The Sparks Foundation: Graduate Rotational Internship Program July 2021

Domain: Data Science & Business Analytics

Name: Anna Miriam Philip

Project Topic: Prediction using Unsupervised Machine Learning

Project Level: Beginner

Project Aim: To find the optimum number of clusters from the given dataset and visualize them.

Programming Language: R

Algorithm: K - Means Clustering - Unsupervised Algorithm

Dataset: Iris - Open source dataset

This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.

"iris" is a data frame with 150 cases (rows) and 5 variables (columns) named Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, and Species.

Installing Required Libraries:

```
# install.packages("patchwork")
# install.packages("tidyverse")
# install.packages("gridExtra")
# install.packages("ggExtra")
# install.packages("gtable")
# install.packages("ggpubr")
```

Exploratory Data Analysis:

```
# Loading the Data
View(iris)

# Shape of the Data
dim(iris)
```

```
## [1] 150   5
```

```
# Top 10 rows of the Data
head(iris, n=10)
```

```
##    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
## 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
```

```
# Structure of the Data
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 ...
```

```
# Summary of the Data
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
##
##
##
```

Observations:

1. No NA values are present in this dataset.

2. We have 50 numbers each of the species - Setosa, Versicolor & Virginica.

3. The mean and median are not far apart thus indicating less number of outliers.

Reading the Data into a .csv file

```
write.csv(iris,"irisdataset.csv")
irisdf = read.csv("irisdataset.csv")
```

Data Visualization:

```
library(gridExtra)
```

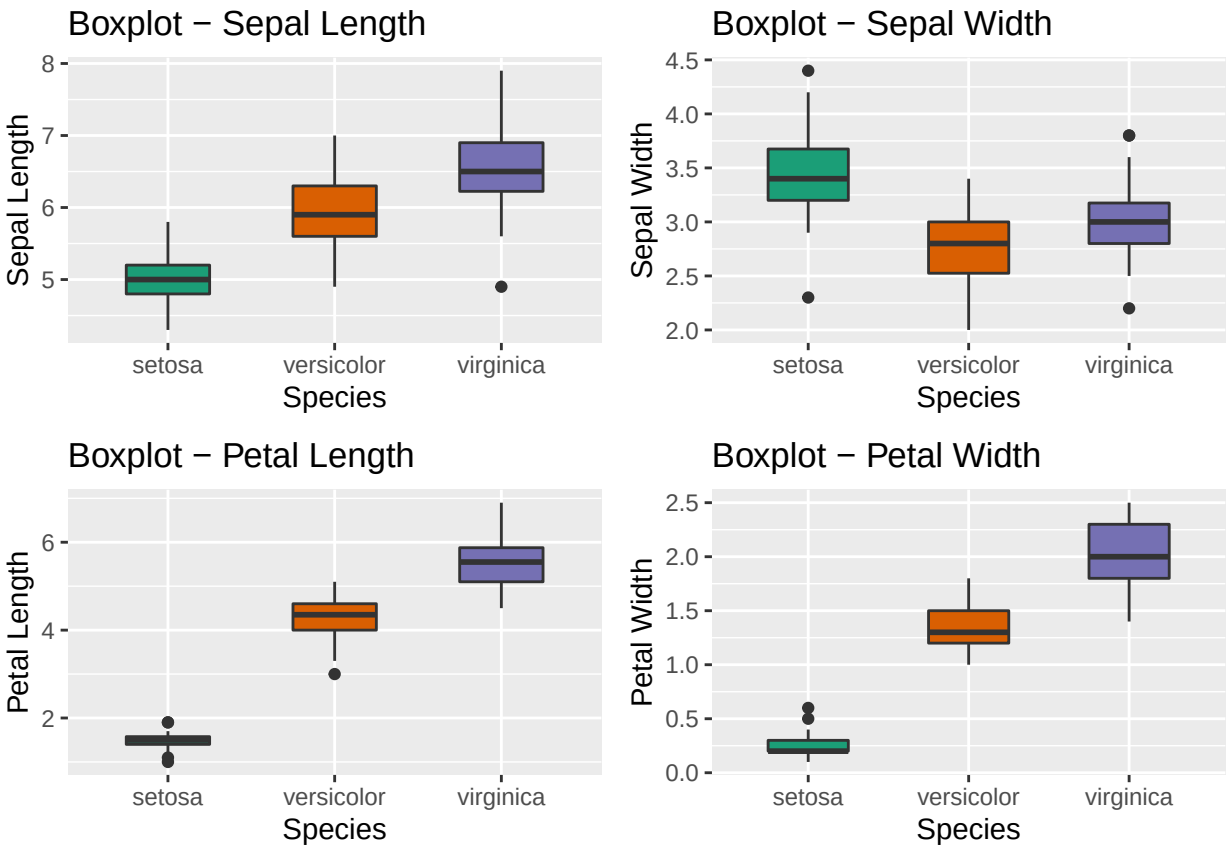
Boxplots - to examine outliers:

```
## Warning: package 'gridExtra' was built under R version 4.0.5
```

```
library(ggplot2)
library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 4.0.5
```

```
# SEPAL LENGTH:
plot1 = ggplot(data = iris)+ geom_boxplot(aes(x = Species,
                                              y = Sepal.Length,
                                              fill = Species), width=0.5) +
  labs(title = "Boxplot - Sepal Length",
       x = "Species", y = "Sepal Length") + theme(legend.position = "none")

plot1 = plot1 + scale_fill_brewer(palette = "Dark2")

# SEPAL WIDTH:
plot2 = ggplot(data = iris)+ geom_boxplot(aes(x = Species,
                                              y = Sepal.Width,
                                              fill = Species), width=0.5) +
  labs(title = "Boxplot - Sepal Width", x = "Species",
       y = "Sepal Width") + theme(legend.position = "none")

plot2 = plot2 + scale_fill_brewer(palette = "Dark2")

# PETAL LENGTH:
plot3 = ggplot(data = iris)+ geom_boxplot(aes(x = Species,
                                              y = Petal.Length,
                                              fill = Species), width = 0.5) +
  labs(title = "Boxplot - Petal Length", x = "Species",
       y = "Petal Length") + theme(legend.position = "none")

plot3 = plot3 + scale_fill_brewer(palette = "Dark2")

# PETAL WIDTH:
plot4 = ggplot(data = iris)+ geom_boxplot(aes(x = Species,
                                              y = Petal.Width,
                                              fill = Species), width = 0.5) +
  labs(title = "Boxplot - Petal Width", x = "Species",
       y = "Petal Width") + theme(legend.position = "none")

plot4 = plot4 + scale_fill_brewer(palette = "Dark2")

grid.arrange(plot1, plot2, plot3, plot4, ncol = 2)
```



