INFORMATICS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF WESTMINSTER

Information Institute of Technology

Department of computer

(B.Eng.) in Software Engineering

Module: 5DATA002W Machine Learning & Data Mining

Module Leader: Dr. V.S. Kontogiannis

Coursework 1

Date of submission: 6th of May, 2021

Student ID: 20191161

Student UOW: 17904023/1

Student First Name: Fathima Rinoza

Student Surname: Mohamed Jiffry

1st Objective (Partitioning Clustering)

Methodologies used for reducing the input dimensionality

The number of input variables or features determines the dimensionality of a dataset. Dimensionality reduction refers to methods for reducing the number of input variables in a dataset. Dimensionality reduction techniques include the feature selection process, matrix factorization, manifold learning, auto encoder methods, and others.

PCA is a linear algebra technique that can be used to automatically reduce dimensionality. PCA is a popular method used in this coursework (Principal Components Analysis).

Pre-processing tasks

Data preprocessing is a data mining technique for converting raw data into a format that is both accessible and effective.

Outlier detection

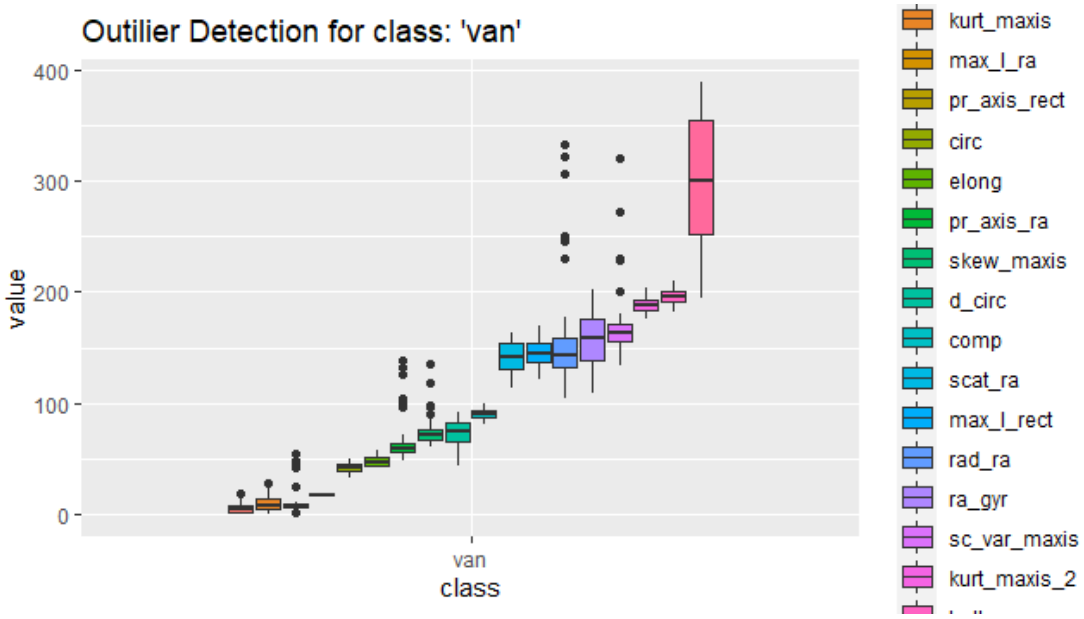
A value in a random collection of data that deviates abnormally from other values is called an outlier. To get both numerical and nominal data, we must first factorize the data.

Sc.Var.maxis	Ra.Gyr	Skew.Maxis	Skew.maxis	Kurt.maxis	Kurt.Maxis	Holl.Ra	Class
379	184	70	6	16	187	197	van
330	158	72	9	14	189	199	van
635	220	73	14	9	188	196	saab
309	127	63	6	10	199	207	van
325	188	127	9	11	180	183	bus
957	264	85	5	9	181	183	bus
361	172	66	13	1	200	204	bus

```
#reading the data
vehicles <- read_excel("E:/Lectures/DataScience/cw/vehicles.xlsx")

#viewing the data
view(vehicles)
summary(vehicles)
```

Outlier detection for van class

```
vehicles_original %>%
  pivot_longer(2:19,names_to = "labels") %>%
  filter(class == "van") %>%
  mutate(class = fct_reorder(class,value,median)) %>%
  ggplot(aes(class, value, fill = reorder(labels,value))) +
  geom_boxplot() +
  labs(title = "Outilier Detection for class: 'van'")
```

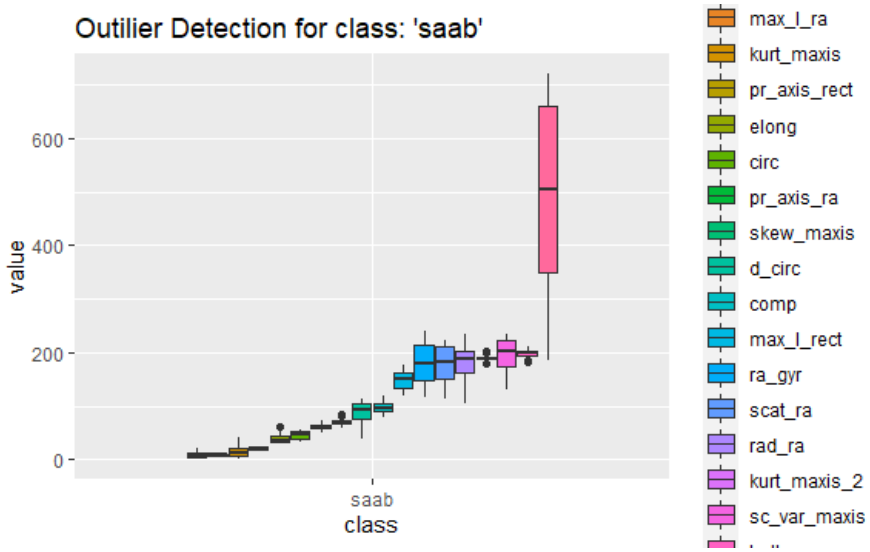


Outlier detection for bus class

```
vehicles_original %>%
  pivot_longer(2:19,names_to = "labels") %>%
  filter(class == "bus") %>%
  mutate(class = fct_reorder(class,value,median)) %>%
  ggplot(aes(class, value, fill = reorder(labels,value))) +
  geom_boxplot() +
  labs(title = "Outilier Detection for class: 'bus'")
```

