Final Question

Huafeng Zhang (217107707)

2019.7.26

1. load data, remove na, log-transform

```
library(ISLR)
hit <- Hitters #322 obs
hit <- na.omit(hit) #263 obs
hit$Salary <- log(hit$Salary)
#names(hit)
```

2. split to train/test

```
tr.hit <- hit[1:200,]
te.hit <- hit[201:nrow(hit),]
```

3. Random Forest

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.6.1

## randomForest 4.6-14

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

```
set.seed(345)

ntree <- c(25,100,500,1000)
tree.list <- list()

for (i in ntree){
  rf <- randomForest(Salary ~ ., data = tr.hit, ntree = i, importance = T)
  rf.ps <- predict(rf, newdata = te.hit)
  rf.mse <- mean((rf.ps-te.hit$Salary)^2)
  print(paste('MSE of', i, 'trees is', rf.mse))
  tree.list[[i]] <- rf
}
```

```
## [1] "MSE of 25 trees is 0.237691767512926"
## [1] "MSE of 100 trees is 0.223005375334518"
## [1] "MSE of 500 trees is 0.215187150925909"
## [1] "MSE of 1000 trees is 0.215948559578728"
```

Given the results, using 500 trees gives the best result with the smallest MSE of 0.215.

4. identify the most important variables

```
#use 500 trees stored from previous step
importance(tree.list[[500]], type = 2)
```

```
##           IncNodePurity
## AtBat         5.9992625
## Hits          5.1796468
## HmRun         1.9908966
## Runs          3.9207067
## RBI           4.5690186
## Walks         5.7286570
## Years         5.5554880
## CAtBat       35.5913442
## CHits        30.7677954
## CHmRun        7.5493866
## CRuns        22.3492466
## CRBI         12.3107149
## CWalks       16.0775664
## League        0.1459558
## Division      0.2032052
## PutOuts       2.6576243
## Assists       1.4374749
## Errors        1.3067447
## NewLeague     0.2155866
```

The most important variables (with top 5 largest values) associated with predicting Salary (with 500 trees) are 'CAtBat', 'CHits', 'CRuns', 'CWalks', and 'CRBI'.

5. Boosting and plot of learning rates vs. train MSEs

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.6.1
```

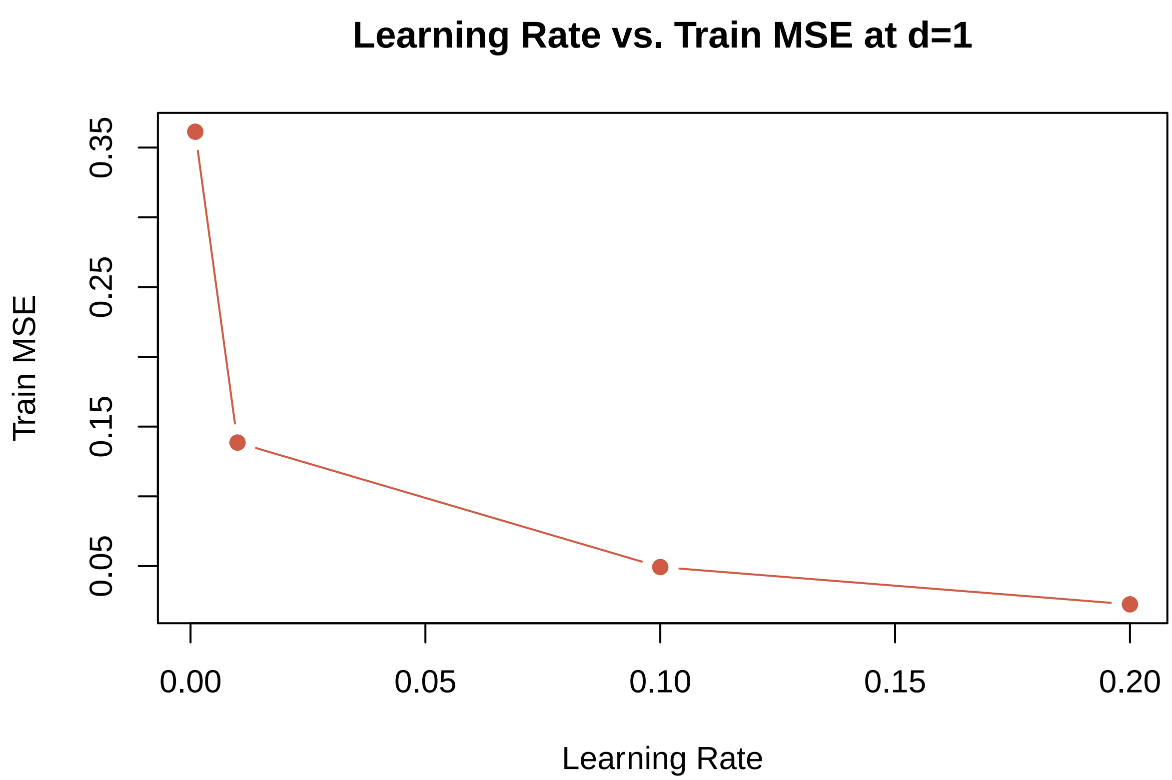
```
## Loaded gbm 2.1.5
```

