HW4_MSA 8150

Anutida Sangkla ID: 002602236

2/24/2021

Question 1

part (a)

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$RSS = \sum_{i=1}^{n} (y_i - \beta x_i)^2$$

$$RSS'(\beta) = \sum_{i=1}^{n} (y_i - \beta x_i)^2\ '$$

$$-2\sum_{i=1}^{n} (y_i - \beta x_i)x_i = 0$$

$$\sum_{i=1}^{n} (x_i y_i - \beta x_i^2) = 0$$

$$\hat{\beta} = \frac{\sum_{i=1}^{n} x_i\, y_i}{\sum_{i=1}^{n} x_i^2}$$

Since we know that $S_{xy} = \sum_{i=1}^{n} x_i\, y_i$ and $S_{xx} = \sum_{i=1}^{n} x_i^2$

Therefore, the fitted values can be acquired via

$$\boldsymbol{\hat{\beta}} = \frac{\boldsymbol{S_{xy}}}{\boldsymbol{S_{xx}}}$$

part (b)

$$RSS = \sum_{i=1}^{n} (y_i - \beta x_i)^2 + (y_j - \beta^{(j)} x_j)^2$$

$$RSS'(\beta) = \sum_{i=1}^{n} (y_i - \beta x_i)^2 + (y_j - \beta^{(j)} x_j)^2\ '$$

$$-2 \sum_{i=1}^{n} (y_i - \beta x_i) x_i - 2(y_j - \beta^{(j)} x_j) x_j = 0$$

$$\sum_{i=1}^{n} (x_i y_i - \beta x_i^2) - (x_j y_j - \beta^{(j)} x_j^2) = 0$$

$$\tilde{\beta} = \frac{\sum_{i=1}^{n} x_i\, y_i - x_j y_j}{\sum_{i=1}^{n} x_i^2 - x_j^2}$$

Since we know that $S_{xy} = \sum_{i=1}^{n} x_i\, y_i$ and $S_{xx} = \sum_{i=1}^{n} x_i^2$

Therefore, the fitted value can be acquired via

$$\boldsymbol{\tilde{\beta} = \frac{S_{xy} - x_j y_j}{S_{xx} - x_j^2}}$$

part (c)

From

$$MSE_j = (y_j - \beta^{(j)} x_j)^2$$

$$MSE_j = (y_j - \frac{S_{xy} - x_j y_j}{S_{xx} - x_j^2} x_j)^2$$

$$MSE_j = \left(\frac{y_j S_{xx} - y_j x_j^2 - x_j S_{xy} - x_j^2 y_j}{S_{xx} - x_j^2}\right)^2$$

Therefore,

$$\boldsymbol{MSE_j = \left(\frac{y_j S_{xx} - x_j S_{xy}}{S_{xx} - x_j^2}\right)^2}$$

part (d)

From

$$CV_n = \frac{1}{n} \sum_{j=1}^{n} \left(\frac{y_j - \hat{\beta} x_j}{1 - h_j}\right)^2$$

Where $h_j = \frac{x_j^2}{S_{xx}}$

$$CV_n = \frac{1}{n}\sum_{j=1}^{n}\left(\frac{y_j - \hat{\beta}x_j}{1 - \frac{x_j^2}{S_{xx}}}\right)^2$$

$$CV_n = \frac{1}{n}\sum_{j=1}^{n}\left(\frac{y_j - \frac{x_j S_{xy}}{S_{xx}}}{1 - \frac{x_j^2}{S_{xx}}}\right)^2$$

$$CV_n = \frac{1}{n}\sum_{j=1}^{n}\left(\frac{\frac{y_j S_{xx} - x_j S_{xy}}{S_{xx}}}{\frac{S_{xx} - x_j^2}{S_{xx}}}\right)^2$$

$$CV_n = \frac{1}{n}\sum_{j=1}^{n}\left(\frac{y_j S_{xx} - x_j S_{xy}}{S_{xx} - x_j^2}\right)^2$$

