Homework 3

Bethany V Smith

February 8, 2018

# 2) Work on Problem 9 Part (a)-(d) and (g) on page 263. Note, you don’t need to include PCR and PLS in Part (g). However, if you want to, then read Section 6.7, and work on Part (e) and (f), too. College data is in the ISLR library. You can use the following code to get it.

## (a) Split the data set into a training set and a test set.

data(College)  
set.seed(1)  
trainid <- sample(1:nrow(College), nrow(College)/2)  
train <- College[trainid, ]  
test <- College[-trainid, ]  
str(College)

## 'data.frame': 777 obs. of 18 variables:  
## $ Private : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...  
## $ Apps : num 1660 2186 1428 417 193 ...  
## $ Accept : num 1232 1924 1097 349 146 ...  
## $ Enroll : num 721 512 336 137 55 158 103 489 227 172 ...  
## $ Top10perc : num 23 16 22 60 16 38 17 37 30 21 ...  
## $ Top25perc : num 52 29 50 89 44 62 45 68 63 44 ...  
## $ F.Undergrad: num 2885 2683 1036 510 249 ...  
## $ P.Undergrad: num 537 1227 99 63 869 ...  
## $ Outstate : num 7440 12280 11250 12960 7560 ...  
## $ Room.Board : num 3300 6450 3750 5450 4120 ...  
## $ Books : num 450 750 400 450 800 500 500 450 300 660 ...  
## $ Personal : num 2200 1500 1165 875 1500 ...  
## $ PhD : num 70 29 53 92 76 67 90 89 79 40 ...  
## $ Terminal : num 78 30 66 97 72 73 93 100 84 41 ...  
## $ S.F.Ratio : num 18.1 12.2 12.9 7.7 11.9 9.4 11.5 13.7 11.3 11.5 ...  
## $ perc.alumni: num 12 16 30 37 2 11 26 37 23 15 ...  
## $ Expend : num 7041 10527 8735 19016 10922 ...  
## $ Grad.Rate : num 60 56 54 59 15 55 63 73 80 52 ...

## 

## (b) Fit a linear model using least squares on the training set, and report the test error obtained.

fit.lm <- lm(Apps~., data=train)  
lm.pred <- predict(fit.lm, test)  
  
summary(fit.lm)

##   
## Call:  
## lm(formula = Apps ~ ., data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -5276.1 -473.2 -63.9 351.9 6574.0   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 78.15204 600.84427 0.130 0.896581   
## PrivateYes -757.22843 205.47577 -3.685 0.000263 \*\*\*  
## Accept 1.67981 0.05196 32.329 < 2e-16 \*\*\*  
## Enroll -0.62380 0.27629 -2.258 0.024544 \*   
## Top10perc 67.45654 8.45231 7.981 1.84e-14 \*\*\*  
## Top25perc -22.37500 6.57093 -3.405 0.000734 \*\*\*  
## F.Undergrad -0.06126 0.05468 -1.120 0.263258   
## P.Undergrad 0.04745 0.06248 0.760 0.448024   
## Outstate -0.09227 0.02889 -3.194 0.001524 \*\*   
## Room.Board 0.24513 0.07300 3.358 0.000867 \*\*\*  
## Books 0.09086 0.36826 0.247 0.805254   
## Personal 0.05886 0.09260 0.636 0.525455   
## PhD -8.89027 7.20890 -1.233 0.218271   
## Terminal -1.71947 8.22589 -0.209 0.834539   
## S.F.Ratio -5.75201 21.32871 -0.270 0.787554   
## perc.alumni -1.46681 6.28702 -0.233 0.815652   
## Expend 0.03487 0.01928 1.808 0.071361 .   
## Grad.Rate 7.57567 4.69602 1.613 0.107551   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1087 on 370 degrees of freedom  
## Multiple R-squared: 0.9397, Adjusted R-squared: 0.9369   
## F-statistic: 339.3 on 17 and 370 DF, p-value: < 2.2e-16

lm.error <- (mean(( test$Apps - lm.pred)^2))  
lm.error

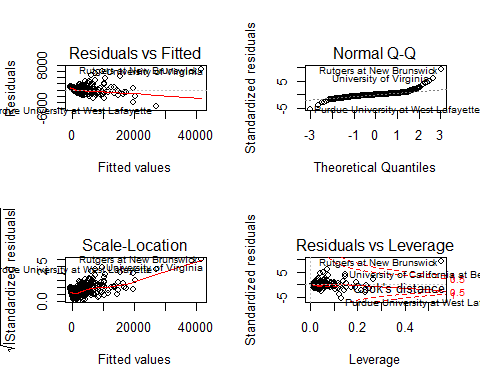
## [1] 1108531

RMSE.lm.error <- sqrt(mean( test$Apps - lm.pred)^2)  
RMSE.lm.error

## [1] 47.83397

**Answer:** *The total test error obtained when fitting the above full, linear regression model is MSE = 1108531. We see from the above output that we have seven significant Independent variables : PrivateYes, Accept, Top10perc, P.Undergrad, Room.Board, Expend, and Grad.RAte.*

par(mfrow=c(2,2))  
plot(fit.lm)

