1600389 Coursework

Rafael Castillo

30/11/2020

Setup

Task 1

Data Exploration

Data Information

The information below shows important details about our datasets which will be useful for pre-processing.

The summary shows that the attributes "analysed" and "media" contain the same value across all the table making this columns non useful. Another useless column is the location where the experiments took place "siteIDSchema" this is because these test are universal and non exclusive to certain parts of the world.

Three of the columns left in the dataset seem to be nominal "WBCategory", "determinandC" and "method" hold 4 values each which can be used to categorized the entire dataset. The last 5 columns hold numerical values "minValue", "meanValue", "maxValue", "sd", "NSamples". #### Checks datasets dimension

```
# Water 1 dimension
dim(water1)
```

```
## [1] 3146   11
```

```
# Water 2 dimension
dim(water2)
```

```
## [1] 2485   11
```

```
# Water 3 dimension
dim(testWater)
```

```
## [1] 2489   11
```

```
# water 1 preview
head(water1)
```

Display the attributes and top 6 instances.

```
##               siteIDScheme WBCategory  determinandC analysed media NSamples
## 1     euMonitoringSiteCode         LW EEA_3131-01-9    total water        5
## 2 eionetMonitoringSiteCode         RW EEA_3132-01-2    total water        4
## 3     euMonitoringSiteCode         LW EEA_3131-01-9    total water        2
## 4     euMonitoringSiteCode         LW CAS_7723-14-0    total water        6
## 5 eionetMonitoringSiteCode         LW EEA_3131-01-9    total water        1
## 6     euMonitoringSiteCode         LW EEA_3131-01-9    total water        1
##   minValue meanValue maxValue     sd        method
## 1   37.200    60.432   81.600 20.174      SFS 3040
## 2   10.500    10.900   11.300     NA EN 25813:1992
## 3   74.857    74.928   75.000  0.101      SFS 3040
## 4    0.005     0.006    0.008  0.000      SFS 3026
## 5   78.500    78.500   78.500     NA      SFS 3040
## 6   95.500    95.500   95.500     NA      SFS 3040
```

```
# water 2 preview
head(water2)
```

```
##               siteIDScheme WBCategory  determinandC analysed media NSamples
## 1     euMonitoringSiteCode         LW CAS_7723-14-0    total water        3
## 2 eionetMonitoringSiteCode         TW EEA_3132-01-2    total water        4
## 3     euMonitoringSiteCode         RW EEA_3131-01-9    total water        5
## 4     euMonitoringSiteCode         LW EEA_3131-01-9    total water        5
## 5     euMonitoringSiteCode         LW CAS_7723-14-0    total water        1
## 6     euMonitoringSiteCode         GW EEA_3132-01-2    total water        1
##   minValue meanValue maxValue      sd        method
## 1    0.041     0.047    0.058  0.0090      SFS 3026
## 2    6.670     8.240   10.600  1.5500 EN 25813:1992
## 3   87.000    89.400   91.000  1.8166 EN 25813:1992
## 4   27.500    41.880   53.250 10.1860      SFS 3040
## 5    0.015     0.015    0.015      NA      SFS 3026
## 6    3.980     3.980    3.980  0.0000 EN 25813:1992
```

```
# testWater preview
head(testWater)
```

```
##               siteIDScheme WBCategory  determinandC analysed media NSamples
## 1     euMonitoringSiteCode         LW CAS_7723-14-0    total water        3
## 2 eionetMonitoringSiteCode         TW EEA_3132-01-2    total water        4
## 3     euMonitoringSiteCode         RW EEA_3131-01-9    total water        5
## 4     euMonitoringSiteCode         LW EEA_3131-01-9    total water        5
## 5     euMonitoringSiteCode         LW CAS_7723-14-0    total water        1
## 6     euMonitoringSiteCode         GW EEA_3132-01-2    total water        1
##   minValue meanValue maxValue      sd        method
## 1    0.041     0.047    0.058  0.0090      SFS 3026
## 2    6.670     8.240   10.600  1.5500 EN 25813:1992
## 3   87.000    89.400   91.000  1.8166 EN 25813:1992
## 4   27.500    41.880   53.250 10.1860      SFS 3040
## 5    0.015     0.015    0.015      NA      SFS 3026
## 6    3.980     3.980    3.980  0.0000 EN 25813:1992
```

```r
# Water 1 summary
summary(water1)
```

Summary of the datasets

```
##                     siteIDScheme  WBCategory         determinandC   analysed
##  eionetMonitoringSiteCode: 154   GW:  64    CAS_14798-03-9: 420   total:3146
##  euMonitoringSiteCode    :2992   LW:2941    CAS_7723-14-0 : 802
##                                  RW: 140    EEA_3131-01-9 : 885
##                                  TW:   1    EEA_3132-01-2 :1039
##
##
##
##    media          NSamples         minValue           meanValue
##  water:3146   Min.   : 1.000   Min.   :  0.00000   Min.   :  0.00100
##               1st Qu.: 2.000   1st Qu.:  0.01619   1st Qu.:  0.02502
##               Median : 2.000   Median :  6.26300   Median :  7.80650
##               Mean   : 3.203   Mean   : 19.07865   Mean   : 22.58591
##               3rd Qu.: 4.000   3rd Qu.: 33.46875   3rd Qu.: 52.27325
##               Max.   :25.000   Max.   :108.33300   Max.   :108.33300
##
##     maxValue               sd                              method
##  Min.   :  0.00129   Min.   : 0.000   EN 25813:1992          :460
##  1st Qu.:  0.03300   1st Qu.: 0.004   SFS 3026               :802
##  Median :  8.92350   Median : 0.809   SFS 3032 + EN ISO 11732:420
##  Mean   : 25.95354   Mean   : 3.917   SFS 3040               :721
##  3rd Qu.: 62.83268   3rd Qu.: 2.987   SFS 3040 SFS-EN 25813  :743
##  Max.   :135.50000   Max.   :53.594
##                      NA's   :314
```

```r
# Water 2 summary
summary(water2)
```

```
##                     siteIDScheme  WBCategory         determinandC  analysed
##  eionetMonitoringSiteCode: 112   GW:  56    CAS_14798-03-9:328   total:2485
##  euMonitoringSiteCode    :2373   LW:2318    CAS_7723-14-0 :632
##                                  RW: 110    EEA_3131-01-9 :700
##                                  TW:   1    EEA_3132-01-2 :825
##
##
##
##    media          NSamples         minValue          meanValue
##  water:2485   Min.   : 1.000   Min.   :  0.000   Min.   :  0.00129
##               1st Qu.: 2.000   1st Qu.:  0.017   1st Qu.:  0.02700
##               Median : 2.000   Median :  6.333   Median :  7.55000
##               Mean   : 3.241   Mean   : 19.627   Mean   : 22.78879
##               3rd Qu.: 4.000   3rd Qu.: 36.666   3rd Qu.: 53.17800
##               Max.   :25.000   Max.   :104.000   Max.   :104.00000
##
##     maxValue               sd                              method
##  Min.   :  0.00129   Min.   : 0.000   EN 25813:1992          :365
```

```
##  1st Qu.:  0.03430   1st Qu.: 0.004   SFS 3026               :632
##  Median :  8.82500   Median : 0.807   SFS 3032 + EN ISO 11732:328
##  Mean   : 25.80856   Mean   : 3.520   SFS 3040               :561
##  3rd Qu.: 60.66600   3rd Qu.: 2.842   SFS 3040 SFS-EN 25813  :599
##  Max.   :137.50000   Max.   :67.882
##                      NA's   :258
```

```
# testWater summary
summary(testWater)
```

```
##                    siteIDScheme  WBCategory         determinandC  analysed
##  eionetMonitoringSiteCode: 116   GW:  56    CAS_14798-03-9:328   total:2489
##  euMonitoringSiteCode    :2373   LW:2318    CAS_7723-14-0 :632
##                                  RW: 110    EEA_3131-01-9 :700
##                                  TW:   5    EEA_3132-01-2 :829
##
##
##
##    media          NSamples         minValue          meanValue
##  water:2489   Min.   : 1.000   Min.   :  0.000   Min.   :  0.00129
##               1st Qu.: 2.000   1st Qu.:  0.017   1st Qu.:  0.02700
##               Median : 2.000   Median :  6.367   Median :  7.55560
##               Mean   : 3.249   Mean   : 19.608   Mean   : 22.76799
##               3rd Qu.: 4.000   3rd Qu.: 36.666   3rd Qu.: 53.16600
##               Max.   :25.000   Max.   :104.000   Max.   :104.00000
##
##     maxValue               sd                              method
##  Min.   :  0.00129   Min.   : 0.000   EN 25813:1992          :369
##  1st Qu.:  0.03430   1st Qu.: 0.004   SFS 3026               :632
##  Median :  8.83300   Median : 0.808   SFS 3032 + EN ISO 11732:328
##  Mean   : 25.78533   Mean   : 3.515   SFS 3040               :561
##  3rd Qu.: 60.60000   3rd Qu.: 2.837   SFS 3040 SFS-EN 25813  :599
##  Max.   :137.50000   Max.   :67.882
##                      NA's   :258
```

```
# Water 1 duplicate count
nrow(water1[duplicated(water1),])
```

Checks for duplicates on the dataset.

```
## [1] 145
```

```
# Water 1 duplicate count
nrow(water2[duplicated(water2),])
```

```
## [1] 113
```

