



INFORMATICS  
INSTITUTE OF  
TECHNOLOGY

UNIVERSITY OF  
WESTMINSTER

## **Informatics Institute of Technology**

### **Department of Computing**

**BSc in Computer Science**

**Module: 5DATA001C**

**Machine Learning & Data Mining**

**Module Leader: Mr. Achala Aponso**

### **COURSE WORK REPORT**

**Tutorial Group** : SE/CS Group B

**Student IIT ID** : 20191191

**Student UoW ID** : W17903352

**Student First Name** : Mahfoos

**Student Surname** : Ahamed

## Table of Contents

<b>Objective 1 (Partitioning clustering)</b> .....	4
<b>Methodologies used for reducing the input dimensionality.</b> .....	4
<b>Pre- Processing tasks</b> .....	4
<b>Code and plot of Outliers detection</b> .....	4
<b>Bus Class</b> .....	4
<b>Van Class</b> .....	5
<b>Saab Class</b> .....	5
<b>Opel Class</b> .....	6
<b>Code of Removing the outliers</b> .....	6
<b>Scaling the data</b> .....	7
<b>Number of cluster centers</b> .....	7
<b>Silhouette Method</b> .....	7
<b>Code</b> .....	7
<b>NbClust</b> .....	8
<b>Code</b> .....	8
.....	8
<b>K-means analysis for each attempt</b> .....	9
<b>Evaluation of the produced outputs against 19<sup>th</sup> column</b> .....	10
<b>Final Winner cluster case</b> .....	10
<b>Objective 2 (MLP)</b> .....	11
<b>Methods for defining the input vector in time-series problems.</b> .....	11
<b>Evidence of various adopted Input vectors and the related input/output metrics</b> .....	11
<b>Evidence of correct normalization</b> .....	12
<b>Implement Number of MLPs, using various structures (layers/nodes)</b> .....	13
<b>Discussion of the meaning of these stat. indices</b> .....	14
<b>Best results (prediction output vs. desired output)</b> .....	15
<b>Appendix</b> .....	16
<b>Objective 1 Code</b> .....	16
<b>Objective 2 Code</b> .....	21
<b>References</b> .....	24

## **Table of Figure**

Figure 1 Outlier Detection bus.....	4
Figure 2 Outlier Detection van.....	5
Figure 3 Outlier Detection saab .....	5
Figure 4 Outlier Detection opel .....	6
Figure 5 Silhouette method .....	7
Figure 6 NbClust.....	8
Figure 7 Table cluster .....	9
Figure 8 Cluster plot.....	10

## Objective 1 (Partitioning clustering)

### Methodologies used for reducing the input dimensionality.

- Here is a brief review of techniques for dimensionality reduction:

#### Principal Component Analysis (PCA).

- Principal component analysis (PCA) is a statistical procedure that orthogonally transforms the original  $n$  numeric dimensions of a dataset into a new set of  $n$  dimensions called principal components. As a result of the transformation, the first principal component has the largest possible variance each succeeding principal component has the highest possible variance under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding principal components.
- Keeping only the first  $m < n$  principal components reduce the data dimensionality while retaining most of the data information, i.e., variation in the data. Notice that the PCA transformation is sensitive to the relative scaling of the original columns, and therefore, the data need to be normalized before applying PCA. Also notice that the new coordinates (PCs) are not real, system-produced variables anymore. Applying PCA to your dataset loses its interpretability. If interpretability of the results is important for your analysis, PCA is not the transformation that you should apply.

#### Pre- Processing tasks

- In the preprocessing task first, identify and detection the outlier of each class (attribute).
- After that removed the outliers

