UNIVERSITY OF WESTMINSTER

University of Westminster

IIT Sri Lanka

Machine Learning & Data Mining

Module Leader:- Mr. Nipuna Senanayake

INDIVIDUAL COURSEWORK REPORT

Clustering & Neural Network Analysis

P. G. Malik Kumara

Table of contents

Part A- Sub Task 1

● This dataset consists of 2700 variants. This dataset is characterized by 12 different properties. ● Before delving into k-means clustering, I started the preprocessing phase.

1. Preprocessing has 4 sections , Data cleaning, Data transformation, Data integration, Data reduction.
    - In my analysis I performed data transformations, which included scaling and outlier removal.
        - My first step was to remove the quality column from the data, thus Clustering is an unsupervised scheme.
        - After that, I began the task of identifying and removing outliers.
    - Outlier removal , as I research on the subject there are 3 main methods for identifying outliers.
        1. Z-Score Method - Measure data with standard deviations is from the mean, then for a given score value(threshold), identify the outliers.
        2. IQR (Interquartile Range) Method - Represent range between the 25th and 75th percentiles (Q1 and Q3) of the data. Then calculate IQR(Q3 - Q1). Define upper and lower bounds. Identify data(outliers) that fall below the lower bound or above the upper bound.
        3. Visualization Methods - Box plots, scatter plots, and histograms can visually reveal outliers.Using the box plot's statistical properties can identify outliers.
    - In my analysis I have used both IQR method and visualization method. Both methods have the same answer.

```
19  # -------Outlier removal with Interquartile Range method
20  # Define a function to remove outliers using the Interquartile Range method
21  Cw_out_iqr <- function(x) {
22    q1 <- quantile(x, 0.25)
23    q3 <- quantile(x, 0.75)
24    iqr <- q3 - q1
25    lower_bound <- q1 - 1.5 * iqr
26    upper_bound <- q3 + 1.5 * iqr
27    x[which(x < lower_bound | x > upper_bound)] <- NA
28    return(x)
29  }
30
31  # Apply the function to each numerical column in the dataset
32  CW_Winedata <- as.data.frame(apply(CW_Winedata, 2, Cw_out_iqr))
33
34  # Remove rows with NA values
35  newCW_Winedata <- drop_na(CW_Winedata)
36
37
```

Figure 1 : IQR function

- IQR function -
  - q1(first quartile of the data x, which is the value below which 25% of the data fall)
  - q3(third quartile of the data x, which is the value below which 75% of the data fall)
  - Calculate the iqr(q3 - q1) , lower bound , upper bound . (1.5 is constant)
  - Any data point less than the lower bound or greater than the upper bound is considered an outlier. So the function replaces those data values to NA.
  - apply function is used to apply the remove_outliers_iqr function to each column of scaled_data.
  - Then using drop_na() function NA values are removed from the dataset.
  - After that boxplot display as follows

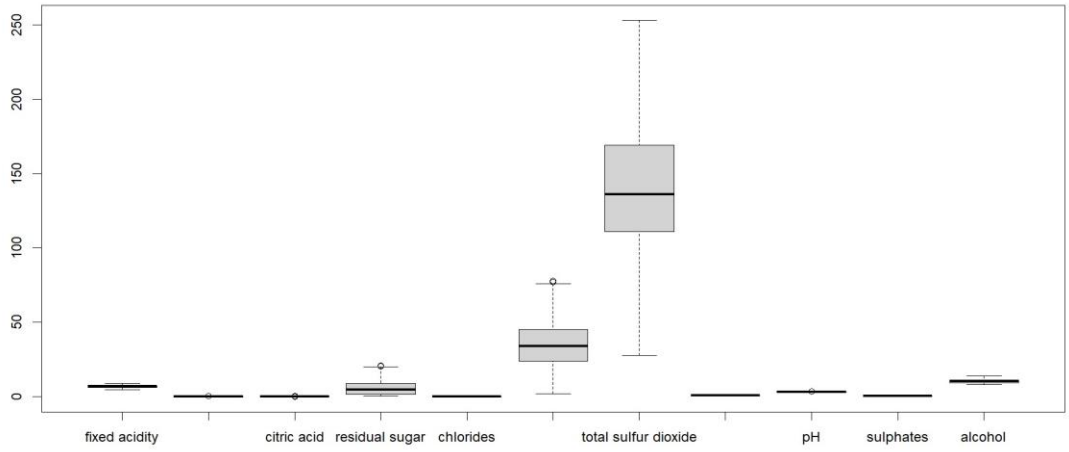

Figure 2 : boxplot after outlier removal with IQR method

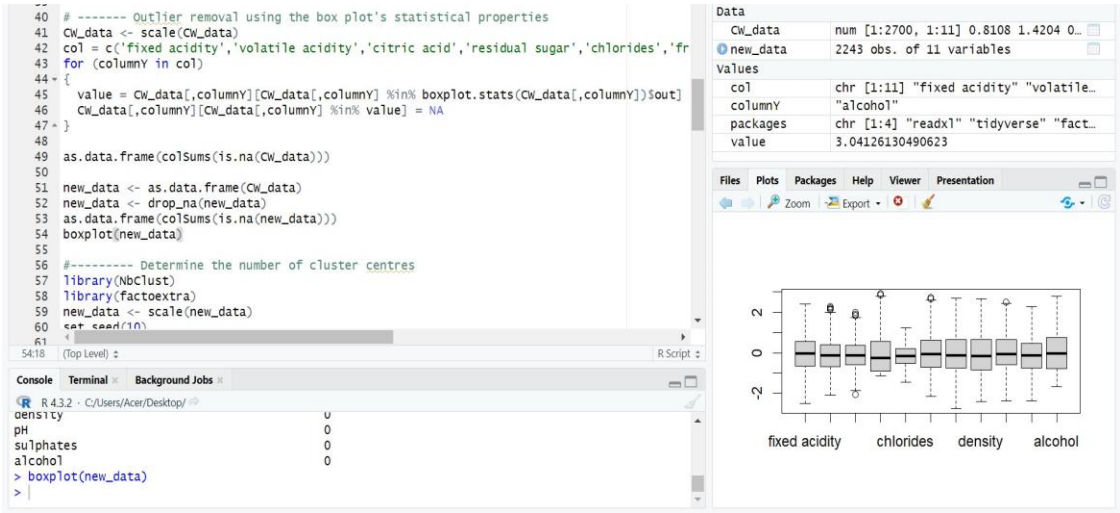

Figure 3 : Outlier Removal Using Box plot Statistical Properties

- Box Plot statistical outlier removal
    - First, I need to **Scale** before using this method.
    - For each column in the dataset, identify the outliers in the current column. It uses the boxplot.stats function.
    - Then replaces the outliers in the current column with NA.
    - After that, using colSums() with is.na() functions counts the number of NA values in each column.

```
> as.data.frame(colSums(is.na(scaled_data)))
                      colSums(is.na(scaled_data))
fixed acidity                                  59
volatile acidity                               94
citric acid                                   176
residual sugar                                  1
chlorides                                     116
free sulfur dioxide                            29
total sulfur dioxide                           10
density                                         0
pH                                             50
sulphates                                     113
alcohol                                         1
>
```

Figure 4 : columns before removing outliers

```
> as.data.frame(colSums(is.na(new_data)))
                      colSums(is.na(new_data))
fixed acidity                                 0
volatile acidity                              0
citric acid                                   0
residual sugar                                0
chlorides                                     0
free sulfur dioxide                           0
total sulfur dioxide                          0
density                                       0
pH                                            0
sulphates                                     0
alcohol                                       0
>
```

Figure 5 : columns after removing outliers

- After removing outliers this is how the box plot shown

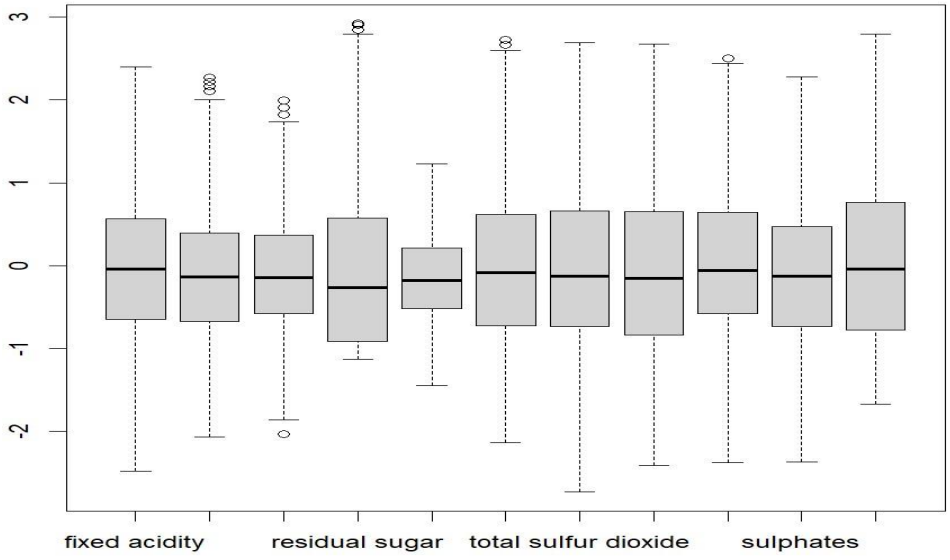

Figure 6 : box plot After removing outliers

- The IQR method needs scaling after removing outliers. Scale to ensure uniformity across all features.Scaling is getting these data into one range. In this case, the data may be in a

scattered range, so it may be difficult in further procedures, so the data should be scaled to one range.

```
57 # scale if used iqr method
58 newCW_Winedata <- scale(newCW_Winedata)
59 boxplot(newCW_Winedata)
60
```

Figure 7 : scaling data

- In this image, the function I use to scale is shown. It is a Z-score function , to achieve z-score standardization, R has built-in scale() function. ● After scaling the dataset, display as follows.

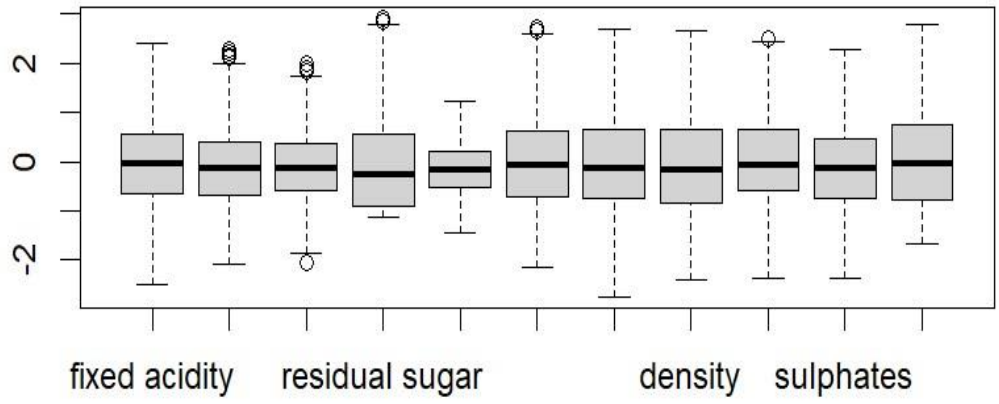

Figure 8 : After scaling boxplot before removing Outliers

- Two methods show the same plot.Thus confirming that outliers are removed.

2. K-means clustering
   - Now that outliers have been removed, the next step in k-means is to determine the number of clusters in this data set. For that I use the 4 tools mentioned in the cw description (nbclust, elbow, gap statistics and silhouette methods).
     - Elbow method -
       - process : optimal number of clusters determined by looking at how much variation occurs in data points when clustering together.
       - Plot displays as follows and from the plot there is a significant drop point , that point is the optimal cluster number.

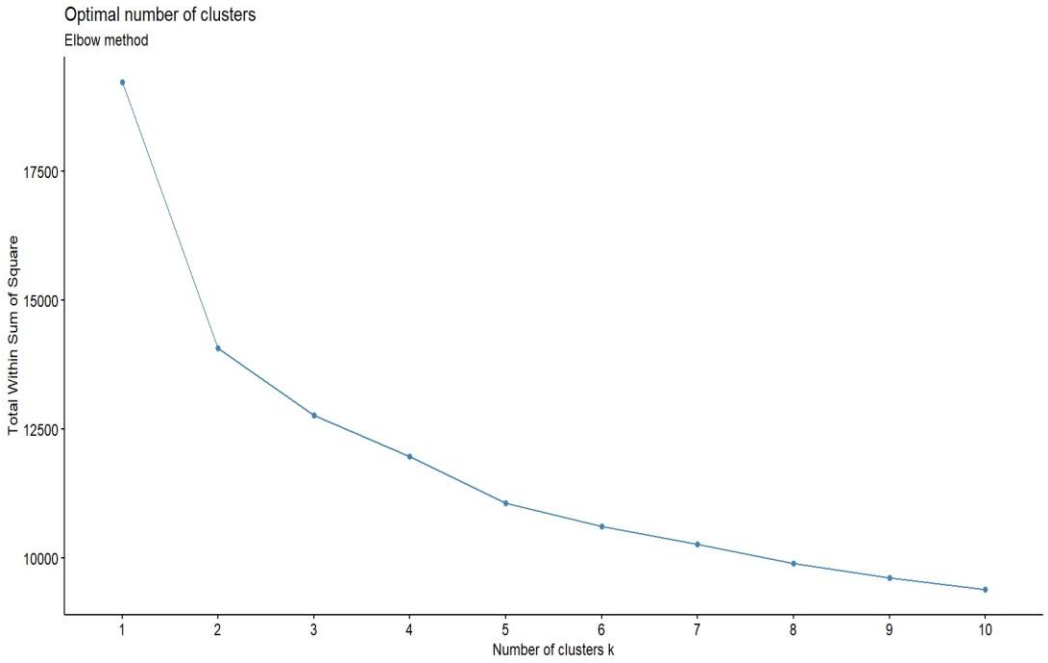

Figure 9 : Elbow method

- This shows that 2 is optimal for the number of clusters.

- Silhouette method -
  - Process: Clustering by calculating the average distance between clusters, higher silhouette score gives better defined clusters.
  - Plot displays as -

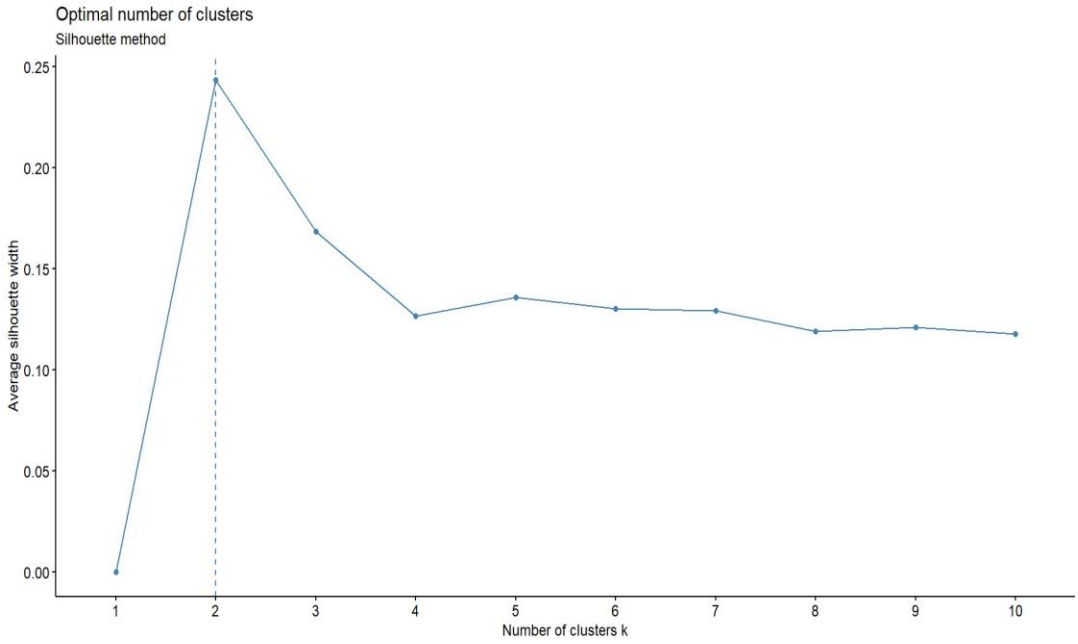

Figure 10 : Silhouette method

  - According to the highest value 2 should be optimal for the number of clusters.

○ Gap-statistics method -

■ Estimate the number of clusters in a dataset by comparing the total intra-cluster variation for different values of k with their expected values under zero-mean reference distribution of the data.

■ Plot displays as -

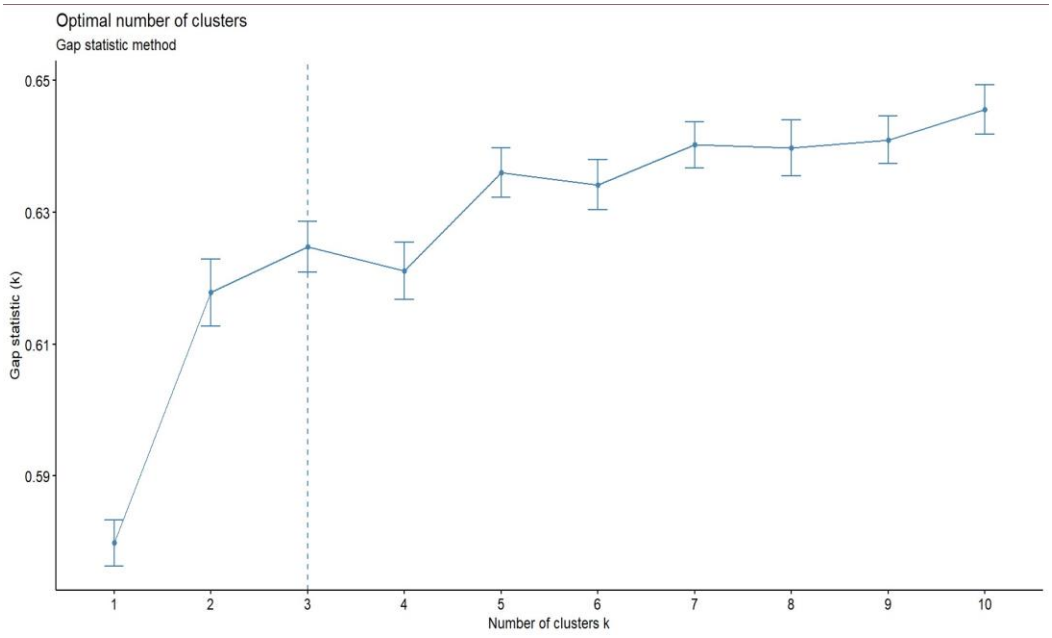

Figure 11 : Gap-statistic method

○ NBclust method -

■ This process calculates clustering indices for multiple partitioning solutions, recommending the optimal number of clusters based on the variability of data point distances across all possible combinations.

