Thota, Sunil Raj - Data Mining.R

# Intermediate Analytics  
# ALY 6015  
# Module 4 - Data Mining  
# 02/09/2021  
# Sunil Raj Thota  
# NUID: 001099670

# Get and set the working directories  
getwd()

## [1] "G:/NEU/Coursework/2021 Q1 Winter/ALY 6015 IA/Discussions & Assignments"

setwd('G:/NEU/Coursework/2021 Q1 Winter/ALY 6015 IA/Discussions & Assignments')  
getwd()

## [1] "G:/NEU/Coursework/2021 Q1 Winter/ALY 6015 IA/Discussions & Assignments"

# Installed the above packages into the work space  
install.packages("datasets")  
install.packages("plyr")  
install.packages("dplyr")  
install.packages("tidyr")  
install.packages("factoextra")  
install.packages("NbClust")  
install.packages("party")  
install.packages("caret")  
install.packages("fpc")  
  
# Loaded the below libraries into the work space  
library(plyr)

library(dplyr)

library(tidyr)

library(factoextra)

library(NbClust)

library(party)

require(datasets)

library(caret)

library(fpc)

# Problem 1  
data(iris)  
View(iris)  
str(iris)

## 'data.frame': 150 obs. of 5 variables:  
## $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...  
## $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...  
## $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...  
## $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...  
## $ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

head(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 1 5.1 3.5 1.4 0.2 setosa  
## 2 4.9 3.0 1.4 0.2 setosa  
## 3 4.7 3.2 1.3 0.2 setosa  
## 4 4.6 3.1 1.5 0.2 setosa  
## 5 5.0 3.6 1.4 0.2 setosa  
## 6 5.4 3.9 1.7 0.4 setosa

tail(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 145 6.7 3.3 5.7 2.5 virginica  
## 146 6.7 3.0 5.2 2.3 virginica  
## 147 6.3 2.5 5.0 1.9 virginica  
## 148 6.5 3.0 5.2 2.0 virginica  
## 149 6.2 3.4 5.4 2.3 virginica  
## 150 5.9 3.0 5.1 1.8 virginica

summary(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width   
## Min. :4.300 Min. :2.000 Min. :1.000 Min. :0.100   
## 1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300   
## Median :5.800 Median :3.000 Median :4.350 Median :1.300   
## Mean :5.843 Mean :3.057 Mean :3.758 Mean :1.199   
## 3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100 3rd Qu.:1.800   
## Max. :7.900 Max. :4.400 Max. :6.900 Max. :2.500   
## Species   
## setosa :50   
## versicolor:50   
## virginica :50   
##   
##   
##

# Let's perform some Exploratory Data Analysis and Data Mining techniques using "iris" data set. To get this data set we need to install the 'data sets' package from the packages tab which is right side to the work space in R Studio.  
  
# Or we can also install the packages by using install.packages("package name") command. Once it is loaded we can use it in the code for further analysis and calculations. Loaded the "data sets" library into the work space. Loaded the iris Data set into the Environment.  
  
# I have also installed factoextra', 'party', 'caret', 'fpc', and 'NbClust' to perform data mining analysis in R. Let's install all the above packages  
  
# To View the diabetes Data set we use View() command, To observe the structure of the Data set we use str() command, and head () and tail() shows first and last few rows in the Data set. Summary() Provides the Descriptive Stats of the iris columns. We noticed 5 variables from the statistics given in the summary.

# Problem 2  
  
# split into training and test datasets  
set.seed(1234)  
sampleData <- sample(2, nrow(iris), replace = T, prob = c(0.7, 0.3))  
sampleData

## [1] 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1 2 1  
## [38] 1 2 2 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 2 1 2 2 1 1 1 1 2 1 1 1 1 1 2 1 2  
## [75] 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2  
## [112] 1 2 1 1 2 2 1 1 2 2 2 2 2 1 1 1 1 1 1 2 1 1 1 2 1 2 1 1 2 1 2 1 1 1 1 2 1  
## [149] 2 1

