Modelling - Ridge & Lasso Regression

```
library(stringr)
```

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

```
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
```

```
library(ggplot2)
```

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

```
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
```

```
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
```

```
library(tidyverse)
```

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

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v tibble  3.0.4     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_conflicts() --
## 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()
```

```
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
```

```
library(MLmetrics)
```

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

```
##
## Attaching package: 'MLmetrics'

## The following object is masked from 'package:base':
##
##     Recall
```

```
data <- read.csv("../data/standardised_data_460.csv", stringsAsFactors=FALSE)
data = select(data, -c(Pass))
```

Ridge Regression

Making train_preds and test_preds store all attributes that could influence response.

```
set.seed(100)
# Divide the data set  into training set and test set, the ratio is 7:3
train_sub = sample(nrow(data),7/10*nrow(data))
train_data = data[train_sub,]
train_preds = data.matrix(select(train_data, -c("Response")))

test_data = data[-train_sub,]
test_preds = data.matrix(select(test_data, -c("Response")))
```

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

```
kfolds_cv_model <-
  cv.glmnet(train_preds, train_data$Response, alpha = 0)

bestLambdaVal = kfolds_cv_model$lambda.min
```

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

```
finalModel <- glmnet(train_preds, train_data$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 = test_preds)

sSE <- sum((predictedResponse - test_data$Response)^2)
sST <- sum((data$Response - mean(data$Response))^2)


rSQ <- 1 - sSE/sST
rSQ
```

```
## [1] 0.8894444
```

Calculating the adjusted R-Squared value:

```
n = dim(data)[1]
k = dim(data)[2]

adrSQ = 1 - ( (1 - rSQ)*(n-1) )/(n-k-1)
adrSQ
```

```
## [1] 0.8864994
```

```
mse = MSE(predictedResponse, test_data$Response)
mse
```

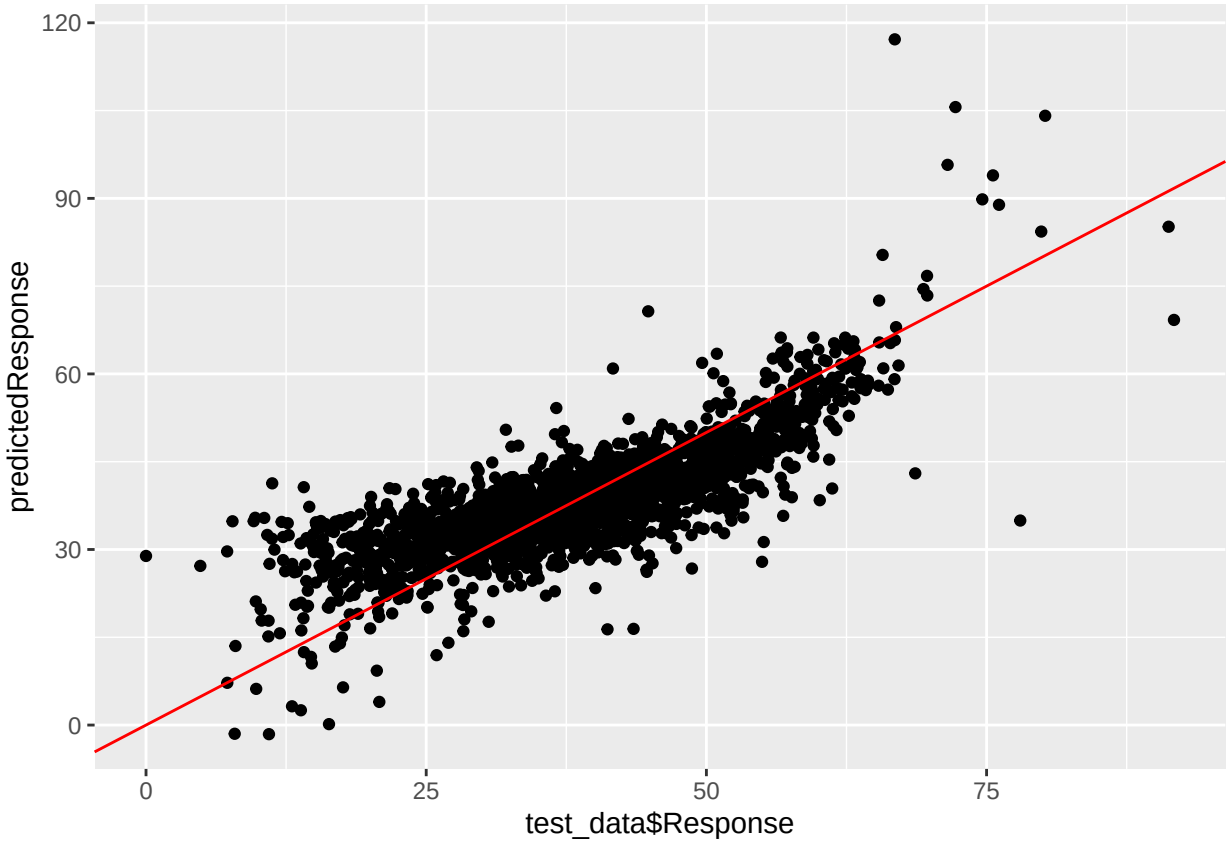
```
## [1] 59.00806
```

```
mae = MAE(predictedResponse, test_data$Response)
mae
```

```
## [1] 5.865087
```

As expected the adjusted R-squared is lower than the normal r-squared value, however the decrease is such that the model is still competitive in its closeness of fit to other models.

