PREDICTIVE ANALYTICS

ALY6020, SPRING 2020

MODULE 1 PROJECT ASSIGNMENT

WEEK 1: CLASSIFICATION USING NEAREST NEIGHBORS

SUBMITTED BY: SHIVANI ADSAR

NUID: 001399374

SUBMITTED TO: PROF. NA YU

DATE: 04/11/2020

Introduction

The assignment focusses on providing practical knowledge through the use of K Nearest Neighbor machine learning algorithm. The classification algorithm is efficient and robust, widely used in medical industry, theft and fraud detection etc. In this assignment, The K Nearest Neighbor is performed on the Breast Cancer and Abalone datasets for predictions. [1]

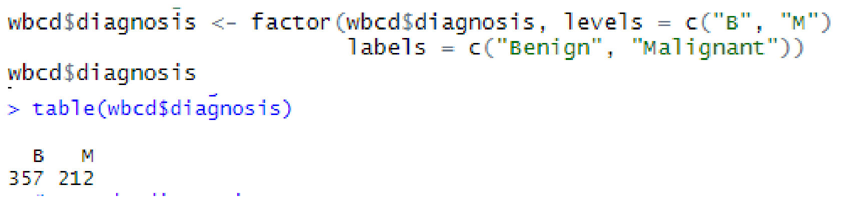
Analysis

Breast Cancer Dataset

- The Breast Cancer dataset has been taken from the UCI Machine Learning Repository. This dataset has 32 attributes and 569 instances. The data consists of the dimensions of cancer cells in women and these attributes have helped in predicting the type of cancer in patients.
- In order to perform the classification of the cancer types as Benign and Malignant, we have used the K Nearest Neighbors classification algorithm. [2]

Importing the Dataset

- The Breast Cancer dataset has been imported initially into a dataframe. Using the "str" function, the structure of the dataset has been analysed.
- We have selected the feature, "diagnosis" for understanding the number of benign and malignant types and further factorized by labelling them.



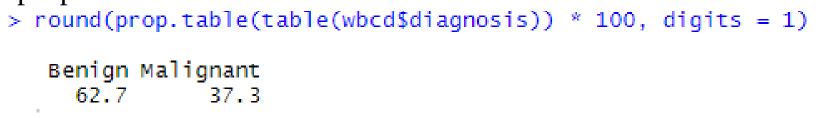
```
wbcd$diagnosis <- factor(wbcd$diagnosis, levels = c("B", "M")
                         labels = c("Benign", "Malignant"))
wbcd$diagnosis
> table(wbcd$diagnosis)

  B   M
357 212
```

We can observe that there are 357 records having Benign cancer while 212 records have Malignant cancer.

Fig.1: Factorization for the Diagnosis

Analysis on Dataset

- The percentages of the labelled values, Benign and Malignant were analysed using the "prop.table" function.

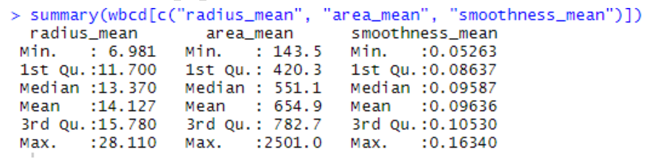


```
> round(prop.table(table(wbcd$diagnosis)) * 100, digits = 1)

   Benign Malignant
     62.7      37.3
```

As seen, the percentage of records having Benign cancer is 62.7 % and for Malignant, is 37.3 %.

Fig.2: Percentages of Labelled Values

- We have selected the three variables to understand and analyze certain numeric parameters such as the radius, area and smoothness. [2]



```
> summary(wbcd[c("radius_mean", "area_mean", "smoothness_mean")])
  radius_mean       area_mean      smoothness_mean
 Min.   : 6.981   Min.   : 143.5   Min.   :0.05263
 1st Qu.:11.700   1st Qu.: 420.3   1st Qu.:0.08637
 Median :13.370   Median : 551.1   Median :0.09587
 Mean   :14.127   Mean   : 654.9   Mean   :0.09636
 3rd Qu.:15.780   3rd Qu.: 782.7   3rd Qu.:0.10530
 Max.   :28.110   Max.   :2501.0   Max.   :0.16340
```

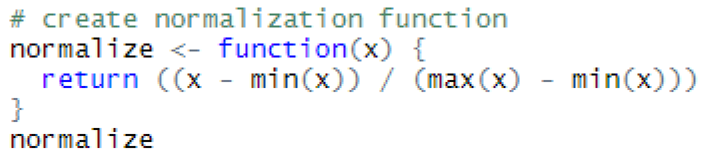
Fig.3: Summary of numeric parameters selected

We know that KNN works primarily on the numeric measurement of datapoints. We notice, the values of smoothness mean vary drastically when compared to area_mean and radius _mean.

- As we can observe that the values of smoothness_mean are varying from 0.05 to 0.16 while, radius_mean varies from 6.98 to 28.11 and area_mean from 143.5 to 2501.0. We can see that the values are not uniform which may give biased results and inaccurate predictions.

Normalisation

- Hence, we will normalize the data in order to have standardized and uniform values. Using the normalization function which will use the difference of the variable from the minimum value and divide by the range, the values will be standardized.

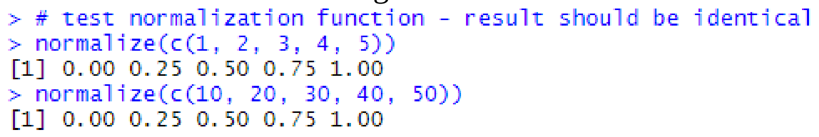


```
# create normalization function
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
normalize
```

Fig.4: Normalization

Using the normalization function which will use the difference of the variable from the minimum value and divide by the range, the values will be standardized.

- We have checked the working of normalization function.

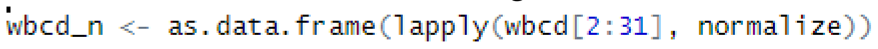


```
> # test normalization function - result should be identical
> normalize(c(1, 2, 3, 4, 5))
[1] 0.00 0.25 0.50 0.75 1.00
> normalize(c(10, 20, 30, 40, 50))
[1] 0.00 0.25 0.50 0.75 1.00
```

Fig.5: Verification Normalization Function

We can see that the values have been normalized for different set of input parameters.