Outlier detection for Saab class

```
vehicles_original %>%
  pivot_longer(2:19,names_to = "labels") %>%
  filter(class == "saab") %>%
  mutate(class = fct_reorder(class,value,median)) %>%
  ggplot(aes(class, value, fill = reorder(labels,value))) +
  geom_boxplot() +
  labs(title = "Outilier Detection for class: 'saab'")
```



Outlier detection for Opel class

```
vehicles_original %>%
  pivot_longer(2:19,names_to = "labels") %>%
  filter(class == "opel") %>%
  mutate(class = fct_reorder(class,value,median)) %>%
  ggplot(aes(class, value, fill = reorder(labels,value))) +
  geom_boxplot() +
  labs(title = "Outilier Detection for class: 'opel'")
```

Removing the outliers

```
vehicles_bus = vehicles_original %>%
  filter(class == "bus") %>%
  mutate(across(2:19, ~squish(.x, quantile(.x, c(.05, .95)))))

vehicles_van = vehicles_original %>%
  filter(class == "van") %>%
  mutate(across(2:19, ~squish(.x, quantile(.x, c(.05, .95)))))

vehicles_opel = vehicles_original %>%
  filter(class == "opel") %>%
  mutate(across(2:19, ~squish(.x, quantile(.x, c(.05, .95)))))

vehicles_saab = vehicles_original %>%
  filter(class == "saab") %>%
  mutate(across(2:19, ~squish(.x, quantile(.x, c(.05, .95)))))
```

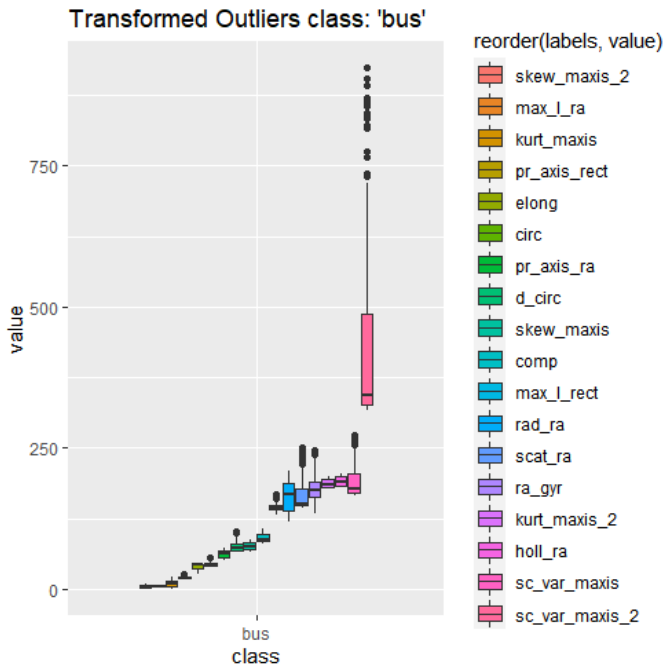
Squish - Squish the values into a range.
Quantile – It removes the outliers in the column.

After removing the outliers and merging them into a list, the transformed outliers for each class were plotted once more.

```
combined = bind_rows(list(vehicles_bus,vehicles_opel,vehicles_saab,vehicles_van)) %>%
  arrange(samples)


print(combined)
```

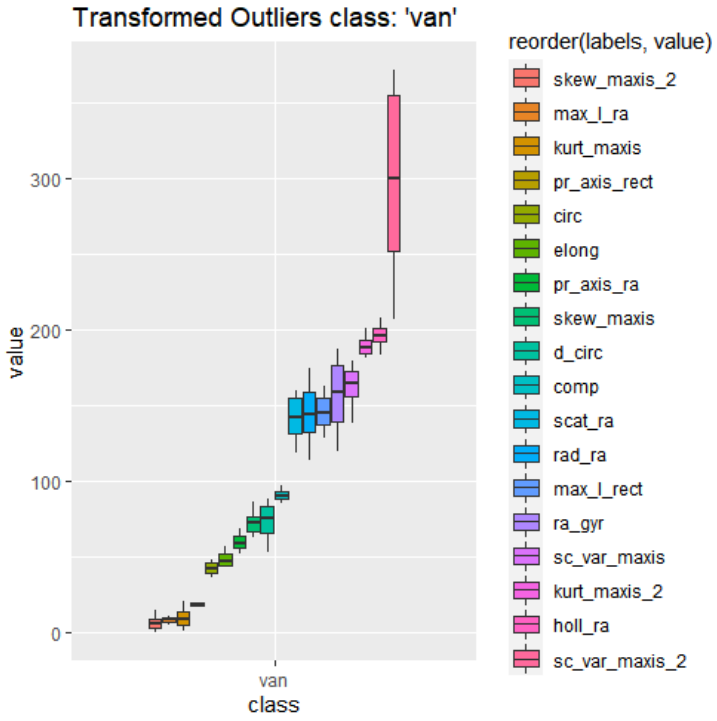
Combined bus class

```
combined %>%
  pivot_longer(2:19,names_to = "labels") %>%
  filter(class == "bus") %>%
  mutate(class = fct_reorder(class,value,median)) %>%
  ggplot(aes(class, value, fill = reorder(labels,value))) +
  geom_boxplot() +
  labs(title = "Transformed Outliers class: 'bus'")
```



