Informatics Institute of Technology

Department of Computing

5DATA001C.2 Machine Learning and Data Mining CW Report

Name : Hasni haleemdeen

UoW ID : w1898945

IIT ID : 20211337

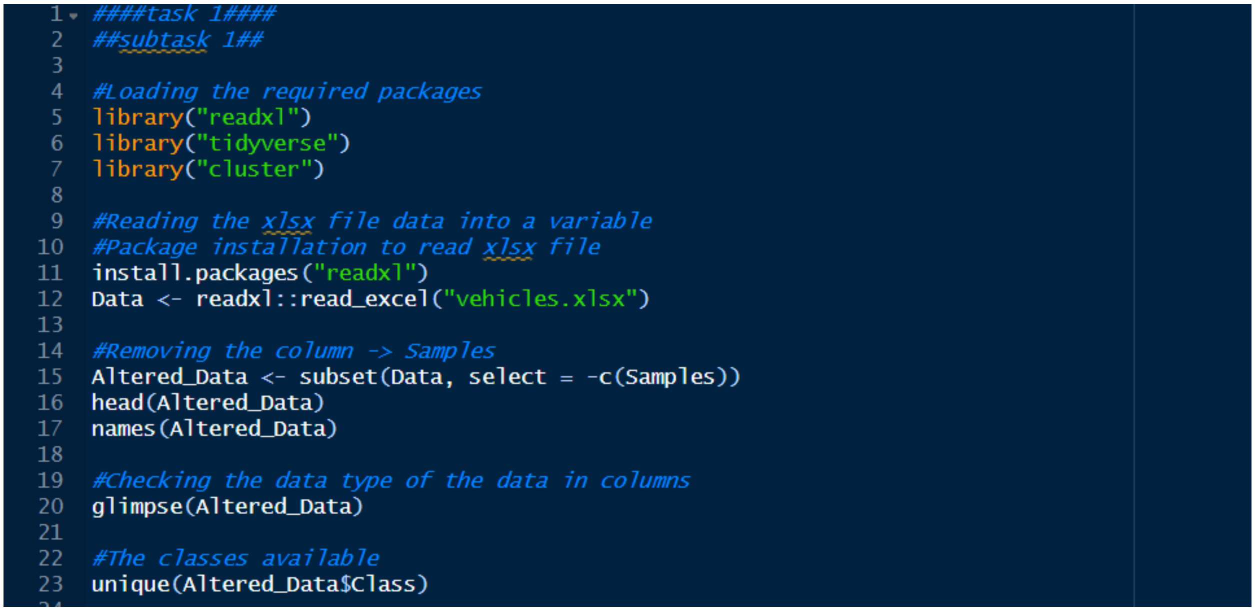
Contents

Partitioning Clustering Part

We are required to cluster input variables in this assignment based on how similar they are without having access to class labels or outputs. The task entails examining a dataset that contains the characteristics of four different kinds of vehicles, including an Opel Manta, a Chevrolet van, a Double Decker bus, and a Saab car. The goal is to locate the various clusters in the data. Three types of vehicle class labels are included in the dataset: bus, van, and car.

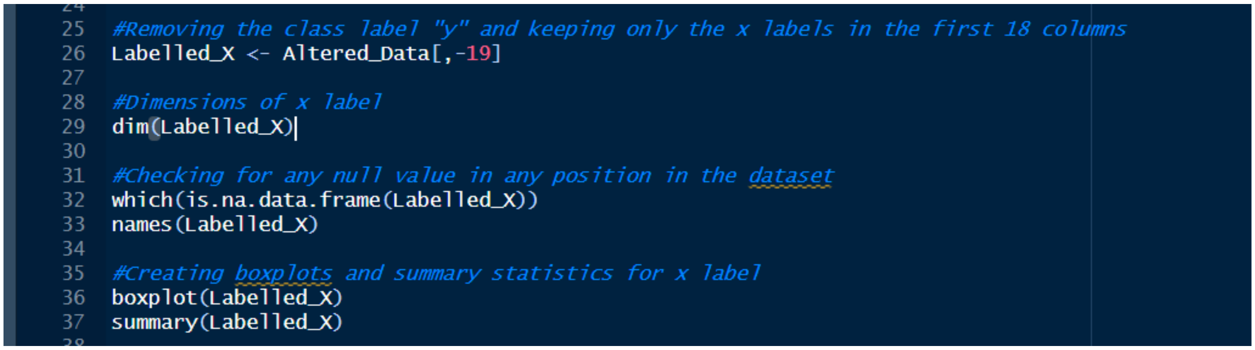
A).Pre-processing and outlier removal.

Since the "Samples" column only provided the dataset's row count and was not a feature that could help identify clusters, it was initially removed during the pre-processing stage.



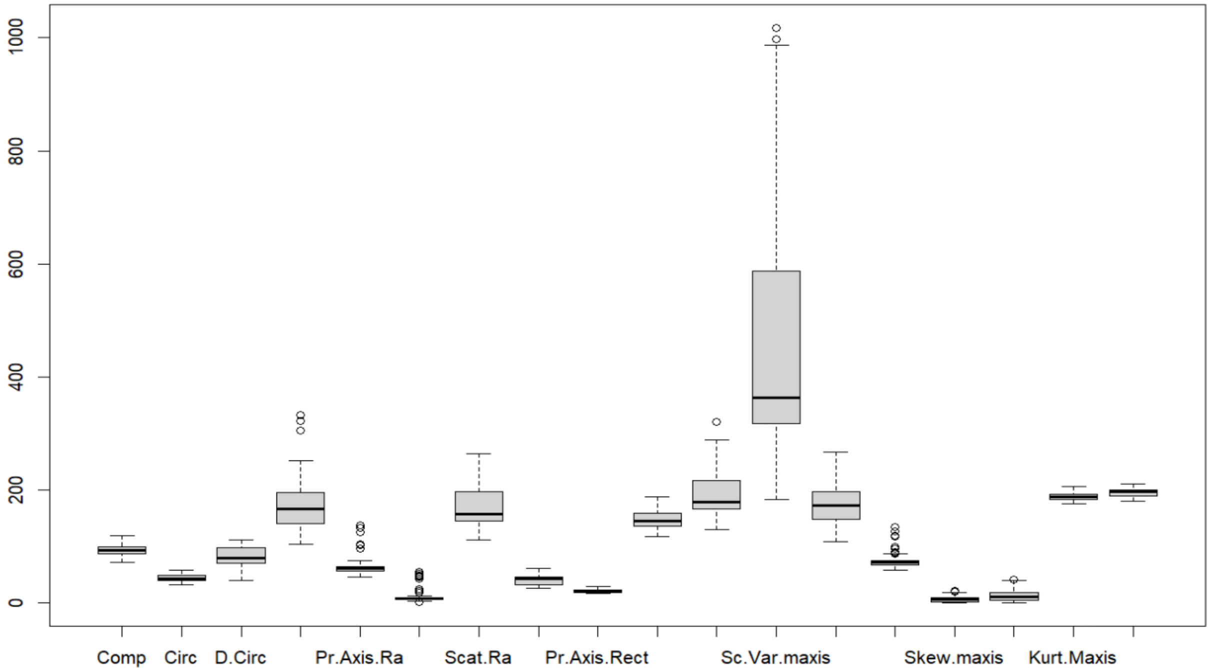
```
####task 1####
##subtask 1##

#Loading the required packages
library("readxl")
library("tidyverse")
library("cluster")

#Reading the xlsx file data into a variable
#Package installation to read xlsx file
install.packages("readxl")
Data <- readxl::read_excel("vehicles.xlsx")

#Removing the column -> Samples
Altered_Data <- subset(Data, select = -c(Samples))
head(Altered_Data)
names(Altered_Data)

#Checking the data type of the data in columns
glimpse(Altered_Data)

#The classes available
unique(Altered_Data$Class)
```

The "vehicles.xlsx" Excel file that contains the data is read and stored using the code that was mentioned above. The "Samples" column has been taken out of the dataset and put into its own variable. Since the goal of clustering is to identify groups or labels based only on input features, the class column (label y) was also removed from the dataset as according to the code picture below.

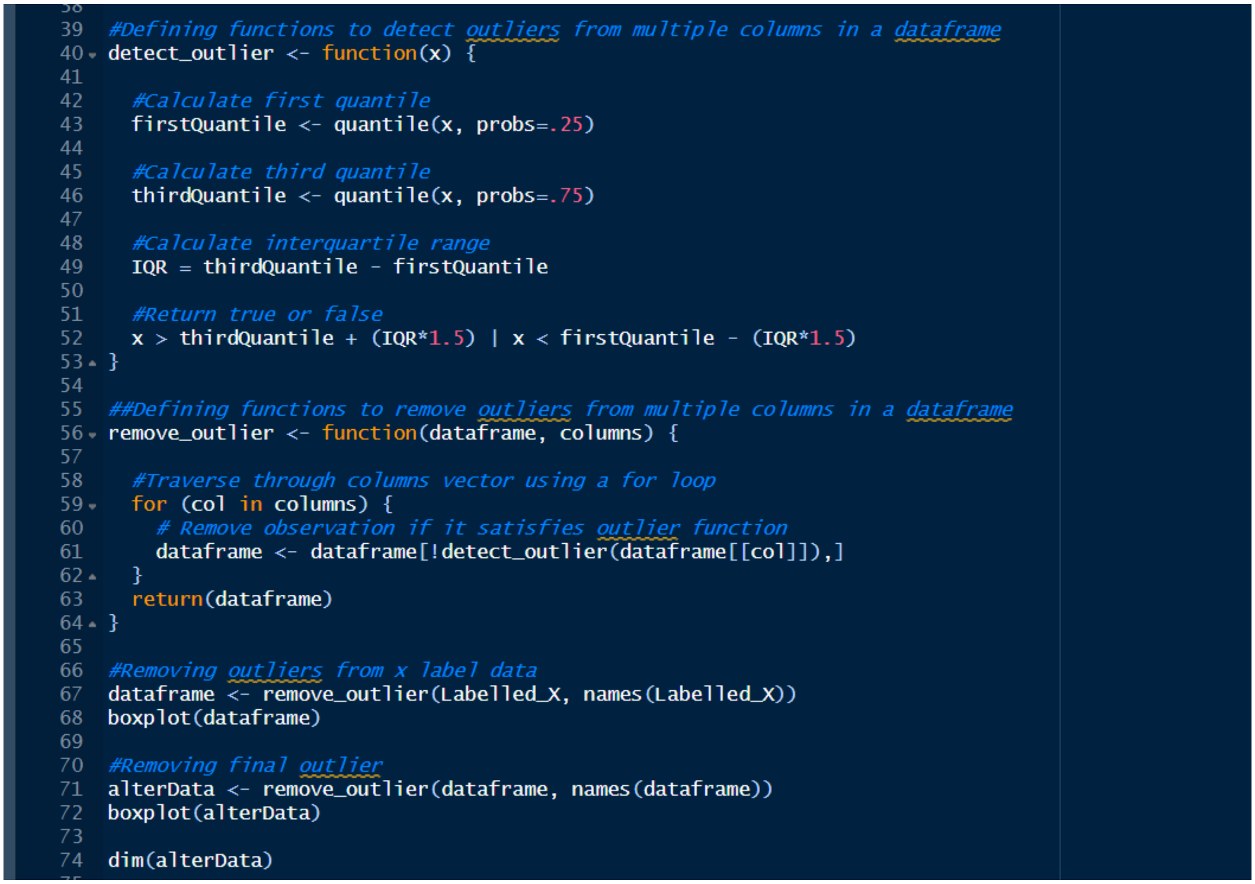


```
#Removing the class label "y" and keeping only the x labels in the first 18 columns
Labelled_X <- Altered_Data[,-19]

#Dimensions of x label
dim(Labelled_X)

#Checking for any null value in any position in the dataset
which(is.na.data.frame(Labelled_X))
names(Labelled_X)

#Creating boxplots and summary statistics for x label
boxplot(Labelled_X)
summary(Labelled_X)
```

The dataset dimensions were discovered to be 846 rows and 18 columns after the y label was removed. The dataset was also checked to see if there were any missing values, and it was found that there were none (null values). The dataset was then examined for outliers using a boxplot visualization as the next step. This made it possible to find any outliers in the data.

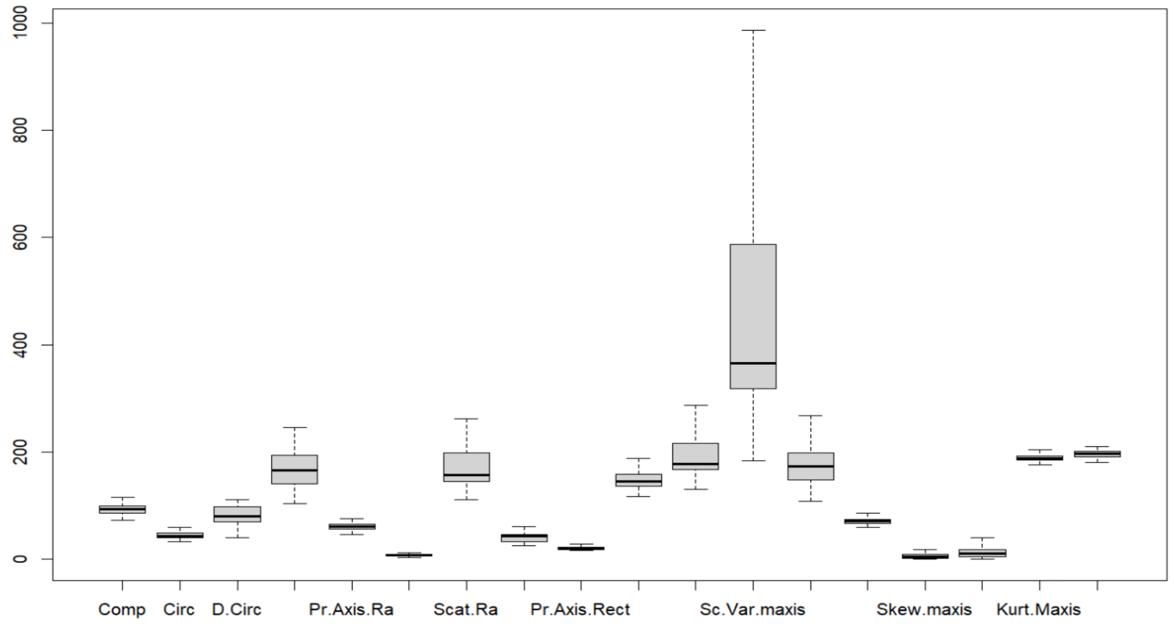


The dataset makes it clear that there are outliers. The interquartile range (IQR) method, which involves removing any data points that fall outside the range, was used to address this.



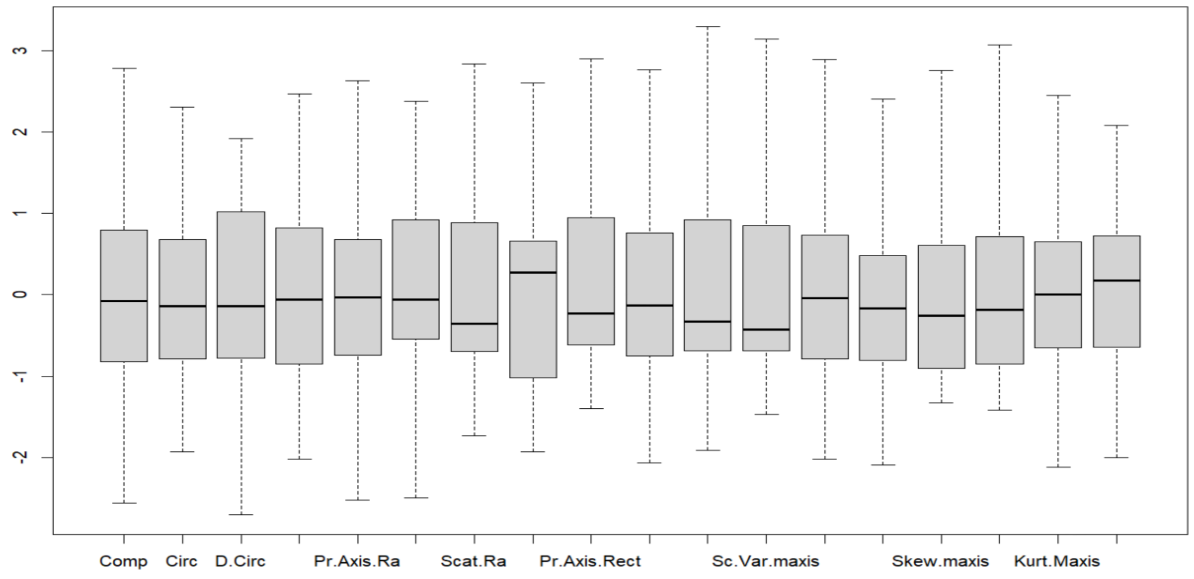
```
#Defining functions to detect outliers from multiple columns in a dataframe
detect_outlier <- function(x) {

  #Calculate first quantile
  firstQuantile <- quantile(x, probs=.25)

  #Calculate third quantile
  thirdQuantile <- quantile(x, probs=.75)

  #Calculate interquartile range
  IQR = thirdQuantile - firstQuantile

  #Return true or false
  x > thirdQuantile + (IQR*1.5) | x < firstQuantile - (IQR*1.5)
}

##Defining functions to remove outliers from multiple columns in a dataframe
remove_outlier <- function(dataframe, columns) {

  #Traverse through columns vector using a for loop
  for (col in columns) {
    # Remove observation if it satisfies outlier function
    dataframe <- dataframe[!detect_outlier(dataframe[[col]]),]
  }
  return(dataframe)
}

#Removing outliers from x label data
dataframe <- remove_outlier(Labelled_X, names(Labelled_X))
boxplot(dataframe)

#Removing final outlier
alterData <- remove_outlier(dataframe, names(dataframe))
boxplot(alterData)

dim(alterData)
```

The above-mentioned code is used to remove all outliers from the dataset. By examining the first and third quartiles, the process determines whether each data point is an outlier. If the third quartile of a data point is greater than 1.5 times the interquartile range, or if the first quartile is less than 1.5 times the interquartile range, the data point is considered an outlier and is removed. After the outliers were removed, the dataset dimensions were discovered to be 813 rows and 18 columns.



The dataset is shown in the box plot above after the outliers have been taken out. The Z-score method was then used to standardize the dataset after this step. The choice of this approach was made in order to help the data be centered around the mean.

As you can see from the boxplot below, all the features have been brought within a range after normalization. After the preprocessing stage was successfully finished, the data was prepared for clustering.



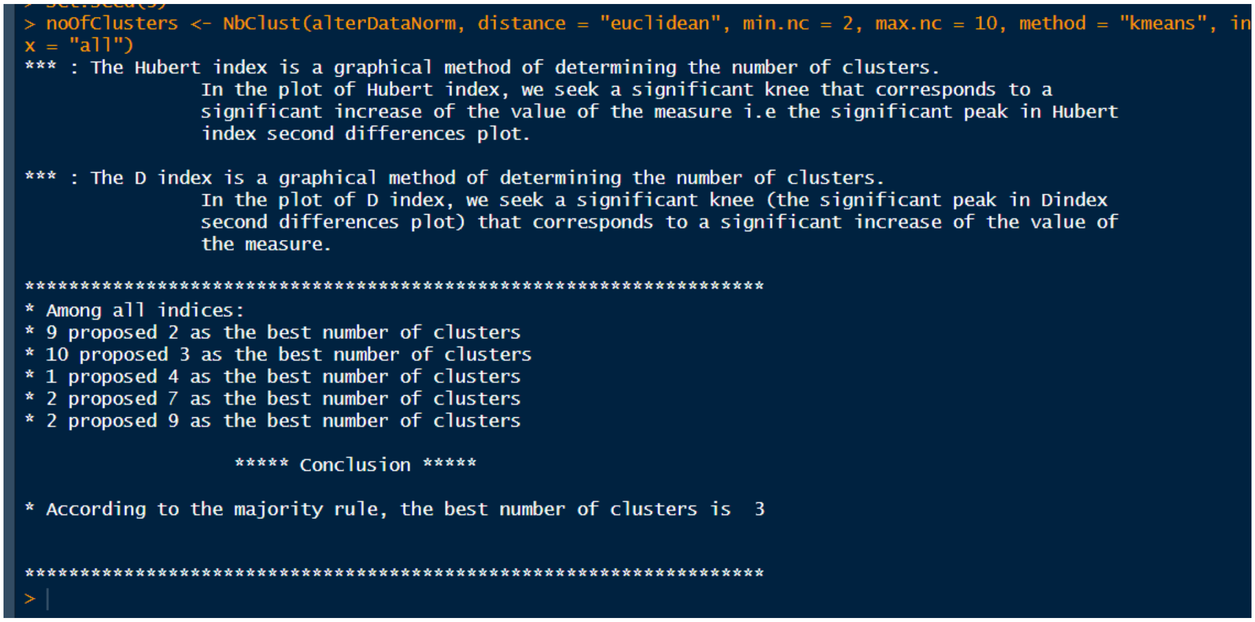
B).Using Automated tools.