trainData <- iris[sampleData == 1, ]  
trainData

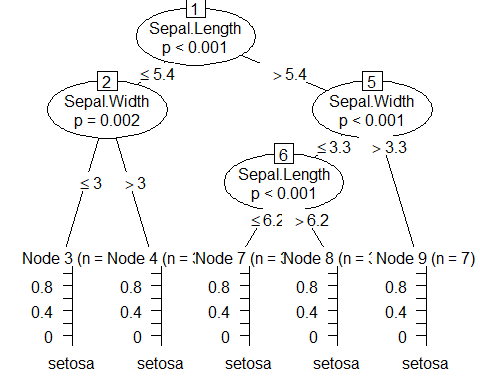
## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 1 5.1 3.5 1.4 0.2 setosa  
## 2 4.9 3.0 1.4 0.2 setosa  
## 3 4.7 3.2 1.3 0.2 setosa  
## 4 4.6 3.1 1.5 0.2 setosa  
## 6 5.4 3.9 1.7 0.4 setosa  
## 7 4.6 3.4 1.4 0.3 setosa  
## 8 5.0 3.4 1.5 0.2 setosa  
## 9 4.4 2.9 1.4 0.2 setosa  
## 10 4.9 3.1 1.5 0.1 setosa  
## 11 5.4 3.7 1.5 0.2 setosa  
## 12 4.8 3.4 1.6 0.2 setosa  
## 13 4.8 3.0 1.4 0.1 setosa  
## 15 5.8 4.0 1.2 0.2 setosa  
## 17 5.4 3.9 1.3 0.4 setosa  
## 18 5.1 3.5 1.4 0.3 setosa  
## 19 5.7 3.8 1.7 0.3 setosa  
## 20 5.1 3.8 1.5 0.3 setosa  
## 21 5.4 3.4 1.7 0.2 setosa  
## 22 5.1 3.7 1.5 0.4 setosa  
## 23 4.6 3.6 1.0 0.2 setosa  
## 24 5.1 3.3 1.7 0.5 setosa  
## 25 4.8 3.4 1.9 0.2 setosa  
## 27 5.0 3.4 1.6 0.4 setosa  
## 30 4.7 3.2 1.6 0.2 setosa  
## 31 4.8 3.1 1.6 0.2 setosa  
## 32 5.4 3.4 1.5 0.4 setosa  
## 33 5.2 4.1 1.5 0.1 setosa  
## 34 5.5 4.2 1.4 0.2 setosa  
## 35 4.9 3.1 1.5 0.2 setosa  
## 37 5.5 3.5 1.3 0.2 setosa  
## 38 4.9 3.6 1.4 0.1 setosa  
## 41 5.0 3.5 1.3 0.3 setosa  
## 42 4.5 2.3 1.3 0.3 setosa  
## 43 4.4 3.2 1.3 0.2 setosa  
## 44 5.0 3.5 1.6 0.6 setosa  
## 45 5.1 3.8 1.9 0.4 setosa  
## 46 4.8 3.0 1.4 0.3 setosa  
## 47 5.1 3.8 1.6 0.2 setosa  
## 48 4.6 3.2 1.4 0.2 setosa  
## 49 5.3 3.7 1.5 0.2 setosa  
## 51 7.0 3.2 4.7 1.4 versicolor  
## 52 6.4 3.2 4.5 1.5 versicolor  
## 54 5.5 2.3 4.0 1.3 versicolor  
## 55 6.5 2.8 4.6 1.5 versicolor  
## 56 5.7 2.8 4.5 1.3 versicolor  
## 57 6.3 3.3 4.7 1.6 versicolor  
## 59 6.6 2.9 4.6 1.3 versicolor  
## 62 5.9 3.0 4.2 1.5 versicolor  
## 63 6.0 2.2 4.0 1.0 versicolor  
## 64 6.1 2.9 4.7 1.4 versicolor  
## 65 5.6 2.9 3.6 1.3 versicolor  
## 67 5.6 3.0 4.5 1.5 versicolor  
## 68 5.8 2.7 4.1 1.0 versicolor  
## 69 6.2 2.2 4.5 1.5 versicolor  
## 70 5.6 2.5 3.9 1.1 versicolor  
## 71 5.9 3.2 4.8 1.8 versicolor  
## 73 6.3 