Homework 4.4 - Hitters dataset

Hamed

2/26/2020

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

library(ISLR)  
Hitters=as.data.frame(Hitters)  
dim(Hitters)

## [1] 322 20

str(Hitters)

## 'data.frame': 322 obs. of 20 variables:  
## $ AtBat : int 293 315 479 496 321 594 185 298 323 401 ...  
## $ Hits : int 66 81 130 141 87 169 37 73 81 92 ...  
## $ HmRun : int 1 7 18 20 10 4 1 0 6 17 ...  
## $ Runs : int 30 24 66 65 39 74 23 24 26 49 ...  
## $ RBI : int 29 38 72 78 42 51 8 24 32 66 ...  
## $ Walks : int 14 39 76 37 30 35 21 7 8 65 ...  
## $ Years : int 1 14 3 11 2 11 2 3 2 13 ...  
## $ CAtBat : int 293 3449 1624 5628 396 4408 214 509 341 5206 ...  
## $ CHits : int 66 835 457 1575 101 1133 42 108 86 1332 ...  
## $ CHmRun : int 1 69 63 225 12 19 1 0 6 253 ...  
## $ CRuns : int 30 321 224 828 48 501 30 41 32 784 ...  
## $ CRBI : int 29 414 266 838 46 336 9 37 34 890 ...  
## $ CWalks : int 14 375 263 354 33 194 24 12 8 866 ...  
## $ League : Factor w/ 2 levels "A","N": 1 2 1 2 2 1 2 1 2 1 ...  
## $ Division : Factor w/ 2 levels "E","W": 1 2 2 1 1 2 1 2 2 1 ...  
## $ PutOuts : int 446 632 880 200 805 282 76 121 143 0 ...  
## $ Assists : int 33 43 82 11 40 421 127 283 290 0 ...  
## $ Errors : int 20 10 14 3 4 25 7 9 19 0 ...  
## $ Salary : num NA 475 480 500 91.5 750 70 100 75 1100 ...  
## $ NewLeague: Factor w/ 2 levels "A","N": 1 2 1 2 2 1 1 1 2 1 ...

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

# remove rows with missing values   
Hitters = Hitters[!is.na(Hitters$Salary),]  
  
#log transform salary  
Hitters$Salary = log(Hitters$Salary)

## (b) Create a training set consisting of 65% of the observations, and a test set consisting of the remaining observations.

set.seed(123)  
smp\_size <-floor(0.65 \* nrow(Hitters))  
train\_ind <-sample(seq\_len(nrow(Hitters)), size = smp\_size)  
train.Hitters = Hitters[train\_ind,]  
dim(train.Hitters)

## [1] 170 20

test.Hitters = Hitters[-train\_ind,]  
dim(test.Hitters)

## [1] 93 20

## (c) Perform boosting using XGBoost 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(xgboost)  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(gbm)

## Loaded gbm 2.1.5

# Ensure the R-studio can handle plots with large margins  
par("mar")

## [1] 5.1 4.1 4.1 2.1

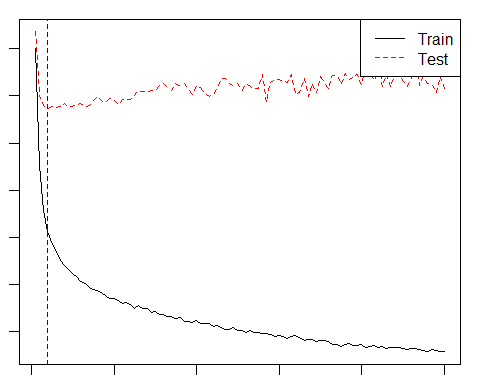
par(mar=c(1,1,1,1))  
  
#Grid search the shrinkage  
set.seed(1)  
shrinkGrid = seq(.1,.001,by=-.001)  
shrinkGrid

## [1] 0.100 0.099 0.098 0.097 0.096 0.095 0.094 0.093 0.092 0.091 0.090 0.089  
## [13] 0.088 0.087 0.086 0.085 0.084 0.083 0.082 0.081 0.080 0.079 0.078 0.077  
## [25] 0.076 0.075 0.074 0.073 0.072 0.071 0.070 0.069 0.068 0.067 0.066 0.065  
## [37] 0.064 0.063 0.062 0.061 0.060 0.059 0.058 0.057 0.056 0.055 0.054 0.053  
## [49] 0.052 0.051 0.050 0.049 0.048 0.047 0.046 0.045 0.044 0.043 0.042 0.041  
## [61] 0.040 0.039 0.038 0.037 0.036 0.035 0.034 0.033 0.032 0.031 0.030 0.029  
## [73] 0.028 0.027 0.026 0.025 0.024 0.023 0.022 0.021 0.020 0.019 0.018 0.017  
## [85] 0.016 0.015 0.014 0.013 0.012 0.011 0.010 0.009 0.008 0.007 0.006 0.005  
## [97] 0.004 0.003 0.002 0.001

#Define the MSE   
MSE = matrix(NA,nrow=length(shrinkGrid),ncol=2)  
  