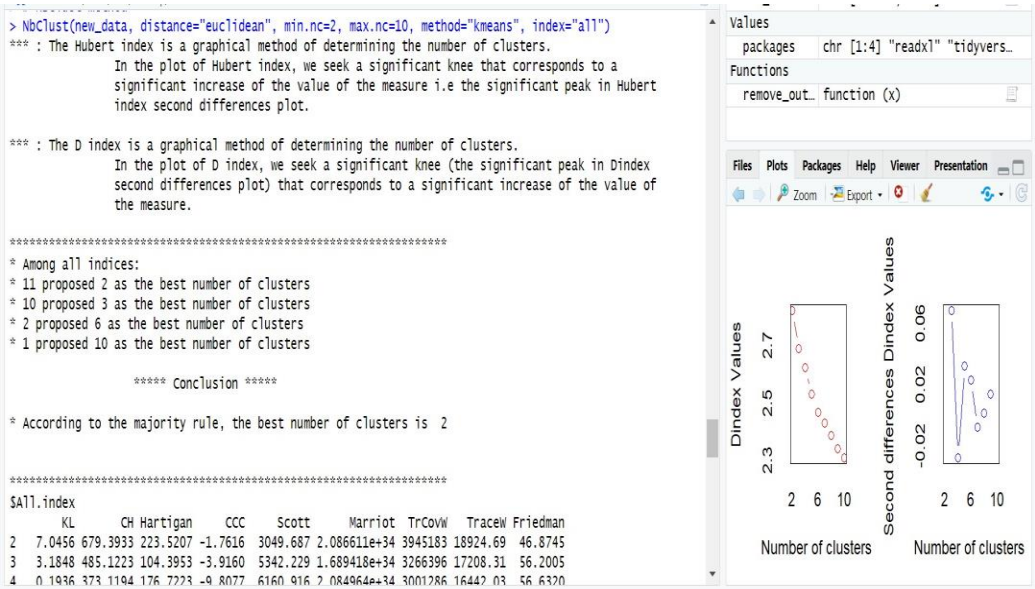

Figure 12 : NBclust method with euclidean distance

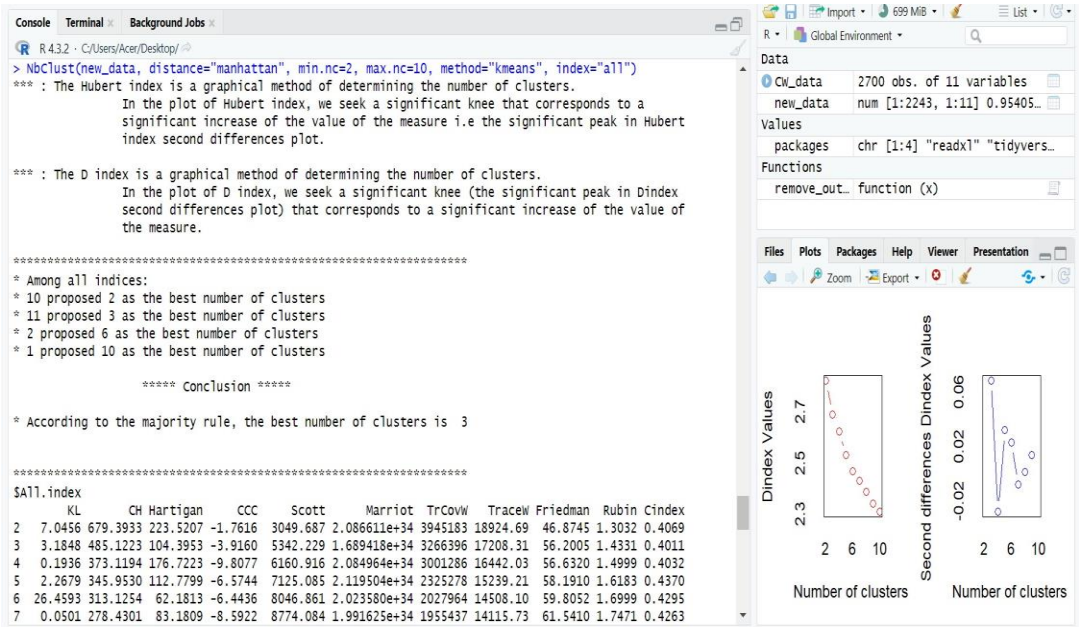

Figure 13 : NBclust method with manhattan distance

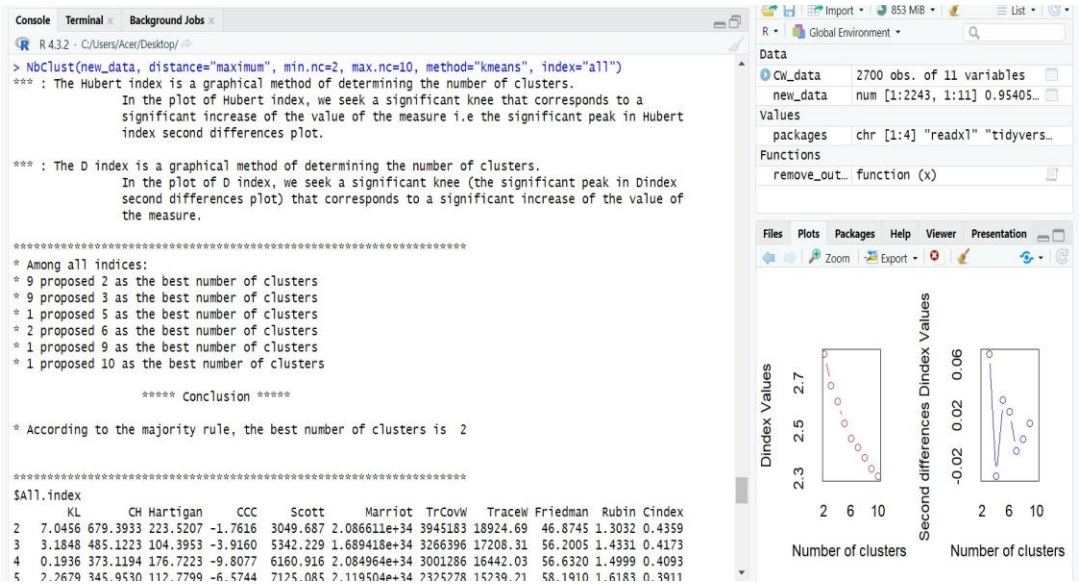

Figure 14 : NBclust method with maximum distance

- From this analysis and the majority of results, it appears that the most suitable value for the number of clusters is 2.

- After identifying the number of clusters, next is to cluster them in k-means. This is how the dataset shows after dividing into 2 clusters.

Figure 15 : Kmeans cluster plot for two clusters

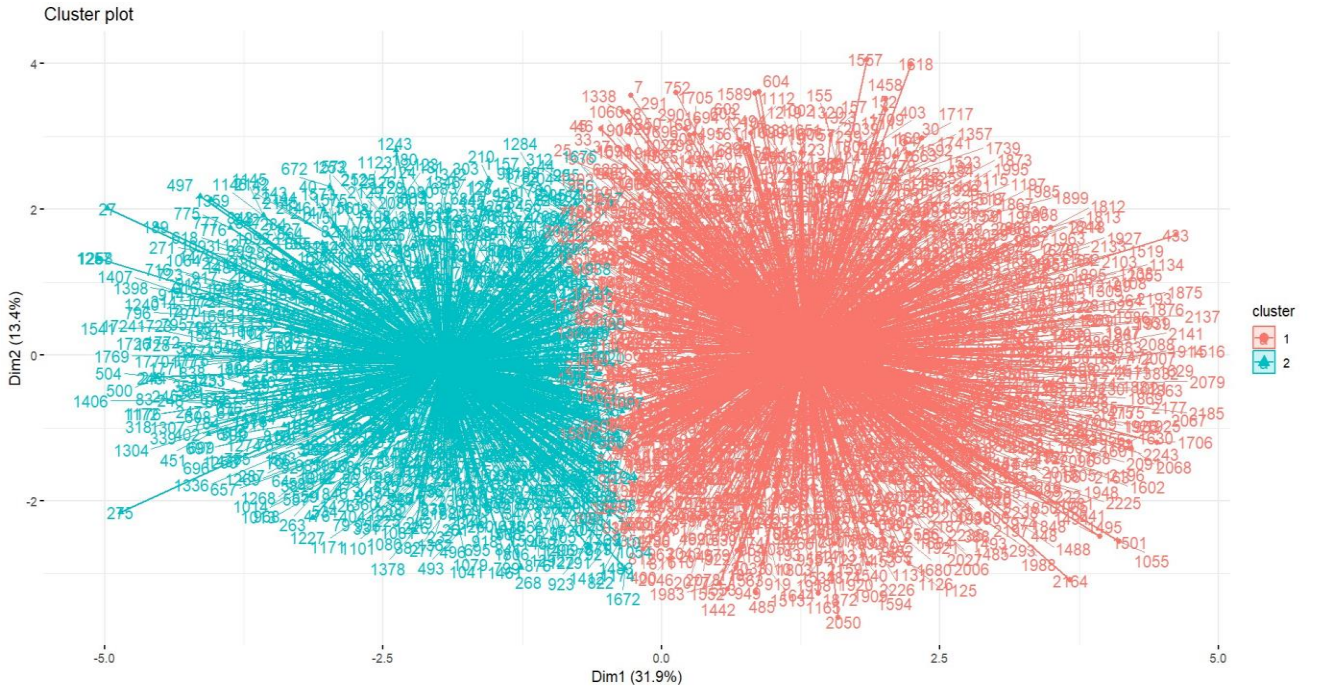

Figure 16 : Kmeans cluster plot for two clusters

- Calculate the WSS, BSS and TSS values.
    - For a good clustering process the BSS and WSS values should be such that BSS > WSS. The values for these 2 clusters are as follows,

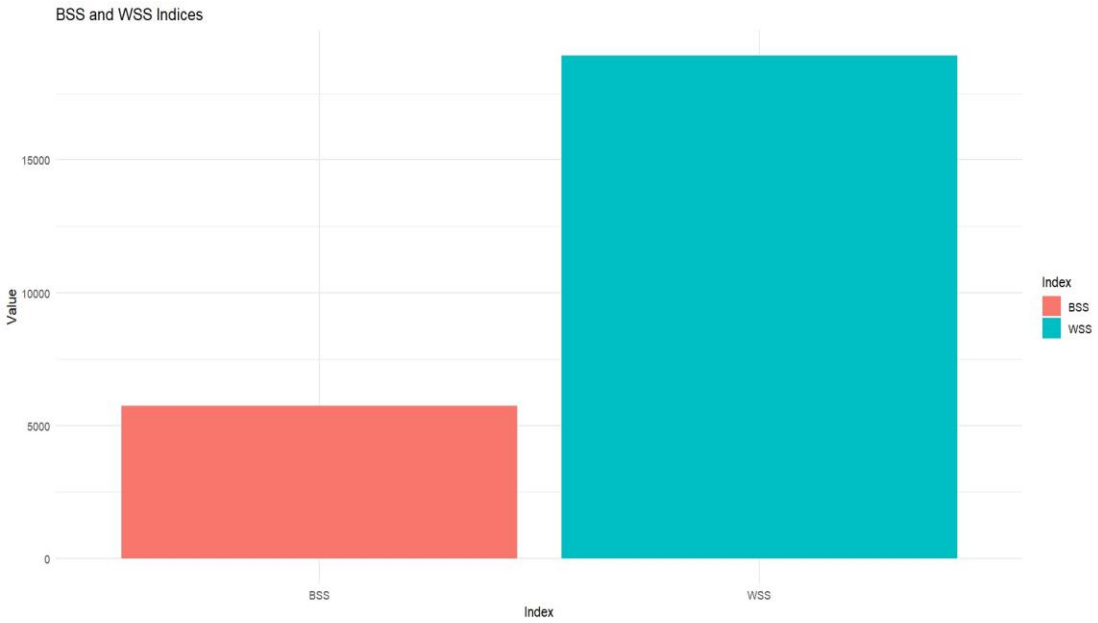

Figure 17 : Graph of BSS and WSS

- From the plot ,

  WSS - 18924.69

  BSS - 5737.309

  TSS - 24662

  Ratio - 23.26376%

- I increased the number of clusters to 3 and took the WSS and BSS values.

  WSS - 17208.16

  BSS - 7453.845 TSS

  - 24662

○ The silhouette plot for this K-means output.

  ■ First calculate silhouette width by comparing the average distance of data points in the same cluster (cohesion) and those average distances to data points in other clusters (separation).

  ■ Then calculate the Average silhouette width (ASW) by averaging the silhouette width.ASW measures the quality of a clustering.

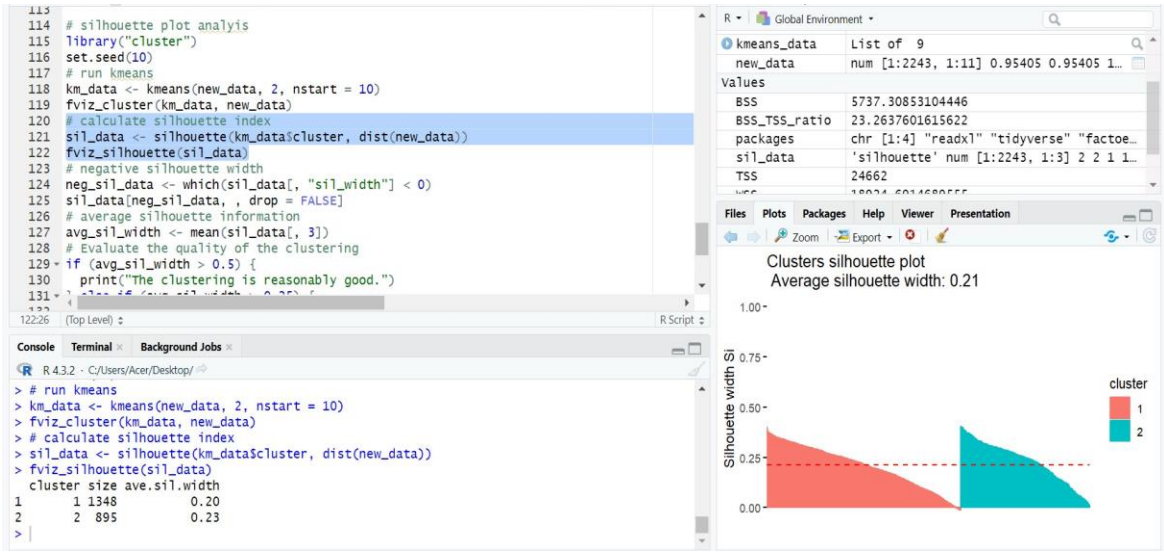

Figure 18 : Silhouette width

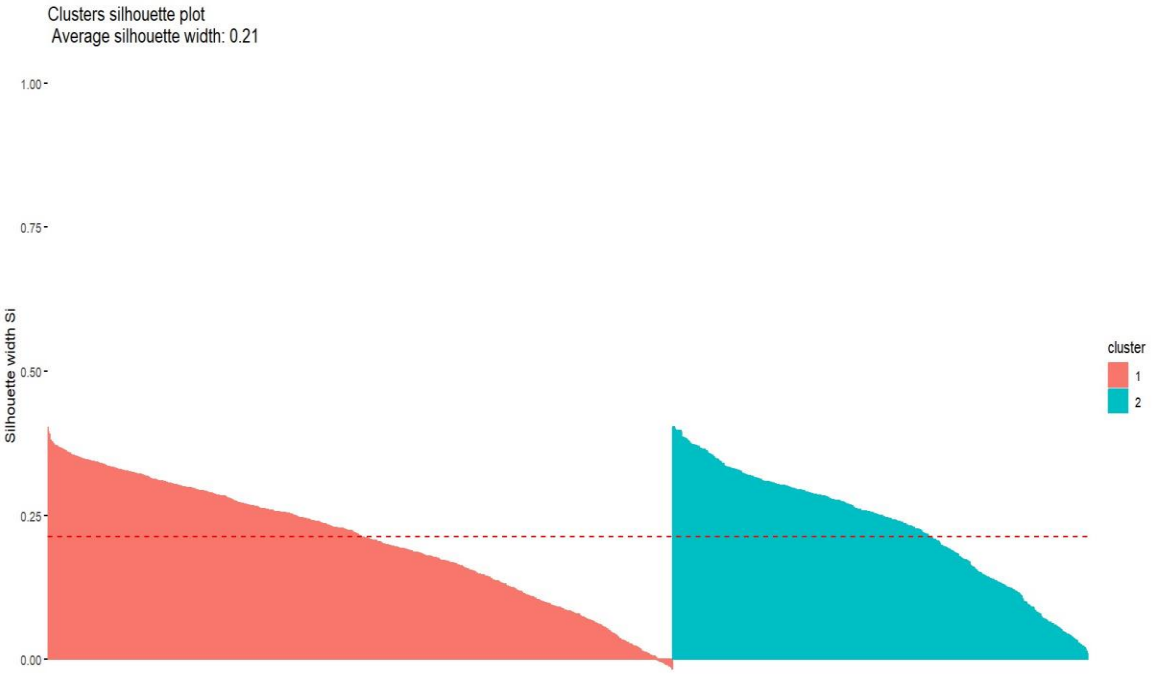

Figure 19 : Silhouette plot with k-means

- As can be seen from the figure, the clusters seem to be separated.
- And I identify the negative silhouette width. Which are the points that are likely misclassified.

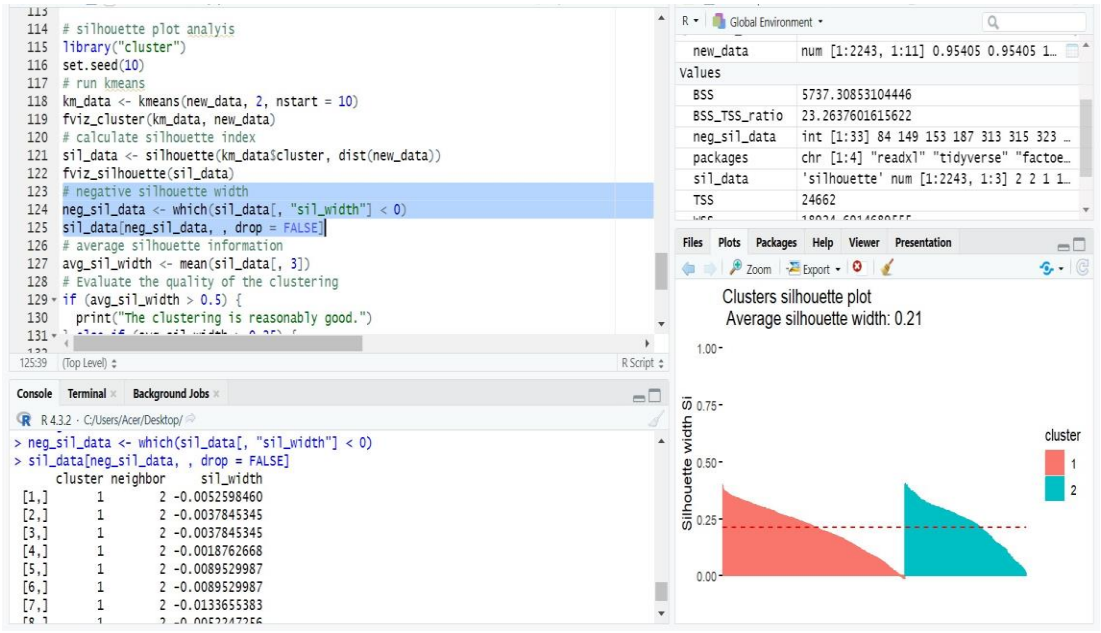

Figure 20 : Negative silhouette values

■ Cluster quality ;

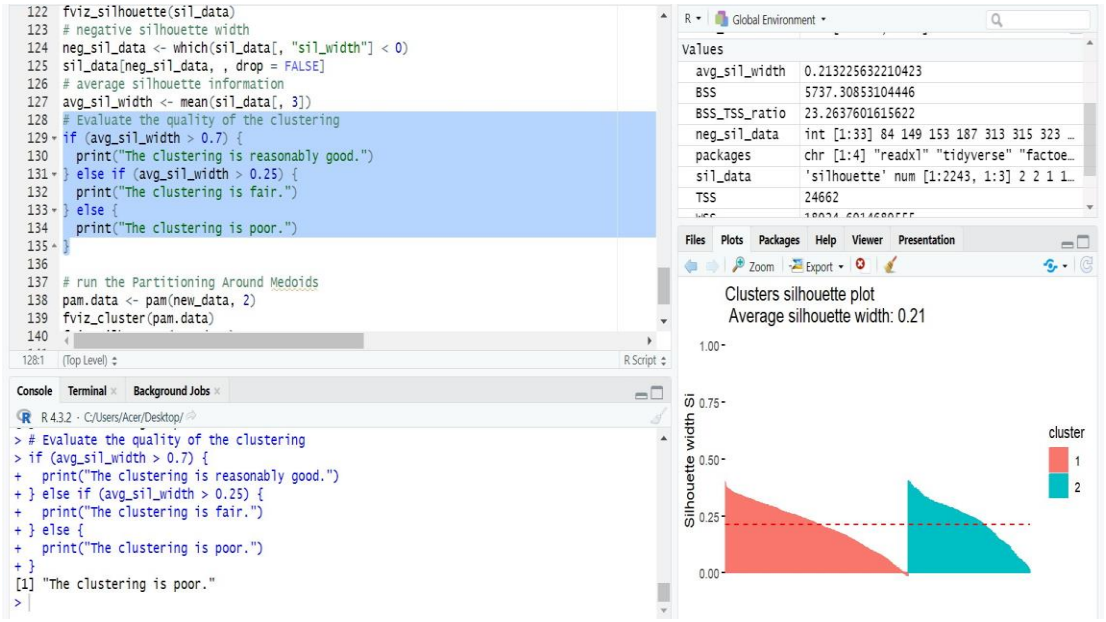

Figure 21 : Cluster quality with Average silhouette width

■ Then I calculate partitioning around medoids (PAM). Rather than using means of data points to represent clusters like K-means, PAM uses actual data points.

