EdX HarvardX Data Science CYO Project - Concrete Compressive Strength

Michael O'Donohoe

6th September 2020

1.0 Introduction/Overview/Executive Summary

This R Markdown document covers the devlopment of multiple machine learning models using the Concrete Compressive Strength Dataset downloaded from the UCI Machine Learning Respository (https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength). The models were compared for accuracy in predicting concrete compressive strength based omn a number of input variables. Compressive Strength is a measure of the resistance of a material to failure under a compressive force (https://www.nrmca.org/aboutconcrete/cips/35p.pdf). This is a critical performance measurement for concrete used in construction and is typically measured by casting cubes or cylinders of concrete and measuring the compressive force required to cause failure. The following machine learning algorithms were employed to build models:

- Multiple Linear Regression (MLR)
- Artificial Neural Networks (Ann)
- Decision Trees
- Random Forest
- Support Vector Machine (SVM)
- K Nearest Neighbour (Knn)
- Naive-Bayes

For the multiple linear regression and artificial neural networks models, the concrete compressive strength data was left as continous data. However for the clssification algorithms (Decition Trees, Random Forest, Support Vector Machine, K nearest neighbour, Naive-Bayes), it was decided to Create four categories for the concrete compressive strength data based on typical classification found in industry:

- Fail - <5Mpa
- Low Strength 5-19MPa
- Standard Strength 20-49MPa
- High Strength >= 50MPa

Overall this projects looked braodly at using many different machine learning algorithms for the purposes of learninga nd comparison rather than focussing deeply on just one

(however the multiple linear regression model was evaluated in quite a significant level of depth)

2.0 Methods/Analysis

2.1 Install the required packages from CRAN

A number of R packages required for the analysis (tidyverse, caret and lubridate) were loaded to RStudio.

2.2 Load & Inryspect the Concrete Compressive Strength Dataset from the UCI Machine Learning Repository

```
ConDat <- read_excel("Data/Concrete_Data.xls")
```

Inspect the data

```
str(ConDat)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    1030 obs. of  9 variables:
##  $ Cement (component 1)(kg in a m^3 mixture)            : num  540 540 332
332 199 ...
##  $ Blast Furnace Slag (component 2)(kg in a m^3 mixture): num  0 0 142 142
132 ...
##  $ Fly Ash (component 3)(kg in a m^3 mixture)           : num  0 0 0 0 0 0
0 0 0 0 ...
##  $ Water  (component 4)(kg in a m^3 mixture)            : num  162 162 228
228 192 228 228 228 228 228 ...
##  $ Superplasticizer (component 5)(kg in a m^3 mixture)  : num  2.5 2.5 0 0
0 0 0 0 0 0 ...
##  $ Coarse Aggregate  (component 6)(kg in a m^3 mixture) : num  1040 1055 9
32 932 978 ...
##  $ Fine Aggregate (component 7)(kg in a m^3 mixture)    : num  676 676 594
594 826 ...
##  $ Age (day)                                            : num  28 28 270 3
65 360 90 365 28 28 28 ...
##  $ Concrete compressive strength(MPa, megapascals)      : num  80 61.9 40.
3 41.1 44.3 ...
```

```
head(ConDat)
```

```
## # A tibble: 6 x 9
##   `Cement (compon~ `Blast Furnace ~ `Fly Ash (compo~ `Water  (compon~
##              <dbl>            <dbl>            <dbl>            <dbl>
## 1              540                0                0              162
## 2              540                0                0              162
## 3              332.             142.               0              228
## 4              332.             142.               0              228
## 5              199.             132.               0              192
```

```
## 6             266              114                 0              228
## # ... with 5 more variables: `Superplasticizer (component 5)(kg in a m^3
## #   mixture)` <dbl>, `Coarse Aggregate (component 6)(kg in a m^3
## #   mixture)` <dbl>, `Fine Aggregate (component 7)(kg in a m^3
## #   mixture)` <dbl>, `Age (day)` <dbl>, `Concrete compressive
## #   strength(MPa, megapascals)` <dbl>
```

The dataset consists of 1030 obs. of 9 variables mean compressive strength is 35.8MPa(Range 2.3-82.6MPa). This seems reasonable as typical concrete compressive strength values for most normal applications are in the range 10-60MPa.

count NAs

```
sapply(ConDat, function(x){sum(is.na(x))})
```

```
##             Cement (component 1)(kg in a m^3 mixture)
##                                                     0
## Blast Furnace Slag (component 2)(kg in a m^3 mixture)
##                                                     0
##            Fly Ash (component 3)(kg in a m^3 mixture)
##                                                     0
##             Water  (component 4)(kg in a m^3 mixture)
##                                                     0
##   Superplasticizer (component 5)(kg in a m^3 mixture)
##                                                     0
##  Coarse Aggregate  (component 6)(kg in a m^3 mixture)
##                                                     0
##     Fine Aggregate (component 7)(kg in a m^3 mixture)
##                                                     0
##                                               Age (day)
##                                                     0
##       Concrete compressive strength(MPa, megapascals)
##                                                     0
```

There are no na values

count blank entries

```
sapply(ConDat, function(x){sum(x=='', na.rm=T)})
```

```
##             Cement (component 1)(kg in a m^3 mixture)
##                                                     0
## Blast Furnace Slag (component 2)(kg in a m^3 mixture)
##                                                     0
##            Fly Ash (component 3)(kg in a m^3 mixture)
##                                                     0
##             Water  (component 4)(kg in a m^3 mixture)
##                                                     0
##   Superplasticizer (component 5)(kg in a m^3 mixture)
##                                                     0
##  Coarse Aggregate  (component 6)(kg in a m^3 mixture)
```

```
##                                                         0
##     Fine Aggregate (component 7)(kg in a m^3 mixture)
##                                                         0
##                                               Age (day)
##                                                         0
##       Concrete compressive strength(MPa, megapascals)
##                                                         0
```

There are no blank entries No requirement to deal with NA's or missing values.

Rename the attributes to simplify & Recheck structure of the data

```
names(ConDat) <- c("Cement", "Blast_Furnace_Slag", "Fly_Ash", "Water", "Super
plasticizer", "Coarse_Aggregate", "Fine_Aggregate", "Age", "Compressive_Stren
gth")
summary(ConDat)
```

```
##      Cement      Blast_Furnace_Slag    Fly_Ash           Water
##  Min.   :102.0   Min.   :  0.0      Min.   :  0.00   Min.   :121.8
##  1st Qu.:192.4   1st Qu.:  0.0      1st Qu.:  0.00   1st Qu.:164.9
##  Median :272.9   Median : 22.0      Median :  0.00   Median :185.0
##  Mean   :281.2   Mean   : 73.9      Mean   : 54.19   Mean   :181.6
##  3rd Qu.:350.0   3rd Qu.:142.9      3rd Qu.:118.27   3rd Qu.:192.0
##  Max.   :540.0   Max.   :359.4      Max.   :200.10   Max.   :247.0
##  Superplasticizer Coarse_Aggregate Fine_Aggregate       Age
##  Min.   : 0.000   Min.   : 801.0   Min.   :594.0   Min.   :  1.00
##  1st Qu.: 0.000   1st Qu.: 932.0   1st Qu.:731.0   1st Qu.:  7.00
##  Median : 6.350   Median : 968.0   Median :779.5   Median : 28.00
##  Mean   : 6.203   Mean   : 972.9   Mean   :773.6   Mean   : 45.66
##  3rd Qu.:10.160   3rd Qu.:1029.4   3rd Qu.:824.0   3rd Qu.: 56.00
##  Max.   :32.200   Max.   :1145.0   Max.   :992.6   Max.   :365.00
##  Compressive_Strength
##  Min.   : 2.332
##  1st Qu.:23.707
##  Median :34.443
##  Mean   :35.818
##  3rd Qu.:46.136
##  Max.   :82.599
```

```
str(ConDat)
```

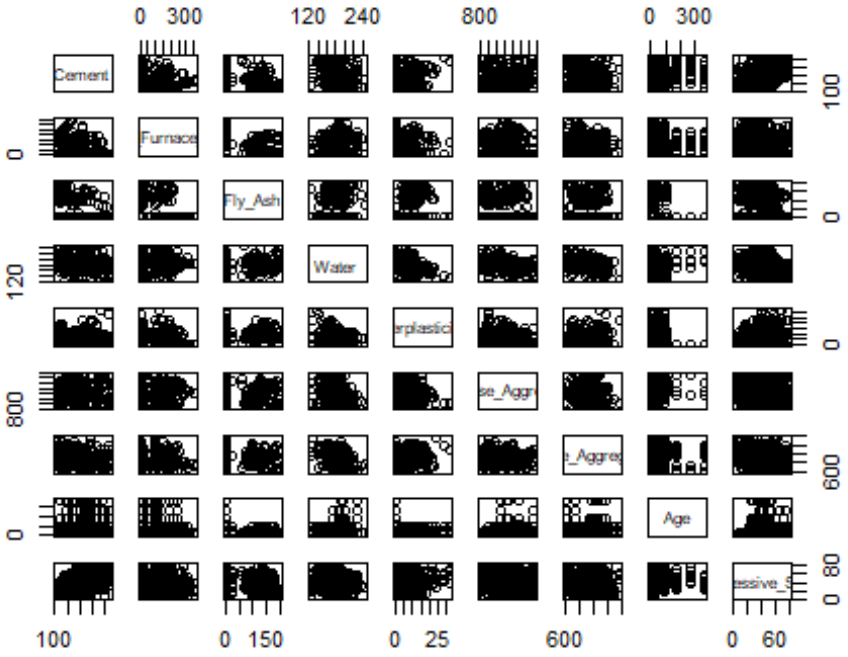
```
## Classes 'tbl_df', 'tbl' and 'data.frame':    1030 obs. of  9 variables:
##  $ Cement              : num  540 540 332 332 199 ...
##  $ Blast_Furnace_Slag  : num  0 0 142 142 132 ...
##  $ Fly_Ash             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Water               : num  162 162 228 228 192 228 228 228 228 228 ...
##  $ Superplasticizer    : num  2.5 2.5 0 0 0 0 0 0 0 0 ...
##  $ Coarse_Aggregate    : num  1040 1055 932 932 978 ...
##  $ Fine_Aggregate      : num  676 676 594 594 826 ...
##  $ Age                 : num  28 28 270 365 360 90 365 28 28 28 ...
##  $ Compressive_Strength: num  80 61.9 40.3 41.1 44.3 ...
```

```
head(ConDat)
```

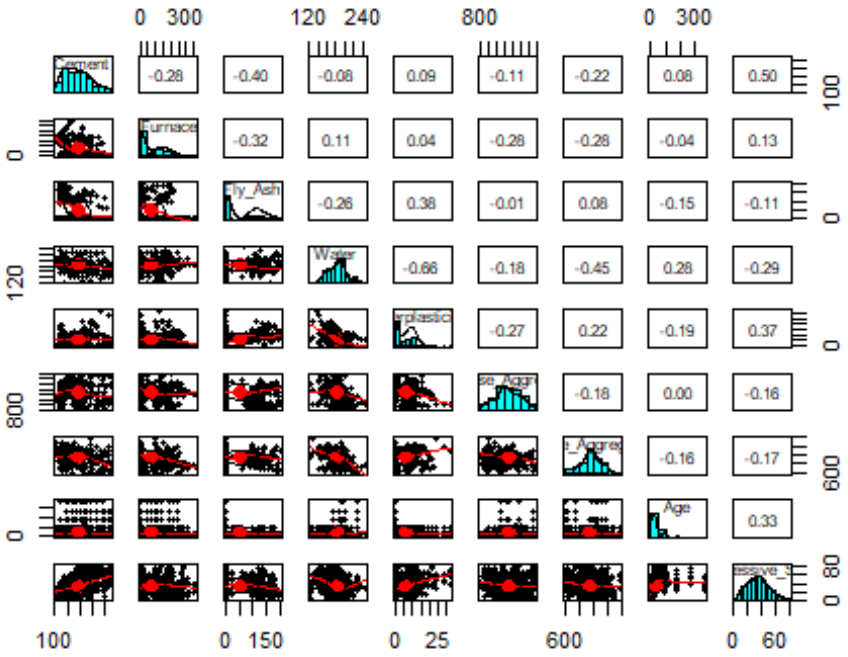
```
## # A tibble: 6 x 9
##   Cement Blast_Furnace_S~ Fly_Ash Water Superplasticizer Coarse_Aggregate
##    <dbl>            <dbl>   <dbl> <dbl>            <dbl>            <dbl>
## 1   540                0       0   162              2.5             1040
## 2   540                0       0   162              2.5             1055
## 3   332.             142.      0   228              0                932
## 4   332.             142.      0   228              0                932
## 5   199.             132.      0   192              0                978.
## 6   266              114       0   228              0                932
## # ... with 3 more variables: Fine_Aggregate <dbl>, Age <dbl>,
## #   Compressive_Strength <dbl>
```

2.3 Exploratory Data Analysis

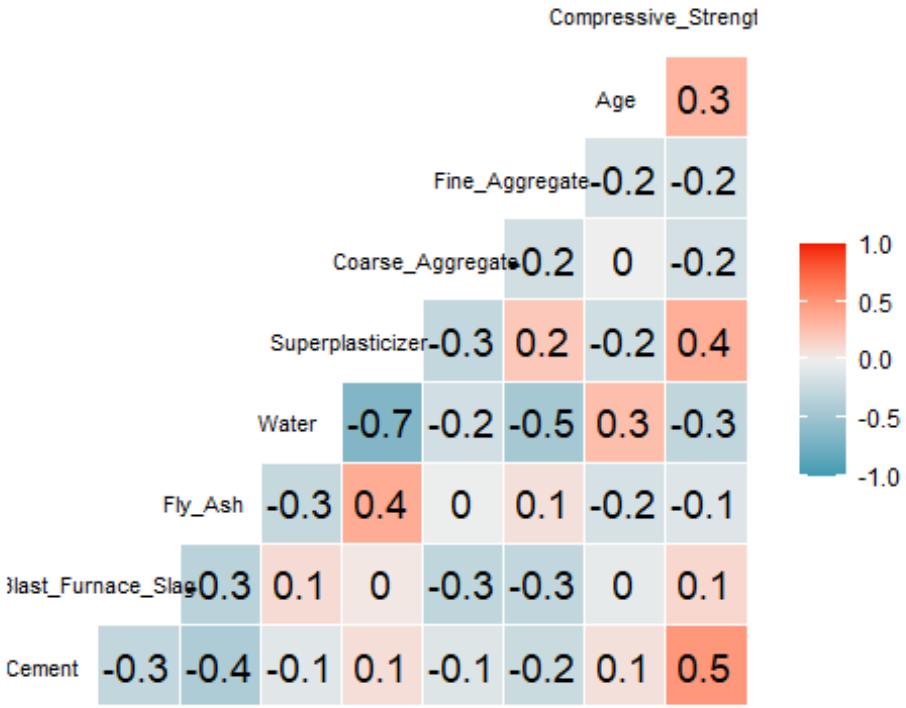
Use pairs function to provide visualization to check for correlations



Use pairs.panels function to provide visualization to check for correlations



Use GGally ggcorr function to visualize correlations

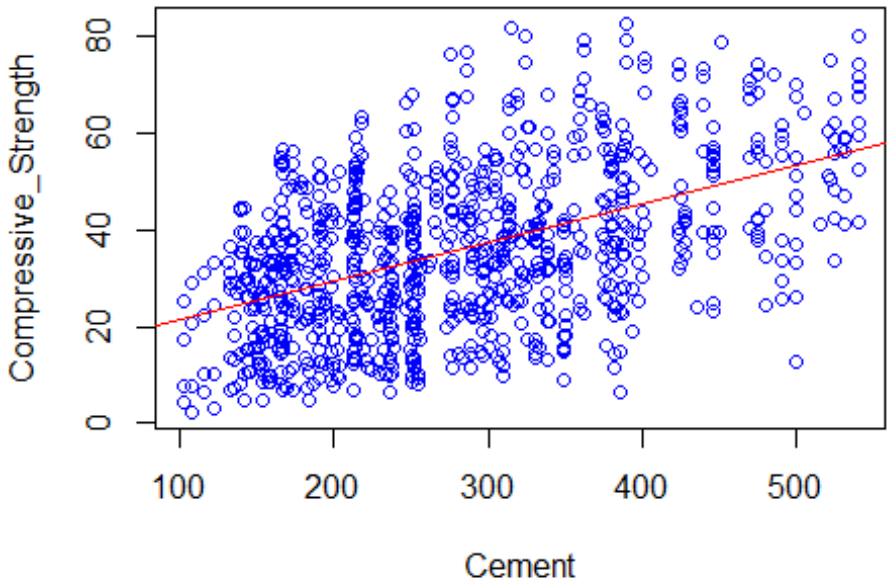


Due to the number of variables, it is difficult to see relationships using the pairs and pairs.panels functions. The GGally ggcorr function does show the relationships more clearly and it is obvious that cement, superplasticizer and age are the three strongest relationships, however all three would be considered weak to moderate correlations. In addition tehre are some interesting correlations between the independent variables that might be worth exploring further (for example the correlation betwwen superplastizer and fly ash, and that between age and water)

It was decided to plot the relationships between each of the independent variables and the dependent variable (compressive strength) to show better visualizations of the correlations.

plot data to view relationships and calculate correlation

Plot relationship between compressive strength and qty cement



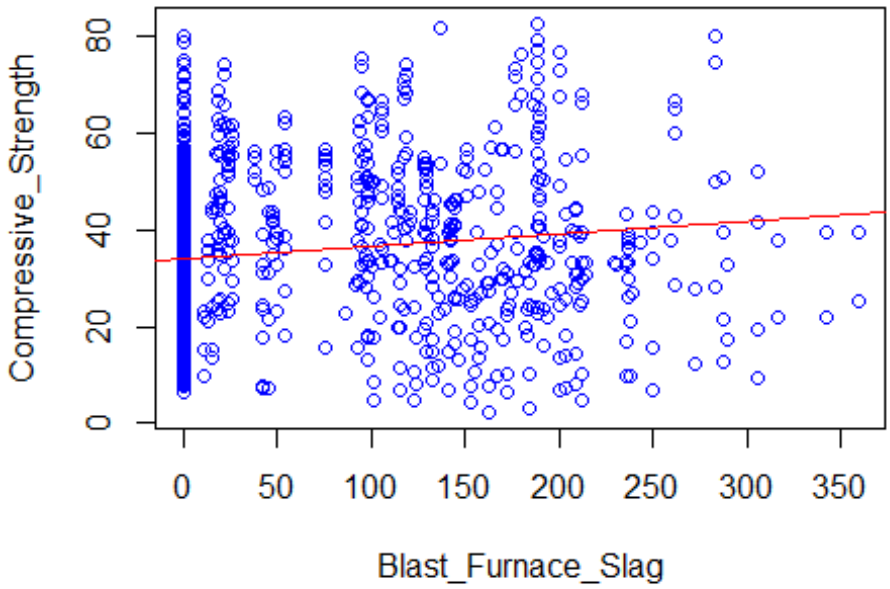
Calculate correlation between compressive strength and qty cement

```
cor(ConDat$Compressive_Strength, ConDat$Cement)
```

```
## [1] 0.4978327
```

There is a moderate positive correlation of 0.5

Plot relationship between compressive strength and qty blast furnace slag



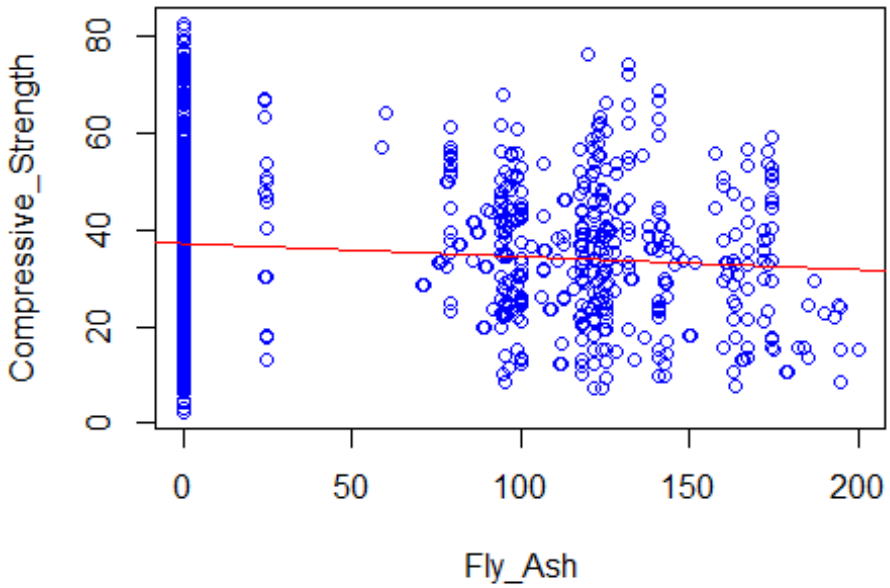
Calculate correlation between compressive strength and qty blast furnace slag

```
cor(ConDat$Compressive_Strength, ConDat$Blast_Furnace_Slag)
```

```
## [1] 0.1348244
```

There is only a very weak positive correlation of 0.13

Plot relationship between compressive strength and qty fly ash



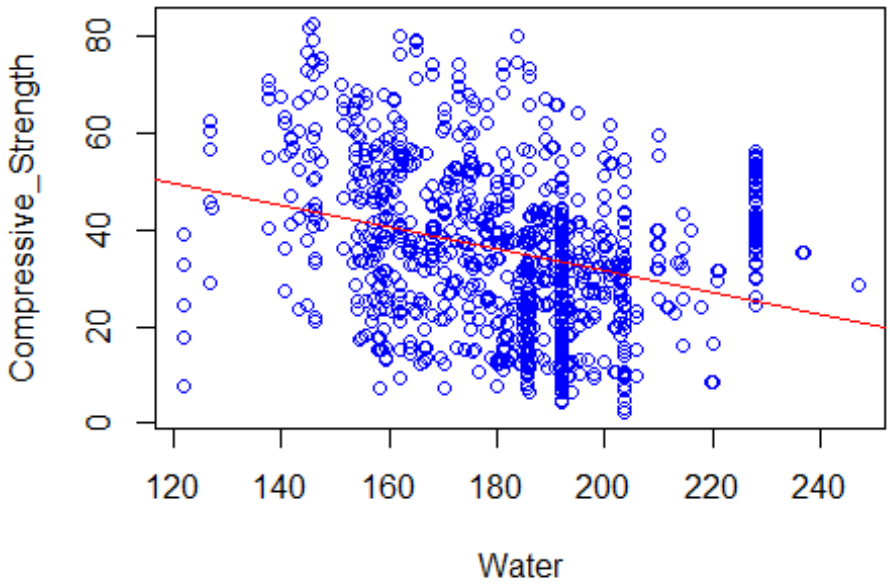
Calculate correlation between compressive strength and qty fly ash

```
cor(ConDat$Compressive_Strength, ConDat$Fly_Ash)
```

```
## [1] -0.1057533
```