```
lambda <- c(0.001,0.01,0.1,0.2)
btrain.list <- list()
```

```r
for (i in lambda){
  set.seed(5671)
  boost <- gbm(formula = Salary ~ ., data = tr.hit,
               distribution = "gaussian", n.trees = 1000,
               interaction.depth = 1, shrinkage = i, verbose = F)
  btr.ps <- predict(boost, n.trees = boost$n.trees, newdata = tr.hit)
  btr.mse <- mean((btr.ps-tr.hit$Salary)^2)
  print(paste('Train MSE of', i, 'learning rate at d=1 is', btr.mse))
  btrain.list <- c(btrain.list, btr.mse)
}
```

```
## [1] "Train MSE of 0.001 learning rate at d=1 is 0.361349642227683"
## [1] "Train MSE of 0.01 learning rate at d=1 is 0.13852943507682"
## [1] "Train MSE of 0.1 learning rate at d=1 is 0.0493054349792426"
## [1] "Train MSE of 0.2 learning rate at d=1 is 0.0225802738767658"
```

```r
plot(lambda, btrain.list,
     type = 'b', pch=19, col = 'coral3',
     main = 'Learning Rate vs. Train MSE at d=1',
     xlab = 'Learning Rate',
     ylab = 'Train MSE')
```



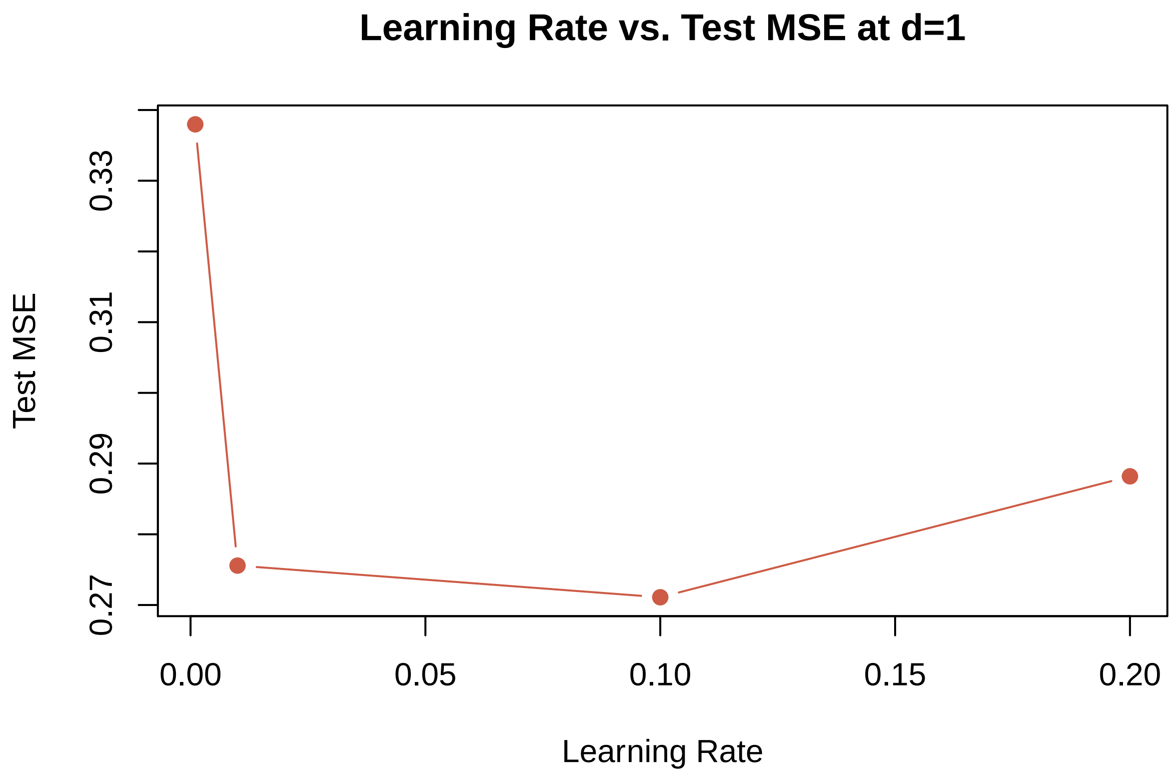
Given the results, using $\lambda = 0.2$ gives the best result with the smallest train MSE of 0.02258 at $d = 1$.

6. plot of learning rates vs. test MSEs

```r
library(gbm)

lambda <- c(0.001,0.01,0.1,0.2)
btest.list1 <- list()

for (i in lambda){
  set.seed(5671)
  #I put the 5671 seed here for the same boosted tree in Q8
  #they have the same MSE (0.27557) as I tested
  boost <- gbm(formula = Salary ~ ., data = tr.hit,
           distribution = "gaussian", n.trees = 1000,
           interaction.depth = 1, shrinkage = i, verbose = F)
  bte.ps <- predict(boost, n.trees = boost$n.trees, newdata = te.hit)
  bte.mse <- mean((bte.ps-te.hit$Salary)^2)
  print(paste('Test MSE of', i, 'learning rate at d=1 is', bte.mse))
  btest.list1 <- c(btest.list1, bte.mse)
}
```

```
## [1] "Test MSE of 0.001 learning rate at d=1 is 0.33797272373508"
## [1] "Test MSE of 0.01 learning rate at d=1 is 0.275573437621729"
## [1] "Test MSE of 0.1 learning rate at d=1 is 0.271098591991139"
## [1] "Test MSE of 0.2 learning rate at d=1 is 0.288197714911153"
```

