Project 1

Part 1 – Regression Analysis on Health Dataset

The health dataset consists of 5 variables (4 predictor variables and 1 response variable) and 53 observations. Below is the data description for the dataset –

X1 = death rate per 1000 residents

X2 = doctor availability per 100,000 residents

X3 = hospital availability per 100,000 residents

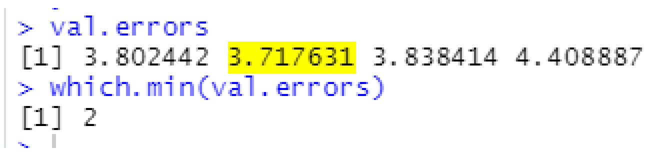
X4 = annual per capita income in thousands of dollars

X5 = population density people per square mile

Out of the above variables, X1 is the response variable and X2 to X5 are the predictor variables.

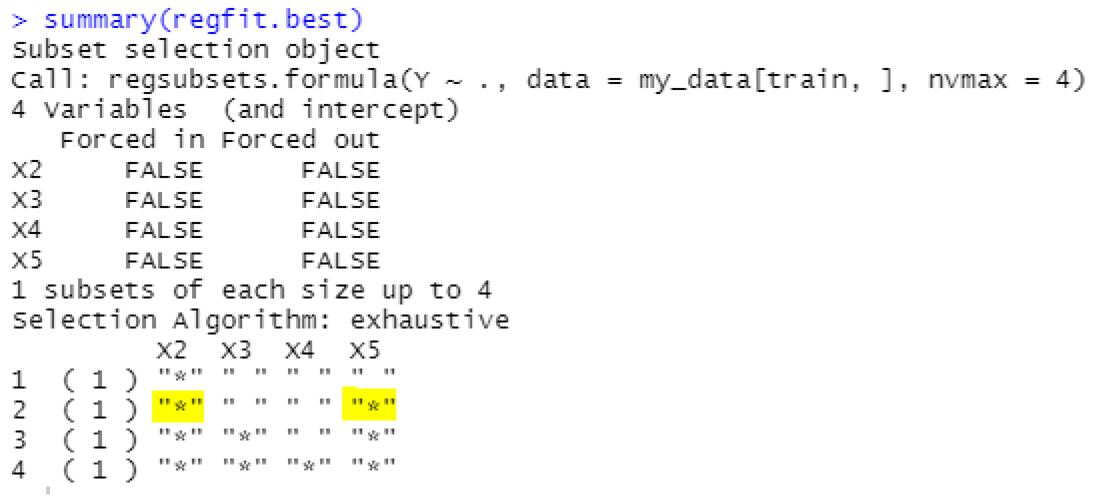
Subset and Model Selection

We use validation set approach and Cross-Validation for selecting the best subset and the model with least error. We see that the two variable model gives us the least MSE as **3.717631**



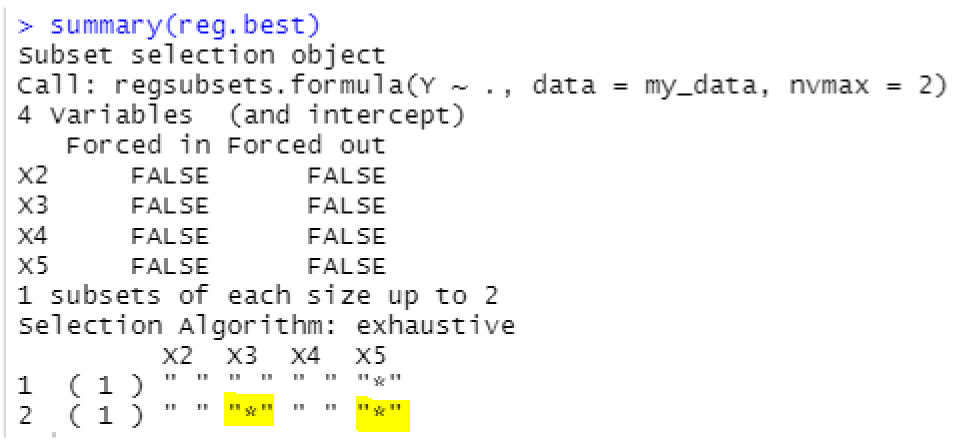
```
> val.errors
[1] 3.802442 3.717631 3.838414 4.408887
> which.min(val.errors)
[1] 2
```

We select the two variables as X2 and X5 as below –



```
> summary(regfit.best)
Subset selection object
Call: regsubsets.formula(Y ~ ., data = my_data[train, ], nvmax = 4)
4 Variables  (and intercept)
   Forced in Forced out
X2     FALSE      FALSE
X3     FALSE      FALSE
X4     FALSE      FALSE
X5     FALSE      FALSE
1 subsets of each size up to 4
Selection Algorithm: exhaustive
         X2  X3  X4  X5 
1  ( 1 ) "*" " " " " " "
2  ( 1 ) "*" " " " " "*"
3  ( 1 ) "*" "*" " " "*"
4  ( 1 ) "*" "*" "*" "*"
```

Now, when we use Cross-Validation method to select the best subset, we again see that it selects the two variable model to produce the least MSE as **2.618851** but selects X3 an X5 as the two variables.

