Linear Models and Regularization Methods

Franky Zhang

2/7/2022

6.2

For parts (a) through (c), indicate which of i. through iv. is correct. Justify your answer.

(a)

The lasso, relative to least squares, is:

i. More flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.
ii. More flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.
iii. Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.
iv. Less flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

Answer:

(iii) is right. Lasso regression add penalty term which results to reduction of predictors' parameters even down to zero. And this causes the decrease of flexibility and variance. However, due to the bias-variance trade-off, the bias increases.

(b) Repeat (a) for ridge regression relative to least squares.

Answer:

Ridge regression is similar to Lasso, thus choose (iii). the only difference is Ridge regression will not result in remove any predictors.

(c) Repeat (a) for non-linear methods relative to least squares.

Answer:

choose (ii), while non-linear regression allows quadratic terms, cubic terms and so on, its flexibility raises and bias decreases. .

6.9

(a)

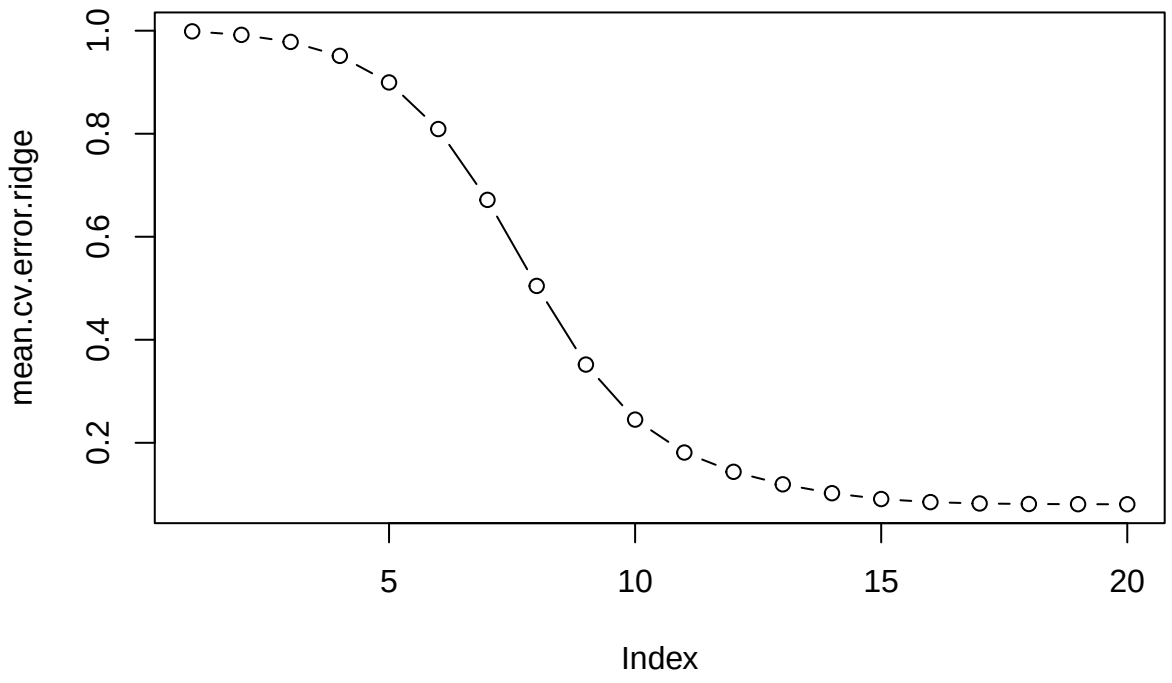
```
# College
set.seed(2)
full_set <- College
full_set$Private <- as.numeric(full_set$Private)
# data centering and scaling
for (i in 1:dim(full_set)[2]) {
  full_set[, i] <- scale(full_set[, i], center = TRUE, scale = TRUE)
}
train <- sample(c(TRUE, FALSE), nrow(College), replace = TRUE, prob = c(.8, .2))
training_set <- full_set[train, ]
test_set <- full_set[!train, ]
```

(b)

```
## [1] 0.0556268
```

test error of linear regression is *0.056*.

