Table of Contents

Abstract

This paper describes and shows some data pre-processing using a company collected information's on concrete and their amounts in the past 10 years of its cement production. The dataset contains about the cement, blast furnace slag, fly ash, water, super plasticizer, coarse aggregate, fine aggregate, age and strength which can also be called as columns in simple term and 1030 rows or the data of the given columns. The objective of this paper is to understand the data by reporting its interesting facts and pre-processing the data (missing values, outliers, multi-collinearity, investigate variables and scaling).

Introduction

The analytical problem of this paper selected case study are to investigate messiness by column and make decision data clearance, stating a technique to detect outliers, multi-collinearity, investigate the variables and scaling. The data set used in this paper is from a cement producer company containing the information about the concrete and their amount in the last 10 years of its cement production. This was collected to optimize the cement testing time by implementing a model that will allow accurate concrete strength prediction.

Before technologies evolvement, concrete strengths were tested by making a cubic block and compressing it with a compression testing machine after certain period of curing. (M.Lessard, 1993) (H.Shi, 2009). It has not proven to be a cost effective test, which is one of the reason researcher have been moving on to new ways to predict the strength. (S.Bhanja, 2002) (B.Bharatkumar, 2001) (M.F.M.Zain, 2009). Using regression method is one way but while using it summarizing the exact regission expression is very hard. (X.Zhu, 2011) (Z.Chen, 2021). That is one reason not to use regression method, this brings researcher to new way which is by predicing using machine learning models to handle regression problem (H.Salehi, 2018). Some of the successful algorithm used cases are, Chithra (S.Chithra, 2016) Use of ANN to predict the strength of silica nanoparticles and copper slag contained concrete.. Ayat (H.Ayat, 2018) Predicting the compressive strength of concrete filled with limestone with a correlation coefficient value as high as 0.976 using ANN.. Nguyen (H.Nguyen, 2021) proposed four machine learning algorithms for predicting the compressive and tensile strength of HPC, with the prediction models based on gradient boosting regressor (GBR) and extreme gradient boosting (XGBoost) having good output accuracy. kumar (A.kumar, 2022) Using SVM to predict the compressive strength of lightweight concrete.. Ashrafian (A.Ashrafian, 2018) Predicting fibrous concrete strength using the heuristic regression and lastly Zhang (J.Zhang, 2019) Predicting the uniaxial compressive strength of lightweight self-compacting concrete using random forest. The highly used algorithms out there are individual learning algorithms because training multiple of individual learning algorithms helps in better accuracy and robustness. (M.S.Barkhordari, 2022).

With all of these experimented proofs out there use technology is without a doubt a better and modern way to measure the strength of concrete.

Understanding the data

In the concrete strength dataset two datasets were given, which was combined into one namely combined dataset for this project. The total number of columns were 9 and rows were 1030.

ADAAssesment.R* | combined | concrete_strength_train | combineSets.R | concrete_strength_te

Filter

	Cement	Blast.Furnace.Slag	Fly.Ash	Water	Superplasticizer	Coarse.Aggregate	Fine.Aggregate	A
1	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	
2	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	
3	332.5	142.5	0.0	228.0	NA	932.0	594.0	
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	
5	266.0	114.0	0.0	228.0	0.0	932.0	670.0	
6	266.0	114.0	0.0	228.0	0.0	932.0	670.0	
7	475.0	0.0	0.0	228.0	0.0	932.0	594.0	
8	198.6	132.4	0.0	192.0	0.0	978.4	825.5	
9	198.6	132.4	0.0	192.0	0.0	978.4	825.5	
10	427.5	47.5	0.0	228.0	NA	932.0	594.0	
11	190.0	190.0	0.0	228.0	0.0	932.0	670.0	
12	304.0	76.0	NA	228.0	0.0	932.0	670.0	
13	380.0	NA	0.0	228.0	0.0	932.0	670.0	
14	139.6	209.4	0.0	192.0	0.0	1047.0	806.9	
15	NA	38.0	0.0	228.0	0.0	932.0	670.0	
16	380.0	95.0	0.0	228.0	0.0	932.0	594.0	
17	475.0	0.0	0.0	228.0	0.0	932.0	594.0	

Showing 1 to 17 of 1,030 entries, 9 total columns

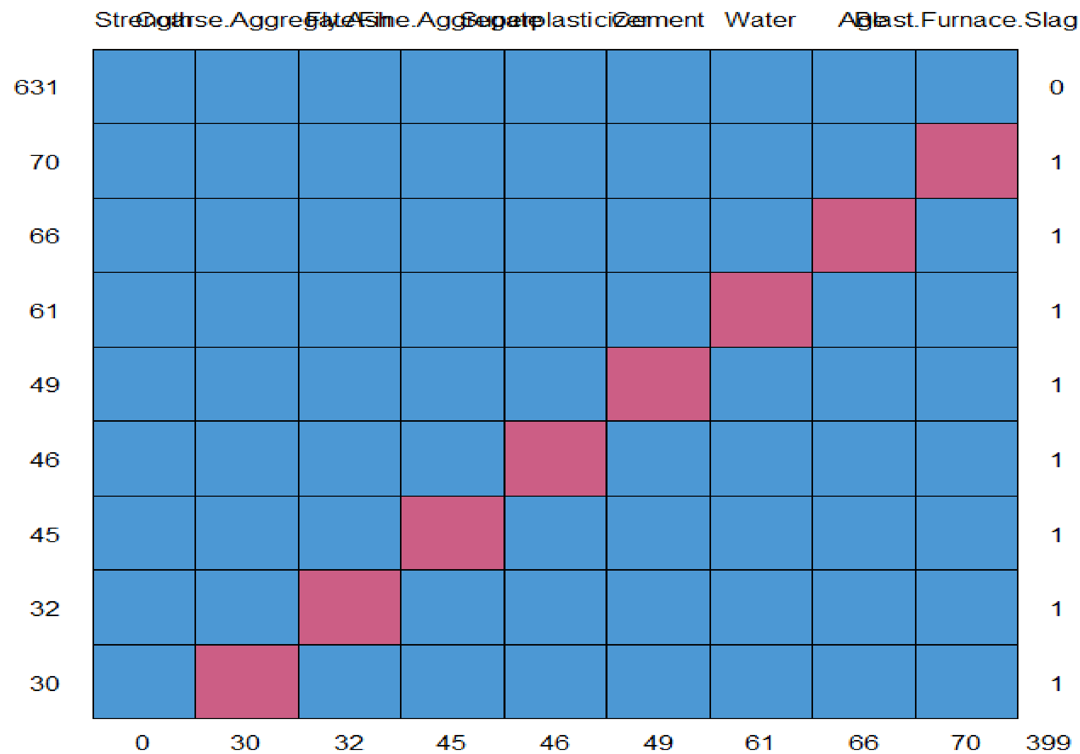
Clearly the dataset has a lot of empty rows, following commands were run to check the missing rows according to the columns.

```
> colSums(sapply(combined, is.na))
            Cement Blast.Furnace.Slag            Fly.Ash              Water   Superplasticizer 
                49                 70                 32                 61                 46 
  Coarse.Aggregate     Fine.Aggregate                Age           Strength 
                30                 45                 66                  0 
```

This stated that there were 49 entries missing in cement, 70 in blast furnace slag, 32 in fly ash, 61 in water, 46 in super plasticizer, 30 in coarse aggregate, 45 in fine aggregate, 66 in age and 0 in strength column.

```
> sum(missing_data)
[1] 399
```

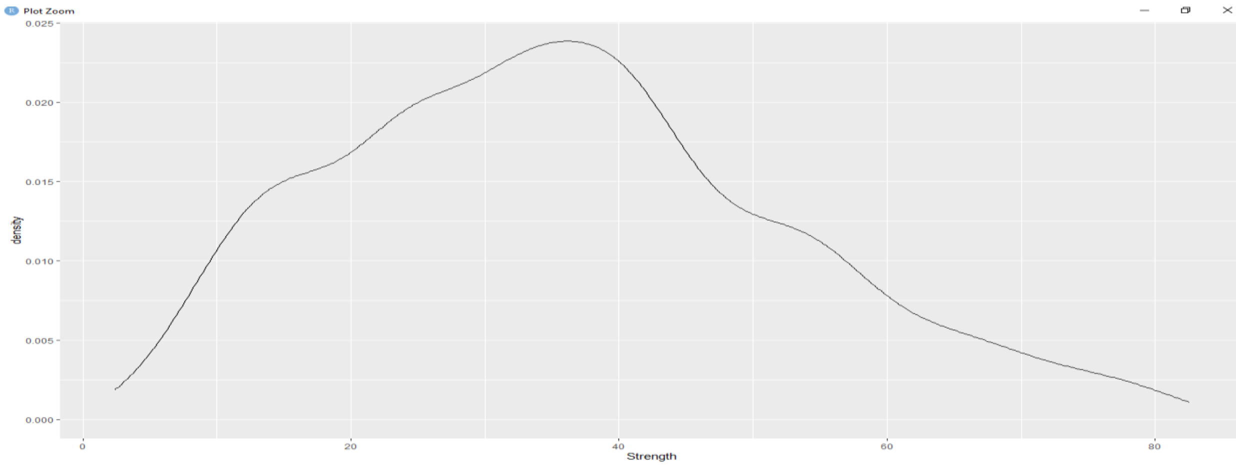
As a sum up 399 entries were missing. As the average missing value is greater than 1% values were imputed using mice.



