Data Preparation and Analysis (CSP571)

Assignment-3

1. **Recitation Exercises**
   Chapter-6

1. **(a) Which of the three models with k predictors has the smallest training RSS?**
   The best subnet selection approach will have the smallest training RSS because it considers all the possible models. Other models mentioned use a greedy approach.
   (b) Which of the three models with k predictors has the smallest test RSS?
   We cannot determine if any of the three models have the small test RSS. The best subnet model may have low-training RSS since it considers all models. But this may overfit the model resulting in low test RSS.
   (c) True or False
   (i) True
   (ii) True
   (iii) False= sometimes it can be true but not always
   (iv) False= sometimes it can be true but not always
   (v) False= Best subnet selection may drop previously chosen predictors when a new one is added since it considers all possible subnets.
2. **(a) The lasso, relative to least squares, is:**
   Option (iii) is correct. When it performs feature selection Lasso generates a sparse model that is less flexible.
   (b) Repeat (a) for ridge regression relative to least squares.
   Option(iii) is correct. When ridge performs shrinkage, it shrinks predictors that don't have a strong relationship with the target variable hence they are less flexible. It reduces the variance at the cost of an increase in bias.
   (c) Repeat (a) for non-linear methods relative to least squares.
   Option(ii) is correct. The non-linear model follows the observation more tightly than least squares hence, it's more flexible. It also reduces the bias at the cost of an increase in variance.
3. **(a) As we increase s from 0, the training RSS will:**
   Option(iv) is correct. As we increase s, the model becomes more and more flexible as the restriction on β is reducing; hence, coefficients increase from 0 to their least square estimate values.
   (b) Repeat (a) for test RSS
   Option(ii) is correct. As the model is becoming more and more flexible the test RSS will reduce first and then start increasing when overfitting will start.
   (c) Repeat (a) for variance.
   Option(iii) is correct. Variance increases with the increase in model flexibility.
   (d) Repeat (a) for (squared) bias.
   Option(iv) is correct. Bias decreases with the increase in flexibility.
   (e) Repeat (a) for the irreducible error.
   Option (v) is correct. The irreducible error model does not depend on the value of s.
4. **(a) As we increase λ from 0, the training RSS will:**
   Option(iii) is correct. As we increase λ, the model becomes less and less flexible because the restriction on β is increasing and hence coefficients value comes closer to 0.
   (b) Repeat (a) for test RSS.
   Option (ii) is correct. As the model becomes less flexible the test RSS will reduce first and then increase when overfitting starts.
   (c) Repeat (a) for variance
   Option(iv) is correct. Variance decreases with the decrease in model flexibility.
   (d) Repeat (a) for (squared) bias.
   Option(iii) is correct. Bias increases with a decrease in model flexibility.
   (e) Repeat (a) for the irreducible error.

Option(v) is correct. The irreducible error model does not depend on λ.

5. (a) Write out the ridge regression optimization problem in this setting.

Chapter-6

Recitation Excercises

5 a) $n=2$

$p=2$

$x_{11} = x_{12}$

$y_1 + y_2 = 0$

$x_{12} + x_{22} = 0$

$x_{11} + x_{21} = 0$

$\beta_0 = 0$

Ridge Regression

$n=2$

$p=2$

$\beta_0 = 0$

a) $$\min\left[\left(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12}\right)^2 + \left(y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22}\right)\right]^2 + \lambda\left(\hat{\beta}_1^2 + \hat{\beta}_2^2\right)$$

(b) From a

$$\left(y_1^2 + \beta_1^2 x_{11} + \beta_2^2 x_{12} - 2\beta_2 x_{12} y_1 - 2\hat{\beta}_1 x_{11} y_1 + 2\hat{\beta}_1\hat{\beta}_2 x_{11} x_{12}\right)$$

$$+$$

$$\left(y_2^2 + \beta_1^2 x_{21}^2 + \hat{\beta}_2 x_{22}^2 - 2\beta_2 x_{22} y_2 + 2\hat{\beta}_1\hat{\beta}_2 x_{21} x_{22} - 2\hat{\beta}_1 x_2 y_1\right)$$

$$+$$

$$\lambda\left(\beta_1^2 + \beta_2^2\right)$$

taking the partial derivative to β_1 & setting exp to 0

$$\frac{\partial}{\partial \beta_1} : \left(2\hat{\beta}_1 x_{11}^2 - 2x_{11}y_1 + 2\hat{\beta}_2 x_{11}x_{12}\right) + \left(2\beta_1 x_{21}^2 - 2x_{21}y_2 + 2\beta_2 x_{21}x_{22}\right.$$

$$+ 2\lambda\beta_1 = 0$$

Let $x_{11} = x_{12} = x_1$ and $x_{21} = x_{22} = x_2$

$$\left(\hat{\beta}_1 x_1^2 - x_1 y_1 + \hat{\beta}_2 x_2^2\right) + \left(\hat{\beta}_1 x_2^2 - x_2 y_2 + \hat{\beta}_2 x_2^2\right) + \lambda\hat{\beta}_1 = 0$$

$$\beta_1\left(x_1^2 + x_2^2\right) + \beta_2\left(x_1^2 + x_2^2\right) + \lambda\beta_1 = x_1y_1 + x_2y_2$$

Adding $2\hat{\beta}_1 x_1x_2$ and $2\beta_2 x_1x_2$ on both sides.

$$\hat{\beta}_1\left(x_1 + x_2\right)^2 + \hat{\beta}_2\left(x_1 + x_2\right)^2 + \lambda\hat{\beta}_1 = x_1y_1 + x_2y_2 + 2\hat{\beta}_1 x_1x_2 + 2\beta_2 x_1x_2$$

Given $x_1 + x_2 = 0$

$$\lambda\beta_1 = x_1y_1 + x_2y_2 + 2\hat{\beta}_1 x_1x_2 + 2\hat{\beta}_2 x_1x_2 \quad - (1)$$

Taking partial derivative to β_2 we get

$$\lambda\beta_2 = x_1y_1 + x_2y_2 + 2\hat{\beta}_1 x_1x_2 + 2\hat{\beta}_2 x_1x_2 \quad - (2)$$

From (1) & (2) we get

$\lambda\beta_1 = \lambda\beta_2 \Rightarrow \beta_1 = \beta_2$

c) Lasso Optimization

$$\min\left[(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12})^2 + (y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22})^2 + \lambda(|\beta_1| + |\beta_2|)\right)$$

d) Replacing the penality term from Ridge Regression the derivative term to β

$$= \frac{\partial}{\partial \beta}(\lambda|\beta|) ; \frac{\lambda|\beta|}{\beta}$$

In ridge Regression

$$\lambda\left(\frac{|\beta_1|}{\beta_1}\right. = \frac{\lambda|\beta_2|}{\beta_2}$$

β_1 and β_2 can be both positive or both negative.

2. **Recitation Exercises**

Chapter-7

2.(a) Suppose that a curve ˆg is computed to smoothly fit a set of n points using the following formula.

(a) λ = ∞, m = 0.

In this case, gˆ is 0 because due to the large smoothing parameter g^0 (x)→0

(b) λ = ∞, m = 1.

In this case, gˆ is c because due to the large smoothing parameter g^1 (x)→0

(c) λ = ∞, m = 2.

In this case, gˆ is cx + d because due to the large smoothing parameter g^2 (x)→0

(d) λ = ∞, m = 3.

