Sta314 HW2 Problem 2

2023-10-17

#Problem 2

```
library(MASS)
set.seed(0)
```

##Part 1 Generate the training data with rho = 0.1 and n = 100

```
rho = 0.1
n = 100
beta = c(0.5,0.5)
mean = c(0,0,0)
sigma = matrix(c(1,0,0,0,1,rho,0,rho,1), nrow = 3)
X = mvrnorm(n, mean, sigma)
epsilon = rnorm(n) # since epsilon(i) with 1<=i<=n i.i.d. from N(0,1)
y = X[c(1,2)]%% beta + epsilon
sample = data.frame(cbind(X,y))
```

```
f_hat_1 = lm(y~V1 + V2, data = sample)
f_hat_2 = lm(y~V1 + V2 + V3, data = sample)
f_hat_3 = lm(y~V1 + V3, data = sample)

MSE_model_1 = mean((y - predict(f_hat_1 , data = sample))**2)
MSE_model_2 = mean((y - predict(f_hat_2 , data = sample))**2)
MSE_model_3 = mean((y - predict(f_hat_3 , data = sample))**2)

cat("MSE Model 1:", MSE_model_1, "\n")
```

```
## MSE Model 1: 0.8737418
```

```
cat("MSE Model 2:", MSE_model_2, "\n")
```

```
## MSE Model 2: 0.860581
```

```
cat("MSE Model 3:", MSE_model_3, "\n")
```

```
## MSE Model 3: 0.8725934
```

Get the same conclusions as in Problem 1. Since the model 2 has 3 features, such that the value of MSE(f_hat_2) is the lowest.

##Part 2 Repeat part 1 above N = 100

```
n = 100
mean = c(0, 0, 0)
sigma = matrix(c(1, 0.5, 0.5, 0.5, 1, 0.5, 0.5, 0.5, 1), nrow = 3)
beta = c(0.5, 0.5)
count = 0

for (i in 1:100) { #Since need to repeat 100 times,trying to use for i in range
  X = as.data.frame(mvrnorm(n, mean, sigma))
```

```
  epsilon = rnorm(n)
  y = X$V1 * beta[1] + X$V2 * beta[2] + epsilon
  sample = data.frame(X, y)

  f_hat_1 = lm(y ~ V1 + V2, data = sample)
  f_hat_2 = lm(y ~ V1 + V2 + V3, data = sample)
  f_hat_3 = lm(y ~ V1 + V3, data = sample)

  MSE_model_1 = mean((y - predict(f_hat_1, data = sample))^2)
  MSE_model_2 = mean((y - predict(f_hat_2, data = sample))^2)
  MSE_model_3 = mean((y - predict(f_hat_3, data = sample))^2)

  if (MSE_model_2 < MSE_model_1) {
    count = count + 1
  }
}

cat("Count:", count, "\n")
```

```
## Count: 100
```

For part 2, we need to repeat part 1 100 times, trying to use for loop function, set up the count = 0 at the begnning, since part 1 we get the conclusion that the model 2 has 3 features,such that the value of MSE(f_hat_2) is the lowest, set up the if statement, which is if (MSE_model_2 < MSE_model_1), finally, get the count = 100, which means MSE_model_2 always lower than MSE_model_1, but we can not compare MSE_model_1 and MSE_model_3, with the same consluion in Question 1 part 3.

##Part 3 Repeat part 2 above rho = 0.95

```
rho = 0.95
n = 100
mean = c(0, 0, 0)
sigma = matrix(c(1, 0.5, 0.5, 0.5, 1, 0.5, 0.5, 0.5, 1), nrow = 3)
beta = c(0.5, 0.5)
count1v2 = count1v3 = 0

for (i in 1:100) { #Since need to repeat 100 times,trying to use for i in range
  X = as.data.frame(mvrnorm(n, mean, sigma))
  epsilon = rnorm(n)
  y = X$V1 * beta[1] + X$V2 * beta[2] + epsilon
  sample = data.frame(X, y)

  f_hat_1 = lm(y ~ V1 + V2, data = sample)
  f_hat_2 = lm(y ~ V1 + V2 + V3, data = sample)
  f_hat_3 = lm(y ~ V1 + V3, data = sample)

  MSE_model_1 = mean((y - predict(f_hat_1, data = sample))^2)
  MSE_model_2 = mean((y - predict(f_hat_2, data = sample))^2)
  MSE_model_3 = mean((y - predict(f_hat_3, data = sample))^2)

  if (MSE_model_2 < MSE_model_1) {
    count1v2 = count1v2 + 1
  }

  if (MSE_model_1 < MSE_model_3) {
```

```
    count1v3 = count1v3 + 1
  }
}
print(paste("MSE Model 2 was lower than Model 1 in", count1v2,
            "out of 100 trials."))
```

```
## [1] "MSE Model 2 was lower than Model 1 in 100 out of 100 trials."
```

```
print(paste("MSE Model 1 was lower than Model 3 in", count1v3,
            "out of 100 trials."))
```

```
## [1] "MSE Model 1 was lower than Model 3 in 99 out of 100 trials."
```

