Stats 101C HW 5

Charles Liu (304804942)

11/22/2020

Loading Necessary Packages:

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

Problem 1 (Exercise 5.4.2)

We will now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap sample from a set of n observations.

(a) What is the probability that the first bootstrap observation is not the j^{th} observation from the original sample? Justify your answer.

ANSWER: We have the probability that the j^{th} observation is the first bootstrap sample with probability of $\frac{1}{n}$. Now to find the probability that the j^{th} observation is *not* the first bootstrap sample is 1 minus that probability, which is $(1 - \frac{1}{n})$.

(b) What is the probability that the second bootstrap observation is not the j^{th} observation from the original sample?

ANSWER: We find that the first observation is the same as the second observation's probability. It will be the j^{th} observation is the second bootstrap sample with probability of $\frac{1}{n}$. Now to find the probability that the j^{th} observation is *not* the second bootstrap sample is 1 minus that probability, which is $(1 - \frac{1}{n})$.

(c) Argue that the probability that the j^{th} observation is not in the bootstrap sample is $(1 - \frac{1}{n})^n$.

ANSWER: We have bootstrap sampling come with replacements and all of the probabilities are independent of each other. From parts (2a) and (2b), we find the probability that the j^{th} observation is *not* the n^{th} bootstrap sample will have $(1-\frac{1}{n})$. Therefore, we combine all what is said to get… $(1-\frac{1}{n})*(1-\frac{1}{n})*...*(1-\frac{1}{n}) = (1 - \frac{1}{n})^n$

(d) When n = 5, what is the probability that the j^{th} observation is in the bootstrap sample?

ANSWER: $(1 - \frac{1}{5})^5 = 0.32768$ to get the probability that the j^{th} observation is *NOT* in the bootstrap sample. $1 - 0.32768 = 0.67232$ is the probability that the j^{th} observation *IS* in the boostrap sample for $n = 5$.

(e) When $n = 100$, what is the probability that the j^{th} observation is in the bootstrap sample?

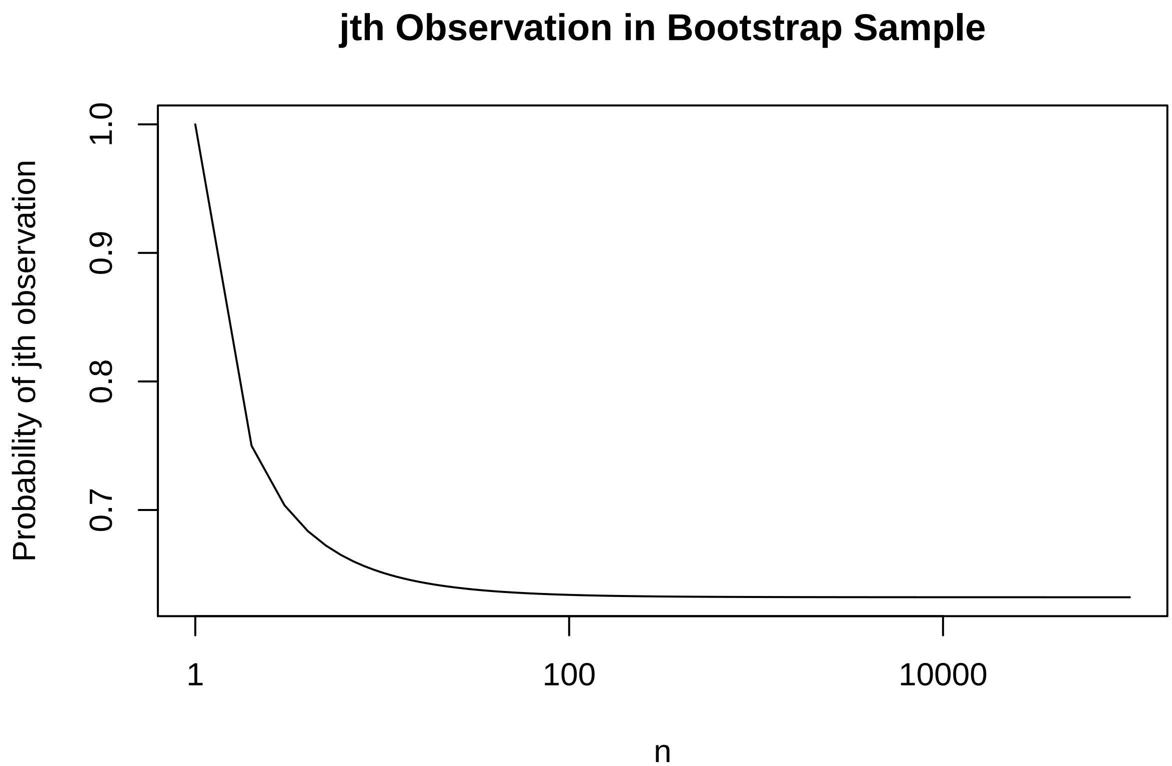
ANSWER: $(1 - \frac{1}{100})^{100} \approx 0.36603$ to get the probability that the j^{th} observation is *NOT* in the bootstrap sample. $1 - 0.36603 = 0.63397$ is the probability that the j^{th} observation *IS* in the boostrap sample for $n = 100$.