In this case, gˆ is cx^2 + dx + e because due to the large smoothing parameter g^3 (x)→0

(e) λ = 0, m = 3

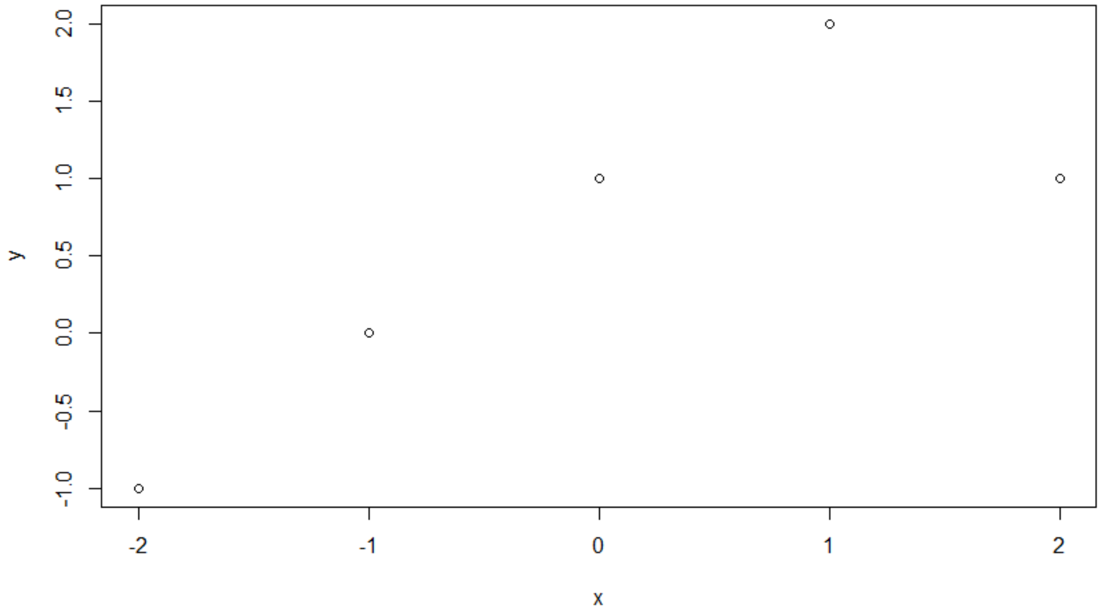
In this case, the smoothing will have no effect as λ = 0 and gˆ will interpolate the data set.

3. x = -2:2

y = 1 + x + -2 * (x-1)^2 * I(x>1)

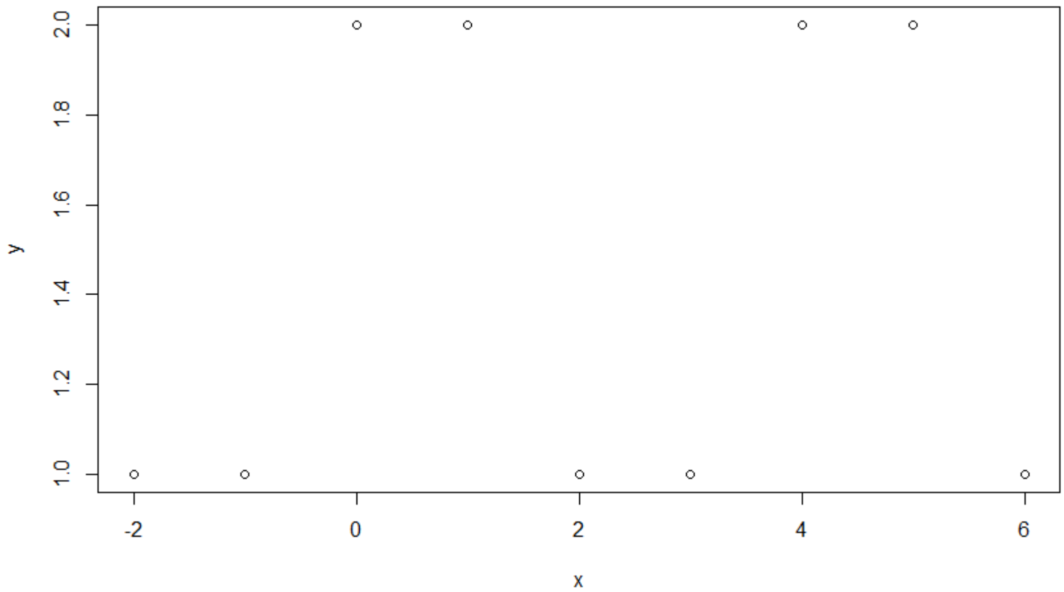
plot(x, y)

```
> x = -2:2
> y = 1 + x + -2 * (x-1)^2 * I(x>1)
> plot(x, y)
> |
```




4. ```
   x = -2:6
   y = c(1 + 0 + 0, # x = -2
       1 + 0 + 0, # x = -1
       1 + 1 + 0, # x = 0
       1 + (1-0) + 0, # x = 1
       1 + (1-1) + 0, # x =2
       1 + (0-0) + 0, # x =3
       1 + (0) + 1, # x =4
       1 + 0 + 1, # x =5
       1 + 0 + 0 # x =6
   )
   plot(x,y)
   ```

```
> x = -2:6
> y = c(1 + 0 + 0, # x = -2
+       1 + 0 + 0, # x = -1
+       1 + 1 + 0, # x = 0
+       1 + (1-0) + 0, # x = 1
+       1 + (1-1) + 0, # x =2
+       1 + (0-0) + 0, # x =3
+       1 + (0) + 1, # x =4
+       1 + 0 + 1, # x =5
+       1 + 0 + 0 # x =6
+ )
> plot(x,y)
>
```
```



5. **(a) As λ → ∞, will ˆg1 or ˆg2 have the smaller training RSS?**
   ˆg2 will have smaller training RSS as it will be a higher-order polynomial due to the order of penalty term.
   (b) As λ → ∞, will ˆg1 or ˆg2 have the smaller test RSS?
   ˆg1 will have a smaller test RSS value.
   (c) For λ = 0, will ˆg1 or ˆg2 have the smaller training and test RSS?
   Here λ = 0 and we have ˆg1 = ˆg2 so they will have equal training and test RSS.

2. Practicum Problems

Problem#1

library(caret)

library(ggplot2)

data("mtcars")

colnames(mtcars)

```
> library(caret)
> library(ggplot2)
> data("mtcars")
> colnames(mtcars)
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear" "carb"
> |
```

#80-20 training- testing split

SampleDataIndex = createDataPartition(y = mtcars$mpg, p = 0.8, list = FALSE)

TrainingDataMcars = mtcars[SampleDataIndex,]

TestingDtataMcars = mtcars[-SampleDataIndex,]

```
> #80-20 training- testing split
> SampleDataIndex = createDataPartition(y = mtcars$mpg, p = 0.8, list = FALSE)
> TrainingDataMcars = mtcars[SampleDataIndex,]
> TestingDtataMcars = mtcars[-SampleDataIndex,]
> |
```

#fitting a linear model

LinearModelMcars1 = lm(mpg~., data = TrainingDataMcars)

summary(LinearModelMcars1)

```
> LinearModelMcars1 = lm(mpg~., data = TrainingDataMcars)
> summary(LinearModelMcars1)

Call:
lm(formula = mpg ~ ., data = TrainingDataMcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.6339 -1.6263 -0.1275  0.7253  4.6465 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)  5.21282   20.13320   0.259   0.7988  
cyl          0.57539    1.09241   0.527   0.6052  
disp         0.01073    0.01813   0.592   0.5616  
hp          -0.03350    0.02365  -1.417   0.1746  
drat         1.51005    1.85994   0.812   0.4281  
wt          -3.76657    1.93958  -1.942   0.0689 .
qsec         1.10211    0.75907   1.452   0.1647  
vs          -0.87715    2.17482  -0.403   0.6917  
am           3.09750    2.11809   1.462   0.1619  
```

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.622 on 17 degrees of freedom
Multiple R-squared:   0.88,	Adjusted R-squared:  0.8094 
F-statistic: 12.47 on 10 and 17 DF,  p-value: 5.649e-06
```

#coefficients values

LinearModelMcars1$coefficients

```
> #coefficients values
> LinearModelMcars1$coefficients
(Intercept)         cyl        disp          hp        drat          wt        qsec          vs          am        gear 
 5.21282197  0.57539004  0.01073177 -0.03349912  1.51005122 -3.76656990  1.10210634 -0.87714708  3.09749975  0.01575818 
       carb 
-0.01052182 
> 
```

library(glmnet)

```
> library(glmnet)
> 
```

```
package 'RcppEigen' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
	C:\Users\aasth\AppData\Local\Temp\RtmpOyylWN\downloaded_packages


[3/3] Installing glmnet...