Since rho represent the correlation between X2 and X3. From part 1 rho = 0.1, and part 3 rho = 0.95, the result shows that MSE Model 2 was lower than Model 1 in 100 out of 100 trials, which same result with different rho. rho = 0.95, which means X2 and X3 both have high correlation, which means f_hat_1 and f_hat_3 essential is the same, such that the result shows that MSE Model 1 was lower than Model 3 in 99 out of 100 trials

Sta314 HW2 Problem 3

2023-10-19

##1 Generate a data set

```
n = 1000
p = 20
beta = c(rep(2,3), rep(0.5,2), rep(0,15))

X = matrix(rnorm(n*p), nrow = n)
epsilion = rnorm(n)
y = X %*% beta + epsilion
```

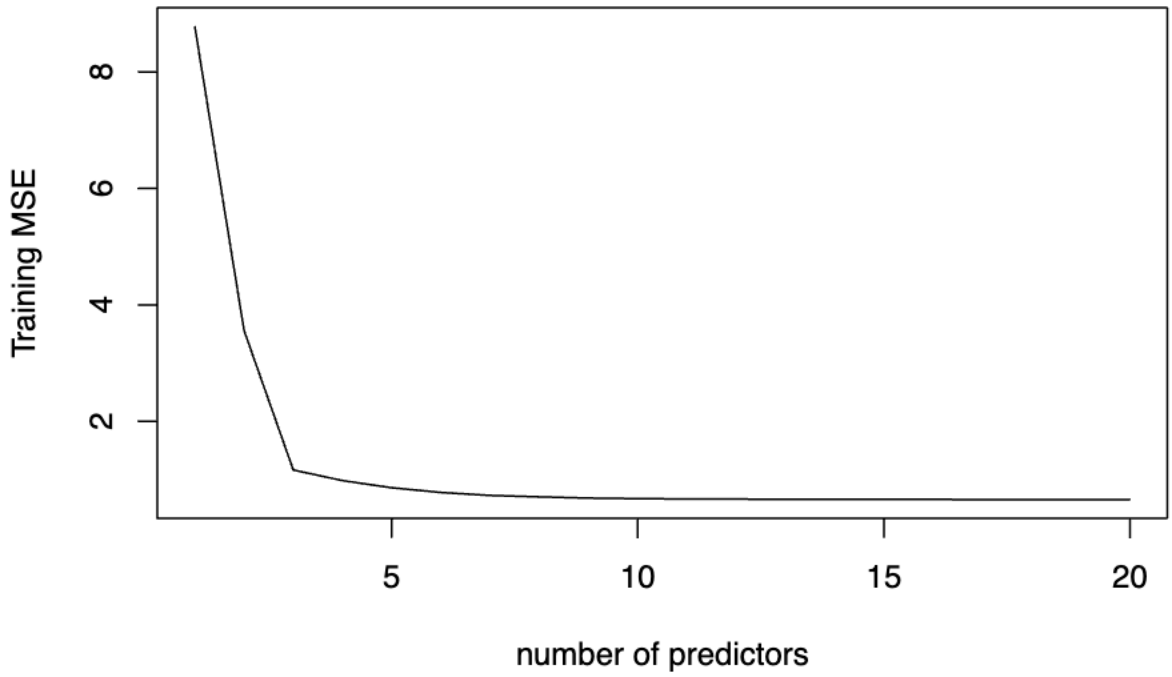
```
# Create data frame
data = data.frame(y, X)
```

##2 Randomly split dataset

```
set.seed(0)
index = sample(1:n, 100)

train.dataset = data[index, ] #training set containing 100 observations
test.dataset = data[-index, ] #test set containing 900 observations
```

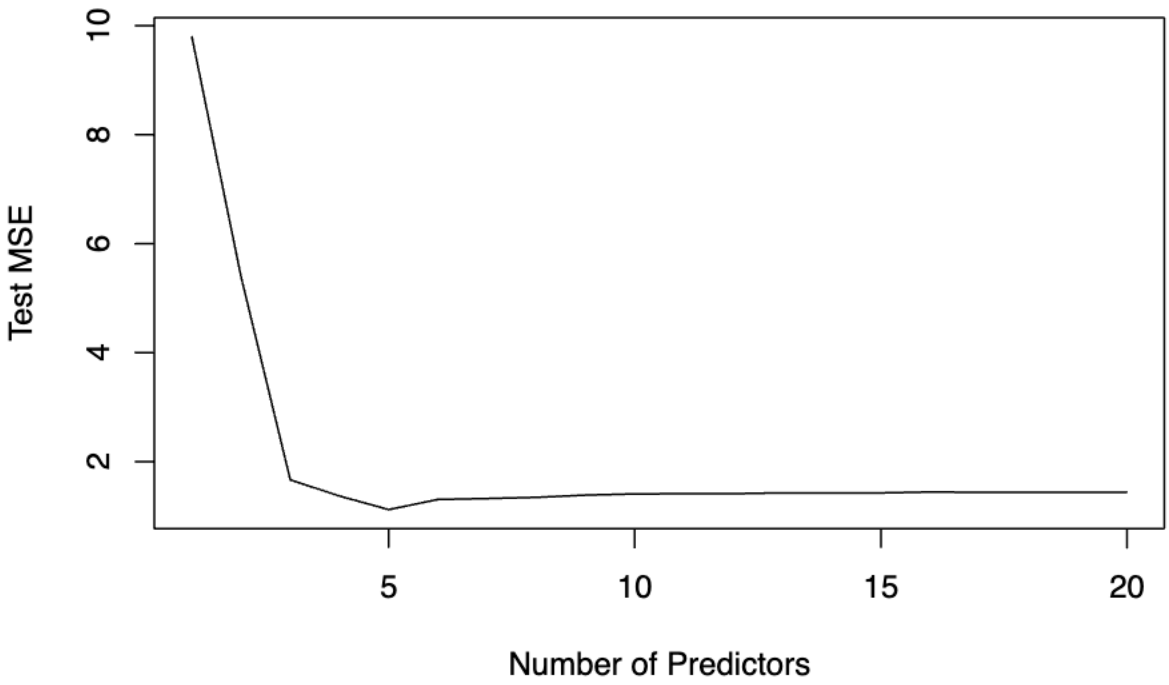
##3 Perform best subset selection

```
library(leaps)
regfit.best = regsubsets(y ~ . , data = train.dataset, nvmax = 20)
reg_summary = summary(regfit.best)
plot(reg_summary$rss/length(index), xlab = 'number of predictors',
     ylab ='Training MSE', type ='l')
```



