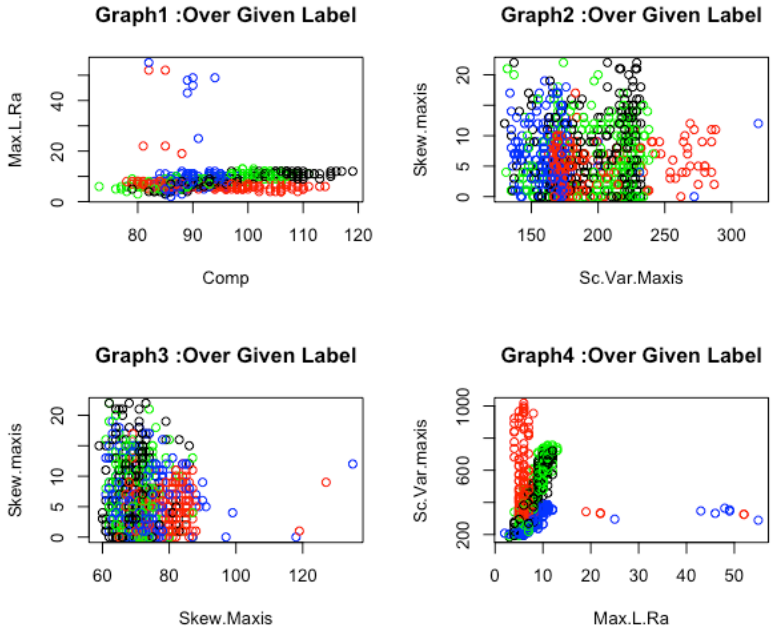
1st Objective One (Partitioning Clustering)

Analyzing data with given classification

Since the data set is given with a classification field, at first, we can analyze the data with some selected columns in order to get a basic idea about the data and the given classification.



Four types of vehicles are shown in different colors. When observing these graphs, specially Graph1 and Graph4, we can see that data are squeezed towards a particular axis. Also, there are significant differences in the scale of some of the features. For example, scale difference between, “Sc.Var.Maxis” and “Max.L.Ra” is huge which impacts to any data algorithm badly. Also, there are clear set of outliers too.

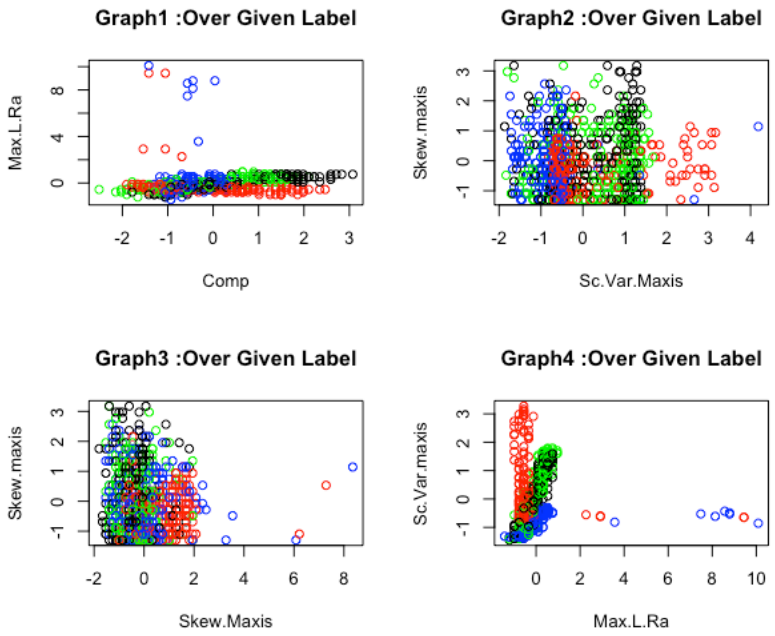
These are clear indication to normalize or scale data before applying any kind of algorithm to the given data set.

Even though, the provided data set is labeled one, after analyzing above graphs, there is no sign of identifying clear data clusters by comparing only two features at a time. Also, it is hard to say that these data set can be labeled into 4 classes, because there is huge overlapping of data points.

program-01 – source code for above graph

```r
vehicles <- read_excel("Documents/IIT-DMML/R/vehicles.xlsx")

#Replace, bus = 'red', van = 'blue', opel = 'green', saab = 'black'
vehicles$Class[vehicles$Class == 'bus']  <- 'red'
vehicles$Class[vehicles$Class == 'van']  <- 'blue'
vehicles$Class[vehicles$Class == 'opel'] <- 'green'
vehicles$Class[vehicles$Class == 'saab'] <- 'black'

#Plot the original data set
#Plot the given data as it is with class

par(mfrow=(c(2, 2)))
plot(vehicles[c("Comp", "Max.L.Ra")], col=vehicles$Class, main="Graph1 :Over Given Label")
plot(vehicles[c("Sc.Var.Maxis", "Skew.maxis")], col=vehicles$Class, main="Graph2 :Over Given Label")
plot(vehicles[c("Skew.Maxis", "Skew.maxis")], col=vehicles$Class, main="Graph3 :Over Given Label")
plot(vehicles[c("Max.L.Ra", "Sc.Var.maxis")], col=vehicles$Class, main="Graph4 :Over Given Label")
```

Let's draw the same four graphs after applying normalization to the given data set.



With the above graph, we can see that the data set is properly normalized into an acceptable scale. So, therefore, before applying any algorithm to the given data set, it is important to scale or normalized the data set. But there are still outliers to be quantile. Let's analyze it later.

program-02 – source code for above graph

```r
vehicles <- read_excel("Documents/IIT-DMML/R/vehicles.xlsx")
#print(vehicles)

#Replace, bus = 'red', van = 'blue', opel = 'green', saab = 'black'
vehicles$Class[vehicles$Class == 'bus']  <- 'red'
vehicles$Class[vehicles$Class == 'van']  <- 'blue'
vehicles$Class[vehicles$Class == 'opel'] <- 'green'
vehicles$Class[vehicles$Class == 'saab'] <- 'black'

# Saclig data set
#Remove classification field from original data set before scaling data.
vehicles_features <- vehicles
vehicles_features$Class <- NULL
vehicles_features_normalized <- as.data.frame(scale(vehicles_features))
#Append the 'Class' field into the scaled data set again
vehicles_features_normalized["Class"] <- vehicles$Class

par(mfrow=(c(2, 2)))
plot(vehicles_features_normalized[c("Comp", "Max.L.Ra")], col=vehicles_features_normalized$Class, main="Graph1 :Over Given Label")
plot(vehicles_features_normalized[c("Sc.Var.Maxis", "Skew.maxis")], col=vehicles_features_normalized$Class, main="Graph2 :Over Given Label")
plot(vehicles_features_normalized[c("Skew.Maxis", "Skew.maxis")], col=vehicles_features_normalized$Class, main="Graph3 :Over Given Label")
plot(vehicles_features_normalized[c("Max.L.Ra", "Sc.Var.maxis")], col=vehicles_features_normalized$Class, main="Graph4 :Over Given Label")
```

Outlier Detection

First, let's run the summary function on the given data set before doing any processing on the data set.