```
ggplot() + geom_point(aes(y=predictedResponse, x=test_data$Response)) +
  geom_abline(slope=1, color="red")
```



Lasso Regression

Making std_data_predictors store all attributes that could influence response.

```
data <- read.csv("../data/standardised_data_460.csv", stringsAsFactors=FALSE)
data = select(data, -c(Pass))
set.seed(100)
# Divide the data set  into training set and test set, the ratio is 7:3
train_sub = sample(nrow(data),7/10*nrow(data))
train_data = data[train_sub,]
train_preds = data.matrix(select(train_data, -c("Response")))

test_data = data[-train_sub,]
test_preds = data.matrix(select(test_data, -c("Response")))
```

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

```
kfolds_cv_model <-
  cv.glmnet(train_preds, train_data$Response, alpha = 1)

bestLambdaVal = kfolds_cv_model$lambda.min
```

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

```
finalModel <- glmnet(train_preds, train_data$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.

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

sSE <- sum((predictedResponse - test_data$Response)^2)
sST <- sum((data$Response - mean(data$Response))^2)


rSQ <- 1 - sSE/sST
rSQ
```

```
## [1] 0.8899865
```

Calculating the adjusted R-Squared value:

```
n = dim(data)[1]
k = dim(data)[2]

adrSQ = 1 - ( (1 - rSQ)*(n-1) )/(n-k-1)
adrSQ
```

```
## [1] 0.8870559
```

```
allAttributeAdrSQ = adrSQ
allAttributeMSE = MSE(predictedResponse, test_data$Response)
allAttributeMSE
```

```
## [1] 58.71872
```

```
mae = MAE(predictedResponse, test_data$Response)
mae
```

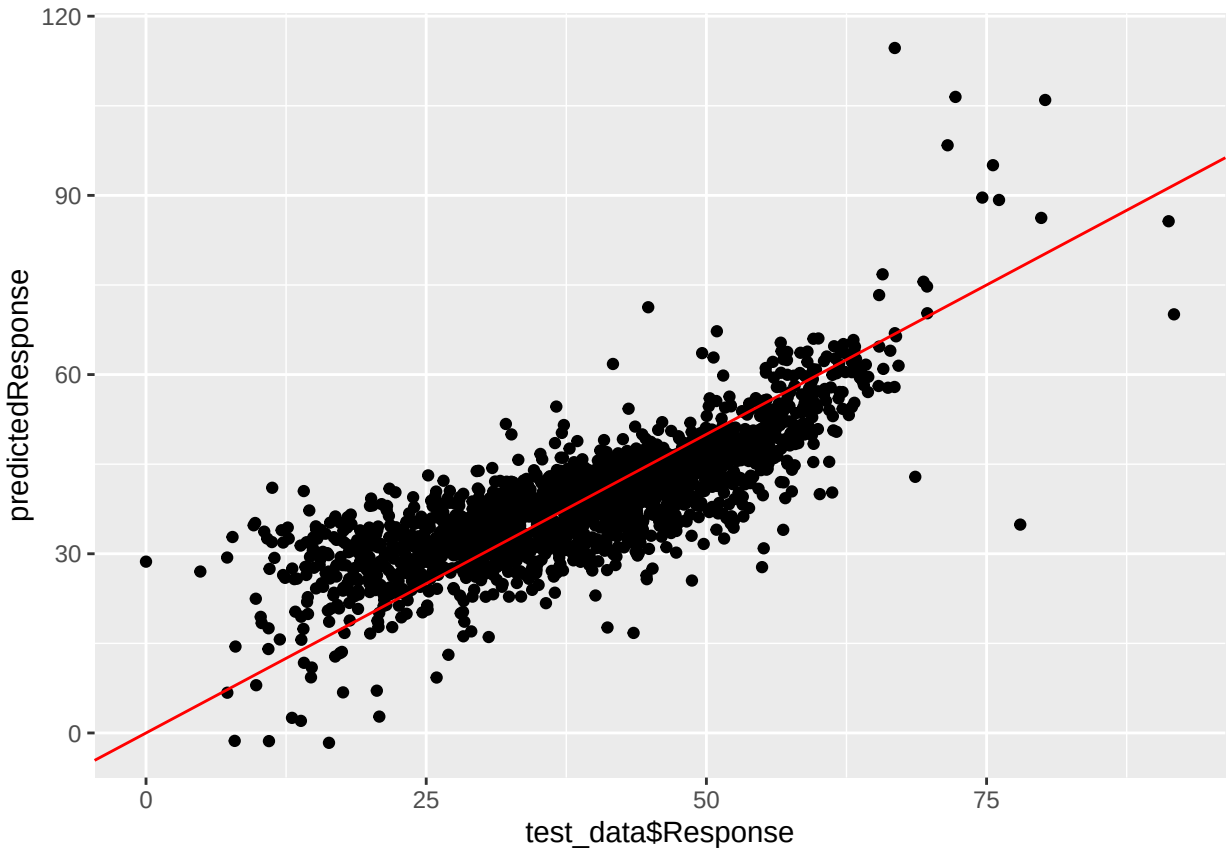
```
## [1] 5.825184
```

The adjusted R-squared value is marginally better than that generated by ridge regression, however the similarity of approach is highlighted by the closeness in quality of fit.

Finding the coefficient values of each attribute in the data, I have taken the highest ten attributes and used them in a new model below.

