# Challenging experiment

Siddharth.S.Chandran (18BCE1003) and Shubham Agarwal(18BCE1184)

23/03/2021

1. Reading the dataset

```
library(readxl)
dat<-read_excel("C:/Users/Siddharth.S.Chandran/Desktop/MyProjects/DataAnalytics/challenging.xlsx")
str(dat)
```

```
## tibble [64 x 7] (S3: tbl_df/tbl/data.frame)
##  $ Sample Number: num [1:64] 1 2 3 4 5 6 7 8 9 10 ...
##  $ X1           : num [1:64] 50 50 50 50 50 50 50 50 50 50 ...
##  $ X2           : num [1:64] 6 6 6 6 8 8 8 8 10 10 ...
##  $ X3           : num [1:64] 30 45 60 75 30 45 60 75 30 45 ...
##  $ Y1           : num [1:64] 9000 8833 7750 7300 8750 ...
##  $ Y2           : num [1:64] 4.86 4.97 5.79 6.28 5.06 ...
##  $ Z1           : num [1:64] 18 19 20 27 19 24 21 23 20 20 ...
```

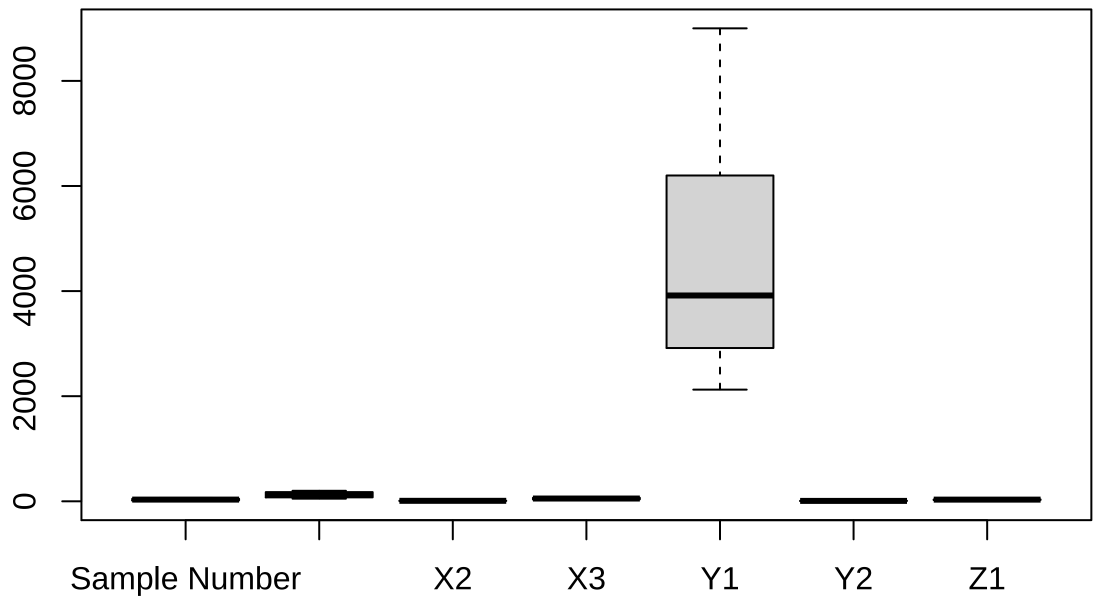
```
summary(dat)
```

```
##  Sample Number         X1              X2              X3              Y1
##  Min.   : 1.00   Min.   : 50.0   Min.   : 6.0   Min.   :30.00   Min.   :2125
##  1st Qu.:16.75   1st Qu.: 87.5   1st Qu.: 7.5   1st Qu.:41.25   1st Qu.:2958
##  Median :32.50   Median :125.0   Median : 9.0   Median :52.50   Median :3916
##  Mean   :32.50   Mean   :125.0   Mean   : 9.0   Mean   :52.50   Mean   :4679
##  3rd Qu.:48.25   3rd Qu.:162.5   3rd Qu.:10.5   3rd Qu.:63.75   3rd Qu.:5850
##  Max.   :64.00   Max.   :200.0   Max.   :12.0   Max.   :75.00   Max.   :9000
##        Y2               Z1
##  Min.   : 4.855   Min.   :18.00
##  1st Qu.: 6.946   1st Qu.:25.75
##  Median : 7.208   Median :32.00
##  Mean   : 7.532   Mean   :30.69
##  3rd Qu.: 8.703   3rd Qu.:35.25
##  Max.   :10.127   Max.   :44.00
```

Normalizing the values

Before normalization

```
boxplot(dat)
```




We can see that the values are higly differing. We therefore have to perform normalization, We apply the standardization technique

```
library(caret)
```

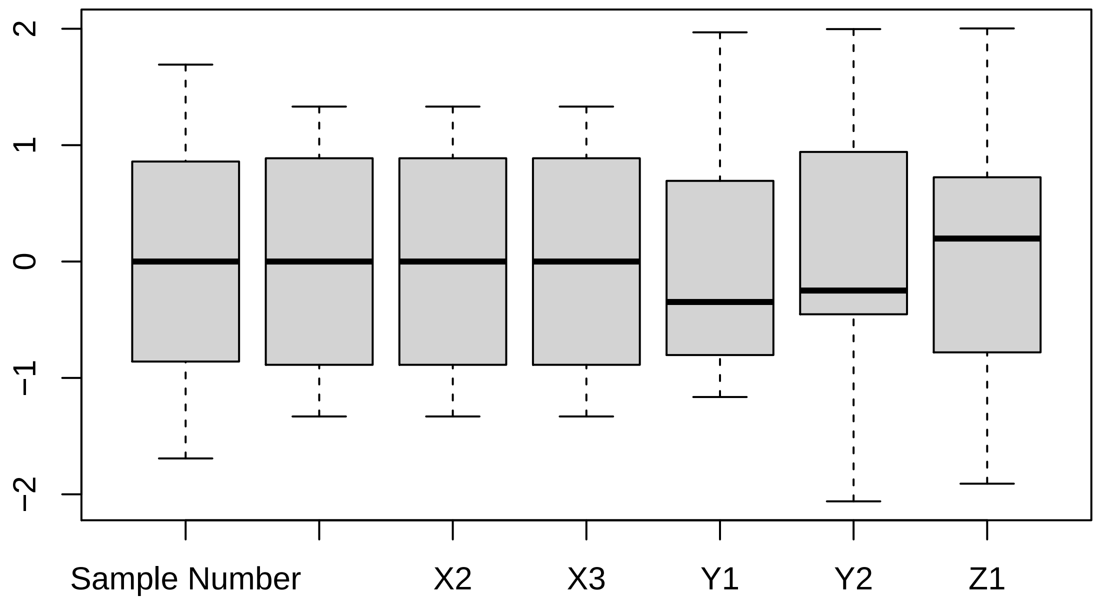
```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
preproc1 <- preProcess(dat, method=c("center", "scale"))
norm1 <- predict(preproc1, dat)
summary(norm1)
```

```
##  Sample Number          X1                 X2                 X3         
##  Min.   :-1.6918   Min.   :-1.3311   Min.   :-1.3311   Min.   :-1.3311  
##  1st Qu.:-0.8459   1st Qu.:-0.6656   1st Qu.:-0.6656   1st Qu.:-0.6656  
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000  
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.8459   3rd Qu.: 0.6656   3rd Qu.: 0.6656   3rd Qu.: 0.6656  
##  Max.   : 1.6918   Max.   : 1.3311   Max.   : 1.3311   Max.   : 1.3311  
##        Y1                 Y2                 Z1         
##  Min.   :-1.1642   Min.   :-2.0604   Min.   :-1.9087  
##  1st Qu.:-0.7844   1st Qu.:-0.4510   1st Qu.:-0.7428  
##  Median :-0.3477   Median :-0.2492   Median : 0.1974  
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
```

```
##   3rd Qu.: 0.5336   3rd Qu.: 0.9010   3rd Qu.: 0.6864
##   Max.   : 1.9694   Max.   : 1.9971   Max.   : 2.0027
```

```
boxplot(norm1)
```




After the standardization process we can see that each and every variable has not much differences in their values, and is hencce normalizaed

2. Stage1 (inputs -> x1, x2, x3 and outputs -> y1, y2)

a. Correlation between the input variables

```
data1<-norm1[,c(2,3,4)]
cor(data1)
```

```
##    X1 X2 X3
## X1  1  0  0
## X2  0  1  0
## X3  0  0  1
```

The input variables have 0 correlation between them

b.

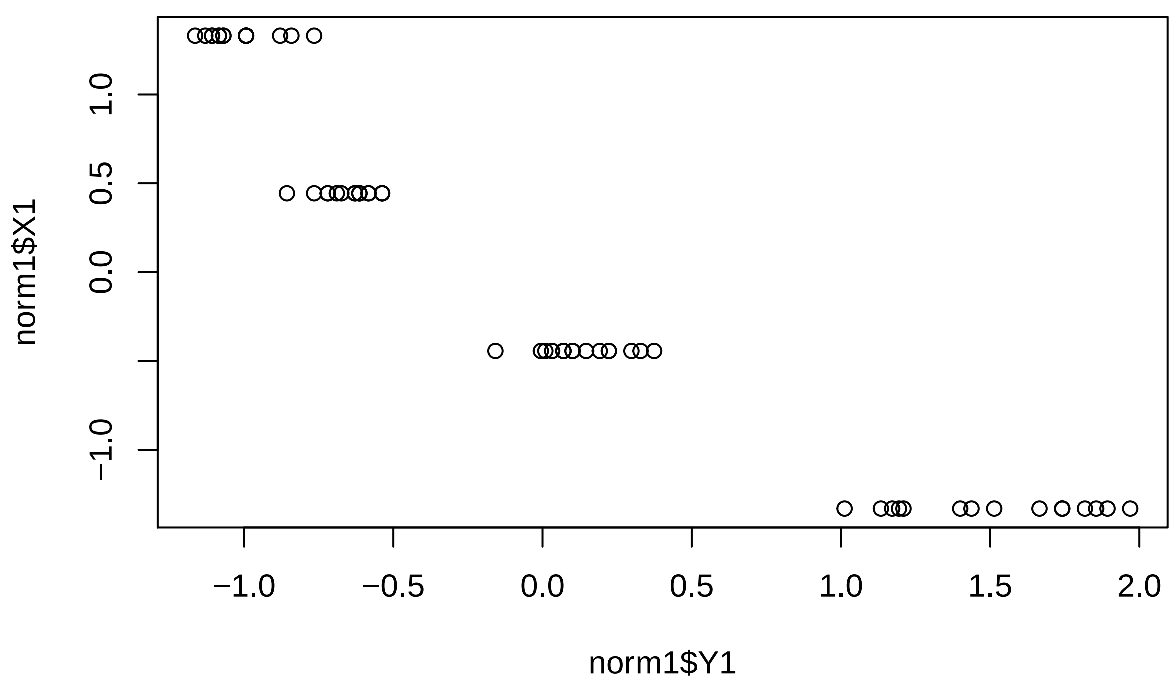
```r
data1<-norm1[,c(2,3,4,5,6)]
cor(data1)
```

```
##            X1          X2         X3          Y1         Y2
## X1  1.0000000  0.00000000  0.0000000 -0.94851220  0.3747101
## X2  0.0000000  1.00000000  0.0000000 -0.05629901  0.3920906
## X3  0.0000000  0.00000000  1.0000000 -0.12093791  0.6623520
## Y1 -0.9485122 -0.05629901 -0.1209379  1.00000000 -0.5223024
## Y2  0.3747101  0.39209061  0.6623520 -0.52230240  1.0000000
```