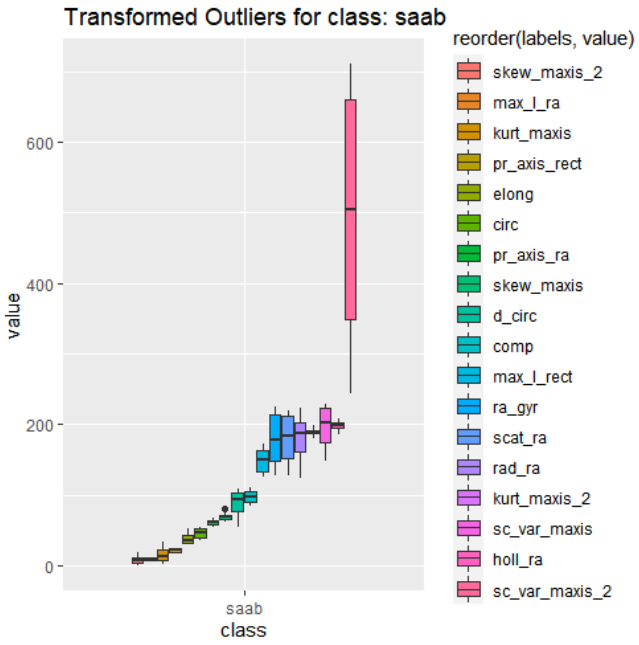
Combined van class

```
combined %>%
  pivot_longer(2:19,names_to = "labels") %>%
  filter(class == "van") %>%
  mutate(class = fct_reorder(class,value,median)) %>%
  ggplot(aes(class, value, fill = reorder(labels,value))) +
  geom_boxplot() +
  labs(title = "Transformed Outliers class: 'van'")
```



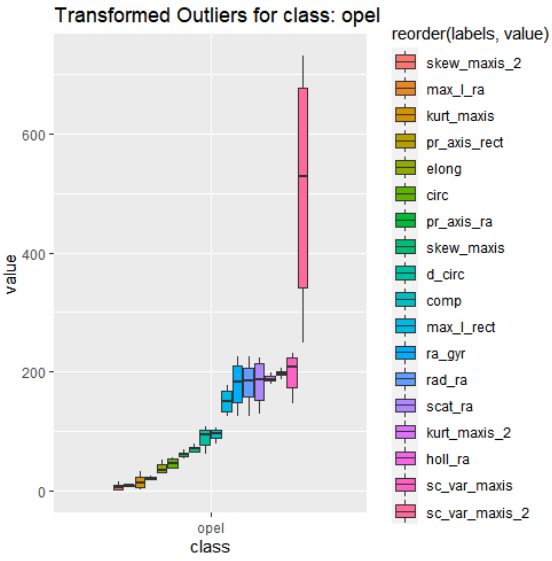
Combined Saab class

```
combined %>%
  pivot_longer(2:19,names_to = "labels") %>%
  filter(class == "saab") %>%
  mutate(class = fct_reorder(class,value,median)) %>%
  ggplot(aes(class, value, fill = reorder(labels,value))) +
  geom_boxplot() +
  labs(title = "Transformed Outliers for class: saab")
```



Combined Opel class

```
combined %>%
  pivot_longer(2:19,names_to = "labels") %>%
  filter(class == "opel") %>%
  mutate(class = fct_reorder(class,value,median)) %>%
  ggplot(aes(class, value, fill = reorder(labels,value))) +
  geom_boxplot() +
  labs(title = "Transformed Outliers for class: opel")
```



Next step is scaling data

Scale Data

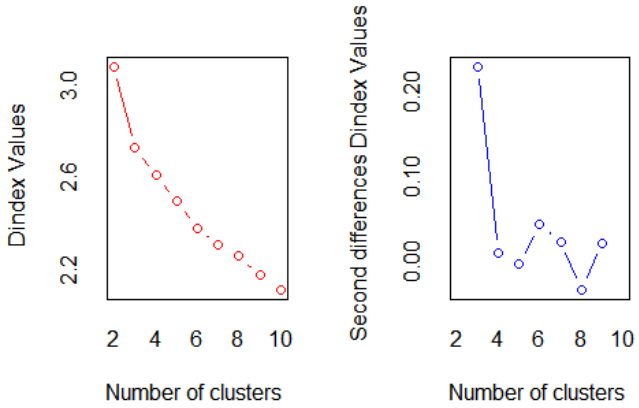
```
# Now that we have the "vehicles_data_points" dataset, scaling is performed

vehicles_scaled = vehicles_data_points %>%
  mutate(across(everything(), scale))
```

Euclidean distance

Euclidean distance is the basic metric for solving geometry problems. It's the distance between two points on a regular basis. It's one of the most common cluster analysis algorithms out there.

```
# Use Euclidean for distance

cluster_euclidean = NbClust(vehicles_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
```



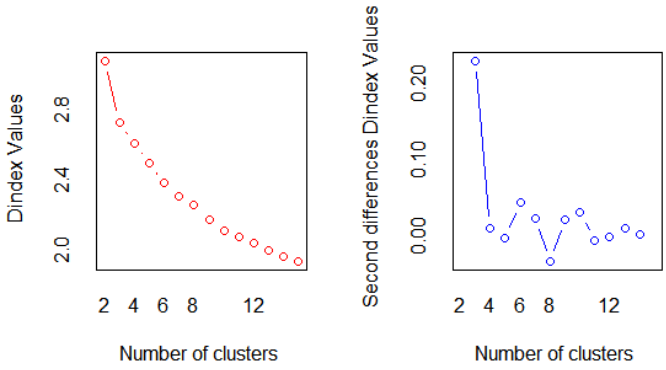
Confusion Matrix

```
       1   2
bus   56 162
opel 117  95
saab 117 100
van    0 199
```

Manhatten distance

It defines the absolute difference between two coordinate pairs.