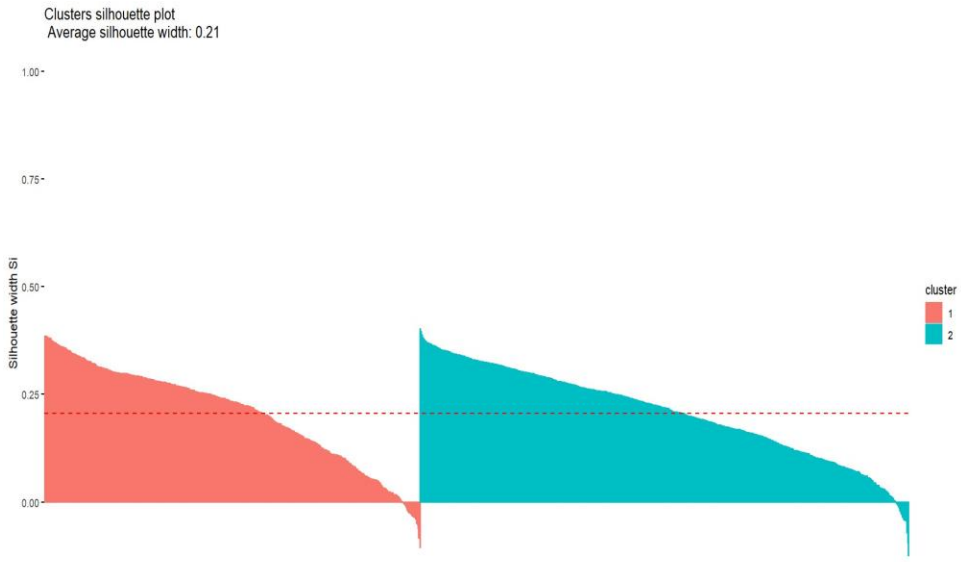

Figure 22 : Silhouette plot with PAM

- The average silhouette width of 0.21 for the two-cluster solution suggests that the clustering might be weak and the data points within each cluster are not very similar to each other. There are several reasons for that,
    - Overlap Between Clusters
    - High-Dimensional Data
    - Incorrect Number of Clusters
- Even if I increase the number of clusters to 3 , the average silhouette width gives a score of 0.14, which is still below 0.25.

Part A- Sub Task 2

- There are steps to calculate pca values for subtask 2. The steps I followed for that are as follows.
- First find out any missing values and if not , remove the quality column from the database, then scale it to the same range.
- First, the covariance matrix is calculated, and the eigenvalues and eigenvectors can be obtained from it. After obtaining the cumulative score using those values, the pca values should be selected according to the condition associated with the cumulative score.
- But there is another way of calculating using R's inbuilt functions.
    - There are 2 functions , princomp() and prcomp() . In my analysis I use the prcomp() function because it provides a more general PCA that considers both variables and individuals. For the summary of prcomp() as follows,

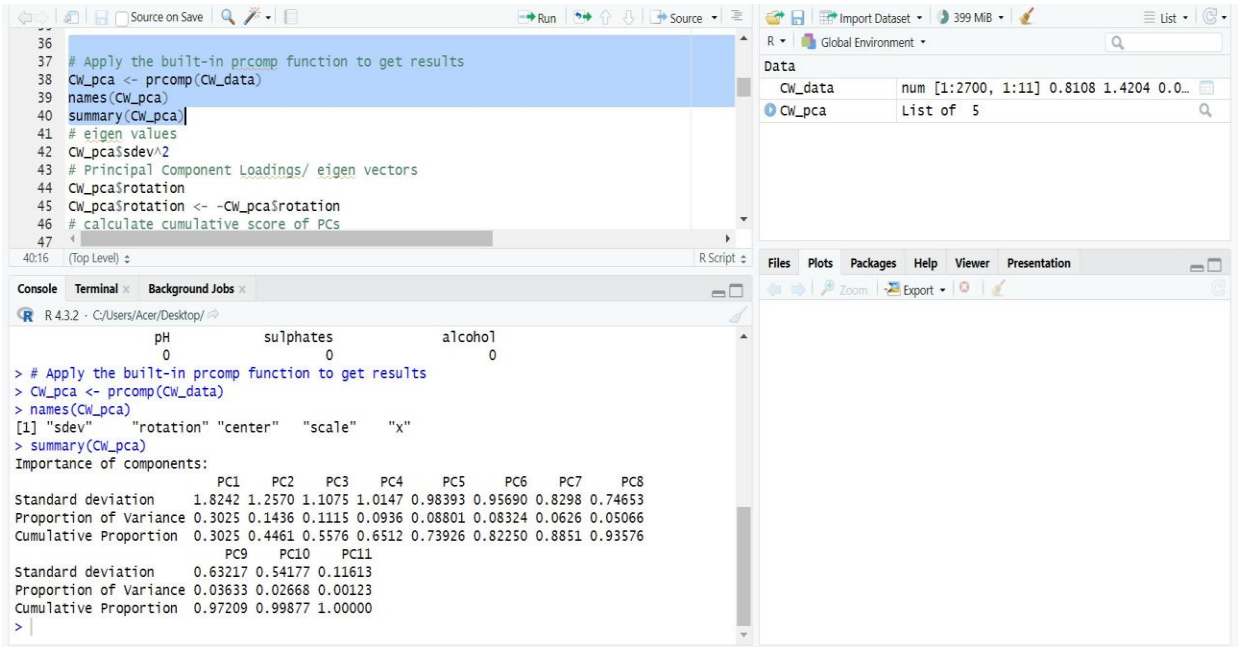

Figure 23 : summary of prcomp()

    - After that , eigenvectors and eigenvalues can be obtained from the result .
    - In the coursework description there is a condition for choosing PCs. Cumulative score must be at least 85%.
    - Cumulative score defines total variance in the data.
    - Given that results PCs 1 to 6 gives 82.2% cumulative score , which is less than 85%. So I need to include the PCs up to PC7, which is greater than 85%.
    - After selecting which PCs , I transformed the dataset with PCs as attributes.

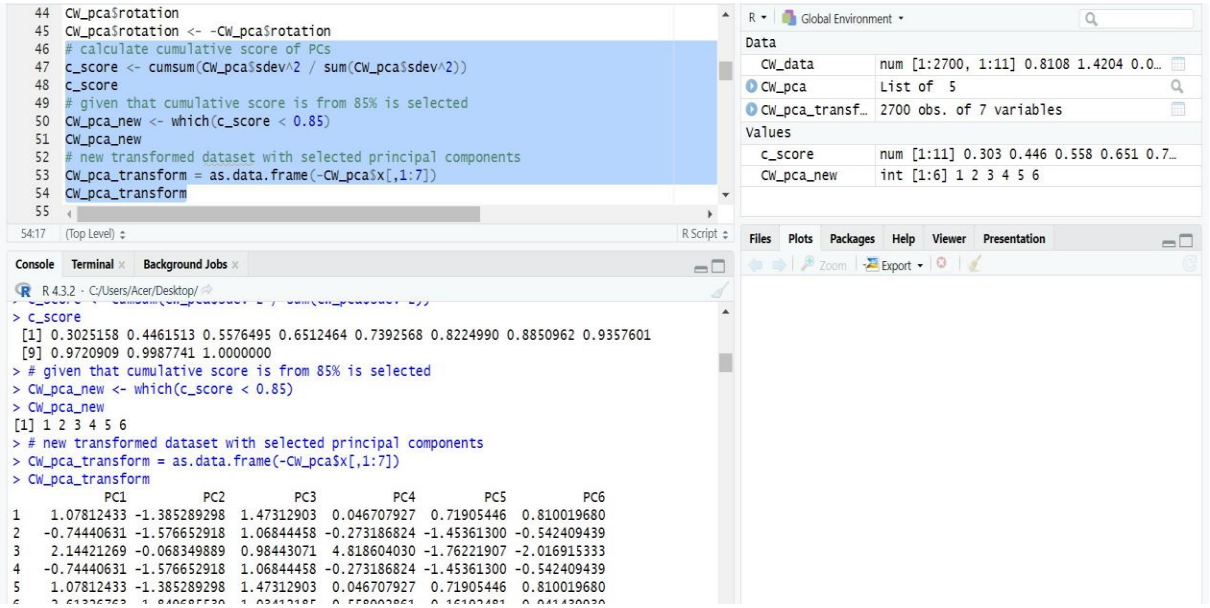

Figure 24 : cumulative scores and dataset transformation

- Analysis of choosing PCs
  - Correlation matrix plot - Relationships between different variables. Red - positive relation, blue - negative relation, white - no relation.

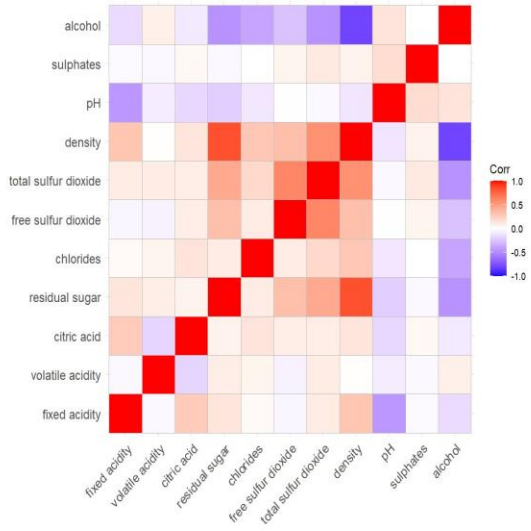

Figure 25 : Correlation matrix plot

  - Scree-plot - plot that shows the fraction of total variance in the data as represented by each Principal Component.

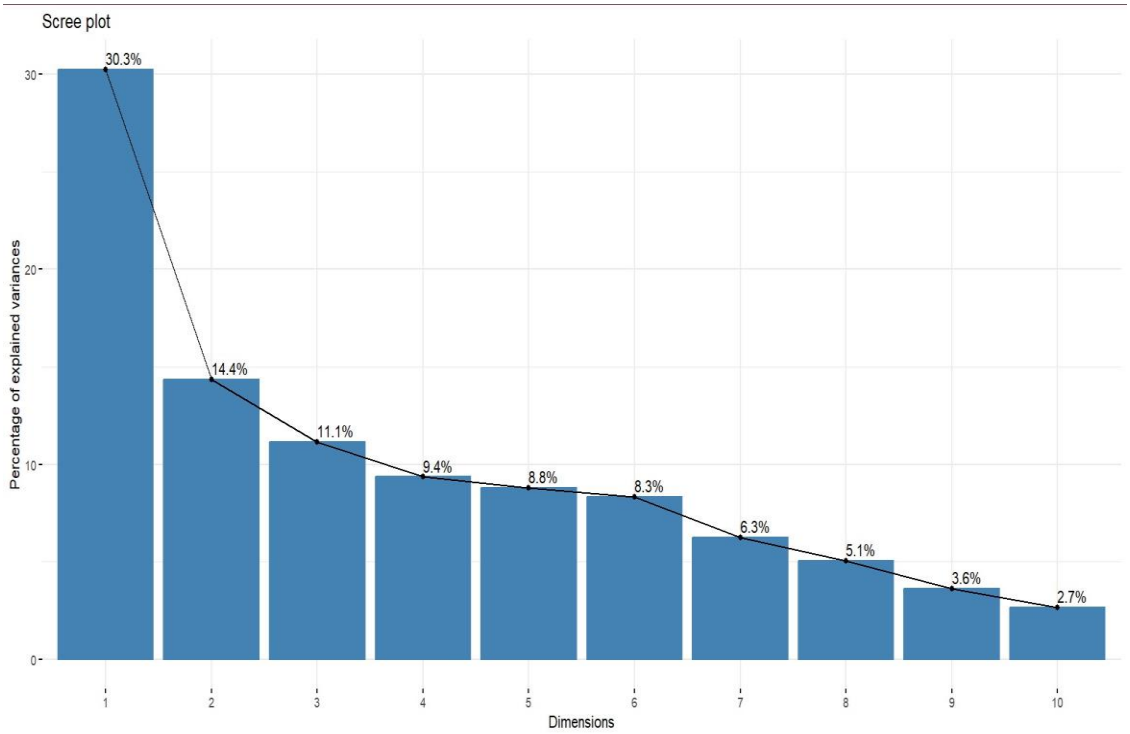

Figure 26 : Scree-plot

■ Bar-plot - same as scree-plot.

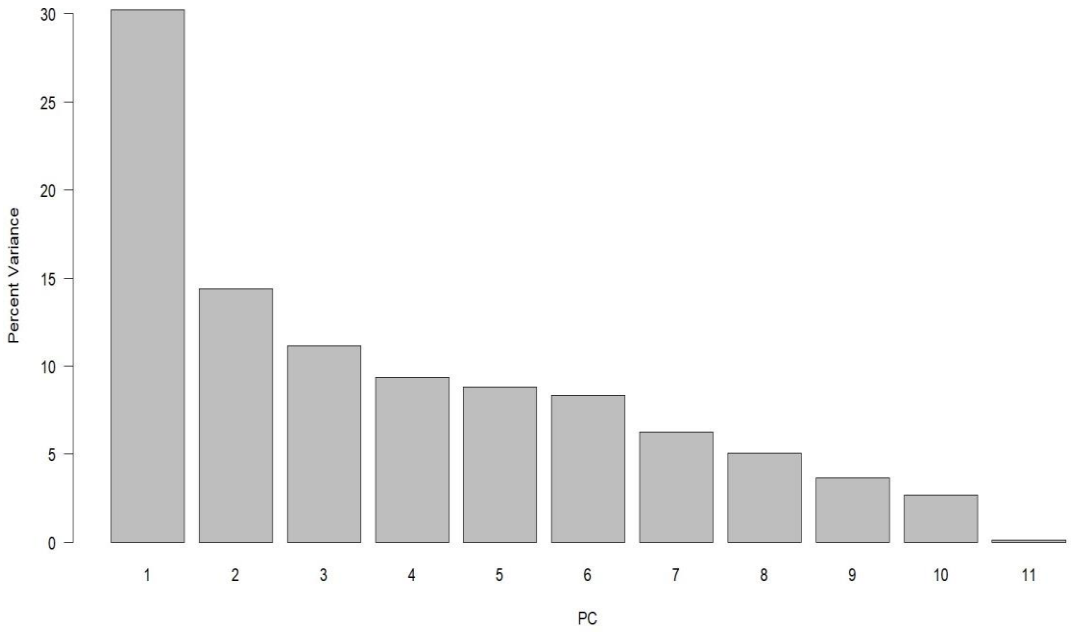

Figure 27 : Bar-plot

■ Bi-plot - plot that shows two-dimensional graphical representation of a multivariate dataset where both the observations and variables are represented,

- View of how the variables structure the observations. The Euclidean distances between points can represent similarity or dissimilarity patterns.
- Here PC1 and PC2 have similar patterns with some variables(point wise).

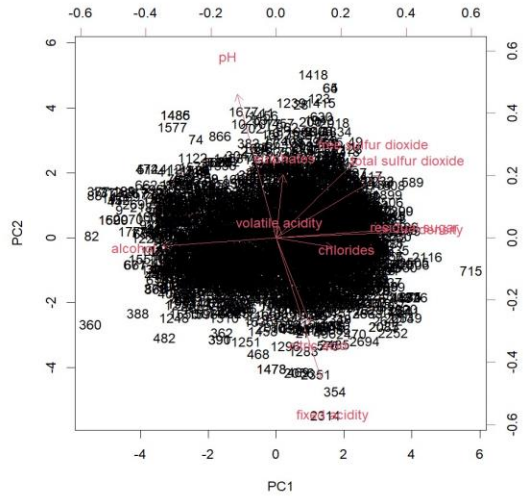

Figure 28 : bi-plot

■ Variables of PCA plot - Visualization of how each variable contributes to the principal components.

- For PC1 and PC2 variable analysis as follows.

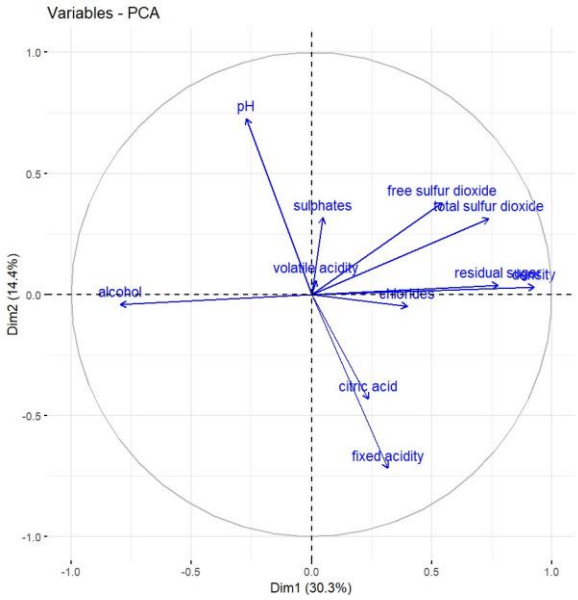

Figure 29 : Variance-plot

- Same direction shows positive relation and opposite direction shows negative relation.

○ For Kmeans clustering I use the transformed dataset with PC attributes.

■ First, choose the number of clusters by NBclust, Elbow, Gap statistics and silhouette methods .