Each clustering tool produced a different set of metrics and visualizations that could be used to determine the ideal number of clusters for the dataset. To determine the optimal number of clusters for the dataset, these metrics and visualizations were compared and further examined. The results of each clustering method are summarized in the sections that follow.

NBClust

Popular automated clustering tool NBClust determines the ideal number of clusters for a given dataset by combining several different clustering algorithms. It generates results for various clustering techniques, including k-means, hierarchical clustering, and model-based clustering, and computes a number of internal clustering validation measures. The measurements comprise, among others, the silhouette width, the Calinski-Harabasz index, and the within-cluster sum of squares. Based on these metrics, NBClust can assess the clustering quality objectively and assist in deciding how many clusters are best for the dataset.

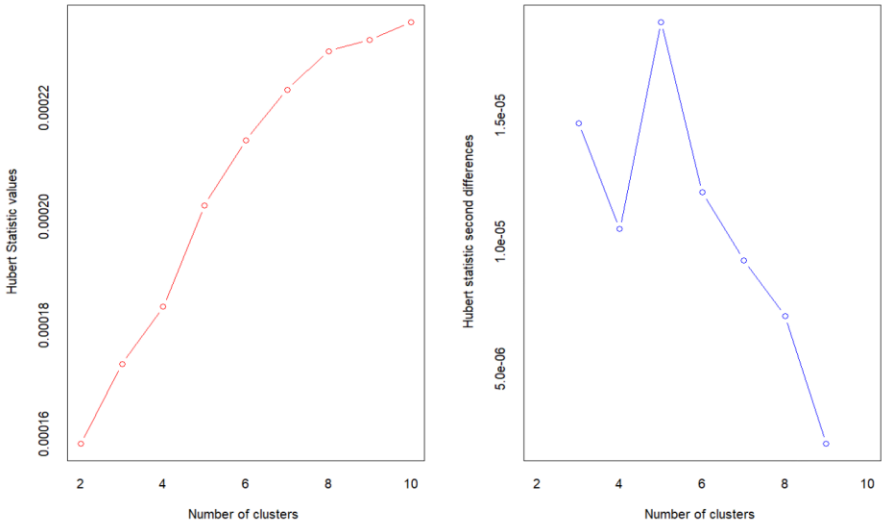
Euclidean distance

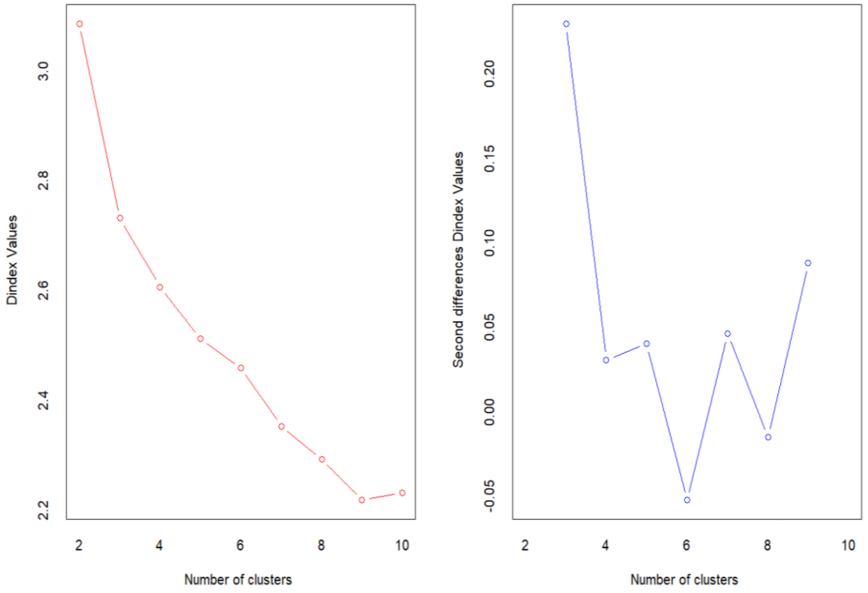


```
> set.seed(3)
> noOfClusters <- NbClust(alterDataNorm, distance = "euclidean", min.nc = 2, max.nc = 10, method = "kmeans", in
x = "all")
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a
                significant increase of the value of the measure i.e the significant peak in Hubert
                index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.
                In the plot of D index, we seek a significant knee (the significant peak in Dindex
                second differences plot) that corresponds to a significant increase of the value of
                the measure.

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

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  3


*******************************************************************
> 
```

Graphical techniques like the Hubert index and D index are used to locate the knee point in the plot where the measure's value significantly increases. The significant peak in the Hubert and D indices is located using the second difference plots.

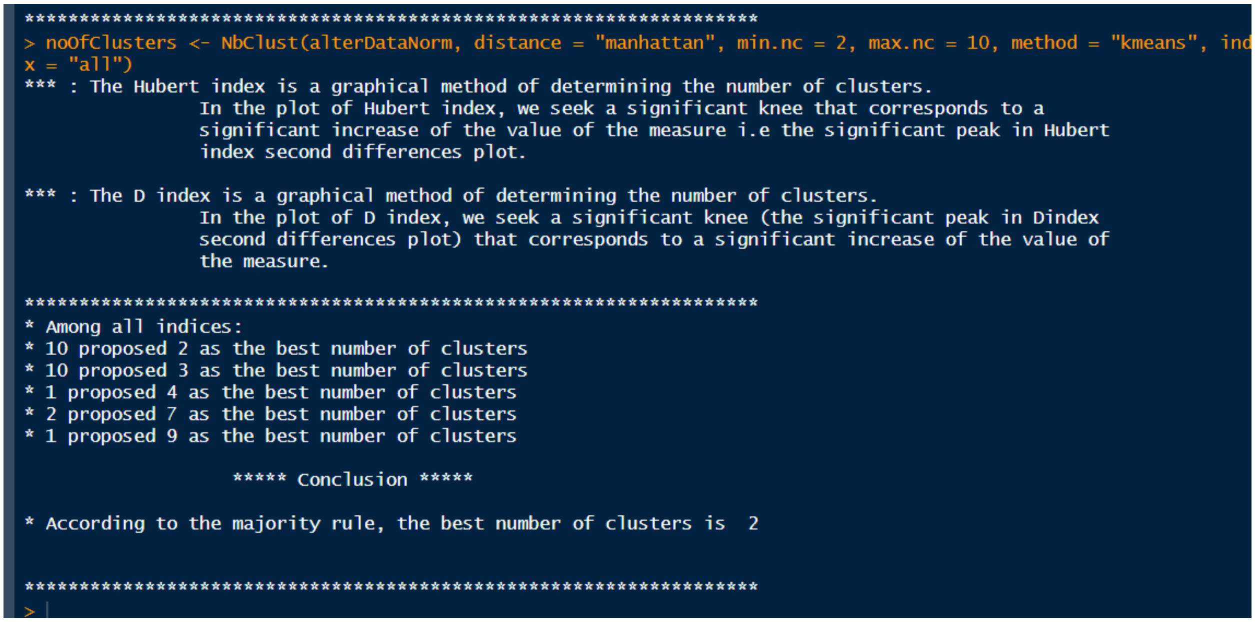
According to the output, out of all the indices, 9 suggest two clusters, 10 suggest three clusters, 1 suggests four clusters, 2 suggest seven clusters, and 2 suggest nine clusters. The highest number of indices (10 out of 24) suggest that 3 clusters are the ideal number, making this the best number of clusters for the given dataset according to the majority rule.



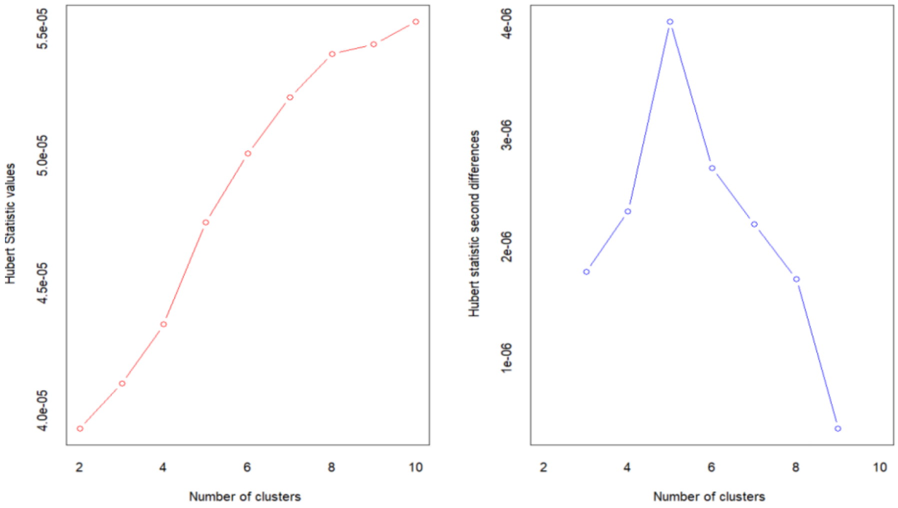


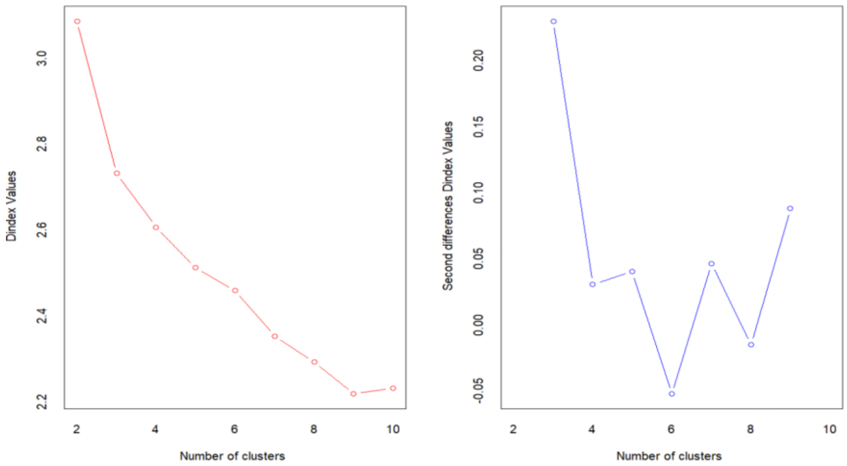
Manhattan distance

The output of the clustering analysis is the result that is shown. Hubert and D indices were used in the analysis to count the number of clusters. Both techniques rely on identifying a knee or peak in their respective plots that signifies a notable rise in the measure's value. The majority of indices suggested 2 or 3 clusters as the ideal number, with three others suggesting 4, two 7, and one 9. The majority rule states that two clusters are the ideal number. In other words, the data can be separated into two separate groups or clusters.



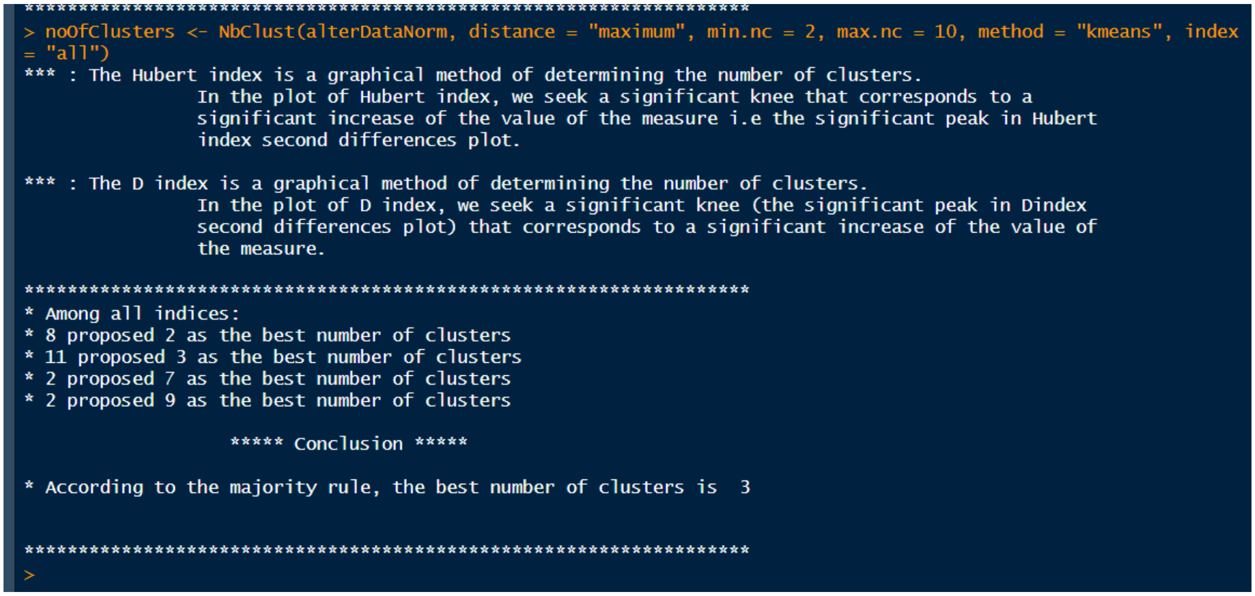
```
*******************************************************************
> noOfClusters <- NbClust(alterDataNorm, distance = "manhattan", min.nc = 2, max.nc = 10, method = "kmeans", ind
x = "all")
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a
                significant increase of the value of the measure i.e the significant peak in Hubert
                index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.
                In the plot of D index, we seek a significant knee (the significant peak in Dindex
                second differences plot) that corresponds to a significant increase of the value of
                the measure.

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

                   ***** Conclusion *****

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


*******************************************************************
>
```



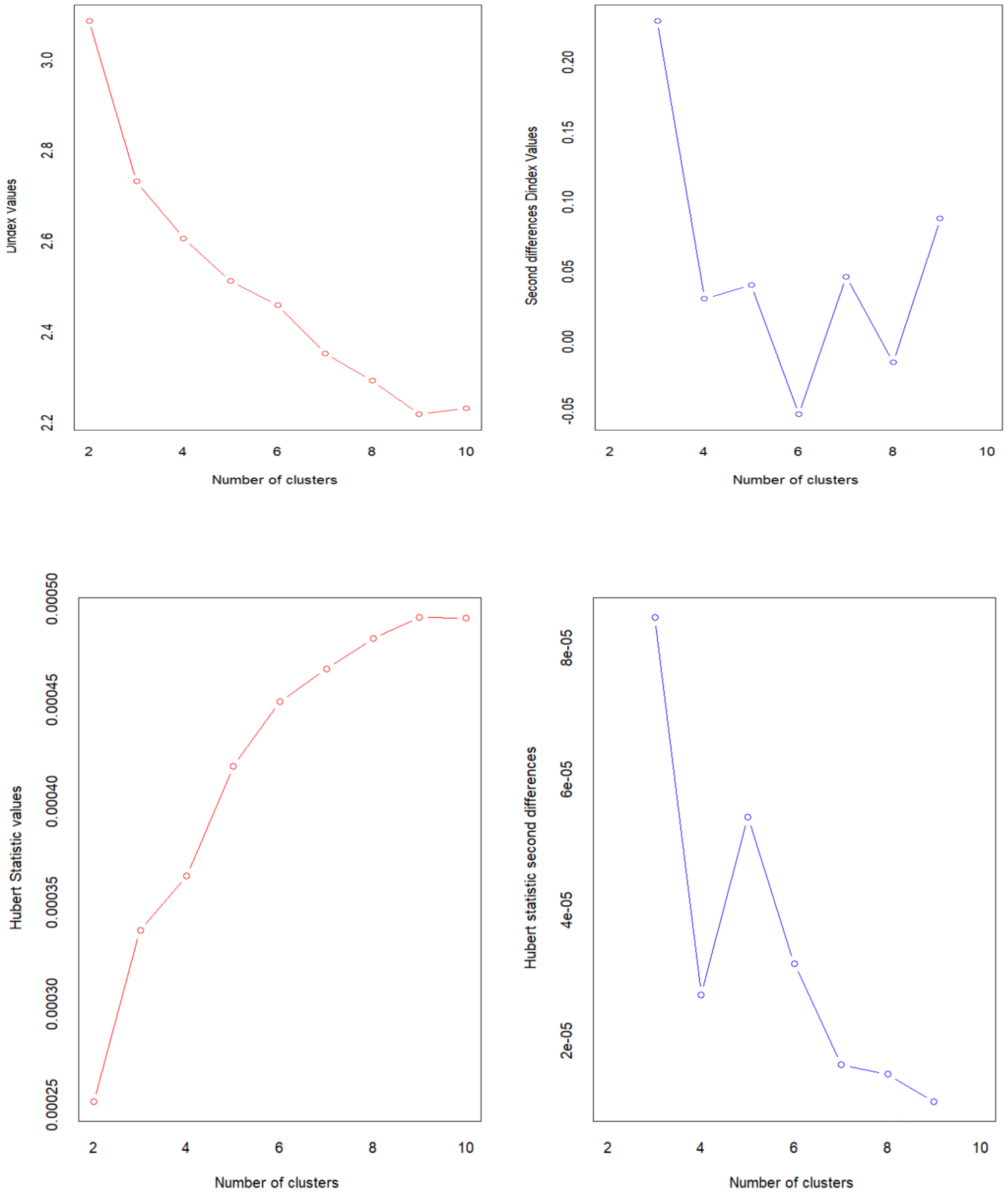


Maximum distance

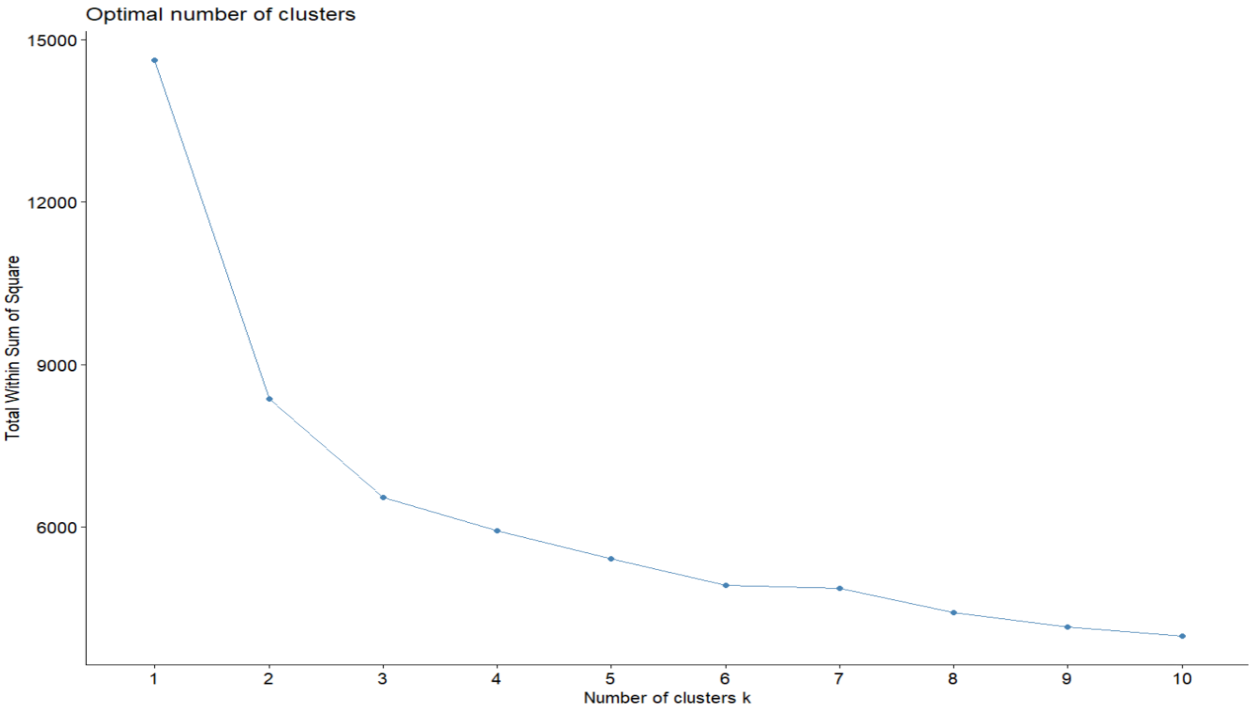
The findings demonstrate that, across all indices, 8 indices suggested 2 as the ideal number of clusters, 11 indices suggested 3, 2 indices suggested 7 as the ideal number of clusters, and 2 indices suggested 9 as the ideal number of clusters. The most effective number of clusters for this dataset, according to the majority rule, is 3. This indicates that using the k-means algorithm with the "maximum" distance measure, the dataset is most likely to be divided into three distinct clusters.



```
*******************************************************************
> noOfClusters <- NbClust(alterDataNorm, distance = "maximum", min.nc = 2, max.nc = 10, method = "kmeans", index
= "all")
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a
                significant increase of the value of the measure i.e the significant peak in Hubert
                index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.
                In the plot of D index, we seek a significant knee (the significant peak in Dindex
                second differences plot) that corresponds to a significant increase of the value of
                the measure.

*******************************************************************
* Among all indices:
* 8 proposed 2 as the best number of clusters
* 11 proposed 3 as the best number of clusters
* 2 proposed 7 as the best number of clusters
* 2 proposed 9 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  3


*******************************************************************
>
```