There is only a very weak negative correlation of -0.11

Plot relationship between compressive strength and qty water



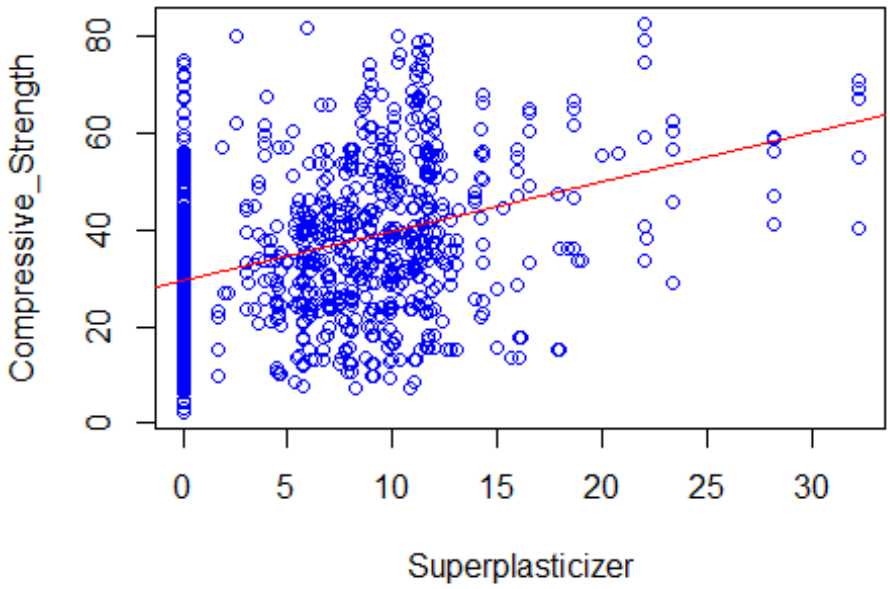
Calculate correlation between compressive strength and qty water

```
cor(ConDat$Compressive_Strength, ConDat$Water)
```

```
## [1] -0.2896135
```

There is a weak negative correlation of -0.29

Plot relationship between compressive strength and qty superplasticizer



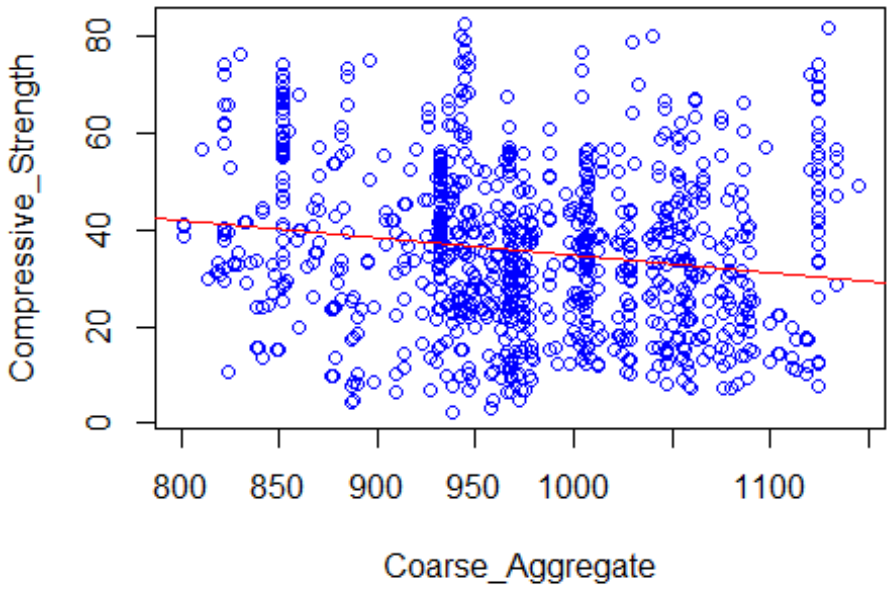
Calculate correlation between compressive strength and qty superplasticizer

```
cor(ConDat$Compressive_Strength, ConDat$Superplasticizer)
```

```
## [1] 0.3661023
```

There is a Weak positive correlation of 0.37

Plot relationship between compressive strength and qty coarse aggregate



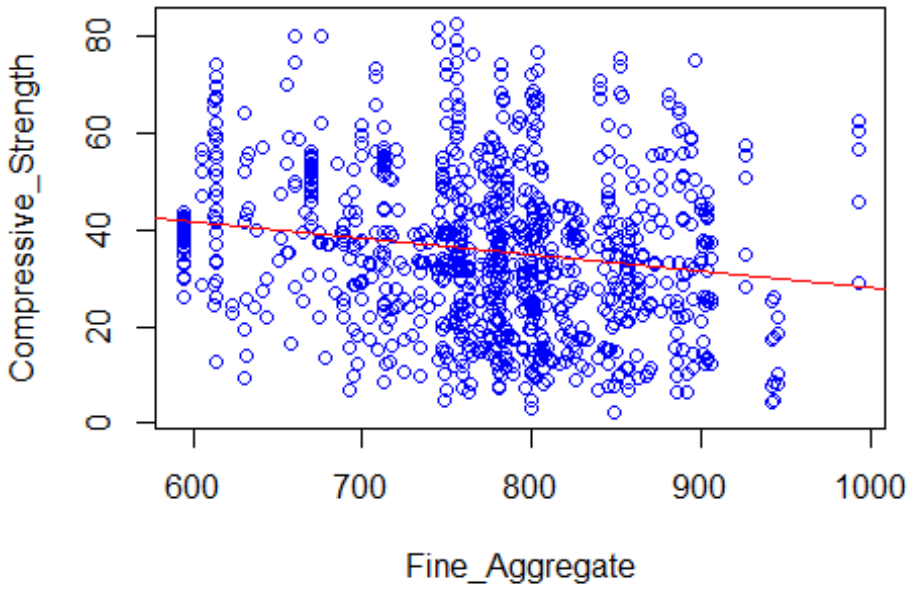
Calculate correlation between compressive strength and qty coarse aggregate

```
cor(ConDat$Compressive_Strength, ConDat$Coarse_Aggregate)
```

```
## [1] -0.1649278
```

There is only a very weak negative correlation of -0.16

Plot relationship between compressive strength and qty fine aggregate



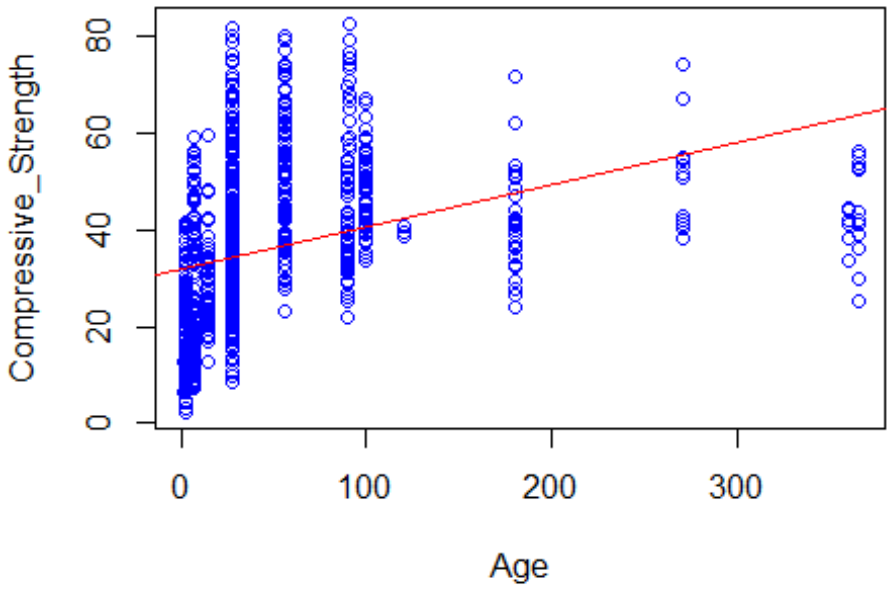
Calculate correlation between compressive strength and qty fine aggegrate

```
cor(ConDat$Compressive_Strength, ConDat$Fine_Aggregate)
```

```
## [1] -0.167249
```

None to very weak negative relationship observed

Plot relationship between compressive strength and age



Calculate correlation between compressive strength and age

```
cor(ConDat$Compressive_Strength, ConDat$Age)
```

```
## [1] 0.328877
```

There is a weak positive correlation of 0.33

Conclusions from Exploratory Data Analysis

Based on this initial evaluation, it appears that that the most significant positive correlations were achieved for Cement (0.4978327), Superplasticizer (0.3661023) and Age (0.32887). The most significant negative correlation is for Water (-0.2896135). None of the correlations are very strong which (in conjunction with the observatiuons from the scatterplots) may indicate that the relationships between Compressive_Strength and the other attributes in the dataset may be non-linear and may require further transformation to develop a more accurate linear regression model.

2.4 Multiple Linear Regression Model

Load data and rename columns

Split ConDat data into training (80%) and validation (20%) sets

Build model csmodel1 with all parameters included

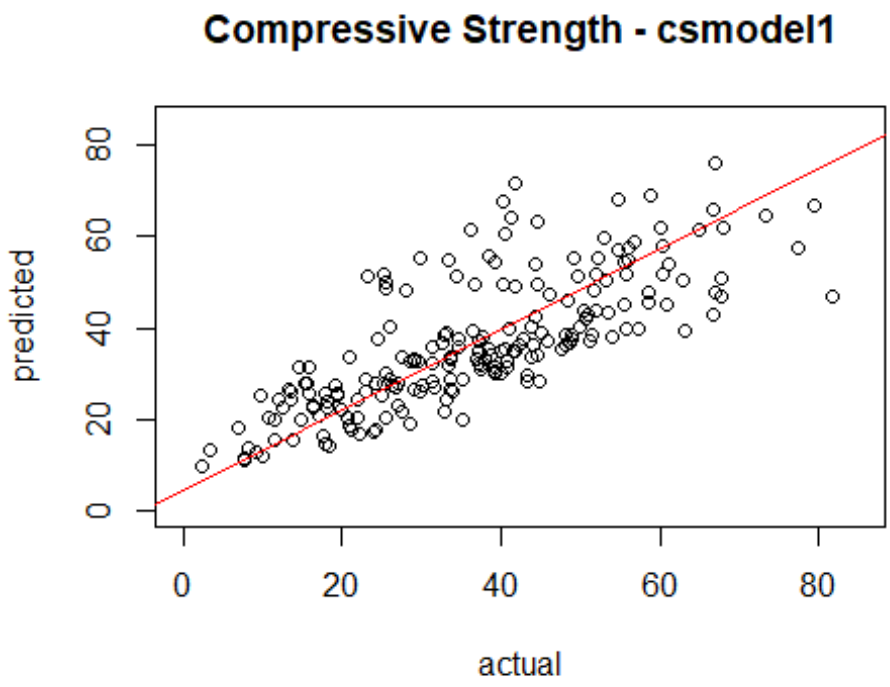
```
csmodel1 <- lm(Compressive_Strength~., data=TrainSet)
```

evaluate csmodel1

```
##
## Call:
## lm(formula = Compressive_Strength ~ ., data = TrainSet)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -29.100  -6.300   0.710   6.509  32.133
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -34.637122  29.306019  -1.182 0.237586
## Cement               0.125080   0.009274  13.487  < 2e-16 ***
## Blast_Furnace_Slag   0.105432   0.011159   9.448  < 2e-16 ***
## Fly_Ash              0.089129   0.013844   6.438 2.06e-10 ***
## Water               -0.128032   0.044668  -2.866 0.004260 **
## Superplasticizer     0.347489   0.104144   3.337 0.000886 ***
## Coarse_Aggregate     0.020055   0.010322   1.943 0.052356 .
## Fine_Aggregate       0.024364   0.011815   2.062 0.039507 *
## Age                  0.117908   0.006286  18.756  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.32 on 815 degrees of freedom
## Multiple R-squared:  0.6242, Adjusted R-squared:  0.6205
## F-statistic: 169.2 on 8 and 815 DF,  p-value: < 2.2e-16
```

The Adj R-squared of 0.6205 so model explains about 62% of the variability

Plot & check correlation between response in ValidSet and response of csmodel1 built using TrainSet



Correlation between predicted and actual results for csmodel1

```
## [1] 0.7636956
```

Correlation is about 76% so model needs more work to improve accuracy

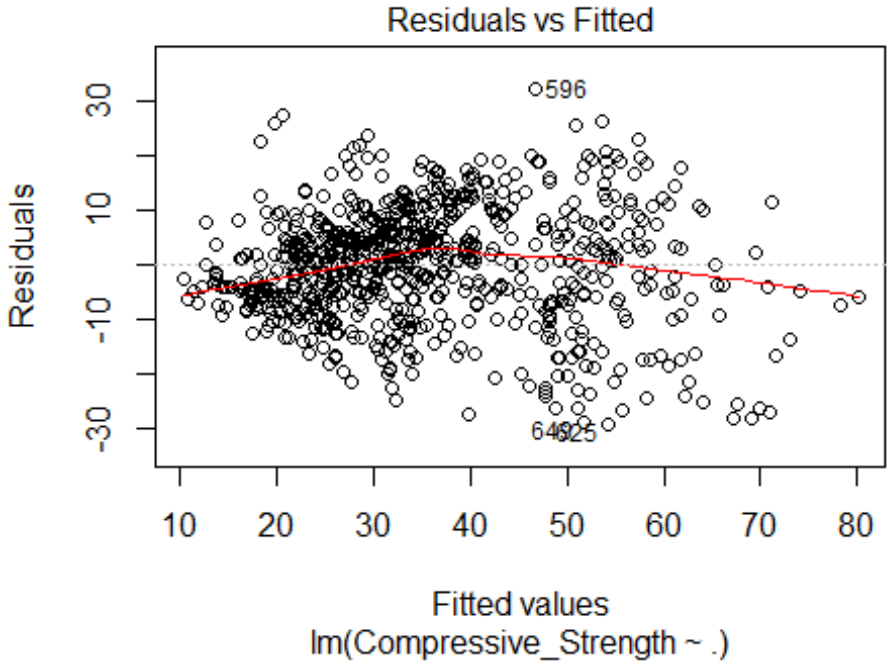
Check for "LINE" assumptions for csmodel1 multiple linear regression model

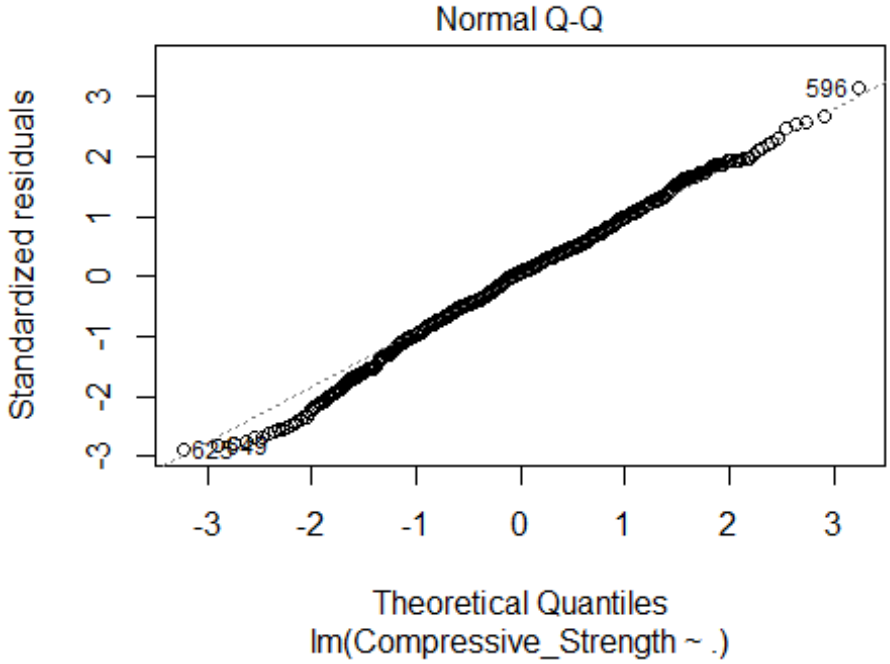
Linear regression models make some assumptions about the data which must be assessed to check the validity of the model. These four key assumptions are often given the acronym LINE:

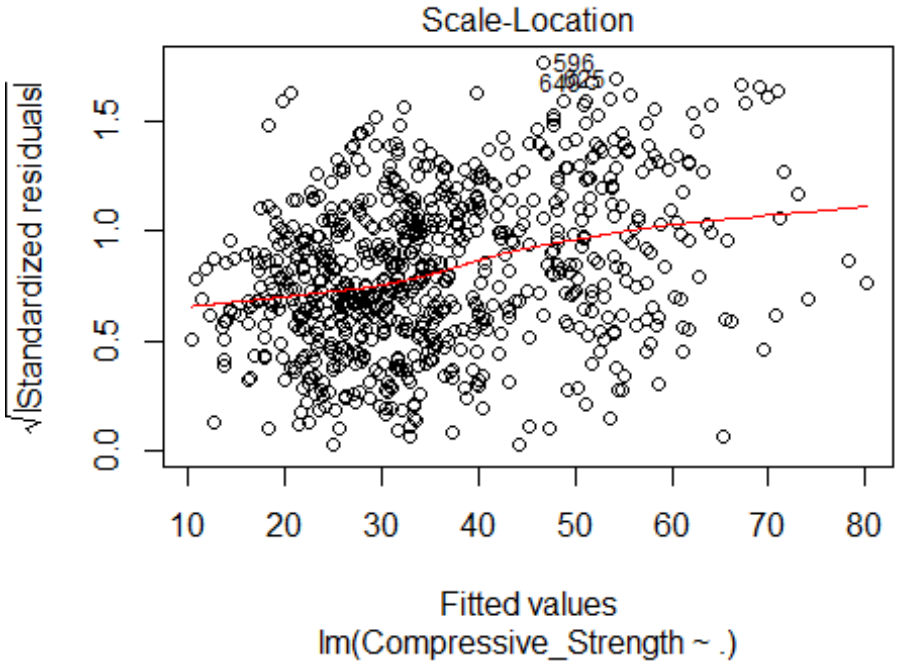
L - Linearity (There is some concern over this based on scatter plots above) I - Independence
N - Normality E - Equal Variances (homoscedasticity)

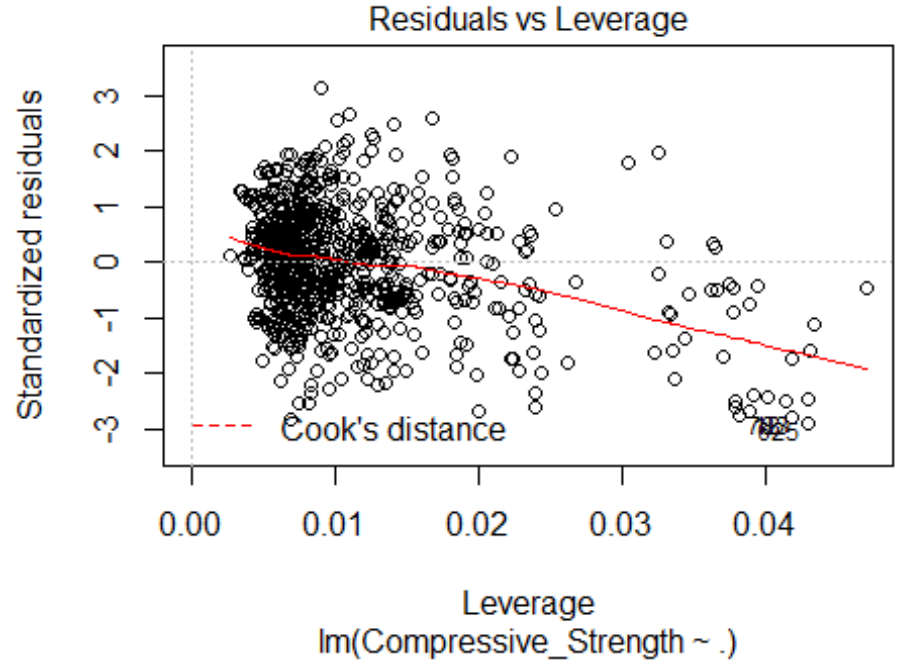
Linearity & Equal Variance

plot residuals to check for linearity and equal variance


Residuals vs Fitted
Residuals
Fitted values
lm(Compressive_Strength ~ .)
596
625
30
10
-10
-30
10
20
30
40
50
60
70
80


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(Compressive_Strength ~ .)
596
3
2
1
0
-1
-2
-3





The residual plot looks reasonably randomly distributed around the zero line however there some concerns that there may be “funnelling” or “bowing” indicating that data is heteroscedastic and that data transformation may be required for modelling