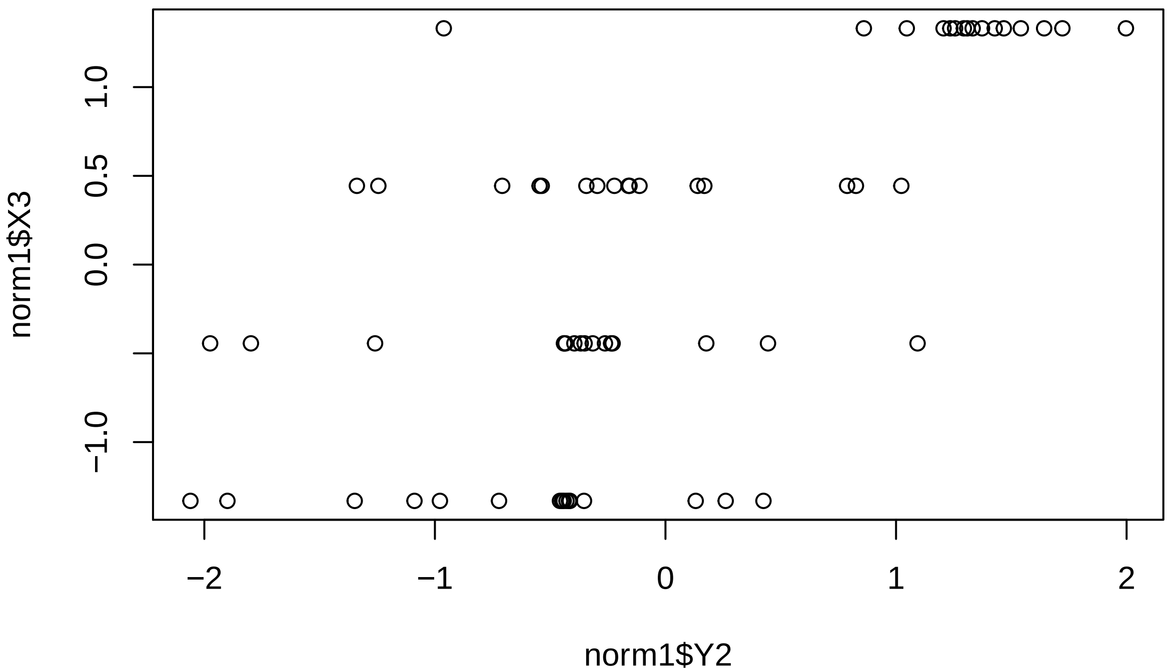
c. Y1 and X1 have the strongest correlation of -0.9 Y2 and X3 follows that with a correlation of 0.66

d. Graphical inferences

```r
plot(norm1$Y1, norm1$X1)
```




```r
plot(norm1$Y2, norm1$X3)
```




3. Stage2 classifier (y2 input and z1 output ) a. Correlation between input variables

```
data1<-norm1[,c(5,6)]
cor(data1)
```

```
##             Y1         Y2
## Y1  1.0000000 -0.5223024
## Y2 -0.5223024  1.0000000
```

The input variables have a good amount of correlation of -0.5

b. Correlation between input and output variables

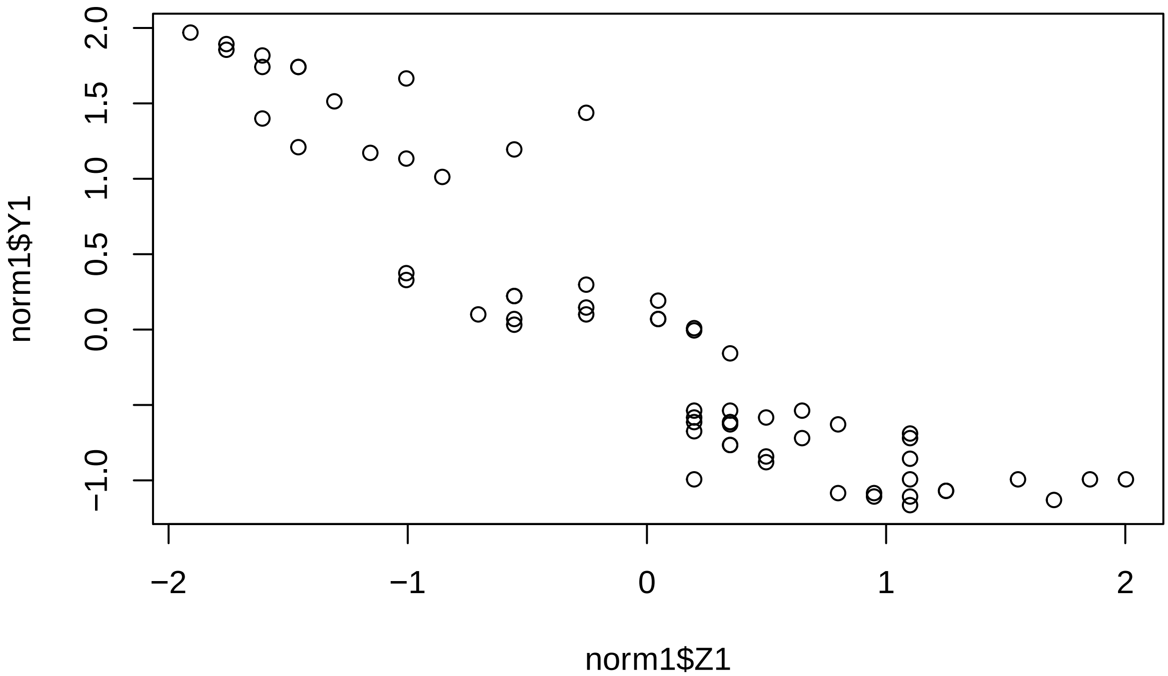
```
data1<-norm1[,c(5,6,7)]
cor(data1)
```

```
##             Y1         Y2         Z1
## Y1  1.0000000 -0.5223024 -0.9200719
## Y2 -0.5223024  1.0000000  0.5538196
## Z1 -0.9200719  0.5538196  1.0000000
```

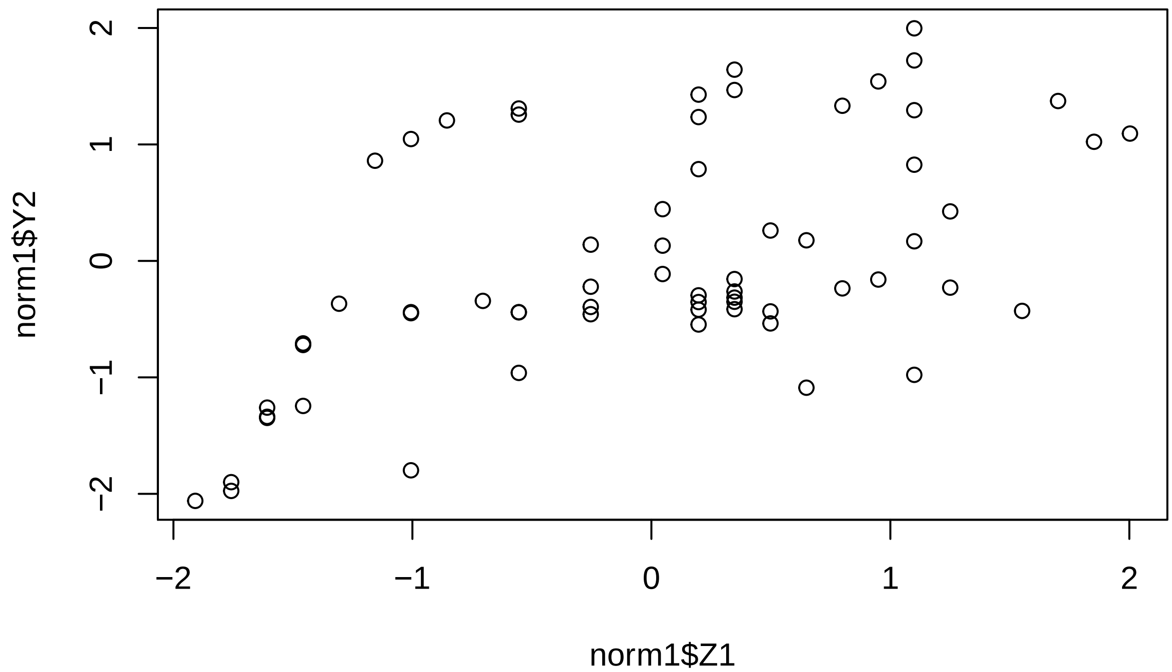
c. Z1 and Y1 have the strongest correlation of -0.92. Z1 and Y2 have a good correlation of 0.5

d. Graphical inference

```
plot(norm1$Z1, norm1$Y1)
```




```
plot(norm1$Z1, norm1$Y2)
```




