Project 2 Regression

Ashish Lamichhane

Prediction of % of Silica on the Iron Ore Concentrate

Link of the data :

https://www.kaggle.com/edumagalhaes/quality-prediction-in-a-mining-process

Background information on data:

The main goal is to use this data to predict how much impurity is in the ore concentrate. As this impurity is measured every hour, if we can predict how much silica (impurity) is in the ore concentrate, we can help the engineers, giving them early information to take actions (empowering!). Hence, they will be able to take corrective actions in advance (reduce impurity, if it is the case) and also help the environment (reducing the amount of ore that goes to tailings as you reduce silica in the ore concentrate). The first column shows time and date range (from march of 2017 until september of 2017). Some columns were sampled every 20 second. Others were sampled on a hourly base. The second and third columns are quality measures of the iron ore pulp right before it is fed into the flotation plant. Column 4 until column 8 are the most important variables that impact in the ore quality in the end of the process. From column 9 until column 22, we can see process data (level and air flow inside the flotation columns, which also impact in ore quality. The last two columns are the final iron ore pulp quality measurement from the lab. Target is to predict the last column, which is the % of silica in the iron ore concentrate.

Citation : Oliveira, Eduardo Magalhães. Quality Prediction in Mining Process. Kaggle, 6 Dec. 2017,www.kaggle.com/edumagalhaes/quality-prediction-in-a-mining-process.

I used the parameter na.strings="NA" to tell R to fill missing cells with NA. I am using stringsAsFactors = FALSE otherwise, columns such as X. Silica.Feed or X.Iron.Feed could be read as factors but we want them as numbers. I have used dec = "," because the number in the table were separated by ",". The original data set has almost 738k instances. I am reducing the data set to 100k because my laptop was not able to handle 738k instances for some algorithms.

```
df <- read.csv("MiningProcess_Flotation_Plant_Database.csv", na.strings = "NA",stringsAsFactors = FALSE
df <- df[sample(nrow(df), 100000, replace = FALSE), ]
```