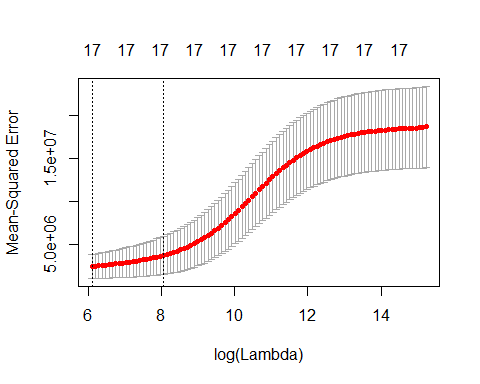


*We see from the above plots that there appears to be some outliers that are exuding leverage on the current fitted model, based off of the QQ plot above it appears as if overall the assumption of normality is valid and true, and there are no inherint visible patterns in the residual plots.*

## (c) Fit a ridge regression model on the training set, with lambda chosen by cross-validation. Report the test error obtained.

x.train <- model.matrix(Apps~., data=train)[,-1]  
x.test <- model.matrix(Apps~., data=test)[,-1]

fit.ridge = cv.glmnet(x.train, train$Apps, alpha=0)  
plot(fit.ridge)



***Answer:*** *The plot above can be used as a visual guide to choosing the best lambda. The lowest point in the curve indicates the optimal lambda: the log value of lambda that best minimised the error in cross-validation. We extract the optimal lambda below and see that the optimal lambda for ridge regression is 450.7435 or a normal scale or on a log scale is equal to 6.110898.*

ridge.lambda <- fit.ridge$lambda.min  
ridge.lambda

## [1] 450.7435

loglam <- log(ridge.lambda)  
loglam

## [1] 6.110898

ridge.predict <- predict(fit.ridge, s=loglam, newx= x.test)  
ridge.error <- mean((( ridge.predict - test$Apps)^2))  
ridge.error

## [1] 1028694

**Answer :** *So we see above that when fitting a ridge model with lambda = 6.1108, it leads us to a much lower test MSE than fitting a model with ols. The test MSE associated with this value of lambda is 1028694. Lastly, we refit our ridge regression model on the full data set, using the value of lambda chosen by cross validation, and examine the coefficient estimates.*

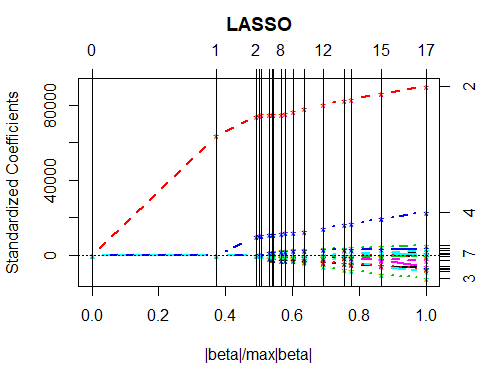
x.college = model.matrix(Apps~., data=College)[,-1]  
out = glmnet(x.college, College$Apps, alpha=0)  
ridge.coeffi<- predict(out, type="coefficients", s=loglam)  
ridge.coeffi

## 18 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -1.468326e+03  
## PrivateYes -5.278781e+02  
## Accept 1.004588e+00  
## Enroll 4.313442e-01  
## Top10perc 2.580619e+01  
## Top25perc 5.501092e-01  
## F.Undergrad 7.258520e-02  
## P.Undergrad 2.420595e-02  
## Outstate -2.407454e-02  
## Room.Board 1.987732e-01  
## Books 1.285477e-01  
## Personal -8.146130e-03  
## PhD -4.028284e+00  
## Terminal -4.811071e+00  
## S.F.Ratio 1.302180e+01  
## perc.alumni -8.544783e+00  
## Expend 7.589013e-02  
## Grad.Rate 1.126699e+01

**Answer:** *As we expected, none of the ridge regression coefficients are exactly set to zero, this is due to the fact that ridge regression does not perform variable selection.*

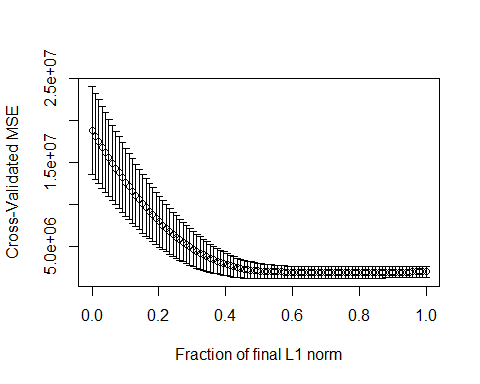
## (d) Fit a lasso model on the training set, with ?? chosen by crossvalidation. Report the test error obtained, along with the number of non-zero coefficient estimates.

lasso.fit <- lars(x.train, train$Apps, type="lasso")  
plot(lasso.fit, lwd=2, breaks=T)



***Answer:*** *We see from the above plot of the training fit of the lasso model to the college dataset, depending on what we set the tunning parameter to in the lasso model, some of the coefficients are exactly set to zero.*