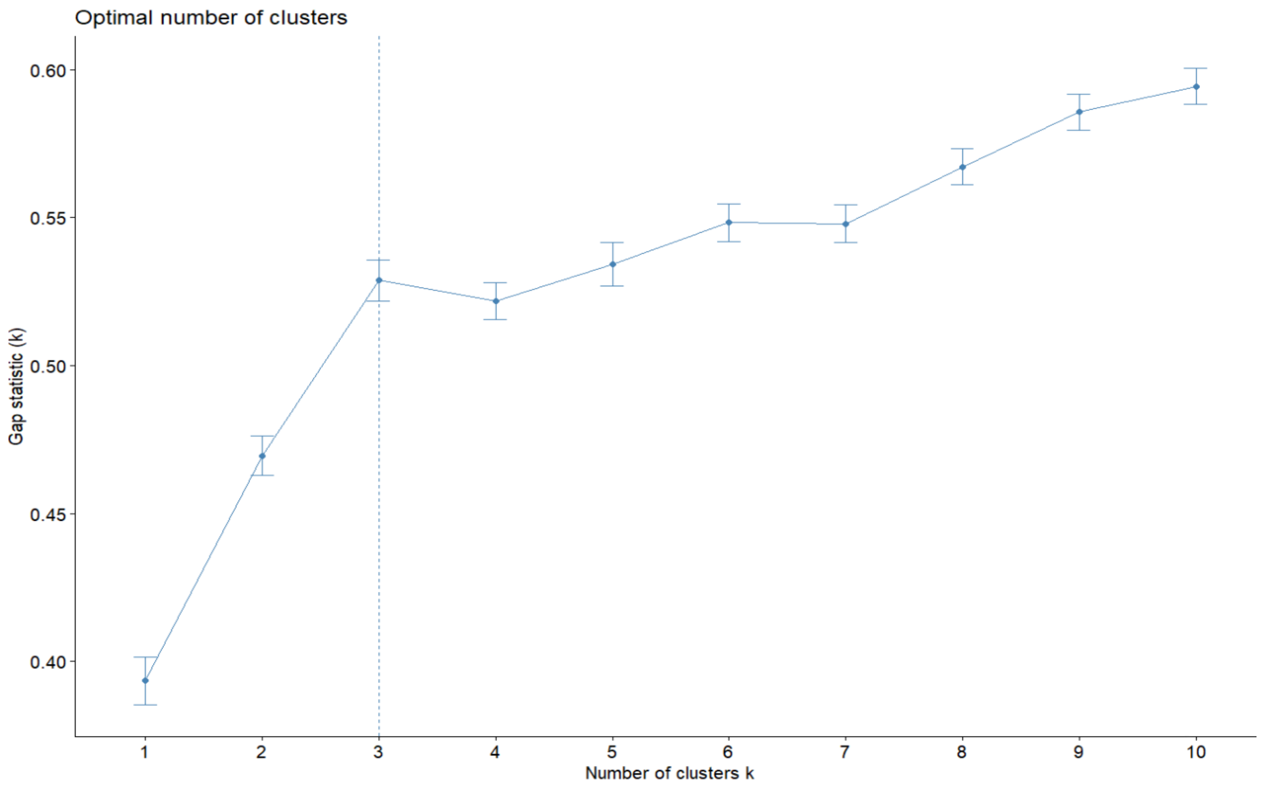

Dindex Values
3.0
2.8
2.6
2.4
2.2
2
4
6
8
10
Number of clusters
Second differences Dindex Values
0.20
0.15
0.10
0.05
0.00
-0.05
2
4
6
8
10
Number of clusters
Hubert Statistic values
0.00050
0.00045
0.00040
0.00035
0.00030
0.00025
2
4
6
8
10
Number of clusters
Hubert statistic second differences
8e-05
6e-05
4e-05
2e-05
2
4
6
8
10
Number of clusters

Elbow



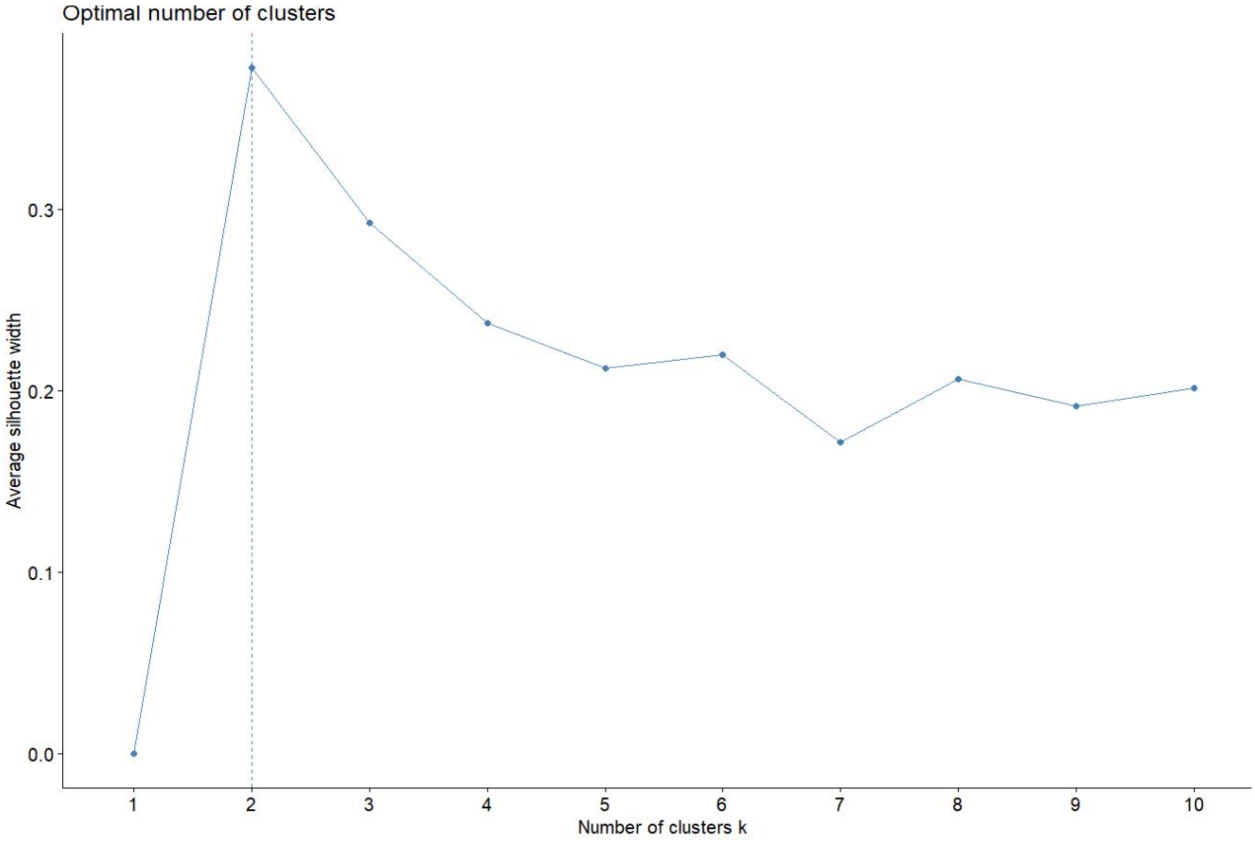
The Elbow method is used to calculate the within clustering sum of squared errors for various k values. Three clusters are discovered to be the most effective for this task based on the elbow plot, which displays the reduction in the sum of squared error as k increases. This is due to the elbow in the plot appearing at k=3, which denotes a significant reduction in the error up until that point and only a modest reduction thereafter.

Gap statistics



The Gap statistic method uses a zero-mean reference distribution of data to assess the total intra-cluster variation for various values of k and compares them to their predicted values. As can be seen in the provided diagram, the method in this case suggests that there should be 3 clusters because this value maximizes the value of the gap statistic.

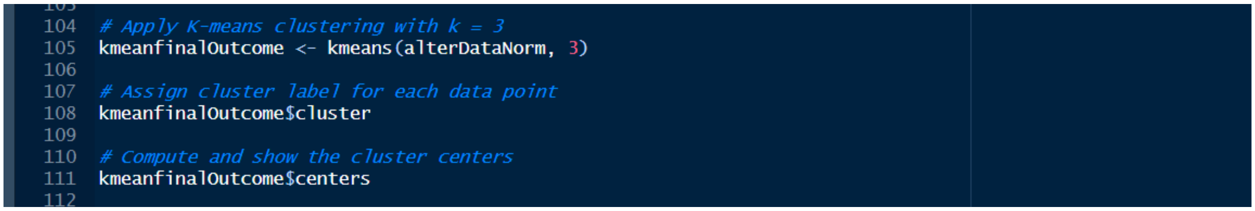
Silhouette



The Silhouette Method assesses the accuracy of clustering and the cohesiveness of the data points within each cluster. The findings show that two clusters are the ideal number, with three clusters being the next-best choice, to maximize the average silhouette values. Three clusters were selected for the kmeans clustering out of all the automated tools used for this task because most methods indicate that this is the optimal number of clusters. Furthermore, three clusters seem appropriate for this issue given that the data represents three different vehicle types and it is difficult to distinguish between two of them.

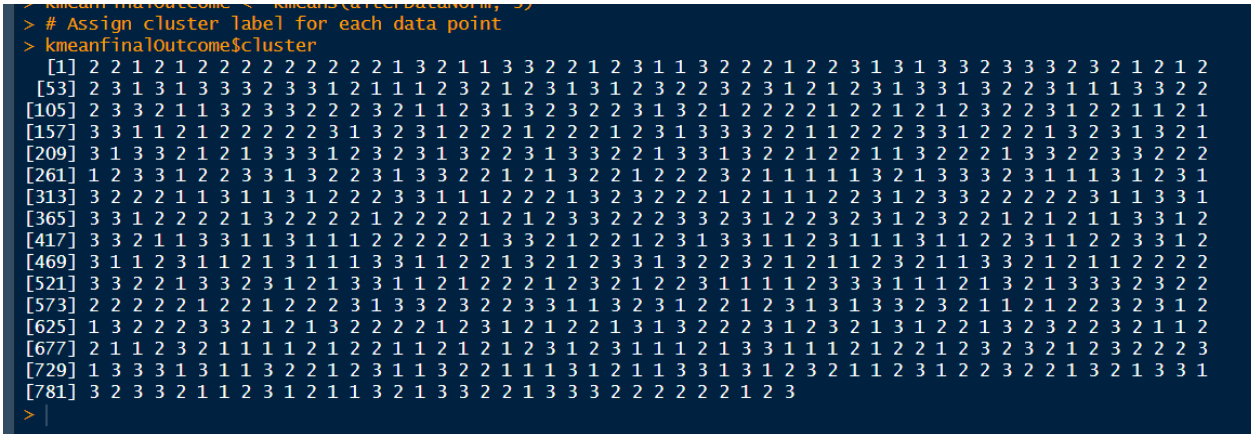
C).K-means clustering investigation.

The pre-processed dataset underwent kmeans clustering after discovering the number of clusters (3). The function for kmeans clustering is displayed in the code below.



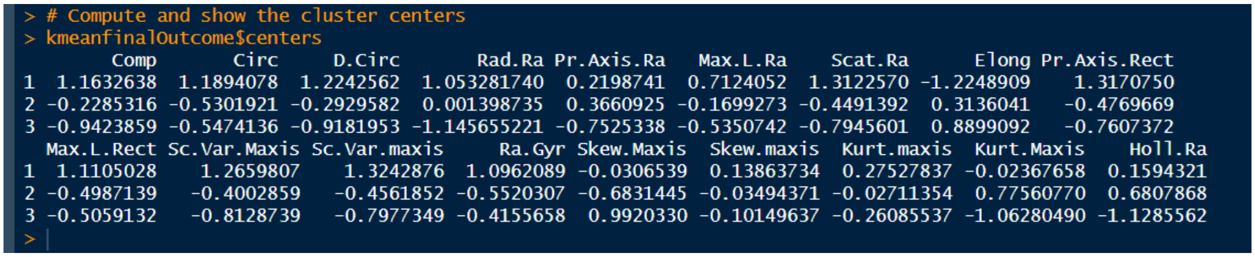
```
# Apply K-means clustering with k = 3
kmeanfinalOutcome <- kmeans(alterDataNorm, 3)

# Assign cluster label for each data point
kmeanfinalOutcome$cluster

# Compute and show the cluster centers
kmeanfinalOutcome$centers
```

After clustering, the results for each row of the cluster can be seen. Each data point is assigned to one of three distinct clusters, 1, 2, or 3.



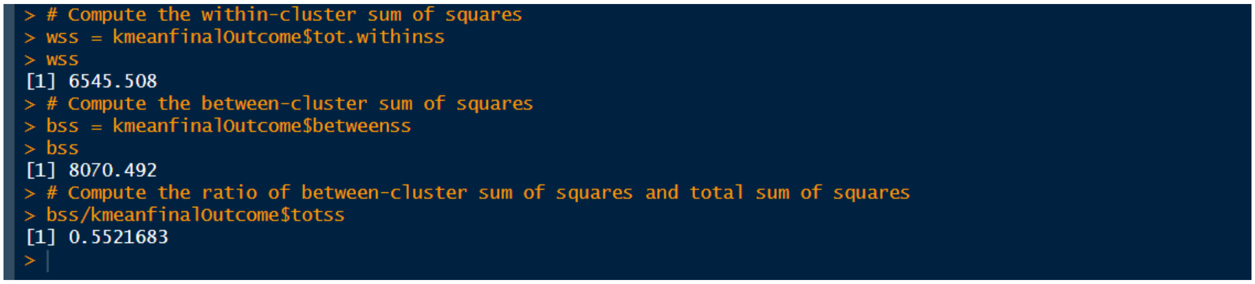
```
> kmeanfinalOutcome <- kmeans(afterDataNorm, 3)
> # Assign cluster label for each data point
> kmeanfinalOutcome$cluster
  [1] 2 2 1 2 1 2 2 2 2 2 2 2 2 2 1 3 2 1 1 3 3 2 2 1 2 3 1 1 3 2 2 2 1 2 2 3 1 3 1 3 3 2 3 3 3 2 3 2 1 2 1 2
 [53] 2 3 1 3 1 3 3 3 2 3 3 1 2 1 1 1 2 3 2 1 2 3 1 3 1 2 3 2 2 3 2 3 1 2 1 2 3 1 3 3 1 3 2 2 3 1 1 1 3 3 2 2
[105] 2 3 3 2 1 1 3 2 3 3 2 2 2 3 2 1 1 2 3 1 3 2 3 2 2 3 1 3 2 1 2 2 2 2 1 2 2 1 2 1 2 3 2 2 3 1 2 2 1 1 2 1
[157] 3 3 1 1 2 1 2 2 2 2 2 3 1 3 2 3 1 2 2 2 1 2 2 2 1 2 3 1 3 3 3 2 2 1 1 2 2 2 3 3 1 2 2 2 1 3 2 3 1 3 2 1
[209] 3 1 3 3 2 1 2 1 3 3 3 1 2 3 2 3 1 3 2 2 3 1 3 3 2 2 1 3 3 1 3 2 2 1 2 2 1 1 3 2 2 2 1 3 3 2 2 3 3 2 2 2
[261] 1 2 3 3 1 2 2 3 3 1 3 2 2 3 1 3 3 2 2 1 2 1 3 2 2 1 2 2 2 3 2 1 1 1 1 1 3 2 1 3 3 3 2 3 1 1 1 3 1 2 3 1
[313] 3 2 2 2 1 1 3 1 1 3 1 2 2 2 3 3 1 1 1 2 2 2 1 3 2 3 2 2 2 1 2 1 1 1 2 2 3 1 2 3 3 2 2 2 2 2 3 1 1 3 3 1
[365] 3 3 1 2 2 2 2 1 3 2 2 2 2 1 2 2 2 2 1 2 1 2 3 3 2 2 2 3 3 2 3 1 2 2 3 2 3 1 2 3 2 2 1 2 1 2 1 1 3 3 1 2
[417] 3 3 2 1 1 3 3 1 1 3 1 1 1 2 2 2 2 2 1 3 3 2 1 2 2 1 2 3 1 3 3 1 1 2 3 1 1 1 3 1 1 2 2 3 1 1 2 2 3 3 1 2
[469] 3 1 1 2 3 1 1 2 1 3 1 1 1 3 3 1 1 2 2 1 3 2 1 2 3 3 1 3 2 2 3 2 1 2 1 1 2 3 2 1 1 3 3 2 1 2 1 1 2 2 2 2
[521] 3 3 2 2 1 3 3 2 3 1 2 1 3 3 1 1 2 1 2 2 2 1 2 3 2 1 2 2 3 1 1 1 1 2 3 3 3 1 1 1 2 1 3 2 1 3 3 3 2 3 2 2
[573] 2 2 2 2 2 1 2 2 1 2 2 2 3 1 3 3 2 3 2 2 3 3 1 1 3 2 3 1 2 2 1 2 3 1 3 1 3 3 2 3 2 1 1 2 1 2 2 3 2 3 1 2
[625] 1 3 2 2 2 3 3 2 1 2 1 3 2 2 2 2 1 2 3 1 2 1 2 2 1 3 1 3 2 2 2 3 1 2 3 2 1 3 1 2 2 1 3 2 3 2 2 3 2 1 1 2
[677] 2 1 1 2 3 2 1 1 1 1 2 1 2 2 1 1 2 1 2 1 2 3 1 2 3 1 1 1 2 1 3 3 1 1 1 2 1 2 2 1 2 3 2 3 2 1 2 3 2 2 2 3
[729] 1 3 3 3 1 3 1 1 3 2 2 1 2 3 1 1 3 2 2 1 1 1 3 1 2 1 1 3 3 1 3 1 2 3 2 1 1 2 3 1 2 2 3 2 2 1 3 2 1 3 3 1
[781] 3 2 3 3 2 1 1 2 3 1 2 1 1 3 2 1 3 3 2 2 1 3 3 3 2 2 2 2 2 1 2 3
> |
```