The following is the five-number summary output.

```
> vehicles <- read_excel("Documents/IIT-DMML/R/vehicles.xlsx")
> summary(vehicles)
    Samples          Comp            Circ          D.Circ          Rad.Ra         Pr.Axis.Ra      Max.L.Ra         Scat.Ra         Elong        Pr.Axis.Rect
 Min.   :  1.0   Min.   : 73.00   Min.   :33.00   Min.   : 40.00   Min.   :104.0   Min.   : 47.00   Min.   : 2.000   Min.   :112.0   Min.   :26.00   Min.   :17.00
 1st Qu.:212.2   1st Qu.: 87.00   1st Qu.:40.00   1st Qu.: 70.00   1st Qu.:141.0   1st Qu.: 57.00   1st Qu.: 7.000   1st Qu.:146.2   1st Qu.:33.00   1st Qu.:19.00
 Median :423.5   Median : 93.00   Median :44.00   Median : 80.00   Median :167.0   Median : 61.00   Median : 8.000   Median :157.0   Median :43.00   Median :20.00
 Mean   :423.5   Mean   : 93.68   Mean   :44.86   Mean   : 82.09   Mean   :168.9   Mean   : 61.69   Mean   : 8.567   Mean   :168.8   Mean   :40.93   Mean   :20.58
 3rd Qu.:634.8   3rd Qu.:100.00   3rd Qu.:49.00   3rd Qu.: 98.00   3rd Qu.:195.0   3rd Qu.: 65.00   3rd Qu.:10.000   3rd Qu.:198.0   3rd Qu.:46.00   3rd Qu.:23.00
 Max.   :846.0   Max.   :119.00   Max.   :59.00   Max.   :112.00   Max.   :333.0   Max.   :138.00   Max.   :55.000   Max.   :265.0   Max.   :61.00   Max.   :29.00
   Max.L.Rect     Sc.Var.Maxis     Sc.Var.maxis      Ra.Gyr        Skew.Maxis       Skew.maxis       Kurt.maxis      Kurt.Maxis      Holl.Ra          Class
 Min.   :118   Min.   :130.0   Min.   : 184.0   Min.   :109.0   Min.   : 59.00   Min.   : 0.000   Min.   : 0.0   Min.   :176.0   Min.   :181.0   Length:846
 1st Qu.:137   1st Qu.:167.0   1st Qu.: 318.2   1st Qu.:149.0   1st Qu.: 67.00   1st Qu.: 2.000   1st Qu.: 5.0   1st Qu.:184.0   1st Qu.:190.2   Class :character
 Median :146   Median :178.5   Median : 364.0   Median :173.0   Median : 71.50   Median : 6.000   Median :11.0   Median :188.0   Median :197.0   Mode  :character
 Mean   :148   Mean   :188.6   Mean   : 439.9   Mean   :174.7   Mean   : 72.46   Mean   : 6.377   Mean   :12.6   Mean   :188.9   Mean   :195.6
 3rd Qu.:159   3rd Qu.:217.0   3rd Qu.: 587.0   3rd Qu.:198.0   3rd Qu.: 75.00   3rd Qu.: 9.000   3rd Qu.:19.0   3rd Qu.:193.0   3rd Qu.:201.0
 Max.   :188   Max.   :320.0   Max.   :1018.0   Max.   :268.0   Max.   :135.00   Max.   :22.000   Max.   :41.0   Max.   :206.0   Max.   :211.0
>
```

<u>Circ Feature Analysis</u>

Q1 = 40.00
Q3 = 49.00
IQR (Inter Quantile Range) = Q3 – Q1 = 49.0 – 40.00 = 9
Q1 - 1.5(IQR) = 40.00 – 1.5 x 9 = 26.5
Q3 + 1.5(IQR) = 49.00 + 1.5x9 = 62.5

The minimum value of 'Circ' feature is 33.00 which is bigger than then 26.5. And also, the maximum value of 'Circ' feature is 59.00 which is below the 62.5. So, there are no outliers for 'Circ' feature values for entire data set.

<u>Sc.Var.maxis Feature Analysis</u>

Q1 = 318.2
Q3 = 587.0
IQR (Inter Quantile Range) = Q3 – Q1 = 587.0 – 318.2 = 268.8
Q1 - 1.5(IQR) = 587.00 – (1.5 x 268.8) = 183.80

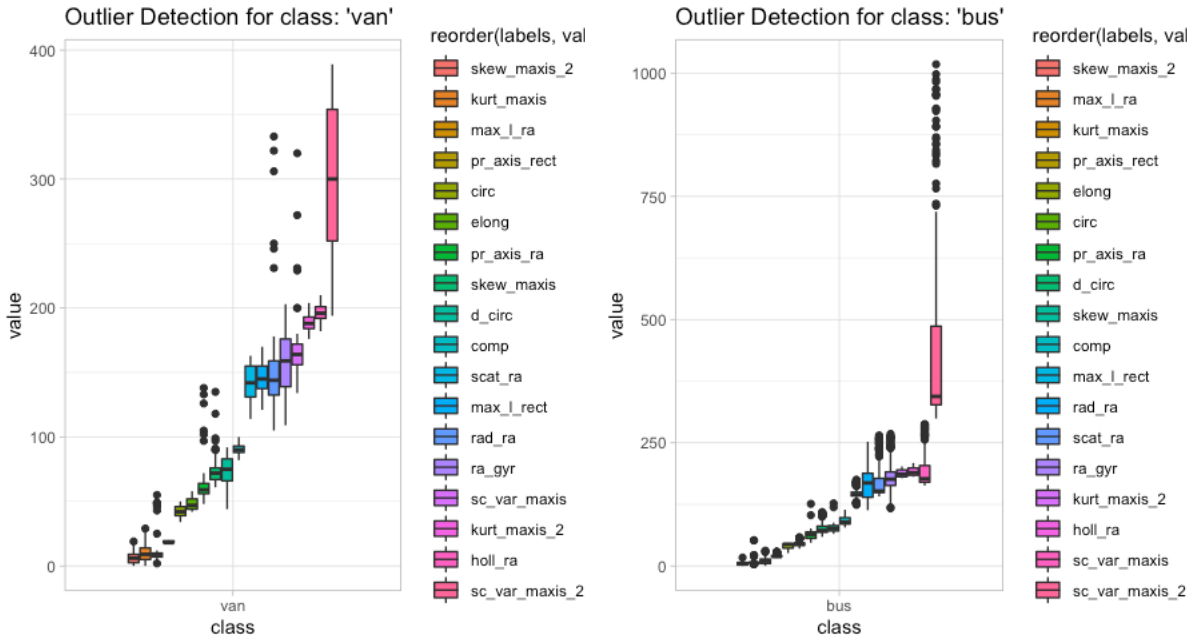
Q3 + 1.5(IQR) = 587.00 + (1.5 x 268.8) = 990.20

The maximum value of ‘Sc.Var.maxis’ is 1018.00 which is bigger than 990.20 which is an outlier.

As you can see in highlighted, the values for ‘Circ’ field is almost equally distributed. There are not outliers. The ‘Min’ and ‘Max’ values are reasonably close to the ‘Median’ and ‘Mean’ values. This is a good quality of a feature of a data set when applying an algorithm.

But ‘Sc. Var.maxis’ feature has significant outliers. It’s ‘Max’ values are significantly different. When an algorithm compares two data points of ‘Sc.Var.maxis’ features, it will result a huge difference which negatively impact to the final result.

We will further check for outliers by generating boxplots for each of the feature under given classification.



Outliers

If you see the above four graphs for each given classification, we can see outliers for many of the given features. Also, there are some features which has no a single outlier. So therefore, it is important to quantile the dataset before applying any algorithm to the given data set in order to get more accurate result.

program-04 – source code for the above graph

```r
library(tidyverse)
library(readxl)
library(NbClust)
library(knitr)
library(tidymodels)
library(flexclust)
library(funtimes)
library(gridExtra)

# Read in the original excel datafile
vehicles <- read_excel("Documents/IIT-DMML/R/vehicles.xlsx")
summary(vehicles)
vehicles_processed <- janitor::clean_names(vehicles)
vehicles_processed <- mutate(vehicles_processed, class = as_factor(class))

#Determine outliers for 'van'
vehicles_processed_van  <- pivot_longer(vehicles_processed,2:19,names_to = "labels")
vehicles_processed_van  <- filter(vehicles_processed_van, class == "van")
vehicles_processed_van  <- mutate(vehicles_processed_van, class = fct_reorder(class,value,median))
ggplot(vehicles_processed_van, aes(class, value, fill = reorder(labels,value))) + geom_boxplot() + labs(title = "Outlier Detection for class: 'van'")

#Determine outliers for 'bus'
vehicles_processed_bus  <- pivot_longer(vehicles_processed,2:19,names_to = "labels")
vehicles_processed_bus  <- filter(vehicles_processed_bus, class == "bus")
vehicles_processed_bus  <- mutate(vehicles_processed_bus, class = fct_reorder(class,value,median))
ggplot(vehicles_processed_bus, aes(class, value, fill = reorder(labels,value))) + geom_boxplot() + labs(title = "Outlier Detection for class: 'bus'")

#Determine outliers for 'saab'
vehicles_processed_saab  <- pivot_longer(vehicles_processed,2:19,names_to = "labels")
vehicles_processed_saab  <- filter(vehicles_processed_saab, class == "saab")
vehicles_processed_saab  <- mutate(vehicles_processed_saab, class = fct_reorder(class,value,median))
ggplot(vehicles_processed_saab, aes(class, value, fill = reorder(labels,value))) + geom_boxplot() + labs(title = "Outlier Detection for class: 'saab'")

#Determine outliers for 'opel'
vehicles_processed_opel  <- pivot_longer(vehicles_processed,2:19,names_to = "labels")
vehicles_processed_opel  <- filter(vehicles_processed_opel, class == "opel")
vehicles_processed_opel  <- mutate(vehicles_processed_opel, class = fct_reorder(class,value,median))
ggplot(vehicles_processed_opel, aes(class, value, fill = reorder(labels,value))) + geom_boxplot() + labs(title = "Outlier Detection for class: 'opel'")
```