```
coefs = coef(finalModel)
coefs = as.matrix(coefs)


#coefs = coefs[order(coefs$s0),]

#coefs
#coefs = data.frame(attributes = coefs_names, vals = coefs_vals)
```

```
ggplot() + geom_point(aes(y=predictedResponse, x=test_data$Response)) +
  geom_abline(slope=1, color="red")
```



Lasso Regression - Reduced to 10 Attributes

Making data store the top ten variables that help to explain the variation in the data based on the previous lasso model utilising all attributes of the data.

```
data <- read.csv("../data/standardised_data_460.csv", stringsAsFactors=FALSE)
data = select(data, -c(Pass))
set.seed(100)

data = select(data, c("Group1_11", "Group13_34", "Group2_9", "Group13_5",
                      "Group2_21", "Group2_45", "Group2_18", "Group9_31",
                      "Group8_7", "Group3_9", "Response"))

# Divide the data set  into training set and test set, the ratio is 7:3
train_sub = sample(nrow(data),7/10*nrow(data))
train_data = data[train_sub,]
train_preds = data.matrix(select(train_data, -c("Response")))

test_data = data[-train_sub,]
test_preds = data.matrix(select(test_data, -c("Response")))
```

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

```
kfolds_cv_model <-
  cv.glmnet(train_preds, train_data$Response, alpha = 1)

bestLambdaVal = kfolds_cv_model$lambda.min
```

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

```
finalModel <- glmnet(train_preds, train_data$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.

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

sSE <- sum((predictedResponse - test_data$Response)^2)
sST <- sum((data$Response - mean(data$Response))^2)


rSQ <- 1 - sSE/sST
rSQ
```

```
## [1] 0.876794
```

Calculating the adjusted R-Squared value:

```
n = dim(data)[1]
k = dim(data)[2]