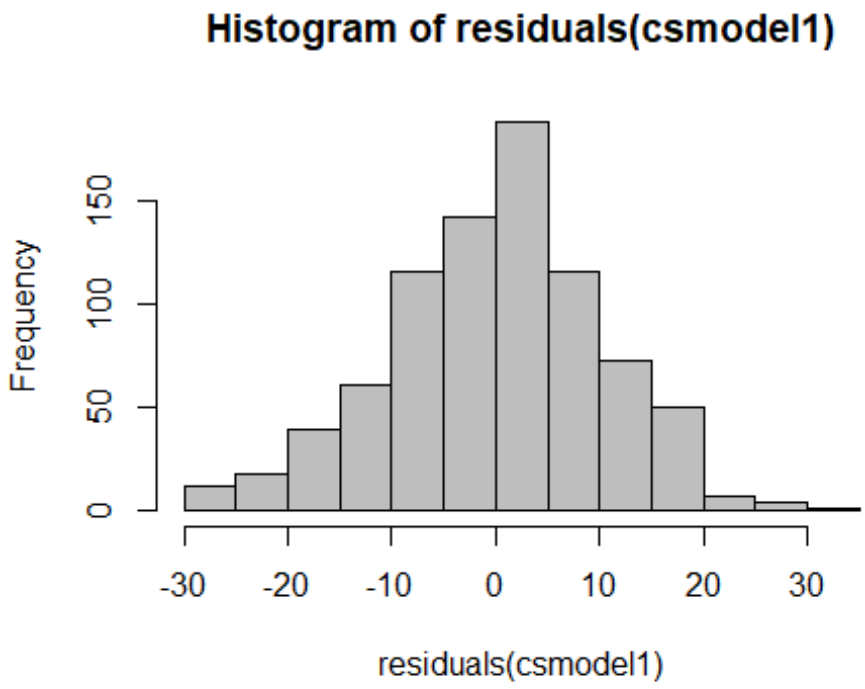
Independence

The Durbin-Watson test was performed to check for Independence

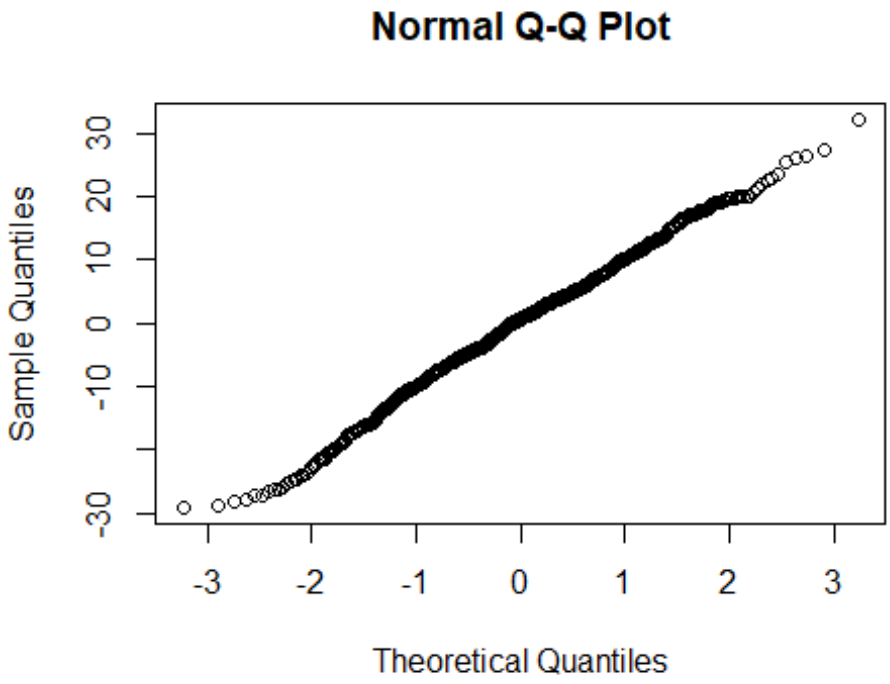
```
##
##  Durbin-Watson test
##
## data:  csmodel1
## DW = 2.0669, p-value = 0.8322
## alternative hypothesis: true autocorrelation is greater than 0
```

If there is no autocorrelation the Durbin-Watson statistic should be between 1.5 and 2.5 and the p-value will be above 0.05. With a DW value of 2.0669 and p value of 0.8322 there is no evidence of autocorrelation

Normality Plot histogram and normal Q-Q plot to confirm residuals are normally distributed



Histogram of residuals for csmodel1



QQ-Plot of residuals for csmodel1

The data looks to be somewhat normally distributed but some deviations at extremes

The Shapiro-wilk and Anderson-Darling tests were completed to confirm residuals are normally distributed

```
##
##  Shapiro-Wilk normality test
##
## data:  csmodel1$residuals
## W = 0.9954, p-value = 0.01455

##
##  Anderson-Darling normality test
##
## data:  csmodel1$residuals
## A = 0.94553, p-value = 0.0167
```

Data looks not to be normally distributed (P value <0.05)

** Data Transformation to Improve mode1**

It was decided to attempt some transformations to see if correlation can be improved by using Transformations on data to improve linearity. The following trsnadoemations were tried:

- square root

- squared
- log
- exp

Compressive strength and qty cement transformations

```
cor(ConDat$Compressive_Strength, ConDat$Cement)
```

```
## [1] 0.4978327
```

```
cor(ConDat$Compressive_Strength, sqrt(ConDat$Cement))
```

```
## [1] 0.4965256
```

```
cor(ConDat$Compressive_Strength, (ConDat$Cement^2))
```

```
## [1] 0.4877756
```

```
cor(ConDat$Compressive_Strength, log(ConDat$Cement))
```

```
## [1] 0.4906377
```

```
cor(ConDat$Compressive_Strength, exp(ConDat$Cement))
```

```
## [1] 0.1601236
```

There was no correlation improvement from these transformations

Compressive strength and qty blast furnace slag

```
cor(ConDat$Compressive_Strength, ConDat$Blast_Furnace_Slag)
```

```
## [1] 0.1348244
```

```
cor(ConDat$Compressive_Strength, sqrt(ConDat$Blast_Furnace_Slag))
```

```
## [1] 0.1807579
```

```
cor(ConDat$Compressive_Strength, (ConDat$Blast_Furnace_Slag^2))
```

```
## [1] 0.07558779
```

```
cor(ConDat$Compressive_Strength, log(ConDat$Blast_Furnace_Slag))
```

```
## [1] NaN
```

```
cor(ConDat$Compressive_Strength, exp(ConDat$Blast_Furnace_Slag))
```

```
## [1] -0.008952402
```

There was no correlation improvement from these transformations

Compressive strength and qty fly ash

```
cor(ConDat$Compressive_Strength, ConDat$Fly_Ash)
```

```
## [1] -0.1057533
```

```
cor(ConDat$Compressive_Strength, sqrt(ConDat$Fly_Ash))
```

```
## [1] -0.08794249
```

```
cor(ConDat$Compressive_Strength, (ConDat$Fly_Ash^2))
```

```
## [1] -0.1280811
```

```
cor(ConDat$Compressive_Strength, log(ConDat$Fly_Ash))
```

```
## [1] NaN
```

```
cor(ConDat$Compressive_Strength, exp(ConDat$Fly_Ash))
```

```
## [1] -0.05545302
```

There was correlation improvement from these transformations

Compressive strength and qty water

```
cor(ConDat$Compressive_Strength, ConDat$Water)
```

```
## [1] -0.2896135
```

```
cor(ConDat$Compressive_Strength, sqrt(ConDat$Water))
```

```
## [1] -0.2983896
```

```
cor(ConDat$Compressive_Strength, (ConDat$Water^2))
```

```
## [1] -0.2689817
```

```
cor(ConDat$Compressive_Strength, log(ConDat$Water))
```

```
## [1] -0.3059733
```

```
cor(ConDat$Compressive_Strength, exp(ConDat$Water))
```

```
## [1] -0.01896802
```

There was no correlation improvement from these transformations

Compressive strength and qty superplasticizer

```
cor(ConDat$Compressive_Strength, ConDat$Superplasticizer)
```

```
## [1] 0.3661023
```

```
cor(ConDat$Compressive_Strength, sqrt(ConDat$Superplasticizer))
```

```
## [1] 0.3483311
```

```
cor(ConDat$Compressive_Strength, (ConDat$Superplasticizer^2))
```

```
## [1] 0.3152159
```

```
cor(ConDat$Compressive_Strength, log(ConDat$Superplasticizer))
```

```
## [1] NaN
```

```
cor(ConDat$Compressive_Strength, exp(ConDat$Superplasticizer))
```

```
## [1] 0.1041011
```

There was no correlation improvement from these transformations

Compressive strength and qty coarse aggregate

```
cor(ConDat$Compressive_Strength, ConDat$Coarse_Aggregate)
```

```
## [1] -0.1649278
```

```
cor(ConDat$Compressive_Strength, sqrt(ConDat$Coarse_Aggregate))
```

```
## [1] -0.1671838
```

```
cor(ConDat$Compressive_Strength, (ConDat$Coarse_Aggregate^2))
```

```
## [1] -0.1599855
```

```
cor(ConDat$Compressive_Strength, log(ConDat$Coarse_Aggregate))
```

```
## [1] -0.1692873
```

```
cor(ConDat$Compressive_Strength, exp(ConDat$Coarse_Aggregate))
```

```
## [1] NaN
```

There was no correlation improvement from these transformations

Compressive strength and qty fine aggregate

```
cor(ConDat$Compressive_Strength, ConDat$Fine_Aggregate)
```

```
## [1] -0.167249
```

```
cor(ConDat$Compressive_Strength, sqrt(ConDat$Fine_Aggregate))
```

```
## [1] -0.1691842
```

```
cor(ConDat$Compressive_Strength, (ConDat$Fine_Aggregate^2))
```

```
## [1] -0.1625184
```

```
cor(ConDat$Compressive_Strength, log(ConDat$Fine_Aggregate))
```

```
## [1] -0.1708168
```

```
cor(ConDat$Compressive_Strength, exp(ConDat$Fine_Aggregate))
```

```
## [1] NaN
```

There was no correlation improvement from these transformations

Compressive strength and age

```
cor(ConDat$Compressive_Strength, ConDat$Age)

## [1] 0.328877

cor(ConDat$Compressive_Strength, sqrt(ConDat$Age))

## [1] 0.4636538

cor(ConDat$Compressive_Strength, (ConDat$Age^2))

## [1] 0.1646082

cor(ConDat$Compressive_Strength, log(ConDat$Age))

## [1] 0.5521848

cor(ConDat$Compressive_Strength, exp(ConDat$Age))

## [1] 0.05451402
```

There was no correlation improvement from these transformations

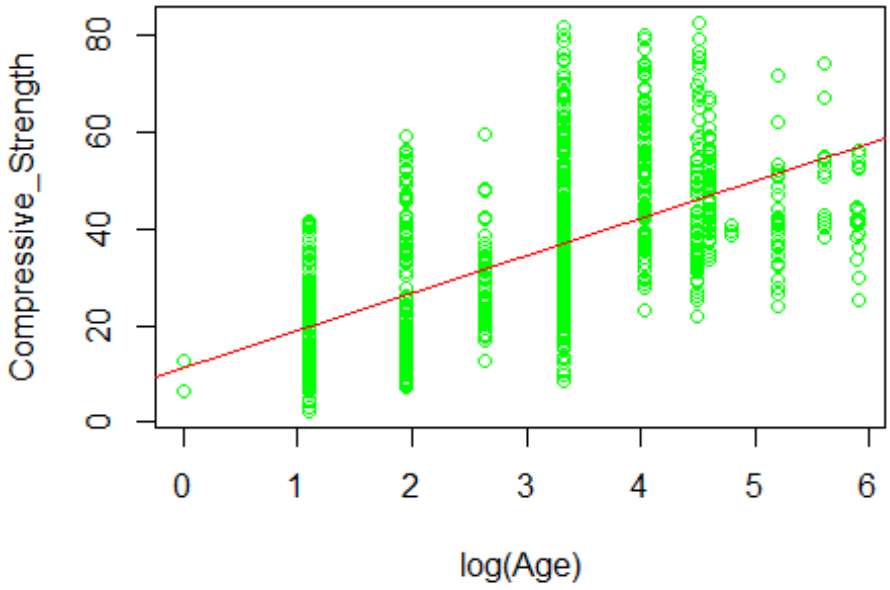
At this point, some background research (https://courses.washington.edu/cm425/strength.pdf) seemed to support the observation that some of the relationships associated with concrete compressive strength may not be linear and would require a transformation. In addition the background research also identified an important relationship between concrete compressive strength and Water/Cement Ratio (a relationship known as Abram's law). Based on this finding transfromations of this ratio were also completed

Compressive strength and water/cement based on Abrams law

```
cor(ConDat$Compressive_Strength, (ConDat$Water/ConDat$Cement))
cor(ConDat$Compressive_Strength, sqrt(ConDat$Water/ConDat$Cement))
cor(ConDat$Compressive_Strength, ((ConDat$Water/ConDat$Cement)^2))
cor(ConDat$Compressive_Strength, log(ConDat$Water/ConDat$Cement))
cor(ConDat$Compressive_Strength, exp(ConDat$Water/ConDat$Cement))
```

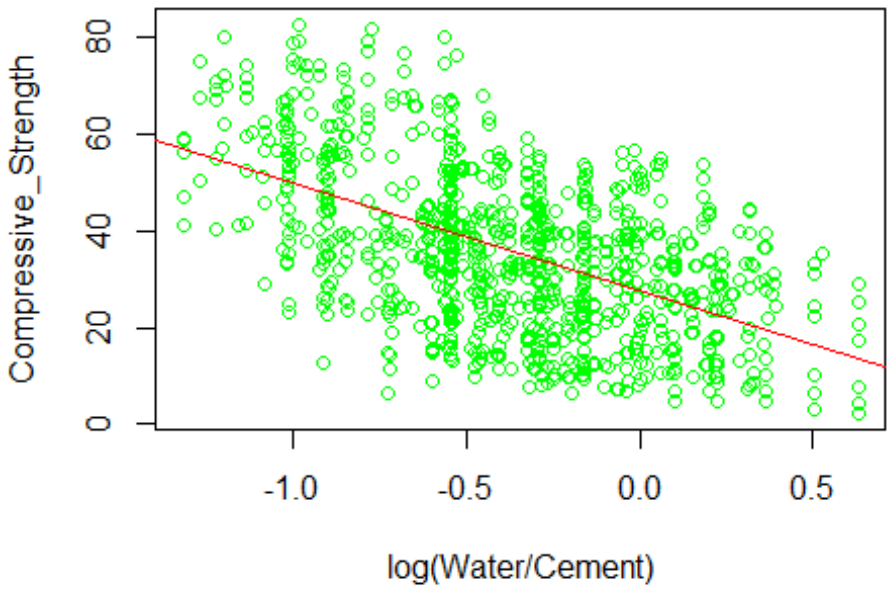
log(Water/Cement) also shows a strong correlation to compressive strength.
log(Age) also shows significant improvement in correlation to compressive strength.

Plot relationship between compressive strength and log(Age)



Correlation between Compressive Strength and log(Age)

Plot relationship between compressive strength and log(water/cement)



The multiple linear regression model was rebuilt as csmodel7 replacing Age with log(Age) and water and Cement variables with log(water/cement) to see if model accuracy is improved

```
##
## Call:
## lm(formula = Compressive_Strength ~ log(Water/Cement) + Blast_Furnace_Slag
+
##     Fly_Ash + Superplasticizer + log(Age), data = TrainSet)
##
## Coefficients:
##        (Intercept)   log(Water/Cement)  Blast_Furnace_Slag
##          -14.17540           -31.77248             0.08912
##            Fly_Ash    Superplasticizer            log(Age)
##            0.06481             0.18124             8.47674

##
## Call:
## lm(formula = Compressive_Strength ~ log(Water/Cement) + Blast_Furnace_Slag
+
##     Fly_Ash + Superplasticizer + log(Age), data = TrainSet)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -23.7722  -4.4773  -0.2502   4.4272  30.7425
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -14.175401   1.016813 -13.941  < 2e-16 ***
## log(Water/Cement)  -31.772484   0.894266 -35.529  < 2e-16 ***
## Blast_Furnace_Slag   0.089115   0.003980  22.389  < 2e-16 ***
## Fly_Ash              0.064813   0.006097  10.630  < 2e-16 ***
## Superplasticizer     0.181241   0.058863   3.079  0.00215 **
## log(Age)             8.476738   0.217709  38.936  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.341 on 818 degrees of freedom
## Multiple R-squared:  0.8093, Adjusted R-squared:  0.8081
## F-statistic: 694.1 on 5 and 818 DF,  p-value: < 2.2e-16
```

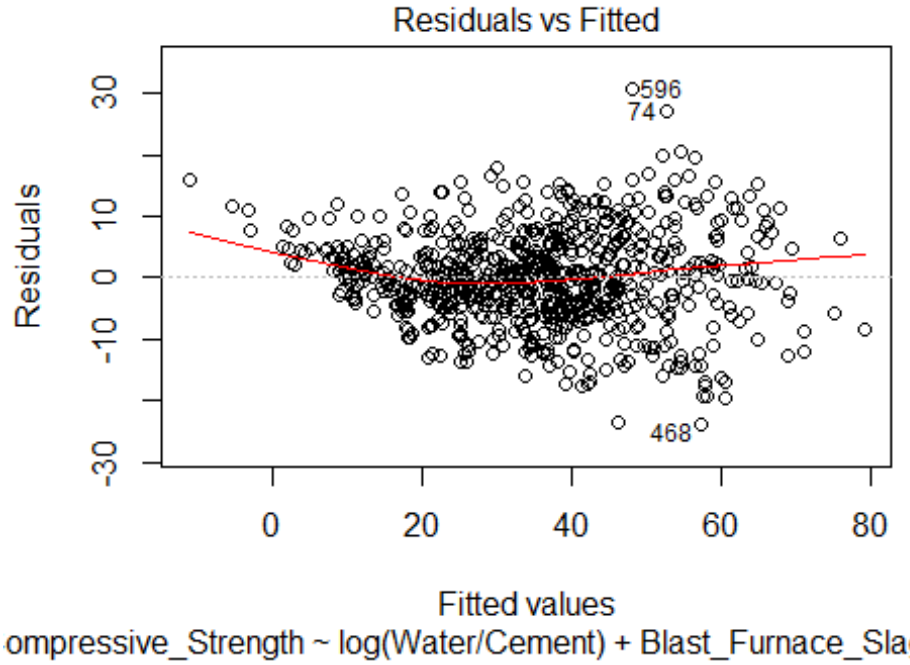
This model is much better with an Adjusted R-squared value of 0.8081

Check for "LINE" assumptions for csmodel7 multiple linear regression model

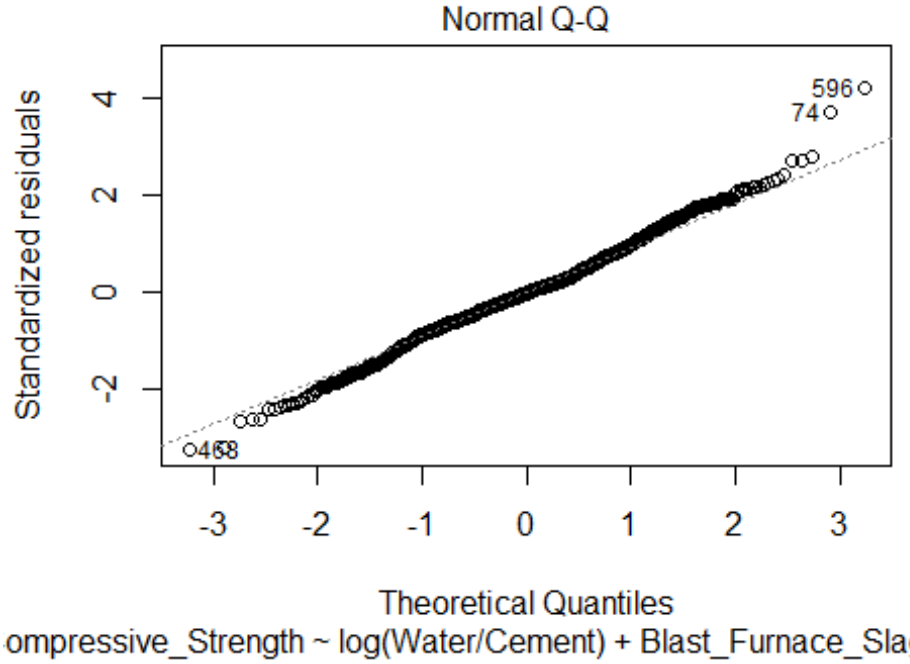
L - Linearity (There is some concern over this based on scatter plots above) I - Independence
N - Normality E - Equal Variances

