# Individual Assignment 8

Group 3

11/3/2021

Exercises 8.4 Problem #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)

set.seed(2)

train = sample(1:nrow(Carseats),200)

Carseats.test = Carseats[-train,]
```

(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.

```
#bagging

library(randomForest)

## randomForest 4.6-14

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

set.seed(1)

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

bag.Carseats

##
## Call:
##  randomForest(formula = Sales ~ ., data = Carseats, mtry = 10, importance = TRUE, subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##           Mean of squared residuals: 2.953352
##                     % Var explained: 60.36
```

```r
yhat.bag = predict(bag.Carseats, newdata = Carseats.test)


mean((yhat.bag-Carseats.test$Sales)^2)

## [1] 2.555523

#test MSE is 2.555523
importance(bag.Carseats)

##                 %IncMSE IncNodePurity
## CompPrice   25.64599599    213.096278
## Income       5.67740403     76.385083
## Advertising 12.22093633    104.986108
## Population   0.59715138     61.294727
## Price       56.50749020    535.237246
## ShelveLoc   41.05022443    283.698935
## Age          9.11485795    118.103039
## Education   -0.05758758     39.442533
## Urban        0.79459812      8.828001
## US           1.45384950      7.611851

#Price, ShelveLoc and CompPrice are most important
```

(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.

```r
library(randomForest)

set.seed(1)

rf.Carseats = randomForest(Sales~.,data=Carseats, subset=train, mtry=3,
importance=TRUE)

rf.Carseats

##
## Call:
##  randomForest(formula = Sales ~ ., data = Carseats, mtry = 3,
importance = TRUE, subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 3.334325
##                     % Var explained: 55.24
```

```r
yhat.rf = predict(rf.Carseats, newdata = Carseats.test)

mean((yhat.rf-Carseats.test$Sales)^2)

## [1] 3.20635

#test MSE is 3.20635
importance(rf.Carseats)

##                 %IncMSE IncNodePurity
## CompPrice   11.34376552     158.83906
## Income       6.40545807     121.23469
## Advertising 10.24191488     123.49672
## Population   0.09842459      98.17871
## Price       36.27025901     403.38433
## ShelveLoc   31.07819285     241.99669
## Age          6.77865583     145.64539
## Education    1.51515779      65.62168
## Urban        1.14914879      14.60700
## US           2.71845074      14.78064

#Price, ShelveLoc are most important
#A larger m will decrease test MSE
```

Problem #10: 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.

```r
Hitters=na.omit(Hitters)
log_salary=log(Hitters$Salary)
Hitters=data.frame(Hitters,log_salary)
```

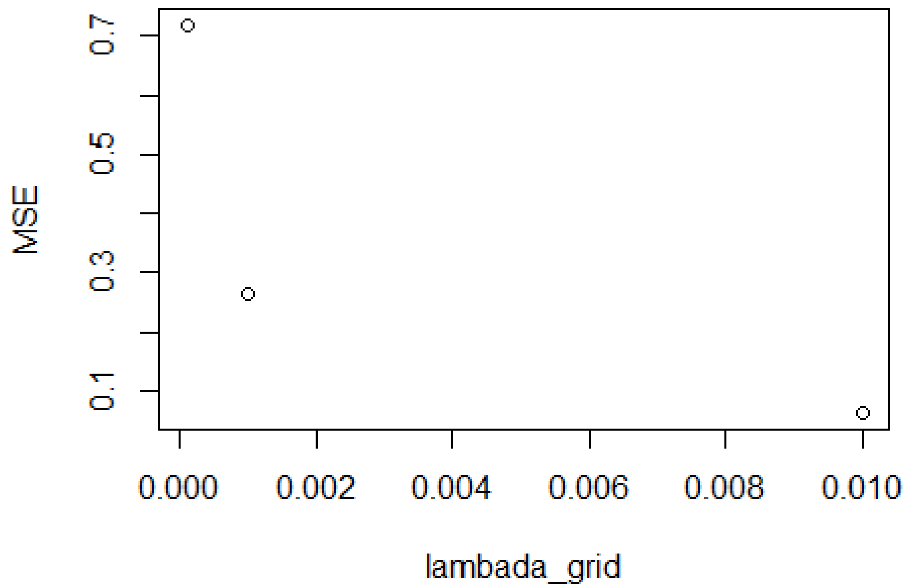
(b) Create a training set consisting of the first 200 observations, and a test set consisting of the remaining observations.

