Individual Assignment 8

Zixiao Wu

2022-11-10

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

Exercises 8.4 Problem #8:

(d).

```
library(ISLR)
library(tree)
```

```
## Warning: 程辑包'tree'是用R版本4.2.2 来建造的
```

```
library(MASS)
library(randomForest)
```

```
## Warning: 程辑包'randomForest'是用R版本4.2.2 来建造的
```

```
## randomForest 4.7-1.1
```

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

```
set.seed(100)

num = sample(1:nrow(Carseats), nrow(Carseats)/2)
train = data.frame(Carseats[num,])
test = data.frame(Carseats[-num,])
```

```
attach(Carseats)
bag.car = randomForest(Sales~., data=train, mtry=10, importance=TRUE)
bag.car
```

```
##
## Call:
##  randomForest(formula = Sales ~ ., data = train, mtry = 10, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##           Mean of squared residuals: 2.493042
##                     % Var explained: 66.49
```

```
importance(bag.car)
```

```
##                %IncMSE IncNodePurity
## CompPrice   23.6034138    123.927793
## Income      -0.1841818     55.713070
## Advertising 13.2961197     71.190078
## Population   2.5422959     54.346026
## Price       47.3201219    351.362625
## ShelveLoc   67.4664116    596.528573
## Age         18.2975161    148.854700
## Education    1.4741897     39.212178
## Urban       -0.8398648      6.043315
## US           4.8741858      5.204826
```

We can find that ShelveLoc and Price are important.

```
set.seed(100)
bag.car = predict(bag.car,newdata = test)
mean((bag.car-test$Sales)^2)
```

```
## [1] 3.249445
```

The test MSE is 3.25.

(e).

```
library(randomForest)
set.seed(100)
for (m in seq(1:10)){
  rf = randomForest(Sales~.,data=train,mtry=m,importance=T)
  mse = mean((predict(rf,newdata = test)-test$Sales)^2)
  print(mse)
}
```

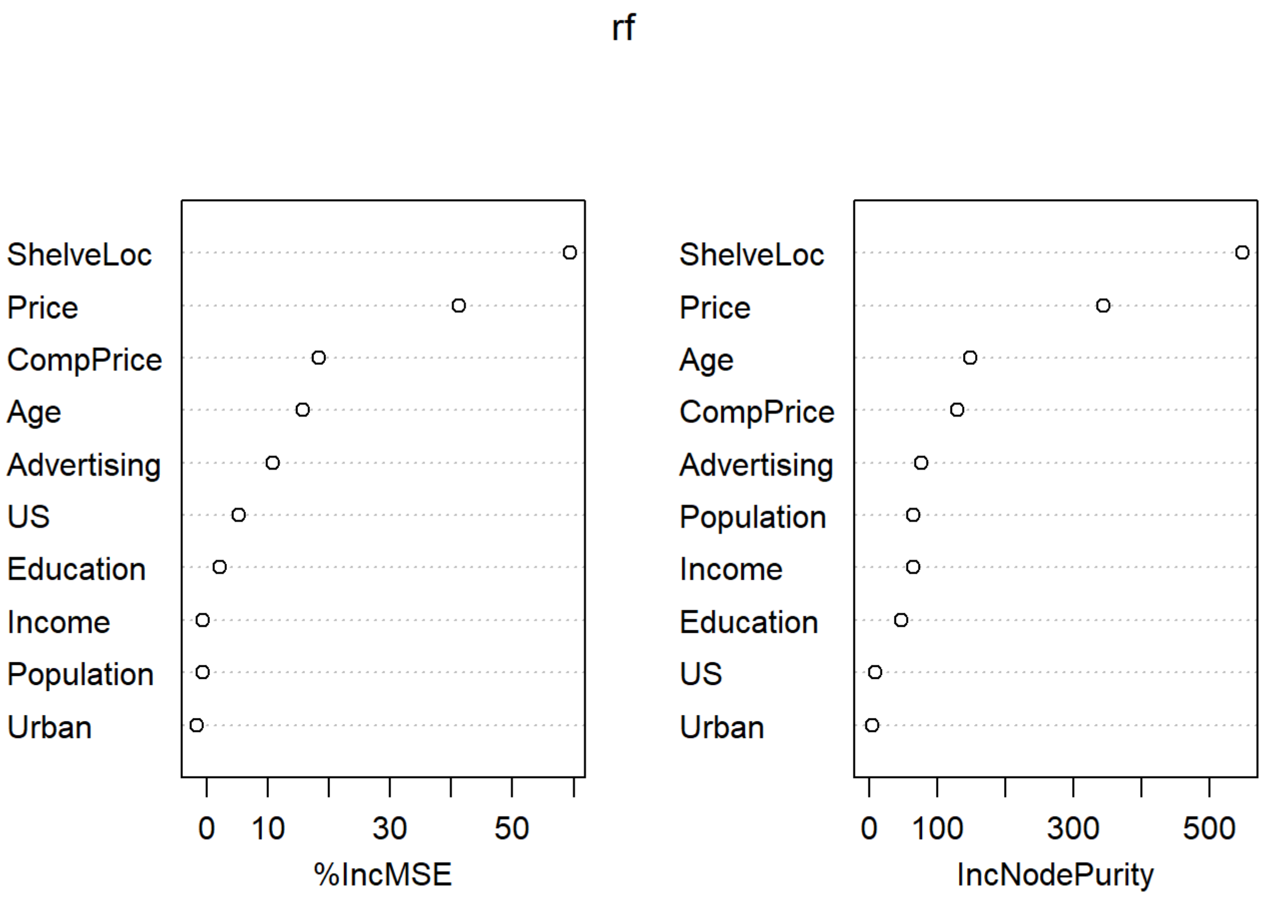
```
## [1] 5.250622
## [1] 3.99048
## [1] 3.483607
## [1] 3.344269
## [1] 3.251978
## [1] 3.195777
## [1] 3.178101
## [1] 3.210043
## [1] 3.22456
## [1] 3.28893
```