$$CV_n = \frac{1}{n}\sum_{j=1}^{n} MSE_j$$

Therefore, $CV_n = \frac{1}{n}\sum_{j=1}^{n} MSE_j$ can be written in the form $CV_n = \frac{1}{n}\sum_{j=1}^{n}\left(\frac{y_j - \hat{\beta}x_j}{1-h_j}\right)^2$

part (e)

```
library(boot)

## Warning: package 'boot' was built under R version 4.0.4

SimpleReg <- read.csv("SimpleReg.csv", header=TRUE,  sep=",")
str(SimpleReg)

## 'data.frame':    1500 obs. of  2 variables:
##  $ x: num  0.997 0.622 0.715 0.464 0.65 ...
##  $ y: num  0.0029 0.3467 0.1276 0.6836 0.2887 ...

## Fitting a linear model

lm.fit<- lm(y~x -1, data = SimpleReg)
summary(lm.fit)

##
## Call:
```

```
## lm(formula = y ~ x - 1, data = SimpleReg)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0929  0.2587  0.9099  1.0040  1.1064
##
## Coefficients:
##   Estimate Std. Error t value Pr(>|t|)
## x -0.09245    0.03793  -2.437   0.0149 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8507 on 1499 degrees of freedom
## Multiple R-squared:  0.003947,   Adjusted R-squared:  0.003283
## F-statistic:  5.94 on 1 and 1499 DF,  p-value: 0.01491
```

```
## Calculating CV using cv.glm

glm.fit<- glm(y~x -1, data = SimpleReg)
cv.err<-cv.glm(SimpleReg, glm.fit)$delta[1]
sprintf('%s = %3.6f', 'The LOOCV CV', cv.err)
```

```
## [1] "The LOOCV CV = 0.724028"
```

```
## Calculating CV using equation 1
h<-lm.influence(lm.fit)$h
cv.err<- mean((residuals(lm.fit)/(1-h))^2)
sprintf('%s = %3.6f', 'The LOOCV CV of equation 1', cv.err)
```

```
## [1] "The LOOCV CV of equation 1 = 0.724028"
```

From the results, we can see that two methods produce the same results which are equal to **0.724028**.

Question 2

```
train <- read.csv("HW4TrainData.csv", header=TRUE,  sep=",")

test <- read.csv("HW4TestData.csv", header=TRUE,  sep=",")

validation <- read.csv("HW4ValidationData.csv", header=TRUE,  sep=",")
```

part (a)

```
library(ggm)
```

```
## Warning: package 'ggm' was built under R version 4.0.3
```

```
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

x <- train %>% select(-y)
vars <- powerset(names(x))
mse<- c()
for (j in 1:length(vars)){
  lm_fit <- lm(y ~ ., data = train[,c("y", vars[[j]])])
   ypred <- predict(lm_fit, newdata=validation)
   ytest <- validation$y
   mse[j] <- mean((ytest-ypred)^2)
}
## best subset
best_sub<- which.min(mse)
print(best_sub)

## [1] 138

## features of the best subset
best_features<- vars[[which.min(mse)]]
print(best_features)

## [1] "x6" "x8" "x9"
```

Therefore, the features are selected by this algorithm are **x6, x8, and x9.**

```
model.A<- lm(y ~ ., data = train[,c("y", vars[[138]])])
coef(summary(model.A))[,1]

## (Intercept)          x6          x8          x9
##   12.008744 1804.463072 -502.593877   -4.597576
```

From the result, we can write the formula of Model A as following:

$$\boldsymbol{y = 12.008744 + 1804.463072\, x_6 - 502.593877\, x_8 - 4.597576\, x_9}$$