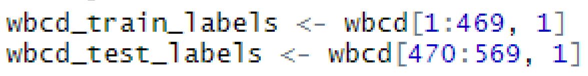
- In order to perform the normalization function for all the values, we have used the “lapply” function which will return the same length for all the values.



```
wbcd_n <- as.data.frame(lapply(wbcd[2:31], normalize))
```

Fig.6: Normalisation for all values

- The data has been divided into Training and Testing Data for better prediction.



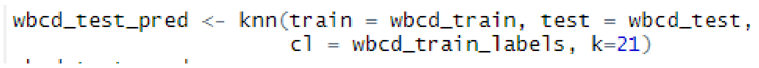
```
wbcd_train_labels <- wbcd[1:469, 1]
wbcd_test_labels <- wbcd[470:569, 1]
```

Fig.7: Training and Testing Datasets

As seen, the first 469 columns have been sed for training the dataset and this is tested on the remaining columns.

K-Nearest Neighbor

- After the initial data analysis is completed, the training for KNN classifier is performed. The “class” package is used which classifies the datapoints for KNN algorithm. The algorithm selects the nearest neighbors based on the Euclidean distance of each datapoint and as per the number of clusters specified.[1]

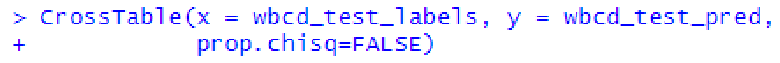


```
wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test,
                      cl = wbcd_train_labels, k=21)
```

Fig.8: KNN Classifier

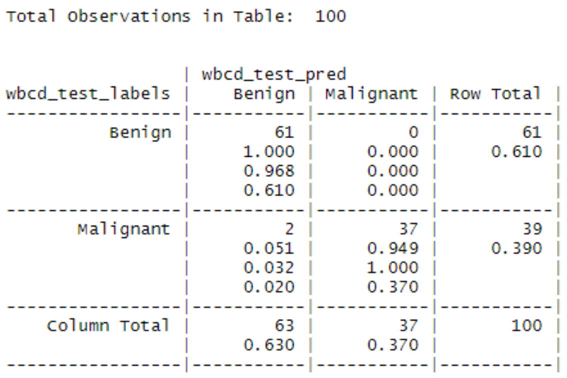
We can built the KNN classifier on the training and testing data considering 21 clusters.

- Since the training data is built upon 469 columns, we have used the number of clusters as 21 which is almost equivalent to the square root of that number.
- We have checked the performance of the model by comparing with the predicted values and test values in the testing dataset. Hence, the Cross Table function included in the “gmodels” library will help in comparisons of two vectors.



```
> CrossTable(x = wbcd_test_labels, y = wbcd_test_pred,
+            prop.chisq=FALSE)
```

Fig.9: Cross Table



```
Total Observations in Table:  100 

 
                 | wbcd_test_pred 
wbcd_test_labels |    Benign | Malignant | Row Total | 
-----------------|-----------|-----------|-----------|
          Benign |        61 |         0 |        61 | 
                 |     1.000 |     0.000 |     0.610 | 
                 |     0.968 |     0.000 |           | 
                 |     0.610 |     0.000 |           | 
-----------------|-----------|-----------|-----------|
       Malignant |         2 |        37 |        39 | 
                 |     0.051 |     0.949 |     0.390 | 
                 |     0.032 |     1.000 |           | 
                 |     0.020 |     0.370 |           | 
-----------------|-----------|-----------|-----------|
    Column Total |        63 |        37 |       100 | 
                 |     0.630 |     0.370 |           | 
-----------------|-----------|-----------|-----------|
```

Fig. 10: Confusion Matrix

True Negative: The top right value of 61 has been accurately been identified as Benign out of 100 values.

True Positive: The values of 37 of 100 was correctly identified as Malignant.

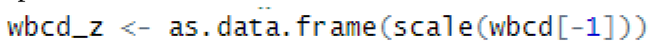
False Negative: The value of 2 shows that the predicted value was Benign while the cancer was malignant.

False Positive: The model precited 0 values for tumors that were malignant but actually benign

- We can see that the model predicted 2 out of 100 values incorrectly and the accuracy was seen to be 98%.
- The improvement of prediction can be very well performed by Normalisation. However, normalization, does not compress the middle values for uniformity always. Hence, outliers are not taking into account as the extreme values of compressed. Therefore, Z-score standardization can be useful in such cases.

Z-Score Standardization

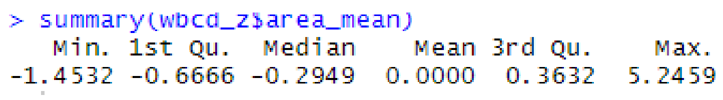
- Z-score standardization uses the scale() function to re-scale the values for better optimization.



```
wbcd_z <- as.data.frame(scale(wbcd[-1]))
```

Fig.11: Re-scaling using Z-score

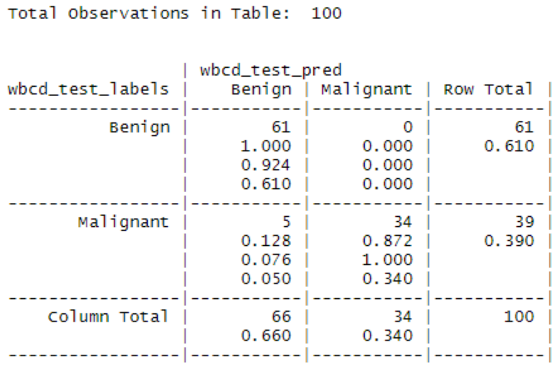
The Z-score value should be 0 as, we can see the mean value is 0 in summary.



```
> summary(wbcd_z$area_mean)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
-1.4532 -0.6666 -0.2949  0.0000  0.3632  5.2459 
```

Fig.12: Summary of Z-Score function

- As performed earlier, we will be dividing the data into testing and training data labels, then perform the KNN classifier and built the cross table.