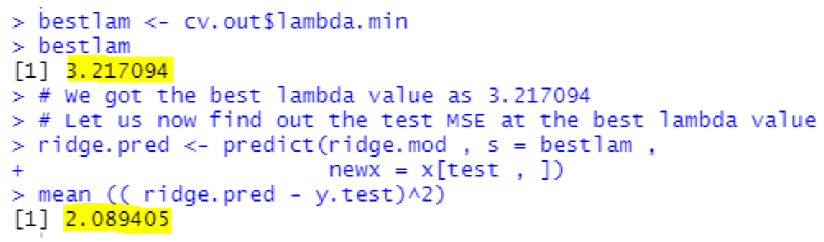


```
> summary(reg.best)
Subset selection object
Call: regsubsets.formula(Y ~ ., data = my_data, nvmax = 2)
4 Variables  (and intercept)
   Forced in Forced out
X2     FALSE      FALSE
X3     FALSE      FALSE
X4     FALSE      FALSE
X5     FALSE      FALSE
1 subsets of each size up to 2
Selection Algorithm: exhaustive
         X2  X3  X4  X5 
1  ( 1 ) " " " " " " "*"
2  ( 1 ) " " "*" " " "*"
```

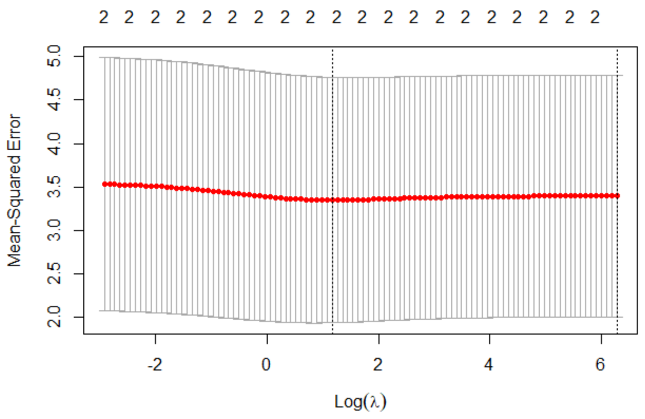
Regression Analysis

Ridge Regression

Using Cross-Validation, we find the best λ value as **3.217094** and test MSE at best λ value is **2.089405**

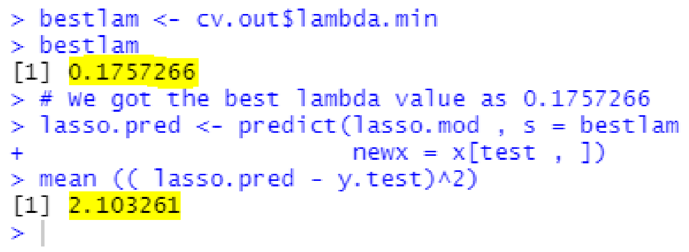


```
> bestlam <- cv.out$lambda.min
> bestlam
[1] 3.217094
> # We got the best lambda value as 3.217094
> # Let us now find out the test MSE at the best lambda value
> ridge.pred <- predict(ridge.mod , s = bestlam ,
+                       newx = x[test , ])
> mean (( ridge.pred - y.test)^2)
[1] 2.089405
```

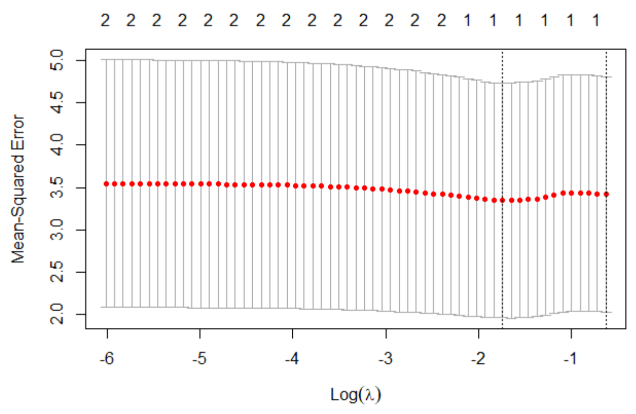


Lasso Regression

Using Cross-Validation, we find the best λ value as **0.1757266** and test MSE at best λ value is **2.103261**

