Assignment 1

#QA1 #The main purpose of regularization is to simplify models in order to improve their performance. Keeping it as simple as possible decreases the chance of a model being to complex and struggling to generalize.

#QA2 #A loss function is a function that is trying to minimize an objective in predictive modeling. An example of a loss the function for regression models is root mean square error, RMSE, or the mean absolute error MAE. A example of a common loss function for classification models is log loss, ROC and AUC.

#QA3 #You can’t fully trust this model because of the chance there may be overfitting. A smaller more specific data setDue to a large number of hyperparameters is the perfect candidate for overfitting. The fact that the training error is low also makes me consider the idea of overfitting because typically a lower training model means themodel is probably more complex. There is a greater likelihood that the model could lose its ability to generalize.

#Q4 #The lambda parameter controls the amount of regularization of a model. The lambda parameter balances the minimization and error of a model to find an optimal solution. In a Lasso regression model the lambda parameter minimize the absolute sum values of coefficients while in Ridge Regression models minimizes the sum square of the error terms.

library(ISLR)   
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-4

library(caret)

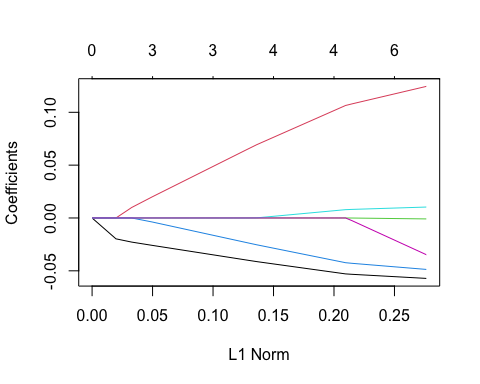
## Loading required package: ggplot2

## Loading required package: lattice

#Problem QB1  
  
Carseats\_Filtered <- Carseats %>% select("Sales", "Price","Advertising","Population","Age","Income","Education")   
preProcess(Carseats\_Filtered)

## Created from 400 samples and 7 variables  
##   
## Pre-processing:  
## - centered (7)  
## - ignored (0)  
## - scaled (7)

y <- Carseats$Sales  
x <- as.matrix(Carseats[, c('Price', 'Advertising', 'Population', 'Age', 'Income', 'Education')])  
#Fit Models   
  
fit = glmnet(x,y)  
plot(fit)



print(fit)

##   
## Call: glmnet(x = x, y = y)   
##   
## Df %Dev Lambda  
## 1 0 0.00 1.25500  
## 2 1 3.36 1.14400  
## 3 1 6.15 1.04200  
## 4 1 8.47 0.94940  
## 5 1 10.39 0.86500  
## 6 1 11.99 0.78820  
## 7 2 14.62 0.71820  
## 8 3 18.08 0.65440  
## 9 3 21.12 0.59620  
## 10 3 23.64 0.54330  
## 11 3 25.73 0.49500  
## 12 3 27.46 0.45100  
## 13 3 28.91 0.41100  
## 14 3 30.10 0.37450  
## 15 4 31.12 0.34120  
## 16 4 32.13 0.31090  
## 17 4 32.97 0.28330  
## 18 4 33.67 0.25810  
## 19 4 34.25 0.23520  
## 20 4 34.73 0.21430  
## 21 4 35.13 0.19520  
## 22 4 35.46 0.17790  
## 23 4 35.74 0.16210  
## 24 4 35.97 0.14770  
## 25 4 36.16 0.13460  
## 26 4 36.31 0.12260  
## 27 4 36.45 0.11170  
## 28 4 36.55 0.10180  
## 29 4 36.64 0.09276  
## 30 6 36.75 0.08451  
## 31 6 36.86 0.07701  
## 32 6 36.95 0.07017  
## 33 6 37.02 0.06393  
## 34 6 37.09 0.05825  
## 35 6 37.14 0.05308  
## 36 6 37.18 0.04836  
## 37 6 37.21 0.04407  
## 38 6 37.24 0.04015  
## 39 6 37.27 0.03658  
## 40 6 37.29 0.03333  
## 41 6 37.30 0.03037  
## 42 6 37.32 0.02767  
## 43 6 37.33 0.02522  
## 44 6 37.34 0.02298  
## 45 6 37.35 0.02094  
## 46 6 37.35 0.01908  
## 47 6 37.36 0.01738  
## 48 6 37.36 0.01584  
## 49 6 37.37 0.01443  
## 50 6 37.37 0.01315  
## 51 6 37.37 0.01198  
## 52 6 37.38 0.01092  
## 53 6 37.38 0.00995  
## 54 6 37.38 0.00906  
## 55 6 37.38 0.00826  
## 56 6 37.38 0.00752  
## 57 6 37.38 0.00686  
## 58 6 37.38 0.00625  
## 59 6 37.38 0.00569  
## 60 6 37.38 0.00519  
## 61 6 37.38 0.00472  
## 62 6 37.38 0.00430