Data Exploration Functions (Before data cleaning)

```
names(df) # lists the column names.
```

```
##  [1] "date"                          "X..Iron.Feed"
##  [3] "X..Silica.Feed"                "Starch.Flow"
##  [5] "Amina.Flow"                    "Ore.Pulp.Flow"
##  [7] "Ore.Pulp.pH"                   "Ore.Pulp.Density"
##  [9] "Flotation.Column.01.Air.Flow" "Flotation.Column.02.Air.Flow"
```

```
## [11] "Flotation.Column.03.Air.Flow" "Flotation.Column.04.Air.Flow"
## [13] "Flotation.Column.05.Air.Flow" "Flotation.Column.06.Air.Flow"
## [15] "Flotation.Column.07.Air.Flow" "Flotation.Column.01.Level"
## [17] "Flotation.Column.02.Level"    "Flotation.Column.03.Level"
## [19] "Flotation.Column.04.Level"    "Flotation.Column.05.Level"
## [21] "Flotation.Column.06.Level"    "Flotation.Column.07.Level"
## [23] "X..Iron.Concentrate"          "X..Silica.Concentrate"
```

```
head(df, n = 5) #see first 10 rows.
```

```
##                       date X..Iron.Feed X..Silica.Feed Starch.Flow
## 16170  2017-03-13 18:00:00        58.95           8.94   4635.5500
## 333161 2017-06-08 09:00:00        64.03           6.26   3889.8800
## 205893 2017-05-09 22:00:00        52.61          20.24    704.4713
## 201083 2017-05-08 20:00:00        56.65          14.83   4353.0700
## 451972 2017-07-05 21:00:00        53.63          15.74   3176.0000
##        Amina.Flow Ore.Pulp.Flow Ore.Pulp.pH Ore.Pulp.Density
## 16170    440.7350       392.285     9.83961          1.74357
## 333161   453.5600       409.033    10.28440          1.73619
## 205893   244.2354       402.246     9.10214          1.52062
## 201083   513.9110       392.021    10.22310          1.72824
## 451972   603.0270       410.742     9.80207          1.75482
##        Flotation.Column.01.Air.Flow Flotation.Column.02.Air.Flow
## 16170                       252.532                       249.247
## 333161                      300.461                       294.628
## 205893                      250.049                       247.632
## 201083                      250.018                       248.057
## 451972                      297.729                       298.389
##        Flotation.Column.03.Air.Flow Flotation.Column.04.Air.Flow
## 16170                       250.719                       295.096
## 333161                      301.576                       299.927
## 205893                      251.807                       300.366
## 201083                      250.043                       300.366
## 451972                      301.245                       295.093
##        Flotation.Column.05.Air.Flow Flotation.Column.06.Air.Flow
## 16170                       306.400                       249.906
## 333161                      297.410                       306.686
## 205893                      297.729                       299.927
## 201083                      299.892                       299.795
## 451972                      300.366                       296.631
##        Flotation.Column.07.Air.Flow Flotation.Column.01.Level
## 16170                       249.170                    614.299
## 333161                      302.185                    393.884
## 205893                      292.676                    464.453
## 201083                      300.313                    457.455
## 451972                      301.685                    405.396
##        Flotation.Column.02.Level Flotation.Column.03.Level
## 16170                    607.215                   635.779
## 333161                   407.158                   399.012
## 205893                   451.426                   447.827
## 201083                   456.614                   457.438
## 451972                   327.383                   371.211
##        Flotation.Column.04.Level Flotation.Column.05.Level
## 16170                    539.490                   533.071
```

```
## 333161                   351.945                   344.551
## 205893                   338.672                   357.457
## 201083                   335.749                   342.399
## 451972                   431.250                   392.707
##        Flotation.Column.06.Level Flotation.Column.07.Level
## 16170                    547.796                   543.342
## 333161                   351.803                   340.173
## 205893                   344.436                   366.367
## 201083                   352.984                   357.610
## 451972                   414.443                   466.235
##        X..Iron.Concentrate X..Silica.Concentrate
## 16170                65.15                  3.01
## 333161               66.25                  1.25
## 205893               64.38                  3.82
## 201083               64.28                  3.17
## 451972               65.64                  1.51
```

```
tail(df, n = 5) # see last 5 rows.
```

```
##                        date X..Iron.Feed X..Silica.Feed Starch.Flow
## 706365 2017-09-02 19:00:00        54.27          16.60     4285.86
## 50968  2017-04-04 02:00:00        56.65          13.99     2648.29
## 421562 2017-06-28 21:00:00        50.70          23.17     3598.22
## 568247 2017-08-01 19:00:00        57.46          10.80     2126.01
## 462957 2017-07-08 11:00:00        55.40          16.83     3294.51
##        Amina.Flow Ore.Pulp.Flow Ore.Pulp.pH Ore.Pulp.Density
## 706365    620.235       413.151     9.74205          1.76650
## 50968     539.982       402.711     9.85672          1.73074
## 421562    484.569       407.334     9.36561          1.59436
## 568247    479.340       380.497     9.75076          1.71953
## 462957    613.766       412.811     9.97342          1.70679
##        Flotation.Column.01.Air.Flow Flotation.Column.02.Air.Flow
## 706365                      299.487                      302.344
## 50968                       250.269                      255.135
## 421562                      300.330                      298.036
## 568247                      300.266                      249.856
## 462957                      297.718                      304.708
##        Flotation.Column.03.Air.Flow Flotation.Column.04.Air.Flow
## 706365                      301.535                     295.5842
## 50968                       249.390                     295.0960
## 421562                      298.260                     300.3660
## 568247                      302.085                     301.9390
## 462957                      298.158                     301.8010
##        Flotation.Column.05.Air.Flow Flotation.Column.06.Air.Flow
## 706365                     292.7910                      347.546
## 50968                      306.4000                      251.555
## 421562                     303.0610                      306.242
## 568247                     298.2459                      326.626
## 462957                     299.2100                      300.078
##        Flotation.Column.07.Air.Flow Flotation.Column.01.Level
## 706365                      306.088                  583.0930
## 50968                       250.451                  602.6500
## 421562                      298.058                  393.4340
## 568247                      349.594                  633.5633
```

```
## 462957                      297.963                   546.2570
##        Flotation.Column.02.Level Flotation.Column.03.Level
## 706365                   478.639                  394.0820
## 50968                    597.117                  879.7377
## 421562                   418.642                  408.9010
## 568247                   506.909                  503.1330
## 462957                   610.291                  566.8550
##        Flotation.Column.04.Level Flotation.Column.05.Level
## 706365                   372.911                   425.887
## 50968                    349.801                   321.229
## 421562                   405.605                   381.387
## 568247                   451.759                   459.605
## 462957                   543.672                   559.719
##        Flotation.Column.06.Level Flotation.Column.07.Level
## 706365                   453.142                  416.5640
## 50968                    673.352                  209.1247
## 421562                   390.037                  392.1630
## 568247                   452.009                  453.3420
## 462957                   433.135                  475.3910
##        X..Iron.Concentrate X..Silica.Concentrate
## 706365               64.54                  1.51
## 50968                66.29                  1.50
## 421562               65.19                  1.93
## 568247               65.44                  2.08
## 462957               66.45                  1.23
```

```
str(df) #finding the structure of the data set.
```

```
## 'data.frame':    100000 obs. of  24 variables:
##  $ date                        : chr  "2017-03-13 18:00:00" "2017-06-08 09:00:00" "2017-05-09 22:00:0
##  $ X..Iron.Feed                : num  59 64 52.6 56.6 53.6 ...
##  $ X..Silica.Feed              : num  8.94 6.26 20.24 14.83 15.74 ...
##  $ Starch.Flow                 : num  4636 3890 704 4353 3176 ...
##  $ Amina.Flow                  : num  441 454 244 514 603 ...
##  $ Ore.Pulp.Flow               : num  392 409 402 392 411 ...
##  $ Ore.Pulp.pH                 : num  9.84 10.28 9.1 10.22 9.8 ...
##  $ Ore.Pulp.Density            : num  1.74 1.74 1.52 1.73 1.75 ...
##  $ Flotation.Column.01.Air.Flow: num  253 300 250 250 298 ...
##  $ Flotation.Column.02.Air.Flow: num  249 295 248 248 298 ...
##  $ Flotation.Column.03.Air.Flow: num  251 302 252 250 301 ...
##  $ Flotation.Column.04.Air.Flow: num  295 300 300 300 295 ...
##  $ Flotation.Column.05.Air.Flow: num  306 297 298 300 300 ...
##  $ Flotation.Column.06.Air.Flow: num  250 307 300 300 297 ...
##  $ Flotation.Column.07.Air.Flow: num  249 302 293 300 302 ...
##  $ Flotation.Column.01.Level   : num  614 394 464 457 405 ...
##  $ Flotation.Column.02.Level   : num  607 407 451 457 327 ...
##  $ Flotation.Column.03.Level   : num  636 399 448 457 371 ...
##  $ Flotation.Column.04.Level   : num  539 352 339 336 431 ...
##  $ Flotation.Column.05.Level   : num  533 345 357 342 393 ...
##  $ Flotation.Column.06.Level   : num  548 352 344 353 414 ...
##  $ Flotation.Column.07.Level   : num  543 340 366 358 466 ...
##  $ X..Iron.Concentrate         : num  65.2 66.2 64.4 64.3 65.6 ...
##  $ X..Silica.Concentrate       : num  3.01 1.25 3.82 3.17 1.51 2.38 2.72 1.67 1.39 2.05 ...
```

```r
summary(df) # summary() function provides a number of useful statistics including range, median, and me
```

```
##      date            X..Iron.Feed   X..Silica.Feed   Starch.Flow
##  Length:100000      Min.   :42.74   Min.   : 1.31   Min.   :   0.002
##  Class :character   1st Qu.:52.67   1st Qu.: 8.98   1st Qu.:2073.622
##  Mode  :character   Median :56.02   Median :13.93   Median :3018.155
##                     Mean   :56.28   Mean   :14.67   Mean   :2867.724
##                     3rd Qu.:59.72   3rd Qu.:19.94   3rd Qu.:3728.253
##                     Max.   :65.78   Max.   :33.40   Max.   :6300.230
##    Amina.Flow    Ore.Pulp.Flow    Ore.Pulp.pH     Ore.Pulp.Density
##  Min.   :241.7   Min.   :376.2   Min.   : 8.753   Min.   :1.520
##  1st Qu.:430.8   1st Qu.:394.2   1st Qu.: 9.528   1st Qu.:1.647
##  Median :504.1   Median :399.3   Median : 9.800   Median :1.698
##  Mean   :487.7   Mean   :397.6   Mean   : 9.770   Mean   :1.680
##  3rd Qu.:553.0   3rd Qu.:402.9   3rd Qu.:10.041   3rd Qu.:1.728
##  Max.   :739.4   Max.   :418.6   Max.   :10.808   Max.   :1.853
##  Flotation.Column.01.Air.Flow Flotation.Column.02.Air.Flow
##  Min.   :175.6                Min.   :176.3
##  1st Qu.:250.3                1st Qu.:250.4
##  Median :299.4                Median :296.2
##  Mean   :280.1                Mean   :277.1
##  3rd Qu.:300.1                3rd Qu.:300.7
##  Max.   :369.3                Max.   :376.0
##  Flotation.Column.03.Air.Flow Flotation.Column.04.Air.Flow
##  Min.   :176.5                Min.   :292.4
##  1st Qu.:250.8                1st Qu.:298.3
##  Median :298.7                Median :299.8
##  Mean   :281.1                Mean   :299.5
##  3rd Qu.:300.4                3rd Qu.:300.7
##  Max.   :357.8                Max.   :305.9
##  Flotation.Column.05.Air.Flow Flotation.Column.06.Air.Flow
##  Min.   :286.9                Min.   :189.9
##  1st Qu.:298.1                1st Qu.:266.1
##  Median :299.9                Median :299.5
##  Mean   :299.9                Mean   :292.2
##  3rd Qu.:301.8                3rd Qu.:303.1
##  Max.   :309.9                Max.   :370.9
##  Flotation.Column.07.Air.Flow Flotation.Column.01.Level
##  Min.   :186.1                Min.   :149.2
##  1st Qu.:256.5                1st Qu.:416.9
##  Median :299.0                Median :492.0
##  Mean   :290.8                Mean   :520.3
##  3rd Qu.:301.9                3rd Qu.:594.2
##  Max.   :371.5                Max.   :862.2
##  Flotation.Column.02.Level Flotation.Column.03.Level
##  Min.   :211.0             Min.   :126.7
##  1st Qu.:442.2             1st Qu.:411.7
##  Median :496.2             Median :494.4
##  Mean   :522.8             Mean   :531.4
##  3rd Qu.:595.1             3rd Qu.:601.2
##  Max.   :828.8             Max.   :886.8
##  Flotation.Column.04.Level Flotation.Column.05.Level
##  Min.   :162.3             Min.   :167.0
```

```
##  1st Qu.:356.8             1st Qu.:357.7
##  Median :412.4             Median :409.2
##  Mean   :420.8             Mean   :425.4
##  3rd Qu.:485.7             3rd Qu.:484.7
##  Max.   :680.3             Max.   :675.6
##  Flotation.Column.06.Level Flotation.Column.07.Level X..Iron.Concentrate
##  Min.   :157.3             Min.   :175.3             Min.   :62.05
##  1st Qu.:358.6             1st Qu.:356.7             1st Qu.:64.37
##  Median :424.8             Median :411.1             Median :65.21
##  Mean   :429.9             Mean   :421.1             Mean   :65.05
##  3rd Qu.:492.5             3rd Qu.:476.6             3rd Qu.:65.87
##  Max.   :698.8             Max.   :659.6             Max.   :68.01
##  X..Silica.Concentrate
##  Min.   :0.600
##  1st Qu.:1.440
##  Median :1.990
##  Mean   :2.325
##  3rd Qu.:3.000
##  Max.   :5.530
```

```
dim(df) #gives the row, col dimensions
```

```
## [1] 100000     24
```

Data Cleaning Process

I used is.na(x) function to check for count of NAs per column. It looks like the data set has no NA values. The first column contains the date and it is removed because it is just for tracking purpose and I do not need time stamp in this data set. Using the coorelation function we can see that there are alot of columns that have coorelation greater than 50. So, for column 8 through 14 I am going to take the average of those columns and make a separate column that contains the average instead of those 7 columns. So, two new columns created are mean_airflow and mean_level. Then I am only selecting the column of our concern. I successfully removed 14 columns and replaced them with two columns that represents average of those columns. I ran correlation to see if there is any correlation between the columns again. It turns out that column 1 and 4 still had correlation with column 7 and 2 respectively. Column 1 and 2 represents the amount of Iron and Silica feed prior the to the mining process so its okay to have a correlation. In short, I have not removed any more columns.