4. Applying suitable regression techniques

a. Linear regression

Scaling the data Dividing it into testing and training

```
set.seed(100)

index = sample(1:nrow(dat), 0.7*nrow(dat))

train = dat[index,] # Create the training data
test = dat[-index,] # Create the test data

dim(train)
```

```
## [1] 44  7
```

```
dim(test)
```

```
## [1] 20  7
```

```r
pre_proc_val <- preProcess(train, method = c("center", "scale"))

train = predict(pre_proc_val, train)
test = predict(pre_proc_val, test)

summary(train)
```

```
##  Sample Number            X1                X2                X3
##  Min.   :-1.59711   Min.   :-1.2245   Min.   :-1.2882   Min.   :-1.4578
##  1st Qu.:-0.91834   1st Qu.:-1.2245   1st Qu.:-0.5797   1st Qu.:-0.5669
##  Median :-0.02662   Median :-0.3693   Median :-0.3435   Median : 0.3239
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.91834   3rd Qu.: 0.6997   3rd Qu.: 0.6012   3rd Qu.: 1.2148
##  Max.   : 1.70358   Max.   : 1.3411   Max.   : 1.5459   Max.   : 1.2148
##        Y1                Y2                Z1
##  Min.   :-1.1528   Min.   :-1.9881   Min.   :-1.7994
##  1st Qu.:-0.8055   1st Qu.:-0.4925   1st Qu.:-0.8963
##  Median :-0.1364   Median :-0.2238   Median : 0.2326
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.0123   3rd Qu.: 1.0489   3rd Qu.: 0.6466
##  Max.   : 1.7990   Max.   : 1.7299   Max.   : 1.9636
```

```r
cols = c('Y1', 'X1','X2', 'X3')

pre_proc_val <- preProcess(train[,cols], method = c("center", "scale"))

train[,cols] = predict(pre_proc_val, train[,cols])
test[,cols] = predict(pre_proc_val, test[,cols])

summary(train)
```

```
##  Sample Number            X1                X2                X3
##  Min.   :-1.59711   Min.   :-1.2245   Min.   :-1.2882   Min.   :-1.4578
##  1st Qu.:-0.91834   1st Qu.:-1.2245   1st Qu.:-0.5797   1st Qu.:-0.5669
##  Median :-0.02662   Median :-0.3693   Median :-0.3435   Median : 0.3239
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.91834   3rd Qu.: 0.6997   3rd Qu.: 0.6012   3rd Qu.: 1.2148
##  Max.   : 1.70358   Max.   : 1.3411   Max.   : 1.5459   Max.   : 1.2148
##        Y1                Y2                Z1
##  Min.   :-1.1528   Min.   :-1.9881   Min.   :-1.7994
##  1st Qu.:-0.8055   1st Qu.:-0.4925   1st Qu.:-0.8963
##  Median :-0.1364   Median :-0.2238   Median : 0.2326
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.0123   3rd Qu.: 1.0489   3rd Qu.: 0.6466
##  Max.   : 1.7990   Max.   : 1.7299   Max.   : 1.9636
```

```r
lr1 = lm(Y1 ~ X1 + X2 + X3, data = train)
summary(lr1)
```

```
##
## Call:
```

```
## lm(formula = Y1 ~ X1 + X2 + X3, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37827 -0.25243 -0.02042  0.25999  0.49129
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.276e-16  4.298e-02   0.000  1.00000
## X1          -9.455e-01  4.405e-02 -21.466  < 2e-16 ***
## X2          -4.172e-02  4.395e-02  -0.949  0.34819
## X3          -1.299e-01  4.374e-02  -2.969  0.00503 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2851 on 40 degrees of freedom
## Multiple R-squared:  0.9244, Adjusted R-squared:  0.9187
## F-statistic:   163 on 3 and 40 DF,  p-value: < 2.2e-16
```

```r
lr1
```

```
##
## Call:
## lm(formula = Y1 ~ X1 + X2 + X3, data = train)
##
## Coefficients:
## (Intercept)           X1           X2           X3
##   1.276e-16   -9.455e-01   -4.172e-02   -1.299e-01
```

```r
lr2 = lm(Y2 ~ X1 + X2 + X3, data = train)
summary(lr2)
```

```
##
## Call:
## lm(formula = Y2 ~ X1 + X2 + X3, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.86377 -0.49254  0.00017  0.45096  1.03215
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.292e-16  8.474e-02   0.000 1.000000
## X1           3.437e-01  8.684e-02   3.958 0.000302 ***
## X2           3.395e-01  8.665e-02   3.918 0.000340 ***
## X3           6.623e-01  8.624e-02   7.680 2.16e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5621 on 40 degrees of freedom
## Multiple R-squared:  0.7061, Adjusted R-squared:  0.6841
## F-statistic: 32.03 on 3 and 40 DF,  p-value: 1.009e-10
```

```r
lr2
```

```
##
## Call:
## lm(formula = Y2 ~ X1 + X2 + X3, data = train)
##
## Coefficients:
## (Intercept)           X1           X2           X3
##  -1.292e-16    3.437e-01    3.395e-01    6.623e-01
```

```r
#Step 1 - create the evaluation metrics function

eval_metrics = function(model, df, predictions, target){
    resids = df[,target] - predictions
    resids2 = resids**2
    N = length(predictions)
    r2 = as.character(round(summary(model)$r.squared, 2))
    adj_r2 = as.character(round(summary(model)$adj.r.squared, 2))
    print(adj_r2) #Adjusted R-squared
    print(as.character(round(sqrt(sum(resids2)/N), 2))) #RMSE
}

# Step 2 - predicting and evaluating the model on train data
predictions = predict(lr1, newdata = train)
eval_metrics(lr1, train, predictions, target = 'Y1')
```

```
## [1] "0.92"
## [1] "0.27"
```

```r
# Step 3 - predicting and evaluating the model on test data
predictions = predict(lr1, newdata = test)
eval_metrics(lr1, test, predictions, target = 'Y1')
```

```
## [1] "0.92"
## [1] "0.28"
```

```r
d<-predictions - test[,c('Y1')]

mse = mean((d[,c('Y1')])^2)
mse
```

```
## [1] 0.07765067
```

```r
mae = mean(abs(d[,c('Y1')]))
mae
```

```
## [1] 0.260944
```

Stage 1 classifier on y1 (linear regression)

RMSE -> 0.92 R squared -> 0.27 MSE -> 0.077 MAE -> 0.26

```r
#Step 1 - create the evaluation metrics function

eval_metrics = function(model, df, predictions, target){
    resids = df[,target] - predictions
    resids2 = resids**2
    N = length(predictions)
    r2 = as.character(round(summary(model)$r.squared, 2))
    adj_r2 = as.character(round(summary(model)$adj.r.squared, 2))
    print(adj_r2) #Adjusted R-squared
    print(as.character(round(sqrt(sum(resids2)/N), 2))) #RMSE
}

# Step 2 - predicting and evaluating the model on train data
predictions = predict(lr2, newdata = train)
eval_metrics(lr1, train, predictions, target = 'Y2')
```

```
## [1] "0.92"
## [1] "0.54"
```

```r
# Step 3 - predicting and evaluating the model on test data
predictions = predict(lr2, newdata = test)
eval_metrics(lr2, test, predictions, target = 'Y2')
```

```
## [1] "0.68"
## [1] "0.44"
```

```r
d<-predictions - test[,c('Y2')]

mse = mean((d[,c('Y2')])^2)
mse
```

```
## [1] 0.1975769
```

```r
mae = mean(abs(d[,c('Y2')]))
mae
```

```
## [1] 0.3403203
```

The values for Stage 1 classifier on y2 (linear regression are) 1. RMSE -> 0.68 2. R sqaured -> 0.44 3. MSE -> 0.197 4. MAE -> 0.34

Stage 2 classifier

```r
cols = c('Y1', 'Y2', 'Z1')

pre_proc_val <- preProcess(train[,cols], method = c("center", "scale"))

train[,cols] = predict(pre_proc_val, train[,cols])
test[,cols] = predict(pre_proc_val, test[,cols])

summary(train)
```

```
##  Sample Number            X1                 X2                 X3
##  Min.   :-1.59711   Min.   :-1.2245   Min.   :-1.2882   Min.   :-1.4578
##  1st Qu.:-0.91834   1st Qu.:-1.2245   1st Qu.:-0.5797   1st Qu.:-0.5669
##  Median :-0.02662   Median :-0.3693   Median :-0.3435   Median : 0.3239
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.91834   3rd Qu.: 0.6997   3rd Qu.: 0.6012   3rd Qu.: 1.2148
##  Max.   : 1.70358   Max.   : 1.3411   Max.   : 1.5459   Max.   : 1.2148
##        Y1                 Y2                 Z1
##  Min.   :-1.1528   Min.   :-1.9881   Min.   :-1.7994
##  1st Qu.:-0.8055   1st Qu.:-0.4925   1st Qu.:-0.8963
##  Median :-0.1364   Median :-0.2238   Median : 0.2326
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.0123   3rd Qu.: 1.0489   3rd Qu.: 0.6466
##  Max.   : 1.7990   Max.   : 1.7299   Max.   : 1.9636
```

```
lr = lm(Z1 ~ Y1 + Y2, data = train)
lr
```

```
##
## Call:
## lm(formula = Z1 ~ Y1 + Y2, data = train)
##
## Coefficients:
## (Intercept)           Y1           Y2
##   9.662e-17   -9.086e-01    5.355e-02
```

```
summary(lr)
```

```
##
## Call:
## lm(formula = Z1 ~ Y1 + Y2, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.68824 -0.22107 -0.01778  0.15522  0.97867
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.662e-17  5.383e-02   0.000    1.000
## Y1          -9.086e-01  6.346e-02 -14.318   <2e-16 ***
## Y2           5.355e-02  6.346e-02   0.844    0.404
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3571 on 41 degrees of freedom
## Multiple R-squared:  0.8784, Adjusted R-squared:  0.8725
## F-statistic: 148.1 on 2 and 41 DF,  p-value: < 2.2e-16
```

```
#Step 1 - create the evaluation metrics function

eval_metrics = function(model, df, predictions, target){
    resids = df[,target] - predictions
```

```
    resids2 = resids**2
    N = length(predictions)
    r2 = as.character(round(summary(model)$r.squared, 2))
    adj_r2 = as.character(round(summary(model)$adj.r.squared, 2))
    print(adj_r2) #Adjusted R-squared
    print(as.character(round(sqrt(sum(resids2)/N), 2))) #RMSE
}

# Step 2 - predicting and evaluating the model on train data
predictions = predict(lr, newdata = train)
eval_metrics(lr, train, predictions, target = 'Z1')
```

```
## [1] "0.87"
## [1] "0.34"
```

```
# Step 3 - predicting and evaluating the model on test data
predictions = predict(lr, newdata = test)
eval_metrics(lr, test, predictions, target = 'Z1')
```

```
## [1] "0.87"
## [1] "0.46"
```

```
d<-predictions - test[,c('Z1')]

mse = mean((d[,c('Z1')])^2)
mse
```

```
## [1] 0.2118346
```

```
mae = mean(abs(d[,c('Z1')]))
mae
```

```
## [1] 0.3390398
```

The values for stage 2 classifier on linear regression are 1. RMSE -> 0.87 2. R squared -> 0.46 3. MSE -> 0.211 4. MAE -> 0.33

b. Ridge regression

Dividing the data into training and testing

```
set.seed(100)

index = sample(1:nrow(dat), 0.7*nrow(dat))

train = dat[index,] # Create the training data
test = dat[-index,] # Create the test data

dim(train)
```

```
## [1] 44  7
```

```r
dim(test)
```

```
## [1] 20  7
```

Scaling the data

```r
pre_proc_val <- preProcess(train, method = c("center", "scale"))

train = predict(pre_proc_val, train)
test = predict(pre_proc_val, test)

summary(train)
```

```
##  Sample Number            X1                X2                X3         
##  Min.   :-1.59711   Min.   :-1.2245   Min.   :-1.2882   Min.   :-1.4578  
##  1st Qu.:-0.91834   1st Qu.:-1.2245   1st Qu.:-0.5797   1st Qu.:-0.5669  
##  Median :-0.02662   Median :-0.3693   Median :-0.3435   Median : 0.3239  
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 0.91834   3rd Qu.: 0.6997   3rd Qu.: 0.6012   3rd Qu.: 1.2148  
##  Max.   : 1.70358   Max.   : 1.3411   Max.   : 1.5459   Max.   : 1.2148  
##        Y1                Y2                Z1         
##  Min.   :-1.1528   Min.   :-1.9881   Min.   :-1.7994  
##  1st Qu.:-0.8055   1st Qu.:-0.4925   1st Qu.:-0.8963  
##  Median :-0.1364   Median :-0.2238   Median : 0.2326  
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000  
##  3rd Qu.: 1.0123   3rd Qu.: 1.0489   3rd Qu.: 0.6466  
##  Max.   : 1.7990   Max.   : 1.7299   Max.   : 1.9636
```

Regularization

Linear regression works with independent variables which minimizes the loss function, however if the data values are large enough it leads to overfitting of the data and such a model cannot be used for generalization

```r
dummies <- dummyVars(Y1 ~ ., data = dat[,c(1:5)])

train_dummies = predict(dummies, newdata = train[,c(1:5)])

test_dummies = predict(dummies, newdata = test[,c(1:5)])

print(dim(train_dummies)); print(dim(test_dummies))
```

```
## [1] 44  4
```

```
## [1] 20  4
```

