ISTM 660 Team 3, Homework 3

Team 3

3/24/2021

## ISTM660 Homework 3 - Team 3

Question 1 - Predicting with Ridge Regression

library(glmnet)

## Loading required package: Matrix

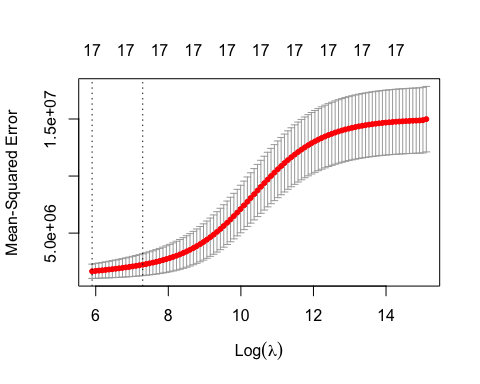
## Loaded glmnet 4.1-1

# Initializing the current file path & clearing variables from   
# the environment of execution.  
  
library(rstudioapi) # This is a external library of functions  
# Getting the path of your current open file  
current\_path = rstudioapi::getActiveDocumentContext()$path   
setwd(dirname(current\_path ))  
rm(list=ls())  
cat("\014")

#a. Loading the required input files for processing  
  
library(ISLR)  
View(College)  
  
sum(is.na(College$Apps))

## [1] 0

#b. Determine the best value of lambda through cross-validation   
  
x=model.matrix(Apps~.,College)[,-1]   
y=College$Apps  
  
#Best value of lambda through cross-validation  
  
set.seed(1)  
cv.out=cv.glmnet(x,y,alpha=0,nfolds=10,type.measure="mse")  
plot(cv.out)



bestlam = cv.out$lambda.min  
bestlam

## [1] 364.8993

# mean cross-validated error for best lambda  
cv.out$cvm[which.min(cv.out$cvm)]

## [1] 1658142

#c. Specify training (75%) and testing (25%) subsets and train the model using best lambda  
  
set.seed(1)  
train=sample(1:nrow(x), 0.75\*nrow(x))  
test=(-train)  
y.test=y[test]  
  
#training the model using best lambda obtained  
out=glmnet(x[train,],y[train],lambda=bestlam,alpha=0)  
ridge.pred=predict(out,s=bestlam ,newx=x[test,])   
coef(out)

## 18 x 1 sparse Matrix of class "dgCMatrix"  
## s0  
## (Intercept) -1.614571e+03  
## PrivateYes -4.008071e+02  
## Accept 1.087952e+00  
## Enroll 3.543005e-01  
## Top10perc 2.645250e+01  
## Top25perc 5.983126e-01  
## F.Undergrad 7.506199e-02  
## P.Undergrad 3.189042e-02  
## Outstate -2.675567e-02  
## Room.Board 2.206571e-01  
## Books 2.516169e-01  
## Personal -3.145862e-02  
## PhD -5.343684e+00  
## Terminal -3.467880e+00  
## S.F.Ratio 1.565884e+01  
## perc.alumni -4.953965e+00  
## Expend 6.059115e-02  
## Grad.Rate 8.913695e+00

#d. Test MSE associated with the best value of lambda  
  
mean((ridge.pred-y.test)^2)

## [1] 1206963

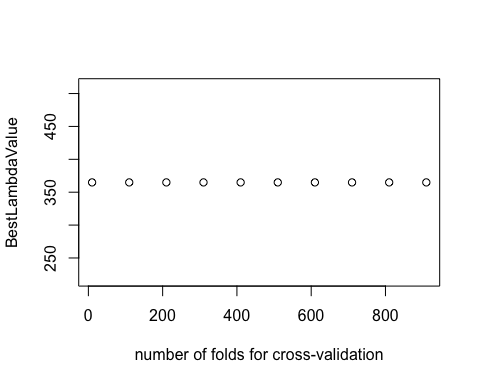
#e.Does your best value of lambda depend on the number of folds for cross-validation?  
  
