5DATA002W

Machine Learning & Data Mining

Coursework (2020/21)

Student Name : Anuja Herath

Student IIT ID : 20191180

Student UOW ID : W1790023

Module Leader : Aponso Achala Chathuranga

Due Date : 6th of may 2021

Table of Contents

Part 1

Description

Supervised Deep Learning

Supervised deep learning entails the use of multi-layered algorithms to determine which output target class it belongs to and to predict its value by mapping its optimal relationship with input predictor data. The two most common supervised deep learning tasks are classification and regression.

Artificial Neural Network Regression

A supervised deep learning artificial neural network predicts output target feature by dynamically processing output target and input predictors data through a multi-layer network of optimally weighted node connections. Nodes are divided into input, hidden, and output layers.

Brief discussion of methodologies used to reduce the dimensionality

In an efficient manner, dimensionality reduction is used to downsize data.

Principal Component Analysis (PCA)

PCA decreases the data set's dimensionality, allowing the majority of the variability to be explained with fewer variables. PCA is sometimes used as one of several steps in a sequence of analyses. If there are so many predictors in relation to the number of observations, PCA can be used to reduce the number of variables.

Principal component analysis (PCA) is a statistical procedure that transforms the original *n* numeric dimensions of a dataset into a new set of *n* dimensions called principal components.

the data need to be normalized before applying PCA.

Normalization

In a data frame, there can be instances where one feature's value is in the range of 1-100, while another feature's value is in the range of 1-10000000. In cases like these, simply having a larger numeric range can have a greater effect on response variables than having a smaller numeric range, which can affect prediction accuracy.

The aim is to increase predictive accuracy by preventing a specific feature from influencing the prediction due to a wide numeric value range. As a result, we can need to normalize or scale values under various features in order for them to fall into a common range.

Scaling and outlier removal

First, remove the outliers, then scale the data. Otherwise, you could end up with different standard deviations for different variables.

outlier detection

An outlier is a value or observation that deviates greatly from other observations, or a data point that is significantly different from other data points.

Because we have both numerical and nominal data, we must first factorize the data.

justification-

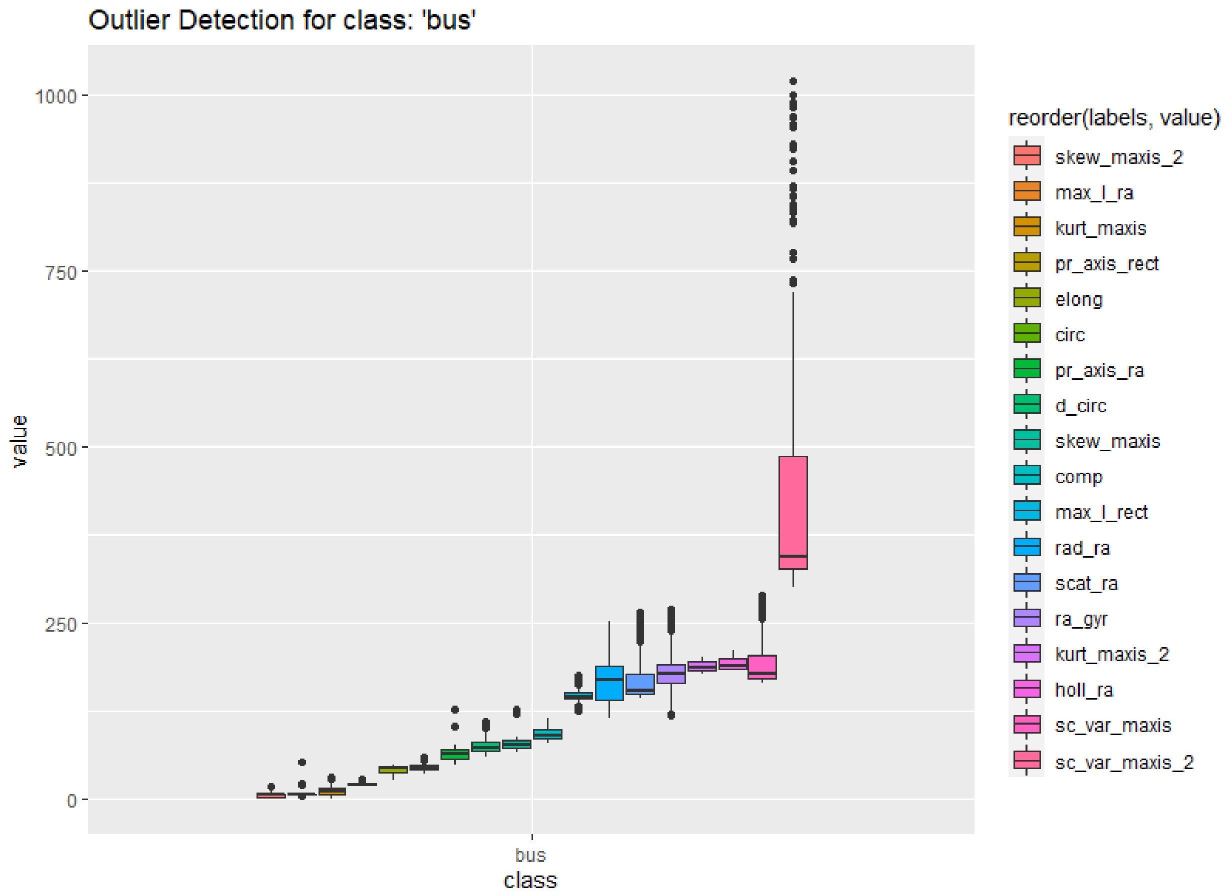
Kurt.maxis	Kurt.Maxis	Holl.Ra	Class
16	187	197	van
14	189	199	van
9	188	196	saab
10	199	207	van
11	180	183	bus
9	181	183	bus

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

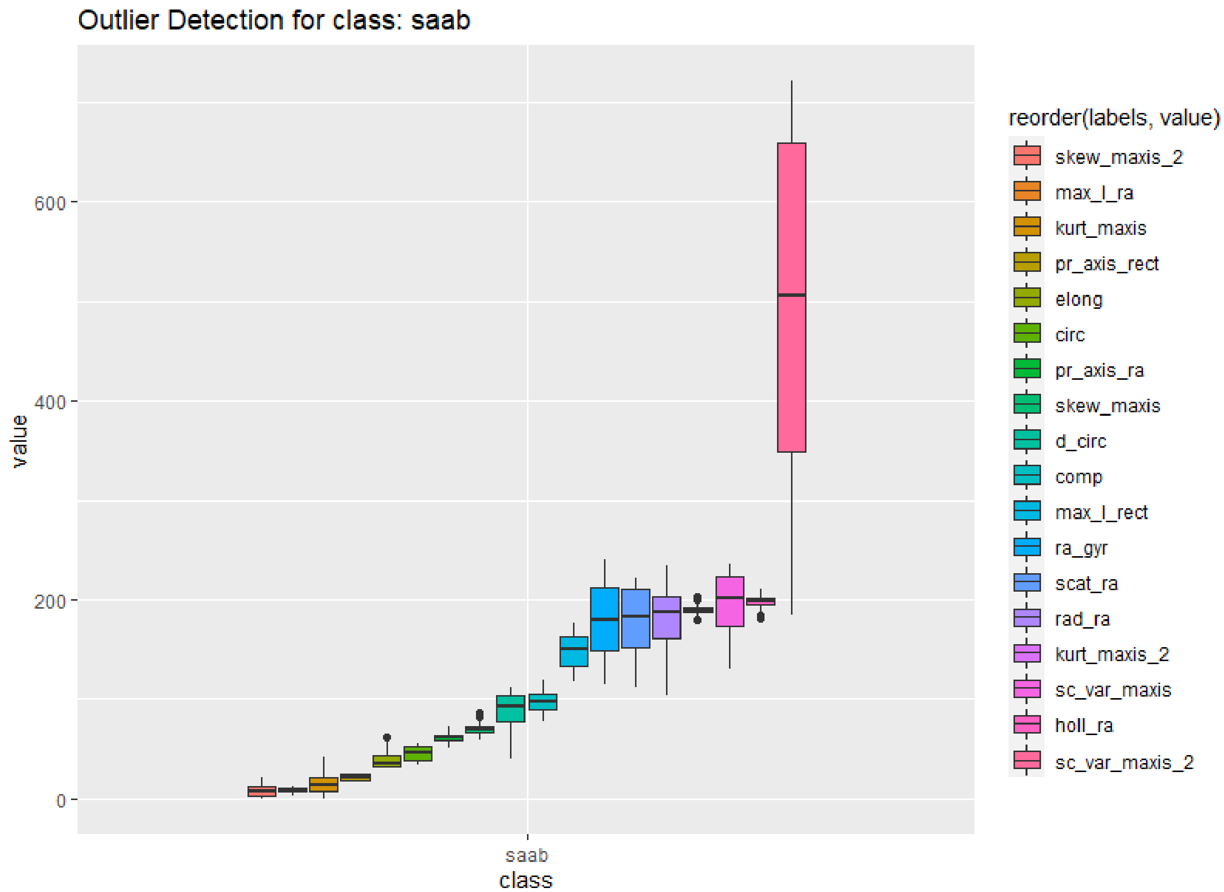
```
summary(vehicles_fact)

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

outlier detection for van class

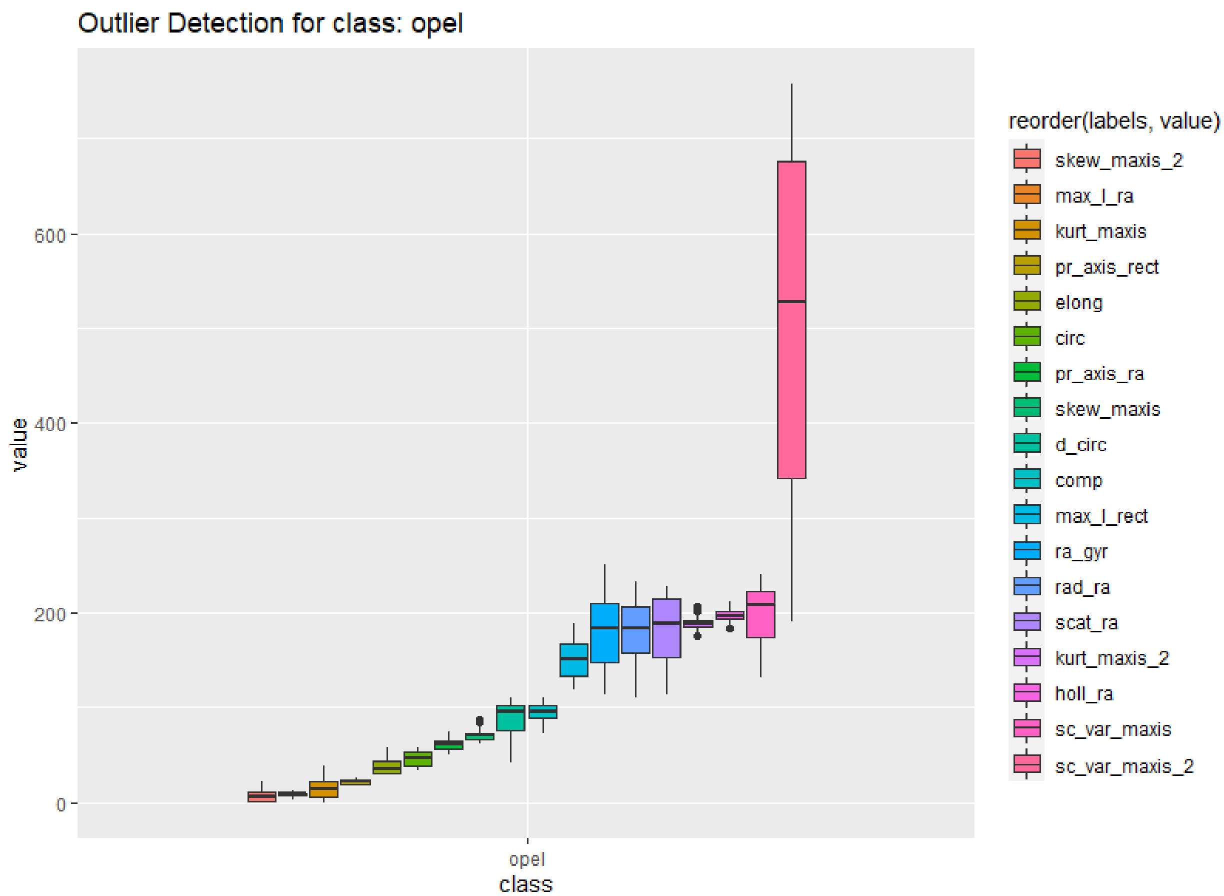
Outlier detection for bus class



Outlier detection for Saab class



Outlier detection for opal class



Removing the outliers

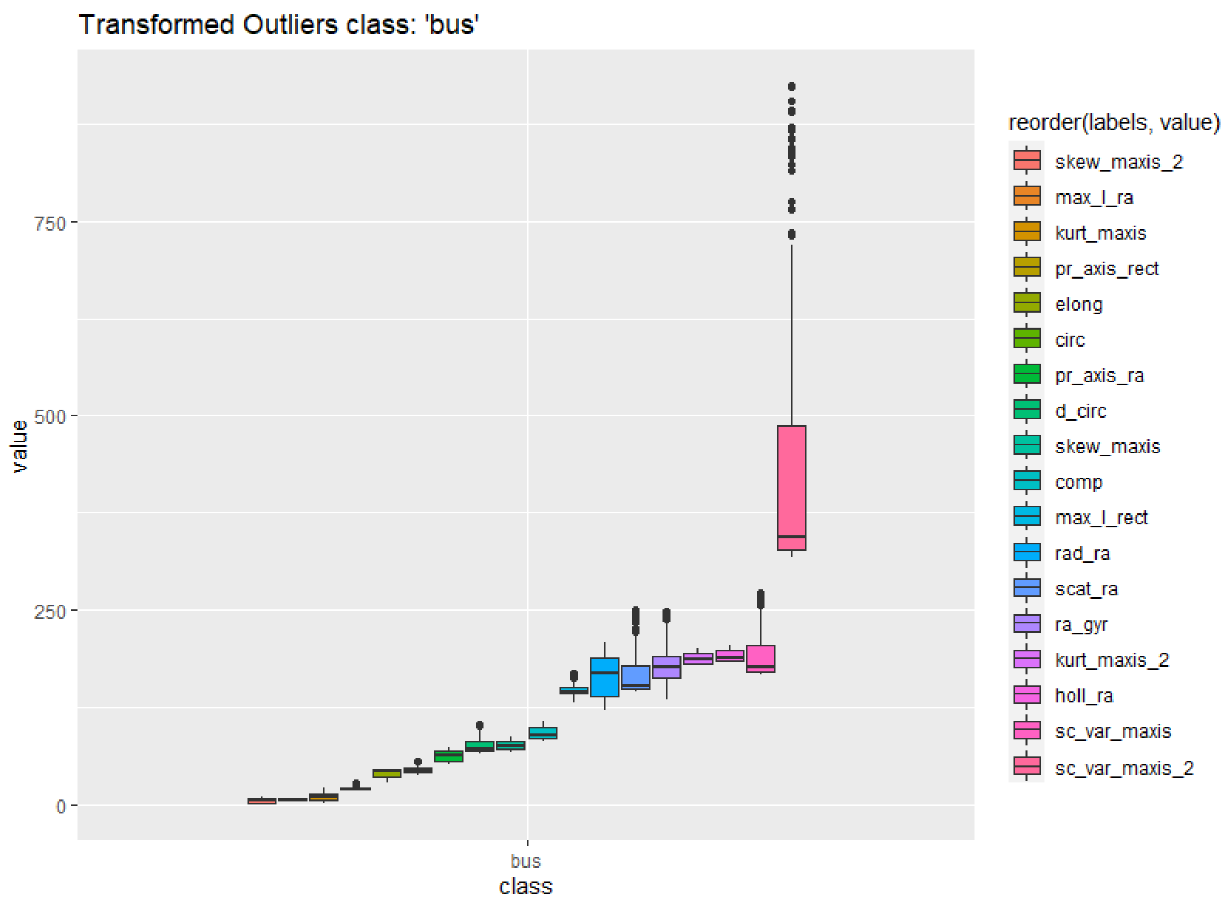
```
## outlier removal
vehicles_bus = vehicles_clean %>%

