HW6

Trey Davidson

2024-10-03

### Question 1

A.)

Subset selection will most likely have smallest RSS as it looks at every possible model.

B.)

This can not be guartneed as Subset collection has the possibility of overfitting to training.

i.) True because once a predicter is added in the foward model, then it will never be removed.

ii.) True because one is removed.

iii).False backward and foward selection do not guartnee the samle predictators.

iv.) False again backward and foward selection do not guartnee the same results.

v.) False subset looks at the best model for k predictors meaning that a preidctor could be useful for k, but not for k + 1. Unlike foward and backward, which just adds or removes the best one meaning if you are in k model, then you will be in k + 1 model.

### Question 2

A.) Lasso is less flexaible meaning bias is higher, and variance is lower. Meaning that if the increase in bias is less then the decrease in variance the model will have more predictave accuracy.

Answer is iii

B.)

Just like lasso ridge decrease flexability so we would epect iii for the same reasons

Answer is iii

C.)

Non linnear methods are more flexabile therefore they will increase accuracy when the abs(increase in varinace) is less then the abs(decrease in bias).

### Linnear Regression

library(ISLR2)  
  
train\_percent <- 0.7  
  
  
data <- College  
  
set.seed(10)  
  
n <- nrow(data)  
  
train\_size <- floor(train\_percent \* n)  
  
train\_indices <- sample(seq\_len(n), size = train\_size)  
  
train\_data <- College[train\_indices, ]  
test\_data <- College[-train\_indices, ]  
  
lr\_model <- lm(Apps ~ ., data=train\_data)  
  
predictions <- predict(lr\_model, newdata = test\_data)  
  
MSE <- sum((predictions-test\_data$Apps)^2) / nrow(test\_data)  
  
RMSE <- sqrt(MSE)  
  
# Linnear Regression RMSE  
RMSE

## [1] 1146.281

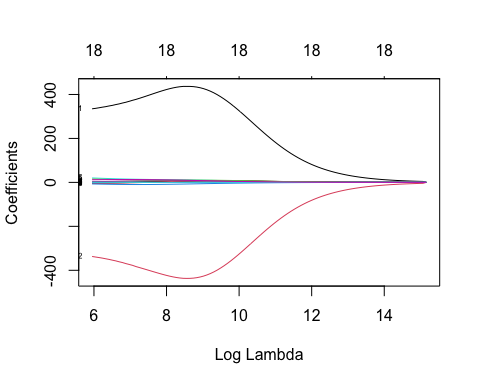
### Ridge

library(ISLR2)  
library(glmnet)

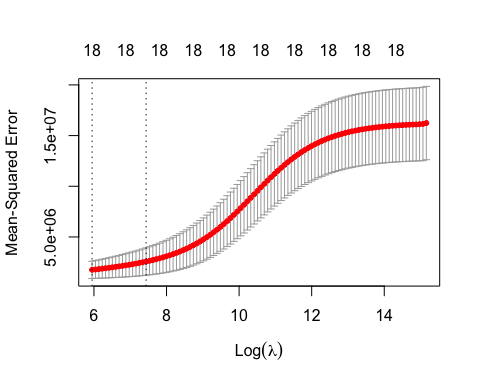
## Loading required package: Matrix

## Loaded glmnet 4.1-8

train\_percent <- 0.7  
  
  
data <- College  
  
set.seed(10)  
  
n <- nrow(data)  
  
train\_size <- floor(train\_percent \* n)  
  
train\_indices <- sample(seq\_len(n), size = train\_size)  
  
train\_data <- College[train\_indices, ]  
test\_data <- College[-train\_indices, ]  
  
  
x=model.matrix(Apps~. -1, data=train\_data)  
y=train\_data$Apps  
  
fit.ridge=glmnet(x, y, alpha=0)  
plot(fit.ridge, xvar="lambda", label = TRUE)



cv.ridge=cv.glmnet(x, y, alpha=0)  
  
#Cross Validation  
plot(cv.ridge)

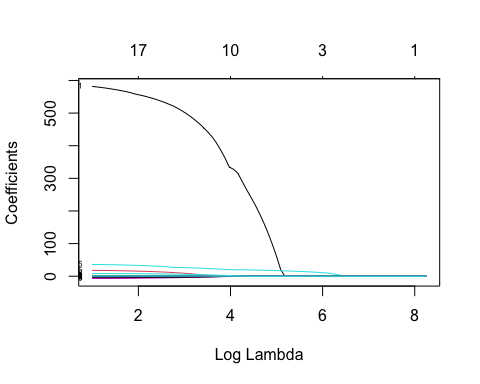


# This picks the best lambda  
best\_lambda <- cv.ridge$lambda.min  
  
  
  
ridge\_model = glmnet(x, y, alpha = 0, lambda = best\_lambda)  
  
  
  
x\_test = model.matrix(Apps ~ . - 1, data = test\_data)  
  
predictions = predict(fit.ridge, s = best\_lambda, newx = x\_test)  
  
MSE <- sum((predictions - test\_data$Apps)^2) / nrow(test\_data)  
  
RMSE <- sqrt(MSE)  
  
  
# Ridge RMSE  
RMSE

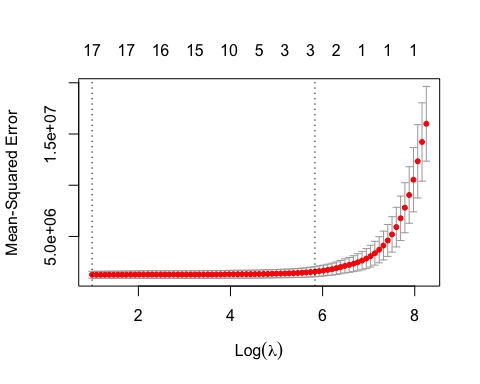
## [1] 1115.371

### Lasso

library(ISLR2)  
library(glmnet)  
  