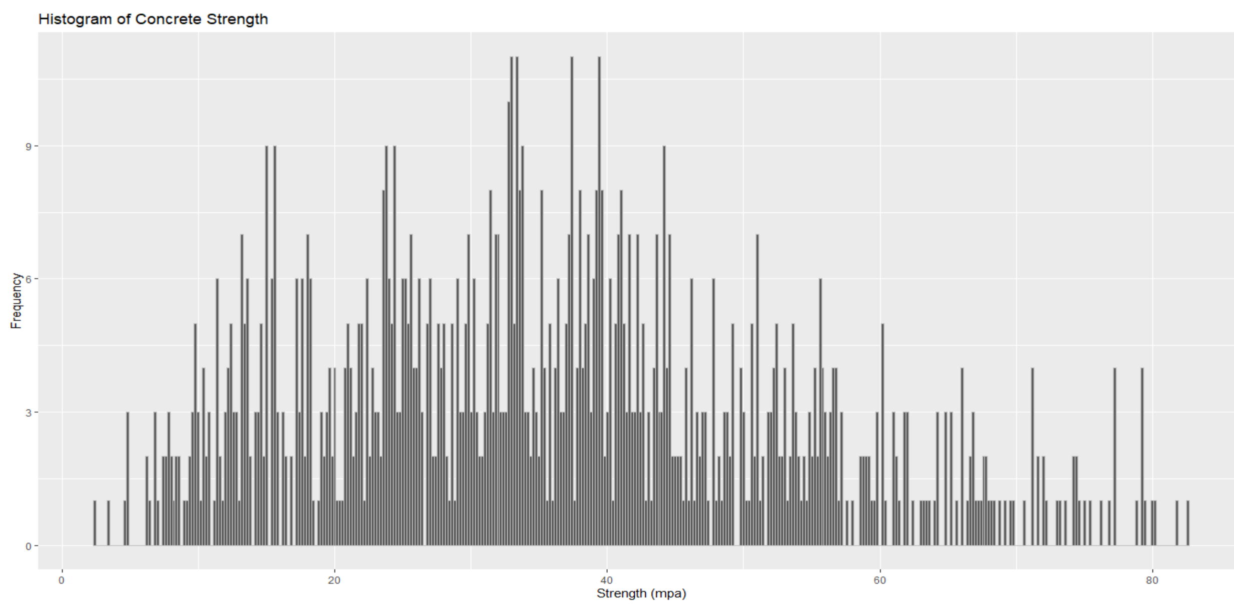
IQR of the strength column of the data is given below.

```
> IQR(concStrength)
[1] 22.425
```

For the density estimation or Kernel density plot, library ggplot2 was used. Visualization of the shape of the data was much more clear or better. The density plot was possible using geom_density () from the ggplot2 library.



To understand the data, histogram was also used as it categorizes a continuous variable into non overlapping intervals. For this too ggplot2 was used as ggplot2 uses a conceptual framework based on the grammar of graphics.

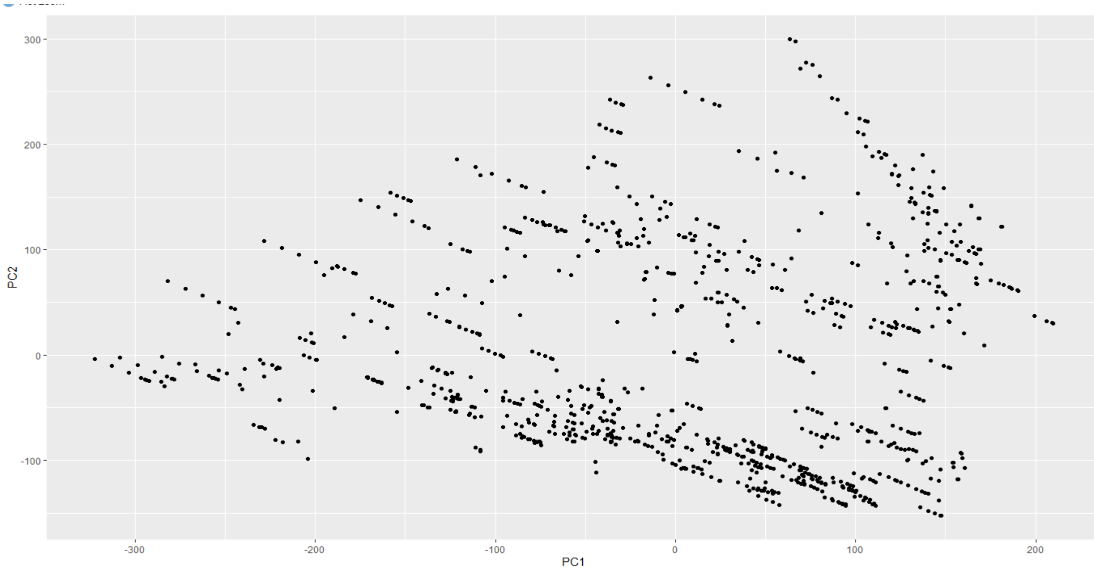


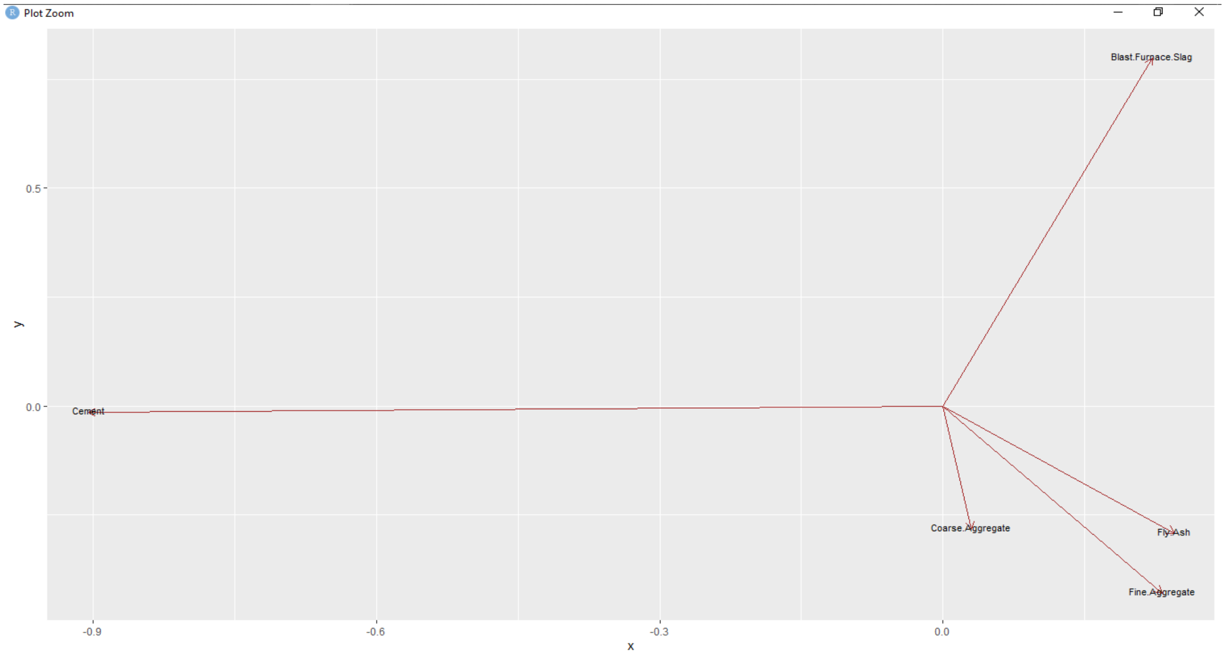
To know the frequency of the variables table () was used to get result below.

```
> table(strength_r)
strength_r
 2  3  5  6  7  8  9 10 11 12 13 14 15 16 17 18
 1  1  4  3  6 10  6 16 12 15 19 14 22 17 11 22
19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
 9 12 15 21 14 37 23 24 15 17 18 24 20 23 40 25
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
21 17 29 22 35 20 28 23 13 27 15 13 12 10 13  9
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66
17 15 11 13 11 19 12  2  9 10  6  7  3  5  6  6
67 68 69 70 71 72 73 74 75 76 77 79 80 82 83
 8  7  2  2  5  5  2  5  2  2  5  6  2  1  1
```

PCA is done by transforming a large set of variables into a smaller. To compute PCA prcomp () function was used. This function takes a matrix of data where the variables are the columns and rows are the sample of the matrix. PCA score is also shown in the graph below.

```
> summary(conc_pca)
Importance of components:
                            PC1     PC2     PC3     PC4
Standard deviation     113.5939 98.9562 85.5250 64.9248
Proportion of Variance   0.3261  0.2474  0.1848  0.1065
Cumulative Proportion    0.3261  0.5735  0.7583  0.8648
                            PC5      PC6      PC7
Standard deviation     62.03895 35.93628 10.63559
Proportion of Variance  0.09725  0.03263  0.00286
Cumulative Proportion   0.96207  0.99470  0.99756
                           PC8     PC9
Standard deviation     9.22097 3.40830
Proportion of Variance 0.00215 0.00029
Cumulative Proportion  0.99971 1.00000
```





Data Pre-processing

Missing Values

While examining the data, it showed the missing data were more than 1% which is why mice () has been used to impute the missing values. The is.na gave a Boolean result on missing entries.

