Student Name: Eoghan Hynes

Student ID Number: A00107408

Class Group: AL_KSWPT_R_5 MSc Software Eng

Class: Data Science 5

Project: R Assignment

Table of Contents

Concrete Compressive Strength Data Set Analysis

Introduction:

Title of Data Set: Concrete Compressive Strength [1].

Data Type: multivariate.

Abstract:

Concrete is the most important material in civil engineering [2]. The concrete compressive strength is a highly nonlinear function of age and ingredients. These ingredients include cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, and fine aggregate.

Data Set Characteristics:	Multivariate	**Number of Instances:**	1030	**Area:**	Physical
Attribute Characteristics:	Real	**Number of Attributes:**	9	**Date Donated**	2007-08-03
Associated Tasks:	Regression	**Missing Values?**	N/A	**Number of Web Hits:**	84178

Figure 1.1 Data Table

Sources:

Original Owner and Donor Prof. I-Cheng Yeh
Department of Information Management
Chung-Hua University,
Hsin Chu, Taiwan 30067, R.O.C.
e-mail:icyeh@chu.edu.tw
TEL:886-3-5186511

Date Donated: August 3, 2007

Data Characteristics:

The actual concrete compressive strength (MPa) for a given mixture under a specific age (days) was determined from laboratory tests. Data is in raw form (not scaled).

The compressive strength is calculated from the failure load divided by the cross-sectional area resisting the load and reported in units of pound-force per square inch (psi) in US customary units or megapascals (MPa) in SI units. [3]

Summary Statistics:

Number of instances (observations): 1030
Number of Attributes: 9
Attribute breakdown: 8 quantitative input variables, and 1 quantitative output variable
Missing Attribute Values: None

Attribute Information:

Given is the variable name, variable type, the measurement unit and a brief description. The concrete compressive strength is the regression problem. The order of this listing corresponds to the order of numerals along the rows of the database.

Name	Data Type	Measurement	Description
Cement (component 1)	quantitative	kg in a m3 mixture	Input Variable
Blast Furnace Slag (component 2)	quantitative	kg in a m3 mixture	Input Variable
Fly Ash (component 3)	quantitative	kg in a m3 mixture	Input Variable
Water (component 4)	quantitative	kg in a m3 mixture	Input Variable
Superplasticizer (component 5)	quantitative	kg in a m3 mixture	Input Variable
Coarse Aggregate (component 6)	quantitative	kg in a m3 mixture	Input Variable
Fine Aggregate (component 7)	quantitative	kg in a m3 mixture	Input Variable
Age	quantitative	Day (1~365)	Input Variable
Concrete compressive strength	quantitative	MPa	Output Variable

Figure 1.2 Table of Attributes

Data Analysis using Linear Regression in R:

Structure of the Concrete Data:

```
> str(ConcData)
'data.frame':   1030 obs. of  9 variables:
 $ Cement             : num  540 540 332 332 199 ...
 $ FurnaceSlag        : num  0 0 142 142 132 ...
 $ FlyAsh             : 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 ...
 $ CoarseAggregate    : num  1040 1055 932 932 978 ...
 $ FineAggregate      : num  676 676 594 594 826 ...
 $ Age                : int  28 28 270 365 360 90 365 28 28 28 ...
 $ CompressiveStrength: num  80 61.9 40.3 41 44.3 ...
```

Figure 4.3 Structure of the Data

Head of the Concrete Data:

```
> head(ConcData)
  Cement FurnaceSlag FlyAsh Water Superplasticizer CoarseAggregate FineAggregate Age CompressiveStrength
1  540.0         0.0      0   162              2.5          1040.0         676.0  28             79.99
2  540.0         0.0      0   162              2.5          1055.0         676.0  28             61.89
3  332.5       142.5      0   228              0.0           932.0         594.0 270             40.27
4  332.5       142.5      0   228              0.0           932.0         594.0 365             41.05
5  198.6       132.4      0   192              0.0           978.4         825.5 360             44.30
6  266.0       114.0      0   228              0.0           932.0         670.0  90             47.03
```

Figure 1.3 Head of the Data

Summary of the Concrete Data:

```
> summary(ConcData)
     Cement       FurnaceSlag        FlyAsh          Water      Superplasticizer CoarseAggregate  FineAggregate      Age        CompressiveStrength
 Min.   :102.0   Min.   :  0.0   Min.   :  0.00   Min.   :121.8   Min.   : 0.000   Min.   : 801.0   Min.   :594.0   Min.   :  1.00   Min.   : 2.33
 1st Qu.:192.4   1st Qu.:  0.0   1st Qu.:  0.00   1st Qu.:164.9   1st Qu.: 0.000   1st Qu.: 932.0   1st Qu.:731.0   1st Qu.:  7.00   1st Qu.:23.71
 Median :272.9   Median : 22.0   Median :  0.00   Median :185.0   Median : 6.400   Median : 968.0   Median :779.5   Median : 28.00   Median :34.45
 Mean   :281.2   Mean   : 73.9   Mean   : 54.19   Mean   :181.6   Mean   : 6.205   Mean   : 972.9   Mean   :773.6   Mean   : 45.66   Mean   :35.82
 3rd Qu.:350.0   3rd Qu.:142.9   3rd Qu.:118.30   3rd Qu.:192.0   3rd Qu.:10.200   3rd Qu.:1029.4   3rd Qu.:824.0   3rd Qu.: 56.00   3rd Qu.:46.13
 Max.   :540.0   Max.   :359.4   Max.   :200.10   Max.   :247.0   Max.   :32.200   Max.   :1145.0   Max.   :992.6   Max.   :365.00   Max.   :82.60
```

Figure 1.4 Summary of the Data

The following Scatter Plots show the Correlations of each Concrete component with the Compressive Strength of the Concrete.

1.) The strongest positive correlation with Compressive Strength in the data set is that with the amount of Cement in the mix.

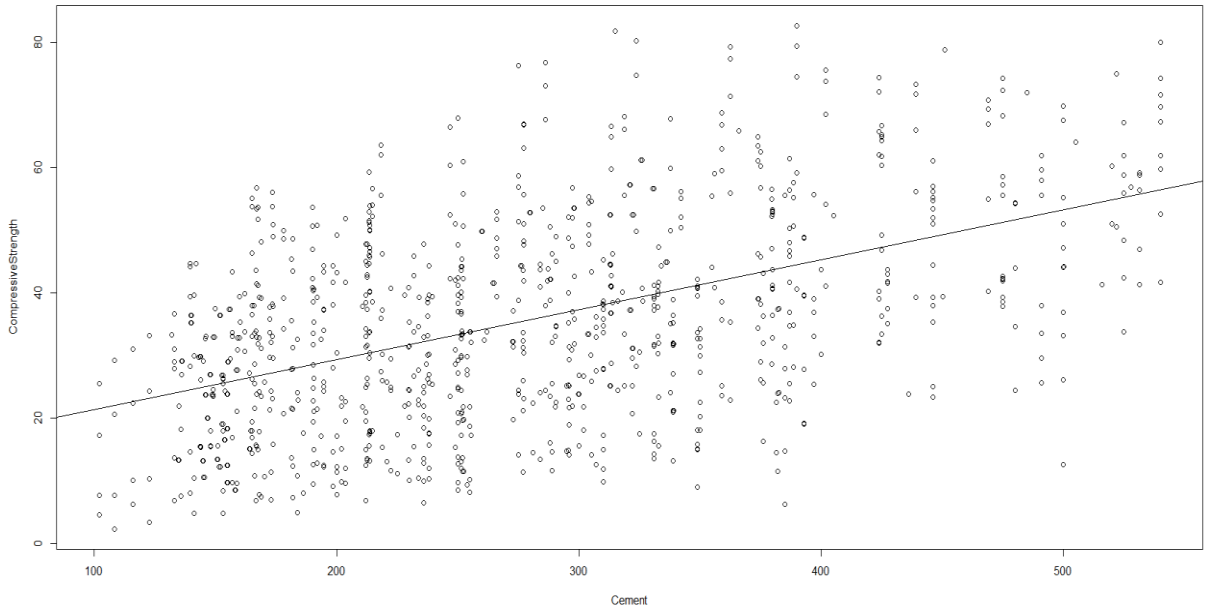


Figure 1.5 Compressive Strength ~ Cement Component

cor(CompressiveStrength, Cement) = `0.4978319`

2.) There is a weak but positive correlation between the Furnace Slag component of the concrete with Compressive Strength.