  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_va

  arrange(samples)
```

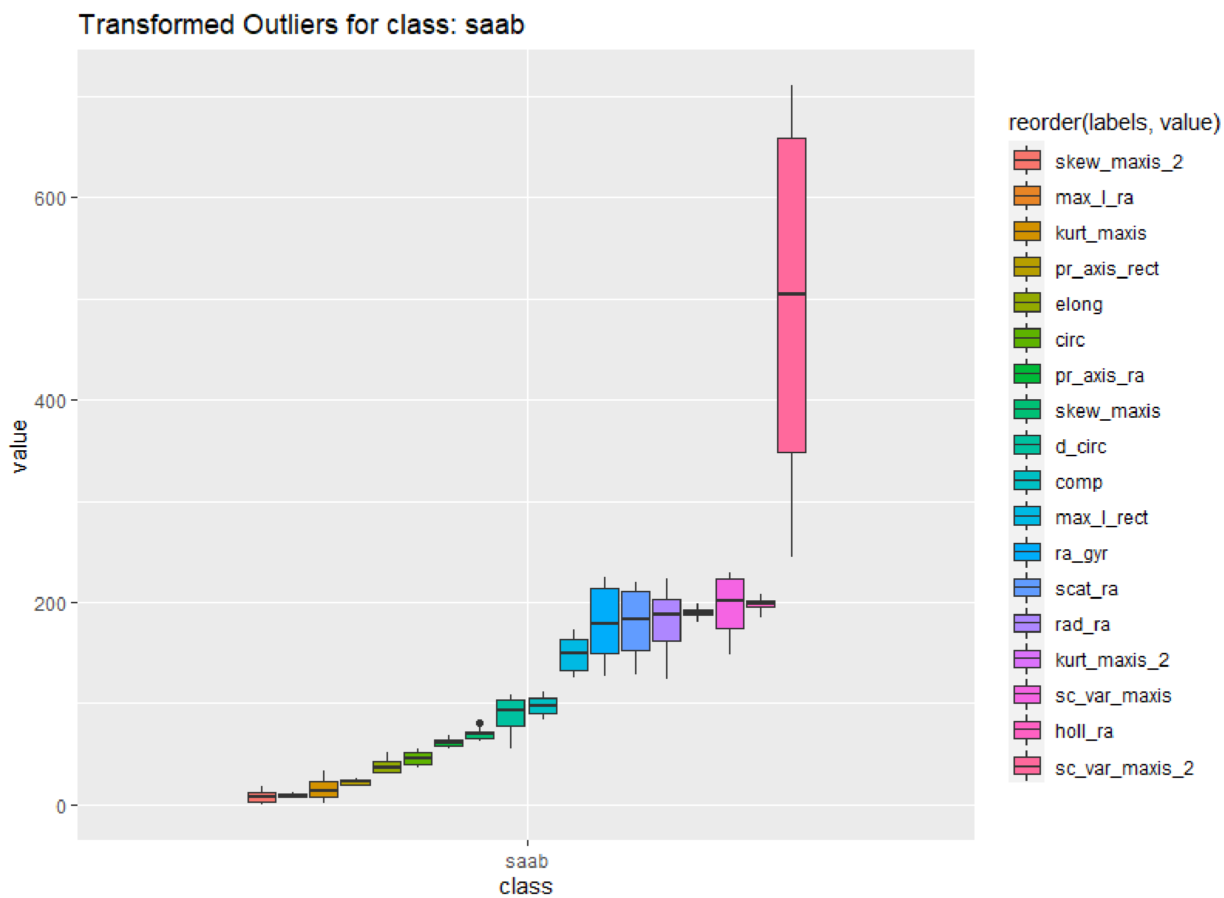
Squish – squish the values into a range

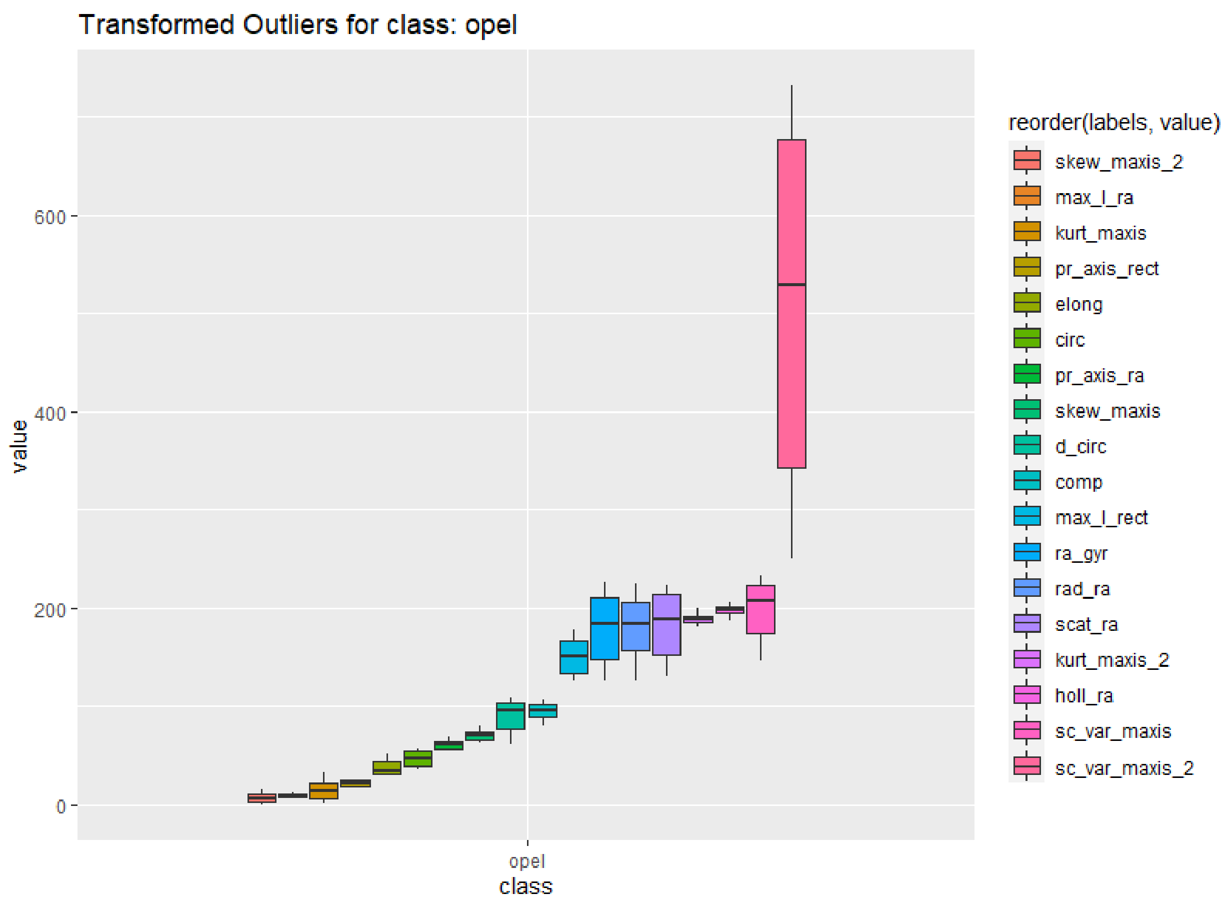
Quantile-removing the outliers in a column

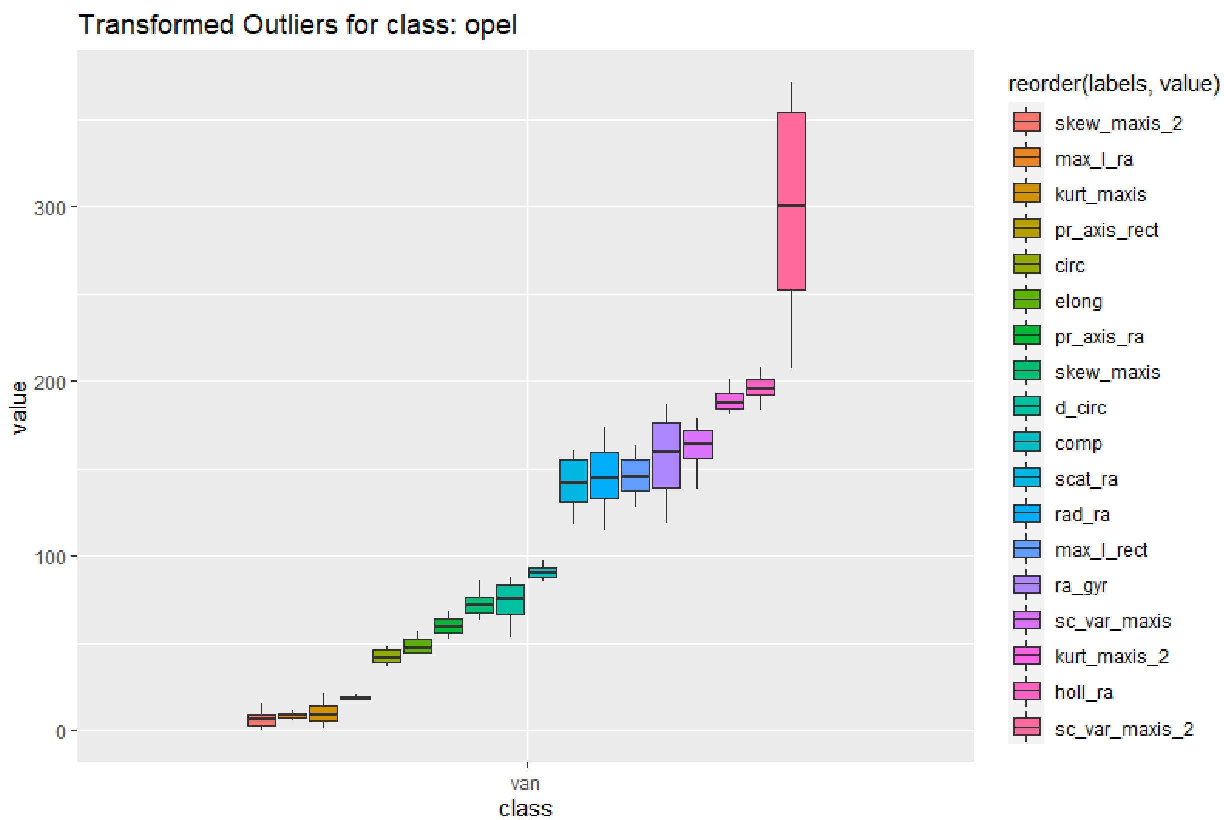
Removing the outliers for every class

And combining them into a list and plotting the transformed outliers for every class again


Transformed Outliers class: 'bus'
value
750
500
250
0
bus
class
reorder(labels, value)
skew_maxis_2
max_l_ra
kurt_maxis
pr_axis_rect
elong
circ
pr_axis_ra
d_circ
skew_maxis
comp
max_l_rect
rad_ra
scat_ra
ra_gyr
kurt_maxis_2
holl_ra
sc_var_maxis
sc_var_maxis_2


Transformed Outliers for class: saab
value
600
400
200
0
saab
class
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


Transformed Outliers for class: opel
value
600
400
200
0
opel
class
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



after the outlier removal scale the data

scaling the data

