END OF TERM ASSESSMENT

DATA MINING & MACHINE LEARNING

Sean Gavin
A00251388

Table of Contents

Decision Trees

1.1 Business Understanding

The dataset which I used to explore decision trees is a car evaluation dataset from UCI Machine Learning Repository. My goal or primary objective from a business perspective is to help out people who are looking into buying a second hand car. My plan is to take these features such as number of doors or quality of the maintenance on the car and see does this affect the condition value for a vehicle.

1.2 Data Understanding & Preparation

The car evaluation dataset includes 1728 observations (rows) and 7 facets (columns) all of which are nominal features (buying, maintenance, doors, persons, boot capacity, safety & condition) that were converted into factors. I intend to investigate the condition of the car based totally on the different features. The dataset is not missing any fields which means no preparation of the data has to be carried out at the moment.

Summary of the dataset below:

```
Buying      Maintenance    Doors        Persons
high :432   high :432      2     :432   2    :576
low  :432   low  :432      3     :432   4    :576
med  :432   med  :432      4     :432   more:576
vhigh:432   vhigh:432      more :432

Boot. Capacity Safety      Condition
big  :576      high:576    acc  : 384
med  :576      low :576    good :  69
small:576      med :576    unacc:1210
                           vgood:  65
```

1.3 Modelling

To construct my training and test sets, I first randomized the order of my dataset as it was given in order of condition. I then took the first 1200 cases and introduced them to my training dataset. The remaining 528 cases were designated as my test data.

To generate the model, I used the C50 library. I used the C5.0 function by assigning condition as y before tilde and using a "." as a shorthand notation to indicate using the rest of the features from the dataset as multiple independent variables.

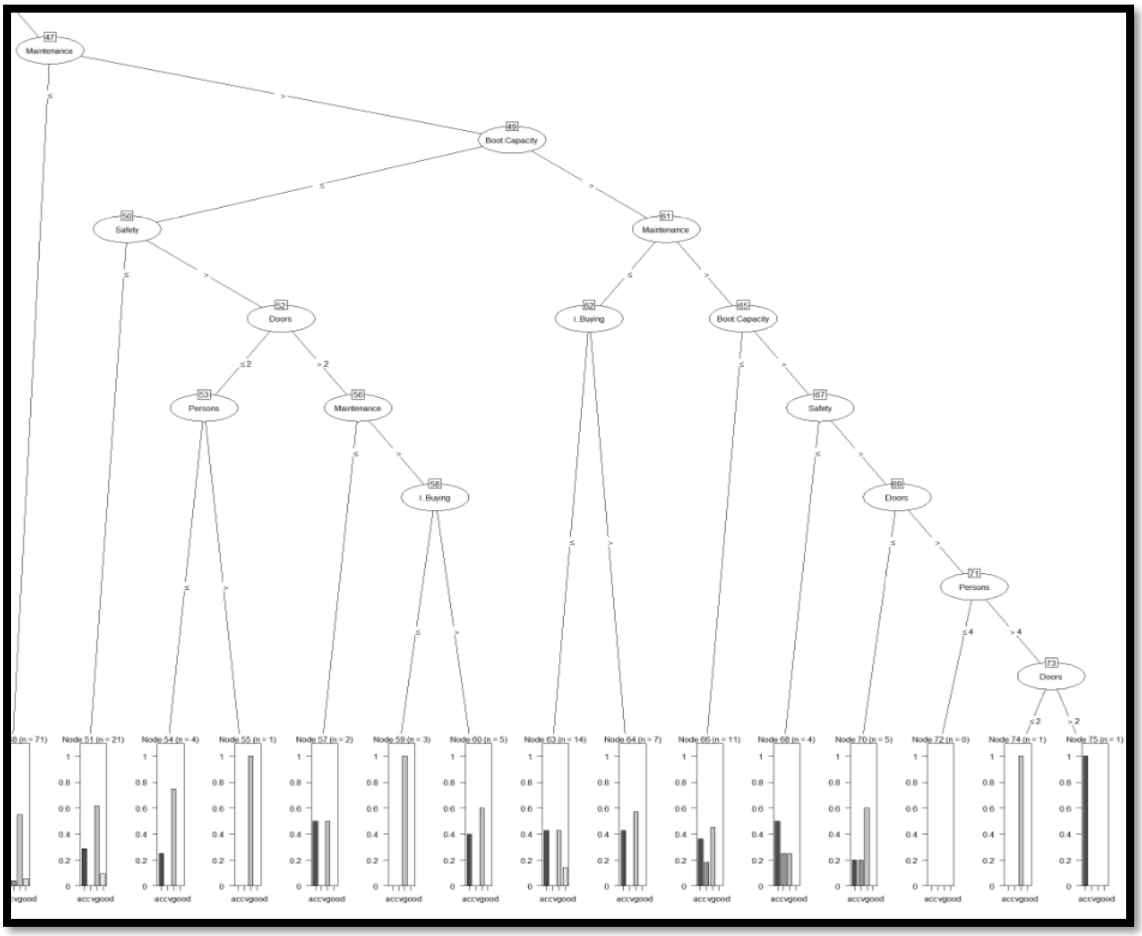
Breakdown of train data and cars data in relation to the Condition independent variable.

```
> prop.table(table(cars_train$Condition))

       acc       good      unacc      vgood 
0.21166667 0.03833333 0.71333333 0.03666667 
> prop.table(table(cars_test$Condition))

       acc       good      unacc      vgood 
0.24621212 0.04356061 0.67045455 0.03977273 
> 
```