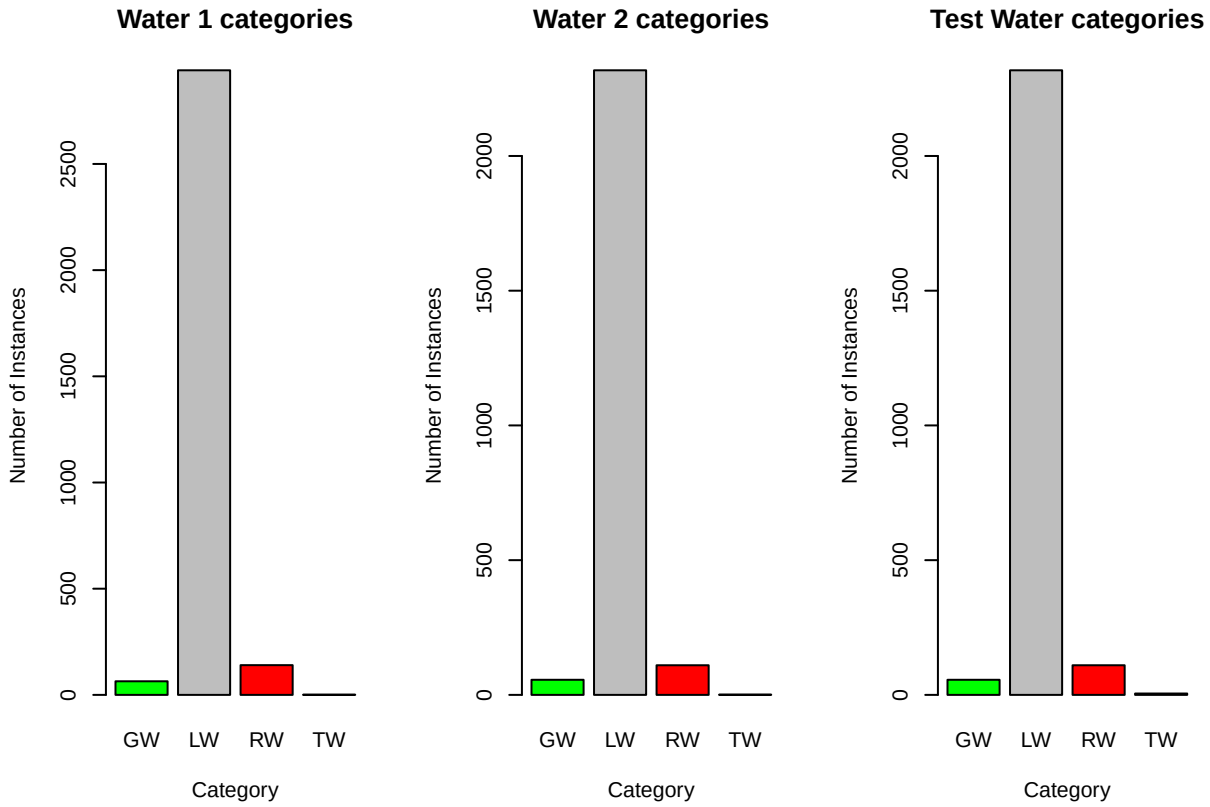
```r
# Water 1 duplicate count
nrow(testWater[duplicated(testWater),])
```

```
## [1] 114
```

Interesting Facts

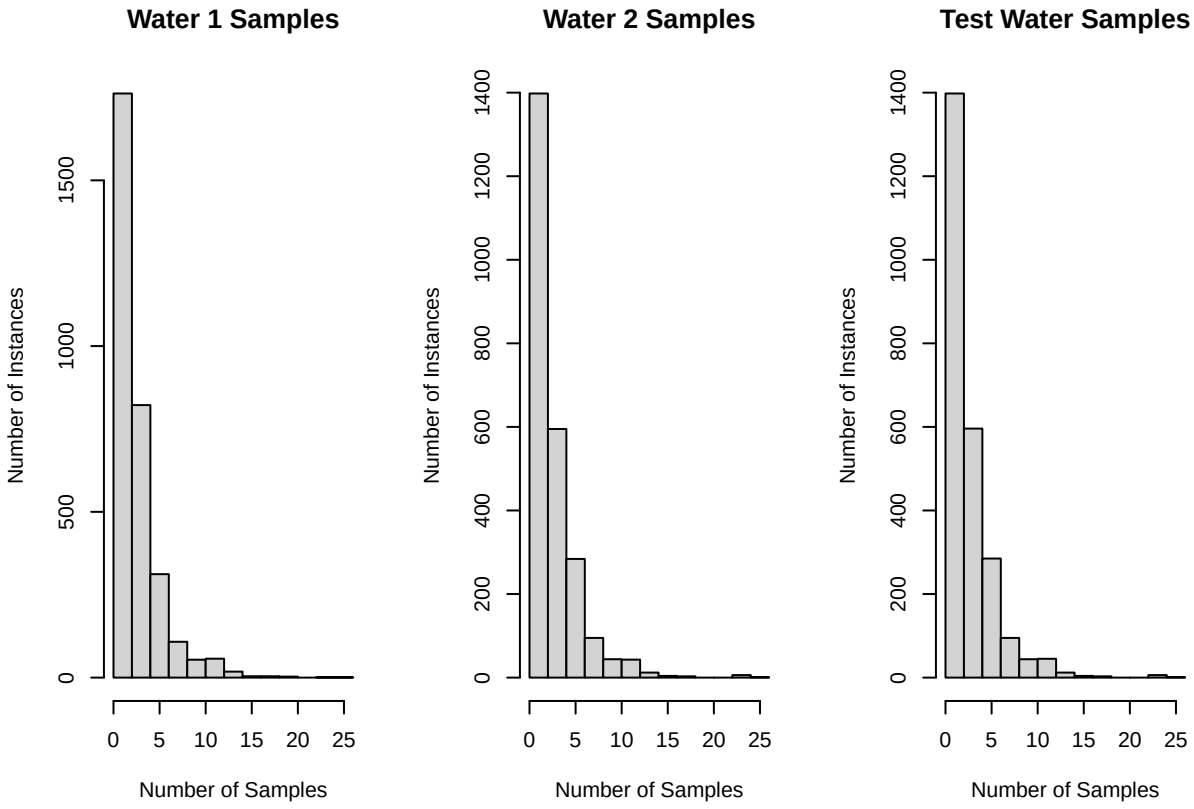
Analysis of WBCategories It seems that this dataset is majorly based on lake waters as most experiments belong to this category across all datasets. On the same note transit water is the least used water for these tests. I am incline to assume that the dataset is focused on lake water but also takes into consideration other types for contrast purposes.

```r
par(mfrow=c(1,3))
# Water 1 WBCategories analysis
plot(water1$WBCategory, main = "Water 1 categories", xlab = "Category",
     ylab = "Number of Instances", col = c("green", "grey", "red", "blue"))
# Water 2 WBCategories analysis
plot(water2$WBCategory, main = "Water 2 categories", xlab = "Category",
     ylab = "Number of Instances", col = c("green", "grey", "red", "blue"))
# testWater WBCategories analysis
plot(testWater$WBCategory, main = "Test Water categories", xlab = "Category",
     ylab = "Number of Instances", col = c("green", "grey", "red", "blue"))
```



Analysis of NSamples Most experiments where taken by considering 1 to 5 samples. I assume that perhaps there is no need in taking many samples to get the necessary data to execute these tests.

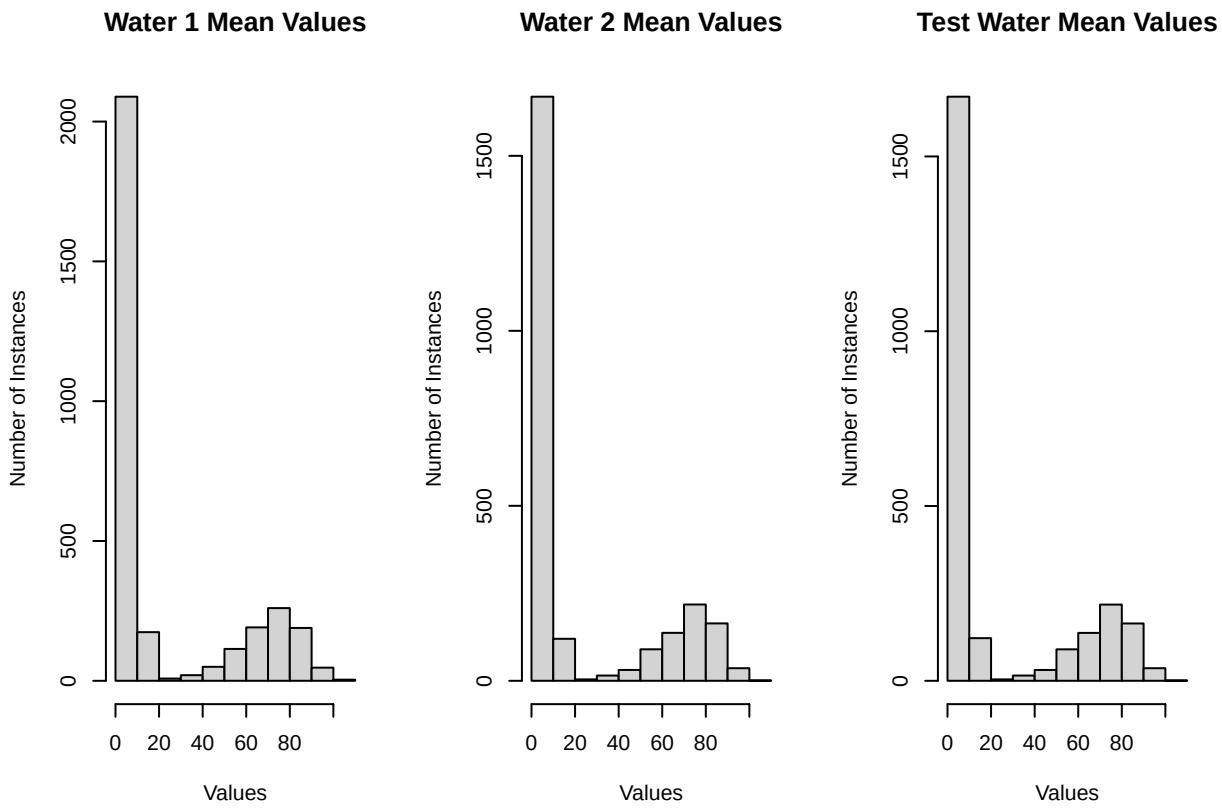
```
par(mfrow=c(1,3))
# Water 1 NSamples analysis
hist(water1$NSamples, main = "Water 1 Samples", xlab = "Number of Samples",
     ylab = "Number of Instances")
# Water 2 NSamples analysis
hist(water2$NSamples, main = "Water 2 Samples", xlab = "Number of Samples",
     ylab = "Number of Instances")
# testWater NSamples analysis
hist(testWater$NSamples, main = "Test Water Samples", xlab = "Number of Samples",
     ylab = "Number of Instances")
```