2.5 4.9 1.5 versicolor  
## 75 6.4 2.9 4.3 1.3 versicolor  
## 76 6.6 3.0 4.4 1.4 versicolor  
## 77 6.8 2.8 4.8 1.4 versicolor  
## 78 6.7 3.0 5.0 1.7 versicolor  
## 79 6.0 2.9 4.5 1.5 versicolor  
## 80 5.7 2.6 3.5 1.0 versicolor  
## 82 5.5 2.4 3.7 1.0 versicolor  
## 83 5.8 2.7 3.9 1.2 versicolor  
## 84 6.0 2.7 5.1 1.6 versicolor  
## 85 5.4 3.0 4.5 1.5 versicolor  
## 87 6.7 3.1 4.7 1.5 versicolor  
## 88 6.3 2.3 4.4 1.3 versicolor  
## 89 5.6 3.0 4.1 1.3 versicolor  
## 91 5.5 2.6 4.4 1.2 versicolor  
## 93 5.8 2.6 4.0 1.2 versicolor  
## 94 5.0 2.3 3.3 1.0 versicolor  
## 95 5.6 2.7 4.2 1.3 versicolor  
## 96 5.7 3.0 4.2 1.2 versicolor  
## 97 5.7 2.9 4.2 1.3 versicolor  
## 98 6.2 2.9 4.3 1.3 versicolor  
## 99 5.1 2.5 3.0 1.1 versicolor  
## 101 6.3 3.3 6.0 2.5 virginica  
## 102 5.8 2.7 5.1 1.9 virginica  
## 103 7.1 3.0 5.9 2.1 virginica  
## 104 6.3 2.9 5.6 1.8 virginica  
## 105 6.5 3.0 5.8 2.2 virginica  
## 106 7.6 3.0 6.6 2.1 virginica  
## 107 4.9 2.5 4.5 1.7 virginica  
## 108 7.3 2.9 6.3 1.8 virginica  
## 109 6.7 2.5 5.8 1.8 virginica  
## 110 7.2 3.6 6.1 2.5 virginica  
## 112 6.4 2.7 5.3 1.9 virginica  
## 114 5.7 2.5 5.0 2.0 virginica  
## 115 5.8 2.8 5.1 2.4 virginica  
## 118 7.7 3.8 6.7 2.2 virginica  
## 119 7.7 2.6 6.9 2.3 virginica  
## 125 6.7 3.3 5.7 2.1 virginica  
## 126 7.2 3.2 6.0 1.8 virginica  
## 127 6.2 2.8 4.8 1.8 virginica  
## 128 6.1 3.0 4.9 1.8 virginica  
## 129 6.4 2.8 5.6 2.1 virginica  
## 130 7.2 3.0 5.8 1.6 virginica  
## 132 7.9 3.8 6.4 2.0 virginica  
## 133 6.4 2.8 5.6 2.2 virginica  
## 134 6.3 2.8 5.1 1.5 virginica  
## 136 7.7 3.0 6.1 2.3 virginica  
## 138 6.4 3.1 5.5 1.8 virginica  
## 139 6.0 3.0 4.8 1.8 virginica  
## 141 6.7 3.1 5.6 2.4 virginica  
## 143 5.8 2.7 5.1 1.9 virginica  
## 144 6.8 3.2 5.9 2.3 virginica  
## 145 6.7 3.3 5.7 2.5 virginica  
## 146 6.7 3.0 5.2 2.3 virginica  
## 148 6.5 3.0 5.2 2.0 virginica  
## 150 5.9 3.0 5.1 1.8 virginica