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
sapply(df, function(x) sum(is.na(x))) #checking # of NAs per column
```

```
##                          date                X..Iron.Feed
##                             0                           0
##               X..Silica.Feed                 Starch.Flow
##                             0                           0
```

```
##                   Amina.Flow                Ore.Pulp.Flow
##                            0                            0
##                  Ore.Pulp.pH             Ore.Pulp.Density
##                            0                            0
## Flotation.Column.01.Air.Flow Flotation.Column.02.Air.Flow
##                            0                            0
## Flotation.Column.03.Air.Flow Flotation.Column.04.Air.Flow
##                            0                            0
## Flotation.Column.05.Air.Flow Flotation.Column.06.Air.Flow
##                            0                            0
## Flotation.Column.07.Air.Flow    Flotation.Column.01.Level
##                            0                            0
##    Flotation.Column.02.Level    Flotation.Column.03.Level
##                            0                            0
##    Flotation.Column.04.Level    Flotation.Column.05.Level
##                            0                            0
##    Flotation.Column.06.Level    Flotation.Column.07.Level
##                            0                            0
##          X..Iron.Concentrate        X..Silica.Concentrate
##                            0                            0
```

```
#removing first column.

df <- df[,-1]

#looking for corelations
corMatrix <- cor(df[])
findCorrelation(corMatrix, cutoff=0.5, verbose=TRUE)
```

```
## Compare row 8  and column  10 with corr  0.956
##   Means:  0.295 vs 0.191 so flagging column 8
## Compare row 10  and column  13 with corr  0.659
##   Means:  0.263 vs 0.182 so flagging column 10
## Compare row 13  and column  9 with corr  0.586
##   Means:  0.256 vs 0.176 so flagging column 13
## Compare row 9  and column  14 with corr  0.577
##   Means:  0.211 vs 0.168 so flagging column 9
## Compare row 14  and column  11 with corr  0.555
##   Means:  0.196 vs 0.165 so flagging column 14
## Compare row 15  and column  17 with corr  0.728
##   Means:  0.241 vs 0.158 so flagging column 15
## Compare row 17  and column  16 with corr  0.655
##   Means:  0.2 vs 0.15 so flagging column 17
## Compare row 19  and column  21 with corr  0.712
##   Means:  0.218 vs 0.142 so flagging column 19
## Compare row 21  and column  20 with corr  0.606
##   Means:  0.171 vs 0.134 so flagging column 21
## Compare row 20  and column  18 with corr  0.512
##   Means:  0.132 vs 0.132 so flagging column 20
## Compare row 4  and column  7 with corr  0.656
##   Means:  0.181 vs 0.13 so flagging column 4
## Compare row 23  and column  22 with corr  0.801
##   Means:  0.128 vs 0.119 so flagging column 23
## Compare row 2  and column  1 with corr  0.972
```

```
##   Means:  0.167 vs 0.115 so flagging column 2
## All correlations <= 0.5
```

```
##  [1]  8 10 13  9 14 15 17 19 21 20  4 23  2
```

```
#making new columns : df$mean_airflow and df$mean_level
df$mean_airflow <- rowMeans(df[c('Flotation.Column.01.Air.Flow', 'Flotation.Column.02.Air.Flow','Flotat
length(df$mean_airflow) # checking to see that the acutal number of rows is preserved.
```

```
## [1] 100000
```

```
df$mean_level <- rowMeans(df[c('Flotation.Column.01.Level', 'Flotation.Column.02.Level','Flotation.Colu
length(df$mean_level)
```

```
## [1] 100000
```

```
# only selecting the columns of our concern
df <- df[,c(1:7,22:25)]

#looking for correlation again
corMatrix <- cor(df[,c(1:8,10,11)])
findCorrelation(corMatrix, cutoff=0.5, verbose=TRUE)
```

```
## Compare row 4  and column  7 with corr  0.656
##   Means:  0.219 vs 0.148 so flagging column 4
## Compare row 1  and column  2 with corr  0.972
##   Means:  0.182 vs 0.126 so flagging column 1
## All correlations <= 0.5
```

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

Data Exploration Functions (Applied on selected subset of the original data)

```
names(df) # lists the column names.
```

```
##  [1] "X..Iron.Feed"          "X..Silica.Feed"
##  [3] "Starch.Flow"           "Amina.Flow"
##  [5] "Ore.Pulp.Flow"         "Ore.Pulp.pH"
##  [7] "Ore.Pulp.Density"      "X..Iron.Concentrate"
##  [9] "X..Silica.Concentrate" "mean_airflow"
## [11] "mean_level"
```

```
head(df, n = 10) #see first 10 rows.
```

```
##        X..Iron.Feed X..Silica.Feed Starch.Flow Amina.Flow Ore.Pulp.Flow
## 16170         58.95           8.94   4635.5500   440.7350      392.2850
## 333161        64.03           6.26   3889.8800   453.5600      409.0330
## 205893        52.61          20.24    704.4713   244.2354      402.2460
## 201083        56.65          14.83   4353.0700   513.9110      392.0210
## 451972        53.63          15.74   3176.0000   603.0270      410.7420
## 622779        48.81          25.31   2821.5800   397.4340      376.8199
## 660352        53.06          19.28   3665.0600   561.9120      386.0540
## 683494        59.67          11.33   2690.6600   411.5490      379.5064
## 77169         56.45          13.57   2135.8100   399.7800      391.1190
## 323892        64.03           6.26   1684.8157   608.7130      401.1250
##        Ore.Pulp.pH Ore.Pulp.Density X..Iron.Concentrate
## 16170      9.83961          1.74357               65.15
## 333161    10.28440          1.73619               66.25
## 205893     9.10214          1.52062               64.38
## 201083    10.22310          1.72824               64.28
## 451972     9.80207          1.75482               65.64
## 622779     9.82449          1.68545               63.79
## 660352     9.54210          1.70028               64.05
## 683494     9.06704          1.70473               65.37
## 77169      9.66602          1.77269               66.10
## 323892     9.44827          1.74261               64.75
##        X..Silica.Concentrate mean_airflow mean_level
## 16170                   3.01     264.7243   574.4274
## 333161                  1.25     300.4104   369.7894
## 205893                  3.82     277.1694   395.8054
## 201083                  3.17     278.3549   394.3213
## 451972                  1.51     298.7340   401.2321
## 622779                  2.38     300.3427   439.6667
## 660352                  2.72     299.7207   492.6599
## 683494                  1.67     299.4681   512.5663
## 77169                   1.39     264.9690   537.8699
## 323892                  2.05     293.4446   363.7743
```

```r
tail(df, n = 5) # see last 5 rows.
```

```
##        X..Iron.Feed X..Silica.Feed Starch.Flow Amina.Flow Ore.Pulp.Flow
## 706365        54.27          16.60     4285.86    620.235       413.151
## 50968         56.65          13.99     2648.29    539.982       402.711
## 421562        50.70          23.17     3598.22    484.569       407.334
## 568247        57.46          10.80     2126.01    479.340       380.497
## 462957        55.40          16.83     3294.51    613.766       412.811
##        Ore.Pulp.pH Ore.Pulp.Density X..Iron.Concentrate
## 706365     9.74205          1.76650               64.54
## 50968      9.85672          1.73074               66.29
## 421562     9.36561          1.59436               65.19
## 568247     9.75076          1.71953               65.44
## 462957     9.97342          1.70679               66.45
##        X..Silica.Concentrate mean_airflow mean_level
## 706365                  1.51     306.4822   446.3311
## 50968                   1.50     265.4709   519.0016
## 421562                  1.93     300.6219   398.5956
## 568247                  2.08     304.0874   494.3315
## 462957                  1.23     299.9480   533.6171
```