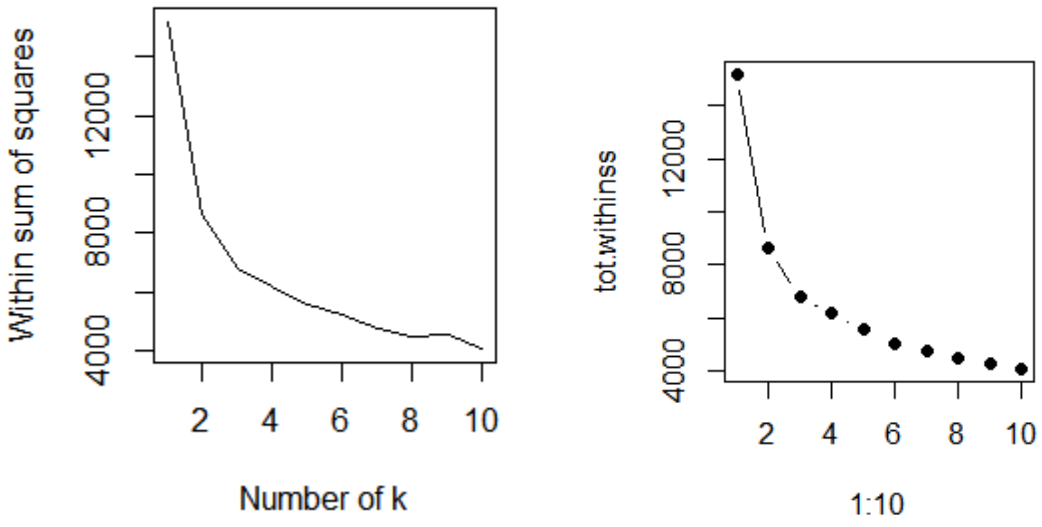
```
* Among all indices:
* 9 proposed 2 as the best number of clusters
* 9 proposed 3 as the best number of clusters
* 1 proposed 4 as the best number of clusters
* 1 proposed 7 as the best number of clusters
* 1 proposed 9 as the best number of clusters
* 1 proposed 14 as the best number of clusters
* 2 proposed 15 as the best number of clusters

                   ***** Conclusion *****

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



Using the elbow technique, determine the optimum number of clusters.

Elbow method

The elbow method is an algorithm used in cluster method to identify the number of clusters in a data set.

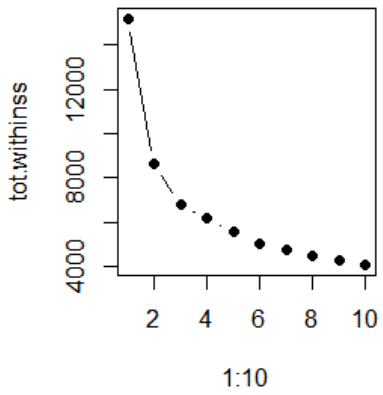