##4 Plot the test set MSE associated with the best model of each size

```r
Test_MSE = rep(NA, 20)

for (i in 1:p) {
  coef_i = coef(regfit.best, id = i)
  has_intercept <- "(Intercept)" %in% names(coef_i)

  if (has_intercept) {
    predictors <- setdiff(names(coef_i), "(Intercept)")
} else {
    predictors <- names(coef_i)
}

  X_test = cbind(1, as.matrix(test.dataset[, predictors]))
  pred_i = X_test %*% coef_i
  Test_MSE[i] = mean((test.dataset$y - pred_i)^2)
}

plot(Test_MSE, type = "l", xlab = "Number of Predictors", ylab = "Test MSE")
```



##5 For which model size do the training set MSE and test set MSE

```
Train_MSE = reg_summary$rss
which.min(Train_MSE)
```

```
## [1] 20
```

```
which.min(Test_MSE)
```

```
## [1] 5
```

The model size is 20, the training MSE reaches its lowest.When the model size is 5, the test MSE is at its minimum. Model with more predictors fit the training data more closely. Such that selecting a model based on the training MSE can result in overfitting, but if using the test MSE as a guide often leads to a model that's more representative of the true model.

##6 How does the model at which the test set MSE is minimized

```
coef(regfit.best, id =5 )
```

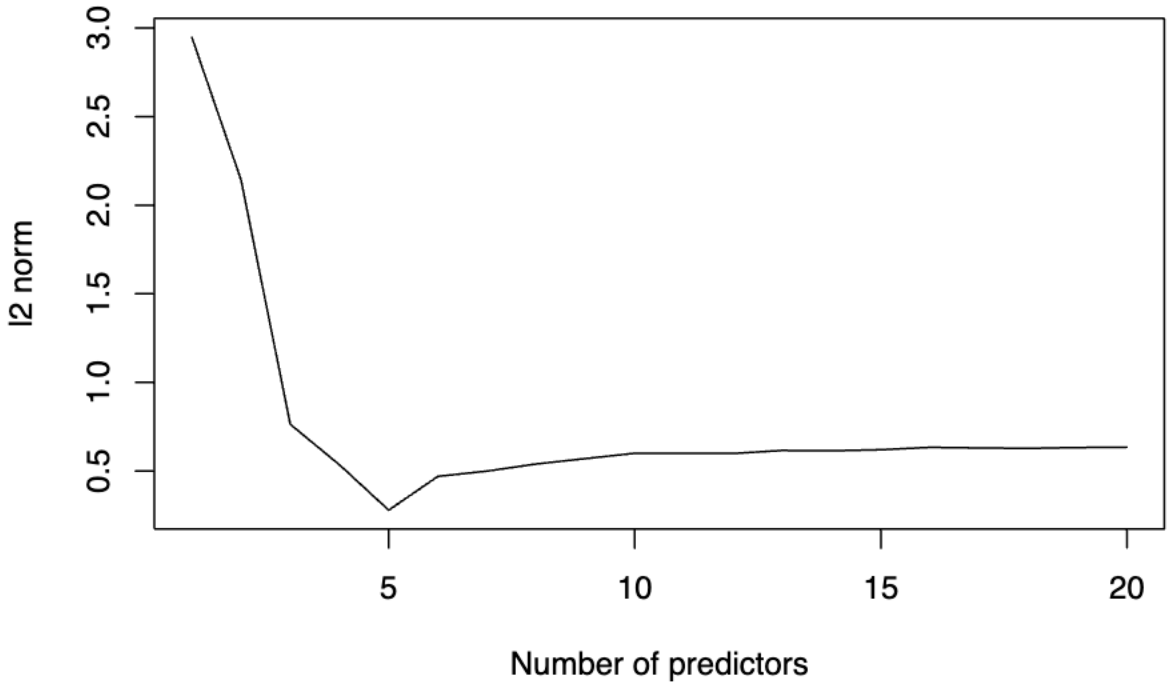
```
## (Intercept)          X1          X2          X3          X4          X5
##  0.06068251  2.11667329  2.19704836  2.01389150  0.34128119  0.51954004
```

Since in the part 1, we set the true model for the beta, beta1 - beta3 =2, and beta4-beta5 = 0.5, beta6 - beta20 = 0. The reslut shows that coefficent of X1_hat - X3_hat more close to 2, and coefficent X4_hat -X5_hatclose to 0.5. Such that selecting a model based on the test MSE will more close to true model. Since the epsilion exist, which the test set not exaclty 100% like true model.