- Elbow method

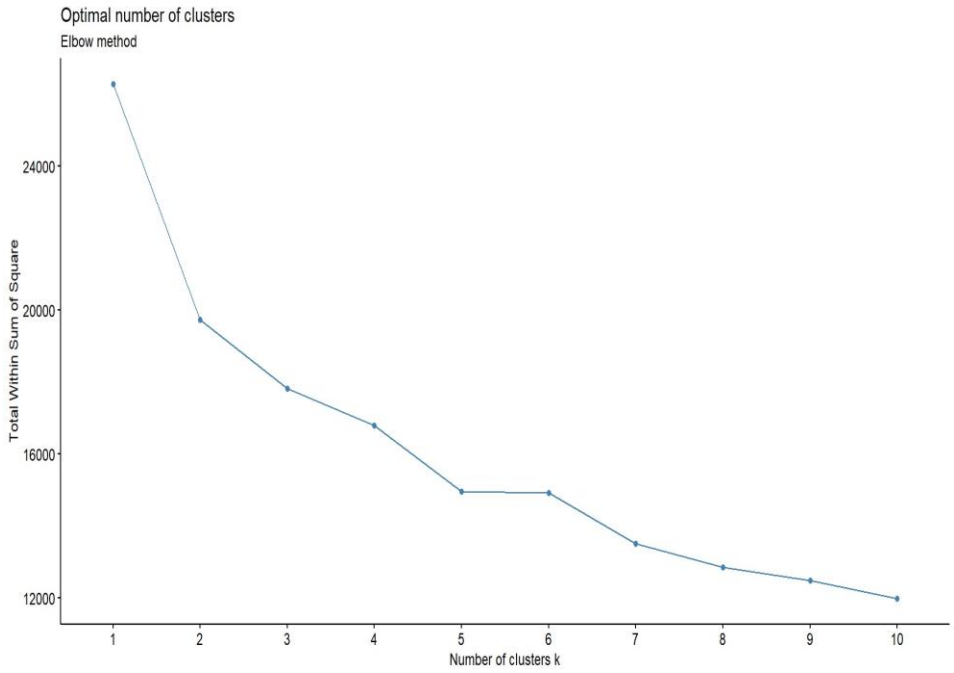

Figure 30: elbow method

- Silhouette method

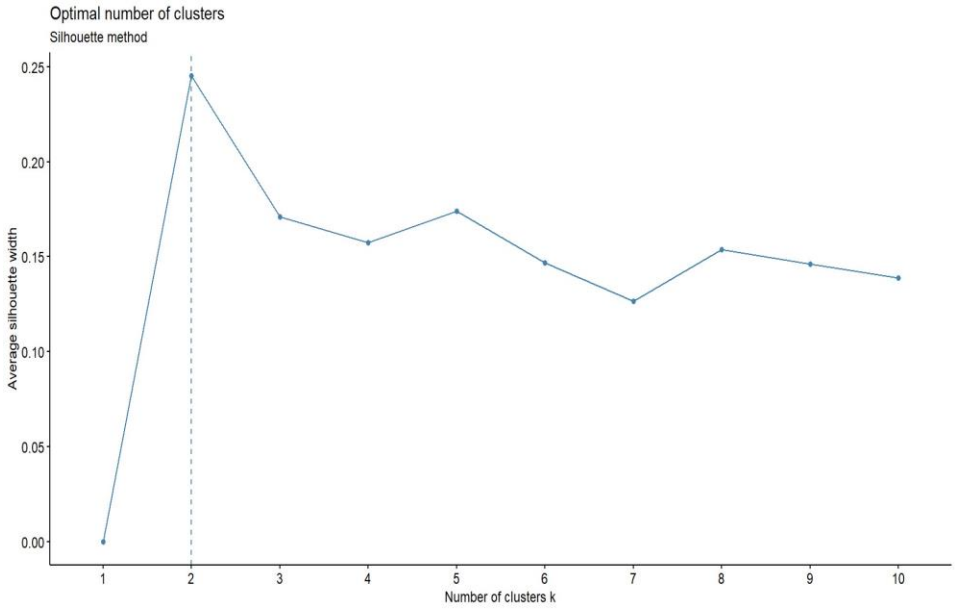

Figure 31: silhouette method

- Gap-statistic method

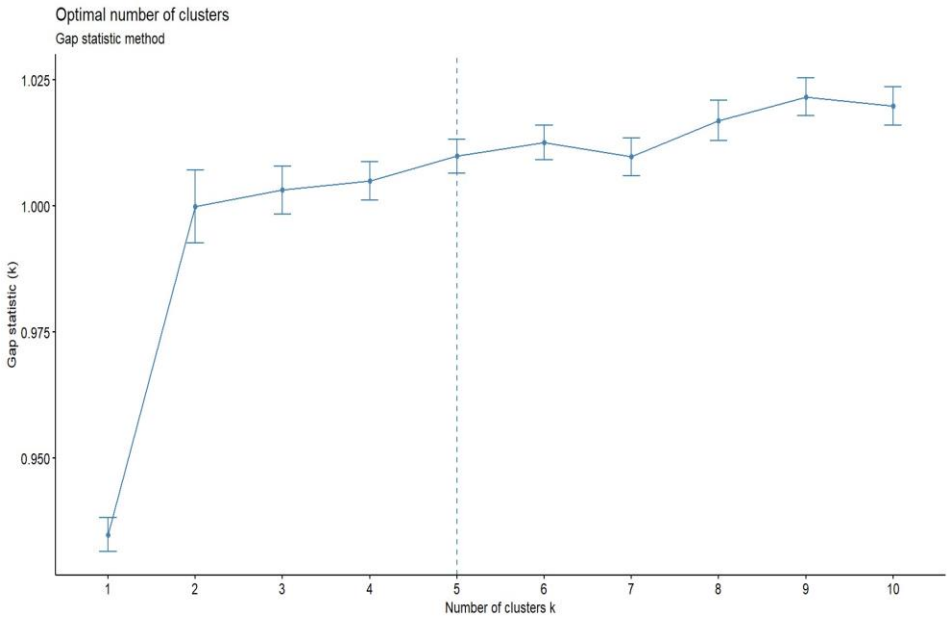

Figure 32: gap-statistic method

- Nbclust method

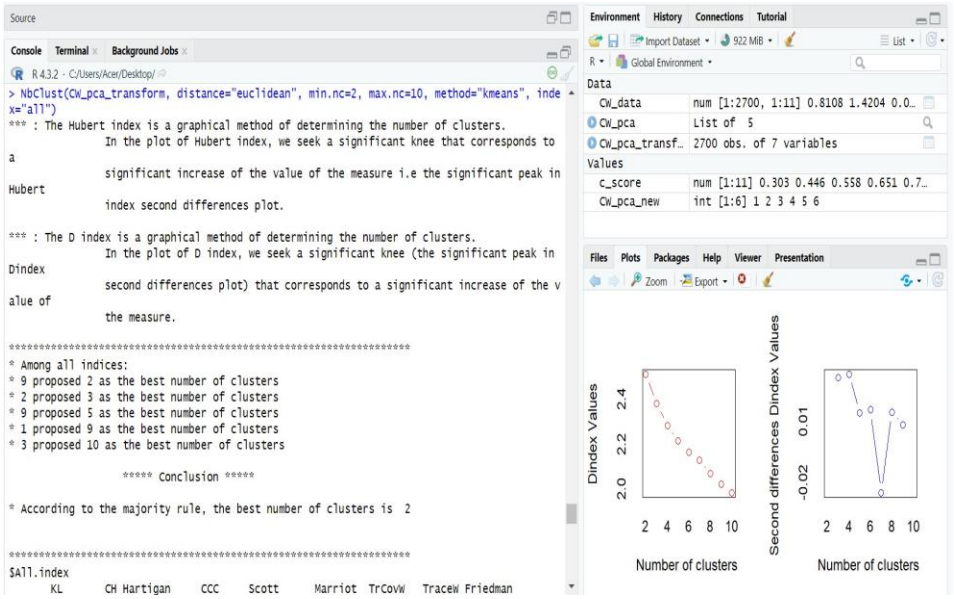

Figure 33: nbclust method with euclidean

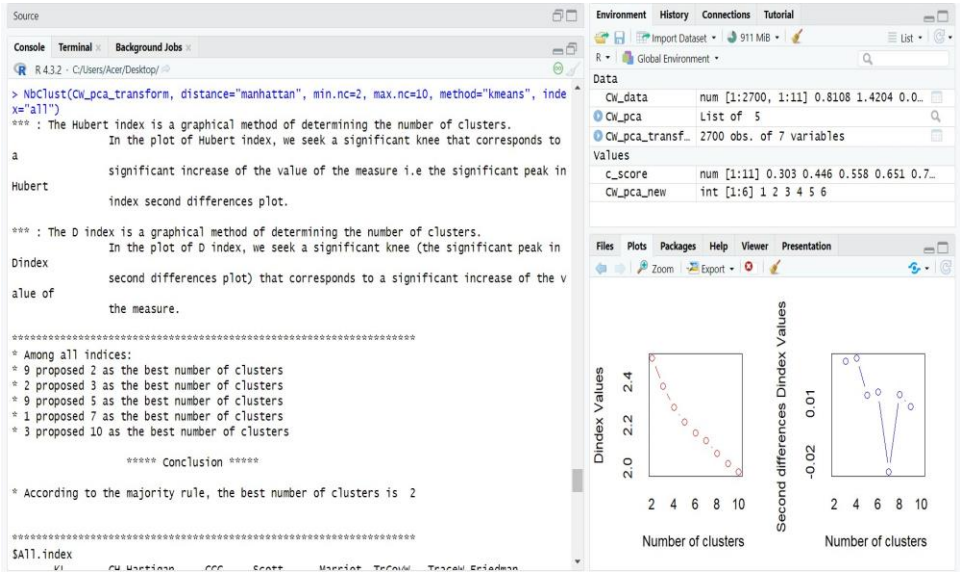

Figure 34: nbclust method with manhattan

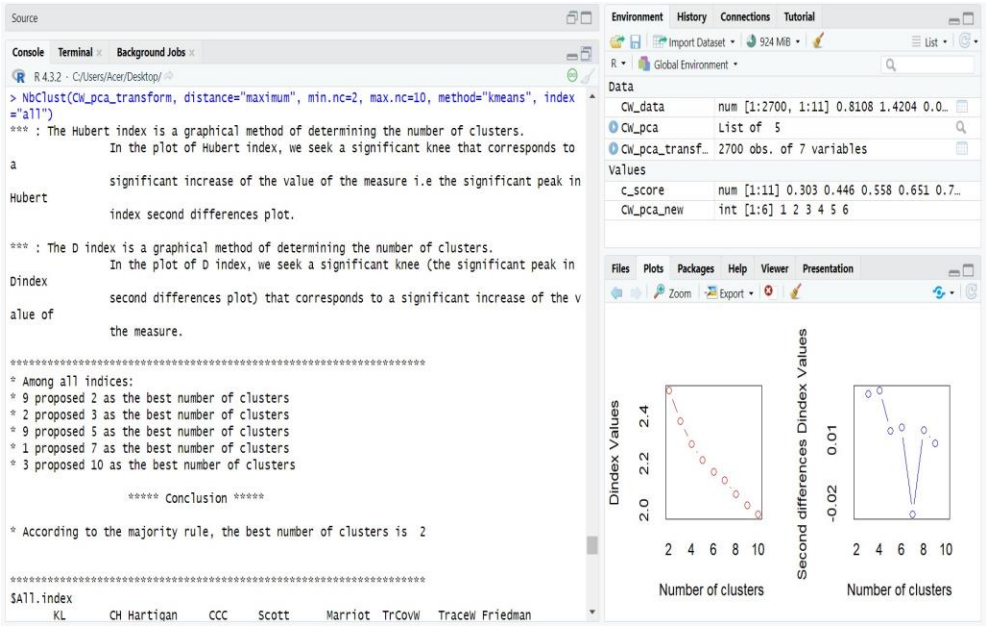

Figure 35: nbclust method with maximum

- In this analysis, the majority of results show that the number of clusters should be 2.

■ Then for the dataset apply the K-means clustering method

- This is how it displays two clusters

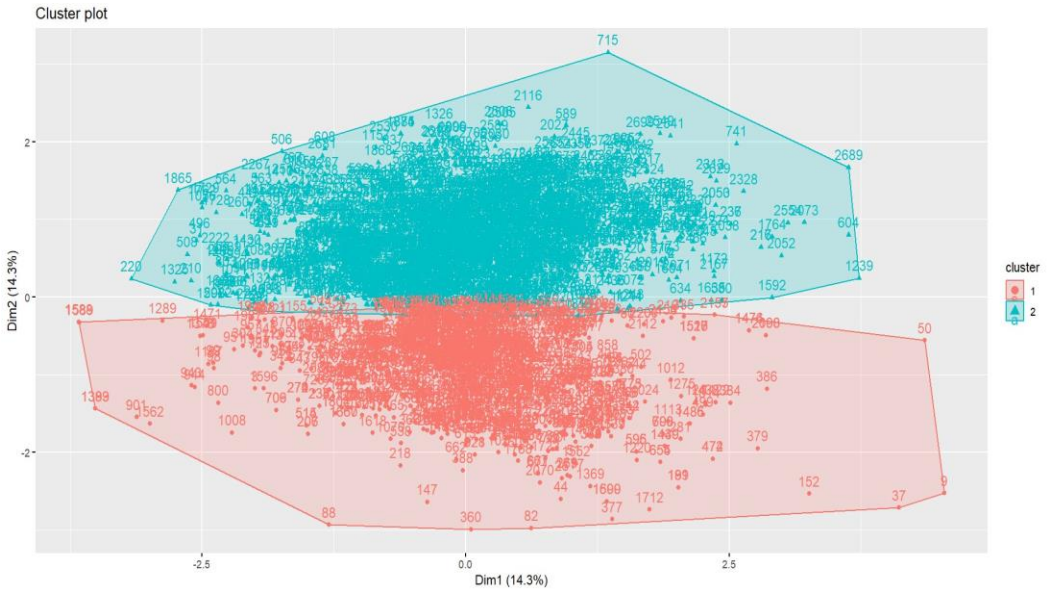

Figure 36: clustering with k means

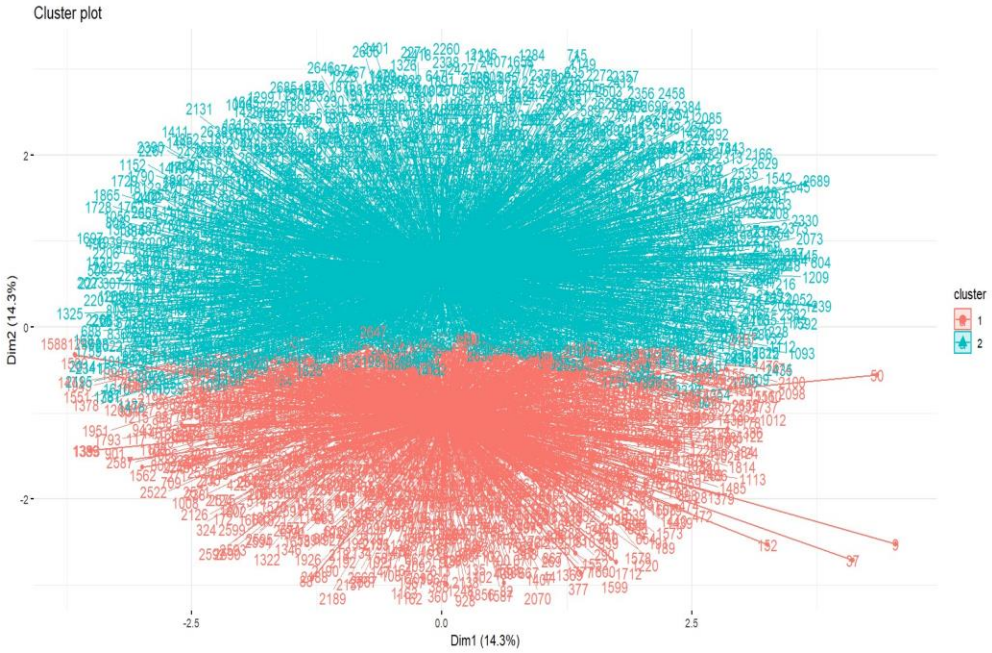

Figure 37: clustering with k means

- Calculate BSS and WSS values,
  - From the Kmeans result ,
    WSS - 19739.04
    BSS - 6538.58
    TSS - 26277.62
    Ratio - 24.88269
- Now to calculate silhouette width.

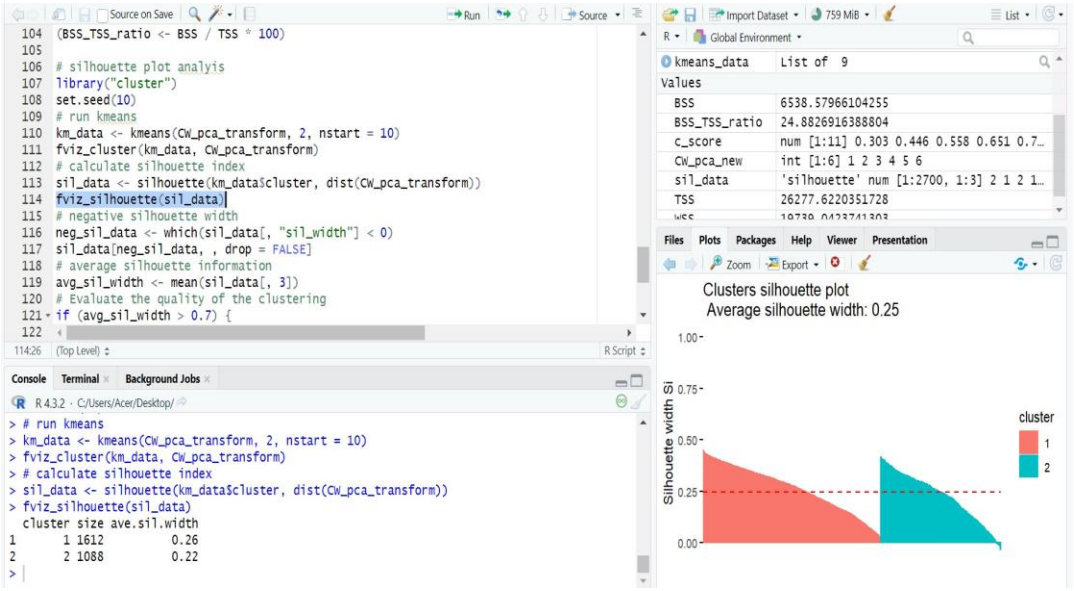

Figure 39: average silhouette width in clusters

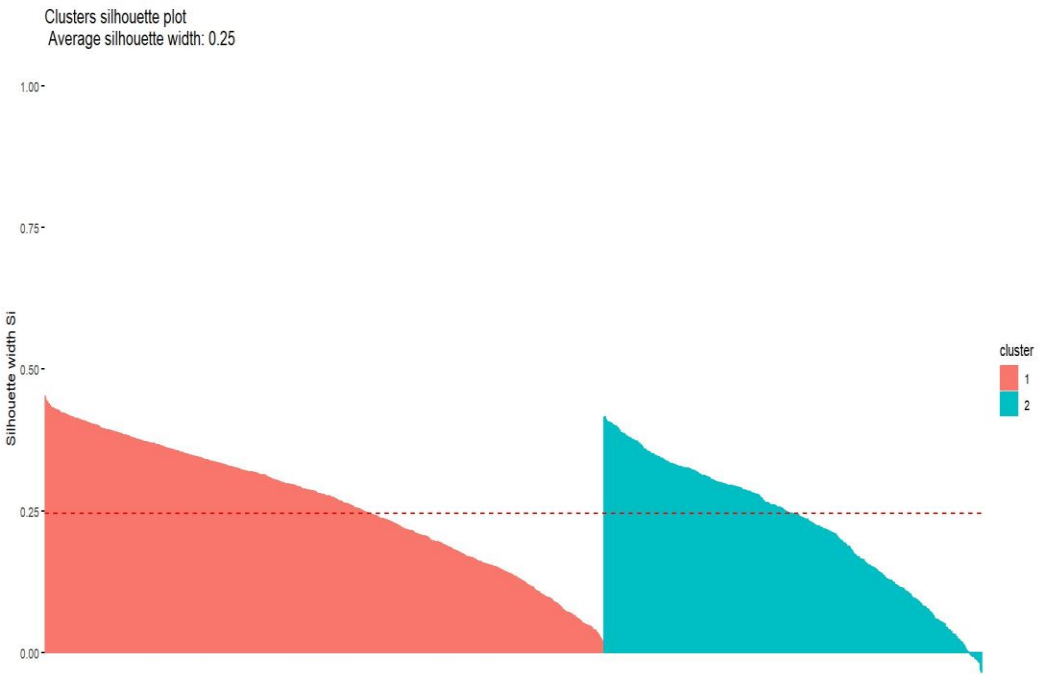

Figure 40: silhouette of 2 clusters

- From the image , clusters are separated with the average width of 0.25.
- This is an improvement from the previous width which is 0.21.
- Negative silhouette width