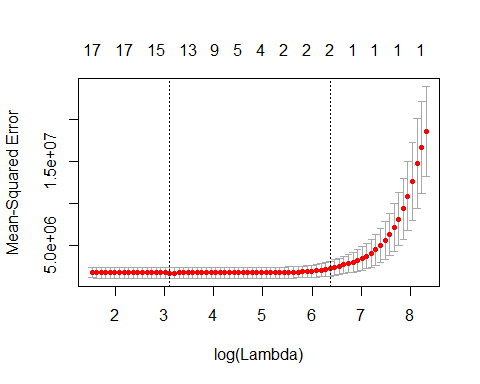
lasso.cv <- cv.lars(x.train, train$Apps, K=10, trace=F, plot.it=T, se=T, type="lasso")



title= ("lasso")

We now perform cross validation to compute the optimal lambda and the associated test error when using the optimal lambda found in the lasso model.

c.train <- model.matrix(Apps~., data=train)[,-1]  
c.test <- model.matrix(Apps~., data=test) [,-1]  
fit.lasso <- cv.glmnet(c.train, train$Apps, alpha=1)  
plot(fit.lasso)



lambdalasso<- fit.lasso$lambda  
lambdalasso

## [1] 4107.006527 3742.151386 3409.708971 3106.799823 2830.800290  
## [6] 2579.319794 2350.180132 2141.396607 1951.160834 1777.825083  
## [11] 1619.888003 1475.981617 1344.859478 1225.385869 1116.525966  
## [16] 1017.336876 926.959472 844.610947 769.578039 701.210847  
## [21] 638.917208 582.157564 530.440290 483.317437 440.380848  
## [26] 401.258627 365.611917 333.131962 303.537436 276.572007  
## [31] 252.002112 229.614939 209.216581 190.630356 173.695281  
## [36] 158.264671 144.204874 131.394110 119.721420 109.085699  
## [41] 99.394826 90.564863 82.519331 75.188541 68.508998  
## [46] 62.422847 56.877374 51.824545 47.220595 43.025647  
## [51] 39.203367 35.720648 32.547324 29.655910 27.021360  
## [56] 24.620857 22.433608 20.440668 18.624776 16.970202  
## [61] 15.462616 14.088960 12.837335 11.696902 10.657781  
## [66] 9.710973 8.848277 8.062221 7.345995 6.693397  
## [71] 6.098774 5.556976 5.063310 4.613499

lambda2<- 29.65591

***Answer :*** *We see from the plot above that when fitting the lasso model to the training data, the two parallel lines indicate two possible lambda values, with differing values of predictors illustrated at the top. After trying varrying values of lambda for prediction, we found that the lambda that gives the lowest prediction test error is equal to 29.65591.*

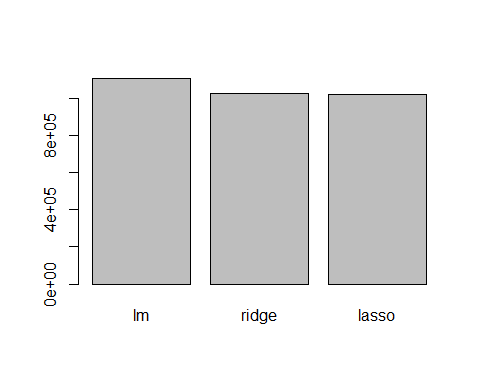
pred.lasso <- predict(fit.lasso, s=lambda2, newx=c.test)  
err.lasso <- (mean(( pred.lasso - test$Apps)^2))  
err.lasso

## [1] 1025248

**Answer :** *We see when using the optimal lambda found above (29.65591) and perform cross validation and compute the associated test error we get a value of 1025248. This error value is substantially less than the test MSE value found with the OLS model (1108531). The lasso test error value is similar but still lower than the test error found with the ridge model (1028694). The lasso model has a substantial advantage over ridge regression in that the resulting coefficient estimates are sparse. Below we see that three of the 17 independent variables are set to exactly zero. The lasso model with lambda chosen by cross-validation contains only 14 independent variables, they are: Private, Accept, Enroll, Top10perc, Top25perc, P.Undergrad, Outstate, Room.board, PhD, Terminal, S.F.Ratio, perc.alumni, Expend, grad.rate.*

## G) Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these five approaches?

MSE\_full <- c(lm.error, ridge.error, err.lasso)  
names(MSE\_full) <- c("lm", "ridge", "lasso")  
barplot(MSE\_full)



**Answer:** *The above barplot illustrates the differences between the test Mean Squared Error achieved for the lm, ridge, and lasso models. We see that the ridge and lasso models in terms of test MSE perform better than the ols model fit in part b of this question. There is only a marginal difference between between the ridge or lasso models, however we choose the lasso model over the ridge due to the lasso methods inherint ability to perform variable selection. We see that when choosing the lasso model, we have an overall test MSE equal to 1025248, compared to OLS = 1108531 or ridge = 1028694. While we do receive a somewhat better test MSE with lasso over ridge or ordinary least squares, the lasso for this particular problem does not provide a huge difference in prediction error over the OLS or Ridge methods.*

**#3) We will now try to predict per capita crime rate in the Boston data set.**

## (a) Try out some of the regression methods explored in this chapter,such as best subset selection, the lasso, ridge regression, and PCR. Present and discuss results for the approaches that you consider.

require(leaps)   
require(glmnet)   
require(MASS)

## Loading required package: MASS

## Warning: package 'MASS' was built under R version 3.4.3

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

data(Boston)