Installing package into 'C:/Users/aasth/AppData/Local/R/win-library/4.2'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/glmnet_4.1-4.zip'
Content type 'application/zip' length 2557243 bytes (2.4 MB)
==================================================
downloaded 2.4 MB

package 'glmnet' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
	C:\Users\aasth\AppData\Local\Temp\RtmpOyylWN\downloaded_packages


✓ Package 'glmnet' successfully installed.
```

#vector of 100 values

LambdaVector = 10^seq(5, -5, by = -.1)

```
> #vector of 100 values
> LambdaVector = 10^seq(5, -5, by = -.1)
> 
```

#Extracting data for minimum value of lambda

y = TrainingDataMcars$mpg

x = model.matrix(mpg~., data = TrainingDataMcars)

```
> y = TrainingDataMcars$mpg
> x = model.matrix(mpg~., data = TrainingDataMcars)
> 
```

#Use cross-validation (via cv.glmnet) to determine the minimum value for λ - what do you obtain?

LambdaVectorCV = cv.glmnet(x, y, lambda = LambdaVector, alpha = 0)

OptimalLambda = LambdaVectorCV$lambda.min

c(OptimalLambda)

```
> LambdaVectorCV = cv.glmnet(x, y, lambda = LambdaVector, alpha = 0 )
Warning message:
Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per fold
> OptimalLambda = LambdaVectorCV$lambda.min
> c(OptimalLambda)
[1] 2.511886
> 
```

Answer- Minimum value of lambda is 2.51186

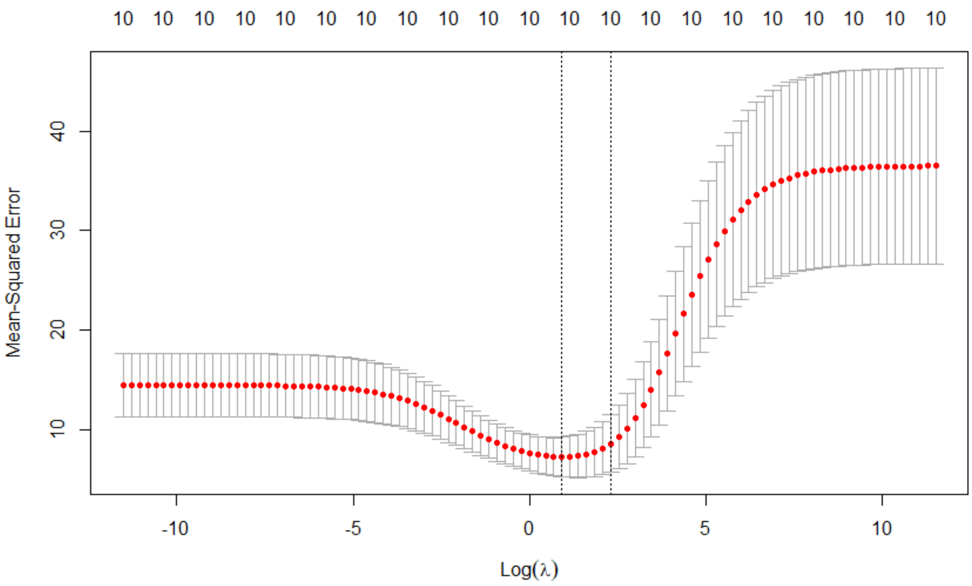
#Fitting the ridge regression

FitModel = glmnet(x, y, lambda = OptimalLambda, alpha = 0)

#Plot training MSE as a function of λ (you may also use log λ).

plot(LambdaVectorCV)

```
> #Fitting the ridge regression
> FitModel = glmnet(x, y, lambda = OptimalLambda, alpha = 0)
> #Plot training MSE as a function of λ (you may also use log λ).
> plot(LambdaVectorCV)
> 
```



#What is out-of-sample test set performance (using predict), and how do the coefficients differ versus the regular linear model?

coef(FitModel)

```
> coef(FitModel)
12 x 1 sparse Matrix of class "dgCMatrix"
                        s0
(Intercept) 20.448198506
(Intercept)  .
cyl         -0.287288332
disp        -0.005342594
hp          -0.013196073
drat         1.256065512
wt          -1.256736164
qsec         0.220717061
vs           0.305492479
am           1.685777455
gear         0.245686328
carb        -0.544637581
>
```

#Out of Sample test set performance for regular Linear model

LinearModelMcarsPredict = predict(LinearModelMcars1, newdata = TestingDtataMcars, type = "response")

LinearModelMcars1Actual = TestingDtataMcars[, "mpg"]

MeanValueRegModel = mean((LinearModelMcars1Actual - LinearModelMcarsPredict)^2)

c(MeanValueRegModel)

```
> LinearModelMcarsPredict = predict(LinearModelMcars1, newdata = TestingDtataMcars, type = "response")
> LinearModelMcars1Actual = TestingDtataMcars[, "mpg"]
> MeanValueRegModel = mean((LinearModelMcars1Actual - LinearModelMcarsPredict)^2)
> c(MeanValueRegModel)
[1] 12.65157
```

Answer- Out of sample training MSE for Linear Model is 12.65157

Has ridge regression performed shrinkage, variable selection, or both?

#Out of sample test set performance for regression Model

RegModel = model.matrix(mpg~., data = TestingDtataMcars)

RegModelPredict = predict(FitModel, s = OptimalLambda, newx = RegModel)

RegModelActual = TestingDtataMcars[, "mpg"]

RegModelMean = mean((RegModelActual - RegModelPredict)^2)

print(RegModelMean)

```
> RegModel = model.matrix(mpg~., data = TestingDtataMcars)
> RegModelPredict = predict(FitModel, s = OptimalLambda, newx = RegModel)
> RegModelActual = TestingDtataMcars[, "mpg"]
> RegModelMean = mean((RegModelActual - RegModelPredict)^2)
> print(RegModelMean)
[1] 7.165049
```

Answer- Out of sample training MSE for Linear Model is 12.65157 and the out-of-sample Training MSE after ridge regression is 7.165049. So, it can be clearly seen that ridge regression has performed shrinkage.

Problem#2

library(ggplot2)

library(caret)

data("swiss")

colnames(swiss)

```
> library(ggplot2)
> library(caret)
> data("swiss")
> colnames(swiss)
[1] "Fertility"        "Agriculture"      "Examination"      "Education"        "Catholic"         "Infant.Mortality"
> |
```

#80-20 training-testing split

SampleSwissIndex = createDataPartition(y = swiss$Fertility, p = 0.8, list = FALSE)

TrainingDataSwiss = swiss[SampleSwissIndex,]

TestingDataSwiss = swiss[-SampleSwissIndex,]

```
> #80-20 training-testing split
> SampleSwissIndex = createDataPartition(y = swiss$Fertility, p = 0.8, list = FALSE)
> TrainingDataSwiss = swiss[SampleSwissIndex,]
> TestingDataSwiss = swiss[-SampleSwissIndex,]
> |
```

#fitting a linear model

LinearModelSwiss1 = lm(Fertility~., data = TrainingDataSwiss)

summary(LinearModelSwiss1)

```
> LinearModelSwiss1 = lm(Fertility~., data = TrainingDataSwiss)
> summary(LinearModelSwiss1)

Call:
lm(formula = Fertility ~ ., data = TrainingDataSwiss)

Residuals:
     Min       1Q   Median       3Q      Max 
-12.9988  -4.3723   0.2933   3.1141  13.9618 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)      62.04634   10.81152   5.739 2.08e-06 ***
Agriculture      -0.10866    0.07534  -1.442 0.158643    
Examination      -0.18441    0.25427  -0.725 0.473397    
Education        -0.75650    0.17782  -4.254 0.000163 ***
Catholic          0.11414    0.03475   3.284 0.002424 ** 
Infant.Mortality  1.03405    0.36984   2.796 0.008562 ** 
---
```


```
Catholic          0.11414    0.03475   3.284 0.002424 ** 
Infant.Mortality  1.03405    0.36984   2.796 0.008562 ** 
```


```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.675 on 33 degrees of freedom
Multiple R-squared:  0.7477,	Adjusted R-squared:  0.7095 
F-statistic: 19.56 on 5 and 33 DF,  p-value: 5.072e-09

> 
```

#coefficient values

#What are the associated coefficient values for relevant features?

LinearModelSwiss1$coefficients

```
> LinearModelSwiss1$coefficients
     (Intercept)      Agriculture      Examination        Education         Catholic Infant.Mortality 
      62.0463423       -0.1086631       -0.1844132       -0.7564982        0.1141418        1.0340487 
> 
```

library(glmnet)

#vector of 100 values

LambdaVectorSwiss = 10^seq(5, -5, by = -.1)

#Extracting x and y values from training data

y = TrainingDataSwiss$Fertility

x = model.matrix(Fertility~., data = TrainingDataSwiss)