Analysis of Mean Values Most of the values within the dataset seems to fall between 2 groups. In the range of 0-10 we can find the higest concentration of mean values and between 50-90 we seem to have another big concentration of values although within a larger range. There seems to be a major gap between both ranges showing the lack of experiment resulting in the mean value between 15-50.

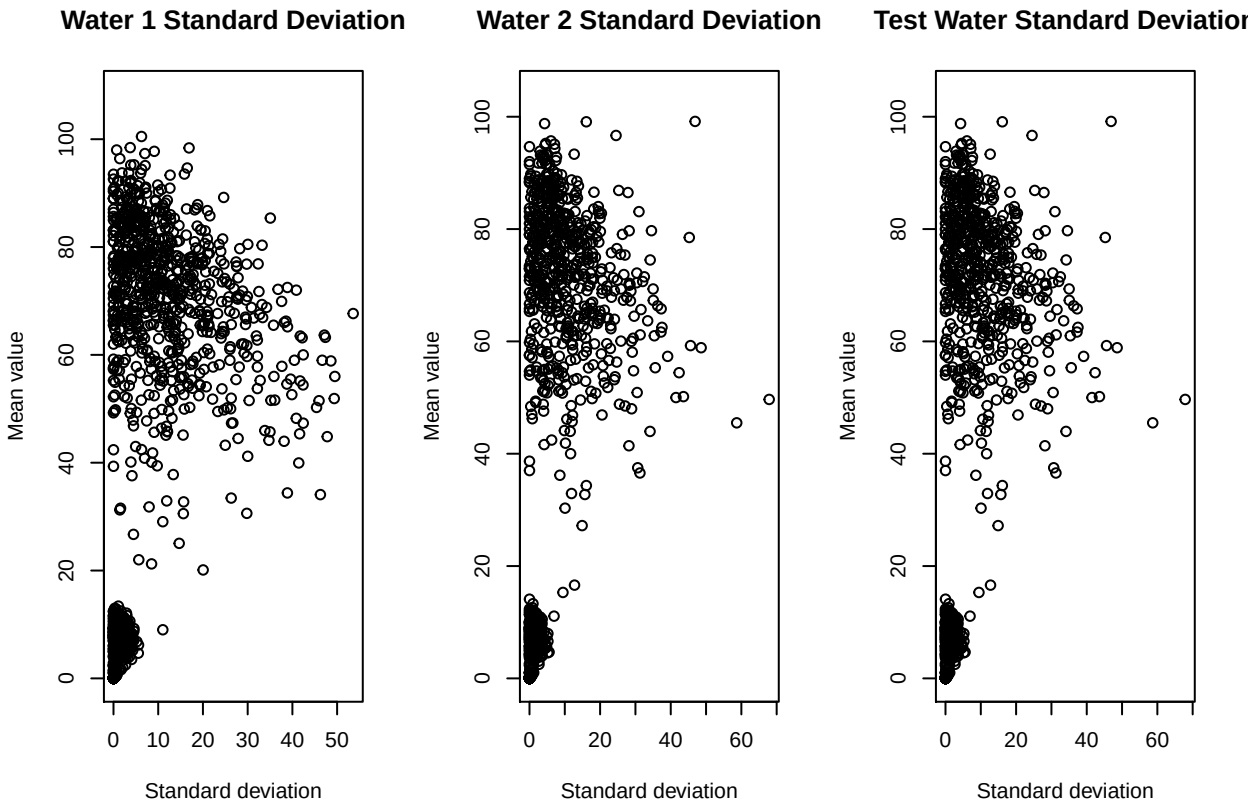
```
par(mfrow=c(1,3))
# Water 1 mean value analysis
hist(water1$meanValue, main = "Water 1 Mean Values", xlab = "Values",
     ylab = "Number of Instances")
# Water 2 mean value analysis
hist(water2$meanValue, main = "Water 2 Mean Values", xlab = "Values",
```

```
     ylab = "Number of Instances")
# testWater mean value analysis
hist(testWater$meanValue, main = "Test Water Mean Values", xlab = "Values",
     ylab = "Number of Instances")
```



Analysis of Standard deviation. The charts below show the mean value and the standard deviation of for each experiment. It is interesting to note that low values also have low standard deviation weas high values are more prompt to a higher standard deviation.

```
par(mfrow=c(1,3))
# Water 1 standart deviation analysis
plot(water1$sd, water1$meanValue, main = "Water 1 Standard Deviation",
     xlab = "Standard deviation", ylab = "Mean value")
# Water 2 standart deviation analysis
plot(water2$sd, water2$meanValue, main = "Water 2 Standard Deviation",
     xlab = "Standard deviation", ylab = "Mean value")
# testWater standart deviation analysis
plot(testWater$sd, testWater$meanValue, main = "Test Water Standard Deviation",
     xlab = "Standard deviation", ylab = "Mean value")
```



Pre-prossesing

Pre-processing allows for models to train quicker and achieve robust results.

Remove useless columns from datasets

Because of the repetitive nature of of the "analyzed" and "media" columns I have decided to remove them from the training as it adds no value. I have also decided to remove the "siteIDScheme" because this test are universal and regardless of the place they are executed they will yield the same results.

```
water1$analysed = NULL
water1$media = NULL
water1$siteIDScheme = NULL
water2$analysed = NULL
water2$media = NULL
water2$siteIDScheme = NULL
testWater$analysed = NULL
testWater$media = NULL
testWater$siteIDScheme = NULL
```

Replacing NA values

This section iterates through every column and if the column is numeric and has missing values it replaces them with the mean value.

```
# Replacing NA values for water 1
for(i in 1:ncol(water1)){
  if(is.numeric(water1[,i])){
  water1[is.na(water1[,i]), i] = mean(water1[,i], na.rm = TRUE)
  }
}
# Replacing NA values for water 2
for(i in 1:ncol(water2)){
   if(is.numeric(water2[,i])){
  water2[is.na(water2[,i]), i] = mean(water2[,i], na.rm = TRUE)
   }
}
# Replacing NA values for testWater
for(i in 1:ncol(testWater)){
   if(is.numeric(testWater[,i])){
  testWater[is.na(testWater[,i]), i] = mean(testWater[,i], na.rm = TRUE)
   }
}
```

Joining water1 with wate2

```
water = rbind(water1,water2)
dim(water)
```

```
## [1] 5631    8
```

Part B

Information gained.

The results show that the most important attribute which influences our results the most is the method used to test the samples.

```
# Checks the attributes that contain the higher information gained
GainRatioAttributeEval(WBCategory ~., data=water1)
```

```
## determinandC     NSamples     minValue    meanValue     maxValue           sd
##   0.03540364   0.04535611   0.04841880   0.05971038   0.06151723   0.02488260
##       method
##   0.08907334
```

```
GainRatioAttributeEval(WBCategory ~., data=water)
```

```
## determinandC     NSamples     minValue    meanValue     maxValue           sd
##   0.03610194   0.04218459   0.04492827   0.05966370   0.05869635   0.02432192
##       method
##   0.09037767
```

Water 1 Classifiers

For the training control I have only chosen repeated cross validation. This allows us to use all our data for testing and training by partitioning in 10 sets for testing. 3 repeats for 10 fold gets us the average of 3 error terms obtained by executing 10 fold CV 3 times getting good results without too much computational time required.

```
# setting seed for reproducible experiment
set.seed(34)
# Preparing training method
ctrl = trainControl(method="repeatedcv", number=10, repeats=3)

# Tree Classifier J48
J48Water1 = train(WBCategory ~., data=water1, method = "J48", metric = "Accuracy",
                  trControl = ctrl)
# Tree classifier CART
cartWater1 = train(WBCategory ~., data= water1, method = "rpart", metric = "Accuracy",
                   trControl = ctrl)
# Instance Base classifier KNN
knnWater1 = train(WBCategory ~., data= water1, method = "knn", metric = "Accuracy",
                  trControl = ctrl)
```