##7 Create a plot displaying

```
result = rep(NA, p)

for(i in 1:p) {
  beta.hati = rep(0, p + 1)
  names(beta.hati) = c("(Intercept)", colnames(test.dataset)[-1])

  coef_i = coef(regfit.best, id = i)
```

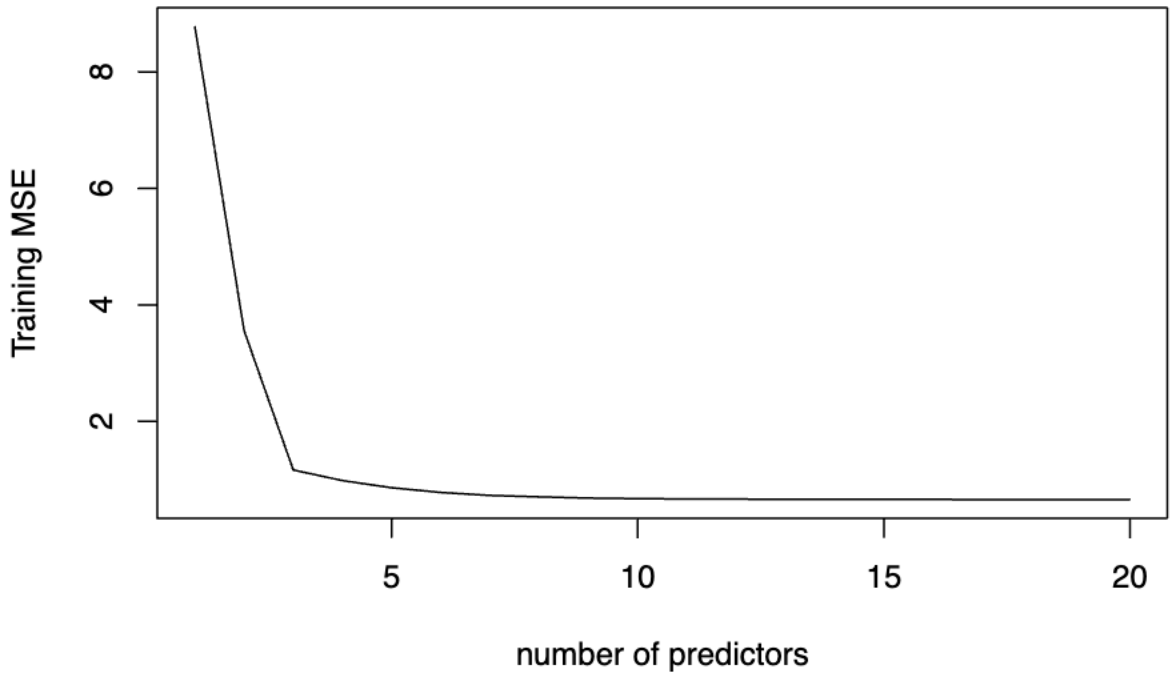
```
  beta.hati[names(coef_i)] = coef_i

  result[i] <- sqrt(sum((beta - beta.hati[-1])^2))
}

plot(result, xlab = "Number of predictors", ylab = "l2 norm", type = "l")
```