```
> #vector of 100 values
> LambdaVectorSwiss = 10^seq(5, -5, by = -.1)
> #Extracting x and y values from training data
> y = TrainingDataSwiss$Fertility
> x = model.matrix(Fertility~., data = TrainingDataSwiss)
> 
```

#Use cross-validation (via cv.glmnet) to determine the minimum value for λ - what do you obtain?

LambdaVectorSwissCV = cv.glmnet(x, y, lambda = LambdaVectorSwiss, alpha = 1)

OptimalLambdaSwiss = LambdaVectorSwissCV$lambda.min c(OptimalLambdaSwiss)

```
> LambdaVectorSwissCV = cv.glmnet(x, y, lambda = LambdaVectorSwiss, alpha = 1)
> OptimalLambdaSwiss = LambdaVectorSwissCV$lambda.min
> c(OptimalLambdaSwiss)
[1] 0.6309573
>
```

Answer- Minimum value of lambda is 0.6309573

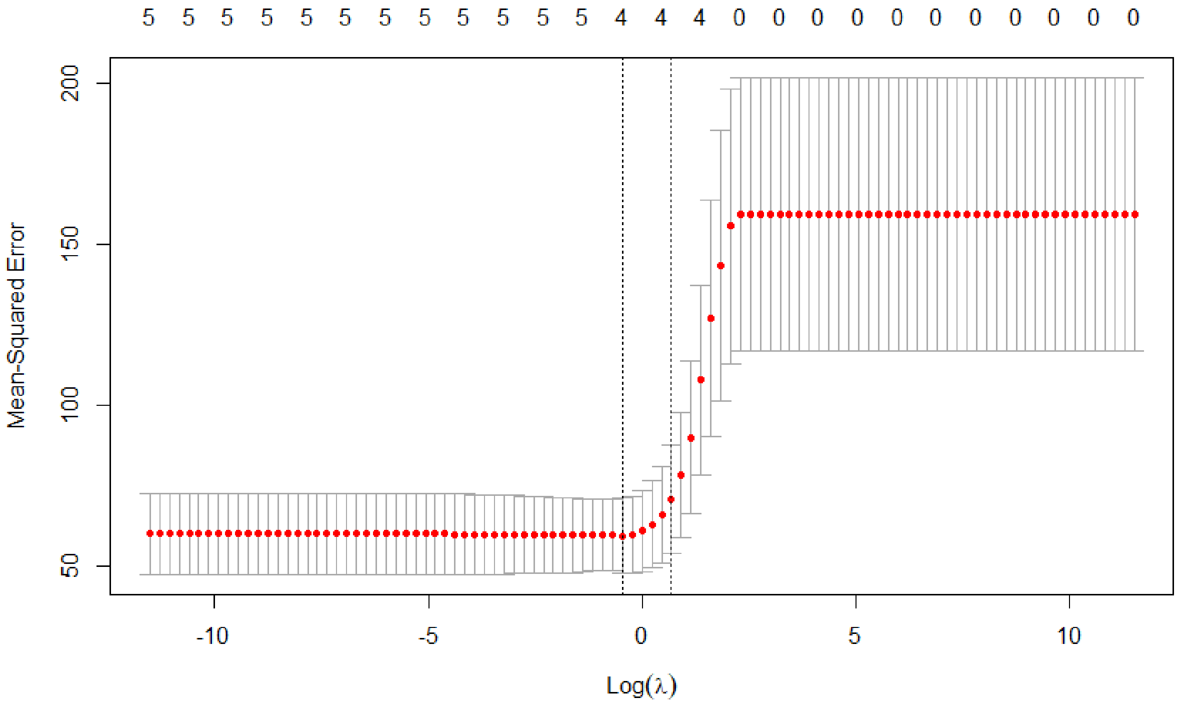
#fitting lasso regression

FitModelSwiss = glmnet(x, y, lambda = OptimalLambdaSwiss, alpha = 1)

#Plot training MSE as a function of λ (you may also use log λ).

plot(LambdaVectorSwissCV)

```
> #fitting lasso regression
> FitModelSwiss = glmnet(x, y, lambda = OptimalLambdaSwiss, alpha = 1)
> #Plot training MSE as a function of λ (you may also use log λ).
> plot(LambdaVectorSwissCV)
>
```



#What is out-of-sample test set performance (using predict), and how do the coefficients differ versus the regular linear model?

coef(FitModelSwiss)

```
> coef(FitModelSwiss)
7 x 1 sparse Matrix of class "dgCMatrix"
                        s0
(Intercept)      54.49765837
(Intercept)       .
Agriculture       .
Examination      -0.06718265
Education        -0.60898884
Catholic          0.10008801
Infant.Mortality  0.97759315
>
```

#Out of Sample test set performance for regular Linear model

LinearModelSwissPredict = predict(LinearModelSwiss1, newdata = TestingDataSwiss, type = "response")

LinearModelSwissActual = TestingDataSwiss[, "Fertility"]

LinearModelSwissMean = mean((LinearModelSwissActual - LinearModelSwissPredict)^2)

print(LinearModelSwissMean)

```
> LinearModelSwissPredict = predict(LinearModelSwiss1, newdata = TestingDataSwiss, type = "response")
> LinearModelSwissActual = TestingDataSwiss[, "Fertility"]
> LinearModelSwissMean = mean((LinearModelSwissActual - LinearModelSwissPredict)^2)
> print(LinearModelSwissMean)
[1] 89.6039
> |
```

Out of sample training MSE for regular Linear Model = 89.6039

#Has lasso regression performed shrinkage, variable selection, or both?

#Out of sample test set performance for lasso regression model

LassoRegModel = model.matrix(Fertility~., data = TestingDataSwiss)

LassoRegModelPredict = predict(FitModelSwiss, s = OptimalLambdaSwiss, newx = LassoRegModel)

LassoRegModelActual = TestingDataSwiss[, "Fertility"]

LassoRegModelMean = mean((LassoRegModelActual - LassoRegModelPredict)^2)

print(LassoRegModelMean)

```
> LassoRegModel = model.matrix(Fertility~., data = TestingDataSwiss)
> LassoRegModelPredict = predict(FitModelSwiss, s = OptimalLambdaSwiss, newx = LassoRegModel)
> LassoRegModelActual = TestingDataSwiss[, "Fertility"]
> LassoRegModelMean = mean((LassoRegModelActual - LassoRegModelPredict)^2)
> print(LassoRegModelMean)
[1] 114.9967
> |
```

Answer- After performing Lasso regression the out-of-sample MSE has raised from 89.6039 to 114.9967. Lasso regression usually performs variable selection but, in this case, it performs shrinkage as the coefficients have shrunk to some extent.

Problem#3

library(readxl)

library(magrittr)

ConcreteData1 = read_excel("C:\\Users\\aasth\\Downloads\\Concrete_Data.xls")

summary(ConcreteData1)

```
> library(readxl)
> library(magrittr)
> ConcreteData1 = read_excel("C:\\Users\\aasth\\Downloads\\Concrete_Data.xls")
> summary(ConcreteData1)
 Cement (component 1)(kg in a m^3 mixture) Blast Furnace Slag (component 2)(kg in a m^3 mixture)
 Min.   :102.0                             Min.   :  0.0
 1st Qu.:192.4                             1st Qu.:  0.0
 Median :272.9                             Median : 22.0
 Mean   :281.2                             Mean   : 73.9
 3rd Qu.:350.0                             3rd Qu.:142.9
 Max.   :540.0                             Max.   :359.4
 Fly Ash (component 3)(kg in a m^3 mixture) Water  (component 4)(kg in a m^3 mixture)
 Min.   :  0.00                             Min.   :121.8
 1st Qu.:  0.00                             1st Qu.:164.9
 Median :  0.00                             Median :185.0
 Mean   : 54.19                             Mean   :181.6
 3rd Qu.:118.27                             3rd Qu.:192.0
 Max.   :200.10                             Max.   :247.0
 Superplasticizer (component 5)(kg in a m^3 mixture) Coarse Aggregate  (component 6)(kg in a m^3 mixture)
```