```
# Now that we have the "vehicles_data_points" dataset, scaling is performed
vehicles_scaled = vehicles_data_points %>%
  mutate(across(everything(), scale))
```

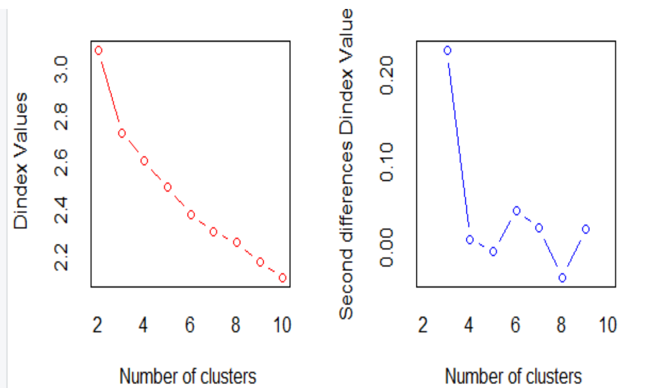
Scale- is generic function whose default method centers and/or scales the columns of a numeric matrix.

	comp	circ	d_circ	rad_ra	pr_axis_ra	max_l_ra	scat_ra	elong	pr_axis_rect	ma
1	0.17130414	0.5215512	0.05203019	0.19359370	1.25746195	0.91249786	-0.27284600	0.2806401	-0.2327082	
2	-0.33746646	-0.6463039	0.11735265	-0.87667479	-0.78315260	0.41219119	-0.60983866	0.5436819	-0.6272396	
3	1.31603800	0.8552241	1.55444680	1.32872694	0.87139974	0.91249786	1.16703172	-1.1660898	0.9508859	
4	-0.08308116	-0.6463039	-0.01329227	-0.29289198	0.31988229	0.41219119	-0.76301714	0.6752028	-0.6272396	
5	-1.10062236	-0.1457945	-0.79716181	1.19899743	2.34211292	-0.08811548	-0.60983866	0.5436819	-0.6272396	
6	1.69761595	1.8562428	1.36827779	0.12872894	-1.88618749	-1.08872881	2.48896201	-1.8236943	2.5290113	

Kmeans clustering

Using Euclidean distance

```
            second differences plot) that corresponds to a significant increase of t
he 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