testData <- iris[sampleData == 2, ]  
testData

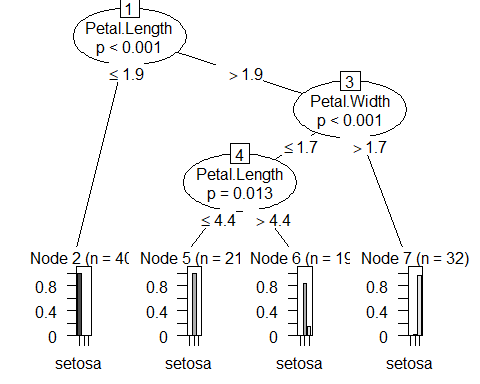
## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 5 5.0 3.6 1.4 0.2 setosa  
## 14 4.3 3.0 1.1 0.1 setosa  
## 16 5.7 4.4 1.5 0.4 setosa  
## 26 5.0 3.0 1.6 0.2 setosa  
## 28 5.2 3.5 1.5 0.2 setosa  
## 29 5.2 3.4 1.4 0.2 setosa  
## 36 5.0 3.2 1.2 0.2 setosa  
## 39 4.4 3.0 1.3 0.2 setosa  
## 40 5.1 3.4 1.5 0.2 setosa  
## 50 5.0 3.3 1.4 0.2 setosa  
## 53 6.9 3.1 4.9 1.5 versicolor  
## 58 4.9 2.4 3.3 1.0 versicolor  
## 60 5.2 2.7 3.9 1.4 versicolor  
## 61 5.0 2.0 3.5 1.0 versicolor  
## 66 6.7 3.1 4.4 1.4 versicolor  
## 72 6.1 2.8 4.0 1.3 versicolor  
## 74 6.1 2.8 4.7 1.2 versicolor  
## 81 5.5 2.4 3.8 1.1 versicolor  
## 86 6.0 3.4 4.5 1.6 versicolor  
## 90 5.5 2.5 4.0 1.3 versicolor  
## 92 6.1 3.0 4.6 1.4 versicolor  
## 100 5.7 2.8 4.1 1.3 versicolor  
## 111 6.5 3.2 5.1 2.0 virginica  
## 113 6.8 3.0 5.5 2.1 virginica  
## 116 6.4 3.2 5.3 2.3 virginica  
## 117 6.5 3.0 5.5 1.8 virginica  
## 120 6.0 2.2 5.0 1.5 virginica  
## 121 6.9 3.2 5.7 2.3 virginica  
## 122 5.6 2.8 4.9 2.0 virginica  
## 123 7.7 2.8 6.7 2.0 virginica  
## 124 6.3 2.7 4.9 1.8 virginica  
## 131 7.4 2.8 6.1 1.9 virginica  
## 135 6.1 2.6 5.6 1.4 virginica  
## 137 6.3 3.4 5.6 2.4 virginica  
## 140 6.9 3.1 5.4 2.1 virginica  
## 142 6.9 3.1 5.1 2.3 virginica  
## 147 6.3 2.5 5.0 1.9 virginica  
## 149 6.2 3.4 5.4 2.3 virginica

# To perform Data Mining, we are taking 'iris' data set which has 150 records and 5 variables. The columns that we are working now are Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, and Species of the iris data set.  
  
# To generate a sequence of random numbers we use set.seed() in R. With the help of sample() method, we can split the data set into two parts i.e., training and testing data sets which will be splitted into 70: 30 ratio of the main data set ('iris').  
  
# To store these splitted data sets we require two different variables that needs to be assigned as 'trainData' and 'testdata'.

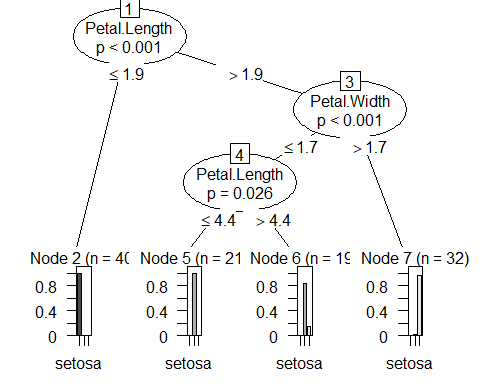
# Problem 3  
irisSepal <- Species ~ Sepal.Length + Sepal.Width  
irisPetal <- Species ~ Petal.Length + Petal.Width  
irisAll <-  
 Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width  
  
irisTree1 <- ctree(irisSepal, data = trainData)  
irisTree2 <- ctree(irisPetal, data = trainData)  
irisTree3 <- ctree(irisAll, data = trainData)  
  
plot(irisTree1)



plot(irisTree2)



plot(irisTree3)



# To apply Decision Tree Model to our data set, we have to use "ctree()" method and include our Training Data set. Here we are performing by taking various independent variables and named as irisSepal, irisPetal, and irisAll. Let us also delve into the analysis by plotting these Decision Trees models to compare and observe the differences in each plot.  
  