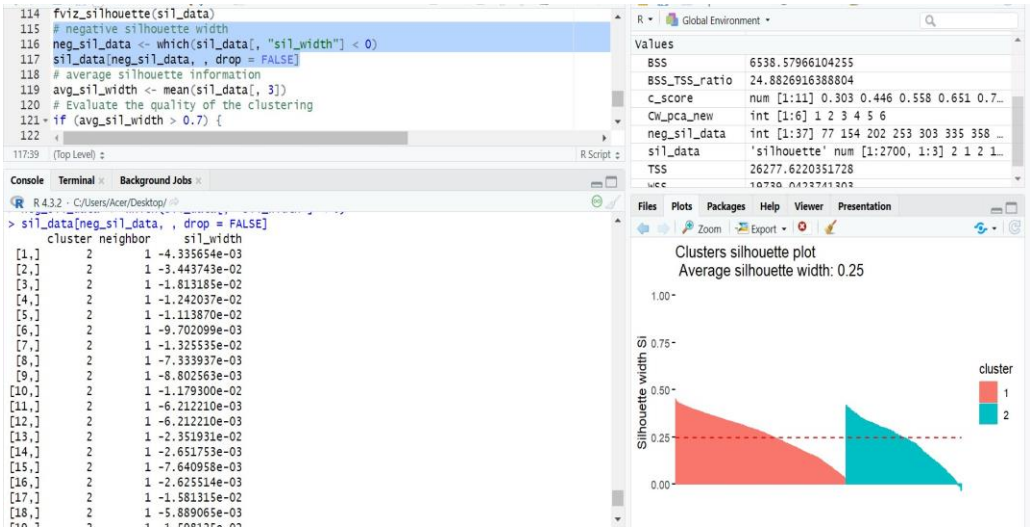

Figure 41: negative silhouette of 2 clusters

- level of "quality" of the cluster calculates with the average silhouette width as follows,

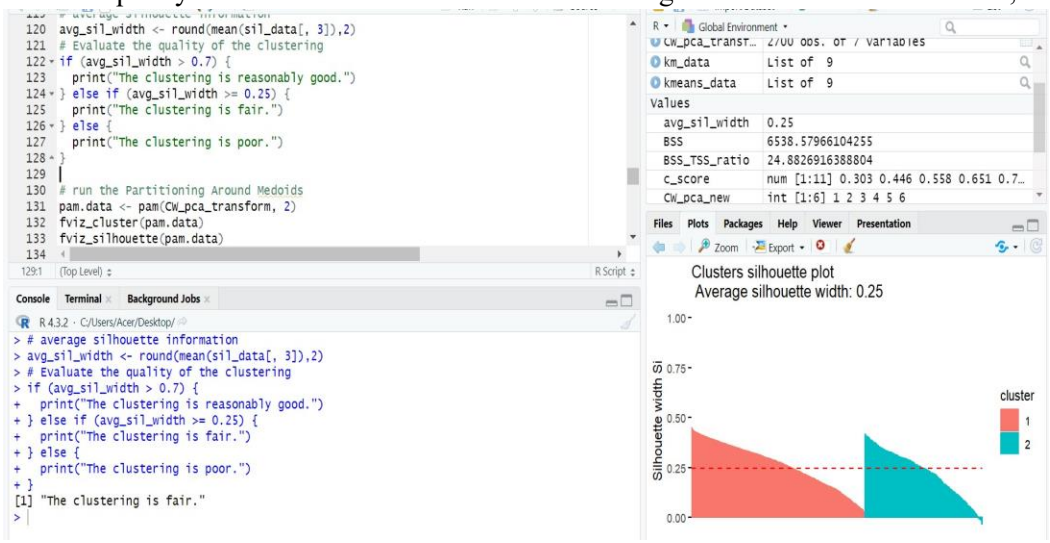

Figure 42: calculate the quality of clusters

- Calinski-Harabasz Index
  - Calinski-Harabasz is a scale used to represent the optimal number of clusters for the data set. It is calculated with BSS, WSS, number of clusters and number of rows in the data set.
  - To calculate CH index i built a function as follows,

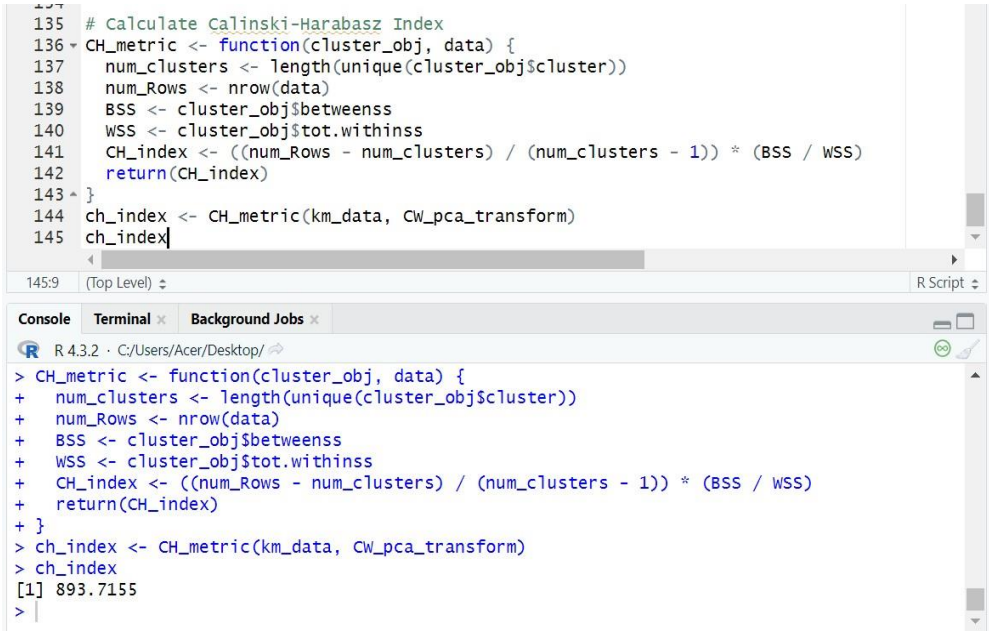

Figure 43: CH Index function

- I created a function to calculated for several clusters

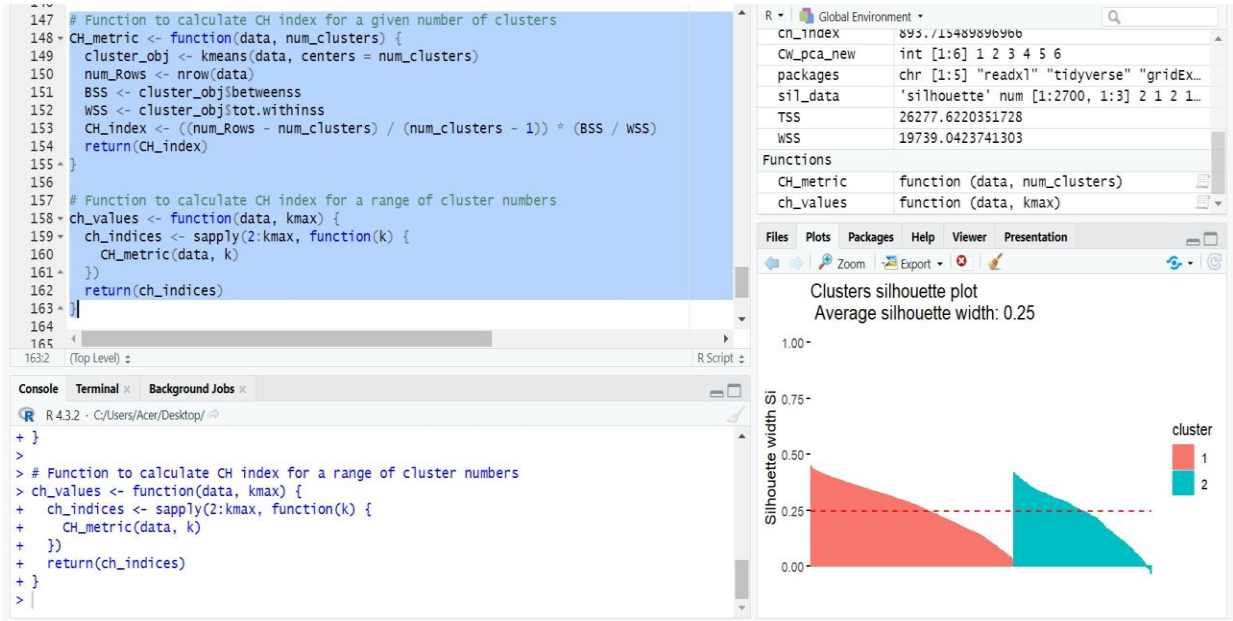

Figure 44: CH Index function for given number of clusters

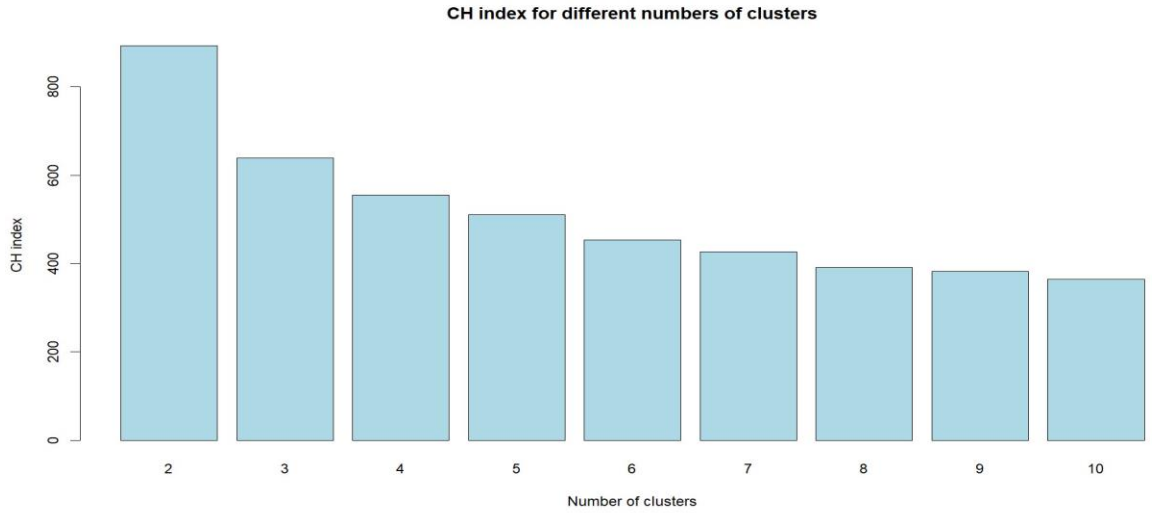

Figure 45: plot for CH Index and given number of clusters

- A decision should be made based on the highest CH index to determine which is the most optimal for the number of clusters.
- From the results obtained 2 is optimal for the number of clusters.

Part B

- According to this task description, when forecasting exchange rates using multi-layer neural networks, the selection of input variables plays a critical role in the performance of the model.
- Several approaches have been proposed in the literature to define the input vector for MLP-NN in this domain.

1. Method discussion:
    a. Fundamental Economic Indicators
    - Fundamental economic indicators, such as interest rates, inflation rates, GDP growth, trade balances, and employment data, can influence exchange rates. These variables can be controlled.
    - Example: The input vector could include the interest rate differential between two countries, inflation rate differential, and GDP growth rate differential.

    b. Exogenous Variables:
    - Exogenous variables refer to external factors that may influence exchange rates, such as political events, natural disasters, or global economic conditions. But these variables cannot be controlled.
    - Example: A binary variable representing the occurrence of a major political event could be included in the input vector.
2. Autoregressive (AR) approach is one of the most commonly used methods as mentioned in this task. It involves using past values (time lagged values) of the exchange rate as input variables.
    ○ According to the description 4 time delay values are specified to be used.
    ○ Then the steps should be done sequentially according to the given data set, data preprocessing, training a model on the data, and evaluating the model performance.

    - Defining the input vector using the "autoregressive" (AR) approach
        ○ As the task suggests I have created 4 input vectors from the dataset as time lagged data, it refers to data points that have been shifted backward in time.
        ○ For (T) - current point/value , (T-1) - point from the one time period , (T-2) - point from the two time period , (T-3) - point from the three time period, (T-4) - point from the four time period .
        ○ (T-4) to (T-1) are taken as inputs and the output value (predicted) obtained from the inputs can be compared with the actual output (T) value to determine the efficiency.
    - Splitting the data into training and testing sets and normalization..
        ○ As given in the description, the training set includes [1:400] and the testing set includes [401:500].
        ○ Creating the input/output matrices for each training and testing set , there are 4 input vectors and one output vector. So each train set and test set have 1 input vector to 4 input vectors with one output vector metrics. As follows for each of the training and testing metrics,
        ○ AR approach 1

input	output

(T-1)	T

- AR approach 2

input	input	output
(T-2)	(T-1)	T

- AR approach 3

input	input	input	output
(T-3)	(T-2)	(T-1)	T

- AR approach 4

input	input	input	input	output
(T-4)	(T-3)	(T-2)	(T-1)	T

3. Then normalization and de-normalisation for both training and testing metrics.
    a. Normalization
        - Neural networks perform better when the input features are scaled to a similar range. This is because the weight updates during training. If the input features have significantly different scales, it can prevent the network from learning effectively.
        - Normalization is the process of rescaling the input features to a common range, typically between 0 and 1. This ensures that no single feature dominates the weight updates and allows the network to learn patterns more efficiently.

	InputV1	InputV2	InputV3	InputV4	Output
1	1.3730	1.3860	1.3768	1.3718	1.3774
2	1.3860	1.3768	1.3718	1.3774	1.3672
3	1.3768	1.3718	1.3774	1.3672	1.3872
4	1.3718	1.3774	1.3672	1.3872	1.3932
5	1.3774	1.3672	1.3872	1.3932	1.3911
6	1.3672	1.3872	1.3932	1.3911	1.3838
7	1.3872	1.3932	1.3911	1.3838	1.4171
8	1.3932	1.3911	1.3838	1.4171	1.4164
9	1.3911	1.3838	1.4171	1.4164	1.3947
10	1.3838	1.4171	1.4164	1.3947	1.3675
11	1.4171	1.4164	1.3947	1.3675	1.3801
12	1.4164	1.3947	1.3675	1.3801	1.3744
13	1.3947	1.3675	1.3801	1.3744	1.3759

Figure 46: before normalization

	InputV1	InputV2	InputV3	InputV4	Output
1	0.7909953	0.8526066	0.8090047	0.7853081	0.8118483
2	0.8526066	0.8090047	0.7853081	0.8118483	0.7635071
3	0.8090047	0.7853081	0.8118483	0.7635071	0.8582938
4	0.7853081	0.8118483	0.7635071	0.8582938	0.8867299
5	0.8118483	0.7635071	0.8582938	0.8867299	0.8767773
6	0.7635071	0.8582938	0.8867299	0.8767773	0.8421801
7	0.8582938	0.8867299	0.8767773	0.8421801	1.0000000
8	0.8867299	0.8767773	0.8421801	1.0000000	0.9966825
9	0.8767773	0.8421801	1.0000000	0.9966825	0.8938389
10	0.8421801	1.0000000	0.9966825	0.8938389	0.7649289
11	1.0000000	0.9966825	0.8938389	0.7649289	0.8246445
12	0.9966825	0.8938389	0.7649289	0.8246445	0.7976303
13	0.8938389	0.7649289	0.8246445	0.7976303	0.8047393

Figure 47: after normalization

- For example, consider an input vector containing the exchange rate. Without normalization, the exchange rate values (e.g., ranging from 1.2 to 1.4) and Normalization would scale these features to a similar range(0 to 1).

b. Denormalization

- After training the MLP network with normalized inputs, it is necessary to de-normalize the outputs to obtain the exchange rate predictions. Denormalization is the reverse process of normalization, where the scaled outputs are transformed back to their original units and range. ● Outputs(predictions) are compared with actual outputs.

4. Implement a number of MLPs for the “AR” approach,
   - Hyper-parameter tuning
     - Although the input and output vectors are given, the neural network has hidden layers. Based on those hidden layers, the performance of the model can be low or high, the number of nodes with hidden layers should be found to get the best model. Here modeling by random values is called hyperparameter tuning.