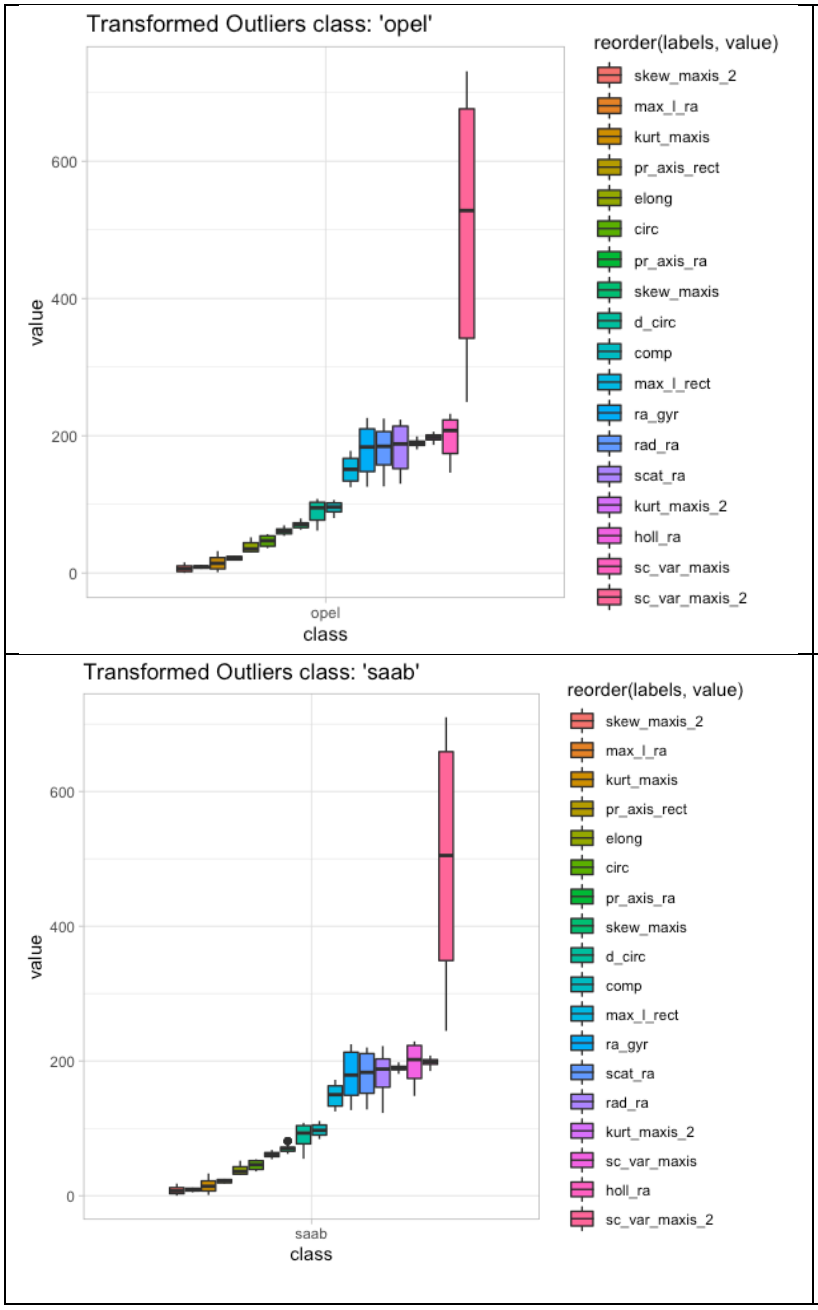
Removing Outliers

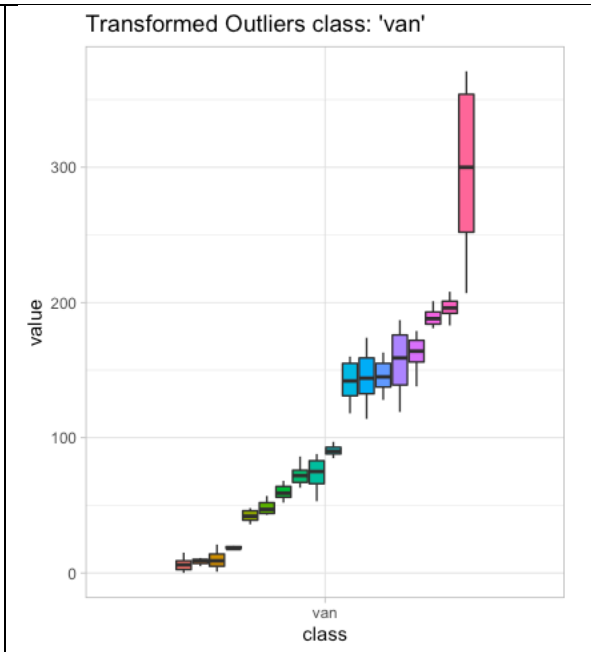
Let's look at the five-number summary out for the whole data set after removing outliers.

```
> summary(vehicle_quantiled)
    samples          comp            circ           d_circ          rad_ra       pr_axis_ra       max_l_ra         scat_ra          elong      
 Min.   :  1.0   Min.   : 80.00   Min.   :35.90   Min.   : 53.0   Min.   :113.9   Min.   :51.00   Min.   : 5.000   Min.   :118.0   Min.   :27.00  
 1st Qu.:212.2   1st Qu.: 87.00   1st Qu.:40.00   1st Qu.: 70.0   1st Qu.:141.0   1st Qu.:57.00   1st Qu.: 7.000   1st Qu.:146.2   1st Qu.:33.00  
 Median :423.5   Median : 93.00   Median :44.00   Median : 80.0   Median :167.0   Median :61.00   Median : 8.000   Median :157.0   Median :43.00  
 Mean   :423.5   Mean   : 93.65   Mean   :44.87   Mean   : 82.2   Mean   :168.0   Mean   :61.26   Mean   : 8.176   Mean   :168.9   Mean   :40.87  
 3rd Qu.:634.8   3rd Qu.:100.00   3rd Qu.:49.00   3rd Qu.: 98.0   3rd Qu.:194.0   3rd Qu.:65.00   3rd Qu.:10.000   3rd Qu.:198.0   3rd Qu.:46.00  
 Max.   :846.0   Max.   :111.00   Max.   :56.45   Max.   :108.2   Max.   :225.0   Max.   :74.00   Max.   :12.000   Max.   :250.2   Max.   :57.00  
  pr_axis_rect     max_l_rect     sc_var_maxis   sc_var_maxis_2      ra_gyr        skew_maxis     skew_maxis_2      kurt_maxis     kurt_maxis_2   
 Min.   :17.00   Min.   :125   Min.   :138.0   Min.   :206.9   Min.   :119.0   Min.   :62.0   Min.   : 0.000   Min.   : 0.90   Min.   :179.0  
 1st Qu.:19.00   1st Qu.:137   1st Qu.:167.0   1st Qu.:318.2   1st Qu.:149.0   1st Qu.:67.0   1st Qu.: 2.000   1st Qu.: 5.00   1st Qu.:184.0  
 Median :20.00   Median :146   Median :178.5   Median :364.0   Median :173.0   Median :71.5   Median : 6.000   Median :11.00   Median :188.0  
 Mean   :20.59   Mean   :148   Mean   :188.2   Mean   :439.9   Mean   :174.4   Mean   :72.2   Mean   : 6.256   Mean   :12.45   Mean   :188.9  
 3rd Qu.:23.00   3rd Qu.:159   3rd Qu.:216.0   3rd Qu.:587.0   3rd Qu.:198.0   3rd Qu.:75.0   3rd Qu.: 9.000   3rd Qu.:19.00   3rd Qu.:193.0  
 Max.   :27.00   Max.   :178   Max.   :272.4   Max.   :923.8   Max.   :247.4   Max.   :87.0   Max.   :18.000   Max.   :33.00   Max.   :201.0  
    holl_ra       class     
 Min.   :182.0   van :199  
 1st Qu.:190.2   saab:217  
 Median :197.0   bus :218  
 Mean   :195.6   opel:212  
 3rd Qu.:201.0             
 Max.   :208.0             
> 
```