#### Code and plot of Outliers detection

##### 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 = "Outlier Detection for class: 'bus'")
```

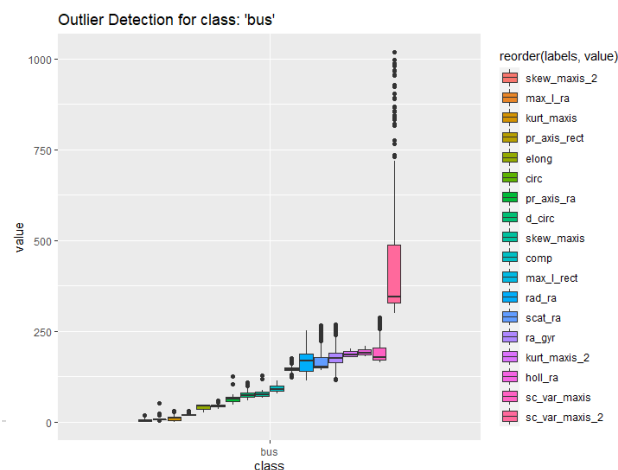


Figure 1 Outlier Detection bus

### 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 = "Outlier Detection for class: 'van'")
```

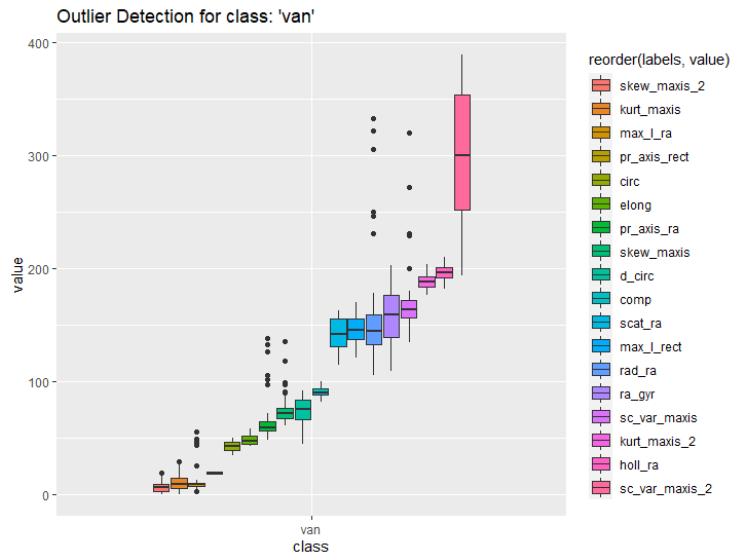


Figure 2 Outlier Detection van

### 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 = "Outlier Detection for class: 'saab'")
```

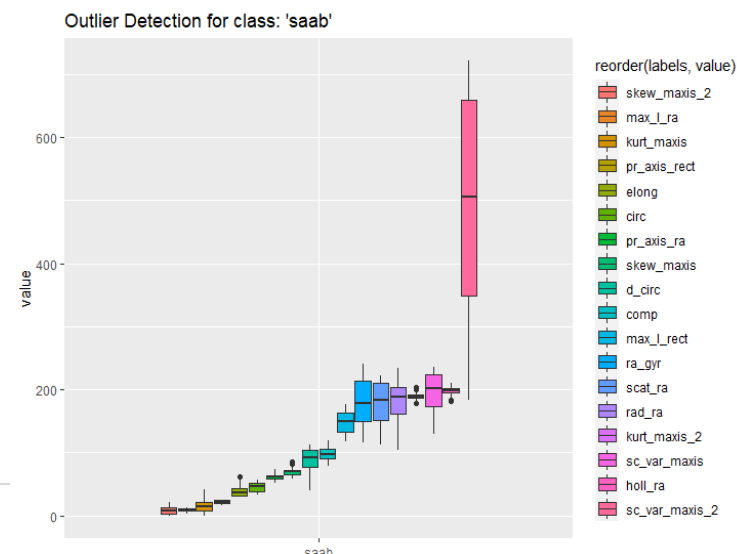


Figure 3 Outlier Detection saab

## 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 = "Outlier Detection for class: 'opel'")
```