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

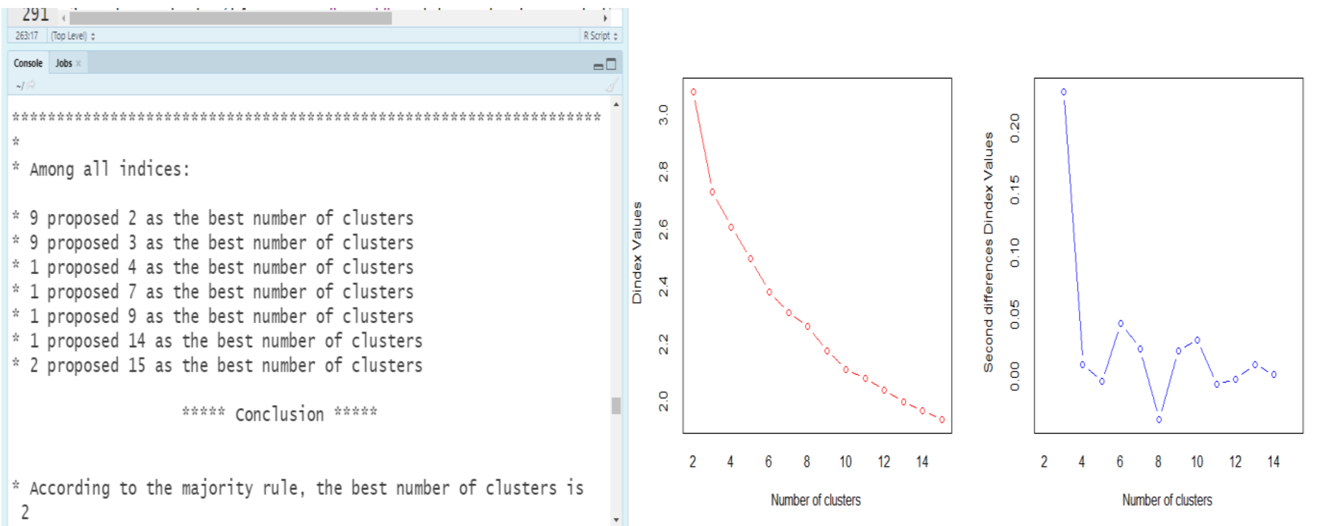



```
# Use Euclidean
#for distance

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

Confusion matrix

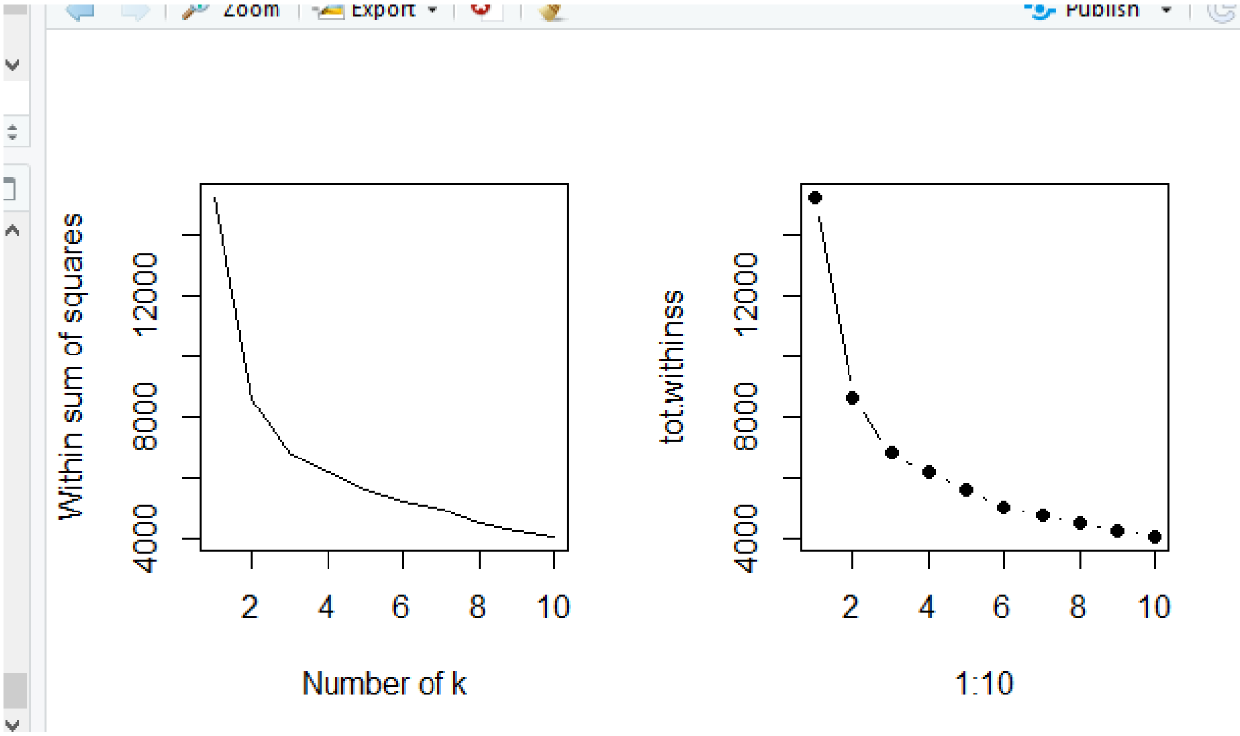
```
        1   2
bus    56 162
opel  117  95
saab  117 100
van     0 199
>
```

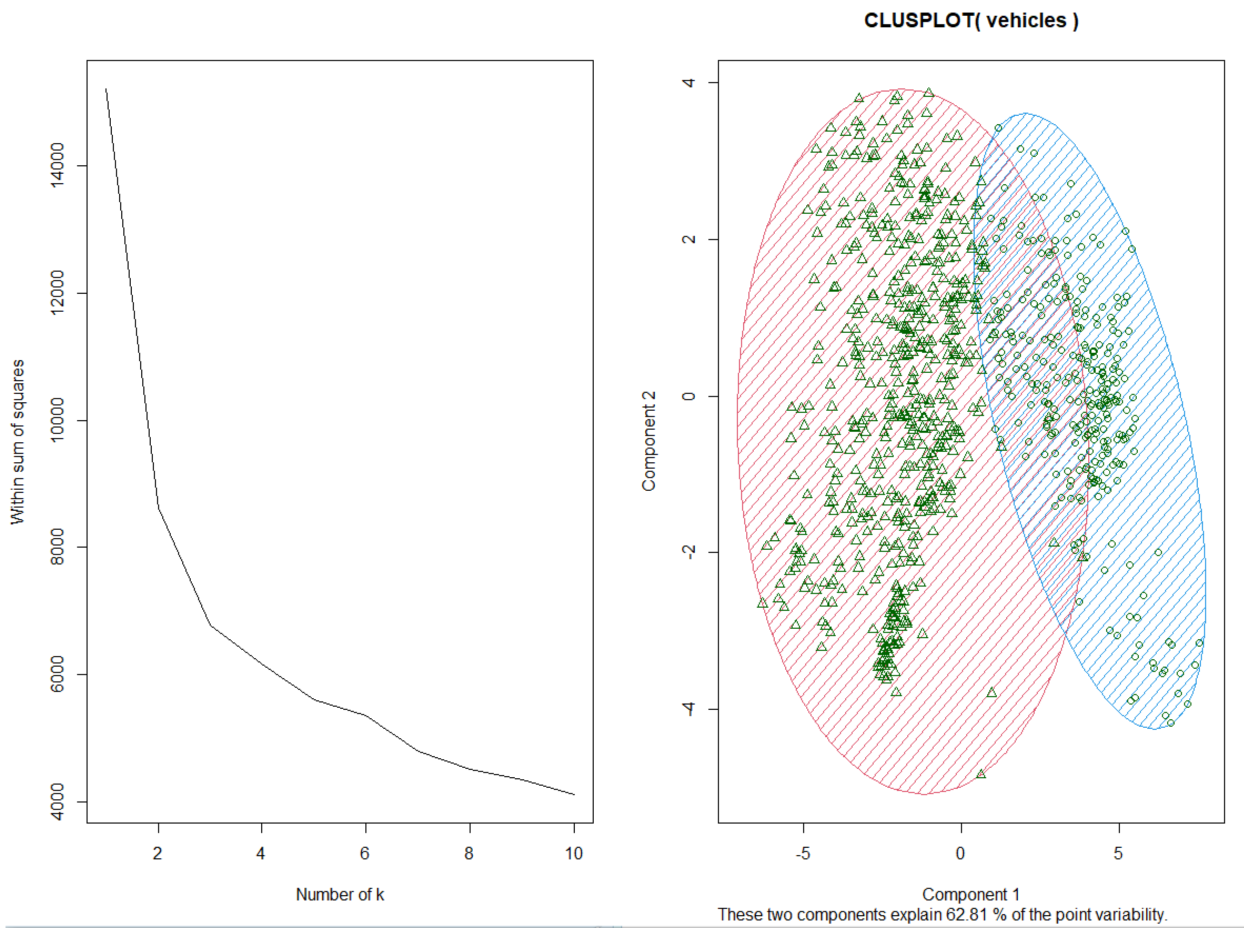
Using Manhattan distance

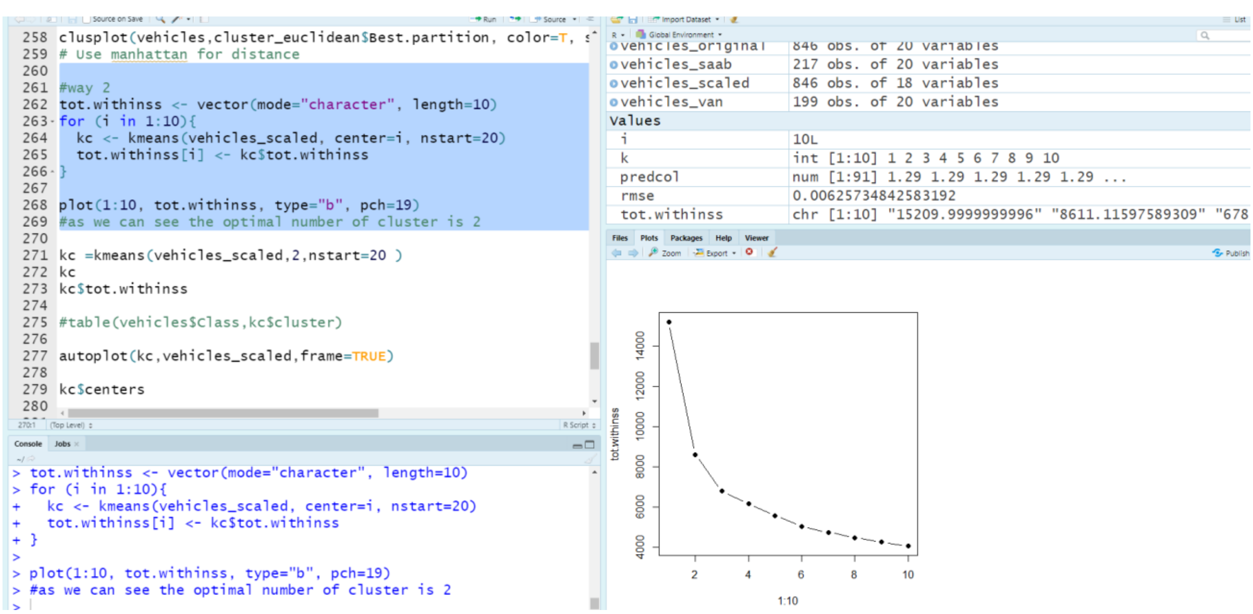


Finding the optimal number of cluster number using elbow method

The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters


Zoom
Export
Publish
Within sum of squares
12000
8000
4000
2
4
6
8
10
Number of k
tot.withinss
12000
8000
4000
2
4
6
8
10
1:10


CLUSPLOT(vehicles)
Within sum of squares
Number of k
Component 2
Component 1
These two components explain 62.81 % of the point variability.


clusplot(vehicles,cluster_euclidean$Best.partition, color=T,
Use manhattan for distance
#way 2
tot.withinss <- vector(mode="character", length=10)
for (i in 1:10){
kc <- kmeans(vehicles_scaled, center=i, nstart=20)
tot.withinss[i] <- kc$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
vehicles_saab 217 obs. of 20 variables
vehicles_scaled 846 obs. of 18 variables
vehicles_van 199 obs. of 20 variables
Values
i 10L
k int [1:10] 1 2 3 4 5 6 7 8 9 10
predcol num [1:91] 1.29 1.29 1.29 1.29 1.29 ...
rmse 0.00625734842583192
tot.withinss chr [1:10] "15209.9999999996" "8611.11597589309" "678
> 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
tot.withinss
1:10

K=2 was

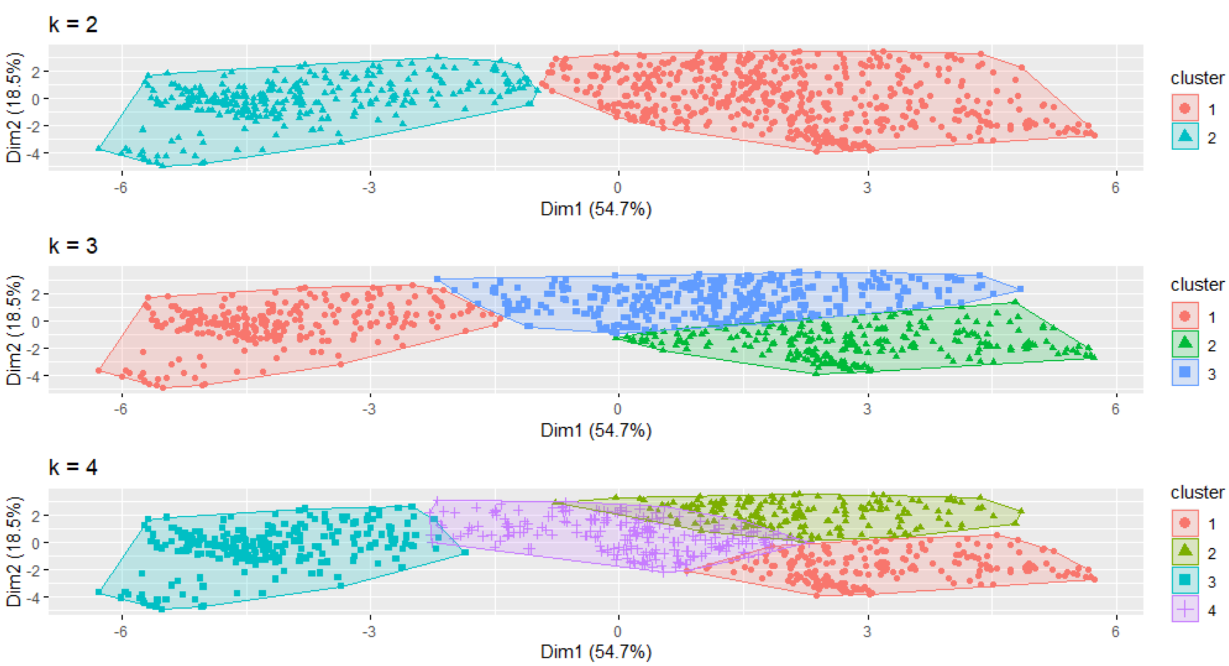
```
Within cluster sum of squares by cluster:
[1] 5852.651 2758.465
 (between_SS / total_SS =  43.4 %)
```

K=3 was