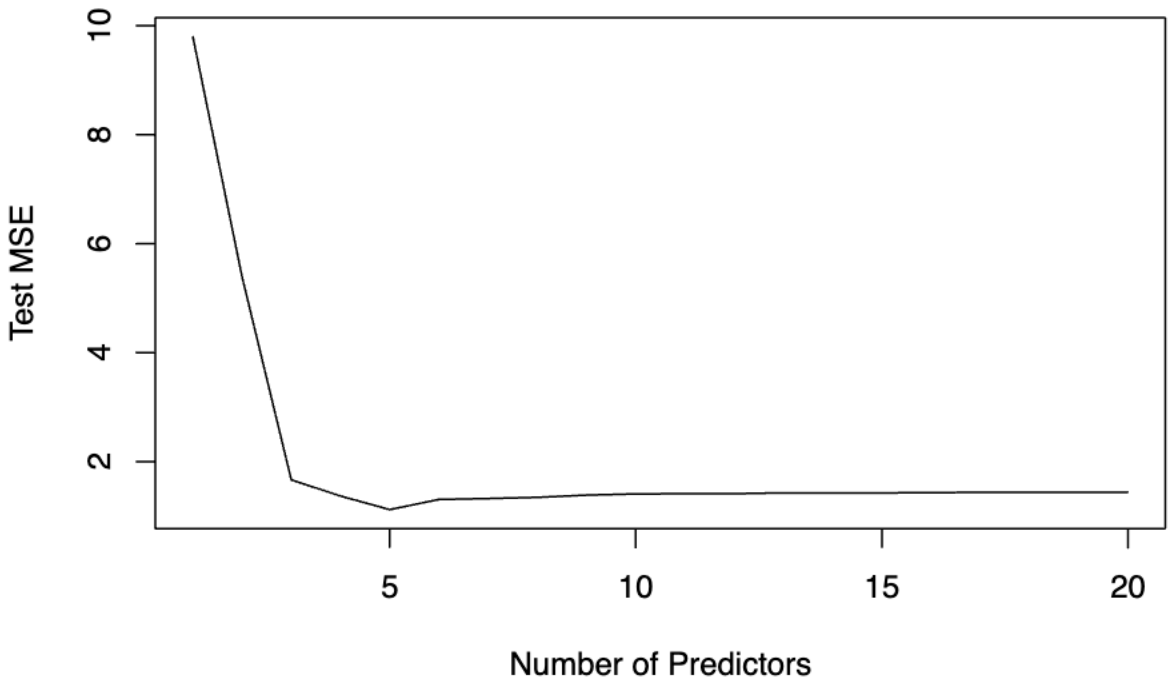
Since the result shows that at size 5, has the minimum value.Which is similar with part 4 test MSE plot.

##8 Repeat parts 3-6 for forward

```
#3
library(leaps)
regfit.forward = regsubsets(y ~ . , data = train.dataset, nvmax = 20,
                            method = 'forward')
reg_summary = summary(regfit.forward)
plot(reg_summary$rss/length(index), xlab = 'number of predictors',
     ylab ='Training MSE', type ='l')
```




```r
#4
Test_MSE = rep(NA, 20)

for (i in 1:p) {
  coef_i = coef(regfit.forward, id = i)
  has_intercept <- "(Intercept)" %in% names(coef_i)

  if (has_intercept) {
    predictors <- setdiff(names(coef_i), "(Intercept)")
} else {
    predictors <- names(coef_i)
}

  X_test = cbind(1, as.matrix(test.dataset[, predictors]))
  pred_i = X_test %*% coef_i
  Test_MSE[i] = mean((test.dataset$y - pred_i)^2)
}

plot(Test_MSE, type = "l", xlab = "Number of Predictors", ylab = "Test MSE")
```




```
#5
Train_MSE = reg_summary$rss
which.min(Train_MSE)
```

```
## [1] 20
```

```
which.min(Test_MSE)
```

```
## [1] 5
```

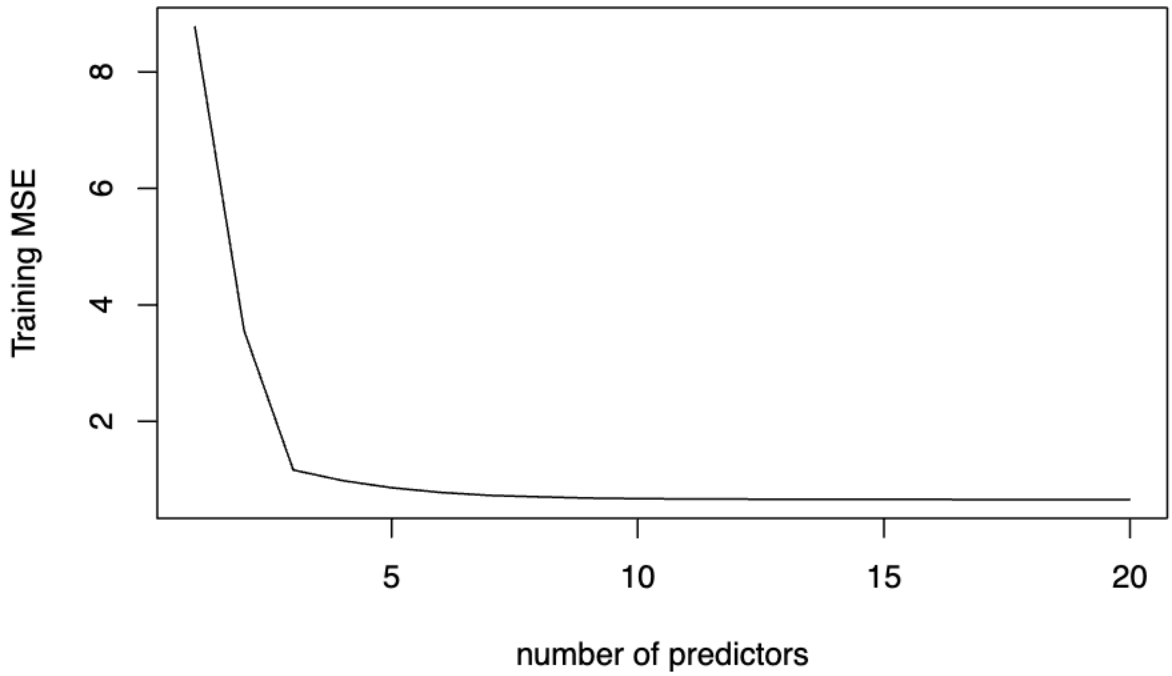
The model size is 20, the training MSE reaches its lowest.When the model size is 5, the test MSE is at its minimum.

```
#6
coef(regfit.forward, id =5 )
```