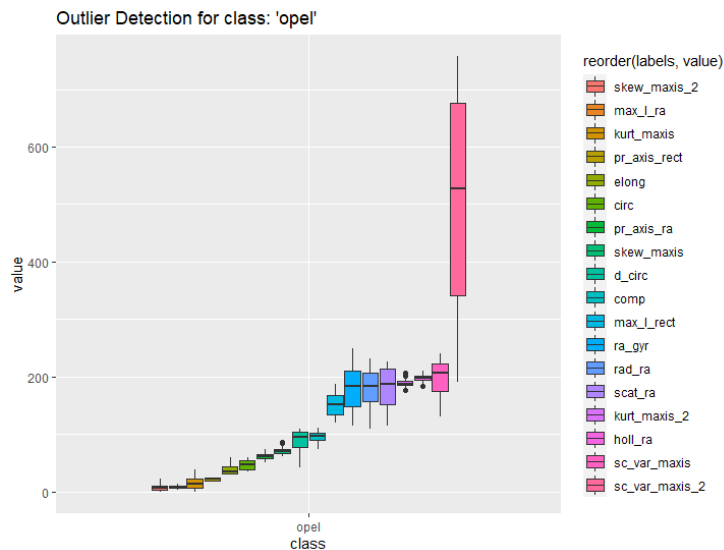


Figure 4 Outlier Detection opel

## Code of Removing the outliers

# Remove the Outlier

```
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)))))
```

```
# Combining all class
```

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

### Scaling the data

- When you have variables, which are measured in different scales it is useful to scale the data.

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

### Number of cluster centers

- A variety of measures have been used in the partition clustering for evaluating clustering results. The term clustering validation is used to design the procedure of evaluating the results of a clustering algorithm. There are more than thirty indices and methods for identifying the optimal number of clusters.

### Silhouette Method

- This method can help determine the optimal number of clusters is called a silhouette method. Average silhouette method computes the average silhouette of observations for different values of k. The optimal number of clusters k is the one that maximize the average silhouette over a range of possible values for k.

#### Code

```
fviz_nbclust (vehicles_scaled, kmeans, method = "silhouette", k.max = 24) +  
  theme_minimal () + ggtitle ("The Silhouette Plot")
```

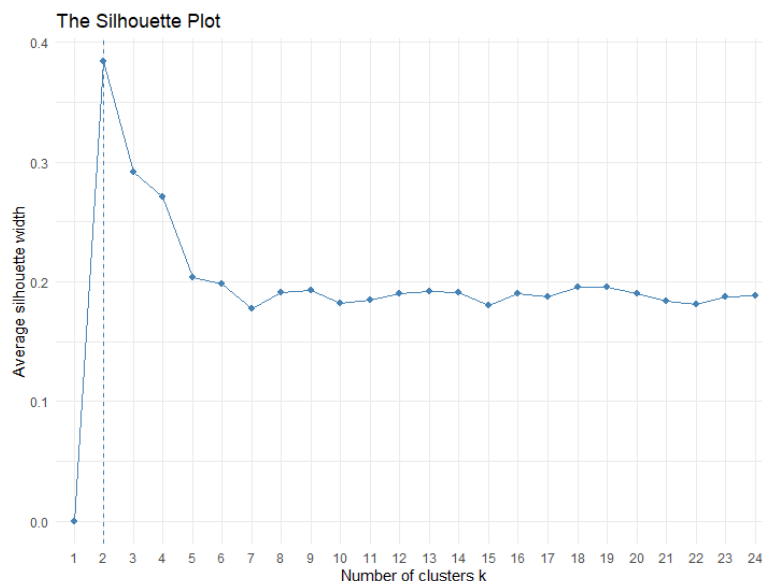


Figure 5 Silhouette method

## NbClust

- The **NbClust** package provides 30 indices for determining the relevant number of clusters and proposes to users the best clustering scheme from the different results obtained by varying all combinations of number of clusters, distance measures, and clustering methods.
- Used Euclidean distance method.

### Code

```
# Use Euclidean for distance
```

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

```
# Plot the best cluster
```

```
factoextra::fviz_nbclust(cluster_euclidean) + theme_minimal() +  
  ggtitle("NbClust's optimal number of clusters")
```