Linearity and Equal Varience

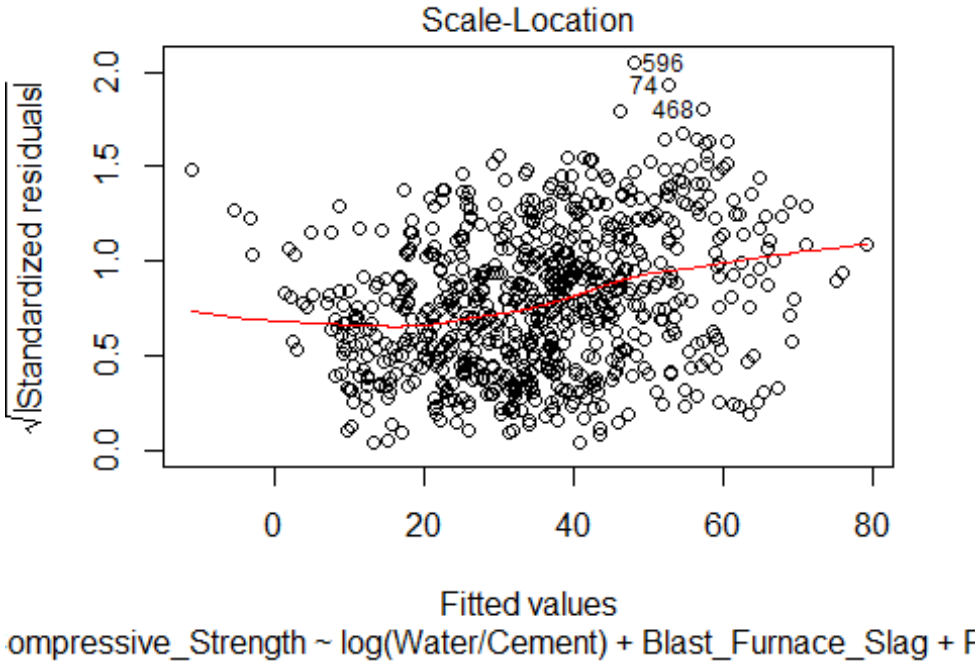
Plot residuals for csmodel7

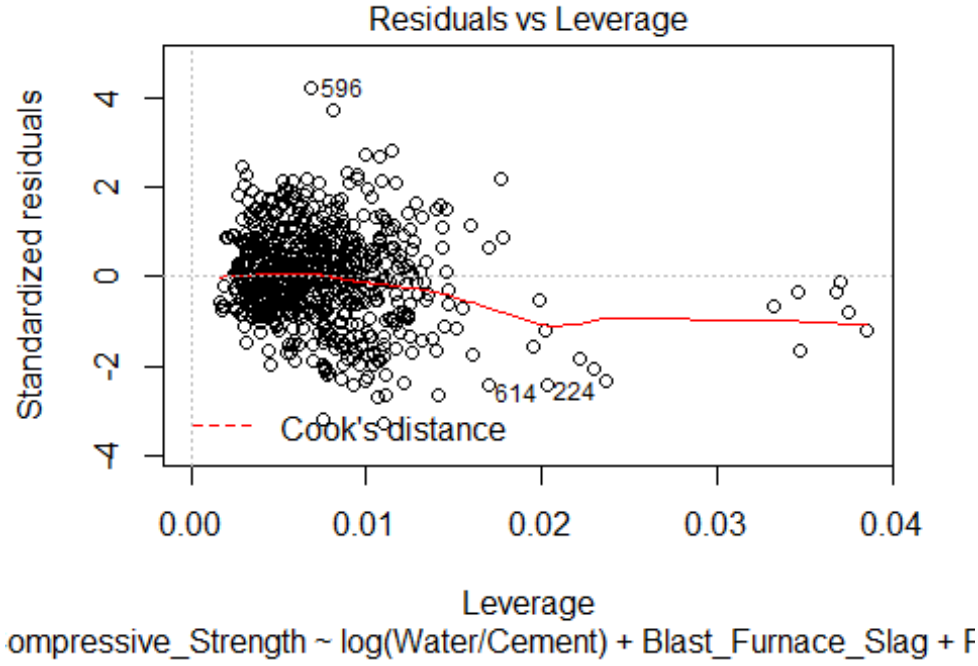


ompressive_Strength ~ log(Water/Cement) + Blast_Furnace_Slag + F



ompressive_Strength ~ log(Water/Cement) + Blast_Furnace_Slag + F





The Residual plot looks reasonably randomly distributed around the zero line. There are still some slight concerns with “funneling” but seems to be better than previous models

Independence

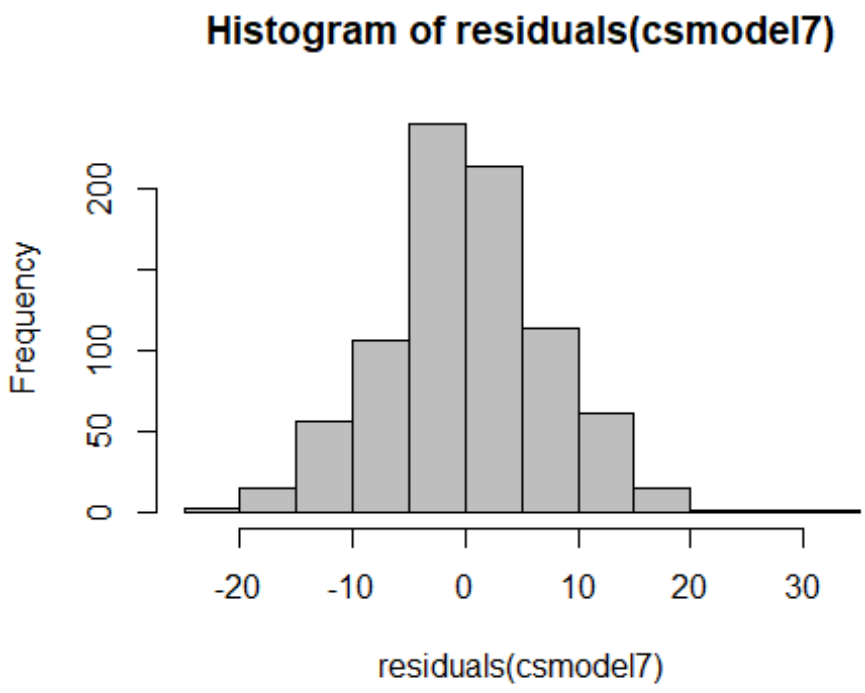
The Durbin-Watson test was completed to check for Independence

```
dwtest(csmodel7)

##
##  Durbin-Watson test
##
## data:  csmodel7
## DW = 2.2018, p-value = 0.9981
## alternative hypothesis: true autocorrelation is greater than 0
```

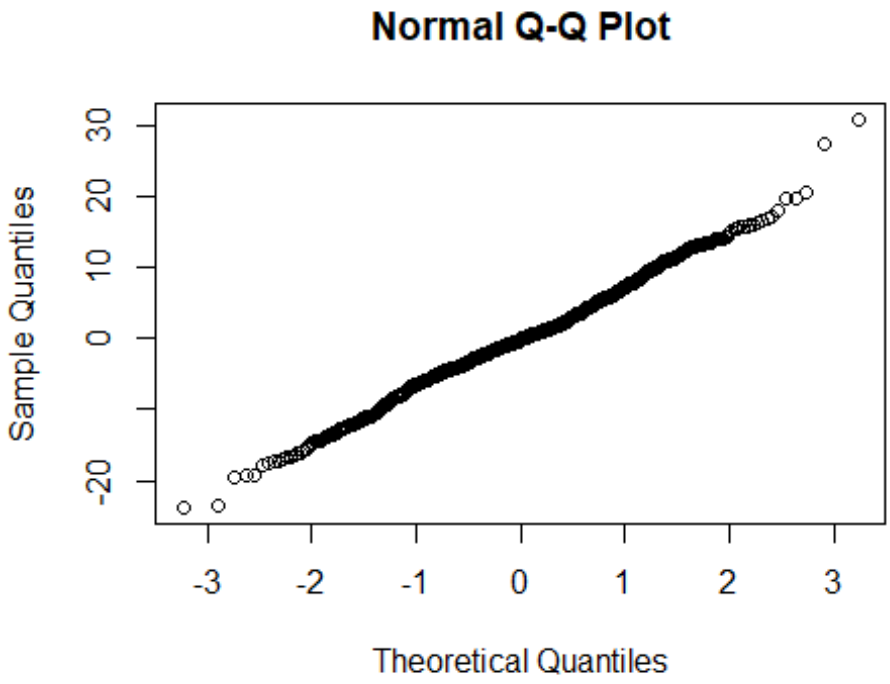
If there is no autocorrelation the Durbin-Watson statistic should be between 1.5 and 2.5 and the p-value will be above 0.05. With a DW value of 2.2018 and p value of 0.9981 there is no evidence of autocorrelation

Normality

Plot histogram and normal Q-Q plot for csmodel7 to confirm residuals are normally distributed



Histogram of residuals for csmodel7



QQ-Plot of residuals for csmodel7

The Shapiro-wilk and Anderson_Darling tests were completed to confirm residuals are normally distributed

```
##
##  Shapiro-Wilk normality test
##
## data:  csmodel7$residuals
## W = 0.99402, p-value = 0.002345

##
##  Anderson-Darling normality test
##
## data:  csmodel7$residuals
## A = 1.7342, p-value = 0.0001917
```

The Data looks not to be normally distributed (P value <0.05), Although this is not an ideal situation, it does not necessarily invalidate the model. Homoscedasticity and linearity tend to be more important than normality for linear regression models and with large datasets, normality of lesser importance for a model to be valid. (Schmidt, A. F. & Finan, C., 2018. Linear regression and the normality assumption. Journal of Clinical Epidemiology, Volume 98, pp. 146-151). As the Training dataset has >800 data points, the requirement for normality for the model to be valid is less important.

Calculate confidence and prediction intervals for csmodel7 (show first 20 rows only)

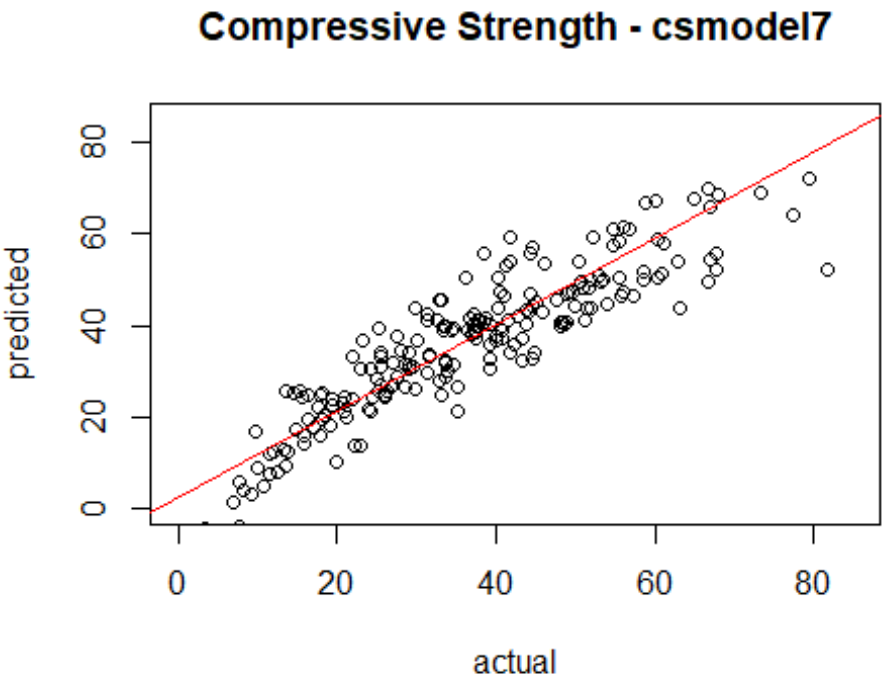
Confidence Interval

```
##         fit      lwr      upr
## 1  32.50273 31.15240 33.85306
## 2  54.04937 52.83243 55.26632
## 3  49.51115 48.13696 50.88535
## 4  38.27626 37.41494 39.13759
## 5  50.89339 49.48927 52.29750
## 6  55.74288 54.11939 57.36637
## 7  36.53275 35.30201 37.76348
## 8  26.52501 25.51840 27.53162
## 9  59.15651 57.64774 60.66527
## 10 48.33786 47.03678 49.63894
## 11 44.25398 42.57531 45.93266
## 12 38.86098 37.45178 40.27018
## 13 50.33244 47.43734 53.22754
## 14 34.59730 33.25480 35.93981
## 15 39.95742 38.35696 41.55789
## 16 39.43522 37.88765 40.98279
## 17 40.20715 38.78470 41.62961
## 18 47.19872 45.20141 49.19602
## 19 57.51476 54.70649 60.32303
## 20 47.13974 45.70015 48.57934
```

Prediction Interval

```
##         fit      lwr      upr
## 1  32.50273 18.03056 46.97490
## 2  54.04937 39.58903 68.50971
## 3  49.51115 35.03673 63.98557
## 4  38.27626 23.84150 52.71102
## 5  50.89339 36.41610 65.37068
## 6  55.74288 41.24267 70.24309
## 7  36.53275 22.07124 50.99425
## 8  26.52501 12.08085 40.96917
## 9  59.15651 44.66869 73.64432
## 10 48.33786 33.87020 62.80552
## 11 44.25398 29.74749 58.76048
## 12 38.86098 24.38319 53.33876
## 13 50.33244 35.63544 65.02945
## 14 34.59730 20.12586 49.06875
## 15 39.95742 25.45977 54.45507
## 16 39.43522 24.94331 53.92713
## 17 40.20715 25.72807 54.68624
## 18 47.19872 32.65191 61.74553
## 19 57.51476 42.83461 72.19492
## 20 47.13974 32.65897 61.62052
```

Plot & check correlation between response in ValidSet and response of csmodel7 built using TrainSet



Residual plot for csmodel7

```
## [1] 0.897965
```

The Correlation has improved to about 90% for csmodel7

Calculate RMSE and MAPE for csmodel7

```
##        mae        mse       rmse       mape
##  5.7398761 53.5153644  7.3154196  0.2227156
```

The RMSE of the model is 7.3 and the MAPE is only 0.22 indicating a reasonably accurate model.

Conclusions from Multiple lInear Regression Model

The final Multiple Linear Regression model (after transformation of the data) has an accuracy of 90% which represents a good model. The Adjusted R-squared value is 0.8081 which means that the model explains about 80% of the variability. The RMSE of the model is 7.3 and the MAPE is only 0.22 indicating a reasonably accurate model. The model meets all of the assumptions for a regression model apart from normality. Homoscedasticity and

linearity tend to be more important than normality for linear regression models and with large datasets, normality of lesser importance for a model to be valid. (Schmidt, A. F. & Finan, C., 2018. Linear regression and the normality assumption. Journal of Clinical Epidemiology, Volume 98, pp. 146-151). As the Training dataset has >800 data points, the requirement for normality for the model to be valid is less important.

2.5 - Artificial Neural Network Modelling

Normalize Dataset

It is necessary to normalize the ConDat dataset as Neural Networks work best when input data are scaled to a narrow range arond zero, This was performed by creating a function to normalize data. The dataset (apart from the Concrete Strength Category) was then normalized using the function

```
##      Cement       Blast_Furnace_Slag    Fly_Ash           Water
##  Min.   :0.0000   Min.   :0.00000    Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.2063   1st Qu.:0.00000    1st Qu.:0.0000   1st Qu.:0.3445
##  Median :0.3902   Median :0.06121    Median :0.0000   Median :0.5050
##  Mean   :0.4091   Mean   :0.20561    Mean   :0.2708   Mean   :0.4776
##  3rd Qu.:0.5662   3rd Qu.:0.39775    3rd Qu.:0.5911   3rd Qu.:0.5609
##  Max.   :1.0000   Max.   :1.00000    Max.   :1.0000   Max.   :1.0000
##  Superplasticizer Coarse_Aggregate Fine_Aggregate        Age
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:0.3808   1st Qu.:0.3436   1st Qu.:0.01648
##  Median :0.1972   Median :0.4855   Median :0.4654   Median :0.07418
##  Mean   :0.1926   Mean   :0.4998   Mean   :0.4505   Mean   :0.12270
##  3rd Qu.:0.3155   3rd Qu.:0.6640   3rd Qu.:0.5770   3rd Qu.:0.15110
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##  Compressive_Strength
##  Min.   :0.0000
##  1st Qu.:0.2663
##  Median :0.4000
##  Mean   :0.4172
##  3rd Qu.:0.5457
##  Max.   :1.0000

## 'data.frame':    1030 obs. of  9 variables:
##  $ Cement              : num  1 1 0.526 0.526 0.221 ...
##  $ Blast_Furnace_Slag  : num  0 0 0.396 0.396 0.368 ...
##  $ Fly_Ash             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Water               : num  0.321 0.321 0.848 0.848 0.561 ...
##  $ Superplasticizer    : num  0.0776 0.0776 0 0 0 ...
##  $ Coarse_Aggregate    : num  0.695 0.738 0.381 0.381 0.516 ...
##  $ Fine_Aggregate      : num  0.206 0.206 0 0 0.581 ...
##  $ Age                 : num  0.0742 0.0742 0.739 1 0.9863 ...
##  $ Compressive_Strength: num  0.967 0.742 0.473 0.482 0.523 ...
```

```
##      Cement Blast_Furnace_Slag Fly_Ash     Water Superplasticizer
## 1 1.0000000          0.0000000       0 0.3213573       0.07763975
## 2 1.0000000          0.0000000       0 0.3213573       0.07763975
## 3 0.5262557          0.3964942       0 0.8483034       0.00000000
## 4 0.5262557          0.3964942       0 0.8483034       0.00000000
## 5 0.2205479          0.3683918       0 0.5608782       0.00000000
## 6 0.3744292          0.3171953       0 0.8483034       0.00000000
##   Coarse_Aggregate Fine_Aggregate        Age Compressive_Strength
## 1        0.6947674      0.2057200 0.07417582            0.9674449
## 2        0.7383721      0.2057200 0.07417582            0.7419643
## 3        0.3808140      0.0000000 0.73901099            0.4726417
## 4        0.3808140      0.0000000 1.00000000            0.4823996
## 5        0.5156977      0.5807827 0.98626374            0.5228058
## 6        0.3808140      0.1906673 0.24450549            0.5568641
```

Create Training and Validation Datasets

The data was Split Data into Training Set (70%) and Validation Set (30%)

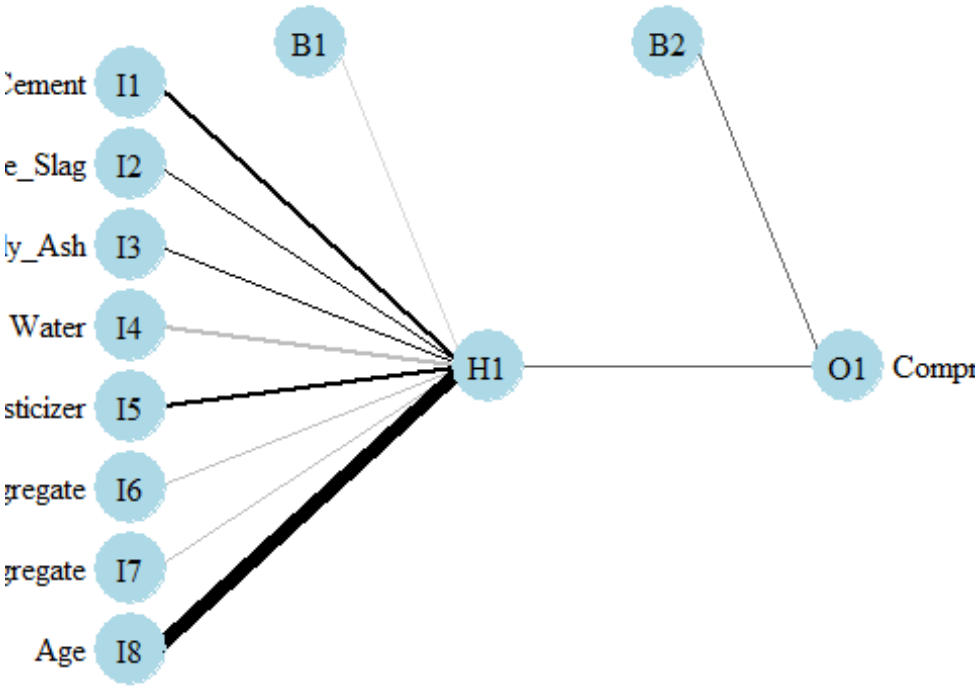
```
##      Cement       Blast_Furnace_Slag    Fly_Ash           Water
##  Min.   :0.0000   Min.   :0.00000    Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.2317   1st Qu.:0.00000    1st Qu.:0.0000   1st Qu.:0.3102
##  Median :0.3432   Median :0.06678    Median :0.0000   Median :0.4651
##  Mean   :0.4151   Mean   :0.20938    Mean   :0.2651   Mean   :0.4540
##  3rd Qu.:0.5950   3rd Qu.:0.38119    3rd Qu.:0.5911   3rd Qu.:0.5609
##  Max.   :1.0000   Max.   :1.00000    Max.   :0.8733   Max.   :0.8483
##  Superplasticizer Coarse_Aggregate Fine_Aggregate        Age
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.005495
##  1st Qu.:0.0000   1st Qu.:0.3924   1st Qu.:0.3828   1st Qu.:0.016484
##  Median :0.1904   Median :0.4855   Median :0.4691   Median :0.074176
##  Mean   :0.1961   Mean   :0.5172   Mean   :0.4694   Mean   :0.131133
##  3rd Qu.:0.3363   3rd Qu.:0.6744   3rd Qu.:0.6297   3rd Qu.:0.151099
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000000
##  Compressive_Strength
##  Min.   :0.0000
##  1st Qu.:0.2708
##  Median :0.4250
##  Mean   :0.4352
##  3rd Qu.:0.5935
##  Max.   :1.0000

##      Cement        Blast_Furnace_Slag    Fly_Ash           Water
##  Min.   :0.06849   Min.   :0.0000     Min.   :0.0000   Min.   :0.04192
##  1st Qu.:0.12100   1st Qu.:0.0000     1st Qu.:0.0000   1st Qu.:0.45629
##  Median :0.44292   Median :0.0000     Median :0.0000   Median :0.54491
##  Mean   :0.39498   Mean   :0.1968     Mean   :0.2841   Mean   :0.53269
##  3rd Qu.:0.52283   3rd Qu.:0.4035     3rd Qu.:0.5647   3rd Qu.:0.58802
##  Max.   :1.00000   Max.   :0.7234     Max.   :1.0000   Max.   :1.00000
##  Superplasticizer Coarse_Aggregate Fine_Aggregate         Age
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.04516   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:0.2267   1st Qu.:0.27847   1st Qu.:0.07418
```

```
##  Median :0.2174   Median :0.4605   Median :0.43904   Median :0.07418
##  Mean   :0.1845   Mean   :0.4592   Mean   :0.40643   Mean   :0.10302
##  3rd Qu.:0.3075   3rd Qu.:0.6453   3rd Qu.:0.53939   3rd Qu.:0.07418
##  Max.   :0.6863   Max.   :0.9419   Max.   :0.75765   Max.   :0.98626
##  Compressive_Strength
##  Min.   :0.04903
##  1st Qu.:0.26395
##  Median :0.37462
##  Mean   :0.37515
##  3rd Qu.:0.47749
##  Max.   :0.90517
```