```
 Max.   :540.0                             Max.   :359.4
 Fly Ash (component 3)(kg in a m^3 mixture) Water  (component 4)(kg in a m^3 mixture)
 Min.   :  0.00                             Min.   :121.8
 1st Qu.:  0.00                             1st Qu.:164.9
 Median :  0.00                             Median :185.0
 Mean   : 54.19                             Mean   :181.6
 3rd Qu.:118.27                             3rd Qu.:192.0
 Max.   :200.10                             Max.   :247.0
 Superplasticizer (component 5)(kg in a m^3 mixture) Coarse Aggregate  (component 6)(kg in a m^3 mixture)
```


```
 Min.   : 0.000                                      Min.   : 801.0
 1st Qu.: 0.000                                      1st Qu.: 932.0
 Median : 6.350                                      Median : 968.0
 Mean   : 6.203                                      Mean   : 972.9
 3rd Qu.:10.160                                      3rd Qu.:1029.4
 Max.   :32.200                                      Max.   :1145.0
 Fine Aggregate (component 7)(kg in a m^3 mixture)  Age (day)       Concrete compressive strength(MPa, megapascals)
 Min.   :594.0                                     Min.   :  1.00   Min.   : 2.332
 1st Qu.:731.0                                     1st Qu.:  7.00   1st Qu.:23.707
 Median :779.5                                     Median : 28.00   Median :34.443
 Mean   :773.6                                     Mean   : 45.66   Mean   :35.818
 3rd Qu.:824.0                                     3rd Qu.: 56.00   3rd Qu.:46.136
 Max.   :992.6                                     Max.   :365.00   Max.   :82.599
>
```

colnames(ConcreteData1)

```
> colnames(ConcreteData1)
[1] "Cement (component 1)(kg in a m^3 mixture)"              "Blast Furnace Slag (component 2)(kg in a m^3 mixture)"
[3] "Fly Ash (component 3)(kg in a m^3 mixture)"             "Water  (component 4)(kg in a m^3 mixture)"
[5] "Superplasticizer (component 5)(kg in a m^3 mixture)"    "Coarse Aggregate  (component 6)(kg in a m^3 mixture)"
[7] "Fine Aggregate (component 7)(kg in a m^3 mixture)"      "Age (day)"
[9] "Concrete compressive strength(MPa, megapascals)"
>
```

library(mgcv)

library(stringr)

names(ConcreteData1)[1] = "cement"   #c1

names(ConcreteData1)[2] = "slag"   #c2

names(ConcreteData1)[3] = "ash"   #c3

names(ConcreteData1)[4] = "water"   #c4

names(ConcreteData1)[5] = "superplasticizer" #c5

names(ConcreteData1)[6] = "coarse"   #c6

names(ConcreteData1)[7] = "fine"

names(ConcreteData1)[8] = "age"

names(ConcreteData1)[9] = "ccs"

names(ConcreteData1)

```
[9] Concrete Compressive Strength(MPa, megapascals)
> library(mgcv)
> library(stringr)
> names(ConcreteData1)[1] = "cement"     #c1
> names(ConcreteData1)[2] = "slag"       #c2
> names(ConcreteData1)[3] = "ash"        #c3
> names(ConcreteData1)[4] = "water"      #c4
> names(ConcreteData1)[5] = "superplasticizer" #c5
> names(ConcreteData1)[6] = "coarse"     #c6
> names(ConcreteData1)[7] = "fine"
> names(ConcreteData1)[8] = "age"
> names(ConcreteData1)[9] = "ccs"
> names(ConcreteData1)
[1] "cement"           "slag"             "ash"              "water"            "superplasticizer" "coarse"
[7] "fine"             "age"              "ccs"
>
```

#creating a generalized addictive model as linear model for C1-C6

LinearModelAddict = gam(ccs ~ cement + slag + ash + water + superplasticizer + coarse, data = ConcreteData1)

summary(LinearModelAddict)

```
> #creating a generalized addictive model as linear model for C1-C6
> LinearModelAddict = gam(ccs ~ cement + slag + ash + water + superplasticizer + coarse, data = ConcreteData1 )
> summary(LinearModelAddict)

Family: gaussian
Link function: identity

Formula:
ccs ~ cement + slag + ash + water + superplasticizer + coarse

Parametric coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       5.326997  10.510518   0.507 0.612387
cement            0.108256   0.005214  20.761  < 2e-16 ***
slag              0.079357   0.006193  12.814  < 2e-16 ***
ash               0.055928   0.009287   6.022  2.4e-09 ***
water            -0.103871   0.027796  -3.737 0.000197 ***
superplasticizer  0.356016   0.110251   3.229 0.001281 **
coarse            0.008027   0.006272   1.280 0.200940
---
```

```
coarse            0.008027   0.006272   1.280 0.200940
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


R-sq.(adj) =  0.445   Deviance explained = 44.9%
GCV = 155.83  Scale est. = 154.77    n = 1030
```

Answer- The value of R2 here is 0.445

#creating generalized addictive model as linear model for all values

LinearModelAddict1 <- gam(ccs ~ cement + slag + ash + water + superplasticizer + coarse + fine +

age, data = ConcreteData1)

summary(LinearModelAddict1)

```
> #creating generalized addictive model as linear model for all values
> LinearModelAddict1 <- gam(ccs ~ cement + slag + ash + water + superplasticizer + coarse + fine +
+ age, data = ConcreteData1)
> summary(LinearModelAddict1)

Family: gaussian
Link function: identity

Formula:
ccs ~ cement + slag + ash + water + superplasticizer + coarse +
    fine + age

Parametric coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)     -23.163756  26.588421  -0.871 0.383851
cement            0.119785   0.008489  14.110  < 2e-16 ***
```

```
Parametric coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)       -23.163756  26.588421  -0.871 0.383851
cement              0.119785   0.008489  14.110  < 2e-16 ***
slag                0.103847   0.010136  10.245  < 2e-16 ***
ash                 0.087943   0.012585   6.988 5.03e-12 ***
water              -0.150298   0.040179  -3.741 0.000194 ***
superplasticizer    0.290687   0.093460   3.110 0.001921 **
coarse              0.018030   0.009394   1.919 0.055227 .
fine                0.020154   0.010703   1.883 0.059968 .
age                 0.114226   0.005427  21.046  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


R-sq.(adj) =  0.612   Deviance explained = 61.5%
GCV = 109.11  Scale est. = 108.16    n = 1030
> |
```

Answer- In this the R2 value for linear model is 0.612 respectively.

#creating a generalized addictive model as non linear for C1-C6

NonLinearModelAddict = gam(ccs ~ s(cement)+ s(slag)+ s(water)+ s(ash)+

s(superplasticizer)+ s(coarse), data = ConcreteData1)

summary(NonLinearModelAddict)

```
GCV = 109.11  Scale est. = 108.16    n = 1030
> NonLinearModelAddict = gam(ccs ~ s(cement)+ s(slag)+ s(water)+ s(ash)+
+ s(superplasticizer)+ s(coarse), data = ConcreteData1)
> summary(NonLinearModelAddict)

Family: gaussian
Link function: identity

Formula:
ccs ~ s(cement) + s(slag) + s(water) + s(ash) + s(superplasticizer) +
    s(coarse)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  35.8178     0.3566   100.4   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
```


```
Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  35.8178     0.3566   100.4   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
```


```
                      edf Ref.df      F  p-value
s(cement)           4.464  5.513 69.530  < 2e-16 ***
s(slag)             2.088  2.578 48.091  < 2e-16 ***
s(water)            8.567  8.936 13.504  < 2e-16 ***
s(ash)              5.332  6.404  1.784    0.101
s(superplasticizer) 7.133  8.143  5.498 1.22e-06 ***
s(coarse)           1.000  1.000  0.018    0.892
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.531   Deviance explained = 54.4%
GCV = 134.84  Scale est. = 130.96    n = 1030
> |
```

Answer- R2 value for non-linear model is 0.531

#Creating a generalised addictive model as non- linear for all values

NonLinearModelAddict2 <- gam(ccs ~ s(cement) + s(slag) + s(ash) + s(water) + s(superplasticizer) +

s(coarse) + s(fine) + s(age), data = ConcreteData1)

summary(NonLinearModelAddict2)