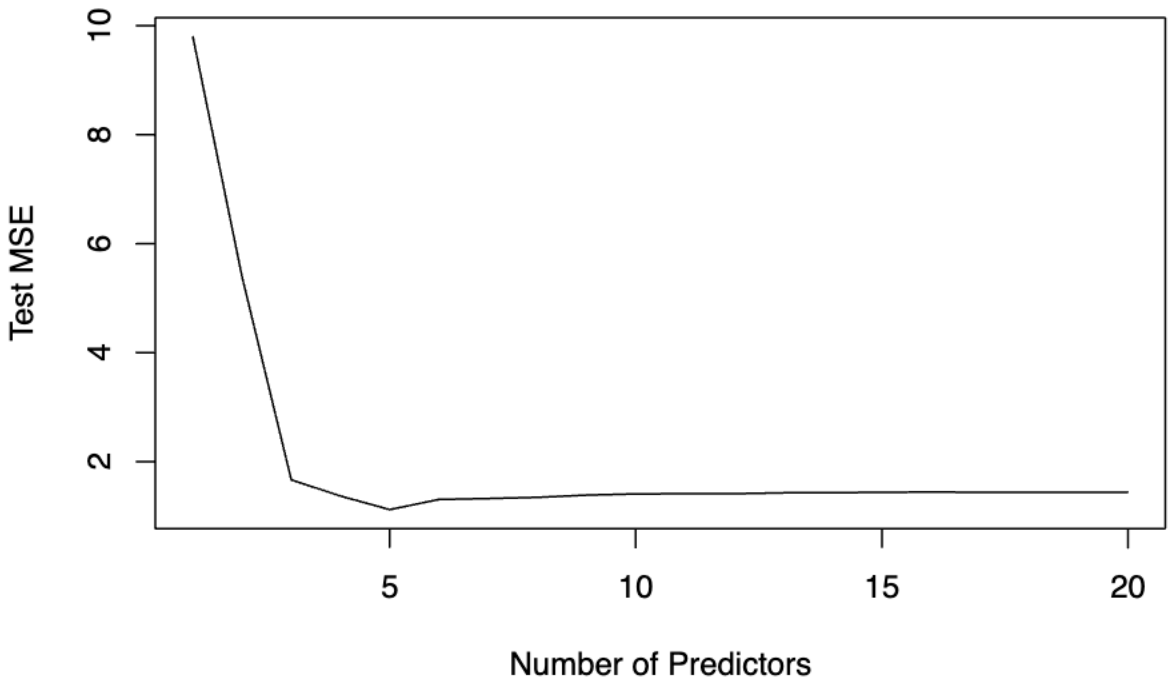
```
## (Intercept)          X1          X2          X3          X4          X5
##  0.06068251  2.11667329  2.19704836  2.01389150  0.34128119  0.51954004
```

##9 Repeat parts 3-6 for backward

```
#3
library(leaps)
regfit.backward = regsubsets(y ~ . , data = train.dataset, nvmax = 20,
                        method = 'backward')
reg_summary = summary(regfit.backward)
plot(reg_summary$rss/length(index), xlab = 'number of predictors',
     ylab ='Training MSE', type ='l')
```




```r
#4
Test_MSE = rep(NA, 20)

for (i in 1:p) {
  coef_i = coef(regfit.backward, id = i)
  has_intercept <- "(Intercept)" %in% names(coef_i)

  if (has_intercept) {
    predictors <- setdiff(names(coef_i), "(Intercept)")
} else {
    predictors <- names(coef_i)
}

  X_test = cbind(1, as.matrix(test.dataset[, predictors]))
  pred_i = X_test %*% coef_i
  Test_MSE[i] = mean((test.dataset$y - pred_i)^2)
}

plot(Test_MSE, type = "l", xlab = "Number of Predictors", ylab = "Test MSE")
```




```
#5
Train_MSE = reg_summary$rss
which.min(Train_MSE)
```

```
## [1] 20
```

```
which.min(Test_MSE)
```

```
## [1] 5
```

The model size is 20, the training MSE reaches its lowest.When the model size is 5, the test MSE is at its minimum.

```
#6
coef(regfit.backward, id =5 )
```

```
## (Intercept)          X1          X2          X3          X4          X5
##  0.06068251  2.11667329  2.19704836  2.01389150  0.34128119  0.51954004
```

Sta314 Hw3 Problem 4

2023-10-18

a~d

```
p = 50
n = 1100
set.seed(0)
X = matrix(rnorm(n * p), nrow = n,)
epsilon = rnorm(n)
beta = c(rep(2, 5), rep(0, 45)) # since p = 50
y = X %*% beta + epsilon

index = sample(1:n, 100) # randomly split the data: training 100, test 1000

train_x = X[index, ]
train_y = y[index]

test_x = X[-index, ]
test_y = y[-index]

grid = 10^seq(10, -2, length = 100)
```