- One input

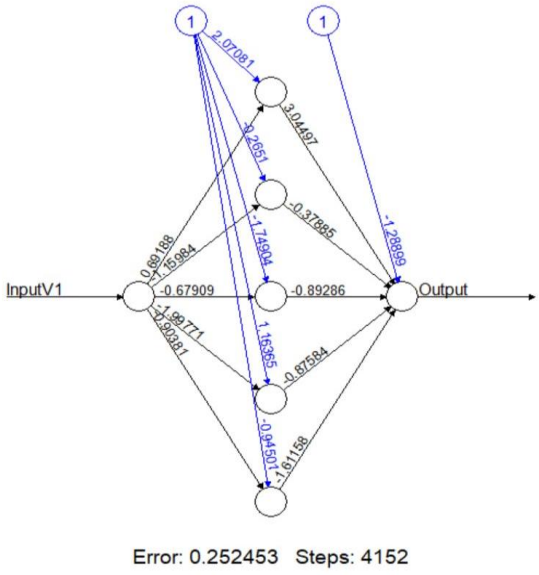

Figure 48: 1 hidden layer, 5 neurons

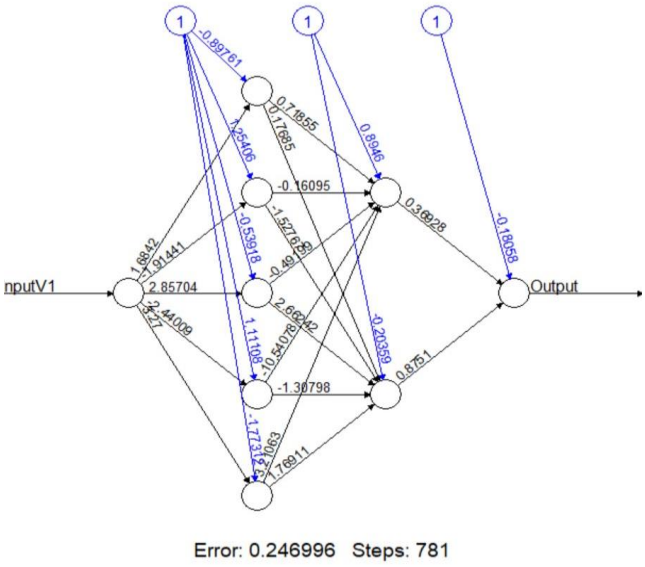

Figure 49: 2 hidden layer , 10 neurons

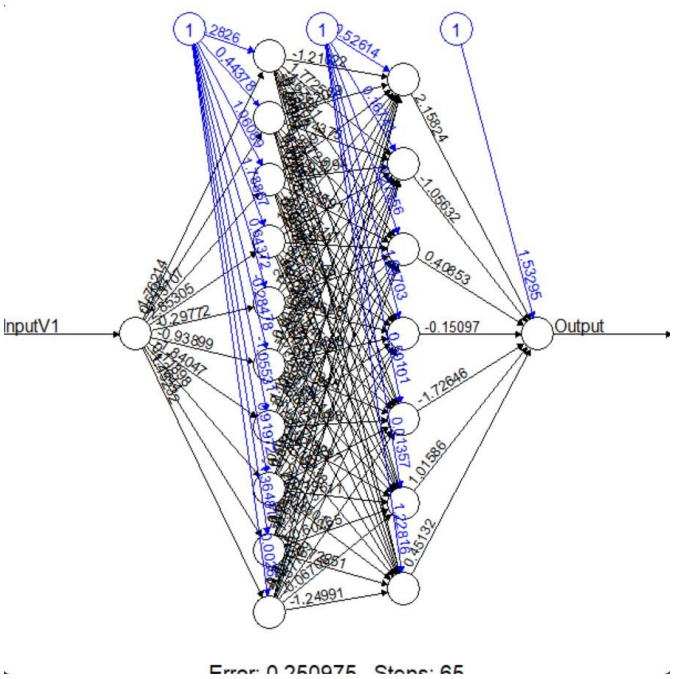

Figure 50: 1 hidden layer, 70 neurons

- Two inputs

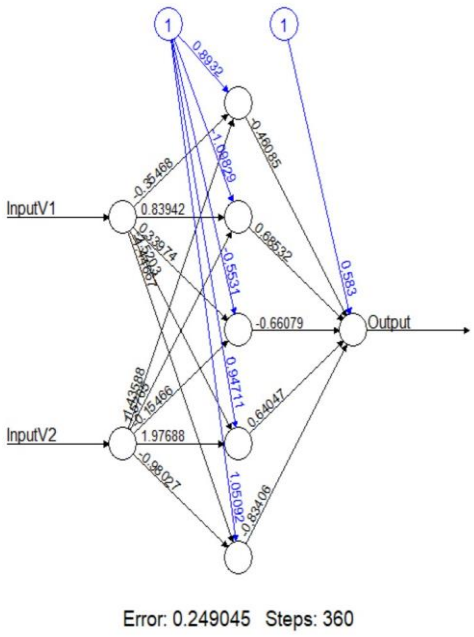

Figure 51: 1 hidden layer, 5 neurons

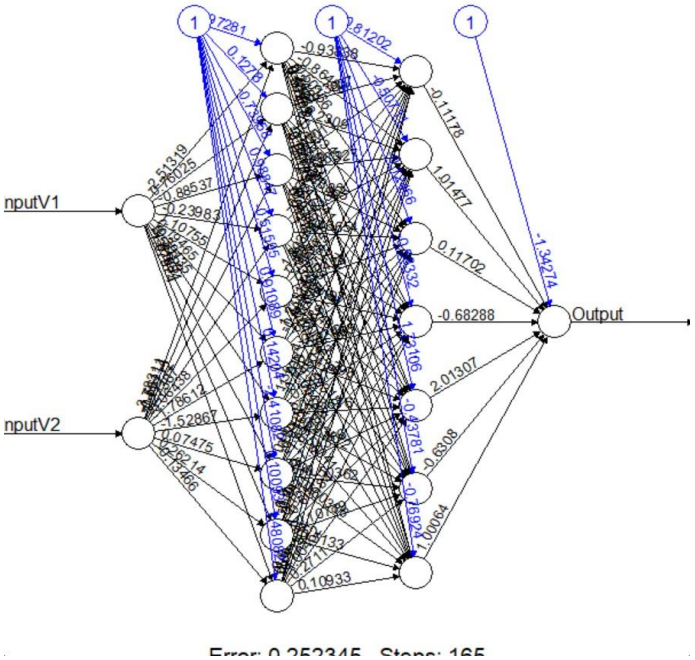

Figure 52: 2 hidden layer, 70 neurons

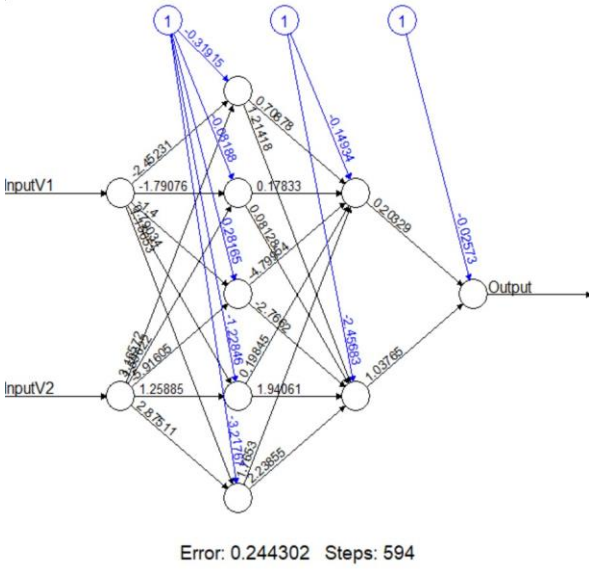

Figure 53: 2 hidden layer, 10 neurons

- Three inputs

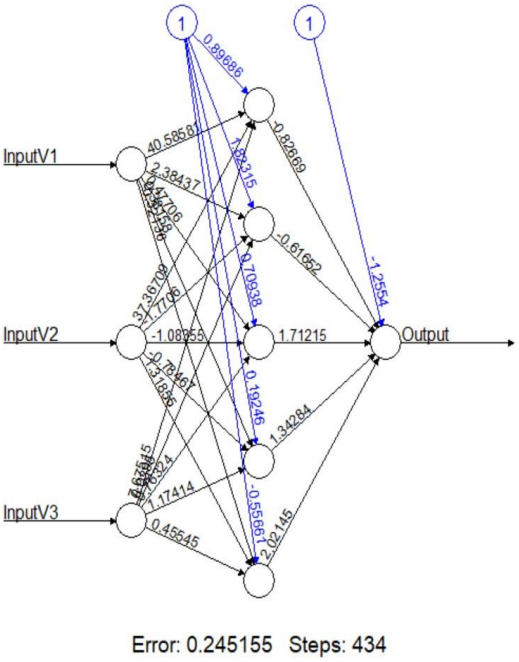

Figure 54: 1 hidden layer, 5 neurons

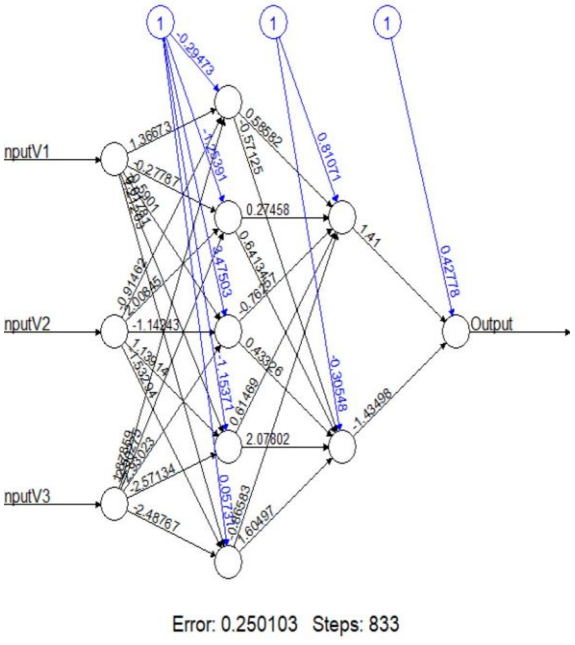

Figure 55: 2 hidden layer, 10 neurons

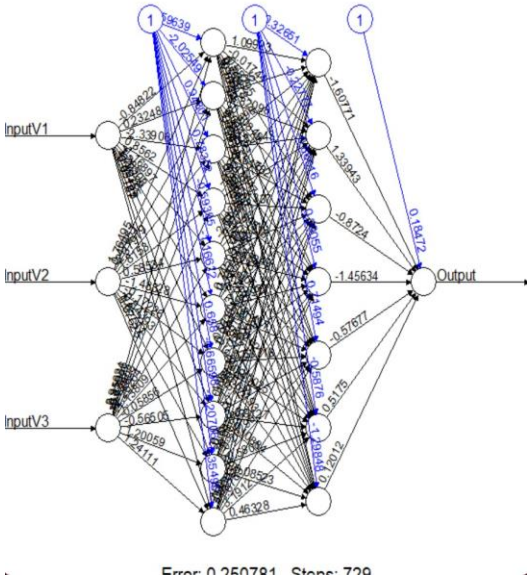

Figure 56: 2 hidden layer, 70 neurons

- Four inputs layer:

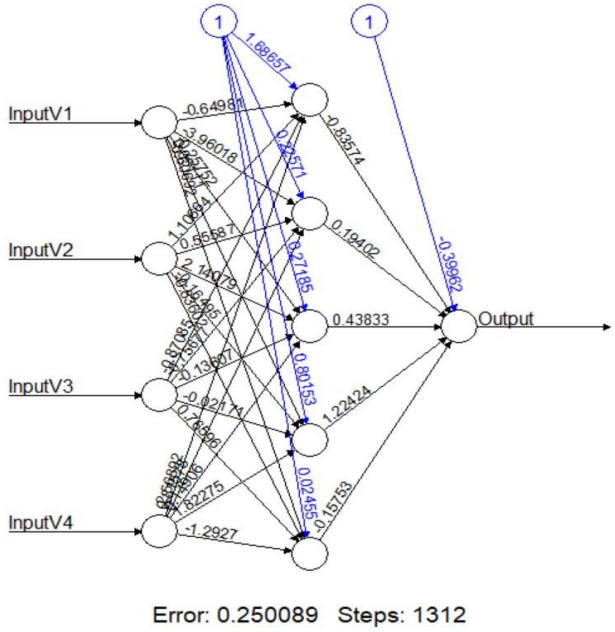

Figure 57: 2 hidden layer, 5 neurons

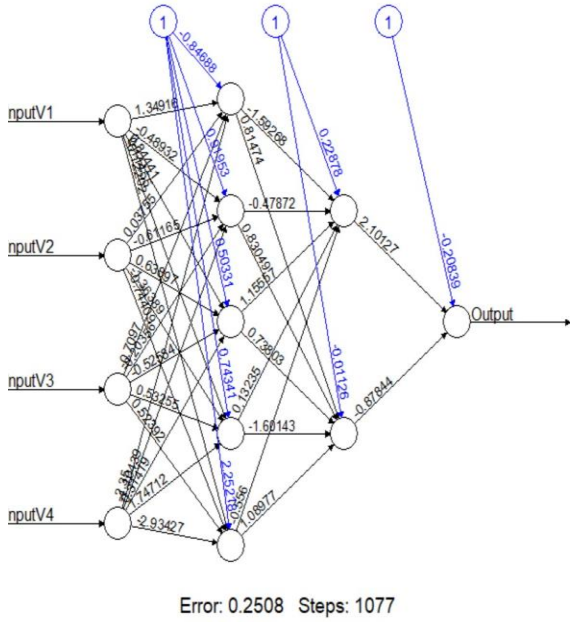

Figure 58: 2 hidden layer, 10 neurons

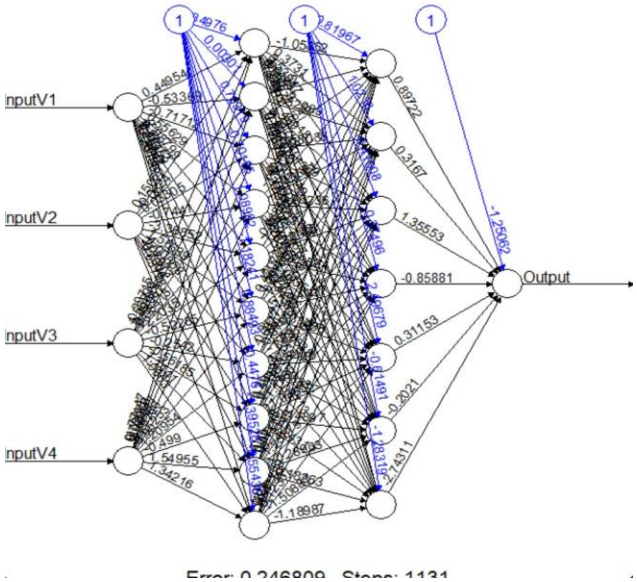

Figure 59: 2 hidden layer, 70 neurons

5. Statistical indices are commonly used to evaluate the performance of the models, including those used for exchange rate prediction.
    - Root Mean Squared Error (RMSE):
        - Square root of the average squared difference between the predicted values and the actual values.
        - It gives higher weights to larger errors, making it sensitive to outliers.

■ Use: Gives the overall magnitude of the errors ●

Mean Absolute Error (MAE):

■ Average absolute difference between the predicted values and the actual values.

■ It is less sensitive to outliers compared to RMSE.

■ Use: Presents an interpretation of the average absolute error in the same units ●

Mean Absolute Percentage Error (MAPE):

■ Average absolute difference between the predicted and actual values as a percentage of the actual values.

■ Use: Comparing the accuracy of predictions across different scales.

- Symmetric Mean Absolute Percentage Error (sMAPE):
  - ■ Average of the absolute percentage errors, where the percentage error is calculated as the absolute difference between the actual and predicted values divided by the sum of their absolute values.
  - ■ sMAPE is bounded between 0 and 200% or (0 to 2).
- A lower RMSE,MAE,MAPE,MAPE value indicates better model performance.

6. Creation of the comparison matrix

```
> comparison_Table
   Input_nodes Hidden_layers HIdden_nodes        RMSE         MAE        MAPE       SMAPE
1            1             1            5 0.006128683 0.004650304 0.003521062 0.003524165
2            2             1            5 0.006132359 0.004541487 0.003438077 0.003440867
3            3             1            5 0.006210147 0.004664654 0.003528511 0.003532015
4            4             1            5 0.006195269 0.004619499 0.003493961 0.003496844
5            1             2          5-2 0.006209847 0.004744705 0.003592022 0.003595946
6            2             2          5-2 0.006164847 0.004680260 0.003540935 0.003544478
7            3             2          5-2 0.006184814 0.004574697 0.003462218 0.003465071
8            4             2          5-2 0.006108971 0.004536396 0.003432946 0.003435433
9            1             2         10-7 0.006135852 0.004613653 0.003494374 0.003497343
10           2             2         10-7 0.006126538 0.004560266 0.003452677 0.003455172
11           3             2         10-7 0.006133680 0.004533118 0.003431977 0.003434546
12           4             2         10-7 0.006116559 0.004634147 0.003505836 0.003508916
```

Figure 60: comparison matrix

- From this table, the model with the lowest overall values can be identified as the best model.
  - ○ 3 input vectors two hidden layer with 10 and 7 nodes ●

And 2nd and best model for one hidden layers

- ○ 2 input vectors one hidden layer with 5 nodes

7. Efficiency of the model

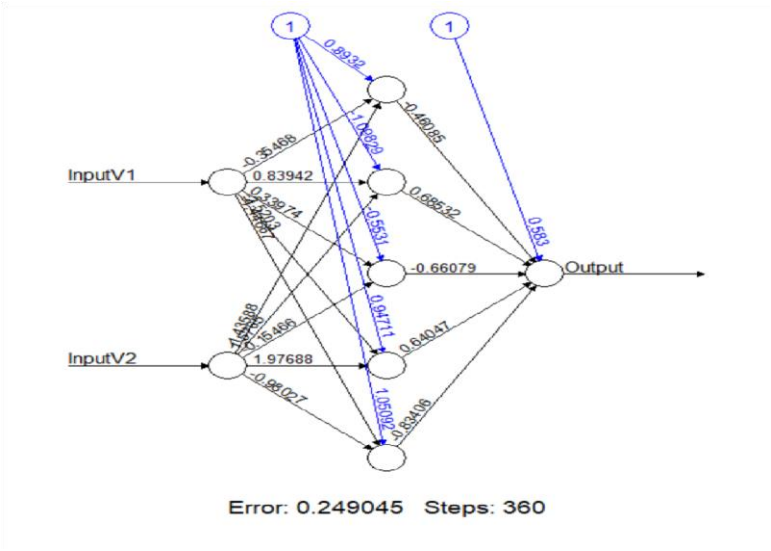

Figure 61: 2 inputs with 1 hidden layer 5 neurons

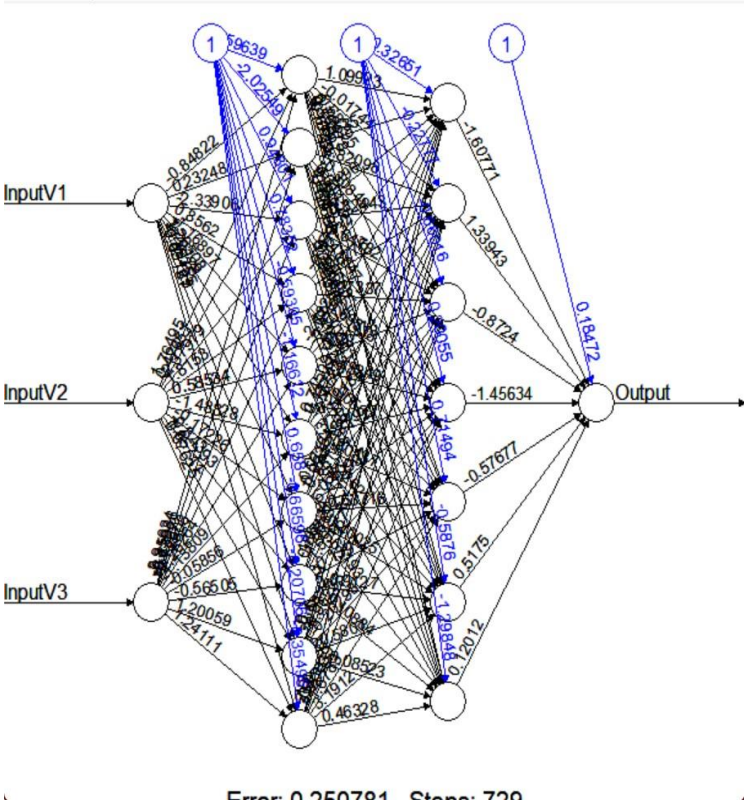

Figure 62: 3 inputs with 2 hidden layer 70 neurons(best model)

8. Prediction output and desired output analysis

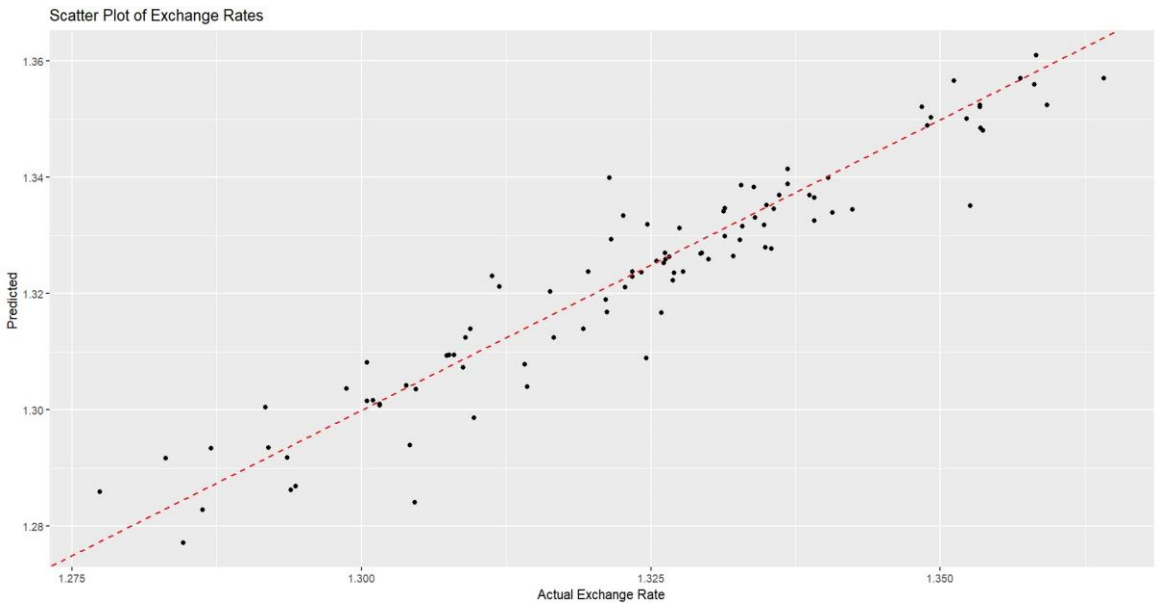

Figure 63: scatter plot of selected model(best model)

- In this scatter plot black dots represent the individual data points and the red dashed line in the diagonal indicates if the predicted values are exactly equal to the actual values then the points lie on the line.
- Looking at this graph, most of the black dots are clustered around this red dashed line, indicating that the model's predictions are generally close to the true values.there are some several outliers can be seen.