# split data into training and test sets  
set.seed(1)  
trainid <- sample(1:nrow(Boston), nrow(Boston)/2)  
trainb <- Boston[trainid,]  
testb <- Boston[-trainid,]  
x.matrix.b <- model.matrix(crim~., data=Boston)[,-1]  
b.train <- model.matrix(crim~., data=trainb)[,-1]  
b.test <- model.matrix(crim~., data=testb)[,-1]  
str(Boston)

## 'data.frame': 506 obs. of 14 variables:  
## $ crim : num 0.00632 0.02731 0.02729 0.03237 0.06905 ...  
## $ zn : num 18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...  
## $ indus : num 2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...  
## $ chas : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ nox : num 0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...  
## $ rm : num 6.58 6.42 7.18 7 7.15 ...  
## $ age : num 65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...  
## $ dis : num 4.09 4.97 4.97 6.06 6.06 ...  
## $ rad : int 1 2 2 3 3 3 5 5 5 5 ...  
## $ tax : num 296 242 242 222 222 222 311 311 311 311 ...  
## $ ptratio: num 15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...  
## $ black : num 397 397 393 395 397 ...  
## $ lstat : num 4.98 9.14 4.03 2.94 5.33 ...  
## $ medv : num 24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...

predict.regsubsets <- function(object, newdata, id, ...){  
 form <- as.formula(object$call[[2]])  
 mat <- model.matrix(form, newdata)  
 coefi <- coef(object, id=id)  
 xvars <- names(coefi)  
 mat[,xvars]%\*%coefi  
}

## Forward Selection

forward.selection.boston <- regsubsets(crim~., data=trainb, nvmax=ncol(Boston)-1)

forward.summary.boston <- summary(forward.selection.boston)  
forward.summary.boston

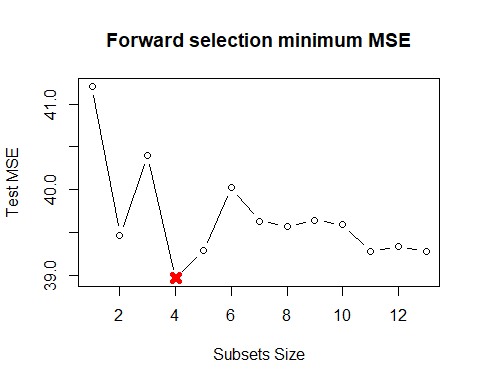
## Subset selection object  
## Call: regsubsets.formula(crim ~ ., data = trainb, nvmax = ncol(Boston) -   
## 1)  
## 13 Variables (and intercept)  
## Forced in Forced out  
## zn FALSE FALSE  
## indus FALSE FALSE  
## chas FALSE FALSE  
## nox FALSE FALSE  
## rm FALSE FALSE  
## age FALSE FALSE  
## dis FALSE FALSE  
## rad FALSE FALSE  
## tax FALSE FALSE  
## ptratio FALSE FALSE  
## black FALSE FALSE  
## lstat FALSE FALSE  
## medv FALSE FALSE  
## 1 subsets of each size up to 13  
## Selection Algorithm: exhaustive  
## zn indus chas nox rm age dis rad tax ptratio black lstat medv  
## 1 ( 1 ) " " " " " " " " " " " " " " "\*" " " " " " " " " " "   
## 2 ( 1 ) " " " " " " " " " " " " " " "\*" " " " " " " "\*" " "   
## 3 ( 1 ) " " " " " " " " "\*" " " " " "\*" " " " " " " "\*" " "   
## 4 ( 1 ) "\*" " " " " " " " " " " "\*" "\*" " " " " " " " " "\*"   
## 5 ( 1 ) "\*" " " " " "\*" " " " " "\*" "\*" " " " " " " " " "\*"   
## 6 ( 1 ) "\*" " " " " "\*" "\*" " " "\*" "\*" " " " " " " " " "\*"   
## 7 ( 1 ) "\*" " " " " "\*" "\*" " " "\*" "\*" " " " " " " "\*" "\*"   
## 8 ( 1 ) "\*" " " " " "\*" "\*" " " "\*" "\*" " " "\*" " " "\*" "\*"   
## 9 ( 1 ) "\*" " " " " "\*" "\*" " " "\*" "\*" "\*" "\*" " " "\*" "\*"   
## 10 ( 1 ) "\*" " " "\*" "\*" "\*" " " "\*" "\*" "\*" "\*" " " "\*" "\*"   
## 11 ( 1 ) "\*" " " "\*" "\*" "\*" " " "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 12 ( 1 ) "\*" " " "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 13 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"

error.forward.selection.boston <- rep(NA, ncol(Boston)-1)  
for (i in 1:(ncol(Boston)-1)){  
 prediction.forward.selection <- predict(forward.selection.boston, testb, id=i)  
 error.forward.selection.boston[i] <- mean((testb$crim - prediction.forward.selection)^2)  
}  
error.forward.selection.boston

