Modelling - Ridge & Lasso Regression

```r
library(stringr)
```

```
## Warning: package 'stringr' was built under R version 3.6.3
```

```r
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
##
## 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
```

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```r
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 3.6.3
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(car)
```

```
## Warning: package 'car' was built under R version 3.6.3
```

```
## Loading required package: carData
## Warning: package 'carData' was built under R version 3.6.3

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.6.3

## -- Attaching packages --------------------------------------------------------------------- tidyverse

## v tibble  3.0.3     v purrr   0.3.4
## v tidyr   1.1.2     v forcats 0.5.0
## v readr   1.4.0

## Warning: package 'tibble' was built under R version 3.6.3

## Warning: package 'tidyr' was built under R version 3.6.3

## Warning: package 'readr' was built under R version 3.6.3

## Warning: package 'purrr' was built under R version 3.6.3

## Warning: package 'forcats' was built under R version 3.6.3

## -- Conflicts ------------------------------------------------------------------------- tidyverse_confli
## x lubridate::as.difftime() masks base::as.difftime()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks stats::filter()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x car::recode()            masks dplyr::recode()
## x lubridate::setdiff()     masks base::setdiff()
## x purrr::some()            masks car::some()
## x lubridate::union()       masks base::union()
```

```r
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.6.3

## Loading required package: Matrix

##
## Attaching package: 'Matrix'
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.0-2
```

```r
data <- read.csv("Data.csv", stringsAsFactors=FALSE)
data$LAB <- as.factor(data$LAB) # convert LAB to be a factor
data[is.na(data)] <- 0 # replace NAs with zero
```

```r
data_model <- data
data_model$X <- NULL # drop identifier column
data_model$LAB <- NULL # drop non-numeric LAB column
data_model <- data_model[1:nrow(data_model),477:ncol(data_model)]
```

A new method is used to eliminate the zero majority of columns by counting the proportion of 0 in each column

```r
a=ncol(data_model) # number of columns
b=nrow(data_model)*0.8 # 80% number of rows
c=c()
# Delete columns with more than 80% zeros
for(i in 1:a){
  # print(sum(data_model[,i]==0)) # Number of 0 per column
  if( sum(data_model[,i]==0)>=b ){
    c=append(c,i)
  }
  }
print(c) # Columns to be deleted
```

```
##   [1]   1   3   4   5  10  12  14  15  16  17  18  19  21  22  23  24  25  26
##  [19]  27  28  38  47  48  55  62  65  66  67  69  71  72  77  78  79  80  82
##  [37]  83  87  93  94  96  97  98  99 100 101 102 103 104 105 106 107 108 109
##  [55] 110 111 112 114 116 117 118 119 122 123 124 125 126 127 128 129 130 132
##  [73] 133 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151
##  [91] 152 153 154 155 156 157 158 160 163 164 165 166 167 168 169 170 171 172
## [109] 173 174 175 176 177 178 180 183 186 187 188 189 190 191 192 193 194 207
## [127] 210 213 214 215 216 217 218 219 220 221 222 230 231 232 233 234 235 236
## [145] 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 261 262
## [163] 267 269 323 324 325 327 332
```

```r
Data_remove=data_model[,-c]
```

Ridge Regression

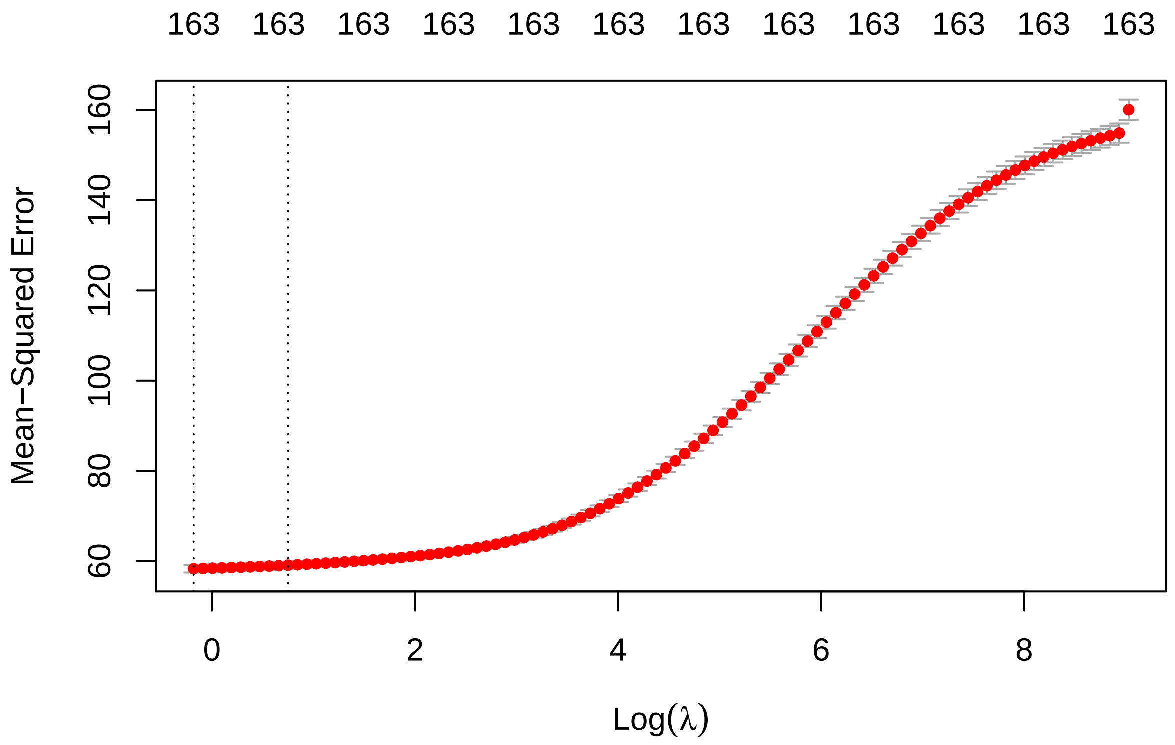
Making std_data_predictors store all attributes that could influence response.