```
> #Creating a generalised addictive model as non- linear for all values
> NonLinearModelAddict2 <- gam(ccs ~ s(cement) + s(slag) + s(ash) + s(water) + s(superplasticizer) +
+ s(coarse) + s(fine) + s(age), data = ConcreteData1)
> summary(NonLinearModelAddict2)

Family: gaussian
Link function: identity

Formula:
ccs ~ s(cement) + s(slag) + s(ash) + s(water) + s(superplasticizer) +
    s(coarse) + s(fine) + s(age)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  35.8178     0.1675   213.9   <2e-16 ***
---
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  35.8178     0.1675   213.9   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                      edf Ref.df       F  p-value    
s(cement)           8.223  8.830  48.690  < 2e-16 ***
s(slag)             8.114  8.757  25.041  < 2e-16 ***
s(ash)              8.256  8.817   9.354  < 2e-16 ***
s(water)            8.742  8.973  26.116  < 2e-16 ***
s(superplasticizer) 8.039  8.743  10.845  < 2e-16 ***
s(coarse)           7.904  8.673   3.403 0.000593 ***
s(fine)             8.618  8.951  18.435  < 2e-16 ***
s(age)              8.559  8.900 365.119  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.896   Deviance explained = 90.3%
GCV = 30.914  Scale est. = 28.89     n = 1030
> |
```

Answer- In this R2 is 0.896 respectively.

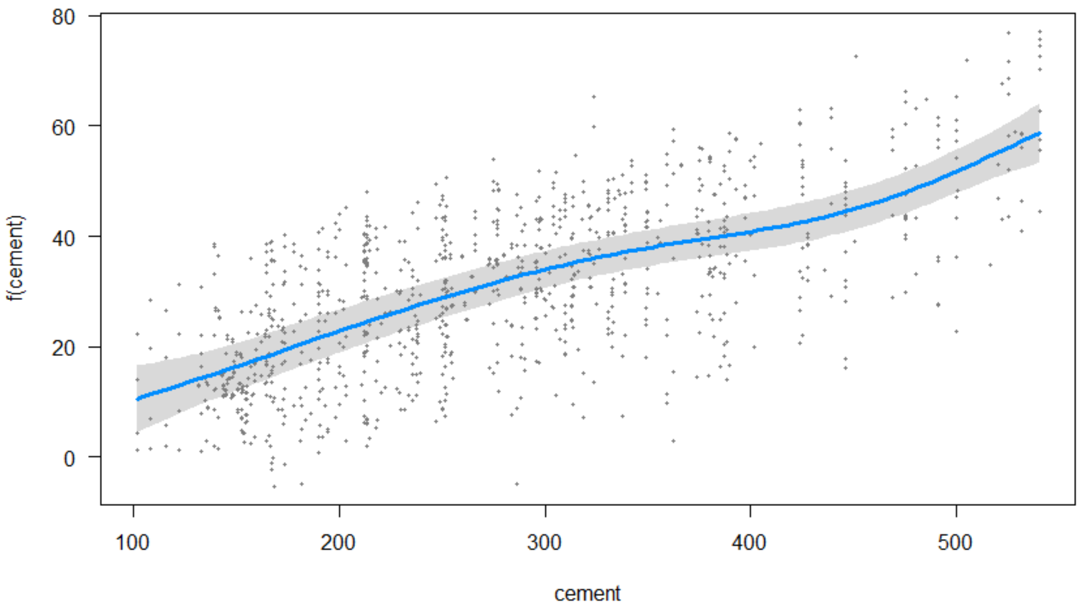
 # Visualize the regression using the visreg package, showing the fit as a function of each predictor with confidence intervals

library(visreg)

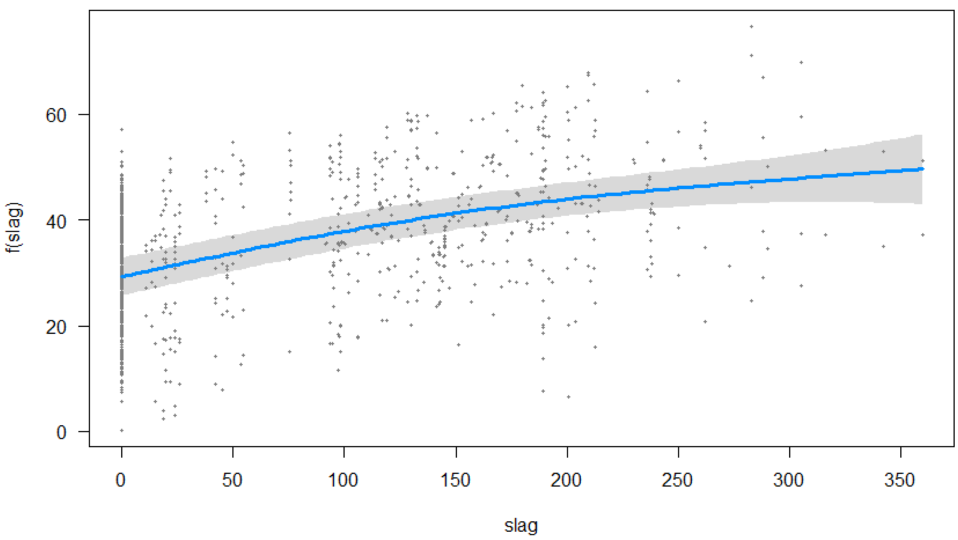
visreg(NonLinearModelAddict)

visreg(NonLinearModelAddict2)

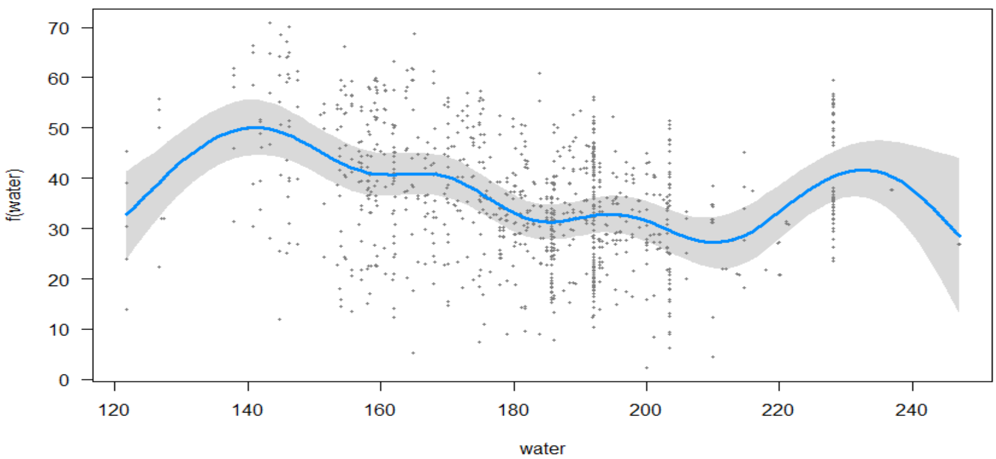
```
> library(visreg)
> visreg(NonLinearModelAddict)
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
>
> |
```



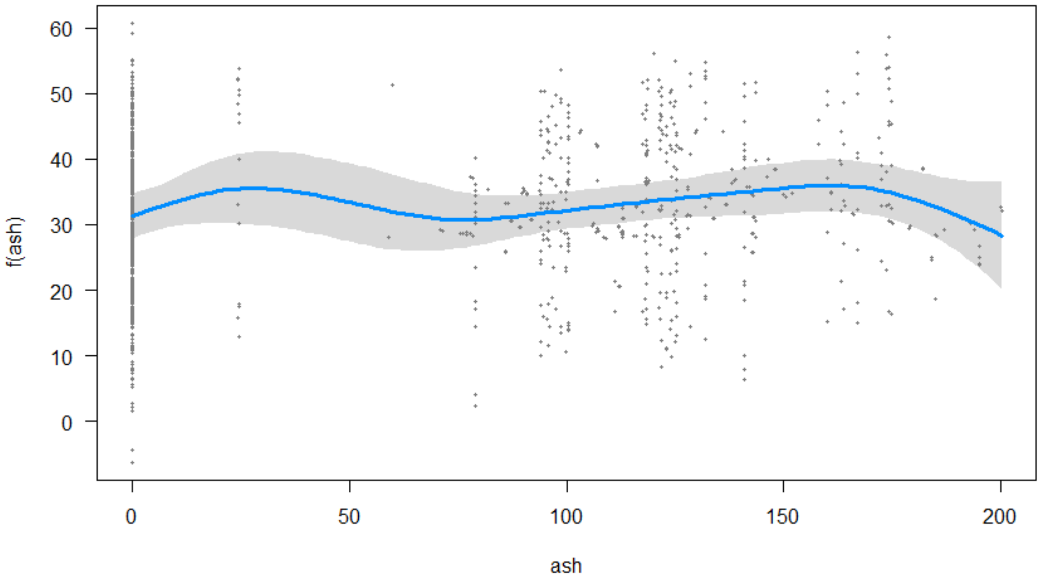
80
60
40
20
0
f(cement)
100
200
300
400
500
cement

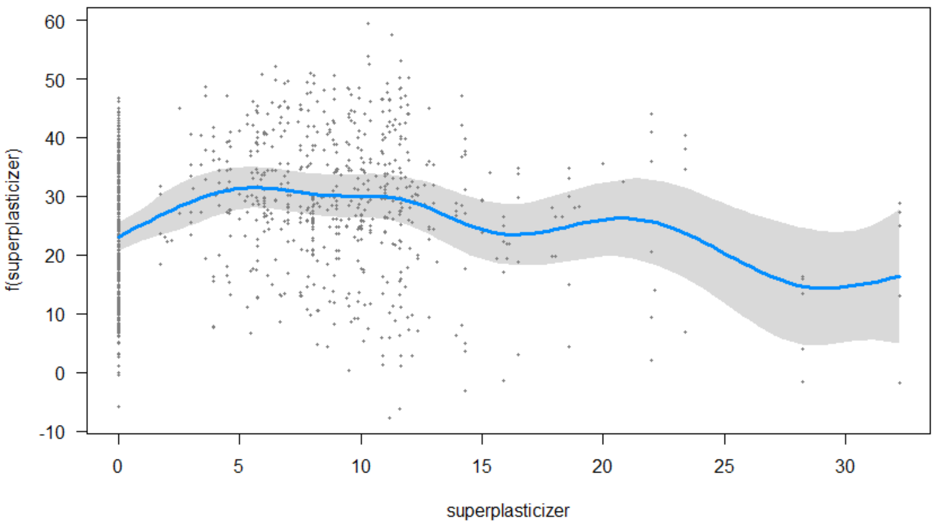


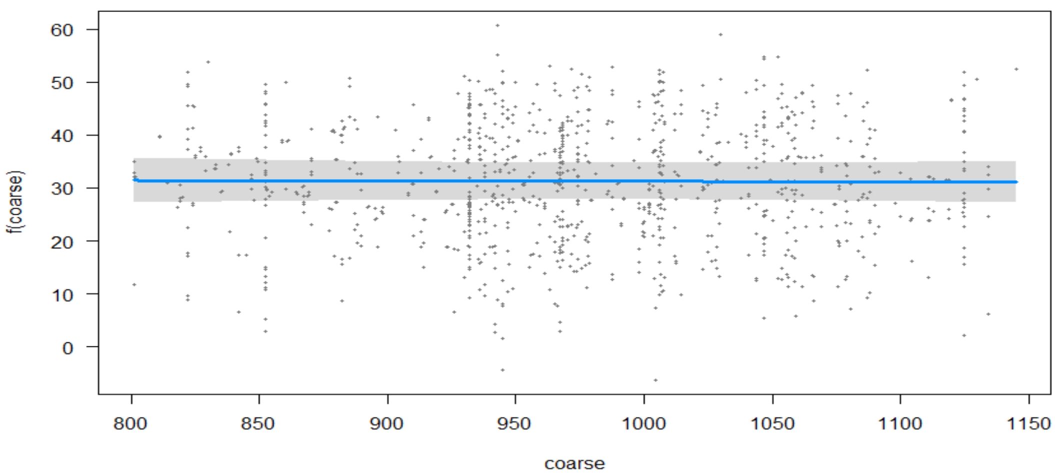
60
40
20
0
f(slag)
0
50
100
150
200
250
300
350