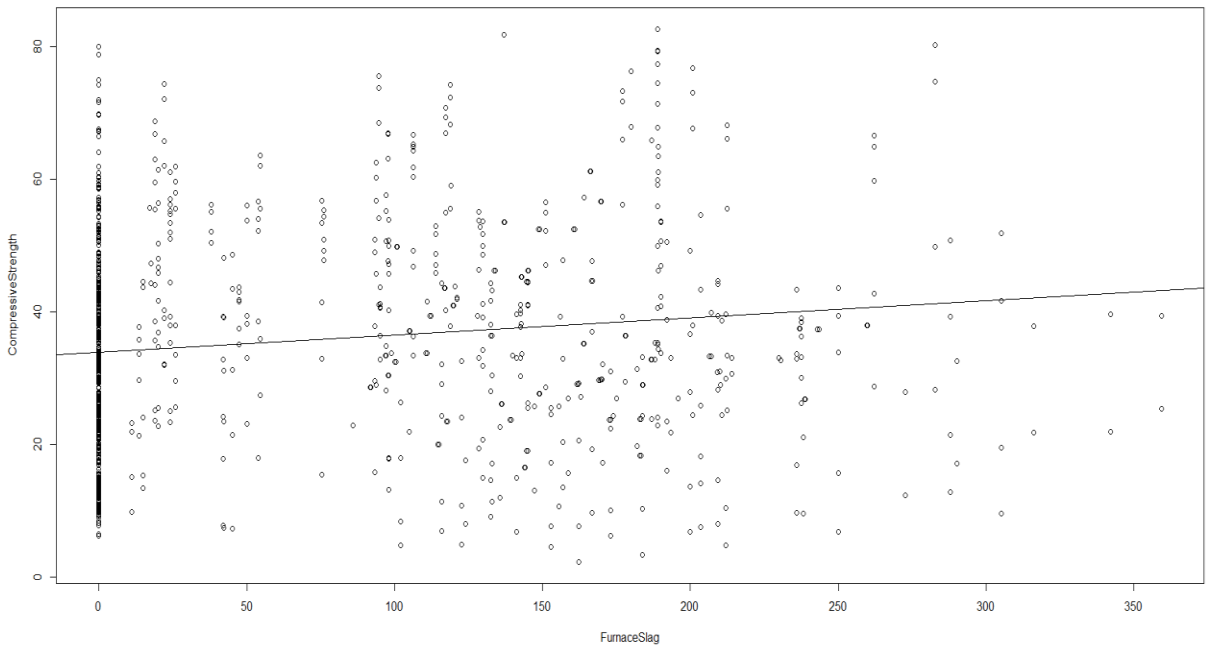


Figure 1.6 Compressive Strength ~ Furnace Slag Component

cor(CompressiveStrength, FurnaceSlag) = `0.1348293`

3.) The Fly Ash component of the Concrete mix negatively impacts on the Compressive Strength of the Concrete.

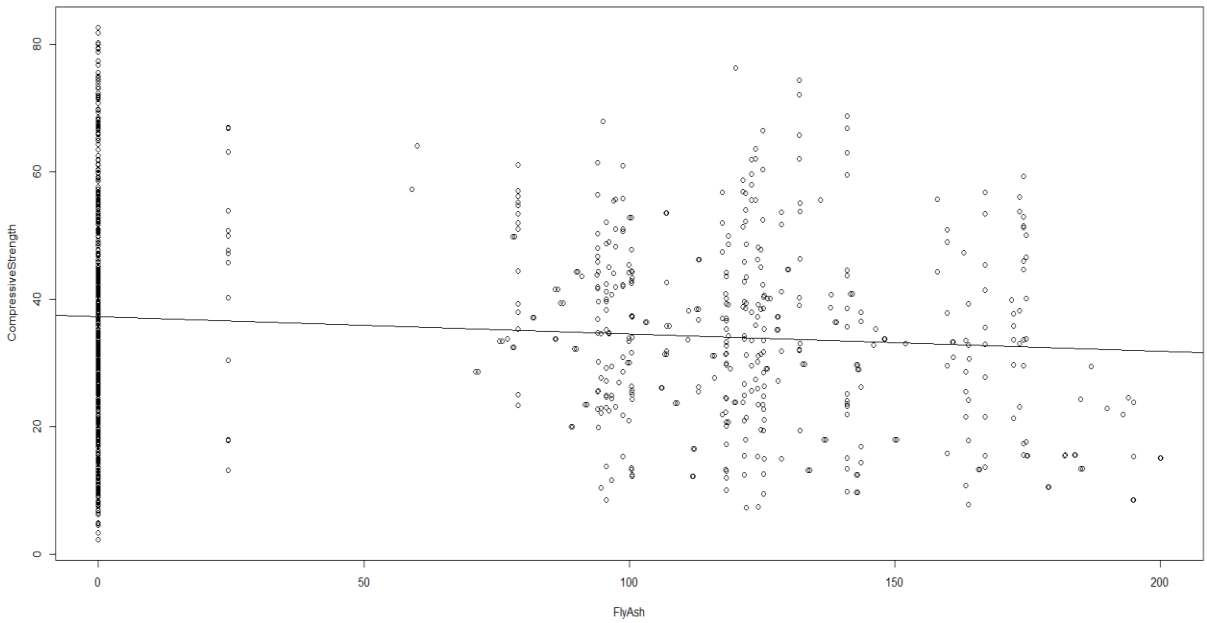


Figure 1.7 Compressive Strength ~ Fly Ash Component

cor(CompressiveStrength, FlyAsh) = -0.1057549

4.) The Water component negatively impacts on Compressive Strength.

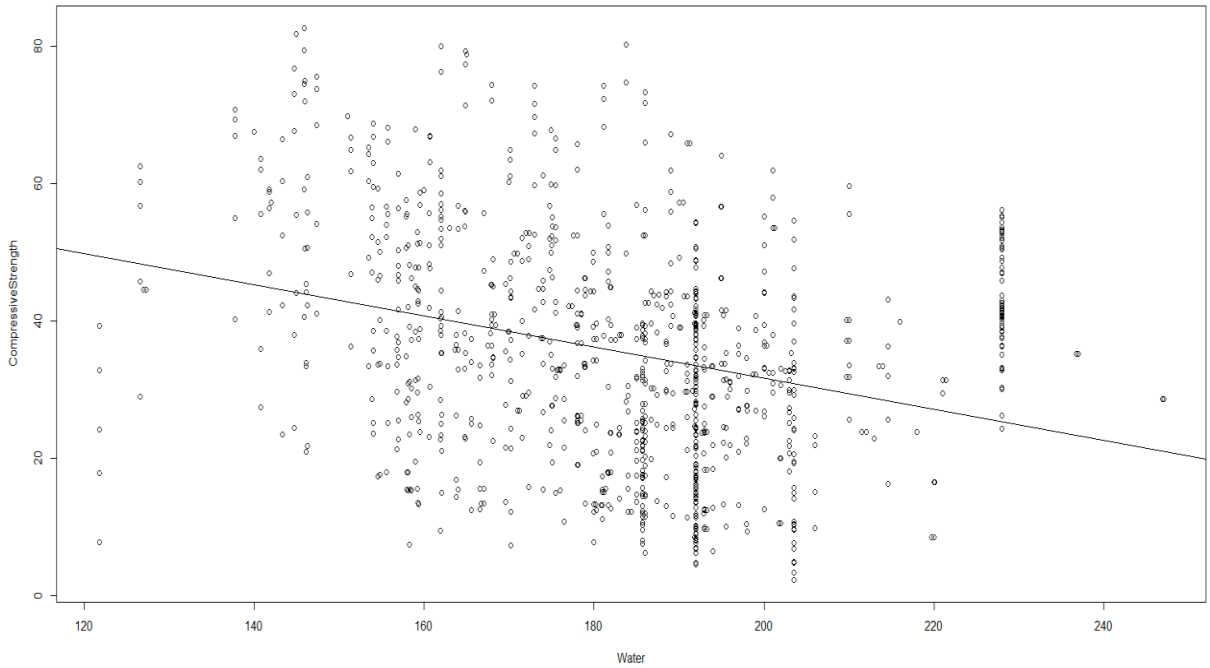


Figure 1.8 Compressive Strength ~ Water Component

cor(CompressiveStrength, Water) = -0.2896334

5.) Another strong positive correlation, this time between plasticizer in Kg/M3 in the mix and Compressive Strength.

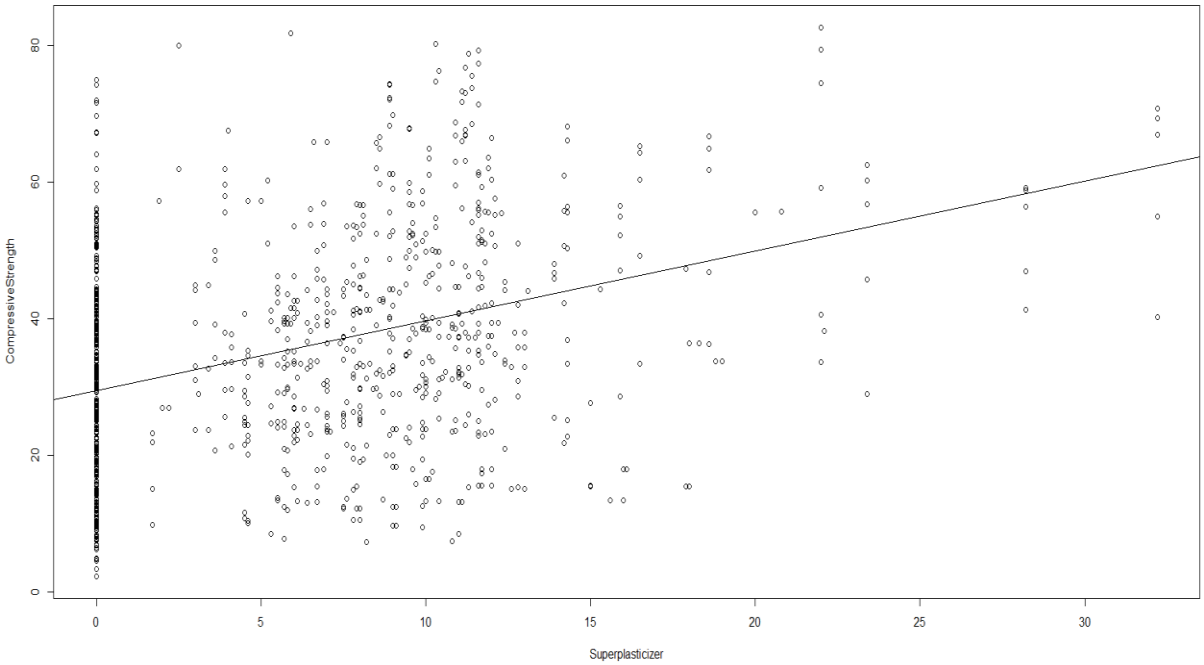


Figure 1.9 Compressive Strength ~ Plasticizer Component

cor(CompressiveStrength, Superplasticizer) = 0.3660788

6.) Too much Coarse Aggregate will weaken the mix.