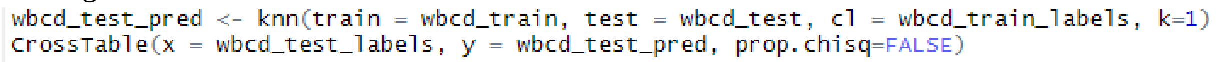


```
Total Observations in Table:  100 

 
                 | wbcd_test_pred 
wbcd_test_labels |    Benign | Malignant | Row Total | 
-----------------|-----------|-----------|-----------|
          Benign |        61 |         0 |        61 | 
                 |     1.000 |     0.000 |     0.610 | 
                 |     0.924 |     0.000 |           | 
                 |     0.610 |     0.000 |           | 
-----------------|-----------|-----------|-----------|
       Malignant |         5 |        34 |        39 | 
                 |     0.128 |     0.872 |     0.390 | 
                 |     0.076 |     1.000 |           | 
                 |     0.050 |     0.340 |           | 
-----------------|-----------|-----------|-----------|
    Column Total |        66 |        34 |       100 | 
                 |     0.660 |     0.340 |           | 
-----------------|-----------|-----------|-----------|
```

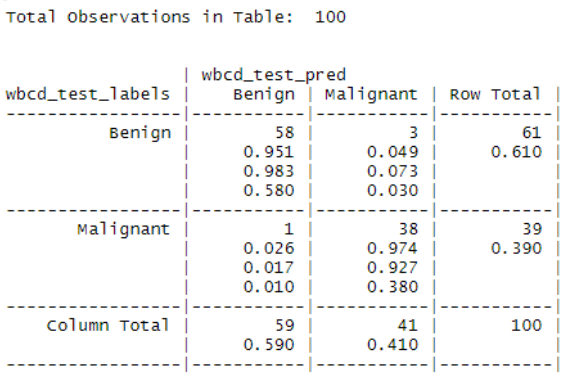
We can see that, the False Negative values were incorrectly classified as 5, Hence the accuracy has reduced to 95%.

Fig.13: Confusion Matrix using Z-Score

- Testing for several values of K has been done as follows on the same function:

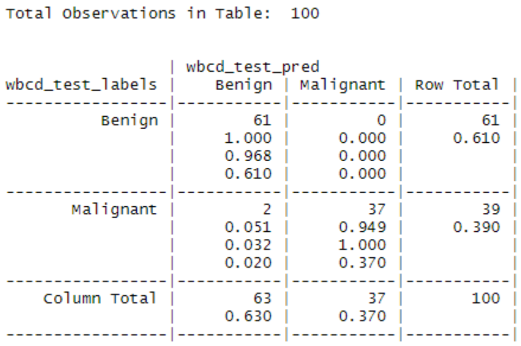


```
wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test, cl = wbcd_train_labels, k=1)
CrossTable(x = wbcd_test_labels, y = wbcd_test_pred, prop.chisq=FALSE)
```



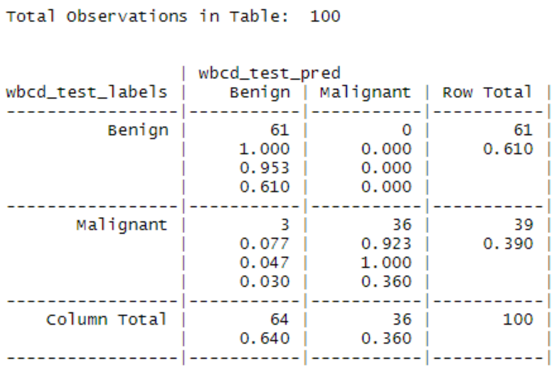
```
Total Observations in Table:  100 

 
                 | wbcd_test_pred 
wbcd_test_labels |    Benign | Malignant | Row Total | 
-----------------|-----------|-----------|-----------|
          Benign |        58 |         3 |        61 | 
                 |     0.951 |     0.049 |     0.610 | 
                 |     0.983 |     0.073 |           | 
                 |     0.580 |     0.030 |           | 
-----------------|-----------|-----------|-----------|
       Malignant |         1 |        38 |        39 | 
                 |     0.026 |     0.974 |     0.390 | 
                 |     0.017 |     0.927 |           | 
                 |     0.010 |     0.380 |           | 
-----------------|-----------|-----------|-----------|
    Column Total |        59 |        41 |       100 | 
                 |     0.590 |     0.410 |           | 
-----------------|-----------|-----------|-----------|
```

K=1



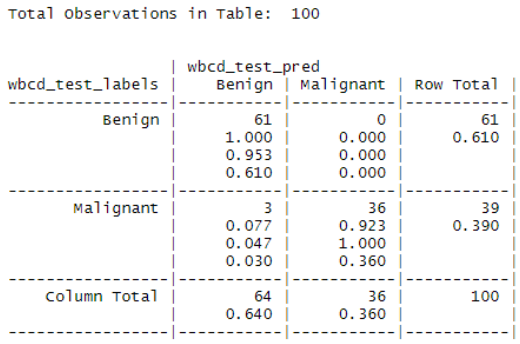
```
Total Observations in Table:  100 

 
                 | wbcd_test_pred 
wbcd_test_labels |    Benign | Malignant | Row Total | 
-----------------|-----------|-----------|-----------|
          Benign |        61 |         0 |        61 | 
                 |     1.000 |     0.000 |     0.610 | 
                 |     0.968 |     0.000 |           | 
                 |     0.610 |     0.000 |           | 
-----------------|-----------|-----------|-----------|
       Malignant |         2 |        37 |        39 | 
                 |     0.051 |     0.949 |     0.390 | 
                 |     0.032 |     1.000 |           | 
                 |     0.020 |     0.370 |           | 
-----------------|-----------|-----------|-----------|
    Column Total |        63 |        37 |       100 | 
                 |     0.630 |     0.370 |           | 
-----------------|-----------|-----------|-----------|
```