part (b)

```
library(leaps)

## Warning: package ‘leaps’ was built under R version 4.0.3
```

```
best.sub<-(regsubsets(y~., data = train, nvmax = 10))
summary(best.sub)

## Subset selection object
## Call: regsubsets.formula(y ~ ., data = train, nvmax = 10)
## 10 Variables  (and intercept)
##     Forced in Forced out
## x1      FALSE      FALSE
## x2      FALSE      FALSE
## x3      FALSE      FALSE
## x4      FALSE      FALSE
## x5      FALSE      FALSE
## x6      FALSE      FALSE
## x7      FALSE      FALSE
## x8      FALSE      FALSE
## x9      FALSE      FALSE
## x10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           x1  x2  x3  x4  x5  x6  x7  x8  x9  x10
## 1  ( 1 ) “ “ “ “ “ “ “ “ “ “ “ “ “ “ “ “ “*” “ “
## 2  ( 1 ) “ “ “ “ “ “ “ “ “*” “ “ “ “ “*” “ “ “ “
## 3  ( 1 ) “ “ “ “ “ “ “*” “*” “ “ “*” “ “ “ “ “ “
## 4  ( 1 ) “ “ “ “ “ “ “ “ “*” “ “ “*” “ “ “*” “*”
## 5  ( 1 ) “ “ “ “ “ “ “*” “*” “ “ “*” “ “ “*” “*”
## 6  ( 1 ) “ “ “*” “ “ “*” “*” “*” “*” “ “ “ “ “*”
## 7  ( 1 ) “ “ “*” “ “ “*” “*” “*” “*” “ “ “*” “*”
## 8  ( 1 ) “ “ “*” “ “ “*” “*” “*” “*” “*” “*” “*”
## 9  ( 1 ) “*” “*” “ “ “*” “*” “*” “*” “*” “*” “*”
## 10  ( 1 ) “*” “*” “*” “*” “*” “*” “*” “*” “*” “*”

## Estimating the test error
sum <- summary(best.sub)
test.error<- data.frame(
  Adj.R2 = which.max(sum$adjr2),
  CP = which.min(sum$cp),
  BIC = which.min(sum$bic)
)
print(test.error)

##   Adj.R2 CP BIC
## 1      4  4   4

 print(‘%s = %3.0f’, ‘The lowest Cp’, test.error$CP)

## [1] “The lowest Cp =   4”

## Model with the lowest Cp
model.B<- coef(best.sub,4)
print(model.B)
```

```
## (Intercept)          x5          x7          x9         x10
##    12.01057  -875.40480 -2583.76799  -826.99088 -2044.38551
```

Therefore, the best subset is **set of 4**, which includes **x5, x7, x9, and x10** features. We can write the formula of Model. B as following

$$y = 12.01057 - 875.40480\ x_5 - 2583.76799\ x_7 - 826.99088\ x_9 - 2044.38551\ x_{10}$$

part (c)

```
library(leaps)
forward <- regsubsets(y ~ ., data= train,  nvmax = 10, method = "forward")
summary(forward)
```

```
## Subset selection object
## Call: regsubsets.formula(y ~ ., data = train, nvmax = 10, method = "forward")
## 10 Variables  (and intercept)
##     Forced in Forced out
## x1      FALSE      FALSE
## x2      FALSE      FALSE
## x3      FALSE      FALSE
## x4      FALSE      FALSE
## x5      FALSE      FALSE
## x6      FALSE      FALSE
## x7      FALSE      FALSE
## x8      FALSE      FALSE
## x9      FALSE      FALSE
## x10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: forward
##           x1  x2  x3  x4  x5  x6  x7  x8  x9  x10
## 1  ( 1 )  " " " " " " " " " " " " " " " " "*" " "
## 2  ( 1 )  "*" " " " " " " " " " " " " " " "*" " "
## 3  ( 1 )  "*" " " " " " " " " "*" " " " " "*" " "
## 4  ( 1 )  "*" " " " " " " " " "*" "*" " " "*" " "
## 5  ( 1 )  "*" " " " " " " " " "*" "*" " " "*" "*"
## 6  ( 1 )  "*" " " " " " " "*" "*" "*" " " "*" "*"
## 7  ( 1 )  "*" " " " " "*" "*" "*" "*" " " "*" "*"
## 8  ( 1 )  "*" "*" " " "*" "*" "*" "*" " " "*" "*"
## 9  ( 1 )  "*" "*" " " "*" "*" "*" "*" "*" "*" "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```
## Estimating the test error
sum <- summary(forward)
test.error<- data.frame(
  Adj.R2 = which.max(sum$adjr2),
  CP = which.min(sum$cp),
  BIC = which.min(sum$bic))
print(test.error)
```

```
##   Adj.R2 CP BIC
## 1      6  6   3
```

```
sprintf('%s = %3.0f', 'The lowest Cp', test.error$CP)
```

```
## [1] "The lowest Cp =   6"
```

```
## Model with the Lowest Cp
model.C<- coef(forward,6)
print(model.C)
```

```
## (Intercept)          x1          x5          x6          x7          x9
##    12.01036  -319.31073 -1174.62217  -687.94739 -2349.99344  -530.31832
##         x10
## -2231.25816
```