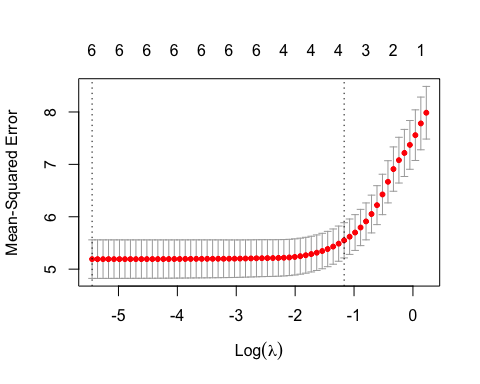
coef(fit, s=1)

## 7 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 8.74510987  
## Price -0.01078445  
## Advertising .   
## Population .   
## Age .   
## Income .   
## Education .

#Predictions  
nx = matrix(rnorm(12\*6),12,6)  
predict(fit,newx=nx,s=c(.1,.05))

## s1 s2  
## [1,] 14.44346 15.08152  
## [2,] 14.51519 15.13949  
## [3,] 14.45676 15.06981  
## [4,] 14.50614 15.15811  
## [5,] 14.71899 15.37290  
## [6,] 14.68192 15.31454  
## [7,] 14.52666 15.15654  
## [8,] 14.65568 15.31831  
## [9,] 14.49955 15.19107  
## [10,] 14.50983 15.16609  
## [11,] 14.52529 15.17359  
## [12,] 14.71063 15.35612

#Cross Validation  
cvfit = cv.glmnet(x,y)  
plot(cvfit)



cvfit$lambda.min

## [1] 0.004305309

cvfit$lambda.1se

## [1] 0.3108781

coef(cvfit, s = "lambda.min")

## 7 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 15.8945811275  
## Price -0.0571800982  
## Advertising 0.1245132007  
## Population -0.0008862575  
## Age -0.0486750291  
## Income 0.0103379764  
## Education -0.0347353012

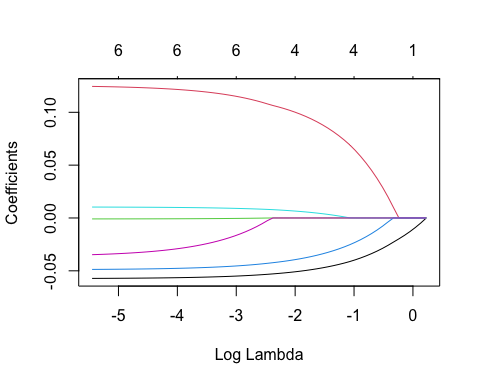
predict(cvfit, newx = x[1:6,], s = "lambda.min")