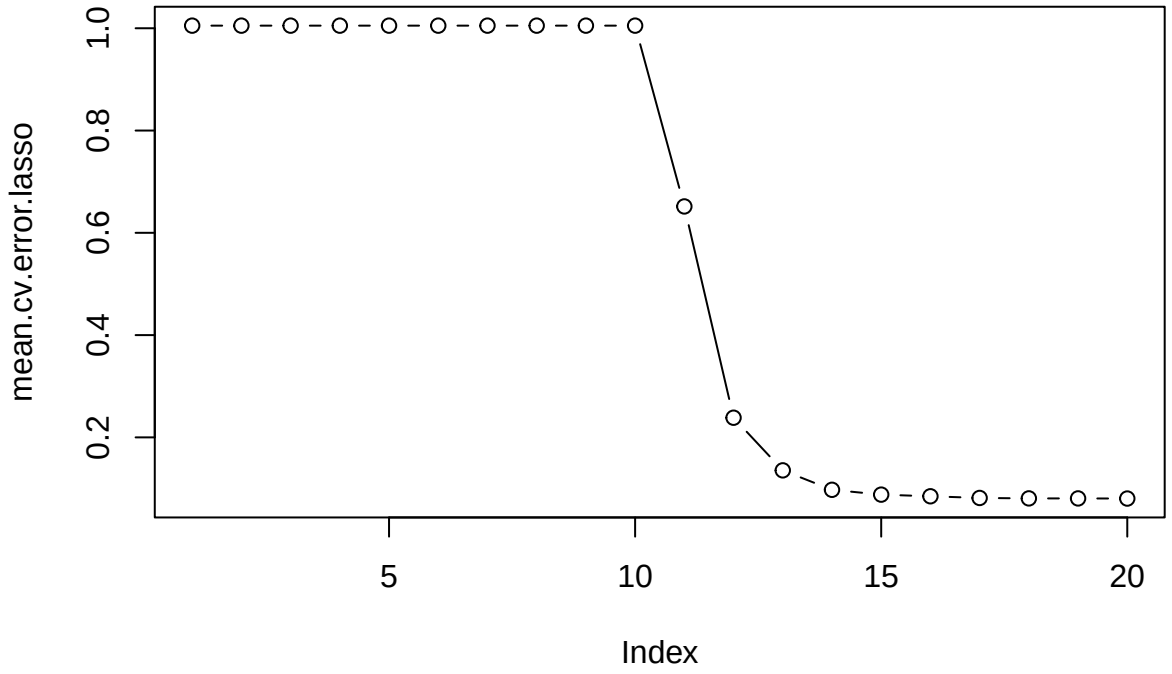
(c)




```
## lambda = 0.001
##      0.08066928
```

the mean test error of Ridge regression is *0.084*.

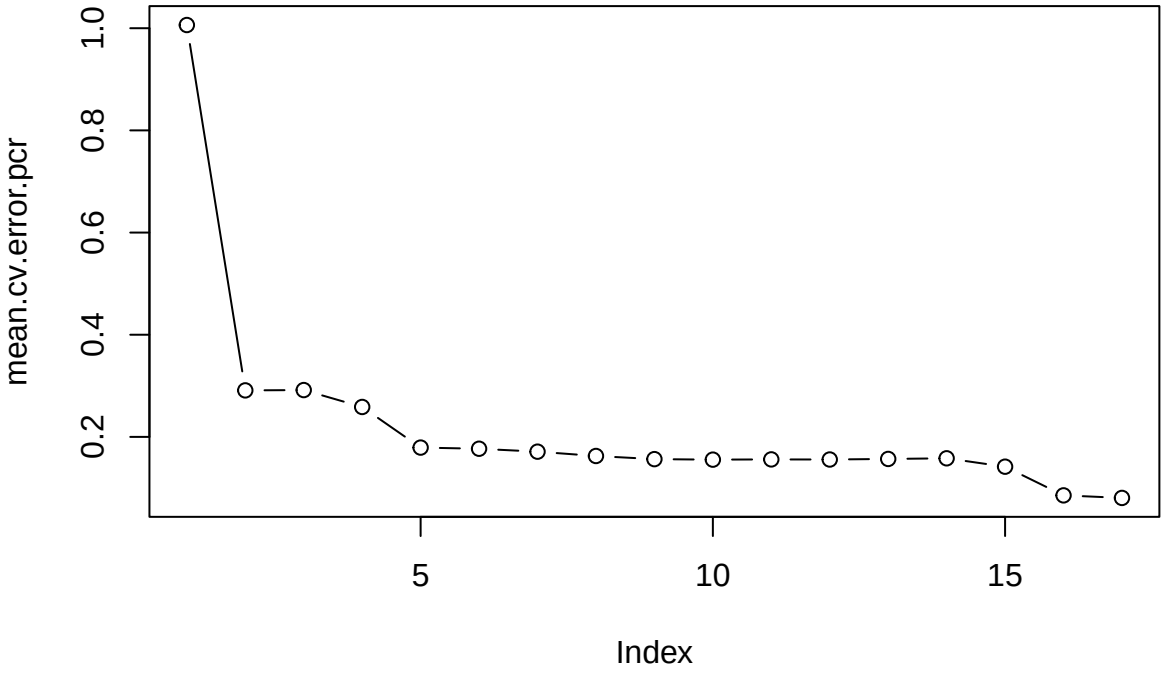
(d)




```
## lambda = 0.001
##      0.08041198
```