(f) When $n = 10,000$, what is the probability that the j^{th} observation is in the bootstrap sample?

ANSWER: $(1 - \frac{1}{10000})^{10000} \approx 0.36786$ to get the probability that the j^{th} observation is *NOT* in the bootstrap sample. $1 - 0.36786 = 0.63214$ is the probability that the j^{th} observation *IS* in the boostrap sample for $n = 10,000$.

(g) Create a plot that displays, for each integer value of n from 1 to 100,000, the probability that the jth observation is in the bootstrap sample. Comment on what you observe.

```r
# Create function of Bootstrap probability sampling
boostrap_sample <- function(n){
  prob <- 1 - ( (1 - (1/n))^(n) )
  return(prob)
}

# Set up the x and y in the plot
n_seq <- 1:10^5
boostrap_prob <- sapply(n_seq,  boostrap_sample)

# Plot the Bootstrap Samplings
plot(n_seq, boostrap_prob,
     main = "jth Observation in Bootstrap Sample",
     xlab = "n",
     ylab = "Probability of jth observation",
     type = "l",
     log = "x")
```



(h) We will now investigate numerically the probability that a bootstrap sample of size $n = 100$ contains the jth observation. Here $j = 4$. We repeatedly create bootstrap samples, and each time we record whether or not the fourth observation is contained in the bootstrap sample. Comment on the results obtained.

```
store=rep (NA , 10000)
for (i in 1:10000) {
  store[i]=sum(sample (1:100 , rep =TRUE)==4) >0
}
mean(store)
```

```
## [1] 0.6271
```

COMMENTS: We find the that stored results are approximately 0.63 for the $j^{th} = 4$ observation. From the Bootstrap Sample before question (2h), we realize that $\lim_{n\to\infty}(1-\frac{1}{n})^n = \frac{1}{e} \approx 0.36788$ is the the probability that the j^{th} observation is *NOT* in the bootstrap sample. Using this fact, we find that the probability that the j^{th} observation is *IS* in the bootstrap sample is $\lim_{n\to\infty} 1 - (1 - \frac{1}{n})^n = (1 - \frac{1}{e}) \approx 0.63212$. This follows the same what we found in (5h), which is approximately 0.63. As the $n^{th} \to \infty$ samples, we see that the probability of *ANY* j^{th} observation will be approximately 0.63.

Problem 2 (Exercise 8.4.10)