```r
plot(lambda, btest.list1,
     type = 'b', pch=19, col = 'coral3',
     main = 'Learning Rate vs. Test MSE at d=1',
     xlab = 'Learning Rate',
     ylab = 'Test MSE')
```



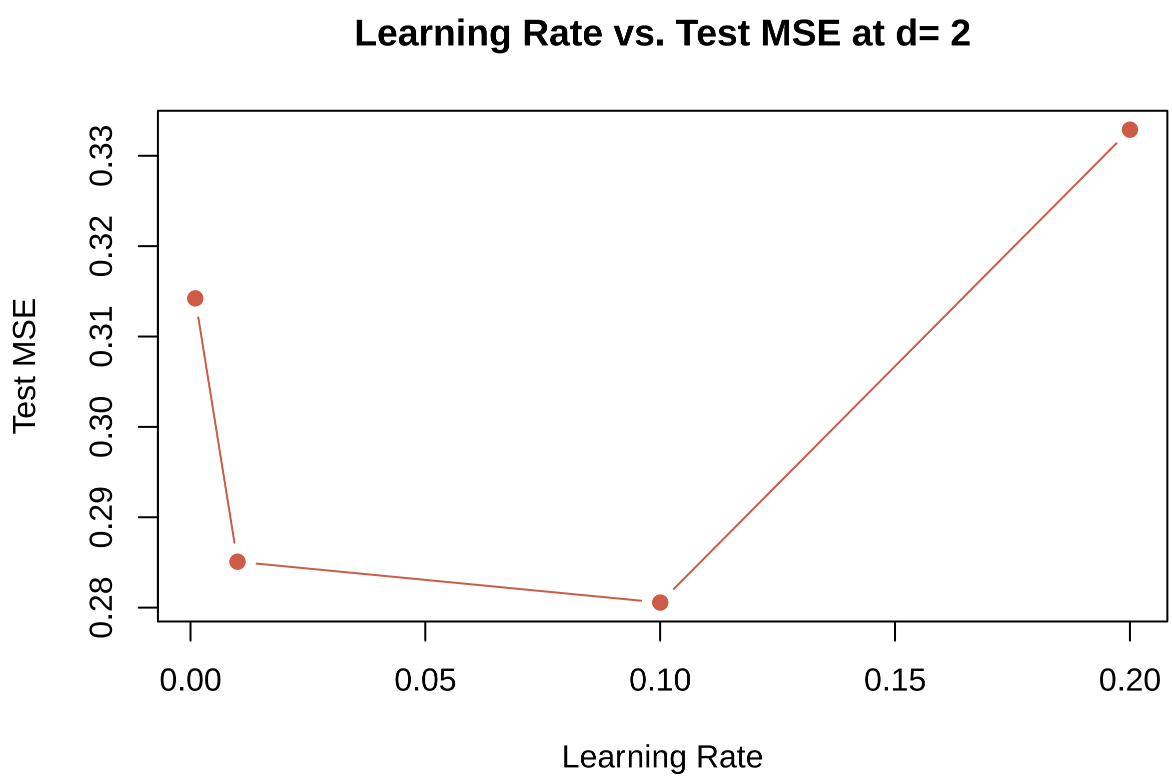