Example of a snippet of the plotted model:



Summary of the Model below:

```
Evaluation on training data (1200 cases):

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

          38   22( 1.8%)   <<


         (a)   (b)   (c)   (d)    <-classified as
        ----  ----  ----  ----
         247     4     1     2    (a): class acc
                46                (b): class good
          12     2   842          (c): class unacc
                 1          43    (d): class vgood


      Attribute usage:

      100.00% Safety
       66.17% Persons
       44.08% ï..Buying
       44.08% Maintenance
       38.08% Boot.Capacity
        9.08% Doors


Time: 0.0 secs
```

Predictions for the test data

```
> predictions <- predict(model, cars_test)
> summary(predictions)
  acc  good unacc vgood 
  129    35   345    19 
```

Prediction Table:

```
> table <- CrossTable(predictions, cars_test$Condition,
+                     prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
+                     dnn = c('predicted default', 'actual default'))

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

 
Total Observations in Table:  528 

 
                  | actual default 
predicted default |       acc |      good |     unacc |     vgood | Row Total | 
------------------|-----------|-----------|-----------|-----------|-----------|
              acc |       120 |         0 |         9 |         0 |       129 | 
                  |     0.227 |     0.000 |     0.017 |     0.000 |           | 
------------------|-----------|-----------|-----------|-----------|-----------|
             good |         8 |        23 |         1 |         3 |        35 | 
                  |     0.015 |     0.044 |     0.002 |     0.006 |           | 
------------------|-----------|-----------|-----------|-----------|-----------|
            unacc |         1 |         0 |       344 |         0 |       345 | 
                  |     0.002 |     0.000 |     0.652 |     0.000 |           | 
------------------|-----------|-----------|-----------|-----------|-----------|
            vgood |         1 |         0 |         0 |        18 |        19 | 
                  |     0.002 |     0.000 |     0.000 |     0.034 |           | 
------------------|-----------|-----------|-----------|-----------|-----------|
     Column Total |       130 |        23 |       354 |        21 |       528 | 
```