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

```r
data(Hitters)
```

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

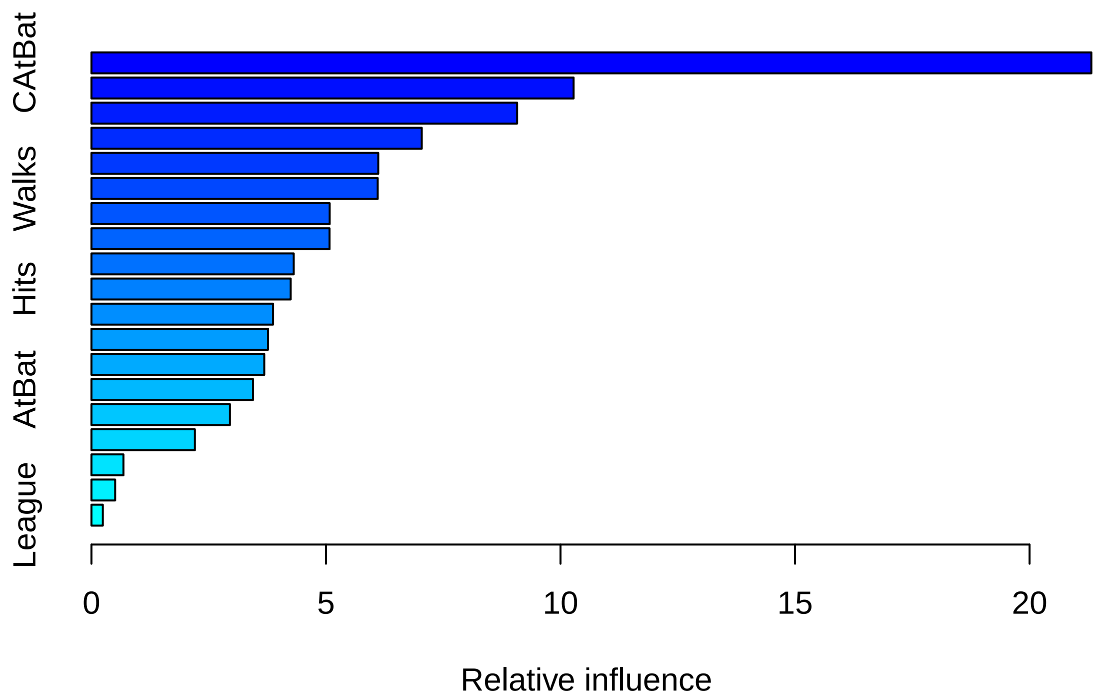
```r
Hitters <- na.omit(Hitters)
Hitters$Salary <- log(Hitters$Salary)
```

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

```r
# First 200 for training set
train <- 1:200
hitters_train <- Hitters[train, ]
hitters_test <- Hitters[-train, ]
```

(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.

```r
set.seed(1)
# the option distribution="gaussian" since this is a regression problem; if it were a binary classifica
hitters_train_boost_summary <- gbm(Salary ~ ., data = hitters_train,
                     distribution = "gaussian",
                     n.trees = 1000)
summary(hitters_train_boost_summary)
```



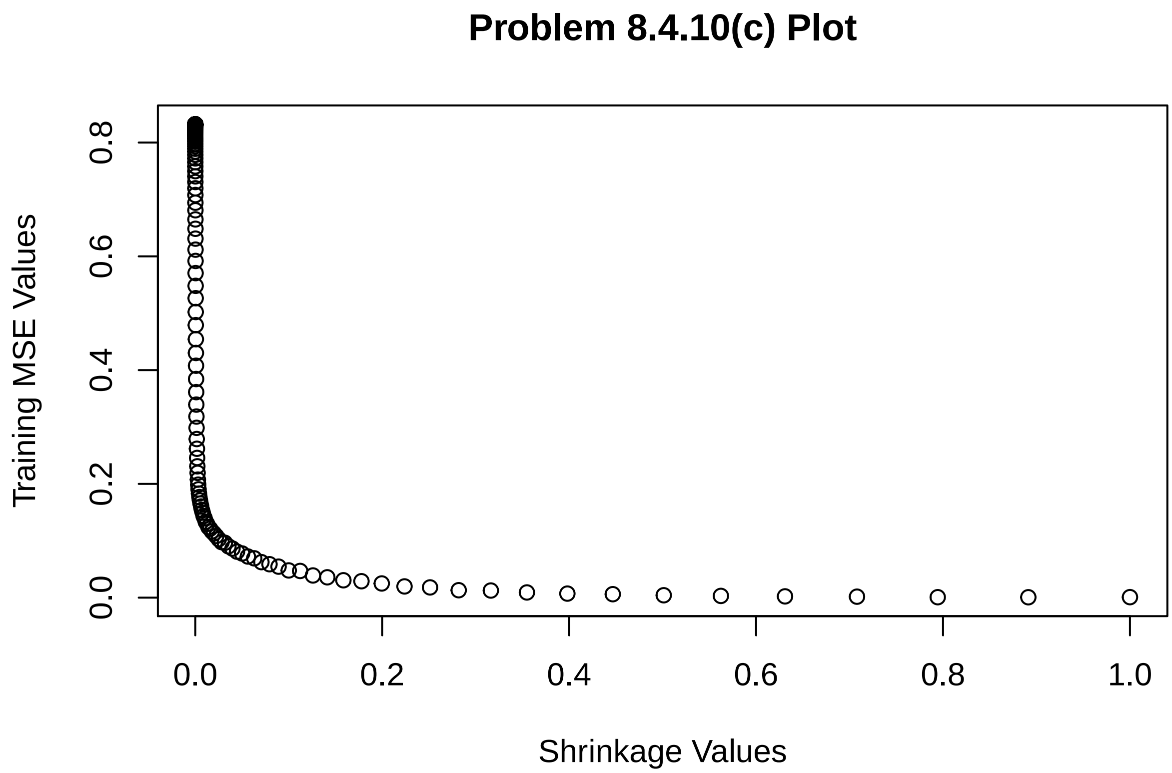

```
##                 var    rel.inf
## CAtBat       CAtBat 21.3171524
## PutOuts     PutOuts 10.2788377
## CRuns         CRuns  9.0751254
## CRBI           CRBI  7.0417974
## CHmRun       CHmRun  6.1148613
## Walks         Walks  6.1031023
## Assists     Assists  5.0788336
## Years         Years  5.0759281
## CWalks       CWalks  4.3126065
## Hits           Hits  4.2479119
## RBI             RBI  3.8718470
## CHits         CHits  3.7653292
## Runs           Runs  3.6845856
## AtBat         AtBat  3.4443531
## HmRun         HmRun  2.9520357
## Errors       Errors  2.2050936
## Division   Division  0.6822530
## NewLeague NewLeague  0.5057473
## League       League  0.2425989
```

```
# Find the Training MSE once
predict_hitters_train_summary <- predict(hitters_train_boost_summary,
                                         hitters_train,
                                         n.trees = 1000)
```

```r
MSE_hitters_train_boost <- mean((predict_hitters_train_summary -
                                    hitters_train$Salary)^2)
MSE_hitters_train_boost
```

```
## [1] 0.05109086
```