Therefore, the forward stepwise selection is **set of 6** including **x1, x5, x6, x7, x9, and x10** features. We can write the formula of Model C as following

$$y = 12.01036 - 319.31073\, x_1 - 1174.62217\, x_5 - 687.94736 x_6 - \ 2349.99344\, x_7$$

$$-530.31832\, x_9 - 2231.25816\, x_{10}$$

part (d)

```
library(leaps)
backward <- regsubsets(y ~ ., data= train,  nvmax = 10, method = "backward")
summary(backward)
```

```
## Subset selection object
## Call: regsubsets.formula(y ~ ., data = train, nvmax = 10, method = "backward")
## 10 Variables  (and intercept)
##     Forced in Forced out
## x1      FALSE      FALSE
## x2      FALSE      FALSE
## x3      FALSE      FALSE
## x4      FALSE      FALSE
## x5      FALSE      FALSE
## x6      FALSE      FALSE
## x7      FALSE      FALSE
## x8      FALSE      FALSE
## x9      FALSE      FALSE
## x10     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: backward
##           x1  x2  x3  x4  x5  x6  x7  x8  x9  x10
## 1  ( 1 )  " " " " " " " " " " "*" " " " " " " " "
## 2  ( 1 )  " " " " " " " " "*" "*" " " " " " " " "
## 3  ( 1 )  " " " " " " " " "*" "*" "*" " " " " " "
## 4  ( 1 )  " " " " " " " " "*" "*" "*" " " " " "*"
## 5  ( 1 )  " " " " " " "*" "*" "*" "*" " " " " "*"
```

```
## 6  ( 1 )  " " "*" " " "*" "*" "*" "*" " " " " "*"
## 7  ( 1 )  " " "*" " " "*" "*" "*" "*" " " "*" "*"
## 8  ( 1 )  " " "*" " " "*" "*" "*" "*" "*" "*" "*"
## 9  ( 1 )  "*" "*" " " "*" "*" "*" "*" "*" "*" "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```
## Estimating the test error
res.sum <- summary(backward)
test.error<- data.frame(
  Adj.R2 = which.max(res.sum$adjr2),
  CP = which.min(res.sum$cp),
  BIC = which.min(res.sum$bic)
)
print(test.error)
```

```
##   Adj.R2 CP BIC
## 1      4  4   4
```

```
sprintf('%s = %3.0f', 'The lowest Cp', test.error$CP)
```

```
## [1] "The lowest Cp =   4"
```

```
## Model with the lowest Cp
model.D<- coef(backward,4)
print(model.D)
```

```
## (Intercept)          x5          x6          x7         x10
##    12.01015 -1026.81061 -1678.05946 -2114.17331 -1759.88784
```

Therefore, the backward stepwise selection is **set of 4** including **x5, x6, x7, and x10**. We can write the formula of Model D as following

$$\boldsymbol{y = 12.01015 - 1026.81061\, x_5 - 1678.05946\, x_6 - 2114.17331\, x_7 - 1759.88784\, x_{10}}$$

part (e)

Model A testing

```
pred <- predict(model.A, newdata = test)
model.a_mse <- (mean((test$y - pred) ^ 2))
sprintf('%s = %5.6f', 'MSE of Model A', model.a_mse)
```

```
## [1] "MSE of Model A = 0.140704"
```

Model B testing

```
test_mat = model.matrix(y ~ ., data = test)
coefs = coef(best.sub, 4)
pred = test_mat[, names(coefs)] %*% coefs
model.b_mse <- (mean((test$y - pred) ^ 2))
sprintf('%s = %5.6f', 'MSE of Model B', model.b_mse)
```

```
## [1] "MSE of Model B = 0.143801"
```