# From the plots, we can depict that there is a change in the root nodes and leaves. Primary node is called as the root and the bottom nodes are known as leaves. In this model, the decision makers are leaves. The final result is validated by a leaf node where the model terminates and has a value on the leaf

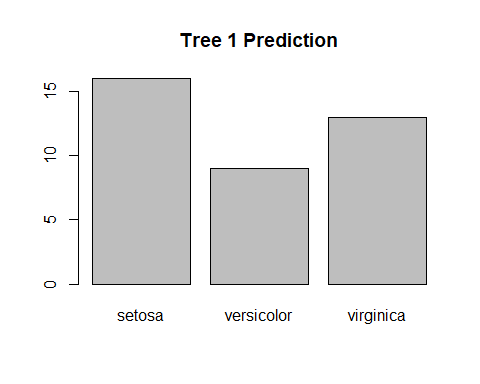
# Problem 4  
pred1 <- predict(irisTree1, testData)  
pred1

## [1] setosa setosa setosa setosa setosa setosa   
## [7] setosa setosa setosa setosa virginica setosa   
## [13] setosa setosa virginica versicolor versicolor versicolor  
## [19] setosa versicolor versicolor versicolor virginica virginica   
## [25] virginica virginica versicolor virginica versicolor virginica   
## [31] virginica virginica versicolor setosa virginica virginica   
## [37] virginica setosa   
## Levels: setosa versicolor virginica

table(pred1, testData$Species)

##   
## pred1 setosa versicolor virginica  
## setosa 10 4 2  
## versicolor 0 6 3  
## virginica 0 2 11

plot(pred1, main = "Tree 1 Prediction")



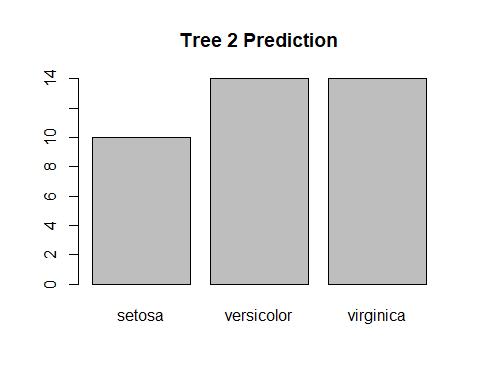
pred2 <- predict(irisTree2, testData)  
pred2

## [1] setosa setosa setosa setosa setosa setosa   
## [7] setosa setosa setosa setosa versicolor versicolor  
## [13] versicolor versicolor versicolor versicolor versicolor versicolor  
## [19] versicolor versicolor versicolor versicolor virginica virginica   
## [25] virginica virginica versicolor virginica virginica virginica   
## [31] virginica virginica versicolor virginica virginica virginica   
## [37] virginica virginica   
## Levels: setosa versicolor virginica

table(pred2, testData$Species)

##   
## pred2 setosa versicolor virginica  
## setosa 10 0 0  
## versicolor 0 12 2  
## virginica 0 0 14

plot(pred2, main = "Tree 2 Prediction")



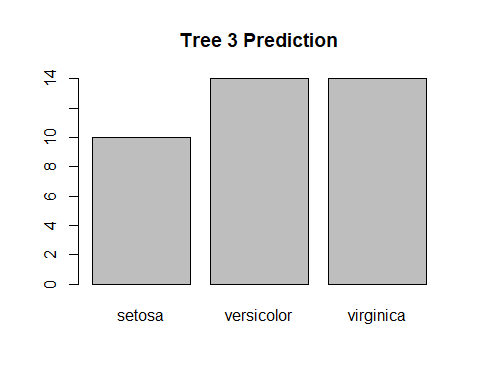
pred3 <- predict(irisTree3, testData)  
pred3

## [1] setosa setosa setosa setosa setosa setosa   
## [7] setosa setosa setosa setosa versicolor versicolor  
## [13] versicolor versicolor versicolor versicolor versicolor versicolor  
## [19] versicolor versicolor versicolor versicolor virginica virginica   
## [25] virginica virginica versicolor virginica virginica virginica   
## [31] virginica virginica versicolor virginica virginica virginica   
## [37] virginica virginica   
## Levels: setosa versicolor virginica

table(pred3, testData$Species)

##   
## pred3 setosa versicolor virginica  
## setosa 10 0 0  
## versicolor 0 12 2  
## virginica 0 0 14

plot(pred3, main = "Tree 3 Prediction")



# Let's predict our model with the help of testing data set which was initially splitted as testData with 305 of main data set. For this, we have to use predict() method on the testing data set. In this, lets mix and match various columns to implement several various decision tree models and compare the prediction results of each.  
  