#Using different values of number of folds for cross-validation to see its impact  
#on the value of lambda  
  
nfold\_value=seq(10,1000,by=100)  
nfold\_value

## [1] 10 110 210 310 410 510 610 710 810 910

d = NULL  
for (i in nfold\_value){  
 set.seed(1) # since we call a cross-validation function - need to use set.seed()  
 cv.out=cv.glmnet(x,y,alpha=0,nfolds=i,type.measure="mse")  
 bestlambda = list(cv.out$lambda.min)  
 d = rbind(d, data.frame(i, bestlambda))  
   
   
}

## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per  
## fold  
  
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per  
## fold  
  
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per  
## fold  
  
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per  
## fold  
  
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per  
## fold  
  
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per  
## fold  
  
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per  
## fold

cols <- c("number of folds for cross-validation", "BestLambdaValue")  
colnames(d) <- cols  
plot(d)



#From the above plot, we can conclude that the best value of lambda doesn't   
#depend on the number of folds for cross-validation  
  
#f.What are the estimated coefficients for your best model?  
  
out=glmnet(x,y,alpha=0)  
predict(out,type="coefficients",s=bestlam)[1:18,]

## (Intercept) PrivateYes Accept Enroll Top10perc   
## -1.468326e+03 -5.278781e+02 1.004588e+00 4.313442e-01 2.580619e+01   
## Top25perc F.Undergrad P.Undergrad Outstate Room.Board   
## 5.501092e-01 7.258520e-02 2.420595e-02 -2.407454e-02 1.987732e-01   
## Books Personal PhD Terminal S.F.Ratio   
## 1.285477e-01 -8.146131e-03 -4.028284e+00 -4.811071e+00 1.302180e+01   
## perc.alumni Expend Grad.Rate   
## -8.544783e+00 7.589013e-02 1.126699e+01

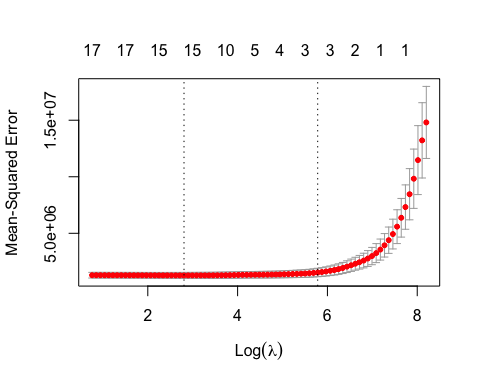
Question 2 - Predicting with The Lasso

#loading the required libraries  
  
  
library(glmnet)  
library(ISLR)  
  
  
# Initializing the current file path & clearing variables from   
# the environment of execution.  
  
library(rstudioapi) # This is a external library of functions  
# Getting the path of your current open file  
current\_path = rstudioapi::getActiveDocumentContext()$path   
setwd(dirname(current\_path ))  
rm(list=ls())  
cat("\014")

#a. Determine the best value of lambda through cross-validation using default   
# values for cross-validations and lambda used by glmnet() function   
  
  
x=model.matrix(Apps~.,College)[,-1]   
y=College$Apps  
  
  
#Best value of lambda through cross-validation  
  
cv.out=cv.glmnet(x,y,alpha=1, type.measure = 'mse')  
plot(cv.out)  
bestlambda = cv.out$lambda.min  
bestlambda

## [1] 16.54774

#b Plot the results of your cross validation.  
  
plot(cv.out)



cv.out$cvm[which.min(cv.out$cvm)]

## [1] 1277016

out=glmnet(x,y,alpha=1,lambda=bestlambda)  
lasso.coef=predict(out,type="coefficients",s=bestlambda)[1:18,]   
lasso.coef

## (Intercept) PrivateYes Accept Enroll Top10perc   
## -5.662259e+02 -4.472324e+02 1.484660e+00 -2.956967e-01 3.753354e+01   
## Top25perc F.Undergrad P.Undergrad Outstate Room.Board   
## -5.120346e+00 0.000000e+00 2.975703e-02 -6.514216e-02 1.316004e-01   
## Books Personal PhD Terminal S.F.Ratio   
## 0.000000e+00 6.703347e-03 -6.369926e+00 -3.158529e+00 7.232447e+00   
## perc.alumni Expend Grad.Rate   
## -8.130372e-01 7.159205e-02 5.951663e+00

#From the above plot we can see that the best value of lambda comes around 16.54774  
# and for that given value there are 15 non-zero coefficients in the model.  
  