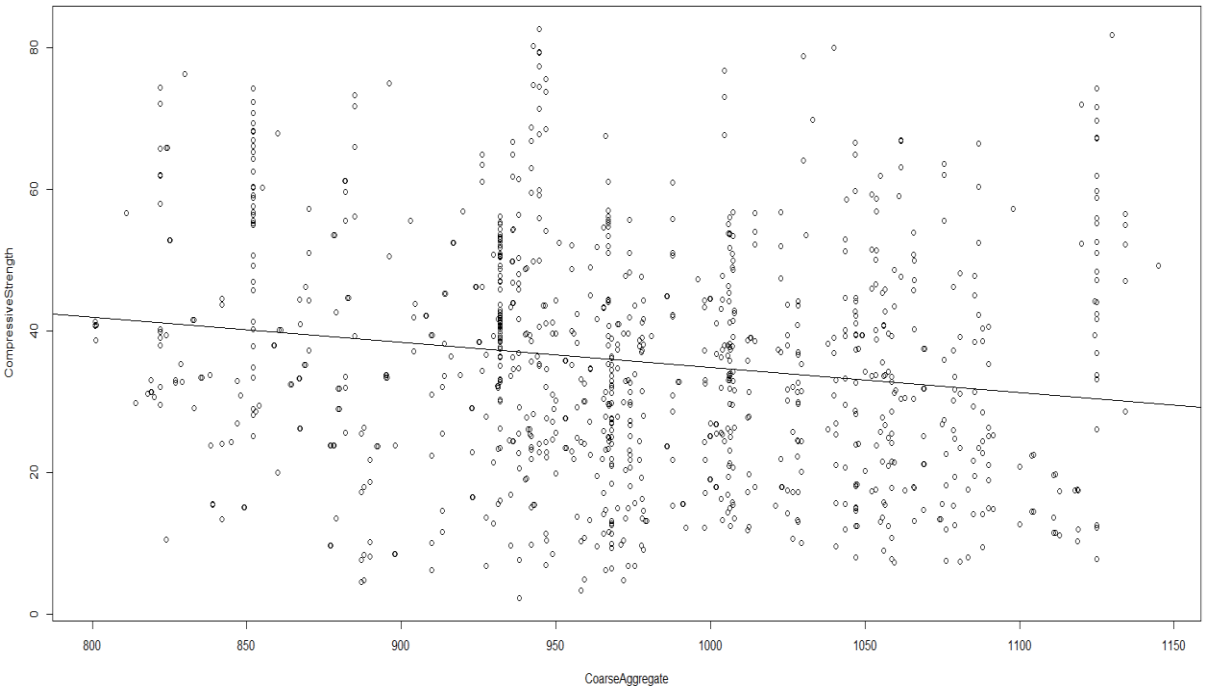


Figure1.10 Compressive Strength ~ Coarse Aggregate Component

cor(CompressiveStrength, CoarseAggregate) = -0.1649346

7.) Too much Fine Aggregate will weaken the mix.

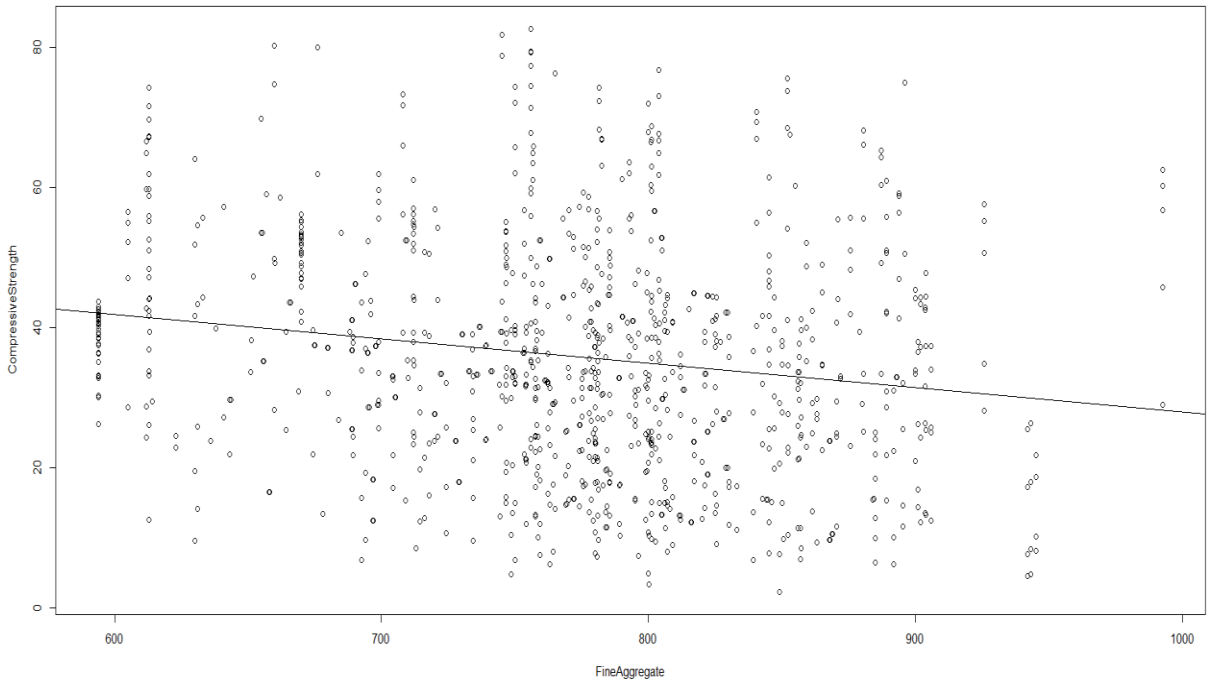


Figure 1.11 Compressive Strength ~ Fine Aggregate Component

cor(CompressiveStrength, FineAggregate) = -0.1672412

8.) Age has the third most positive correlation with Compressive Strength in the data set.

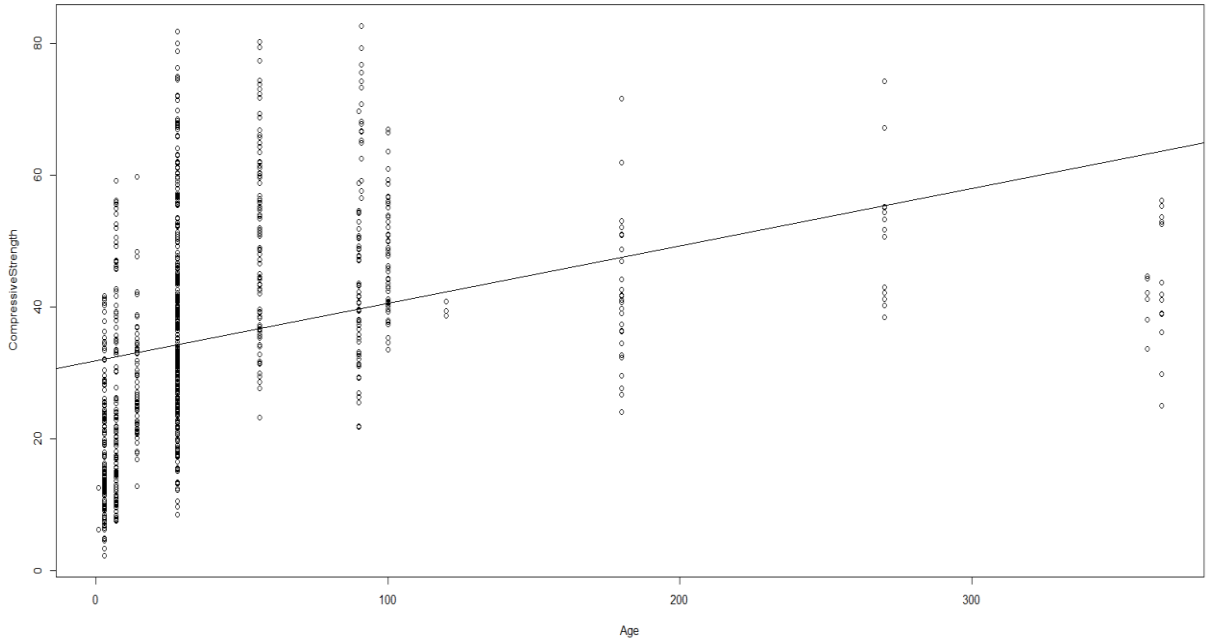


Figure 1.12 Compressive Strength ~ Age

cor(CompressiveStrength, Age) = 0.328873

I use R to create a Linear Model for all of the attributes in the data set which will be used to make predictions for the compressive strength of future data with different component values. However, a summary of the model shows that the Coarse and Fine Aggregate components are not statistically significant.

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-23.331214	26.585504	-0.878	0.380372	
Cement	0.119804	0.008489	14.113	< 2e-16	***
FurnaceSlag	0.103866	0.010136	10.247	< 2e-16	***
FlyAsh	0.087934	0.012583	6.988	5.02e-12	***
Water	-0.149918	0.040177	-3.731	0.000201	***
Superplasticizer	0.292225	0.093424	3.128	0.001810	**
CoarseAggregate	0.018086	0.009392	1.926	0.054425	.
FineAggregate	0.020190	0.010702	1.887	0.059491	.
Age	0.114222	0.005427	21.046	< 2e-16	***

Figure 1.13 Summary of Linear Model

The Scatter Plot Matrix below created using the GGally library in R summarises the correlations of the significant components only, with the Compressive Strength of the concrete. We can see the positive correlations with Cement, Furnace Slag, Superplasticizer and Age. We can also see the negative correlations with Fly Ash and Water. The strongest correlation to Compressive Strength is with Cement, followed by SuperPlasticizer.