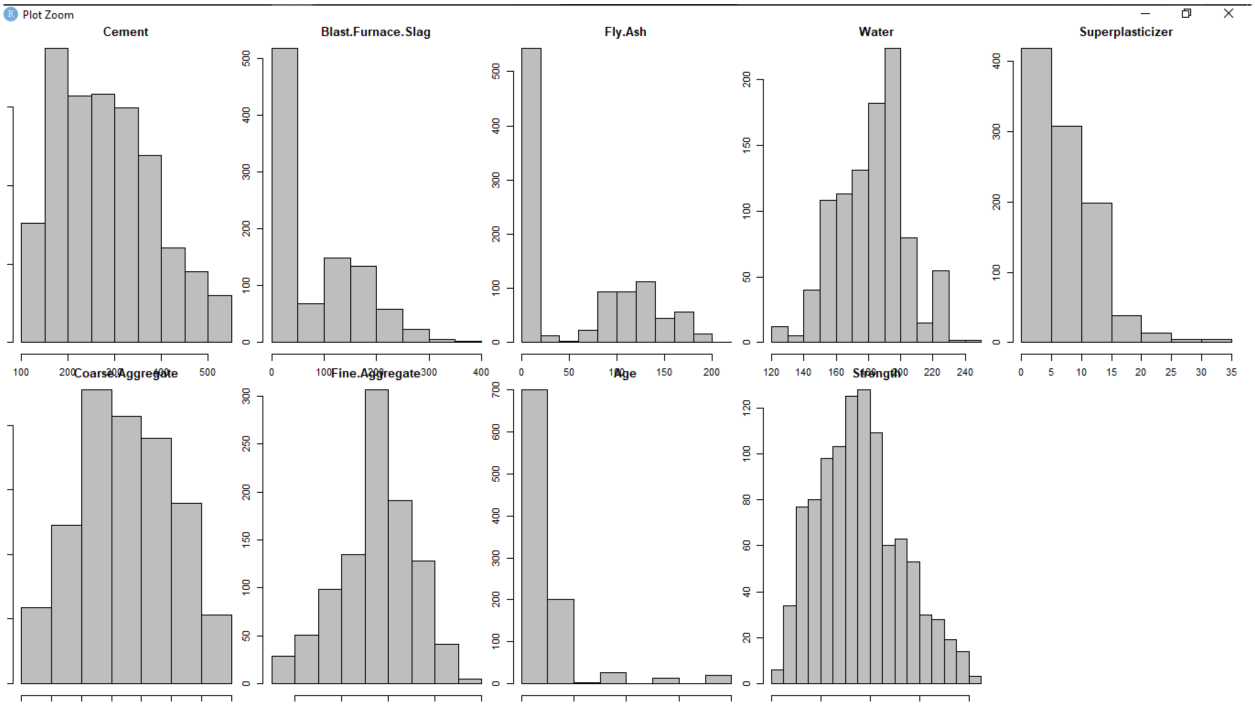
```
> is.na(combined)
      Cement Blast.Furnace.Slag Fly.Ash Water Superplasticizer Coarse.Aggregate
 [1,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
 [2,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
 [3,]  FALSE              FALSE   FALSE FALSE             TRUE            FALSE
 [4,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
 [5,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
 [6,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
 [7,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
 [8,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
 [9,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
[10,]  FALSE              FALSE   FALSE FALSE             TRUE            FALSE
[11,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
[12,]  FALSE              FALSE    TRUE FALSE            FALSE            FALSE
[13,]  FALSE               TRUE   FALSE FALSE            FALSE            FALSE
[14,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
[15,]   TRUE              FALSE   FALSE FALSE            FALSE            FALSE
[16,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
[17,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
[18,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
[19,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
[20,]  FALSE              FALSE   FALSE FALSE            FALSE             TRUE
[21,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
[22,]  FALSE              FALSE   FALSE FALSE             TRUE            FALSE
[23,]  FALSE              FALSE   FALSE FALSE            FALSE             TRUE
[24,]  FALSE               TRUE   FALSE FALSE            FALSE            FALSE
[25,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
[26,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
[27,]  FALSE              FALSE   FALSE FALSE            FALSE            FALSE
```

md.pattern showed the missing values according to its column, strength is the only column without missing values.

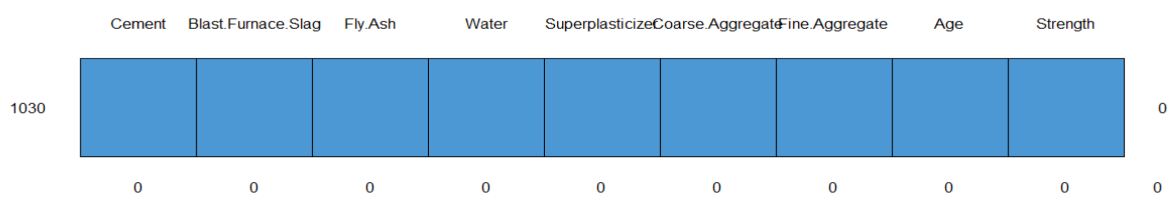
```
> md.pattern(combined)
    Strength Coarse.Aggregate Fly.Ash Fine.Aggregate Superplasticizer Cement
631        1                1       1              1                1      1
70         1                1       1              1                1      1
66         1                1       1              1                1      1
61         1                1       1              1                1      1
49         1                1       1              1                1      0
46         1                1       1              1                0      1
45         1                1       1              0                1      1
32         1                1       0              1                1      1
30         1                0       1              1                1      1
           0               30      32             45               46     49
    Water Age Blast.Furnace.Slag
631     1   1                  1   0
70      1   1                  0   1
66      1   0                  1   1
61      0   1                  1   1
49      1   1                  1   1
46      1   1                  1   1
45      1   1                  1   1
32      1   1                  1   1
30      1   1                  1   1
       61  66                 70 399
```

Mice package was installed and imported. Then the missing values where checked and stored in a variable, and its average was calculated using mean function. The data was checked if its normal or not in the histogram. As the distribution was not normal random forest is used as a method for imputation. Then mice function was called where m refers to the number of imputed data sets. With m = 1 only one dataset was generated. Maxit refers to the number of iterations taken while imputing the values and lastly method refers to the method used while imputing the values. Then complete function was called in order to extract the imputed data. Finally, md. pattern was called to check if missing data was imputed.

```
#missing values, greater than 1%
install.packages('mice')
library("mice")
is.na(combined)
missing_data <- apply(combined, 2, function(x) (sum(is.na(x)))/nrow(combined))
avgMissing <- mean(missing_data) * 100
md.pattern(combined)
par(mar=c(1,1,1,1))
lapply(names(combined), function(col){
  hist(combined[[col]], main=col, xlab=col, col = "gray", border = "black")
})
imputed_data <- mice(data = combined, m = 1, method="rf", maxit=10)
concreteImputed <- complete(imputed_data)
View(concreteImputed)
missingSum <- sum(is.na(concreteImputed))
missingSum
md.pattern(concreteImputed)

for(i in 1:9){
  boxplot(concreteImputed[i], col="blue", main="Box Plot", xlab=colnames(concreteImputed)[[i]])
}
```




```
> md.pattern(concreteImputed)
 /\     /\
{  `---'  }
{  O   O  }
==>  V <==  No need for mice. This data set is completely observed.
 \  \|/  /
  `-----'

     Cement Blast.Furnace.Slag Fly.Ash Water Superplasticizer Coarse.Aggregate Fine.Aggregate Age Strength
1030      1                  1       1     1                1                1              1   1        1 0
          0                  0       0     0                0                0              0   0        0 0
```



Outliers

The method used to figure out the outlier is by using the IQR. For this at first Q1, Q2 and IQR were calculated. With the help of IQR lower and upper whiskers were calculated. And finally using rbind and unique functions outliers were taken out and store in a variable and data without outliers were stored in a separate variable too.

```r
#outlier using IQR
outliersDF <- data.frame()
cols = ncol(concreteImputed)
for(i in 1:cols){
  Q1 <- quantile(concreteImputed[[i]], 0.25)
  Q3 <- quantile(concreteImputed[[i]], 0.75)
  IQR <- IQR(concreteImputed[[i]])

  lowwhisker <- Q1 - 1.5 * IQR
  upwhisker <- Q3 + 1.5 * IQR

  outliersDF <- rbind(outliers, concreteImputed[concreteImputed[[i]] < lowwhisker | concreteImputed[[i]] > upwhisker,
}
outliers <- unique(outliersDF)
outliers
ouliersNull <- concreteImputed[!(rownames(concreteImputed) %in% rownames(outliers)), ]
View(ouliersNull)
summary(ouliersNull)
```