#c Test MSE associated with the best value of lambda  
  
#Splitting the data into 75% training and 25% testing set  
set.seed(1)  
train=sample(1:nrow(x), 0.75\*nrow(x))  
test=(-train)  
y.test=y[test]  
  
#Training the model on using the best value of lambda determined  
  
lasso.mod=glmnet(x[train ,],y[train],alpha=1,lambda=bestlambda)  
  
lasso.pred=predict(lasso.mod,s=bestlambda ,newx=x[test,])  
  
#Test MSE  
  
mean((lasso.pred-y.test)^2)

## [1] 1331051

#d. Which model is better: ridge or lasso.  
  
out=glmnet(x,y,alpha=1,lambda=bestlambda)  
lasso.coef=predict(out,type="coefficients",s=bestlambda)[1:18,]   
lasso.coef

## (Intercept) PrivateYes Accept Enroll Top10perc   
## -5.662259e+02 -4.472324e+02 1.484660e+00 -2.956967e-01 3.753354e+01   
## Top25perc F.Undergrad P.Undergrad Outstate Room.Board   
## -5.120346e+00 0.000000e+00 2.975703e-02 -6.514216e-02 1.316004e-01   
## Books Personal PhD Terminal S.F.Ratio   
## 0.000000e+00 6.703347e-03 -6.369926e+00 -3.158529e+00 7.232447e+00   
## perc.alumni Expend Grad.Rate   
## -8.130372e-01 7.159205e-02 5.951663e+00

#Test MSE associated with Ridge and Lasso regression is very similar.   
#However, lasso regression has substantial advantage over ridge as it reduces 2   
#coefficients to zero which helps to reduce noise and for small bias we can get   
#rid of substantial amount of variance.   
  
  
  
#e.Combine ridge regression with lasso (as in elastic net)   
# to arrive at a better predictive model  
  
  
library("caret")

## Loading required package: lattice

## Loading required package: ggplot2

#Building the model for elastic net regression  
control <- trainControl(method = "cv",number = 10)  
  
  
#Training the elastic net model  
elastic\_model = train(  
 Apps ~ ., data = College,  
 method = "glmnet",  
 tuneLength = 10,  
 trControl = control  
)  
  