Water 1 Results

It looks like a small pruning of 0.010 and a minimum number of objects if 1 yield the best results for J48. The bigger the pruning confidence the more complex is our tree which impacts the accuracy negatively. The CART algorithm found the best accuracy by using a complexity parameter of 0.048 " optimal size of decision tree. KNN found the best accuracy by selecting 5 nearest neighbors, the more neighbors selected the smaller the accuracy.

```
# water2 J48 run values
J48Water1
```

```
## C4.5-like Trees
##
## 3146 samples
##    7 predictor
##    4 classes: 'GW', 'LW', 'RW', 'TW'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 2831, 2832, 2832, 2831, 2832, 2831, ...
## Resampling results across tuning parameters:
##
##   C      M  Accuracy   Kappa
##   0.010  1  0.9723475  0.7704837
##   0.010  2  0.9718178  0.7653886
##   0.010  3  0.9711822  0.7611370
##   0.255  1  0.9692771  0.7392184
##   0.255  2  0.9685340  0.7328817
##   0.255  3  0.9672614  0.7227822
##   0.500  1  0.9700185  0.7456413
##   0.500  2  0.9700165  0.7449855
```

```
##   0.500  3  0.9691672  0.7395959
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were C = 0.01 and M = 1.
```

```
# water1 CART run values
cartWater1
```

```
## CART
##
## 3146 samples
##    7 predictor
##    4 classes: 'GW', 'LW', 'RW', 'TW'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 2831, 2832, 2832, 2832, 2832, 2832, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.04878049  0.9515771  0.5423229
##   0.05203252  0.9476597  0.4647621
##   0.12195122  0.9359005  0.1404302
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.04878049.
```

```
# water1 KNN run values
knnWater1
```

```
## k-Nearest Neighbors
##
## 3146 samples
##    7 predictor
##    4 classes: 'GW', 'LW', 'RW', 'TW'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 2832, 2832, 2831, 2831, 2831, 2831, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.9591089  0.6149224
##   7  0.9576254  0.5775071
##   9  0.9564610  0.5399164
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

```
# water1 J48 outputs values for final model
print(J48Water1$finalModel)
```

```
## J48 pruned tree
```

```
## ------------------
##
## NSamples <= 3
## |   determinandCEEA_3132-01-2 <= 0: LW (1485.0/9.0)
## |   determinandCEEA_3132-01-2 > 0
## |   |   methodSFS 3040 SFS-EN 25813 <= 0
## |   |   |   meanValue <= 7.05
## |   |   |   |   NSamples <= 2
## |   |   |   |   |   minValue <= 3.924: GW (28.0)
## |   |   |   |   |   minValue > 3.924
## |   |   |   |   |   |   sd <= 2.43: GW (28.0/6.0)
## |   |   |   |   |   |   sd > 2.43: LW (9.0)
## |   |   |   |   NSamples > 2: LW (4.0)
## |   |   |   meanValue > 7.05: LW (114.0/26.0)
## |   |   methodSFS 3040 SFS-EN 25813 > 0: LW (519.0)
## NSamples > 3
## |   NSamples <= 11
## |   |   minValue <= 6.77: LW (486.0/3.0)
## |   |   minValue > 6.77
## |   |   |   methodSFS 3040 <= 0
## |   |   |   |   methodSFS 3040 SFS-EN 25813 <= 0
## |   |   |   |   |   sd <= 8.09958: RW (90.0/23.0)
## |   |   |   |   |   sd > 8.09958: LW (20.0/2.0)
## |   |   |   |   methodSFS 3040 SFS-EN 25813 > 0: LW (92.0)
## |   |   |   methodSFS 3040 > 0: LW (200.0)
## |   NSamples > 11
## |   |   methodSFS 3040 <= 0
## |   |   |   maxValue <= 10.188: LW (9.0)
## |   |   |   maxValue > 10.188
## |   |   |   |   methodSFS 3040 SFS-EN 25813 <= 0: RW (50.0/2.0)
## |   |   |   |   methodSFS 3040 SFS-EN 25813 > 0: LW (2.0)
## |   |   methodSFS 3040 > 0: LW (10.0)
##
## Number of Leaves  :  16
##
## Size of the tree :   31
```

```
# water1 CART outputs values for final model
print(cartWater1$finalModel)
```

```
## n= 3146
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 3146 205 LW (0.0203432931 0.9348378894 0.0445009536 0.0003178640)
##    2) NSamples< 11.5 3075 157 LW (0.0208130081 0.9489430894 0.0299186992 0.0003252033)
##      4) determinandCEEA_3132-01-2>=0.5 1006 125 LW (0.0636182903 0.8757455268 0.0596421471 0.0009940
##        8) methodSFS 3040 SFS-EN 25813< 0.5 267 125 LW (0.2397003745 0.5318352060 0.2247191011 0.0037
##         16) meanValue< 5.7354 48   8 GW (0.8333333333 0.1666666667 0.0000000000 0.0000000000) *
##         17) meanValue>=5.7354 219  85 LW (0.1095890411 0.6118721461 0.2739726027 0.0045662100)
##           34) NSamples< 3.5 136  36 LW (0.1764705882 0.7352941176 0.0882352941 0.0000000000) *
##           35) NSamples>=3.5 83  35 RW (0.0000000000 0.4096385542 0.5783132530 0.0120481928)
##             70) meanValue< 9.0846 30   7 LW (0.0000000000 0.7666666667 0.2000000000 0.0333333333) *
```

```
##             71) meanValue>=9.0846 53  11 RW (0.0000000000 0.2075471698 0.7924528302 0.0000000000) *
##        9) methodSFS 3040 SFS-EN 25813>=0.5 739   0 LW (0.0000000000 1.0000000000 0.0000000000 0.0000
##      5) determinandCEEA_3132-01-2< 0.5 2069  32 LW (0.0000000000 0.9845335911 0.0154664089 0.0000000
##    3) NSamples>=11.5 71  23 RW (0.0000000000 0.3239436620 0.6760563380 0.0000000000) *
```

```
# water1 KNN outputs values for final model
print(knnWater1$finalModel)
```

```
## 5-nearest neighbor model
## Training set outcome distribution:
##
##   GW   LW   RW   TW
##   64 2941  140    1
```

```
# water1 J48 output results
print(J48Water1$results)
```

```
##       C M  Accuracy     Kappa  AccuracySD    KappaSD
## 1 0.010 1 0.9723475 0.7704837 0.008759308 0.07584109
## 2 0.010 2 0.9718178 0.7653886 0.008821371 0.07442038
## 3 0.010 3 0.9711822 0.7611370 0.008749117 0.07387794
## 4 0.255 1 0.9692771 0.7392184 0.010629660 0.09232798
## 5 0.255 2 0.9685340 0.7328817 0.010989713 0.09202604
## 6 0.255 3 0.9672614 0.7227822 0.011266232 0.09435919
## 7 0.500 1 0.9700185 0.7456413 0.008199927 0.07897745
## 8 0.500 2 0.9700165 0.7449855 0.008620659 0.08080036
## 9 0.500 3 0.9691672 0.7395959 0.009305481 0.08680583
```

```
# water1 CART output results
print(cartWater1$results)
```

```
##           cp  Accuracy     Kappa  AccuracySD   KappaSD
## 1 0.04878049 0.9515771 0.5423229 0.011805206 0.1616750
## 2 0.05203252 0.9476597 0.4647621 0.010936421 0.1672294
## 3 0.12195122 0.9359005 0.1404302 0.003387642 0.1464974
```

```
# water1 KNN output results
print(knnWater1$results)
```

```
##   k  Accuracy     Kappa  AccuracySD    KappaSD
## 1 5 0.9591089 0.6149224 0.010217858 0.10105704
## 2 7 0.9576254 0.5775071 0.009889002 0.09636391
## 3 9 0.9564610 0.5399164 0.009106149 0.10007619
```

Water classifiers

The same conclusion discussed in water1 applies for water classifiers.

```r
# setting seed for reproducible experiment
set.seed(34)
# Preparing training method
ctrl = trainControl(method="repeatedcv", number=10, repeats=3)


# Tree classifier for water
J48Water = train(WBCategory ~., data=water,method = "J48", metric = "Accuracy",
                 trControl = ctrl)
# Tree classifier for water
cartWater = train(WBCategory ~., data=water,method = "rpart", metric = "Accuracy",
                  trControl = ctrl)
# Instance base classifier for water
knnWater = train(WBCategory ~., data=water,method = "knn", metric = "Accuracy",
                 trControl = ctrl)


# water1 run values
J48Water
```

```
## C4.5-like Trees
##
## 5631 samples
##    7 predictor
##    4 classes: 'GW', 'LW', 'RW', 'TW'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 5068, 5068, 5068, 5067, 5068, 5068, ...
## Resampling results across tuning parameters:
##
##   C      M  Accuracy   Kappa
##   0.010  1  0.9724746  0.7617824
##   0.010  2  0.9722380  0.7604196
##   0.010  3  0.9713500  0.7545526
##   0.255  1  0.9721192  0.7643863
##   0.255  2  0.9721785  0.7633524
##   0.255  3  0.9706990  0.7526735
##   0.500  1  0.9720594  0.7620788
##   0.500  2  0.9718823  0.7593927
##   0.500  3  0.9709357  0.7530569
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were C = 0.01 and M = 1.
```

```r
# water run values
cartWater
```

```
## CART
##
## 5631 samples
##    7 predictor
##    4 classes: 'GW', 'LW', 'RW', 'TW'
```

```
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 5068, 5068, 5069, 5068, 5068, 5068, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.04749104  0.9530009  0.5436650
##   0.06451613  0.9415753  0.3035310
##   0.08870968  0.9351215  0.1380476
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.04749104.
```