Given the results, using $\lambda = 0.1$ gives the best result with the smallest test MSE of 0.27557 at $d = 1$.

7. try different interaction depths

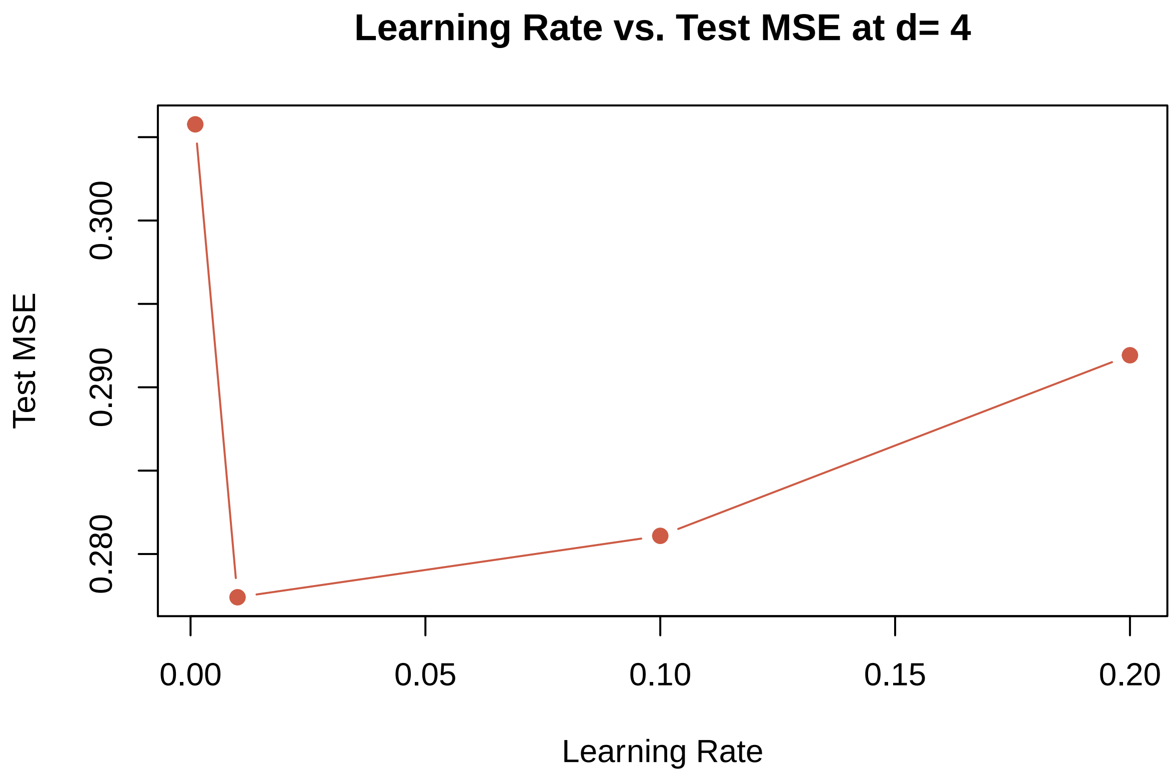
```
library(gbm)
set.seed(8432)