Observations:

1. We have some outliers in all the columns.

2. Setosa and Virginica are respectively the smallest and the largest of the flowers.

3. Sepal Width is different from the other attributes.

Correlations:

```
# Correlation:
cor(iris[1:4])
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
## Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
## Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
## Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```

```
# Kendall's Correlation Method:
# The Kendall rank correlation coefficient or Kendall's tau statistic is used to estimate a rank-based
cor(iris[1:4], method="kendall")
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length   1.00000000 -0.07699679    0.7185159   0.6553086
## Sepal.Width   -0.07699679  1.00000000   -0.1859944  -0.1571257
```

```
## Petal.Length   0.71851593 -0.18599442    1.0000000   0.8068907
## Petal.Width    0.65530856 -0.15712566    0.8068907   1.0000000
```

```
# Spearman's Correlation Method:
# Spearman's rho statistic is used to estimate a rank-based measure of association. This test may be us
cor(iris[1:4], method="spearman")
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    1.0000000  -0.1667777    0.8818981   0.8342888
## Sepal.Width    -0.1667777   1.0000000   -0.3096351  -0.2890317
## Petal.Length    0.8818981  -0.3096351    1.0000000   0.9376668
## Petal.Width     0.8342888  -0.2890317    0.9376668   1.0000000
```

```
# Correlation between Sepal.Length and Sepal.Width
cor(iris[1:2])
```

```
##              Sepal.Length Sepal.Width
## Sepal.Length    1.0000000  -0.1175698
## Sepal.Width    -0.1175698   1.0000000
```

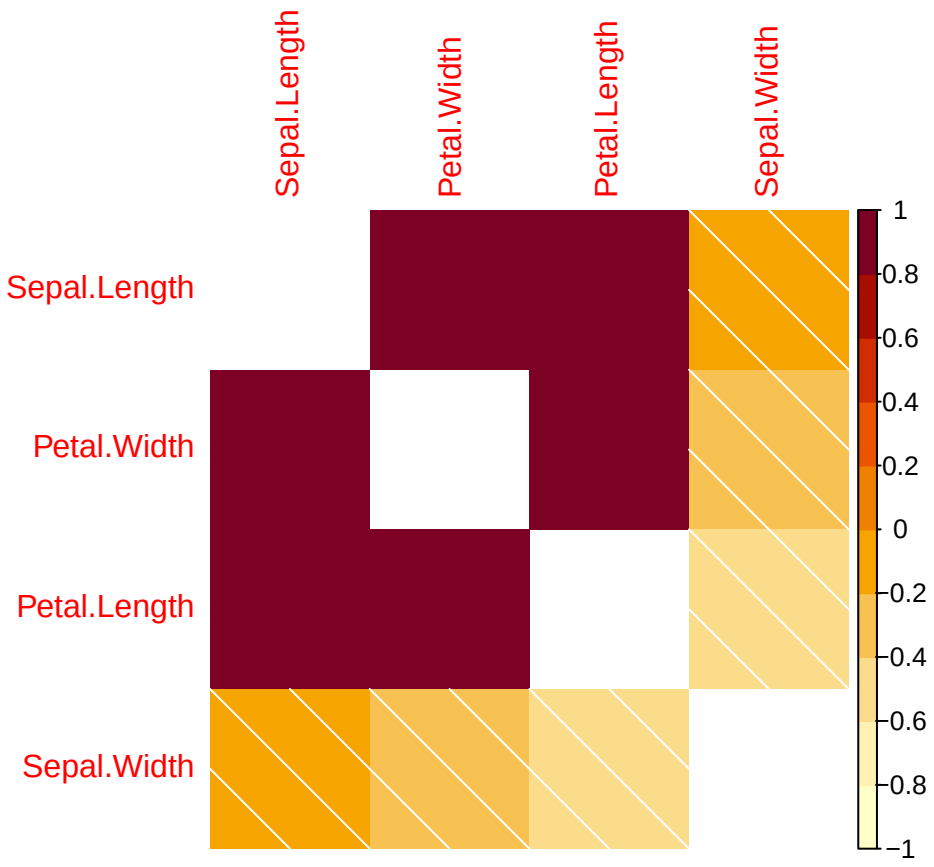
```
# Correlation between Petal.length and Petal.Width
cor(iris[3:4])
```

```
##              Petal.Length Petal.Width
## Petal.Length    1.0000000   0.9628654
## Petal.Width     0.9628654   1.0000000
```

```
cr = cor(iris[1:4])

library(corrplot)
```

```
## corrplot 0.90 loaded
```

```
col3 = hcl.colors(10, "YlOrRd", rev = TRUE)
corrplot(cr, method = 'shade', order = 'AOE', col = col3, diag = FALSE)
```

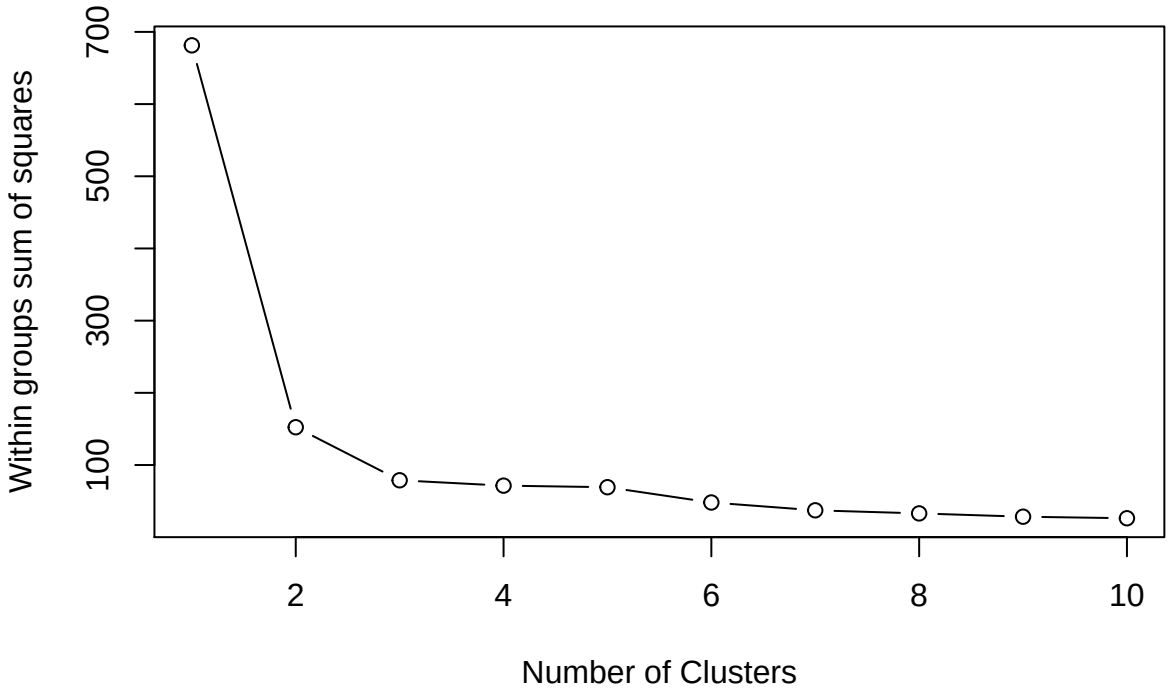


Finding the optimal number of clusters:

During our implementation, we need to calculate *"With-In-Sum-Of-Squares (WSS)"* iteratively.

