Assignment: Chapter 8 Exercises (Week 7)

Amol Gote

08/23/2020

# Problem 9 This problem involves the OJ data set which is part of the ISLR package.

training\_oj\_subset <- sample(nrow(OJ), 800)  
training\_oj\_ds = OJ[training\_oj\_subset, ]  
test\_oj\_ds = OJ[-training\_oj\_subset, ]  
nrow(training\_oj\_ds)

## [1] 800

nrow(test\_oj\_ds)

## [1] 270

tree\_oj <- tree(Purchase∼., OJ ,subset = training\_oj\_subset )

prediction <- predict(tree\_oj, test\_oj\_ds, type="class")  
table(prediction, test\_oj\_ds$Purchase)

##   
## prediction CH MM  
## CH 137 32  
## MM 20 81

test\_error\_rate <- mean(prediction != test\_oj\_ds$Purchase)  
test\_error\_rate\_percentage <- test\_error\_rate \* 100  
test\_error\_rate\_percentage

## [1] 19.25926

test\_accuracy\_rate <- mean(prediction == test\_oj\_ds$Purchase)  
test\_accuracy\_rate\_percentage <- test\_accuracy\_rate \* 100  
test\_accuracy\_rate\_percentage

## [1] 80.74074

optimal\_tree <- cv.tree(tree\_oj, FUN = prune.tree)

# 9g Produce a plot with tree size on the x-axis and cross-validated classification error rate on the y-axis.

tree\_plot <- data.frame(x=optimal\_tree$size, y=optimal\_tree$dev)  
ggplot(tree\_plot, aes(x=x,y=y)) +   
 geom\_point() +   
 geom\_line() +   
 xlab("Tree Size") +   
 ylab("Deviance")



# 9h Which tree size corresponds to the lowest cross-validated classification error rate?

Tree size which corresponds to the lowest cross-validated classification error rate is: 7

# 9i Produce a pruned tree corresponding to the optimal tree size obtained using cross-validation. If cross-validation does not lead to selection of a pruned tree, then create a pruned tree with five terminal nodes.

pruned\_tree <- prune.tree(tree\_oj, best = 7)

# 9j Compare the training error rates between the pruned and unpruned trees. Which is higher?

summary(pruned\_tree)

##   
## Classification tree:  
## snip.tree(tree = tree\_oj, nodes = 5L)  
## Variables actually used in tree construction:  
## [1] "LoyalCH" "PriceDiff"  
## Number of terminal nodes: 7   
## Residual mean deviance: 0.7583 = 601.3 / 793   
## Misclassification error rate: 0.1662 = 133 / 800

summary(tree\_oj)

##   
## Classification tree:  
## tree(formula = Purchase ~ ., data = OJ, subset = training\_oj\_subset)  
## Variables actually used in tree construction:  
## [1] "LoyalCH" "PriceDiff"  
## Number of terminal nodes: 8   
## Residual mean deviance: 0.7402 = 586.3 / 792   
## Misclassification error rate: 0.16 = 128 / 800

Misclassification rate on pruned tree is is higher

# 9k Compare the test error rates between the pruned and unpruned trees. Which is higher?

test\_error\_rate\_pruned <- mean(predict(pruned\_tree, test\_oj\_ds, type = "class") != test\_oj\_ds$Purchase)  
test\_error\_rate\_pruned

## [1] 0.1814815

test\_error\_rate\_unpruned <- mean(prediction != test\_oj\_ds$Purchase)  
test\_error\_rate\_unpruned

## [1] 0.1925926

Test error rate of pruned tree is 0.1814815 and unpruned tree is 0.1925926. Pruned tree error rate is lower.

# Problem 10 - We now use boosting to predict Salary in the Hitters data set.

# 10a Remove the observations for whom the salary information is unknown, and then log-transform the salaries.

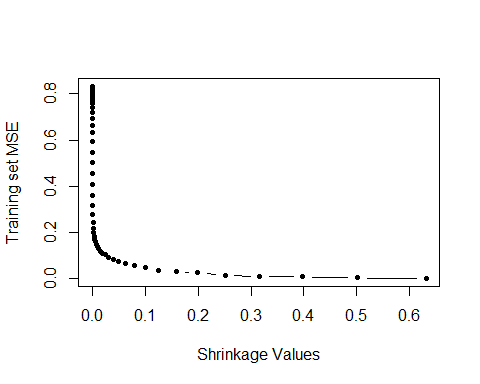
hitters\_ds <- Hitters   
no\_salary <- hitters\_ds %>%  
 filter(is.na(Salary))  
no\_salary\_row\_count <- nrow(no\_salary)  
  
  
filtered\_hitters\_ds <- hitters\_ds %>%  
 filter(!is.na(Salary))  
filtered\_hitters\_ds$Salary <- log(filtered\_hitters\_ds$Salary)

# 10b Create a training set consisting of the first 200 observations, and a test set consisting of the remaining observations.

hitters\_ds\_training <- filtered\_hitters\_ds[1:200,]  
hitters\_ds\_testing <- filtered\_hitters\_ds[-(1:200),]

