**Clustering**

Clustering is a data analysis and machine learning method that involves grouping similar objects or data points based on their similarity. The main objective of clustering is to uncover and identify natural groupings or patterns in a dataset. Clustering has main two classifications which are Partitioning clustering and Hierarchical clustering.

**Partitional Clustering**

Partitioning clustering is a frequently used method for categorizing related objects or data points into separate clusters. Partitioning clustering divides the data set into a predetermined number of clusters and assigns each data point to a cluster depending on how similar it is to other data points in that cluster. It is important to make sure that each cluster's items are as separate from those in other clusters as feasible while yet being as similar to one another as possible.

K-means is a method of grouping objects or data points into clusters based on their similarities. It works by selecting a pre-determined number of clusters and randomly assigning each data point to one of these clusters. The algorithm then calculates the center point, or centroid, of each cluster by taking the average of all the data points assigned to that cluster. This is repeated iteratively until the centroids no longer change or a maximum number of iterations is reached. The resulting clusters represent groups of similar data points located close together in the data space.

According to the assignment we are analysing the Statlog (Vehicle Silhouettes) Data Set which contains 846 sample instances of four vehicle models and 18 attribute features. All the data handling, preprocessing, and calculations were initiated through R language and RStudio.

**Objective 01 - Find the ideal number of clusters**

As the first step, load the dataset into the RStudio environment. By installing the **NbClust package,** I am determined to find the ideal number of clusters using the euclidean method.
In this dataset, we are analysing 4 models of vehicles such as van, bus, opel, or saab. We are assigning numbers in to each model to feature the clustering calculations.

| **Vehicle Type/ Model** | **Assigned No.** |
|---|---|
| Van | 1 |
| Bus | 2 |
| Opel | 3 |
| Saab | 4 |

Code is demonstrated as follows

```
# loading the dataset
vehicle = read.csv("C:/Users/Gihan/Documents/Data mining course work/Q2 clustering/vehicle.csv")

# Converting the vehicle models into numbers
vehicle$Class<-c(van=1,bus=2,opel=3,saab=4)[vehicle$Class]

# Print the dataset
head(vehicle)

# Loading the NbClust library
library(NbClust)
```

Next, the "class" column is removed from the data set, as it is the target variable and should not be included in the clustering.

Then, the "set.seed( )" function is used to set a random seed for reproducibility.
Using the euclidean method find Kmeans with min.nc=2, max.nc=15 parameter setting to test the minimum number of clusters is 02 and the maximum number of clusters is 15.

Code is demonstrated as follows

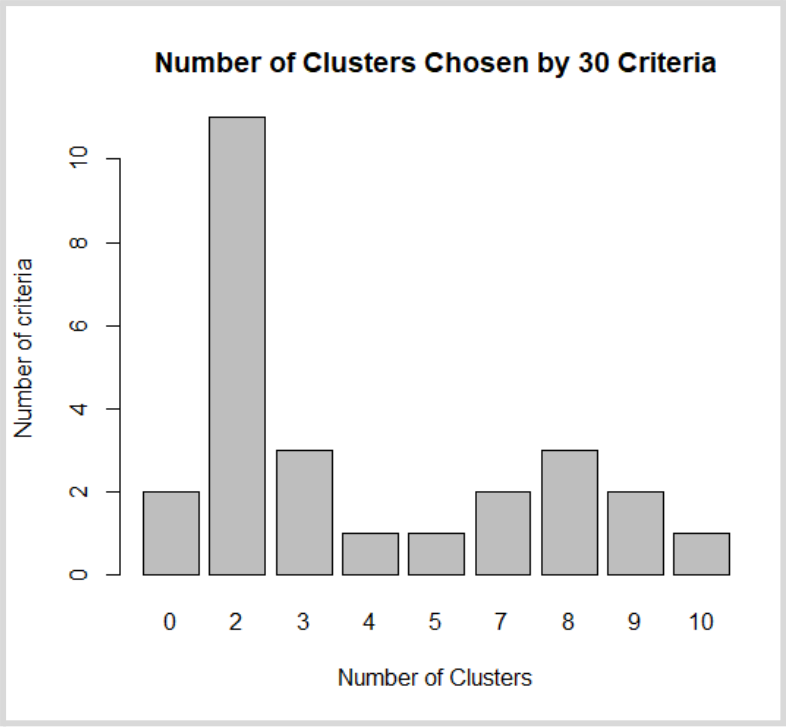
```
classes = vehicle$Class
vehicle$class=NULL

#set the seed to No.55 to ensure reproducibility
set.seed(55)

#Setting parameters for euclidean method to find No of clusters
clusterNo=NbClust(vehicle,distance="euclidean",
min.nc=2,max.nc=15,method="kmeans",index="all")
```

**Results output**

```
*******************************************************************
* Among all indices:
* 4 proposed 2 as the best number of clusters
* 8 proposed 3 as the best number of clusters
* 7 proposed 5 as the best number of clusters
* 1 proposed 6 as the best number of clusters
* 1 proposed 11 as the best number of clusters
* 2 proposed 15 as the best number of clusters

                 ***** Conclusion *****

* According to the majority rule, the best number of clusters is  3
```

We can visualise in a barplot above result. Code is demonstrated as follows

```
# Selecting the dataframe and scaling the data
str(vehicle)
data.train<-scale(vehicle[-1])
summary(data.train)

# Setting the parameter settings
set.seed(1500)
nc<-NbClust(data.train,
          min.nc = 2, max.nc = 10,
          method="kmeans")
table(nc$Best.n[1,])

# Plotting the bar chart
barplot(table(nc$Best.n[1,]),
       xlab="Number of Clusters",
       ylab="Number of criteria",
       main="Number of Clusters Chosen by 30 Criteria")
```