```r
#way 2
tot.withinss <- vector(mode="character", length=10)
for (i in 1:10){
  kc <- kmeans(vehicles_scaled, center=i, nstart=20)
  tot.withinss[i] <- kc$tot.withinss
}
```

Values	
i	10L
k	int [1:10] 1 2 3 4 5 6
tot.withinss	chr [1:10] "15209.99999
WSS	num [1:10] 15210 8611 6

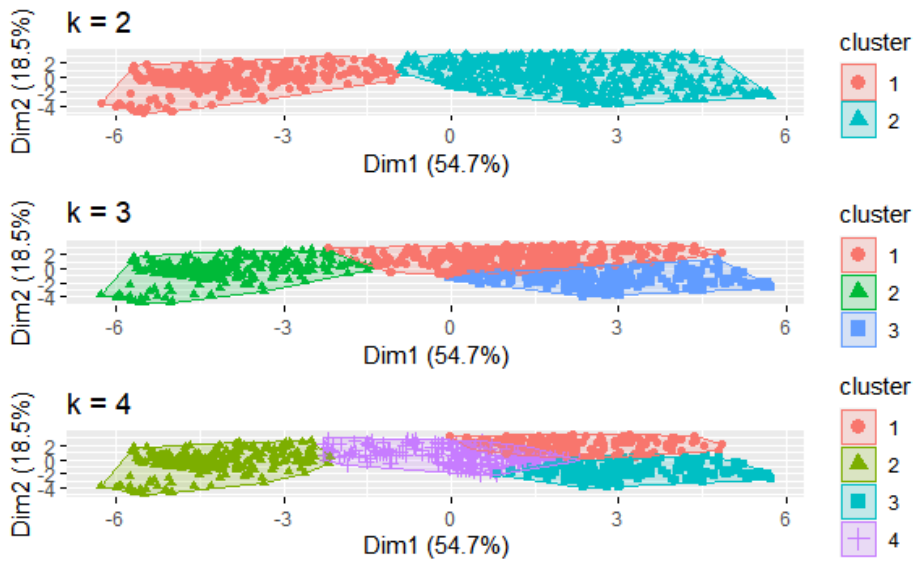
Files | Plots | Packages | Help | Viewer

Zoom | Export



K = 2 WSS

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



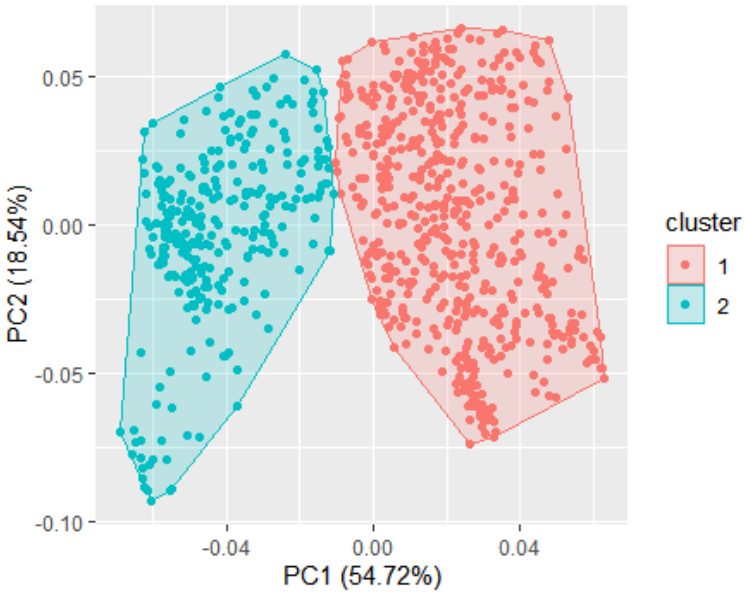
We wanted the total within-cluster sum of square (wss) to be as small as possible, so we found the within-cluster sum of square for k=2 that was the smallest.

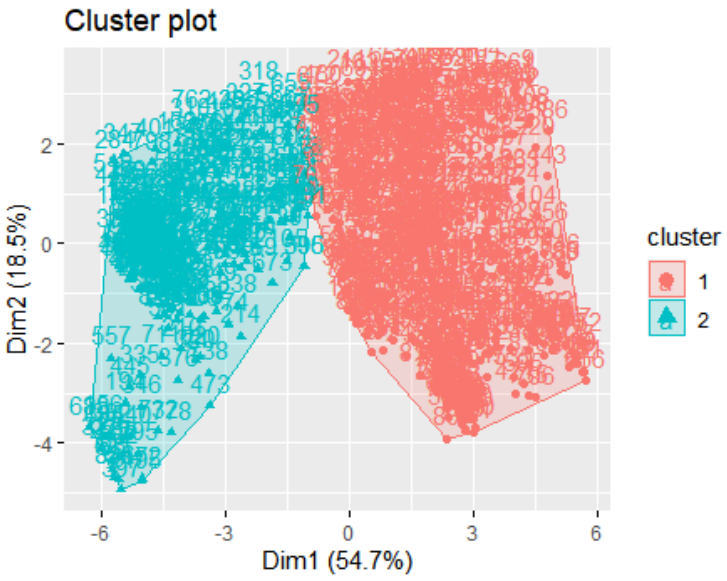
According to the graph, the cluster's optimal number is two.

As a result, the winning cluster is k= 2.

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

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





2nd Objective (MLP)

01) Time series analysis

A time series is a set of data points, which have a timestamp attached to it.

We used some methods for defining the input vector in time-series problems in this piece of code.

Xts

We need a R data structure that can deal with time series data. As a result, there are two sets. The first is zoo, and the second is xts bundles. We're using xts bundles in this case. When working with time series data, it has a lot of functions that come in handy. The index must be a date or time class that has been accepted. The xts package is more restricted than zoo because it implements powerful operations that require a time-based index.

```
library(xts)

# convert ExchangeUSD excel file to time series
time_series <- xts(ExchangeUSD$`USD/EUR`,ExchangeUSD$`YYYY/MM/DD`)
```

Lag

The relationship between individual values and their previous values will decide the lag value we choose.

Univariate

A single-variable time series is a set of measurements taken over time. A single variable or variate is referred to as univariate.

Multivariate

A multivariate time series is a set of multiple variables evaluated over time: multiple variates or multiple variables.

02) Lag takes an atomic vector and returns it with an additional attribute.

```
# target and predictor features
original_rate <-(time_series)
leg_rate <- stats::lag(original_rate,2)
all_rate <- cbind(original_rate,leg_rate)
colnames(all_rate) <- c('original_rate', 'leg_rate')
all_rate <- na.exclude(all_rate)
```

Before lag function

```
             [,1]
2011-10-13 1.3730
2011-10-14 1.3860
2011-10-17 1.3768
2011-10-18 1.3718
2011-10-19 1.3774
2011-10-20 1.3672
2011-10-21 1.3872
2011-10-24 1.3932
2011-10-25 1.3911
2011-10-26 1.3838
2011-10-27 1.4171
```

Processing lag function

```
> head(temporary_test)
           leg_rate
2013-05-21   1.2898
2013-05-22   1.2817
2013-05-23   1.2910
2013-05-24   1.2865
2013-05-27   1.2939
2013-05-28   1.2920
```

After lag function

```
           original_rate predicted_result
2013-05-21        1.2910         1.289367
2013-05-22        1.2865         1.281819
2013-05-23        1.2939         1.290491
2013-05-24        1.2920         1.286284
2013-05-27        1.2936         1.293211
2013-05-28        1.2870         1.291428
2013-05-29        1.2943         1.292930
2013-05-30        1.3042         1.286751
2013-05-31        1.2987         1.293587
2013-06-03        1.3097         1.302937
2013-06-04        1.3074         1.297732
2013-06-05        1.3088         1.308168
2013-06-06        1.3246         1.305977
```

03) Normalization

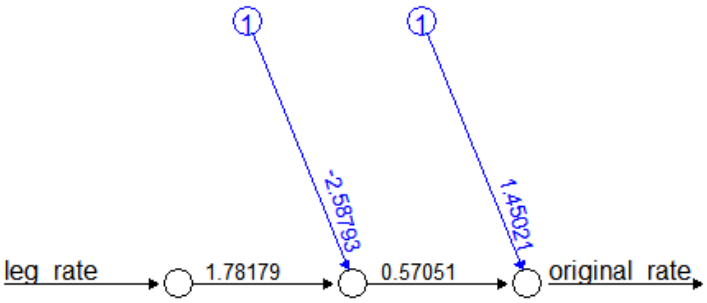
Data normalization techniques are used to create equivalent values for variables evaluated on different scales. Clustering and heat map visualization, principal component analysis (PCA), and other machine learning algorithms based on distance measures all benefit from this preprocessing phase.