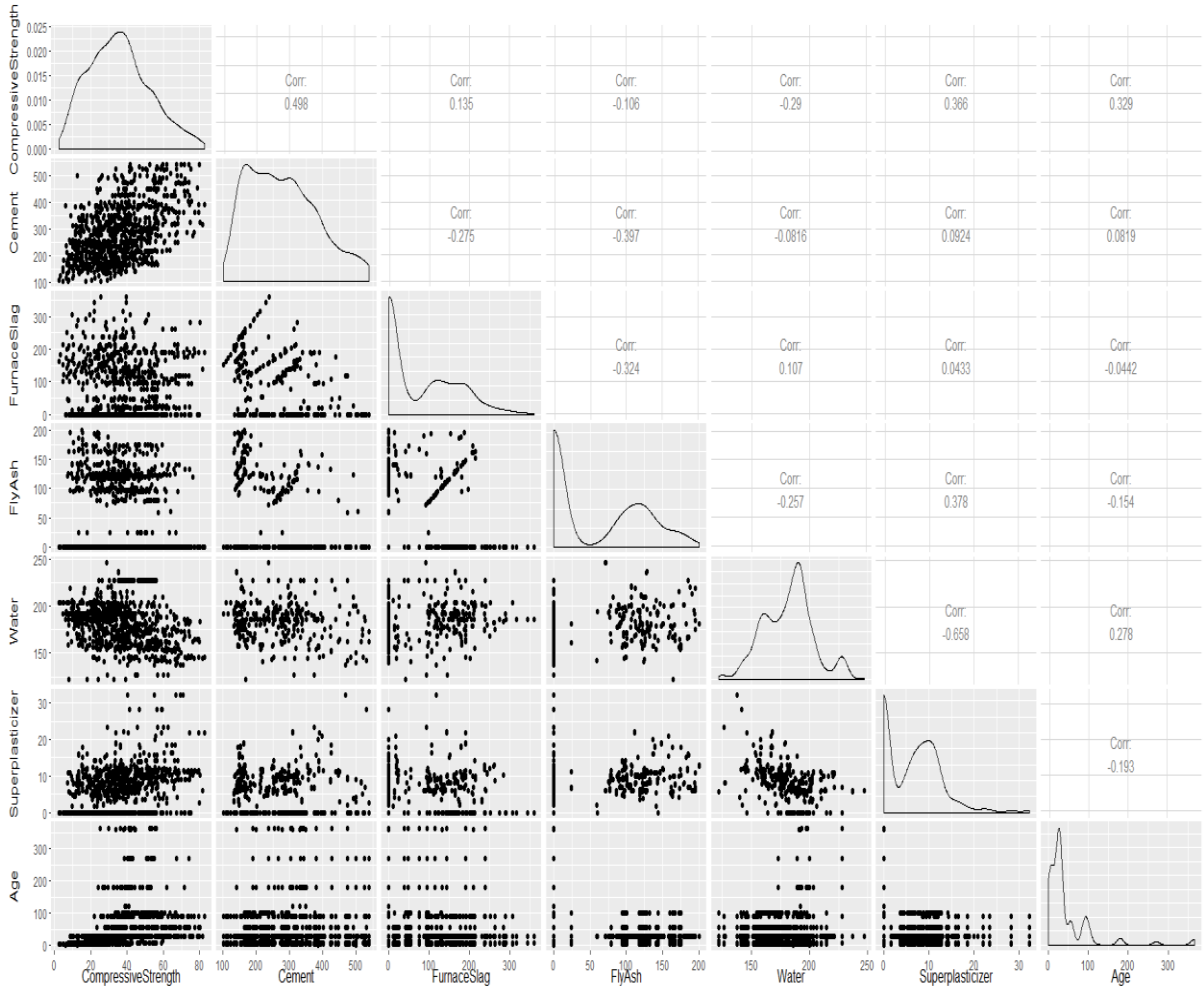


Figure 1.14 GGally Scatter Plot Matrix of significant components

The “ConcDatalm” linear model is created in R from the Concrete Data set for Compressive Strength dependant on the other significant attributes as follows:

```
ConcDatalm = lm(CompressiveStrength ~  Cement + FurnaceSlag + FlyAsh + Water
                                    + Superplasticizer + Age, data=ConcData)
```

With the model in place, I use it to predict Compressive Strength values for new data frames containing different values for the significant components. The first data frame takes the Mean values for the components from the data as seen in the Concrete Data Summary on page 4 of this report, where:

Cement=281.2
FurnaceSlag=22
FlyAsh=54.19
Water=181.6
Superplasticizer=6.205
Age=35.82

= Compressive Strength of 30.20989.

Which gives a predicted Compressive Strength of 30.20989 MPa. The next data frame takes the minimum values of all of the significant components. These values are:

Cement=102.0
FurnaceSlag=0
FlyAsh=0
Water=121.8
Superplasticizer=0
Age=1

= Compressive Strength of 13.295549.

Which give a predicted Compressive Strength of 13.29549 MPa. The next data frame I pass into the model takes the Maximum Attribute values for all significant components as seen in the data summary. These are:

Cement=540.0
FurnaceSlag=359.4
FlyAsh=200.10
Water=247.0
Superplasticizer=32.2
Age=365

= Compressive Strength of 126.0278.

Which give a predicted Compressive Strength of 126.0278 MPa.

Following this, I tweak the component values to arrive at a high prediction for Compressive Strength. I arrive at the following values:

Cement=1000 KG/M^3
FurnaceSlag=500 KG/M^3
FlyAsh=50 KG/M^3
Water=150 KG/M^3
Superplasticizer=50 KG/M^3
Age=365 Days

A typical cubic meter of concrete weighs 2,406.53 kg so almost half of this imaginary mix would be made up of cement. I keep the age of the new batch within the same maximum time frame of 365 days in line with the lab tested. These values give a predicted Compressive Strength of 201.802 MPa.

Lastly I use the model to predict a Compressive Strength taking the minimum values for all significant components in the data set but change the age of the sample from 35.82 days to 10 years. The age of the concrete has the third highest correlation to Compressive Strength in the data set. This is what makes the Compressive Strength of concrete a non-linear problem. The compressive strength of concrete appears to increase indefinitely as it cures. 'Ardnacrusha' power station, as one of the oldest concrete constructions in Europe, is examined regularly by concrete experts to examine this behaviour, which has held true for almost 100 years so far. These minimum values are:

Cement=102.0
FurnaceSlag=0
FlyAsh=0
Water=121.8
Superplasticizer=0
+
Age=3650

Which give a predicted Compressive Strength of 427.4274 MPa. The same batch of concrete left to cure for only 35.82 days had a Compressive Strength of only 30.20989 MPa. The mixture disregards Fine and Coarse Aggregates, and would not stand up well to the effects of frost.

Conclusion:

The Cement component of a Concrete mix has the greatest linear correlation to the Compressive Strength of the mix. However, the Compressive Strength of a given Concrete mix will improve over time as the Concrete cures.

The Compressive strength of an older Concrete mix may be greater than that of a younger mix containing more Cement. This is shown below in the final correlations.

Once again, the first data frame passed to the model contains the minimum attribute values taken from the data set, with an age of 631 days. The second data frame passed to the model uses the maximum values for the attributes with an age of 1 day. These concrete batches both give a Compressive Strength of 85 when rounded.

Cement=102.0	Cement=540.0
FurnaceSlag=0	FurnaceSlag=359.4
FlyAsh=0	FlyAsh=200.10
Water=121.8	Water=247.0
Superplasticizer=0	Superplasticizer=32.2
Age=631	Age=1
= 85MPa	=85MPa

To conclude, a Concrete mix taking all of the minimum component values of the significant attributes in the data set would have to cure for 631 days to have the equivalent Compressive Strength of a Concrete mix taking all of the maximum component values for all of the significant attributes in the data set with an absolute minimum test age of one day old.

The examples used to make predictions leave out Coarse and Fine aggregates as relatively insignificant in the mix. A mix without Coarse aggregates would be classed as Mortar rather than Concrete. Nevertheless, a mix of Cement=255 & Water=10 with 0 for all other values will roughly double in Compressive Strength every year.

Wilt Data Set Analysis

Introduction:

Title of Data Set: Wilt Data Set [4].

Data Type: Multivariate.

Abstract:

Data created using high-resolution Remote Sensing data set (Quickbird) satellite imagery. Small number of training samples of diseased trees, large number for other land cover. Testing data set from stratified random sample of image.

Data Set Characteristics:	Multivariate	**Number of Instances:**	4889	**Area:**	Life
Attribute Characteristics:	N/A	**Number of Attributes:**	6	**Date Donated**	2014-03-13
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	22976

Figure2.1Data Table

Source:

Brian Johnson;
Institute for Global Environmental Strategies;
2108-11 Kamiyamaguchi, Hayama, Kanagawa,240-0115 Japan;
Email: Johnson '@' iges.or.jp

Data Characteristics:

This data set contains some training and testing data from a remote sensing study by Johnson et al. (2013) that involved detecting diseased trees in Quickbird imagery. There are few training samples for the 'diseased trees' class (74) and many for 'other land cover' class (4265).

The data set consists of image segments, generated by segmenting the pansharpened image. The segments contain spectral information from the Quickbird multispectral image bands and texture information from the panchromatic (Pan) image band.