```
## Finding optimal number of clusters from WSS

wssplot = function(data, nc = 15, seed = 123){
  wss = (nrow(data)-1)*sum(apply(data,2,var))
  for (i in 2:nc){
    set.seed(seed)
    wss[i] = sum(kmeans(data, centers = i)$withinss)}
  plot(1:nc, wss, type = "b", xlab = "Number of Clusters",
       ylab = "Within groups sum of squares")}
wssplot(irisdf[,2:5], nc = 10)
```

WSS is a measure to explain the homogeneity within a cluster. Let's create a function to plot WSS against the number of clusters, so that we can call it iteratively whenever required (Function name – "wssplot"). We will be using "NbClust" library for this illustration.

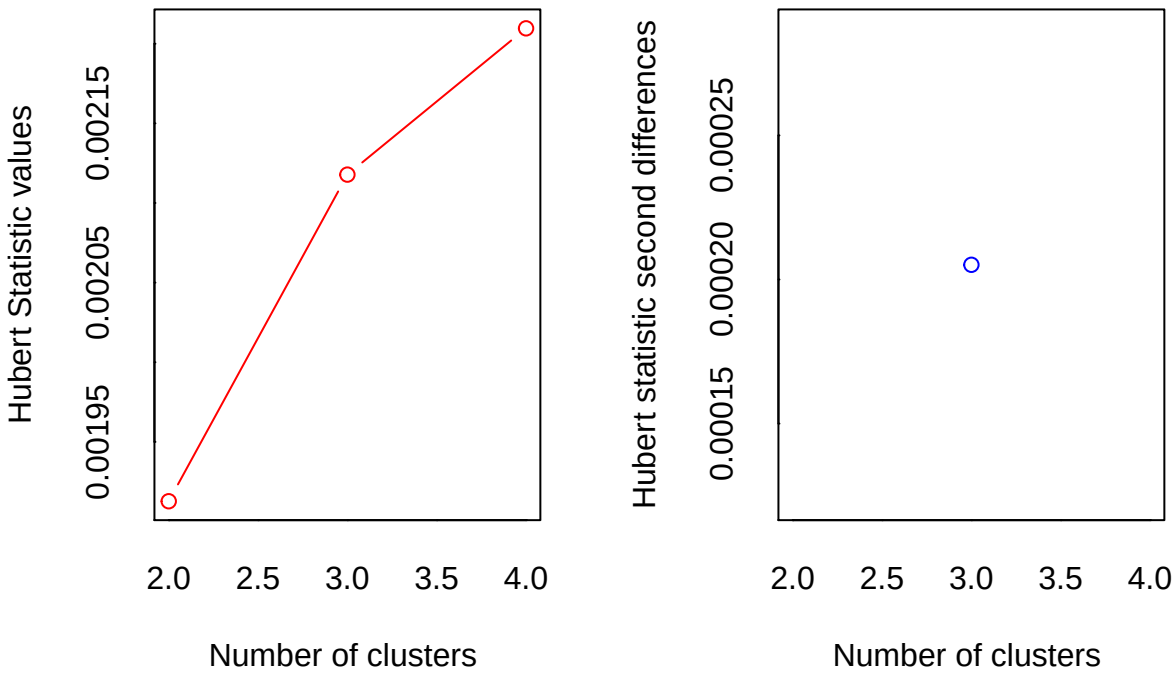


nc – maximum number of clusters we are giving

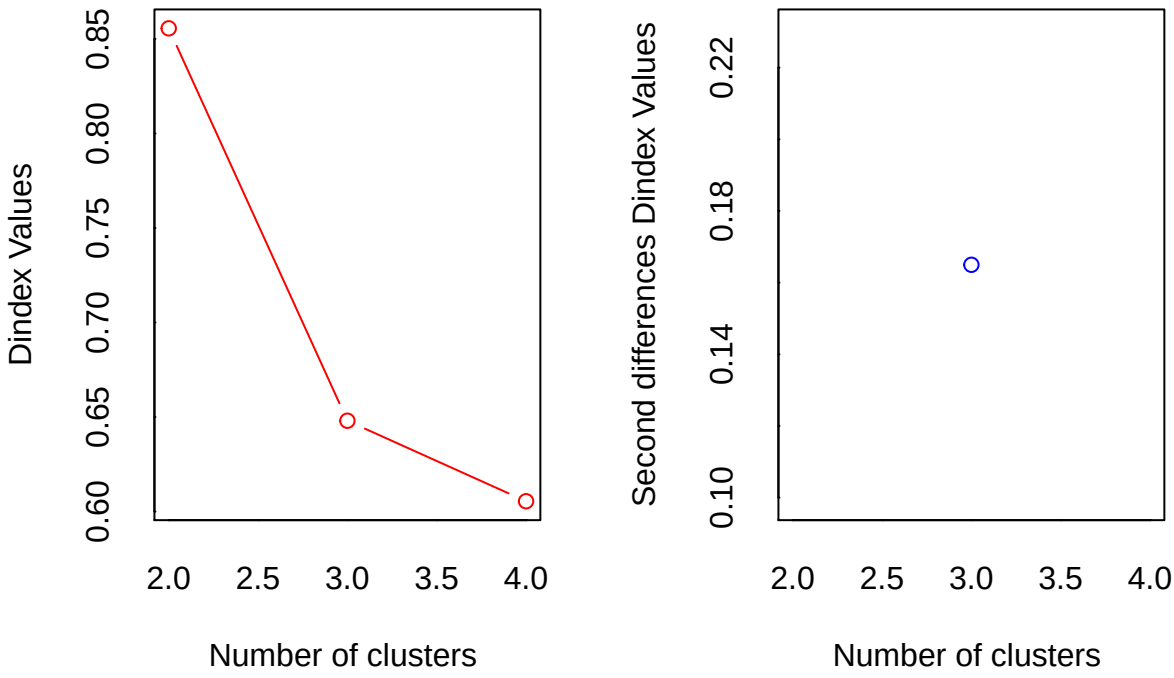
seed – random initialization of clusters