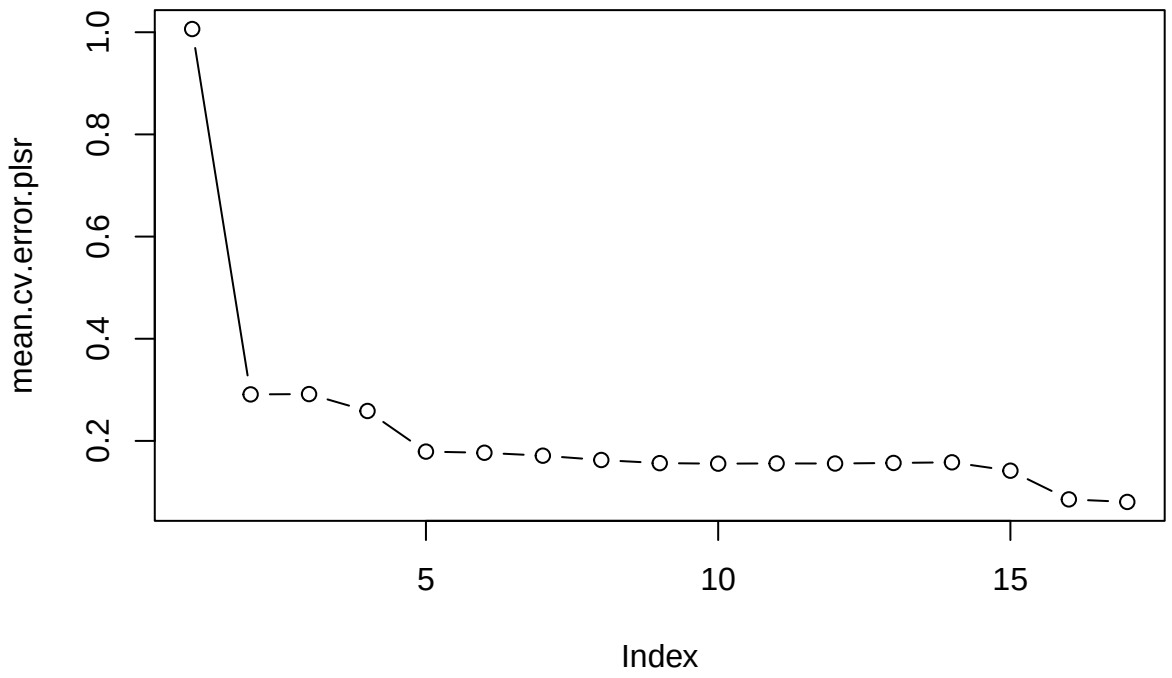
the mean test error of Lasso regression is *0.084*.

(e)



```
##   ncomp 1    ncomp 2    ncomp 3    ncomp 4    ncomp 5    ncomp 6    ncomp 7
## 1.00625687 0.29106085 0.29164830 0.25856127 0.17907407 0.17680360 0.17106198
##   ncomp 8    ncomp 9   ncomp 10   ncomp 11   ncomp 12   ncomp 13   ncomp 14
## 0.16261399 0.15646245 0.15546253 0.15592322 0.15571932 0.15683340 0.15809525
##  ncomp 15   ncomp 16   ncomp 17
## 0.14167029 0.08553699 0.08052947
```

(f)




```
##   ncomp 1    ncomp 2    ncomp 3    ncomp 4    ncomp 5    ncomp 6    ncomp 7
## 1.00625687 0.29106085 0.29164830 0.25856127 0.17907407 0.17680360 0.17106198
##   ncomp 8    ncomp 9   ncomp 10   ncomp 11   ncomp 12   ncomp 13   ncomp 14
## 0.16261399 0.15646245 0.15546253 0.15592322 0.15571932 0.15683340 0.15809525
##  ncomp 15   ncomp 16   ncomp 17
## 0.14167029 0.08553699 0.08052947
```

(g)

the test error rate is between 0.05 ~ 0.08 on scaled data. There's not much difference between these approaches.