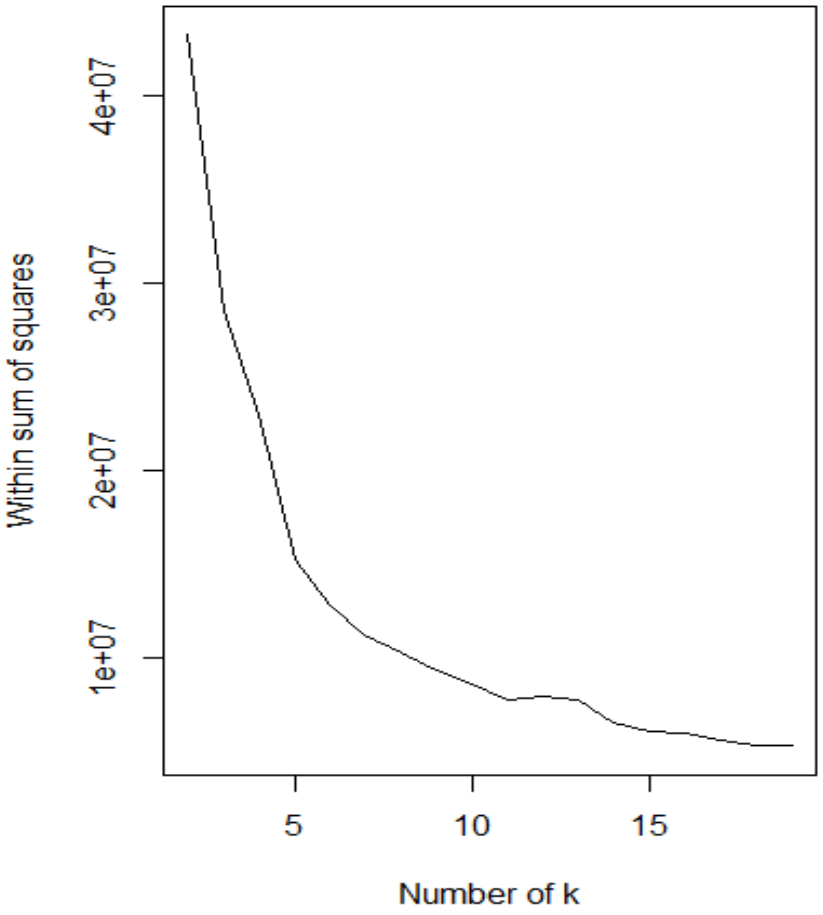

By analysing this bar chart we can consider best number of cluster are 02 and 03 with using the kmeans.

To further clarify and determine the best number of clusters are can plot the chart with elbow method by simply calculate the within Cluster Sum of Squares (WSS).

Code is demonstrated as follows

```
#Define a range of k values
k=2:19

#Set the seed for reproducibility
set.seed(55)

#Calculate the within sum of squares for each k value using k-means clustering
WSS=sapply(k,function(k){kmeans(vehicle[1:19],centers = k)$tot.withinss})

#Plot the results
plot(k, WSS, type="l", xlab= "Number of k", ylab="Within sum of squares")
```

**Results Output**




Based on the review of above two charts project the insights that best number of clusters are 02.

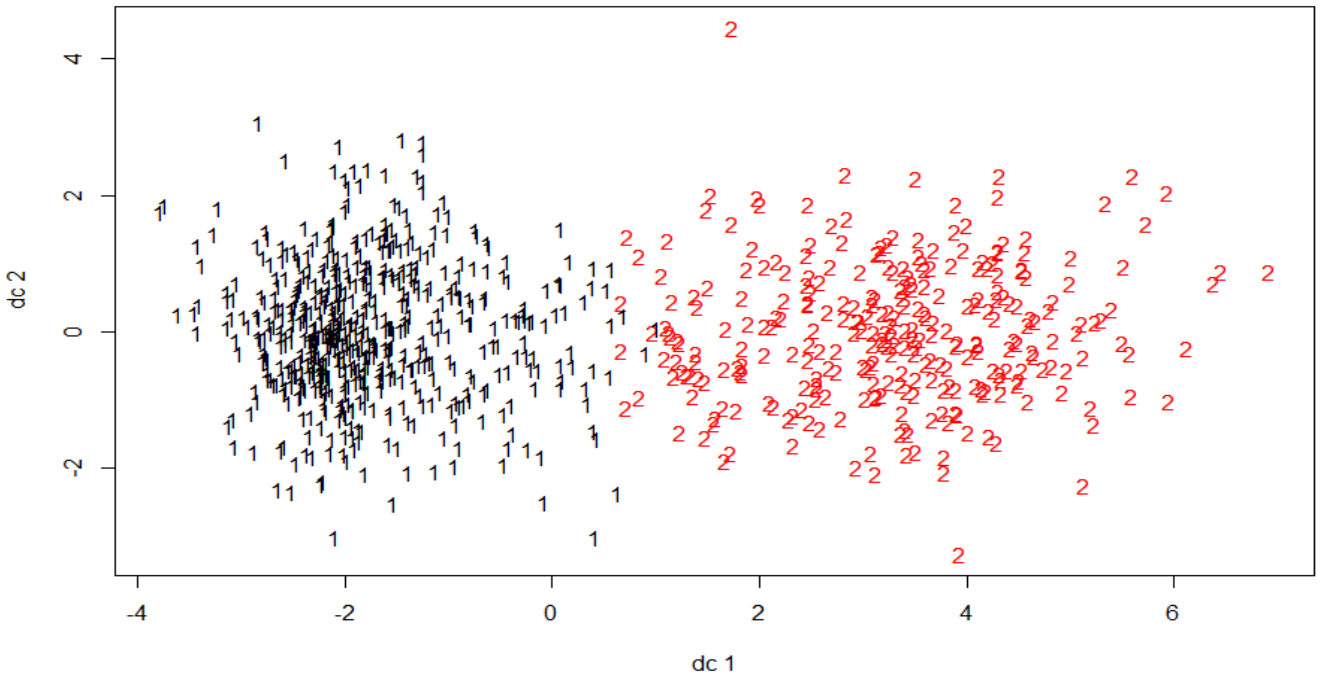
Although the best result is 02 clusters by calculating within Cluster Sum of Squares (WSS), we receive the best number of clusters is 03 from the euclidean method. To further clarify which number of clusters is optimum number, we need to evaluate the clustering results using an external validation metric. Therefore I am going to use the Adjusted Rand Index (ARI) method to prove the optimum number of clusters for both 02 and 03 by fitting the model.