# From the plots, we can depict that there is change in each prediction with respect to the columns. If the independent variable is tweaked then the final result will also get altered with respect to the columns

# Problem 5  
  
set.seed(9000)  
irisData2 <- iris  
irisData2

irisData2$Species <- NULL  
irisData2$species

## NULL

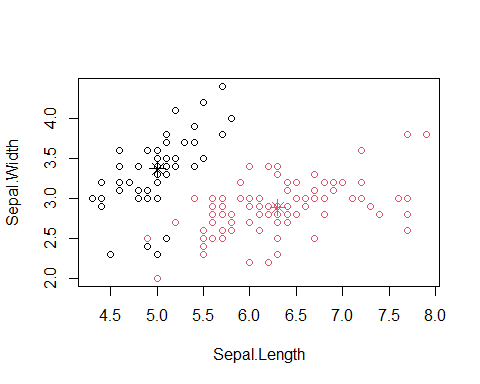
irisKMeansClustering2 <- kmeans(irisData2, 2)  
irisKMeansClustering2

## K-means clustering with 2 clusters of sizes 53, 97  
##   
## Cluster means:  
## Sepal.Length Sepal.Width Petal.Length Petal.Width  
## 1 5.005660 3.369811 1.560377 0.290566  
## 2 6.301031 2.886598 4.958763 1.695876  
##   
## Clustering vector:  
## [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 1 1 1 1 1 1 1 1  
## [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2  
## [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [149] 2 2  
##   
## Within cluster sum of squares by cluster:  
## [1] 28.55208 123.79588  
## (between\_SS / total\_SS = 77.6 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss" "tot.withinss"  
## [6] "betweenss" "size" "iter" "ifault"

table(irisKMeansClustering2$cluster, iris$Species)

##   
## setosa versicolor virginica  
## 1 50 3 0  
## 2 0 47 50

plot(iris[c("Sepal.Length", "Sepal.Width")], col = irisKMeansClustering2$cluster)  
points(  
 irisKMeansClustering2$centers[, c("Sepal.Length", "Sepal.Width")],  
 col = 1:3,  
 pch = 8,  
 cex = 1.5  
)



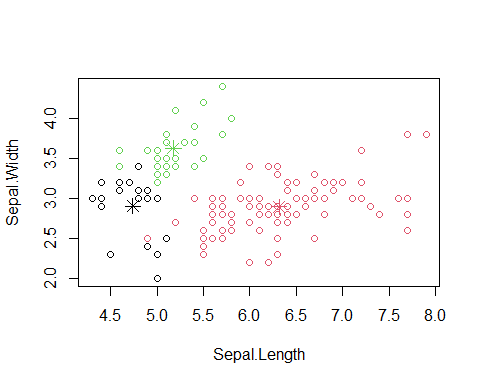
irisKMeansClustering3 <- kmeans(irisData2, 3)  
irisKMeansClustering3

## K-means clustering with 3 clusters of sizes 21, 96, 33  
##   
## Cluster means:  
## Sepal.Length Sepal.Width Petal.Length Petal.Width  
## 1 4.738095 2.904762 1.790476 0.3523810  
## 2 6.314583 2.895833 4.973958 1.7031250  
## 3 5.175758 3.624242 1.472727 0.2727273  
##   
## Clustering vector:  
## [1] 3 1 1 1 3 3 3 3 1 1 3 3 1 1 3 3 3 3 3 3 3 3 3 3 1 1 3 3 3 1 1 3 3 3 1 3 3  
## [38] 3 1 3 3 1 1 3 3 1 3 1 3 3 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2  
## [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [149] 2 2  
##   
## Within cluster sum of squares by cluster:  
## [1] 17.669524 118.651875 6.432121  
## (between\_SS / total\_SS = 79.0 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss" "tot.withinss"  
## [6] "betweenss" "size" "iter" "ifault"

table(irisKMeansClustering3$cluster, iris$Species)