```
# Identifying the optimal number of clusters using NbClust:

library(NbClust)
set.seed(123)
Nclus = NbClust(irisdf[,2:5], min.nc = 2, max.nc = 4, method = "kmeans")
```

Here, we have plotted WSS with number of clusters. From here we can see that there is not much decrease in WSS even if we increase the number of clusters beyond 6. This graph is also known as "Elbow Curve" where the bending point (E.g, nc = 6 in our case) is known as "Elbow Point". From the above plot we can conclude that if we keep number of clusters = 2, we should be able to get good clusters with good homogeneity within themselves.




```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```

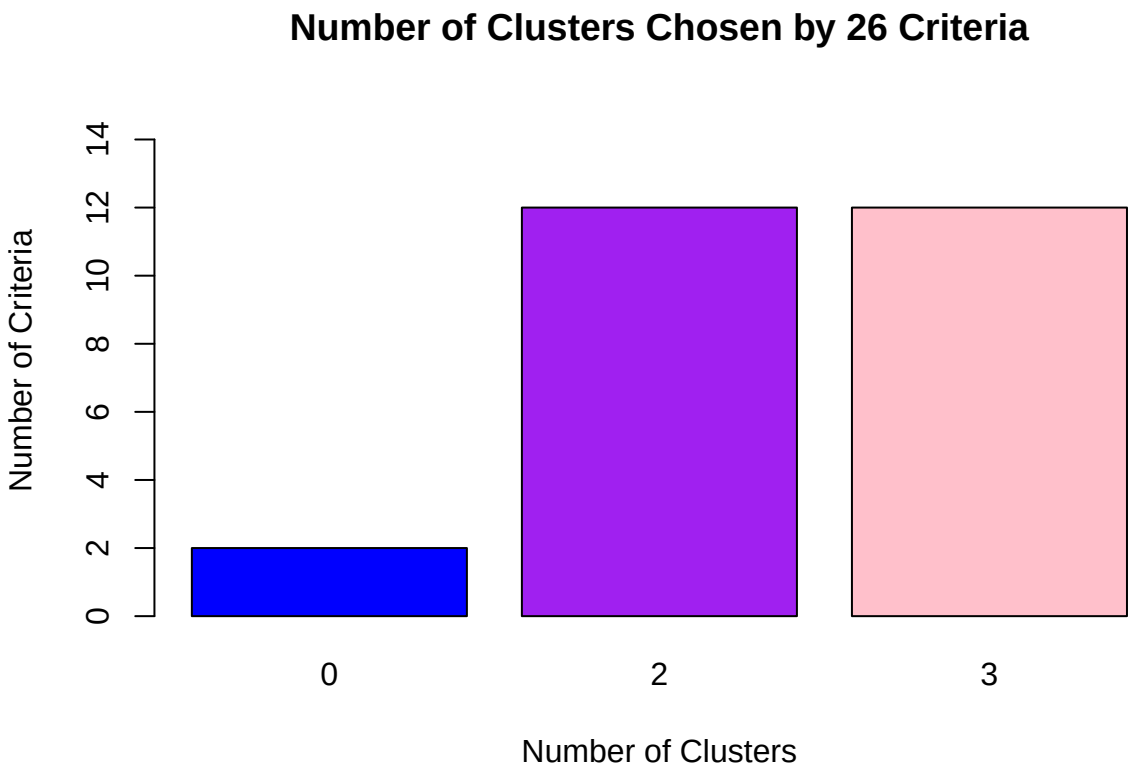



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 12 proposed 2 as the best number of clusters
## * 12 proposed 3 as the best number of clusters
##
##                    ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

```
table(Nclus$Best.n[1,])
```

```
##
##  0  2  3
##  2 12 12
```

```
barplot(table(Nclus$Best.n[1,]), ylim = range(0,15),
        xlab = "Number of Clusters", ylab = "Number of Criteria",