K=5



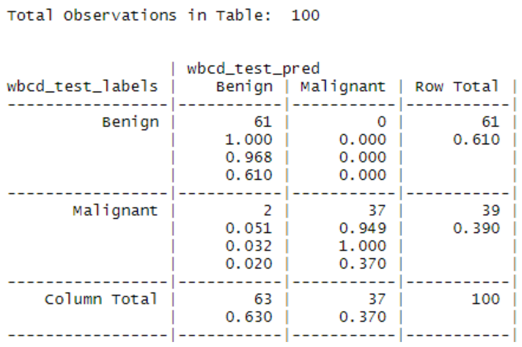
```
Total Observations in Table:  100 

 
                 | wbcd_test_pred 
wbcd_test_labels |    Benign | Malignant | Row Total | 
-----------------|-----------|-----------|-----------|
          Benign |        61 |         0 |        61 | 
                 |     1.000 |     0.000 |     0.610 | 
                 |     0.953 |     0.000 |           | 
                 |     0.610 |     0.000 |           | 
-----------------|-----------|-----------|-----------|
       Malignant |         3 |        36 |        39 | 
                 |     0.077 |     0.923 |     0.390 | 
                 |     0.047 |     1.000 |           | 
                 |     0.030 |     0.360 |           | 
-----------------|-----------|-----------|-----------|
    Column Total |        64 |        36 |       100 | 
                 |     0.640 |     0.360 |           | 
-----------------|-----------|-----------|-----------|
```

K=11



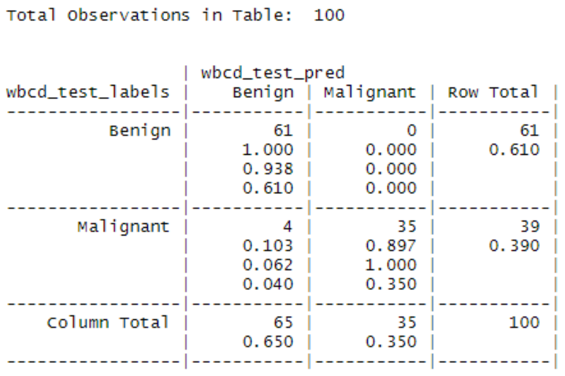
```
Total Observations in Table:  100 

 
                 | wbcd_test_pred 
wbcd_test_labels |    Benign | Malignant | Row Total | 
-----------------|-----------|-----------|-----------|
          Benign |        61 |         0 |        61 | 
                 |     1.000 |     0.000 |     0.610 | 
                 |     0.953 |     0.000 |           | 
                 |     0.610 |     0.000 |           | 
-----------------|-----------|-----------|-----------|
       Malignant |         3 |        36 |        39 | 
                 |     0.077 |     0.923 |     0.390 | 
                 |     0.047 |     1.000 |           | 
                 |     0.030 |     0.360 |           | 
-----------------|-----------|-----------|-----------|
    Column Total |        64 |        36 |       100 | 
                 |     0.640 |     0.360 |           | 
-----------------|-----------|-----------|-----------|
```

K=15



```
Total Observations in Table:  100

                 | wbcd_test_pred
wbcd_test_labels |    Benign | Malignant | Row Total |
-----------------|-----------|-----------|-----------|
          Benign |        61 |         0 |        61 |
                 |     1.000 |     0.000 |     0.610 |
                 |     0.968 |     0.000 |           |
                 |     0.610 |     0.000 |           |
-----------------|-----------|-----------|-----------|
       Malignant |         2 |        37 |        39 |
                 |     0.051 |     0.949 |     0.390 |
                 |     0.032 |     1.000 |           |
                 |     0.020 |     0.370 |           |
-----------------|-----------|-----------|-----------|
    Column Total |        63 |        37 |       100 |
                 |     0.630 |     0.370 |           |
-----------------|-----------|-----------|-----------|
```

K=21



```
Total Observations in Table:  100

                 | wbcd_test_pred
wbcd_test_labels |    Benign | Malignant | Row Total |
-----------------|-----------|-----------|-----------|
          Benign |        61 |         0 |        61 |
                 |     1.000 |     0.000 |     0.610 |
                 |     0.938 |     0.000 |           |
                 |     0.610 |     0.000 |           |
-----------------|-----------|-----------|-----------|
       Malignant |         4 |        35 |        39 |
                 |     0.103 |     0.897 |     0.390 |
                 |     0.062 |     1.000 |           |
                 |     0.040 |     0.350 |           |
-----------------|-----------|-----------|-----------|
    Column Total |        65 |        35 |       100 |
                 |     0.650 |     0.350 |           |
-----------------|-----------|-----------|-----------|
```

K=27

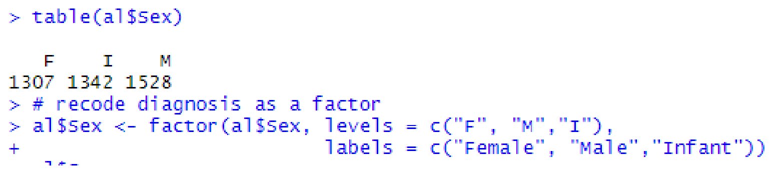
- As we can see, the algorithm gives more accurate results for K=5 and K=21 with accuracy of 98% as compared to other values.

Abalone Dataset

- The Abalone dataset is used to predict the gender and age of the shell on the basis of certain characteristics as sex, length, height, number of rings and weight. The dataset has 4177 instances and 8 attributes. [1]

Importing the Dataset

- The Abalone dataset has been imported initially into a dataframe. Using the "str" function, the structure of the dataset has been analysed.
- We have selected the feature, "Sex" for understanding the gender of shells and further factorized by labelling them.



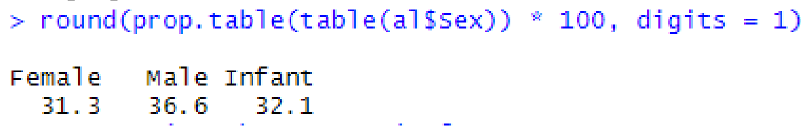
```
> table(al$sex)

   F    I    M 
1307 1342 1528 
> # recode diagnosis as a factor
> al$sex <- factor(al$sex, levels = c("F", "M","I"),
+                          labels = c("Female", "Male","Infant"))
```

We can observe that there are 1307 female, 1342 Infants and 1528 Males.