# Train xgboost model using the shrinkage  
for(i in 1:length(shrinkGrid)){  
 lambda = shrinkGrid[i]  
 boost.hitters = gbm(Salary~.,data=train.Hitters,distribution="gaussian",n.trees=1000,shrinkage=lambda)  
 boost.pred.train = predict(boost.hitters,train.Hitters,n.trees=1000)  
 boost.pred.test = predict(boost.hitters,test.Hitters,n.trees=1000)  
 train.MSE = mean((boost.pred.train - train.Hitters$Salary)^2)  
 test.MSE = mean((boost.pred.test - test.Hitters$Salary)^2)  
 MSE[i,1] = train.MSE  
 MSE[i,2] = test.MSE  
}  
best.lambda = shrinkGrid[which.min(MSE[,2])]  
best.lambda

## [1] 0.004

# Plot the MSE against the shrinkage values for th train and test dataset  
matplot(x=shrinkGrid,y=MSE,type="l",xlab="lambda",ylab="MSE")  
legend("topright",legend = c("Train","Test"),col=c("black","red"),lty=c(1,2))  
abline(v=best.lambda,col="blue",lty=2,lwd=.5)



#The best MSE   
print(paste0("Best Test MSE: ", MSE[which.min(MSE[,2]),2]))

## [1] "Best Test MSE: 0.286644491660698"

## (d) Compare the test MSE of boosting to the test MSE that results from applying a linear regression approach. We shall apply two regression approaches; (ridge and lasso)

library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 3.0-2

#split the dataset  
x.train.Hitters = model.matrix(Salary~.,train.Hitters)[,-1]  
x.test.Hitters = model.matrix(Salary~.,test.Hitters)[,-1]  
y.train.Hitters = train.Hitters$Salary  
y.test.Hitters = test.Hitters$Salary  
  
#define lambda   
lambdaGrid = 10^seq(10,-2,length=100)  
  
# Ridge Linear regression cross-validated  
cv.ridge = cv.glmnet(x.train.Hitters,y.train.Hitters,alpha=0)  
bestlam.ridge = cv.ridge$lambda.min  
  
#Lasso linear regression cross-validated  
cv.lasso = cv.glmnet(x.train.Hitters,y.train.Hitters,alpha=1)  
bestlam.lasso = cv.lasso$lambda.min  
  
#Ridge and lasso regression without cross-validation   
ridge.hitters = glmnet(x.train.Hitters,y.train.Hitters,alpha=0,lambda=lambdaGrid)  
lasso.hitters = glmnet(x.train.Hitters,y.train.Hitters,alpha=1,lambda=lambdaGrid)  
  
#Prediction  
ridge.pred = predict(ridge.hitters,s=bestlam.ridge,newx=x.test.Hitters)  
lasso.pred = predict(lasso.hitters,s=bestlam.lasso,newx=x.test.Hitters)  
  
#Mean squared error   
ridge.MSE.test = mean((ridge.pred - y.test.Hitters)^2)  
lasso.MSE.test = mean((lasso.pred - y.test.Hitters)^2)  
print(paste0("Boost Test MSE: ", MSE[which.min(MSE[,2]),2]))

## [1] "Boost Test MSE: 0.286644491660698"

print(paste0("Ridge Test MSE: ", ridge.MSE.test))

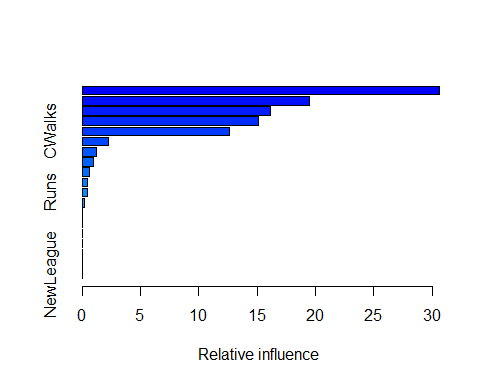
## [1] "Ridge Test MSE: 0.493528930698876"

print(paste0("Lasso Test MSE: ", lasso.MSE.test))

## [1] "Lasso Test MSE: 0.505343822864917"

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

summary(boost.hitters)



## var rel.inf  
## CRuns CRuns 30.63233511  
## CHits CHits 19.43477600  
## CAtBat CAtBat 16.12177144  
## CRBI CRBI 15.12685201  
## CWalks CWalks 12.58921424  
## CHmRun CHmRun 2.24536342  
## Years Years 1.19225050  
## Hits Hits 0.94377105  
## Walks Walks 0.60363977  
## RBI RBI 0.45304197  
## Runs Runs 0.41438550  
## AtBat AtBat 0.20095293  
## HmRun HmRun 0.04164606  
## League League 0.00000000  
## Division Division 0.00000000  
## PutOuts PutOuts 0.00000000  
## Assists Assists 0.00000000  
## Errors Errors 0.00000000  
## NewLeague NewLeague 0.00000000

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

library(randomForest)

## randomForest 4.6-14

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

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

bag.hitters = randomForest(Salary~.,data=train.Hitters,ntree=1000,mtry=19,importance=TRUE)  
  
bag.pred = predict(bag.hitters,newdata=test.Hitters)  
bag.MSE.test = mean((bag.pred - test.Hitters$Salary)^2)  
  
print(paste0("Bagging Test MSE: ",bag.MSE.test))

## [1] "Bagging Test MSE: 0.317231830929847"

# Bagging appears to have slightly outperformed boosting, ridge, and lasso.