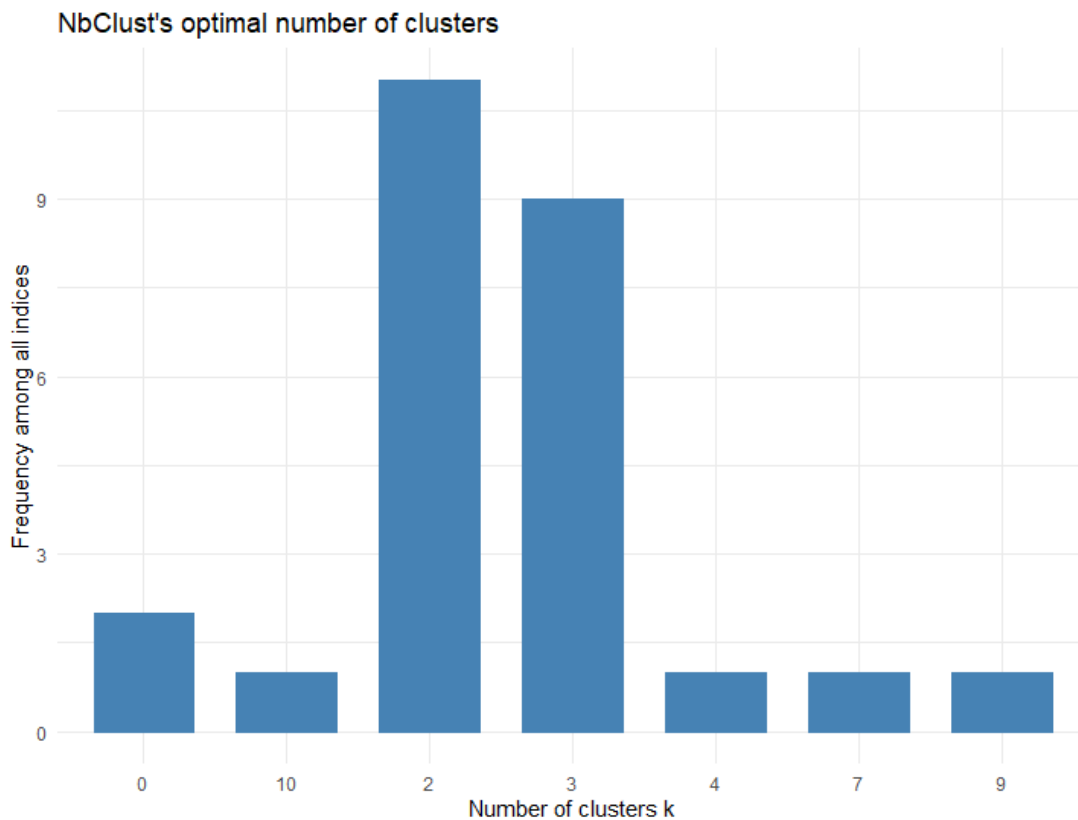


Figure 6 NbClust



### K-means analysis for each attempt

- There are few steps need to do for get the final cluster.
  1. Specify the number of clusters(k)
  2. Allocate objects to clusters.
  3. Compute cluster means.
  4. Allocate each observation to the closet cluster center.
  5. Repeat steps 2 and 4 until the solution converges.
- In this vehicle data set, there are 2 cluster below code for that

### Code of k-mean

#compute k-means in R with the kmeans() function:

```
kmean_result<- kmeans(vehicles_scaled ,centers = 2, nstart = 30) # aplly k-means  
algorithm with no. of k = 2
```

- In this code using kmeans function
- Assume the center 2 because in top identified the cluster using Euclidean method

### The output from k-means cluster analysis

- The main output from k-means cluster analysis is a table showing the mean values of each cluster on the clustering variables. The table of means produced from examining the data is shown below.

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

*Figure 7 Table cluster*

## Evaluation of the produced outputs against 19<sup>th</sup> column

- When it comes to evaluating clusters, some classes would not differentiate according to the receiving data sets but there are some which would. In this graph, it is clear that two out of the four clusters we identified, are differentiating in a good manner. Other two classes do not differentiate better because they have so many similarities in their data sets.