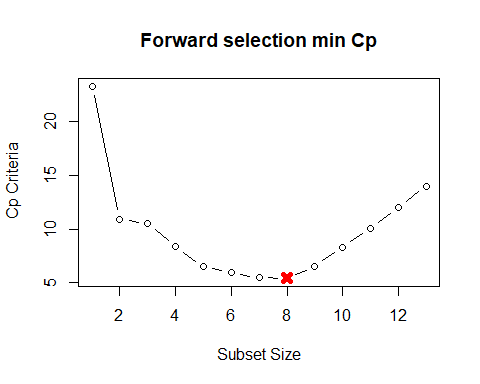
## [1] 41.20712 39.46705 40.40271 38.96427 39.28537 40.02194 39.63462  
## [8] 39.56733 39.64198 39.59028 39.27845 39.33711 39.27592

forward.error.subset\_4 <- 38.96427

forward.subset <- which.min(error.forward.selection.boston)  
plot(error.forward.selection.boston, type="b", main= "Forward selection minimum MSE", xlab = "Subsets Size", ylab="Test MSE")  
points(forward.subset, error.forward.selection.boston[forward.subset], col="red", pch=4, lwd=5)



forward.min.cp <- which.min(forward.summary.boston$cp)  
plot(forward.summary.boston$cp,type="b",main="Forward selection min Cp", xlab = "Subset Size", ylab= " Cp Criteria")  
points(forward.min.cp, forward.summary.boston$cp[forward.min.cp], col="red", pch=4, lwd=5)

   
  
**Answer:** *We see that when performing forward selection, best subset on training and validation data, the best Forward selection model that provides us with the lowest Cp value is a subset size of 8. However, in terms of test error MSE, the forward selection model with four variables proves the lowest estimated test MSE = 38.94 including the following four variables: zn, dis, rad, and medv.*

# Backward Selection

fit.backward.boston <- regsubsets(crim~., data=trainb, nvmax=ncol(Boston)-1)  
backward.summary.boston<- summary(fit.backward.boston)  
backward.summary.boston

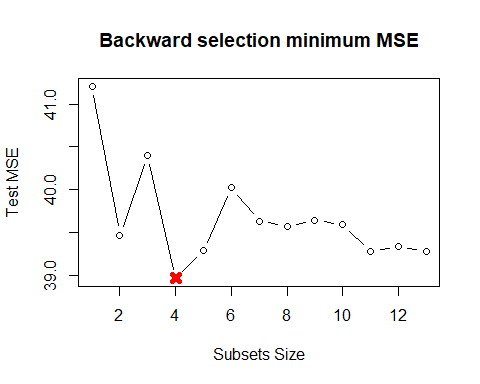
## Subset selection object  
## Call: regsubsets.formula(crim ~ ., data = trainb, nvmax = ncol(Boston) -   
## 1)  
## 13 Variables (and intercept)  
## Forced in Forced out  
## zn FALSE FALSE  
## indus FALSE FALSE  
## chas FALSE FALSE  
## nox FALSE FALSE  
## rm FALSE FALSE  
## age FALSE FALSE  
## dis FALSE FALSE  
## rad FALSE FALSE  
## tax FALSE FALSE  
## ptratio FALSE FALSE  
## black FALSE FALSE  
## lstat FALSE FALSE  
## medv FALSE FALSE  
## 1 subsets of each size up to 13  
## Selection Algorithm: exhaustive  
## zn indus chas nox rm age dis rad tax ptratio black lstat medv  
## 1 ( 1 ) " " " " " " " " " " " " " " "\*" " " " " " " " " " "   
## 2 ( 1 ) " " " " " " " " " " " " " " "\*" " " " " " " "\*" " "   
## 3 ( 1 ) " " " " " " " " "\*" " " " " "\*" " " " " " " "\*" " "   
## 4 ( 1 ) "\*" " " " " " " " " " " "\*" "\*" " " " " " " " " "\*"   
## 5 ( 1 ) "\*" " " " " "\*" " " " " "\*" "\*" " " " " " " " " "\*"   
## 6 ( 1 ) "\*" " " " " "\*" "\*" " " "\*" "\*" " " " " " " " " "\*"   
## 7 ( 1 ) "\*" " " " " "\*" "\*" " " "\*" "\*" " " " " " " "\*" "\*"   
## 8 ( 1 ) "\*" " " " " "\*" "\*" " " "\*" "\*" " " "\*" " " "\*" "\*"   
## 9 ( 1 ) "\*" " " " " "\*" "\*" " " "\*" "\*" "\*" "\*" " " "\*" "\*"   
## 10 ( 1 ) "\*" " " "\*" "\*" "\*" " " "\*" "\*" "\*" "\*" " " "\*" "\*"   
## 11 ( 1 ) "\*" " " "\*" "\*" "\*" " " "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 12 ( 1 ) "\*" " " "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 13 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"

error.backward.selection.boston <- rep(NA, ncol(Boston)-1)  
for (i in 1:(ncol(Boston)-1)){  
 prediction.backward.selection <- predict(fit.backward.boston, testb, id=i)  
 error.backward.selection.boston[i] <- mean((testb$crim - prediction.backward.selection)^2)  
}  
  
error.backward.selection.boston

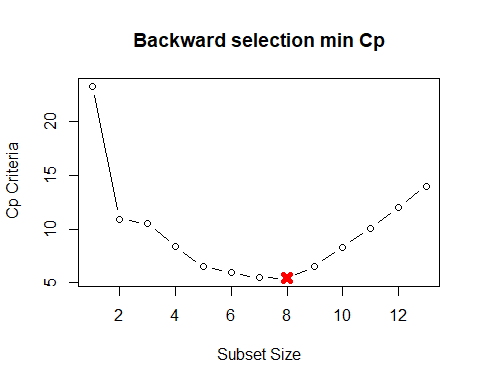
## [1] 41.20712 39.46705 40.40271 38.96427 39.28537 40.02194 39.63462  
## [8] 39.56733 39.64198 39.59028 39.27845 39.33711 39.27592