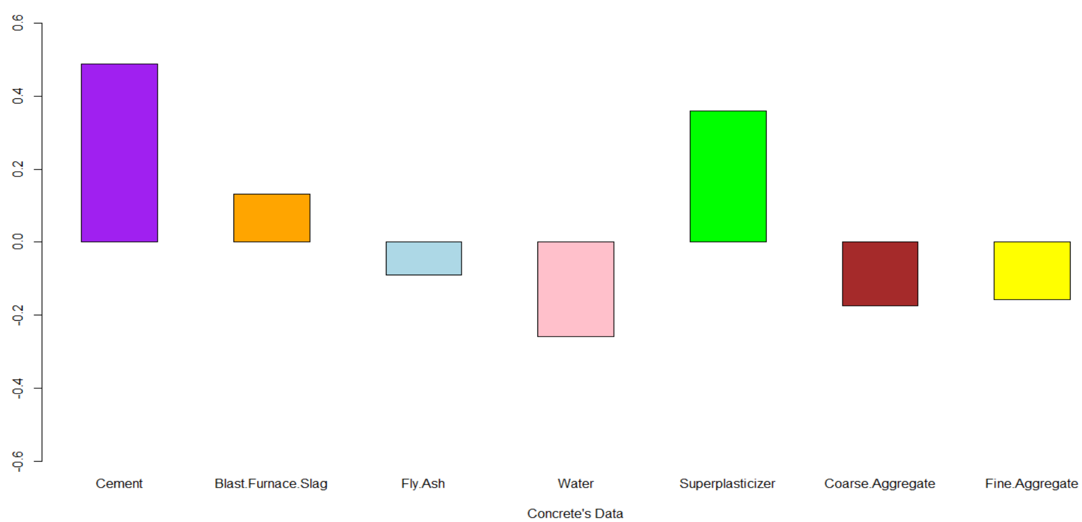
```
> outliers <- unique(outliersDF)
> outliers
    Cement Blast.Furnace.Slag Fly.Ash Water
1    540.0                0.0       0 162.0
109  323.7              282.8       0 183.8
266  315.0              137.0       0 145.0
774  389.9              189.0       0 145.9
    Superplasticizer Coarse.Aggregate Fine.Aggregate
1           2.500000         1040.000          676.0
109        10.300000          973.258          659.9
266         6.282927         1130.000          745.0
774        22.000000          944.700          755.8
         Age Strength
1   28.00000    79.99
109 56.00000    80.20
266 28.00000    81.75
774 45.96058    82.60
```

```
> View(ouliersNull)
> summary(ouliersNull)
     Cement      Blast.Furnace.Slag    Fly.Ash
 Min.   :102.0   Min.   :  0.00     Min.   :  0.00
 1st Qu.:200.0   1st Qu.:  0.00     1st Qu.:  0.00
 Median :281.0   Median : 47.50     Median :  0.00
 Mean   :283.1   Mean   : 75.35     Mean   : 55.31
 3rd Qu.:350.0   3rd Qu.:139.97     3rd Qu.:118.30
 Max.   :540.0   Max.   :359.40     Max.   :200.10
     Water       Superplasticizer Coarse.Aggregate
 Min.   :121.8   Min.   : 0.000   Min.   : 801.0
 1st Qu.:168.0   1st Qu.: 0.000   1st Qu.: 932.0
 Median :183.9   Median : 6.283   Median : 968.0
 Mean   :182.2   Mean   : 6.267   Mean   : 973.1
 3rd Qu.:192.0   3rd Qu.:10.075   3rd Qu.:1028.4
 Max.   :247.0   Max.   :32.200   Max.   :1145.0
 Fine.Aggregate       Age            Strength
 Min.   :594.0   Min.   :  1.00   Min.   : 2.33
 1st Qu.:739.0   1st Qu.: 14.00   1st Qu.:23.69
 Median :777.8   Median : 28.00   Median :34.27
 Mean   :774.6   Mean   : 45.99   Mean   :35.64
 3rd Qu.:821.0   3rd Qu.: 56.00   3rd Qu.:45.85
 Max.   :992.6   Max.   :365.00   Max.   :79.40
```

Multicollinearity

Using the variable without the outlier's, strength values were calculated and stored in a new variable, similarly a variable without strength was created using the same oulierNull variable, except for the age column every other column were used. With the cor function correlation with strength were calculated. The results were displayed with bar plot.

```
#Multicollinearity
varStrength <- ouliersNull[9]
nullStrength <- ouliersNull[c(1:7)]
corStrength <- cor(varStrength, nullStrength)
legend <- colnames(ouliersNull)[-8][-8]
par(mfrow=c(1,1))
barplot(corStrength, beside = TRUE, col=c("purple", "orange", "lightblue", "pink", "green", "brown", "yellow"), xlab='
```



Investigate Variables

First variance of a numeric variable was calculated and store in a variable, using the apply function. Using sum function total variance was calculated with the help of calculated single variance variable. Threshold was set in multiplication to total variance. Low variance was identified using the names function. Constant variables were checked using sapply and names function. Noise was checked using a new vector of low variance and constant variance variables. Finally, the noise was removed using the null outlier variable and names function.

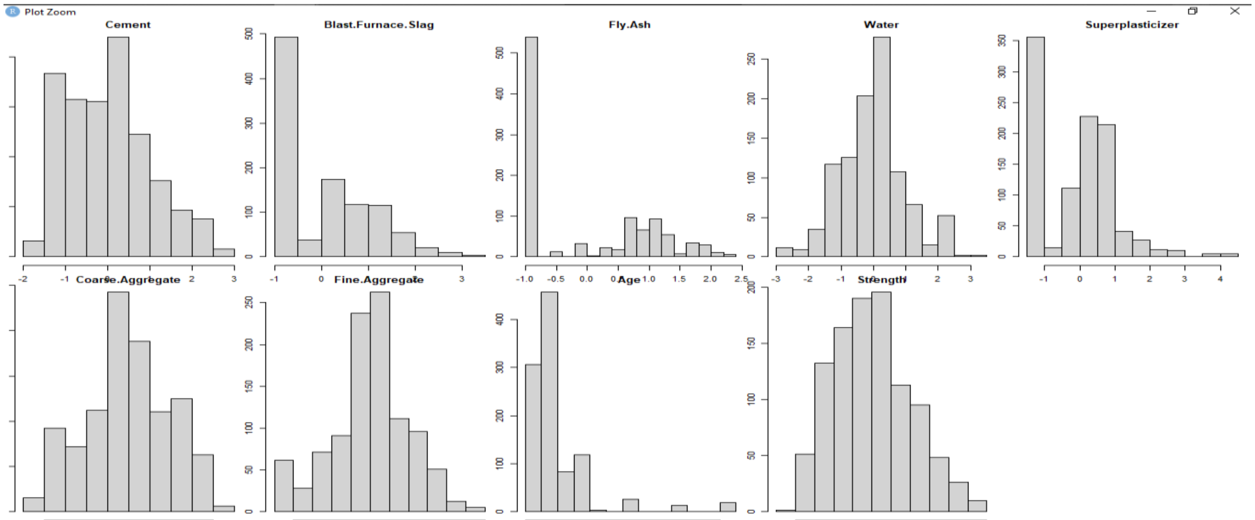
```
#investigate variables
variance <- apply(ouliersNull[, sapply(ouliersNull, is.numeric)], 2, var)
variance
varianceTotal <- sum(variance)
threshold <- 0.05 * varianceTotal
lowVariance <- names(variance[variance < threshold])
lowVariance
constVariance  <- sapply(ouliersNull, function(x) length(unique(x))) == 1
constVariance <- names(ouliersNull[constVariance])
noises <- c(lowVariance, constVariance)
noises
noiseNull <- ouliersNull[, !(names(ouliersNull) %in% noises)]
noiseNull
```

```
> lowVariance <- names(variance[variance < threshold])
> lowVariance
[1] "Water"            "Superplasticizer"
[3] "Strength"
> constVariance  <- sapply(ouliersNull, function(x) length(unique(x))) == 1
> constVariance <- names(ouliersNull[constVariance])
> noises <- c(lowVariance, constVariance)
> noises
[1] "Water"            "Superplasticizer"
[3] "Strength"
> noiseNull <- ouliersNull[, !(names(ouliersNull) %in% noises)]
> noiseNull
     Cement Blast.Furnace.Slag  Fly.Ash
2  540.0000            0.00000   0.0000
3  332.5000          142.50000   0.0000
4  198.6000          132.40000   0.0000
5  266.0000          114.00000   0.0000
6  266.0000          114.00000   0.0000
7  475.0000            0.00000   0.0000
8  198.6000          132.40000   0.0000
9  198.6000          132.40000   0.0000
10 427.5000           47.50000   0.0000
11 190.0000          190.00000   0.0000
12 304.0000           76.00000  55.0986
13 380.0000           75.64469   0.0000
14 139.6000          209.40000   0.0000
15 283.5065           38.00000   0.0000
16 380.0000           95.00000   0.0000
17 475.0000            0.00000   0.0000
18 139.6000          209.40000   0.0000
19 139.6000          209.40000   0.0000
20 139.6000          209.40000   0.0000
21 380.0000            0.00000   0.0000
22 380.0000            0.00000   0.0000
23 380.0000           95.00000   0.0000
24 427.5000           75.64469   0.0000
25 475.0000            0.00000   0.0000
```

Scaling

Scaling was performed using the scale function and null outlier variable. After scaling a loop of histograms were called.

```
#Scaling
numCols <- sapply(ouliersNull, is.numeric)
scaling <- scale(ouliersNull[, numCols], center = TRUE, scale = TRUE)
scaling
par(mar = c(1,1,1,1))
for(i in 1:ncol(scaling)){
  hist(scaling[,i], main=colnames(scaling)[i], xlab="Scaled Variable Values")
}
```



Data Modelling

The process of creating visual representation as a whole or as parts of data system in order to communicate connections between data point and structure. Main point of data modelling is to illustrate the used and stored data within the system. Usually models are created as per the business needs. This kind of data models are living documents which evolve along with the change in business nature. The best and easy way to visualize what data modelling is, is to think about it as a building architect plan, which consists of all the subsequent conceptual models.