### Plot of the cluster

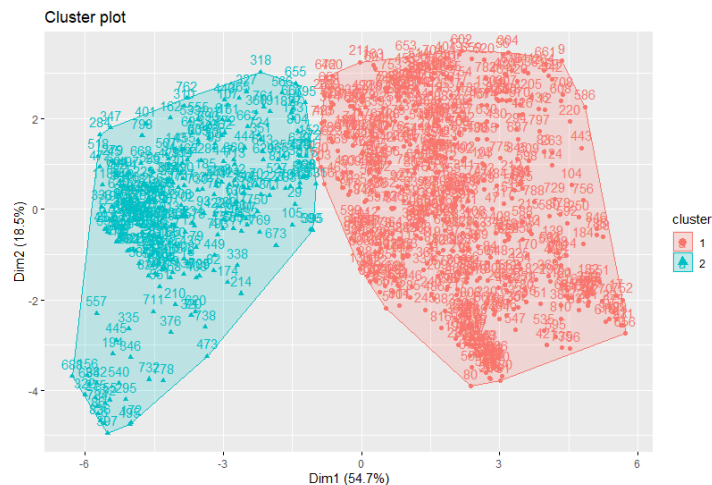


Figure 8 Cluster plot

## Final Winner cluster case

- Final winner cluster is determined according to the auto suggested number of clusters. That is basically suggested according to the majority rule. As the number of clusters which was suggested is 2, the clustering of the scenario is carried out with 2 cluster.
- Used two method to find the cluster.
  - Euclidean method
  - Manhattan method

### Euclidean method

```
*****
* 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 method

```
*****
* 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

*****
```

## Objective 2 (MLP)

### Methods for defining the input vector in time-series problems.

- There are some input vectors available for defining time series problem in this task Performed lag operation for input vectors.
- First converted to timeseries from the data set using xts package after that used lag operation

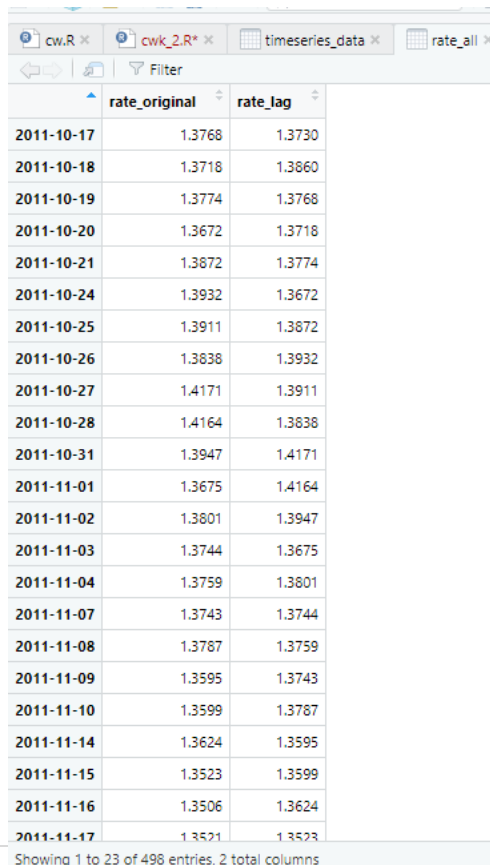
#### Code for lag operation

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

# target and predictor features
rate_original <- (timeseries_data)
rate_lag <- stats::lag(rate_original,2)
rate_all <- cbind(rate_original,rate_lag)
colnames(rate_all) <- c('rate_original', 'rate_lag')
rate_all <- na.exclude(rate_all)
```

### Evidence of various adopted Input vectors and the related input/output metrics

- In this dataset has input vectors below shown the table it includes original data and lagged data.



The screenshot shows an RStudio window with four tabs: 'cw.R', 'cwk\_2.R\*', 'timeseries\_data', and 'rate\_all'. The 'rate\_all' tab is active, displaying a data table with two columns: 'rate\_original' and 'rate\_lag'. The table contains 23 rows of data, each representing a date from 2011-10-17 to 2011-11-17. The 'rate\_original' column shows values ranging from 1.3506 to 1.4171, and the 'rate\_lag' column shows values ranging from 1.3523 to 1.3947. The table is filtered to show the first 23 entries.