adrSQTEN = 1 - ( (1 - rSQ)*(n-1) )/(n-k-1)
adrSQTEN
```

```
## [1] 0.8765805
```

```
mse = MSE(predictedResponse, test_data$Response)
mse
```

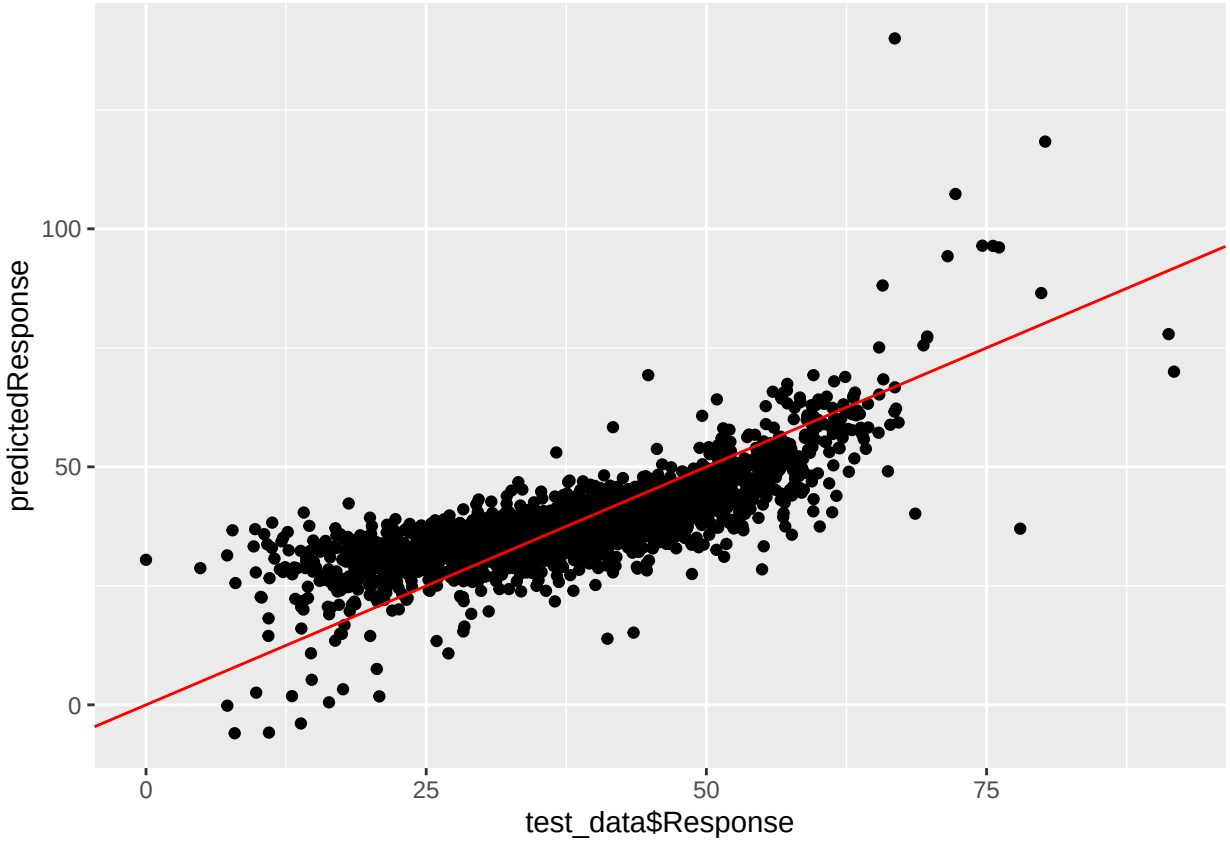
```
## [1] 65.76013
```

```
mae = MAE(predictedResponse, test_data$Response)
mae
```

```
## [1] 6.170414
```

The adjusted r-squared of this model shows that the ten attributes utilised in this model (“Group1_11”, “Group13_34”, “Group2_9”, “Group13_5”, “Group2_21”, “Group2_45”, “Group2_18”, “Group9_31”, “Group8_7”, “Group3_9”) explains 0.8765805 of the variation in the Response variable, this is important to note given that the model using all attributes explained 0.8870559 of the variation.

```
ggplot() + geom_point(aes(y=predictedResponse, x=test_data$Response)) +
  geom_abline(slope=1, color="red")
```



Lasso Regression - Determining Top 10 Variables Impacts

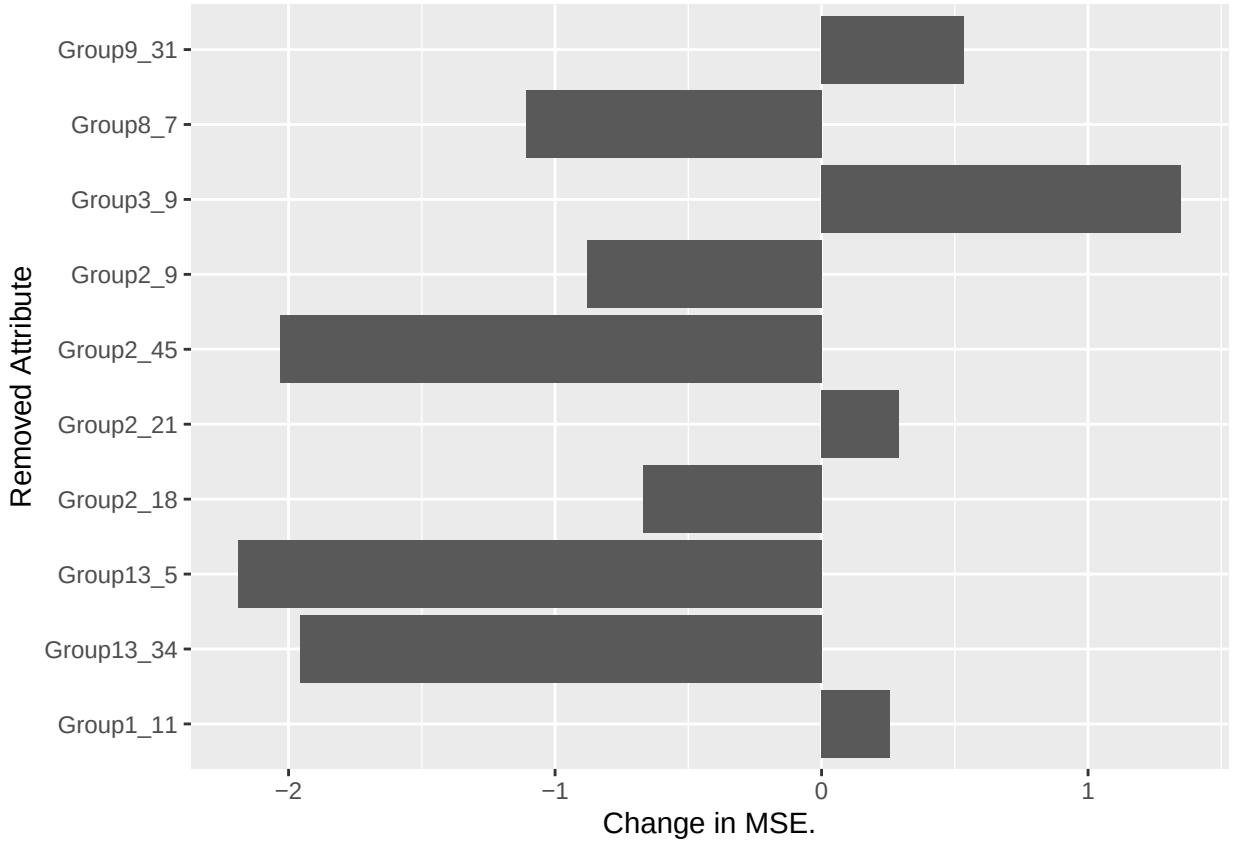
In this section I will use a model containing all attributes, except I will remove one of the previously explored ten attributes to analyses the individual impact of removal on mean squared error (MSE).