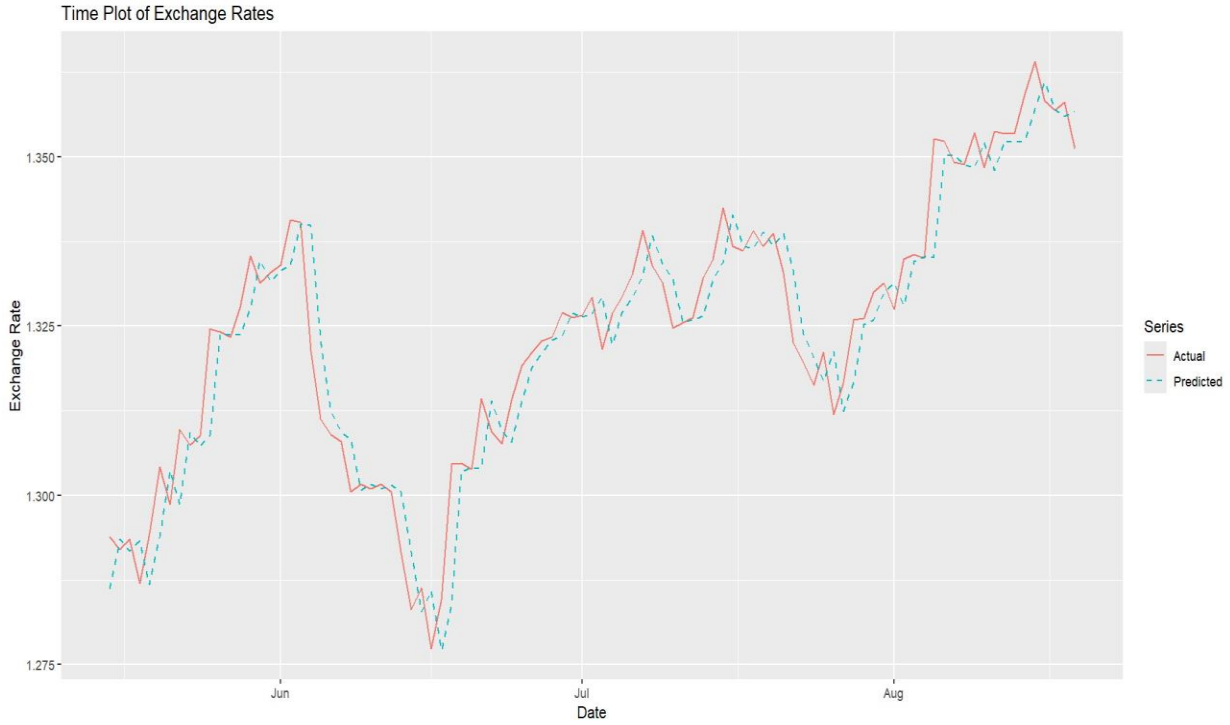

Figure 64: time plot of selected model(best model)

- This graph shows how exchange rates change over time for model predicted values and actual values.
- Both lines follow a similar pattern of ups and downs, indicating that the predictions are somewhat aligned with the actual results. This suggests that the model is performing well in general.

References

- Bhandari, P. (2024, January 17). *How to find outliers: 4 ways with examples & explanation*. Scribbr. https://www.scribbr.com/statistics/outliers/ [Accessed 7 April 2024]
- Chollet, F., & Allaire, J. J. (2017, December 20). *Time series forecasting with Recurrent Neural Networks*. Posit AI Blog. https://blogs.rstudio.com/ai/posts/2017-12-20-time-series-forecasting-with-recurrent-neural-networks/ [Accessed 15 April 2024]
- Fatih Ozturk, E. (2023, March 22). *Unsupervised learning in R: Determination of cluster number*. Medium. https://medium.com/@ozturkfemre/unsupervised-learning-determination-of-cluster-number-be8842cdb11 [Accessed 7 April 2024]
- GeeksforGeeks. (2023, December 13). *Autoregressive (AR) model for time series forecasting*. GeeksforGeeks. https://www.geeksforgeeks.org/autoregressive-ar-model-for-time-series-forecasting/ [Accessed 15 April 2024]
- guest blog (2020). Creating & Visualizing Neural Network in R. Analytics Vidhya. Available from https://www.analyticsvidhya.com/blog/2017/09/creating-visualizing-neural-network-in-r/ [Accessed 15 April 2024]
- *How to remove outliers from data in R*. Universe of Data Science. (2022, December 11). https://universeofdatascience.com/how-to-remove-outliers-from-data-in-r/#:~:text=In%20addition%2C%20we%20calculate%20Q1,()%20function%20to%20remove%20outliers.&text=Secondly%2C%20we%20use%20boxplot(),by%20using%20which()%20function. [Accessed 7 April 2024]
- John. (2020, January 19). *How to remove outliers in R: R-bloggers*. R. https://www.r-bloggers.com/2020/01/how-to-remove-outliers-in-r/ [Accessed 7 April 2024]
- Kumaratunga, S. (2020, May 11). *KMEANS hyper-parameters explained with examples*. Medium. https://towardsdatascience.com/kmeans-hyper-parameters-explained-with-examples-c93505820cd3 [Accessed 7 April 2024]
- Tiwari, A. (2020, August 13). *Let's forecast your time series using classical approaches*. Medium. https://towardsdatascience.com/lets-forecast-your-time-series-using-classical-approaches-f84eb982212c [Accessed 15 April 2024]
- Views, R. (2020, July 20). *Building a neural net from scratch using R - part 1*. · R Views. https://rviews.rstudio.com/2020/07/20/shallow-neural-net-from-scratch-using-r-part-1/ [Accessed 15 April 2024]

Appendix

- Part A subtask 1

```
# Install and load required packages packages <-
c("readxl", "tidyverse","factoextra", "NbClust")
# install.packages(packages)
library(readxl)
library(tidyverse)
library(tidyr)
# read the dataset
CW_Winedata <- read_excel('G:\\iit campus\\course\\2 ND YEAR\\2 sem\\Machine Learning and Data mining\\cw\\Whitewine_v6.xlsx')
```

```
# remove 12th column
CW_Winedata <- CW_Winedata[,1:11]
# check the dimensions
dim(CW_Winedata)
# get first 10 lines of the dataset
head(CW_Winedata,10) #
Before outlier removal
boxplot(CW_Winedata)


# -------Outlier removal with Interquartile Range method
# Define a function to remove outliers using the Interquartile Range method
Cw_out_iqr <- function(x) { q1 <- quantile(x, 0.25) q3
  <- quantile(x, 0.75) iqr <- q3 - q1 lower_bound <-
  q1 - 1.5 * iqr upper_bound <- q3 + 1.5 * iqr
  x[which(x < lower_bound | x > upper_bound)] <- NA
  return(x)
}

# Apply the function to each numerical column in the dataset
CW_Winedata <- as.data.frame(apply(CW_Winedata, 2, Cw_out_iqr))

# Remove rows with NA values
newCW_Winedata <- drop_na(CW_Winedata)

# Visualize boxplots after removing outliers
boxplot(newCW_Winedata)

# ------- Outlier removal using the box plot's statistical properties
CW_Winedata <- scale(CW_Winedata)
col = c('fixed acidity','volatile acidity','citric acid','residual sugar','chlorides','free sulfur dioxide','total sulfur
dioxide','density','pH','sulphates','alcohol') for (columnY in col)
{
  value = CW_Winedata[,columnY][CW_Winedata[,columnY] %in%
boxplot.stats(CW_Winedata[,columnY])$out]
  CW_Winedata[,columnY][CW_Winedata[,columnY] %in% value] = NA
}

as.data.frame(colSums(is.na(CW_Winedata)))

newCW_Winedata <- as.data.frame(CW_Winedata)
newCW_Winedata <- drop_na(newCW_Winedata)
as.data.frame(colSums(is.na(newCW_Winedata)))
boxplot(newCW_Winedata)
```

```
# scale if used iqr method newCW_Winedata
<- scale(newCW_Winedata)
boxplot(newCW_Winedata)

#--------- Determine the number of cluster centres
library(NbClust) library(factoextra) set.seed(10)
# elbow method fviz_nbclust(newCW_Winedata, kmeans,
method = "wss") +
  labs(subtitle = "Elbow method")
set.seed(10) # Silhouette method
fviz_nbclust(newCW_Winedata, kmeans, method = "silhouette")+
  labs(subtitle = "Silhouette method")
set.seed(10) # Gap statistic method
fviz_nbclust(newCW_Winedata, kmeans, method = "gap_stat")+
  labs(subtitle = "Gap statistic method")
set.seed(10)
# NBclust method
NbClust(newCW_Winedata, distance="euclidean", min.nc=2, max.nc=10, method="kmeans", index="all")
NbClust(newCW_Winedata, distance="manhattan", min.nc=2, max.nc=10, method="kmeans", index="all")
NbClust(newCW_Winedata, distance="maximum", min.nc=2, max.nc=10, method="kmeans", index="all")

# --- K- means clustering kmeans_data <-
kmeans(newCW_Winedata, centers = 2, nstart = 20)
# display the structure of object
str(kmeans_data)
# print the object
kmeans_data
# display clusters fviz_cluster(kmeans_data, data =
newCW_Winedata)
# display clusters fviz_cluster(kmeans_data, data = newCW_Winedata, ellipse.type = "euclid",star.plot =
TRUE, repel = TRUE, ggtheme = theme_minimal())


# get BSS and TSS from kmeans_data object
(BSS <- kmeans_data$betweenss)    # 5737.309
(TSS <- kmeans_data$totss)       # 24662
(WSS <- TSS - BSS)               # 18924.69
# Calculate the ratio of BSS over TSS
(BSS_TSS_ratio <- BSS / TSS * 100) # 23.26376%

# Create a data frame
df <- data.frame(
  Index = c("BSS", "WSS"),
  Value = c(BSS, WSS)
```

```
)

# Create a bar plot
library(ggplot2)
ggplot(df, aes(x=Index, y=Value, fill=Index)) +
  geom_bar(stat="identity") + theme_minimal() + labs(x="Index",
  y="Value", title="BSS and WSS Indices", fill="Index")

# silhouette plot analysis library("cluster") set.seed(10) # run
kmeans km_Resultsdata <- kmeans(newCW_Winedata, 2,
nstart = 10) fviz_cluster(km_Resultsdata, newCW_Winedata)
# calculate silhouette index sil_data <-
silhouette(km_Resultsdata$cluster, dist(newCW_Winedata))
fviz_silhouette(sil_data)
# negative silhouette width neg_sil_data <-
which(sil_data[, "sil_width"] < 0)
sil_data[neg_sil_data, , drop = FALSE] #
average silhouette information avg_sil_width <-
mean(sil_data[, 3]) # Evaluate the quality of the
clustering if (avg_sil_width > 0.7) {
  print("The clustering is reasonably good.")
} else if (avg_sil_width > 0.25) {
  print("The clustering is fair.")
} else { print("The clustering is
  poor.")
}

# run the Partitioning Around Medoids
pam.data <- pam(newCW_Winedata, 2)
fviz_cluster(pam.data)
fviz_silhouette(pam.data)
```

- Part A subtask 2

```
#Installing and loading the required packages and libraries
packages <- c("readxl", "tidyverse","gridExtra","ggcorrplot","dplyr")
#install.packages(packages)

library(readxl)
library(tidyr)
library(tidyverse)
library(ggplot2)
```

```
library(gridExtra)
library(ggcorrplot)
library(rlang)
library(factoextra)
library(NbClust)
library(cluster)

# read the dataset
CW_Winedata <- read_excel('G:\\iit campus\\course\\2 ND YEAR\\2 sem\\Machine Learning and Data
mining\\cw\\Whitewine_v6.xlsx')
# remove 12th column
CW_Winedata <- CW_Winedata[,1:11]
# scale the dataset
CW_Winedata <- scale(CW_Winedata)
# check if any missing values
colSums(is.na(CW_Winedata))

# calculate the Covariance Matrix
cw.cov <- cov(CW_Winedata)
ggcorrplot(cw.cov)


# Apply the built-in prcomp function to get results
CW_pca <- prcomp(CW_Winedata)
names(CW_pca)
summary(CW_pca)
# eigen values
CW_pca$sdev^2
# Principal Component Loadings/ eigen vectors
CW_pca$rotation
CW_pca$rotation <- -CW_pca$rotation # calculate cumulative
score of PCs c_score <- cumsum(CW_pca$sdev^2 /
sum(CW_pca$sdev^2)) c_score
# given that cumulative score is from 85% is selected
CW_pca_new <- which(c_score < 0.85)
CW_pca_new
# new transformed dataset with selected principal components
CW_pca_transform = as.data.frame(-CW_pca$x[,1:7])
CW_pca_transform

# Disscussion--# scree plot
fviz_eig(CW_pca, addlabels = TRUE)
# variables of the PCA
fviz_pca_var(CW_pca, col.var = "blue")
# barplot
```

```
var_precent <- (CW_pca$sdev^2 / sum(CW_pca$sdev^2))*100
barplot(var_precent, xlab='PC', ylab='Percent Variance',
names.arg=1:length(var_precent), las=1, ylim=c(0,
max(var_precent)), col='gray') # biplot biplot(CW_pca, scale =
0)

#--------- Determine the number of cluster centres
set.seed(10)
# elbow method fviz_nbclust(CW_pca_transform, kmeans,
method = "wss") +
  labs(subtitle = "Elbow method")
set.seed(10) # Silhouette method
fviz_nbclust(CW_pca_transform, kmeans, method = "silhouette")+
  labs(subtitle = "Silhouette method")
set.seed(10) # Gap statistic method
fviz_nbclust(CW_pca_transform, kmeans, method = "gap_stat")+
  labs(subtitle = "Gap statistic method")
set.seed(10)
# NBclust method
NbClust(CW_pca_transform, distance="euclidean", min.nc=2, max.nc=10, method="kmeans", index="all")
NbClust(CW_pca_transform, distance="manhattan", min.nc=2, max.nc=10, method="kmeans", index="all")
NbClust(CW_pca_transform, distance="maximum", min.nc=2, max.nc=10, method="kmeans", index="all")

# kmeans kmeans_data <- kmeans(CW_pca_transform, centers = 2,
nstart = 20)
# display the structure of object str(kmeans_data) #
print the object kmeans_data # display clusters
fviz_cluster(kmeans_data, data = CW_pca_transform)
# display clusters fviz_cluster(kmeans_data, data = CW_pca_transform, ellipse.type = "euclid",star.plot =
TRUE, repel = TRUE, ggtheme = theme_minimal())


# get BSS and TSS from kmeans_data object
(BSS <- kmeans_data$betweenss)
(TSS <- kmeans_data$totss)
(WSS <- TSS - BSS)
# Calculate the ratio of BSS over TSS
(BSS_TSS_ratio <- BSS / TSS * 100)

# silhouette plot analyis set.seed(10) # run kmeans
km_data <- kmeans(CW_pca_transform, 2, nstart = 10)
fviz_cluster(km_data, CW_pca_transform)
# calculate silhouette index sil_data <-
silhouette(km_data$cluster, dist(CW_pca_transform))
fviz_silhouette(sil_data)
```

```
# negative silhouette width neg_sil_data <-
which(sil_data[, "sil_width"] < 0)
sil_data[neg_sil_data, , drop = FALSE] #
average silhouette information
avg_sil_width <- round(mean(sil_data[, 3]),2)
# Evaluate the quality of the clustering if
(avg_sil_width > 0.7) {
  print("The clustering is reasonably good.")
} else if (avg_sil_width >= 0.25) {
  print("The clustering is fair.")
} else { print("The clustering is
  poor.")
}


# Calculate Calinski-Harabasz Index
CH_metric <- function(cluster_obj, data) {
  num_clusters <- length(unique(cluster_obj$cluster))
  num_Rows <- nrow(data) BSS <-
  cluster_obj$betweenss
  WSS <- cluster_obj$tot.withinss
  CH_index <- ((num_Rows - num_clusters) / (num_clusters - 1)) * (BSS / WSS)
  return(CH_index)
}
ch_index <- CH_metric(km_data, CW_pca_transform)
ch_index

# Function to calculate CH index for a given number of clusters
CH_metric <- function(data, num_clusters) {
  cluster_obj <- kmeans(data, centers = num_clusters)
  num_Rows <- nrow(data) BSS <-
  cluster_obj$betweenss
  WSS <- cluster_obj$tot.withinss
  CH_index <- ((num_Rows - num_clusters) / (num_clusters - 1)) * (BSS / WSS)
  return(CH_index)
}

# Function to calculate CH index for a range of cluster numbers
ch_values <- function(data, kmax) {
  ch_indices <- sapply(2:kmax, function(k) {
    CH_metric(data, k)
  })
  return(ch_indices)
}
```

```
# Calculate CH indices ch_indices <-
ch_values(CW_pca_transform, 10)

# Plot CH indices barplot(ch_indices, names.arg = 2:10, xlab = "Number of clusters",
ylab = "CH index", main = "CH index for different numbers of clusters", col =
"lightblue")
```