Applying ridge regression on stage 1 classifier

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-1
```

```r
x = as.matrix(train_dummies)
y_train = train$Y1

x_test = as.matrix(test_dummies)
y_test = test$Y1

lambdas <- 10^seq(2, -3, by = -.1)
ridge_reg = glmnet(x, y_train, nlambda = 25, alpha = 0, family = 'gaussian', lambda = lambdas)

summary(ridge_reg)
```

```
##           Length Class     Mode
## a0         51    -none-    numeric
## beta      204    dgCMatrix S4
## df         51    -none-    numeric
## dim         2    -none-    numeric
## lambda     51    -none-    numeric
## dev.ratio  51    -none-    numeric
## nulldev     1    -none-    numeric
## npasses     1    -none-    numeric
## jerr        1    -none-    numeric
## offset      1    -none-    logical
## call        7    -none-    call
## nobs        1    -none-    numeric
```

This code runs for several values of lambda. Let us find the optimal lambda

```r
cv_ridge <- cv.glmnet(x, y_train, alpha = 0, lambda = lambdas)
optimal_lambda <- cv_ridge$lambda.min
optimal_lambda
```

```
## [1] 0.02511886
```

The optimal lambda comes to be 0.025

Evaluating the model results based on the lambda value

```r
# Compute R^2 from true and predicted values
eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))


  # Model performance metrics
data.frame(
  RMSE = RMSE,
  Rsquare = R_square
)

}
```

```r
# Prediction and evaluation on train data
predictions_train <- predict(ridge_reg, s = optimal_lambda, newx = x)
eval_results(y_train, predictions_train, train)
```

```
##        RMSE   Rsquare
## 1 0.2720975 0.9242411
```

```r
# Prediction and evaluation on test data
predictions_test <- predict(ridge_reg, s = optimal_lambda, newx = x_test)
eval_results(y_test, predictions_test, test)
```

```
##        RMSE  Rsquare
## 1 0.2777209 0.884149
```

```r
d<-predictions_test - test[,c('Y1')]

mse = mean((d[,c('Y1')])^2)
mse
```

```
## [1] 0.07712892
```

```r
mae = mean(abs(d[,c('Y1')]))
mae
```

```
## [1] 0.2595318
```

For testing dataset in stage 1 classifier using ridge regression the vaules are

1. RMSE -> 0.2777
2. R squared -> 0.884
3. MSE -> 0.077
4. MAE -> 0.259

Stage 1 classifier for Y2 generalization

```r
dummies <- dummyVars(Y2 ~ ., data = dat[,c('X1', 'X2', 'X3', 'Y2')])

train_dummies = predict(dummies, newdata = train[,c('X1', 'X2', 'X3', 'Y2')])

test_dummies = predict(dummies, newdata = test[,c('X1', 'X2', 'X3', 'Y2')])

print(dim(train_dummies)); print(dim(test_dummies))
```

```
## [1] 44  3
```

```
## [1] 20  3
```

Applying ridge regression on stage 1 classifier

```
library(glmnet)

x = as.matrix(train_dummies)
y_train = train$Y2

x_test = as.matrix(test_dummies)
y_test = test$Y2

lambdas <- 10^seq(2, -3, by = -.1)
ridge_reg = glmnet(x, y_train, nlambda = 25, alpha = 0, family = 'gaussian', lambda = lambdas)

summary(ridge_reg)
```

```
##           Length Class     Mode
## a0         51    -none-    numeric
## beta      153    dgCMatrix S4
## df         51    -none-    numeric
## dim         2    -none-    numeric
## lambda     51    -none-    numeric
## dev.ratio  51    -none-    numeric
## nulldev     1    -none-    numeric
## npasses     1    -none-    numeric
## jerr        1    -none-    numeric
## offset      1    -none-    logical
## call        7    -none-    call
## nobs        1    -none-    numeric
```

This code runs for several values of lambda. Let us find the optimal lambda

```
cv_ridge <- cv.glmnet(x, y_train, alpha = 0, lambda = lambdas)
optimal_lambda <- cv_ridge$lambda.min
optimal_lambda
```

```
## [1] 0.05011872
```

The optimal lambda comes to be 0.039

Evaluating the model results based on the lambda value

```
# Compute R^2 from true and predicted values
eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))


  # Model performance metrics
data.frame(
  RMSE = RMSE,
  Rsquare = R_square
)
```

```r
}

# Prediction and evaluation on train data
predictions_train <- predict(ridge_reg, s = optimal_lambda, newx = x)
eval_results(y_train, predictions_train, train)
```

```
##        RMSE   Rsquare
## 1 0.5372944 0.7046012
```

```r
# Prediction and evaluation on test data
predictions_test <- predict(ridge_reg, s = optimal_lambda, newx = x_test)
eval_results(y_test, predictions_test, test)
```

```
##        RMSE   Rsquare
## 1 0.4500715 0.7709925
```

```r
d<-predictions_test - test[,c('Y2')]
d
```

```
##             Y2
## 1   0.66660581
## 2   0.39623980
## 3   0.12785126
## 4  -1.05503122
## 5  -0.03178435
## 6   0.06547289
## 7  -0.59993421
## 8  -0.45539176
## 9  -0.22933476
## 10 -0.83533297
## 11  0.15896329
## 12 -0.19506018
## 13  0.21507350
## 14 -0.55837111
## 15  0.01322360
## 16  0.05351780
## 17  0.09558760
## 18 -0.43942767
## 19 -0.53308654
## 20 -0.29444774
```

```r
mse = mean((d[,c('Y2')])^2)
mse
```

```
## [1] 0.2025644
```

```r
mae = mean(abs(d[,c('Y2')]))
mae
```

```
## [1] 0.3509869
```

The values for stage 1 classifier on y2 are (ridge regression) RMSE -> 0.45 R squared -> 0.77 MSE -> 0.202 MAE -> 0.35

Stage 2 classifier

```r
cols_reg = c('Y2', 'Z1', 'Y1')
dummies <- dummyVars(Z1 ~ ., data = dat[,cols_reg])

train_dummies = predict(dummies, newdata = train[,cols_reg])

test_dummies = predict(dummies, newdata = test[,cols_reg])

print(dim(train_dummies)); print(dim(test_dummies))
```

```
## [1] 44  2
```

```
## [1] 20  2
```

```r
library(glmnet)

x = as.matrix(train_dummies)
y_train = train$Z1

x_test = as.matrix(test_dummies)
y_test = test$Z1

lambdas <- 10^seq(2, -3, by = -.1)
ridge_reg = glmnet(x, y_train, nlambda = 25, alpha = 0, family = 'gaussian', lambda = lambdas)

summary(ridge_reg)
```

```
##           Length Class     Mode
## a0         51    -none-    numeric
## beta      102    dgCMatrix S4
## df         51    -none-    numeric
## dim         2    -none-    numeric
## lambda     51    -none-    numeric
## dev.ratio  51    -none-    numeric
## nulldev     1    -none-    numeric
## npasses     1    -none-    numeric
## jerr        1    -none-    numeric
## offset      1    -none-    logical
## call        7    -none-    call
## nobs        1    -none-    numeric
```

```r
cv_ridge <- cv.glmnet(x, y_train, alpha = 0, lambda = lambdas)
optimal_lambda <- cv_ridge$lambda.min
optimal_lambda
```

```
## [1] 0.01
```

```r
# Compute R^2 from true and predicted values
eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))


  # Model performance metrics
data.frame(
  RMSE = RMSE,
  Rsquare = R_square
)

}

# Prediction and evaluation on train data
predictions_train <- predict(ridge_reg, s = optimal_lambda, newx = x)
eval_results(y_train, predictions_train, train)
```

```
##        RMSE   Rsquare
## 1 0.3448454 0.8783161
```

```r
# Prediction and evaluation on test data
predictions_test <- predict(ridge_reg, s = optimal_lambda, newx = x_test)
eval_results(y_test, predictions_test, test)
```

```
##        RMSE   Rsquare
## 1 0.4608319 0.7686179
```

```r
d<-predictions_test - test[,c('Z1')]
d
```

```
##             Z1
## 1   0.29623356
## 2   0.02490690
## 3  -1.00490105
## 4   0.62162097
## 5   0.56025517
## 6   0.10566803
## 7  -0.14929349
## 8  -0.12050983
## 9  -0.19932270
## 10 -0.22484635
## 11 -0.29280134
## 12  0.23661172
## 13  0.13407758
## 14 -0.02495441
## 15 -0.36067546
## 16  0.15540859
## 17 -0.76838423
## 18 -0.35271884
```

```
## 19 -1.13149598
## 20 -0.02991481
```

```
mse = mean((d[,c('Z1')])^2)
mse
```

```
## [1] 0.212366
```

```
mae = mean(abs(d[,c('Z1')]))
mae
```

```
## [1] 0.3397301
```

The values for stage 2 classifier on ridge regression are

1. RMSE -> 0.46
2. R squared -> 0.76
3. MSE ->0.212
4. MAE -> 0.33

Same applies for Stage 2 too. The linear model is found to be better.

b. Lasso regression

For stage 1 classifier (for Y1)

```
cols = c('X1','X2','X3','Y1')

pre_proc_val <- preProcess(train[,cols], method = c("center", "scale"))

train[,cols] = predict(pre_proc_val, train[,cols])
test[,cols] = predict(pre_proc_val, test[,cols])

summary(train)
```

```
##  Sample Number            X1                 X2                 X3
##  Min.   :-1.59711   Min.   :-1.2245   Min.   :-1.2882   Min.   :-1.4578
##  1st Qu.:-0.91834   1st Qu.:-1.2245   1st Qu.:-0.5797   1st Qu.:-0.5669
##  Median :-0.02662   Median :-0.3693   Median :-0.3435   Median : 0.3239
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.91834   3rd Qu.: 0.6997   3rd Qu.: 0.6012   3rd Qu.: 1.2148
##  Max.   : 1.70358   Max.   : 1.3411   Max.   : 1.5459   Max.   : 1.2148
##        Y1                 Y2                 Z1
##  Min.   :-1.1528   Min.   :-1.9881   Min.   :-1.7994
##  1st Qu.:-0.8055   1st Qu.:-0.4925   1st Qu.:-0.8963
##  Median :-0.1364   Median :-0.2238   Median : 0.2326
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.0123   3rd Qu.: 1.0489   3rd Qu.: 0.6466
##  Max.   : 1.7990   Max.   : 1.7299   Max.   : 1.9636
```

```r
cols_reg = c(1:5)

dummies <- dummyVars(Y1 ~ ., data = dat[,cols_reg])

train_dummies = predict(dummies, newdata = train[,cols_reg])

test_dummies = predict(dummies, newdata = test[,cols_reg])

print(dim(train_dummies)); print(dim(test_dummies))
```

```
## [1] 44  4
```

```
## [1] 20  4
```

Finding the optimal lamba value for lasso regression

```r
lambdas <- 10^seq(2, -3, by = -.1)

# Setting alpha = 1 implements lasso regression
lasso_reg <- cv.glmnet(x, y_train, alpha = 1, lambda = lambdas, standardize = TRUE, nfolds = 5)

# Best
lambda_best <- lasso_reg$lambda.min
lambda_best
```

```
## [1] 0.01995262
```

Now we train the lasso model using the obtained lambda values

```r
lasso_model <- glmnet(x, y_train, alpha = 1, lambda = lambda_best, standardize = TRUE)

predictions_train <- predict(lasso_model, s = lambda_best, newx = x)
eval_results(y_train, predictions_train, train)
```

```
##        RMSE   Rsquare
## 1 0.3454587 0.8778829
```

```r
predictions_test <- predict(lasso_model, s = lambda_best, newx = x_test)
eval_results(y_test, predictions_test, test)
```