Files:

training.csv: training data set (4339 image segments)
testing.csv: testing data set (500 image segments)

Attribute Information:

class: 'w' (diseased trees), 'n' (all other land cover)
GLCM_Pan: GLCM mean texture (Pan band)
Mean_G: Mean green value
Mean_R: Mean red value
Mean_NIR: Mean NIR value
SD_Pan: Standard deviation (Pan band)

A summary of the class column from the Wilt data set shows a total of 74 Wilted classifications in the data set.

summary(class)

```
   n    w
4265   74
```

A quick check of the proportion of classifications shows a greater than 98% healthy classification and a less than 2% diseased classification in the data set.

prop.table(table(wilt$class))

```
        n         w
0.98294538 0.01705462
```

A Histogram of Mean_Red shows the frequency of the mean red values in the colour of the leaves in the data set, which for a particular time of year will indicate if the tree is diseased or not. A higher value of mean red in a leaf's colour is indicative of disease, ie > 125. The Mean Red attribute is used heavily in the Tree Map in figure x.x, which can be seen in the Tree Map summary in figure x.x. The histogram below reflects the correlation between the small number of w classifications in the data set with the red colour in the leaves of the data set ie, the number of samples with a mean red value above 200 is very small (around 500 samples).

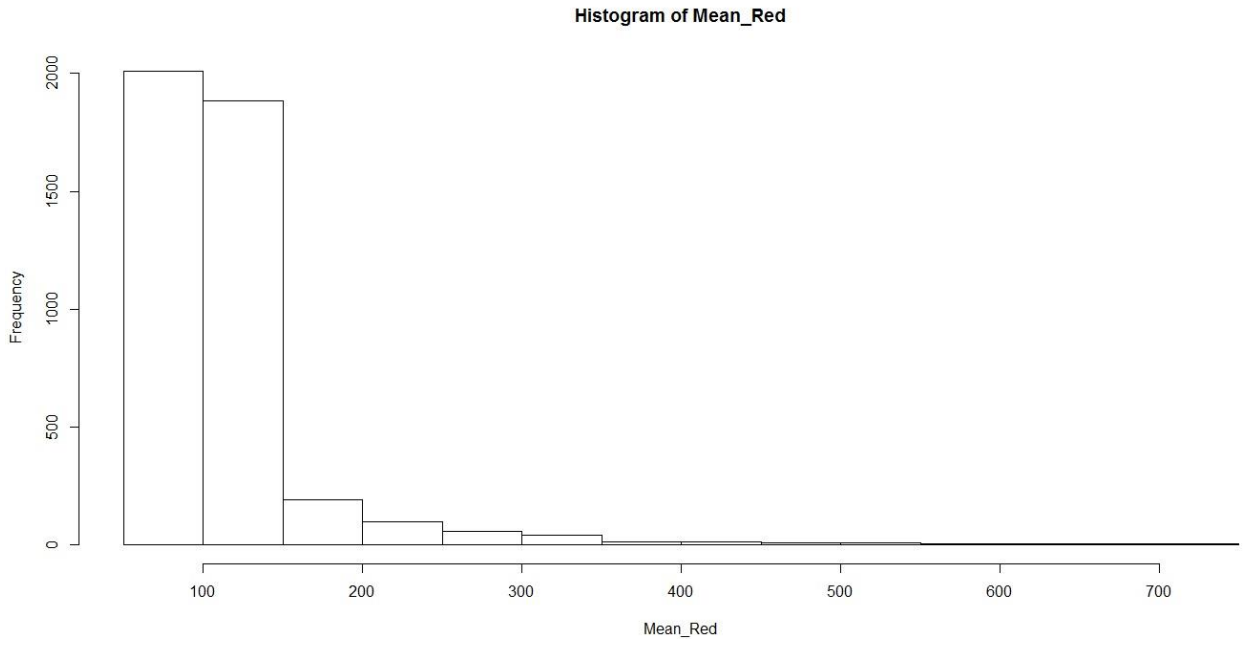


Figure 2.2 Histogram of mean red

A histogram of the GLCM_pan (Grey-Level Co-occurrence Matrix of the pan chromatic satellite imagery) shows that the majority of the leaves in the data set have a GLCM value in the range of 110 to 140. Therefore, considering the small number of w classifications in the data set, it seems that a GLCM_pan value of less than about 200 & greater than 150 is probably indicative of a diseased tree.

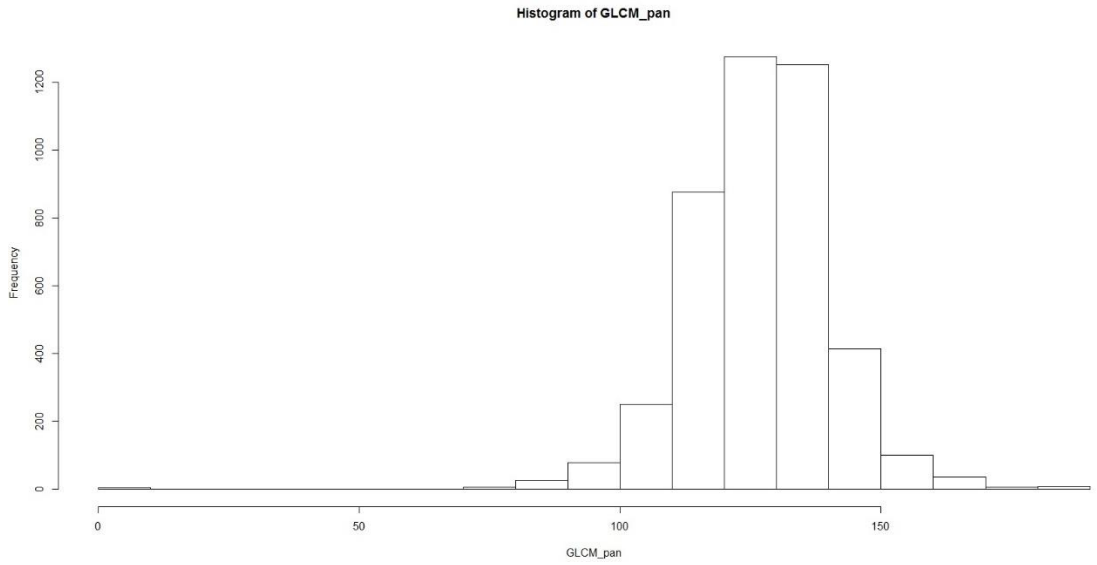


Figure 2.3 Histogram of GLCM_pan

A histogram of the Mean_NIR (Near Infra-Red) reflects the majority of trees in the data set, which are healthy, with a NIR signature of between 200 and 800. This range of NIR values must reflect healthy leaf infra-red values. Presumably the healthy leaves emit this infra-red signature as they photosynthesise. It follows then that lower mean NIR values are indicative of diseased (likely dead) leaves and the higher NIR are from decaying leaves, hence diseased trees.

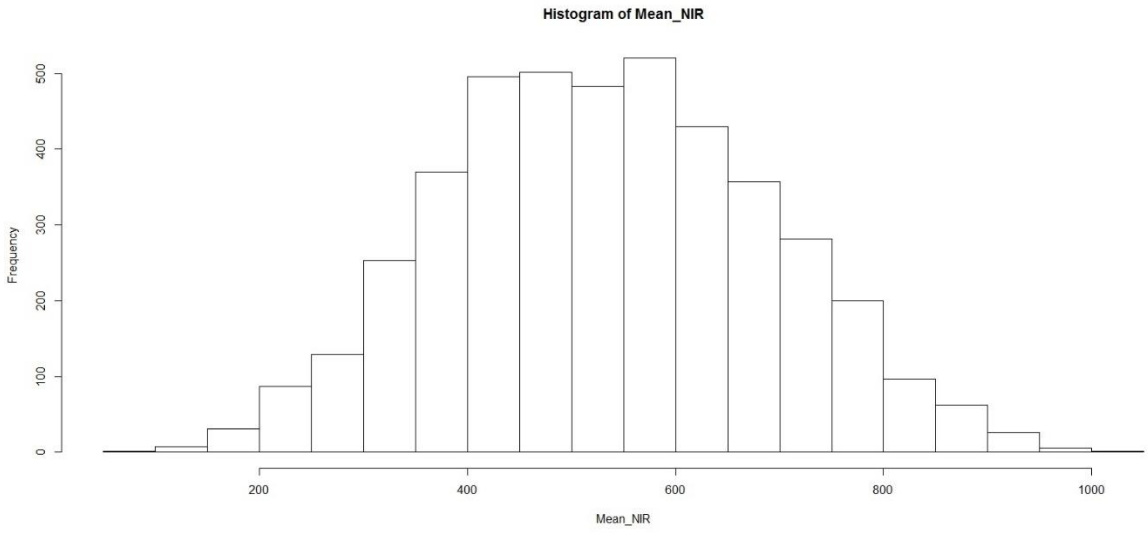


Figure 2.5 Histogram of Mean_NIR

Wilt Prediction using a Decision Tree

I use the C5.0 algorithm in R to create a Tree Map decision model for the wilt data, dependent on the class attribute which is either w = wilt or n = not wilt. The resulting tree map is seen here:

```
WiltTreeModel <- C5.0(class ~ ., data = wilt)
```