Fig.1: Factorization for the Gender

Analysis on Dataset

- The percentages of the labelled values, Males, Females and Infants were analysed using the "prop.table" function.[2]

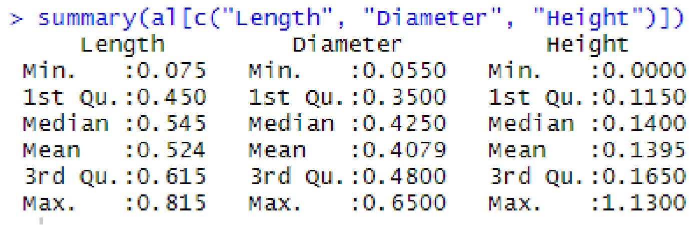


```
> round(prop.table(table(al$sex)) * 100, digits = 1)

Female   Male Infant 
  31.3   36.6   32.1 
```

As seen, the percentage of records having Female are 31.3%, Male are 36.6% and Infants are 32.1 %.

Fig.2: Percentages of Labelled Values

- We have selected the three variables to understand and analyze certain numeric parameters such as the Length, Diameter and Height.



```
> summary(al[c("Length", "Diameter", "Height")])
     Length          Diameter          Height
 Min.   :0.075   Min.   :0.0550   Min.   :0.0000
 1st Qu.:0.450   1st Qu.:0.3500   1st Qu.:0.1150
 Median :0.545   Median :0.4250   Median :0.1400
 Mean   :0.524   Mean   :0.4079   Mean   :0.1395
 3rd Qu.:0.615   3rd Qu.:0.4800   3rd Qu.:0.1650
 Max.   :0.815   Max.   :0.6500   Max.   :1.1300
```

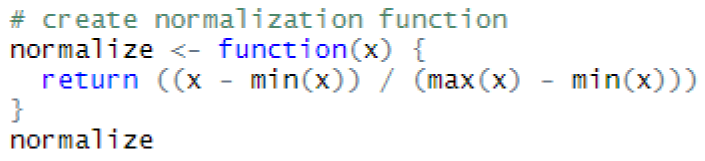
Fig.3: Summary of numeric parameters selected

We know that KNN works primarily on the numeric measurement of datapoints. We notice, the values of smoothness mean vary drastically when compared to Length, Diameter and Height.

- As we can observe that the values of Height values are varying from 0.00 to 1.13 while, Diameter varies from 0.05 to 0.65 and Length from 0.075 to 0.815. We can see that the values are not uniform which may give biased results and inaccurate predictions.

<u>Normalisation</u>

- Hence, we will normalize the data in order to have standardized and uniform values. Using the normalization function which will use the difference of the variable from the minimum value and divide by the range, the values will be standardized.

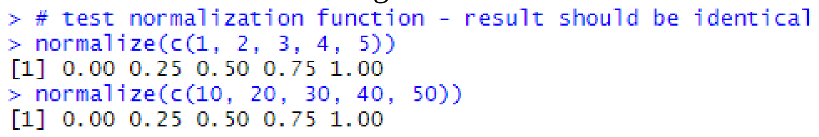


```
# create normalization function
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
normalize
```

Fig.4: Normalization

Using the normalization function which will use the difference of the variable from the minimum value and divide by the range, the values will be standardized.

- We have checked the working of normalization function.

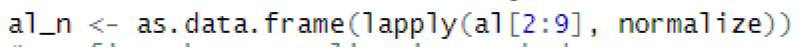


```
> # test normalization function - result should be identical
> normalize(c(1, 2, 3, 4, 5))
[1] 0.00 0.25 0.50 0.75 1.00
> normalize(c(10, 20, 30, 40, 50))
[1] 0.00 0.25 0.50 0.75 1.00
```

Fig.5: Verification Normalization Function

We can see that the values have been normalized for different set of input parameters.

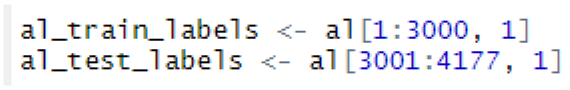
- In order to perform the normalization function for all the values, we have used the “lapply” function which will return the same length for all the values.



```
al_n <- as.data.frame(lapply(al[2:9], normalize))
```

Fig.6: Normalisation for all values

- The data has been divided into Training and Testing Data for better prediction.



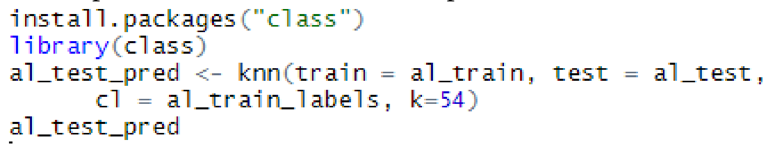
```
al_train_labels <- al[1:3000, 1]
al_test_labels <- al[3001:4177, 1]
```

Fig.7: Training and Testing Datasets

As seen, the first 3000 columns have been used for training the dataset and this is tested on the remaining columns.

K-Nearest Neighbor

- After the initial data analysis is completed, the training for KNN classifier is performed. The "class" package is used which classifies the datapoints for KNN algorithm. The algorithm selects the nearest neighbors based on the Euclidean distance of each datapoint and as per the number of clusters specified. [2]

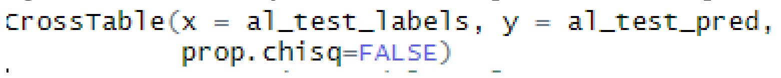


```
install.packages("class")
library(class)
al_test_pred <- knn(train = al_train, test = al_test,
      cl = al_train_labels, k=54)
al_test_pred
```

Fig.8: KNN Classifier

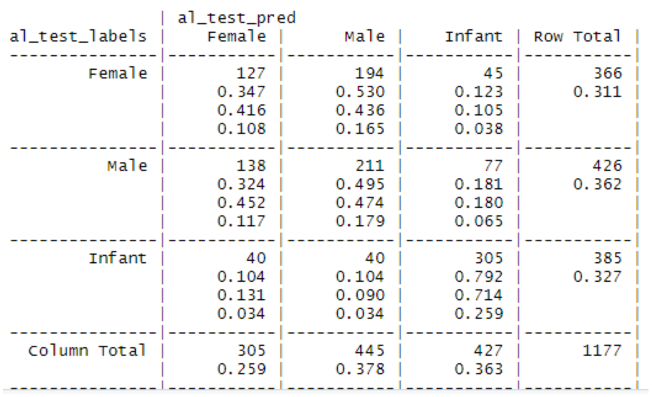
We can built the KNN classifier on the training and testing data considering 54 clusters.