6.10

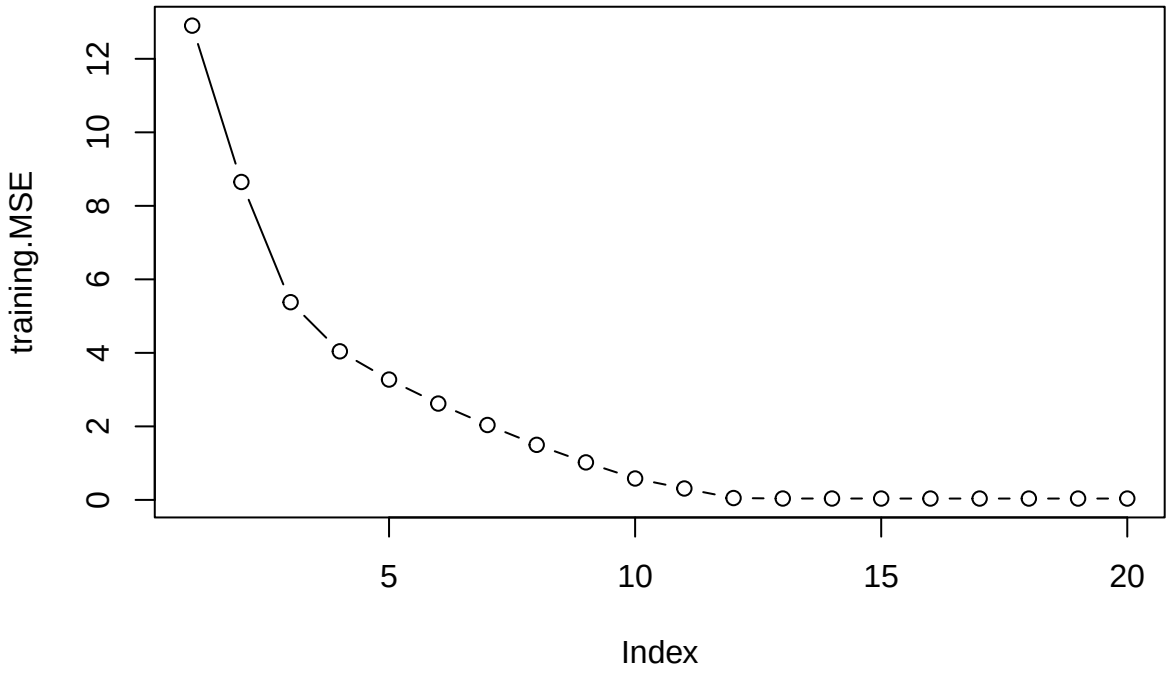
(a)

```
set.seed(2007)
df <- data.frame(replicate(20, rnorm(n = 1000)))
# select some of variables exactly equal to 0, let the rate to be .4
set.seed(6212)
index <- sample(c(TRUE, FALSE), dim(df)[2], replace = TRUE, prob = c(.6, .4))
# sum(index)
# select 13 variables
beta <- rep(NA, dim(df)[2])
beta[!index] <- 0
beta[index] <- replicate(sum(index), rnorm(1, mean = rbinom(1, 1, prob = .5)))
# beta
epsilon <- rnorm(n = 1000, mean = 0, sd <- .2)
y <- rep(0, 1000)
for (i in 1:1000) {
  for (j in 1:20) {
    y[i] <- df[i, j]*beta[j] + y[i]
  }
}
# add white noises
y <- y + epsilon
```

(b)

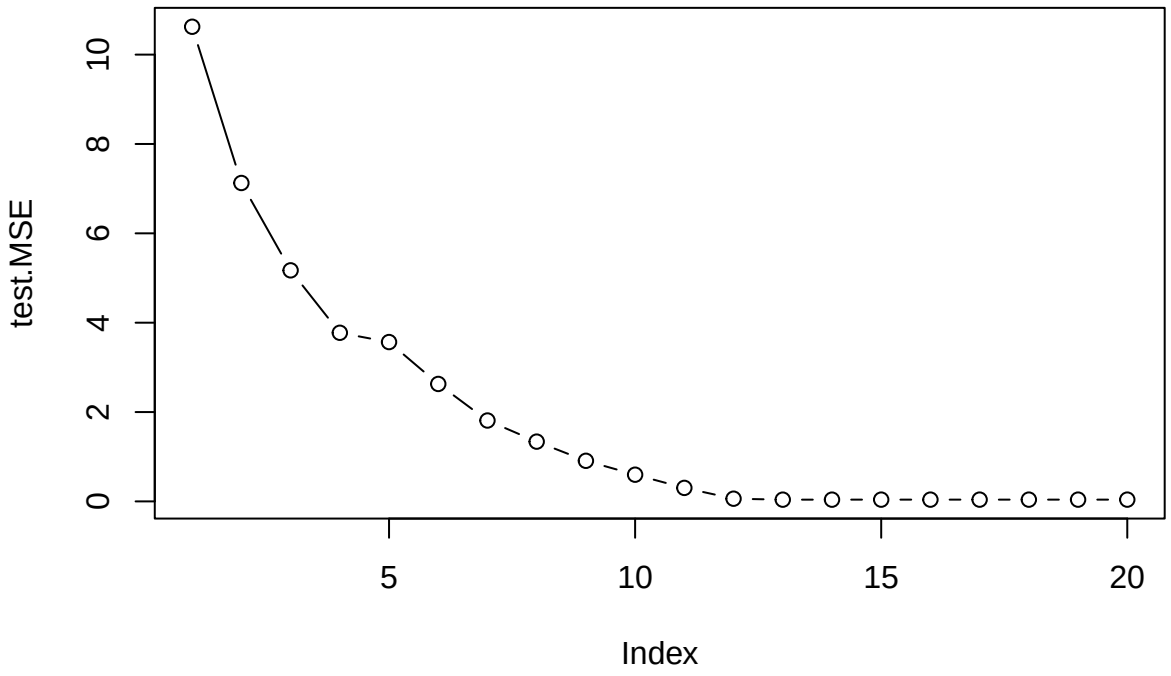
```
train <- sample(c(1: 1000), 900, replace = FALSE)
y_train <- y[train]
x_train <- df[train, ]
y_test <- y[-train]
x_test <- df[-train, ]

training <- cbind(y = y_train, x_train)
test <- cbind(y = y_test, x_test)
```

(c)


training.MSE
0
2
4
6
8
10
12
5
10
15
20
Index

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

(d)


test.MSE
0
2
4
6
8
10
5
10
15
20
Index

```
## [1] 13
```

(e)

Answer:

When the number of varibles equals to 13, the test MSE reaches it minimum, which illustrate that including all predictors performs worse on test data.