```
##        RMSE   Rsquare
## 1 0.4679396 0.7614254
```

```r
d<-predictions_test - test[,c('Y1')]
d
```

```
##             Y1
## 1  -2.43260573
## 2  -3.27750139
## 3  -2.44244462
```

```
## 4  -0.54635245
## 5  -0.04855876
## 6  -0.05061772
## 7   0.21316355
## 8   0.03771121
## 9   0.16070286
## 10  1.33400858
## 11  1.07374236
## 12  1.18250996
## 13  1.27094893
## 14  1.20938119
## 15  1.72534046
## 16  1.64839477
## 17  1.92251428
## 18  2.09368226
## 19  1.98272325
## 20  2.32696680
```

```
mse = mean((d[,c('Y1')])^2)
mse
```

```
## [1] 2.676476
```

```
mae = mean(abs(d[,c('Y1')]))
mae
```

```
## [1] 1.348994
```

The values for stage 1 classifier (on Y1) using Lasso regression are

1. RMSE -> 0.46
2. R square -> 0.76
3. MSE -> 2.67
4. MAE -> 1.34

The model shows decreased R square value for testing and increased RMSE for testing which indicates that this model is also not good enough

Stage 1 classifier for Y2

```
cols = c('X1','X2','X3','Y2')

pre_proc_val <- preProcess(train[,cols], method = c("center", "scale"))

train[,cols] = predict(pre_proc_val, train[,cols])
test[,cols] = predict(pre_proc_val, test[,cols])

summary(train)
```

```
##  Sample Number            X1                 X2                 X3
##  Min.   :-1.59711   Min.   :-1.2245   Min.   :-1.2882   Min.   :-1.4578
##  1st Qu.:-0.91834   1st Qu.:-1.2245   1st Qu.:-0.5797   1st Qu.:-0.5669
```

```
##  Median :-0.02662   Median :-0.3693   Median :-0.3435   Median : 0.3239
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.91834   3rd Qu.: 0.6997   3rd Qu.: 0.6012   3rd Qu.: 1.2148
##  Max.   : 1.70358   Max.   : 1.3411   Max.   : 1.5459   Max.   : 1.2148
##        Y1                 Y2                 Z1
##  Min.   :-1.1528   Min.   :-1.9881   Min.   :-1.7994
##  1st Qu.:-0.8055   1st Qu.:-0.4925   1st Qu.:-0.8963
##  Median :-0.1364   Median :-0.2238   Median : 0.2326
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.0123   3rd Qu.: 1.0489   3rd Qu.: 0.6466
##  Max.   : 1.7990   Max.   : 1.7299   Max.   : 1.9636
```

```
cols_reg = c(1:5)

dummies <- dummyVars(Y1 ~ ., data = dat[,cols_reg])

train_dummies = predict(dummies, newdata = train[,cols_reg])

test_dummies = predict(dummies, newdata = test[,cols_reg])

print(dim(train_dummies)); print(dim(test_dummies))
```

```
## [1] 44  4
```

```
## [1] 20  4
```

Finding the optimal lamba value for lasso regression

```
lambdas <- 10^seq(2, -3, by = -.1)

# Setting alpha = 1 implements lasso regression
lasso_reg <- cv.glmnet(x, y_train, alpha = 1, lambda = lambdas, standardize = TRUE, nfolds = 5)

# Best
lambda_best <- lasso_reg$lambda.min
lambda_best
```

```
## [1] 0.01995262
```

Now we train the lasso model using the obtained lambda values

```
lasso_model <- glmnet(x, y_train, alpha = 1, lambda = lambda_best, standardize = TRUE)

predictions_train <- predict(lasso_model, s = lambda_best, newx = x)
eval_results(y_train, predictions_train, train)
```

```
##        RMSE   Rsquare
## 1 0.3454587 0.8778829
```

```
predictions_test <- predict(lasso_model, s = lambda_best, newx = x_test)
eval_results(y_test, predictions_test, test)
```

```
##        RMSE   Rsquare
## 1 0.4679396 0.7614254
```

```
d<-predictions_test - test[,c('Y2')]
d
```

```
##             Y2
## 1   0.10211846
## 2   0.24317554
## 3  -1.32481743
## 4   0.13700521
## 5   0.27056803
## 6   0.31970104
## 7  -1.31009577
## 8  -0.44670825
## 9  -0.71890738
## 10 -0.68853794
## 11  1.51806354
## 12  0.92460952
## 13  0.81625896
## 14  0.28337950
## 15 -0.01654426
## 16  1.15893117
## 17  1.28450712
## 18  0.54337842
## 19 -0.15226374
## 20 -0.85919149
```

```
mse = mean((d[,c('Y2')])^2)
mse
```

```
## [1] 0.6440856
```

```
mae = mean(abs(d[,c('Y2')]))
mae
```

```
## [1] 0.6559381
```

Values of stage1 classifier on Y2 using lasso regression are 1. RMSE -> 0.46 2. R square -> 0.76 3. MSE -> 0.63 4. MAE -> 0.64

b. Stage 2 classifier

```
cols = c('Y1','Y2','Z1')

pre_proc_val <- preProcess(train[,cols], method = c("center", "scale"))
```

```
train[,cols] = predict(pre_proc_val, train[,cols])
test[,cols] = predict(pre_proc_val, test[,cols])

summary(train)
```

```
##  Sample Number            X1                X2                X3
##  Min.   :-1.59711   Min.   :-1.2245   Min.   :-1.2882   Min.   :-1.4578
##  1st Qu.:-0.91834   1st Qu.:-1.2245   1st Qu.:-0.5797   1st Qu.:-0.5669
##  Median :-0.02662   Median :-0.3693   Median :-0.3435   Median : 0.3239
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.91834   3rd Qu.: 0.6997   3rd Qu.: 0.6012   3rd Qu.: 1.2148
##  Max.   : 1.70358   Max.   : 1.3411   Max.   : 1.5459   Max.   : 1.2148
##        Y1                Y2                Z1
##  Min.   :-1.1528   Min.   :-1.9881   Min.   :-1.7994
##  1st Qu.:-0.8055   1st Qu.:-0.4925   1st Qu.:-0.8963
##  Median :-0.1364   Median :-0.2238   Median : 0.2326
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.0123   3rd Qu.: 1.0489   3rd Qu.: 0.6466
##  Max.   : 1.7990   Max.   : 1.7299   Max.   : 1.9636
```

```
cols_reg = c('Y1','Y2','Z1')

dummies <- dummyVars(Z1 ~ ., data = dat[,cols_reg])

train_dummies = predict(dummies, newdata = train[,cols_reg])

test_dummies = predict(dummies, newdata = test[,cols_reg])

print(dim(train_dummies)); print(dim(test_dummies))
```

```
## [1] 44  2
```

```
## [1] 20  2
```

Finding the optimal lambda value for lasso regression

```
lambdas <- 10^seq(2, -3, by = -.1)

# Setting alpha = 1 implements lasso regression
lasso_reg <- cv.glmnet(x, y_train, alpha = 1, lambda = lambdas, standardize = TRUE, nfolds = 5)

# Best
lambda_best <- lasso_reg$lambda.min
lambda_best
```

```
## [1] 0.001
```

Now we train the lasso model using the obtained lambda values

```r
lasso_model <- glmnet(x, y_train, alpha = 1, lambda = lambda_best, standardize = TRUE)

predictions_train <- predict(lasso_model, s = lambda_best, newx = x)
eval_results(y_train, predictions_train, train)
```

```
##        RMSE   Rsquare
## 1 0.3446986 0.8784197
```

```r
predictions_test <- predict(lasso_model, s = lambda_best, newx = x_test)
eval_results(y_test, predictions_test, test)
```

```
##        RMSE   Rsquare
## 1 0.4606175 0.7688331
```

```r
d<-predictions_test - test[,c('Z1')]
d
```

```
##             Z1
## 1   0.29003305
## 2   0.01724723
## 3  -1.02051370
## 4   0.62097855
## 5   0.56189740
## 6   0.10762757
## 7  -0.15730724
## 8  -0.12334110
## 9  -0.20359666
## 10 -0.22560600
## 11 -0.27978437
## 12  0.24603307
## 13  0.14303619
## 14 -0.01967424
## 15 -0.35591018
## 16  0.16768710
## 17 -0.75450549
## 18 -0.34323063
## 19 -1.12689617
## 20 -0.02899168
```