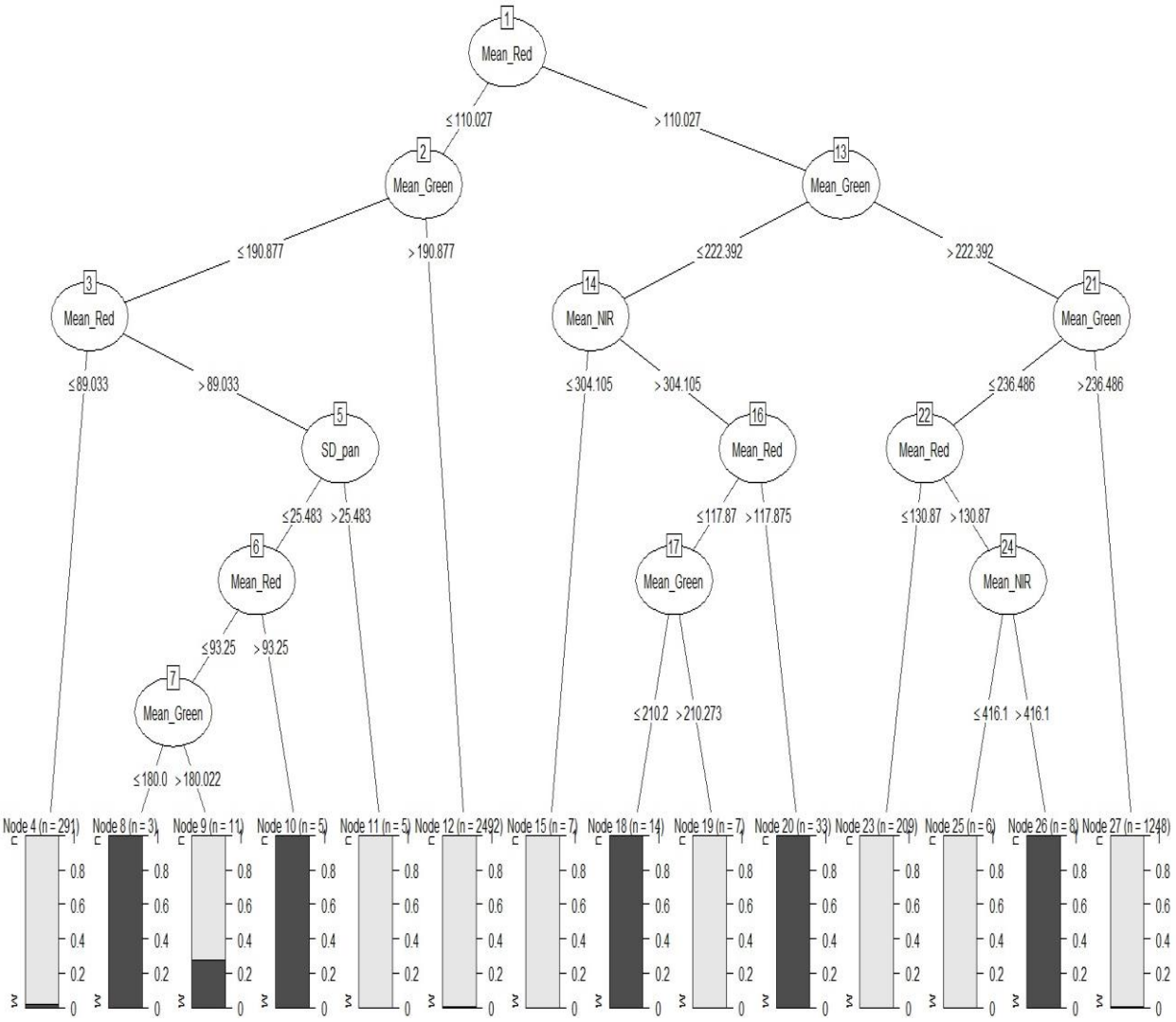


Figure 2.6 Wilt Decision Tree

A summary of the Wilt Tree Model is seen below:

```
C5.0 [Release 2.07 GPL Edition]         Wed Nov 02 16:35:29
-------------------------------

Class specified by attribute `outcome'

Read 4339 cases (6 attributes) from undefined.data

Decision tree:

Mean_Red > 110.0274:
:...Mean_Green <= 222.3922:
:   :...Mean_NIR <= 304.1053: n (7)
:   :   Mean_NIR > 304.1053:
:   :   :...Mean_Red > 117.875: w (33)
:   :       Mean_Red <= 117.875:
:   :       :...Mean_Green <= 210.2727: w (14)
:   :           Mean_Green > 210.2727: n (7)
:   Mean_Green > 222.3922:
:   :...Mean_Green > 236.4861: n (1247)
:       Mean_Green <= 236.4861:
:       :...Mean_Red <= 130.8696: n (209)
:           Mean_Red > 130.8696:
:           :...Mean_NIR <= 416.1: n (6)
:               Mean_NIR > 416.1: w (9)
Mean_Red <= 110.0274:
:...Mean_Green > 190.8775: n (2491/1)
    Mean_Green <= 190.8775:
    :...Mean_Red <= 89.03297: n (291/5)
        Mean_Red > 89.03297:
        :...SD_pan > 25.48325: n (5)
            SD_pan <= 25.48325:
            :...Mean_Red > 93.25: w (6)
                Mean_Red <= 93.25:
                :...Mean_Green <= 180.0222: w (3)
                    Mean_Green > 180.0222: n (11/3)


Evaluation on training data (4339 cases):

           Decision Tree
         ----------------
         Size      Errors

           14    9( 0.2%)   <<


          (a)   (b)    <-classified as
         ----  ----
         4265          (a): class n
            9    65    (b): class w


        Attribute usage:

        100.00% Mean_Green
        100.00% Mean_Red
          1.75% Mean_NIR
          0.58% SD_pan


Time: 0.0 secs
```

Figure 2.7 Wilt Decision Tree Summary

The first attribute tested for in the tree model is the mean red colour of the leaf. If this is above 110.0274, then we check the mean green level. If this is less than 222.3923 then we check the mean NIR value for the leaf. If this is below 304.1054 then we have a healthy tree. 7 Trees in the data set fall into this particular group. However, if the mean NIR is above 304.1053 then we test the mean red value again. This time, if it is above 117.875, we have a diseased tree of which 33 Trees in the data set are. Otherwise we test the mean green value again. This time, if it is below 210.2727 we now have a diseased tree. 14 of the trees in the data set fall into this particular group. Otherwise, they are healthy, which is the case for 7 of them.

If the first node's mean red value is greater than 110.0274 but node 13's mean green value is above 222.3923, then we check if its mean green value is above 236.4861 and if it is, we classify it as a healthy tree, which is the group that the vast majority of trees in the data set belong at 1247 of them. However, if the mean green is less than 236,4861, then we check if its mean red value is less than 130.8697 and if it is we classify it as a healthy tree. Otherwise, we check its mean NIR. If this is below 416.2 then we classify it as a healthy tree with 6 tree in this group. Otherwise it is classified as a diseased tree of which a further 9 trees are.

If the first nodes mean red value is less than 110.0275 then we check its mean green value. If this is above 190.8775 we classify the tree as healthy. Otherwise, we check if its mean red value falls below 89.03298. If so, it is healthy. Otherwise we check its SD_Pan value. If this is greater than 25.48325 it is classed as healthy. Otherwise we check if its mean red value is above 93.25. If so, it is classed as diseased. Otherwise, we perform one final check. If it's mean green value is above 180.022, it is classed as healthy, otherwise it is classed as diseased. The GLCM_pan attribute does not seem to be used in the decision-making process.

Evaluation of the model:

A check of the proportion of classifications in the test data reveal a 63% healthy to 37% diseased ration.

```
prop.table(table(wilttest$class))

    n     w
0.626 0.374
```

When the tree model is tested against the 500 items of test data, we see from the confusion matrix in Figure 2.8, that it was 84.4% accurate in its predictions with only a 0.2% errors overall, which seems good, but there were 74 false negatives which if weighted could be a bad result. The sensitivity of the model is 113 / 113+74 = 60% with a Precision of 113 / 113 + 4 = 96%.

```
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  500

                | actual wilt
predicted wilt |         n |         w | Row Total |
---------------|-----------|-----------|-----------|
             n |       309 |        74 |       383 |
               |     0.618 |     0.148 |           |
---------------|-----------|-----------|-----------|
             w |         4 |       113 |       117 |
               |     0.008 |     0.226 |           |
---------------|-----------|-----------|-----------|
  Column Total |       313 |       187 |       500 |
---------------|-----------|-----------|-----------|
```

Figure 2.8 Wilt Decision Tree Confusion Matrix

If a diseased tree can infect neighbouring trees and needs to be felled, it could be worse to misdiagnose a tree as healthy when it is in fact diseased. Therefore, I introduce a cost matrix giving a higher cost to making false negative classifications. I find that a maximum cost value of 45 for false negatives reduces false negatives the most. With this cost value, false negatives are reduced by more than half, as seen the confusion matrix in figure 2.11.

```
error_cost <- matrix(c(0, 1, 45, 0), nrow = 2)
```

```
     [,1] [,2]
[1,]    0   45
[2,]    1    0
```

When we pass these cost values into the C5.0 algorithm to make a new Tree Map decision model, it only uses two out of five attributes for classification. Namely Mean_Red and Mean_Green.

```
wilt_cost_model <- C5.0(class ~ ., data = wilt, costs = error_cost)
```