- Since the training data is built upon 3000 columns, we have used the number of clusters as 54 which is almost equivalent to the square root of that number.
- We have checked the performance of the model by comparing with the predicted values and test values in the testing dataset. Hence, the Cross Table function included in the "gmodels" library will help in comparisons of two vectors.



```
CrossTable(x = al_test_labels, y = al_test_pred,
           prop.chisq=FALSE)
```

Fig.9: Cross Table



```
               | al_test_pred
al_test_labels |    Female |      Male |    Infant | Row Total |
---------------|-----------|-----------|-----------|-----------|
        Female |       127 |       194 |        45 |       366 |
               |     0.347 |     0.530 |     0.123 |     0.311 |
               |     0.416 |     0.436 |     0.105 |           |
               |     0.108 |     0.165 |     0.038 |           |
---------------|-----------|-----------|-----------|-----------|
          Male |       138 |       211 |        77 |       426 |
               |     0.324 |     0.495 |     0.181 |     0.362 |
               |     0.452 |     0.474 |     0.180 |           |
               |     0.117 |     0.179 |     0.065 |           |
---------------|-----------|-----------|-----------|-----------|
        Infant |        40 |        40 |       305 |       385 |
               |     0.104 |     0.104 |     0.792 |     0.327 |
               |     0.131 |     0.090 |     0.714 |           |
               |     0.034 |     0.034 |     0.259 |           |
---------------|-----------|-----------|-----------|-----------|
  Column Total |       305 |       445 |       427 |      1177 |
               |     0.259 |     0.378 |     0.363 |           |
---------------|-----------|-----------|-----------|-----------|
```

Fig. 10: Confusion Matrix

True Negative: The top left value of 127 has been accurately been identified as Female out of 1177 values.

True Positive: The values of 305 of 1177 was correctly identified as Infant and 77 as Male

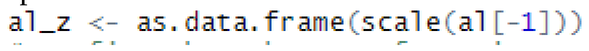
False Negative: The value of 40 for Infants shows that the predicted value was male while gender was infant.

False Positive: The model precited 45 values for Females when they were actually infants

- We can see that the model predicted 40 out of 1177 values incorrectly and the accuracy was seen to be 97%.
- The improvement of prediction can be very well performed by Normalisation. However, normalization, does not compress the middle values for uniformity always. Therefore, Z-score standardization can be useful in such cases.

Z-Score Standardization

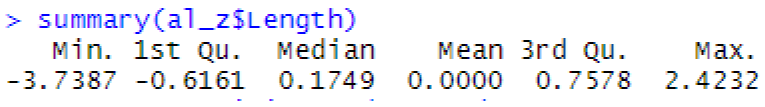
- Z-score standardization uses the scale() function to re-scale the values for better optimization.



```
al_z <- as.data.frame(scale(al[-1]))
```

Fig.11: Re-scaling using Z-score

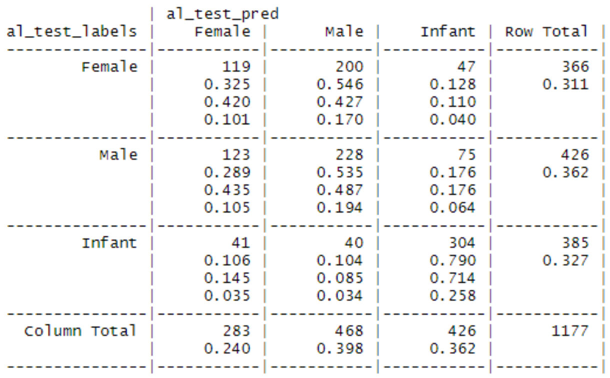
The Z-score value should be 0 as, we can see the mean value is 0 in summary.



```
> summary(al_z$Length)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-3.7387 -0.6161  0.1749  0.0000  0.7578  2.4232
```

Fig.12: Summary of Z-Score function

- As performed earlier, we will be dividing the data into testing and training data labels, then perform the KNN classifier and built the cross table.[2]



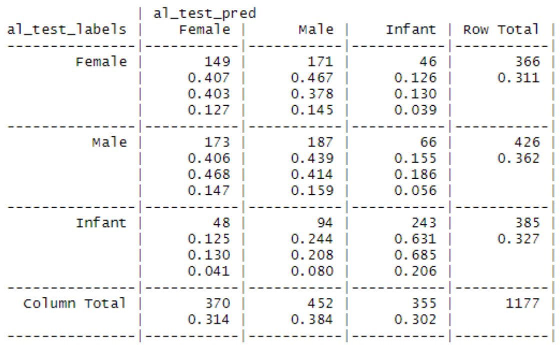
```
               | al_test_pred
al_test_labels |    Female |      Male |    Infant | Row Total |
---------------|-----------|-----------|-----------|-----------|
        Female |       119 |       200 |        47 |       366 |
               |     0.325 |     0.546 |     0.128 |     0.311 |
               |     0.420 |     0.427 |     0.110 |           |
               |     0.101 |     0.170 |     0.040 |           |
---------------|-----------|-----------|-----------|-----------|
          Male |       123 |       228 |        75 |       426 |
               |     0.289 |     0.535 |     0.176 |     0.362 |
               |     0.435 |     0.487 |     0.176 |           |
               |     0.105 |     0.194 |     0.064 |           |
---------------|-----------|-----------|-----------|-----------|
        Infant |        41 |        40 |       304 |       385 |
               |     0.106 |     0.104 |     0.790 |     0.327 |
               |     0.145 |     0.085 |     0.714 |           |
               |     0.035 |     0.034 |     0.258 |           |
---------------|-----------|-----------|-----------|-----------|
  Column Total |       283 |       468 |       426 |      1177 |
               |     0.240 |     0.398 |     0.362 |           |
---------------|-----------|-----------|-----------|-----------|
```

We can see that, the False Negative values were incorrectly classified as 41 out of 1177, Hence the accuracy has reduced to 95%.