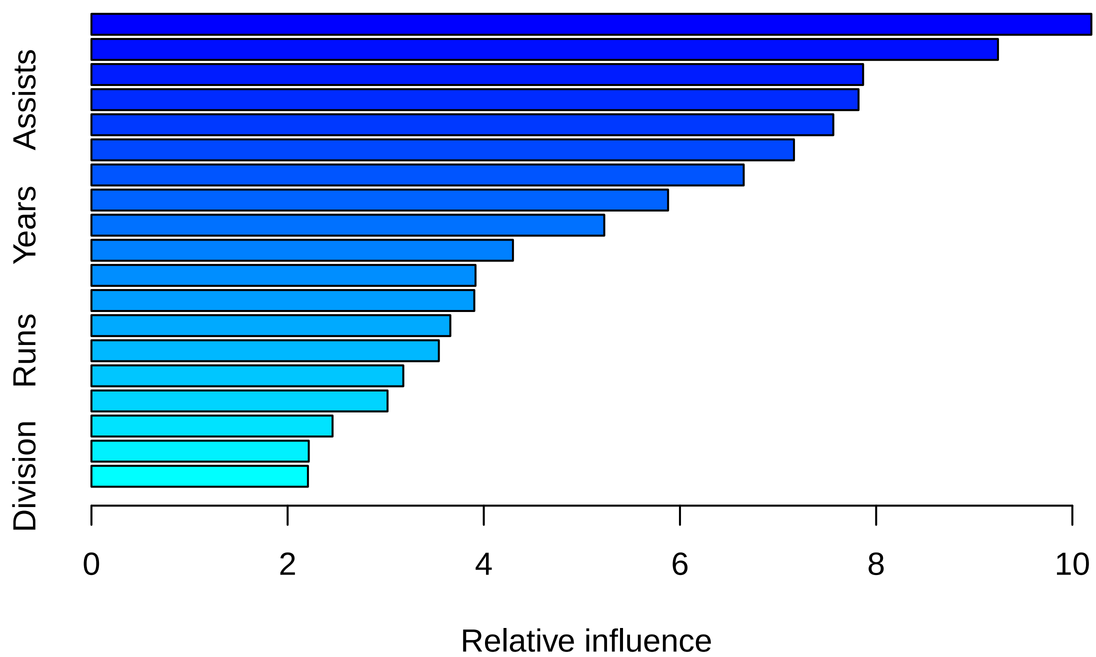
```r
# Set up for the plot
lambdas <- 10^( seq(-10, 0, by = 0.05) )
train_MSE <- rep(NA, length(lambdas))

for (i in 1:length(lambdas)) {
    hitters_train_boost <- gbm(Salary ~ .,
                          data = hitters_train,
                          distribution = "gaussian",
                          n.trees = 1000,
                          shrinkage = lambdas[i])
    predict_hitters_train <- predict(hitters_train_boost,
                                     hitters_train, n.trees = 1000)
    train_MSE[i] <- mean((predict_hitters_train - hitters_train$Salary)^2)
}

# Plot the Shrinkage Vales vs. the Training MSE Values
plot(lambdas, train_MSE,
     main = "Problem 8.4.10(c) Plot",
     xlab = "Shrinkage Values",
     ylab = "Training MSE Values")
```