```
[641] 2 2 2 1 2 3

Within cluster sum of squares by cluster:
[1] 2336.638 2668.274 1777.084
 (between_SS / total_SS =  55.4 %)

Available components:
```

K=4 was

```
Within cluster sum of squares by cluster:
[1] 1107.920 1799.132 1148.070 2097.289
 (between_SS / total_SS =  59.6 %)
```



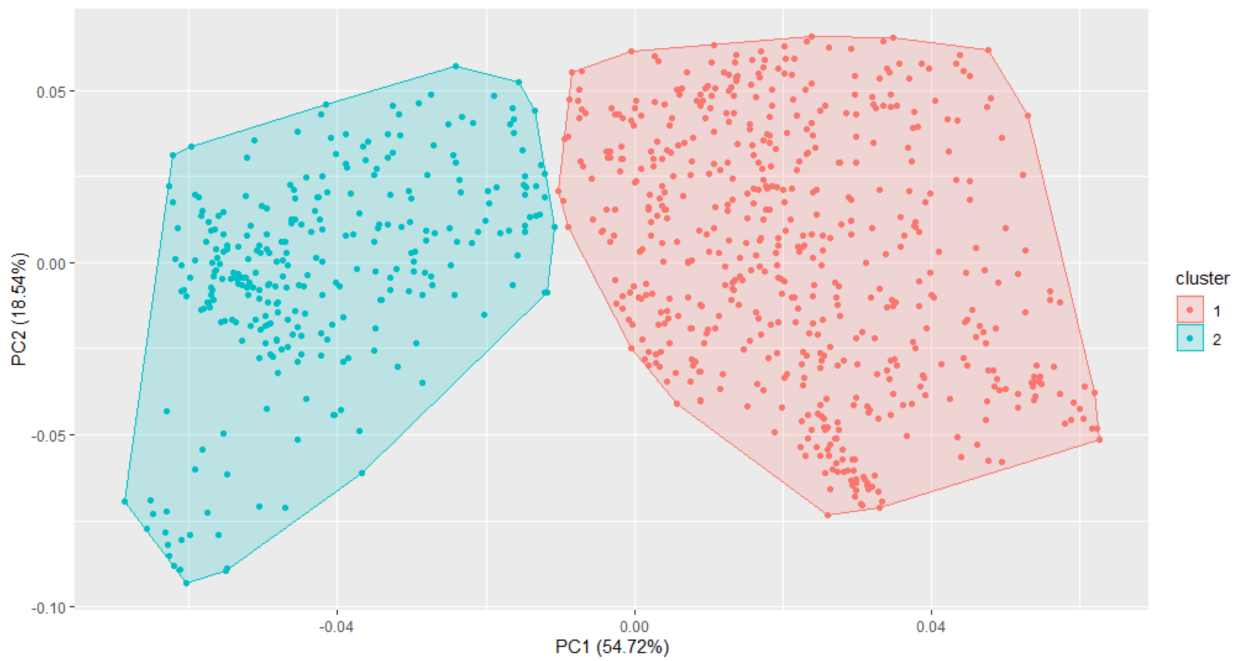
The total within-cluster sum of square (wss) measures the compactness of the clustering and we want it to be as small as possible and we got the minimum within cluster sum of square for k=2

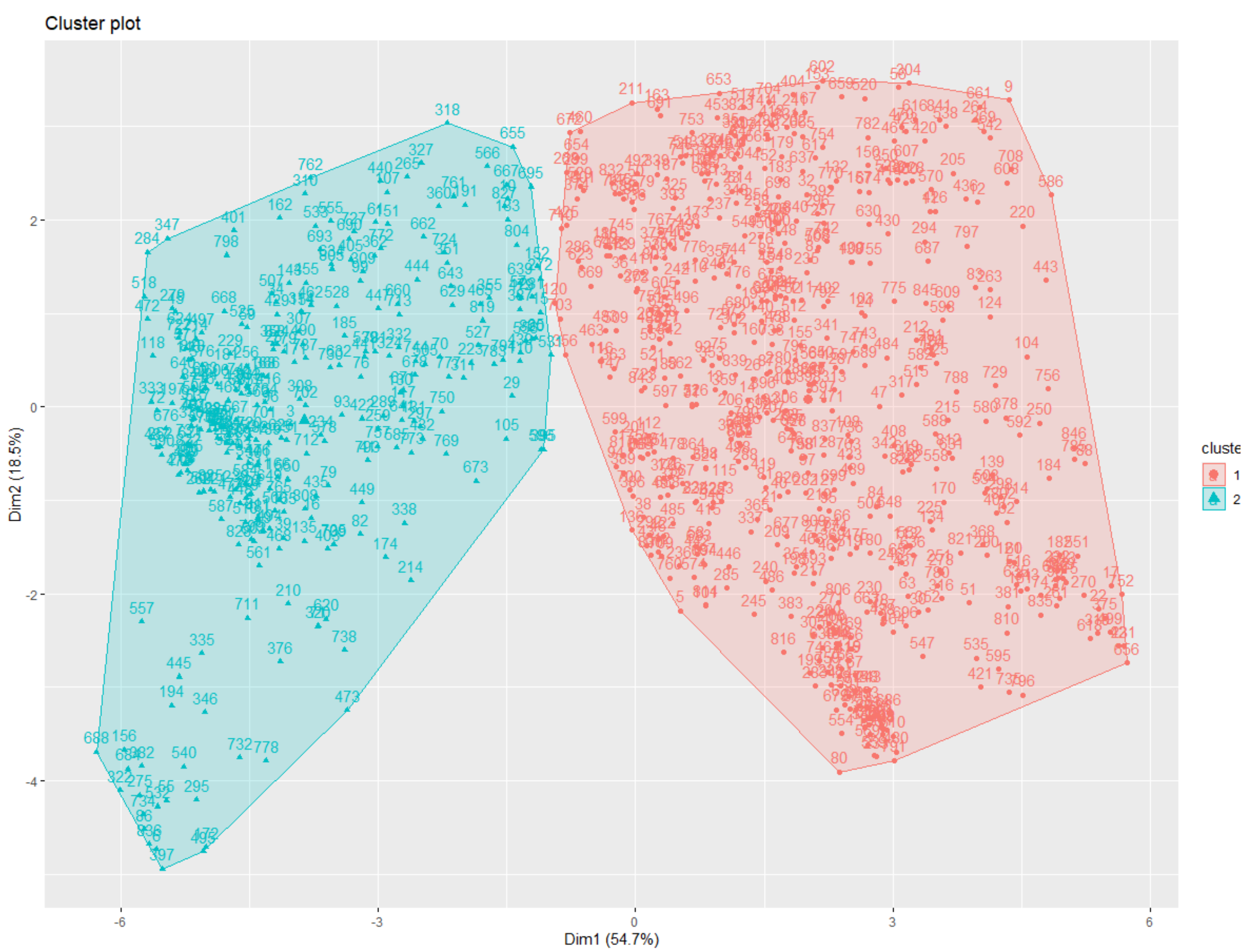
As you can see the graph the optimal number of the cluster is 2

So, the winner case is k= 2

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

