Chapter 8 HW

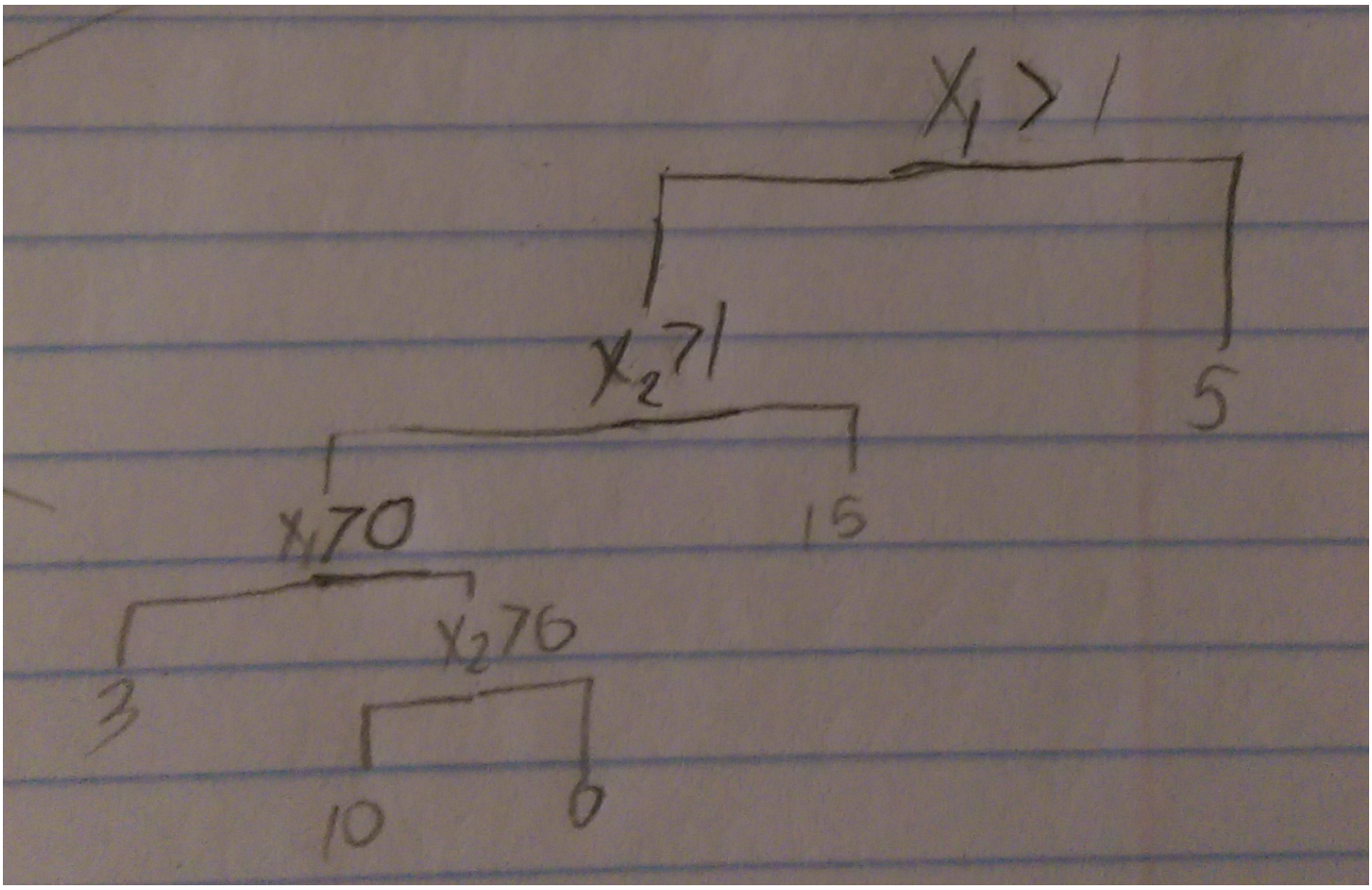
Nehemya McCarter-Ribakoff

4 May 2017

Conceptual Questions

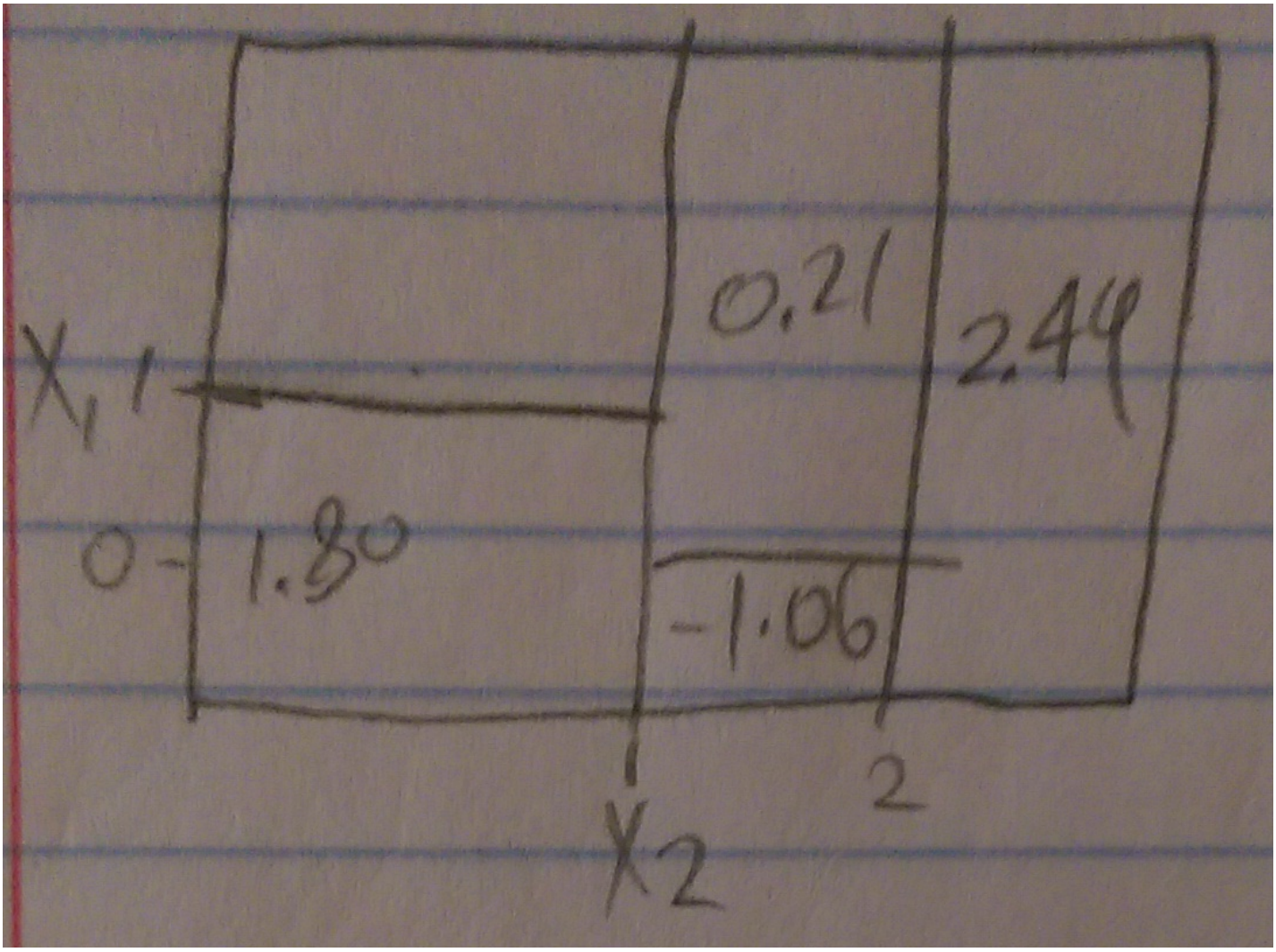
Exercise 4: This question relates to the plots in Figure 8.12.

a. Sketch the tree corresponding to the partition of the predictor space illustrated in the left-hand panel of Figure 8.12. The numbers inside the boxes indicate the mean of Y within each region.