ANSWER: The MSE from Boosting from the training dataset will give us 0.05109086. The plot above shows a multitude of difference shrinkage values with their corresponding training MSE values.

(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)
# the option distribution="gaussian" since this is a regression problem; if it were a binary classifica
hitters_test_boost_summary <- gbm(Salary ~ ., data = hitters_test,
                     distribution = "gaussian",
                     n.trees = 1000)
summary(hitters_test_boost_summary)
```



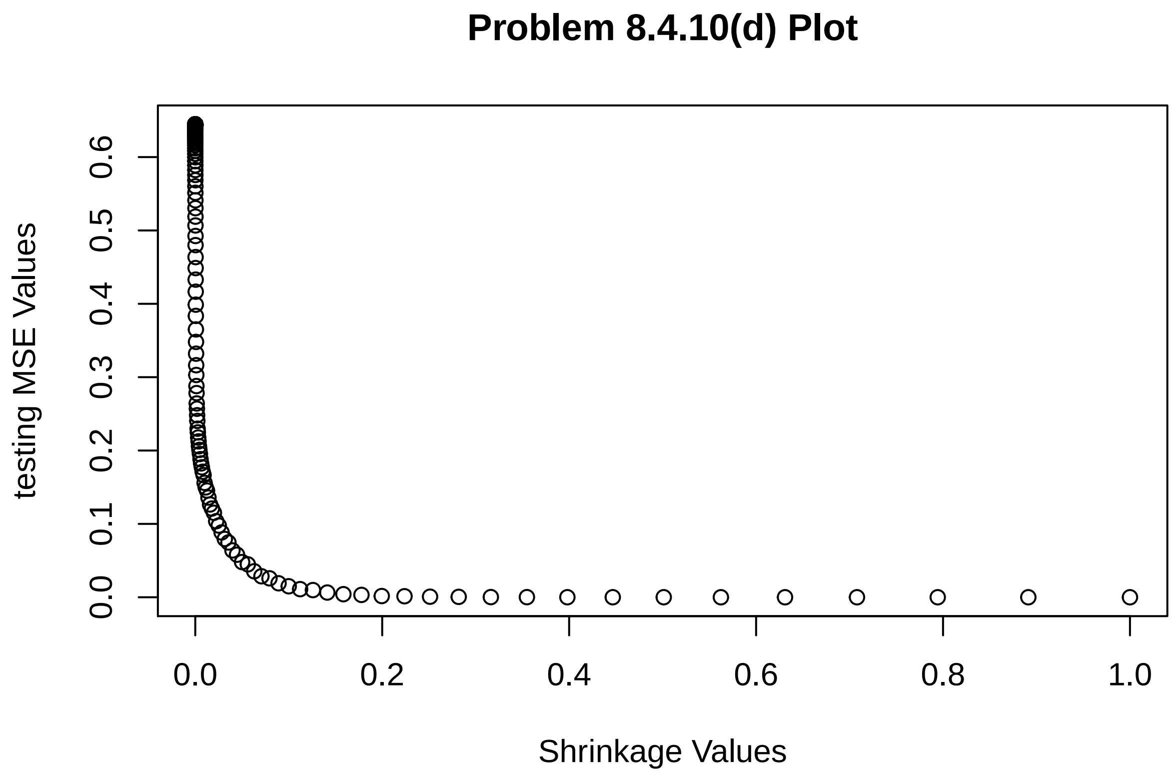