The cluster centers were found.



```
> # Compute and show the cluster centers
> kmeanfinalOutcome$centers
        Comp       Circ     D.Circ       Rad.Ra Pr.Axis.Ra   Max.L.Ra    Scat.Ra      Elong Pr.Axis.Rect
1  1.1632638  1.1894078  1.2242562  1.053281740  0.2198741  0.7124052  1.3122570 -1.2248909    1.3170750
2 -0.2285316 -0.5301921 -0.2929582  0.001398735  0.3660925 -0.1699273 -0.4491392  0.3136041   -0.4769669
3 -0.9423859 -0.5474136 -0.9181953 -1.145655221 -0.7525338 -0.5350742 -0.7945601  0.8899092   -0.7607372
  Max.L.Rect Sc.Var.Maxis Sc.Var.maxis     Ra.Gyr Skew.Maxis  Skew.maxis  Kurt.maxis  Kurt.Maxis    Holl.Ra
1  1.1105028    1.2659807    1.3242876  1.0962089 -0.0306539  0.13863734  0.27527837 -0.02367658  0.1594321
2 -0.4987139   -0.4002859   -0.4561852 -0.5520307 -0.6831445 -0.03494371 -0.02711354  0.77560770  0.6807868
3 -0.5059132   -0.8128739   -0.7977349 -0.4155658  0.9920330 -0.10149637 -0.26085537 -1.06280490 -1.1285562
> |
```

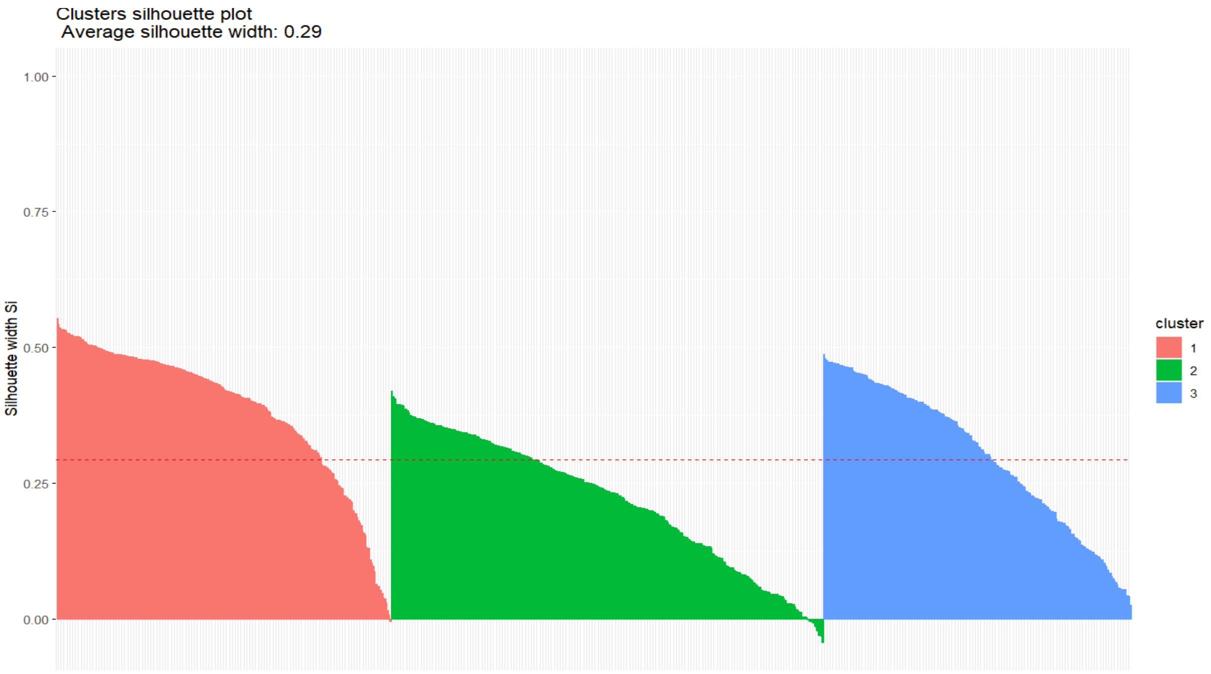
Calculated between cluster sums of squares over the total sum of squares.



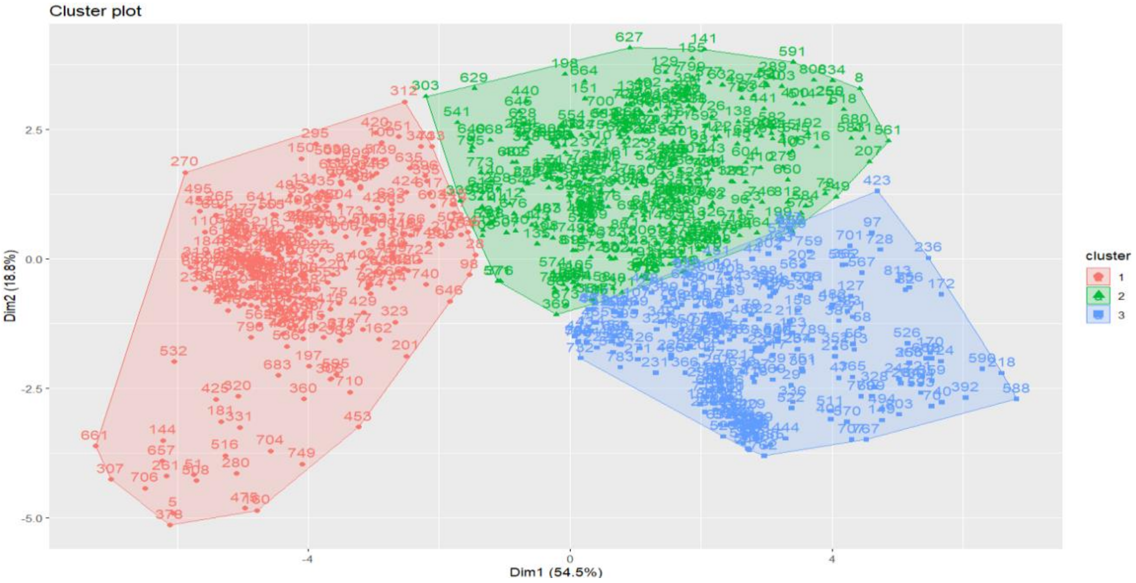
```
> # Compute the within-cluster sum of squares
> wss = kmeanfinalOutcome$tot.withinss
> wss
[1] 6545.508
> # Compute the between-cluster sum of squares
> bss = kmeanfinalOutcome$betweenss
> bss
[1] 8070.492
> # Compute the ratio of between-cluster sum of squares and total sum of squares
> bss/kmeanfinalOutcome$totss
[1] 0.5521683
> |
```

The clustering was successful and the clusters were distinct, according to the value of 0.55. Squares within clusters add up to 6545, while those between clusters add up to 8070. This demonstrates that the data points in various clusters are more dispersed than those in the same cluster. A lower sum of squares within clusters denotes greater similarity among data points within a given cluster, while a higher sum of squares between clusters denotes greater dissimilarity among data points within different clusters.

D).Silhouette plot for the K-means clustering.



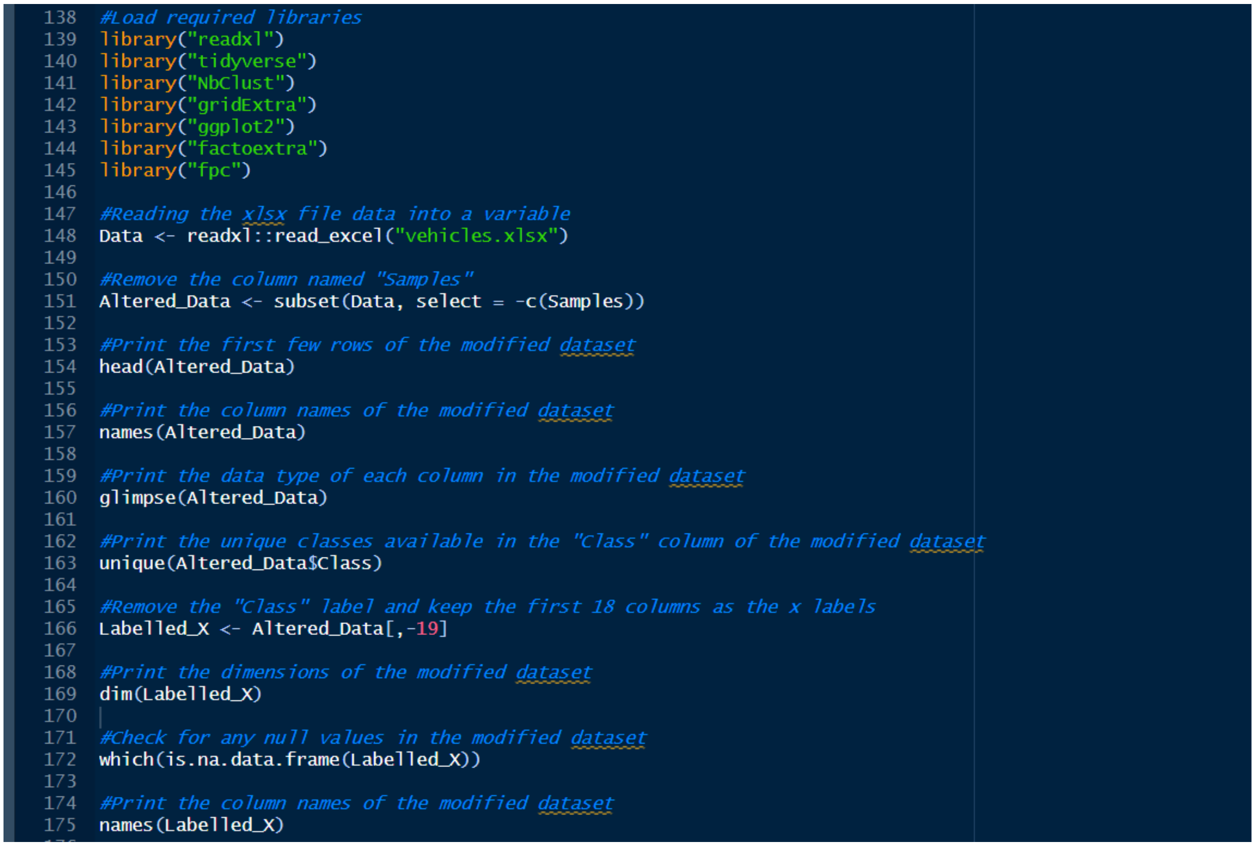
The silhouette method determines the average distance between clusters and assesses how effective clustering is. Each cluster has a width that is greater than the average width score, demonstrating successful clustering. The average silhouette width score, on the other hand, is close to 0, indicating that it is located between two clusters. The 0.29 score indicates that the clustering was not very successful and that the clusters were not well-separated.



Three distinct clusters are represented by three different colors in the plot above. The green and blue clusters are situated closer to one another than the red cluster, whose datapoints are more evenly dispersed, despite the fact that all of the clusters are clearly separated from one another. This shows that the kmeans clustering process was successful in identifying three clusters, which might correspond to a car, van, and bus.

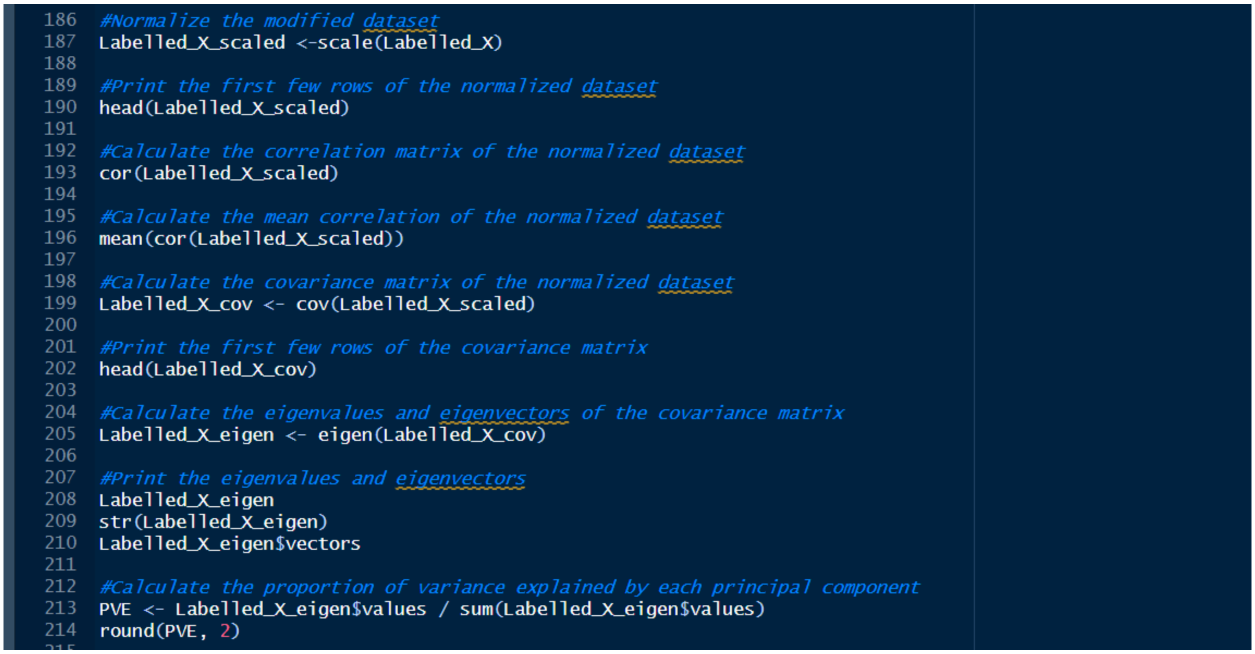
E).PCA method.

The dataset was loaded in the first step of the second subtask, and some columns like "Samples" and "Class" were removed because clustering did not require them. As outliers might significantly affect the dataset, they weren't eliminated. The dataset was normalized using Z-score standardization and then PCA was applied. To decrease the number of dimensions and enhance the clustering outcomes, PCA should be used prior to clustering.



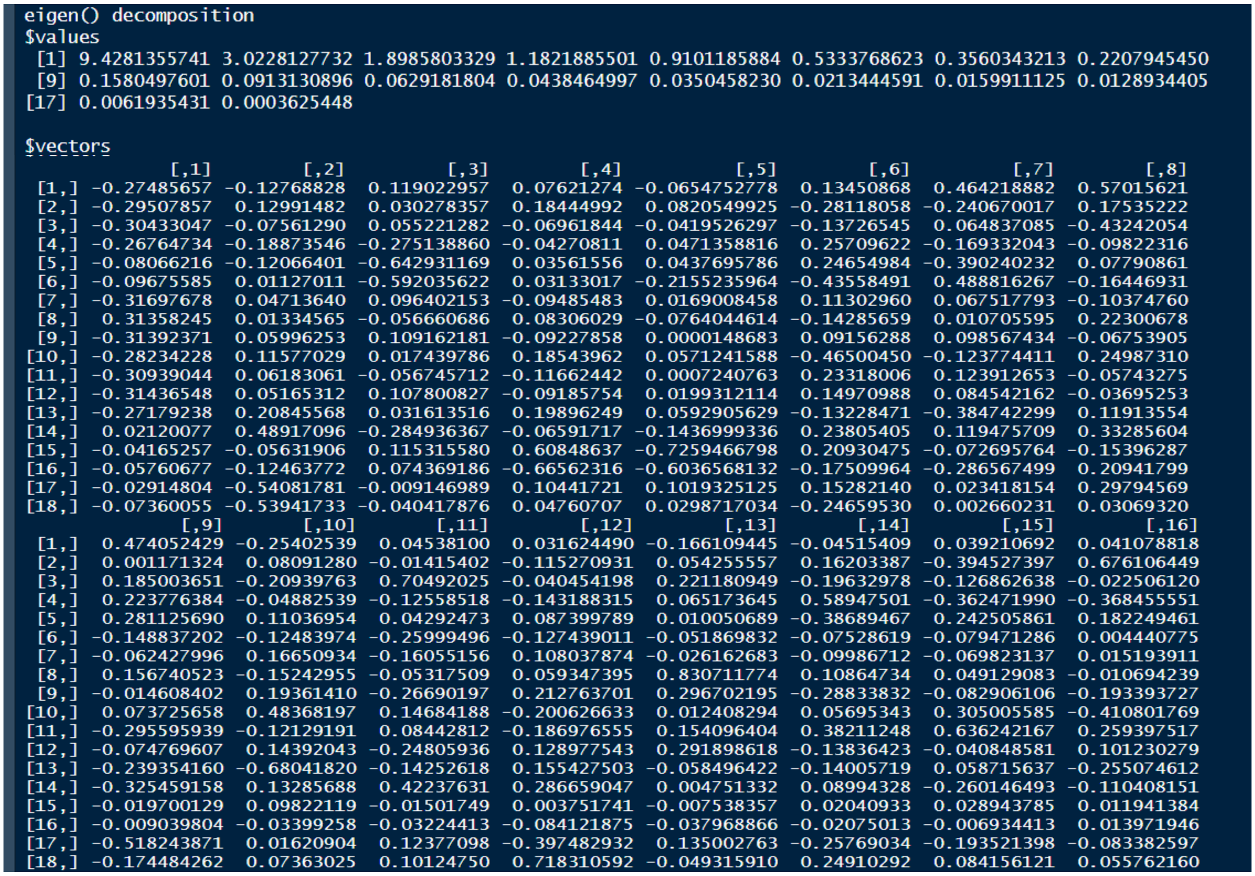
```r
#Load required libraries
library("readxl")
library("tidyverse")
library("NbClust")
library("gridExtra")
library("ggplot2")
library("factoextra")
library("fpc")

#Reading the xlsx file data into a variable
Data <- readxl::read_excel("vehicles.xlsx")

#Remove the column named "Samples"
Altered_Data <- subset(Data, select = -c(Samples))

#Print the first few rows of the modified dataset
head(Altered_Data)

#Print the column names of the modified dataset
names(Altered_Data)

#Print the data type of each column in the modified dataset
glimpse(Altered_Data)

#Print the unique classes available in the "Class" column of the modified dataset
unique(Altered_Data$Class)

#Remove the "Class" label and keep the first 18 columns as the x labels
Labelled_X <- Altered_Data[,-19]

#Print the dimensions of the modified dataset
dim(Labelled_X)

#Check for any null values in the modified dataset
which(is.na.data.frame(Labelled_X))

#Print the column names of the modified dataset
names(Labelled_X)
```

Principal component analysis makes an effort to shrink the dataset's dimension while maintaining the dataset's integrity. PCA converts correlated variables from a dataset to a set of principal components, which are uncorrelated variables. In order to perform PCA in R, first we calculate the covariance matrix, and then we use the covariance matrix to calculate the eigenvalues and eigenvectors.