Tree

b. Create a diagram similar to the left-hand panel of Figure 8.12, using the tree illustrated in the right-hand panel of the same figure. You should divide up the predictor space into the correct regions, and indicate the mean for each region



Block

Applied Questions

Exercise 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.

```
library(ISLR)
library(tree)
library(caTools)
data(Carseats)
```

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

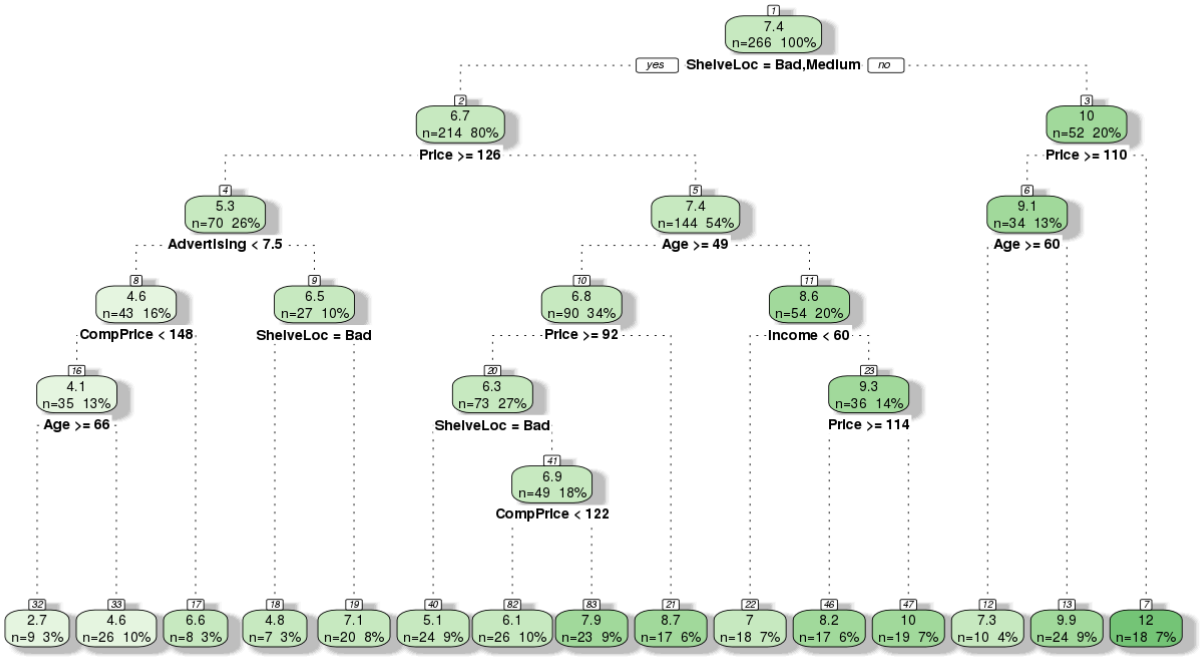
```
data(Carseats)
carseats.true = sample.split(Carseats$Sales, 2/3)
carseats.train = subset(Carseats, carseats.true == TRUE)
carseats.test = subset(Carseats, carseats.true == FALSE)
```

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

```
library(rpart)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
tree.carseats = tree(Sales~., carseats.train)
rpart.carseats = rpart(Sales~., carseats.train)
fancyRpartPlot(rpart.carseats)
```



Rattle 2017-May-04 08:36:01 rickygoals

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

```
yhat.rtree = predict(rpart.carseats, carseats.test)
mean((yhat.rtree - carseats.test$Sales)^2)
```

```
## [1] 4.614792
```

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.

```
library(randomForest)
```

```
## randomForest 4.6-12
```