elastic\_model

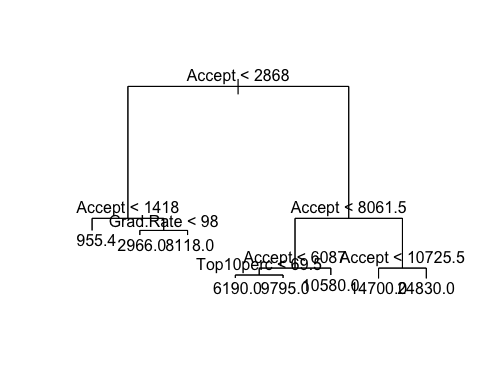
## glmnet   
##   
## 777 samples  
## 17 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 699, 698, 701, 699, 701, 700, ...   
## Resampling results across tuning parameters:  
##   
## alpha lambda RMSE Rsquared MAE   
## 0.1 1.685929 1119.077 0.9169469 632.5130  
## 0.1 3.894715 1119.077 0.9169469 632.5130  
## 0.1 8.997297 1117.269 0.9172793 630.8957  
## 0.1 20.784925 1116.191 0.9180369 627.0775  
## 0.1 48.015873 1117.659 0.9189080 622.2412  
## 0.1 110.922897 1127.184 0.9188765 619.9630  
## 0.1 256.246284 1156.270 0.9159903 627.4501  
## 0.1 591.962166 1218.684 0.9091314 656.3250  
## 0.1 1367.509416 1345.961 0.8983813 729.5350  
## 0.1 3159.124195 1651.293 0.8818508 973.1038  
## 0.2 1.685929 1119.160 0.9169227 632.2901  
## 0.2 3.894715 1119.160 0.9169227 632.2901  
## 0.2 8.997297 1116.366 0.9174029 629.7515  
## 0.2 20.784925 1114.911 0.9181999 624.9714  
## 0.2 48.015873 1116.608 0.9189814 618.7464  
## 0.2 110.922897 1128.143 0.9183212 616.4961  
## 0.2 256.246284 1162.501 0.9143935 625.2680  
## 0.2 591.962166 1227.060 0.9086528 641.8816  
## 0.2 1367.509416 1404.242 0.8955625 772.4126  
## 0.2 3159.124195 1847.068 0.8686267 1137.8261  
## 0.3 1.685929 1118.560 0.9169497 631.9769  
## 0.3 3.894715 1118.560 0.9169497 631.9769  
## 0.3 8.997297 1115.477 0.9175169 628.6855  
## 0.3 20.784925 1113.824 0.9183163 623.2479  
## 0.3 48.015873 1116.273 0.9189073 616.1784  
## 0.3 110.922897 1129.451 0.9177109 613.6392  
## 0.3 256.246284 1164.114 0.9136437 616.7134  
## 0.3 591.962166 1242.760 0.9071627 643.9616  
## 0.3 1367.509416 1462.944 0.8927327 824.9777  
## 0.3 3159.124195 2035.139 0.8516646 1281.9039  
## 0.4 1.685929 1118.068 0.9169571 631.7813  
## 0.4 3.894715 1117.837 0.9169944 631.5805  
## 0.4 8.997297 1114.659 0.9176151 627.6321  
## 0.4 20.784925 1112.795 0.9184106 621.6098  
## 0.4 48.015873 1116.199 0.9187839 614.5476  
## 0.4 110.922897 1131.599 0.9168682 612.5787  
## 0.4 256.246284 1165.258 0.9129940 608.2849  
## 0.4 591.962166 1252.025 0.9068133 650.1520  
## 0.4 1367.509416 1533.889 0.8870784 882.3776  
## 0.4 3159.124195 2184.869 0.8580711 1398.0861  
## 0.5 1.685929 1117.384 0.9169767 631.6120  
## 0.5 3.894715 1116.711 0.9170868 630.9405  
## 0.5 8.997297 1113.940 0.9177024 626.5931  
## 0.5 20.784925 1111.681 0.9185044 619.5678  
## 0.5 48.015873 1115.061 0.9188130 613.0128  
## 0.5 110.922897 1135.257 0.9157321 612.9138  
## 0.5 256.246284 1165.936 0.9124181 604.1449  
## 0.5 591.962166 1263.126 0.9061193 657.8681  
## 0.5 1367.509416 1606.114 0.8809326 941.9735  
## 0.5 3159.124195 2354.137 0.8665253 1528.7850  
## 0.6 1.685929 1117.001 0.9169945 631.4099  
## 0.6 3.894715 1116.036 0.9171557 630.3785  
## 0.6 8.997297 1113.257 0.9177753 625.6497  
## 0.6 20.784925 1110.853 0.9185465 617.7411  
## 0.6 48.015873 1113.542 0.9188586 611.5157  
## 0.6 110.922897 1137.849 0.9148135 613.4716  
## 0.6 256.246284 1167.745 0.9116386 601.9567  
## 0.6 591.962166 1279.732 0.9043110 670.3492  
## 0.6 1367.509416 1676.094 0.8753084 999.4158  
## 0.6 3159.124195 2541.792 0.8757106 1676.6707  
## 0.7 1.685929 1116.716 0.9170013 631.3517  
## 0.7 3.894715 1115.514 0.9172112 629.9089  
## 0.7 8.997297 1112.670 0.9178370 624.7743  
## 0.7 20.784925 1110.280 0.9185478 616.2800  
## 0.7 48.015873 1112.816 0.9187695 609.8768  
## 0.7 110.922897 1136.554 0.9144645 610.9025  
## 0.7 256.246284 1171.573 0.9103973 602.6164  
## 0.7 591.962166 1299.562 0.9014803 685.7874  
## 0.7 1367.509416 1721.376 0.8781048 1042.2984  
## 0.7 3159.124195 2733.998 0.8800061 1823.6521  
## 0.8 1.685929 1116.468 0.9170008 631.2355  
## 0.8 3.894715 1114.923 0.9172626 629.4249  
## 0.8 8.997297 1112.195 0.9178841 623.9420  
## 0.8 20.784925 1109.942 0.9185161 615.0236  
## 0.8 48.015873 1112.659 0.9185458 608.3742  
## 0.8 110.922897 1134.484 0.9142664 607.3281  
## 0.8 256.246284 1172.671 0.9096141 601.5567  
## 0.8 591.962166 1312.660 0.8992297 699.7383  
## 0.8 1367.509416 1766.665 0.8795981 1082.1808  
## 0.8 3159.124195 2905.599 0.8815409 1948.7024  
## 0.9 1.685929 1116.271 0.9169915 631.1965  
## 0.9 3.894715 1114.486 0.9173112 628.8855  
## 0.9 8.997297 1111.464 0.9179518 622.9943  
## 0.9 20.784925 1109.591 0.9184829 613.9977  
## 0.9 48.015873 1113.061 0.9181697 607.8277  
## 0.9 110.922897 1132.309 0.9140360 604.0607  
## 0.9 256.246284 1170.060 0.9093222 598.3283  
## 0.9 591.962166 1320.801 0.8965483 713.1886  
## 0.9 1367.509416 1790.347 0.8813156 1110.7245  
## 0.9 3159.124195 3104.067 0.8815409 2087.3730  
## 1.0 1.685929 1115.933 0.9170171 631.0316  
## 1.0 3.894715 1114.029 0.9173622 628.3668  
## 1.0 8.997297 1110.608 0.9180252 622.1379  
## 1.0 20.784925 1108.935 0.9184751 613.0070  
## 1.0 48.015873 1113.539 0.9177154 607.7018  
## 1.0 110.922897 1129.794 0.9137680 601.3808  
## 1.0 256.246284 1166.566 0.9089108 595.6879  
## 1.0 591.962166 1333.499 0.8927230 729.4348  
## 1.0 1367.509416 1827.205 0.8815409 1142.1799  
## 1.0 3159.124195 3338.707 0.8815409 2249.4575  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were alpha = 1 and lambda = 20.78492.