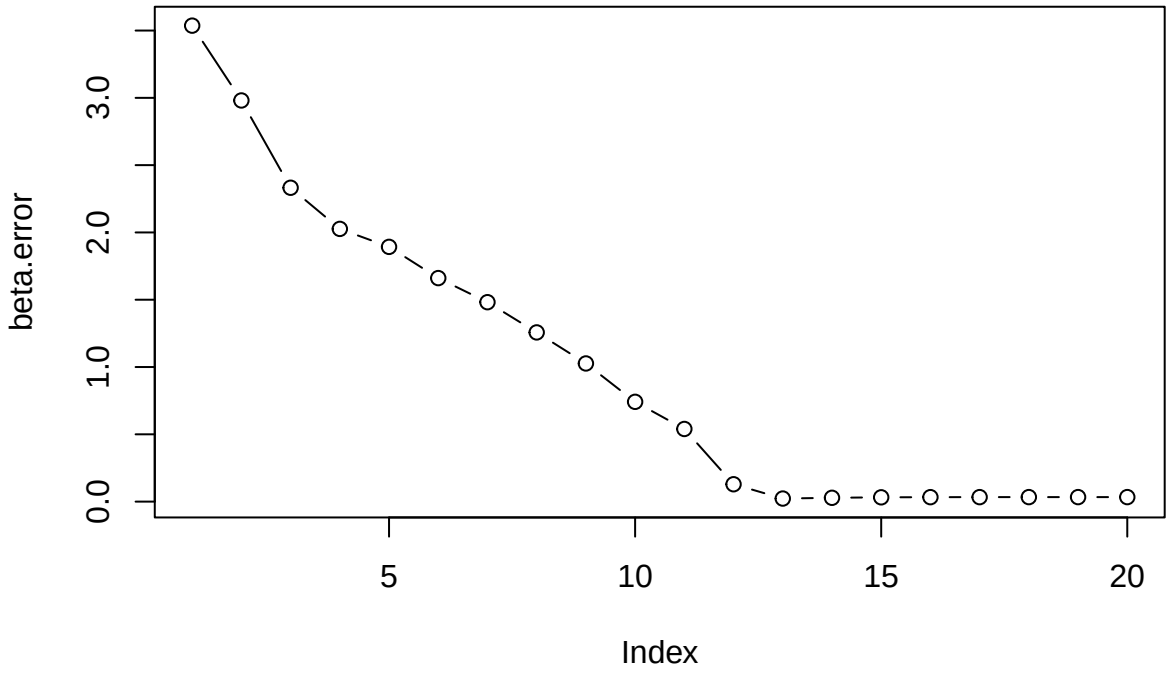
(f)

```
##      var.names     actual Estimation_13
## 1  (Intercept)  0.0000000  -0.008976285
## 2           X1  0.0000000            NA
## 3           X2  2.5910606   2.601353439
## 4           X3 -0.1259393  -0.115866937
## 5           X4 -0.7817413  -0.782639284
## 6           X5 -0.5045241  -0.514731327
## 7           X6 -1.9071117  -1.899527747
## 8           X7  0.7468964   0.746798126
## 9           X8  0.0000000            NA
## 10          X9  0.0000000            NA
## 11         X10  0.0000000            NA
## 12         X11 -0.9071135  -0.906914148
## 13         X12  0.5199715   0.519273519
## 14         X13  0.0000000            NA
## 15         X14  0.0000000            NA
## 16         X15  0.0000000            NA
## 17         X16  0.7224114   0.727166656
## 18         X17  0.7092484   0.704347771
## 19         X18  1.1534229   1.154421375
## 20         X19 -0.7377347  -0.740171146
## 21         X20  1.8522093   1.856445271
```

Answer:

The result is exactly constant with the true model generating data.

(g)




```
## [1] 13
```

Answer:

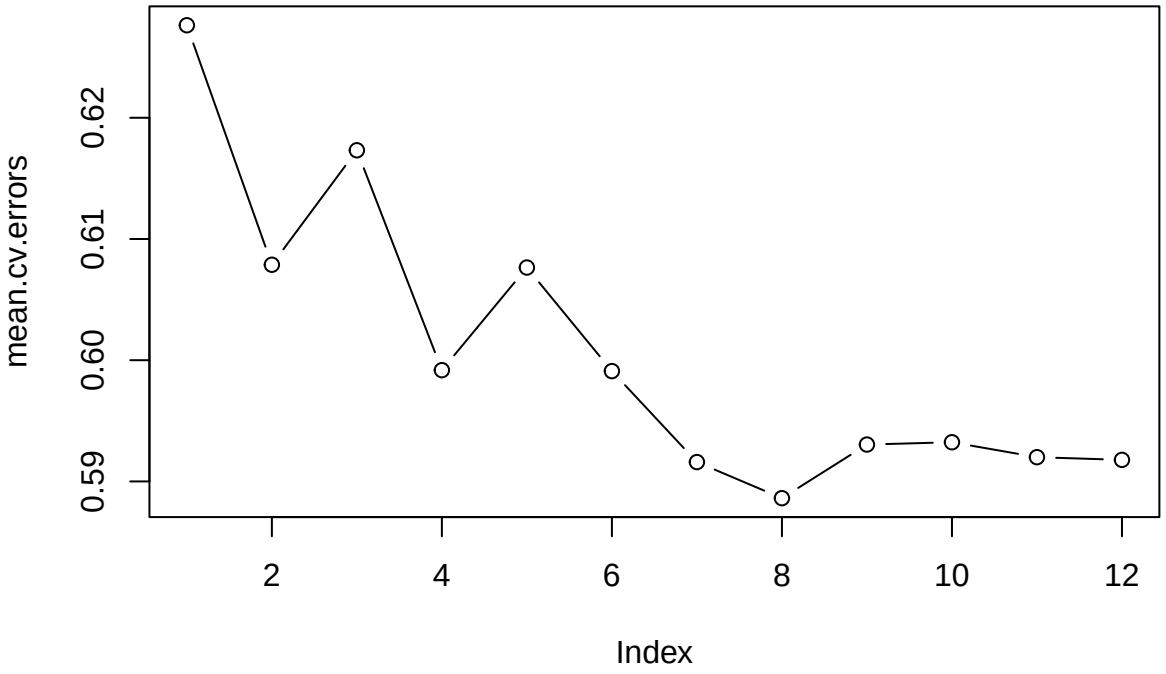
This beta error plot is constant with test MSE.

6.11

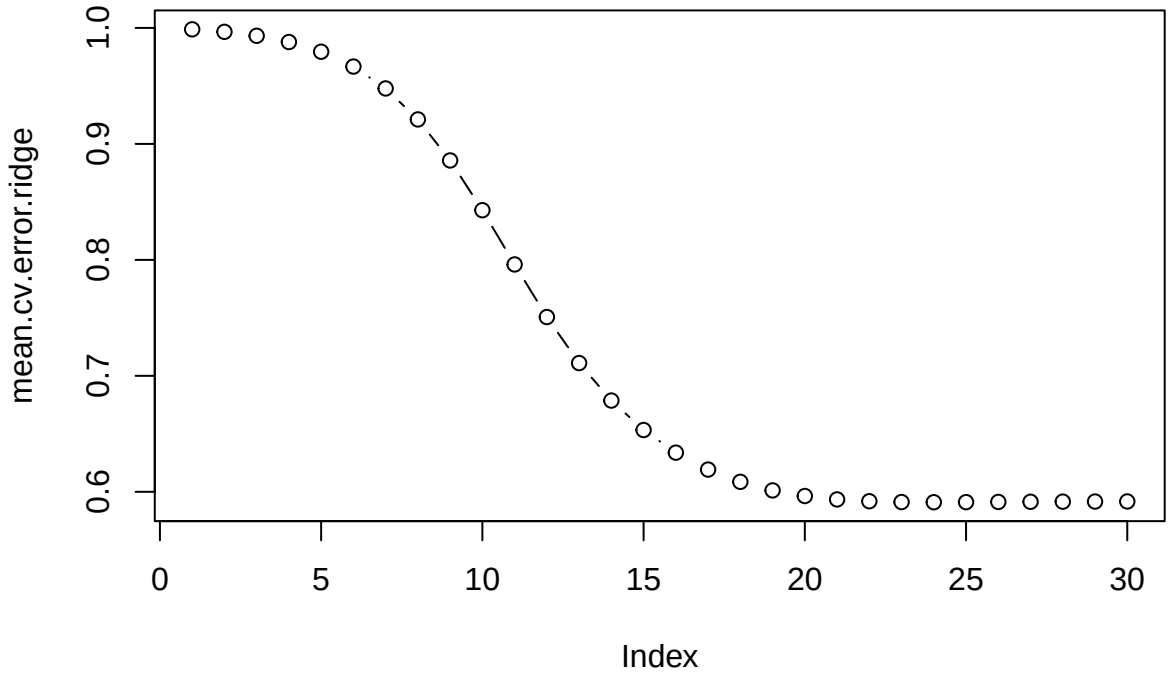
(a)