```
# water run values
knnWater
```

```
## k-Nearest Neighbors
##
## 5631 samples
##    7 predictor
##    4 classes: 'GW', 'LW', 'RW', 'TW'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 5068, 5068, 5068, 5068, 5068, 5068, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.9587402  0.6357211
##   7  0.9591542  0.6260564
##   9  0.9582072  0.6073090
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 7.
```

```
# water outputs values for final model
print(J48Water$finalModel)
```

```
## J48 pruned tree
## ------------------
##
## NSamples <= 3
## |   determinandCEEA_3132-01-2 <= 0: LW (2641.0/19.0)
## |   determinandCEEA_3132-01-2 > 0
## |   |   methodSFS 3040 SFS-EN 25813 <= 0
## |   |   |   meanValue <= 5.808
## |   |   |   |   sd <= 3.425
## |   |   |   |   |   NSamples <= 2
## |   |   |   |   |   |   minValue <= 0.9: LW (1.0)
## |   |   |   |   |   |   minValue > 0.9: GW (74.0/2.0)
## |   |   |   |   |   NSamples > 2: LW (3.0)
## |   |   |   |   sd > 3.425
```

```
## |   |   |   |   |   NSamples <= 1: LW (10.0)
## |   |   |   |   |   NSamples > 1: GW (3.0)
## |   |   |   meanValue > 5.808
## |   |   |   |   meanValue <= 11.359: LW (215.0/59.0)
## |   |   |   |   meanValue > 11.359
## |   |   |   |   |   minValue <= 11.4: RW (7.0)
## |   |   |   |   |   minValue > 11.4
## |   |   |   |   |   |   minValue <= 12.36: LW (10.0/2.0)
## |   |   |   |   |   |   minValue > 12.36: RW (2.0)
## |   |   methodSFS 3040 SFS-EN 25813 > 0: LW (948.0)
## NSamples > 3
## |   NSamples <= 11
## |   |   minValue <= 6.633: LW (854.0/3.0)
## |   |   minValue > 6.633
## |   |   |   methodSFS 3040 <= 0
## |   |   |   |   methodSFS 3040 SFS-EN 25813 <= 0
## |   |   |   |   |   sd <= 8.721
## |   |   |   |   |   |   maxValue <= 10.645
## |   |   |   |   |   |   |   sd <= 1.512
## |   |   |   |   |   |   |   |   maxValue <= 10.575: LW (25.0/5.0)
## |   |   |   |   |   |   |   |   maxValue > 10.575: RW (1.0)
## |   |   |   |   |   |   |   sd > 1.512
## |   |   |   |   |   |   |   |   sd <= 1.6061: TW (2.0)
## |   |   |   |   |   |   |   |   sd > 1.6061: RW (7.0)
## |   |   |   |   |   |   maxValue > 10.645: RW (134.0/26.0)
## |   |   |   |   |   sd > 8.721: LW (34.0/3.0)
## |   |   |   |   methodSFS 3040 SFS-EN 25813 > 0: LW (172.0)
## |   |   |   methodSFS 3040 > 0: LW (363.0)
## |   NSamples > 11
## |   |   maxValue <= 10.07: LW (24.0)
## |   |   maxValue > 10.07
## |   |   |   methodSFS 3040 <= 0
## |   |   |   |   methodSFS 3040 SFS-EN 25813 <= 0: RW (83.0/4.0)
## |   |   |   |   methodSFS 3040 SFS-EN 25813 > 0: LW (3.0)
## |   |   |   methodSFS 3040 > 0: LW (15.0)
##
## Number of Leaves  :  24
##
## Size of the tree :   47
```

```
# water outputs values for final model
print(cartWater$finalModel)
```

```
## n= 5631
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 5631 372 LW (0.0213106020 0.9339371337 0.0443970876 0.0003551767)
##   2) NSamples< 11.5 5506 293 LW (0.0217944061 0.9467853251 0.0310570287 0.0003632401) *
##   3) NSamples>=11.5 125  46 RW (0.0000000000 0.3680000000 0.6320000000 0.0000000000)
##     6) maxValue< 10.07 24   0 LW (0.0000000000 1.0000000000 0.0000000000 0.0000000000) *
##     7) maxValue>=10.07 101  22 RW (0.0000000000 0.2178217822 0.7821782178 0.0000000000) *
```

```r
# water outputs values for final model
print(knnWater$finalModel)
```

```
## 7-nearest neighbor model
## Training set outcome distribution:
##
##   GW   LW   RW   TW
##  120 5259  250    2
```

```r
# water output results
print(J48Water$results)
```

```
##       C M  Accuracy     Kappa  AccuracySD    KappaSD
## 1 0.010 1 0.9724746 0.7617824 0.005722802 0.05592209
## 2 0.010 2 0.9722380 0.7604196 0.005736137 0.05533032
## 3 0.010 3 0.9713500 0.7545526 0.005816352 0.05459117
## 4 0.255 1 0.9721192 0.7643863 0.006996146 0.06340781
## 5 0.255 2 0.9721785 0.7633524 0.006762019 0.06245029
## 6 0.255 3 0.9706990 0.7526735 0.006878348 0.06179681
## 7 0.500 1 0.9720594 0.7620788 0.007343542 0.06637805
## 8 0.500 2 0.9718823 0.7593927 0.007134735 0.06431922
## 9 0.500 3 0.9709357 0.7530569 0.007107408 0.06317549
```

```r
# water output results
print(cartWater$results)
```

```
##           cp  Accuracy     Kappa  AccuracySD    KappaSD
## 1 0.04749104 0.9530009 0.5436650 0.009784914 0.17105007
## 2 0.06451613 0.9415753 0.3035310 0.005849207 0.08112397
## 3 0.08870968 0.9351215 0.1380476 0.002353288 0.12750191
```

```r
# water output results
print(knnWater$results)
```

```
##   k  Accuracy     Kappa  AccuracySD    KappaSD
## 1 5 0.9587402 0.6357211 0.007176380 0.06612933
## 2 7 0.9591542 0.6260564 0.007494113 0.07130132
## 3 9 0.9582072 0.6073090 0.007268601 0.07060501
```

Comparing results for water1 and water

For the dataset water1 we found J48 to obtain the best performance in accuracy and kappa compared to CART and KNN. The performance of J48 doesent overlap in the dotplot with the other models because of how superior it was. We can't say they same form KNN and CART, even though KNN obtained better results it still overlaps with CART even at a confidence of 0.95 it still overlaps I conclude the performance is not statistically significant.

Our water dataset contains twice the size of water1 with 5,631 rows, I would have assumed this to result in better training hence better results but we can see a extreme overlap between each of the twin methods performed for each dataset. This means that the difference in size shows not to be of significant when it comes to performance.

KNN performed much lower than J48 but KNN is not a divide and conquer algorithm but an instance base algorithm. Under different circumstances KNN would have been more inclined to score better with a dataset that has been pre-processed by normalization, standardization and undergo a one hot encoding. Under this new dataset I assume KNN would have been more suited than a tree classifier which work better when dealing with nominal values.

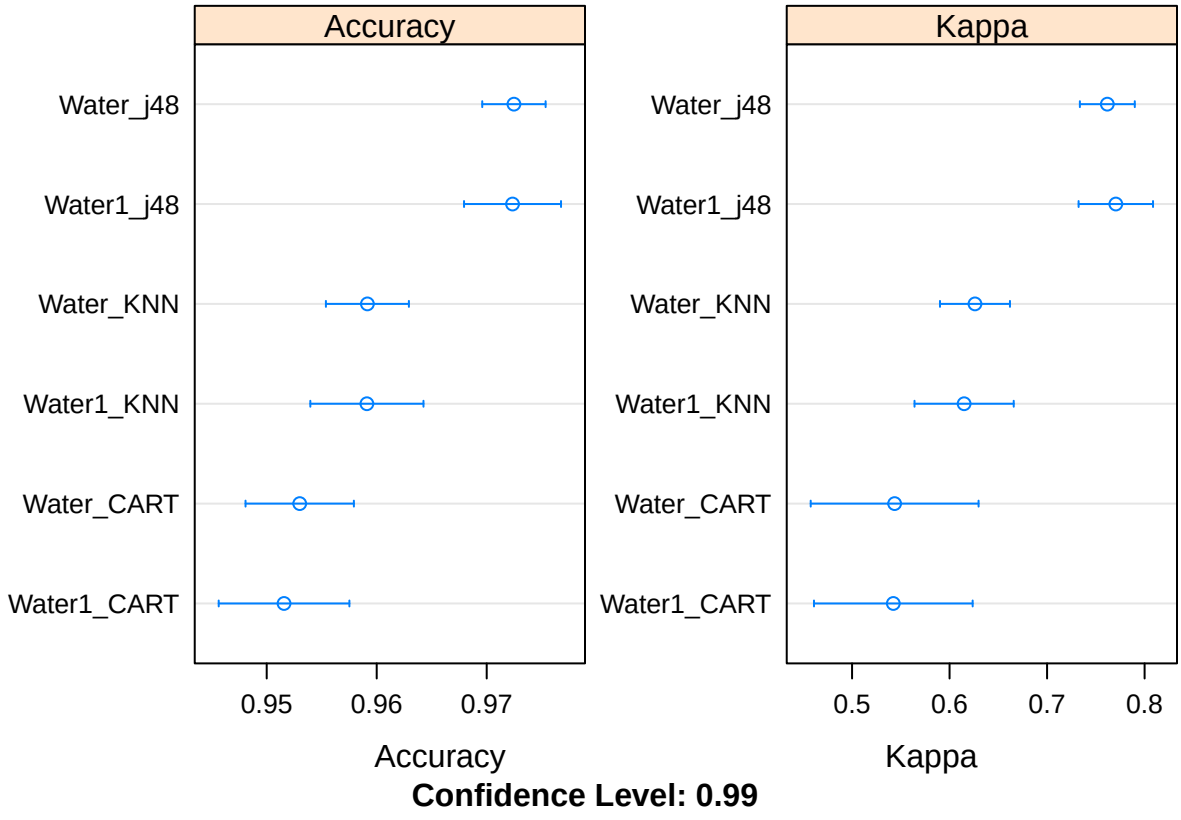
```
# collect resamples
results = resamples(list(Water1_j48=J48Water1,Water1_CART=cartWater1,
                         Water1_KNN=knnWater1, Water_j48=J48Water,
                         Water_CART=cartWater, Water_KNN=knnWater))
# show accuracy and kappa details
results
```