```r
train=c(1:200)
test=-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.

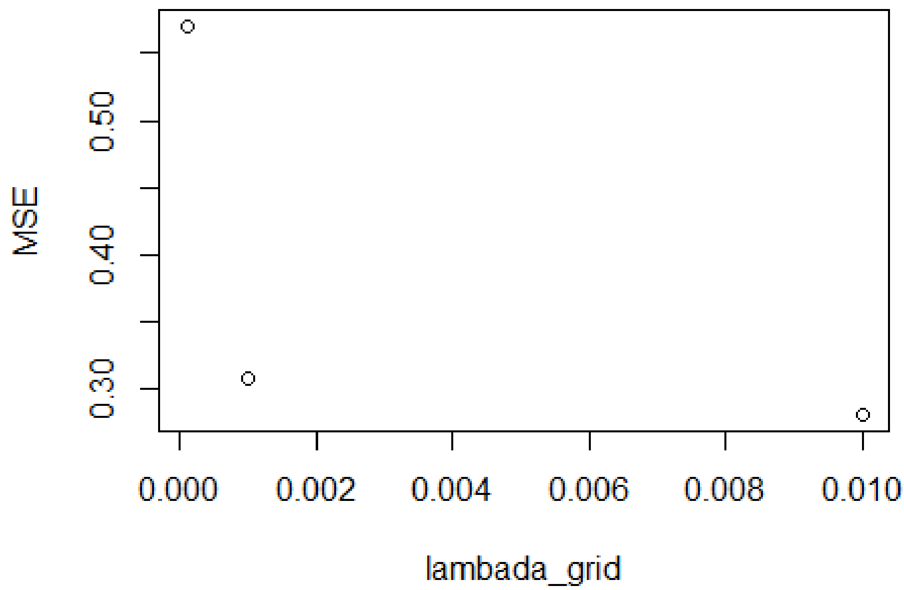
```r
library(gbm)

## Loaded gbm 2.1.8

lambada_grid=c(0.01,0.001,0.0001)
MSE=rep(0,3)
for(i in 1:3){
  boost.Hitters= gbm(log_salary~.-Salary,Hitters[train,],distribution =
"gaussian", n.trees=1000,interaction.depth=4,shrinkage = lambada_grid[i] )

yhat.boost=predict(boost.Hitters,newdata=Hitters[train,],n.trees=1000,interac
```

```
tion.depth=4,shrinkage = lambada_grid[i] )
  MSE[i]=mean((yhat.boost - Hitters[train,]$log_salary)^2) }


plot(lambada_grid,MSE)
```




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

```
for(i in 1:3){
  boost.Hitters= gbm(log_salary~.-Salary,Hitters[train,],distribution =
"gaussian", n.trees=1000,interaction.depth=4,shrinkage = lambada_grid[i] )

yhat.boost=predict(boost.Hitters,newdata=Hitters[test,],n.trees=1000,interact
ion.depth=4,shrinkage = lambada_grid[i] )
  MSE[i]=mean((yhat.boost - Hitters[test,]$log_salary)^2) }


plot(lambada_grid,MSE)
```




```
MSE

## [1] 0.2799206 0.3065863 0.5705474
```