        main = "Number of Clusters Chosen by 26 Criteria", col = c("blue","purple","pink"))
```



Observations: **According to the majority rule, the best number of clusters is 2.**

Forming & Plotting the clusters:

```
kmeans_clust = kmeans(x = irisdf[,2:5], centers = 2, nstart = 5)
kmeans_clust
```

```
## K-means clustering with 2 clusters of sizes 97, 53
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     6.301031    2.886598     4.958763    1.695876
## 2     5.005660    3.369811     1.560377    0.290566
##
## Clustering vector:
##   [1] 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
##  [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
## [112] 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
## [149] 1 1
##
## Within cluster sum of squares by cluster:
## [1] 123.79588  28.55208
##  (between_SS / total_SS =  77.6 %)
##
## Available components:
```

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

Observations:

1. This is a K-means clustering with 2 clusters of sizes 97 and 53.

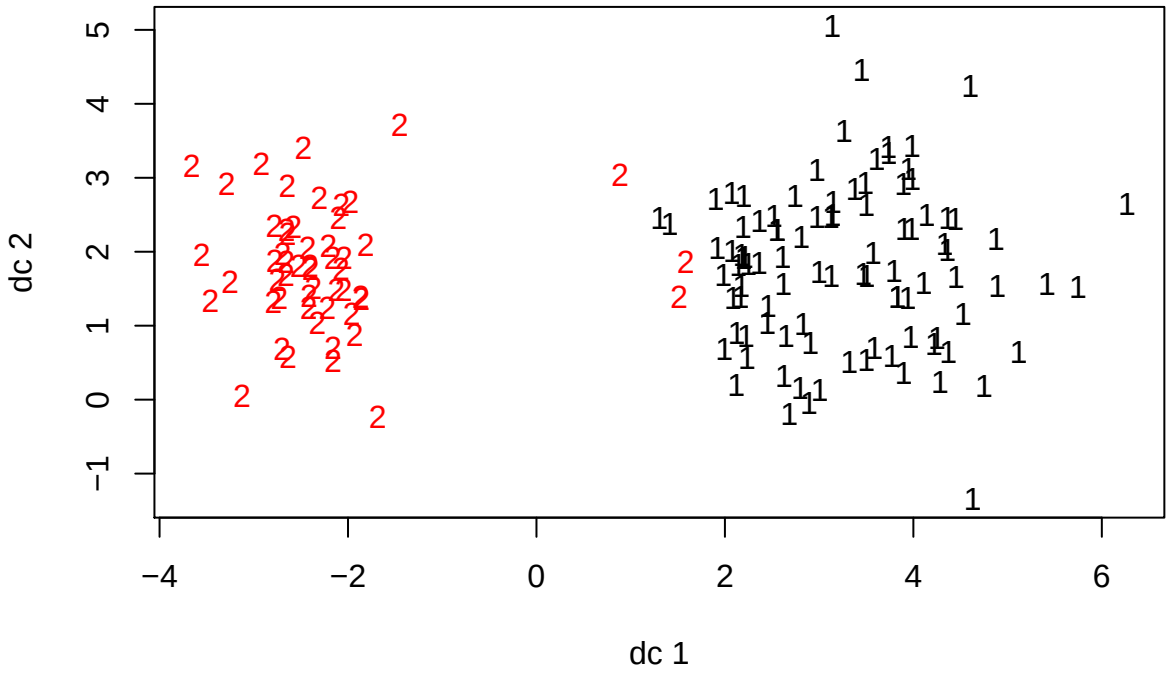
2. The percentage similarity between the data in the same cluster is 77.6%.