	rate_original	rate_lag
2011-10-17	1.3768	1.3730
2011-10-18	1.3718	1.3860
2011-10-19	1.3774	1.3768
2011-10-20	1.3672	1.3718
2011-10-21	1.3872	1.3774
2011-10-24	1.3932	1.3672
2011-10-25	1.3911	1.3872
2011-10-26	1.3838	1.3932
2011-10-27	1.4171	1.3911
2011-10-28	1.4164	1.3838
2011-10-31	1.3947	1.4171
2011-11-01	1.3675	1.4164
2011-11-02	1.3801	1.3947
2011-11-03	1.3744	1.3675
2011-11-04	1.3759	1.3801
2011-11-07	1.3743	1.3744
2011-11-08	1.3787	1.3759
2011-11-09	1.3595	1.3743
2011-11-10	1.3599	1.3787
2011-11-14	1.3624	1.3595
2011-11-15	1.3523	1.3599
2011-11-16	1.3506	1.3624
2011-11-17	1.3521	1.3523

Showing 1 to 23 of 498 entries, 2 total columns

## Evidence of correct normalization

### Normalization

- Normalization is a rescaling of the data from the original range so that all values are within the range of 0 and 1.
- Normalization requires that you know or can accurately estimate the minimum and maximum observable values.

### why normalization procedure is necessary.

- Among the best practices for training a Neural Network is to normalize your data to obtain a mean close to 0. Normalizing the data generally speeds up learning and leads to faster convergence. Also, the (logistic) sigmoid function is hardly ever used anymore as an activation function in hidden layers of Neural Networks because the **tanh** function (among others) seems to be strictly superior.
- While this might not be immediately evident, there are very similar reasons for why this is the case. The **tanh** function is quite like the logistic sigmoid. The main difference, however, is that the tanh function outputs result between -1 and 1, while the sigmoid function outputs values that are between 0 and 1 — therefore they are always positive.

### Normalization code

```
normalize <- function (x, na.rm = TRUE) {  
  return ((x- min(x)) /(max(x)-min(x)))  
}  
nomralizes_data <- as.data.frame(lapply(rate_all,normalize))
```

## Implement Number of MLPs, using various structures (layers/nodes)

- Implemented MLPs using various Structure.
- In this neural network used the **tanh** activation function

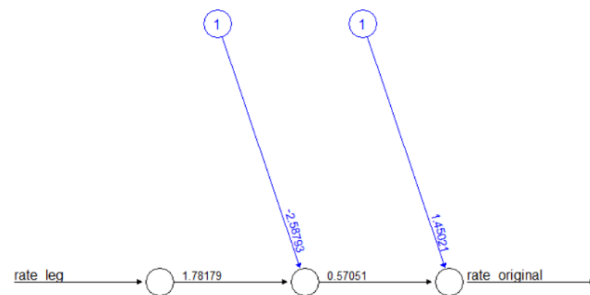
### Hidden layer (node 1)

```
set.seed(123)
```

```
# ANN Regression Fitting
```

```
nueral_fit <- neuralnet(rate_original~rate_leg, data=rate_train, hidden=1, act.fct= tanh) #ternage  
active function used
```

```
nueral_fit$result.matrix
```



Error: 0.020988 Steps: 388

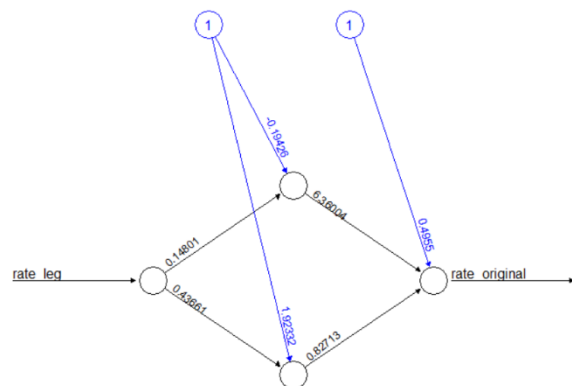
### Hidden layer (node 2)

```
set.seed(123)
```

```
# ANN Regression Fitting
```

```
nueral_fit <- neuralnet(rate_original~rate_leg, data=rate_train, hidden=2, act.fct= tanh) #ternage  
active function used
```

```
nueral_fit$result.matrix
```