Fig.13: Confusion Matrix using Z-Score

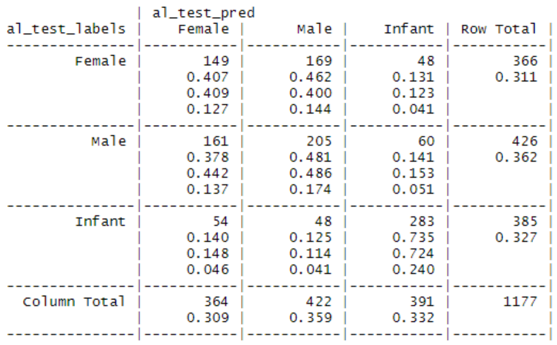
- Testing for several values of K has been done as follows on the same function:

```
al_test_pred <- knn(train = al_train, test = al_test, cl = al_train_labels, k=1)
CrossTable(x = al_test_labels, y = al_test_pred, prop.chisq=FALSE)
```



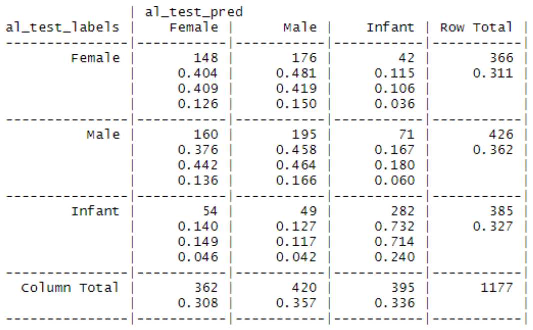
```
               | al_test_pred
al_test_labels |    Female |      Male |    Infant | Row Total |
---------------|-----------|-----------|-----------|-----------|
        Female |       149 |       171 |        46 |       366 |
               |     0.407 |     0.467 |     0.126 |     0.311 |
               |     0.403 |     0.378 |     0.130 |           |
               |     0.127 |     0.145 |     0.039 |           |
---------------|-----------|-----------|-----------|-----------|
          Male |       173 |       187 |        66 |       426 |
               |     0.406 |     0.439 |     0.155 |     0.362 |
               |     0.468 |     0.414 |     0.186 |           |
               |     0.147 |     0.159 |     0.056 |           |
---------------|-----------|-----------|-----------|-----------|
        Infant |        48 |        94 |       243 |       385 |
               |     0.125 |     0.244 |     0.631 |     0.327 |
               |     0.130 |     0.208 |     0.685 |           |
               |     0.041 |     0.080 |     0.206 |           |
---------------|-----------|-----------|-----------|-----------|
  Column Total |       370 |       452 |       355 |      1177 |
               |     0.314 |     0.384 |     0.302 |           |
---------------|-----------|-----------|-----------|-----------|
```

K=1

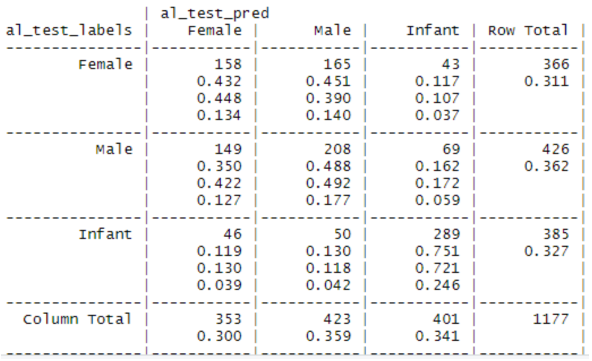


```
               | al_test_pred
al_test_labels |    Female |      Male |    Infant | Row Total |
---------------|-----------|-----------|-----------|-----------|
        Female |       149 |       169 |        48 |       366 |
               |     0.407 |     0.462 |     0.131 |     0.311 |
               |     0.409 |     0.400 |     0.123 |           |
               |     0.127 |     0.144 |     0.041 |           |
---------------|-----------|-----------|-----------|-----------|
          Male |       161 |       205 |        60 |       426 |
               |     0.378 |     0.481 |     0.141 |     0.362 |
               |     0.442 |     0.486 |     0.153 |           |
               |     0.137 |     0.174 |     0.051 |           |
---------------|-----------|-----------|-----------|-----------|
        Infant |        54 |        48 |       283 |       385 |
               |     0.140 |     0.125 |     0.735 |     0.327 |
               |     0.148 |     0.114 |     0.724 |           |
               |     0.046 |     0.041 |     0.240 |           |
---------------|-----------|-----------|-----------|-----------|
  Column Total |       364 |       422 |       391 |      1177 |
               |     0.309 |     0.359 |     0.332 |           |
---------------|-----------|-----------|-----------|-----------|
```