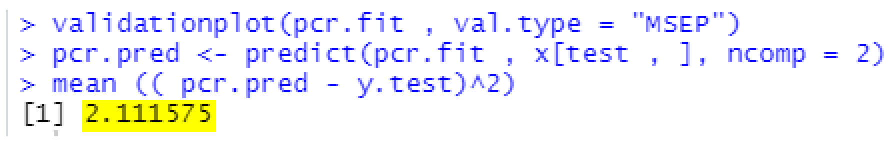


```
> bestlam <- cv.out$lambda.min
> bestlam
[1] 0.1757266
> # We got the best lambda value as 0.1757266
> lasso.pred <- predict(lasso.mod , s = bestlam
+                       newx = x[test , ])
> mean (( lasso.pred - y.test)^2)
[1] 2.103261
> 
```



Principal Component Regression

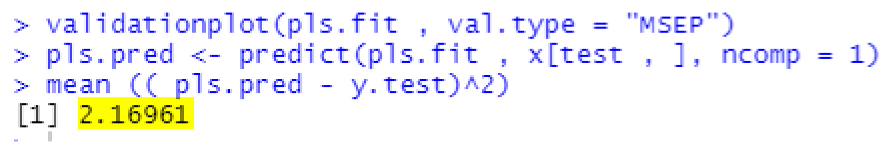
The test MSE obtained for the dataset using PCR is **2.111575**



```
> validationplot(pcr.fit , val.type = "MSEP")
> pcr.pred <- predict(pcr.fit , x[test , ], ncomp = 2)
> mean (( pcr.pred - y.test)^2)
[1] 2.111575
```

Partial Least Squares Regression

The test MSE obtained using Partial Least Squares Regression is **2.16961**

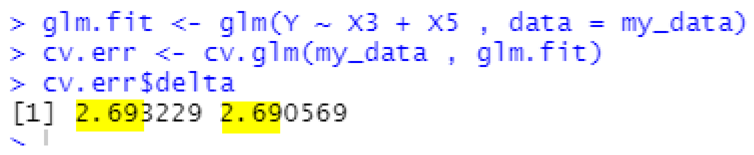


```
> validationplot(pls.fit , val.type = "MSEP")
> pls.pred <- predict(pls.fit , x[test , ], ncomp = 1)
> mean (( pls.pred - y.test)^2)
[1] 2.16961
```

Cross – Validation

Leave One Out Cross Validation (LOOCV)

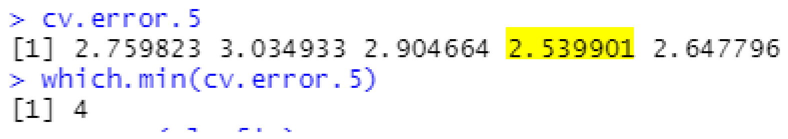
We obtained test MSE as **2.69** by using LOOCV



```
> glm.fit <- glm(Y ~ X3 + X5 , data = my_data)
> cv.err <- cv.glm(my_data , glm.fit)
> cv.err$delta
[1] 2.693229 2.690569
```

5-Fold Cross Validation

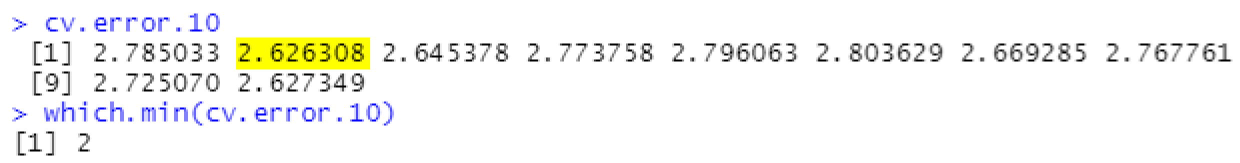
By using 5-Fold CV, we observe that the test MSE obtained is **2.539901**



```
> cv.error.5
[1] 2.759823 3.034933 2.904664 2.539901 2.647796
> which.min(cv.error.5)
[1] 4
```

10-Fold Cross Validation

By using 10-Fold CV, we observe that the test MSE obtained is **2.626308**, which is little higher than that of 5-Fold CV.



```
> cv.error.10
 [1] 2.785033 2.626308 2.645378 2.773758 2.796063 2.803629 2.669285 2.767761
 [9] 2.725070 2.627349
> which.min(cv.error.10)
[1] 2
```

Part 2 - Combined cycle power plant dataset

The dataset contains 9568 data points collected from a Combined Cycle Power Plant over 6 years (2006-2011), when the power plant was set to work with full load. Features consist of hourly average ambient variables Temperature (T), Ambient Pressure (AP), Relative Humidity (RH) and Exhaust Vacuum (V) to predict the net hourly electrical energy output (PE) of the plant.

A combined cycle power plant (CCPP) is composed of gas turbines (GT), steam turbines (ST) and heat recovery steam generators. In a CCPP, the electricity is generated by gas and steam turbines, which are combined in one cycle, and is transferred from one turbine to another. While the Vacuum is collected from and has effect on the Steam Turbine, the other three of the ambient variables effect the GT performance.