#Based on the output, we can see that for alpha=0.9 and lambda=20.78492 we can get  
#the best model

Question 3 - Predicting with Decision Trees

# Q3 - a.  
# Continue using the College data set.  
# Create a regression tree using the training data with Apps as the response  
# variable and all the predictors.   
# Note that you will probably need to recreate training and testing subsets as   
# the tree routines work with dataframes.   
# Plot your tree.  
  
# Get active path, clear environment variables, clear console, load data & packages  
library(rstudioapi)  
current\_path = rstudioapi::getActiveDocumentContext()$path   
setwd(dirname(current\_path ))  
rm(list=ls())  
cat("\014")

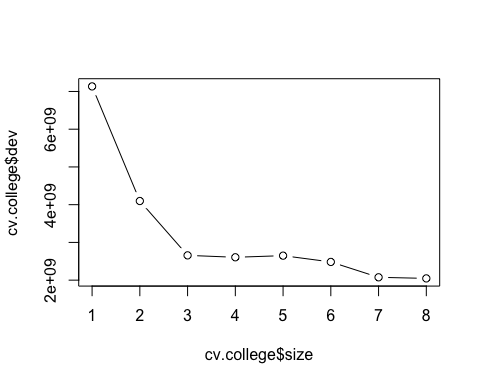
library(ISLR)  
library(tree)  
  
# Set seed & sample the dataset of College and divide it into training & testing  
set.seed(1)  
train = sample(1:nrow(College), nrow(College)/2)  
college.test=College[-train,"Apps"]  
  
# Create decision tree by training College dataset and store its predictions in yhat  
tree.college=tree(Apps~.,College,subset=train)  
yhat=predict(tree.college,newdata=data.frame(College[-train,]))  
  