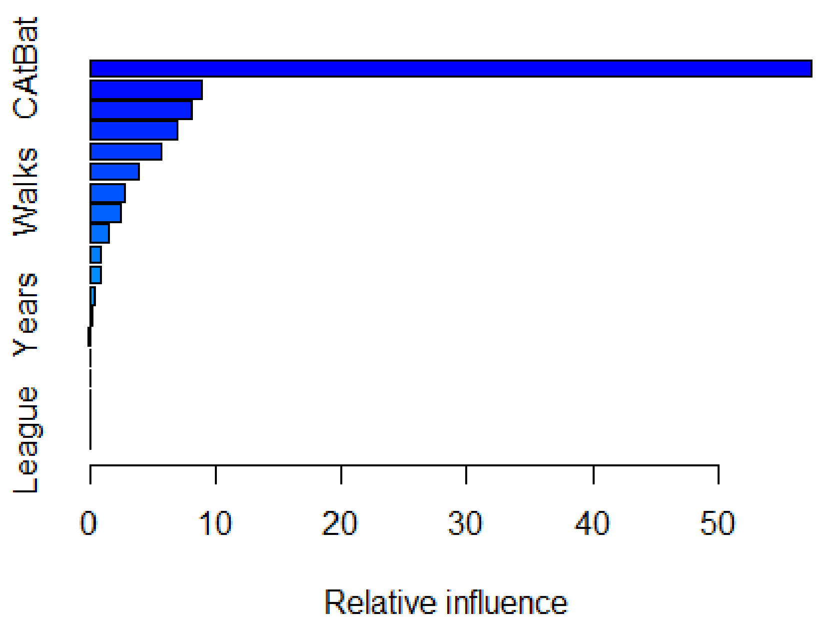
(e) Compare the test MSE of boosting to the test MSE that results from applying two of the regression approaches seen in Chapters 3 and 6.

```
library(glmnet)

## 载入需要的程辑包：Matrix

## Loaded glmnet 4.1-2

x = model.matrix(log_salary~.-Salary,Hitters)[,-2]
y = Hitters$log_salary
grid = 10^seq(10,-2,length=100)
#ridge regression
ridge.mod = glmnet(x,y,alpha=0,lambda=grid,thresh = 1e-12)
set.seed(1)
cv.out = cv.glmnet(x[train,],y[train], alpha=0) ##default is 10-fold cross-validation.
bestlam = cv.out$lambda.min
ridge.pred=predict(ridge.mod,s=bestlam,x=x[train,],y=y[train],newx=x[test,],exact = T)
mean((ridge.pred-y[test])^2)

## [1] 0.4521353
```

```r
#LASSO
lasso.mod = glmnet(x,y,alpha=1,lambda=grid,thresh = 1e-12)
set.seed(1)
cv.out = cv.glmnet(x[train,],y[train], alpha=1) ##default is 10-fold cross-validation.
bestlam = cv.out$lambda.min
lasso.pred=predict(lasso.mod,s=bestlam,x=x[train,],y=y[train],newx=x[test,],exact = T)
mean((lasso.pred-y[test])^2)

## [1] 0.4696766

#linear regression

glm.fit = glm(log_salary~.-Salary,data = Hitters ,subset = train)
yhat.glm = predict(glm.fit, newdata = Hitters[test,], type = "response")
mean((yhat.glm - Hitters[test,]$log_salary) ^ 2)

## [1] 0.4917959

# The test MSE of boosting With shrinkage of 0.01 is smaller than linear
regression, ridge regression and LASSO
```

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

```r
summary(boost.Hitters)
```




```
##                  var       rel.inf
## CAtBat        CAtBat 57.436983418
```

```
## CRuns         CRuns  8.843024418
## CHits         CHits  8.051207483
## CWalks       CWalks  6.904736267
## CRBI           CRBI  5.717567478
## AtBat         AtBat  3.874379438
## Walks         Walks  2.758656329
## CHmRun       CHmRun  2.423226262
## Hits           Hits  1.541525651
## RBI             RBI  0.913012418
## Runs           Runs  0.801367327
## PutOuts     PutOuts  0.382220942
## Years         Years  0.116283565
## HmRun         HmRun  0.104311732
## Assists     Assists  0.072362761
## Errors       Errors  0.027614873
## NewLeague NewLeague  0.011956870
## Division   Division  0.011114482
## League       League  0.008448285

#CAtBat appears to be the most important predictors in the boosted model
```

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

```
set.seed(1)
bag.Hitters = randomForest(log_salary~.-Salary, data=Hitters, subset=train,
mtry=19, importance=TRUE)

bag.Hitters

##
## Call:
##  randomForest(formula = log_salary ~ . - Salary, data = Hitters,      mtry
= 19, importance = TRUE, subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 19
##
##           Mean of squared residuals: 0.2178554
##                     % Var explained: 73.82

yhat.bag = predict(bag.Hitters, newdata = Hitters[test,])


mean((yhat.bag-Hitters[test,]$log_salary)^2)

## [1] 0.2301184

#test MSE is 0.2301184
```