**Fit the model for k=2**

Code is demonstrated as follows

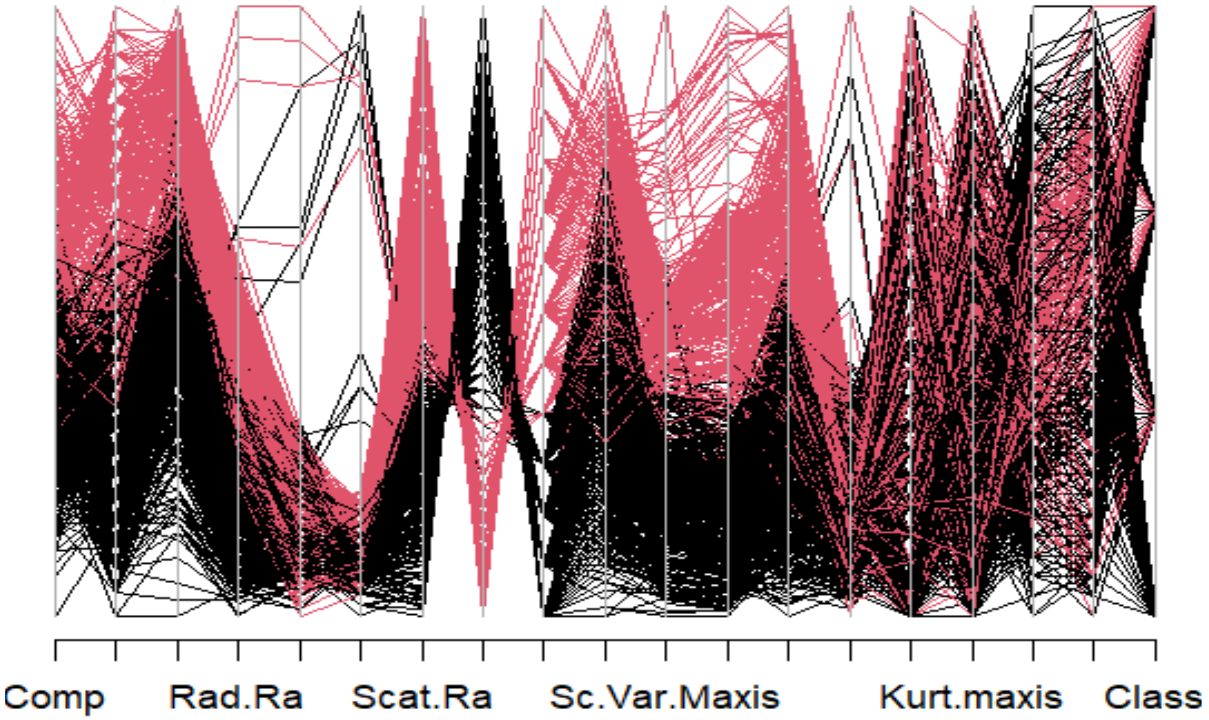
```
#k=2/Set the number of clusters to 2.
#Set the seed for reproducibility
set.seed(1500)

#Perform k-means clustering on scaled data with 2 clusters
fit.km<-kmeans(data.train,2)
fit.km

#Display cluster centers
fit.km$centers

#Display number of observations in each cluster
fit.km$size

#Load fpc package
installed.packages("fpc")
library(fpc)

#Plot data points with colored clusters using k-means results
plotcluster(data.train, fit.km$cluster)
```

With execution above code K-means clustering with 2 clusters, the resulting clusters have **sizes of 552 and 294** respectively. Two clusters have been plotted below.




I have initiated a parallel coordinates plot, which displays the values of each variable on a separate vertical axis and connects the values for each observation with a line. By examining the plot, I could identify patterns and relationships between variables that may have contributed to the formation of the clusters.
Code is demonstrated as follows

```
#Load the library
library(MASS)

# Plot parallel coordinates with colored clusters using k-means results
parcoord(data.train, fit.km$cluster)
```




We can evaluate the performance of the k-means clustering model by analysing a confusion matrix table.
Code is demonstrated as follows

```
#create a confusion matrix
confuseTable.km <- table(classes, fit.km$cluster)
confuseTable.km
```

| **Classes** | **1** | **2** |
|---|---|---|
| **1** | 195 | 4 |
| **2** | 162 | 56 |
| **3** | 95 | 117 |
| **4** | 100 | 117 |

Confusion matrix table interprets most of the data points distributed in one cluster and there are few data points distributed in other cluster

The Adjusted Rand Index (ARI) can be used to measure the level of agreement between the two cluster solutions. The following code computes the ARI using the randIndex() function from the flexclust library.

```
# Load the flexclust package
installed.packages("flexclust")
library(flexclust)

# Calculate the Adjusted Rand Index(ARI)
randIndex(confuseTable.km)
```

**Calculated ARI value is 0.08358285**
**This calculated ARI value interprets that the agreement between the cluster solutions has poor performance.**