```r
mse = mean((d[,c('Z1')])^2)
mse
```

```
## [1] 0.2121685
```

```r
mae = mean(abs(d[,c('Z1')]))
mae
```

```
## [1] 0.3396949
```

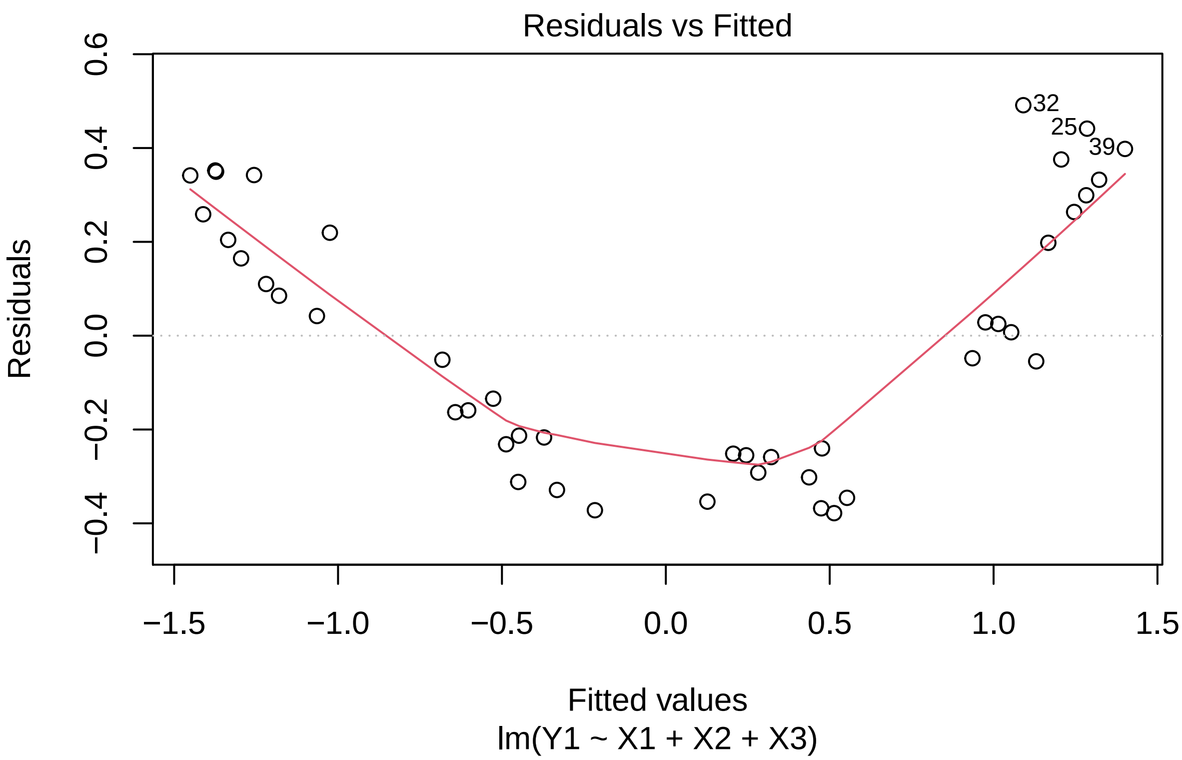
The values for stage 2 classifier using Lasso regression are 1. RMSE -> 0.46 2. R sqaure -> 0.76 3. MSE -> 0.21 4. MAE -> 0.34

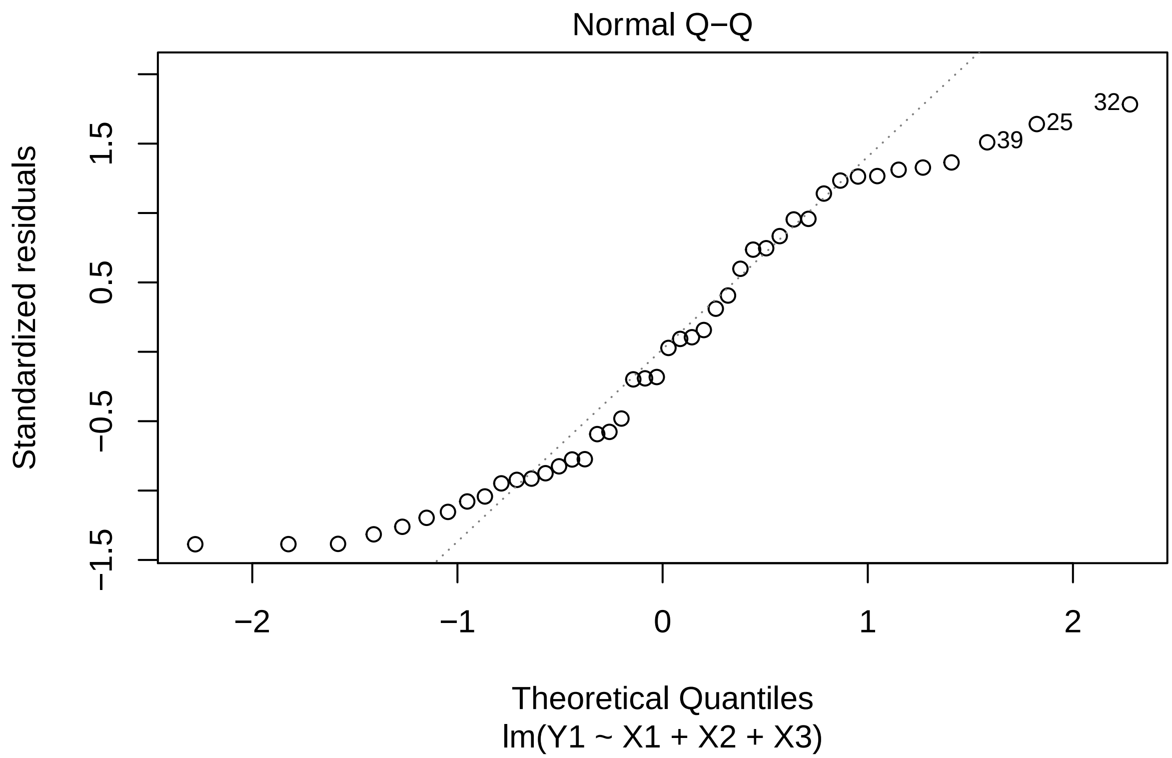
Shows the same and hence is again not a good model

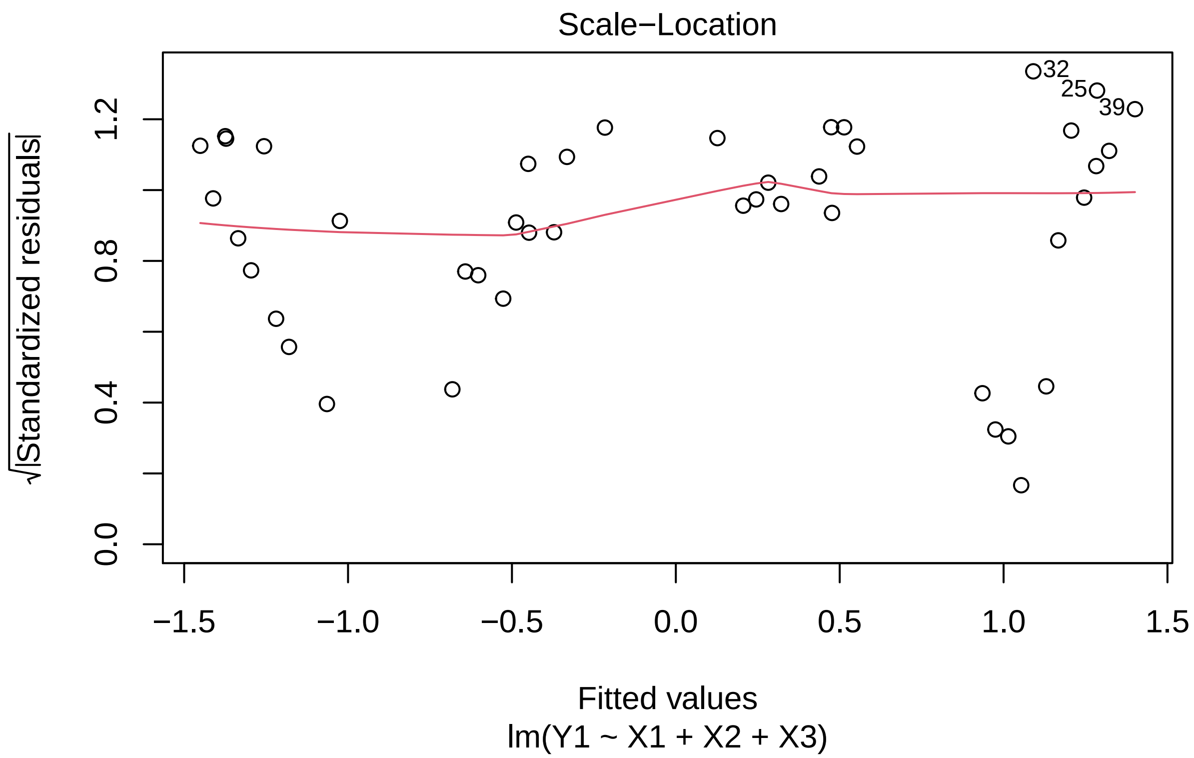
8. Plotting

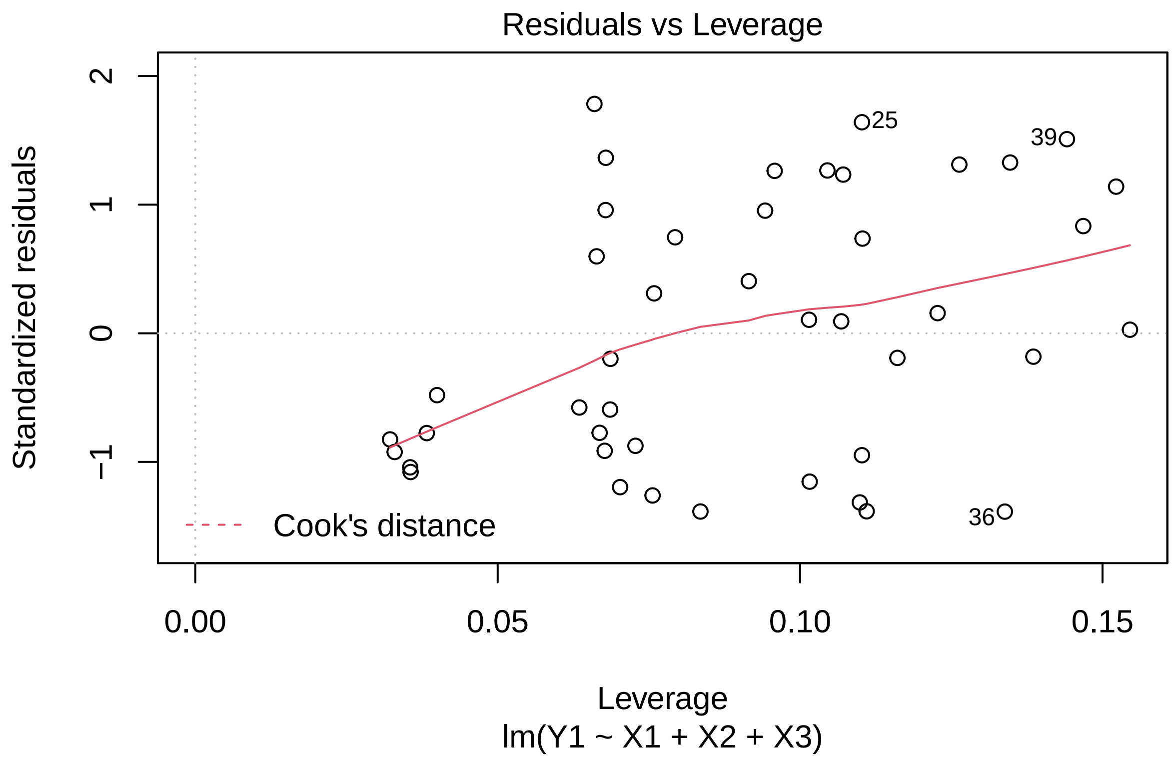
Linear regression model will be a better model