Confusion Matrix and Statistics of the predictions model:

```
> confusionMatrix(table(predictions, cars_test$Condition))
Confusion Matrix and Statistics

           
predictions acc good unacc vgood
      acc   120    0     9     0
      good    8   23     1     3
      unacc   1    0   344     0
      vgood   1    0     0    18

Overall Statistics
                                          
               Accuracy : 0.9564          
                 95% CI : (0.9354, 0.9722)
    No Information Rate : 0.6705          
    P-Value [Acc > NIR] : < 2.2e-16       
                                          
                  Kappa : 0.9124          
                                          
 Mcnemar's Test P-Value : NA              

Statistics by Class:

                     Class: acc Class: good Class: unacc Class: vgood
Sensitivity              0.9231     1.00000       0.9718      0.85714
Specificity              0.9774     0.97624       0.9943      0.99803
Pos Pred Value           0.9302     0.65714       0.9971      0.94737
Neg Pred Value           0.9749     1.00000       0.9454      0.99411
Prevalence               0.2462     0.04356       0.6705      0.03977
Detection Rate           0.2273     0.04356       0.6515      0.03409
Detection Prevalence     0.2443     0.06629       0.6534      0.03598
Balanced Accuracy        0.9502     0.98812       0.9830      0.92759
```

1.4 Evaluation

From the results we can see that there is a 96% accuracy in the prediction of the vehicles condition based on the other nominal factors. We can also see that there are no bad predictions, i.e. an unacceptable vehicle being returned as very good etc.

Vehicles with a high level of maintenance and a low level of safety tend to result in the vehicles condition being unacceptable. Boot capacity and no. of doors didn't have a lot of a bearing in finding out the condition for a vehicle and accordingly should be excluded from the model. Vehicles with a low stage of upkeep and a high degree of safety resulted, normally in the vehicles condition being good or very good. In conclusion, a decision tree is originally modelled using several nominal features such as safety, maintenance, number of doors, number of persons and boot capacity. The model performance can be evaluated by looking at the confusion matrix and the overall statistics. We can decide that the predictions have been very exact based on the 97% accuracy and the fact that we acquired no "bad" predictions.

kNN

2.1 Business Understanding

The dataset which I used to explore kNN analysis is an banknote validity dataset from UCI Machine Learning Repository. My goal or primary objective from a business perspective is to help out the federal reserve to see predict percentage of banknotes which are counterfeit or fake. My plan is to take the data such as numerical aspects such as variation, skewness, curtosis & entropy of wavelet transformed image and predict percentage of real and fake banknotes in order to help the federal reserve review banknotes in circulation.

2.2 Data Understanding & Preparation

The banknote validity dataset contains 1372 observations (rows) and 5 elements (columns) 4 of which are numerical features (variation, skewness, curtosis & entropy of wavelet transformed image) and class, which is a nominal feature that is modified into a element representing real or fake. In relation to the class feature zero represents a real bank note, while one equals a fake or counterfeit back note.

summary of the bank notes dataset:

```
> summary(banknotes)
 ï..Variance.of.Wavelet.Transformed.Edge
 Min.   :-7.0421
 1st Qu.:-1.7730
 Median : 0.4962
 Mean   : 0.4337
 3rd Qu.: 2.8215
 Max.   : 6.8248
 Skewness.of.Wavelet.Transformed.image
 Min.   :-13.773
 1st Qu.: -1.708
 Median :  2.320
 Mean   :  1.922
 3rd Qu.:  6.815
 Max.   : 12.952
 curtosis.of.Wavelet.Transformed.image entropy.of.image      class
 Min.   :-5.2861                       Min.   :-8.5482   Min.   :0.0000
 1st Qu.:-1.5750                       1st Qu.:-2.4135   1st Qu.:0.0000
 Median : 0.6166                       Median :-0.5867   Median :0.0000
 Mean   : 1.3976                       Mean   :-1.1917   Mean   :0.4446
 3rd Qu.: 3.1793                       3rd Qu.: 0.3948   3rd Qu.:1.0000
 Max.   :17.9274                       Max.   : 2.4495   Max.   :1.0000
```

2.3 Modelling

To build my training and test sets, I first randomized the order of my dataset as it used to be given in order real then fake. I then took the first a thousand instances and added them to my training dataset. The remaining 372 cases have been specified as my test data.