##   
## setosa versicolor virginica  
## 1 17 4 0  
## 2 0 46 50  
## 3 33 0 0

plot(iris[c("Sepal.Length", "Sepal.Width")], col = irisKMeansClustering3$cluster)  
points(  
 irisKMeansClustering3$centers[, c("Sepal.Length", "Sepal.Width")],  
 col = 1:3,  
 pch = 8,  
 cex = 1.5  
)



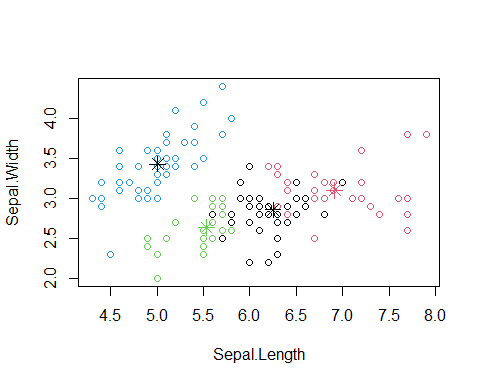
irisKMeansClustering4 <- kmeans(irisData2, 4)  
irisKMeansClustering4

## K-means clustering with 4 clusters of sizes 40, 32, 28, 50  
##   
## Cluster means:  
## Sepal.Length Sepal.Width Petal.Length Petal.Width  
## 1 6.252500 2.855000 4.815000 1.625000  
## 2 6.912500 3.100000 5.846875 2.131250  
## 3 5.532143 2.635714 3.960714 1.228571  
## 4 5.006000 3.428000 1.462000 0.246000  
##   
## Clustering vector:  
## [1] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4  
## [38] 4 4 4 4 4 4 4 4 4 4 4 4 4 1 1 1 3 1 3 1 3 1 3 3 3 3 1 3 1 3 3 1 3 1 3 1 1  
## [75] 1 1 1 1 1 3 3 3 3 1 3 1 1 1 3 3 3 1 3 3 3 3 3 1 3 3 2 1 2 2 2 2 3 2 2 2 1  
## [112] 1 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 1 2 2 2 1 2 2 2 1 2 2 2 1 1  
## [149] 2 1  
##   
## Within cluster sum of squares by cluster:  
## [1] 13.624750 18.703437 9.749286 15.151000  
## (between\_SS / total\_SS = 91.6 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss" "tot.withinss"  
## [6] "betweenss" "size" "iter" "ifault"

table(irisKMeansClustering4$cluster, iris$Species)

##   
## setosa versicolor virginica  
## 1 0 23 17  
## 2 0 0 32  
## 3 0 27 1  
## 4 50 0 0

plot(iris[c("Sepal.Length", "Sepal.Width")], col = irisKMeansClustering4$cluster)  
points(  
 irisKMeansClustering4$centers[, c("Sepal.Length", "Sepal.Width")],  
 col = 1:3,  
 pch = 8,  
 cex = 1.5  
)



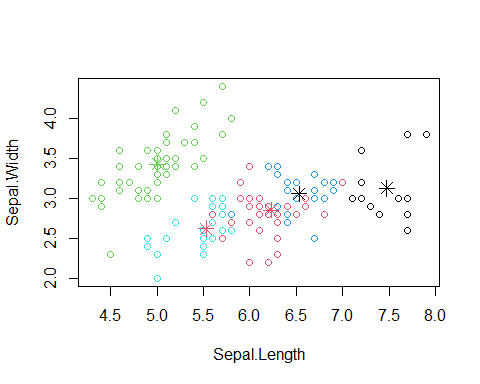
irisKMeansClustering5 <- kmeans(irisData2, 5)  
irisKMeansClustering5