We can find that when m = 7, the model has the lowest test MSE.

```
set.seed(100)
rf = randomForest(Sales~.,data=train,mtry=7,importance=T)
importance(rf)
```

```
##                %IncMSE IncNodePurity
## CompPrice   18.3505793    129.024359
## Income      -0.6206208     64.756104
## Advertising 10.8607804     77.356103
## Population  -0.6829439     64.844157
## Price       41.3146040    343.886846
## ShelveLoc   59.4448919    548.399181
## Age         15.7617751    148.289803
## Education    2.1232042     47.537831
## Urban       -1.6803039      5.200473
## US           5.2675693      8.887731
```

```
varImpPlot(rf)
```



We can find that ShelveLoc and Price are also important in random forest model.

Problem #10: (a).

```
library(MASS)
library(gbm)
```

```
## Warning: 程辑包'gbm'是用R版本4.2.2 来建造的
```

```
## Loaded gbm 2.1.8.1
```

```
library(magrittr)
```

```
## Warning: 程辑包'magrittr'是用R版本4.2.2 来建造的
```

```
library(dplyr)
```

```
## Warning: 程辑包'dplyr'是用R版本4.2.2 来建造的
```

```
##
## 载入程辑包：'dplyr'
```

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

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

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

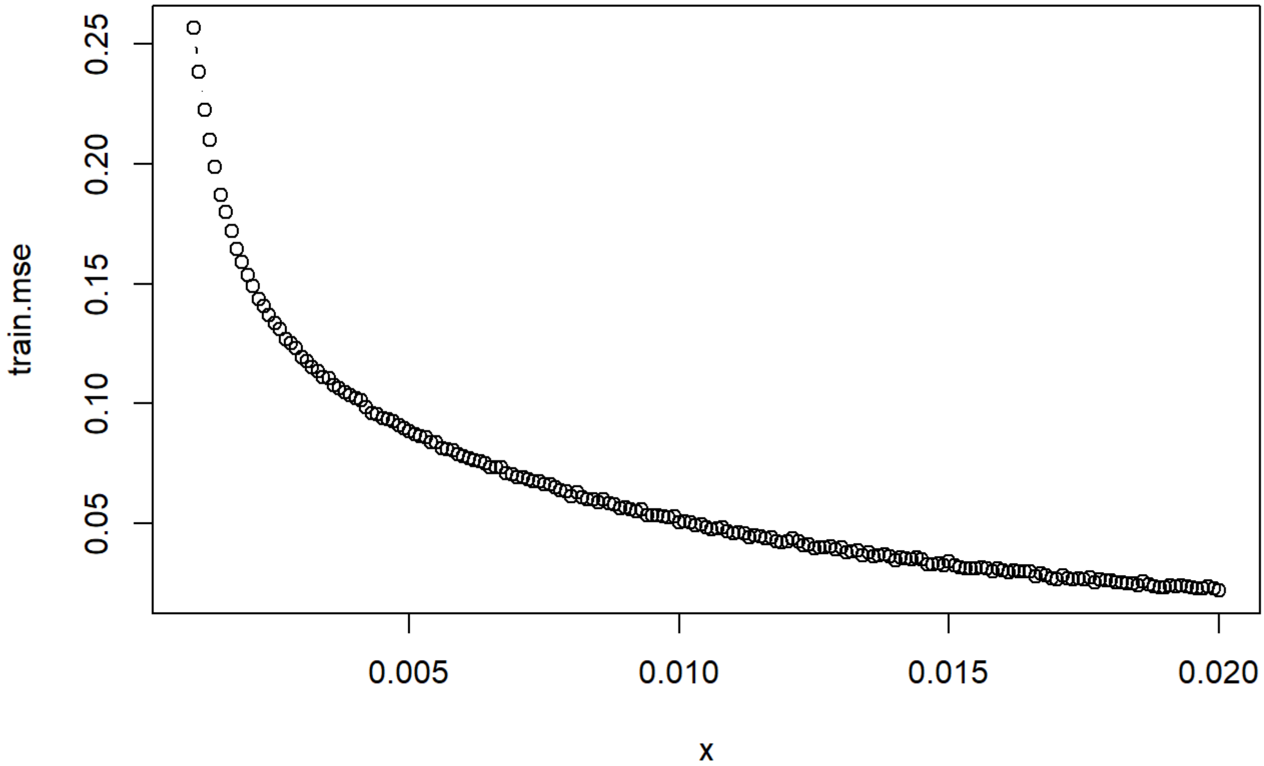
```
set.seed(100)

data = Hitters %>% dplyr :: filter(!is.na(Salary))
data$Salary = log(data$Salary)
```

(b).