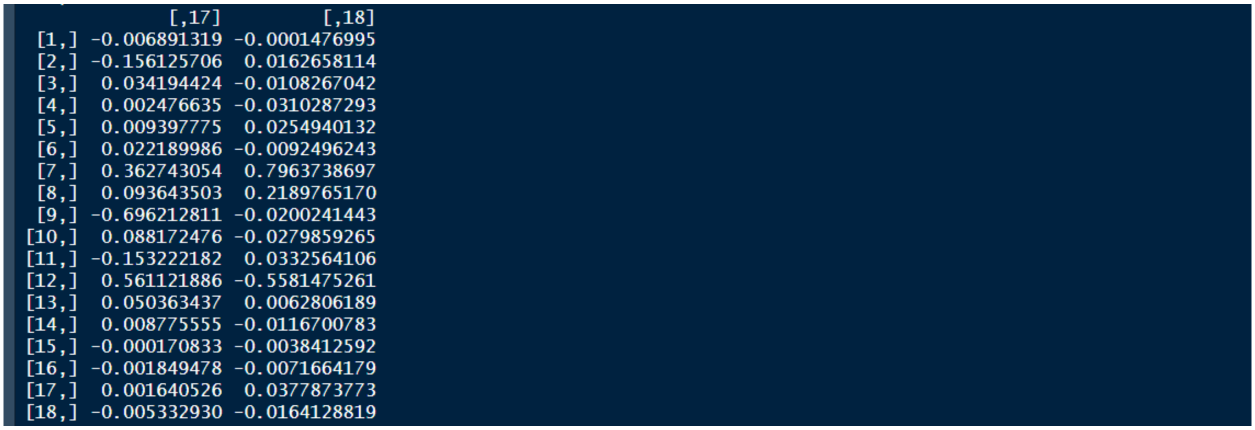


```r
#Normalize the modified dataset
Labelled_X_scaled <-scale(Labelled_X)

#Print the first few rows of the normalized dataset
head(Labelled_X_scaled)

#Calculate the correlation matrix of the normalized dataset
cor(Labelled_X_scaled)

#Calculate the mean correlation of the normalized dataset
mean(cor(Labelled_X_scaled))

#Calculate the covariance matrix of the normalized dataset
Labelled_X_cov <- cov(Labelled_X_scaled)

#Print the first few rows of the covariance matrix
head(Labelled_X_cov)

#Calculate the eigenvalues and eigenvectors of the covariance matrix
Labelled_X_eigen <- eigen(Labelled_X_cov)

#Print the eigenvalues and eigenvectors
Labelled_X_eigen
str(Labelled_X_eigen)
Labelled_X_eigen$vectors

#Calculate the proportion of variance explained by each principal component
PVE <- Labelled_X_eigen$values / sum(Labelled_X_eigen$values)
round(PVE, 2)
```

Below are the calculated eigen vectors and eigen values.

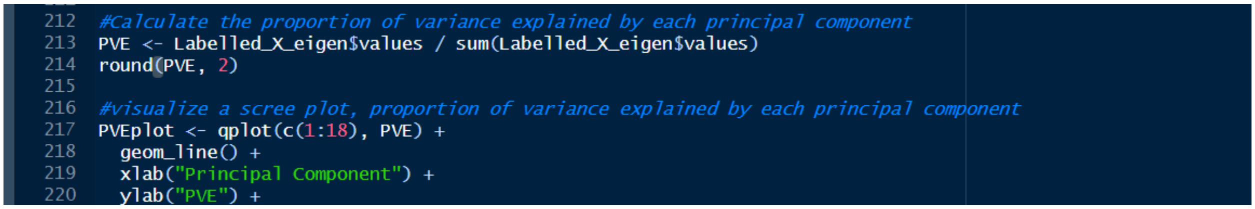


```
eigen() decomposition
$values
 [1] 9.4281355741 3.0228127732 1.8985803329 1.1821885501 0.9101185884 0.5333768623 0.3560343213 0.2207945450
 [9] 0.1580497601 0.0913130896 0.0629181804 0.0438464997 0.0350458230 0.0213444591 0.0159911125 0.0128934405
[17] 0.0061935431 0.0003625448

$vectors
             [,1]        [,2]         [,3]        [,4]          [,5]        [,6]         [,7]        [,8]
 [1,] -0.27485657 -0.12768828  0.119022957  0.07621274 -0.0654752778  0.13450868  0.464218882  0.57015621
 [2,] -0.29507857  0.12991482  0.030278357  0.18444992  0.0820549925 -0.28118058 -0.240670017  0.17535222
 [3,] -0.30433047 -0.07561290  0.055221282 -0.06961844 -0.0419526297 -0.13726545  0.064837085 -0.43242054
 [4,] -0.26764734 -0.18873546 -0.275138860 -0.04270811  0.0471358816  0.25709622 -0.169332043 -0.09822316
 [5,] -0.08066216 -0.12066401 -0.642931169  0.03561556  0.0437695786  0.24654984 -0.390240232  0.07790861
 [6,] -0.09675585  0.01127011 -0.592035622  0.03133017 -0.2155235964 -0.43558491  0.488816267 -0.16446931
 [7,] -0.31697678  0.04713640  0.096402153 -0.09485483  0.0169008458  0.11302960  0.067517793 -0.10374760
 [8,]  0.31358245  0.01334565 -0.056660686  0.08306029 -0.0764044614 -0.14285659  0.010705595  0.22300678
 [9,] -0.31392371  0.05996253  0.109162181 -0.09227858  0.0000148683  0.09156288  0.098567434 -0.06753905
[10,] -0.28234228  0.11577029  0.017439786  0.18543962  0.0571241588 -0.46500450 -0.123774411  0.24987310
[11,] -0.30939044  0.06183061 -0.056745712 -0.11662442  0.0007240763  0.23318006  0.123912653 -0.05743275
[12,] -0.31436548  0.05165312  0.107800827 -0.09185754  0.0199312114  0.14970988  0.084542162 -0.03695253
[13,] -0.27179238  0.20845568  0.031613516  0.19896249  0.0592905629 -0.13228471 -0.384742299  0.11913554
[14,]  0.02120077  0.48917096 -0.284936367 -0.06591717 -0.1436999336  0.23805405  0.119475709  0.33285604
[15,] -0.04165257 -0.05631906  0.115315580  0.60848637 -0.7259466798  0.20930475 -0.072695764 -0.15396287
[16,] -0.05760677 -0.12463772  0.074369186 -0.66562316 -0.6036568132 -0.17509964 -0.286567499  0.20941799
[17,] -0.02914804 -0.54081781 -0.009146989  0.10441721  0.1019325125  0.15282140  0.023418154  0.29794569
[18,] -0.07360055 -0.53941733 -0.040417876  0.04760707  0.0298717034 -0.24659530  0.002660231  0.03069320
              [,9]       [,10]       [,11]        [,12]        [,13]       [,14]        [,15]        [,16]
 [1,]  0.474052429 -0.25402539  0.04538100  0.031624490 -0.166109445 -0.04515409  0.039210692  0.041078818
 [2,]  0.001171324  0.08091280 -0.01415402 -0.115270931  0.054255557  0.16203387 -0.394527397  0.676106449
 [3,]  0.185003651 -0.20939763  0.70492025 -0.040454198  0.221180949 -0.19632978 -0.126862638 -0.022506120
 [4,]  0.223776384 -0.04882539 -0.12558518 -0.143188315  0.065173645  0.58947501 -0.362471990 -0.368455551
 [5,]  0.281125690  0.11036954  0.04292473  0.087399789  0.010050689 -0.38689467  0.242505861  0.182249461
 [6,] -0.148837202 -0.12483974 -0.25999496 -0.127439011 -0.051869832 -0.07528619 -0.079471286  0.004440775
 [7,] -0.062427996  0.16650934 -0.16055156  0.108037874 -0.026162683 -0.09986712 -0.069823137  0.015193911
 [8,]  0.156740523 -0.15242955 -0.05317509  0.059347395  0.830711774  0.10864734  0.049129083 -0.010694239
 [9,] -0.014608402  0.19361410 -0.26690197  0.212763701  0.296702195 -0.28833832 -0.082906106 -0.193393727
[10,]  0.073725658  0.48368197  0.14684188 -0.200626633  0.012408294  0.05695343  0.305005585 -0.410801769
[11,] -0.295595939 -0.12129191  0.08442812 -0.186976555  0.154096404  0.38211248  0.636242167  0.259397517
[12,] -0.074769607  0.14392043 -0.24805936  0.128977543  0.291898618 -0.13836423 -0.040848581  0.101230279
[13,] -0.239354160 -0.68041820 -0.14252618  0.155427503 -0.058496422 -0.14005719  0.058715637 -0.255074612
[14,] -0.325459158  0.13285688  0.42237631  0.286659047  0.004751332  0.08994328 -0.260146493 -0.110408151
[15,] -0.019700129  0.09822119 -0.01501749  0.003751741 -0.007538357  0.02040933  0.028943785  0.011941384
[16,] -0.009039804 -0.03399258 -0.03224413 -0.084121875 -0.037968866 -0.02075013 -0.006934413  0.013971946
[17,] -0.518243871  0.01620904  0.12377098 -0.397482932  0.135002763 -0.25769034 -0.193521398 -0.083382597
[18,] -0.174484262  0.07363025  0.10124750  0.718310592 -0.049315910  0.24910292  0.084156121  0.055762160
```

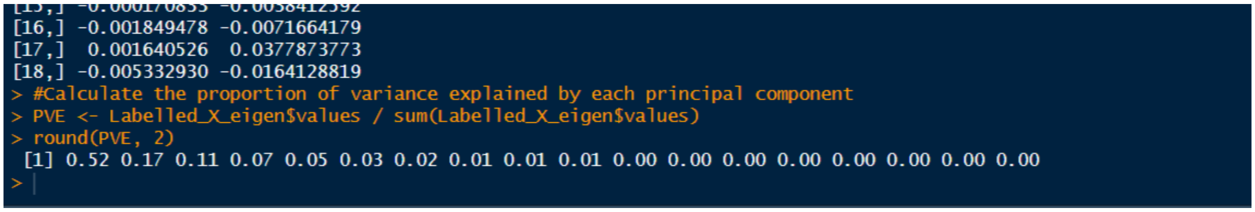


```
              [,17]         [,18]
 [1,] -0.006891319 -0.0001476995
 [2,] -0.156125706  0.0162658114
 [3,]  0.034194424 -0.0108267042
 [4,]  0.002476635 -0.0310287293
 [5,]  0.009397775  0.0254940132
 [6,]  0.022189986 -0.0092496243
 [7,]  0.362743054  0.7963738697
 [8,]  0.093643503  0.2189765170
 [9,] -0.696212811 -0.0200241443
[10,]  0.088172476 -0.0279859265
[11,] -0.153222182  0.0332564106
[12,]  0.561121886 -0.5581475261
[13,]  0.050363437  0.0062806189
[14,]  0.008775555 -0.0116700783
[15,] -0.000170833 -0.0038412592
[16,] -0.001849478 -0.0071664179
[17,]  0.001640526  0.0377873773
[18,] -0.005332930 -0.0164128819
```

The proportion of variance explained was calculated following the determination of the eigen vectors and values.

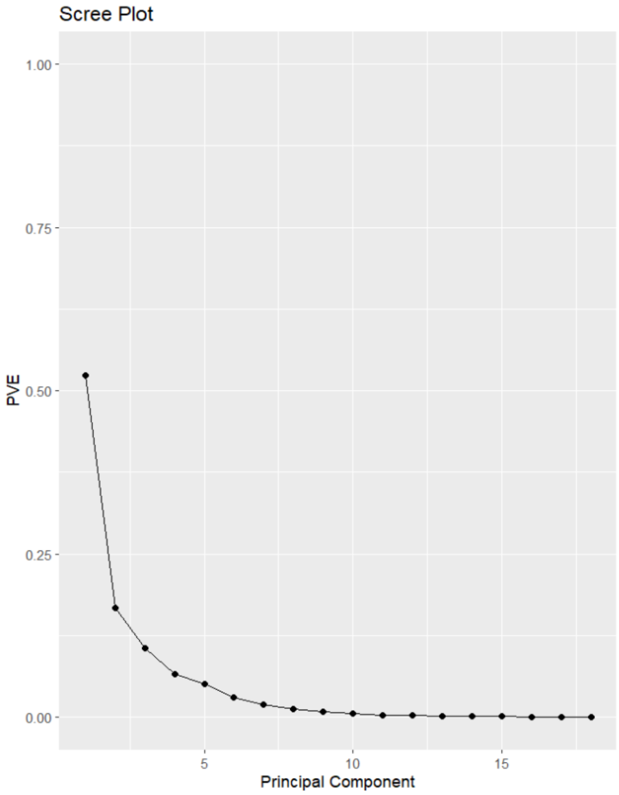


```
#Calculate the proportion of variance explained by each principal component
PVE <- Labelled_X_eigen$values / sum(Labelled_X_eigen$values)
round(PVE, 2)

#visualize a scree plot, proportion of variance explained by each principal component
PVEplot <- qplot(c(1:18), PVE) +
  geom_line() +
  xlab("Principal Component") +
  ylab("PVE") +
```

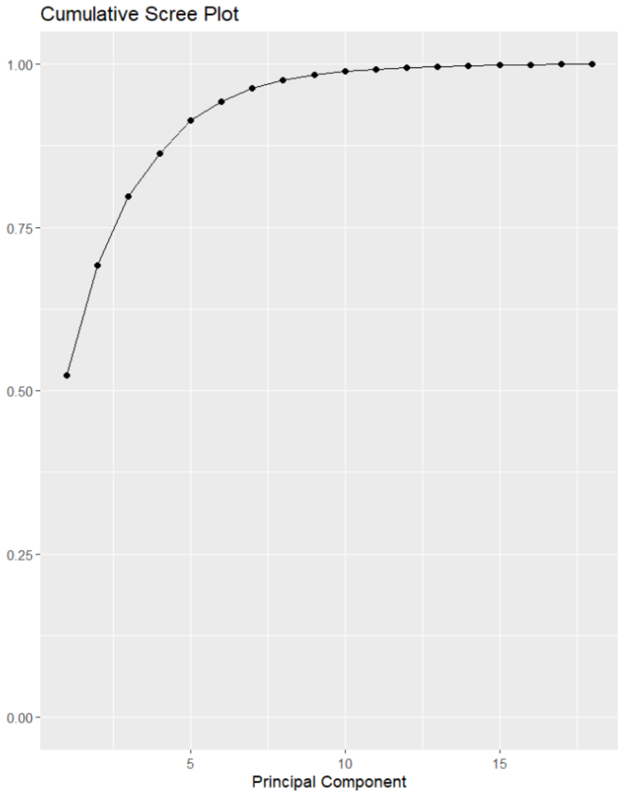


```
[15,] -0.000170833 -0.0038412592
[16,] -0.001849478 -0.0071664179
[17,]  0.001640526  0.0377873773
[18,] -0.005332930 -0.0164128819
> #Calculate the proportion of variance explained by each principal component
> PVE <- Labelled_X_eigen$values / sum(Labelled_X_eigen$values)
> round(PVE, 2)
 [1] 0.52 0.17 0.11 0.07 0.05 0.03 0.02 0.01 0.01 0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
> 
```

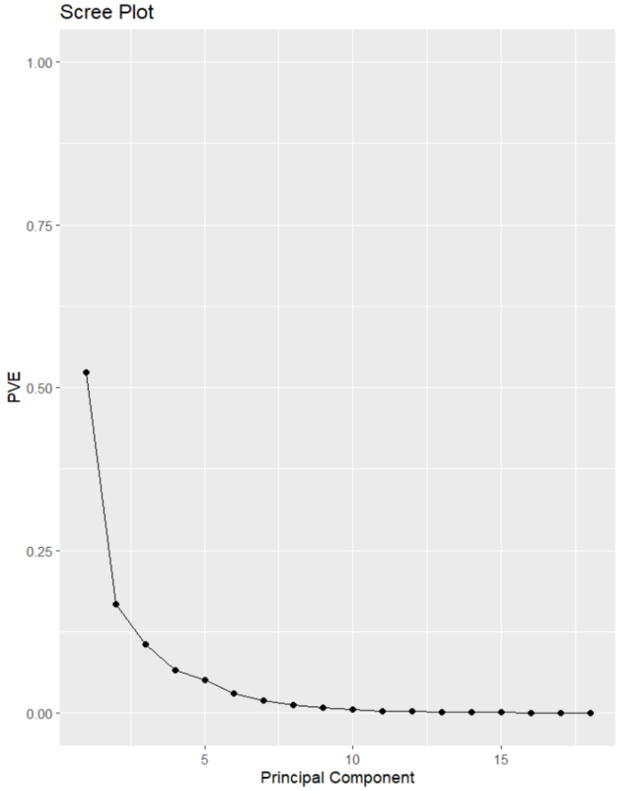
According to the aforementioned diagram, the first principal component accounts for 52% of the variance in the data and the second principal component, 17% and etc.. Because they exceed the required level of 92%, the first six principal components, with a combined variance of 95%, offer good variance coverage. This indicates that a substantial portion of the variance is explained by the six features. Additionally, the scree plot, cumulative plot, and PVE plot were employed to depict this variance.