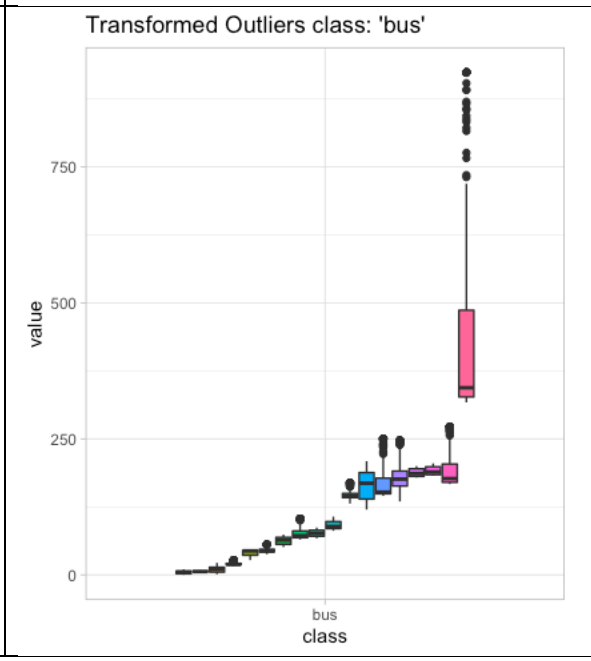
Now you can see that the maximum value of 'sc_var_maxis_2' ('Sc. Var.maxis') is below the upper outlier limit. So, this indicates, outliers of the data set is properly adjusted.

Further, the following graphs were generated again after removing outliers.


Transformed Outliers class: 'opel'
reorder(labels, value)
skew_maxis_2
max_l_ra
kurt_maxis
pr_axis_rect
elong
circ
pr_axis_ra
skew_maxis
d_circ
comp
max_l_rect
ra_gyr
rad_ra
scat_ra
kurt_maxis_2
holl_ra
sc_var_maxis
sc_var_maxis_2
value
0
200
400
600
opel
class


Transformed Outliers class: 'van'
value
0
100
200
300
van
class

Transformed Outliers class: 'saab'
reorder(labels, value)
skew_maxis_2
max_l_ra
kurt_maxis
pr_axis_rect
elong
circ
pr_axis_ra
skew_maxis
d_circ
comp
max_l_rect
ra_gyr
scat_ra
rad_ra
kurt_maxis_2
sc_var_maxis
holl_ra
sc_var_maxis_2
value
0
200
400
600
saab
class


Transformed Outliers class: 'bus'
value
0
250
500
750
bus
class

program-04 – source code to remove outliers.

```
library(tidyverse)
library(readxl)
library(NbClust)
library(knitr)
library(tidymodels)
library(flexclust)
library(funtimes)
library(gridExtra)
theme_set(theme_light())

# Read in the original excel datafile
vehicles <- read_excel("Documents/IIT-DMML/R/vehicles.xlsx")
vehicles_processed <- janitor::clean_names(vehicles)
vehicles_processed <- mutate(vehicles_processed, class = as_factor(class))
summary(vehicles_processed)

#Remove outliers of 'bus'
vehicles_bus_quantiled = vehicles_processed
vehicles_bus_quantiled <- filter(vehicles_bus_quantiled, class == "bus")
vehicles_bus_quantiled <- mutate(vehicles_bus_quantiled, across(2:19, ~squish(.x, quantile(.x, c(.05, .95)))))
print(vehicles_bus_quantiled)

#Remove outliers of 'van'
vehicles_van_quantiled = vehicles_processed
vehicles_van_quantiled <- filter(vehicles_van_quantiled, class == "van")
vehicles_van_quantiled <- mutate(vehicles_van_quantiled, across(2:19, ~squish(.x, quantile(.x, c(.05, .95)))))
print(vehicles_van_quantiled)

#Remove outliers of 'opel'
vehicles_opel_quantiled = vehicles_processed
vehicles_opel_quantiled <- filter(vehicles_opel_quantiled, class == "opel")
vehicles_opel_quantiled <- mutate(vehicles_opel_quantiled, across(2:19, ~squish(.x, quantile(.x, c(.05, .95)))))
print(vehicles_opel_quantiled)

#Remove outliers of 'saab'
vehicles_saab_quantiled = vehicles_processed
vehicles_saab_quantiled <- filter(vehicles_saab_quantiled, class == "saab")
vehicles_saab_quantiled <- mutate(vehicles_saab_quantiled, across(2:19, ~squish(.x, quantile(.x, c(.05, .95)))))
print(vehicles_saab_quantiled)

vehicle_quantiled = bind_rows(list(vehicles_bus_quantiled,vehicles_van_quantiled,vehicles_opel_quantiled,vehicles_saab_quantiled))
vehicle_quantiled <-  arrange(vehicle_quantiled, samples)
summary(vehicle_quantiled)

#Plot quantile data 'bus'
vehicle_quantiled_bus <- pivot_longer(vehicle_quantiled, 2:19,names_to = "labels")
vehicle_quantiled_bus <- filter(vehicle_quantiled_bus, class == "bus")
vehicle_quantiled_bus <- mutate(vehicle_quantiled_bus, class = fct_reorder(class,value,median))
ggplot(vehicle_quantiled_bus, aes(class, value, fill = reorder(labels,value))) + geom_boxplot() + labs(title = "Transformed Outliers class: 'bus'")

#Plot quantile data 'saab'
vehicle_quantiled_saab <- pivot_longer(vehicle_quantiled, 2:19,names_to = "labels")
vehicle_quantiled_saab <- filter(vehicle_quantiled_saab, class == "saab")
```

5:40 (Top Level) R Script

Normalizing dataset

In our program above, 'vehicle_quantiled' field contains the data set after removing outliers. Since, we already identified that our data set is needed to normalize, let's normalized the data set before degerming the number of clusters required. We can R's scale function for this.

```

#Normalized the data set.
vehicle_quantiled_scaled <- vehicle_quantiled
vehicle_quantiled_scaled$samples <- NULL
vehicle_quantiled_scaled$class <- NULL
vehicle_quantiled_scaled <- mutate(vehicle_quantiled_scaled, across(everything(), scale))
print(vehicle_quantiled_scaled)


```

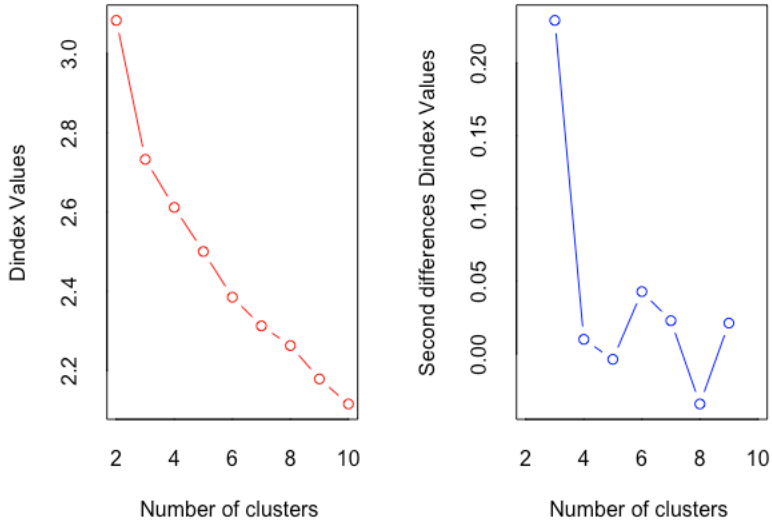
Determining Number of Clusters

We use **NbClust** package in order to determine the most appropriate number of clusters for the given data set. We will use two distance measures with **NbClust** in order to find out the number of clusters for K-Mean algorithm. The results of both analysis are given bellow.

```
#Determine the number of cluster
set.seed(123)
# Perform the kmeans using the NbClust function
# Use Euclidean for distance
cluster_euclidean = NbClust(vehicle_quantiled_scaled,distance="euclidean", min.nc=2,max.nc=10,method="kmeans",index="all")

# Use manhattan for distance
cluster_manhattan = NbClust(vehicle_quantiled_scaled,distance="manhattan", min.nc=2,max.nc=15,method="kmeans",index="all")
```