Five predictive models developed are:

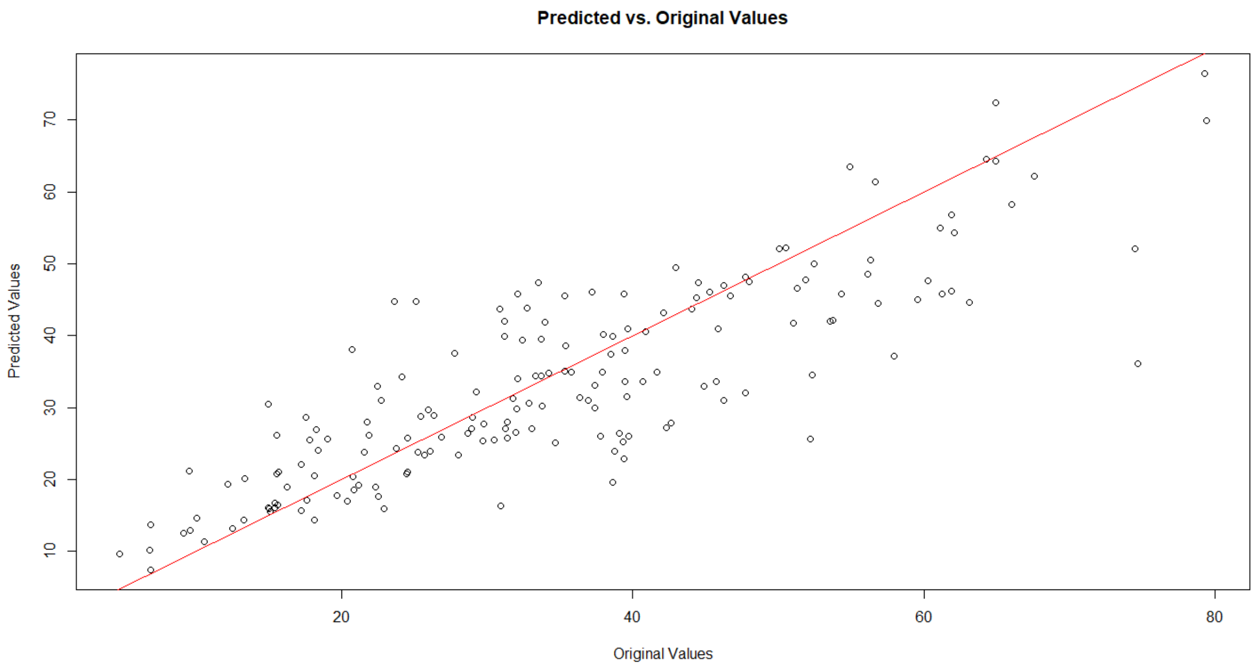
1. KNN (K Nearest Neighbour)
2. Random Forest
3. Decision Tree
4. Logistic Regression
5. Naïve Bayes

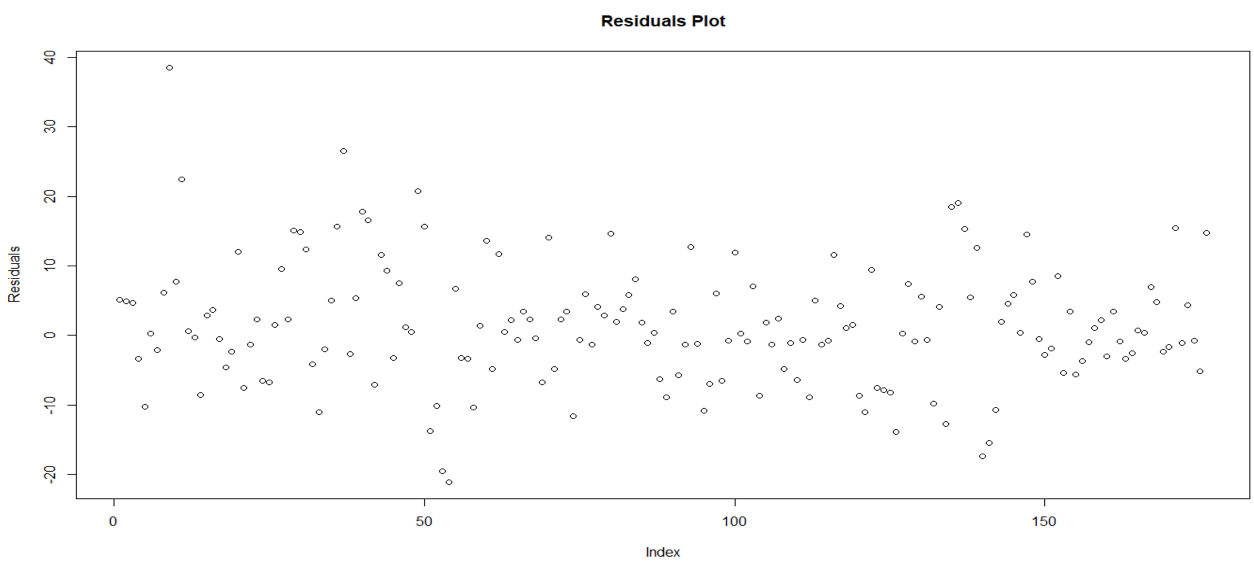
1. KNN

To predict values of new data points KNN uses feature similarity. Based on the clones of the point resemblance KNN assigns the value to any new point. This is a supervised machine learning algorithm, which classifies a point based on the labels of the K nearest neighbour point. (Anon., n.d.)

In KNN k is the number of nearest neighbors. Because of its ease of use and versatility, KNN is an excellent place to start for a variety of classification and regression tasks. Its effectiveness, however, is highly dependent on the selection of 'k' and the distance metric, so it might not be the best choice for high-dimensional or large-scale datasets.

```
> # Print the table of performance metrics
> print(performance_metrics)
     Metric     Value
1      RMSE 8.8508969
2 R-squared 0.6995156
> 
```



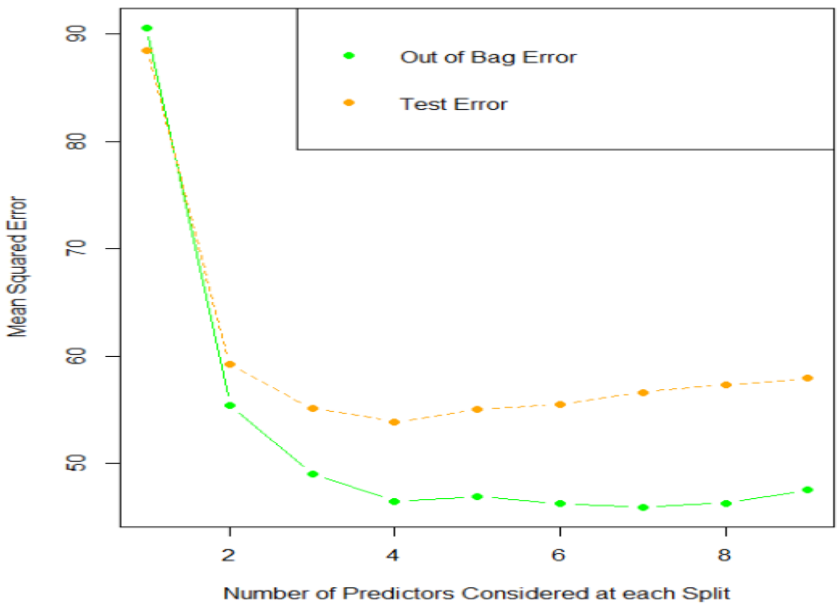


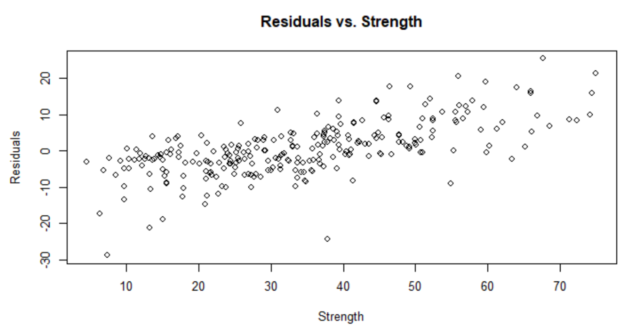
The output shows the RMSE, R2 using the KNN, below it is the predicted vs original values. And lastly residuals plot.