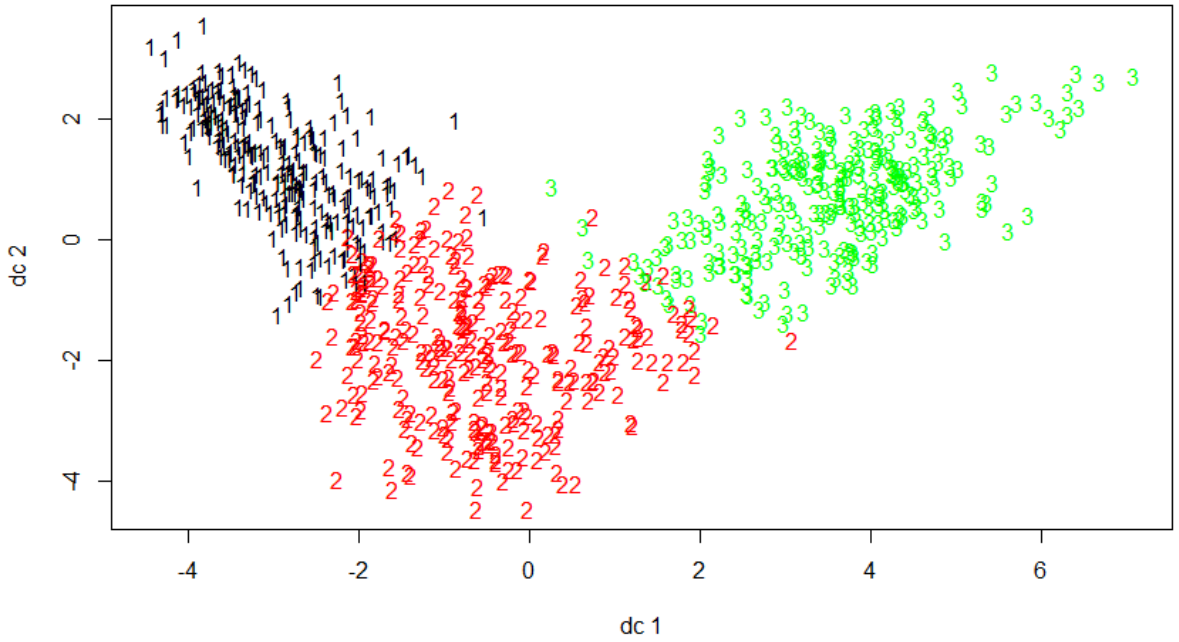
As I mentioned beginning of the task, there are number 02 and 03 suggested as best performing number of clusters. As we calculated the k=2, now we are following the same steps for the k=3 to calculate the ARI value.

**Fit the model for k=3**

Code is demonstrated as follows

```
#k=3/Set the number of clusters to 3

#Set the seed for reproducibility
set.seed(1500)

#Perform k-means clustering on scaled data with 2 clusters
fit.km<-kmeans(data.train,3)
fit.km

#Display cluster centers
fit.km$centers

#Display number of observations in each cluster
fit.km$size

#Load fpc package
installed.packages("fpc")
library(fpc)

#Plot data points with colored clusters using k-means results
plotcluster(data.train, fit.km$cluster)
```

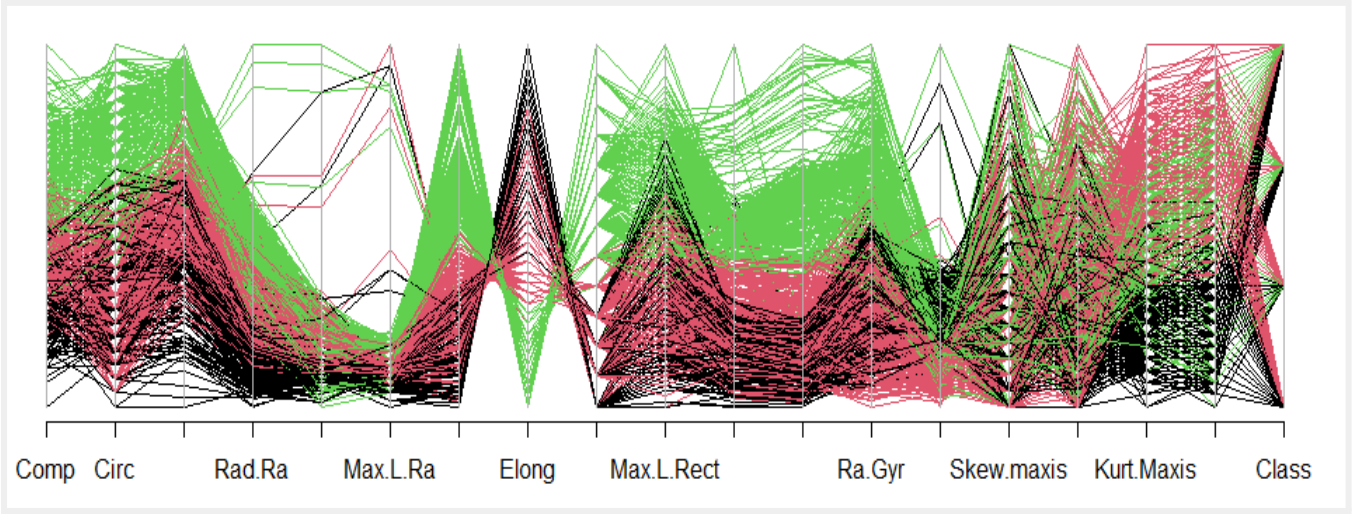
With execution above code K-means clustering with 3 clusters, the resulting clusters have **sizes of 275, 300 and 271** respectively. Three clusters have been plotted below.




Initiating a parallel coordinates plot  identify patterns and relationships between variables that may have contributed to the formation of the clusters.

Code is demonstrated as follows

```
#Load the library
library(MASS)

# Plot parallel coordinates with colored clusters using k-means results
parcoord(data.train, fit.km$cluster)
```



Evaluating the performance of the k-means clustering model by analysing a confusion matrix table.