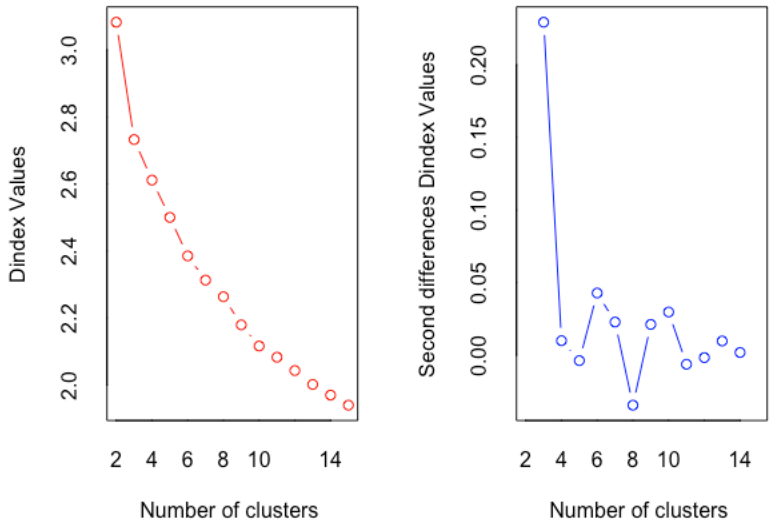
Spellcheck

Euclidean distance measure




```
> # Use Euclidean for distance
> cluster_euclidean = NbClust(vehicle_quantiled_scaled,distance="euclidean", min.nc=2,max.nc=10,method="kmeans",index="all")
*** : 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:
* 11 proposed 2 as the best number of clusters
* 9 proposed 3 as the best number of clusters
* 1 proposed 4 as the best number of clusters
* 1 proposed 7 as the best number of clusters
* 1 proposed 9 as the best number of clusters
* 1 proposed 10 as the best number of clusters

                   ***** Conclusion *****

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


*******************************************************************
```

Manhattan distance measure




```
*****************************************************************
> # Use manhattan for distance
> cluster_manhattan = NbClust(vehicle_quantiled_scaled,distance="manhattan", min.nc=2,max.nc=15,method="kmeans",index="all")
*** : 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:
* 9 proposed 2 as the best number of clusters
* 9 proposed 3 as the best number of clusters
* 1 proposed 4 as the best number of clusters
* 1 proposed 7 as the best number of clusters
* 1 proposed 9 as the best number of clusters
* 1 proposed 14 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  2


*****************************************************************
```

Both the analysis suggest that the most appropriate number of clusters in order to perform the K-Mean algorithm for the given data set is 2.

Perform K-Mean

Even though the data set is given with 4 labels, according to the analysis, we should be able to identify two clear clusters, not 4 clusters. Let's apply the K-Means for both scenario and analyze the output.

K-means with 4 number of clusters

```
#Apply K-Means
km = kmeans(vehicle_quantiled_scaled, centers = 4, nstart = 25)
km
km$cluster
km$centers
#Plot the given data as it is with class
plot(vehicle_quantiled_scaled[c("sc_var_maxis", "sc_var_maxis_2")], col=km$cluster)
points(km$centers[,c("sc_var_maxis", "sc_var_maxis_2")], col=1:4, pch=23, cex=4)

#compare cluster results with original data
table(vehicles_with_numeric_label$Class,km$cluster)
```

K-means with 2 number of clusters

The k-means summary

```
> #Apply K-Means
> km = kmeans(vehicle_quantiled_scaled, centers = 2, nstart = 25)
> km
K-means clustering with 2 clusters of sizes 556, 290

Cluster means:
       comp       circ    d_circ     rad_ra pr_axis_ra   max_l_ra    scat_ra      elong pr_axis_rect max_l_rect sc_var_maxis sc_var_maxis_2     ra_gyr  skew_maxis
1 -0.5708055 -0.5804845 -0.607386 -0.5422655 -0.1231833 -0.3382688 -0.6413009  0.6151195   -0.6435601 -0.5363494   -0.6179139     -0.6438746 -0.5335758  0.06183379
2  1.0943719  1.1129289  1.164505  1.0396538  0.2361721  0.6485430  1.2295285 -1.1793326    1.2338601  1.0283113    1.1846900      1.2344631  1.0229936 -0.11855029
  skew_maxis_2 kurt_maxis kurt_maxis_2    holl_ra
1  -0.07954352 -0.1195446  -0.04250306 -0.1192648
2   0.15250413  0.2291959   0.08148862  0.2286593

Clustering vector:
  [1] 1 1 2 1 1 2 1 1 1 2 1 1 1 1 2 2 1 1 2 2 1 1 1 1 2 1 1 2 2 1 1 1 1 2 1 1 1 1 2 1 2 1 1 1 2 1 1 1 2 1 1 2 1 1 2 1
 [84] 1 1 2 1 1 1 1 2 1 2 1 1 2 1 1 2 1 1 1 1 1 2 2 2 1 1 2 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 1 2 1 1 1 1 1 1 2 1 1 2 1 1 1 1 2 2 1 2 1 2 1 1 1 1 2 1 1 2 1 1 2 2
[167] 1 2 1 1 2 2 1 2 1 1 1 1 1 1 2 1 1 1 2 1 1 1 2 1 2 1 1 2 1 1 2 1 1 2 1 1 1 1 1 2 2 1 1 1 1 1 2 1 1 1 2 1 1 1 2 1 1 2 1 2 1 1 1 1 2 1 2 1 1 1 1 2 1 1 1 1 2 1 1 1 1 2 1 1 1 1 2 1
[250] 1 1 2 1 1 1 2 1 1 2 2 1 1 1 1 2 1 1 1 1 1 1 2 1 1 2 1 1 1 2 1 2 1 1 1 2 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 2 1 2 1 1 1 2 1 1 1 1 1 1 2 2 2 2 2 1 1 2 1 1 1 2 1 2 2 2 1 2 1 1 2 1 1 1 1 1 2
[333] 2 1 2 2 1 2 1 1 1 1 1 2 2 2 2 1 1 1 2 1 1 1 2 1 1 2 1 2 2 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 2 1 1 2 1 2 1 2 1 2 1 1 2 1 1 1 2 1 2 1 1 1 1 2 1 2 1 1 1 2 1 2 1 2 1 2 1 1 1 1 1 1 1 1 1 1
[416] 2 1 1 1 1 1 2 1 1 1 1 2 1 2 1 2 2 1 1 2 1 1 1 2 2 2 1 1 2 2 1 2 2 1 2 2 2 1 1 1 1 1 2 1 1 1 2 1 1 2 1 1 2 1 1 2 2 1 1 2 2 2 1 2 2 1 2 1 2 2 1 1 1 1 2 1 1 2 2 1 1 2 2 1 2 1
[499] 1 2 2 2 1 1 2 2 2 1 1 2 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 2 1 2 2 1 1 2 2 2 1 1 1 2 1 2 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 2 2 1 1 2 2 1 2 1 1 2 2 1 1 1 2 1 1 1 2 2 2 2 1 1 1
[582] 1 2 2 2 1 2 1 1 2 1 1 1 1 1 2 1 1 1 1 1 2 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 2 1 2 1 2 1 1 2 1 1 2 1 2 1 1 1 1 1 2 2 1 2 1 1 1 1 1 2 1 2 1 1 1 2 1 1 1 2 1 2 1 1
[665] 1 1 2 2 1 1 2 1 2 1 1 2 1 2 1 1 1 1 1 2 1 1 1 2 1 2 1 2 1 1 2 1 2 1 1 1 1 1 2 2 1 1 2 2 1 1 1 1 2 2 2 2 1 2 1 1 2 2 1 1 2 2 1 1 1 2 1 1 2 2 2 1 2 1 1 2 1 1 1 1 2 1 1 1
[748] 1 1 2 1 1 1 1 1 1 2 1 1 1 2 2 1 2 2 1 1 1 2 1 1 2 2 1 1 1 2 2 2 1 2 1 2 2 1 1 2 1 2 1 1 2 1 2 1 1 1 2 2 1 1 1 2 1 1 1 2 2 1 1 1 2 1 1 2 1 1 1 1 1 1 2 2 2 1 2 1 2 2 1 2 2 1 1
[831] 1 1 2 1 1 2 1 1 1 1 1 1 1 2 1 1

Within cluster sum of squares by cluster:
[1] 5852.651 2758.465
 (between_SS / total_SS =  43.4 %)
```