```
model1<-lm(Y1~X1+X2+X3, data = train)
plot(model1)
```

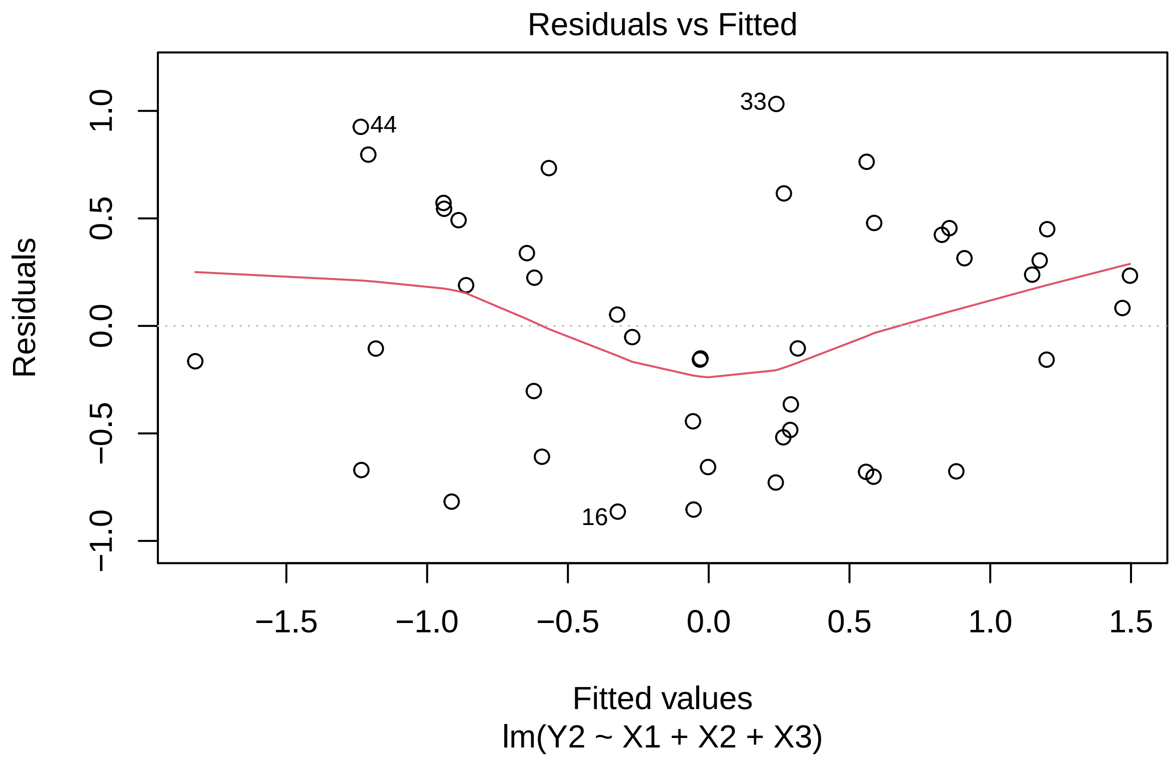


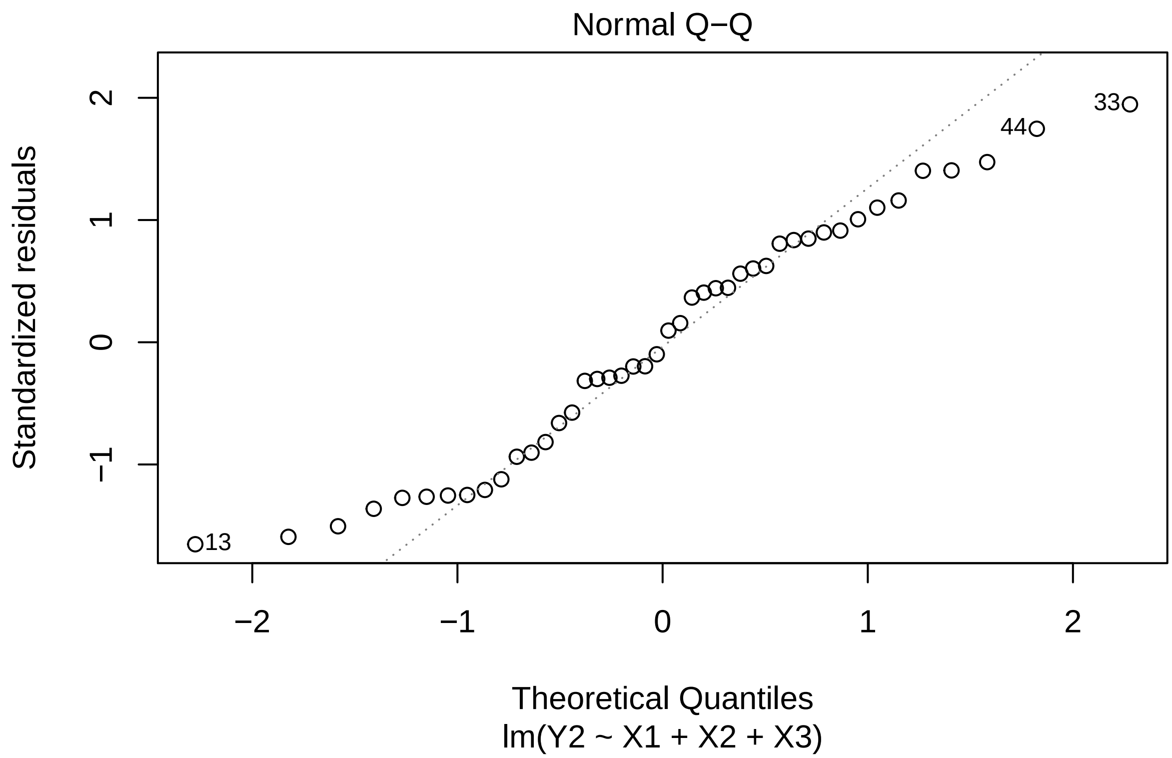

Normal Q−Q
Standardized residuals
1.5
0.5
−0.5
−1.5
−2
−1
0
1
2
32
25
39
Theoretical Quantiles
lm(Y1 ~ X1 + X2 + X3)

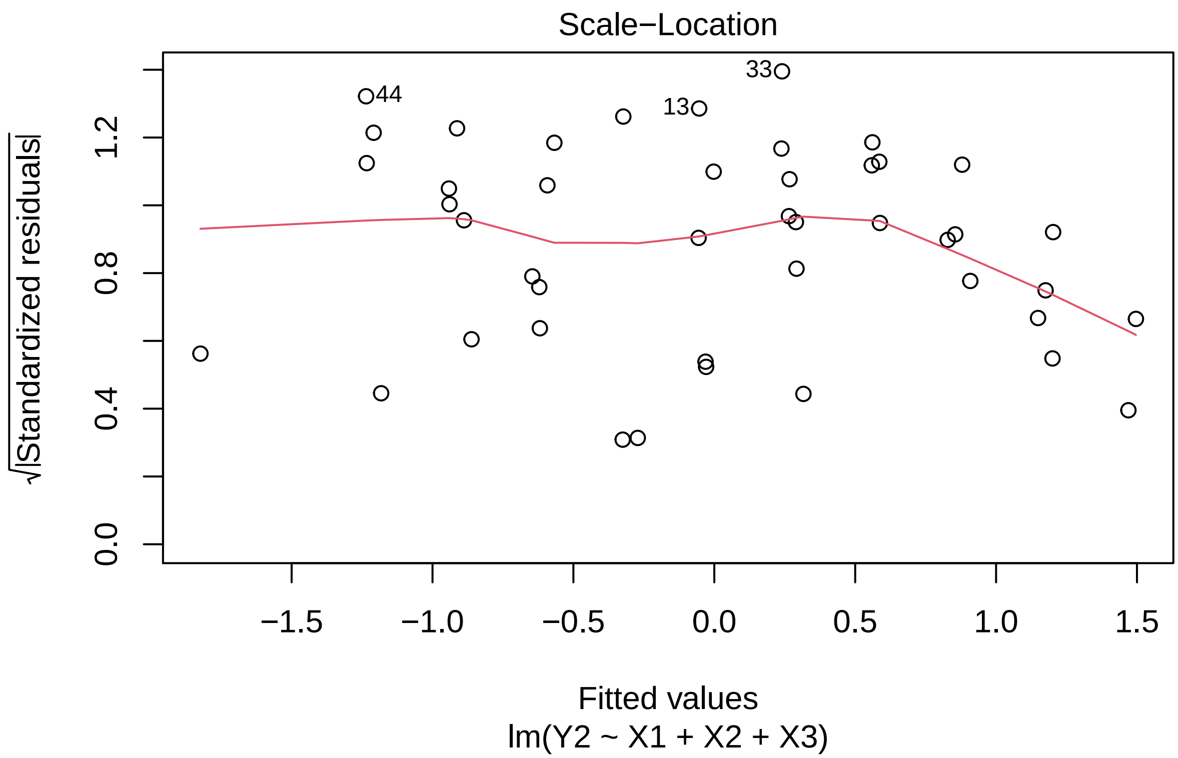

Scale−Location
√|Standardized residuals|
0.0
0.4
0.8
1.2
32
25
39
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
Fitted values
lm(Y1 ~ X1 + X2 + X3)

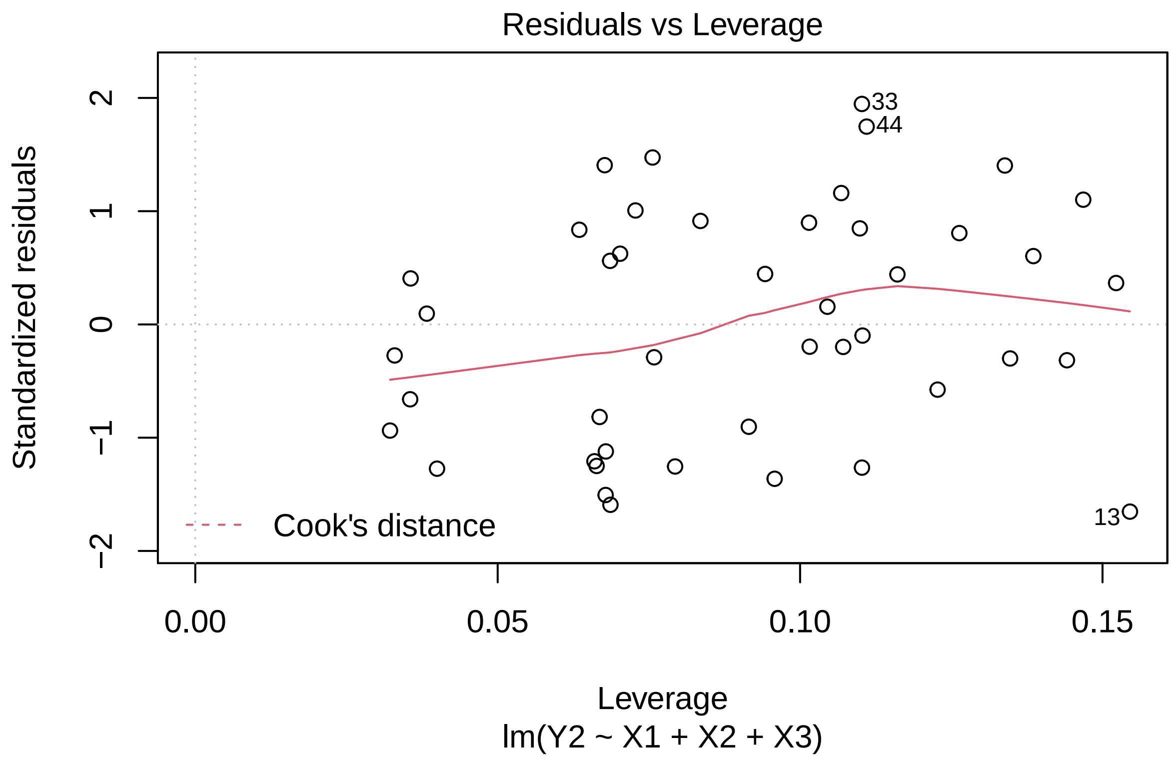