Code is demonstrated as follows

```
#create a confusion matrix
confuseTable.km <- table(classes, fit.km$cluster)
confuseTable.km
```

| **Classes** | **1** | **2** | **3** |
|---|---|---|---|
| **1** | 105 | 90 | 4 |
| **2** | 96 | 73 | 49 |
| **3** | 36 | 64 | 112 |
| **4** | 38 | 73 | 106 |

Confusion matrix table interprets most of the data points distributed in three clusters. All the clusters reflects the moderate distribution of data points between four classes.

```
# Load the flexclust package
installed.packages("flexclust")
library(flexclust)

# Calculate the Adjusted Rand Index(ARI)
randIndex(confuseTable.km)
```

**Calculated ARI value is 0.07904893**
**This calculated ARI value interprets that the agreement between the cluster solutions has poor performance.**

**Data Preprocessing for optimum performance**

In order to enhance the accuracy of clustering metrics results several data preprocessing steps were initiated. Plus ARI value and confusion matrix value table computed in the stage of preprocessing.

**Preprocessing Method 01 - Dropping the highly correlated attributes**

Setting the cutoff value as **0.75** and Code is demonstrated as follows.

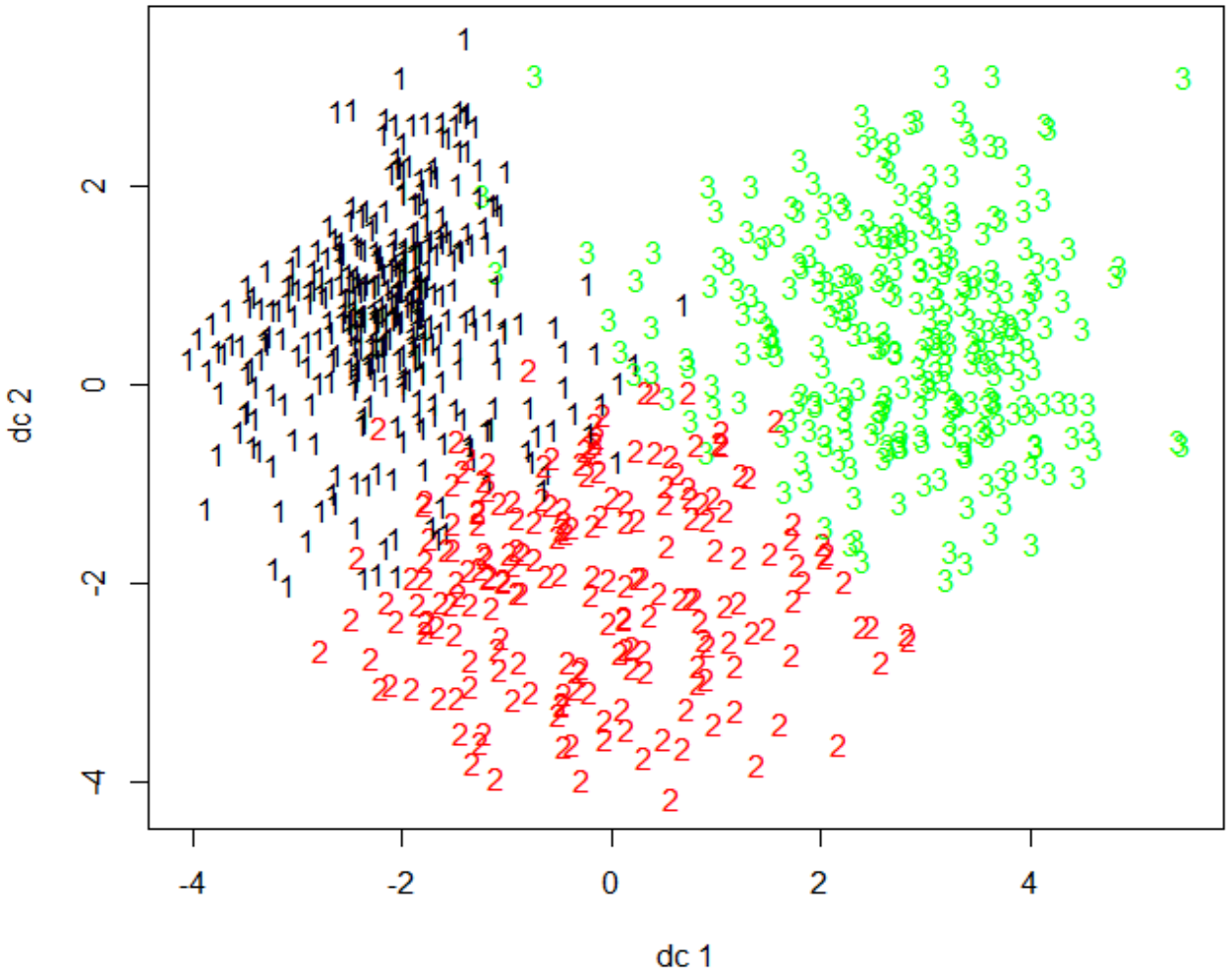
```
# Preprocess - Droping highly correlated columns

# Set the seed to ensure reproducibility of results
set.seed(50)

# load the library
library(caret)

# Load the vehicle data
#data(vehicle)

# Compute the correlation matrix for the first 19 columns of the data
correlationMatrix <- cor(vehicle[,1:19])
print(correlationMatrix)

# Find the highly correlated features using a correlation cutoff of 0.75
highycorrelated <- findCorrelation(correlationMatrix,cutoff = 0.75)
print(highycorrelated)

#Create an empty list to store the names of the highly correlated features
namelist = list()

# Extract the name of the column corresponding to the current index
for (x in highycorrelated) {
  #print(x)
  name <- colnames(vehicle[x])
  print(name[1])
  namelist <- append(namelist,name)
  #print(namelist)
}
#Print the list of highly correlated feature names
print(namelist)

# Remove the namelist
drop <- c(namelist)
print(drop)

#Print the dataset without highly correlated features
vehicle = vehicle[,!(names(vehicle) %in% drop)]
print(vehicle)
```