lambda <- c(0.001,0.01,0.1,0.2)
depth <- c(2,4) #test d=2,4 since we already have d=1 from Q6

for (i in depth){
  test.list24 <- list()
  for (j in lambda){
    boost <- gbm(formula = Salary ~ ., data = tr.hit,
           distribution = "gaussian", n.trees = 1000,
           interaction.depth = i, shrinkage = j, verbose = F)
    bte.ps <- predict(boost, n.trees = boost$n.trees, newdata = te.hit)
    bte.mse <- mean((bte.ps-te.hit$Salary)^2)
    print(paste('Test MSE of', j, 'learning rate at d=', i, 'is', bte.mse))
    test.list24 <- c(test.list24, bte.mse)
  }
  plot(lambda, test.list24,
```

```
        type = 'b', pch=19, col = 'coral3',
        main = paste('Learning Rate vs. Test MSE at d=', i),
        xlab = 'Learning Rate',
        ylab = 'Test MSE')
}
```

```
## [1] "Test MSE of 0.001 learning rate at d= 2 is 0.314219169368212"
## [1] "Test MSE of 0.01 learning rate at d= 2 is 0.285078549797528"
## [1] "Test MSE of 0.1 learning rate at d= 2 is 0.280559872912274"
## [1] "Test MSE of 0.2 learning rate at d= 2 is 0.332887072450714"
```