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

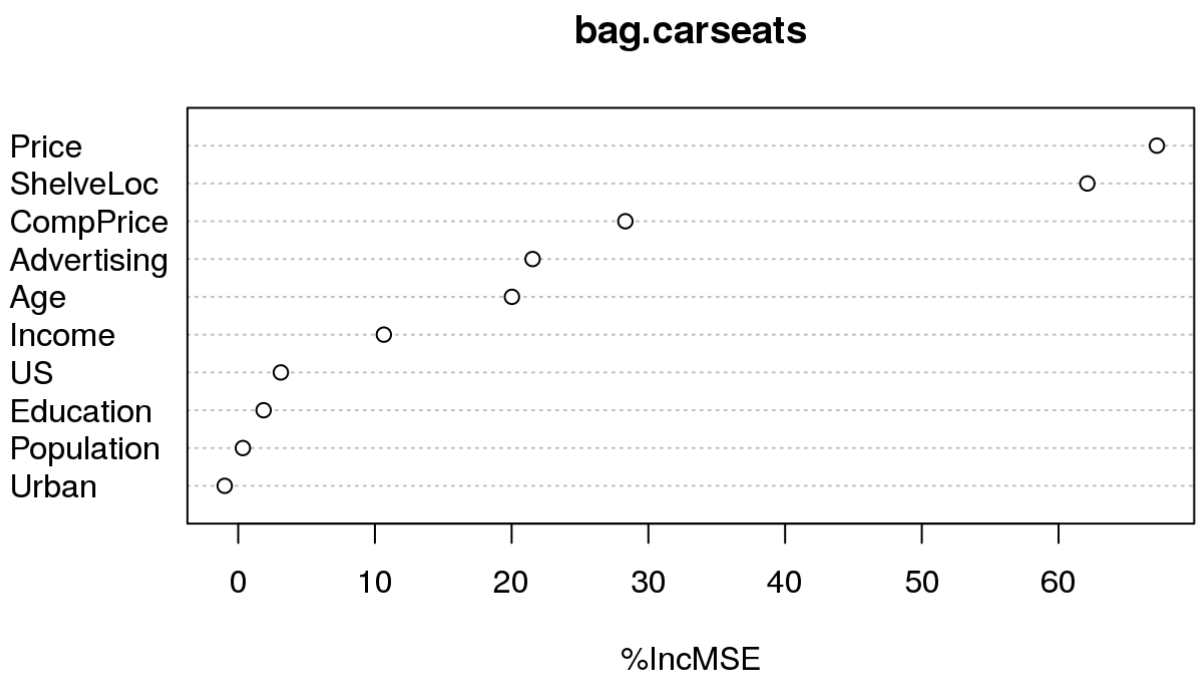
```
##
## Attaching package: 'randomForest'
```

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

```
set.seed(121)
bag.carseats = randomForest(Sales~., carseats.train, mtry=10, importance=TRUE)
bag.predict = predict(bag.carseats, carseats.test)
mean((bag.predict - carseats.test$Sales)^2)
```

```
## [1] 2.700373
```

```
varImpPlot(bag.carseats, type=1)
```



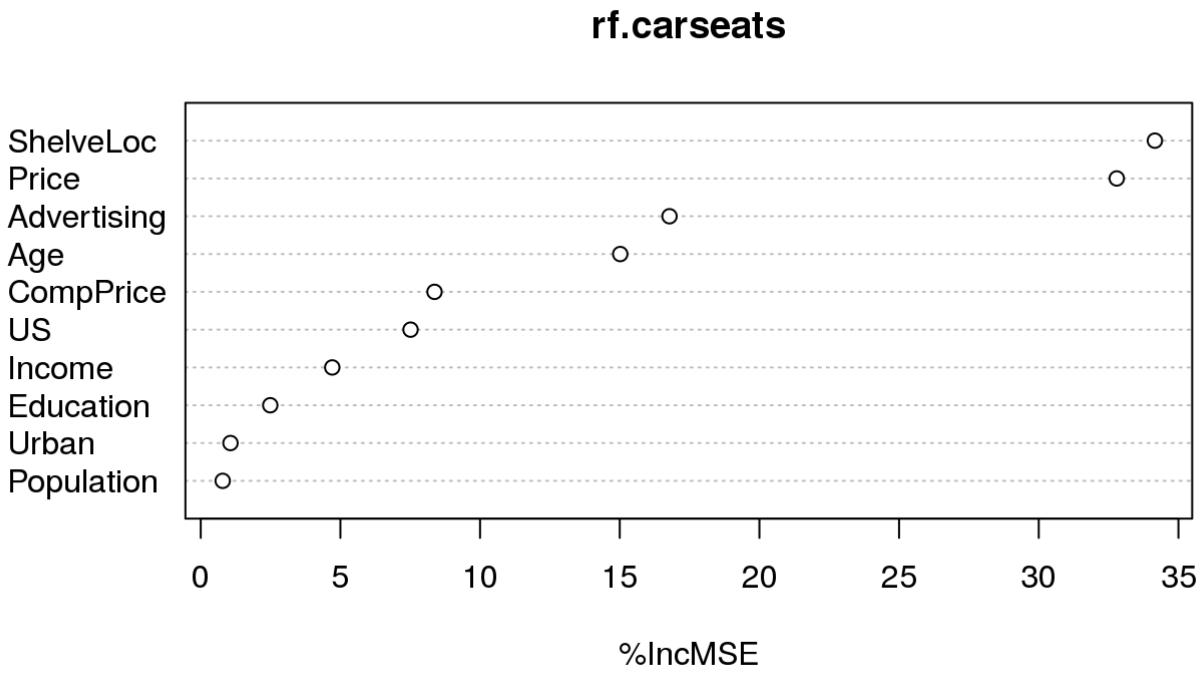
Bagging has reduced the MSE down quite considerably.