To generate the kNN model:

```
predictions <- knn(train = notes_train, test = notes_test,
                   cl = notes_train_labels, k=n)
```

K = sqRt of N = 19:

```
CrossTable(predictions, notes_test_labels,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE)
|
predictions <- knn(train = notes_train, test = notes_test,
                   cl = notes_train_labels, k=19)
```

Prediction Table K = 19:

```
Total Observations in Table:  372


             | notes_test_labels
 predictions |         0 |         1 | Row Total |
-------------|-----------|-----------|-----------|
           0 |       201 |         0 |       201 |
             |     0.540 |     0.000 |           |
-------------|-----------|-----------|-----------|
           1 |         5 |       166 |       171 |
             |     0.013 |     0.446 |           |
-------------|-----------|-----------|-----------|
Column Total |       206 |       166 |       372 |
-------------|-----------|-----------|-----------|
```

Confusion Matrix where K = 19.

```
> confusionMatrix(table(predictions ,notes_test_labels))
Confusion Matrix and Statistics

           notes_test_labels
predictions   0   1
          0 201   0
          1   5 166

               Accuracy : 0.9866
                 95% CI : (0.9689, 0.9956)
    No Information Rate : 0.5538
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9729

 Mcnemar's Test P-Value : 0.07364

            Sensitivity : 0.9757
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9708
             Prevalence : 0.5538
         Detection Rate : 0.5403
   Detection Prevalence : 0.5403
      Balanced Accuracy : 0.9879

       'Positive' Class : 0
```

K = 13:

```
predictions <- knn(train = notes_train, test = notes_test,
                   cl = notes_train_labels, k=13)
CrossTable(predictions, notes_test_labels,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE)
confusionMatrix(table(predictions ,notes_test_labels))
```

Prediction Table K = 13

```
Total Observations in Table:  372

             | notes_test_labels
 predictions |         0 |         1 | Row Total |
-------------|-----------|-----------|-----------|
           0 |       202 |         0 |       202 |
             |     0.543 |     0.000 |           |
-------------|-----------|-----------|-----------|
           1 |         4 |       166 |       170 |
             |     0.011 |     0.446 |           |
-------------|-----------|-----------|-----------|
Column Total |       206 |       166 |       372 |
-------------|-----------|-----------|-----------|
```

Confusion Matrix where K = 13:

```
Confusion Matrix and Statistics

           notes_test_labels
predictions   0   1
          0 202   0
          1   4 166

               Accuracy : 0.9892
                 95% CI : (0.9727, 0.9971)
    No Information Rate : 0.5538
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9783

 Mcnemar's Test P-Value : 0.1336

            Sensitivity : 0.9806
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9765
             Prevalence : 0.5538
         Detection Rate : 0.5430
   Detection Prevalence : 0.5430
      Balanced Accuracy : 0.9903

       'Positive' Class : 0
```

K = 11

```
predictions <- knn(train = notes_train, test = notes_test,
                      cl = notes_train_labels, k=11)

CrossTable(predictions, notes_test_labels,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE)
confusionMatrix(table(predictions ,notes_test_labels))
```

Prediction Table K = 11

```
Total Observations in Table:  372

             | notes_test_labels
 predictions |         0 |         1 | Row Total |
-------------|-----------|-----------|-----------|
           0 |       205 |         0 |       205 |
             |     0.551 |     0.000 |           |
-------------|-----------|-----------|-----------|
           1 |         1 |       166 |       167 |
             |     0.003 |     0.446 |           |
-------------|-----------|-----------|-----------|
Column Total |       206 |       166 |       372 |
-------------|-----------|-----------|-----------|
```