Error: 0.020604 Steps: 85

### Comparison table for each statistical indices

Hidden layer nodes	RMSE	MAE	MAPE
One node (lag 2)	0.008976236	0.006892914	0.055208714
One node (lag 1)	0.006274847	0.04821299	0.003648057
Two nodes (lag 1)	0.006216536	0.004793206	0.003626394
Two nodes (lag 2)	0.00882458	0.006728537	0.005084909

### Discussion of the meaning of these stat. indices

#### Statistical indices

- These values describe the distribution of the elements in a set of elements. These **indices** are useful for examining the relation among subsets of elements, to verify or to contradict some rules and correlations which control the behavior of the subsets of elements.
- In this project calculated the various indices

#### RMSE (Root Mean Square Error)

- **RMSE** Is a standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; **RMSE** is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.

#### MAE (Mean Absolute Error)

- **MAE** is a measure of errors between paired observations expressing the same phenomenon. Examples of Y versus X include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement.

#### MAPE (Mean Absolute Percentage Error)

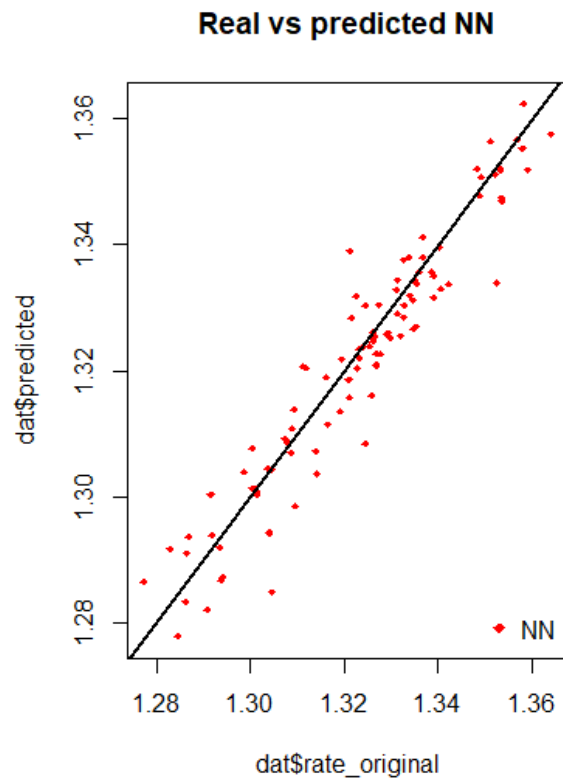
- The mean absolute percentage error (**MAPE**) is a measure of how accurate a forecast system is. It measures this accuracy as a percentage and can be calculated as the average absolute percent error for each time minus actual values divided by actual values.

```
> RMSE(predicted,actual)
[1] 0.008976236
> MAE(predicted,actual)
[1] 0.006892914
> MAPE(predicted,actual)
[1] 0.055208714
```

### Best results (prediction output vs. desired output)

- Below plots the Actual vs predicted graph and best statistical indices

#### Plot of the actual vs Predicted.



- Best statistical indices finding based on low value of it.

#### Best statistical indices

RMSE(predicted,actual)

[1] 0.006201589

MAE(predicted,actual)

[1] 0.004717682

MAPE(predicted,actual)

[1] 0.003571156

## Appendix

### Objective 1 Code

```
library(tidyverse)

library(readxl) # using for read the excel file

library(factoextra)

library(NbClust) # For finding cluster

library(caret)

library(tidymodels)


# Read in the original excel datafile

vehicles_original <- read_excel("D:/Accademic Materials/IIT/02nd Year/Second
Semester/Machine Learnig/CourseWork/vehicles.xlsx") %>%

#plot(vehicles_original)

janitor::clean_names() %>%

#https://www.rdocumentation.org/packages/janitor/versions/1.2.0/topics/clean_names

mutate(class = as_factor(class))


# Get a birds eye view of how the dataset looks like and detect the Outlier

summary(vehicles_original)


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 = "Outlier Detection for class: 'van'")
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 = "Outlier Detection for class: 'bus'")
```

```
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 = "Outlier Detection for class: 'saab'")
```

```
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 = "Outlier Detection for class: 'opel'")
```

# Remove the Outlier

```
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)))))

# Combined the all claas

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

print(combined)

# Transformed Outliers 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 %>%
  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 %>%
  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")
```