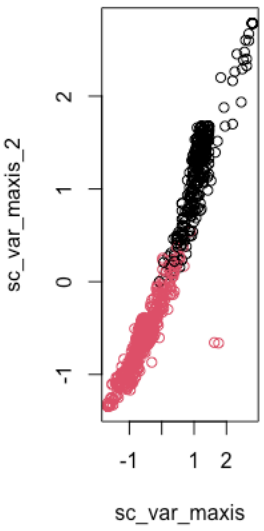
Let's compare the cluster results with original data set's classification field.

```
>
> #compare cluster results with original data
> table(vehicles_with_numeric_label$Class,km$cluster)

     1   2
  1 162  56
  2 199   0
  3  95 117
  4 100 117
```

I did relabel 'bus', 'van', 'opel' and 'saab' as 1,2 3,4 respectively in order to compare with the cluster results with original data set. Here, we can see that, K-means has successfully

identified two clusters even though original data set is classified into 4. One cluster with all the ‘van’s and the other cluster with other 3 types. All ‘van’s are properly identified as a single cluster and it has been unable to distinguish all other three types. There are so many overlapping when compare with original classification.

Let’s plot same two features after applying the K-means using scaled data set. As you can see bellow, two clear clusters can be identified.



2nd Objective (MLP)

Determining input vectors

Since the given data set is time series single feature value, in order to perform MLP-NN, we need to create different input vectors. We are going to create 10 new input vectors by using given date ordered GBP values.

First 8 vectors will be created using lag() function. This will create 8 input vectors with last 8 days GBP values. Among those 8 fields, the latest day values will be on left side in the data set.

Last 2 vectors are created by getting the mean values of nearest 5 rows and 10 rows respectively for a given GPB value.

So, the final input data set will be as follows.

```
> gbp_exchange_full <- drop_na(gbp_exchange_full)
> gbp_exchange_full
# A tibble: 487 x 12
   date_in_ymd gbp_eur  lag1  lag2  lag3  lag4  lag5  lag6  lag7  lag8 five_day_rolling ten_day_rolling
   <date>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>            <dbl>           <dbl>
 1 2010-01-14    0.887 0.890 0.898 0.900 0.898 0.900 0.899 0.900 0.895            0.888           0.885
 2 2010-01-15    0.885 0.887 0.890 0.898 0.900 0.898 0.900 0.899 0.900            0.883           0.883
 3 2010-01-18    0.881 0.885 0.887 0.890 0.898 0.900 0.898 0.900 0.899            0.878           0.880
 4 2010-01-19    0.872 0.881 0.885 0.887 0.890 0.898 0.900 0.898 0.900            0.875           0.877
 5 2010-01-20    0.865 0.872 0.881 0.885 0.887 0.890 0.898 0.900 0.898            0.873           0.875
 6 2010-01-21    0.870 0.865 0.872 0.881 0.885 0.887 0.890 0.898 0.900            0.871           0.873
 7 2010-01-22    0.878 0.870 0.865 0.872 0.881 0.885 0.887 0.890 0.898            0.871           0.871
 8 2010-01-25    0.871 0.878 0.870 0.865 0.872 0.881 0.885 0.887 0.890            0.872           0.870
 9 2010-01-26    0.871 0.871 0.878 0.870 0.865 0.872 0.881 0.885 0.887            0.871           0.870
10 2010-01-27    0.867 0.871 0.871 0.878 0.870 0.865 0.872 0.881 0.885            0.868           0.871
# … with 477 more rows
>
```

The following peace of code shows the imported libraries and how initial set of vectors were created.

```
knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
library(readxl)
library(lubridate)
library(zoo)
library(tidymodels)
library(readxl)
library(neuralnet)
library(knitr)

exchangeGBP <- read_excel("Documents/IIT-DMML/R/exchangeGBP.xlsx")
exchangeGBP <- janitor::clean_names(exchangeGBP)
exchangeGBP <- mutate(exchangeGBP, date_in_ymd = ymd(yyyy_mm_dd))
exchangeGBP <- select(exchangeGBP, -1)
exchangeGBP <- select(exchangeGBP, date_in_ymd, everything())


#all the input is in only one dataframe to be able to preserve the testing and training
#dataset for the two sets of input variables
#creating a new feature by using previous date values
gbp_exchange_full <- exchangeGBP
gbp_exchange_full <- mutate(gbp_exchange_full, lag1 = lag(exchangeGBP$gbp_eur,1))
gbp_exchange_full <- mutate(gbp_exchange_full, lag2 = lag(exchangeGBP$gbp_eur,2))
gbp_exchange_full <- mutate(gbp_exchange_full, lag3 = lag(exchangeGBP$gbp_eur,3))
gbp_exchange_full <- mutate(gbp_exchange_full, lag4 = lag(exchangeGBP$gbp_eur,4))
gbp_exchange_full <- mutate(gbp_exchange_full, lag5 = lag(exchangeGBP$gbp_eur,5))
gbp_exchange_full <- mutate(gbp_exchange_full, lag6 = lag(exchangeGBP$gbp_eur,6))
gbp_exchange_full <- mutate(gbp_exchange_full, lag7 = lag(exchangeGBP$gbp_eur,7))
gbp_exchange_full <- mutate(gbp_exchange_full, lag8 = lag(exchangeGBP$gbp_eur,8))
gbp_exchange_full <- mutate(gbp_exchange_full, five_day_rolling = rollmean(gbp_eur,5, fill = NA))
gbp_exchange_full <- mutate(gbp_exchange_full, ten_day_rolling = rollmean(gbp_eur,10, fill = NA))

gbp_exchange_full <- drop_na(gbp_exchange_full)
gbp_exchange_full
summary(gbp_exchange_full)

```

Outlier Detection

Let's see the summary of the data set before proceeding in order to see for any outliers.