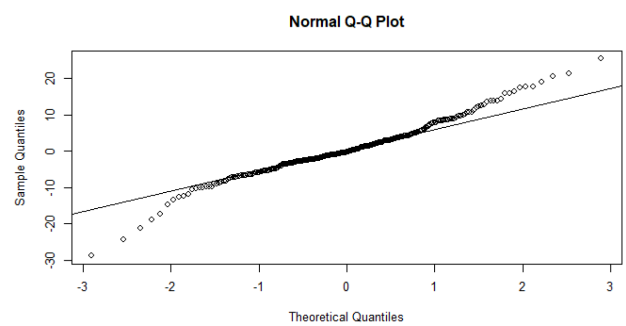
2. Random Forest

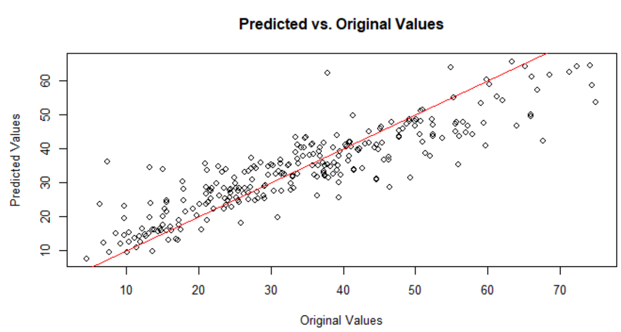
Random Forest is an algorithm that combines various decision trees output to get one single output. This algorithm can handle both classification and regression problems. This is a user friendly algorithm which is easily adaptive. This algorithm can handle very complex datasets and mitigate overfitting which makes it as one of the most important/ useful machine learning algorithm to perform various predictive tasks. (Anon., 2024) Random Forest is extensively employed in diverse fields like finance, healthcare, and marketing to accomplish tasks like disease prediction, fraud detection, and customer segmentation. Renowned for its resilience and efficacy across an extensive array of machine learning assignments, Random Forest is an algorithm that is both adaptable and strong.

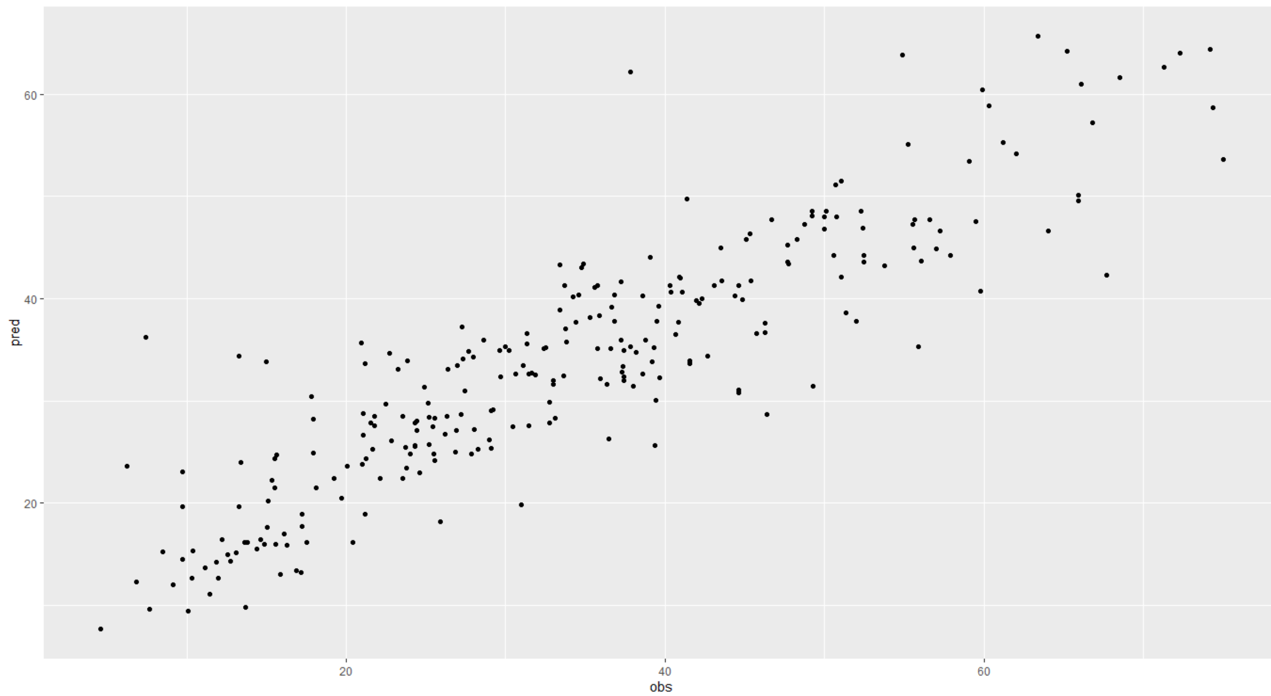
```
> c(metrics_rmse,metrics_r2,metrics_MAE)
[1] 7.4105508 0.8001011 5.4765657
> test.err
[1] 88.45859 59.25553 55.11838 53.85860 55.02865 55.50251 56.62094 57.33919 57.91219
> oob.err
[1] 90.56596 55.40832 49.01673 46.46401 46.90615 46.27541 45.90228 46.30180 47.53276
>
```











The output shows the numbers of predictor considered at each split, the predicted vs original values.

3. Decision Tree

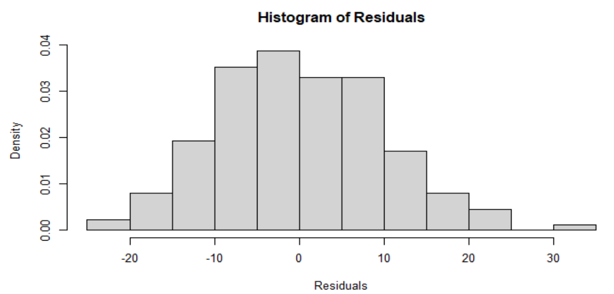
It is an algorithm from supervised learning can also be used in classification and regression problems like random forest. The hierarchical, tree structure in decision tree contains a root node, internal nodes and leaf nodes. This follows divide and conquer strategy by performing greedy search within a tree to find out the optimal split points. Pruning is usually employed, to reduce complexity and prevent overfitting that removes branches which split on features with low importance. (IBM, n.d.) The feature space is recursively divided into regions by a decision tree, with each region being linked to a particular class label in classification or predicted value in regression. Decision trees are a flexible algorithm that can be applied to a wide range of tasks, particularly those where interpretability is crucial. Nonetheless, their limitations can be lessened by employing regularized tree-based models, pruning, or ensembling (such as Random Forests).

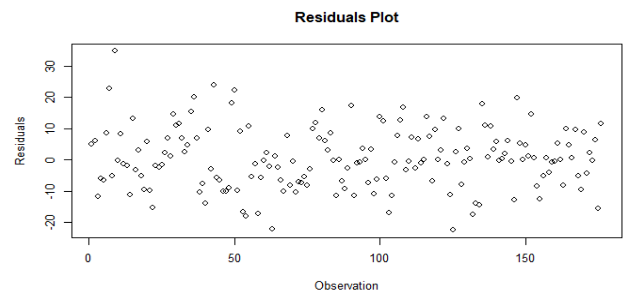
```
> pruned_rmse
[1] 9.72605
> # Calculate Mean Absolute Error (MAE)
> mae <- mean(abs(pruned_predict - test$Strength))
> print(paste("Mean Absolute Error (MAE) for pruned model:", mae))
[1] "Mean Absolute Error (MAE) for pruned model: 7.61223401822699"
```

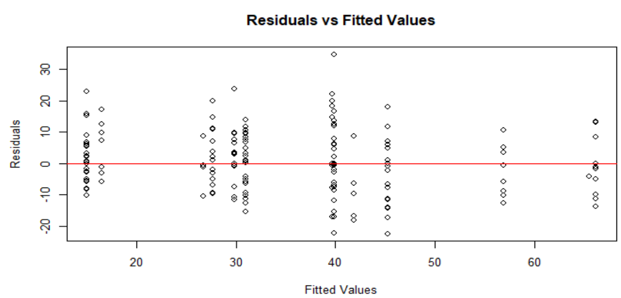
```
> print(paste("Root Mean Squared Error (RMSE):", rmse))
[1] "Root Mean Squared Error (RMSE): 9.72604962157089"
```

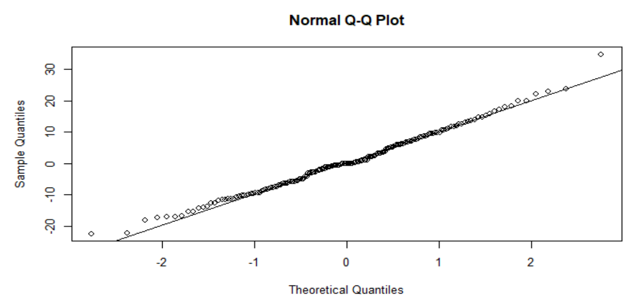
```
> rsquared <- 1 - SSE/SST
> print(paste("R-squared (R2) for pruned model:", rsquared))
[1] "R-squared (R2) for pruned model: 0.637155687034924"
```

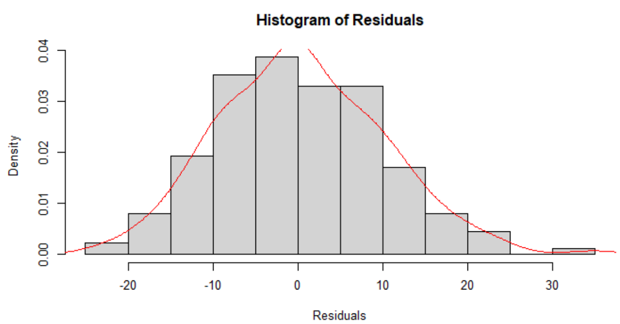
The output shows the MAE of the pruned model. Root mean squared error RMSE, R2 for the pruned model.