# Remove the sample name and the class name. Both of these will be remove so that only n

```

#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))

set.seed(123)

#Determining Optimal Number of Clusters
# Use Euclidean for distance
cluster_euclidean = NbClust(vehicles_scaled,distance="euclidean",
min.nc=2,max.nc=10,method="kmeans",index="all")

# Plot the best cluster
factoextra::fviz_nbclust(cluster_euclidean) + theme_minimal() +
  ggtitle("NbClust's optimal number of clusters")

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

# Plot the best cluster
factoextra::fviz_nbclust(cluster_manhattan) + theme_minimal() +
  ggtitle("NbClust's optimal number of clusters")

# The Silhouette Method
fviz_nbclust(vehicles_scaled, kmeans, method = "silhouette", k.max = 24) + theme_minimal() +
  ggtitle("The Silhouette Plot")

#compute k-means in R with the kmeans() function:
result<- kmeans(vehicles_scaled ,centers = 2, nstart = 30) # aply k-means algorithm with no.
of k = 2

```

```
# Cluster plot  
fviz_cluster(result, data = vehicles_scaled)  
  
#Printing the table  
table(vehicles_original$class,result$cluster)
```

### Objective 2 Code

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

```
Exchange_usd <- read_excel("D:/Accademic Materials/IIT/02nd Year/Second  
Semester/Machine Learnig/CourseWork/ExchangeUSD.xlsx")
```

```
# data Exploration
```

```
plot(Exchange_usd$`YYYY/MM/DD`) # Plotting the YYYY/MM/DD Column  
plot(Exchange_usd$`USD/EUR`) # Plotting the USD/EUR Column  
Exchange_usd$`YYYY/MM/DD` <- as.Date(Exchange_usd$`YYYY/MM/DD`) # Convert character  
string column to date  
timeseries_data <- xts(Exchange_usd$`USD/EUR`,Exchange_usd$`YYYY/MM/DD`) # convert  
excel file to time series  
timeseries_data # Print time series  
  
# target and predictor features
```

```

rate_original <- (timeseries_data)
rate_lag <- stats::lag(rate_original,1)
rate_all <- cbind(rate_original,rate_lag)
colnames(rate_all) <- c('rate_original', 'rate_lag')
rate_all <- na.exclude(rate_all)

# Normalizing
normalize <- function(x, na.rm = TRUE) {
  return((x- min(x)) /(max(x)-min(x)))
}
nomralizes_data <- as.data.frame(lapply(rate_all,normalize))
# Training and testing ranges
rate_train <- window(rate_all, end = '2013-05-21')
rate_test <- window(rate_all, start = '2013-05-21')

plot(rate_train) # plotting the train data

set.seed(123)
# ANN Regression Fitting
nueral_fit <- neuralnet(rate_original~rate_lag, data=rate_train, hidden=1, act.fct= tanh) #ternage
active function used
nueral_fit$result.matrix

# Graphic Neural network
plot(nueral_fit)

# Test the accuracy of the model
temp_test <- subset(rate_test, select = 'rate_lag')
head(temp_test)
nueral_fit.results <- compute(nueral_fit, temp_test)
results <- data.frame(actual = rate_test$rate_original, predicted= nueral_fit.results$net.result)
results

```

```
predicted <- nueral_fit.results$net.result
actual <- rate_test$rate_original

# standard statistical indices
RMSE(predicted,actual)
MAE(predicted,actual)
MAPE(predicted,actual)

data <-data.frame(actual=actual, predicted = predicted)

par(mfrow=c(1,2))
plot(data$rate_original,data$predicted,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN',pch=18,col='red', bty='n')
```

## References

- [10 Tips for Choosing the Optimal Number of Clusters | by Matt.O | Towards Data Science](#)
- [neuralnet: Train and Test Neural Networks Using R | DataScience+ \(datascienceplus.com\)](#)
- [\(Tutorial\) NEURAL NETWORK Models in R - DataCamp](#)
- [PCA and K-means Clustering of Delta Aircraft | R-bloggers](#)
- [R Course. Forecasting Models - YouTube](#)