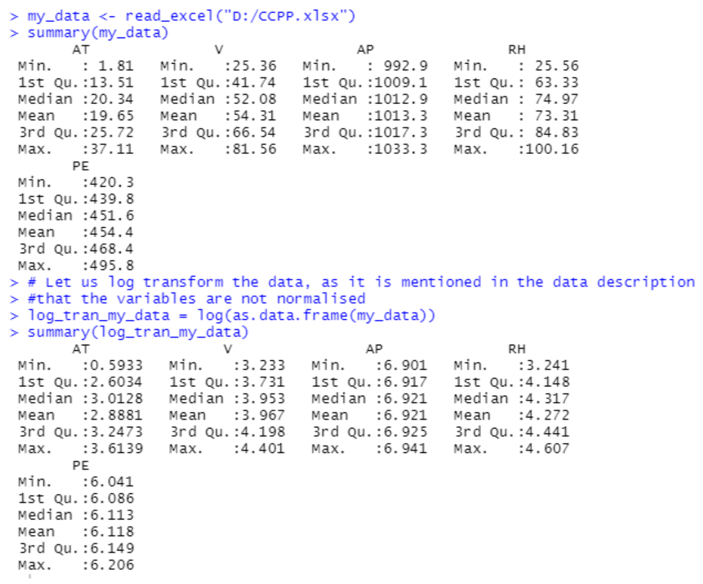
Attribute Information:

Features consist of hourly average ambient variables

- Temperature (T) in the range 1.81°C and 37.11°C,
- Ambient Pressure (AP) in the range 992.89-1033.30 millibar,
- Relative Humidity (RH) in the range 25.56% to 100.16%
- Exhaust Vacuum (V) in the range 25.36-81.56 cm Hg
- Net hourly electrical energy output (EP) 420.26-495.76 MW

The averages are taken from various sensors located around the plant that record the ambient variables every second. The variables are given without normalization.

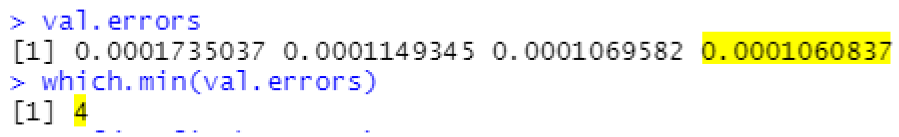
Since, the variables are not scaled properly in the dataset, let us first apply log transform to normalize them as below –



```
> my_data <- read_excel("D:/CCPP.xlsx")
> summary(my_data)
       AT               V               AP               RH        
 Min.   : 1.81   Min.   :25.36   Min.   : 992.9   Min.   : 25.56  
 1st Qu.:13.51   1st Qu.:41.74   1st Qu.:1009.1   1st Qu.: 63.33  
 Median :20.34   Median :52.08   Median :1012.9   Median : 74.97  
 Mean   :19.65   Mean   :54.31   Mean   :1013.3   Mean   : 73.31  
 3rd Qu.:25.72   3rd Qu.:66.54   3rd Qu.:1017.3   3rd Qu.: 84.83  
 Max.   :37.11   Max.   :81.56   Max.   :1033.3   Max.   :100.16  
       PE       
 Min.   :420.3  
 1st Qu.:439.8  
 Median :451.6  
 Mean   :454.4  
 3rd Qu.:468.4  
 Max.   :495.8  
> # Let us log transform the data, as it is mentioned in the data description
> #that the variables are not normalised
> log_tran_my_data = log(as.data.frame(my_data))
> summary(log_tran_my_data)
       AT                V               AP              RH       
 Min.   :0.5933   Min.   :3.233   Min.   :6.901   Min.   :3.241  
 1st Qu.:2.6034   1st Qu.:3.731   1st Qu.:6.917   1st Qu.:4.148  
 Median :3.0128   Median :3.953   Median :6.921   Median :4.317  
 Mean   :2.8881   Mean   :3.967   Mean   :6.921   Mean   :4.272  
 3rd Qu.:3.2473   3rd Qu.:4.198   3rd Qu.:6.925   3rd Qu.:4.441  
 Max.   :3.6139   Max.   :4.401   Max.   :6.941   Max.   :4.607  
       PE       
 Min.   :6.041  
 1st Qu.:6.086  
 Median :6.113  
 Mean   :6.118  
 3rd Qu.:6.149  
 Max.   :6.206  
```

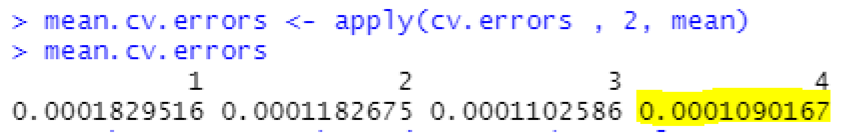
Subset and Model Selection

We use validation set approach and Cross-Validation for selecting the best subset and the model with least error. We see that the four variable model gives us the least MSE as **0.0001060837**