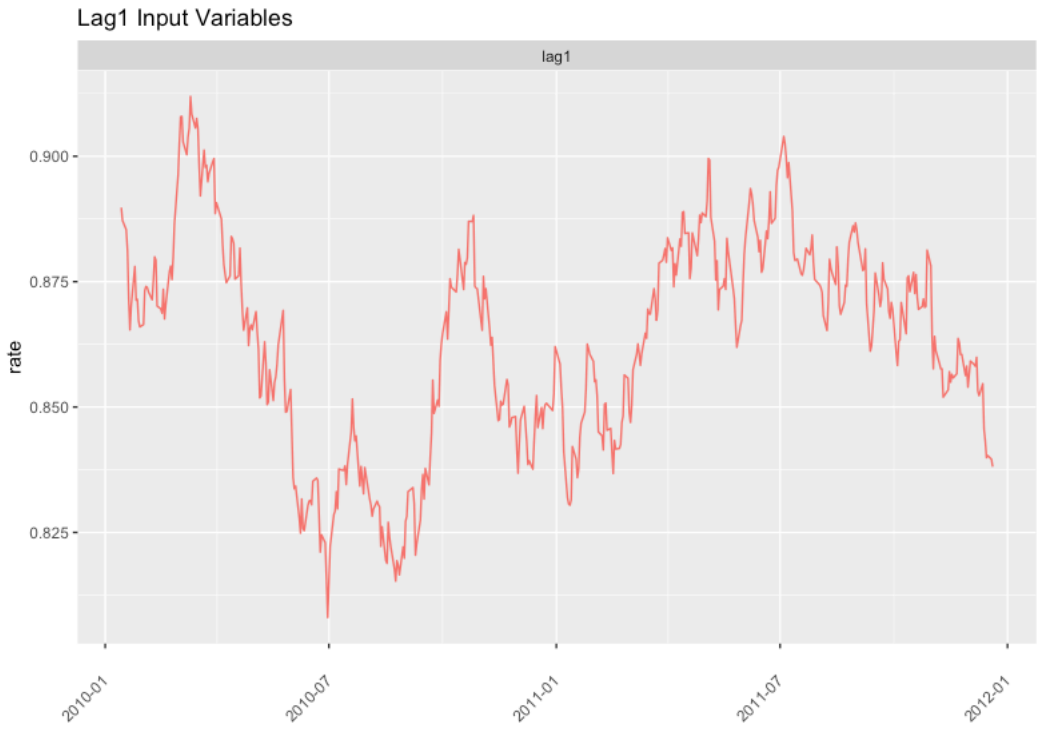
```
> summary(gbp_exchange_full)
  date_in_ymd            gbp_eur            lag1              lag2              lag3              lag4              lag5              lag6        
 Min.   :2010-01-14   Min.   :0.8080   Min.   :0.8080   Min.   :0.8080   Min.   :0.8080   Min.   :0.8080   Min.   :0.8080   Min.   :0.8080  
 1st Qu.:2010-07-07   1st Qu.:0.8471   1st Qu.:0.8474   1st Qu.:0.8475   1st Qu.:0.8477   1st Qu.:0.8480   1st Qu.:0.8481   1st Qu.:0.8482  
 Median :2011-01-04   Median :0.8656   Median :0.8657   Median :0.8660   Median :0.8662   Median :0.8664   Median :0.8664   Median :0.8664  
 Mean   :2010-12-31   Mean   :0.8629   Mean   :0.8630   Mean   :0.8631   Mean   :0.8632   Mean   :0.8634   Mean   :0.8635   Mean   :0.8636  
 3rd Qu.:2011-06-25   3rd Qu.:0.8784   3rd Qu.:0.8785   3rd Qu.:0.8786   3rd Qu.:0.8787   3rd Qu.:0.8787   3rd Qu.:0.8788   3rd Qu.:0.8789  
 Max.   :2011-12-20   Max.   :0.9119   Max.   :0.9119   Max.   :0.9119   Max.   :0.9119   Max.   :0.9119   Max.   :0.9119   Max.   :0.9119  
      lag7              lag8         five_day_rolling ten_day_rolling
 Min.   :0.8080   Min.   :0.8080   Min.   :0.8175   Min.   :0.8199  
 1st Qu.:0.8485   1st Qu.:0.8485   1st Qu.:0.8468   1st Qu.:0.8467  
 Median :0.8672   Median :0.8672   Median :0.8673   Median :0.8676  
 Mean   :0.8637   Mean   :0.8638   Mean   :0.8629   Mean   :0.8629  
 3rd Qu.:0.8790   3rd Qu.:0.8792   3rd Qu.:0.8781   3rd Qu.:0.8782  
 Max.   :0.9119   Max.   :0.9119   Max.   :0.9078   Max.   :0.9062  
```

Lag1 Outlier Analysis

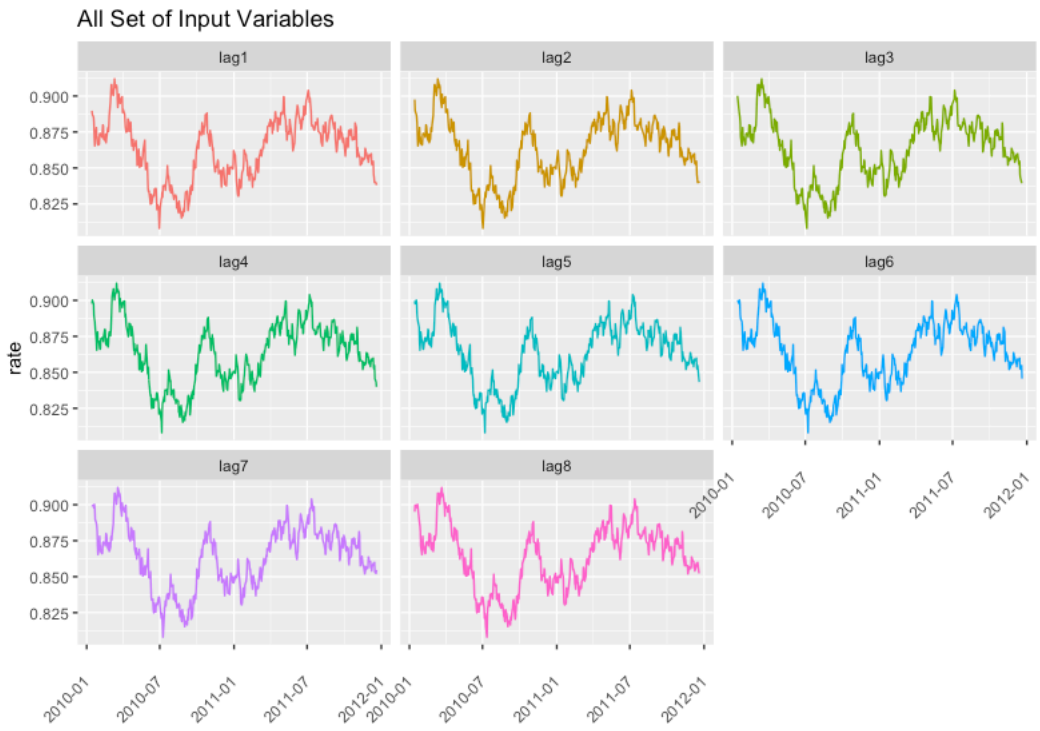
Q1 = 0.8474
Q3 = 0.8785
IQR (Inter Quantile Range) = Q3 – Q1 = 0.8785 – 0.8474 = 0.0311
Q1 - 1.5(IQR) = 0.8474– (1.5 x 0.0311) = 0.80075
Q3 + 1.5(IQR) = 0.8785+ (1.5 x 0.0311) = 0.92515

As per above analysis, both 'Min' and 'Max' values are within the range for 'lag1' summary analysis. Also, the 'Min' and 'Max' values are same for the first 8 vectors which are within 0 and 1 range which is acceptable for 'logistic' activation function for MLP-ANN.

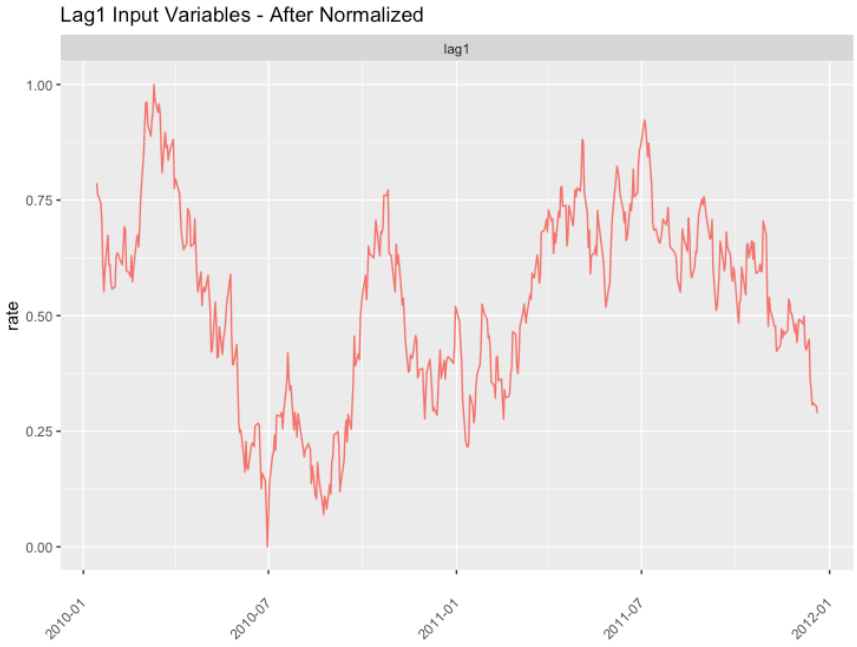
Let's further see a graphical view for 'lag1' data set. The following graph shows the variation of GBP for whole data set for a single input vector.



The same graph was generated for all the input vectors for whole data set as follows.



Let's see the same graph after applying the normalization for the data set. The following graph shows the result of 'lag1' after applying normalization.



As you can see above, there is no significant different of the graph even after normalizing the data set. Based on above all analysis, we can decide that the normalization the given data set is not necessary before applying MLP-ANN model for the given specific data set.

If we use ‘logistic’ activation function, having all the values in between 0 and 1 will result more accurate output. When analyzing the given data set, there is not high requirement to perform normalization because, almost all the values are in between 0 and 1. And also there are no significant outliers. But for better accuracy, we can normalize the data set before applying MLP-ANN, because it does not impact the final output negatively. If there is an impact, it always be a positive impact to the final output.