```r
std_data_predictors = data.matrix(select(Data_remove, -c(Response)))
```

Finding the lambda value that creates the lowest mean squared error using k fold cross validation.

```
kfolds_cv_model <-
  cv.glmnet(std_data_predictors, Data_remove$Response, alpha = 0)

plot(kfolds_cv_model)
```



```
bestLambdaVal = kfolds_cv_model$lambda.min
```

Using the best lambda value found, we create a ridge regression model utilising all predictor variables.

```
finalModel <- glmnet(std_data_predictors, Data_remove$Response, alpha = 0, lambda = bestLambdaVal)
```

Predict the response variable using the model for each set of attribute values, then calculate the R^2 value to see the percentage of the variance which is explained by the model.

```
predictedResponse <- predict(finalModel, s = bestLambdaVal, newx = std_data_predictors)

sse <- sum((predictedResponse - Data_remove$Response)^2)
sst <- sum((Data_remove$Response - mean(Data_remove$Response))^2)


rsq <- 1 - sse/sst
rsq
```

```
## [1] 0.6464178
```

Lasso Regression

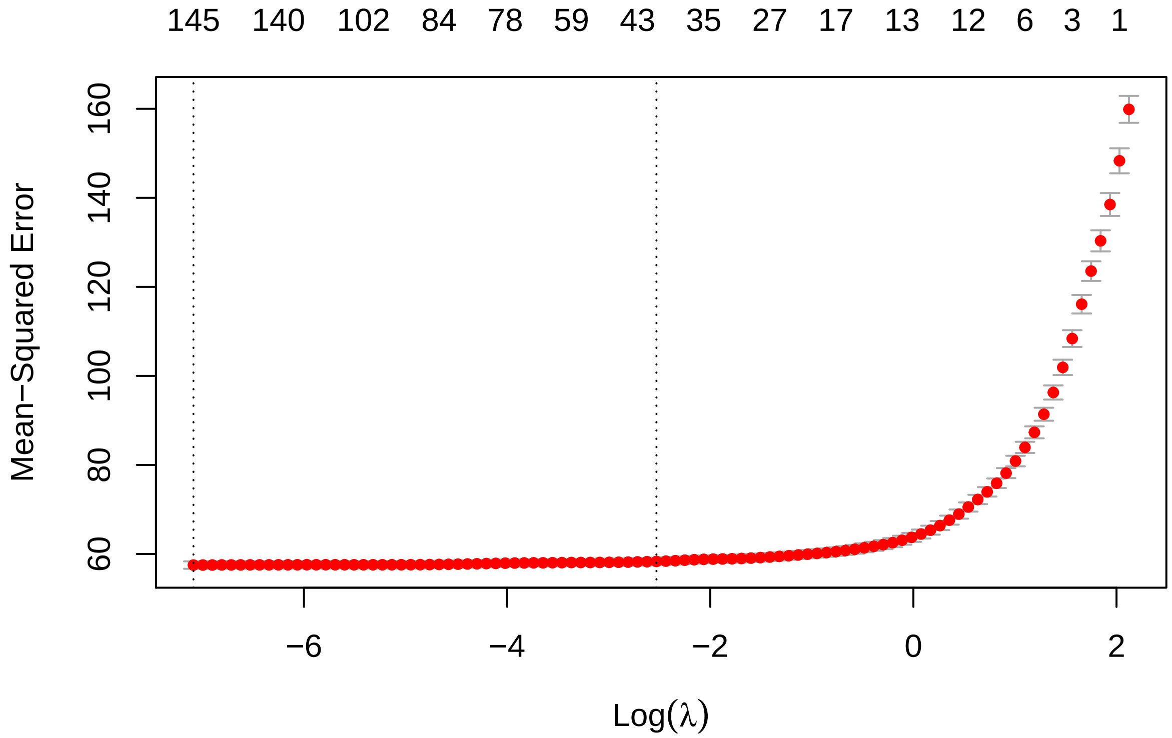
Making std_data_predictors store all attributes that could influence response.

```
std_data_predictors = data.matrix(select(Data_remove, -c("Response")))
```

Finding the lambda value that creates the lowest mean squared error using k fold cross validation.

```
kfolds_cv_model <-
  cv.glmnet(std_data_predictors, Data_remove$Response, alpha = 1)

plot(kfolds_cv_model)
```



```
bestLambdaVal = kfolds_cv_model$lambda.min
```

Using the best lambda value found, we create a lasso regression model utilising all predictor variables.

```
finalModel <- glmnet(std_data_predictors, Data_remove$Response, alpha = 1, lambda = bestLambdaVal)
```

Predict the response variable using the model for each set of attribute values, then calculate the R^2 value to see the percentage of the variance which is explained by the model.

```r
predictedResponse <- predict(finalModel, s = bestLambdaVal, newx = std_data_predictors)

sse <- sum((predictedResponse - Data_remove$Response)^2)
sst <- sum((Data_remove$Response - mean(Data_remove$Response))^2)


rsq <- 1 - sse/sst
rsq
```

```
## [1] 0.6593165
```