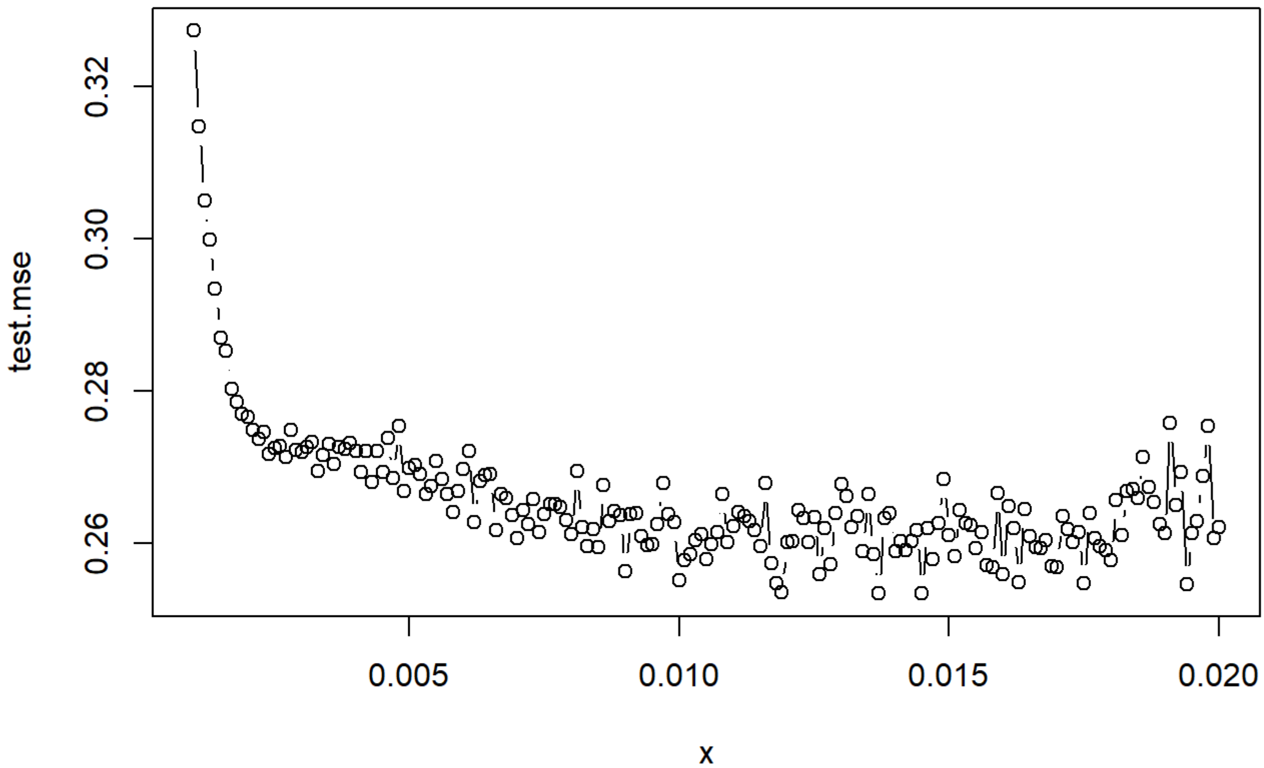
```
set.seed(100)
num = sample(1:nrow(data), 200)
train = data.frame(data[num,])
test = data.frame(data[-num,])
```

(c),(d).

```
set.seed(100)

x = seq(0.001,0.02,0.0001)

train.mse = rep(NA,length(x))
test.mse = rep(NA,length(x))

for (i in x){
  boost.Hitters = gbm(Salary~., data=train,distribution = "gaussian", n.trees = 1000,interactio
n.depth = 4, shrinkage = i)

  Hitters.pred1 = predict(boost.Hitters,newdata = train, n.trees = 1000)
  train.mse[which(i==x)] = mean((Hitters.pred1-train$Salary)^2)

  Hitters.pred2 = predict(boost.Hitters,newdata = test, n.trees = 1000)
  test.mse[which(i==x)] = mean((Hitters.pred2-test$Salary)^2)
}

plot(x,train.mse,type="b")
```




```
plot(x,test.mse,type="b")
```



```
min(test.mse)
```

```
## [1] 0.2534242
```

(e).

```
#linear model
set.seed(100)
lm.fit = lm(Salary~., data=train)
lm.preds = predict(lm.fit, newdata = test)
lm.mse = mean((test$Salary-lm.preds)^2)
lm.mse
```

```
## [1] 0.5832229
```

```
# ridge model with cross validation
library(glmnet)
```

```
## 载入需要的程辑包：Matrix
```

```
## Loaded glmnet 4.1-4
```