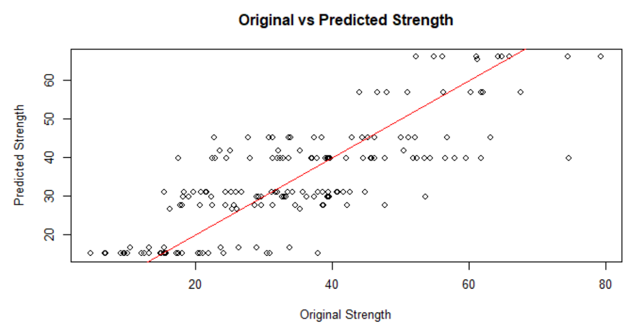

Histogram of Residuals
Density
Residuals

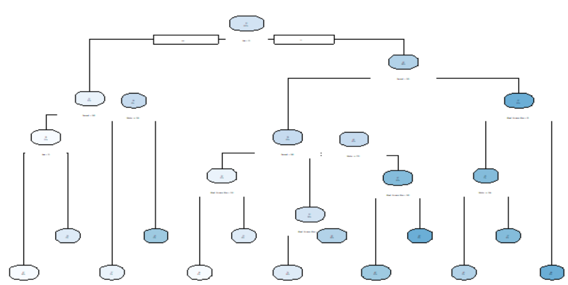

Residuals Plot
Residuals
Observation

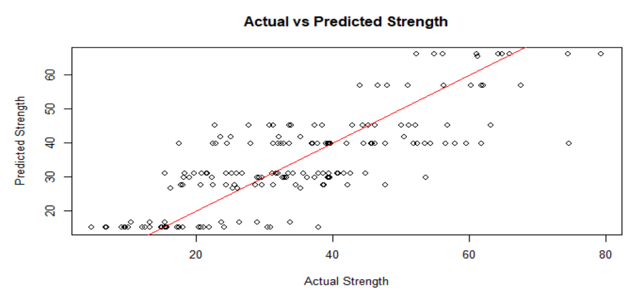

Residuals vs Fitted Values
Residuals
Fitted Values

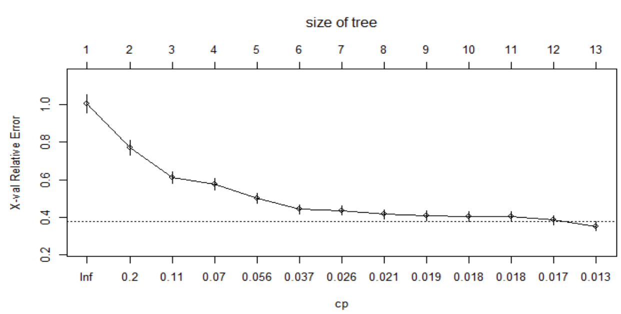

Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles

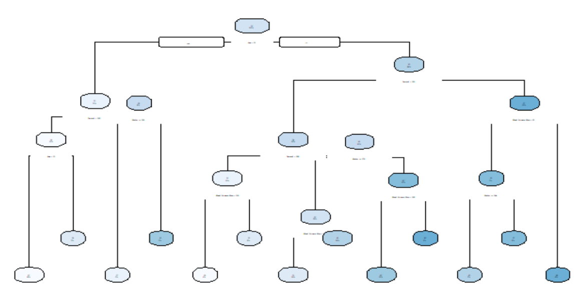

Histogram of Residuals
Density
Residuals


Original vs Predicted Strength
Predicted Strength
Original Strength




Actual vs Predicted Strength
Predicted Strength
Actual Strength


size of tree
X-val Relative Error
cp



4. Logistic Regression

Supervised learning algorithm that performs binary classification tasks by predicting the probability of an outcome, event or observation. This algorithm analyses the relation between one to many independent variables and classifies data into discrete classes. This modelling is highly used in predictive modelling, in which model usually finds the mathematical probability of whether the given instance belongs to the specific category or not. (Anon., n.d.) Because of its ease of use, interpretability, and efficacy in numerous real-world classification problems—particularly those with linear decision boundaries or when probabilistic interpretations are crucial—logistic regression is a popular algorithm.

```
> auc_result <- auc(roc_obj)
> print(paste("AUC:", auc_result))
[1] "AUC: 0.780117349970291"
> # Residual plot
```

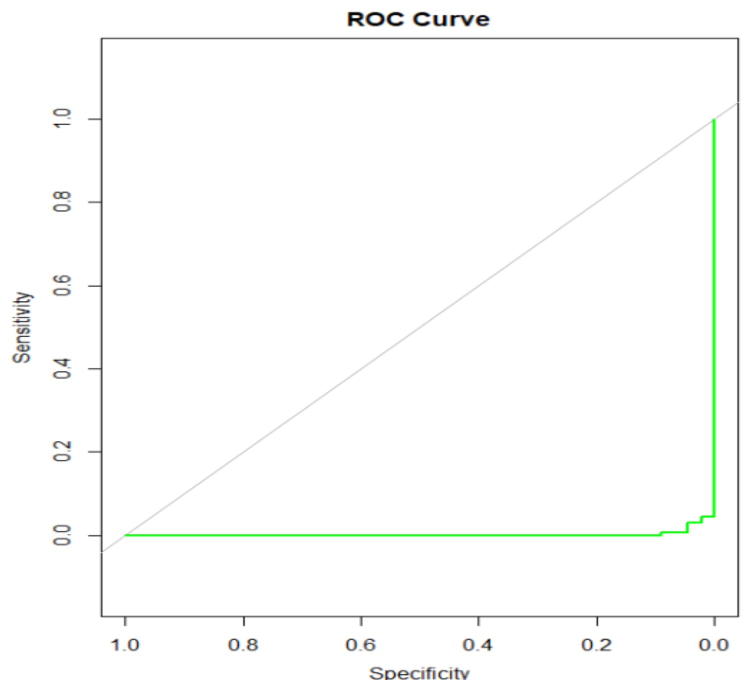
```
> predicted_classes <- ifelse(predicted > 0.5, 1, 0)
> actual_classes <- ifelse(test_data$Strength > q3, 1, 0)
> confusionMatrix(table(predicted_classes, actual_classes))
Confusion Matrix and Statistics

                 actual_classes
predicted_classes   0   1
                0 197  47
                1   7  19

               Accuracy : 0.8
                 95% CI : (0.7472, 0.846)
    No Information Rate : 0.7556
    P-Value [Acc > NIR] : 0.04939

                  Kappa : 0.3189

 Mcnemar's Test P-Value : 1.113e-07

            Sensitivity : 0.9657
            Specificity : 0.2879
         Pos Pred Value : 0.8074
         Neg Pred Value : 0.7308
             Prevalence : 0.7556
         Detection Rate : 0.7296
   Detection Prevalence : 0.9037
      Balanced Accuracy : 0.6268

       'Positive' Class : 0
```