```
# Normalization
normalization <- function(x, na.rm = TRUE) {
  return((x- min(x)) /(max(x)-min(x)))
}
```

The distance between the data points is so significant; some machine learning algorithms (SVM and KNN) are more sensitive to scale data than other algorithm. To avoid this problem, we convert the dataset to a common scale which is between 0 and 1, while maintaining the variable distributions. Min-max scaling is a term used to describe this form of scaling.

04) Various NN

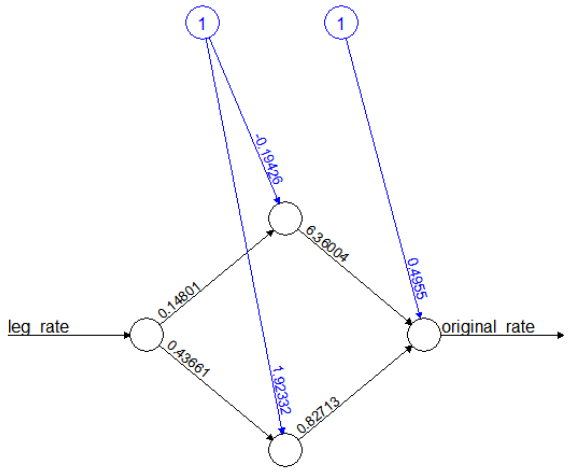
<u>If we have one hidden layer</u>



Error: 0.020988 Steps: 388

```
> nueralNetwork_fitting$result.matrix
                                    [,1]
error                         0.02098803
reached.threshold             0.00892346
steps                       388.00000000
Intercept.to.1layhid1        -2.58792712
leg_rate.to.1layhid1          1.78178550
Intercept.to.original_rate    1.45021269
1layhid1.to.original_rate     0.57051085
```

If we have two hidden layers



Error: 0.020604 Steps: 85

```
                                [,1]
error                      0.020604241
reached.threshold          0.002611445
steps                     85.000000000
Intercept.to.1layhid1     -0.194259714
leg_rate.to.1layhid1       0.148006112
Intercept.to.1layhid2      1.923318211
leg_rate.to.1layhid2       0.436614973
Intercept.to.original_rate 0.495503667
1layhid1.to.original_rate  6.360042107
1layhid2.to.original_rate  0.827132138
```

05) Statistical Indices

A statistical index is plural form of statistical index. The R-index is a crucial metric for evaluating and measuring product effects.

RMSE

RMSE mean Root Mean Squared Error. It is regarded as a good all-around error metric for numerical predictions. And, since it is scale-dependent, is a reasonable measure of accuracy, but it can only be used to compare prediction errors of different models or model configurations for a single variable, not between variables.

MAE

The abbreviation of MAE is Mean absolute Error. We are using this to calculate the average difference between two numeric vectors.

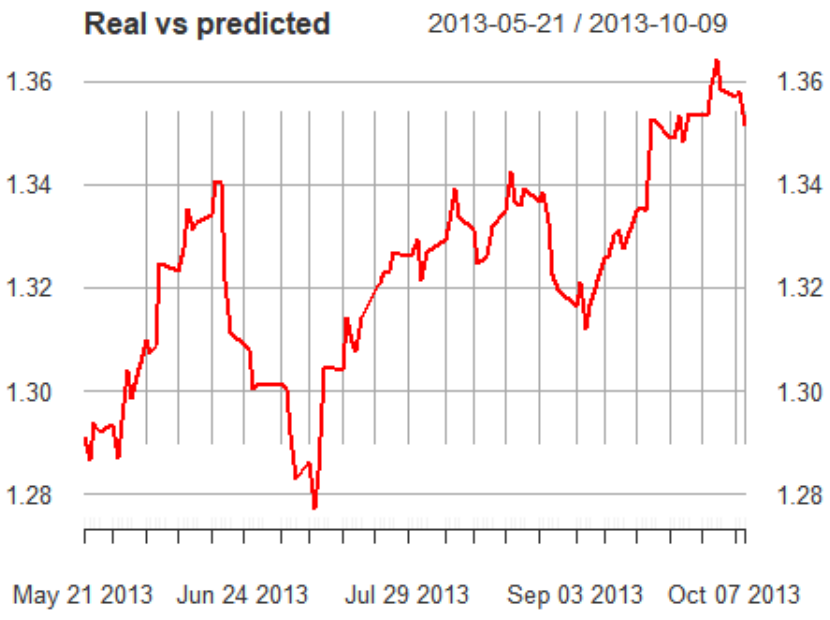
MAPE

The Abbreviation of MAPE is Mean Absolute Percent Error. We are using this method to verify forecast models. It is a standardized value. It never affected by the measurement unit.

07) You can see the below out which has original result and predicted result

	original_rate	predicted_result
2013-05-21	1.2910	1.289884
2013-05-22	1.2865	1.282179
2013-05-23	1.2939	1.291025
2013-05-24	1.2920	1.286745
2013-05-27	1.2936	1.293783
2013-05-28	1.2870	1.291976
2013-05-29	1.2943	1.293498
2013-05-30	1.3042	1.287220
2013-05-31	1.2987	1.294164
2013-06-03	1.3097	1.303580
2013-06-04	1.3074	1.298349
2013-06-05	1.3088	1.308810
2013-06-06	1.3246	1.306623
2013-06-07	1.3242	1.307955
2013-06-10	1.3234	1.322980
2013-06-11	1.3278	1.322599
2013-06-12	1.3354	1.321839
2013-06-13	1.3314	1.326023

Predicted results are more accurate than original one. As the decimal points of predicted results are more accurate than original value.



References

https://search.r-project.org/CRAN/refmans/ie2misc/html/mape.html#:~:text=MAPE%20is%20%22a%20measure%20to,all%20X_i%20values%20are%20positive%22%2C&text=%22If%20X%20and%20Y%20are,result%20in%20a%20different%20value.%22

https://en.wikipedia.org/wiki/Elbow_method_(clustering)#:~:text=In%20cluster%20analysis%2C%20the%20elbow,number%20of%20clusters%20to%20use.

https://www.geeksforgeeks.org/root-mean-square-error-in-r-programming/#:~:text=The%20rmse()%20function%20available,actual%20values%20and%20predicted%20values.&text=predicted%3A%20The%20predicted%20numeric%20vector,the%20corresponding%20element%20in%20actual.

**********Appendix**********

Objective 1