```
>
> kc$centers
         comp        circ       d_circ     rad_ra pr_axis_ra   max_l_ra    scat_r
1 -0.970408883 -0.33140009 -0.564450539 -0.3528919  0.5708227 -0.5377661 -0.610604
2 -0.132317021 -1.02240347 -0.132698921  0.3669151  0.4977911 -0.4162736 -0.170397
3 -0.515536167 -1.05493891 -0.924227417 -0.7513881 -0.4076674 -0.3821313 -1.100009
4 -1.173601748 -0.26887061 -0.978511952 -1.3163728 -0.9247992 -0.6376326 -0.593265
5 -0.901189222 -1.12446039 -1.217033026 -1.3345381 -1.0401704 -0.9575805 -1.096738
6  0.698530149  0.55262862  0.852678632  1.1090135  0.5936584  0.2944720  0.700287
7 -0.004128646 -0.05423796 -0.309872178  0.1275424  0.5958652 -0.1790249 -0.449188
```


PC2 (18.54%)
PC1 (54.72%)
cluster
1
2


Cluster plot
Dim2 (18.5%)
Dim1 (54.7%)

Part 2

Discussion of methods used to define the input vector for time series analysis

Time series analysis is classified into two types: univariate and multivariate. Time series forecasting is a type of regression problem in which the input variables are ordered by time.

Univariate

A univariate time series, as the name implies, is one with only one time-dependent variable.

Multivariate

There is more than one time-dependent variable in a multivariate time series. Each variable is dependent not only on its previous values, but also on other variables. This dependency is employed in order to forecast future values.

If we want to forecast today's USD price, wouldn't it be useful to know what the price was yesterday? Similarly, forecasting website traffic would be much easier if we had data from the previous few months or years.

Quantitative forecasting

Quantitative forecasting approaches are focused on historical data analysis, with the assumption that past data trends can be used to predict future data points.

Auto Regressive Integrated Moving Average (ARIMA)

Commonly used forecasting technique is the Autoregressive Integrated Moving Average (ARIMA) model, which combines two or more time series models. For multivariate non-stationary results, this model works well.

Moving average (MA)

The Moving Average (MA) approach is the most straightforward and fundamental of all time series forecasting techniques. For a univariate (one variable) time series, this model is used. The output (or future) variable in an MA model is presumed to have a linear relationship with the current and previous values. As a result, the new series is built by averaging the previous values.

1. **Time based feature**

   if the time stamp is available For example, we can figure out what time of day the data was collected and compare patterns between business and non-business hours. We can draw more informed conclusions about the data if we can remove the ‘hour' function from the time stamp.

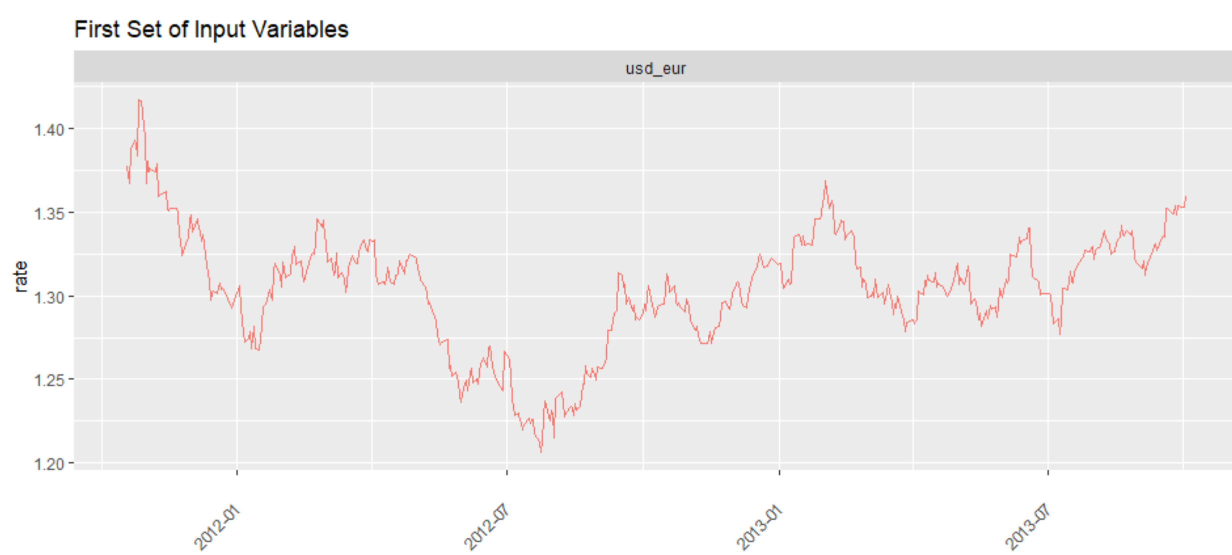
2. Leg feature

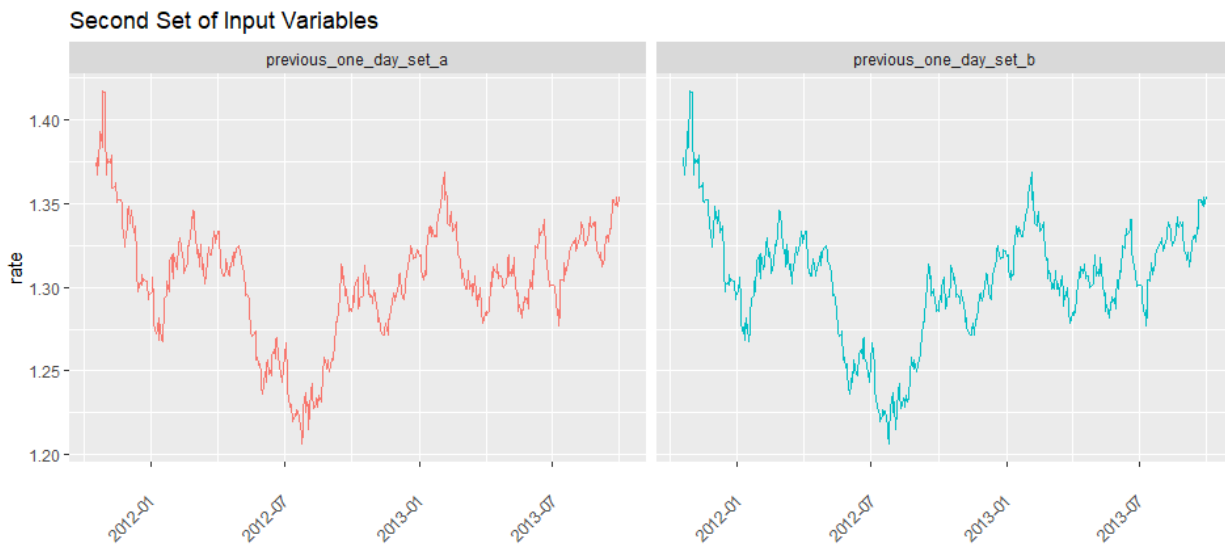
The lag value we select will be determined by the relationship between individual values and their previous values. Assume you're forecasting a company's stock price. So, in order to make a forecast, the previous day's stock price is crucial, right? To put it another way, the value at a given point in time *t* is greatly affected by the value at time *t-1*. The past values are known as lags, so *t-1* is lag *1, t-2* is lag *2*, and so on.

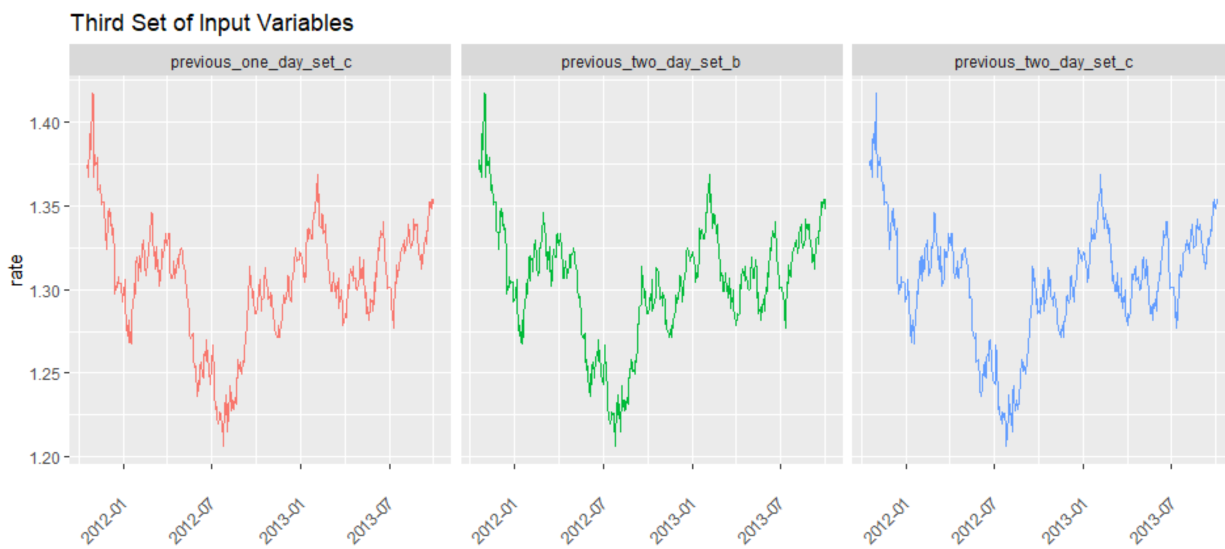
3. Rolling window method

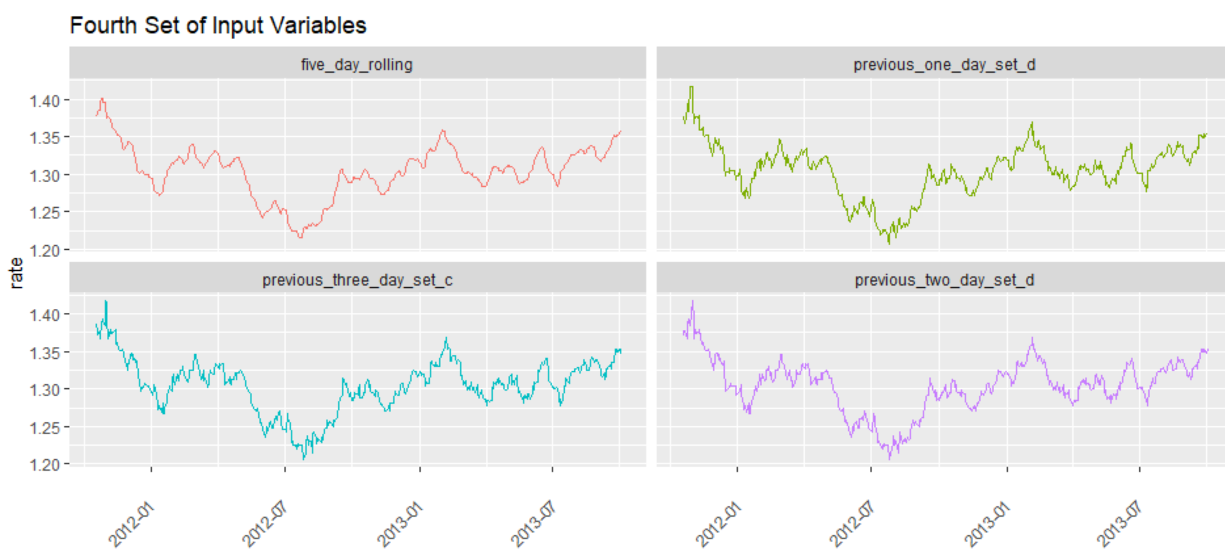
Since the window will be different with each data point, this approach is known as the rolling window method. The features generated using this method are known as the 'rolling window' features because they resemble a window that is sliding with each subsequent stage. We'll choose a window size, average the values in the window, and use the average as a function.

```
#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
usd_exchange_full = ExchangeUSD %>%
  mutate(previous_one_day_set_a = lag(ExchangeUSD$usd_eur,1),
         previous_one_day_set_b = lag(ExchangeUSD$usd_eur,1),
         previous_two_day_set_b = lag(ExchangeUSD$usd_eur,2),
         previous_one_day_set_c = lag(ExchangeUSD$usd_eur,1),
         previous_two_day_set_c = lag(ExchangeUSD$usd_eur,2),
         previous_three_day_set_c = lag(ExchangeUSD$usd_eur,3),
         previous_one_day_set_d = lag(ExchangeUSD$usd_eur,1),
         previous_two_day_set_d = lag(ExchangeUSD$usd_eur,2),
         five_day_rolling = rollmean(usd_eur,5, fill = NA),
         ten_day_rolling = rollmean(usd_eur,10, fill = NA)) %>%