```
data <- read.csv("../data/standardised_data_460.csv", stringsAsFactors=FALSE)
data = select(data, -c(Pass))
set.seed(100)

attributeToRemove = c("Group1_11", "Group13_34", "Group2_9", "Group13_5", "Group2_21",
                      "Group2_45", "Group2_18", "Group9_31", "Group8_7", "Group3_9")

changeInRSQ = c()
changeInMSE = c()
changeInMAE = c()

for (i in 1:10){
  newData = select(data, -c(attributeToRemove[i]))

  # Divide the data set  into training set and test set, the ratio is 7:3
  train_sub = sample(nrow(newData),7/10*nrow(newData))
```

```
  train_data = newData[train_sub,]
  train_preds = data.matrix(select(train_data, -c("Response")))

  test_data = newData[-train_sub,]
  test_preds = data.matrix(select(test_data, -c("Response")))

  kfolds_cv_model <-
    cv.glmnet(train_preds, train_data$Response, alpha = 1)

  bestLambdaVal = kfolds_cv_model$lambda.min

  finalModel <- glmnet(train_preds, train_data$Response, alpha = 1,
                       lambda = bestLambdaVal)

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

  sSE <- sum((predictedResponse - test_data$Response)^2)
  sST <- sum((newData$Response - mean(newData$Response))^2)


  rSQ <- 1 - sSE/sST
  rSQ

  n = dim(newData)[1]
  k = dim(newData)[2]

  adrSQ = 1 - ( (1 - rSQ)*(n-1) )/(n-k-1)

  changeInRSQ[i] = adrSQ - allAttributeAdrSQ
  changeInMSE[i] = MSE(predictedResponse, test_data$Response) - allAttributeMSE
  changeInMAE[i] = MAE(predictedResponse, test_data$Response)
}
```

From this graph we can see that the results of removing the ten explored variables on MSE is mixed, with most showing that removing the attribute caused MSE to decrease in comparison to a model utilising all variables. This is intuitive as we would always anticipate more information improves the models closeness of fit, as the coefficient of completely unrelated attributes would be minimised to become unimpactfull.

```
effectOfRemoval = data.frame(name=attributeToRemove,
                             changeRSQ=changeInRSQ,
                             changeMSE=changeInMSE)