```
##
## Call:
## resamples.default(x = list(Water1_j48 = J48Water1, Water1_CART =
##  cartWater1, Water1_KNN = knnWater1, Water_j48 = J48Water, Water_CART
##  = cartWater, Water_KNN = knnWater))
##
## Models: Water1_j48, Water1_CART, Water1_KNN, Water_j48, Water_CART, Water_KNN
## Number of resamples: 30
## Performance metrics: Accuracy, Kappa
## Time estimates for: everything, final model fit
```

```
summary(results)
```

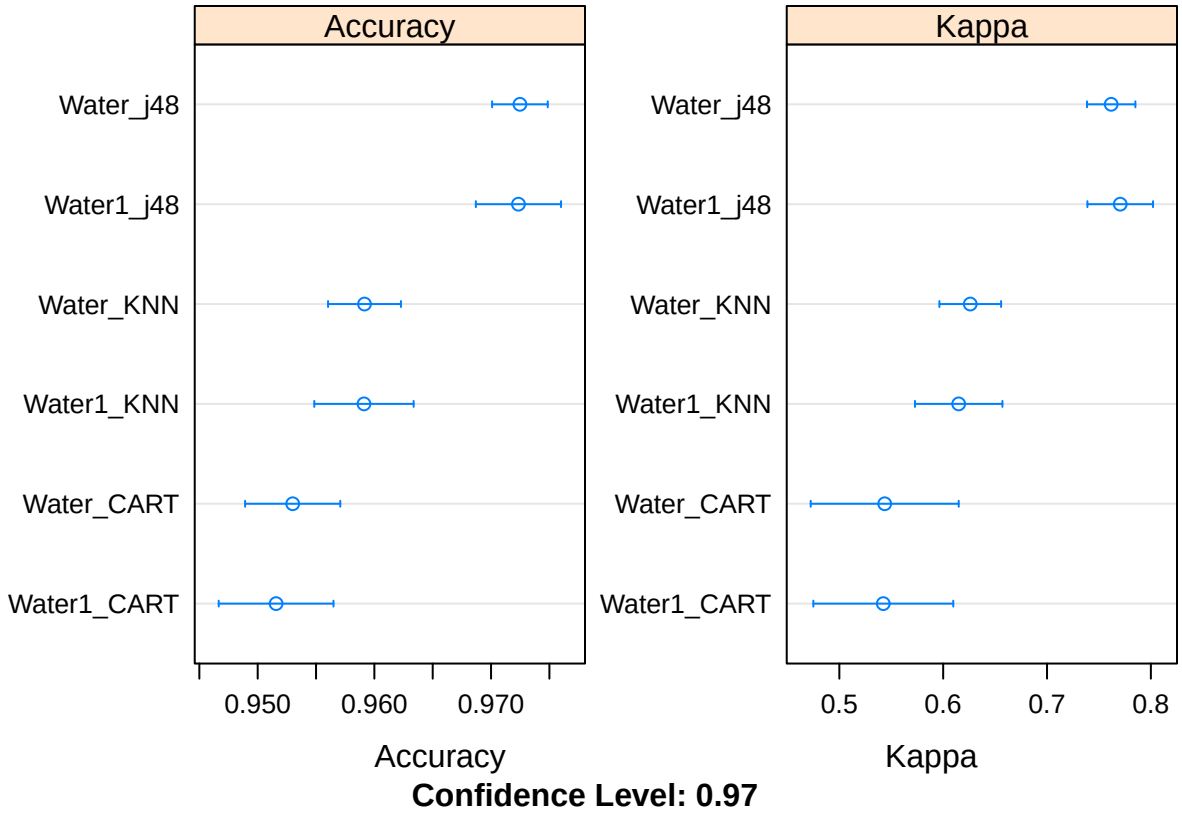
```
##
## Call:
## summary.resamples(object = results)
##
## Models: Water1_j48, Water1_CART, Water1_KNN, Water_j48, Water_CART, Water_KNN
## Number of resamples: 30
##
## Accuracy
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Water1_j48  0.9555556 0.9626984 0.9730611 0.9723475 0.9800955 0.9872611    0
## Water1_CART 0.9331210 0.9395385 0.9523102 0.9515771 0.9618744 0.9682540    0
## Water1_KNN  0.9428571 0.9522672 0.9555556 0.9591089 0.9681529 0.9808917    0
## Water_j48   0.9609236 0.9680284 0.9724689 0.9724746 0.9769094 0.9840142    0
## Water_CART  0.9361702 0.9431868 0.9555950 0.9530009 0.9609236 0.9715808    0
## Water_KNN   0.9378330 0.9560792 0.9609583 0.9591542 0.9626998 0.9698046    0
##
## Kappa
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Water1_j48  0.6311668 0.7187543 0.7766332 0.7704837 0.8328058 0.8947015    0
## Water1_CART 0.1414063 0.3835774 0.5850716 0.5423229 0.6788157 0.7365772    0
## Water1_KNN  0.4502754 0.5302045 0.5937730 0.6149224 0.7012570 0.8255556    0
## Water_j48   0.6515109 0.7273389 0.7608451 0.7617824 0.8058174 0.8630281    0
## Water_CART  0.1691628 0.4242175 0.6133503 0.5436650 0.6698253 0.7598315    0
## Water_KNN   0.4532919 0.5862078 0.6490961 0.6260564 0.6774372 0.7299976    0
```