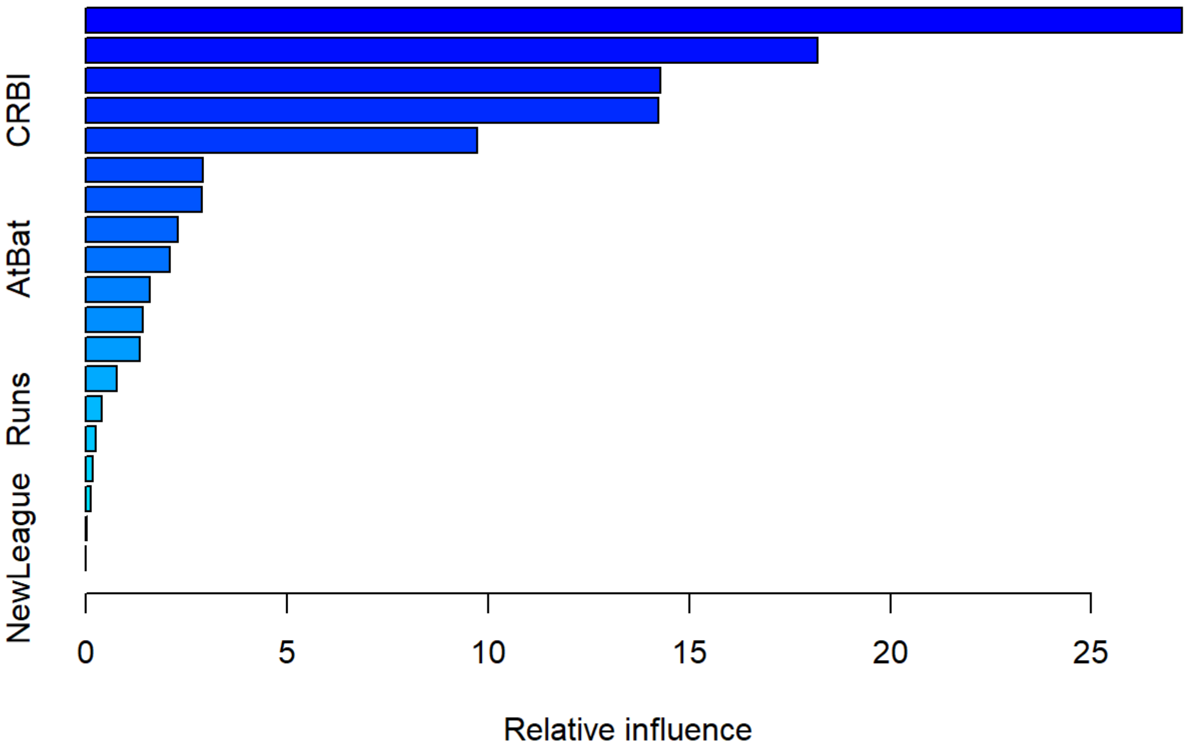
```
set.seed(100)

train_mat = model.matrix(Salary~.,train)
test_mat = model.matrix(Salary~.,test)
y.train = train$Salary
ridge.mod = glmnet(train_mat, y.train, alpha = 0)

crossv=cv.glmnet(train_mat, y.train, alpha=0)
bestlam=crossv$lambda.min
ridge.pred=predict(ridge.mod, s=bestlam, newx = test_mat)
mean((test$Salary-ridge.pred)^2)
```

```
## [1] 0.5384082
```

We can find that test MSE of linear model and ridge regression are 0.58 and 0.53, which is higher than that of boosting model.

f.

```
set.seed(100)
best = gbm(Salary~., data=train, distribution = "gaussian", n.trees = 1000,interaction.depth =
4, shrinkage = x[which.min(x)])
summary(best)
```




```
##                 var      rel.inf
## CAtBat       CAtBat 27.249636823
## CHits         CHits 18.199142445
## CRuns         CRuns 14.277416468
## CRBI           CRBI 14.228234822
## CWalks       CWalks  9.736542219
## CHmRun       CHmRun  2.911205736
## Years         Years  2.890612111
## Hits           Hits  2.284344489
## AtBat         AtBat  2.098943514
## PutOuts     PutOuts  1.605977431
## HmRun         HmRun  1.430591117
## Walks         Walks  1.339469485
## RBI             RBI  0.765145726
## Runs           Runs  0.398947381
## Errors       Errors  0.251281790
## Assists     Assists  0.166257793
## League       League  0.122210002
## Division   Division  0.039889722
## NewLeague NewLeague  0.004150927
```

The most important variables are CRuns, CAtBat and CHits.

(g).

```
set.seed(100)
library(randomForest)
bag = randomForest(Salary~.,data = train,mtry=19,importance=T)
bag.pred = predict(bag,newdata = test)
mean((test$Salary-bag.pred)^2)
```

```
## [1] 0.2053029
```

The test MSE is 0.20, similar to boosting model.