```
##                 var   rel.inf
## CRuns         CRuns 10.193963
## CHmRun       CHmRun  9.241813
## PutOuts     PutOuts  7.866975
## Assists     Assists  7.820207
## CHits         CHits  7.563049
## CRBI           CRBI  7.162059
## HmRun         HmRun  6.649630
## Errors       Errors  5.878650
## Years         Years  5.228160
## Walks         Walks  4.297178
## CWalks       CWalks  3.915518
## CAtBat       CAtBat  3.903139
## League       League  3.658747
## Runs           Runs  3.541810
## RBI             RBI  3.179360
## Hits           Hits  3.018607
## AtBat         AtBat  2.458151
## NewLeague NewLeague  2.215721
## Division   Division  2.207262
```

```
# Find the testing MSE once
predict_hitters_test_summary <- predict(hitters_test_boost_summary,
                                        hitters_test,
                                        n.trees = 1000)
```

```r
MSE_hitters_test_boost <- mean((predict_hitters_test_summary -
                               hitters_test$Salary)^2)
MSE_hitters_test_boost
```

```
## [1] 0.01494023
```

```r
# Set up for the plot
lambdas <- 10^( seq(-10, 0, by = 0.05) )
test_MSE <- rep(NA, length(lambdas))

for (i in 1:length(lambdas)) {
    hitters_test_boost <- gbm(Salary ~ .,
                         data = hitters_test,
                         distribution = "gaussian",
                         n.trees = 1000,
                         shrinkage = lambdas[i])
    predict_hitters_test <- predict(hitters_test_boost,
                                    hitters_test, n.trees = 1000)
    test_MSE[i] <- mean((predict_hitters_test - hitters_test$Salary)^2)
}

# Plot the Shrinkage Vales vs. the testing MSE Values
plot(lambdas, test_MSE,
     main = "Problem 8.4.10(d) Plot",
     xlab = "Shrinkage Values",
     ylab = "testing MSE Values")
```



ANSWER: The MSE from Boosting from the testing dataset will give us 0.01494023. The plot above shows a multitude of difference shrinkage values with their corresponding testing MSE values.

(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.

```
# Linear Model
hitters_lm <- lm(Salary ~ ., data = hitters_train)
predict_hitters_lm <- predict(hitters_lm, hitters_test)
MSE_hitters_lm <- mean((predict_hitters_lm - hitters_test$Salary)^2)
MSE_hitters_lm
```

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

```
# Setting up for Chapter 6 models
x_train <- model.matrix(Salary ~ ., data = hitters_train)
x_test <- model.matrix(Salary ~ ., data = hitters_test)
y_train <- hitters_train$Salary
y_test <- hitters_test$Salary

# Ridge Regression Model
hitters_ridge <- glmnet(x_test, y_test, alpha = 0)
predict_hitters_ridge <- predict(hitters_ridge, x_test)
MSE_hitters_ridge <- mean((predict_hitters_ridge - hitters_test$Salary)^2)
MSE_hitters_ridge
```

```
## [1] 0.4803726
```

```r
# LASSO Model
hitters_LASSO <- glmnet(x_test, y_test, alpha = 1)
predict_hitters_LASSO <- predict(hitters_LASSO, x_test)
MSE_hitters_LASSO <- mean((predict_hitters_LASSO - hitters_test$Salary)^2)
MSE_hitters_LASSO
```