Confusion Matrix where K = 11. Which is most accurate:

```
Confusion Matrix and Statistics

           notes_test_labels
predictions   0   1
          0 205   0
          1   1 166

               Accuracy : 0.9973
                 95% CI : (0.9851, 0.9999)
    No Information Rate : 0.5538
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9946

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9951
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9940
             Prevalence : 0.5538
         Detection Rate : 0.5511
   Detection Prevalence : 0.5511
      Balanced Accuracy : 0.9976

       'Positive' Class : 0
```

2.4 Evaluation

The first k value I used in the kNN model was once the square root of N, which is 19. The model performed well with a 98% accuracy. I then tried a few different values for k and found that k = 11 used to be the value that returned the greatest results, with an accuracy of 99.7%. In conclusion, a kNN model is at first modelled the use of quite a few numerical aspects such as variation, skewness, curtosis and entropy of wavelet transformed image. The models overall performance can be evaluated by looking at the confusion matrix and the overall statistics. We can determine that the predictions have been satisfactory when k = 11, based on the 99.7% accuracy.

kMeans Clustering

3.1 Business Understanding

The dataset which I used to explore kMeans Clustering is a wheat seed dataset from UCI Machine Learning Repository. My goal or primary objective from a business perspective is to help out farmers or agronomist to measure geometrical properties of kernels belonging to three different varieties of wheat (Kama, Rosa and Canadian). My plan is to take the geometrical properties of kernels and cluster the data and examine correlation between some variables from the data such as length and width.

3.2 Data Understanding & Preparation

The seed dataset includes 210 observations (rows) and 7 facets (columns) all of which are numerical features (Area, Perimeter, Compactness, Length, Width ,Asymetry.coef , Grove.length, Type). The three different varieties of wheat: Kama, Rosa and Canadian contain 70 elements each. The dataset is not missing any fields, however the data field contained in the csv file are shortened. I modified the field names to present their full name. The type variable is also converted into a factor value.

```
seed <- read.csv("data/Seed_Data.csv")
View(seed)
# scaling numerical data
set.seed(4)
seed_z <- scale(seed[,-8])
# rename column and correcting data type
names(seed) <- c("Area", "Perimeter", "Compactness", "Length", "Width", "Asymetry.coef", "Grove.length", "Type")
seed$Type <- as.factor(seed$Type)
```

3.3 Modelling

From carrying out different values for k, I have determined the ideal number for k is 3.

```
model <- kmeans(seed, 3)
model
model$cluster
model$tot.withinss
model$centers
seed$Type
```

```
> model
K-means clustering with 3 clusters of sizes 61, 74, 75

Cluster means:
      Area Perimeter Compactness   Length    Width Asymetry.coef Grove.length      Type
1 18.72180  16.29738   0.8850869 6.208934 3.722672      3.603590     6.066098 0.9836066
2 14.63203  14.45324   0.8790973 5.561784 3.274892      2.744043     5.184932 0.1351351
3 11.90907  13.25027   0.8515493 5.222333 2.865093      4.722187     5.093040 1.8666667

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 2 2
 [66] 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1
[131] 1 1 2 2 2 2 1 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[196] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

Within cluster sum of squares by cluster:
[1] 185.0922 223.1591 217.4687
 (between_SS / total_SS =  78.1 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
```