- Part B

```
library(readxl)
library(neuralnet)
library(Metrics)
library(ggplot2)

CW_exchangeData = read_excel('G:\\iit campus\\course\\2 ND YEAR\\2 sem\\Machine Learning and Data
mining\\cw\\ExchangeUSD.xlsx')

# scaling functions
normalization = function(x){
  return((x-min(x))/(max(x)-min(x)))
} unNormalization = function(x, min, max)
{
  return( (max - min)*x + min )
}

# select USD/EUR column
CW_exchangeData = CW_exchangeData[,3]
head(CW_exchangeData)


##### Various input vectors up to (t-4) level.
CW_exchangeData_Tl4 = bind_cols(t_4 = lag(CW_exchangeData, 4),
                        t_3 = lag(CW_exchangeData, 3), t_2 =
                        lag(CW_exchangeData, 2), t_1 =
                        lag(CW_exchangeData, 1), t =
                        `CW_exchangeData`)
# NA values removal and structure
CW_exchangeData_Tl4 = CW_exchangeData_Tl4[complete.cases(CW_exchangeData_Tl4),]
colnames(CW_exchangeData_Tl4) = c("InputV1","InputV2","InputV3","InputV4","Output")

# training and testing matrices trainMat_Tl4 =
CW_exchangeData_Tl4[1:400,] testMat_Tl4 =
CW_exchangeData_Tl4[401:nrow(CW_exchangeData_Tl4),]
```

```
# normalizing the i/o metrics minV_Tl4 = min(trainMat_Tl4) maxV_Tl4 =
max(trainMat_Tl4) normData_Tl4 =
as.data.frame(lapply(CW_exchangeData_Tl4, normalization))

# spliting normalize data into train and test sets
norm_trainMat_Tl4 = normData_Tl4[1:400,] norm_testMat_Tl4 =
normData_Tl4[401:nrow(CW_exchangeData_Tl4),]

# Hyperparameter tuning ---------------------------------------
# 1st model V(1:4) with one hidden layer nn4_1 <- neuralnet(Output ~ InputV1 + InputV2 +
InputV3 + InputV4, data = norm_trainMat_Tl4,
                hidden = 5, linear.output = TRUE, act.fct
                ="logistic")

# Compute predictions on the test set modelTl4_h1_5 <-
neuralnet::compute(nn4_1, norm_testMat_Tl4)
# normalization of model result predictionTl4_5 <-
unNormalization(modelTl4_h1_5$net.result, minV_Tl4, maxV_Tl4)

# Testing performance
(rmseTl4_h1_5 = rmse(testMat_Tl4$Output, predictionTl4_5))
(maeTl4_h1_5 = mae(testMat_Tl4$Output, predictionTl4_5))
(mapeTl4_h1_5 = mape(testMat_Tl4$Output, predictionTl4_5))
(smapeTl4_h1_5 = smape(testMat_Tl4$Output, predictionTl4_5))



# 2nd model V(1:4) with two hidden layer nn4_2 <- neuralnet(Output ~ InputV1 + InputV2 +
InputV3 + InputV4, data = norm_trainMat_Tl4,
                hidden = c(5,2), linear.output = TRUE,
                act.fct ="logistic")

# Compute predictions on the test set modelTl4_h2_7 <- neuralnet::compute(nn4_2,
norm_testMat_Tl4) predictionTl4_7 <- unNormalization(modelTl4_h2_7$net.result,
minV_Tl4, maxV_Tl4)

(rmseTl4_h2_7 = rmse(testMat_Tl4$Output, predictionTl4_7))
(maeTl4_h2_7 = mae(testMat_Tl4$Output, predictionTl4_7))
(mapeTl4_h2_7 = mape(testMat_Tl4$Output, predictionTl4_7))
(smapeTl4_h2_7 = smape(testMat_Tl4$Output, predictionTl4_7))

# 3rd model V(1:4) with two hidden layer nn4_3 <- neuralnet(Output ~ InputV1 + InputV2 +
InputV3 + InputV4, data = norm_trainMat_Tl4,
```

```
                hidden = c(10,7), linear.output = TRUE,
                act.fct ="logistic")

# Compute predictions on the test set modelTl4_h2_17 <- neuralnet::compute(nn4_3,
norm_testMat_Tl4) predictionTl4_17 <- unNormalization(modelTl4_h2_17$net.result,
minV_Tl4, maxV_Tl4)

(rmseTl4_h2_17 = rmse(testMat_Tl4$Output, predictionTl4_17))
(maeTl4_h2_17 = mae(testMat_Tl4$Output, predictionTl4_17))
(mapeTl4_h2_17 = mape(testMat_Tl4$Output, predictionTl4_17))
(smapeTl4_h2_17 = smape(testMat_Tl4$Output, predictionTl4_17))


##### Various input vectors up to (t-3) level.
CW_exchangeData_Tl3 = bind_cols( t_3 = lag(CW_exchangeData, 3),
                        t_2 = lag(CW_exchangeData, 2), t_1 =
                        lag(CW_exchangeData, 1), t =
                        `CW_exchangeData`)
# NA values removal and structure
CW_exchangeData_Tl3 = CW_exchangeData_Tl3[complete.cases(CW_exchangeData_Tl3),]
colnames(CW_exchangeData_Tl3) = c("InputV1","InputV2","InputV3","Output")

# training and testing matrices trainMat_Tl3 =
CW_exchangeData_Tl3[1:400,] head(trainMat_Tl3) testMat_Tl3 =
CW_exchangeData_Tl3[401:nrow(CW_exchangeData_Tl3),]

# normalizing the i/o metrics minV_Tl3 = min(trainMat_Tl3) maxV_Tl3 =
max(trainMat_Tl3) normData_Tl3 =
as.data.frame(lapply(CW_exchangeData_Tl3, normalization))

# spliting normalize data into train and test sets norm_trainMat_Tl3 =
normData_Tl3[1:400,] head(norm_trainMat_Tl3) norm_testMat_Tl3 =
normData_Tl3[401:nrow(CW_exchangeData_Tl3),]
head(norm_testMat_Tl3)

# Hyperparameter tuning ---------------------------------------
# 1st model V(1:3) with one hidden layer nn3_1 <- neuralnet(Output ~ InputV1 +
InputV2 + InputV3, data = norm_trainMat_Tl3, hidden = 5, linear.output = TRUE,
act.fct ="logistic")

# Compute predictions on the test set modelTl3_h1_5 <- neuralnet::compute(nn3_1,
norm_testMat_Tl3) predictionTl3_5 <- unNormalization(modelTl3_h1_5$net.result,
minV_Tl3, maxV_Tl3)

# Testing performance
```

```
(rmseTl3_h1_5 = rmse(testMat_Tl3$Output, predictionTl3_5))
(maeTl3_h1_5 = mae(testMat_Tl3$Output, predictionTl3_5))
(mapeTl3_h1_5 = mape(testMat_Tl3$Output, predictionTl3_5))
(smapeTl3_h1_5 = smape(testMat_Tl3$Output, predictionTl3_5))

# 2nd model V(1:3) with two hidden layer nn3_2 <- neuralnet(Output ~ InputV1 +
InputV2 + InputV3, data = norm_trainMat_Tl3,
                hidden = c(5,2),
                linear.output = TRUE, act.fct ="logistic")

# Compute predictions on the test set modelTl3_h2_7 <- neuralnet::compute(nn3_2,
norm_testMat_Tl3) predictionTl3_7 <- unNormalization(modelTl3_h2_7$net.result,
minV_Tl3, maxV_Tl3)

# Testing performance
(rmseTl3_h2_7 = rmse(testMat_Tl3$Output, predictionTl3_7))
(maeTl3_h2_7 = mae(testMat_Tl3$Output, predictionTl3_7))
(mapeTl3_h2_7 = mape(testMat_Tl3$Output, predictionTl3_7))
(smapeTl3_h2_7 = smape(testMat_Tl3$Output, predictionTl3_7))

# 3rd model V(1:3) with two hidden layer nn3_3 <- neuralnet(Output ~ InputV1 +
InputV2 + InputV3, data = norm_trainMat_Tl3,
                hidden = c(10,7), linear.output = TRUE,
                act.fct ="logistic")

# Compute predictions on the test set modelTl3_h2_17 <- neuralnet::compute(nn3_3,
norm_testMat_Tl3) predictionTl3_17 <- unNormalization(modelTl3_h2_17$net.result,
minV_Tl3, maxV_Tl3)

# Testing performance
(rmseTl3_h2_17 = rmse(testMat_Tl3$Output, predictionTl3_17))
(maeTl3_h2_17 = mae(testMat_Tl3$Output, predictionTl3_17))
(mapeTl3_h2_17 = mape(testMat_Tl3$Output, predictionTl3_17))
(smapeTl3_h2_17 = smape(testMat_Tl3$Output, predictionTl3_17))


##### Various input vectors up to (t-2) level.
CW_exchangeData_Tl2 = bind_cols(t_2 = lag(CW_exchangeData, 2),
                        t_1 = lag(CW_exchangeData, 1), t =
                        `CW_exchangeData`)
# NA values removal and structure
CW_exchangeData_Tl2 = CW_exchangeData_Tl2[complete.cases(CW_exchangeData_Tl2),]
colnames(CW_exchangeData_Tl2) = c("InputV1","InputV2","Output")
```

```
# training and testing matrices trainMat_Tl2 =
CW_exchangeData_Tl2[1:400,] head(trainMat_Tl2) testMat_Tl2 =
CW_exchangeData_Tl2[401:nrow(CW_exchangeData_Tl2),]

# normalizing the i/o metrics minV_Tl2 = min(trainMat_Tl2) maxV_Tl2 =
max(trainMat_Tl2) normData_Tl2 =
as.data.frame(lapply(CW_exchangeData_Tl2, normalization))

# spliting normalize data into train and test sets
norm_trainMat_Tl2 = normData_Tl2[1:400,] head(norm_trainMat_Tl2)
norm_testMat_Tl2 =
normData_Tl2[401:nrow(CW_exchangeData_Tl2),]
head(norm_testMat_Tl2)

# Hyperparameter tuning ---------------------------------------
# 1st model V(1:2) with one hidden layer nn2_1 <- neuralnet(Output ~
InputV1 + InputV2 , data = norm_trainMat_Tl2, hidden = 5, linear.output =
TRUE, act.fct ="logistic")

# Compute predictions on the test set modelTl2_h1_5 <- neuralnet::compute(nn2_1,
norm_testMat_Tl2) predictionTl2_5 <- unNormalization(modelTl2_h1_5$net.result,
minV_Tl2, maxV_Tl2)

# Testing performance
(rmseTl2_h1_5 = rmse(testMat_Tl2$Output, predictionTl2_5))
(maeTl2_h1_5 = mae(testMat_Tl2$Output, predictionTl2_5))
(mapeTl2_h1_5 = mape(testMat_Tl2$Output, predictionTl2_5))
(smapeTl2_h1_5 = smape(testMat_Tl2$Output, predictionTl2_5))

# 2nd model V(1:2) with two hidden layer nn2_2 <- neuralnet(Output ~
InputV1 + InputV2, data = norm_trainMat_Tl2,
                hidden = c(5,2), linear.output = TRUE,
                act.fct ="logistic")

# Compute predictions on the test set modelTl2_h2_7 <- neuralnet::compute(nn2_2,
norm_testMat_Tl2) predictionTl2_7 <- unNormalization(modelTl2_h2_7$net.result,
minV_Tl2, maxV_Tl2)

# Testing performance
(rmseTl2_h2_7 = rmse(testMat_Tl2$Output, predictionTl2_7))
(maeTl2_h2_7 = mae(testMat_Tl2$Output, predictionTl2_7))
(mapeTl2_h2_7 = mape(testMat_Tl2$Output, predictionTl2_7))
(smapeTl2_h2_7 = smape(testMat_Tl2$Output, predictionTl2_7))
```

```
# 3rd model V(1:2) with two hidden layer nn2_3 <- neuralnet(Output ~
InputV1 + InputV2, data = norm_trainMat_Tl2,
                hidden = c(10,7), linear.output = TRUE,
                act.fct ="logistic")

# Compute predictions on the test set modelTl2_h2_17 <- neuralnet::compute(nn2_3,
norm_testMat_Tl2) predictionTl2_17 <- unNormalization(modelTl2_h2_17$net.result,
minV_Tl2, maxV_Tl2)

# Testing performance
(rmseTl2_h2_17 = rmse(testMat_Tl2$Output, predictionTl2_17))
(maeTl2_h2_17 = mae(testMat_Tl2$Output, predictionTl2_17))
(mapeTl2_h2_17 = mape(testMat_Tl2$Output, predictionTl2_17))
(smapeTl2_h2_17 = smape(testMat_Tl2$Output, predictionTl2_17))

##### Various input vectors up to (t-1) level
CW_exchangeData_Tl1 = bind_cols(t_1 = lag(CW_exchangeData, 1),
                        t = `CW_exchangeData`)
# NA values removal and structure
CW_exchangeData_Tl1 = CW_exchangeData_Tl1[complete.cases(CW_exchangeData_Tl1),]
colnames(CW_exchangeData_Tl1) = c("InputV1","Output")

# training and testing matrices trainMat_Tl1 =
CW_exchangeData_Tl1[1:400,] head(trainMat_Tl2) testMat_Tl1 =
CW_exchangeData_Tl1[401:nrow(CW_exchangeData_Tl1),]

# normalizing the i/o metrics minV_Tl1 = min(trainMat_Tl1) maxV_Tl1 =
max(trainMat_Tl1) normData_Tl1 =
as.data.frame(lapply(CW_exchangeData_Tl1, normalization))

# spliting normalize data into train and test sets norm_trainMat_Tl1 =
normData_Tl1[1:400,] head(norm_trainMat_Tl1) norm_testMat_Tl1 =
normData_Tl1[401:nrow(CW_exchangeData_Tl1),]
head(norm_testMat_Tl1)

# 1st model V(1:1) with one hidden layer nn1_1 <-
neuralnet(Output ~ InputV1 , data = norm_trainMat_Tl1, hidden =
5, linear.output = TRUE, act.fct ="logistic")

# Compute predictions on the test set modelTl1_h1_5 <- neuralnet::compute(nn1_1,
norm_testMat_Tl1) predictionTl1_5 <- unNormalization(modelTl1_h1_5$net.result,
minV_Tl1, maxV_Tl1)

# Testing performance
(rmseTl1_h1_5 = rmse(testMat_Tl1$Output, predictionTl1_5))
```

```
(maeTl1_h1_5 = mae(testMat_Tl1$Output, predictionTl1_5))
(mapeTl1_h1_5 = mape(testMat_Tl1$Output, predictionTl1_5))
(smapeTl1_h1_5 = smape(testMat_Tl1$Output, predictionTl1_5))

# 2nd model V(1:1) with two hidden layer nn1_2 <-
neuralnet(Output ~ InputV1, data = norm_trainMat_Tl1,
                hidden = c(5,2), linear.output = TRUE,
                act.fct ="logistic") # Compute
                predictions on the test set
                modelTl1_h2_7 <-
                neuralnet::compute(nn1_2,
                norm_testMat_Tl1) predictionTl1_7 <-
                unNormalization(modelTl1_h2_7$net.re
                sult, minV_Tl1, maxV_Tl1)

# Testing performance
(rmseTl1_h2_7 = rmse(testMat_Tl1$Output, predictionTl1_7))
(maeTl1_h2_7 = mae(testMat_Tl1$Output, predictionTl1_7))
(mapeTl1_h2_7 = mape(testMat_Tl1$Output, predictionTl1_7))
(smapeTl1_h2_7 = smape(testMat_Tl1$Output, predictionTl1_7))

# 3rd model V(1:1) with two hidden layer nn1_3 <-
neuralnet(Output ~ InputV1, data = norm_trainMat_Tl1,
                hidden = c(10,7), linear.output = TRUE,
                act.fct ="logistic")

# Compute predictions on the test set modelTl1_h2_17 <- neuralnet::compute(nn1_3,
norm_testMat_Tl1) predictionTl1_17 <- unNormalization(modelTl1_h2_17$net.result,
minV_Tl1, maxV_Tl1)

# Testing performance
(rmseTl1_h2_17 = rmse(testMat_Tl1$Output, predictionTl1_17))
(maeTl1_h2_17 = mae(testMat_Tl1$Output, predictionTl1_17))
(mapeTl1_h2_17 = mape(testMat_Tl1$Output, predictionTl1_17))
(smapeTl1_h2_17 = smape(testMat_Tl1$Output, predictionTl1_17))



# all performance evaluation rmse_values = c(rmseTl1_h1_5, rmseTl2_h1_5,
rmseTl3_h1_5, rmseTl4_h1_5, rmseTl1_h2_7, rmseTl2_h2_7, rmseTl3_h2_7,
rmseTl4_h2_7, rmseTl1_h2_17, rmseTl2_h2_17, rmseTl3_h2_17,
rmseTl4_h2_17)

mae_values = c(maeTl1_h1_5, maeTl2_h1_5, maeTl3_h1_5, maeTl4_h1_5,
        maeTl1_h2_7, maeTl2_h2_7, maeTl3_h2_7, maeTl4_h2_7,
        maeTl1_h2_17,maeTl2_h2_17, maeTl3_h2_17, maeTl4_h2_17)
```

```
mape_values = c(mapeTl1_h1_5, mapeTl2_h1_5, mapeTl3_h1_5, mapeTl4_h1_5,
          mapeTl1_h2_7, mapeTl2_h2_7, mapeTl3_h2_7, mapeTl4_h2_7,
          mapeTl1_h2_17,mapeTl2_h2_17, mapeTl3_h2_17, mapeTl4_h2_17)

smape_values = c(smapeTl1_h1_5, smapeTl2_h1_5, smapeTl3_h1_5, smapeTl4_h1_5,
           smapeTl1_h2_7, smapeTl2_h2_7, smapeTl3_h2_7, smapeTl4_h2_7,
           smapeTl1_h2_17,smapeTl2_h2_17, smapeTl3_h2_17, smapeTl4_h2_17)
# all plots of nueral net models
plot(nn1_1) plot(nn1_2)
plot(nn1_3) plot(nn2_1)
plot(nn2_2)
plot(nn2_3)
plot(nn3_1)
plot(nn3_2)
plot(nn3_3)
plot(nn4_1)
plot(nn4_2)
plot(nn4_3) #
Comparison table
comparison_Table = data.frame(Input_nodes = c("1","2","3","4","1","2","3","4","1","2","3","4"),
                        Hidden_layers = c("1","1","1","1","2","2","2","2","2","2","2","2"),
                        HIdden_nodes = c("5","5","5","5", "5-2", "5-2", "5-2", "5-2", "10-7", "10-7", "10-7", "10-7"),
                        RMSE = rmse_values,
                        MAE = mae_values,
                        MAPE = mape_values,
                        SMAPE = smape_values)
comparison_Table


# Evaluate the most accurate model
# plot for the selected model
plot(nn2_1) plot(nn3_3)
# Actual and Predicted Exchange Rates Time plot for the model
exchange_ratesPlot <- data.frame(
  DateSeq = seq(as.Date("2013-05-15"), by = "day", length.out = length(testMat_Tl3$Output)),
  ActualRate = testMat_Tl3$Output,
  PredictedRate = predictionTl3_17
)

ggplot(exchange_ratesPlot, aes(x = DateSeq)) +
  geom_line(aes(y = ActualRate, color = "Actual")) + geom_line(aes(y =
  PredictedRate, color = "Predicted"), linetype = "dashed") + labs(x = "Date", y
  = "Exchange Rate", color = "Series") + ggtitle("Time Plot of Exchange
  Rates")
```

```
# Actual and Predicted Exchange Rates scatter plot for the model
exchange_ScatPlot <- data.frame(
  ActualRate = testMat_Tl3$Output,
  PredictedRate = predictionTl3_17
)

ggplot(exchange_ScatPlot, aes(x = ActualRate, y = PredictedRate)) +
  geom_point() + geom_abline(intercept = 0, slope = 1, color = "red",
  linetype = "dashed") + labs(x = "Actual Exchange Rate", y = "Predicted ")
  + ggtitle("Scatter Plot of Exchange Rates")
```