# Plotting the initial tree  
par(mfrow=c(1,1))  
plot(tree.college)  
text(tree.college)



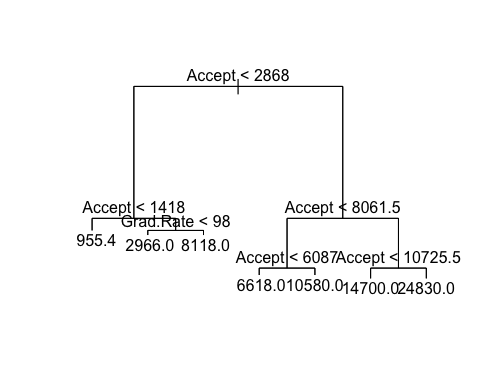
# Compute the test MSE  
mean((yhat-college.test)^2)

## [1] 2586142

# Q3 - b.  
# Prune your tree using cost complexity pruning.   
# What size is the best tree?  
  
# Set seed, use cross-validation and plot the deviance graph  
set.seed(1)  
cv.college=cv.tree(tree.college)  
plot(cv.college$size,cv.college$dev,type='b')



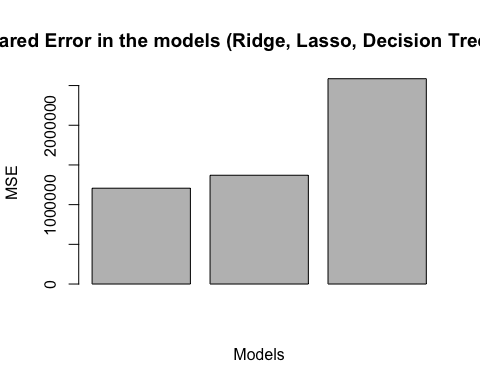
# This shows that the best size is 7 based on the deviance levels  
  
  
# Q3 - c.  
# Plot the tree corresponding to the size you identified in the prior question.  
  
# Plotting the new tree based on the best deviance level, i.e. best = 7  
prune.college=prune.tree(tree.college,best=7)  
plot(prune.college)  
text(prune.college)



# Q3 - d.  
# Using the test data, what is the test MSE and/or RMSE? Is this a good predictive model?  
  
# Compute the test MSE  
mse = mean((yhat-college.test)^2)  
  
# Compute the test RMSE  
rmse = sqrt(mse)  
  
# Print both MSE & RMSE respectively  
paste(mse, rmse, sep=" | ")

## [1] "2586141.99151328 | 1608.14862233354"

# Is this a good predictive model?  
# Decision trees are relatively easy to understand and implement, however, they do not have  
# the best prediction as at times they can overfit. This is taken care of by the way of cost  
# cost complexity pruning, i.e. by limiting the depth of the tree.   
# After pruning, the resultant decision tree can be considered a good predictive model.  
# The comparison of the three models, however, is shown in the plot in the below question.  
  
  
# Q3 - e.  
# How does the accuracy of your best tree compare to ridge regression and the lasso?  
  
# Plotting a graph with Decision Tree's MSE with MSE from Ridge & Lasso, to judge the performance of models  
ridge.mse=1206963  
lasso.mse=1370181  
dt.mse=mse  
par(mfrow=c(1,1))  
plot.df <- data.frame(  
 name = c("1. Ridge", "2. Lasso", "3. Decision Tree"),  
 mse = c(ridge.mse, lasso.mse, dt.mse)  
)  
barplot(plot.df$mse, main="Mean-Squared Error in the models (Ridge, Lasso, Decision Tree respectively)", xlab="Models", ylab="MSE")



# As seen from the plot of MSE for the 3 models, we can see that Decision Tree is relatively  
# a worse model than Ridge and Lasso for this problem as it has a higher MSE than the two models.  
# However, the interpretability of the model is much higher.

Question 4 - Classifying with the Lasso

# Q4 - a.  
# Use the stocks.rdata file that was provided to you in the tree-based methods resources   
# and that you should have worked with on the Decision Tree Lab.   
# Create a binary variable called "target" that is equal to 1 if the value nreturn1 is greater than 0   
# and otherwise, set target to 0 if it is less than or equal to 0. Then get rid of that nreturn1 column.  
  