Plotting the clusters:

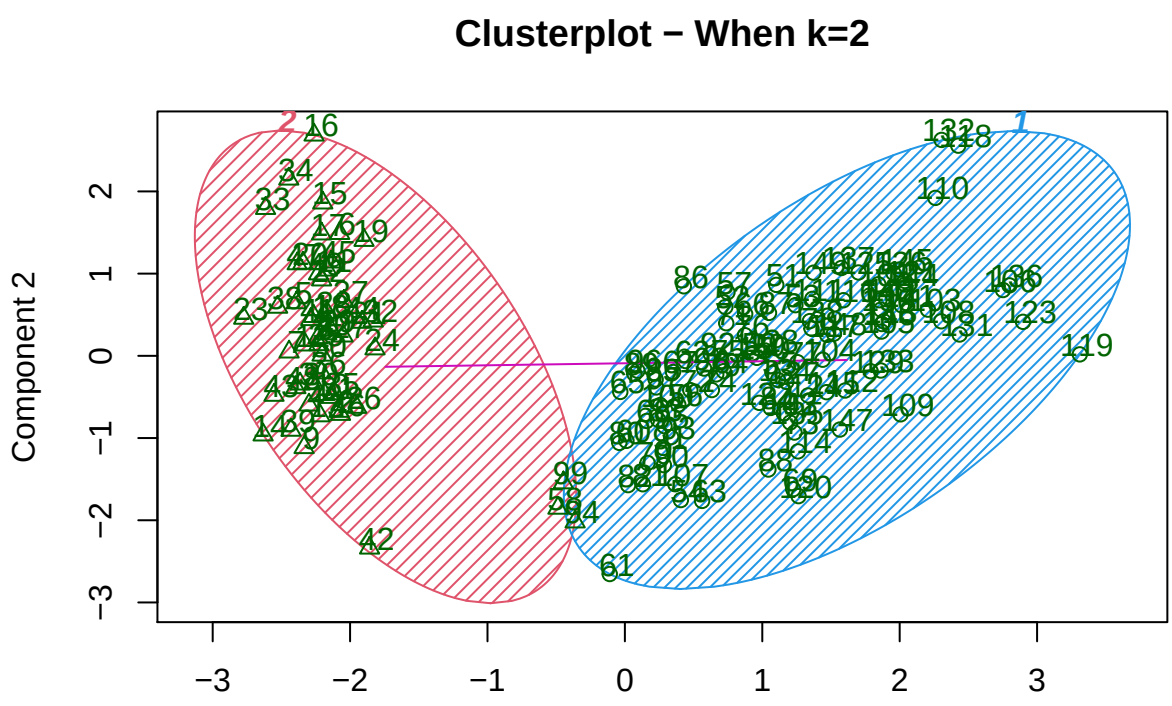
```
library(fpc)
```

```
## Warning: package 'fpc' was built under R version 4.0.5
```

```
library(cluster)
plotcluster(irisdf[,2:5], kmeans_clust$cluster)
```




```
# Better plot:
clusplot(irisdf[,2:5], main = "Clusterplot - When k=2",
         kmeans_clust$cluster,
         color = TRUE, shade = TRUE, labels = 2, lines = 1)
```