```
model2<-lm(Y2~X1+X2+X3, data = train)
plot(model2)
```

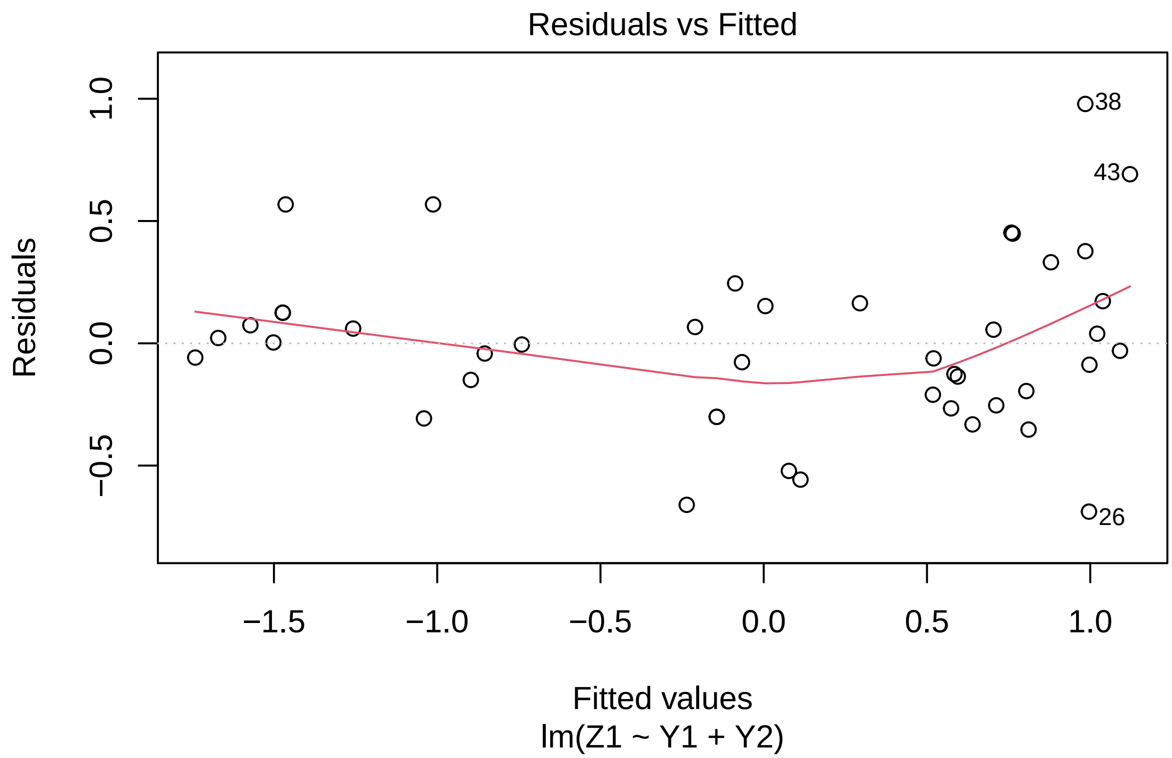

Residuals vs Fitted
Residuals
Fitted values
lm(Y2 ~ X1 + X2 + X3)
44
33
16
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.5

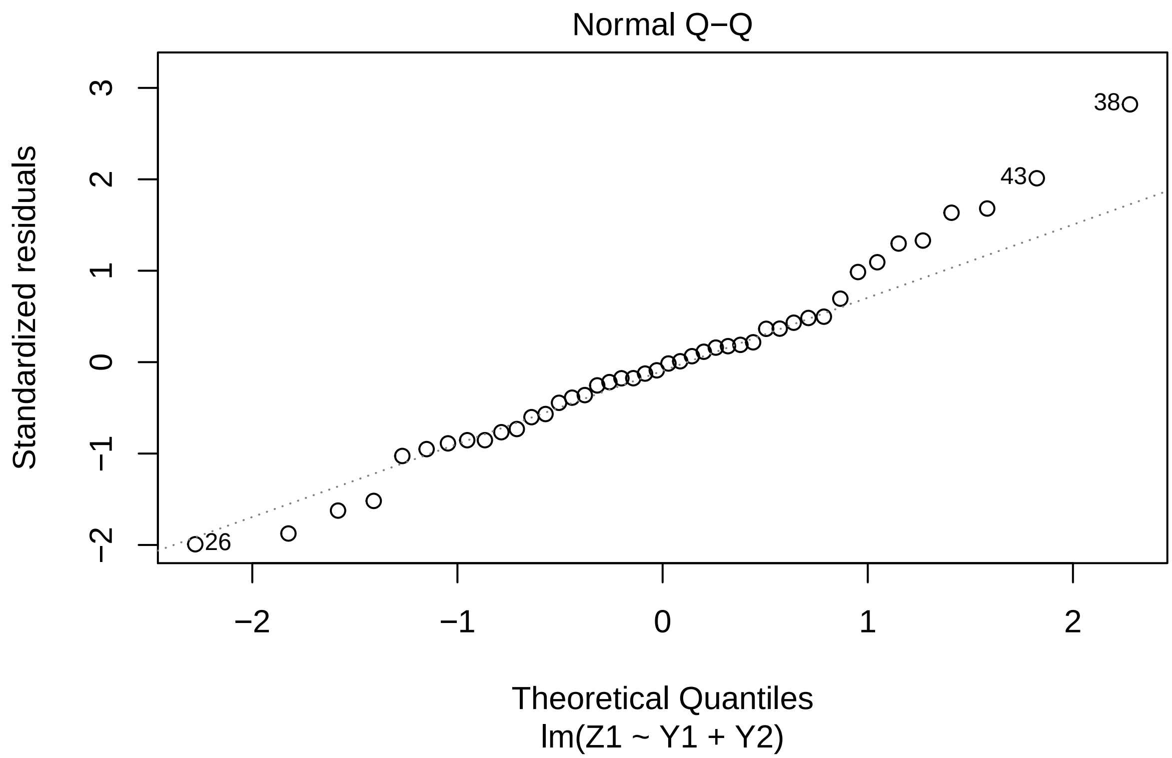

Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Y2 ~ X1 + X2 + X3)
33
44
13
−2
−1
0
1
2

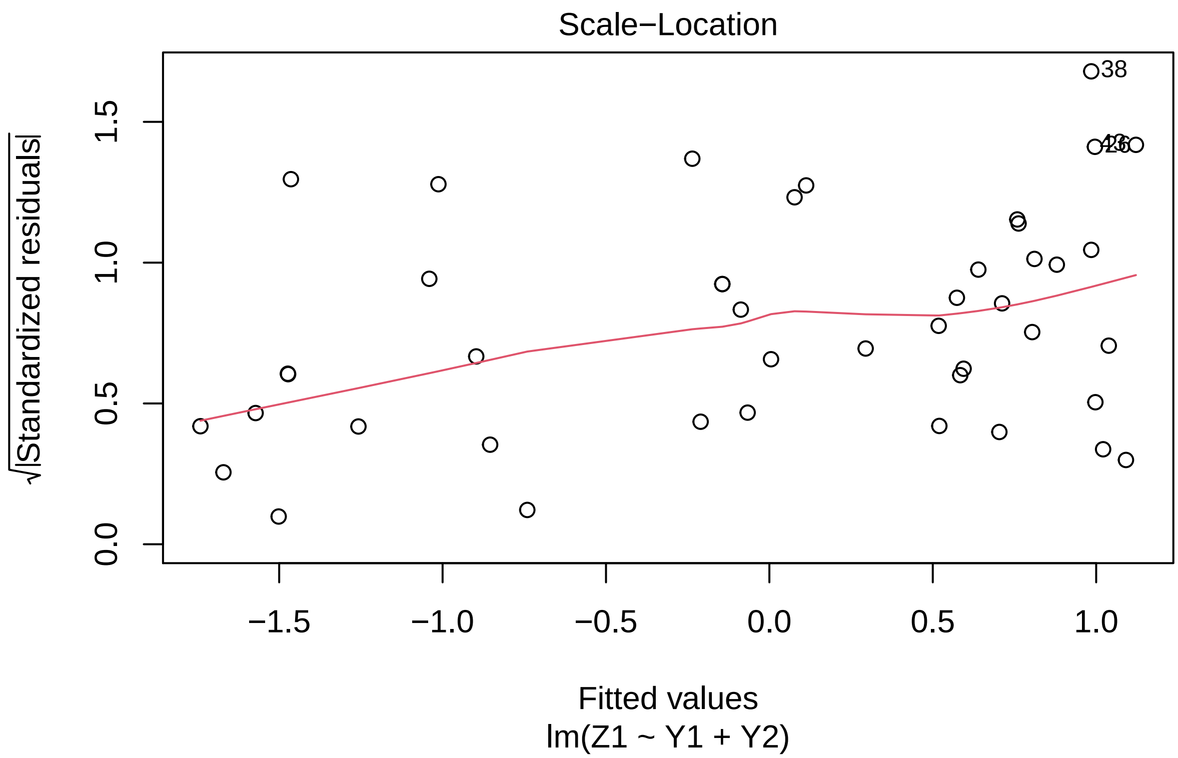

Scale−Location
√|Standardized residuals|
0.0
0.4
0.8
1.2
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
44
13
33
Fitted values
lm(Y2 ~ X1 + X2 + X3)

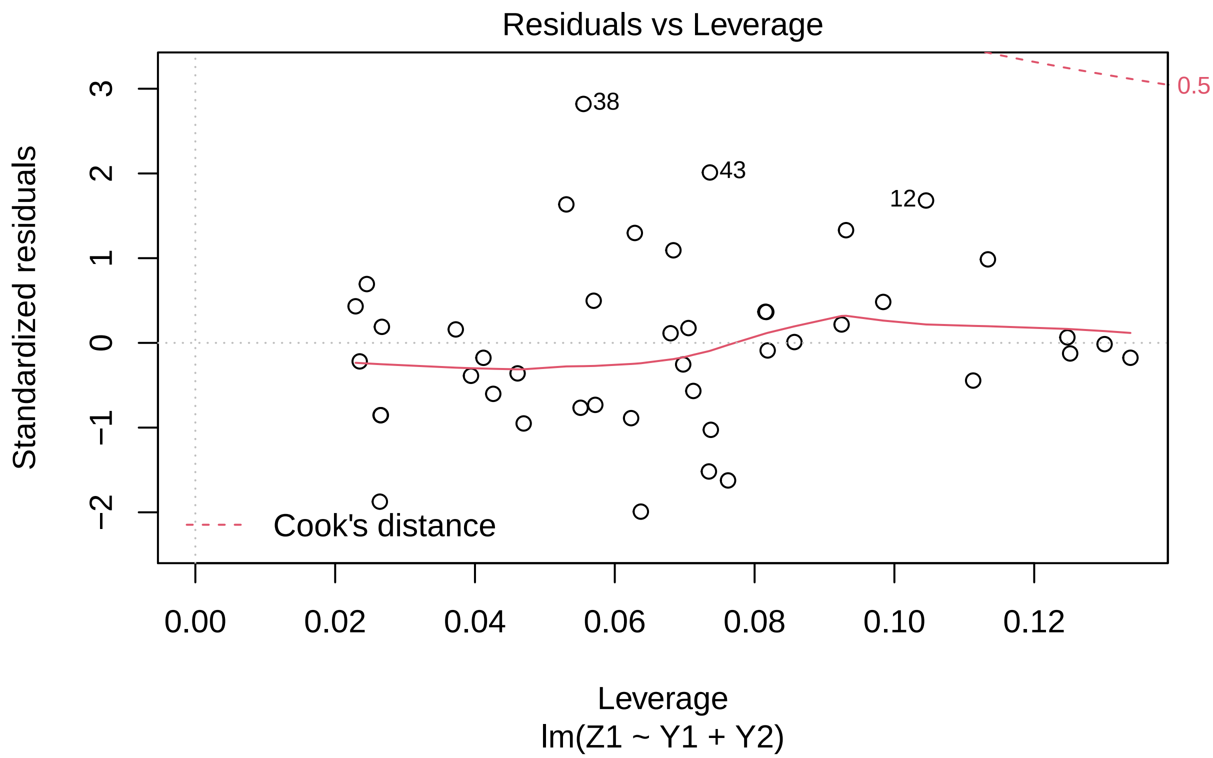



```
model3<-lm(Z1~Y1+Y2, data=train)
plot(model3)
```


Residuals vs Fitted
Residuals
1.0
0.5
0.0
−0.5
38
43
26
−1.5
−1.0
−0.5
0.0
0.5
1.0
Fitted values
lm(Z1 ~ Y1 + Y2)


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Z1 ~ Y1 + Y2)
38
43
26
3
2
1
0
−1
−2
−2
−1
0
1
2


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
38
Fitted values
lm(Z1 ~ Y1 + Y2)




From the graphs we can say that linear graph suits well.