```r
str(df) #finding the structure of the data set.
```

```
## 'data.frame':    100000 obs. of  11 variables:
##  $ X..Iron.Feed         : num  59 64 52.6 56.6 53.6 ...
##  $ X..Silica.Feed       : num  8.94 6.26 20.24 14.83 15.74 ...
##  $ Starch.Flow          : num  4636 3890 704 4353 3176 ...
##  $ Amina.Flow           : num  441 454 244 514 603 ...
##  $ Ore.Pulp.Flow        : num  392 409 402 392 411 ...
##  $ Ore.Pulp.pH          : num  9.84 10.28 9.1 10.22 9.8 ...
##  $ Ore.Pulp.Density     : num  1.74 1.74 1.52 1.73 1.75 ...
##  $ X..Iron.Concentrate  : num  65.2 66.2 64.4 64.3 65.6 ...
##  $ X..Silica.Concentrate: num  3.01 1.25 3.82 3.17 1.51 2.38 2.72 1.67 1.39 2.05 ...
##  $ mean_airflow         : num  265 300 277 278 299 ...
##  $ mean_level           : num  574 370 396 394 401 ...
```

```r
summary(df) # summary() function provides a number of useful statistics including range, median, and me
```

```
##   X..Iron.Feed   X..Silica.Feed   Starch.Flow         Amina.Flow
##  Min.   :42.74   Min.   : 1.31   Min.   :   0.002   Min.   :241.7
##  1st Qu.:52.67   1st Qu.: 8.98   1st Qu.:2073.622   1st Qu.:430.8
##  Median :56.02   Median :13.93   Median :3018.155   Median :504.1
##  Mean   :56.28   Mean   :14.67   Mean   :2867.724   Mean   :487.7
##  3rd Qu.:59.72   3rd Qu.:19.94   3rd Qu.:3728.253   3rd Qu.:553.0
##  Max.   :65.78   Max.   :33.40   Max.   :6300.230   Max.   :739.4
##  Ore.Pulp.Flow    Ore.Pulp.pH     Ore.Pulp.Density X..Iron.Concentrate
##  Min.   :376.2   Min.   : 8.753   Min.   :1.520    Min.   :62.05
##  1st Qu.:394.2   1st Qu.: 9.528   1st Qu.:1.647    1st Qu.:64.37
##  Median :399.3   Median : 9.800   Median :1.698    Median :65.21
##  Mean   :397.6   Mean   : 9.770   Mean   :1.680    Mean   :65.05
##  3rd Qu.:402.9   3rd Qu.:10.041   3rd Qu.:1.728    3rd Qu.:65.87
##  Max.   :418.6   Max.   :10.808   Max.   :1.853    Max.   :68.01
##  X..Silica.Concentrate  mean_airflow     mean_level
##  Min.   :0.600         Min.   :218.6   Min.   :263.4
##  1st Qu.:1.440         1st Qu.:277.8   1st Qu.:399.6
##  Median :1.990         Median :298.7   Median :456.0
##  Mean   :2.325         Mean   :288.7   Mean   :467.4
##  3rd Qu.:3.000         3rd Qu.:300.5   3rd Qu.:518.1
##  Max.   :5.530         Max.   :325.8   Max.   :745.8
```

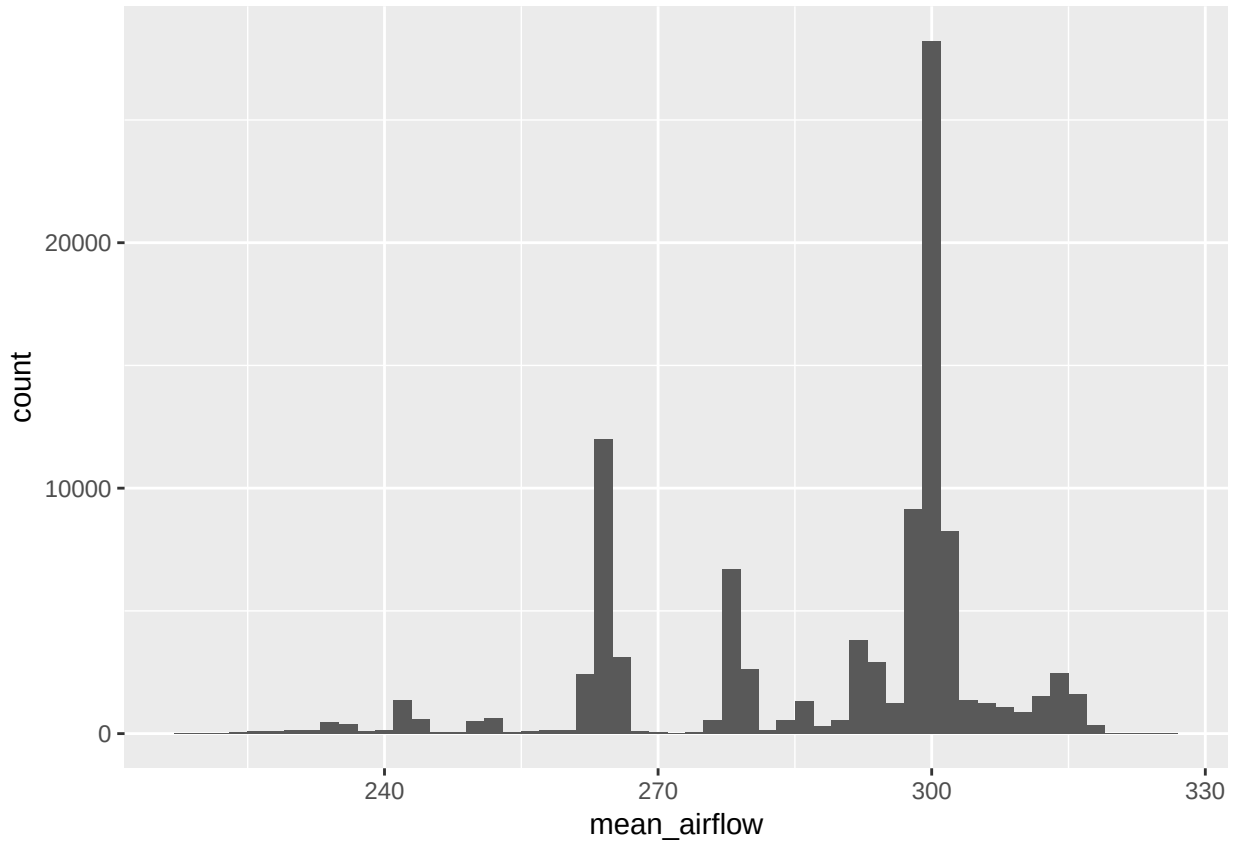
```r
dim(df) #gives the row, col dimensions
```