  drop_na()
```




Second Set of Input Variables
previous_one_day_set_a
previous_one_day_set_b
rate
1.40
1.35
1.30
1.25
1.20
2012-01
2012-07
2013-01
2013-07


Third Set of Input Variables
previous_one_day_set_c
previous_two_day_set_b
previous_two_day_set_c
rate
1.40
1.35
1.30
1.25
1.20
2012-01
2012-07
2013-01
2013-07


Fourth Set of Input Variables
five_day_rolling
previous_one_day_set_d
previous_three_day_set_c
previous_two_day_set_d
rate
1.40
1.35
1.30
1.25
1.20
2012-01
2012-07
2013-01
2013-07

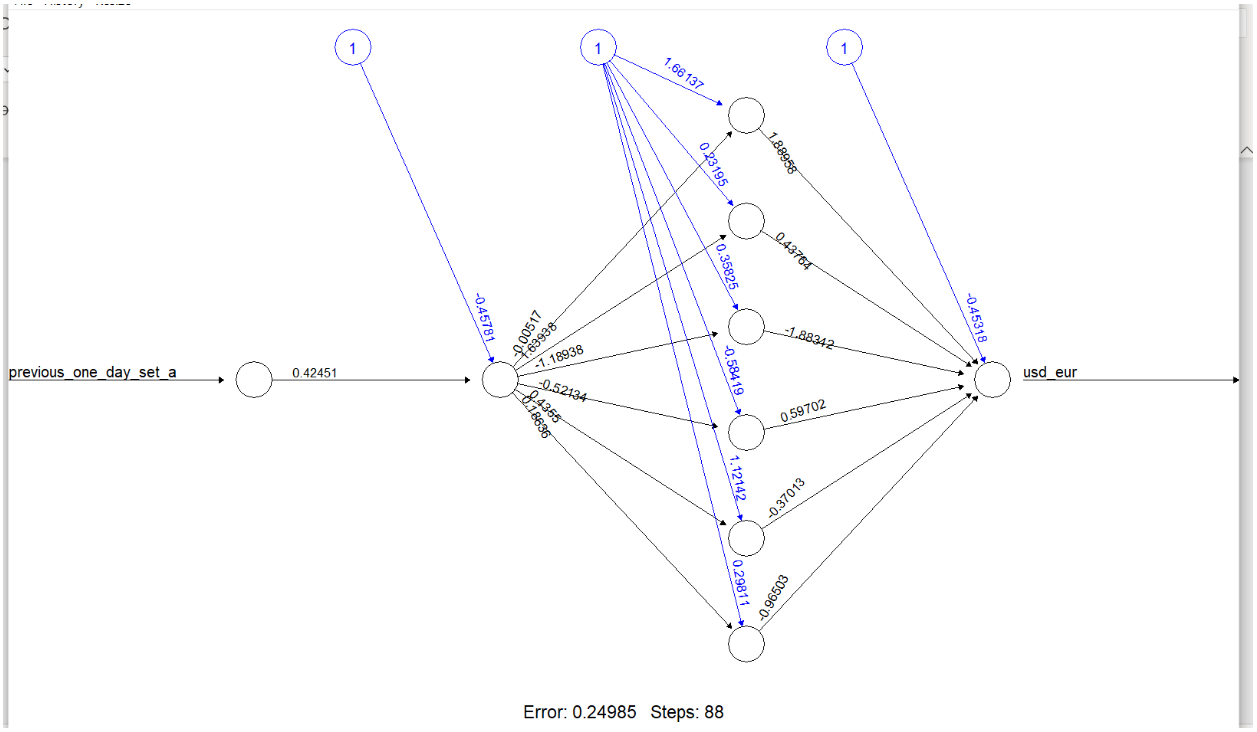
Min max normalization

Normalization is the process of rescaling data from its original range such that all values are between 0 and 1. Normalization necessitates knowing or being able to measure the minimum and maximum measurable values with accuracy. As the scale of the target variable is reduced, the size of the gradient used to change the weights is reduced, resulting in a more robust model and training phase. When fitting a model for regression with a widely distributed target variable, data scaling will help stabilize the training process. Scaling the input variables can also help to improve the model's stability and efficiency. For each of the features, we'll use a different scale. For datasets with multiple targets, the same considerations apply. This situation could lead to more impact in the final results for some of the inputs, with an imbalance caused by their original measurement scales rather than the data's intrinsic nature. This type of problem is avoided by normalizing all features in the same range.

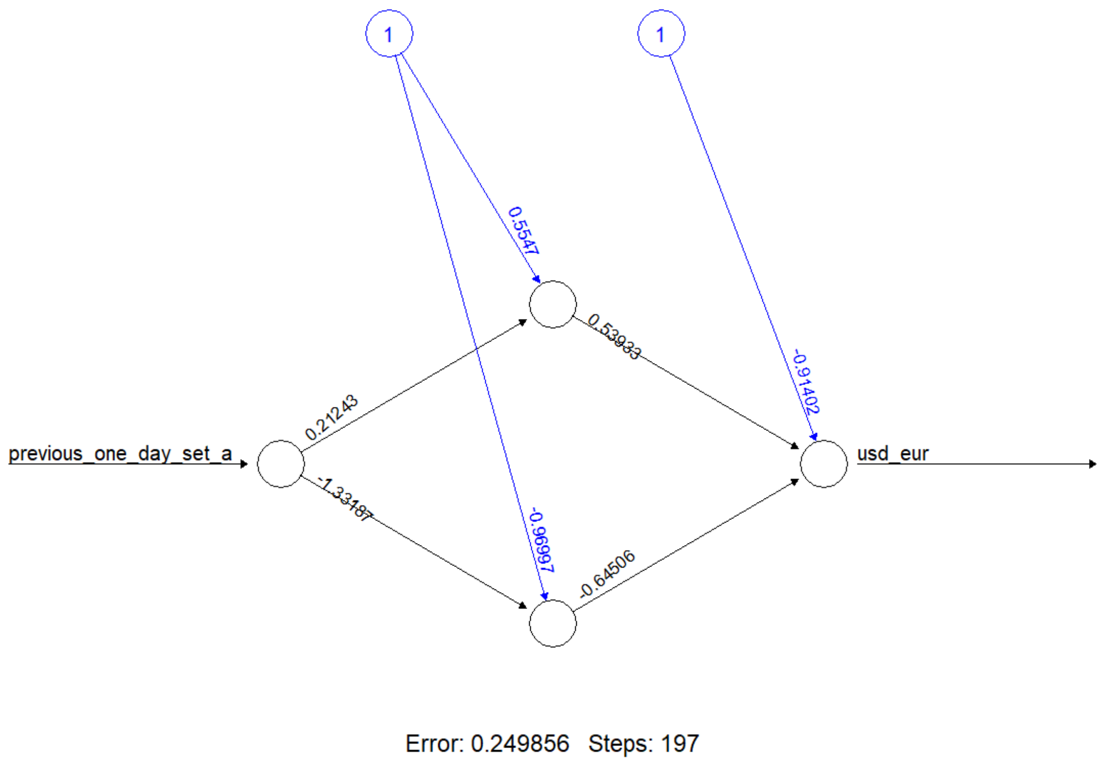
Various NN

```
> # save the stat indices to a dataframe
> set_a_models_two_layers = results_two_hidden_layers %>%
+   select(-estimator) %>%
+   pivot_wider(names_from = metric, values_from = estimate) %>%
+   arrange(rmse)
> kable(set_a_models_two_layers[1:10,])


|type              |hiddel_layers |input_set |      rmse|       mae|      mape|
|:-----------------|:-------------|:---------|---------:|---------:|---------:|
|Two Hidden Layers |7 and 5       |A         | 0.0061760| 0.0045661| 0.3458930|
|Two Hidden Layers |2 and 5       |A         | 0.0061767| 0.0045613| 0.3455457|
|Two Hidden Layers |9 and 3       |A         | 0.0061775| 0.0045637| 0.3457219|
|Two Hidden Layers |2 and 4       |A         | 0.0061788| 0.0045729| 0.3463951|
|Two Hidden Layers |5 and 1       |A         | 0.0061794| 0.0045508| 0.3447456|
|Two Hidden Layers |4 and 5       |A         | 0.0061796| 0.0045740| 0.3464743|
|Two Hidden Layers |4 and 1       |A         | 0.0061797| 0.0045708| 0.3462586|
|Two Hidden Layers |3 and 4       |A         | 0.0061813| 0.0045916| 0.3477651|
|Two Hidden Layers |9 and 5       |A         | 0.0061820| 0.0045419| 0.3441292|
|Two Hidden Layers |9 and 4       |A         | 0.0061824| 0.0045518| 0.3448450|
>
```

Graphical Structure of NN



```
> ann$result.matrix
                                            [,1]
error                                0.249849939
reached.threshold                    0.006795260
steps                               88.000000000
Intercept.to.1layhid1               -0.457810421
previous_one_day_set_a.to.1layhid1   0.424507554
Intercept.to.2layhid1                1.661373539
1layhid1.to.2layhid1                -0.005173381
Intercept.to.2layhid2                0.231952961
1layhid1.to.2layhid2                 1.639383215
Intercept.to.2layhid3                0.358250981
1layhid1.to.2layhid3                -1.189379463
Intercept.to.2layhid4               -0.584187627
1layhid1.to.2layhid4                -0.521343742
Intercept.to.2layhid5                1.121416572
1layhid1.to.2layhid5                 0.435495599
Intercept.to.2layhid6                0.298106225
1layhid1.to.2layhid6                 0.186364488
Intercept.to.usd_eur                -0.453175909
2layhid1.to.usd_eur                  1.889578362
2layhid2.to.usd_eur                  0.437638958
2layhid3.to.usd_eur                 -1.883421799
2layhid4.to.usd_eur                  0.597018090
2layhid5.to.usd_eur                 -0.370126182
2layhid6.to.usd_eur                 -0.965033569
```




```
> ann3$result.matrix
                                          [,1]
error                              0.249855580
reached.threshold                  0.007786183
steps                            197.000000000
Intercept.to.1layhid1              0.554700933
previous_one_day_set_a.to.1layhid1 0.212427066
Intercept.to.1layhid2             -0.969967077
previous_one_day_set_a.to.1layhid2 -1.331870783
Intercept.to.usd_eur              -0.914017463
1layhid1.to.usd_eur                0.539328290
1layhid2.to.usd_eur               -0.645055474
```

Statistical indices