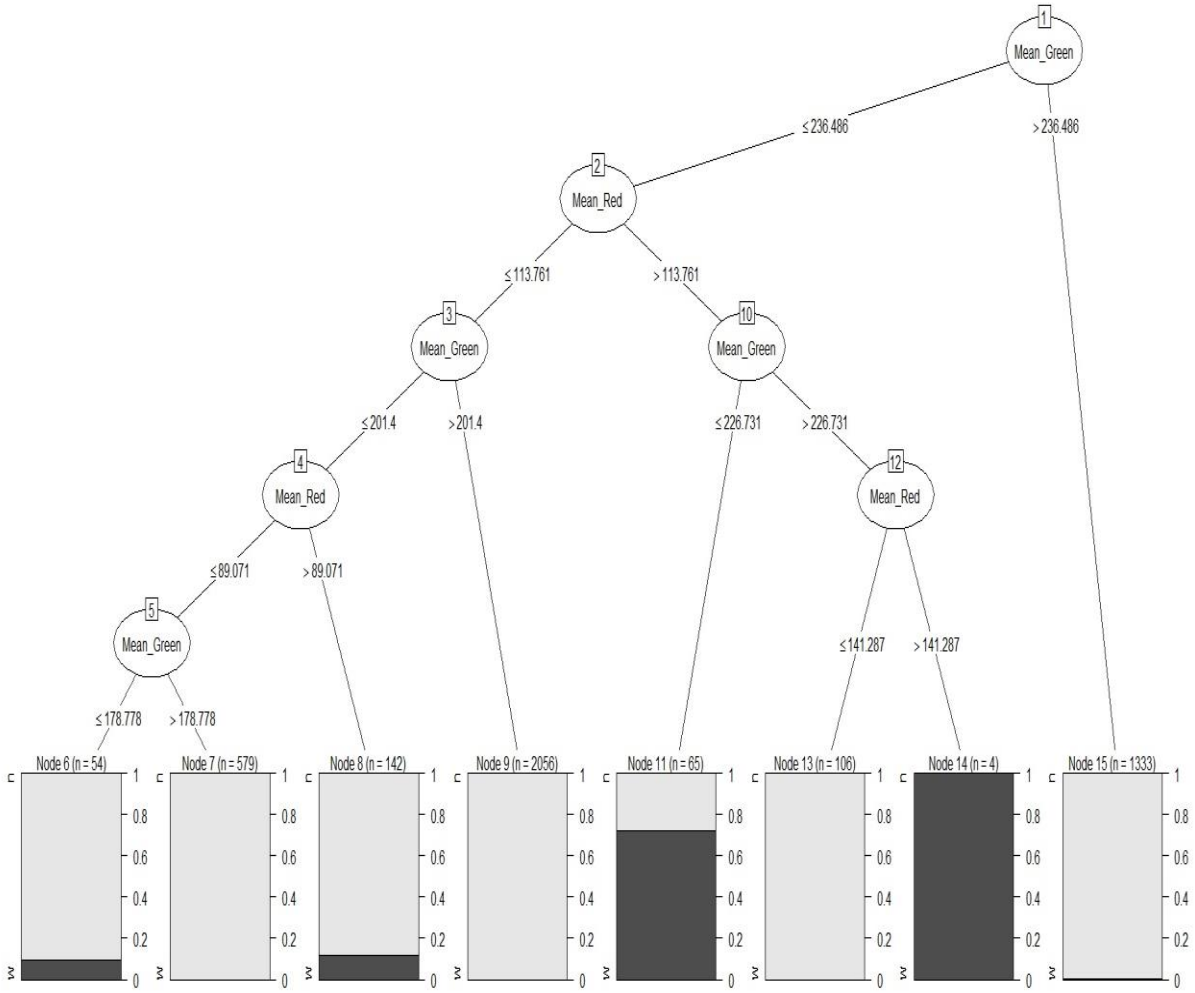


Figure 2.9 Wilt Decision Tree Weighted for False Negatives

A summary of this Tree Map is seen in figure 2.10, where the overall number of errors has increased from 0.2% to 4.4%:

```
C5.0 [Release 2.07 GPL Edition]         Thu Nov 03 10:20:33 2016
-------------------------------

Class specified by attribute `outcome'

Read 4339 cases (6 attributes) from undefined.data
Read misclassification costs from undefined.costs

Decision tree:

Mean_Green > 236.4861: n (1332)
Mean_Green <= 236.4861:
:...Mean_Red > 113.7609:
    :...Mean_Green <= 226.7308: w (65/18)
    :   Mean_Green > 226.7308:
    :   :...Mean_Red <= 141.2873: n (106)
    :       Mean_Red > 141.2873: w (5)
    Mean_Red <= 113.7609:
    :...Mean_Green > 201.4: n (2056)
        Mean_Green <= 201.4:
        :...Mean_Red > 89.07143: w (142/125)
            Mean_Red <= 89.07143:
            :...Mean_Green <= 178.7778: w (54/49)
                Mean_Green > 178.7778: n (579)


Evaluation on training data (4339 cases):

          Decision Tree
        ----------------
        Size      Errors   Cost

           8  192( 4.4%)   0.04   <<


         (a)   (b)    <-classified as
        ----  ----
        4073   192    (a): class n
                74    (b): class w


       Attribute usage:

       100.00% Mean_Green
        69.30% Mean_Red
```

Figure 2.10 Weighted Wilt Decision Tree Summary

The confusion matrix for the new decision tree seen in figure 2.11 shows that indeed the number of false negatives has reduced by more than half but the number of false positives has increased from 4 to 76, which would be bad if it resulted in the felling of healthy trees. The sensitivity of the model has increased to 151 / 151+36 = 80% but the Precision has decreased to 151 / 151 + 76 = 66%.

```
  Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  500


                | actual wilt
predicted wilt |         n |         w | Row Total |
---------------|-----------|-----------|-----------|
             n |       237 |        36 |       273 |
               |     0.474 |     0.072 |           |
---------------|-----------|-----------|-----------|
             w |        76 |       151 |       227 |
               |     0.152 |     0.302 |           |
---------------|-----------|-----------|-----------|
  Column Total |       313 |       187 |       500 |
---------------|-----------|-----------|-----------|
```

Figure 2.11 Weighted Decision Tree Confusion Matrix Cost = 45

Perhaps a better middle ground would be a false negative cost of half of the maximum of 45 that reduced false negatives by more than half. Any cost greater than 45 did not reduce the false negative result any further. Therefore, I give a cost of 20 and apply it to a new Tree Map to reduce false negatives with the minimum number of false positives.

```
error_cost <- matrix(c(0, 1, 20, 0), nrow = 2)
```

```
     [,1] [,2]
[1,]    0   20
[2,]    1    0
```

```
wilt_cost_model <- C5.0(class ~ ., data = wilt, costs = error_cost)
```

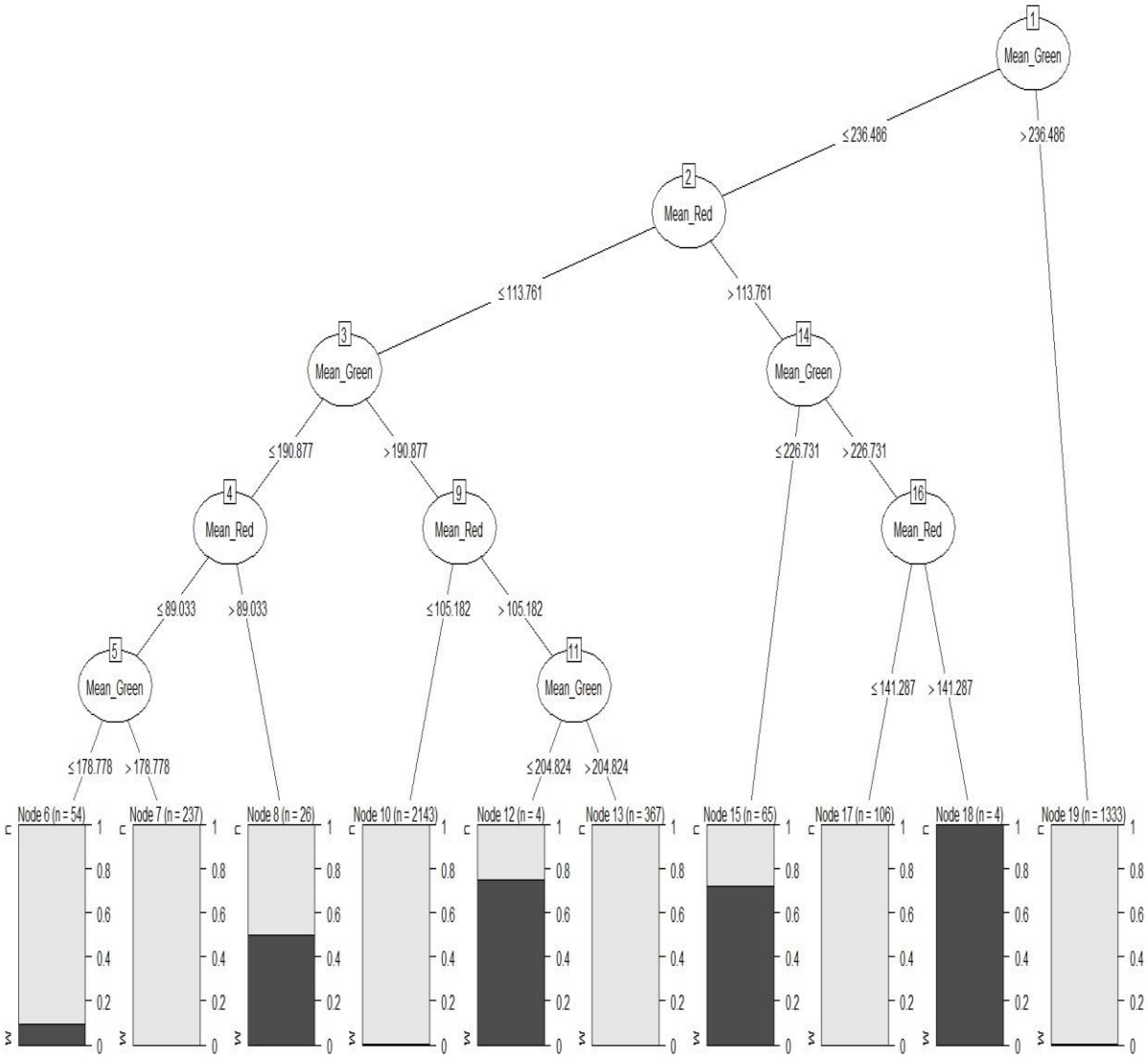


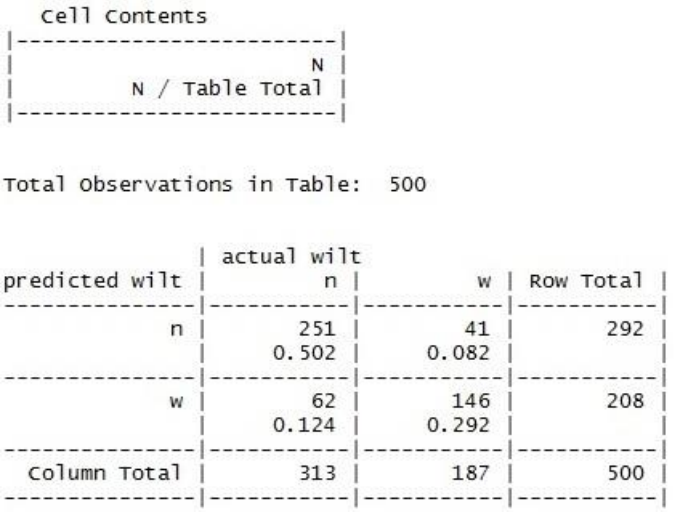
Figure 2.12 Tree Map for False Negative Cost = 20