# Get active path, clear environment variables, clear console, load data & packages  
library(rstudioapi)  
current\_path = rstudioapi::getActiveDocumentContext()$path   
setwd(dirname(current\_path ))  
rm(list=ls())  
cat("\014")

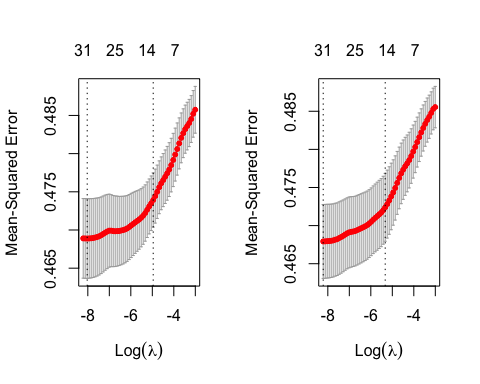
load("stocks.rdata")  
  
# Create target variable using nreturn1 and then drop nreturn1 variable  
stocks$target = ifelse(stocks$nreturn1>0, 1, 0)  
stocks = subset(stocks, select = -c(ncol(stocks)-1))  
  
  
# Q4 - b.  
# Use the Lasso and cross validation to arrive at a best model.   
# Be sure to set.seed(1) prior to the cross-validation.   
# Note that this is a classification problem and not regression.  
  
# Create a matrix with predictors, i.e. x, and the response variable, i.e. y  
x=model.matrix(target~.,stocks)[,-1]  
y=stocks$target  
  
# Set the default plotting view of 1 plot  
par(mfrow=c(1,1))  
  
# Set seed and create Lasso with family=binomial for classification and alpha=1 for Lasso. Plot it  
set.seed(1)  
cv.out=cv.glmnet(x,y,alpha=1,family="binomial",type.measure="mse")  
plot(cv.out)  
  
  
# Q4 - c.  
# Plot the cross-validation results.   
# What does this visualization tell you about the selection of your best model   
# and the number of variables that are in it, relative to other sized models?  
  
# Store the best lambda value  
bestlam=cv.out$lambda.min  
  
# Print cvm & coefficients  
cv.out$cvm[which.min(cv.out$cvm)]

## [1] 0.4688778

# Print the coefficients to get the variables in usage:  
# - variables with dot are NOT used by the model  
# - variables with numeric values are used by the model  
# Therefore, number of variables in use in this model = 12  
coef(cv.out)

## 44 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -6.148830e+00  
## open .   
## high .   
## low .   
## close .   
## volume .   
## avgofday .   
## ma5 .   
## ma10 .   
## ma20 .   
## ma50 .   
## ma100 .   
## ma200 9.474255e-04  
## ma5slope .   
## ma10slope .   
## ma20slope .   
## ma50slope -2.837001e+01  
## ma100slope .   
## ma200slope -6.984602e+01  
## volma5 .   
## preturn1 3.213838e+00  
## preturn2 4.721347e-01  
## preturn3 2.670282e+00  
## preturn4 -3.561933e+00  
## preturn5 .   
## croc1 .   
## croc5 .   
## obv .   
## obvma5 -3.689263e-10  
## macd .   
## ma5ratio .   
## ma10ratio .   
## ma20ratio .   
## ma50ratio .   
## ma100ratio .   
## ma200ratio .   
## ma5toma10 6.851644e+00  
## ma5toma20 .   
## ma20toma50 .   
## ma50toma100 .   
## ma50toma200 -1.247190e-01  
## traderangep -3.080685e+00  
## pma5 .   
## vvmma5 -2.227925e-01

# The visualization shows two dashed lines: the left dashed line is for min(lambda) and  
# the right dashed line is for 1 SD away.  
# Furthermore, the left dashed line is the value where the model has the lowest cross-validation MSE.  
# The numbers on top are the decreasing number of features in the model, and going from left to right,  
# the MSE increases with increase in lambda with a decrease in features included.  
# This is because the penalty keeps increasing once you go left to right.  
# Lastly, this is why we use shrinkage methods reduce the impact of unimportant features leading to noise reduction.  
  