```
## [1] 100000     11
```

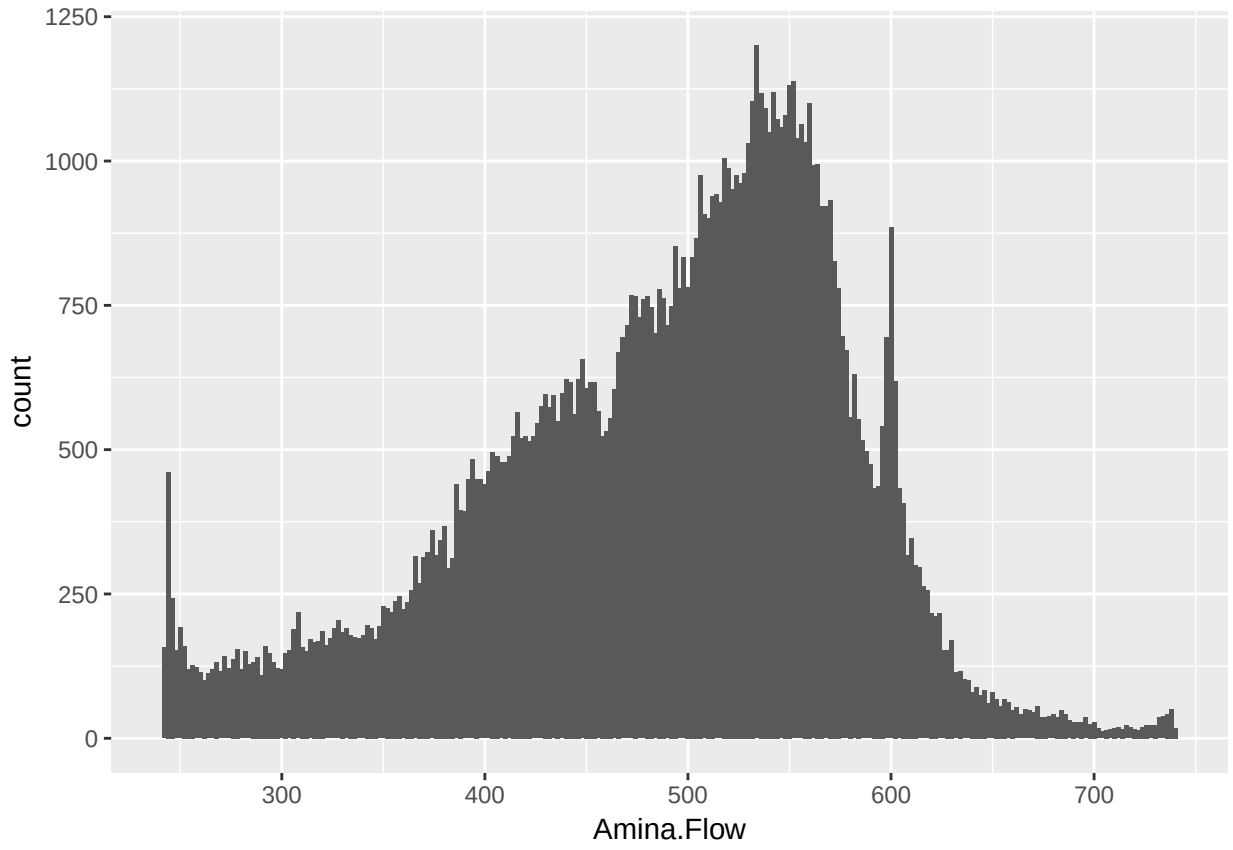
Visual Data Exploration

```r
#Source of some ggplot codes are : https://rstudio-pubs-static.s3.amazonaws.com/278745_60156813ccd2466e

par = par(mfrow=c(2,3))
#Spread of mean_airflow
ggplot(df,aes(df[,10])) + geom_histogram(binwidth = 2) + xlab(colnames(df)[10])
```

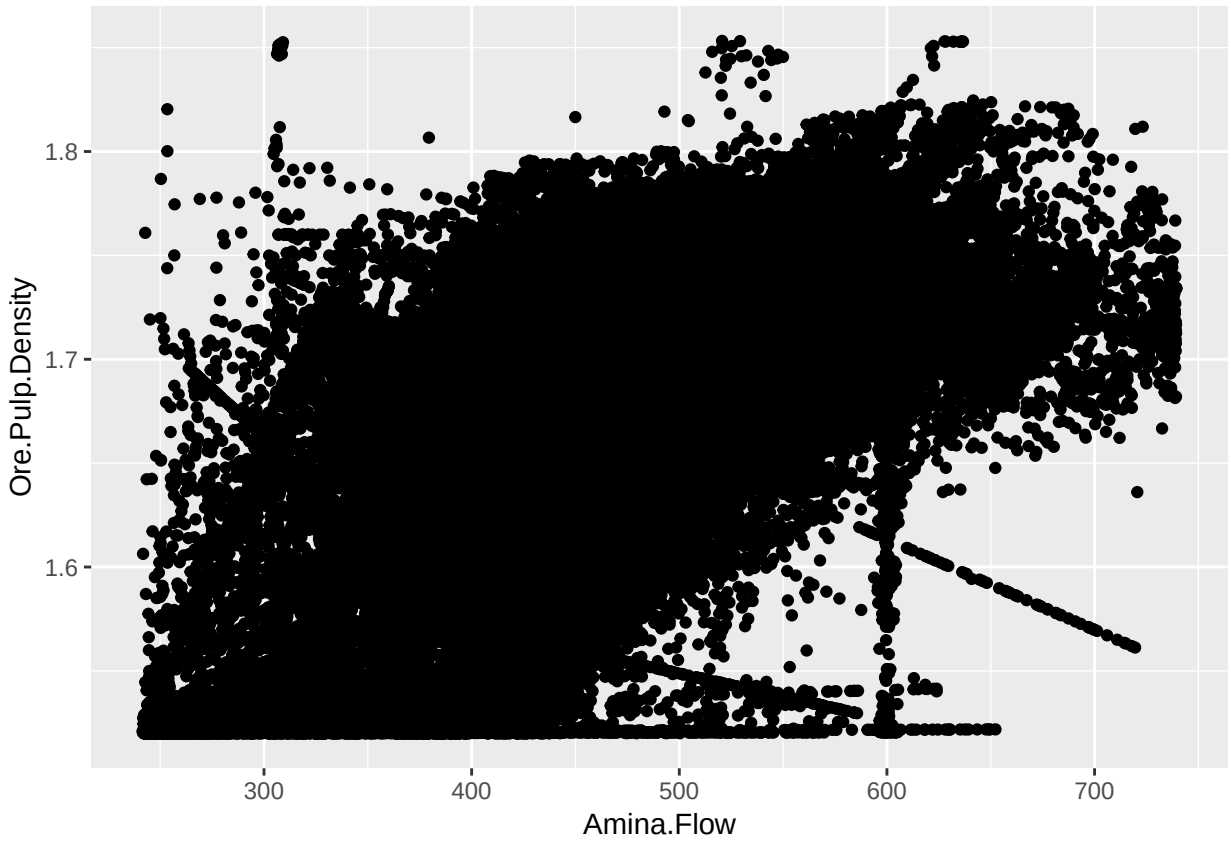



```
#Spread of Amina.Flow
ggplot(df,aes(df[,4])) + geom_histogram(binwidth = 2) + xlab(colnames(df)[4])
```




```
#Scatter plot of Silica Concentrate and Iron Feed
plot(df$X..Silica.Concentrate~df$X..Iron.Feed, pch = '*', col = "Blue",xlab = "Iron Feed", ylab = "Sili
# Scatter plot of Silica Concentration and Silica Feed.
plot(df$X..Silica.Concentrate~df$X..Silica.Feed, pch='+', col="Red", xlab="Silica Feed", ylab="Silica C

#Correlation between DEWP concentration and TEMP
print(ggplot(df, aes(df[,4], df[,7])) + geom_point() + xlab(colnames(df)[4]) + ylab(colnames(df)[7]))
```



Linear Regression

Divide into train and test (Using the same sample for all algorithms).

Predictors selected are : [1] “X..Iron.Feed”
[2] “X..Silica.Feed”
[3] “Starch.Flow”
[4] “Amina.Flow”
[5] “Ore.Pulp.Flow”
[6] “Ore.Pulp.pH”
[7] “Ore.Pulp.Density”
[8] “X..Iron.Concentrate”
[9] “mean_airflow”
[10] “mean_level”

The variable to be predicted is : [1] “X..Silica.Concentrate” The reason for selecting those features is as follows: - mean_airflow is the average of all columns from column 8 to 14 in original data set. - mean_level is the average of all columns from column 15 to 21 in original data set. - After some understanding of the domain knowledge, it seems fair to include every column in the data set for prediction. - I am predicting Silica Concentration based on all columns.