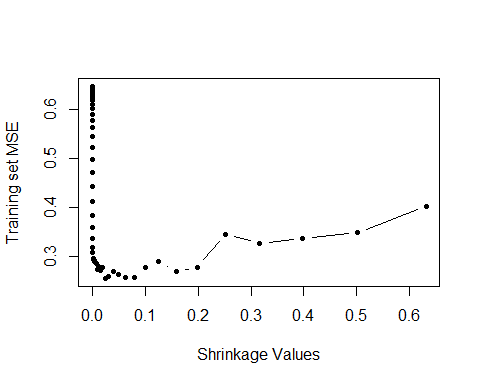
# 10c Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter λ. Produce a plot with different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

pows <- seq(-10, -0.2, by = 0.1)  
lambdas <- 10^pows  
training\_errors <- rep(NA, length(lambdas))  
  
  
for (i in 1:length(lambdas)) {  
 boosting\_model <- gbm(Salary ~ . , data = hitters\_ds\_training, distribution = "gaussian",   
 n.trees = 1000, shrinkage = lambdas[i])  
   
 training\_predictions <- predict(boosting\_model, hitters\_ds\_training, n.trees = 1000)  
 training\_errors[i] <- mean((training\_predictions - hitters\_ds\_training$Salary)^2)  
}  
  
plot(lambdas, training\_errors, xlab = "Shrinkage Values", ylab = "Training set MSE", type = "b", pch = 20)



# 10d Produce a plot with different shrinkage values on the x-axis and the corresponding test set MSE on the y-axis.

test\_errors <- rep(NA, length(lambdas))  
  
  
for (i in 1:length(lambdas)) {  
 boosting\_model <- gbm(Salary ~ . , data = hitters\_ds\_training, distribution = "gaussian",   
 n.trees = 1000, shrinkage = lambdas[i])  
   
 test\_predictions <- predict(boosting\_model, hitters\_ds\_testing, n.trees = 1000)  
 test\_errors[i] <- mean((test\_predictions - hitters\_ds\_testing$Salary)^2)  
}  
  
plot(lambdas, test\_errors, xlab = "Shrinkage Values", ylab = "Training set MSE", type = "b", pch = 20)



min(test\_errors)

## [1] 0.2557404

min\_test\_err <- min(test\_errors)  
min\_test\_err\_at <- lambdas[which.min(test\_errors)]

Minimum test error is obtained at λ = 0.0251189

# 10e 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\_model <- lm(Salary ~ . , data = hitters\_ds\_training)  
lm\_model\_predictions <- predict(lm\_model, hitters\_ds\_testing)  
lm\_model\_test\_mse <- mean((lm\_model\_predictions - hitters\_ds\_testing$Salary)^2)  
lm\_model\_test\_mse

## [1] 0.4917959

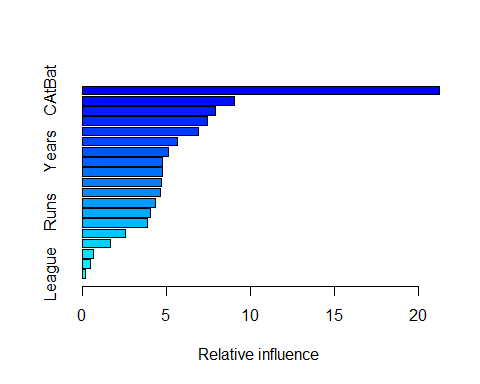
lasso\_model <- lm(Salary ~ . , data = hitters\_ds\_training)  
  
hitters\_ds\_training\_matrix <- model.matrix(Salary ~ . , data = hitters\_ds\_training)  
hitters\_ds\_testing\_matrix <- model.matrix(Salary ~ . , data = hitters\_ds\_testing)  
  
lasso\_model <- glmnet(hitters\_ds\_training\_matrix, hitters\_ds\_training$Salary, alpha = 1)  
lasso\_model\_predictions <- predict(lasso\_model, s=0.01, newx=hitters\_ds\_testing\_matrix)  
  
  
lasso\_model\_test\_mse <- mean((lasso\_model\_predictions - hitters\_ds\_testing$Salary)^2)  
lasso\_model\_test\_mse

## [1] 0.4700537

Both regression approaches lm and lasso have higher MSE compared to that of boosting.

# 10f Which variables appear to be the most important predictors in the boosted model?

boosting\_model <- gbm(Salary ~ . , data = hitters\_ds\_training, distribution = "gaussian",   
 n.trees = 1000, shrinkage = min\_test\_err)  
summary(boosting\_model)



## var rel.inf  
## CAtBat CAtBat 21.2484603  
## PutOuts PutOuts 9.0208616  
## CRuns CRuns 7.9273379  
## Walks Walks 7.4314427  
## CRBI CRBI 6.8794741  
## Assists Assists 5.6582958  
## Years Years 5.1497895  
## CWalks CWalks 4.7599174  
## AtBat AtBat 4.7433505  
## RBI RBI 4.7339007  
## CHmRun CHmRun 4.6621621  
## Hits Hits 4.3321089  
## Runs Runs 4.0771840  
## HmRun HmRun 3.8735363  
## Errors Errors 2.5592030  
## CHits CHits 1.6622569  
## NewLeague NewLeague 0.6368087  
## Division Division 0.4746364  
## League League 0.1692733

Variables that appear to be most important predictors are

1. CAtBat

2. PutOuts

3. CRuns

4. CRBI

# 10g Now apply bagging to the training set. What is the test set MSE for this approach?

random\_forest\_model <- randomForest(Salary ~ . , data = hitters\_ds\_training, ntree = 500, mtry = ncol(hitters\_ds\_training)-1)  
random\_forest\_predictions <- predict(random\_forest\_model, hitters\_ds\_testing)  
  
random\_forest\_test\_mse <- mean((random\_forest\_predictions - hitters\_ds\_testing$Salary)^2)  
random\_forest\_test\_mse

## [1] 0.2328366

Test MSE for bagging is 0.2328366 which is better than 0.2557404 which is best MSE from boosting