# Set parameters  
train\_percent <- 0.7  
data <- College  
set.seed(10)  
  
# Split data  
n <- nrow(data)  
train\_size <- floor(train\_percent \* n)  
train\_indices <- sample(seq\_len(n), size = train\_size)  
  
train\_data <- College[train\_indices, ]  
test\_data <- College[-train\_indices, ]  
  
x <- model.matrix(Apps ~ . - 1, data = train\_data)  
y <- train\_data$Apps  
  
# Fit Lasso regression  
fit.lasso <- glmnet(x, y, alpha = 1) # Set alpha = 1 for Lasso  
plot(fit.lasso, xvar = "lambda", label = TRUE)



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



# Get the best lambda  
best\_lambda <- cv.lasso$lambda.min  
  
# Fit the Lasso model with the best lambda  
lasso\_model <- glmnet(x, y, alpha = 1, lambda = best\_lambda)  
  
  
x\_test <- model.matrix(Apps ~ . - 1, data = test\_data)  
  
# Make predictions  
predictions <- predict(lasso\_model, s = best\_lambda, newx = x\_test)  
  
# Calculate MSE and RMSE  
MSE <- sum((predictions - test\_data$Apps)^2) / nrow(test\_data)  
RMSE <- sqrt(MSE)  
  
  
  
# Get coefficients  
coefficients <- coef(lasso\_model)  
  
  
# Prints number of non zero coeeficents  
  
sum(coefficients != 0)

## [1] 18

RMSE

## [1] 1146.328

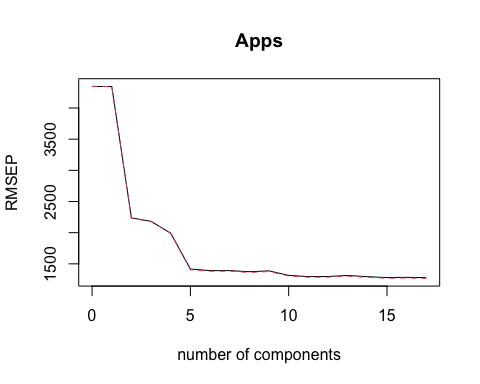
### PCR

library(ISLR2)  
library(pls)

##   
## Attaching package: 'pls'

## The following object is masked from 'package:stats':  
##   
## loadings

set.seed(10)  
  
  
train\_fraction <- 0.5  
train\_indices <- sample(seq\_len(nrow(College)), size = floor(train\_fraction \* nrow(College)))  
train\_data <- College[train\_indices, ]  
test\_data <- College[-train\_indices, ]  
  
  
pcr\_model <- pcr(Apps ~ ., data = train\_data, validation = "CV")  
  
  
  
  
  
  
plot(pcr\_model, plottype = "validation")



# Finds optimal by minimizeng validation error  
optimal\_ncomp <- which.min(pcr\_model$validation$PRESS)  
  
  
  
predictions <- predict(pcr\_model, newdata = test\_data, ncomp = optimal\_ncomp)  
  
  
MSE <- sum((predictions - test\_data$Apps)^2) / nrow(test\_data)  
  
RMSE <- sqrt(MSE)  
  
RMSE

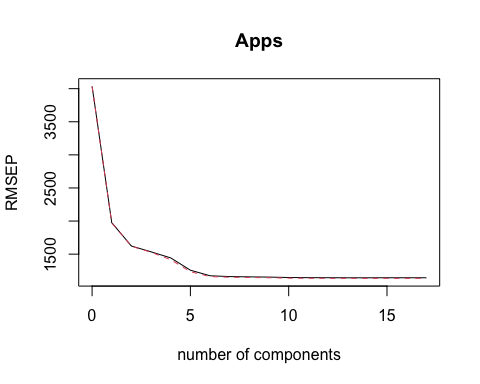
## [1] 1010

# Number of Components  
optimal\_ncomp

## [1] 17

### PLS

library(ISLR2)  
library(pls)  
  
  
set.seed(10)  
  
  
train\_fraction <- 0.7  
train\_indices <- sample(seq\_len(nrow(College)), size = floor(train\_fraction \* nrow(College)))  
train\_data <- College[train\_indices, ]  
test\_data <- College[-train\_indices, ]  
  
  
pls\_model <- plsr(Apps ~ ., data = train\_data, scale=TRUE, validation = "CV")  
  
  
plot(pls\_model, plottype = "validation")



optimal\_ncomp <- which.min(pls\_model$validation$PRESS)  
  
  
predictions <- predict(pls\_model, newdata = test\_data, ncomp = optimal\_ncomp)  
  
  
  
MSE <- sum((predictions - test\_data$Apps)^2) / nrow(test\_data)  
  
RMSE <- sqrt(MSE)  
  
  
# RMSE on test   
RMSE

## [1] 1146.367

# Number of Components  
optimal\_ncomp

## [1] 13

The Ridge Selection had the lowest Root Mean Sqaure Error on the test data. I did a 70% Train and 30% Test Split. I did them all on seed 10, which was a randomly picked by me. I tried other ones and some drastically lowered the RMSE of certain models, but on this seed all of them finished +- 100 of each other in terms of RMSE.

For me, if I did this code right, this shows how good of a regression linear is because its RMSE wasn’t that much worse then the best model and it is significant easier to implement.