slag



70
60
50
40
30
20
10
0
f(water)
120
140
160
180
200
220
240
water

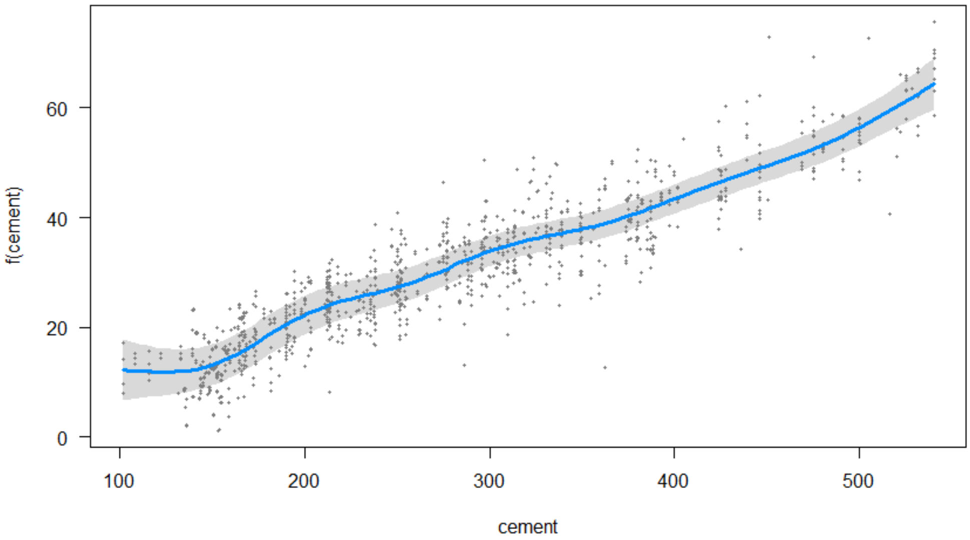

f(ash)
ash
60
50
40
30
20
10
0
0
50
100
150
200

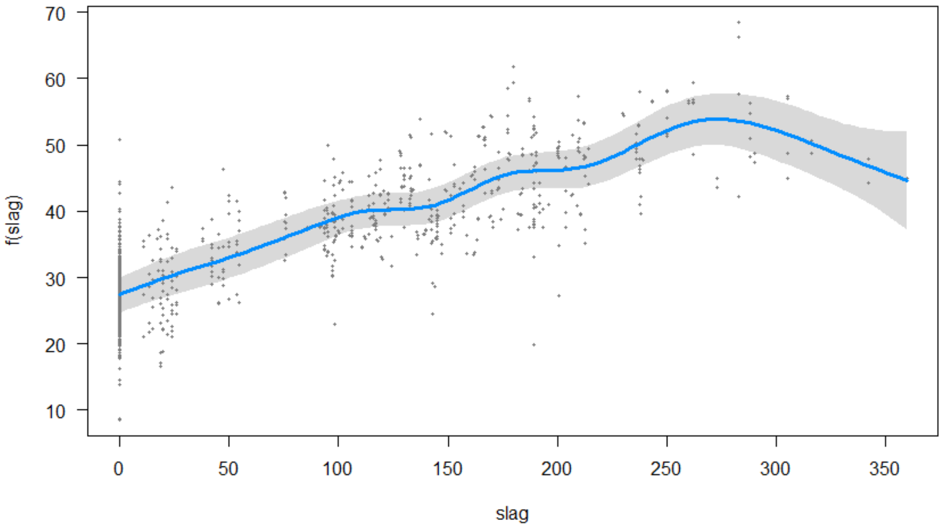

f(superplasticizer)
superplasticizer
60
50
40
30
20
10
0
-10
0
5
10
15
20
25
30

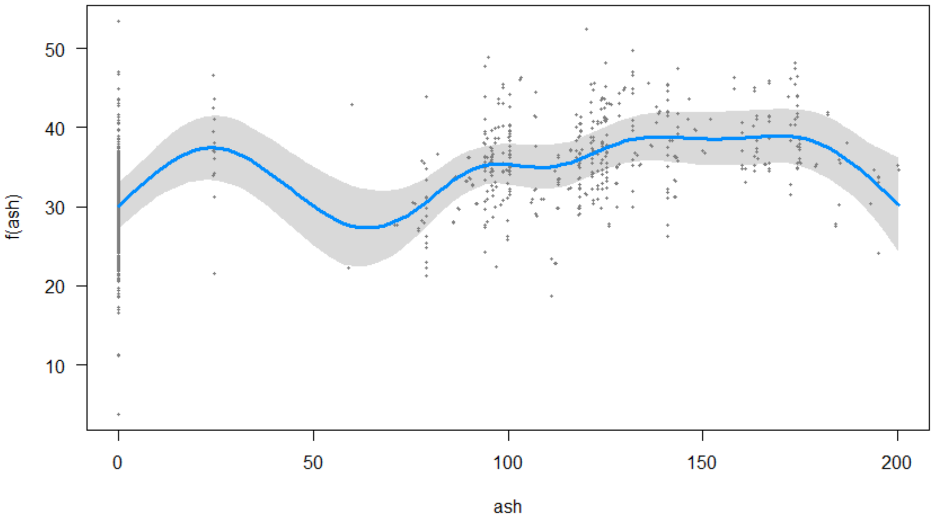

f(coarse)
coarse
60
50
40
30
20
10
0
800
850
900
950
1000
1050
1100
1150

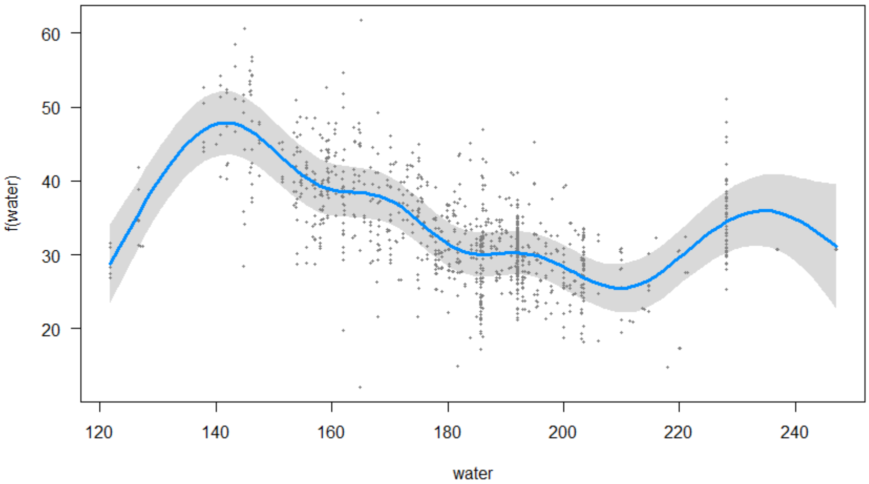
visreg(NonLinearModelAddict2)

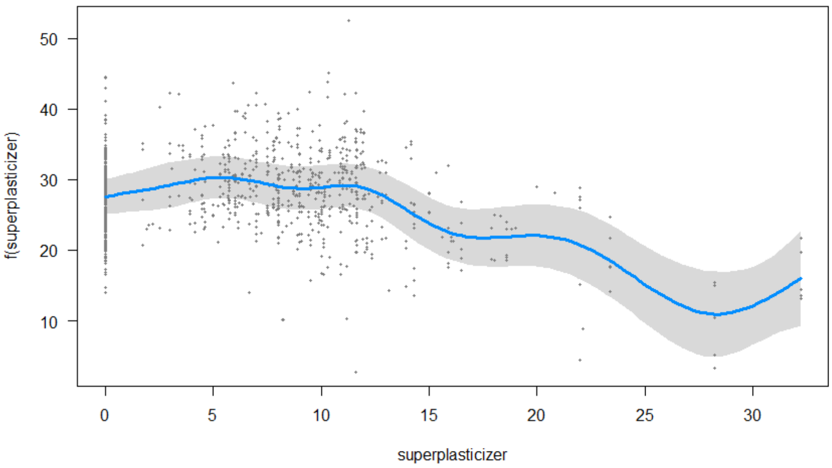
```
>
> visreg(NonLinearModelAddict2)
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
>
```

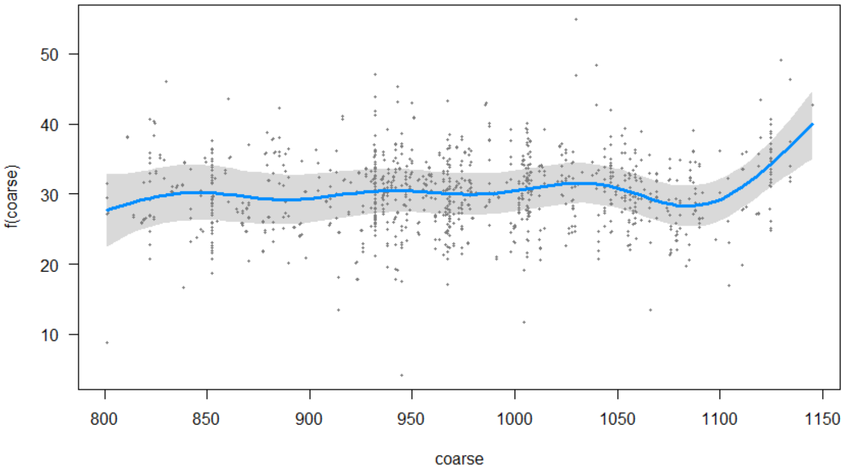


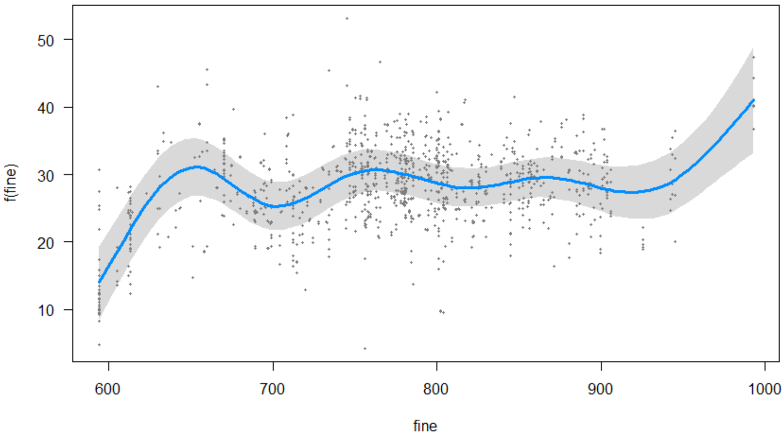


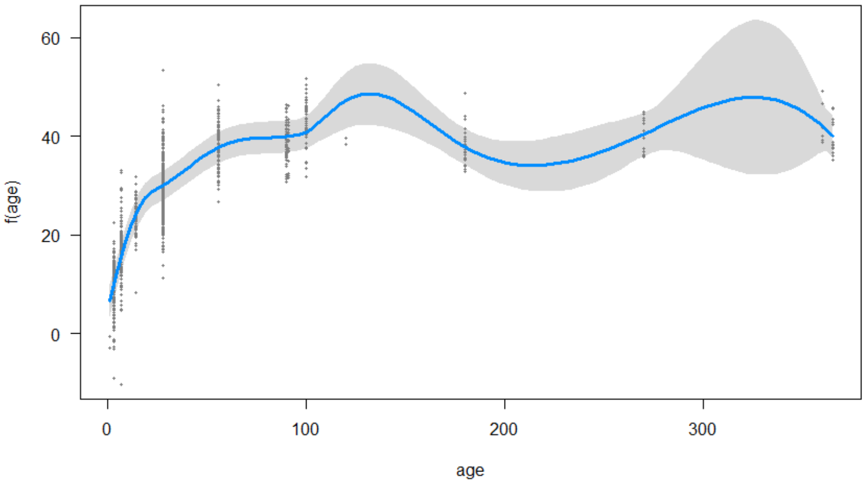

f(ash)
50
40
30
20
10
0
50
100
150
200
ash


f(water)
60
50
40
30
20
120
140
160
180
200
220
240
water


f(superplasticizer)
50
40
30
20
10
0
5
10
15
20
25
30
superplasticizer







Submitted By: -

Aastha Dhir

CWID-A20468022

adhir2@hawk.iit.edu