Baseline: linear regression

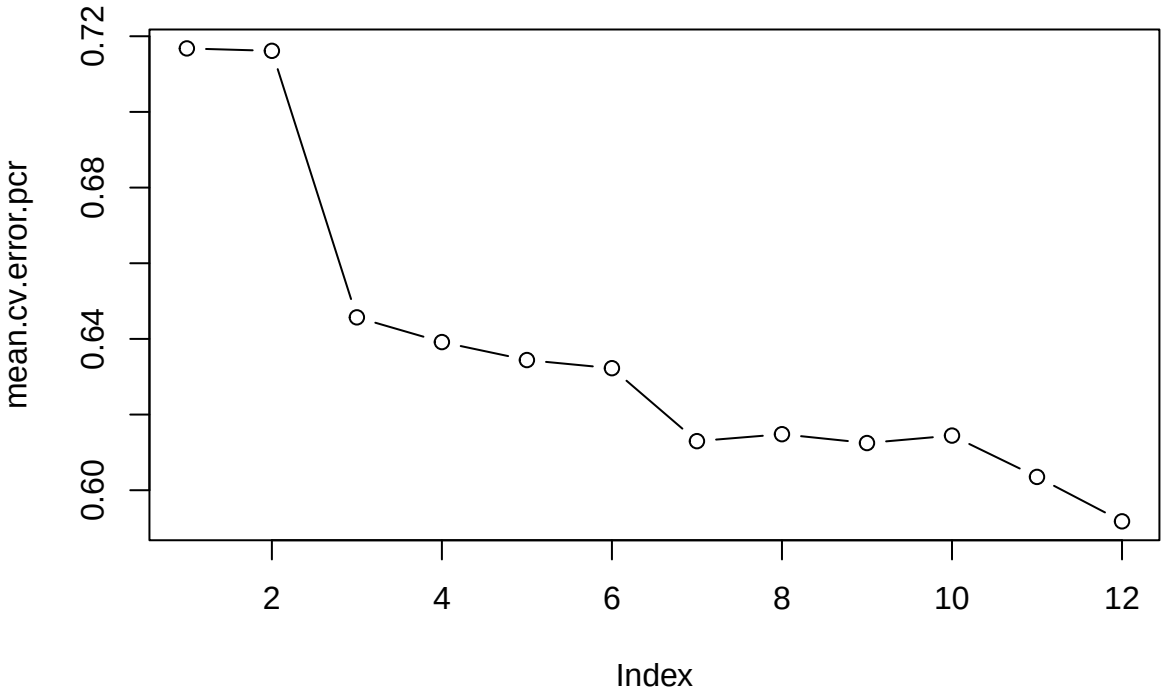
best subset selection(Cross Validation) + linear regression



Ridge regression (Cross Validation)



PCR (Cross Validation)



Discussion:

the test MSE for simple linear regression is 1.12, which acts as the baseline. The preferred number of best subset selection is 8, and the best model composed by 8 variable gives a test MSE = 1.05, slightly better than our baseline(1.12). Then, the Ridge regression performs best when $\lambda = 0.591$ and the best model offered by this model has test MSE = 1.24. Eventally, PCR prefer the model with 12 predictors, which is exactly the same as simple linear regression.

(b)

Answer: Prefer Best Subset Selection.

(c)

Answer: I won't chose model involve all of the features in the data set. Since Best Subset Selection tells me that the model with 8 predictors performs better on test data and the fewer number of predictors means better and easier to interpret. Additionally, it is obvious that the robustness of model with fewer predictors is better. In all, I prefer the model with 8 predictors.

5.8

(a)

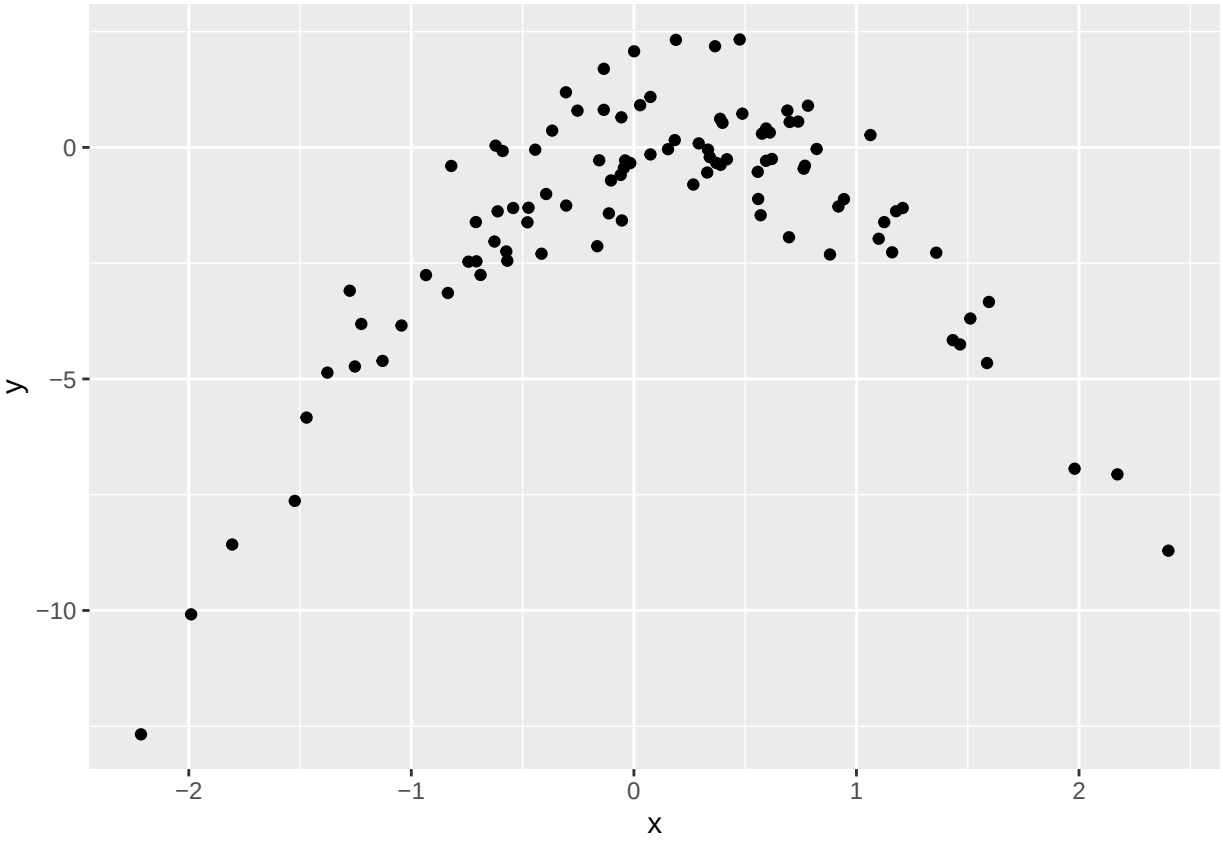
```
set.seed(1)
x <- rnorm(100)
y <- x - 2 * x^2 + rnorm(100)
```