Clustering Validation - Silhouette Width:

The term *cluster validation* is used to design the procedure of evaluating the goodness of clustering algorithm results. This is important to avoid finding patterns in a random data, as well as, in the situation where you want to compare two clustering algorithms.

```
library(factoextra)
```

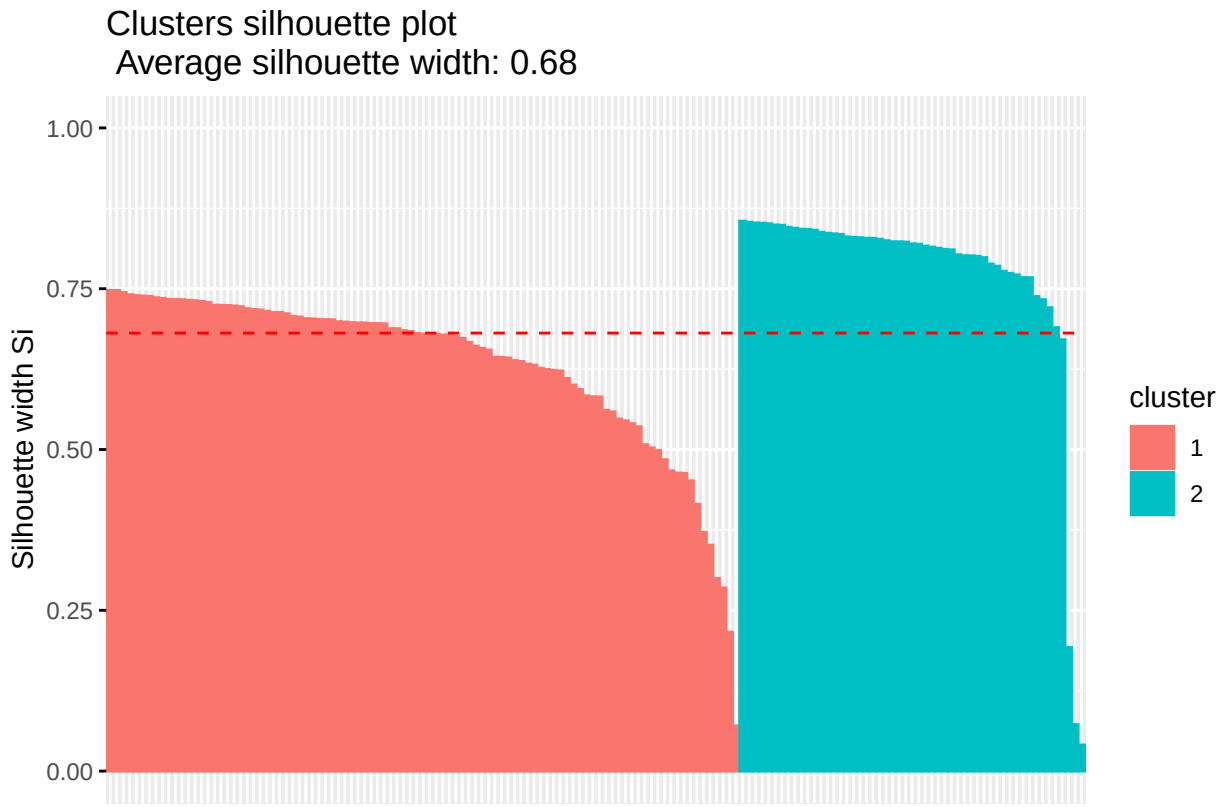
The *silhouette analysis* measures how well an observation is clustered and it estimates the average distance between clusters. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters.

```
## Warning: package 'factoextra' was built under R version 4.0.5
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
# When k = 2

sil_k2 = silhouette(kmeans_clust$cluster, dist(irisdf[,2:5]))
fviz_silhouette(sil_k2)
```

```
##   cluster size ave.sil.width
## 1       1   97          0.63
## 2       2   53          0.77
```



Observations:

1. Cluster 1 (97 data points) has an average silhouette width of 0.63.

2. Cluster 2 (53 data points) has an average silhouette width of 0.77.

3. Therefore, average silhouette width is 0.68.

Inference:

We have separated or categorised our data into two clusters. The data within each cluster are about 77.6% similar and the K-Means model has performed well having a Silhouette width of 0.68 on an average, making the model good for categorisation.