backward.best.subset <- c(Boston$zn,Boston$dis, Boston$rad, Boston$medv)  
  
backward.MSE.subset4 <- 38.96427

backward.subset <- which.min(error.backward.selection.boston)  
plot(error.backward.selection.boston, type="b", main= "Backward selection minimum MSE", xlab = "Subsets Size", ylab="Test MSE")  
points(backward.subset, error.backward.selection.boston[backward.subset], col="red", pch=4, lwd=5)



backward.min.cp <- which.min(backward.summary.boston$cp)  
plot(backward.summary.boston$cp,type="b",main="Backward selection min Cp", xlab = "Subset Size", ylab= " Cp Criteria")  
points(backward.min.cp, backward.summary.boston$cp[backward.min.cp], col="red", pch=4, lwd=5)

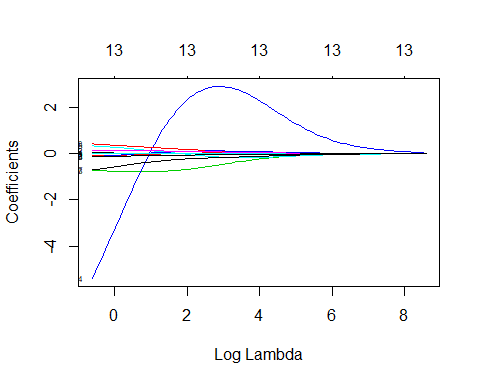


**Answer:** *We see that when performing backward selection, best subset on training and validation data, the best backward selection model that provides us with the lowest Cp value is a subset size of 8. However, in terms of test error MSE, the backward selection model with four variables proves the lowest estimated test MSE = 38.94 including the following four variables: zn, dis, rad, and medv. The forward and backward slection models perform virtually the exact same produceing the same model chosen in terms of subset size, Cp and those variables selected in the model with the lowest test MSE.*

subsetsize.backward <- which.min(error.backward.selection.boston)  
subsetsize.forward <- which.min(error.forward.selection.boston)

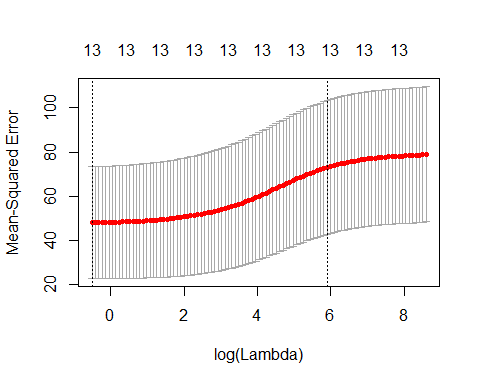
## Ridge

fit.ridge.boston.full <- glmnet(x.matrix.b, Boston$crim, alpha=0)  
plot(fit.ridge.boston.full, xvar="lambda", label=TRUE)



### Cross Validation of Ridge

ridge.fit.boston <- cv.glmnet(b.train, trainb$crim, alpha=0)  
plot(ridge.fit.boston)



ridge.lambda.boston <- ridge.fit.boston$lambda.min  
ridge.lambda.boston

## [1] 0.5982585

*We see above that when choosing the cross validation, the optimum lambda that provides us with the lowest Mean squared error we get a lambda value equal to .5982585.*

predict.ridge.boston <- predict(ridge.fit.boston, s=ridge.lambda.boston, newx = b.test)  
  
boston.ridge.error <- mean((testb$crim - predict.ridge.boston)^2)  
boston.ridge.error

## [1] 38.36353

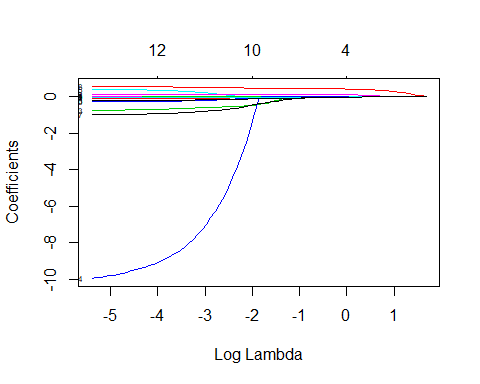
**Answer:** *When predicting the crime rate with the Boston data using the optimum lambda identified through cross validation, we get a prediction test error equal to 38.3653, this is only slightly better than the forward and backward prediction error observed. The ridge regression model, does not perform variable selection, thus the ridge method provides us with only slightly better test MSE, but at the sacrafice of complexity of the model. We would not favor the ridge regression method over the forward or backward selection methods. The coefficients for the variables in the final ridge model can be seen below.*

ridge.output <- predict(ridge.fit.boston, s=ridge.lambda.boston, type="coefficients")  
ridge.output

## 14 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) 4.429222285  
## zn 0.036521710  
## indus -0.061214283  
## chas -0.775621731  
## nox -8.045252067  
## rm 1.179585530  
## age 0.005683303  
## dis -0.884805550  
## rad 0.429755276  
## tax 0.003261068  
## ptratio -0.169564880  
## black -0.004036776  
## lstat 0.193028502  
## medv -0.171441090