Build first ANN model annconmodel_1

Plot first ANN model

Plot first ANN model with plotnet



Plot of ann_conmodel1 using plotnet

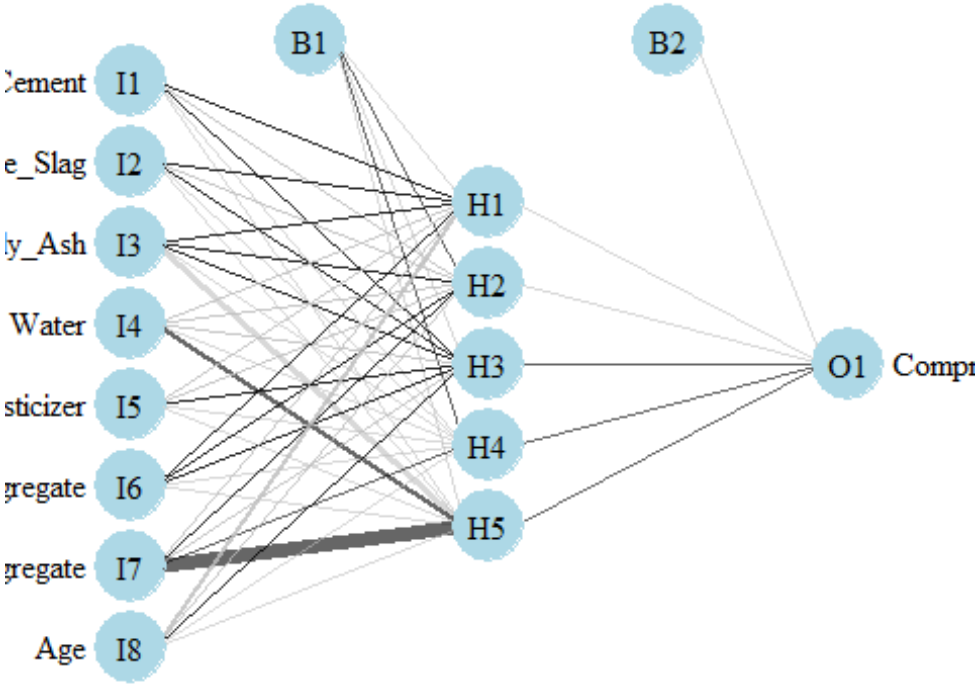
Check prediction accuracy of first ANN model

```
##           [,1]
## [1,] 0.7528747
```

Build second ANN model using 5 hidden nodes

Plot second ANN model ann_conmodel2

Plot second ANN model with plotnet



Plot of ann_conmodel1 with plotnet

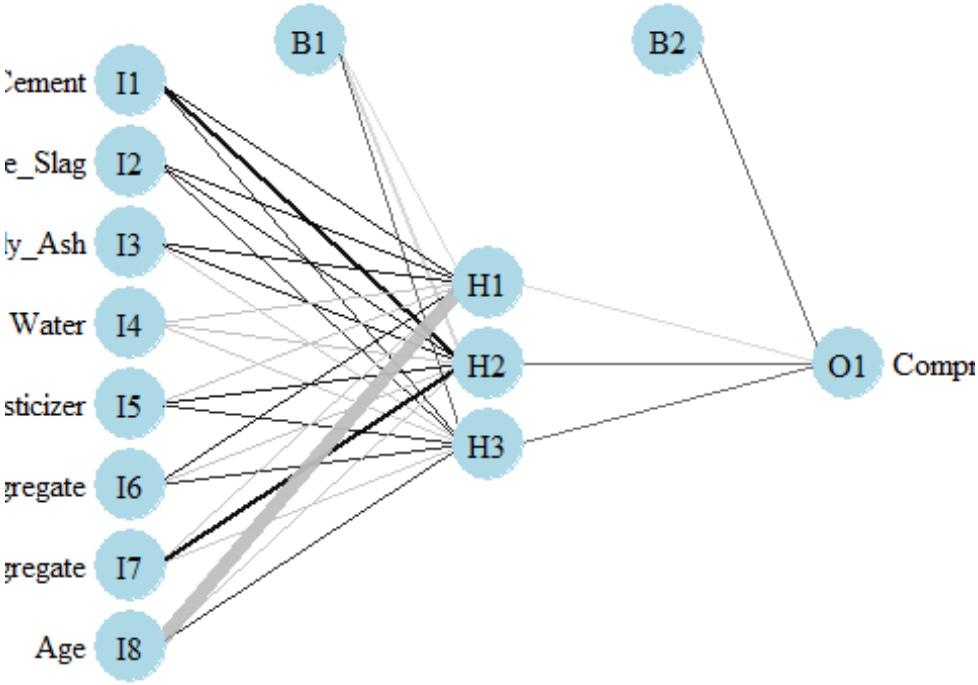
check prediction accuracy of second ANN model

```
##           [,1]
## [1,] 0.7555812
```

Build third ANN model using 5 hidden neurons and logistic activation function

Plot third ANN model

Plot third ANN model with plotnet



Plot of ann_conmodel3 with plotnet

Check prediction accuracy of third ANN model

```
##           [,1]
## [1,] 0.7910229
```

Conclusions from Artificial Neural Network Model

The best ANN model (after transformation of the data) was achieved using 5 hidden neurons and logistic activation function. The model has an accuracy of 79% which represents a reasobally accurate model. THis model is not as good as the multiple linear regression model. ANN is best used for very large datasets rather than small data sets like this.

2.6 Data Visualization for Classification Algorithms

It was decided to use notched box plots to see which features showed the best separation for Concrete Strength and therefore which features might be most useful in classification machine learning alogrithms

Create 4 categories for classification purposes

The concrete comprerssive strengh data was classifiied into four categories:

- Fail - <5Mpa
- Low Strength 5-19MPa

- Standard Strength 20-49MPa
- High Strength >= 50MPa

Concrete classification is based on compressive strength in MPa was obtained from the website linked below:

https://www.baseconcrete.co.uk/different-types-of-concrete-grades-and-their-uses/

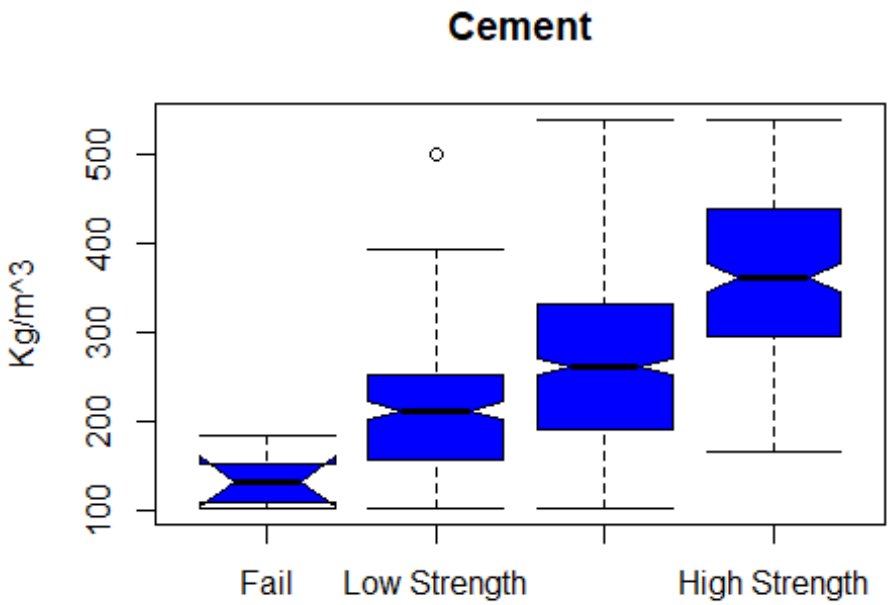
```
## Classes 'tbl_df', 'tbl' and 'data.frame':    1030 obs. of  10 variables:
##  $ Cement              : num  540 540 332 332 199 ...
##  $ Blast_Furnace_Slag  : num  0 0 142 142 132 ...
##  $ Fly_Ash             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Water               : num  162 162 228 228 192 228 228 228 228 228 ...
##  $ Superplasticizer    : num  2.5 2.5 0 0 0 0 0 0 0 0 ...
##  $ Coarse_Aggregate    : num  1040 1055 932 932 978 ...
##  $ Fine_Aggregate      : num  676 676 594 594 826 ...
##  $ Age                 : num  28 28 270 365 360 90 365 28 28 28 ...
##  $ Compressive_Strength: num  80 61.9 40.3 41.1 44.3 ...
##  $ Concrete_Category   : num  4 4 3 3 3 3 3 3 3 3 ...

##      Cement      Blast_Furnace_Slag    Fly_Ash           Water
##  Min.   :102.0   Min.   :  0.0      Min.   :  0.00   Min.   :121.8
##  1st Qu.:192.4   1st Qu.:  0.0      1st Qu.:  0.00   1st Qu.:164.9
##  Median :272.9   Median : 22.0      Median :  0.00   Median :185.0
##  Mean   :281.2   Mean   : 73.9      Mean   : 54.19   Mean   :181.6
##  3rd Qu.:350.0   3rd Qu.:142.9      3rd Qu.:118.27   3rd Qu.:192.0
##  Max.   :540.0   Max.   :359.4      Max.   :200.10   Max.   :247.0
##  Superplasticizer Coarse_Aggregate Fine_Aggregate       Age
##  Min.   : 0.000   Min.   : 801.0   Min.   :594.0   Min.   :  1.00
##  1st Qu.: 0.000   1st Qu.: 932.0   1st Qu.:731.0   1st Qu.:  7.00
##  Median : 6.350   Median : 968.0   Median :779.5   Median : 28.00
##  Mean   : 6.203   Mean   : 972.9   Mean   :773.6   Mean   : 45.66
##  3rd Qu.:10.160   3rd Qu.:1029.4   3rd Qu.:824.0   3rd Qu.: 56.00
##  Max.   :32.200   Max.   :1145.0   Max.   :992.6   Max.   :365.00
##  Compressive_Strength Concrete_Category
##  Min.   : 2.332       Min.   :1.000
##  1st Qu.:23.707       1st Qu.:3.000
##  Median :34.443       Median :3.000
##  Mean   :35.818       Mean   :3.007
##  3rd Qu.:46.136       3rd Qu.:3.000
##  Max.   :82.599       Max.   :4.000

## # A tibble: 6 x 10
##   Cement Blast_Furnace_S~ Fly_Ash Water Superplasticizer Coarse_Aggregate
##    <dbl>            <dbl>   <dbl> <dbl>            <dbl>            <dbl>
## 1   540                0        0   162              2.5             1040
## 2   540                0        0   162              2.5             1055
## 3   332.             142.       0   228              0                932
## 4   332.             142.       0   228              0                932
## 5   199.             132.       0   192              0                978.
## 6   266              114        0   228              0                932
```

```
## # ... with 4 more variables: Fine_Aggregate <dbl>, Age <dbl>,
## #   Compressive_Strength <dbl>, Concrete_Category <dbl>

## Classes 'tbl_df', 'tbl' and 'data.frame':    1030 obs. of  9 variables:
##  $ Cement            : num  540 540 332 332 199 ...
##  $ Blast_Furnace_Slag: num  0 0 142 142 132 ...
##  $ Fly_Ash           : num  0 0 0 0 0 0 0 0 0 ...
##  $ Water             : num  162 162 228 228 192 228 228 228 228 228 ...
##  $ Superplasticizer  : num  2.5 2.5 0 0 0 0 0 0 0 ...
##  $ Coarse_Aggregate  : num  1040 1055 932 932 978 ...
##  $ Fine_Aggregate    : num  676 676 594 594 826 ...
##  $ Age               : num  28 28 270 365 360 90 365 28 28 28 ...
##  $ Concrete_Category : Factor w/ 4 levels "Fail","Low Strength",..: 4 4 3
3 3 3 3 3 3 3 ...

##      Cement      Blast_Furnace_Slag    Fly_Ash           Water
##  Min.   :102.0   Min.   :  0.0      Min.   :  0.00   Min.   :121.8
##  1st Qu.:192.4   1st Qu.:  0.0      1st Qu.:  0.00   1st Qu.:164.9
##  Median :272.9   Median : 22.0      Median :  0.00   Median :185.0
##  Mean   :281.2   Mean   : 73.9      Mean   : 54.19   Mean   :181.6
##  3rd Qu.:350.0   3rd Qu.:142.9      3rd Qu.:118.27   3rd Qu.:192.0
##  Max.   :540.0   Max.   :359.4      Max.   :200.10   Max.   :247.0
##  Superplasticizer Coarse_Aggregate Fine_Aggregate       Age
##  Min.   : 0.000   Min.   : 801.0   Min.   :594.0   Min.   :  1.00
##  1st Qu.: 0.000   1st Qu.: 932.0   1st Qu.:731.0   1st Qu.:  7.00
##  Median : 6.350   Median : 968.0   Median :779.5   Median : 28.00
##  Mean   : 6.203   Mean   : 972.9   Mean   :773.6   Mean   : 45.66
##  3rd Qu.:10.160   3rd Qu.:1029.4   3rd Qu.:824.0   3rd Qu.: 56.00
##  Max.   :32.200   Max.   :1145.0   Max.   :992.6   Max.   :365.00
##          Concrete_Category
##  Fail             :  6
##  Low Strength     :191
##  Standard Strength:623
##  High Strength    :210
##
##

## # A tibble: 6 x 9
##   Cement Blast_Furnace_S~ Fly_Ash Water Superplasticizer Coarse_Aggregate
##    <dbl>            <dbl>   <dbl> <dbl>            <dbl>            <dbl>
## 1   540               0        0   162              2.5             1040
## 2   540               0        0   162              2.5             1055
## 3   332.            142.       0   228              0                932
## 4   332.            142.       0   228              0                932
## 5   199.            132.       0   192              0                978.
## 6   266             114        0   228              0                932
## # ... with 3 more variables: Fine_Aggregate <dbl>, Age <dbl>,
## #   Concrete_Category <fct>
```

Visualize features by strength cartegory using notched box plots

It was decided to use box plots to visualize which features showed best separataion based on compressive strength category. If there is no overlap of the notches then there is a signifcant difference in the median values.

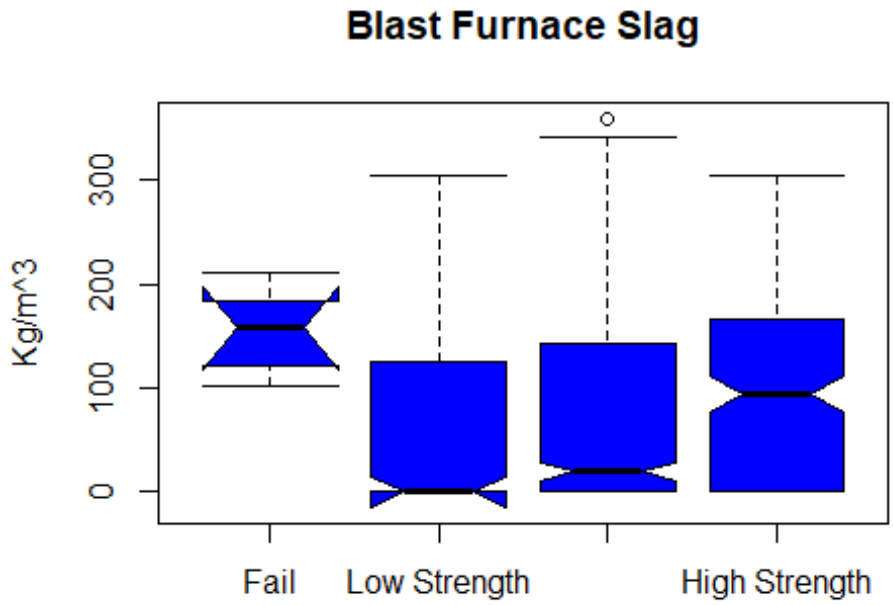
Cement



Cement

The cement variable seems to show good separation based on compressive strength category.

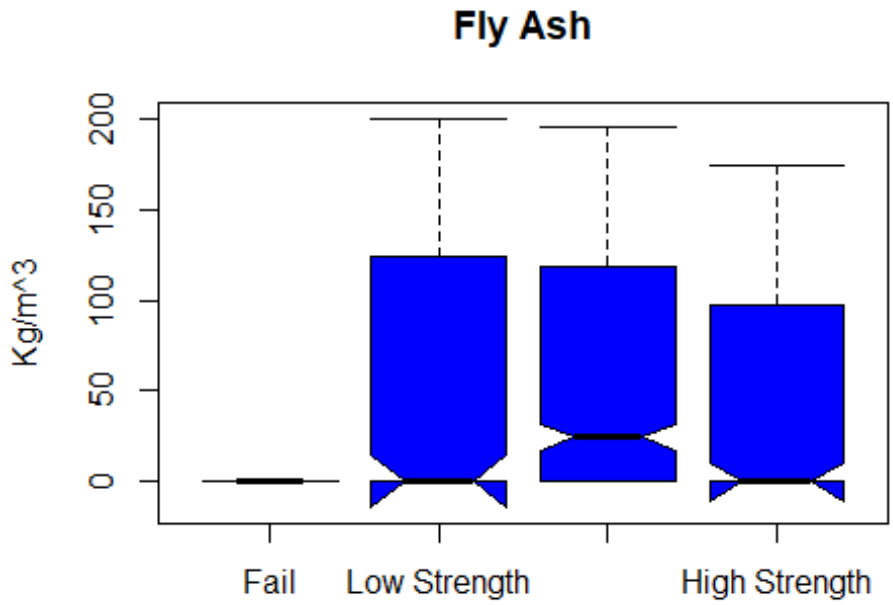
Blast Furnace Slag



Blast Furnace Slag

The blast furnace slag variable does not show good separation based on compressive strength category.

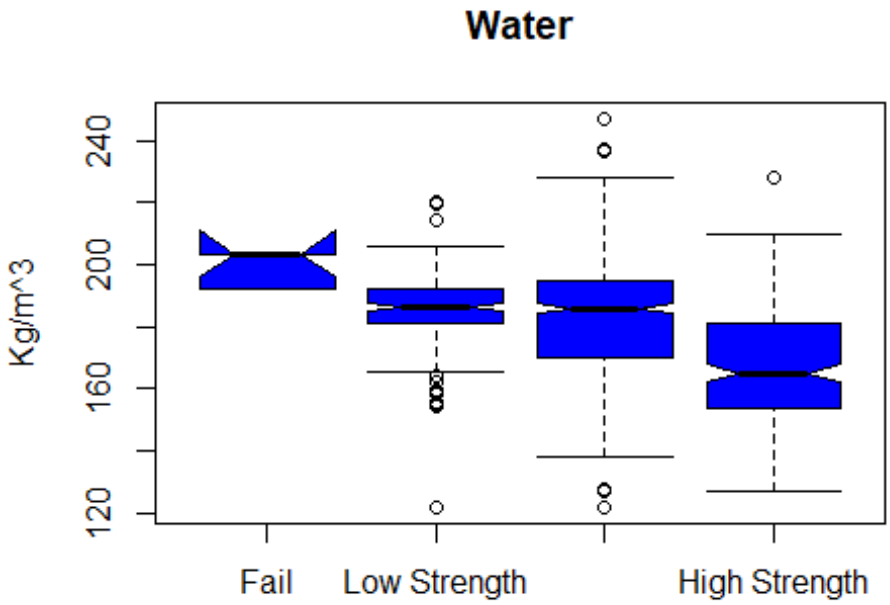
Fly Ash



Fly Ash

The fly ash variable does not show good separation based on compressive strength category.

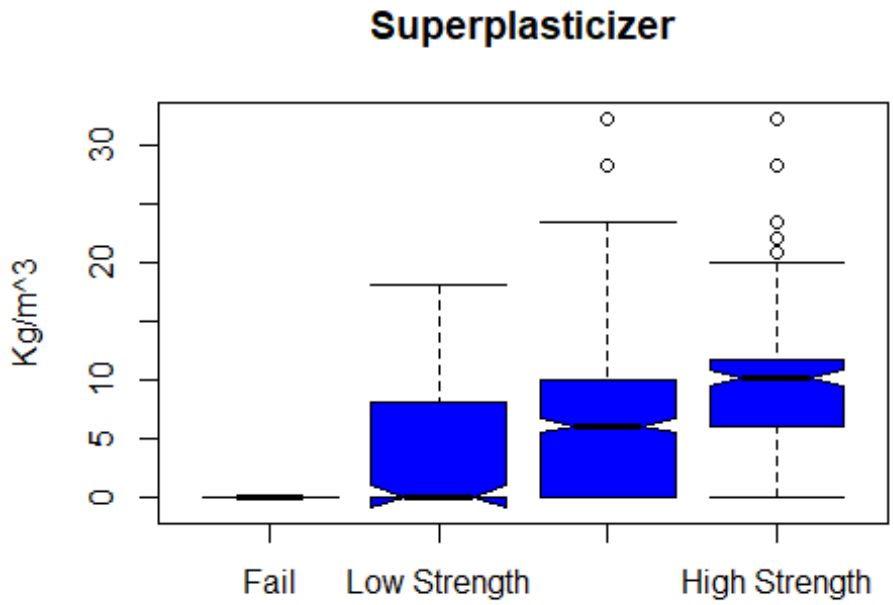
Water



Water

The water variable seems to show a reasonably good separation based on compressive strength category.

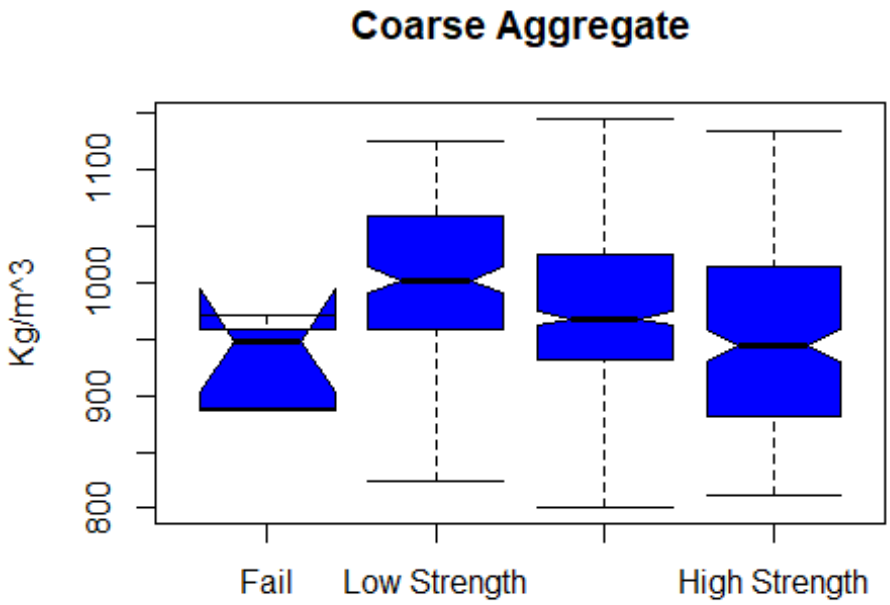
Superplasticizer



Superplasticizer

The superplasticizer variable seems to show a reasonably good separation based on compressive strength category.