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.carseats = randomForest(Sales ~., carseats.train, mtry=sqrt(3), importance=TRUE)
rf.predict = predict(rf.carseats, carseats.test)
mean((rf.predict - carseats.test$Sales)^2)
```

```
## [1] 3.678517
```

```
varImpPlot(rf.carseats, type=1)
```



RF does not do as well as bagging. Decorrelating the nodes had had a negative effect on the test score.

Exercise 10: We now use boosting to predict Salary in the Hitters data set.

```
library(ISLR)
data(Hitters)
```

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

```
Hitters = na.omit(Hitters)
logSalary = log(Hitters$Salary)
```

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

```
# logSalary has 263 observations
split = sample.split(Hitters$Salary, 2/3)
Hitters.train = subset(Hitters, split == TRUE)
Hitters.test = subset(Hitters, split == FALSE)
```

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.

```
library(gbm)
```

```
## Loading required package: survival
```

```
## Loading required package: splines
```

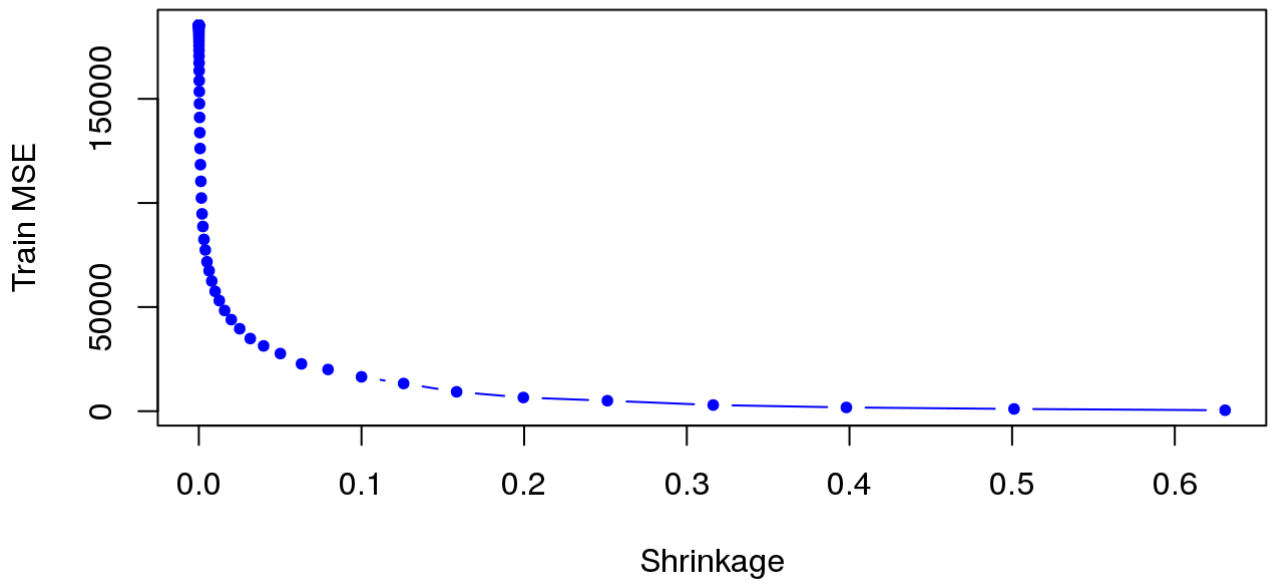
```