The pieces of code to generate graph and normalization is as follows.

```
#Plot few input feature vectors to see any outlizers
gbp_exchange_full %>%
  pivot_longer(cols = 3:3,names_to = "kind",values_to = "rate") %>%
  ggplot(aes(date_in_ymd,rate, color = kind)) + geom_line() + facet_wrap(~kind)
      + theme(axis.text.x = element_text(angle = 45, vjust = 0.5, hjust=1)) +
  labs(x = "", title = "Lag1 Input Variables") +
  theme(legend.position = "none")
```

```
#Normalize the data set before applying into the model
# All the variables are normalized
normalized_gbp <- gbp_exchange_full
normalized_gbp <- mutate(normalized_gbp, across(2:12, ~normalize(.x)))
```

```
# We can create a function to normalize the data from 0 to 1
# If we use leania activation function like 'logistic'
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
```

Testing with two layers and different combination of nodes and capture the results of all performs operations.

Get the lag values for last 10 days

Network Architecture

We are going to use two hidden layers with 50 different combination of nodes by using two activation function, ‘logistic’ and ‘sigmoid’. After getting summary details of all these, best network structure will be selected.

The following tables shows the best fit out of 4 different network structure results. Each execution will result 50 results. Out of those 50, the best results are highlighted bellow table.

Structure	Input Vector	Hidden Layers	Nodes Layer1	Nodes Layer2	Activation Function	RMSE	MAE	MAPE
1	Lag1	2	2	4	logistic	0.00449	0.00344	**0.397**
2	Lag1 + lag2	2	4	2	logistic	0.000119	0.0000948	**0.0109**
3	Lag1	2	7	4	tanh	0.00448	0.00343	0.396
4	Lag1 + lag2	2	8	4	tanh	0.000210	0.000156	0.0179

Structure 1 Result

```
> print(set_a_models_two_layers, n=50)
# A tibble: 50 x 6
   type              hiddel_layers input_set    rmse      mae  mape
   <chr>             <chr>         <chr>       <dbl>    <dbl> <dbl>
 1 Two Hidden Layers 2 and 4       lag1      0.00449 0.00344 0.397
 2 Two Hidden Layers 8 and 5       lag1      0.00453 0.00347 0.401
 3 Two Hidden Layers 10 and 4      lag1      0.00454 0.00347 0.401
 4 Two Hidden Layers 6 and 5       lag1      0.00454 0.00348 0.401
 5 Two Hidden Layers 4 and 1       lag1      0.00454 0.00348 0.401
 6 Two Hidden Layers 9 and 4       lag1      0.00454 0.00348 0.401
 7 Two Hidden Layers 5 and 1       lag1      0.00454 0.00347 0.401
 8 Two Hidden Layers 10 and 3      lag1      0.00454 0.00347 0.401
 9 Two Hidden Layers 5 and 2       lag1      0.00455 0.00348 0.402
10 Two Hidden Layers 8 and 3       lag1      0.00455 0.00347 0.401
11 Two Hidden Layers 7 and 1       lag1      0.00455 0.00348 0.402
12 Two Hidden Layers 3 and 4       lag1      0.00455 0.00348 0.402
13 Two Hidden Layers 9 and 5       lag1      0.00455 0.00348 0.402
14 Two Hidden Layers 9 and 3       lag1      0.00455 0.00348 0.402
15 Two Hidden Layers 5 and 4       lag1      0.00455 0.00348 0.401
16 Two Hidden Layers 4 and 3       lag1      0.00455 0.00348 0.402
```

Structure 2 Result

```
# A tibble: 50 x 6
   type              hiddel_layers input_set     rmse       mae   mape
   <chr>             <chr>         <chr>        <dbl>     <dbl>  <dbl>
 1 Two Hidden Layers 4 and 2       lag1      0.000119 0.0000948 0.0109
 2 Two Hidden Layers 9 and 4       lag1      0.000128 0.000107  0.0124
 3 Two Hidden Layers 10 and 1      lag1      0.000162 0.000127  0.0147
 4 Two Hidden Layers 4 and 3       lag1      0.000170 0.000143  0.0166
 5 Two Hidden Layers 1 and 5       lag1      0.000170 0.000139  0.0161
 6 Two Hidden Layers 8 and 5       lag1      0.000176 0.000149  0.0172
 7 Two Hidden Layers 3 and 1       lag1      0.000177 0.000145  0.0168
 8 Two Hidden Layers 3 and 5       lag1      0.000182 0.000129  0.0150
 9 Two Hidden Layers 10 and 2      lag1      0.000201 0.000159  0.0184
10 Two Hidden Layers 8 and 4       lag1      0.000206 0.000151  0.0174
11 Two Hidden Layers 7 and 5       lag1      0.000211 0.000144  0.0167
12 Two Hidden Layers 6 and 3       lag1      0.000212 0.000148  0.0172
13 Two Hidden Layers 4 and 4       lag1      0.000214 0.000174  0.0202
14 Two Hidden Layers 6 and 5       lag1      0.000220 0.000185  0.0214
15 Two Hidden Layers 8 and 2       lag1      0.000229 0.000190  0.0220
16 Two Hidden Layers 7 and 2       lag1      0.000230 0.000186  0.0215
17 Two Hidden Layers 10 and 3      lag1      0.000234 0.000196  0.0227
```

Structure 3 Result

```
# A tibble: 50 x 6
   type              hiddel_layers input_set    rmse      mae  mape
   <chr>             <chr>         <chr>       <dbl>    <dbl> <dbl>
 1 Two Hidden Layers 7 and 4       lag1      0.00448 0.00343 0.396
 2 Two Hidden Layers 8 and 4       lag1      0.00450 0.00345 0.398
 3 Two Hidden Layers 7 and 2       lag1      0.00450 0.00345 0.398
 4 Two Hidden Layers 8 and 3       lag1      0.00450 0.00345 0.398
 5 Two Hidden Layers 5 and 5       lag1      0.00450 0.00345 0.398
 6 Two Hidden Layers 10 and 4      lag1      0.00451 0.00345 0.398
 7 Two Hidden Layers 10 and 2      lag1      0.00451 0.00345 0.398
 8 Two Hidden Layers 1 and 3       lag1      0.00451 0.00346 0.399
 9 Two Hidden Layers 9 and 2       lag1      0.00451 0.00345 0.398
10 Two Hidden Layers 5 and 3       lag1      0.00451 0.00345 0.399
11 Two Hidden Layers 9 and 3       lag1      0.00451 0.00346 0.399
```

Structure 4 Result

```
> print(set_a_models_two_layers, n=50)
# A tibble: 50 x 6
   type              hiddel_layers input_set     rmse      mae   mape
   <chr>             <chr>         <chr>        <dbl>    <dbl>  <dbl>
 1 Two Hidden Layers 8 and 4       lag1      0.000210 0.000156 0.0179
 2 Two Hidden Layers 5 and 4       lag1      0.000214 0.000162 0.0187
 3 Two Hidden Layers 6 and 2       lag1      0.000250 0.000201 0.0232
 4 Two Hidden Layers 6 and 3       lag1      0.000265 0.000182 0.0209
 5 Two Hidden Layers 6 and 5       lag1      0.000270 0.000213 0.0246
 6 Two Hidden Layers 8 and 1       lag1      0.000286 0.000225 0.0260
 7 Two Hidden Layers 10 and 4      lag1      0.000287 0.000230 0.0265
 8 Two Hidden Layers 5 and 1       lag1      0.000306 0.000244 0.0282
 9 Two Hidden Layers 4 and 1       lag1      0.000323 0.000254 0.0293
10 Two Hidden Layers 10 and 5      lag1      0.000334 0.000258 0.0297
11 Two Hidden Layers 1 and 3       lag1      0.000350 0.000296 0.0342
12 Two Hidden Layers 1 and 5       lag1      0.000353 0.000298 0.0345
13 Two Hidden Layers 8 and 3       lag1      0.000353 0.000298 0.0345
14 Two Hidden Layers 2 and 4       lag1      0.000362 0.000269 0.0310
```

If we consider structure 1 and structure3, we can notice that error values are little bit higher than the structure 2 and structure4. So, having one than one input vector will produced much better results rather than having a single input vector.

As per the above analysis, structure 2 will give the best result, since it has the lowest error factor values.

Model with the lowest RMST will be the best.

Need to specify how many hidden layers are going to be used and how many nodes or neurons are going to be under one layer. In our case, we are going to use two layers with different combination of nodes.