```
## [1] 0.3417042
```

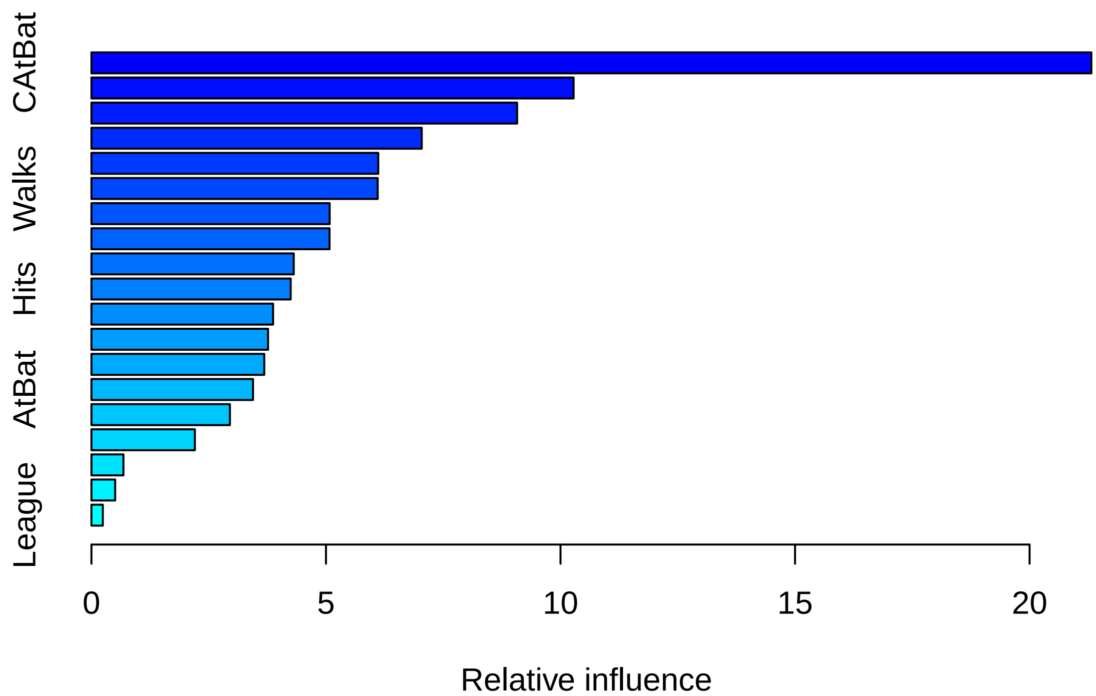
```r
# Compare all the MSE values
compare_matrix <- matrix(c(MSE_hitters_test_boost,
                           MSE_hitters_lm,
                           MSE_hitters_ridge,
                           MSE_hitters_LASSO))
colnames(compare_matrix) <- "Test MSE Values"
rownames(compare_matrix) <- c("Boosting", "Linear Regression",
                              "Ridge Regression", "LASSO")
compare_matrix
```

```
##                   Test MSE Values
## Boosting               0.01494023
## Linear Regression      0.49179594
## Ridge Regression       0.48037258
## LASSO                  0.34170417
```

COMMENTS: We can see that *Boosting* model yields the lowest MSE at 0.01494023. This tells us this is the preferred model method to use. Meanwhile, the second lowest MSE is LASSO, third is Ridge Regression, and finally the highest MSE is Linear Regression.

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

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




```
##                 var    rel.inf
## CAtBat       CAtBat 21.3171524
## PutOuts     PutOuts 10.2788377
## CRuns         CRuns  9.0751254
## CRBI           CRBI  7.0417974
## CHmRun       CHmRun  6.1148613
## Walks         Walks  6.1031023
## Assists     Assists  5.0788336
## Years         Years  5.0759281
## CWalks       CWalks  4.3126065
## Hits           Hits  4.2479119
## RBI             RBI  3.8718470
## CHits         CHits  3.7653292
## Runs           Runs  3.6845856
## AtBat         AtBat  3.4443531
## HmRun         HmRun  2.9520357
## Errors       Errors  2.2050936
## Division   Division  0.6822530
## NewLeague NewLeague  0.5057473
## League       League  0.2425989
```

ANSWER: The most important variable is *CAtBat*. Then, we have *PutOuts*, *CRuns*, and *CRBI*.

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

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

predict_hitters_bag <- predict(hitters_bag, hitters_test)
MSE_hitters_bag <- mean((predict_hitters_bag - hitters_test$Salary)^2)
MSE_hitters_bag
```

```
## [1] 0.2301184
```

```r
# Compare with the rest of the Test MSE
compare_matrix <- rbind(compare_matrix, MSE_hitters_bag)
rownames(compare_matrix)[5] <- "Bagging"
compare_matrix
```

```
##                   Test MSE Values
## Boosting               0.01494023
## Linear Regression      0.49179594
## Ridge Regression       0.48037258
## LASSO                  0.34170417
## Bagging                0.23011836
```

ANSWER: The MSE for the *Bagging* model method is 0.2301184. We can see that *Boosting* is still the best method to use for this dataset. Meanwhile, the second lowest MSE is now Bagging, third is LASSO, fourth is Ridge Regression, and finally the highest MSE is still Linear Regression.