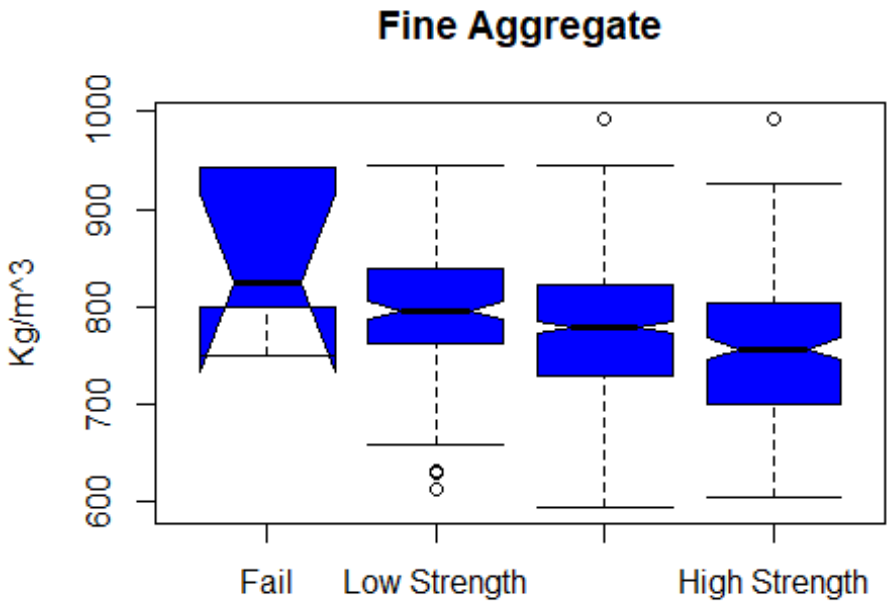
Coarse Aggregate



Coarse Aggregate

The coarse aggregate variable does not show good separation based on compressive strength category.

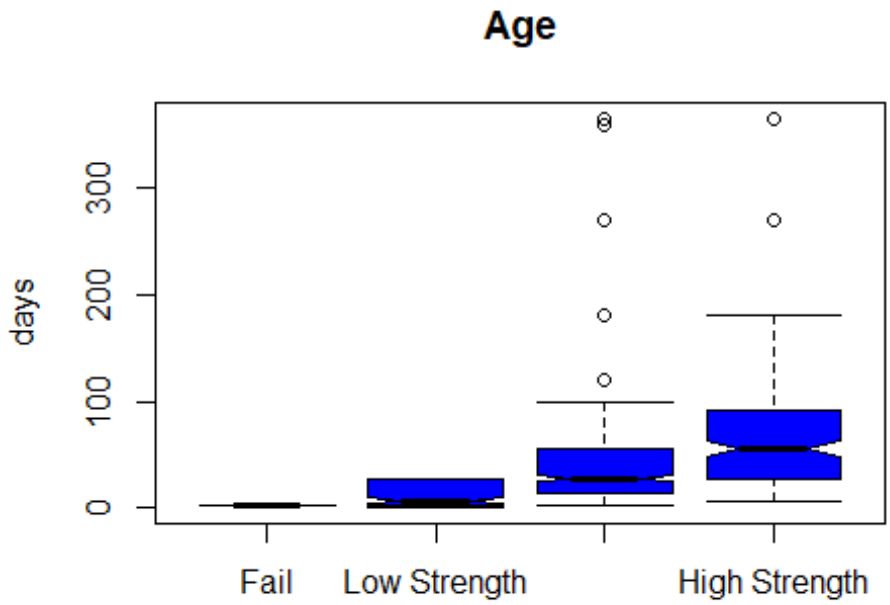
Fine Aggregate



Fine Aggregate

The fine aggregate variable seems to show reasonably good separation based on compressive strength category.

Age



Age

Conclusions from notched box plot viusalization

Based on the notched box plots, the features which might be best suited for separating the classifications are Cement, Age, Water, Superplasticizer and Fine Aggregate.

2.7 - Decision Tree Modelling

Show max and min compressive strength values

```
max(ConDat$Compressive_Strength)
```

```
## [1] 82.59922
```

```
min(ConDat$Compressive_Strength)
```

```
## [1] 2.331808
```

Table of Concrete Strength Categories

```
table(ConDat$Concrete_Category)
```

```
##
##               Fail      Low Strength Standard Strength     High Strength
##                  6               191               623               210
```

There are 6 Fails, 191 Low Strength, 623 standard strength and 210 High Strength

Proportion Table of Concrete Strength Categories

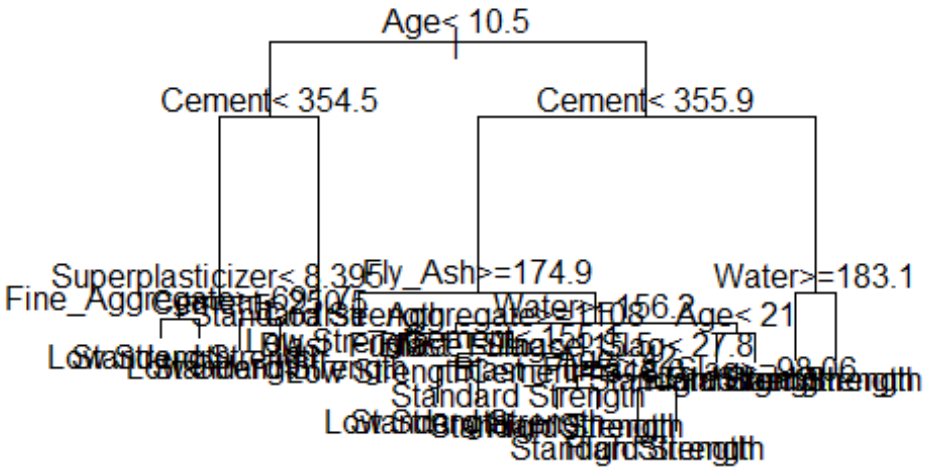
```
prop.table(table(ConDat$Concrete_Category))
```

```
##
##               Fail      Low Strength Standard Strength     High Strength
##       0.005825243       0.185436893       0.604854369       0.203883495
```

In advance of building decision tree model, the dataset Split data into training (80%) and test (20%) datset

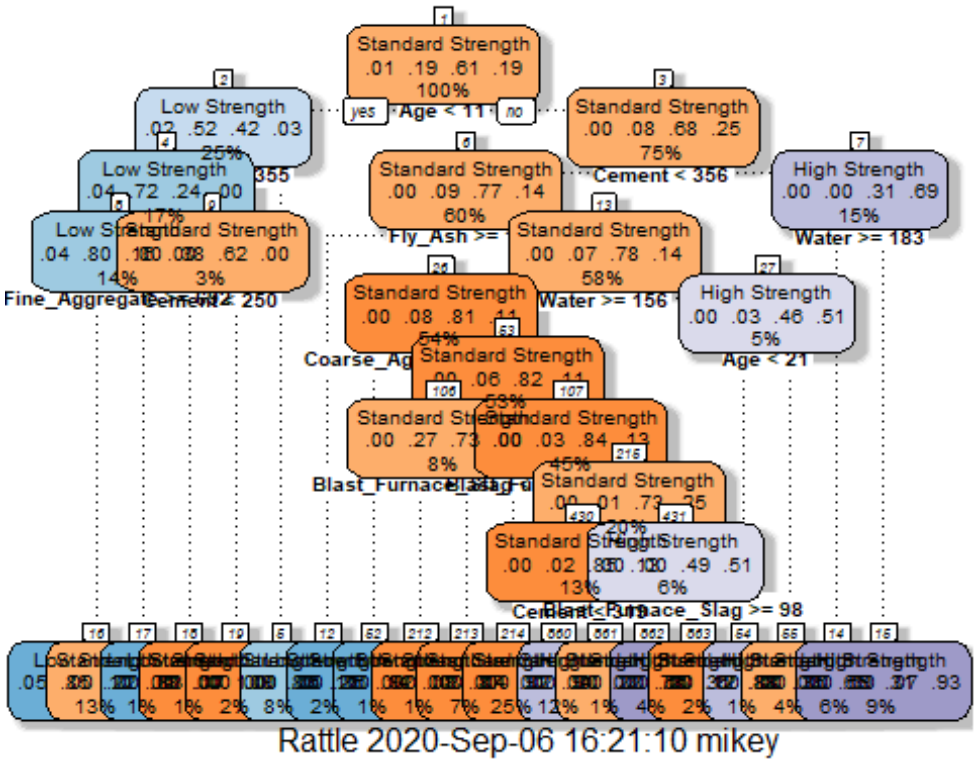
Build Model using rpart

Plot the trees



Plot of Decision Tree Model dt_model1

The tree plot is messy and difficult to interpret - It was decided to create a nicer plot using the fancyRpartPlot function



fancyRpartPlot of Decision Tree Model dt_model1

Make predictions on the test data

```
##                  1                  2                  3                  4
## Standard Strength Standard Strength Standard Strength     High Strength
##                  5                  6
##      Low Strength Standard Strength
## Levels: Fail Low Strength Standard Strength High Strength
```

Compute model accuracy rate on test data

```
## [1] 0.8106796
```

Generate Confusion Matrix

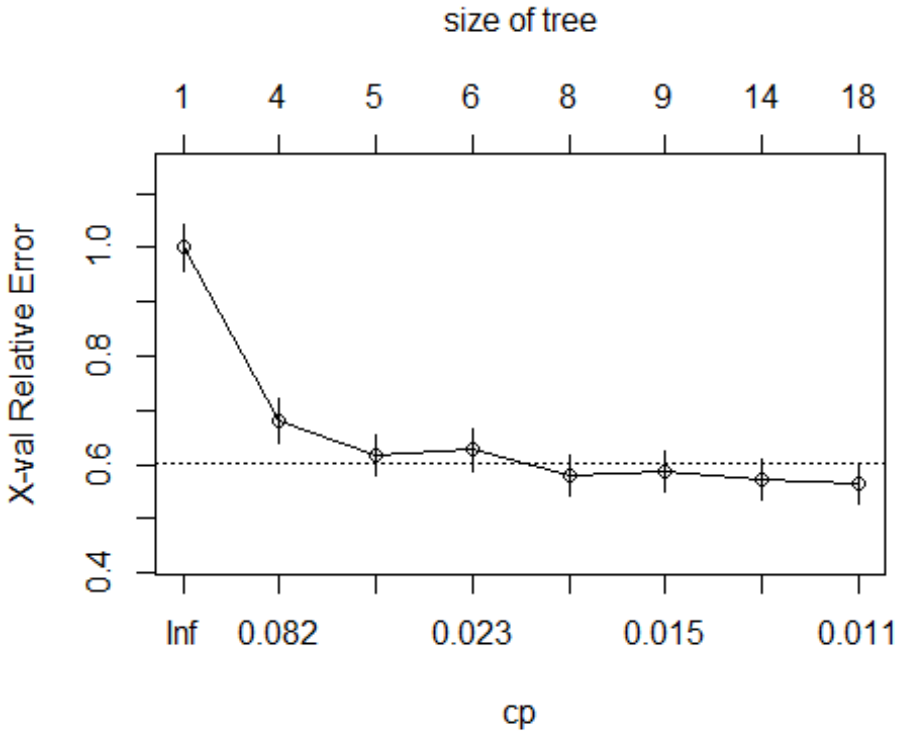
```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction           Fail Low Strength Standard Strength High Strength
##   Fail                  0            0                  0             0
##   Low Strength          1           33                  5             0
##   Standard Strength     0            4                107            23
##   High Strength         0            0                  6            27
##
## Overall Statistics
##
##                Accuracy : 0.8107
```

```
##                  95% CI : (0.7504, 0.8618)
##     No Information Rate : 0.5728
##     P-Value [Acc > NIR] : 4.011e-13
##
##                   Kappa : 0.6586
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Fail Class: Low Strength
## Sensitivity            0.000000              0.8919
## Specificity            1.000000              0.9645
## Pos Pred Value              NaN              0.8462
## Neg Pred Value         0.995146              0.9760
## Prevalence             0.004854              0.1796
## Detection Rate         0.000000              0.1602
## Detection Prevalence   0.000000              0.1893
## Balanced Accuracy      0.500000              0.9282
##                      Class: Standard Strength Class: High Strength
## Sensitivity                            0.9068               0.5400
## Specificity                            0.6932               0.9615
## Pos Pred Value                         0.7985               0.8182
## Neg Pred Value                         0.8472               0.8671
## Prevalence                             0.5728               0.2427
## Detection Rate                         0.5194               0.1311
## Detection Prevalence                   0.6505               0.1602
## Balanced Accuracy                      0.8000               0.7508
```

Print & Plot cp values

```
##
## Classification tree:
## rpart(formula = Concrete_Category ~ ., data = DTTrainSet, method = "class"
)
##
## Variables actually used in tree construction:
## [1] Age               Blast_Furnace_Slag Cement
## [4] Coarse_Aggregate  Fine_Aggregate     Fly_Ash
## [7] Superplasticizer  Water
##
## Root node error: 319/824 = 0.38714
##
## n= 824
##
##         CP nsplit rel error  xerror     xstd
## 1 0.118077      0   1.00000 1.00000 0.043832
## 2 0.056426      3   0.64577 0.68025 0.039634
## 3 0.025078      4   0.58934 0.61755 0.038381
## 4 0.021944      5   0.56426 0.62696 0.038579
```

```
## 5 0.015674      7   0.52038 0.57994 0.037548
## 6 0.015047      8   0.50470 0.58621 0.037691
## 7 0.012539     13   0.42947 0.57053 0.037329
## 8 0.010000     17   0.37618 0.56426 0.037181
```



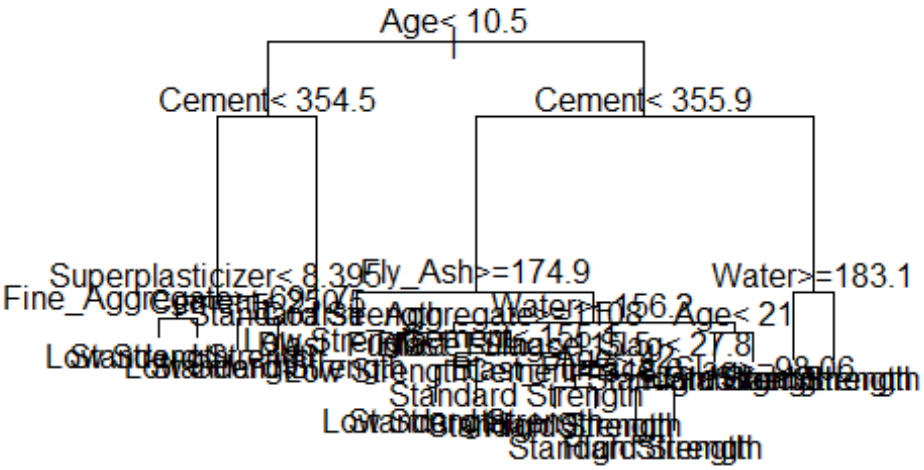
cp Plot of model dt_model1

Find cp value with lowest cross validation error

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

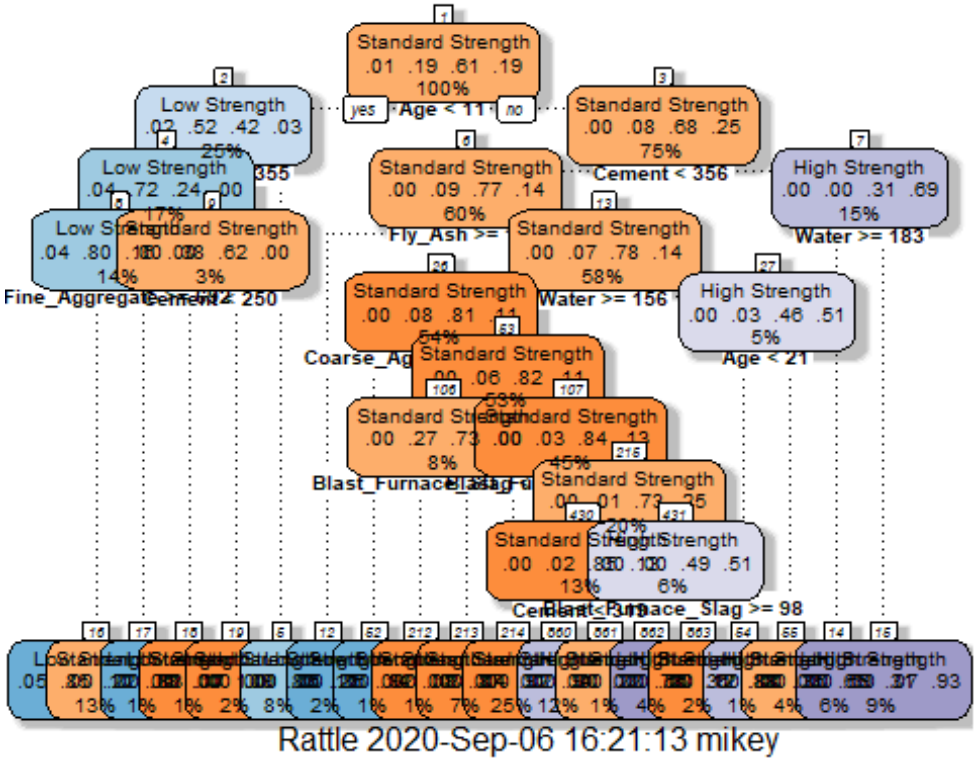
Prune the tree to create a second model dt_model2

plot the dt_model2 trees



Plot of Decision Tree Model dt_model2

Generate nicer plot using fancyRpartPlot



fancyRpartPlot of Decision Tree Model dt_model1

Make predictions on the test data

```
##                  1                  2                  3                  4
## Standard Strength Standard Strength Standard Strength     High Strength
##                  5                  6
##      Low Strength Standard Strength
## Levels: Fail Low Strength Standard Strength High Strength
```

Compute model accuracy rate on test data

```
## [1] 0.8106796
```

Generate Confusion Matrix

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction           Fail Low Strength Standard Strength High Strength
##   Fail                  0            0                  0             0
##   Low Strength          1           33                  5             0
##   Standard Strength     0            4                107            23
```

```
##   High Strength        0            0                 6            27
##
## Overall Statistics
##
##                Accuracy : 0.8107
##                  95% CI : (0.7504, 0.8618)
##     No Information Rate : 0.5728
##     P-Value [Acc > NIR] : 4.011e-13
##
##                   Kappa : 0.6586
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Fail Class: Low Strength
## Sensitivity             0.000000              0.8919
## Specificity             1.000000              0.9645
## Pos Pred Value               NaN              0.8462
## Neg Pred Value          0.995146              0.9760
## Prevalence              0.004854              0.1796
## Detection Rate          0.000000              0.1602
## Detection Prevalence    0.000000              0.1893
## Balanced Accuracy       0.500000              0.9282
##                      Class: Standard Strength Class: High Strength
## Sensitivity                            0.9068               0.5400
## Specificity                            0.6932               0.9615
## Pos Pred Value                         0.7985               0.8182
## Neg Pred Value                         0.8472               0.8671
## Prevalence                             0.5728               0.2427
## Detection Rate                         0.5194               0.1311
## Detection Prevalence                   0.6505               0.1602
## Balanced Accuracy                      0.8000               0.7508
```

Conclusions from Decision Tree Model The final Decision Tree model has an accuracy of 81% which is a reasonably accurate classification model. The kappa value is 0.66. The kappa value takes into account chance agreement, and is defined as:

(observed agreement – expected agreement)/(1 – expected agreement).

When two measurements agree only at the chance level, the value of kappa is zero. When the two measurements agree perfectly, the value of kappa is 1. Sensitivity (also called the true positive rate) measures the percentage of actual positives that are correctly identified. Specificity (also called the true negative rate) measures the percentage of actual negatives that are correctly identified. Sensitivities and Specificities are reasonably high for this model for most categories.

2.8 - Random Forest Modelling

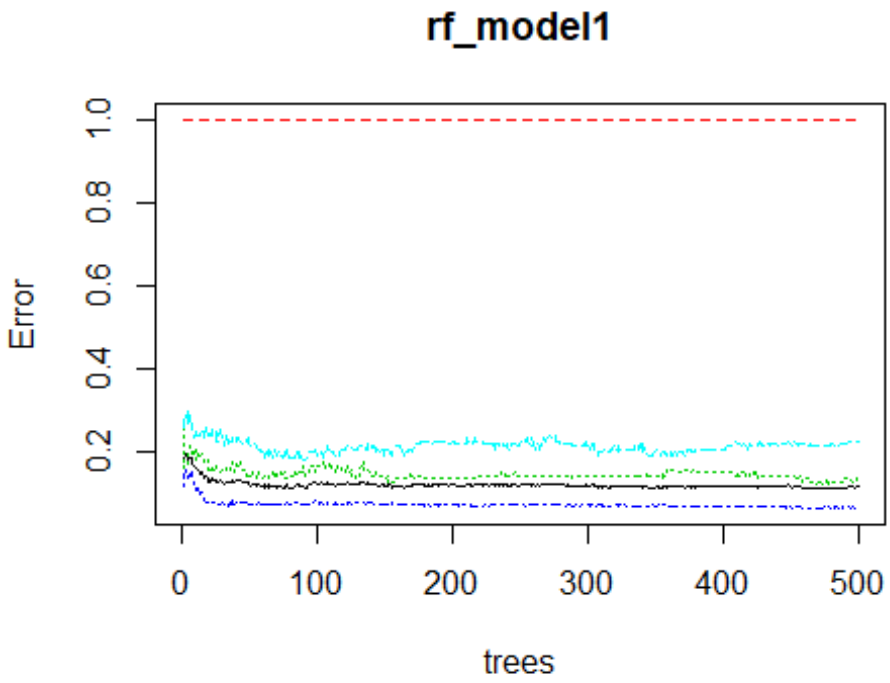
Inspect pre-processed data

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    1030 obs. of  9 variables:
##  $ Cement            : num  540 540 332 332 199 ...
##  $ Blast_Furnace_Slag: num  0 0 142 142 132 ...
##  $ Fly_Ash           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Water             : num  162 162 228 228 192 228 228 228 228 228 ...
##  $ Superplasticizer  : num  2.5 2.5 0 0 0 0 0 0 0 0 ...
##  $ Coarse_Aggregate  : num  1040 1055 932 932 978 ...
##  $ Fine_Aggregate    : num  676 676 594 594 826 ...
##  $ Age               : num  28 28 270 365 360 90 365 28 28 28 ...
##  $ Concrete_Category : Factor w/ 4 levels "Fail","Low Strength",..: 4 4 3
3 3 3 3 3 3 3 ...

##      Cement      Blast_Furnace_Slag    Fly_Ash           Water
##  Min.   :102.0   Min.   :  0.0      Min.   :  0.00   Min.   :121.8
##  1st Qu.:192.4   1st Qu.:  0.0      1st Qu.:  0.00   1st Qu.:164.9
##  Median :272.9   Median : 22.0      Median :  0.00   Median :185.0
##  Mean   :281.2   Mean   : 73.9      Mean   : 54.19   Mean   :181.6
##  3rd Qu.:350.0   3rd Qu.:142.9      3rd Qu.:118.27   3rd Qu.:192.0
##  Max.   :540.0   Max.   :359.4      Max.   :200.10   Max.   :247.0
##  Superplasticizer Coarse_Aggregate Fine_Aggregate       Age
##  Min.   : 0.000   Min.   : 801.0   Min.   :594.0   Min.   :  1.00
##  1st Qu.: 0.000   1st Qu.: 932.0   1st Qu.:731.0   1st Qu.:  7.00
##  Median : 6.350   Median : 968.0   Median :779.5   Median : 28.00
##  Mean   : 6.203   Mean   : 972.9   Mean   :773.6   Mean   : 45.66
##  3rd Qu.:10.160   3rd Qu.:1029.4   3rd Qu.:824.0   3rd Qu.: 56.00
##  Max.   :32.200   Max.   :1145.0   Max.   :992.6   Max.   :365.00
##         Concrete_Category
##  Fail             :  6
##  Low Strength     :191
##  Standard Strength:623
##  High Strength    :210
##
##

## # A tibble: 6 x 9
##   Cement Blast_Furnace_S~ Fly_Ash Water Superplasticizer Coarse_Aggregate
##    <dbl>            <dbl>   <dbl> <dbl>            <dbl>            <dbl>
## 1   540               0        0   162              2.5             1040
## 2   540               0        0   162              2.5             1055
## 3   332.            142.       0   228              0                932
## 4   332.            142.       0   228              0                932
## 5   199.            132.       0   192              0                978.
## 6   266             114        0   228              0                932
## # ... with 3 more variables: Fine_Aggregate <dbl>, Age <dbl>,
## #   Concrete_Category <fct>
```