ggplot(effectOfRemoval) + geom_col(aes(x=changeMSE, y=name)) +
  labs(y="Removed Attribute", x="Change in MSE.")
```



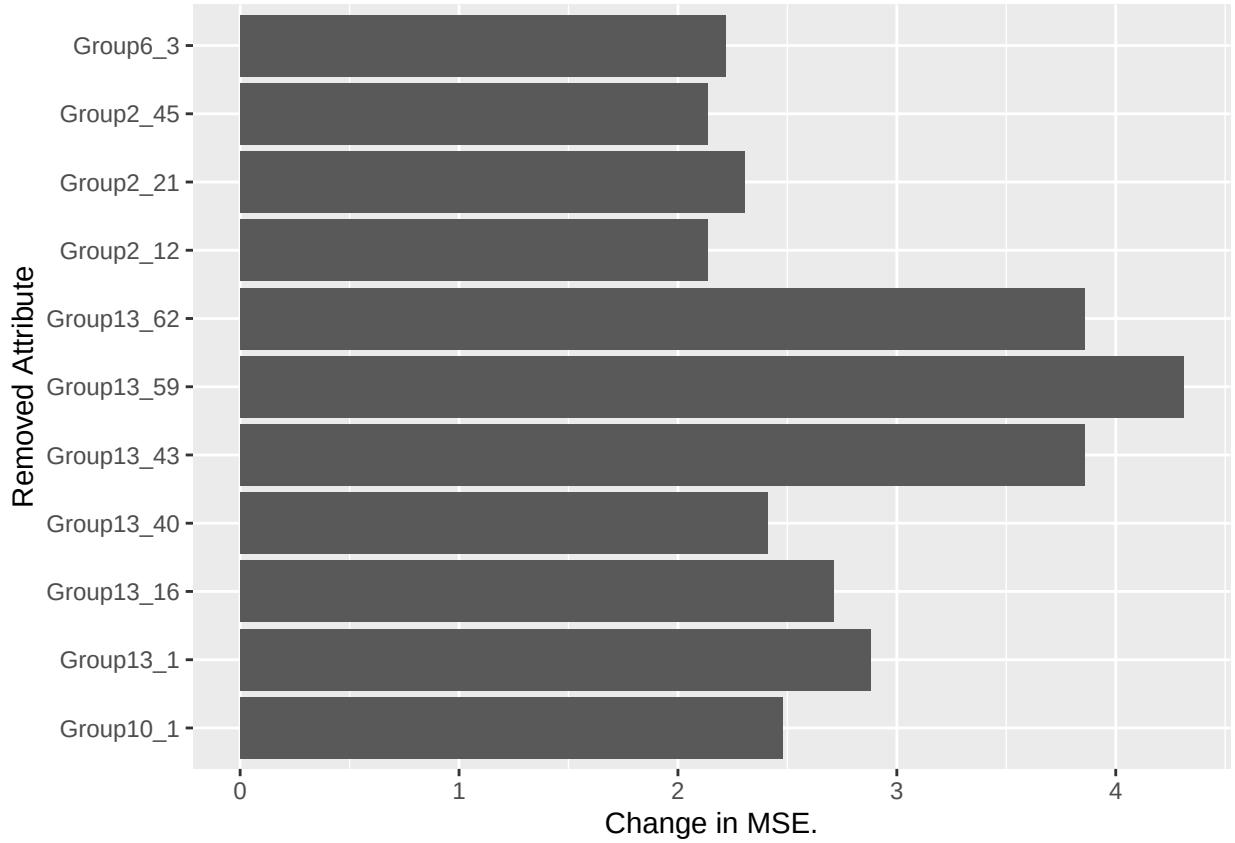
Lasso Regression - Determining The MSE Impact of Each Variable

Here I am recording the effect that removing each variable has from the model, as it may provide insight into how to extract performance from the model by highlighting influential attributes.

```
data <- read.csv("../data/standardised_data_460.csv", stringsAsFactors=FALSE)
data = select(data, -c(Pass))
set.seed(100)

attributeToRemove = names(data)
attributeToRemove = attributeToRemove[attributeToRemove %in% "Response" == FALSE]

changeInRSQ = c()
changeInMSE = c()
changeInMAE = c()

for (i in 1:length(attributeToRemove)){
  newData = select(data, -c(attributeToRemove[i]))

  # Divide the data set  into training set and test set, the ratio is 7:3
  train_sub = sample(nrow(newData),7/10*nrow(newData))
  train_data = newData[train_sub,]
  train_preds = data.matrix(select(train_data, -c("Response")))