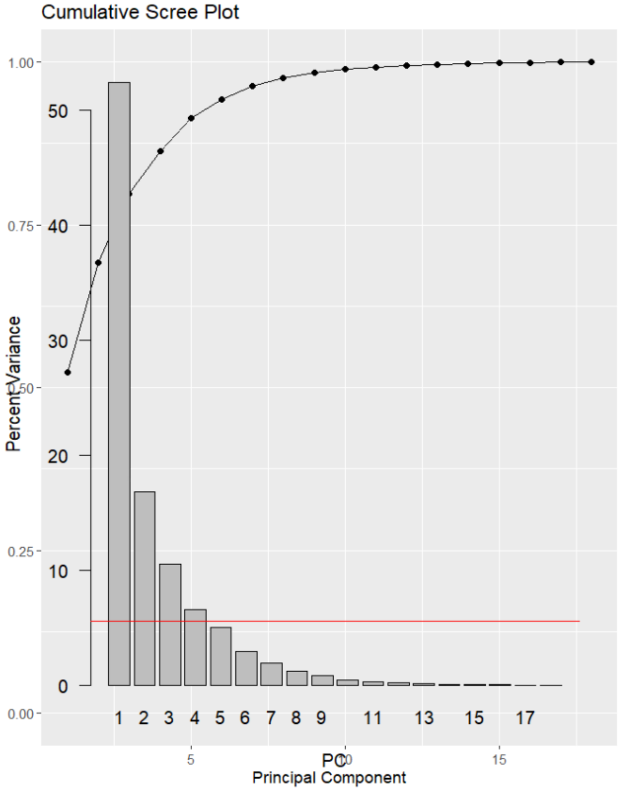
Scree Plot



Cumulative Scree Plot

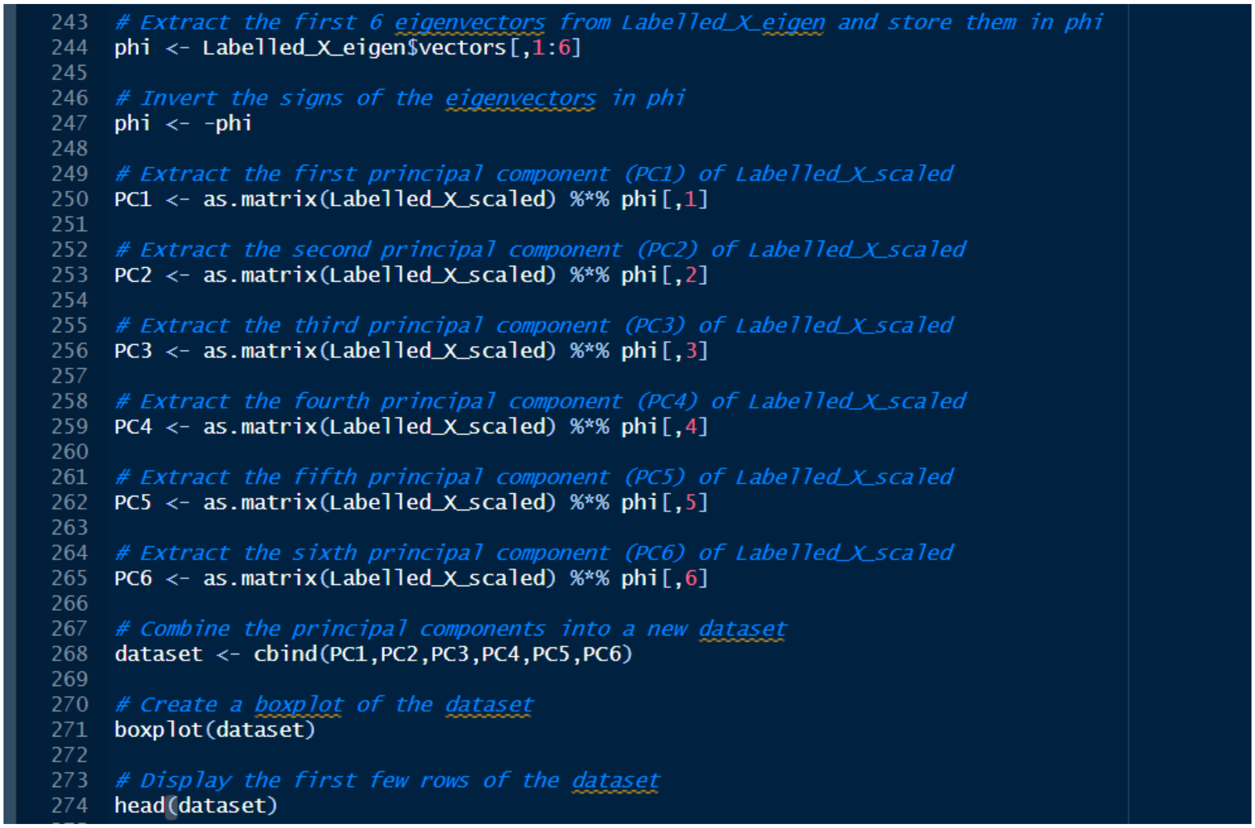


Scree Plot

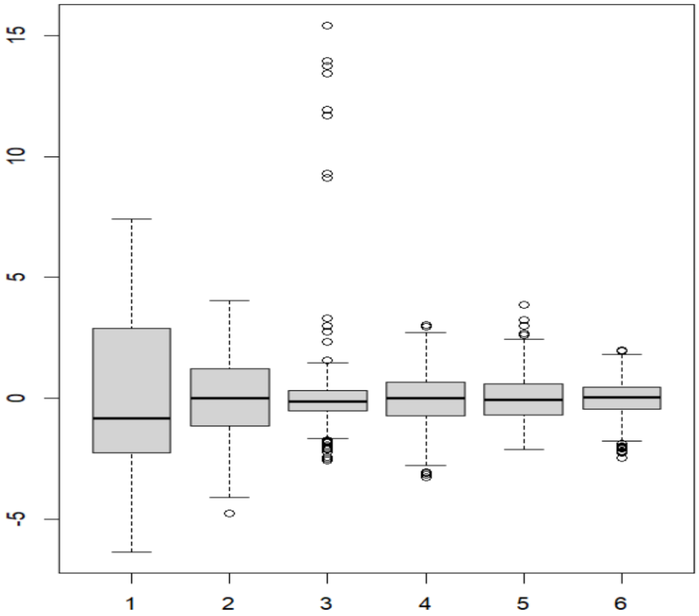


Cumulative Scree Plot

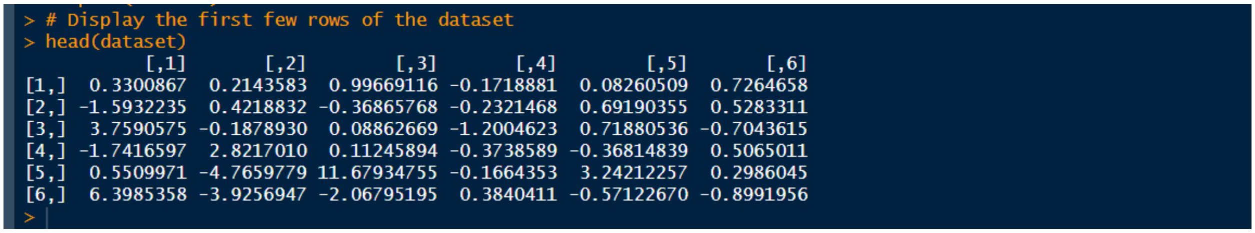
The elbow point indicates that 5 features give a good variance and can be chosen as our principal components because, as we can see from the plots, the variance up to number 5 is high. However, because the task required taking a cumulative score greater than 92, the sixth variance was also chosen to create the transformed dataset.



```r
# Extract the first 6 eigenvectors from Labelled_X_eigen and store them in phi
phi <- Labelled_X_eigen$vectors[,1:6]

# Invert the signs of the eigenvectors in phi
phi <- -phi

# Extract the first principal component (PC1) of Labelled_X_scaled
PC1 <- as.matrix(Labelled_X_scaled) %*% phi[,1]

# Extract the second principal component (PC2) of Labelled_X_scaled
PC2 <- as.matrix(Labelled_X_scaled) %*% phi[,2]

# Extract the third principal component (PC3) of Labelled_X_scaled
PC3 <- as.matrix(Labelled_X_scaled) %*% phi[,3]

# Extract the fourth principal component (PC4) of Labelled_X_scaled
PC4 <- as.matrix(Labelled_X_scaled) %*% phi[,4]

# Extract the fifth principal component (PC5) of Labelled_X_scaled
PC5 <- as.matrix(Labelled_X_scaled) %*% phi[,5]

# Extract the sixth principal component (PC6) of Labelled_X_scaled
PC6 <- as.matrix(Labelled_X_scaled) %*% phi[,6]

# Combine the principal components into a new dataset
dataset <- cbind(PC1,PC2,PC3,PC4,PC5,PC6)

# Create a boxplot of the dataset
boxplot(dataset)

# Display the first few rows of the dataset
head(dataset)
```



The transformed dataset was then produced by calculating the principal components scores.



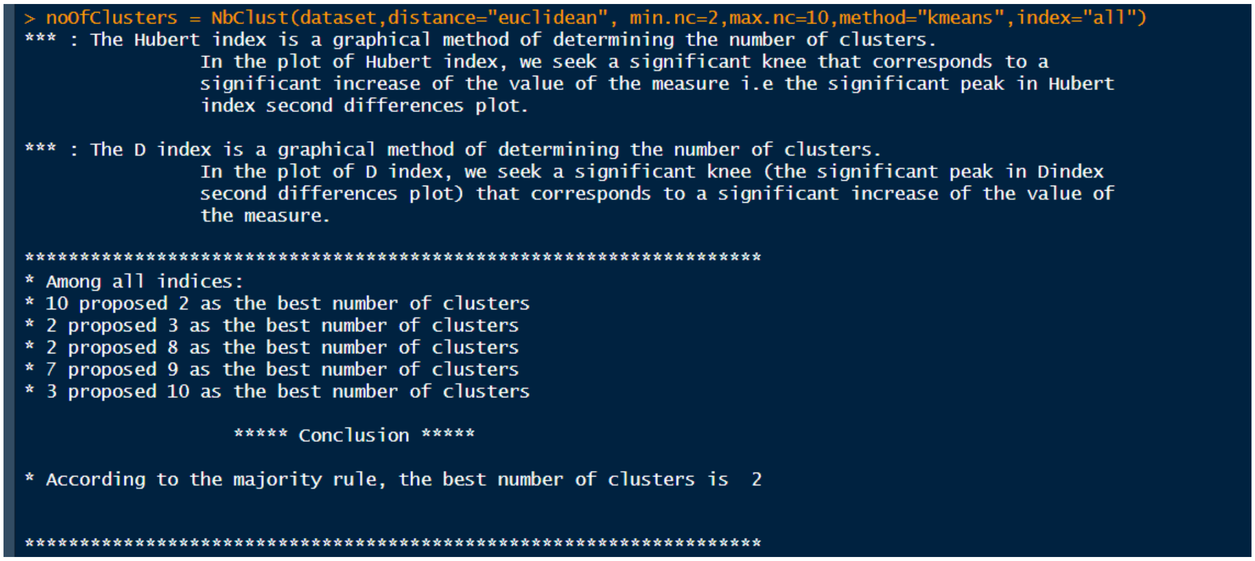
```
> # Display the first few rows of the dataset
> head(dataset)
           [,1]       [,2]        [,3]       [,4]        [,5]       [,6]
[1,]  0.3300867  0.2143583  0.99669116 -0.1718881  0.08260509  0.7264658
[2,] -1.5932235  0.4218832 -0.36865768 -0.2321468  0.69190355  0.5283311
[3,]  3.7590575 -0.1878930  0.08862669 -1.2004623  0.71880536 -0.7043615
[4,] -1.7416597  2.8217010  0.11245894 -0.3738589 -0.36814839  0.5065011
[5,]  0.5509971 -4.7659779 11.67934755 -0.1664353  3.24212257  0.2986045
[6,]  6.3985358 -3.9256947 -2.06795195  0.3840411 -0.57122670 -0.8991956
>
```

F).Using Automated tools after performing PCA method.

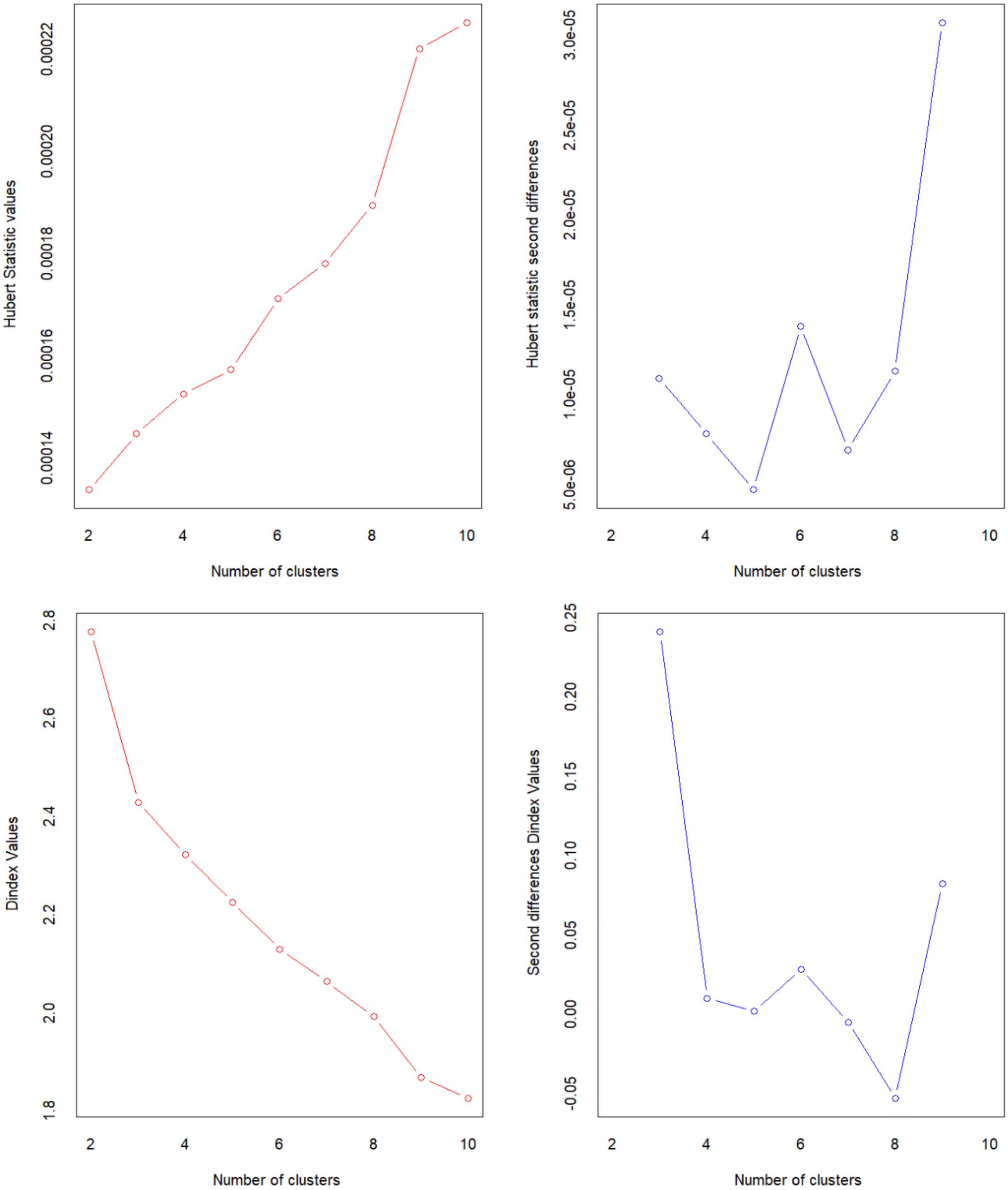
Below i have mentioned all the outputs on each of the following automated tools.

NBClust

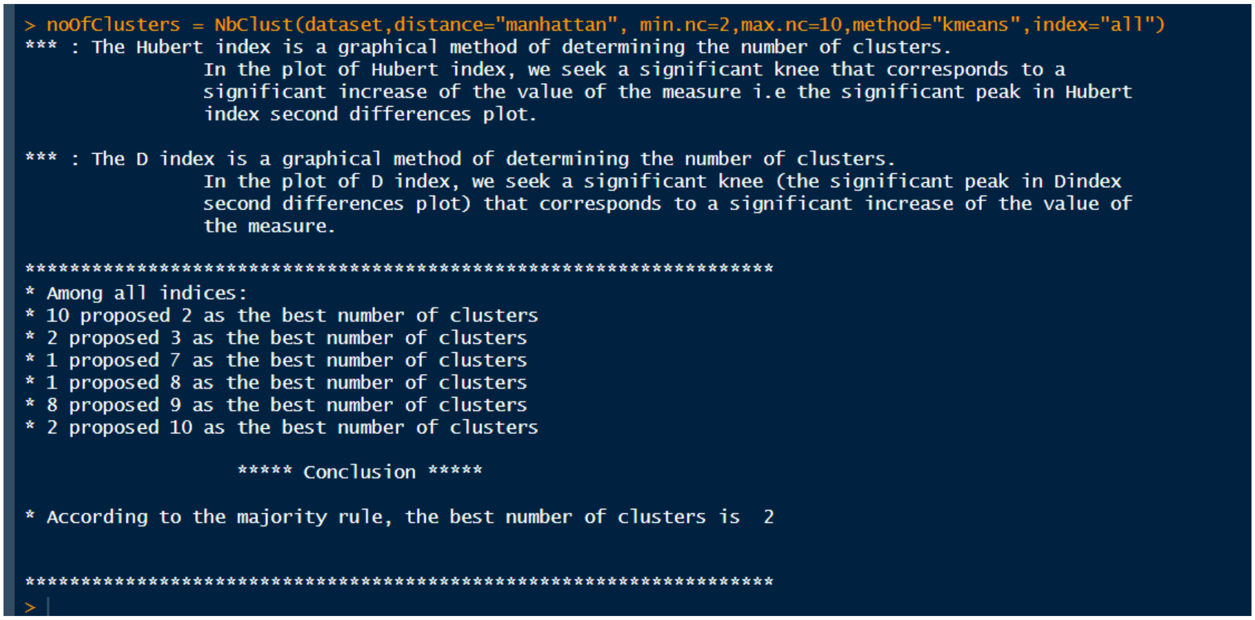
Euclidean distance



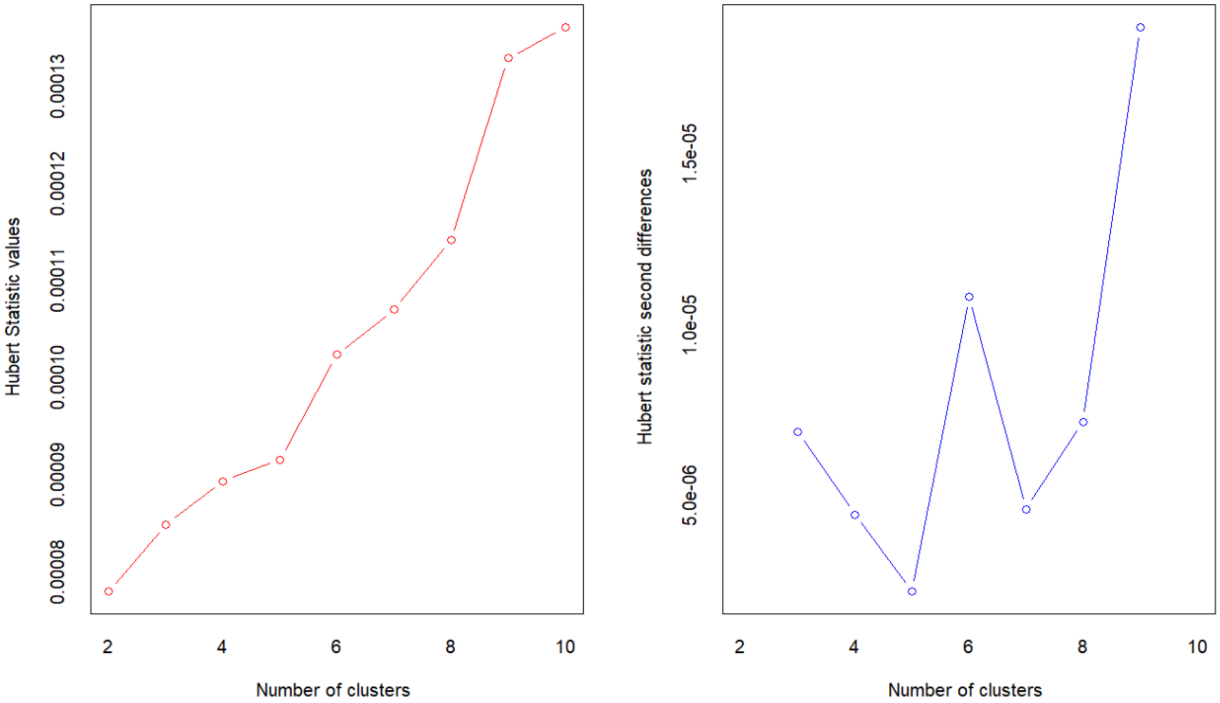
```
> noOfClusters = NbClust(dataset,distance="euclidean", min.nc=2,max.nc=10,method="kmeans",index="all")
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a
                significant increase of the value of the measure i.e the significant peak in Hubert
                index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.
                In the plot of D index, we seek a significant knee (the significant peak in Dindex
                second differences plot) that corresponds to a significant increase of the value of
                the measure.

*******************************************************************
* Among all indices:
* 10 proposed 2 as the best number of clusters
* 2 proposed 3 as the best number of clusters
* 2 proposed 8 as the best number of clusters
* 7 proposed 9 as the best number of clusters
* 3 proposed 10 as the best number of clusters

                   ***** Conclusion *****

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


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

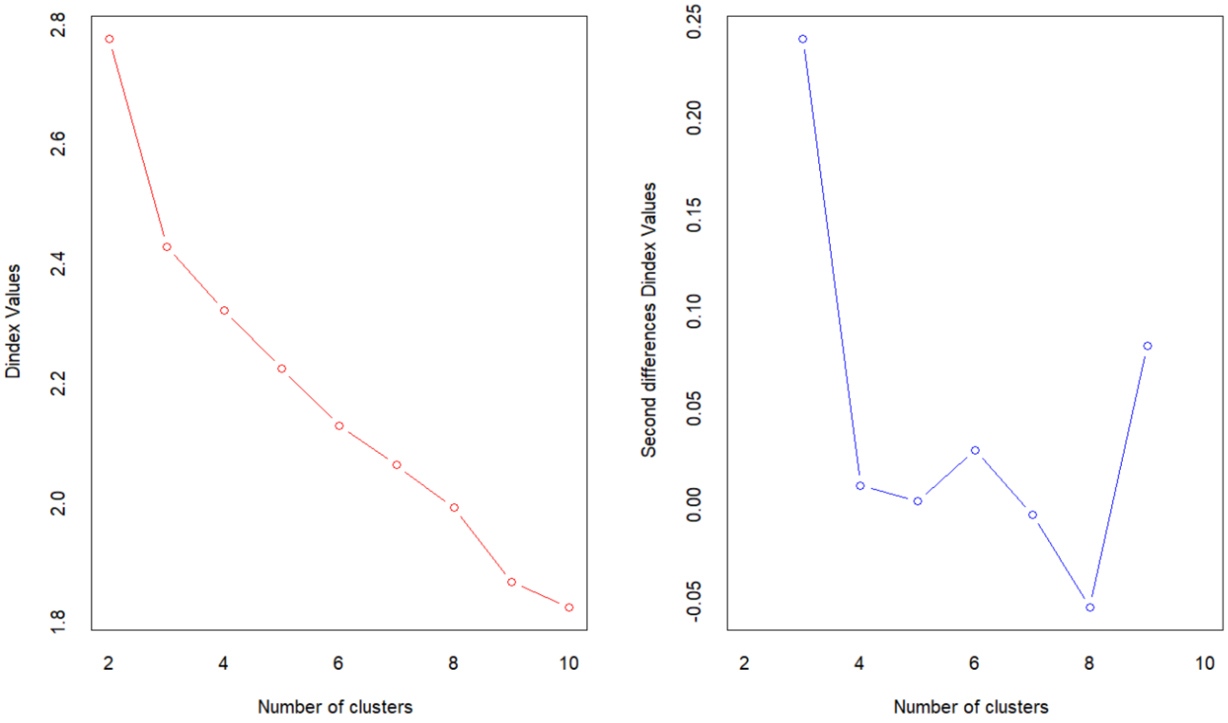


Manhattan distance

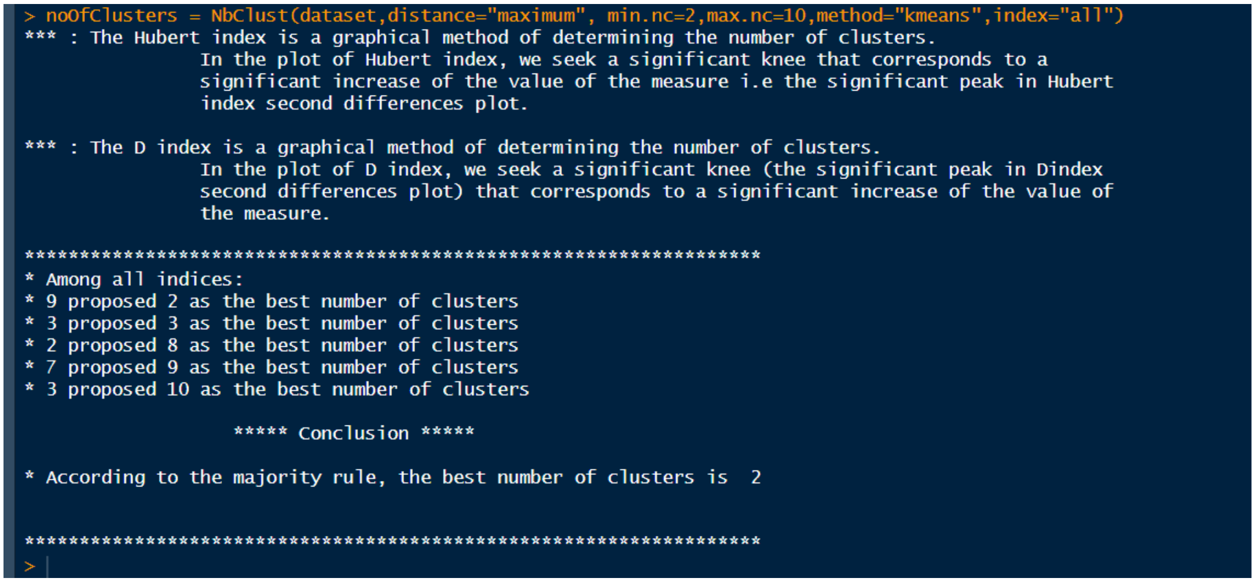