```
set.seed(1234)  # setting a seed gets the same results every time
i <- sample(1: nrow(df),0.80 * nrow(df), replace = TRUE)
```

```
#Creating train and test split for linear regression
linear_train <- df[i,]
linear_test <- df[-i,]
```

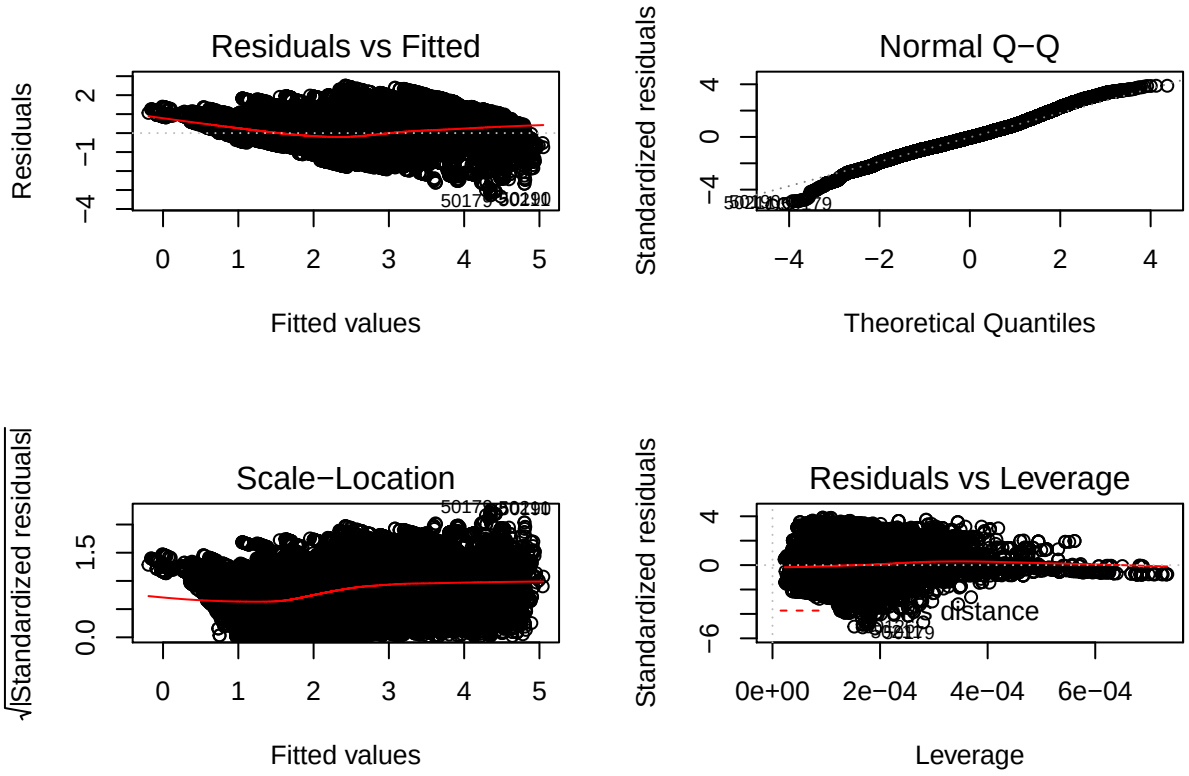
Key Points: - p-value was low for all columns except starch.flow, ore.pulp.ph and ore.pulp.density. - RSE was calculated as 0.643. R-Square was calculated as 0.6712. R-squared tells us that aproximately 67% of the variance in the model can be explained by our predictor which is not that bad.The residual standard error, RSE, is in units of y. In this case our RSE was 0.643, so the average error of the model was about aproximately .64 percentage. This statistic was calculated on 79989 degrees of freedom: we had 80000 data points minus 11 predictors. - F-statistic: 1.633e+04 which is far from 0 so is good. It provides evidence against the null hypothesis.

Building linear model on train data

```
set.seed(1234)
linear_model <- lm(X..Silica.Concentrate~., data = linear_train)
summary(linear_model)
```

```
##
## Call:
## lm(formula = X..Silica.Concentrate ~ ., data = linear_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3388 -0.4202 -0.0321  0.3825  2.4991
##
## Coefficients:
##                       Estimate Std. Error  t value Pr(>|t|)
## (Intercept)          5.250e+01  2.177e-01  241.184  < 2e-16 ***
## X..Iron.Feed         2.568e-02  1.897e-03   13.533  < 2e-16 ***
## X..Silica.Feed       2.838e-02  1.433e-03   19.812  < 2e-16 ***
## Starch.Flow         -1.412e-05  2.052e-06   -6.884 5.86e-12 ***
## Amina.Flow           5.835e-04  3.709e-05   15.730  < 2e-16 ***
## Ore.Pulp.Flow        4.337e-03  2.567e-04   16.892  < 2e-16 ***
## Ore.Pulp.pH          4.220e-02  6.280e-03    6.719 1.84e-11 ***
## Ore.Pulp.Density     1.943e-01  4.610e-02    4.216 2.49e-05 ***
## X..Iron.Concentrate -7.914e-01  2.137e-03 -370.259  < 2e-16 ***
## mean_airflow        -9.695e-03  1.436e-04  -67.497  < 2e-16 ***
## mean_level          -9.910e-04  3.400e-05  -29.145  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6433 on 79989 degrees of freedom
## Multiple R-squared:  0.6728, Adjusted R-squared:  0.6727
## F-statistic: 1.645e+04 on 10 and 79989 DF,  p-value: < 2.2e-16
```

```
par (mfrow = c(2,2)) # 2*2 grid.
plot(linear_model)
```



Analysis of the plot : - Residual vs Fitted : Although the red line doesn't align with the dotted white line, it is fairly horizontal as the fitted values increases. - Normal Q-Q: The residuals follows the dotted line (except in the beginning). It means that residuals are normally distributed. - Scale-Location: We want to see a fairly horizontal red line with the residuals distributed equally around it. The plot suggests that data is homoscedastic. - Residuals vs Leverage : Outliers is unusual y value and leverage is unusual x value. The red line is fairly horizontal and follows the dotted white line.

Predict on test data

```
linear_pred <- predict(linear_model,newdata = linear_test)
```

Metrics for test set evaluation

One of the metrics that is used to evalute the linear regression is correlation. The correlation for the linear model is calculated as 0.82. The mse and rmse are calculated as 0.416 and 0.645 respectively. The MSE tells that the model is off by 0.42% (of silica concentration).

```
#finding correlation
linear_cor <- cor(linear_pred,linear_test$X..Silica.Concentrate)
#finding mse - mean square error
linear_mse <- mean((linear_pred-linear_test$X..Silica.Concentrate)^2)
linear_rmse <- sqrt(linear_mse)
print(paste("MSE and RMSE for linear model :",linear_mse, linear_rmse))
```

```
## [1] "MSE and RMSE for linear model : 0.413658918841583 0.643163213221638"
```

Ridge Regression

Using the same divide (80/20) for train and test.

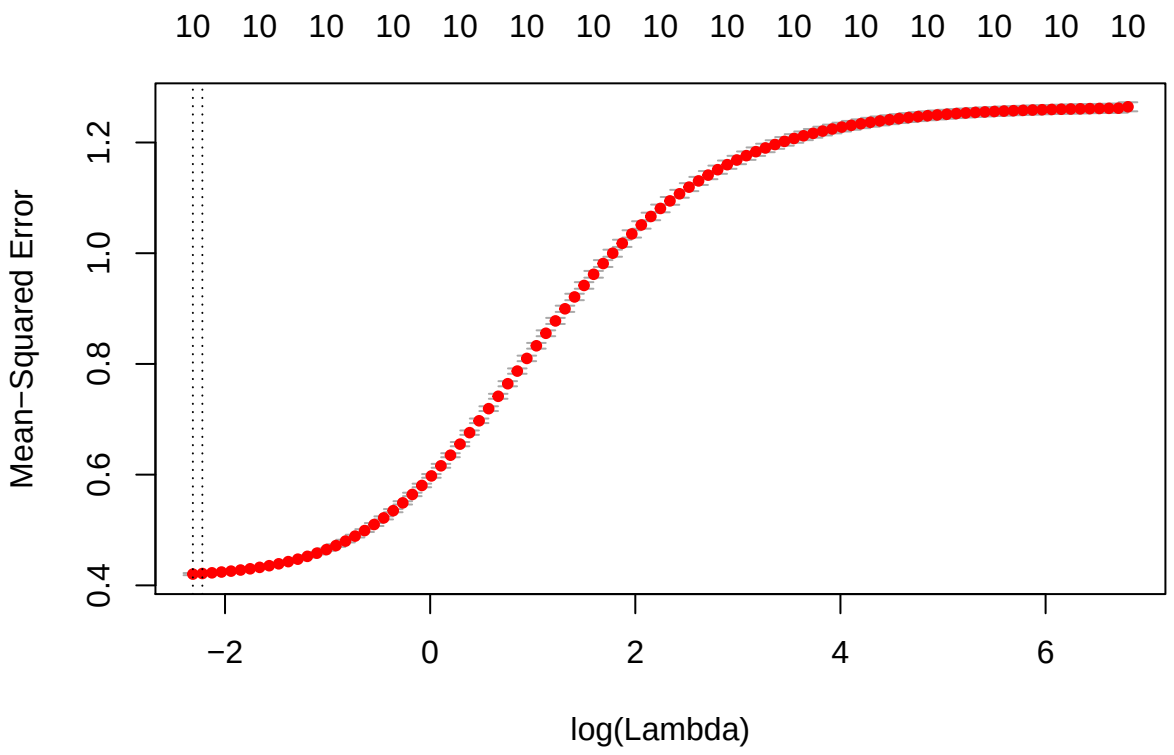
```
library(glmnet)
```

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

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```
x <- model.matrix(X..Silica.Concentrate~., df)
y <- df$X..Silica.Concentrate
train_x <- x[i,]
train_y <- y[i]
test_x <- x[-i,]
test_y <- y[-i]
set.seed(1234)
#build a ridge regression model.
rm<- glmnet(train_x,train_y,alpha=0)
#use cv to see which lambda is best
cv_results <- cv.glmnet(train_x,train_y,alpha =0)
plot(cv_results)
```




```
l <- cv_results$lambda.min

#get data for best lambda, which is the 100th
```

```
#as determined by looking at rm$lamda

pred <- predict(rm,s=l,newx = test_x)
mse <- mean((pred-test_y)^2)
rmse <-sqrt(mse)
coef2 <- coef(rm)[,100]
ridge_cor <- cor(pred,test_y)
print(paste("MSE and RMSE for ridge regression model :",mse, rmse))
```

```
## [1] "MSE and RMSE for ridge regression model : 0.420167311638019 0.648203140719033"
```