```
library(readxl)
library(NbClust)
library(janitor)
library(dplyr)
library(tidyr)
library(ggplot2)
library(tidyverse)
library(cluster)
library(knitr)
library(ggfortify)
library(tidymodels)
library(factoextra)
library(flexclust)

library(funtimes)
```

```
#reading the data
vehicles <- read_excel("E:/Lectures/DataScience/cw/vehicles.xlsx")

#viewing the data
view(vehicles)
summary(vehicles)


#pca_result <- prcomp(vehicles_scaled, scale = TRUE)
#names(pca_result)

#pca_result$rotation



#firs twe have to factor the data since we have the numerical and nominal data
#then used the  clean names method from the janitor package  to clean the dirty data
from the dataset
vehicles_fact <- mutate(vehicles,Class=as_factor(vehicles$Class))



summary(vehicles_fact)

vehicles_clean <-janitor::clean_names(vehicles_fact)
summary(vehicles_clean)

#van outlier

vehicles_clean %>%

  pivot_longer(2:19,names_to = "labels") %>%

  dplyr::filter(class == "van") %>%

  mutate(class = fct_reorder(class,value,median)) %>%
```

```
  ggplot(aes(class, value, fill = reorder(labels,value))) +
  geom_boxplot() +
  labs(title = "Outlier Detection for class: 'van'")
#bus outlier
vehicles_clean %>%
  pivot_longer(2:19,names_to = "labels") %>%
  filter(class == "bus") %>%
  mutate(class = fct_reorder(class,value,median)) %>%
  ggplot(aes(class, value, fill = reorder(labels,value))) +
  geom_boxplot() +
  labs(title = "Outlier Detection for class: 'bus'")
#saab outlier
vehicles_clean %>%
  pivot_longer(2:19,names_to = "labels") %>%
  filter(class == "saab") %>%
  mutate(class = fct_reorder(class,value,median)) %>%
  ggplot(aes(class, value, fill = reorder(labels,value))) +
  geom_boxplot() +
  labs(title = "Outlier Detection for class: saab")
#opel outlier
```

```
vehicles_clean %>%

  pivot_longer(2:19,names_to = "labels") %>%

  filter(class == "opel") %>%
  mutate(class = fct_reorder(class,value,median)) %>%

  ggplot(aes(class, value, fill = reorder(labels,value))) +

  geom_boxplot() +

  labs(title = "Outlier Detection for class: opel")


## outlier removal
vehicles_bus = vehicles_clean %>%

  filter(class == "bus") %>%

  mutate(across(2:19, ~squish(.x, quantile(.x, c(.05, .95)))))

vehicles_van = vehicles_clean %>%

  filter(class == "van") %>%

  mutate(across(2:19, ~squish(.x, quantile(.x, c(.05, .95)))))

vehicles_opel = vehicles_clean %>%

  filter(class == "opel") %>%

  mutate(across(2:19, ~squish(.x, quantile(.x, c(.05, .95)))))

vehicles_saab = vehicles_clean %>%

  filter(class == "saab") %>%

  mutate(across(2:19, ~squish(.x, quantile(.x, c(.05, .95)))))
```

```
combined = bind_rows(list(vehicles_bus,vehicles_opel,vehicles_saab,vehicles_van))
%>%

  arrange(samples)



print(combined)

combined %>%

  pivot_longer(2:19,names_to = "labels") %>%

  filter(class == "bus") %>%

  mutate(class = fct_reorder(class,value,median)) %>%

  ggplot(aes(class, value, fill = reorder(labels,value))) +

  geom_boxplot() +

  labs(title = "Transformed Outliers class: 'bus'")

combined %>%

  pivot_longer(2:19,names_to = "labels") %>%

  filter(class == "saab") %>%

  mutate(class = fct_reorder(class,value,median)) %>%

  ggplot(aes(class, value, fill = reorder(labels,value))) +

  geom_boxplot() +

  labs(title = "Transformed Outliers for class: saab")
```

```
combined %>%
  pivot_longer(2:19,names_to = "labels") %>%
  filter(class == "opel") %>%
  mutate(class = fct_reorder(class,value,median)) %>%
  ggplot(aes(class, value, fill = reorder(labels,value))) +
  geom_boxplot() +
  labs(title = "Transformed Outliers for class: opel")
combined %>%
  pivot_longer(2:19,names_to = "labels") %>%
  filter(class == "van") %>%
  mutate(class = fct_reorder(class,value,median)) %>%
  ggplot(aes(class, value, fill = reorder(labels,value))) +
  geom_boxplot() +
  labs(title = "Transformed Outliers for class: van")
# Remove the sample name and the class name. Both of these will be remove so that
only
#numerical data is left for the algorithm.
vehicles_data_points = combined %>%
  select(-samples, -class )
```