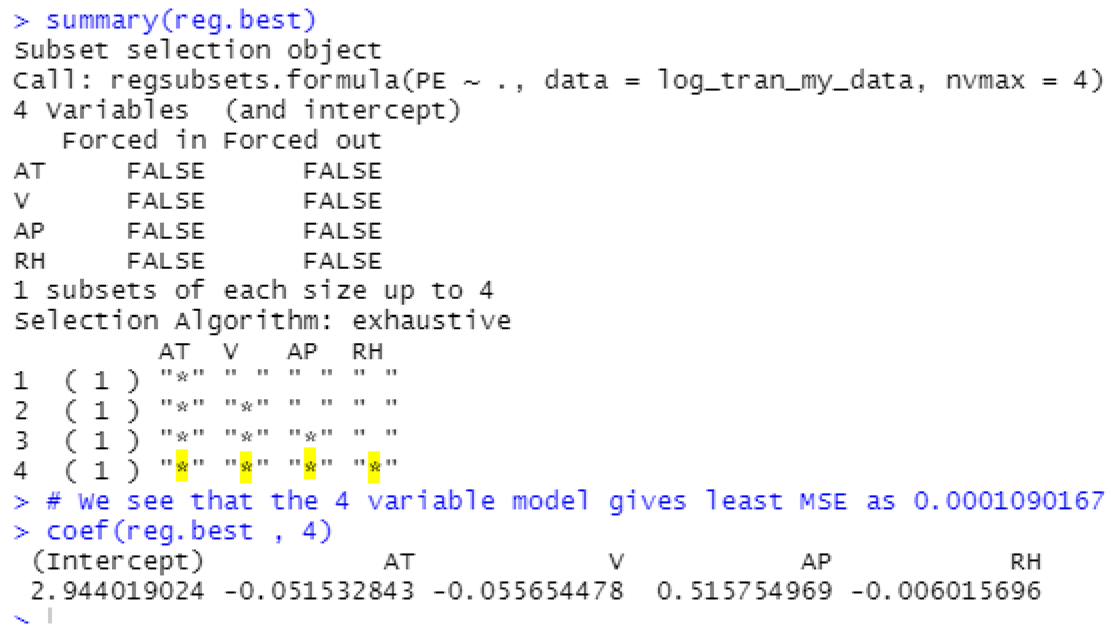


```
> val.errors
[1] 0.0001735037 0.0001149345 0.0001069582 0.0001060837
> which.min(val.errors)
[1] 4
```

The cross validation tells us that the 4 variable model has the least MSE. Hence, we select that and apply best subset selection on that model and get below coefficients.



```
> mean.cv.errors <- apply(cv.errors , 2, mean)
> mean.cv.errors
           1            2            3            4 
0.0001829516 0.0001182675 0.0001102586 0.0001090167
```

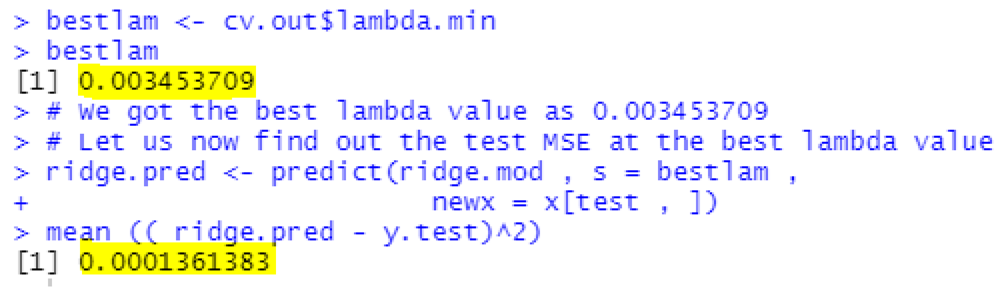


```
> summary(reg.best)
Subset selection object
Call: regsubsets.formula(PE ~ ., data = log_tran_my_data, nvmax = 4)
4 Variables  (and intercept)
   Forced in Forced out
AT     FALSE      FALSE
V      FALSE      FALSE
AP     FALSE      FALSE
RH     FALSE      FALSE
1 subsets of each size up to 4
Selection Algorithm: exhaustive
         AT  V   AP  RH 
1  ( 1 ) "*" " " " " " "
2  ( 1 ) "*" "*" " " " "
3  ( 1 ) "*" "*" "*" " "
4  ( 1 ) "*" "*" "*" "*"
> # We see that the 4 variable model gives least MSE as 0.0001090167
> coef(reg.best , 4)
 (Intercept)           AT            V           AP           RH 
 2.944019024 -0.051532843 -0.055654478  0.515754969 -0.006015696 
```

