values", ylab="Test set MSE")
```



```
test_error[which.min(test_error)]
```

```
## [1] 0.2639862
```

e. Compare the test MSE of boosting to the test MSE that results from applying linear regression and LASSO.

```
lm_fit <- lm(Salary ~., data=train_Hit)
lm_pred <- predict(lm_fit, test_Hit)
mean((lm_pred - test_Hit$Salary)^2)
```

```
## [1] 0.4917959
```

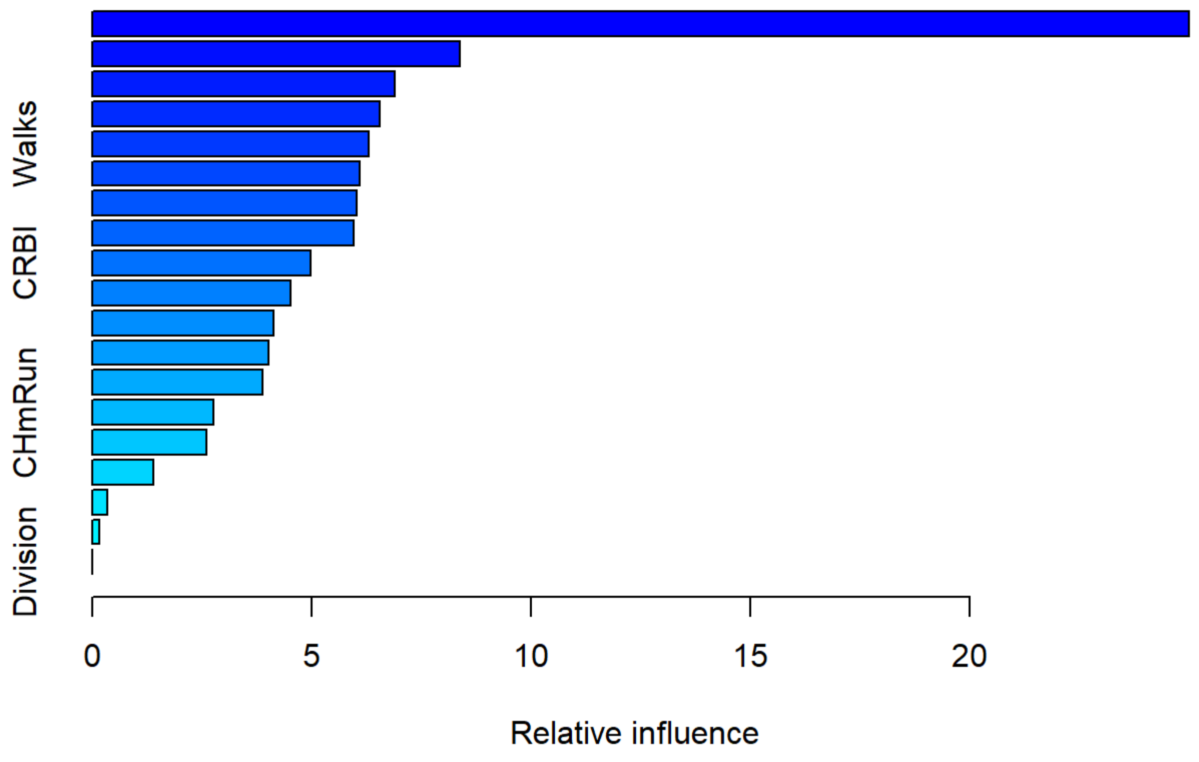
```
x_mat <- model.matrix(Salary ~ ., data=Hitters)[,-1]
x_train <- x_mat[train, ]
x_test <- x_mat[test, ]
y_train <- Hitters$Salary[train]
y_test <- Hitters$Salary[test]
ridge_fit <- glmnet(x_train, y_train, alpha=0)
cv_ridge <- cv.glmnet(x_train, y_train, alpha=0)
best.lam <- cv_ridge$lambda.min

ridge_pred <- predict(ridge_fit, s=best.lam, newx=x_test)
mean((ridge_pred - y_test)^2)
```

```
## [1] 0.4570283
```

boosting < ridge < linear regression

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

```
summary(boost_Hitt)
```



```
##                 var      rel.inf
## CAtBat       CAtBat 24.994418543
## PutOuts     PutOuts  8.378134196
## CRuns         CRuns  6.889335457
## AtBat         AtBat  6.563894673
## Walks         Walks  6.300506565
## Assists     Assists  6.089895068
## Hits           Hits  6.023070886
## RBI             RBI  5.949843408
## CRBI           CRBI  4.986349523
## Errors       Errors  4.514164721
## Years         Years  4.132279644
## HmRun         HmRun  4.009288970
## CWalks       CWalks  3.884160300
## CHmRun       CHmRun  2.772957635
## Runs           Runs  2.601003121
## CHits         CHits  1.396752662
## NewLeague NewLeague  0.342207515
## League       League  0.167290721
## Division   Division  0.004446391
```

CAtBat is the most important variable

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

```
bag.Hit <- randomForest(Salary ~ .,
                        data=train_Hit,
                        mtry=19,
                        importance=TRUE,
                        n.trees=1000)

bag.pred <- predict(bag.Hit, newdata=test_Hit)
mean((bag.pred-y_test)^2)
```

```
## [1] 0.2317199
```

Homework 5

收获与感悟:

在这门课中，我学习了很多机器学习的算法，对监督学习和非监督学习的很多算法有了新的领悟，通过实验我对每个算法的逻辑有了更深层次的理解，我也对许多数据集进行了实操，感觉很有趣。

看法与建议:

考核形式上可能会更倾向于带cheating sheet的半开卷考试，也会减少一些大家死记硬背的内容，感觉我也能更深入的研究和准备每个算法，可能会比论文形式掌握的更牢固。