```r
# Building dotplot with different confidence levels
scales = list(x=list(relation="free"), y=list(relation= "free"))
dotplot(results, scales=scales, conf.level = 0.99)
```

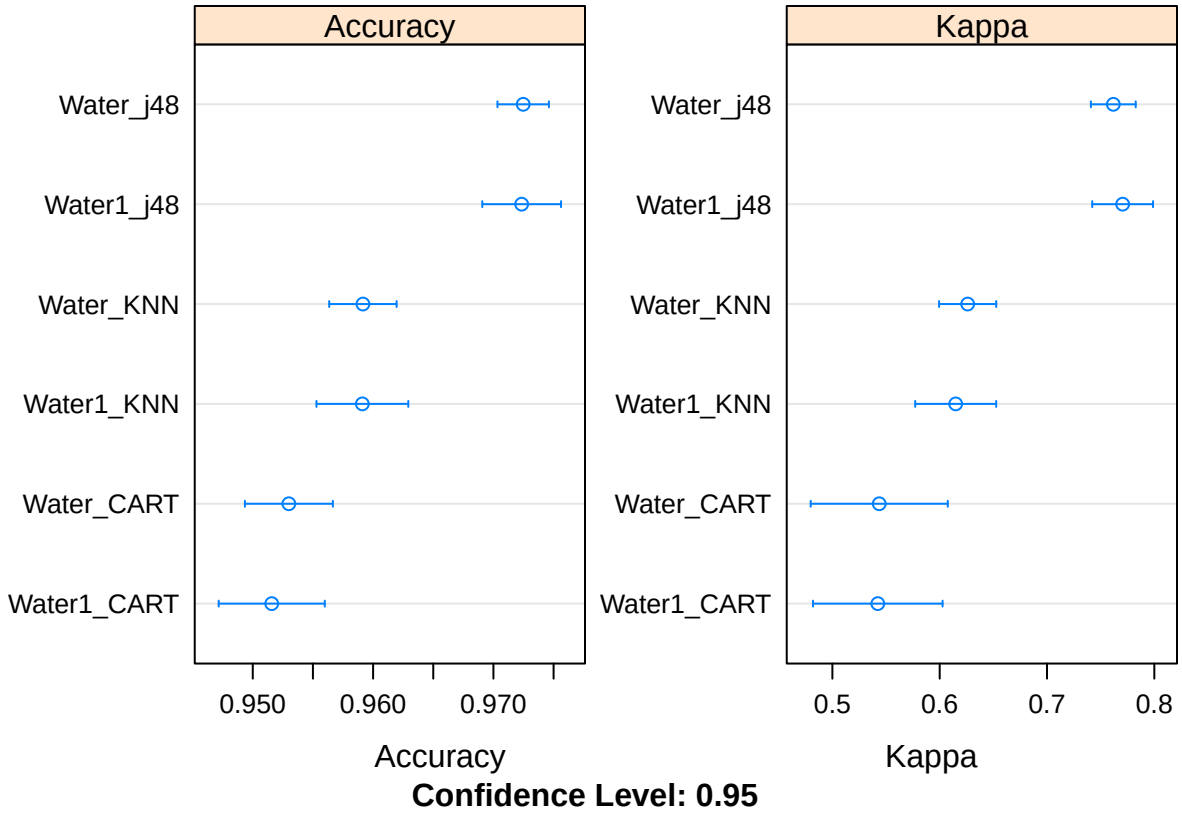


Confidence Level: 0.99

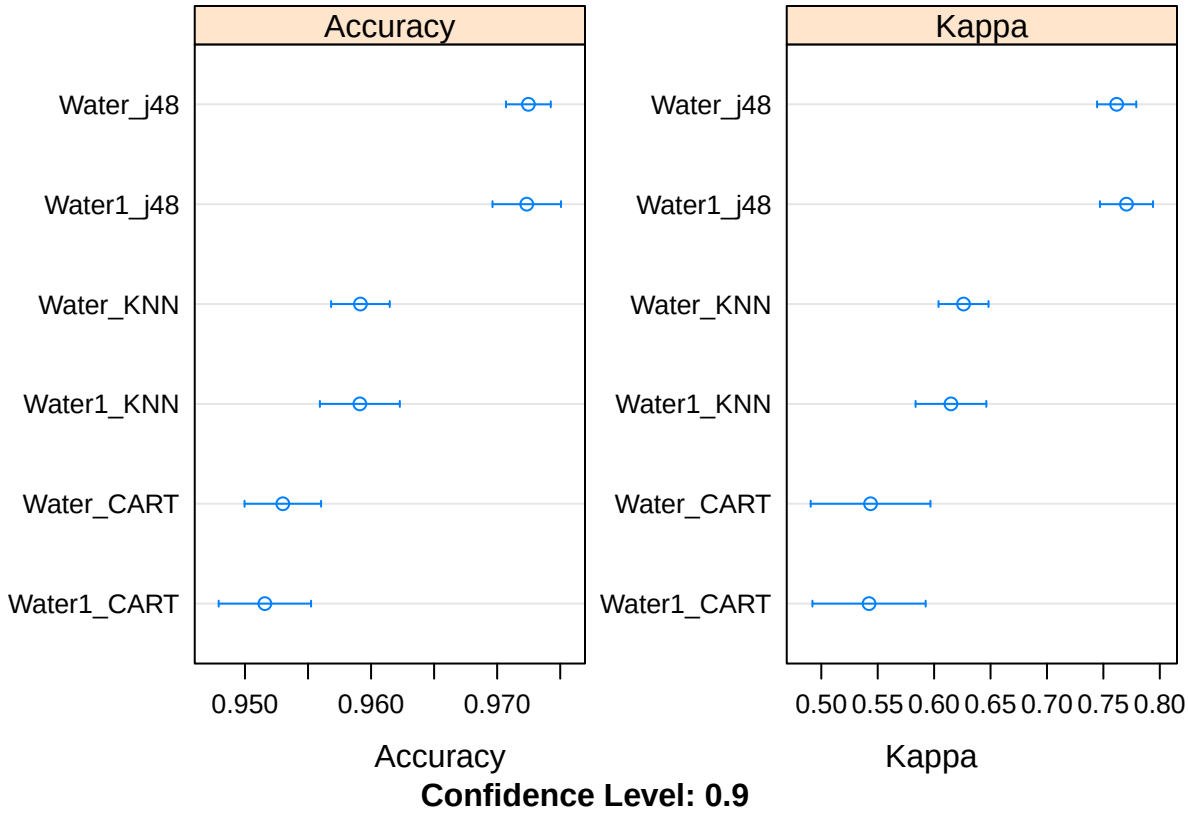
```r
dotplot(results, scales=scales, conf.level = 0.97)
```




```
dotplot(results, scales=scales, conf.level = 0.95)
```




```
dotplot(results, scales=scales, conf.level = 0.90)
```



Testing datasets

It seems like the CART algorithm had the highest accuracy when predicting LW 99.6%, this of course comes with a big problem which is that 150 of the instances that belong to other categories where also classified as LW. CART wasn't able to obtain any true positives when it came to GW and TW and also obtained a unreliable score of 28% for RW. In contrast KNN had a slightly harder chance at predicting LW with 98.8%, still this is a much more desirable model since the number of false positives wasn't as high as CART KNN predicted (GW = 78.5%, RW = 55.4%, TW = 0%.) J48 on the other hand didn't just got the second best score at predicting LW with 99.4% and a high 80% at RW. J48 was the only algorithm that predicted instances of TW right but did not scored as high in GW 62% as KNN. I can conclude that KNN seems to be better at predicting Ground water and J48 at predicting all other categories. J48 proves to be the most balanced model obtaining the highes overall score of all. J48 = Accuracy : 0.9767 KNN = Accuracy : 0.963 CART = Accuracy : 0.9405

```
#Testing J48
J48TestRes = predict(J48Water, newdata = testWater, type="raw")
confusionMatrix(J48TestRes, testWater$WBCategory)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction   GW   LW   RW   TW
##         GW   35    0    0    0
##         LW   21 2306   22    0
##         RW    0   12   88    3
```

```
##         TW    0    0    0    2
##
## Overall Statistics
##
##                Accuracy : 0.9767
##                  95% CI : (0.97, 0.9823)
##     No Information Rate : 0.9313
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8041
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: GW Class: LW Class: RW Class: TW
## Sensitivity            0.62500    0.9948   0.80000 0.4000000
## Specificity            1.00000    0.7485   0.99369 1.0000000
## Pos Pred Value         1.00000    0.9817   0.85437 1.0000000
## Neg Pred Value         0.99144    0.9143   0.99078 0.9987937
## Prevalence             0.02250    0.9313   0.04419 0.0020088
## Detection Rate         0.01406    0.9265   0.03536 0.0008035
## Detection Prevalence   0.01406    0.9438   0.04138 0.0008035
## Balanced Accuracy      0.81250    0.8717   0.89685 0.7000000
```

```
#Testing CART
cartWaterTestRes = predict(cartWater, newdata = testWater, type="raw")
confusionMatrix(cartWaterTestRes, testWater$WBCategory)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   GW   LW   RW   TW
##         GW    0    0    0    0
##         LW   56 2310   79    3
##         RW    0    8   31    2
##         TW    0    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.9405
##                  95% CI : (0.9305, 0.9495)
##     No Information Rate : 0.9313
##     P-Value [Acc > NIR] : 0.03518
##
##                   Kappa : 0.2863
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: GW Class: LW Class: RW Class: TW
## Sensitivity             0.0000    0.9965   0.28182  0.000000
## Specificity             1.0000    0.1930   0.99580  1.000000
```

```
## Pos Pred Value              NaN    0.9436   0.75610       NaN
## Neg Pred Value           0.9775    0.8049   0.96773  0.997991
## Prevalence               0.0225    0.9313   0.04419  0.002009
## Detection Rate           0.0000    0.9281   0.01245  0.000000
## Detection Prevalence     0.0000    0.9835   0.01647  0.000000
## Balanced Accuracy        0.5000    0.5948   0.63881  0.500000
```

```
#Testing KNN
knnWaterTestRes = predict(knnWater, newdata = testWater, type="raw")
confusionMatrix(knnWaterTestRes, testWater$WBCategory)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction   GW   LW   RW   TW
##         GW   44   14    1    0
##         LW   12 2292   48    2
##         RW    0   12   61    3
##         TW    0    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.963
##                  95% CI : (0.9549, 0.9701)
##     No Information Rate : 0.9313
##     P-Value [Acc > NIR] : 6.606e-12
##
##                   Kappa : 0.685
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: GW Class: LW Class: RW Class: TW
## Sensitivity            0.78571    0.9888   0.55455  0.000000
## Specificity            0.99383    0.6374   0.99369  1.000000
## Pos Pred Value         0.74576    0.9737   0.80263       NaN
## Neg Pred Value         0.99506    0.8074   0.97969  0.997991
## Prevalence             0.02250    0.9313   0.04419  0.002009
## Detection Rate         0.01768    0.9209   0.02451  0.000000
## Detection Prevalence   0.02370    0.9458   0.03053  0.000000
## Balanced Accuracy      0.88977    0.8131   0.77412  0.500000
```

Clustering

Pre-processing

```
#binarising nominal attributes - one-hot encoding
# make a copy
noClass = water
```

```r
# remove the class - it is not transformed
noClass$WBCategory = NULL

set.seed(34)
#binarise nominal attributes - one-hot encoding
binaryVars = dummyVars(~ ., data = noClass)
newWater = predict(binaryVars, newdata = noClass)

# add the class to the binarised dataset
binWater = cbind(newWater, water[1])
# check the result.
# View(binWeather) could have been used instead but it does not appear in the knitted document.
# previewing dataset
head(binWater, 10)
```