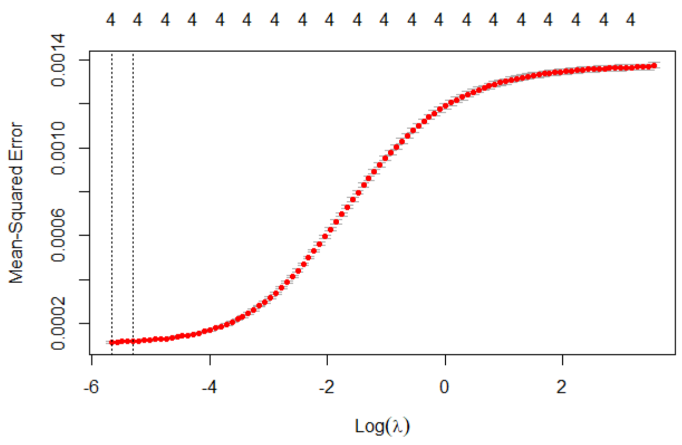
Regression Analysis

Ridge Regression

Using Cross-Validation, we find the best λ value as **0.003453709** and test MSE at best λ value is **0.0001361383**

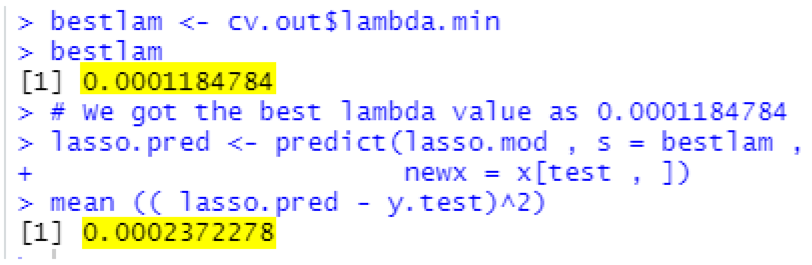


```
> bestlam <- cv.out$lambda.min
> bestlam
[1] 0.003453709
> # We got the best lambda value as 0.003453709
> # Let us now find out the test MSE at the best lambda value
> ridge.pred <- predict(ridge.mod , s = bestlam ,
+                       newx = x[test , ])
> mean (( ridge.pred - y.test)^2)
[1] 0.0001361383
```

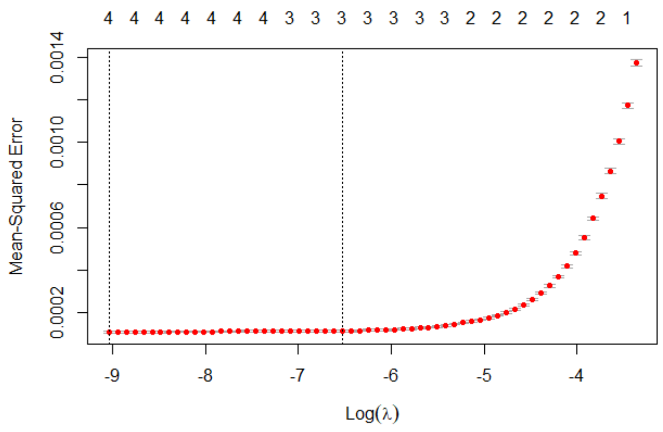


Lasso Regression

Using Cross-Validation, we find the best λ value as **0.0001184784** and test MSE at best λ value is **0.0002372278**

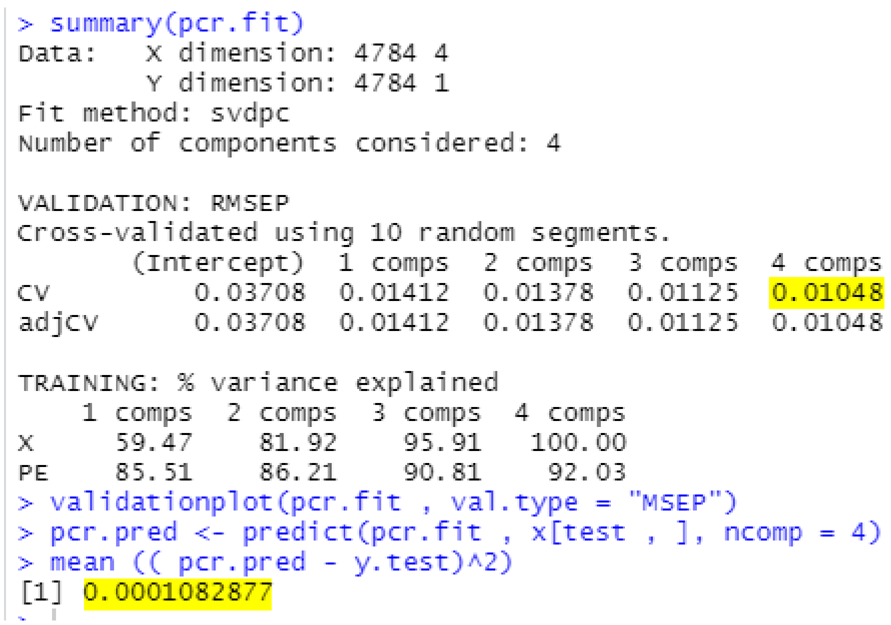


```
> bestlam <- cv.out$lambda.min
> bestlam
[1] 0.0001184784
> # we got the best lambda value as 0.0001184784
> lasso.pred <- predict(lasso.mod , s = bestlam ,
+                        newx = x[test , ])
> mean (( lasso.pred - y.test)^2)
[1] 0.0002372278
```