## lambda.min  
## 1 8.277828  
## 2 9.895410  
## 3 9.400081  
## 4 8.303545  
## 5 7.008161  
## 6 9.768045

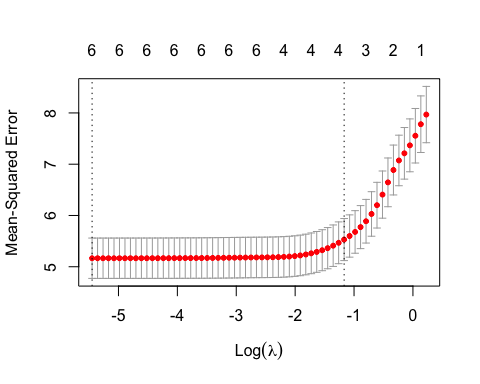
cvfit = cv.glmnet(x, y, type.measure = "mae", nfolds = 5)  
coef(cvfit, s = "lambda.min")

## 7 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 15.8945811275  
## Price -0.0571800982  
## Advertising 0.1245132007  
## Population -0.0008862575  
## Age -0.0486750291  
## Income 0.0103379764  
## Education -0.0347353012

#Build Lasso   
fit.lasso <- glmnet(x, y, alpha = 1)  
plot(fit.lasso, xvar = "lambda")



plot(cv.glmnet(x, y, alpha=1))



#Question 2  
   
coef(cvfit, s = "lambda.min")

## 7 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 15.8945811275  
## Price -0.0571800982  
## Advertising 0.1245132007  
## Population -0.0008862575  
## Age -0.0486750291  
## Income 0.0103379764  
## Education -0.0347353012

predict(cvfit, newx = x[1:6,], s ="lambda.min")

## lambda.min  
## 1 8.277828  
## 2 9.895410  
## 3 9.400081  
## 4 8.303545  
## 5 7.008161  
## 6 9.768045

# The coefficient for the price in the best model is 8.2777.

#Question 3  
coef(fit, s=1)

## 7 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 8.74510987  
## Price -0.01078445  
## Advertising .   
## Population .   
## Age .   
## Income .   
## Education .

coef(fit,s=0.01)

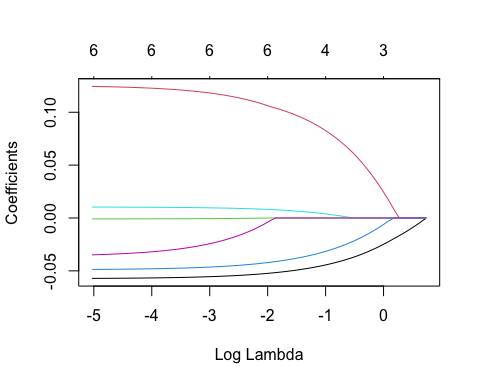
## 7 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 15.8120357462  
## Price -0.0569053296  
## Advertising 0.1233400736  
## Population -0.0008269647  
## Age -0.0482649088  
## Income 0.0101796053  
## Education -0.0324465331

coef(fit, s=0.1)

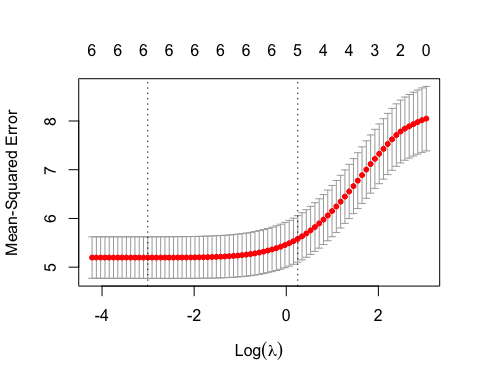
## 7 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 14.590306998  
## Price -0.052573918  
## Advertising 0.105366130  
## Population .   
## Age -0.041822865  
## Income 0.007643889  
## Education .

#All 6 attributes stay in the model if the lambda is set to 0.01. Where as only 5 stayin the model if the lambda is set to 0.10. I would expect less variables to stay in the model as you increase the lambda.