Prior to builiding the Random Forest model, the dataset was split intoTraining & Validation sets

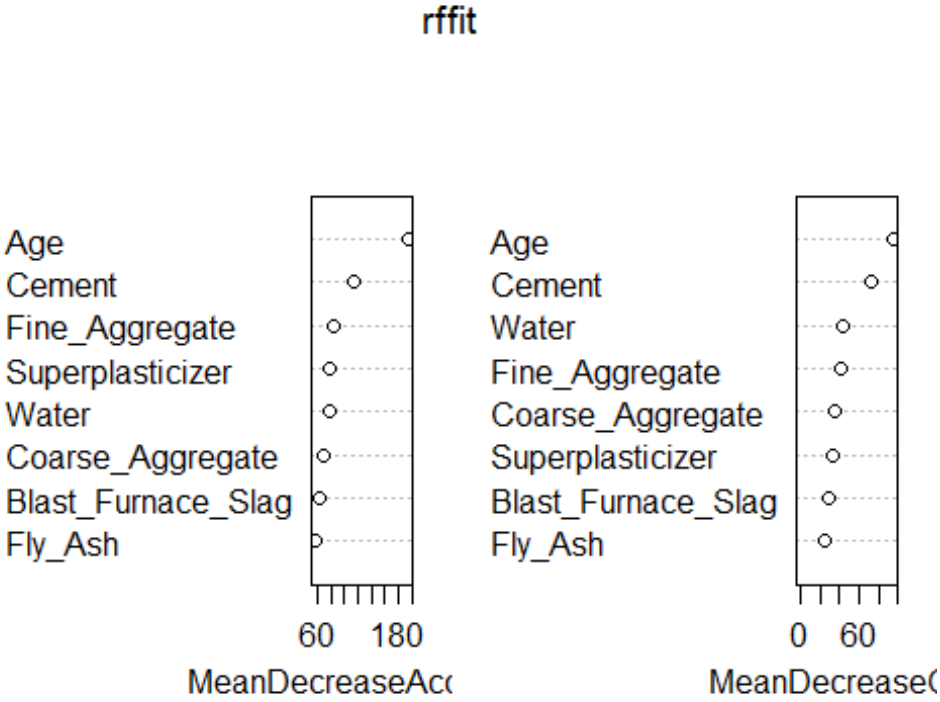
Create first Random Forest Model using all data apart from Concrete Compressive Strength

Plot Random Forest model



Plot of Random Forest model rf_model1

Check which parameters are most influential

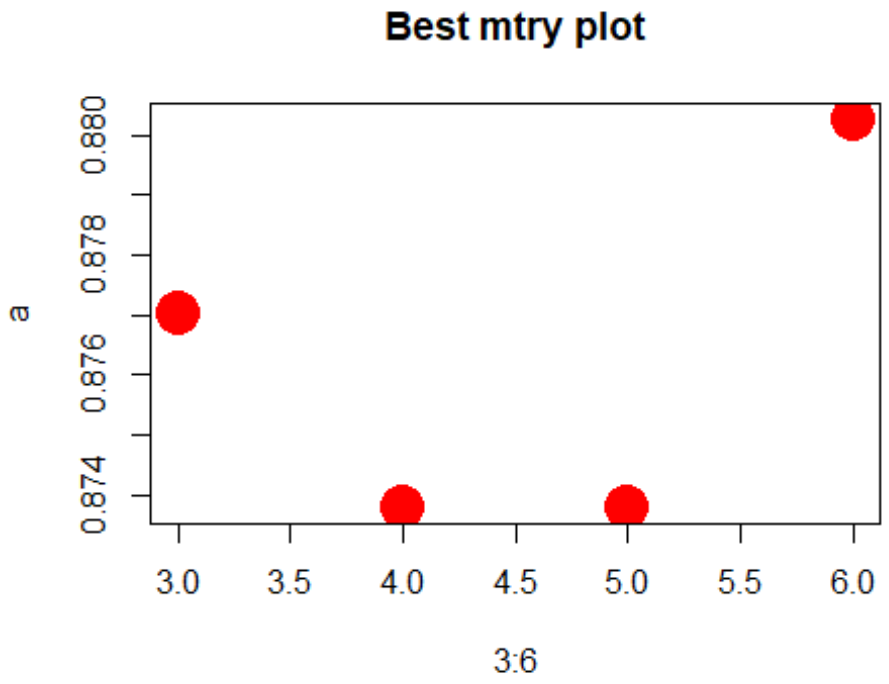


Plot of most influential parameters

From the plot, the most important variables in the random forest model are age, cement, water and fine aggregate.

Using For Loop to identify the right mtry for model

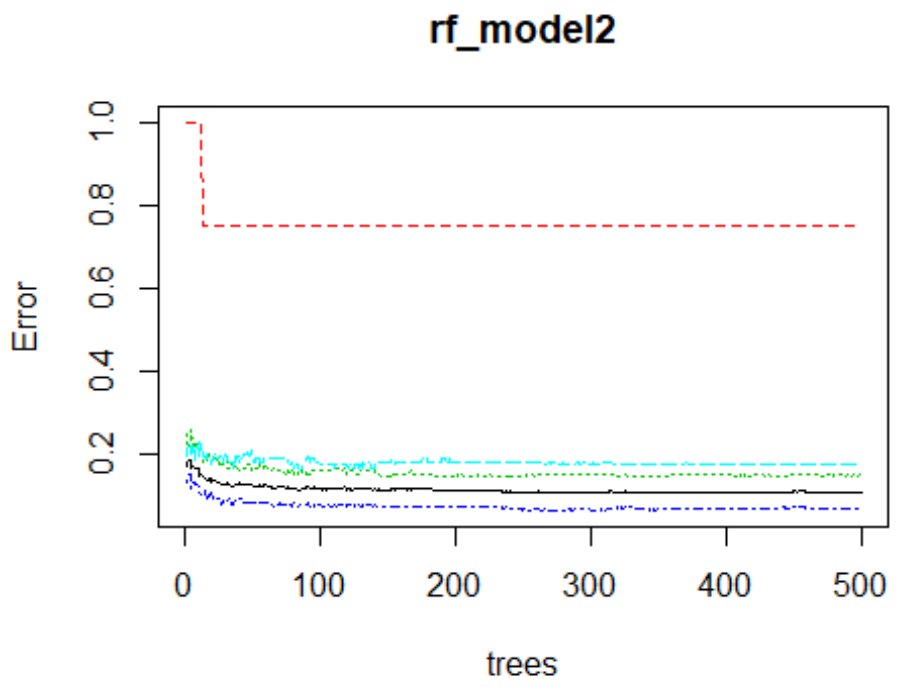
```
## [1] 0.8770227 0.8737864 0.8737864 0.8802589
```



Plot mtry

Fine tuning parameters of Random Forest Model with mtry = 6

```
##
## Call:
##  randomForest(formula = Concrete_Category ~ ., data = RFTrainSet,      ntr
ee = 500, mtry = 4, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 10.96%
## Confusion matrix:
##                   Fail Low Strength Standard Strength High Strength
## Fail                 1            3                 0             0
## Low Strength         2          111                18             0
## Standard Strength    0           16               407            14
## High Strength        0            0                26           123
##                   class.error
## Fail               0.75000000
## Low Strength       0.15267176
## Standard Strength  0.06864989
## High Strength      0.17449664
```



Plot of model rf_model1

Predicting on train set

Checking classification accuracy

```
##
## predTrain           Fail Low Strength Standard Strength High Strength
##   Fail                 4            0                 0             0
##   Low Strength         0          131                 0             0
##   Standard Strength    0            0               436             1
##   High Strength        0            0                 1           148
```

Predicting on Validation Set

Checking classification accuracy

```
## [1] 0.8737864

##
## predValid           Fail Low Strength Standard Strength High Strength
##   Fail                 0            1                 0             0
##   Low Strength         2           50                 4             0
##   Standard Strength    0            9               177            18
##   High Strength        0            0                 5            43

## Confusion Matrix and Statistics
##
```

```
##                     Reference
## Prediction          Fail Low Strength Standard Strength High Strength
##   Fail                 0            1                 0             0
##   Low Strength         2           50                 4             0
##   Standard Strength    0            9               177            18
##   High Strength        0            0                 5            43
##
## Overall Statistics
##
##                Accuracy : 0.8738
##                  95% CI : (0.8315, 0.9087)
##     No Information Rate : 0.6019
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7648
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Fail Class: Low Strength
## Sensitivity             0.000000              0.8333
## Specificity             0.996743              0.9759
## Pos Pred Value          0.000000              0.8929
## Neg Pred Value          0.993506              0.9605
## Prevalence              0.006472              0.1942
## Detection Rate          0.000000              0.1618
## Detection Prevalence    0.003236              0.1812
## Balanced Accuracy       0.498371              0.9046
##                      Class: Standard Strength Class: High Strength
## Sensitivity                            0.9516               0.7049
## Specificity                            0.7805               0.9798
## Pos Pred Value                         0.8676               0.8958
## Neg Pred Value                         0.9143               0.9310
## Prevalence                             0.6019               0.1974
## Detection Rate                         0.5728               0.1392
## Detection Prevalence                   0.6602               0.1553
## Balanced Accuracy                      0.8661               0.8424
```

Conclusions from Random Forest Model

The final Random Forest model has an accuracy of 87% which is a reasonably accurate classification model. The kappa value is 0.76. The kappa value takes into account chance agreement, and is defined as:

(observed agreement – expected agreement)/(1 – expected agreement).

When two measurements agree only at the chance level, the value of kappa is zero. When the two measurements agree perfectly, the value of kappa is 1. Sensitivity (also called the

true positive rate) measures the percentage of actual positives that are correctly identified. Specificity (also called the true negative rate) measures the percentage of actual negatives that are correctly identified. Sensitivities and Specificities are reasonably high for this model for most categories. The Random Forest model has a slightly lower balanced accuracy than the decision tree model but the kappa value is higher. In addition the sensetivities and specificities of the random forest model are higher.

2.9 - Support Vector Machine Modelling

Table of Concrete Category

```
##
##              Fail      Low Strength Standard Strength     High Strength
##                 6               191               623               210
```

Proportion Table of Concrete Category

```
##
##              Fail      Low Strength Standard Strength     High Strength
##              0.01              0.19              0.60              0.20
```

efore building teh SWM model, the datset was scaled and split into Training and Validation datasets.

Build SVM classifier model svm_model1 with linear kernel

```
##
## Call:
## svm(formula = Concrete_Category ~ ., data = svmTrainSet, type = "C-classif
ication",
##     kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  418
##
##  ( 204 98 112 4 )
##
##
## Number of Classes:  4
##
## Levels:
##  Fail Low Strength Standard Strength High Strength
```

Test Model svm_model1 using Validation Set

```
## [1] 0.7055016

##
## svmpred             Fail Low Strength Standard Strength High Strength
##   Fail                 0            0                 0             0
##   Low Strength         2           35                 3             0
##   Standard Strength    0           25               146            24
##   High Strength        0            0                37            37

## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          Fail Low Strength Standard Strength High Strength
##   Fail                 0            0                 0             0
##   Low Strength         2           35                 3             0
##   Standard Strength    0           25               146            24
##   High Strength        0            0                37            37
##
## Overall Statistics
##
##                Accuracy : 0.7055
##                  95% CI : (0.6513, 0.7558)
##     No Information Rate : 0.6019
##     P-Value [Acc > NIR] : 9.747e-05
##
##                   Kappa : 0.4623
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Fail Class: Low Strength
## Sensitivity             0.000000              0.5833
## Specificity             1.000000              0.9799
## Pos Pred Value               NaN              0.8750
## Neg Pred Value          0.993528              0.9071
## Prevalence              0.006472              0.1942
## Detection Rate          0.000000              0.1133
## Detection Prevalence    0.000000              0.1294
## Balanced Accuracy       0.500000              0.7816
##                      Class: Standard Strength Class: High Strength
## Sensitivity                            0.7849               0.6066
## Specificity                            0.6016               0.8508
## Pos Pred Value                         0.7487               0.5000
## Neg Pred Value                         0.6491               0.8979
## Prevalence                             0.6019               0.1974
## Detection Rate                         0.4725               0.1197
## Detection Prevalence                   0.6311               0.2395
## Balanced Accuracy                      0.6933               0.7287
```

Build SVM classifier model svm_model2with radial kernel

```
##
## Call:
## svm(formula = Concrete_Category ~ ., data = svmTrainSet, type = "C-classif
ication",
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  479
##
##  ( 241 119 115 4 )
##
##
## Number of Classes:  4
##
## Levels:
##  Fail Low Strength Standard Strength High Strength
```

Test Model svm_model2 using Validation Set

```
## [1] 0.7508091

##
## svmpred             Fail Low Strength Standard Strength High Strength
##   Fail                 0            0                 0             0
##   Low Strength         2           30                 6             0
##   Standard Strength    0           30               165            24
##   High Strength        0            0                15            37

## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          Fail Low Strength Standard Strength High Strength
##   Fail                 0            0                 0             0
##   Low Strength         2           30                 6             0
##   Standard Strength    0           30               165            24
##   High Strength        0            0                15            37
##
## Overall Statistics
##
##                Accuracy : 0.7508
##                  95% CI : (0.6987, 0.798)
##     No Information Rate : 0.6019
##     P-Value [Acc > NIR] : 2.57e-08
##
```

```
##                   Kappa : 0.5173
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Fail Class: Low Strength
## Sensitivity             0.000000             0.50000
## Specificity             1.000000             0.96787
## Pos Pred Value               NaN             0.78947
## Neg Pred Value          0.993528             0.88930
## Prevalence              0.006472             0.19417
## Detection Rate          0.000000             0.09709
## Detection Prevalence    0.000000             0.12298
## Balanced Accuracy       0.500000             0.73394
##                      Class: Standard Strength Class: High Strength
## Sensitivity                            0.8871               0.6066
## Specificity                            0.5610               0.9395
## Pos Pred Value                         0.7534               0.7115
## Neg Pred Value                         0.7667               0.9066
## Prevalence                             0.6019               0.1974
## Detection Rate                         0.5340               0.1197
## Detection Prevalence                   0.7087               0.1683
## Balanced Accuracy                      0.7240               0.7730
```

Conclusions from SVM model

The best Support Vector Machine model was achieved using a radial kernel. The model has a balanced accuracy of 75% which is a reasonably accurate classification model. The kappa value is 0.51 which is quite low. The kappa value takes into account chance agreement, and is defined as:

(observed agreement – expected agreement)/(1 – expected agreement).

When two measurements agree only at the chance level, the value of kappa is zero. When the two measurements agree perfectly, the value of kappa is 1. Sensitivity (also called the true positive rate) measures the percentage of actual positives that are correctly identified. Specificity (also called the true negative rate) measures the percentage of actual negatives that are correctly identified. Sensitivities and Specificities are moderate to high for this model for most categories. The SWM model is not as good as either the Decision Tree or Random Forest Models.

2.10 - K-Nearest Neighbour Modelling

Prior to building the knn model, the dataset was normalized and split into Training and Validation datasets.

Calculate prediction accuracy

```
## [1] 0.6504854

## Confusion Matrix and Statistics
##
##                     Reference
## Prediction           Fail Low Strength Standard Strength High Strength
##   Fail                  0            0                 0             0
##   Low Strength          0           18                12             0
##   Standard Strength     0           48               170            18
##   High Strength         0            1                29            13
##
## Overall Statistics
##
##                Accuracy : 0.6505
##                  95% CI : (0.5945, 0.7036)
##     No Information Rate : 0.6828
##     P-Value [Acc > NIR] : 0.8996
##
##                   Kappa : 0.2118
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Fail Class: Low Strength
## Sensitivity                   NA             0.26866
## Specificity                    1             0.95041
## Pos Pred Value                NA             0.60000
## Neg Pred Value                NA             0.82437
## Prevalence                     0             0.21683
## Detection Rate                 0             0.05825
## Detection Prevalence           0             0.09709
## Balanced Accuracy             NA             0.60953
##                      Class: Standard Strength Class: High Strength
## Sensitivity                            0.8057              0.41935
## Specificity                            0.3265              0.89209
## Pos Pred Value                         0.7203              0.30233
## Neg Pred Value                         0.4384              0.93233
## Prevalence                             0.6828              0.10032
## Detection Rate                         0.5502              0.04207
```

```
## Detection Prevalence                    0.7638              0.13916
## Balanced Accuracy                       0.5661              0.65572
```

Conclusions from the K Nearest Neighbour Model

The model has an accuracy of 65% which is not a very accurate classification model. The kappa value is 0.21 which is very low. The kappa value takes into account chance agreement, and is defined as:

(observed agreement – expected agreement)/(1 – expected agreement).

When two measurements agree only at the chance level, the value of kappa is zero. When the two measurements agree perfectly, the value of kappa is 1. Sensitivity (also called the true positive rate) measures the percentage of actual positives that are correctly identified. Specificity (also called the true negative rate) measures the percentage of actual negatives that are correctly identified. Sensitivities and Specificities are low to moderate for this model for most categories. The knn model is not as good as the Decision Tree, Random Forest or Support Vector Machines Models.

2.11 - Naive-Bayes Modelling

Table of Concrete Strength Categories

```
##
##              Fail      Low Strength Standard Strength     High Strength
##                 6               191               623               210

##
##              Fail      Low Strength Standard Strength     High Strength
##              0.01              0.19              0.60              0.20
```

Prior to building the Naive-Bayes model, the datset was split into Training and Validation sets.