## Lasso

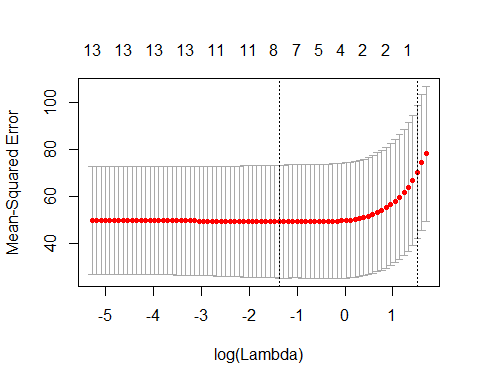
fit.lasso.boston.full <- glmnet(x.matrix.b, Boston$crim, alpha=1)  
plot(fit.lasso.boston.full, xvar="lambda", label=TRUE)



*Unlike ridge, we see that for the lasso model, at certain values for lambda coefficients are set exactly to zero. Essentially, this can be understood to mean that lasso regression performs variable selection.*

### Cross Validation of Lasso

fit.lasso.boston <- cv.glmnet(b.train, trainb$crim, alpha=1)  
plot(fit.lasso.boston)



lasso.boston.lambda <- fit.lasso.boston$lambda.min  
lasso.boston.lambda

## [1] 0.2530181

*The plot above helps to illustrate what is happening when looking for the optimum lambda value for the lasso model. We perform cross validation in order to choose the optimum lambda that provides us with the best test error, that is the lowest test error. As we see above, the optimum lambda for the laso model is a lambda equal to 0.082852.*

predict.lasso.boston <- predict(fit.lasso.boston, s=lasso.boston.lambda, newx=b.test)  
  
err.lasso.boston <- mean((testb$crim - predict.lasso.boston)^2)  
err.lasso.boston

## [1] 38.4593

lasso.subset<- predict(fit.lasso.boston, s=lasso.boston.lambda, type="coefficients")  
lasso.subset

## 14 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -1.803083100  
## zn 0.017767800  
## indus .   
## chas -0.266407052  
## nox .   
## rm 0.510067274  
## age .   
## dis -0.377582057  
## rad 0.457627436  
## tax .   
## ptratio .   
## black -0.002413684  
## lstat 0.159515883  
## medv -0.093917991

*The lasso model with a lambda equal to .082852 provides the smallest test mean squared error of all methods previously explored, the fnal test mean squared error for the lasso model is equal to 38.4593.. This is only a marginal improvement over the forward, backward or ridge methods explored above.*

## (b) Propose a model (or set of models) that seem to perform well on this data set, and justify your answer. Make sure that you are evaluating model performance using validation set error, crossvalidation, or some other reasonable alternative, as opposed to using training error.

# Forward Output and MSE

forward.error.subset\_4

## [1] 38.96427

forward.summary.boston

## Subset selection object  
## Call: regsubsets.formula(crim ~ ., data = trainb, nvmax = ncol(Boston) -   
## 1)  
## 13 Variables (and intercept)  
## Forced in Forced out  
## zn FALSE FALSE  
## indus FALSE FALSE  
## chas FALSE FALSE  
## nox FALSE FALSE  
## rm FALSE FALSE  
## age FALSE FALSE  
## dis FALSE FALSE  
## rad FALSE FALSE  
## tax FALSE FALSE  
## ptratio FALSE FALSE  
## black FALSE FALSE  
## lstat FALSE FALSE  
## medv FALSE FALSE  
## 1 subsets of each size up to 13  
## Selection Algorithm: exhaustive  
## zn indus chas nox rm age dis rad tax ptratio black lstat medv  
## 1 ( 1 ) " " " " " " " " " " " " " " "\*" " " " " " " " " " "   
## 2 ( 1 ) " " " " " " " " " " " " " " "\*" " " " " " " "\*" " "   
## 3 ( 1 ) " " " " " " " " "\*" " " " " "\*" " " " " " " "\*" " "   
## 4 ( 1 ) "\*" " " " " " " " " " " "\*" "\*" " " " " " " " " "\*"   
## 5 ( 1 ) "\*" " " " " "\*" " " " " "\*" "\*" " " " " " " " " "\*"   
## 6 ( 1 ) "\*" " " " " "\*" "\*" " " "\*" "\*" " " " " " " " " "\*"   
## 7 ( 1 ) "\*" " " " " "\*" "\*" " " "\*" "\*" " " " " " " "\*" "\*"   
## 8 ( 1 ) "\*" " " " " "\*" "\*" " " "\*" "\*" " " "\*" " " "\*" "\*"   
## 9 ( 1 ) "\*" " " " " "\*" "\*" " " "\*" "\*" "\*" "\*" " " "\*" "\*"   
## 10 ( 1 ) "\*" " " "\*" "\*" "\*" " " "\*" "\*" "\*" "\*" " " "\*" "\*"   
## 11 ( 1 ) "\*" " " "\*" "\*" "\*" " " "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 12 ( 1 ) "\*" " " "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 13 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"