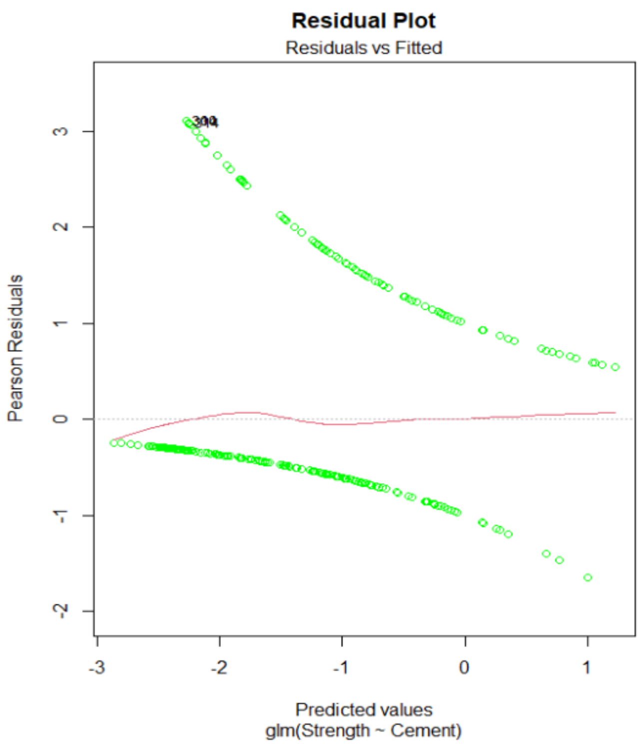
The output shows the AUC result, sensitivity, specificity, accuracy of the model.

```
> train_data <- cleaned_dataset[cleaned_dataset$isTrain == "yes", ]
> test_data <- cleaned_dataset[cleaned_dataset$isTrain == "no", ]
> library(mlbench)
> q1 <- quantile(train_data$Strength, 0.25)
> q3 <- quantile(train_data$Strength, 0.75)
> print(q3)
    75%
44.8675
> filtered_df <- train_data[train_data$Strength < q3 | train_data$Strength > q1, ]
> filtered_df$Strength <- ifelse(filtered_df$Strength > q3, 1, 0)
> datasets_for_regression = filtered_df
> datasets_for_regression$Strength = filtered_df$Strength
> data_to_be_trained = datasets_for_regression
> # Fit logistic regression model
> logit <- glm(Strength ~ Cement, family = binomial, data = datasets_for_regression)
> summary(logit)

Call:
glm(formula = Strength ~ Cement, family = binomial, data = datasets_for_regression)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.811647   0.347802  -10.96   <2e-16 ***
Cement       0.009325   0.001086    8.59   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 700.64  on 621  degrees of freedom
Residual deviance: 614.13  on 620  degrees of freedom
AIC: 618.13

Number of Fisher Scoring iterations: 4
```

```
> conf_matrix <- confusionMatrix(table(predicted_classes, actual_classes))
> sensitivity <- conf_matrix$byClass["Sensitivity"]
> specificity <- conf_matrix$byClass["Specificity"]
> print(paste("Sensitivity:", sensitivity))
[1] "Sensitivity: 0.965686274509804"
> print(paste("Specificity:", specificity))
[1] "Specificity: 0.287878787878788"
```

```
> # Accuracy
> accuracy <- sum(predicted_classes == actual_classes) / length(predicted_classes)
> print(paste("Accuracy:", accuracy))
[1] "Accuracy: 0.8"
>
```





The graph shows the ROC Curve, predicted values.

5. Naïve Bayes

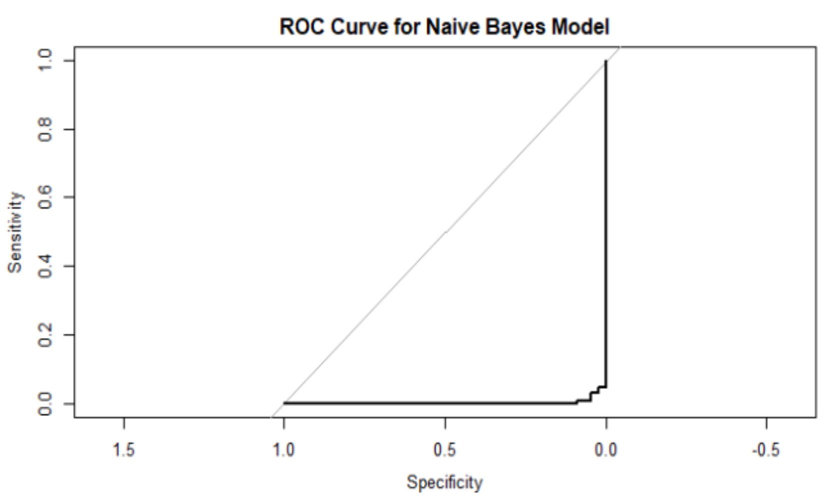
This algorithm utilizes Bayes' rule together. Naïve bayes provides a simple function / mechanism in order to use information in simple data to estimate the posterior probability. Which such estimation, the classification process gets easier. This algorithm is a form of Bayesian Network Classifier based on Bayes' rule. Naive Bayes is a popular option for many applications because, despite its simplicity and the "naive" assumption, it frequently performs surprisingly well, especially in text classification tasks. Text classification tasks like document categorization, sentiment analysis, and spam detection frequently employ Naive Bayes.

$$P(y \mid \mathbf{x}) = P(y)P(\mathbf{x} \mid y)/P(\mathbf{x})$$

```
> print(performance_table)
                 AUC  Accuracy Sensitivity Specificity Balanced_Accuracy
Accuracy 0.002050581 0.9717514   0.9924812   0.9090909         0.9507861
> |
```

The output shows the accuracy of the model, AUC, sensitivity, specificity, balanced accuracy.

```
> conf_matrix <- confusionMatrix(nb_predictions, test_dat
> # Print confusion matrix
> print(conf_matrix)
Confusion Matrix and Statistics

          Reference
Prediction Low High
      Low  132    4
      High   1   40

               Accuracy : 0.9718
                 95% CI : (0.9353, 0.9908)
    No Information Rate : 0.7514
    P-Value [Acc > NIR] : 6.358e-16

                  Kappa : 0.9226

 Mcnemar's Test P-Value : 0.3711

            Sensitivity : 0.9925
            Specificity : 0.9091
         Pos Pred Value : 0.9706
         Neg Pred Value : 0.9756
             Prevalence : 0.7514
         Detection Rate : 0.7458
   Detection Prevalence : 0.7684
      Balanced Accuracy : 0.9508

       'Positive' Class : Low

> |
```



The results for classification models aggregated in a table:

Model	Accuracy	Sensitivity	Specificity	FP	FN	Kappa	AUC
LR	0.8	0.9657	0.2879	47	7	0.3189	0.78011
Naïve Bayes	0.97	1	0.909	4	0	0.9378	0.004

The results for regression models aggregated in a table:

Model	R2	Adjust R2	MSE	RMSE	MAE
KNN	0.6459281	0.6359753	96.19994	9.8081572	0.5671265
Decision Tree	0.6935944	0.6946695	82.32912	8.944214	7.114847
Random Forest	0.8766517	0.8726000	35.0497000	5.8126508	4.1170068

Bibliography

A.Ashrafian, M. M.-B. T. O.-O., 2018. *. Prediction of compressive strength and ultrasonic pulse velocity of fiber reinforced concrete incorporating nano silica using heuristic regression methods.* s.l.:s.n.

A.kumar, H. N. M. K. A. O., 2022. *Compressive strength prediction of lightweight concrete: Machine Learning models.* s.l.:s.n.

B.Bharatkumar, R. B. D., 2001. *Mix proportioning of high performance concrete.* s.l.:s.n.

H.Ayat, Y. M. B., 2018. *Compressive strength prediction of limestone filler concrete using artificial neural networks.* s.l.:s.n.

H.Nguyen, T. T. H., 2021. *Efficient machine learning models for prediction of concrete strengths.* s.l.:s.n.

H.Salehi, R., 2018. *Emerging artificial intelligence methods in structural engineering.* s.l.:s.n.

H.Shi, B. X., 2009. *Influence of mineral admixtures on compressive strength, gas permeability and carbonation of high performance concrete.* s.l.:s.n.

J.Zhang, G. Y. F. B., 2019. *Modelling uniaxial compressive strength of lightweight self-compacting concrete using random forest regression.* s.l.:s.n.

M.F.M.Zain, S., 2009. *Multiple regression model for compressive strength prediction of high performance concrete.* s.l.:s.n.

M.Lessard, 0. P., 1993. *Testing high-strength concrete compressive strength.* s.l.:s.n.

M.S.Barkhordari, D. A. D., 2022. *Data-Driven Compressive Strength Prediction of Fly Ash Concrete Using Ensemble Learner Algorithms.* s.l.:s.n.

P.H.Bischoff, S., 1991. *Compressive behaviour of concrete at high strain rates.* s.l.:s.n.

S.Bhanja, B., 2002. *. Investigations on the compressive strength of silica fume concrete using statistical methods.* s.l.:s.n.

S.Chithra, S. K. F., 2016. *A comparative study on the compressive strength prediction models for high performance concrete contatinning nano silica and copper slag using regression analysis and artificial neural netwroks.* s.l.:s.n.

X.Zhu, 2011. *Strength prediction of high strength concrete using two nonlinear methods..* s.l.:s.n.

Z.Chen, X. L. S. X., 2021. *Design of a three-dimensional earth pressure device and its application in a tailings dam construction simulation experiment.* s.l.:s.n.

Anon., 2024. *analytics vidhya.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/
Anon., n.d. *codecademy.* [Online]
Available at: https://www.codecademy.com/learn/introduction-to-supervised-learning-skill-path-2024/modules/k-nearest-neighbors-skill-path/cheatsheet
Anon., n.d. *spice works.* [Online]
Available at: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/

IBM, n.d. [Online]
Available at: https://www.ibm.com/topics/decision-trees#:~:text=A%20decision%20tree%20is%20a,internal%20nodes%20and%20leaf%20nodes.