  test_data = newData[-train_sub,]
```

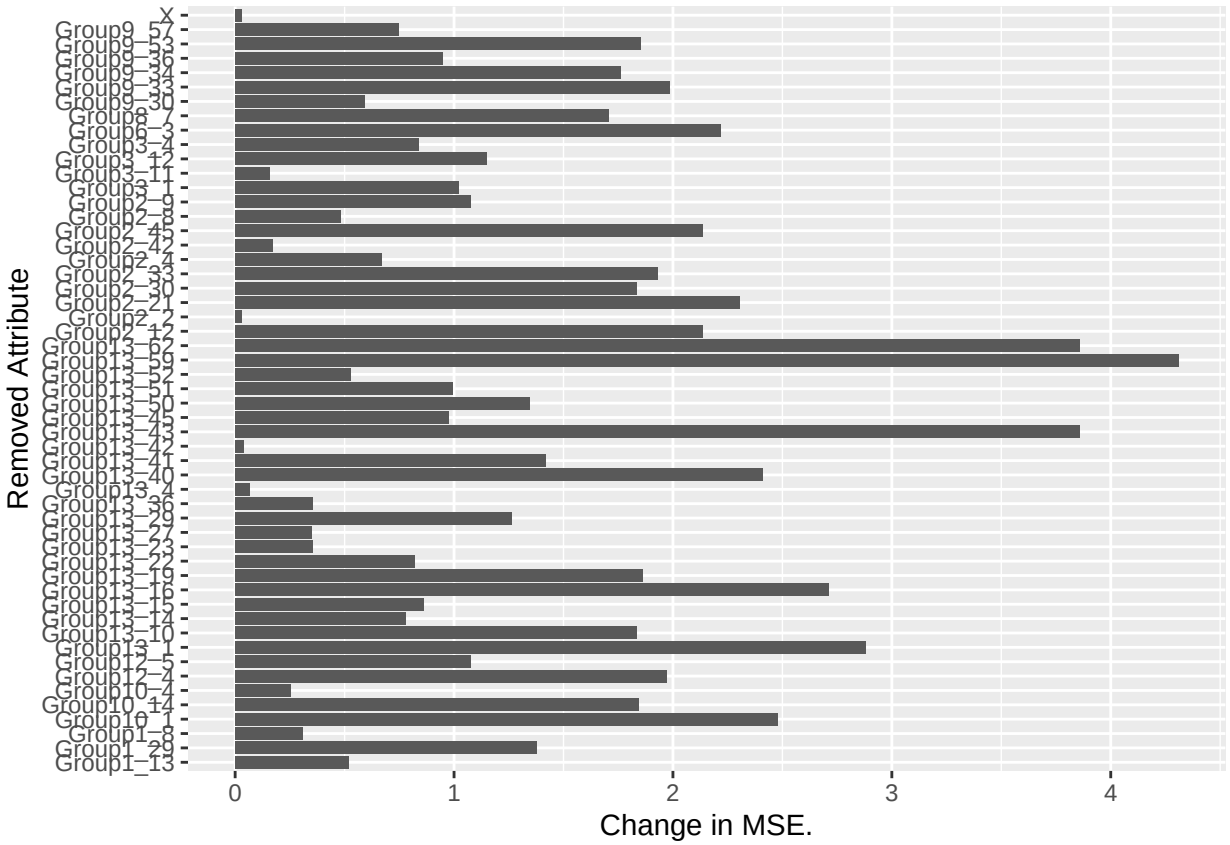
```
  test_preds = data.matrix(select(test_data, -c("Response")))

  kfolds_cv_model <-
    cv.glmnet(train_preds, train_data$Response, alpha = 1)

  bestLambdaVal = kfolds_cv_model$lambda.min

  finalModel <- glmnet(train_preds, train_data$Response, alpha = 1,
                       lambda = bestLambdaVal)

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

  sSE <- sum((predictedResponse - test_data$Response)^2)
  sST <- sum((newData$Response - mean(newData$Response))^2)


  rSQ <- 1 - sSE/sST
  rSQ

  n = dim(newData)[1]
  k = dim(newData)[2]

  adrSQ = 1 - ( (1 - rSQ)*(n-1) )/(n-k-1)

  changeInRSQ[i] = adrSQ - allAttributeAdrSQ
  changeInMSE[i] = MSE(predictedResponse, test_data$Response) - allAttributeMSE
  changeInMAE[i] = MAE(predictedResponse, test_data$Response)
}
```

```
effectOfRemovalOG = data.frame(name=attributeToRemove,
                               changeRSQ=changeInRSQ,
                               changeMSE=changeInMSE)

effectOfRemoval = arrange(effectOfRemovalOG, desc(changeMSE))
effectOfRemoval = filter(effectOfRemoval, changeMSE > 2)
effectOfRemoval = filter(effectOfRemoval, changeMSE < 7.5)

ggplot(effectOfRemoval) + geom_col(aes(x=changeMSE, y=name)) +
  labs(y="Removed Attribute", x="Change in MSE.")
```



Finding the attributes that have a positive effect on MSE:

```
positiveMSE = filter(effectOfRemovalOG, changeMSE > 0)

ggplot(positiveMSE) + geom_col(aes(x=changeMSE, y=name)) +
  labs(y="Removed Attribute", x="Change in MSE.")
```



As the graph shows there are still too many variables (53) to be easily interpretable, so I will attempt to reduce them further while maintaining an explanatory effect over Response.

```
variables = positiveMSE$name

variables = as.character(variables)
variables[length(variables)+1] = "Response"
```

```
data <- read.csv("../data/standardised_data_460.csv", stringsAsFactors=FALSE)
data = select(data, -c(Pass))
set.seed(100)

attributeToRemove = variables
newData = select(data, c(attributeToRemove))
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use ‘all_of(attributeToRemove)‘ instead of ‘attributeToRemove‘ to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
# Divide the data set  into training set and test set, the ratio is 7:3
train_sub = sample(nrow(newData),7/10*nrow(newData))
train_data = newData[train_sub,]
train_preds = data.matrix(select(train_data, -c("Response")))

test_data = newData[-train_sub,]
```

```
test_preds = data.matrix(select(test_data, -c("Response")))

kfolds_cv_model <-
  cv.glmnet(train_preds, train_data$Response, alpha = 1)

bestLambdaVal = kfolds_cv_model$lambda.min

finalModel <- glmnet(train_preds, train_data$Response, alpha = 1, lambda = bestLambdaVal)