Model C testing

```
test_mat = model.matrix(y ~ ., data = test)
coefs = coef(forward, 6)
pred = test_mat[, names(coefs)] %*% coefs
model.c_mse <- (mean((test$y - pred) ^ 2))
sprintf('%s = %5.6f', 'MSE of Model C', model.c_mse)

## [1] "MSE of Model C = 0.144195"
```

Model D testing

```
test_mat = model.matrix(y ~ ., data = test)
coefs = coef(backward, 4)
pred = test_mat[, names(coefs)] %*% coefs
model.d_mse <- (mean((test$y - pred) ^ 2))
sprintf('%s = %5.6f', 'MSE of Model D', model.d_mse)

## [1] "MSE of Model D = 0.144338"
```

Comparing MSE of all four models

```
data.frame(Model = c("Model A", "Model B", "Model C", "Model D"),
           test_MSE = c(model.a_mse, model.b_mse, model.c_mse, model.d_mse)) %>%
arrange(test_MSE)

##     Model  test_MSE
## 1 Model A 0.1407035
## 2 Model B 0.1438013
## 3 Model C 0.1441947
## 4 Model D 0.1443381
```

Therefore, the model that did the best performance on the Test data is **Model A** since its **MSE** value is **0.1407035** which is the lowest one.

Question 3

```
set.seed(1)
PartialAuto <- read.csv("PartialAuto.csv", header=TRUE,  sep=",")
str(PartialAuto)

## 'data.frame':    392 obs. of  6 variables:
##  $ mpg         : num  22.4 29 19 44.3 19.4 17.5 17.6 30 20.2 23.8 ...
##  $ cylinders   : int  6 4 4 4 8 6 8 4 6 4 ...
##  $ displacement: num  231 135 122 90 318 258 302 88 200 151 ...
##  $ horsepower  : int  110 84 85 48 140 95 129 76 85 85 ...
##  $ weight      : int  3415 2525 2310 2085 3735 3193 3725 2065 2965 2855 ...
##  $ acceleration: num  15.8 16 18.5 21.7 13.2 17.8 13.4 14.5 15.8 17.6 ...

train<- PartialAuto[1:300,]
str(train)
```

```
## 'data.frame':    300 obs. of  6 variables:
##  $ mpg         : num  22.4 29 19 44.3 19.4 17.5 17.6 30 20.2 23.8 ...
##  $ cylinders   : int  6 4 4 4 8 6 8 4 6 4 ...
##  $ displacement: num  231 135 122 90 318 258 302 88 200 151 ...
##  $ horsepower  : int  110 84 85 48 140 95 129 76 85 85 ...
##  $ weight      : int  3415 2525 2310 2085 3735 3193 3725 2065 2965 2855 ...
##  $ acceleration: num  15.8 16 18.5 21.7 13.2 17.8 13.4 14.5 15.8 17.6 ...
```

```
test<- PartialAuto[301:392,]
str(test)
```

```
## 'data.frame':    92 obs. of  6 variables:
##  $ mpg         : num  22.3 32 18 13 40.8 36 16 25.5 26 20 ...
##  $ cylinders   : int  4 4 3 8 4 4 6 4 4 6 ...
##  $ displacement: num  140 71 70 318 85 98 225 122 79 198 ...
##  $ horsepower  : int  88 65 90 150 65 70 105 96 67 95 ...
##  $ weight      : int  2890 1836 2124 3755 2110 2125 3439 2300 1963 3102 ...
##  $ acceleration: num  17.3 21 13.5 14 19.2 17.3 15.5 15.5 15.5 16.5 ...
```