Principal Component Regression

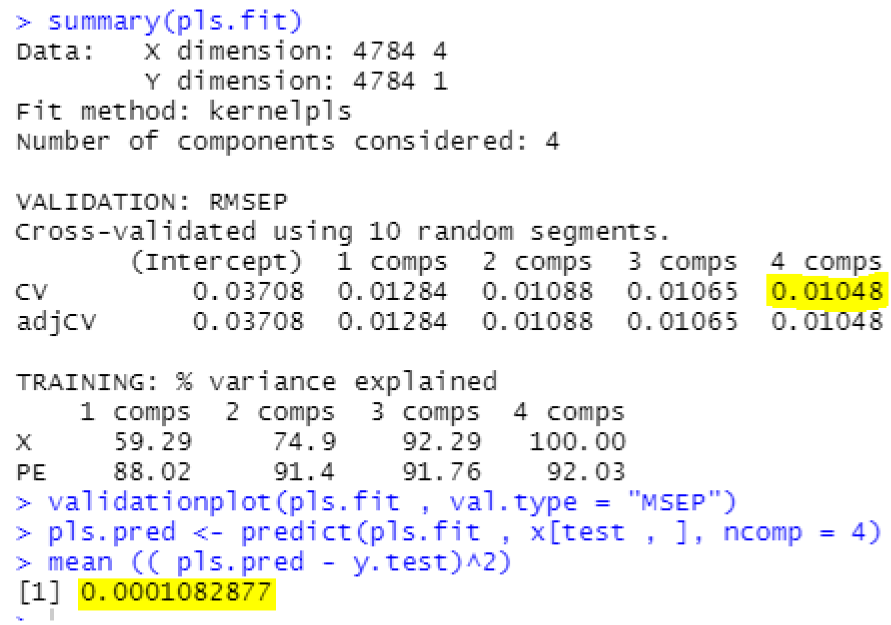
The test MSE obtained using PCR is **0.0001082877**



```
> summary(pcr.fit)
Data:   X dimension: 4784 4
        Y dimension: 4784 1
Fit method: svdpc
Number of components considered: 4

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps
CV         0.03708  0.01412  0.01378  0.01125  0.01048
adjCV      0.03708  0.01412  0.01378  0.01125  0.01048

TRAINING: % variance explained
    1 comps  2 comps  3 comps  4 comps
X     59.47    81.92    95.91   100.00
PE    85.51    86.21    90.81    92.03
> validationplot(pcr.fit , val.type = "MSEP")
> pcr.pred <- predict(pcr.fit , x[test , ], ncomp = 4)
> mean (( pcr.pred - y.test)^2)
[1] 0.0001082877
```

Partial Least Squares Regression

The test MSE obtained using PLS is **0.0001082877**, which is identical to that of PCR.

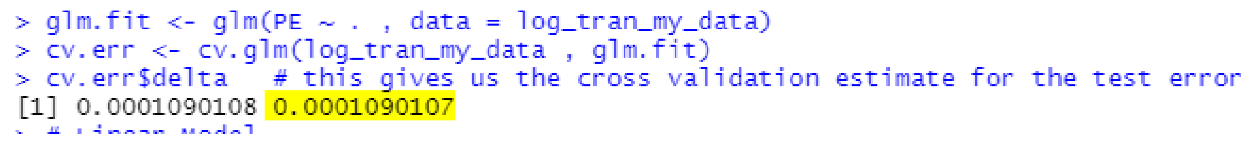


```
> summary(pls.fit)
Data:   X dimension: 4784 4
        Y dimension: 4784 1
Fit method: kernelpls
Number of components considered: 4

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps
CV         0.03708  0.01284  0.01088  0.01065  0.01048
adjCV      0.03708  0.01284  0.01088  0.01065  0.01048

TRAINING: % variance explained
    1 comps  2 comps  3 comps  4 comps
X     59.29     74.9    92.29   100.00
PE    88.02     91.4    91.76    92.03
> validationplot(pls.fit , val.type = "MSEP")
> pls.pred <- predict(pls.fit , x[test , ], ncomp = 4)
> mean (( pls.pred - y.test)^2)
[1] 0.0001082877
```

Cross – Validation

Leave One Out Cross Validation (LOOCV)