```
## [1] "Test MSE of 0.001 learning rate at d= 4 is 0.305767065846022"
## [1] "Test MSE of 0.01 learning rate at d= 4 is 0.277407785450083"
## [1] "Test MSE of 0.1 learning rate at d= 4 is 0.281089652114456"
## [1] "Test MSE of 0.2 learning rate at d= 4 is 0.291919767871503"
```



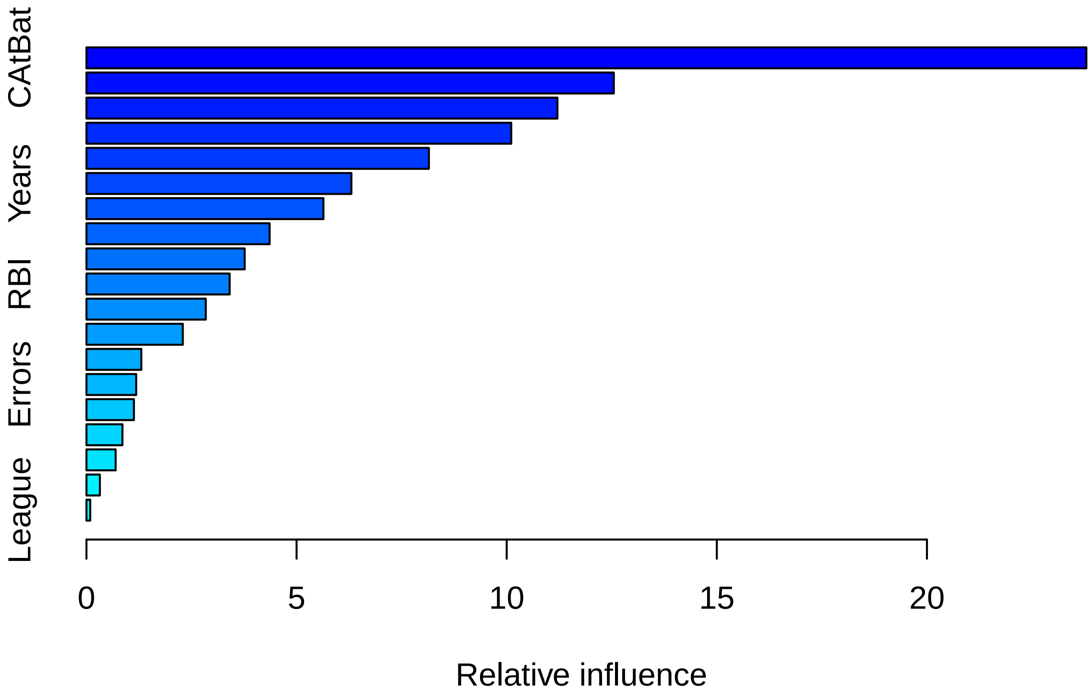
Given the results, using $\lambda = 0.1$ gives the best result with the smallest MSE of 0.281 for $d = 2$.
Given the results, using $\lambda = 0.01$ gives the best result with the smallest MSE of 0.277 for $d = 4$.

8. most important variables from boosted tree

```
#use the best result (smallest MSE) at lambda=0.01 and d=1
set.seed(5671)
b1 <- gbm(Salary ~ ., data = tr.hit,
          distribution = "gaussian", n.trees = 1000,
          interaction.depth = 1, shrinkage = 0.01, verbose = F)
#the following steps for testing the same MSE (0.27557) as in Q6
ps1 <- predict(b1, n.trees = b1$n.trees, newdata = te.hit)
mse1 <- mean((ps1-te.hit$Salary)^2)
print(paste('Test MSE of 0.01 learning rate at d=1 is', mse1))
```

```
## [1] "Test MSE of 0.01 learning rate at d=1 is 0.275573437621729"
```

```
summary(b1)
```




```
##                 var     rel.inf
## CAtBat       CAtBat 23.79318550
## CHits         CHits 12.54906053
## CWalks       CWalks 11.20742421
## CRBI           CRBI 10.10825402
## CRuns         CRuns  8.14898301
## Years         Years  6.30444862
## CHmRun       CHmRun  5.63893116
## Hits           Hits  4.35933434
## Walks         Walks  3.76603402
## RBI             RBI  3.40836290
## PutOuts     PutOuts  2.84039720
## AtBat         AtBat  2.29325151
## HmRun         HmRun  1.30613742
## Errors       Errors  1.18429451
## Runs           Runs  1.13020572
## Assists     Assists  0.85673855
## Division   Division  0.69522877
## NewLeague NewLeague  0.31950074
## League       League  0.09022727
```

Given the result, 'CAtBat', 'CHits', 'CWalks', 'CRBI', 'CRuns' are the most important (top 5) variables.

9. compare test MSEs from RF and boosting

In random forest, using 500 trees gives the best result with the smallest test MSE of 0.215, which is better than any of the test MSE from boosting.
The determinations of the most important variables gives the same top 5 variables from both methods, although the ranks are different.