```
##    determinandC.CAS_14798-03-9 determinandC.CAS_7723-14-0
## 1                            0                          0
## 2                            0                          0
## 3                            0                          0
## 4                            0                          1
## 5                            0                          0
## 6                            0                          0
## 7                            0                          1
## 8                            0                          0
## 9                            0                          1
## 10                           0                          1
##    determinandC.EEA_3131-01-9 determinandC.EEA_3132-01-2 NSamples minValue
## 1                           1                          0        5  37.2000
## 2                           0                          1        4  10.5000
## 3                           1                          0        2  74.8570
## 4                           0                          0        6   0.0050
## 5                           1                          0        1  78.5000
## 6                           1                          0        1  95.5000
## 7                           0                          0        1   0.0040
## 8                           0                          1        2   4.4000
## 9                           0                          0        7   0.0090
## 10                          0                          0        1   0.0047
##    meanValue maxValue        sd method.EN 25813:1992 method.SFS 3026
## 1    60.4320  81.6000 20.174000                    0               0
## 2    10.9000  11.3000  3.916616                    1               0
## 3    74.9280  75.0000  0.101000                    0               0
## 4     0.0060   0.0080  0.000000                    0               1
## 5    78.5000  78.5000  3.916616                    0               0
## 6    95.5000  95.5000  3.916616                    0               0
## 7     0.0040   0.0040  3.916616                    0               1
## 8     6.4667   8.5333  2.922700                    1               0
## 9     0.0130   0.0170  0.002000                    0               1
## 10    0.0047   0.0047  3.916616                    0               1
##    method.SFS 3032 + EN ISO 11732 method.SFS 3040 method.SFS 3040 SFS-EN 25813
## 1                               0               1                           0
## 2                               0               0                           0
## 3                               0               1                           0
## 4                               0               0                           0
```

```
## 5                              0               1                              0
## 6                              0               1                              0
## 7                              0               0                              0
## 8                              0               0                              0
## 9                              0               0                              0
## 10                             0               0                              0
##    WBCategory
## 1          LW
## 2          RW
## 3          LW
## 4          LW
## 5          LW
## 6          LW
## 7          LW
## 8          LW
## 9          LW
## 10         LW
```

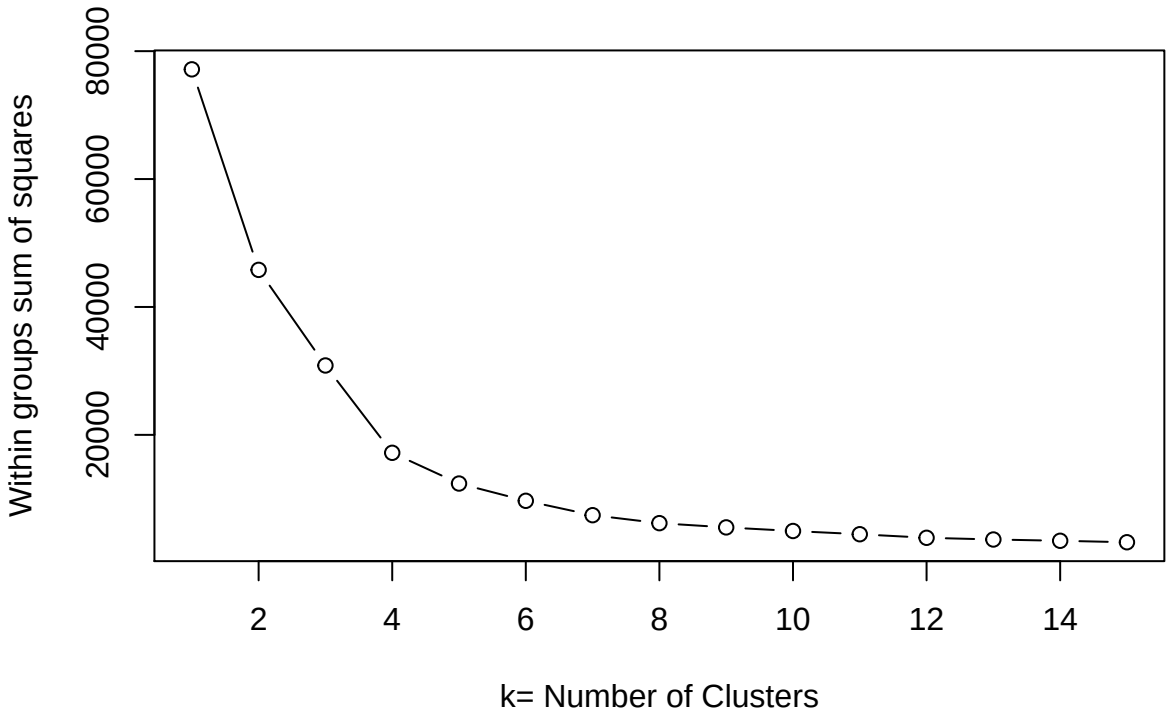
Finding ideal number of clusters

By computing an elbow curve we can see a highe reduction in variation in the ranges of 4-5. I have selected K = 4 because is inside the range and it accurately matches the 4 categories in our class.

```
# Data standardization
pca_water = preProcess(binWater,
                       method = c("center", "scale", "pca"))
# checking standardlised data
waterTreated = predict(pca_water, newdata = binWater)
summary(waterTreated)
```

```
##  WBCategory      PC1               PC2                PC3
##  GW: 120    Min.   :-6.381   Min.   :-2.27126   Min.   :-1.4259
##  LW:5259    1st Qu.:-2.968   1st Qu.:-2.15626   1st Qu.:-0.9635
##  RW: 250    Median : 1.192   Median :-0.07256   Median :-0.8040
##  TW:   2    Mean   : 0.000   Mean   : 0.00000   Mean   : 0.0000
##             3rd Qu.: 1.767   3rd Qu.: 1.68605   3rd Qu.: 0.1275
##             Max.   : 1.926   Max.   : 2.37232   Max.   : 3.8308
##       PC4                PC5                PC6
##  Min.   :-1.4618   Min.   :-7.0389   Min.   :-7.72053
##  1st Qu.:-0.7656   1st Qu.:-0.3875   1st Qu.:-0.18956
##  Median :-0.1775   Median : 0.2110   Median : 0.02415
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.: 0.1313   3rd Qu.: 0.4324   3rd Qu.: 0.32609
##  Max.   : 5.6346   Max.   : 2.4021   Max.   : 2.77131
```

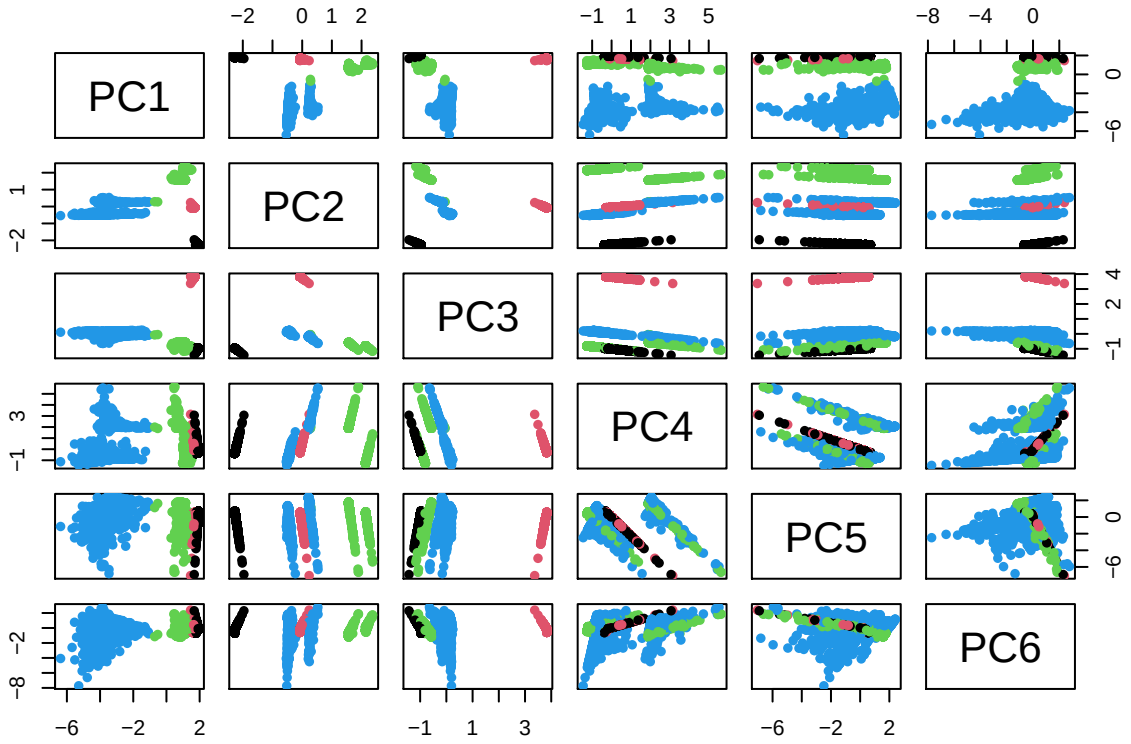
```
set.seed(34)
# Using the Elbow approach to find the number of K to use for clsutering.
wss = (nrow(waterTreated[, -1])-1)*sum(apply(waterTreated[,-1],2,var))


for (i in 2:15)
  wss[i] = sum(kmeans(waterTreated[, -1], centers=i, nstart=100, iter.max=1000)$withinss)
plot(1:15, wss, type="b", xlab="k= Number of Clusters", ylab="Within groups sum of squares")
```



Clustering algorithm

I chose K-means as the clustering algorithm because I did not think the extra flexibility that K-medoids was important. This dataset has no noise and has been preprocess to fit K-means appropriately. K-means is then easier to compute hence faster.

```
set.seed(34)
# Building clusters
km = kmeans(waterTreated[, -1], 4, nstart=25, iter.max=1000)
# Plotting the different clusters
plot(waterTreated[, -1], col=km$clust, pch=16)
```



Clustering results

The results found in the chart above show very good results mainly from the left side of the plot (i.e. PC2-PC1, PC3-PC2, PC5-PC2) this clusters seems to have a strong separation between each other and found this to be the best results for the clustering of this experiment. In contrast we can see PC4-PC6 and other similar results containing and immense overlap in between the classes and finding this results to be the least useful for our cluster model.

-