# Backward Output and MSE

backward.MSE.subset4

## [1] 38.96427

backward.summary.boston

## Subset selection object  
## Call: regsubsets.formula(crim ~ ., data = trainb, nvmax = ncol(Boston) -   
## 1)  
## 13 Variables (and intercept)  
## Forced in Forced out  
## zn FALSE FALSE  
## indus FALSE FALSE  
## chas FALSE FALSE  
## nox FALSE FALSE  
## rm FALSE FALSE  
## age FALSE FALSE  
## dis FALSE FALSE  
## rad FALSE FALSE  
## tax FALSE FALSE  
## ptratio FALSE FALSE  
## black FALSE FALSE  
## lstat FALSE FALSE  
## medv FALSE FALSE  
## 1 subsets of each size up to 13  
## Selection Algorithm: exhaustive  
## zn indus chas nox rm age dis rad tax ptratio black lstat medv  
## 1 ( 1 ) " " " " " " " " " " " " " " "\*" " " " " " " " " " "   
## 2 ( 1 ) " " " " " " " " " " " " " " "\*" " " " " " " "\*" " "   
## 3 ( 1 ) " " " " " " " " "\*" " " " " "\*" " " " " " " "\*" " "   
## 4 ( 1 ) "\*" " " " " " " " " " " "\*" "\*" " " " " " " " " "\*"   
## 5 ( 1 ) "\*" " " " " "\*" " " " " "\*" "\*" " " " " " " " " "\*"   
## 6 ( 1 ) "\*" " " " " "\*" "\*" " " "\*" "\*" " " " " " " " " "\*"   
## 7 ( 1 ) "\*" " " " " "\*" "\*" " " "\*" "\*" " " " " " " "\*" "\*"   
## 8 ( 1 ) "\*" " " " " "\*" "\*" " " "\*" "\*" " " "\*" " " "\*" "\*"   
## 9 ( 1 ) "\*" " " " " "\*" "\*" " " "\*" "\*" "\*" "\*" " " "\*" "\*"   
## 10 ( 1 ) "\*" " " "\*" "\*" "\*" " " "\*" "\*" "\*" "\*" " " "\*" "\*"   
## 11 ( 1 ) "\*" " " "\*" "\*" "\*" " " "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 12 ( 1 ) "\*" " " "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 13 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"

# 

# Ridge Model Output and MSE

ridge.output

## 14 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) 4.429222285  
## zn 0.036521710  
## indus -0.061214283  
## chas -0.775621731  
## nox -8.045252067  
## rm 1.179585530  
## age 0.005683303  
## dis -0.884805550  
## rad 0.429755276  
## tax 0.003261068  
## ptratio -0.169564880  
## black -0.004036776  
## lstat 0.193028502  
## medv -0.171441090

boston.ridge.error

## [1] 38.36353

# Lasso Model output and MSE

err.lasso.boston

## [1] 38.4593

lasso.subset

## 14 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -1.803083100  
## zn 0.017767800  
## indus .   
## chas -0.266407052  
## nox .   
## rm 0.510067274  
## age .   
## dis -0.377582057  
## rad 0.457627436  
## tax .   
## ptratio .   
## black -0.002413684  
## lstat 0.159515883  
## medv -0.093917991

**Answer:** *We see that when evaluating all models run in part a of the question above, backward selection gives us a test MSE = 38.9642 with a variable subset size of 4. Forward selection gives us a test MSE = 38.96427 with a variable subset of 4. Both forward and backward selection provide models with the same subset size, same test MSE, and the exact same predictors which are : zn, dis, rad, and medv. The ridge regression gives us a test MSE = 38.36353. The ridge regression model performs only slightly better than the backward and forward selection models in terms of test MSE. However, we do not favor the ridge regression model due to the fact that it provides on a slightly better prediction in terms at MSE and this is accomplished at the cost of creating an incredibly complex model with all variables inside of it. Lastly, we look to the lasso model which provides a test MSE value of in the middle of all MSEs analyzed, with a test MSE = 38.4593. The lasso model we choose as the best model for predicting crime, from the Boston data. We choose the lasso model, due to its test MSE value which is a better MSE value than the Forward or Backward selection models. We favor the lasso model over the ridge due to its ability to perform variable selection, the lasso model has a variable subset consisting of 8 independent variables rather than the 13 independent variables found in the ridge model.*

## (c) Does your chosen model involve all of the features in the data set? Why or why not?

err.lasso.boston

## [1] 38.4593

lasso.subset

## 14 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -1.803083100  
## zn 0.017767800  
## indus .   
## chas -0.266407052  
## nox .   
## rm 0.510067274  
## age .   
## dis -0.377582057  
## rad 0.457627436  
## tax .   
## ptratio .   
## black -0.002413684  
## lstat 0.159515883  
## medv -0.093917991

**Answer:** *As stated previously, the chosen model, the lasso regression model, is the superior model of all explored. The lasso model was chosen partly due to its inherit feature selection, and ability to remove insignificant predictors in the model. Those predictors determined to be insignificant, and not able to provide any additional predictive value or reduction in test MSE were tax , indus, nox, ptratio and age. The lasso model does however contain the following independent variables found to be significant when predicting crime in Boston : zn, chas, rm, dis, rad, black, lstat, and medv. The lasso model provides us with a less complex model than the ridge regression, due to the variable subset, but a more predictively accurate model that the forward or backward selection models. The lasso’s test MSE is estimated to be equal to 38.4593.*