```
# Now that we have the "vehicles_data_points" dataset, scaling is performed

vehicles_scaled = vehicles_data_points %>%

  mutate(across(everything(), scale))



# Using a seed because the points taken for the cluster always changes when u run
the code
#set.seed(123)



# Perform the kmeans using the NbClust function



# Use Euclidean
#for distance

cluster_euclidean = NbClust(vehicles_scaled,distance="euclidean",
                    min.nc=2,max.nc=10,method="kmeans",index="all")



table(vehicles$Class,cluster_euclidean$Best.partition)



#clustering using manhattan distance

cluster_manhattan = NbClust(vehicles_scaled,distance="manhattan",
```

```
                    min.nc=2,max.nc=15,method="kmeans",index="all")

# Comparing the predicted clusters with the original data
table(vehicles$Class,cluster_manhattan$Best.partition)



 #finding the optimal number of cluster using elbow methods
  #way 1
 k = 1:10
 #set.seed(42)
 WSS = sapply(k, function(k) {kmeans(vehicles_scaled, centers=k)$tot.withinss})
 plot(k, WSS, type="1", xlab= "Number of k", ylab="Within sum of squares")



clusplot(vehicles,cluster_euclidean$Best.partition, color=T, shade=T, labels=0, lines=0)
# Use manhattan for distance

#way 2
tot.withinss <- vector(mode="character", length=10)
for (i in 1:10){
  kc <- kmeans(vehicles_scaled, center=i, nstart=20)
  tot.withinss[i] <- kc$tot.withinss
}

plot(1:10, tot.withinss, type="b", pch=19)
#as we can see the optimal number of cluster is 2

kc =kmeans(vehicles_scaled,2,nstart=20 )
kc
kc$tot.withinss

#table(vehicles$Class,kc$cluster)

autoplot(kc,vehicles_scaled,frame=TRUE)
```

```
kc$centers

k2 <- kmeans(vehicles_scaled, centers = 2, nstart = 25)
k3 <- kmeans(vehicles_scaled, centers = 3, nstart = 25)
k4 <- kmeans(vehicles_scaled, centers = 4, nstart = 25)
k5 <- kmeans(vehicles_scaled, centers = 5, nstart = 25)


k4$tot.withinss
# plots to compare
p1 <- fviz_cluster(k2, geom = "point", data = vehicles_scaled) + ggtitle("k = 2")
p2 <- fviz_cluster(k3, geom = "point",  data =vehicles_scaled) + ggtitle("k = 3")
p3 <- fviz_cluster(k4, geom = "point",  data =vehicles_scaled) + ggtitle("k = 4")


library(gridExtra)
grid.arrange(p1, p2, p3,  nrow = 3)

fviz_cluster(k2, data = vehicles_scaled)
```

Objective 2

```
library(tidyverse)
library(tseries)
library(readxl)
library(neuralnet)
library(quantmod)
library(xts)
library(funtimes)
library(dplyr)
library(MLmetrics)

ExchangeUSD <- read_excel("E:/Lectures/DataScience/cw/ExchangeUSD.xlsx")


# data Exploration
```

```
# YYYY/MM/DD Column Plotting
plot(ExchangeUSD$`YYYY/MM/DD`)

#  USD/EUR Column Plotting
plot(ExchangeUSD$`USD/EUR`)

# Convert a date column from a character string
ExchangeUSD$`YYYY/MM/DD` <- as.Date(ExchangeUSD$`YYYY/MM/DD`)

# convert ExchangeUSD excel file to time series
time_series <- xts(ExchangeUSD$`USD/EUR`,ExchangeUSD$`YYYY/MM/DD`)

# Printing time series
time_series

# target and predictor features
original_rate <-(time_series)
leg_rate <- stats::lag(original_rate,2)
all_rate <- cbind(original_rate,leg_rate)
colnames(all_rate) <- c('original_rate', 'leg_rate')
all_rate <- na.exclude(all_rate)

# Normalization
normalization <- function(x, na.rm = TRUE) {
  return((x- min(x)) /(max(x)-min(x)))
}

normalized <- as.data.frame(lapply(all_rate,normalization))

# Training and testing ranges
training_rate <- window(all_rate, end = '2013-05-21')
testing_rate <- window(all_rate, start = '2013-05-21')

# Plotting the training data
plot(training_rate)
```

```
set.seed(123)
# ANN Regression Fitting

#ternage active function used
nueralNetwork_fitting <- neuralnet(original_rate~leg_rate, data=training_rate,
hidden=2, act.fct= tanh)
nueralNetwork_fitting$result.matrix

# Graphic Neural network
plot(nueralNetwork_fitting)



# Test the accuracy of the model
temporary_test <- subset(testing_rate, select = 'leg_rate')
head(temporary_test)
nueralNetwork_fitting.results <- compute(nueralNetwork_fitting, temporary_test)
results <- data.frame(actual_result = testing_rate$original_rate, predicted_result=
nueralNetwork_fitting.results$net.result)
results

predicted_result <- nueralNetwork_fitting.results$net.result
actual_result <- testing_rate$original_rate

# standard statistical indices
RMSE(predicted_result,actual_result)
MAE(predicted_result,actual_result)
MAPE(predicted_result,actual_result)

kmeans(RMSE,MAE)
plot(actual_result,main = "Real vs predicted", col='red')
```