Compare mse and coefficients

```
print(paste("MSE for linear regression = ", linear_mse))
```

```
## [1] "MSE for linear regression =  0.413658918841583"
```

```
coef(linear_model)
```

```
##         (Intercept)        X..Iron.Feed      X..Silica.Feed
##        5.250002e+01        2.567531e-02        2.838316e-02
##         Starch.Flow          Amina.Flow       Ore.Pulp.Flow
##       -1.412531e-05        5.834815e-04        4.336508e-03
##         Ore.Pulp.pH    Ore.Pulp.Density X..Iron.Concentrate
##        4.219525e-02        1.943413e-01       -7.913582e-01
##        mean_airflow          mean_level
##       -9.695277e-03       -9.909639e-04
```

```
print(paste("MSE for ridge regression = ",mse))
```

```
## [1] "MSE for ridge regression =  0.420167311638019"
```

```
coef2
```

```
##         (Intercept)         (Intercept)        X..Iron.Feed
##        5.049271e+01        0.000000e+00        1.492628e-03
##      X..Silica.Feed         Starch.Flow          Amina.Flow
##        9.967410e-03       -1.529416e-05        6.877113e-04
##       Ore.Pulp.Flow         Ore.Pulp.pH    Ore.Pulp.Density
##        3.775504e-03        6.900071e-03        8.516693e-02
## X..Iron.Concentrate        mean_airflow          mean_level
##       -7.288940e-01       -8.838578e-03       -9.230337e-04
```

The mean squared error as well as RMSE remains almost same for ridge regression model and linear regression model.

kNN

Key Points: The same train and test is used as in linear and ridge regression model. kNN is an instance based learning and doesn't create a model. So, it makes on assumption about the shape of the data. The goal is to maximize the correlation and minimize the mse. I am choosing k=3 and running the kNN in unscaled data set first.

```
library(caret)
library(DMwR)
```

```
## Loading required package: grid
```

```
fit <- knnreg(linear_train[,c(1:8,10,11)],linear_train[,9], k = 3)
predictions <- predict(fit,linear_test[,c(1:8,10,11)])
cor(predictions,linear_test$X..Silica.Concentrate)
```

```
## [1] 0.4397259
```

Additional Metrices

```
knn_mse <- mean((predictions - linear_test$X..Silica.Concentrate)^2)
knn_rmse <- sqrt(knn_mse)
print(paste("MSE and RMSE for kNN regression :",knn_mse, knn_rmse))
```

```
## [1] "MSE and RMSE for kNN regression : 1.19112185796251 1.09138529308513"
```

The correlation obtained is only 0.43, which is way worse than linear and ridge regression models. The MSE is also worse than both previous models. kNN suggests that it is off by 1.21% (silica concentration). However, it is true that know that clustering algorithm works better on a scaled data. So, the next step will be to scale the data.

Scaling the data

```
scaled_data <- scale(df[]) # using scale function to scale the data set.
df <- data.frame(scaled_data)
train <- df[i,] # selecting train (scaled)
test <- df[-i,] #selecting test (scaled)
fit <- knnreg(train[,c(1:8,10,11)],train[,9], k =3) #selecting k=3 and running kNN regression
p <- predict(fit, test[,c(1:8,10,11)]) # predicting..
knn_3 <- cor(p,test$X..Silica.Concentrate) # finding the coorelation
mse <- mean((p-test$X..Silica.Concentrate)^2) # finding mse.
rmse <- sqrt(mse) # finding rmse for comaprison
mse
```

```
## [1] 0.06198492
```

```
rmse
```

```
## [1] 0.2489677
```

WOW! kNN performed way better on a scaled data set. The correlation obtained after scaling the data is 0.96. The MSE and RMSE obtained is 0.06 and 0.24 respectively. This is the result for kNN with k=3. Now, lets try different values of k and see which k value is the best. I am trying odd k values in between 1 and 39.

Finding the best k

Try various values of k and plot the results.

```
cor_k <- rep(0, 20)
mse_k <- rep(0, 20)
i <- 1
for (k in seq(1, 39, 2)){
  fit_k <- knnreg(train[,c(1:8,10,11)],train[,9], k=k)
  pred_k <- predict(fit_k, test[,c(1:8,10,11)])
  cor_k[i] <- cor(pred_k, test$X..Silica.Concentrate)
  mse_k[i] <- mean((pred_k - test$X..Silica.Concentrate)^2)
  print(paste("k=", k, cor_k[i], mse_k[i]))
  i <- i + 1
}
```

```
## [1] "k= 1 0.971511083978621 0.0568747513086576"
## [1] "k= 3 0.968608346214981 0.0619849204250468"
## [1] "k= 5 0.965732846848371 0.0673606145723374"
## [1] "k= 7 0.962608962028702 0.0733349757434112"
## [1] "k= 9 0.95936983012156 0.0795975349280133"
## [1] "k= 11 0.956312502591422 0.0855418307249453"
## [1] "k= 13 0.953151317056159 0.0917063660864837"
## [1] "k= 15 0.950405403542599 0.097085631173001"
## [1] "k= 17 0.947642680039917 0.102489470637187"
## [1] "k= 19 0.945106443551081 0.107477383647191"
## [1] "k= 21 0.942732825996556 0.112158131243171"
## [1] "k= 23 0.940421117619751 0.116728262412204"
## [1] "k= 25 0.938317633544532 0.120900085044471"
## [1] "k= 27 0.93615134201873 0.125144929928669"
## [1] "k= 29 0.934045586375457 0.129272368629252"
## [1] "k= 31 0.932205677305674 0.1329347309137"
## [1] "k= 33 0.930483177436367 0.136369847094487"
## [1] "k= 35 0.92871322521753 0.139842014026877"
## [1] "k= 37 0.927183067785976 0.142849037140756"
## [1] "k= 39 0.925805572578983 0.145621259849147"
```