#Question 4  
fit.elnet <- glmnet(x,y, alpha = 0.6)  
plot(fit.elnet, xvar = "lambda")



plot(cv.glmnet(x,y,alpha=.06))



print(fit.elnet)

##   
## Call: glmnet(x = x, y = y, alpha = 0.6)   
##   
## Df %Dev Lambda  
## 1 0 0.00 2.09200  
## 2 1 2.67 1.90600  
## 3 1 5.03 1.73700  
## 4 1 7.09 1.58200  
## 5 1 8.90 1.44200  
## 6 1 10.47 1.31400  
## 7 2 12.89 1.19700  
## 8 3 16.00 1.09100  
## 9 3 18.95 0.99370  
## 10 3 21.49 0.90540  
## 11 3 23.67 0.82500  
## 12 3 25.55 0.75170  
## 13 3 27.15 0.68490  
## 14 3 28.52 0.62410  
## 15 4 29.75 0.56860  
## 16 4 30.91 0.51810  
## 17 4 31.89 0.47210  
## 18 4 32.72 0.43020  
## 19 4 33.43 0.39190  
## 20 4 34.02 0.35710  
## 21 4 34.52 0.32540  
## 22 4 34.93 0.29650  
## 23 4 35.29 0.27020  
## 24 4 35.58 0.24620  
## 25 4 35.83 0.22430  
## 26 4 36.04 0.20440  
## 27 4 36.21 0.18620  
## 28 4 36.36 0.16970  
## 29 4 36.48 0.15460  
## 30 6 36.60 0.14090  
## 31 6 36.73 0.12830  
## 32 6 36.84 0.11690  
## 33 6 36.93 0.10660  
## 34 6 37.01 0.09709  
## 35 6 37.07 0.08846  
## 36 6 37.12 0.08060  
## 37 6 37.17 0.07344  
## 38 6 37.20 0.06692  
## 39 6 37.23 0.06097  
## 40 6 37.26 0.05556  
## 41 6 37.28 0.05062  
## 42 6 37.30 0.04612  
## 43 6 37.31 0.04203  
## 44 6 37.33 0.03829  
## 45 6 37.34 0.03489  
## 46 6 37.34 0.03179  
## 47 6 37.35 0.02897  
## 48 6 37.36 0.02639  
## 49 6 37.36 0.02405  
## 50 6 37.37 0.02191  
## 51 6 37.37 0.01997  
## 52 6 37.37 0.01819  
## 53 6 37.37 0.01658  
## 54 6 37.38 0.01510  
## 55 6 37.38 0.01376  
## 56 6 37.38 0.01254  
## 57 6 37.38 0.01143  
## 58 6 37.38 0.01041  
## 59 6 37.38 0.00949  
## 60 6 37.38 0.00864  
## 61 6 37.38 0.00788  
## 62 6 37.38 0.00718  
## 63 6 37.38 0.00654

# The Optimal lambda is 0.00654 in this model.

for (a in seq(0,1, by=0.1)) {  
 cvfit=cv.glmnet(x, y, alpha=a)  
 print(paste0("alpha is ", a, " and best MSE is " ,min(cvfit$cvm)))  
}

## [1] "alpha is 0 and best MSE is 5.11978014178814"  
## [1] "alpha is 0.1 and best MSE is 5.15774548111804"  
## [1] "alpha is 0.2 and best MSE is 5.12154780020159"  
## [1] "alpha is 0.3 and best MSE is 5.14786382234857"  
## [1] "alpha is 0.4 and best MSE is 5.1740602348602"  
## [1] "alpha is 0.5 and best MSE is 5.13891500887064"  
## [1] "alpha is 0.6 and best MSE is 5.10421837020004"  
## [1] "alpha is 0.7 and best MSE is 5.21579866847452"  
## [1] "alpha is 0.8 and best MSE is 5.10585860029709"  
## [1] "alpha is 0.9 and best MSE is 5.12769305322967"  
## [1] "alpha is 1 and best MSE is 5.15585937366293"