```
# A tibble: 4 x 4
  .metric .estimator .estimate type
  <chr>   <chr>          <dbl> <chr>
1 rmse    standard     0.00626 Two Hidden Layers
2 rsq     standard     0.891   Two Hidden Layers
3 mae     standard     0.00470 Two Hidden Layers
4 mape    standard     0.356   Two Hidden Layers
>
>
```

RMSE

The root mean square error is a widely used measure of the difference between a model's predicted value and the actual value seen in the data. It compares the forecasting errors of different models for a specific variable.

R – squared

It is a statistical indicator of how close the data are to the fitted regression line. It is also known as the coefficient of predicted value determination.

MSE

Mean squared error

It's also a crucial loss function for algorithms that fit or optimize regression problems using the least squares framework. In this context, "least squares" refers to minimizing the mean squared error between predictions and expected values. The mean or average of the squared differences between predicted and expected target values in a dataset is used to calculate the MSE.

MAE

The MAE score is determined by taking the average of the absolute error values. Absolute, also known as abs(), is a mathematical function that simply turns a negative number into a positive one. As a result, the difference between an expected and predicted value can be either positive or negative, but it must be positive when calculating the MAE.

MAPE

The most commonly used metric for assessing forecast accuracy is mean absolute percentage error. It falls under the category of scale-independent percentage errors and can be used to compare series on different scales.

Predicted output

	x
1	1.293867
2	1.292015
3	1.293575
4	1.287143
5	1.294257
6	1.303904
7	1.298544
8	1.309263
9	1.307022
10	1.308386
11	1.323783
12	1.323393
13	1.322613
14	1.326901
15	1.334307
16	1.330409
17	1.331871
18	1.332943
19	1.339471
20	1.339082
21	1.320664
22	1.310822
23	1.308581
24	1.307607
25	1.300298
26	1.301370
27	1.300785
28	1.301370
29	1.300298
30	1.291723
31	1.283343
32	1.286461
33	1.277788

Showing 1 to 34 of 91 ent

Console

	x
34	1.284804
35	1.304294
36	1.304391
37	1.303611
38	1.313746
39	1.308971
40	1.307217
41	1.313551
42	1.318521
43	1.320372
44	1.322029
45	1.322613
46	1.326121
47	1.325342
48	1.325732
49	1.328363
50	1.320859
51	1.326024
52	1.328460
53	1.331676
54	1.337912
55	1.332845
56	1.330312
57	1.323880
58	1.324660
59	1.325439
60	1.331091
61	1.333722
62	1.341128
63	1.335671
64	1.334989
65	1.337912
66	1.335671

Showing 33 to 66 of 91 entr

64	1.334989
65	1.337912
66	1.335671
67	1.337522
68	1.331773
69	1.321834
70	1.318910
71	1.315695
72	1.320469
73	1.311407
74	1.315987
75	1.325049
76	1.325244
77	1.329045
78	1.330409
79	1.326609
80	1.333820
81	1.334502
82	1.333917
83	1.351067
84	1.350775
85	1.347754
86	1.347462
87	1.351944
88	1.346975
89	1.352139
90	1.351847
91	1.351847

Desired output

	usd_eur
1	1.2920
2	1.2936
3	1.2870
4	1.2943
5	1.3042
6	1.2987
7	1.3097
8	1.3074
9	1.3088
10	1.3246
11	1.3242
12	1.3234
13	1.3278
14	1.3354
15	1.3314
16	1.3329
17	1.3340
18	1.3407
19	1.3403
20	1.3214
21	1.3113
22	1.3090
23	1.3080
24	1.3005
25	1.3016
26	1.3010
27	1.3016
28	1.3005
29	1.2917
30	1.2831
31	1.2863
32	1.2774
33	1.2846

Showing 1 to 34 of 91 entrie

	usd_eur
34	1.3046
35	1.3047
36	1.3039
37	1.3143
38	1.3094
39	1.3076
40	1.3141
41	1.3192
42	1.3211
43	1.3228
44	1.3234
45	1.3270
46	1.3262
47	1.3266
48	1.3293
49	1.3216
50	1.3269
51	1.3294
52	1.3327
53	1.3391
54	1.3339
55	1.3313
56	1.3247
57	1.3255
58	1.3263
59	1.3321
60	1.3348
61	1.3424
62	1.3368
63	1.3361
64	1.3391
65	1.3368
66	1.3387

Showing 33 to 66 of 91 en

	usd_eur
60	1.3348
61	1.3424
62	1.3368
63	1.3361
64	1.3391
65	1.3368
66	1.3387
67	1.3328
68	1.3226
69	1.3196
70	1.3163
71	1.3212
72	1.3119
73	1.3166
74	1.3259
75	1.3261
76	1.3300
77	1.3314
78	1.3275
79	1.3349
80	1.3356
81	1.3350
82	1.3526
83	1.3523
84	1.3492
85	1.3489
86	1.3535
87	1.3484
88	1.3537
89	1.3534
90	1.3534
91	1.3592

Appendix

Part 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("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: opel")
# 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)$tot.withinss})
 plot(k, WSS, type="l", xlab= "Number of k", ylab="Within sum of squares")
clusplot(vehicles,cluster_euclidean$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)
```

Part 2

```
knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
library(readxl)
library(lubridate)
library(zoo)
library(tidymodels)
library(readxl)
library(neuralnet)
library(knitr)
library(tseries)
library(xts)
library(funtimes)
library(dplyr)
library(MLmetrics)
ExchangeUSD <- read_excel("ExchangeUSD.xlsx") %>%
  janitor::clean_names() %>%
  mutate(date_in_ymd = ymd(yyyy_mm_dd)) %>%
  select(-1) %>%
  select(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
usd_exchange_full = ExchangeUSD %>%
  mutate(previous_one_day_set_a = lag(ExchangeUSD$usd_eur,1),
         previous_one_day_set_b = lag(ExchangeUSD$usd_eur,1),
         previous_two_day_set_b = lag(ExchangeUSD$usd_eur,2),
```

```
        previous_one_day_set_c = lag(ExchangeUSD$usd_eur,1),
        previous_two_day_set_c = lag(ExchangeUSD$usd_eur,2),
        previous_three_day_set_c = lag(ExchangeUSD$usd_eur,3),
        previous_one_day_set_d = lag(ExchangeUSD$usd_eur,1),
        previous_two_day_set_d = lag(ExchangeUSD$usd_eur,2),
        five_day_rolling = rollmean(usd_eur,5, fill = NA),
        ten_day_rolling = rollmean(usd_eur,10, fill = NA)) %>%


 drop_na()

usd_exchange_full %>%
 pivot_longer(cols = 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 = "First Set of Input Variables") +
 theme(legend.position = "none")


usd_exchange_full %>%
 pivot_longer(cols = c(4,5),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 = "Second Set of Input Variables") +
  theme(legend.position = "none")

usd_exchange_full %>%
  pivot_longer(cols = 6:8,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 = "Third Set of Input Variables") +
  theme(legend.position = "none")

usd_exchange_full %>%
  pivot_longer(cols = 9:12,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 = "Fourth Set of Input Variables") +
  theme(legend.position = "none")

# We can create a function to normalize the data from 0 to 1
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x))) }
# All the variables are normalized
normalized_usd = usd_exchange_full %>%
  mutate(across(3:12, ~normalize(.x)))
```

```
# Look at the data that has been normalized
summary(normalized_usd)

set.seed(123)
usd_train <- normalized_usd[1:400,]
usd_test <- normalized_usd[401:491,]

# We can create a function to unnormalize the data=
unnormalize <- function(x, min, max) {
  return( (max - min)*x + min ) }
# Get the min and max of the original training values
usd_min_train <- min(usd_exchange_full[1:400,3])
usd_max_train <- max(usd_exchange_full[1:400,3])
# Get the min and max of the original testing values
usd_min_test <- min(usd_exchange_full[401:491,3])
usd_max_test <- max(usd_exchange_full[401:491,3])
# Check the range of the min and max of the training dataset
usd_min_test

usd_min_train

usd_max_test
usd_max_train
relevant_pred_stat <- function(true_value, predicted_value, model_kind) {
  rbind((tibble(truth = true_value,
               prediction = predicted_value) %>%
          metrics(truth,prediction) %>%
          mutate(type = model_kind)),(tibble(truth = true_value,
                                            prediction = predicted_value) %>%
```

```
                    mape(truth,prediction) %>%
                    mutate(type = model_kind)))
}
set.seed(12345)
# function setup that creates 2 layer model
model_two_hidden_layers = function(hidden,sec_hidden) {
  nn_model_true = neuralnet(usd_eur ~ previous_one_day_set_a, data=usd_train, hidden=c(
    hidden,sec_hidden), linear.output=TRUE)
  train_results = compute(nn_model_true,usd_test[,3:4])
  truthcol = usd_exchange_full[401:491,3]$usd_eur
  predcol = unnormalize(train_results$net.result,usd_min_train, usd_max_train)[,1]
  relevant_pred_stat(truthcol,predcol,
          "Two Hidden Layers") %>%
    mutate(hiddel_layers = paste0(hidden, " and ",sec_hidden),
       input_set = "A") %>%
    filter(.metric != "rsq")
}
# creation of different models with varying number of nodes
results_two_hidden_layers = bind_rows(
  lapply(1:10, function(n) {
    bind_rows(
      lapply(1:5, function(m) {
        model_two_hidden_layers(n,m)
      })
    )
  })) %>%
  janitor::clean_names()
# save the stat indices to a dataframe
set_a_models_two_layers = results_two_hidden_layers %>%
```

```
  select(-estimator) %>%
  pivot_wider(names_from = metric, values_from = estimate) %>%
  arrange(rmse)
kable(set_a_models_two_layers[1:10,])
# Combine the dataframes
set_a_models = rbind(set_a_models_two_layers[1:2,],set_a_models_two_layers)
##################################testing2##################################
# We can create a function to normalize the data from 0 to 1
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x))) }
# All the variables are normalized
normalized_usd = usd_exchange_full %>%
  mutate(across(3:12, ~normalize(.x)))
# Look at the data that has been normalized
summary(normalized_usd)

set.seed(123)
usd_train <- normalized_usd[1:400,]
usd_test <- normalized_usd[401:491,]

# We can create a function to unnormalize the data=
unnormalize <- function(x, min, max) {
  return( (max - min)*x + min ) }
# Get the min and max of the original training values
usd_min_train <- min(usd_exchange_full[1:400,3])
usd_max_train <- max(usd_exchange_full[1:400,3])
# Get the min and max of the original testing values
usd_min_test <- min(usd_exchange_full[401:491,3])
usd_max_test <- max(usd_exchange_full[401:491,3])
```

```
# Check the range of the min and max of the training dataset
usd_min_test


usd_min_train


usd_max_test
usd_max_train




data <- read_excel('ExchangeUSD.xlsx')%>%
  janitor::clean_names() %>%
  mutate(date_in_ymd = ymd(yyyy_mm_dd)) %>%
  select(-1) %>%
  select(date_in_ymd,everything())
spy <- xts(data[,3],order.by = as.Date(data$date_in_ymd) )


rspy <- dailyReturn(spy)
rspy1 <-stats::lag(rspy,k=1)
rspyall <- cbind(rspy,rspy1)
colnames(rspyall) <- c('rspy','rspy1')
rspyall <- na.exclude(rspyall)


rspyt = window(rspyall,end = '2013-03-11')
rspyf =window(rspyall, start ='2013-03-11' ,end = '2013-10-02' )



ann = neuralnet(usd_eur~previous_one_day_set_a,data = usd_train,hidden = c(1,6),act.fct =
function(x){x})
```

```
ann$result.matrix

plot(ann)


ann1 = neuralnet(rspy~rspy1,data = rspyt,hidden = 2,act.fct = function(x){x})

ann1$result.matrix

plot(ann1)




ann3 = neuralnet(usd_eur~previous_one_day_set_a,data = gbp_train,hidden =2,act.fct =
function(x){x})

ann3$result.matrix

plot(ann3)


preds=compute(ann,gbp_test)

view(preds$net.result)

truthcol=gbp_exchange_full[401:491,3]

predcol = unnormalize(preds$net.result,gbp_min_train, gbp_max_train)[,1]



view(predcol)

view(truthcol)


rmse=sqrt(mean((predcol-truthcol$usd_eur)^2))

rmse

summary(ann$call)

table(truthcol$usd_eur[26],predcol[26])


relevant_pred_stat(truthcol$usd_eur,predcol,
```

```
        "Two Hidden Layers")



preds1=compute(ann1,rspyf_1)

view(preds1$net.result)

truthcol1=gbp_exchange_full[401:491,3]

predcol1 = unnormalize(preds1$net.result,gbp_min_train, gbp_max_train)[,1]

view(predcol1)

relevant_pred_stat(truthcol1$usd_eur,predcol1,

        "Two Hidden Layers")



preds2=predict(ann3,newdata =  gbp_test)

view(preds2$net.result)

truthcol2=gbp_exchange_full[401:491,3]

predcol2 = unnormalize(preds2$net.result,gbp_min_train, gbp_max_train)[,1]

view(predcol2)

relevant_pred_stat(truthcol2$usd_eur,predcol2,

        "Two Hidden Layers")
```

References

https://www.youtube.com/watch?v=eDcPolK_j8E