```
> noOfClusters = NbClust(dataset,distance="manhattan", min.nc=2,max.nc=10,method="kmeans",index="all")
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a 
                significant increase of the value of the measure i.e the significant peak in Hubert
                index second differences plot. 
 
*** : The D index is a graphical method of determining the number of clusters. 
                In the plot of D index, we seek a significant knee (the significant peak in Dindex
                second differences plot) that corresponds to a significant increase of the value of
                the measure. 
 
******************************************************************* 
* Among all indices:                                                
* 10 proposed 2 as the best number of clusters 
* 2 proposed 3 as the best number of clusters 
* 1 proposed 7 as the best number of clusters 
* 1 proposed 8 as the best number of clusters 
* 8 proposed 9 as the best number of clusters 
* 2 proposed 10 as the best number of clusters 

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

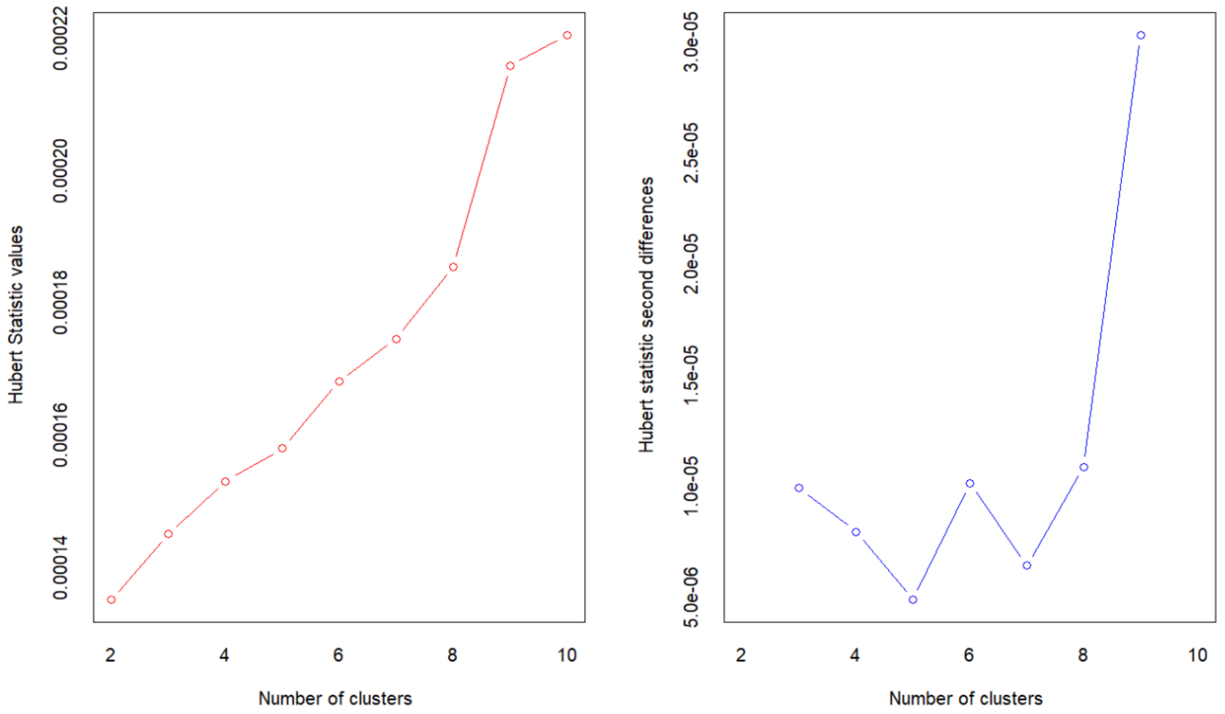


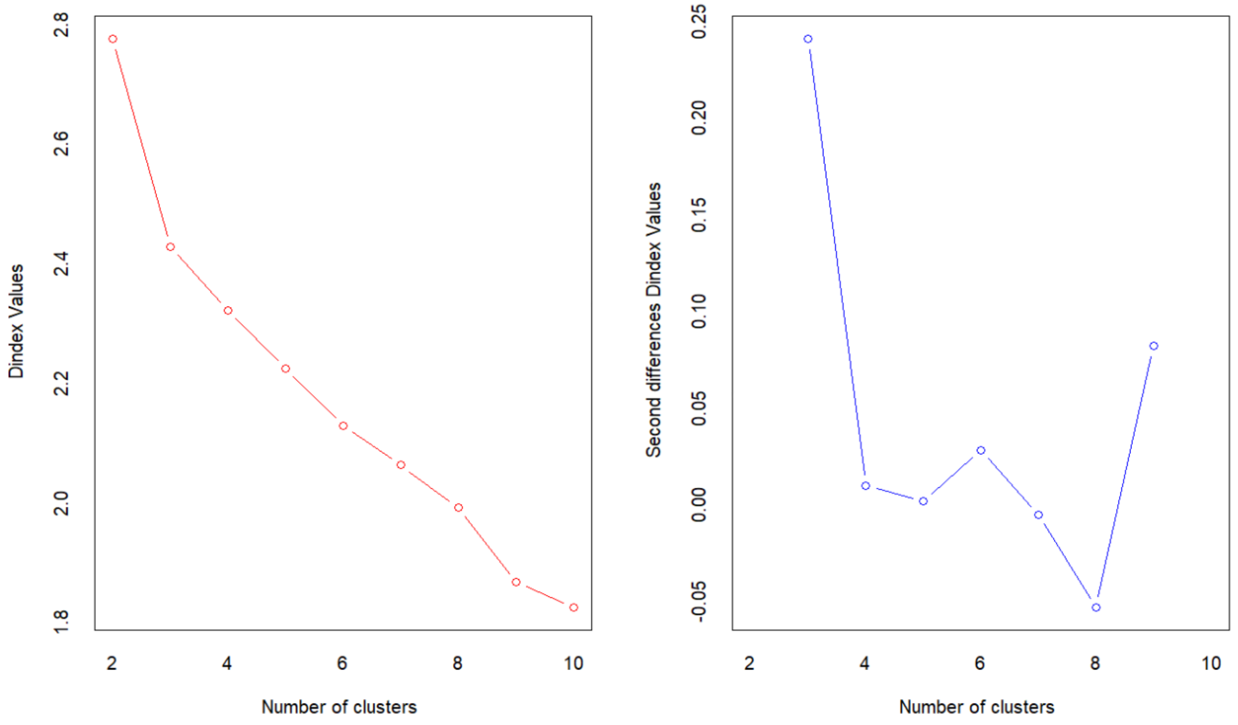


Maximum distance



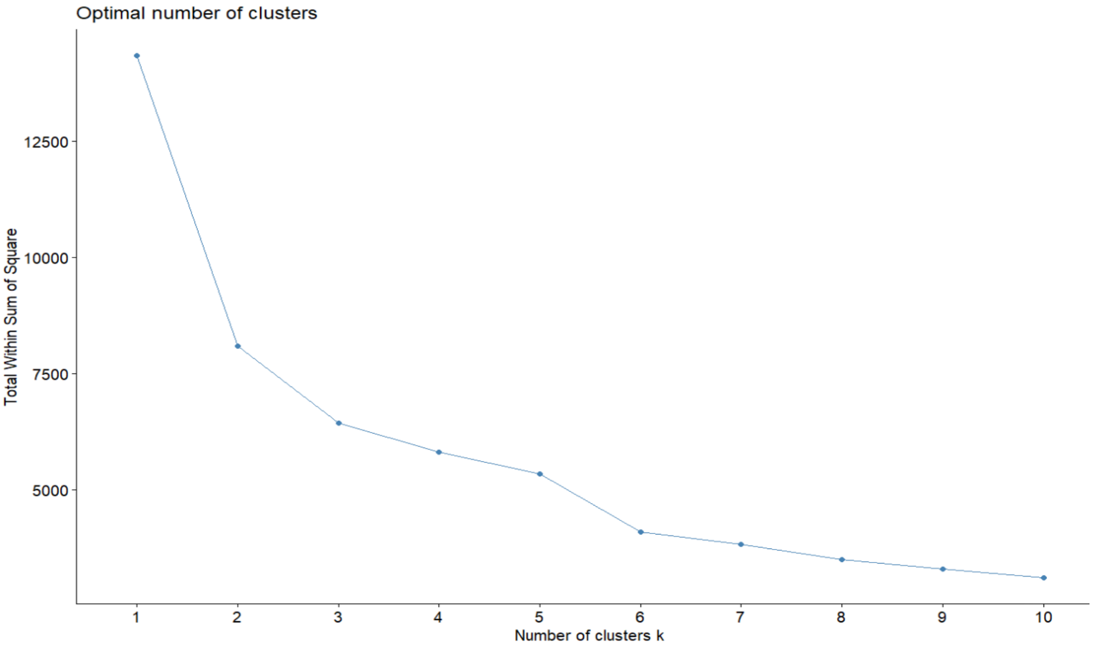
```
> noOfClusters = NbClust(dataset,distance="maximum", min.nc=2,max.nc=10,method="kmeans",index="all")
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a 
                significant increase of the value of the measure i.e the significant peak in Hubert
                index second differences plot. 
 
*** : The D index is a graphical method of determining the number of clusters. 
                In the plot of D index, we seek a significant knee (the significant peak in Dindex
                second differences plot) that corresponds to a significant increase of the value of
                the measure. 
 
******************************************************************* 
* Among all indices:                                                
* 9 proposed 2 as the best number of clusters 
* 3 proposed 3 as the best number of clusters 
* 2 proposed 8 as the best number of clusters 
* 7 proposed 9 as the best number of clusters 
* 3 proposed 10 as the best number of clusters 

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





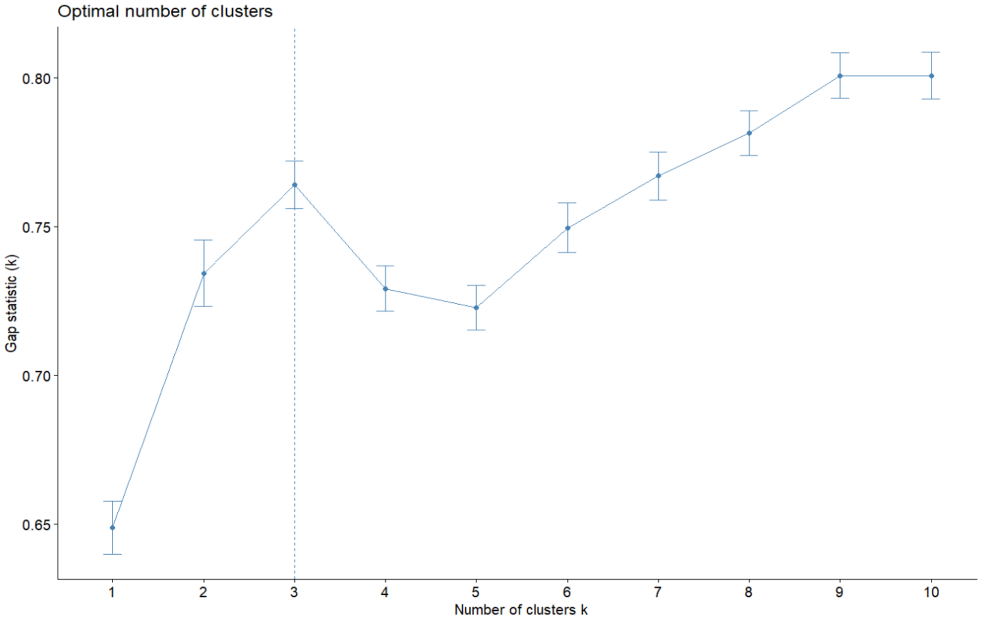
Euclidean, Manhattan, and Maximum distances were all checked with NBClust, and all three distances indicate that 2 clusters are the best fit for this transformed dataset.

Elbow



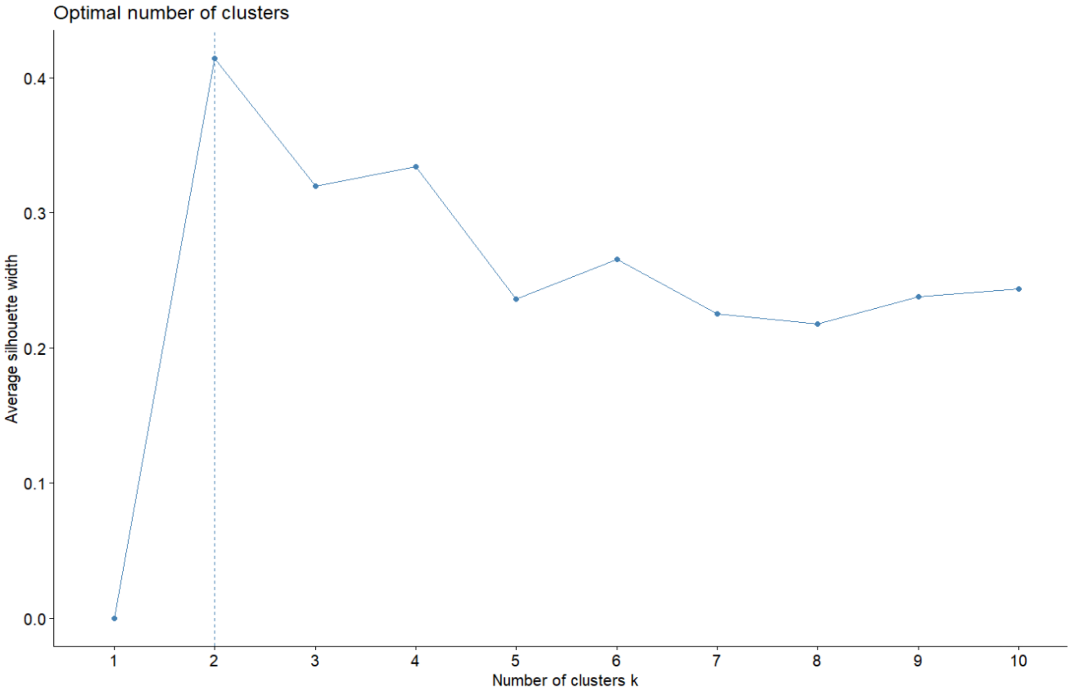
The elbow method states that the transformed dataset should be divided into three clusters. This is due to the fact that the elbow plot clearly delineates a bend or "elbow" at three clusters, indicating a significant decline in the within-cluster sum of squared error. Beyond three clusters, the performance of clustering does not significantly improve.

Gap statistics



The method described above produces 3 clusters for the transformed dataset; 3 is the number that maximizes the gap statistic value as depicted in the diagram above.

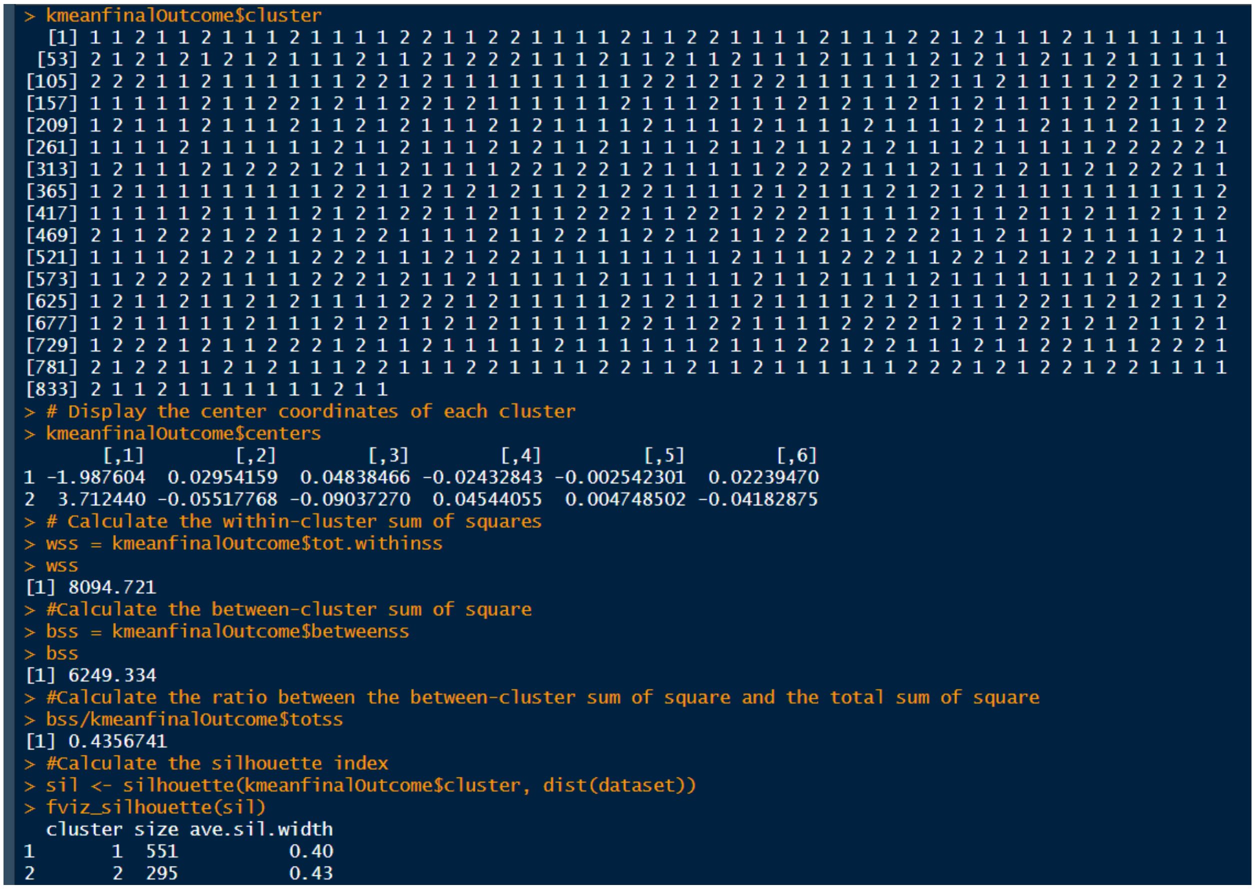
Silhouette



According to this method, 2 clusters maximize the average silhouette values, and 4 clusters are the next best number.

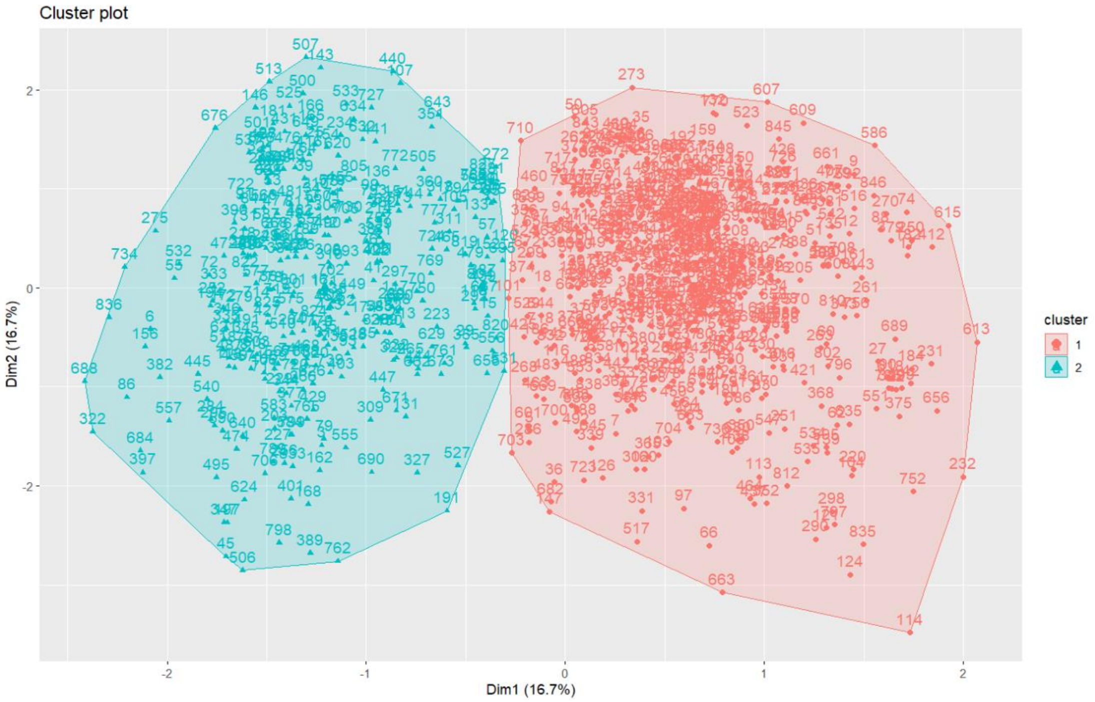
The number of clusters for k-means clustering on the transformed dataset could be either 2 or 3, according to an analysis of several automated tools. Elbow and Gap statistics recommend three clusters, while NBClust and Silhouette believe that two would be the ideal number. But for my k-means clustering, I've decided to work with just two clusters.

G).K-means clustering investigation after performing PCA method.



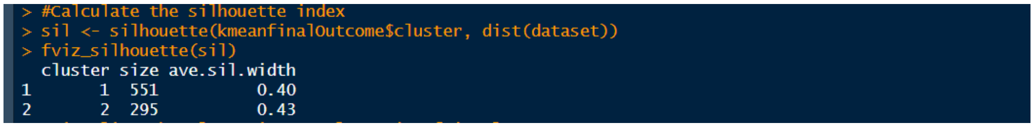
```
> kmeanfinalOutcome$cluster
  [1] 1 1 2 1 1 2 1 1 1 2 1 1 1 1 2 2 1 1 2 2 1 1 1 1 2 1 1 2 2 1 1 1 1 2 1 1 1 2 2 1 2 1 1 1 2 1 1 1 1 1 1 1
 [53] 2 1 2 1 2 1 2 1 2 1 1 1 2 1 1 2 1 2 2 2 1 1 1 2 1 1 2 1 1 2 1 1 1 2 1 1 1 1 2 1 2 1 1 2 1 1 2 1 1 1 1 1
[105] 2 2 2 1 1 2 1 1 1 1 1 1 2 2 1 2 1 1 1 1 1 1 1 1 1 2 2 1 2 1 2 2 1 1 1 1 1 1 2 1 1 2 1 1 1 1 2 2 1 2 1 2
[157] 1 1 1 1 1 2 1 1 2 2 1 2 1 1 2 2 1 2 1 1 1 1 1 1 2 1 1 1 2 1 1 1 2 1 2 1 1 2 1 1 2 1 1 1 1 1 2 2 1 1 1 1
[209] 1 2 1 1 1 2 1 1 1 2 1 1 2 1 2 1 1 1 2 1 2 1 1 1 1 2 1 1 1 1 2 1 1 1 1 2 1 1 1 1 2 1 1 2 1 1 1 2 1 1 2 2
[261] 1 1 1 1 2 1 1 1 1 1 1 2 1 1 2 1 1 1 2 1 2 1 1 2 1 1 1 1 2 1 1 2 1 1 2 1 2 1 1 1 2 1 1 1 1 1 2 2 2 2 2 1
[313] 1 2 1 1 1 2 1 2 2 2 1 2 1 1 2 1 1 1 1 2 2 1 2 2 1 2 1 1 1 1 1 2 2 2 2 1 1 1 2 1 1 1 2 1 1 2 1 2 2 2 1 1
[365] 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 2 1 2 1 2 1 1 2 1 2 2 1 1 1 1 2 1 2 1 1 1 2 1 2 1 2 1 1 1 1 1 1 1 1 1 1 2
[417] 1 1 1 1 1 2 1 1 1 1 2 1 2 1 2 2 1 1 2 1 1 1 2 2 2 1 1 2 2 1 2 2 2 1 1 1 1 1 2 1 1 1 2 1 1 2 1 1 2 1 1 2
[469] 2 1 1 2 2 2 1 2 2 1 2 1 2 2 1 1 1 1 2 1 1 2 2 1 1 2 2 1 2 1 1 2 2 2 1 1 2 2 2 1 1 2 1 1 2 1 1 1 1 2 1 1
[521] 1 1 1 1 2 1 2 2 1 1 2 2 2 1 1 1 2 1 2 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 2 2 1 1 2 2 1 2 1 1 2 2 1 1 1 2 1
[573] 1 1 2 2 2 2 1 1 1 1 2 2 2 1 2 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 2 2 1 1 2
[625] 1 2 1 1 2 1 1 2 1 2 1 1 1 1 2 2 2 1 2 1 1 1 1 1 2 1 2 1 1 1 2 1 1 1 1 2 1 2 1 1 1 1 2 2 1 1 2 1 2 1 1 2
[677] 1 2 1 1 1 1 1 2 1 1 1 2 1 2 1 1 2 1 2 1 1 1 1 1 2 2 1 1 2 2 1 1 1 1 2 2 2 2 1 2 1 1 2 2 1 2 1 2 1 1 2 1
[729] 1 2 2 2 1 2 1 1 2 2 2 1 2 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 2 2 1 2 2 1 1 1 2 1 1 2 2 1 1 1 2 2 2 1