# Q4 - d.  
# Rerun the cross-validation, but use set.seed(25). What insights do these new results offer?  
  
# To compare the two plots, set the new par value  
par(mfrow=c(1,2))  
  
# Repeat the steps of creating another classifying Lasso, plot it  
set.seed(25)  
cv.out.new=cv.glmnet(x,y,alpha=1,family="binomial",type.measure="mse")  
plot(cv.out)  
plot(cv.out.new)



# Store the best lambda value  
bestlam.new=cv.out.new$lambda.min  
cv.out.new$cvm[which.min(cv.out.new$cvm)]

## [1] 0.4679217

coef(cv.out.new)

## 44 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -7.535386e+00  
## open .   
## high .   
## low .   
## close .   
## volume .   
## avgofday .   
## ma5 .   
## ma10 .   
## ma20 .   
## ma50 .   
## ma100 .   
## ma200 1.129451e-03  
## ma5slope .   
## ma10slope .   
## ma20slope .   
## ma50slope -3.177602e+01  
## ma100slope .   
## ma200slope -7.042084e+01  
## volma5 2.738704e-09  
## preturn1 5.509936e+00  
## preturn2 1.531614e+00  
## preturn3 4.702062e+00  
## preturn4 -5.838099e+00  
## preturn5 .   
## croc1 -2.944836e+00  
## croc5 .   
## obv .   
## obvma5 -4.713514e-10  
## macd .   
## ma5ratio .   
## ma10ratio .   
## ma20ratio .   
## ma50ratio .   
## ma100ratio .   
## ma200ratio .   
## ma5toma10 8.231816e+00  
## ma5toma20 .   
## ma20toma50 .   
## ma50toma100 .   
## ma50toma200 -1.072444e-01  
## traderangep -3.184800e+00  
## pma5 .   
## vvmma5 -2.533852e-01

coef(cv.out)

## 44 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -6.148830e+00  
## open .   
## high .   
## low .   
## close .   
## volume .   
## avgofday .   
## ma5 .   
## ma10 .   
## ma20 .   
## ma50 .   
## ma100 .   
## ma200 9.474255e-04  
## ma5slope .   
## ma10slope .   
## ma20slope .   
## ma50slope -2.837001e+01  
## ma100slope .   
## ma200slope -6.984602e+01  
## volma5 .   
## preturn1 3.213838e+00  
## preturn2 4.721347e-01  
## preturn3 2.670282e+00  
## preturn4 -3.561933e+00  
## preturn5 .   
## croc1 .   
## croc5 .   
## obv .   
## obvma5 -3.689263e-10  
## macd .   
## ma5ratio .   
## ma10ratio .   
## ma20ratio .   
## ma50ratio .   
## ma100ratio .   
## ma200ratio .   
## ma5toma10 6.851644e+00  
## ma5toma20 .   
## ma20toma50 .   
## ma50toma100 .   
## ma50toma200 -1.247190e-01  
## traderangep -3.080685e+00  
## pma5 .   
## vvmma5 -2.227925e-01

# Insights - Even with a change in sample of data, there is not much difference between the two lambda values.  
# However, when we compare the two coefficients, we see that there are a few variables that are not included in  
# the new sample. This could probably mean that in the dataset, there is not one proper solution to solve the problem.  
# That is why, there seems to be an inconsistency in results, as the method does not provide one common solution with  
# a high degree of confidence.  
  
  
# Q4 - e.  
# Use 80% of the data to train a new lasso model using the value of lambda you found best   
# using your first cross-validation results from part b.  
  
# Creating training and testing datasets  
set.seed(1)  
train=sample(1:nrow(x), 0.80\*nrow(x))  
test=(-train)  
y.test=y[test]  
  
# Constructing a new Lasso model using the lambda from part b, i.e. bestlam  
lasso.mod=glmnet(x[train ,],y[train],alpha=1,lambda=bestlam)  
  
# Predict using test dataset  
lasso.pred=predict(lasso.mod,s=bestlam,newx=x[test,])  
  
# As predictions are stored as probabilities, classification will be done with the threshold as 0.5  
lasso.predict <- rep(0,length(y.test))  
lasso.predict[lasso.pred>.5] <- 1  
  
# Confusion matrix and accuracy  
table(pred=lasso.predict,true=y.test)

## true  
## pred 0 1  
## 0 59 45  
## 1 188 287

mean(lasso.predict==y.test)

## [1] 0.597582