part (a)

```
set.seed(1)
## fitting a linear regression model
lm.fit<- lm(mpg~., data = train)
summary(lm.fit)
```

```
## 
## Call:
## lm(formula = mpg ~ ., data = train)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -9.6998 -2.7582 -0.3938  2.3470 16.0336 
## 
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  47.6291075  3.0050174  15.850  < 2e-16 ***
## cylinders    -0.7005520  0.4707578  -1.488  0.13779    
## displacement  0.0063256  0.0101679   0.622  0.53435    
## horsepower   -0.0518382  0.0192443  -2.694  0.00747 ** 
## weight       -0.0052159  0.0009663  -5.398 1.39e-07 ***
## acceleration -0.0239893  0.1449448  -0.166  0.86866    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 4.357 on 294 degrees of freedom
## Multiple R-squared:  0.7073, Adjusted R-squared:  0.7023 
## F-statistic: 142.1 on 5 and 294 DF,  p-value: < 2.2e-16
```

```
## Reporting the test MSE
y_pred <- predict(lm.fit, test)
y_actual<- test$mpg
```

```r
lm_mse <- (mean((y_actual - y_pred) ^ 2))
sprintf('%s = %5.6f', 'The test MSE', lm_mse)

## [1] "The test MSE = 15.497936"
```

From the result, the test MSE is 15.4979.

part (b)

```r
set.seed(1)
library(glmnet)

## Warning: package 'glmnet' was built under R version 4.0.3

## Loading required package: Matrix

## Loaded glmnet 4.1

x <- model.matrix(mpg~., train)[,-1]
y<- train$mpg

## Using cross validation to choose the best lambda
grid <- 10^seq(3,-5, length = 1000)
cv<- cv.glmnet(x,y, alpha = 0, lambda = grid)
bestlam<- cv$lambda.min
sprintf('%s = %5.6f', 'The best lambda obtained from cross validation', bestlam)

## [1] "The best lambda obtained from cross validation = 0.188919"

## Fitting a Ridge regression model on the train data with lambda chosen by cross-
validation
ridge.mod <- glmnet(x,y, alpha = 0, lambda = bestlam)
coef(ridge.mod)

## 6 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  46.692863673
## cylinders    -0.644024611
## displacement -0.001356226
## horsepower   -0.050149274
## weight       -0.004433257
## acceleration -0.048988022

## Evaluating MSE on the test data
x_test<- model.matrix(mpg~., test)[,-1]
y_pred <- predict(ridge.mod,  newx = x_test)
y_actual<- test$mpg
ridge_mse <- (mean((y_actual - y_pred) ^ 2))
sprintf('%s = %5.4f', 'The test MSE', ridge_mse)

## [1] "The test MSE = 15.3658"
```

From the result, the MSE obtained on the test data is 15.3658.

part (c)

```r
set.seed(1)
library(glmnet)
x <- model.matrix(mpg~., train)[,-1]
y<- train$mpg

## Using cross validation to choose the best lambda
grid <- 10^seq(3,-5, length = 1000)
cv<- cv.glmnet(x,y, alpha = 1, lambda = grid)
bestlam<- cv$lambda.min
sprintf('%s = %5.6f', 'The best lambda obtained from cross validation', bestlam)

## [1] "The best lambda obtained from cross validation = 0.112733"

## Fitting a Lasso regression model on the train data with lambda chosen by cross-
validation
lasso.mod <- glmnet(x,y, alpha = 1, lambda = bestlam)
coef(lasso.mod)

## 6 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  45.930610381
## cylinders    -0.472315177
## displacement  .
## horsepower   -0.044149853
## weight       -0.005047985
## acceleration  .

## Evaluating MSE on the test data
x_test<- model.matrix(mpg~., test)[,-1]
y_pred <- predict(lasso.mod, newx = x_test)
y_actual<- test$mpg
lasso_mse <- (mean((y_actual - y_pred)^ 2))
sprintf('%s = %5.4f', 'The test MSE', lasso_mse)

## [1] "The test MSE = 15.2049"
```

From the result, the MSE obtained on the test data of Lasso model is 15.2049.

```r
library(dplyr)
data.frame(Models = c("Linear reg (part a)", "Ridge(part b)", "Lasso(part c)"),
           test_MSE = c(lm_mse, ridge_mse, lasso_mse)) %>% arrange(test_MSE)

##                Models test_MSE
## 1       Lasso(part c) 15.20492
## 2       Ridge(part b) 15.36582
## 3 Linear reg (part a) 15.49794
```

From the table above, we can conclude that the **model in part (c)** which is **Lasso model** seems to perform the best since it has **the lowest MSE which is equal to 15.205.**