A summary of this final Tree Map shows a dependence on only two of the attributes in the data set with an overall error of 1.9%. The summary is seen here in figure 2.13

```
Class specified by attribute `outcome'

Read 4339 cases (6 attributes) from undefined.data
Read misclassification costs from undefined.costs

Decision tree:

Mean_Green > 236.4861: n (1332)
Mean_Green <= 236.4861:
:...Mean_Red > 113.7609:
    :...Mean_Green <= 226.7308: w (65/18)
    :   Mean_Green > 226.7308:
    :   :...Mean_Red <= 141.2873: n (106)
    :       Mean_Red > 141.2873: w (5)
    Mean_Red <= 113.7609:
    :...Mean_Green <= 190.8775:
        :...Mean_Red > 89.03297: w (27/13)
        :   Mean_Red <= 89.03297:
        :   :...Mean_Green <= 178.7778: w (54/49)
        :       Mean_Green > 178.7778: n (237)
        Mean_Green > 190.8775:
        :...Mean_Red <= 105.1818: n (2143)
            Mean_Red > 105.1818:
            :...Mean_Green <= 204.8238: w (4/1)
                Mean_Green > 204.8238: n (366)


Evaluation on training data (4339 cases):

            Decision Tree
          ----------------
          Size      Errors   Cost

            10   81( 1.9%)   0.02   <<


           (a)   (b)    <-classified as
          ----  ----
          4184    81    (a): class n
                  74    (b): class w


        Attribute usage:

        100.00% Mean_Green
         69.30% Mean_Red
```

Figure 2.13 Summary of Tree Map for False Negative Cost = 20

Figure 2.14 below shows the confusion matrix for a false negative costing of 20, which reduces false negatives from the original 74 down to 41. The correct detection of diseased trees has increased from 113 up to 146. Naturally there is a higher percentage of errors but the errors have been shifted in a favourable direction by weighting the false negatives as being a graver error.



Cell Contents
N
N / Table Total

Total Observations in Table: 500

predicted wilt	actual wilt: n	actual wilt: w	Row Total
n	251	41	292
	0.502	0.082	
w	62	146	208
	0.124	0.292	
Column Total	313	187	500

Figure 2.14 Confusion Matrix for False Negative Cost = 20

Conclusion:

In conclusion, my final decision tree is derived using only two attributes but it has:

Accuracy: (146 + 251) / (313 + 187) = 397/500 = 79%.

Sensitivity: 146 / 146 + 41 = 78%.

Precision: 146 / 146 + 62 = 70%.

This should result in the least amount of unnecessary felling of healthy trees in proportion to infection of healthy trees by contamination with miss diagnosed diseased trees.

Wilt Prediction using kNN

Summary of Wilt training data:

```
class       GLCM_pan       Mean_Green      Mean_Red        Mean_NIR        SD_pan
n:4265   Min.   :  0.0   Min.   :164.6   Min.   : 59.14   Min.   :  86.5   Min.   :  0.00
w:  74   1st Qu.:118.6   1st Qu.:206.0   1st Qu.: 91.98   1st Qu.: 422.9   1st Qu.: 18.01
         Median :127.5   Median :221.5   Median :101.73   Median : 528.5   Median : 23.61
         Mean   :126.8   Mean   :233.9   Mean   :117.29   Mean   : 534.1   Mean   : 24.92
         3rd Qu.:135.0   3rd Qu.:241.8   3rd Qu.:116.87   3rd Qu.: 643.1   3rd Qu.: 29.90
         Max.   :183.3   Max.   :955.7   Max.   :746.33   Max.   :1005.5   Max.   :156.51
```

Figure 3.1 Summary of Wilt Training Data

The data obtained from the UCI website consists of two files. One is a training data set and the other is for testing. As I am not splitting the training data, I strictly don't need to randomise the data. I do it here simply to ensure an even distribution of classifications throughout the creation of the model:

```
set.seed(1)

wilt_rand <- wilt[order(runif(4339)), ]
```

I scale the attribute values relative to one another so they all have the same weight. I drop the first attribute which is the class attribute which is not used in this case for classification. Then I create a data frame for the training and testing data sets.

```
wilt_z <- as.data.frame(scale(wilt_rand[-1])).

wilt_zt <- as.data.frame(scale(wilttest[-1]))
```

Next I use the kNN algorithm in the class library to make predictions based on the training and testing data frames.

```
wilt_predictions <- knn(train = wilt_train, test = wilt_test, cl = wilt_train_class, k=1)
```

Evaluation of the model:

K=1:

Cell Contents
N
N / Row Total
N / Col Total
N / Table Total

Total Observations in Table: 500

wilt_test_class	wilt_predictions n	wilt_predictions w	Row Total
n	313	0	313
	1.000	0.000	0.626
	0.635	0.000	
	0.626	0.000	
w	180	7	187
	0.963	0.037	0.374
	0.365	1.000	
	0.360	0.014	
Column Total	493	7	500
	0.986	0.014	

Figure 3.2 Wilt kNN Confusion Matrix K=1

Accuracy: (TP + TN) / (P + N) = (7 + 313) / (7 + 493) = 320/500 = 64%.

Sensitivity: TP / (TP + FN) = 7 / 7 + 0 = 1%.

Precision: TP/ (TP + FP) = 7 / 7 + 180 = 0.03%.

This model only correctly diagnosed 7 trees as diseased out of 187, which taken with the 100% correct True Negatives gives it a precision of only 0.03%. It incorrectly guessed that 180 healthy trees would Wilt. This would be a bad if it resulted in the unnecessary felling of 180 healthy trees.

K=2:

Cell Contents
N
N / Row Total
N / Col Total
N / Table Total

Total Observations in Table: 500

wilt_test_class	wilt_predictions n	wilt_predictions w	Row Total
n	313	0	313
	1.000	0.000	0.626
	0.640	0.000	
	0.626	0.000	
w	176	11	187
	0.941	0.059	0.374
	0.360	1.000	
	0.352	0.022	
Column Total	489	11	500
	0.978	0.022	

Figure 3.3 Wilt kNN Confusion Matrix k=2

Accuracy: (TP + TN) / (P + N) = (11 + 313) / (11 + 489) = 324/500 = 65%.

Sensitivity: TP / (TP + FN) = 11 / 11 + 0 = 1%.

Precision: TP/ (TP + FP) = 11 / 11 + 176 = 0.05%.

This new model for K=2 is only marginally better than K=1. It Correctly diagnosed an extra 4 diseased trees out of the 187 diseased trees in the test set giving a precision increase from 0.03 up to 0.05%, and a 1% increase in Accuracy.

K = 3 => 63% accuracy with no false negatives and 185 false positives.

Cell Contents
N
N / Row Total
N / Col Total
N / Table Total

Total Observations in Table: 500

	wilt_predictions		
wilt_test_class	n	w	Row Total
n	313	0	313
	1.000	0.000	0.626
	0.629	0.000	
	0.626	0.000	
w	185	2	187
	0.989	0.011	0.374
	0.371	1.000	
	0.370	0.004	
Column Total	498	2	500
	0.996	0.004	

Figure 3.3 Wilt kNN Confusion Matrix K=3

Accuracy: (TP + TN) / (P + N) = (2 + 313) / (2 + 498) = 315/500 = 63%.

Sensitivity: TP / (TP + FN) = 2 / 2 + 0 = 1%.

Precision: TP/ (TP + FP) = 2 / 2 + 185 = 0.01%.

The model for K=3 is less accurate and less precise then K=2.

Conclusion:

I find that the best value for K = 2. The maximum value for K is 5 before all 500 trees in the test set are correctly classed as not diseased or falsely classed as healthy when diseased. The resulting confusion matrix only has one column for TN and FP. An accuracy of 65% for K=2 is the best kNN model for this data set.

References

[1] I-Cheng Yeh, "Modeling of strength of high performance concrete using artificial neural networks," Cement and Concrete Research, Vol. 28, No. 12, pp. 1797-1808 (1998).

http://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength (Last Visited 02.11.16).

[2]http://archive.ics.uci.edu/ml/machine-learning-databases/concrete/compressive/Concrete_Readme.txt(Last visited 02.11.16)

[3] www.nrmca.org/aboutconcrete/cips/35p.pdf (Last Visited 02.11.16)

[4] Johnson, B., Tateishi, R., Hoan, N., 2013. A hybrid pansharpening approach and multiscale object-based image analysis for mapping diseased pine and oak trees. International Journal of Remote Sensing, 34 (20), 6969-6982. http://archive.ics.uci.edu/ml/datasets/Wilt (Last Visited 02.11.16)