1 Fit both the ridge regression and the lasso

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
# Ridge
ridge_cv = cv.glmnet(train_x, train_y, alpha = 0, lambda = grid)
best_lam = ridge_cv$lambda.min
ridge.mod = glmnet(train_x, train_y, alpha = 0, lambda = best_lam)
pred.ridge = predict(ridge.mod, test_x, lambda = best_lam)
MSE.ridge = mean((test_y - pred.ridge)**2)
MSE.ridge
```

```
## [1] 1.87885
```

lasso

```
# Lasso
lasso_cv = cv.glmnet(train_x, train_y, alpha = 1, lambda = grid)
best_lam = lasso_cv$lambda.min
lasso.mod = glmnet(train_x, train_y, alpha = 1, lambda = best_lam)
pred.lasso = predict(lasso.mod, test_x, lambda = best_lam)
```

```r
MSE.lasso = mean((test_y - pred.lasso)^2)
MSE.lasso
```

```
## [1] 1.150411
```

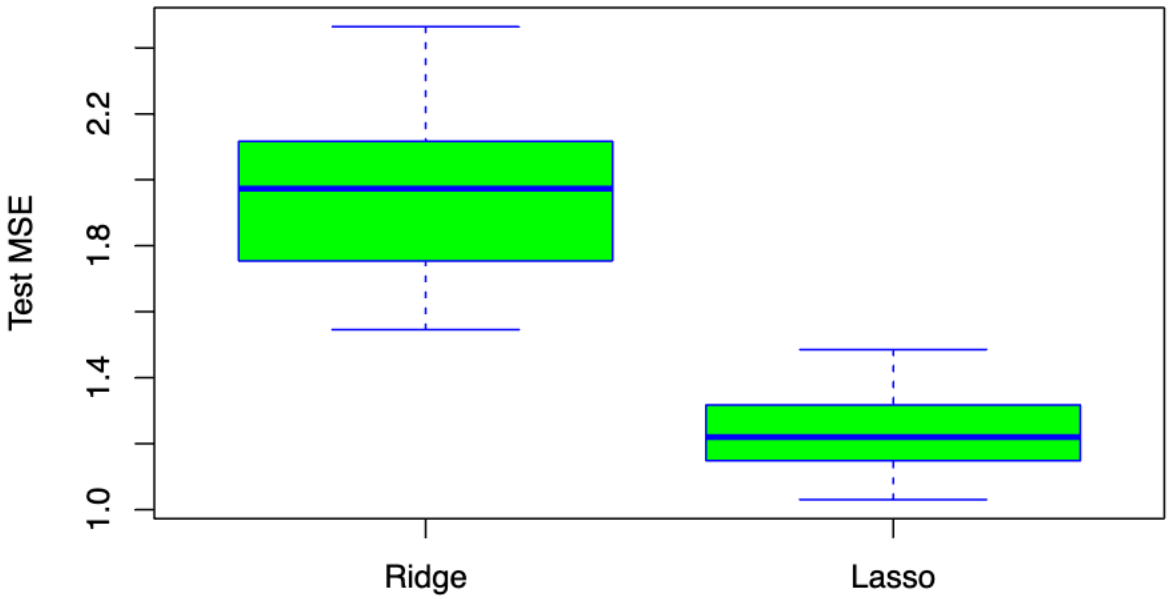
Lasso leads to smaller test set MSE.

2 Repeat steps a~d for generating the data by using different seeds

```r
Cal_MSE <- function(seed, p = 50, n = 1100) {
  set.seed(seed)
  X = matrix(rnorm(n * p), nrow = n)
  epsilon = rnorm(n)
  beta = c(rep(2, 5), rep(0, 45)) # since p = 50
  y = X %*% beta + epsilon

  index = sample(1:n, 100)
  train_x = X[index, ]
  train_y = y[index]
  test_x = X[-index, ]
  test_y = y[-index]
  grid = 10^seq(10, -2, length = 100)

  # Ridge
  ridge_cv = cv.glmnet(train_x, train_y, alpha = 0, lambda = grid)
  best_lam = ridge_cv$lambda.min
  ridge.mod = glmnet(train_x, train_y, alpha = 0, lambda = best_lam)
  pred.ridge = predict(ridge.mod, test_x, lambda = best_lam)
  MSE.ridge = mean((test_y - pred.ridge)**2)
  MSE.ridge

  # Lasso
  lasso_cv = cv.glmnet(train_x, train_y, alpha = 1, lambda = grid)
  best_lam = lasso_cv$lambda.min
  lasso.mod = glmnet(train_x, train_y, alpha = 1, lambda = best_lam)
  pred.lasso = predict(lasso.mod, test_x, lambda = best_lam)
  MSE.lasso = mean((test_y - pred.lasso)^2)
  MSE.lasso
  return(c(MSE_ridge = MSE.ridge , MSE_lasso = MSE.lasso ))
}
```