K=5



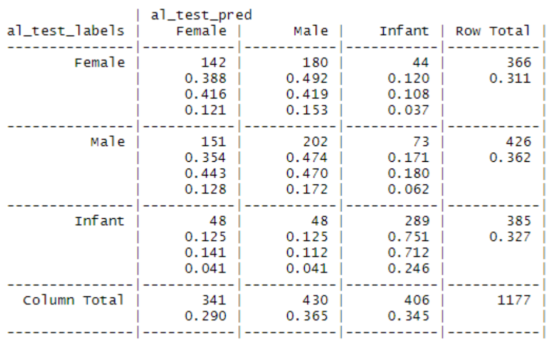
```
               | al_test_pred
al_test_labels |    Female |      Male |    Infant | Row Total |
---------------|-----------|-----------|-----------|-----------|
        Female |       148 |       176 |        42 |       366 |
               |     0.404 |     0.481 |     0.115 |     0.311 |
               |     0.409 |     0.419 |     0.106 |           |
               |     0.126 |     0.150 |     0.036 |           |
---------------|-----------|-----------|-----------|-----------|
          Male |       160 |       195 |        71 |       426 |
               |     0.376 |     0.458 |     0.167 |     0.362 |
               |     0.442 |     0.464 |     0.180 |           |
               |     0.136 |     0.166 |     0.060 |           |
---------------|-----------|-----------|-----------|-----------|
        Infant |        54 |        49 |       282 |       385 |
               |     0.140 |     0.127 |     0.732 |     0.327 |
               |     0.149 |     0.117 |     0.714 |           |
               |     0.046 |     0.042 |     0.240 |           |
---------------|-----------|-----------|-----------|-----------|
  Column Total |       362 |       420 |       395 |      1177 |
               |     0.308 |     0.357 |     0.336 |           |
---------------|-----------|-----------|-----------|-----------|
```



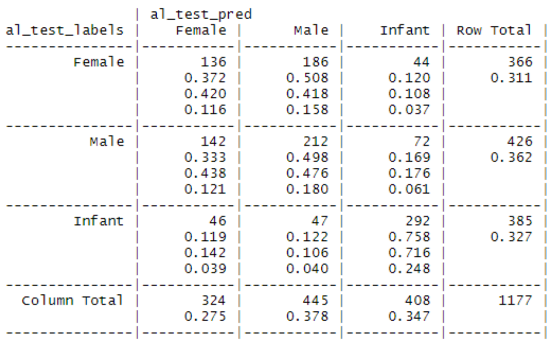
```
               | al_test_pred
al_test_labels |    Female |      Male |    Infant | Row Total |
---------------|-----------|-----------|-----------|-----------|
        Female |       158 |       165 |        43 |       366 |
               |     0.432 |     0.451 |     0.117 |     0.311 |
               |     0.448 |     0.390 |     0.107 |           |
               |     0.134 |     0.140 |     0.037 |           |
---------------|-----------|-----------|-----------|-----------|
          Male |       149 |       208 |        69 |       426 |
               |     0.350 |     0.488 |     0.162 |     0.362 |
               |     0.422 |     0.492 |     0.172 |           |
               |     0.127 |     0.177 |     0.059 |           |
---------------|-----------|-----------|-----------|-----------|
        Infant |        46 |        50 |       289 |       385 |
               |     0.119 |     0.130 |     0.751 |     0.327 |
               |     0.130 |     0.118 |     0.721 |           |
               |     0.039 |     0.042 |     0.246 |           |
---------------|-----------|-----------|-----------|-----------|
  Column Total |       353 |       423 |       401 |      1177 |
               |     0.300 |     0.359 |     0.341 |           |
---------------|-----------|-----------|-----------|-----------|
```

K=11

K=15



```
               | al_test_pred
al_test_labels |     Female |       Male |     Infant |  Row Total |
---------------|------------|------------|------------|------------|
        Female |        142 |        180 |         44 |        366 |
               |      0.388 |      0.492 |      0.120 |      0.311 |
               |      0.416 |      0.419 |      0.108 |            |
               |      0.121 |      0.153 |      0.037 |            |
---------------|------------|------------|------------|------------|
          Male |        151 |        202 |         73 |        426 |
               |      0.354 |      0.474 |      0.171 |      0.362 |
               |      0.443 |      0.470 |      0.180 |            |
               |      0.128 |      0.172 |      0.062 |            |
---------------|------------|------------|------------|------------|
        Infant |         48 |         48 |        289 |        385 |
               |      0.125 |      0.125 |      0.751 |      0.327 |
               |      0.141 |      0.112 |      0.712 |            |
               |      0.041 |      0.041 |      0.246 |            |
---------------|------------|------------|------------|------------|
  Column Total |        341 |        430 |        406 |       1177 |
               |      0.290 |      0.365 |      0.345 |            |
---------------|------------|------------|------------|------------|
```



```
               | al_test_pred
al_test_labels |     Female |       Male |     Infant |  Row Total |
---------------|------------|------------|------------|------------|
        Female |        136 |        186 |         44 |        366 |
               |      0.372 |      0.508 |      0.120 |      0.311 |
               |      0.420 |      0.418 |      0.108 |            |
               |      0.116 |      0.158 |      0.037 |            |
---------------|------------|------------|------------|------------|
          Male |        142 |        212 |         72 |        426 |
               |      0.333 |      0.498 |      0.169 |      0.362 |
               |      0.438 |      0.476 |      0.176 |            |
               |      0.121 |      0.180 |      0.061 |            |
---------------|------------|------------|------------|------------|
        Infant |         46 |         47 |        292 |        385 |
               |      0.119 |      0.122 |      0.758 |      0.327 |
               |      0.142 |      0.106 |      0.716 |            |
               |      0.039 |      0.040 |      0.248 |            |
---------------|------------|------------|------------|------------|
  Column Total |        324 |        445 |        408 |       1177 |
               |      0.275 |      0.378 |      0.347 |            |
---------------|------------|------------|------------|------------|
```

K=21 K=27

- As we can see, the algorithm gives more accurate results for K=15 and K=27 with accuracy of 97% as compared to other values.

Conclusion

- As per the analysis in Breast Cancer dataset, The cancer was seen to be more benign more women as compared to malignant. The number of records with benign cancer were 61 and hence more than patients having malignant cancer.
- Using the Abalone Dataset, The KNN algorithm helped in predicting that with the increase in rings, the shell age increased. Also, the females seemed to be more older than males.
- The normalization technique is efficient in some cases, however, may not consider outliers for compression as it considers extreme parameters. Hence, z-score is used for re-scaling. Clearly, Normalisation seemed more effective on both the datasets as compared to z-score with higher accuracy.

References

1. (n.d.). Retrieved from http://archive.ics.uci.edu/ml/datasets/Abalone
2. Sign In. (n.d.). Retrieved from https://rpubs.com/jkklim/ABALONE_KNN