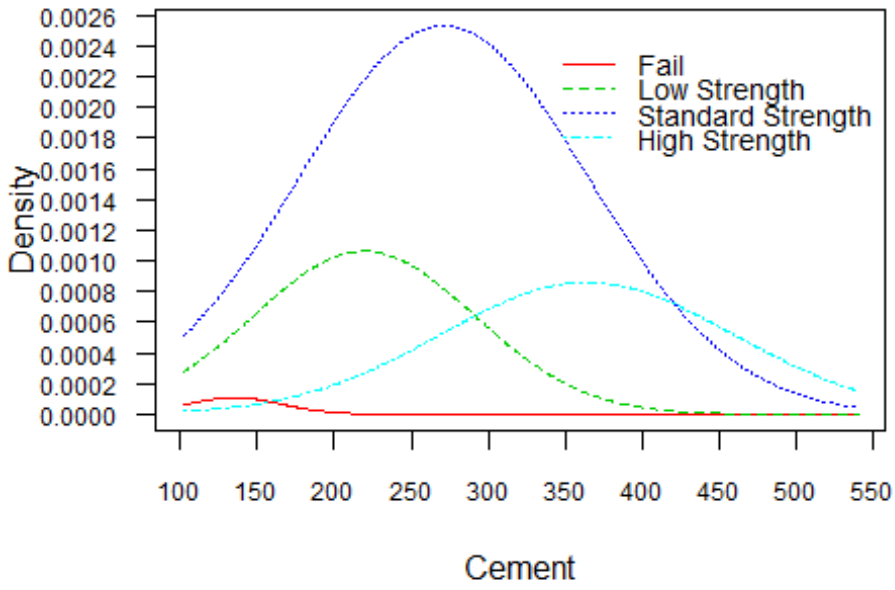
Build Naive-Bayes Model nb_model1

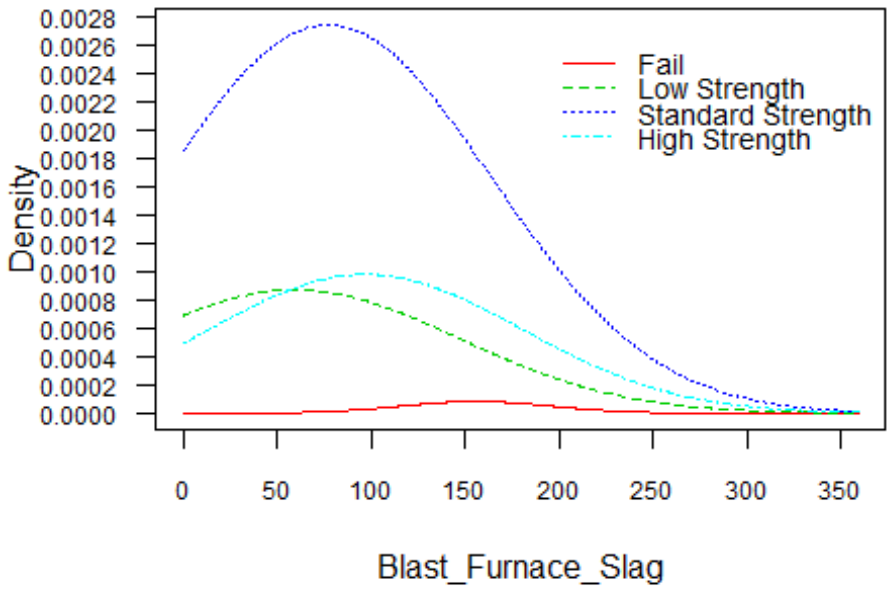
```
##
## ================================ Naive Bayes ============================
====
##
##  Call:
## naive_bayes.formula(formula = Concrete_Category ~ ., data = nbTrainSet,
##     laplace = 0)
##
## ------------------------------------------------------------------------
----
##
## Laplace smoothing: 0
##
## ------------------------------------------------------------------------
----
```

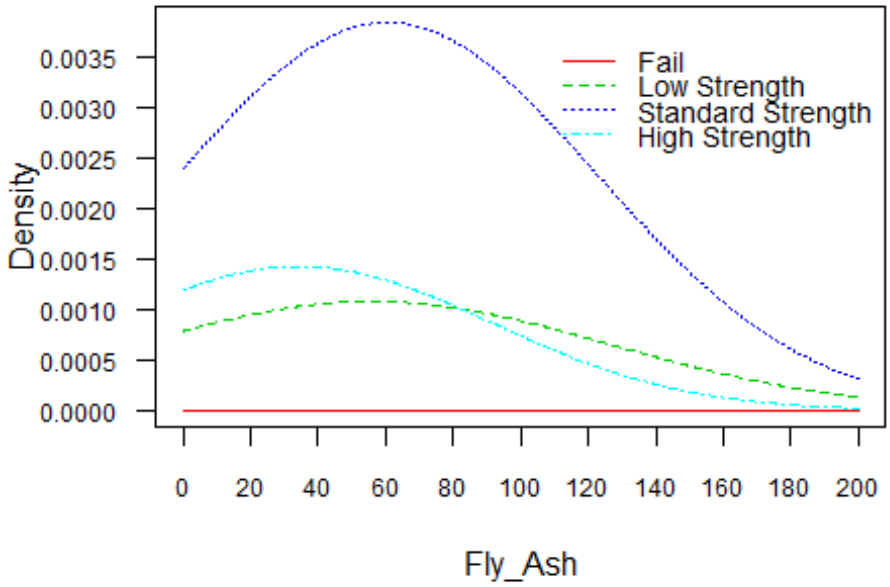
```
##
##  A priori probabilities:
##
##              Fail      Low Strength Standard Strength     High Strength
##       0.008321775       0.190013870       0.597780860       0.203883495
##
## ------------------------------------------------------------------------
----
##
##  Tables:
##
## ------------------------------------------------------------------------
----
##  ::: Cement (Gaussian)
## ------------------------------------------------------------------------
----
##
## Cement      Fail Low Strength Standard Strength High Strength
##   mean 135.18333    219.57234         271.06195     364.01088
##   sd    30.69315     71.36910          94.23713      94.71073
##
## ------------------------------------------------------------------------
----
##  ::: Blast_Furnace_Slag (Gaussian)
## ------------------------------------------------------------------------
----
##
## Blast_Furnace_Slag      Fail Low Strength Standard Strength High Strength
##               mean 155.98333     59.84416          77.24817      97.10796
##               sd    39.98982     87.01107          86.96463      82.99382
##
## ------------------------------------------------------------------------
----
##  ::: Fly_Ash (Gaussian)
## ------------------------------------------------------------------------
----
##
## Fly_Ash     Fail Low Strength Standard Strength High Strength
##    mean  0.00000     56.00818          60.59561      34.49755
##    sd    0.00000     70.38872          62.23867      57.13770
##
## ------------------------------------------------------------------------
----
##  ::: Water (Gaussian)
## ------------------------------------------------------------------------
----
##
## Water        Fail Low Strength Standard Strength High Strength
##   mean 199.666667   185.776934        184.135452    170.187211
##   sd     5.938574    15.189181         21.045006     24.270141
```

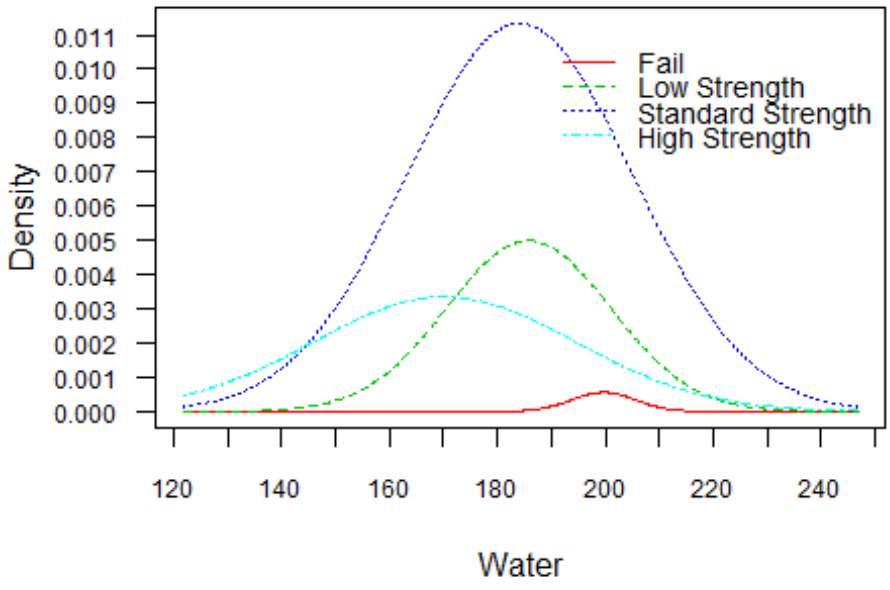
```
##
## ------------------------------------------------------------------------
----
##  ::: Superplasticizer (Gaussian)
## ------------------------------------------------------------------------
----
##
## Superplasticizer     Fail Low Strength Standard Strength High Strength
##             mean 0.000000     3.866328          5.973768      9.907415
##             sd   0.000000     4.833989          5.163850      7.216659
##
## ------------------------------------------------------------------------
----
##
## # ... and 3 more tables
##
## ------------------------------------------------------------------------
----

##
## ================================ Naive Bayes ============================
====
##
## - Call: naive_bayes.formula(formula = Concrete_Category ~ ., data = nbTrai
nSet,      laplace = 0)
## - Laplace: 0
## - Classes: 4
## - Samples: 721
## - Features: 8
## - Conditional distributions:
##     - Gaussian: 8
## - Prior probabilities:
##     - Fail: 0.0083
##     - Low Strength: 0.19
##     - Standard Strength: 0.5978
##     - High Strength: 0.2039
##
## ------------------------------------------------------------------------
----
```

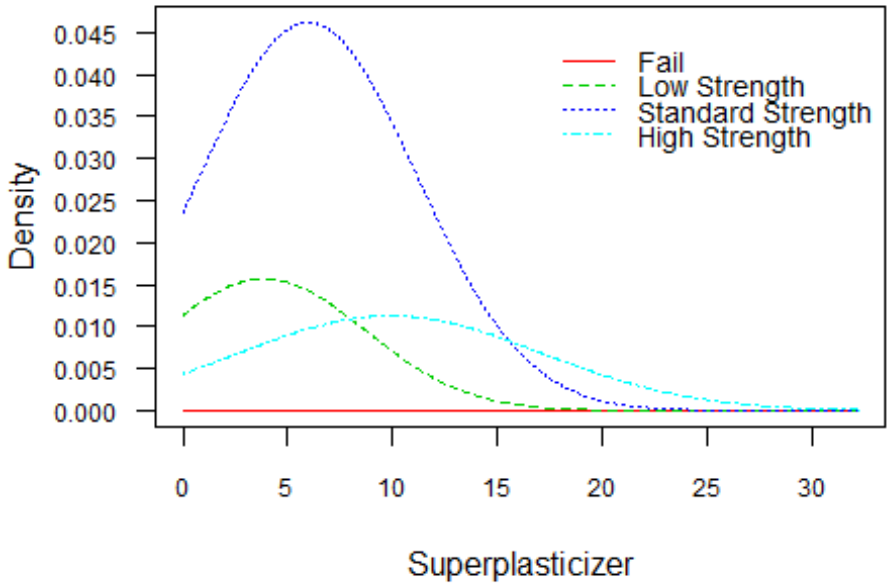
Plot Naive-Bayes Model

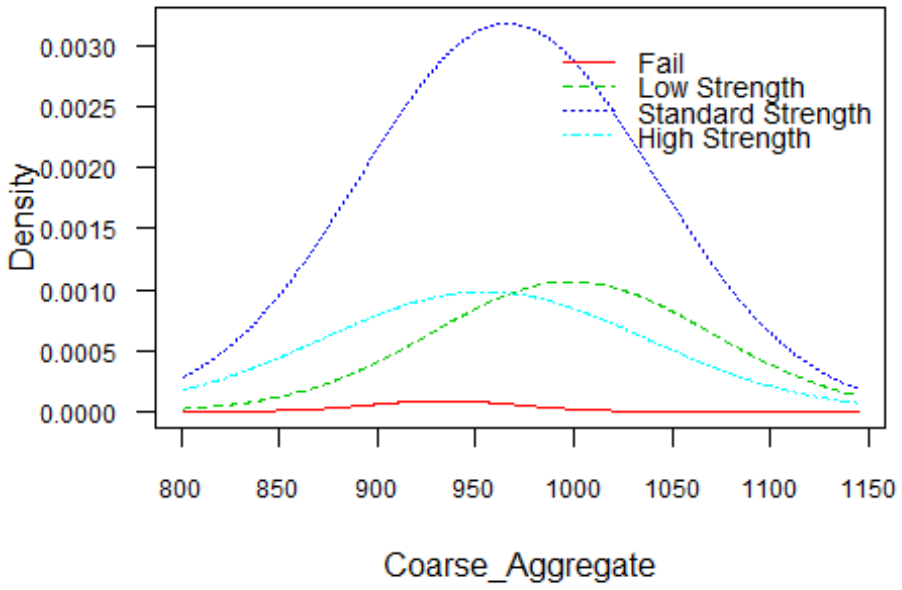


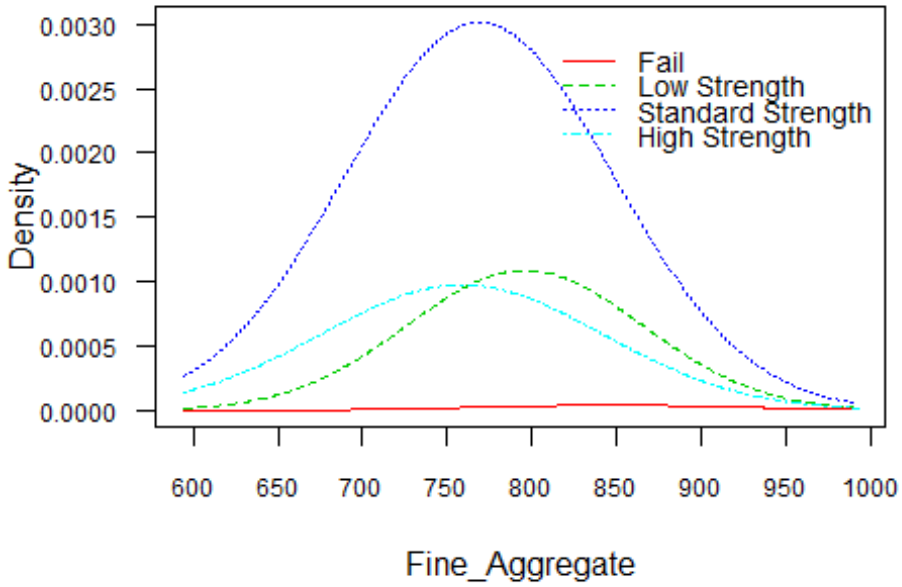


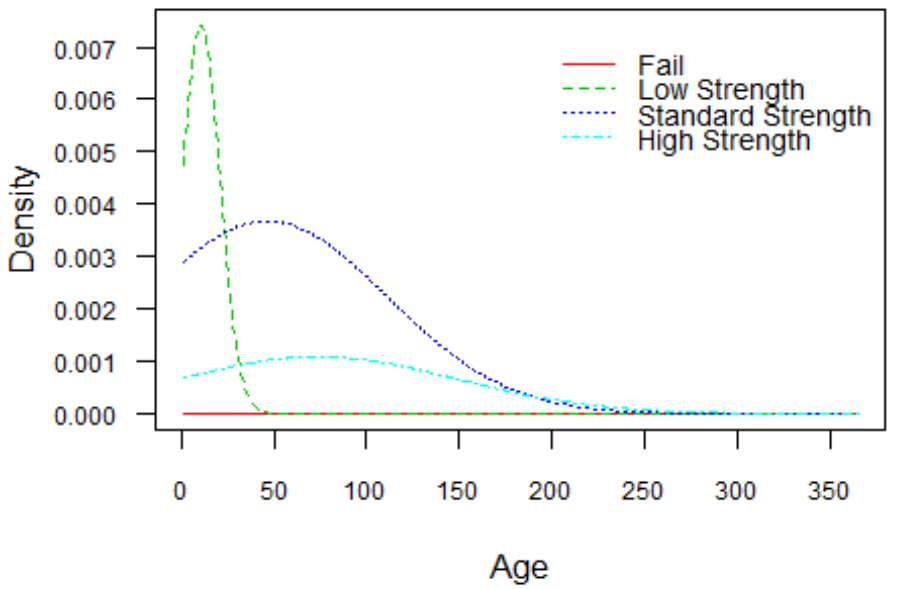

Density
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
20
40
60
80
100
120
140
160
180
200
Fly_Ash
Fail
Low Strength
Standard Strength
High Strength


Density
0.011
0.010
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
120
140
160
180
200
220
240
Water
Fail
Low Strength
Standard Strength
High Strength


Density
0.045
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
5
10
15
20
25
30
Superplasticizer
Fail
Low Strength
Standard Strength
High Strength


Density
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
800
850
900
950
1000
1050
1100
1150
Coarse_Aggregate
Fail
Low Strength
Standard Strength
High Strength


Density
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
600
650
700
750
800
850
900
950
1000
Fine_Aggregate
Fail
Low Strength
Standard Strength
High Strength


Density
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
0
50
100
150
200
250
300
350
Age
Fail
Low Strength
Standard Strength
High Strength

Predict using model and Validation Set

Check model Accuracy using Validation Set

```
## [1] 0.5242718

##
## nbPred              Fail Low Strength Standard Strength High Strength
##   Fail                 0           17                27             1
##   Low Strength         0           26                40             0
##   Standard Strength    0           11               107            33
##   High Strength        0            0                18            29

## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          Fail Low Strength Standard Strength High Strength
##   Fail                 0           17                27             1
##   Low Strength         0           26                40             0
##   Standard Strength    0           11               107            33
##   High Strength        0            0                18            29
##
## Overall Statistics
##
##                Accuracy : 0.5243
##                  95% CI : (0.467, 0.5811)
##     No Information Rate : 0.6214
##     P-Value [Acc > NIR] : 0.9998
##
##                   Kappa : 0.2425
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Fail Class: Low Strength
## Sensitivity                   NA             0.48148
## Specificity               0.8544             0.84314
## Pos Pred Value                NA             0.39394
## Neg Pred Value                NA             0.88477
## Prevalence                0.0000             0.17476
## Detection Rate            0.0000             0.08414
## Detection Prevalence      0.1456             0.21359
## Balanced Accuracy             NA             0.66231
##                      Class: Standard Strength Class: High Strength
## Sensitivity                            0.5573              0.46032
## Specificity                            0.6239              0.92683
## Pos Pred Value                         0.7086              0.61702
## Neg Pred Value                         0.4620              0.87023
```

```
## Prevalence                          0.6214              0.20388
## Detection Rate                      0.3463              0.09385
## Detection Prevalence                0.4887              0.15210
## Balanced Accuracy                   0.5906              0.69357
```

Conclusions from the Naive-Bayes Model

The model has an accuracy of 52% which is not a very accurate classification model. The kappa value is 0.24 which is very low. The kappa value takes into account chance agreement, and is defined as:

(observed agreement – expected agreement)/(1 – expected agreement).

When two measurements agree only at the chance level, the value of kappa is zero. When the two measurements agree perfectly, the value of kappa is 1. Sensitivity (also called the true positive rate) measures the percentage of actual positives that are correctly identified. Specificity (also called the true negative rate) measures the percentage of actual negatives that are correctly identified. Sensitivities and Specificities are low to moderate for this model for most categories. The Naive-Bayes model was the least accurate of the models built.

3.0 Results

The model accuracy for each of the models is detailed below.

Table of Concrete Compressive Strength M/L Model Results

	Multiple Linear Regression	Artificial Neural Network	Decision Trees	Random Forest	Support Vector Machine	K Nearest Neighbour	Naive-Bayes
Accuracy	0.90	0.79	0.81	0.87	0.77	0.66	0.52
Adjusted R-Squared	0.89	NA	NA	NA	NA	NA	NA
RMSE	7.31	NA	NA	NA	NA	NA	NA
MAPE	0.22	NA	NA	NA	NA	NA	NA
Kappa	NA	NA	0.66	0.76	0.51	0.21	0.24

4.0 Conclusion/Discussion

During this project a number of machine learning algorithms were used to build models to predict concrete compressive strengtth:

- Multiple Linear Regression (MLR)
- Artificial Neural Networks (Ann)

- Decision Trees
- Random Forest
- Support Vector Machine (SVM)
- K Nearest Neighbour (Knn)
- Naive-Bayes

For the multiple linear regression and artificial neural networks models, the concrete compressive strength data was left as continous data. However for the clssification algorithms (Decition Trees, Random Forest, Support Vector Machine, K nearest neighbour, Naive-Bayes), it was decided to Create four categories for the concrete compressive strength data based on typical classification found in industry:

- Fail - <5Mpa
- Low Strength 5-19MPa
- Standard Strength 20-49MPa
- High Strength >= 50MPa

Based on the initial exploratory data abnalysis, it appeared that that the most significant positive correlations were achieved for Cement (0.4978327), Superplasticizer (0.3661023) and Age (0.32887). The most significant negative correlation is for Water (-0.2896135).

Multiple Linear Regression Model

The majority of the project effort focused on the The multiple linear regression model. None of the correlations are very strong which (in conjunction with the observatiuons from the scatterplots) indicated that the relationships between Compressive_Strength and the other attributes in the dataset may be non-linear and required further transformation to develop a more accurate linear regression model. Through research and data transformations The final model has an accuracy of approximately 90% (0.8979650) which is quite an accurate model. The final Multiple Linear Regression model (after transformation of the data) has an accuracy of 90% which represents a good model. The Adjusted R-squared value is 0.8081 which means that the model explains about 80% of the variability. The RMSE of the model is 7.3 and the MAPE is only 0.22 indicating a reasonably accurate model. The model meets all of the assumptions for a regression model apart from normality. Homoscedasticity and linearity tend to be more important than normality for linear regression models and with large datasets, normality of lesser importance for a model to be valid. (Schmidt, A. F. & Finan, C., 2018. Linear regression and the normality assumption. Journal of Clinical Epidemiology, Volume 98, pp. 146-151). As the Training dataset has >800 data points, the requirement for normality for the model to be valid is less important.

Artificial Neural Network Model

The best ANN model (after transformation of the data) was achieved using 5 hidden neurons and logistic activation function. The model has an accuracy of 79% which represents a reasobally accurate model. THis model is not as good as the multiple linear

regression model. ANN is best used for very large datasets rather than small data sets like this.

Decision Tree Model

The best Decision Tree model has an accuracy of 81% which is a reasonably accurate classification model. The kappa value is 0.66. The kappa value takes into account chance agreement, and is defined as:

- (observed agreement – expected agreement)/(1 – expected agreement).

When two measurements agree only at the chance level, the value of kappa is zero. When the two measurements agree perfectly, the value of kappa is 1. Sensitivity (also called the true positive rate) measures the percentage of actual positives that are correctly identified. Specificity (also called the true negative rate) measures the percentage of actual negatives that are correctly identified. Sensitivities and Specificities are reasonably high for this model for most categories.

Random Forest Model

The final Random Forest model has an accuracy of 87% which is a reasonably accurate classification model. The kappa value is 0.76. Sensitivities and Specificities are reasonably high for this model for most categories. The Random Forest model has a slightly lower balanced accuracy than the decision tree model but the kappa value is higher. In addition the sensetivities and specificities of the random forest model are higher.

Support Vector Machine Model

The best Support Vector Machine model was achieved using a radial kernel. The model has a balanced accuracy of 75% which is a reasonably accurate classification model. The kappa value is 0.51 which is quite low. Sensitivities and Specificities are moderate to high for this model for most categories. The SWM model is not as good as either the Decision Tree or Random Forest Models.

K Nearest Neighbour Model

The knn model has an accuracy of 65% which is not a very accurate classification model. The kappa value is 0.21 which is very low. Sensitivities and Specificities are low to moderate for this model for most categories. The knn model is not as good as the Decision Tree, Random Forest or Support Vector Machines Models.

K Naive-Bayes Model

The model has a balanced accuracy of 52% which is not a very accurate classification model. The kappa value is 0.24 which is very low. Sensitivities and Specificities are low to moderate for this model for most categories. The Naive-Bayes model was the least accurate of the models built

Overall Conclusion

In summary, the best model was achieved using multiple linear regression. The best classification model werse achieved using the Random Forest algorithm. The worst models were obtained using knn and Naive-Bayes algorithms.