```r
MSE_ridge_list = c(MSE.ridge)
MSE_lasso_list = c(MSE.lasso)

for (i in 2:50){ #repeat ,since the first MSE for ridge and lasso exist
  MSE = Cal_MSE(i)
  MSE_ridge_list = c(MSE_ridge_list, MSE[1])
  MSE_lasso_list = c(MSE_lasso_list, MSE[2])
}
```

```r
boxplot(MSE_ridge_list, MSE_lasso_list, names= c('Ridge', 'Lasso'),
        ylab = 'Test MSE',
        col='green',
        border ='blue')
```



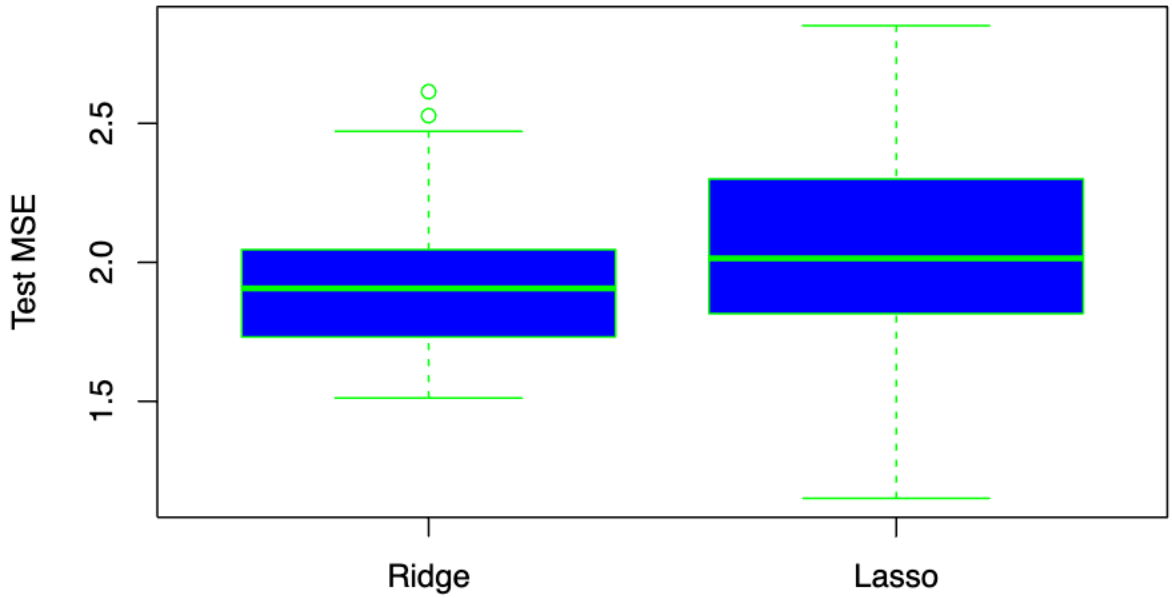
Using the boxplot,Ridge's MSE large than Lasso's MSE, also Ridge's MSE variance larger than Lasso's MSE.

3 Redo part 1 and 2 by using betaj = 0.5 for all j=1„,50 in step b

```
Cal_MSE <- function(seed, p = 50, n = 1100) {
  set.seed(seed)
  X = matrix(rnorm(n * p), nrow = n)
  epsilon = rnorm(n)
  beta = c(rep(0.5, 50)) # since p = 50
  y = X %*% beta + epsilon

  index = sample(1:n, 100)
  train_x = X[index, ]
  train_y = y[index]
  test_x = X[-index, ]
  test_y = y[-index]
  grid = 10^seq(10, -2, length = 100)

  # Ridge
  ridge_cv = cv.glmnet(train_x, train_y, alpha = 0, lambda = grid)
  best_lam = ridge_cv$lambda.min
  ridge.mod = glmnet(train_x, train_y, alpha = 0, lambda = best_lam)
  pred.ridge = predict(ridge.mod, test_x, lambda = best_lam)
  MSE.ridge = mean((test_y - pred.ridge)**2)
  MSE.ridge

  # Lasso
  lasso_cv = cv.glmnet(train_x, train_y, alpha = 1, lambda = grid)
  best_lam = lasso_cv$lambda.min
  lasso.mod = glmnet(train_x, train_y, alpha = 1, lambda = best_lam)
  pred.lasso = predict(lasso.mod, test_x, lambda = best_lam)
  MSE.lasso = mean((test_y - pred.lasso)^2)
  MSE.lasso
  return(c(MSE_ridge = MSE.ridge , MSE_lasso = MSE.lasso ))
}
```

```r
MSE_ridge_list = c(MSE.ridge)
MSE_lasso_list = c(MSE.lasso)

for (i in 2:50){ #repeat ,since the first MSE for ridge and lasso exist
  MSE = Cal_MSE(i)
  MSE_ridge_list = c(MSE_ridge_list, MSE[1])
  MSE_lasso_list = c(MSE_lasso_list, MSE[2])
}
```

```r
boxplot(MSE_ridge_list, MSE_lasso_list, names= c('Ridge', 'Lasso'),
        ylab = 'Test MSE',
        col='blue',
        border ='green')
```



Since change the beta to 0.5, Lasso's MSE bigger than Ridge's MSE. The variance of Lasso MSE slightly large than Ridge's MSE.