Code Result Output as follows.
These attributes are dropped from the dataset due to high corelation between other attributes. Since now, the dataset moving forward with the calculations with nine attributes.

```
[1] "D.Circ"
[1] "Elong"
[1] "Scat.Ra"
[1] "Sc.Var.Maxis"
[1] "Pr.Axis.Rect"
[1] "Sc.Var.maxis"
[1] "Circ"
[1] "Ra.Gyr"
[1] "Holl.Ra"
[1] "Kurt.Maxis"
```

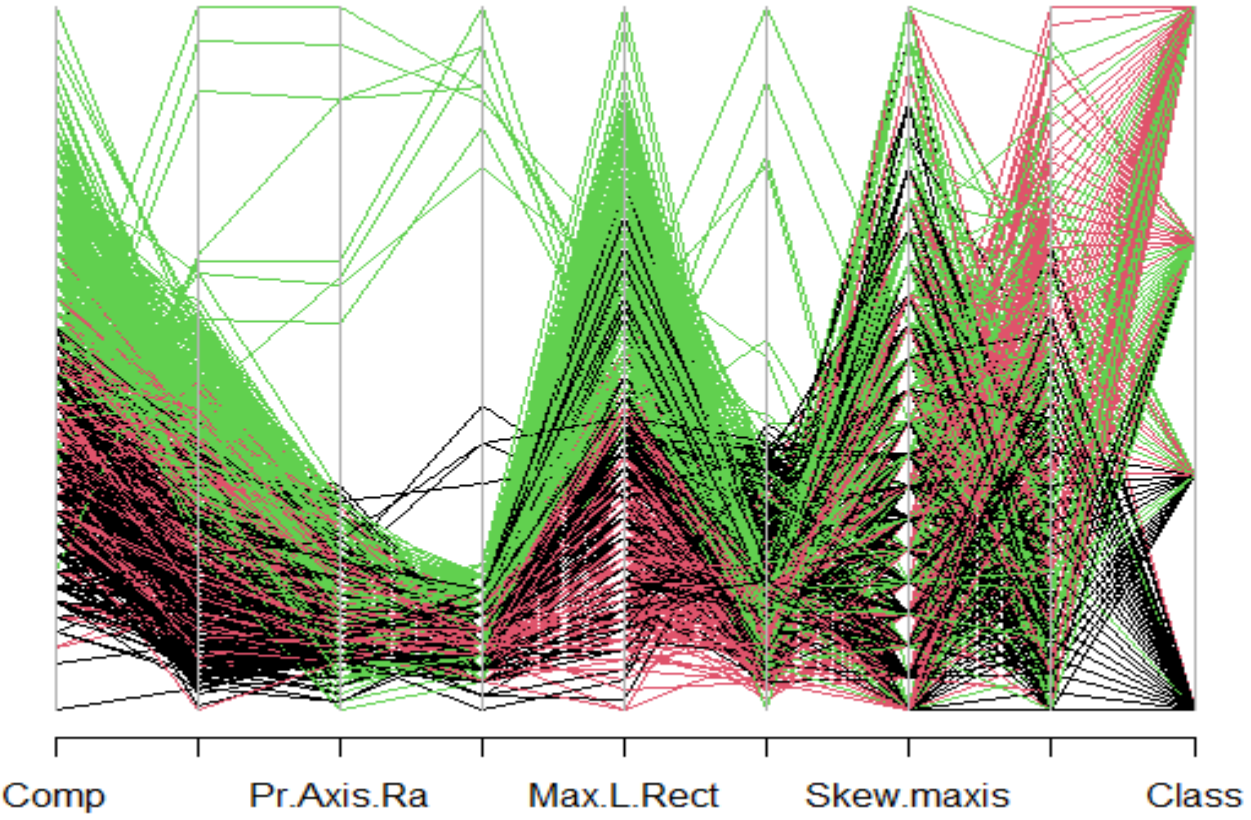
```
#Load fpc package
installed.packages("fpc")
library(fpc)
plotcluster(data.train, fit.km$cluster)
```

With execution above code preprocessing of data, the resulting clusters have **sizes of 358, 204 and 284** respectively. Three clusters have plotted below.



Creating a parallel coordinates plot to analyse the patterns and relationships between variables that may have contributed to the formation of the clusters.

```
#Load the library
library(MASS)
# Plot parallel coordinates with colored clusters using k-means results
parcoord(data.train, fit.km$cluster)
```




Evaluating the performance of the k-means clustering model by analysing a confusion matrix table.

Code is demonstrated as follows

```
#create a confusion matrix
confuseTable.km <- table(class, fit.km$cluster)
confuseTable.km
```

| **Classes** | **1** | **2** | **3** |
|---|---|---|---|
| **1** | 189 | 1 | 9 |
| **2** | 133 | 225 | 60 |
| **3** | 26 | 74 | 112 |
| **4** | 10 | 104 | 103 |

Confusion matrix table interprets most of the data points distributed in three specific clusters and All the clusters reflects moderate distribution of data points between four classes.