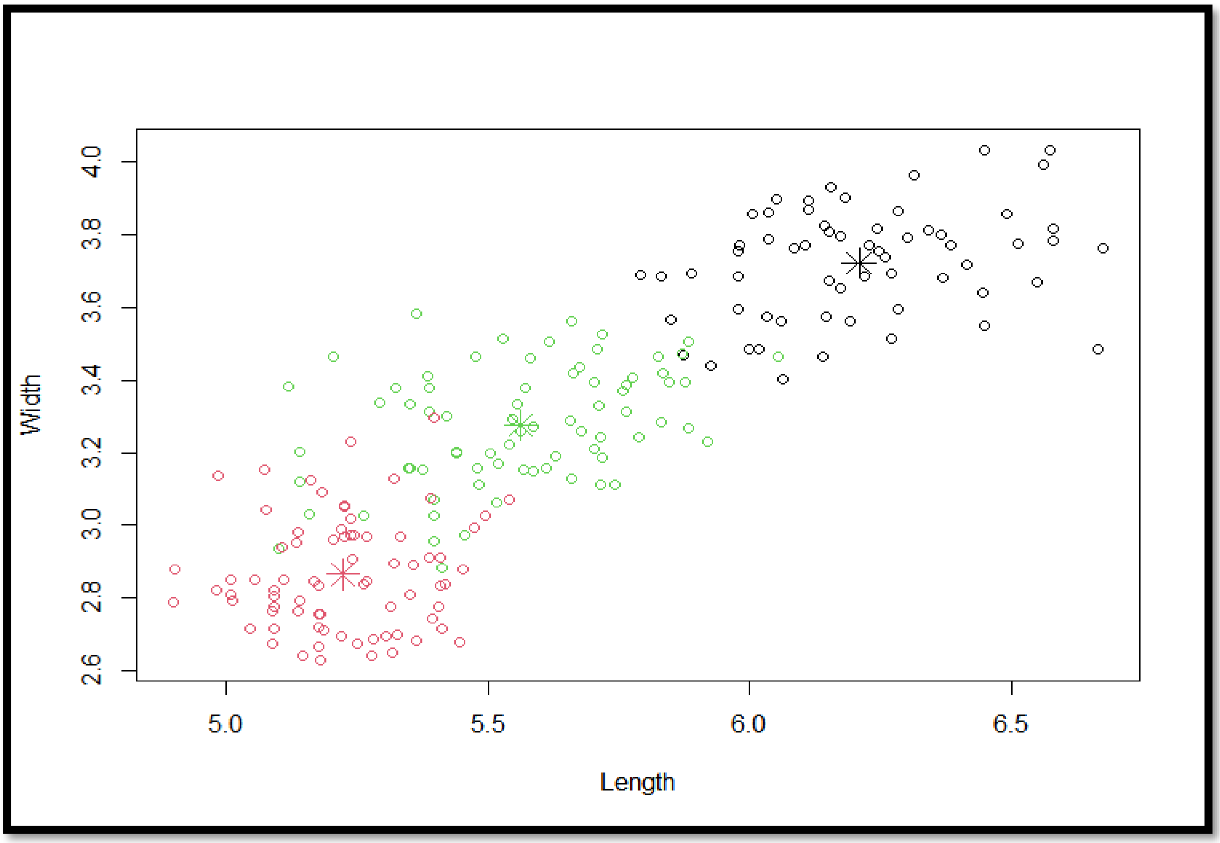
From analysing the results above, you can see the cluster size is not proportional to each kernel type. Cluster one: 61, Cluster two: 74, Cluster three: 75, this reveals a different insight to the true observations from each kernel type (70 each). This shows that there is probably kernels with comparable geometrical properties which originate from different type or species.

Matrix table breakdown in relation to cluster types:

```
> table(seed$Type, model$cluster)

     1  2  3
  0  1  5 64
  1 60  0 10
  2  0 70  0
```

My next step was to take geometrical properties such as height and width and see if they correlate with them that resembles different kernel types. I did this by generating a cluster plot chart:



3.4 Evaluation

It is evident from the matrix above that the geometrical properties of kernels alone are not sufficient to achieve a clustering that resembles kernel types. If I was to analyse this data set again, I would add additional geometrical properties such as genetic and metabolites properties. From the plot graph above you can see that height and width is not an efficient enough feature to overcome this problem. Also, the plot gives us an indication that the clusters were located really close to one another and in some cases an overlapping of clusters can be seen. Once again k-means can be carried out on this dataset, however geometrical properties of kernels alone are not sufficient enough to obtain a clustering that resembles kernel types.