## Loading required package: lattice
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.3
```

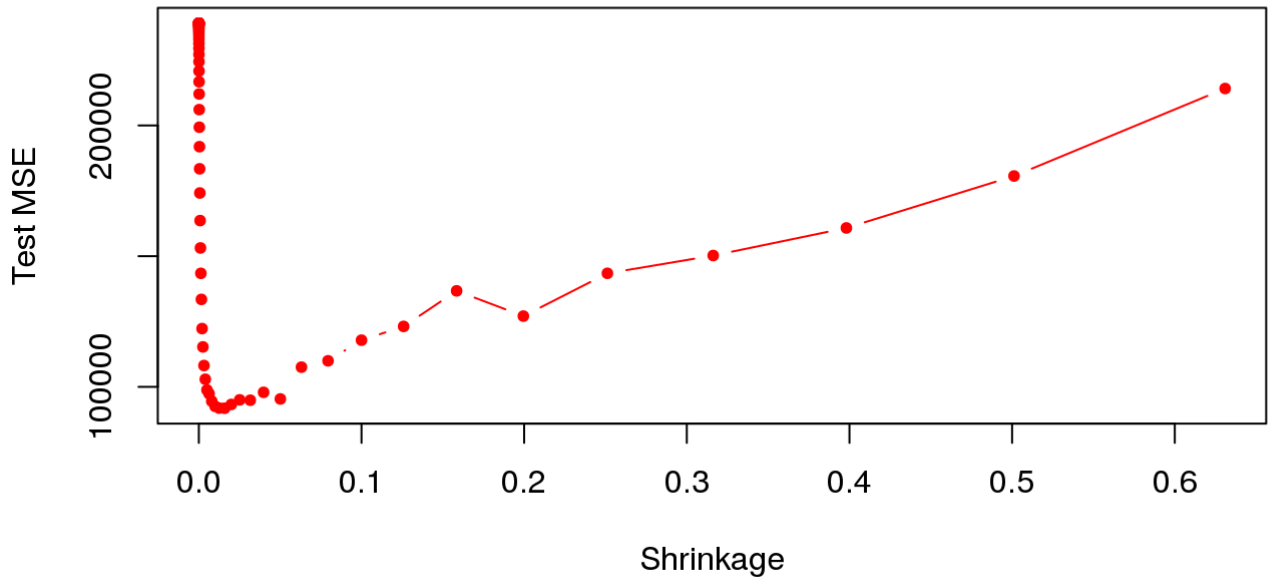
```
set.seed(1)
pows = seq(-10, -0.2, by=0.1)
lambdas = 10 ^ pows
length.lambdas = length(lambdas)
train.errors = rep(NA, length.lambdas)
test.errors = rep(NA, length.lambdas)
for (i in 1:length.lambdas) {
        boost.hitters = gbm(Salary~., data=Hitters.train, distribution="gaussian", n.t
rees=1000, shrinkage=lambdas[i])
        train.pred = predict(boost.hitters, Hitters.train, n.trees=1000)
        test.pred = predict(boost.hitters, Hitters.test, n.trees=1000)
        train.errors[i] = mean((Hitters.train$Salary - train.pred)^2)
        test.errors[i] = mean((Hitters.test$Salary - test.pred)^2)
}

plot(lambdas, train.errors, 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(lambdas, test.errors, type="b", xlab="Shrinkage", ylab="Test MSE", col="red", pch=20)
```




```
min(test.errors)
```

```
## [1] 91826.68
```

```
lambdas[which.min(test.errors)]
```

```
## [1] 0.01584893
```

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.