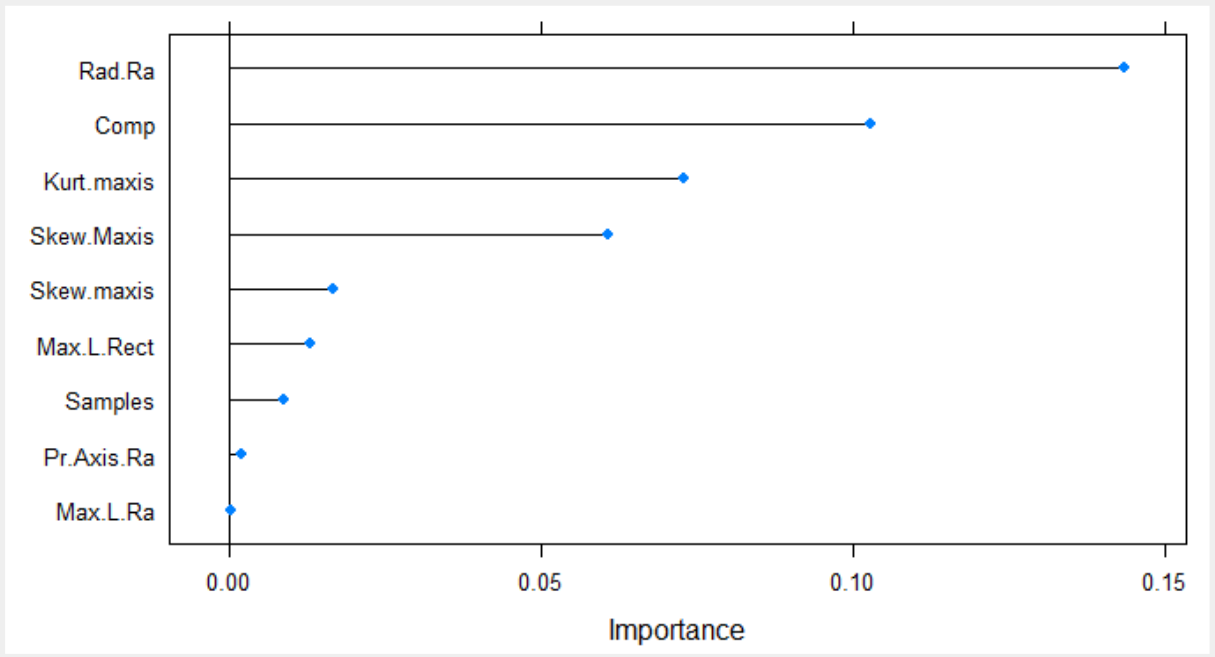
```
# Load the flexclust package
installed.packages("flexclust")
library(flexclust)

# Calculate the Adjusted Rand Index(ARI)
randIndex(confuseTable.km)
```

**Calculated ARI value is 0.2296454**
**This calculated ARI value interprets that the agreement between the cluster solutions has poor performance.**

**Preprocessing Method 02 - Drop unnecessary attribute values in the dataset.**

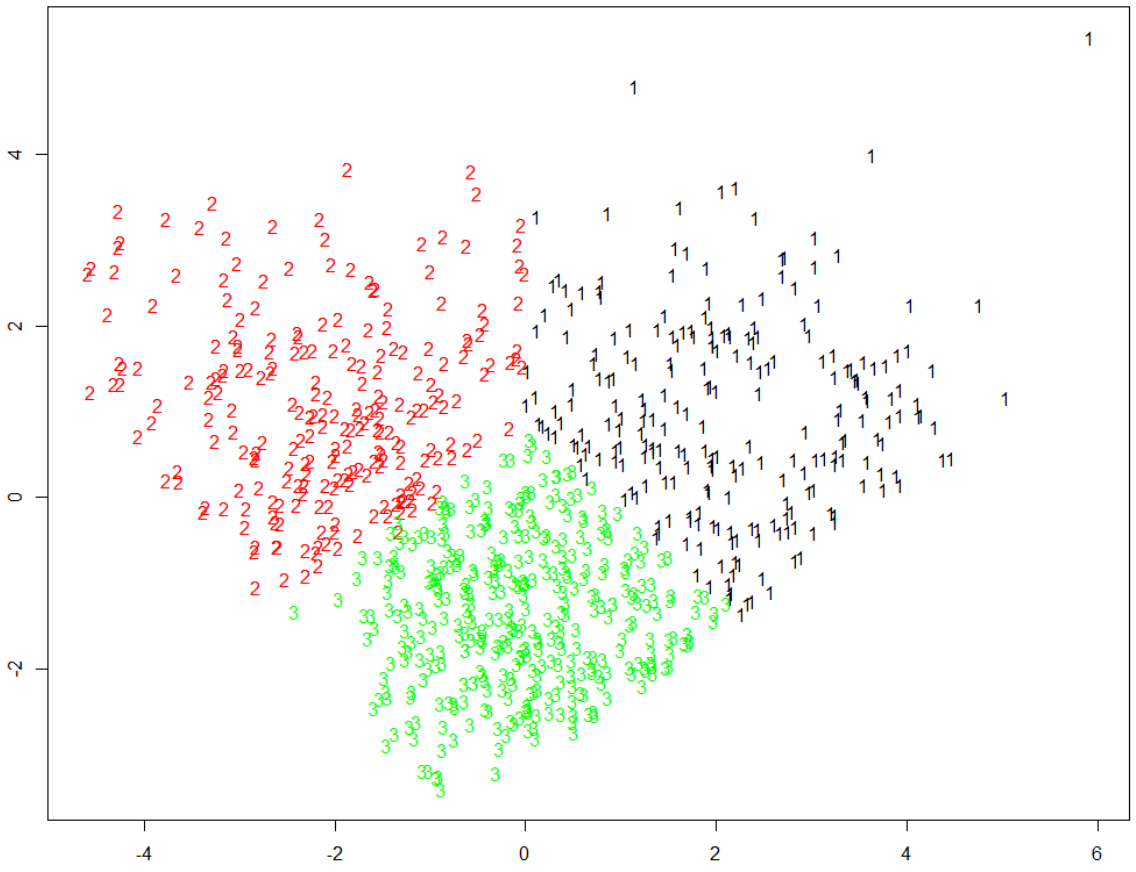
Demonstrating a code to reduce and drop the less related data as follows

```
classes = vehicle$class
vehicle$Class = NULL
data(vehicle)

control <- trainControl(method = "repeatedcv",number = 10,repeats = 3)

model <- train(vehicle, classes, method="lars2", preProcess="scale",
               trControl=control)
# estimate variable importance
importance <- varImp(model, scale=FALSE)

# summarize importance
print(importance)

# plot importance
plot(importance)

# remove less import colmns
namelist = list("Max.L.Ra","Skew.maxis","Max.L.Rect","Samples","Pr.Axis.Ra")

# Remove the namelist
drop <- c(namelist)
print(drop)
vehicle = vehicle[,!(names(vehicle) %in% drop)]
print(vehicle)
```