Answer:

n = 100, p = 1, the equation is as followed:

$$y_i = x_i - 2 \times x_i^2 + \epsilon_i$$

(b)



Comment:

Y has strong relationship with quadratic form of X.

(c)

```
data <- data.frame(x, y)
set.seed(2247)
fit.1 <- glm(y ~ x, data = data)
cv.glm(data, fit.1)$delta[1]
```

```
## [1] 7.288162
```

```
fit.2 <- glm(y ~ poly(x, 2), data = data)
cv.glm(data, fit.2)$delta[1]
```

```
## [1] 0.9374236
```

```r
fit.3 <- glm(y ~ poly(x, 3), data = data)
cv.glm(data, fit.3)$delta[1]
```

```
## [1] 0.9566218
```

```r
fit.4 <- glm(y ~ poly(x, 4), data = data)
cv.glm(data, fit.4)$delta[1]
```

```
## [1] 0.9539049
```

(d)

Answer:

the result must be the same because whatever the order, LOOCV calculate n times where each observation acts as the validation and take its mean. this process is not affected by the order or the randomm seed.

(e)

Answer:

model (ii) has the smallest LOOCV error, it is exactly what we expected. This due to the way of data generation.

(f)

Answer:

```r
summary(fit.1)
```

```
##
## Call:
## glm(formula = y ~ x, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -9.5161  -0.6800   0.6812   1.5491   3.8183
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.6254     0.2619  -6.205 1.31e-08 ***
## x             0.6925     0.2909   2.380   0.0192 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 6.760719)
##
##     Null deviance: 700.85  on 99  degrees of freedom
## Residual deviance: 662.55  on 98  degrees of freedom
## AIC: 478.88
##
## Number of Fisher Scoring iterations: 2
```

```
summary(fit.2)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 2), data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9650  -0.6254  -0.1288   0.5803   2.2700
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.5500     0.0958  -16.18  < 2e-16 ***
## poly(x, 2)1   6.1888     0.9580    6.46 4.18e-09 ***
## poly(x, 2)2 -23.9483     0.9580  -25.00  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9178258)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  89.029  on 97  degrees of freedom
## AIC: 280.17
##
## Number of Fisher Scoring iterations: 2
```

```
summary(fit.3)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 3), data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9765  -0.6302  -0.1227   0.5545   2.2843
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.55002    0.09626 -16.102  < 2e-16 ***
## poly(x, 3)1   6.18883    0.96263   6.429 4.97e-09 ***
## poly(x, 3)2 -23.94830    0.96263 -24.878  < 2e-16 ***
## poly(x, 3)3   0.26411    0.96263   0.274    0.784
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9266599)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  88.959  on 96  degrees of freedom
## AIC: 282.09
##
## Number of Fisher Scoring iterations: 2
```

```
summary(fit.4)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4), data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0550  -0.6212  -0.1567   0.5952   2.2267
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.55002    0.09591 -16.162  < 2e-16 ***
## poly(x, 4)1   6.18883    0.95905   6.453 4.59e-09 ***
## poly(x, 4)2 -23.94830    0.95905 -24.971  < 2e-16 ***
## poly(x, 4)3   0.26411    0.95905   0.275    0.784
## poly(x, 4)4   1.25710    0.95905   1.311    0.193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9197797)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  87.379  on 95  degrees of freedom
## AIC: 282.3
##
## Number of Fisher Scoring iterations: 2
```

Comment:

whatever the number of polynomials, only x and qudratic x show statistial significance, which indicates that it it better to include only these two terms. this agrees the conclusion draws from LOOCV!