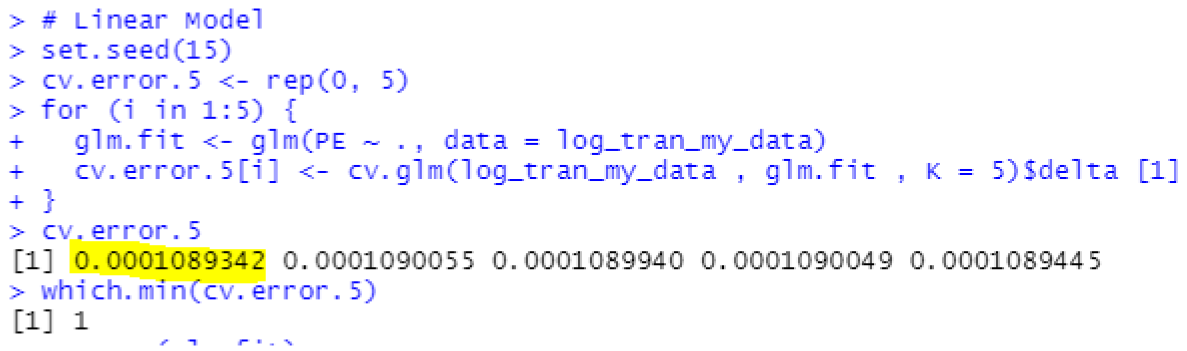
The test MSE obtained by using LOOCV is **0.0001090107**



```
> glm.fit <- glm(PE ~ . , data = log_tran_my_data)
> cv.err <- cv.glm(log_tran_my_data , glm.fit)
> cv.err$delta   # this gives us the cross validation estimate for the test error
[1] 0.0001090108 0.0001090107
```

5-Fold Cross Validation on first degree linear model

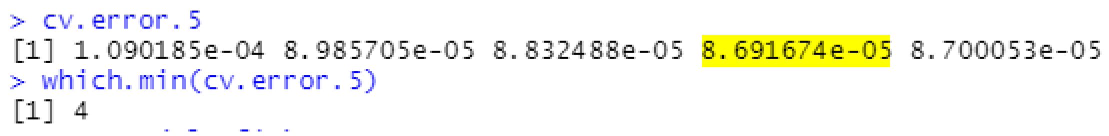
By using 5-Fold CV on the linear model, we obtain test MSE as **0.0001089342**



```
> # Linear Model
> set.seed(15)
> cv.error.5 <- rep(0, 5)
> for (i in 1:5) {
+   glm.fit <- glm(PE ~ ., data = log_tran_my_data)
+   cv.error.5[i] <- cv.glm(log_tran_my_data , glm.fit , K = 5)$delta [1]
+ }
> cv.error.5
[1] 0.0001089342 0.0001090055 0.0001089940 0.0001090049 0.0001089445
> which.min(cv.error.5)
[1] 1
```

5-Fold Cross Validation on polynomial model

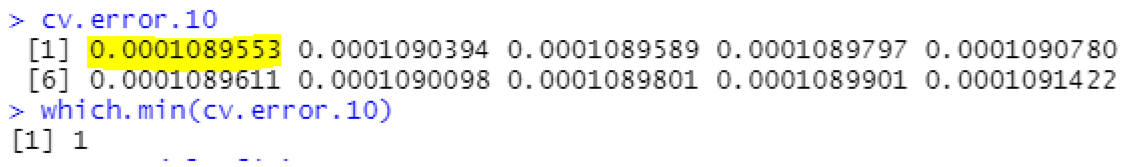
By using 5-Fold CV on the polynomial model, we obtain test MSE as **0.00008691.**



```
> cv.error.5
[1] 1.090185e-04 8.985705e-05 8.832488e-05 8.691674e-05 8.700053e-05
> which.min(cv.error.5)
[1] 4
```

10-Fold Cross Validation on first degree linear model

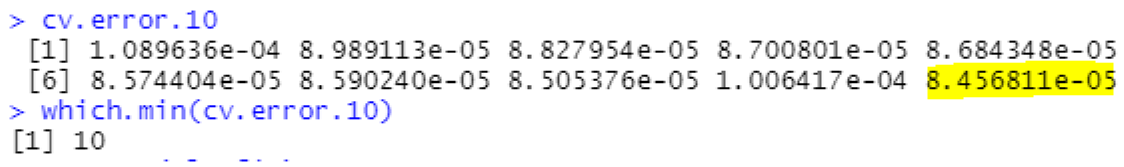
By using 10-Fold CV on linear model, we get test MSE as **0.0001089553**, which is almost identical to that of 5-fold CV test MSE.



```
> cv.error.10
 [1] 0.0001089553 0.0001090394 0.0001089589 0.0001089797 0.0001090780
 [6] 0.0001089611 0.0001090098 0.0001089801 0.0001089901 0.0001091422
> which.min(cv.error.10)
[1] 1
```

10-Fold Cross Validation on polynomial model

By using 10-Fold CV on polynomial model, we get test MSE as **0.00008456**



```
> cv.error.10
 [1] 1.089636e-04 8.989113e-05 8.827954e-05 8.700801e-05 8.684348e-05
 [6] 8.574404e-05 8.590240e-05 8.505376e-05 1.006417e-04 8.456811e-05
> which.min(cv.error.10)
[1] 10
```