## K-means clustering with 5 clusters of sizes 12, 37, 50, 24, 27  
##   
## Cluster means:  
## Sepal.Length Sepal.Width Petal.Length Petal.Width  
## 1 7.475000 3.125000 6.300000 2.050000  
## 2 6.229730 2.851351 4.767568 1.572973  
## 3 5.006000 3.428000 1.462000 0.246000  
## 4 6.529167 3.058333 5.508333 2.162500  
## 5 5.529630 2.622222 3.940741 1.218519  
##   
## Clustering vector:  
## [1] 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  
## [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 5 2 5 2 5 2 5 5 5 5 2 5 2 2 5 2 5 2 5 2 2  
## [75] 2 2 2 2 2 5 5 5 5 2 5 2 2 2 5 5 5 2 5 5 5 5 5 2 5 5 4 2 1 4 4 1 5 1 4 1 4  
## [112] 4 4 2 4 4 4 1 1 2 4 2 1 2 4 1 2 2 4 1 1 1 4 2 2 1 4 4 2 4 4 4 2 4 4 4 2 4  
## [149] 4 2  
##   
## Within cluster sum of squares by cluster:  
## [1] 4.655000 11.963784 15.151000 5.462500 9.228889  
## (between\_SS / total\_SS = 93.2 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss" "tot.withinss"  
## [6] "betweenss" "size" "iter" "ifault"

table(irisKMeansClustering5$cluster, iris$Species)

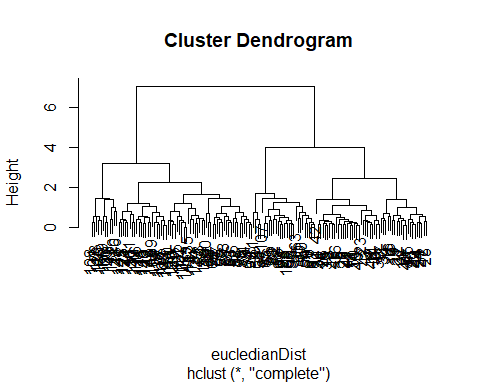
##   
## setosa versicolor virginica  
## 1 0 0 12  
## 2 0 24 13  
## 3 50 0 0  
## 4 0 0 24  
## 5 0 26 1

plot(iris[c("Sepal.Length", "Sepal.Width")], col = irisKMeansClustering5$cluster)  
points(  
 irisKMeansClustering5$centers[, c("Sepal.Length", "Sepal.Width")],  
 col = 1:3,  
 pch = 8,  
 cex = 1.5  
)



# We have set the seed to 9000 and duplicated the data set with irisData2. Then removed the Species column by assigning it to the NULL. Let's perform some K-Means Clustering model also known as K-Means/ K Nearest Neighbors which segregates the the whole data set into various clusters/ groups/ partitions. After this, it further splits the data into alike clusters. groups/ partitions as nearer as available and cluster as apart as possible.  
  
# We don't know the data sets features and parameters they show. And , also the clustering technique will use numerical data and drop non-numeric columns. After that, we will perform clustering by using the kmeans() method with 2 clusters with our normalized data set. Once it is performed, we again feed with 3 clusters, and then 4, and then 5.  
  
# By doing like so, we can observe the change in the distances between them. The Euclidean Distances becomes lesser with increase in the clusters. Here, we can plot to see the clusters.

# Problem 6  
  
eucledianDist <- dist(irisData2)  
hierarachyCluster <- hclust(eucledianDist)  
plot(hierarachyCluster)

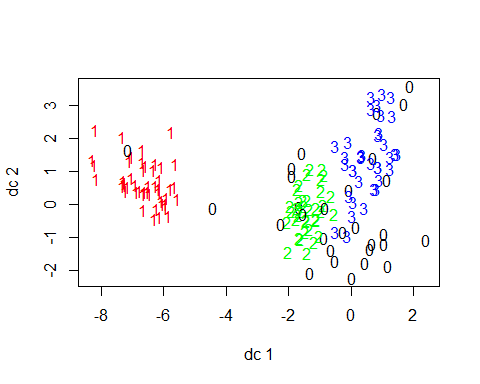


# In this, we perform Hierarchical Clustering and it is not necessary to specify the number of clusters required to get the best result. This is known as a Dendogram and it looks like an inverted tree structure.

# Problem 7  
  
densityCluster <- dbscan(irisData2, eps = 0.42, MinPts = 5)  
table(densityCluster$cluster, iris$Species)

##   
## setosa versicolor virginica  
## 0 2 10 17  
## 1 48 0 0  
## 2 0 37 0  
## 3 0 3 33

plotcluster(irisData2, densityCluster$cluster)



# In this, let us remove the Species ID's and use dbscan() method for clustering with the irisData2 data set. After that, let us compare with the original Species ID's in a table and plot the clusters. We can see few 0's in the graphs which are looked as outliers.