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

sSE <- sum((predictedResponse - test_data$Response)^2)
sST <- sum((newData$Response - mean(newData$Response))^2)


rSQ <- 1 - sSE/sST

n = dim(newData)[1]
k = dim(newData)[2]

adrSQ = 1 - ( (1 - rSQ)*(n-1) )/(n-k-1)
adrSQ
```

```
## [1] 0.8860878
```

```
mse = MSE(predictedResponse, test_data$Response)
mse
```

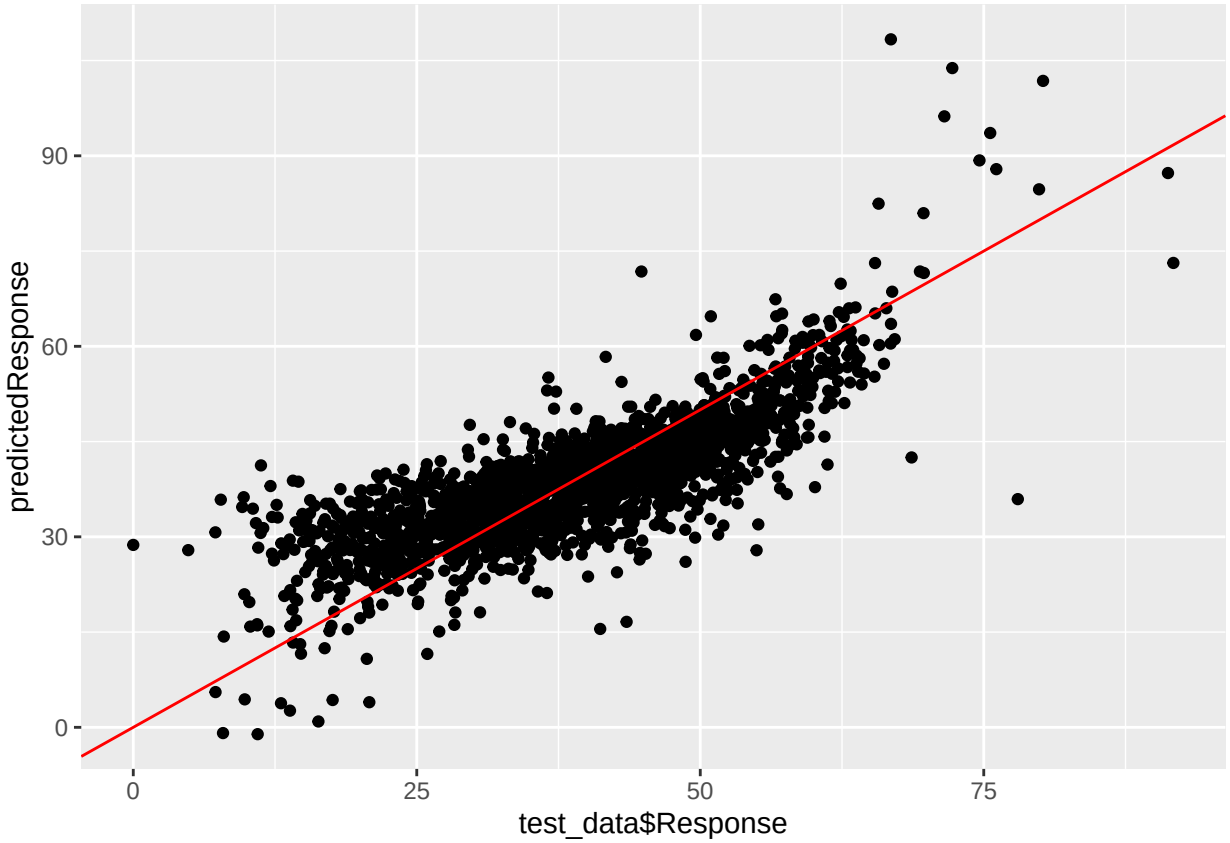
```
## [1] 60.28334
```

```
mae = MAE(predictedResponse, test_data$Response)
mae
```

```
## [1] 5.968986
```

When comparing the above lasso regression model that utilises 43 more variables than the initial 10 variable approach using lasso, the difference between the two models adjusted r-squared values is only 0.0095073. This further highlights the impact that the ten variables ("Group1_11", "Group13_34", "Group2_9", "Group13_5", "Group2_21", "Group2_45", "Group2_18", "Group9_31", "Group8_7", "Group3_9") have on the response variable.

```
ggplot() + geom_point(aes(y=predictedResponse, x=test_data$Response)) +
  geom_abline(slope=1, color="red")
```


predictedResponse
90
60
30
0
0
25
50
75
test_data$Response