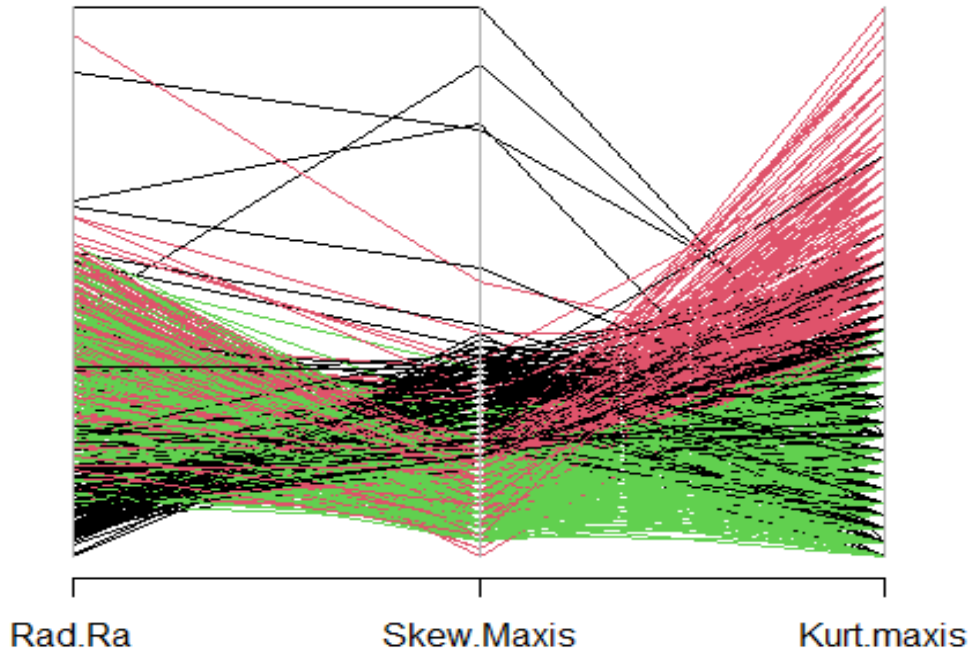
With execution above code preprocessing of data, I have generated the above chart based on the importance of each attribute. As we see, I have dropped the following attributes from the dataset. **["Max.L.Ra", "Pr.Axis.Ra", "Samples", "Skew.maxis", "Max.L.Rect"].** Therefore now the dataset remaining 5 attributes only. Moreover, the resulting clusters have **sizes of 241, 236, and 369** respectively. Three clusters have been plotted below.

```
#Load fpc package
installed.packages("fpc")
library(fpc)
plotcluster(data.train, fit.km$cluster)
```



Creating a parallel coordinates plot to analyse the patterns and relationships between variables that may have contributed to the formation of the clusters.

```
#Load the library
library(MASS)
# Plot parallel coordinates with colored clusters using k-means results
parcoord(data.train, fit.km$cluster)
```




Evaluating the performance of the k-means clustering model by analysing a confusion matrix table. Code is demonstrated as follows

```
#create a confusion matrix
confuseTable.km <- table(class, fit.km$cluster)
confuseTable.km
```

| Classes | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 90 | 18 | 91 |
| 2 | 96 | 35 | 87 |
| 3 | 24 | 92 | 96 |
| 4 | 31 | 91 | 95 |

Confusion matrix table interprets most of the data points distributed in specific one cluster and there are few data points distributed in other cluster

```
# Load the flexclust package
installed.packages("flexclust")
library(flexclust)

# Calculate the Adjusted Rand Index(ARI)
randIndex(confuseTable.km)
```

**Calculated ARI value is 0.05449811**

**This calculated ARI value interprets that the agreement between the cluster solutions has poor performance.**

**Comparison of Final Results of Each Testing of calculation methods and data preprocessing.**

| Test | Testing Method | Tested No of Clusters | Size of each Clusters | ARI Value |
|---|---|---|---|---|
| **Test 01** | Without data preprocessing | 02 | 552 | 0.08358285 |
| | | | 294 | |
| **Test 02** | Without data preprocessing | 03 | 275 | 0.07904893 |
| | | | 300 | |
| | | | 271 | |
| **Test 03** | Removing attributes which have the highest correlation | 03 | 358 | 0.2296454 |
| | | | 204 | |
| | | | 284 | |
| **Test04** | Removing attributes which have the highest correlation and unnecessary attributes | 03 | 241 | 0.05449811 |
| | | | 236 | |
| | | | 369 | |

**Conclusion**

Testing showed that removing attributes with the highest correlation and unnecessary attributes had a negative impact on clustering performance. Test 03 showed the best performance with an ARI value of 0.2296, compared to Test 04 which had the lowest ARI value. Increasing the number of clusters from 2 to 3 did not always lead to better performance.
The overall performance of the dataset is poor because after carrying out several calculations and tests on the dataset ARI value is still 0.2296454 and it is very low.
According to my analysis, the best number of clusters is no. 02 but the same time, number of clusters 03 also can suggest as an option.