```
lm.fit = lm(Salary~., data=Hitters.train)
lm.pred = predict(lm.fit, Hitters.test)
mean((Hitters.test$Salary - lm.pred)^2)
```

```
## [1] 117327.9
```

```
library(glmnet)
```

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

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-5
```

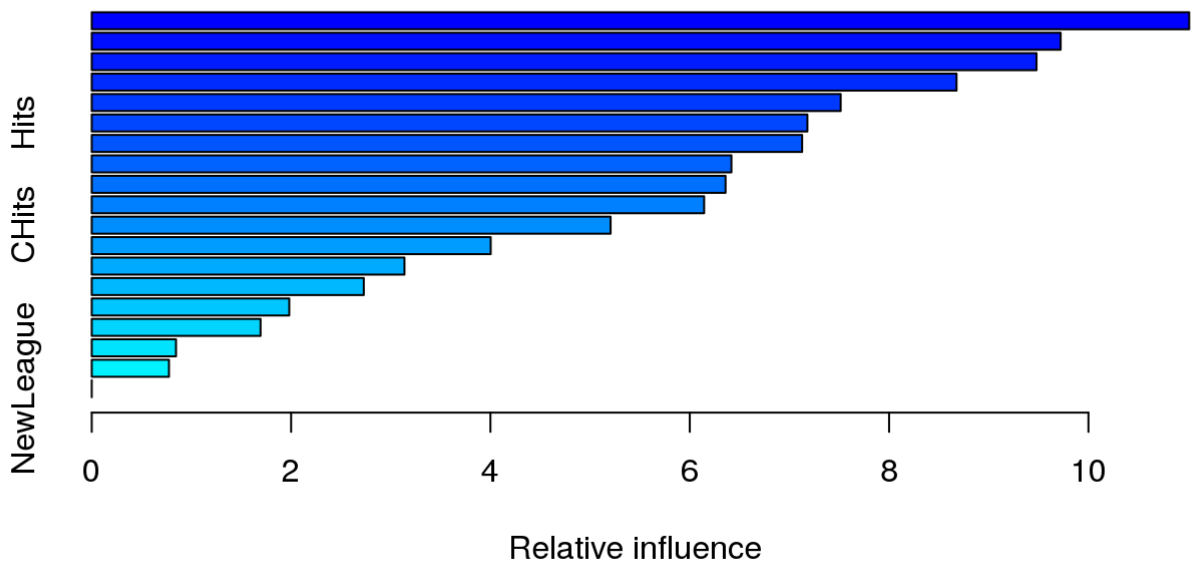
```
set.seed(134)
x = model.matrix(Salary~., data=Hitters.train)
y = Hitters.train$Salary
x.test = model.matrix(Salary~., data=Hitters.test)
lasso.fit = glmnet(x, y, alpha=1)
lasso.pred = predict(lasso.fit, s=0.01, newx=x.test)
mean((Hitters.test$Salary - lasso.pred)^2)
```

```
## [1] 116130.2
```

Test MSE reaches a far lower value with boosting.

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

```
summary(gbm(Salary~., data=Hitters.train, distribution="gaussian", n.trees=1000, shrin
kage=lambdas[which.min(test.errors)]))
```




```
##                 var    rel.inf
## PutOuts     PutOuts 11.0107533
## Walks         Walks  9.7211337
## CRBI           CRBI  9.4791782
## Years         Years  8.6766924
## CHmRun       CHmRun  7.5141256
## Hits           Hits  7.1800953
## RBI             RBI  7.1272807
## CRuns         CRuns  6.4177745
## CAtBat       CAtBat  6.3589182
## CWalks       CWalks  6.1439193
## CHits         CHits  5.2055833
## Assists     Assists  4.0030046
## HmRun         HmRun  3.1375608
## Runs           Runs  2.7292957
## Errors       Errors  1.9814409
## AtBat         AtBat  1.6933875
## League       League  0.8451764
## Division   Division  0.7746796
## NewLeague NewLeague  0.0000000
```

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

```
set.seed(21)
rf.hitters = randomForest(Salary~., data=Hitters.train, ntree=500, mtry=19)
rf.pred = predict(rf.hitters, Hitters.test)
mean((Hitters.test$Salary - rf.pred)^2)
```

```
## [1] 87754.97
```

Teamwork report

Team member	Conceptual	Applied	Contribution %
Nehemya	Yes	Yes	100%
Total			100%