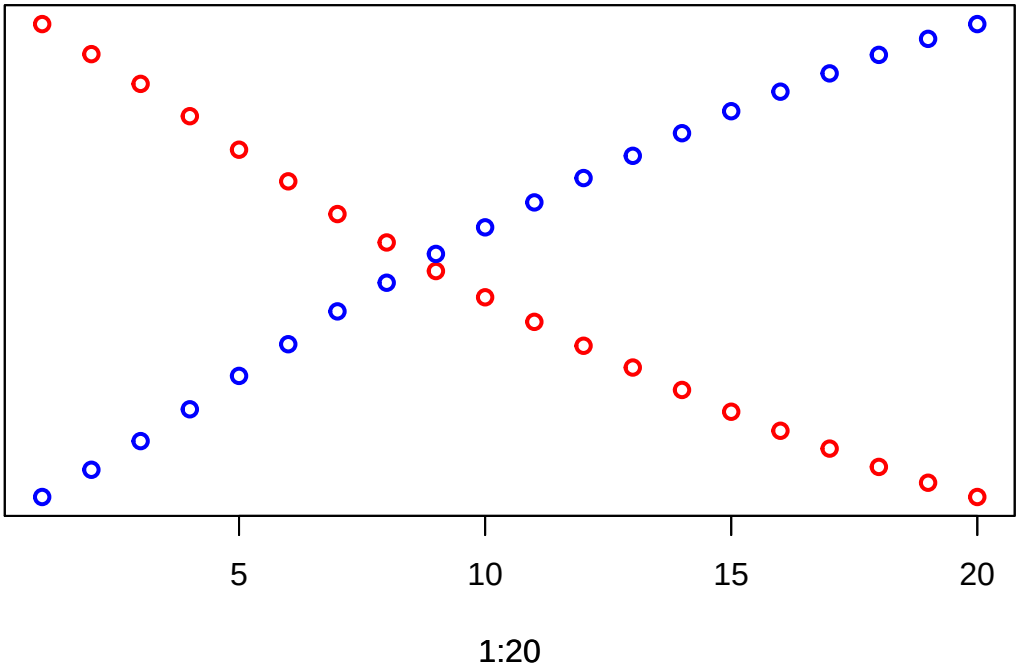
```
plot(1:20, cor_k, lwd=2, col='red', ylab="", yaxt='n')
par(new=TRUE)
plot(1:20, mse_k, lwd=2, col='blue', labels=FALSE, ylab="", yaxt='n')
```

```
## Warning in plot.window(...): "labels" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "labels" is not a graphical parameter

## Warning in box(...): "labels" is not a graphical parameter

## Warning in title(...): "labels" is not a graphical parameter
```



Turns out that k=1 is the best with correlation of .9697 and mse of 0.0604. The correlation seems to decrease as the k value increases.

[1] "k= 1 0.96978118940472 0.0603883607842249" [1] "k= 3 0.966998994607761 0.0651318680136435" [1] "k= 5 0.963435215367703 0.0718097971825848" [1] "k= 7 0.959881924782445 0.078588035972286" [1] "k= 9 0.956705874747346 0.0846819704473681" [1] "k= 11 0.95393788461106 0.0900399527780337" [1] "k= 13 0.951064128754392 0.0956365824177667" [1] "k= 15 0.948274327484662 0.101085551028143" [1] "k= 17 0.945594511950433 0.106326912963887" [1] "k= 19 0.943076988091601 0.111284952162709" [1] "k= 21 0.940433499058742 0.116422531268831" [1] "k= 23 0.938164275270582 0.120891162198145" [1] "k= 25 0.936123820664801 0.124924295724734" [1] "k= 27 0.933993506124818 0.12908199786376" [1] "k= 29 0.932141253562812 0.132726204798412" [1] "k= 31 0.930382059639316 0.136222971768213" [1] "k= 33 0.928651835189829 0.1396407635996" [1] "k= 35 0.926914297078217 0.143056203571157" [1] "k= 37 0.925293111584829 0.146286862595958" [1] "k= 39 0.923887490729068 0.149085124090067"

kNN for k=1

```
fit_1 <- knnreg(train[,c(1:8,10,11)],train[,9], k=1)
predictions_1 <- predict(fit_1,test[,c(1:8,10,11)])
cor_1 <- cor(predictions_1,test$X..Silica.Concentrate)
mse_1 <- mean((predictions_1-test$X..Silica.Concentrate)^2)
```

Comparing co-relation and MSE for linear, ridge and kNN regression

```
print(paste("Co-relation and MSE for linear regreesion are : ",linear_mse, linear_cor))
```

```
## [1] "Co-relation and MSE for linear regreesion are :  0.413658918841583 0.820232644838594"
```

```
print(paste("Co-relation and MSE for ridge regression are: ",ridge_cor, mse ))
```

```
## [1] "Co-relation and MSE for ridge regression are:  0.81959231002691 0.0619849204250468"
```

```
print(paste("Co-relation and MSE for kNN where k=3 (initial)",knn_3,knn_mse ))
```

```
## [1] "Co-relation and MSE for kNN where k=3 (initial) 0.968608346214981 1.19112185796251"
```

```
print(paste("Co-relation and MSE for kNN where k=1 (best)", cor_1,mse_1))
```

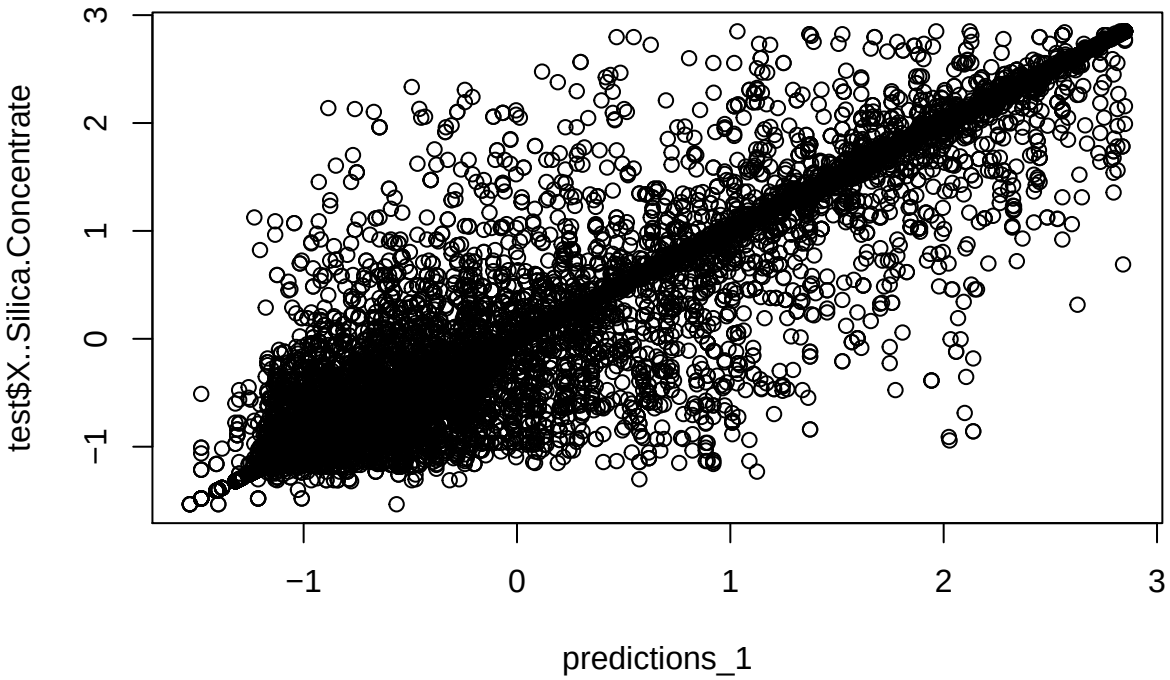
```
## [1] "Co-relation and MSE for kNN where k=1 (best) 0.971511083978621 0.0568747513086576"
```

Analysis of the best algorithm:

The algorithm that was able to obtain the highest correlation and lowset mean squared error was kNN with k=1. kNN doesn not build a model of the data as it is an example of an "instance-based" learning. kNN was able to get the best correlation as compared to linear and ridge regression because the later algorithms assumes a linear relationship between predictors and target. (However, linear doesn't mean straight) On the other hand, kNN makes no assumption about the shape of the data. Also, the data is scaled so it was able to preform better.

What was learnt from the data

```
plot(predictions_1,test$X..Silica.Concentrate)
```



kNN is not an easy algorithm to interpret. The plot above, of the predicted % of Silica and acutal data from test data set, further confirms the good correlation between test and prediction. It can be concluded that the predictors that I choose for the model (not technically a model because kNN doesn't create one) were good predictors. In other words, % of silica in the iron ore concentrate is affected by all ten predictors. Looking at the linear regression, it can be further predicted that Iron Concentrate, mean_airflow, mean_level and Starch Flow had negative effect on the % of Silica. All other columns had positive effect on the target variable. However, it has to be taken into the consideration that the linear model was assuming linear relationship between those predictors and the target.