[781] 2 1 2 2 1 1 2 1 2 1 1 1 2 2 1 1 1 2 2 1 1 1 1 2 2 1 1 2 1 1 2 1 1 1 1 1 1 2 2 2 1 2 1 2 2 1 2 2 1 1 1 1
[833] 2 1 1 2 1 1 1 1 1 1 1 2 1 1
> # Display the center coordinates of each cluster
> kmeanfinalOutcome$centers
       [,1]        [,2]        [,3]        [,4]         [,5]        [,6]
1 -1.987604  0.02954159  0.04838466 -0.02432843 -0.002542301  0.02239470
2  3.712440 -0.05517768 -0.09037270  0.04544055  0.004748502 -0.04182875
> # Calculate the within-cluster sum of squares
> wss = kmeanfinalOutcome$tot.withinss
> wss
[1] 8094.721
> #Calculate the between-cluster sum of square
> bss = kmeanfinalOutcome$betweenss
> bss
[1] 6249.334
> #Calculate the ratio between the between-cluster sum of square and the total sum of square
> bss/kmeanfinalOutcome$totss
[1] 0.4356741
> #Calculate the silhouette index
> sil <- silhouette(kmeanfinalOutcome$cluster, dist(dataset))
> fviz_silhouette(sil)
  cluster size ave.sil.width
1       1  551          0.40
2       2  295          0.43
```

The output of kmeans clustering on the dataset is shown in the image above. Based on its characteristics, each data point is categorized into either cluster 1 or cluster 2. For each feature, the center of each cluster was also identified. It was discovered that the between cluster sum of square was 6249 and the within cluster sum of square was 8094. It was calculated that there is a 0.43 ratio between the cluster sum of squares and the total sum of squares.

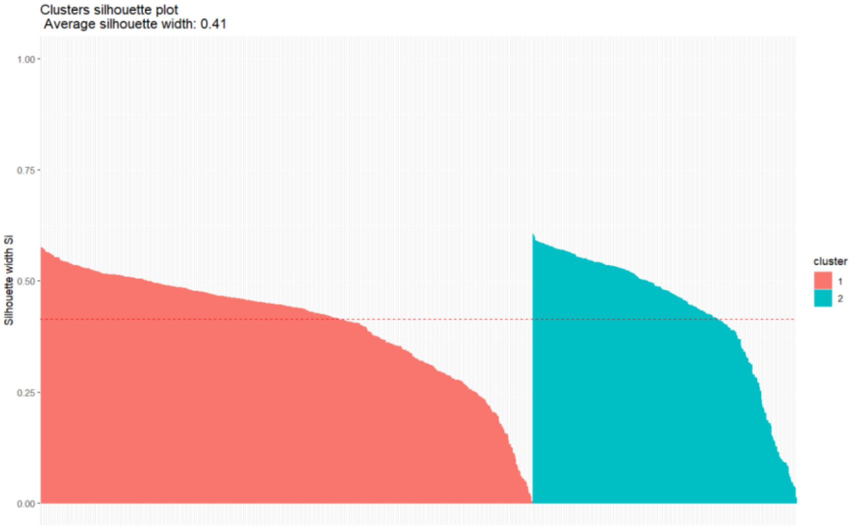


As shown in the above diagram, two clusters were used for kmeans clustering. As the plot demonstrates, two clusters were correctly recognized and distinguished.

H).Silhouette plot after performing PCA method.

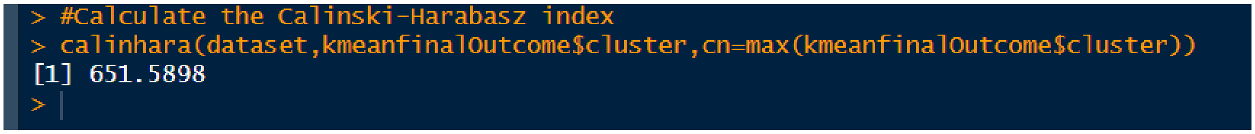


```
> #Calculate the silhouette index
> sil <- silhouette(kmeanfinalOutcome$cluster, dist(dataset))
> fviz_silhouette(sil)
  cluster size ave.sil.width
1       1  551          0.40
2       2  295          0.43
```



When plotting, the average silhouette width score for k = 2 is 0.41, which is a good value and indicates that the clusters are well separated. Additionally, we can see that both clusters' widths are greater than the average width score, proving that the clustering was done well.

I).Calinski-Harabasz Index performing PCA method.



```
> #Calculate the Calinski-Harabasz index
> calinhara(dataset,kmeanfinalOutcome$cluster,cn=max(kmeanfinalOutcome$cluster))
[1] 651.5898
> 
```

A data point's proximity to its cluster relative to other clusters is indicated by the Calinski-Harabasz Index. K = 2 yields 651, indicating that the data points are nearer the cluster.

Energy Forecasting Part

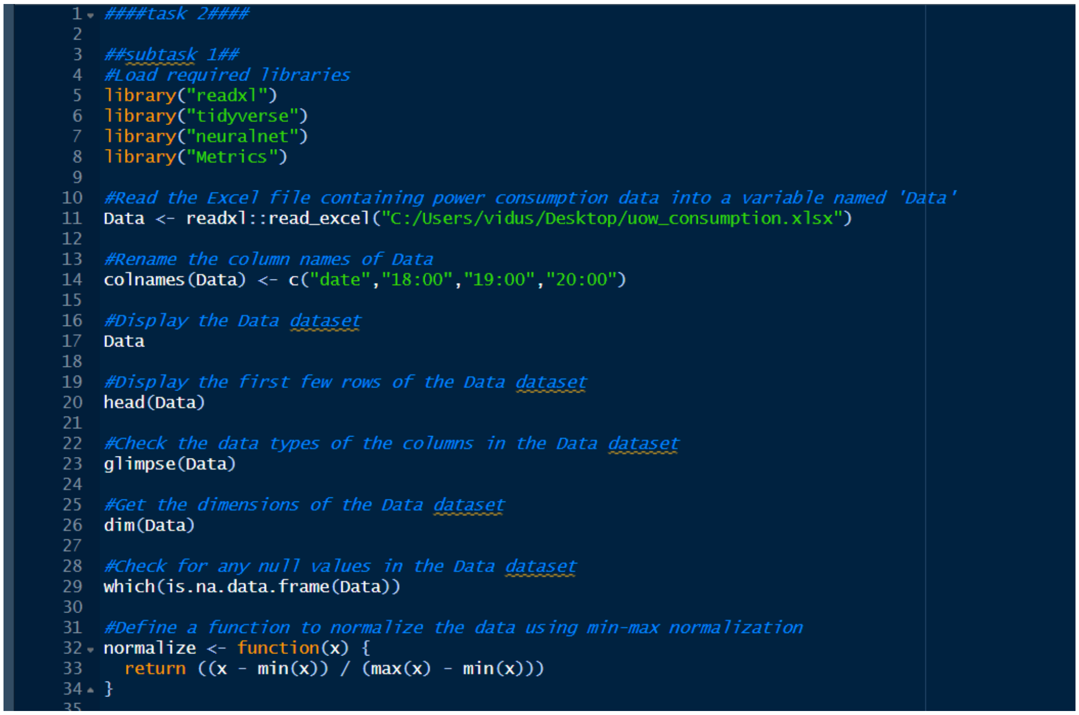
A).AR approach Definition

The dataset "UoW_consumption.xlsx" contains energy values for every day between January 1, 2018, and April 15, 2019, and is used to predict the current day's 20th hour energy value using the previous day's 20th hour value. In order to create the input-output matrix for energy forecasting, the input variables are lagged, with the previous day's energy value acting as the input and the current day's energy value acting as the output node. Using the "lag()" function in R, one can obtain the lagged time series data. It is necessary to use an autoregressive time series forecasting method because the time series data is constantly changing. Autoregression uses historical data to predict future values. Moving average, on the other hand, is an additional strategy that forecasts future values using errors from previous forecasts.

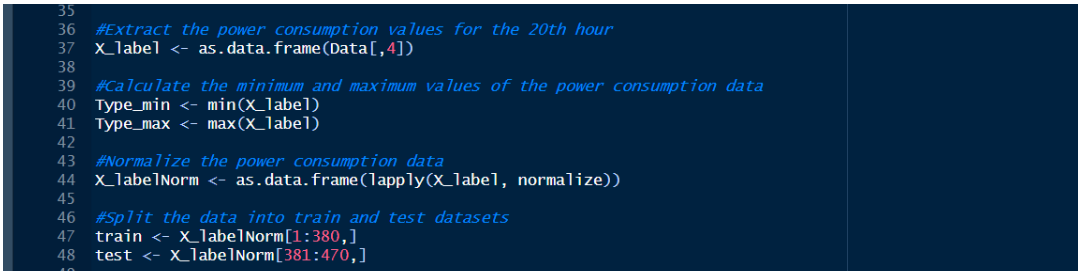
Each t value, where t is the number of prior days used as input, can be used to create an input-output matrix. For instance, t1 only has the previous day's energy value as input and the current day's energy value as output, t2 has the energy values of the previous two days and the current day as output, t3 has the energy values of the previous three days, and so on. The time series moves back that many days as the t value rises and the number of inputs rises along with it.

B).Constructing an input/output matrix (I/O)

The "UoW_consumption.xlsx" dataset, which has 470 rows and 4 columns, was initially imported. After that, no missing values were discovered despite a scan of the dataset. Then, because it was the only column necessary for the task, the fourth column, "20th hour," was extracted from the other columns and saved in a variable called "dataset." The dataset was then normalized using the min-max method. The dataset was split into two sets, the training set and the test set, after normalization was finished.

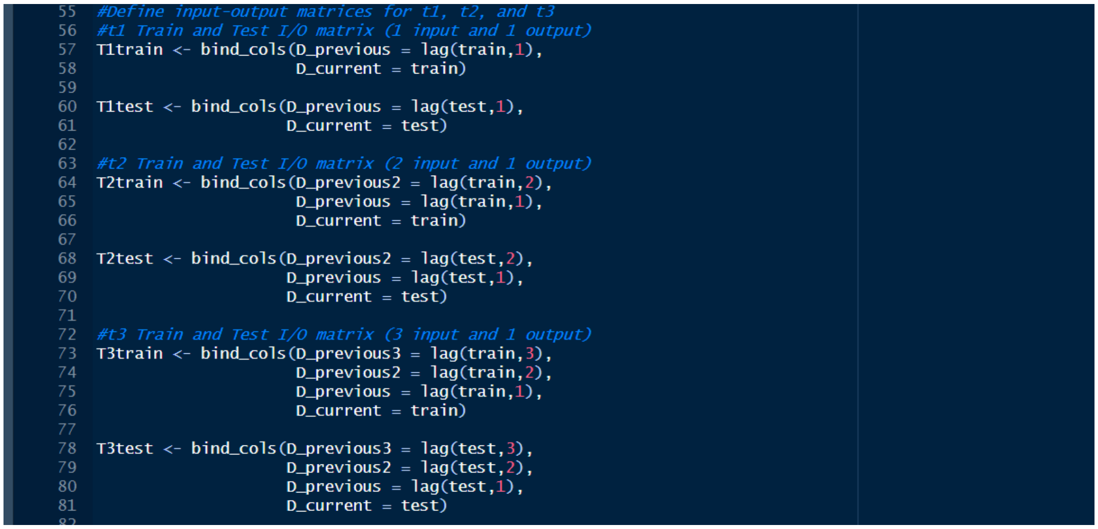


```r
####task 2####

##subtask 1##
#Load required libraries
library("readxl")
library("tidyverse")
library("neuralnet")
library("Metrics")

#Read the Excel file containing power consumption data into a variable named 'Data'
Data <- readxl::read_excel("C:/Users/vidus/Desktop/uow_consumption.xlsx")

#Rename the column names of Data
colnames(Data) <- c("date","18:00","19:00","20:00")

#Display the Data dataset
Data

#Display the first few rows of the Data dataset
head(Data)

#Check the data types of the columns in the Data dataset
glimpse(Data)

#Get the dimensions of the Data dataset
dim(Data)

#Check for any null values in the Data dataset
which(is.na.data.frame(Data))

#Define a function to normalize the data using min-max normalization
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
```

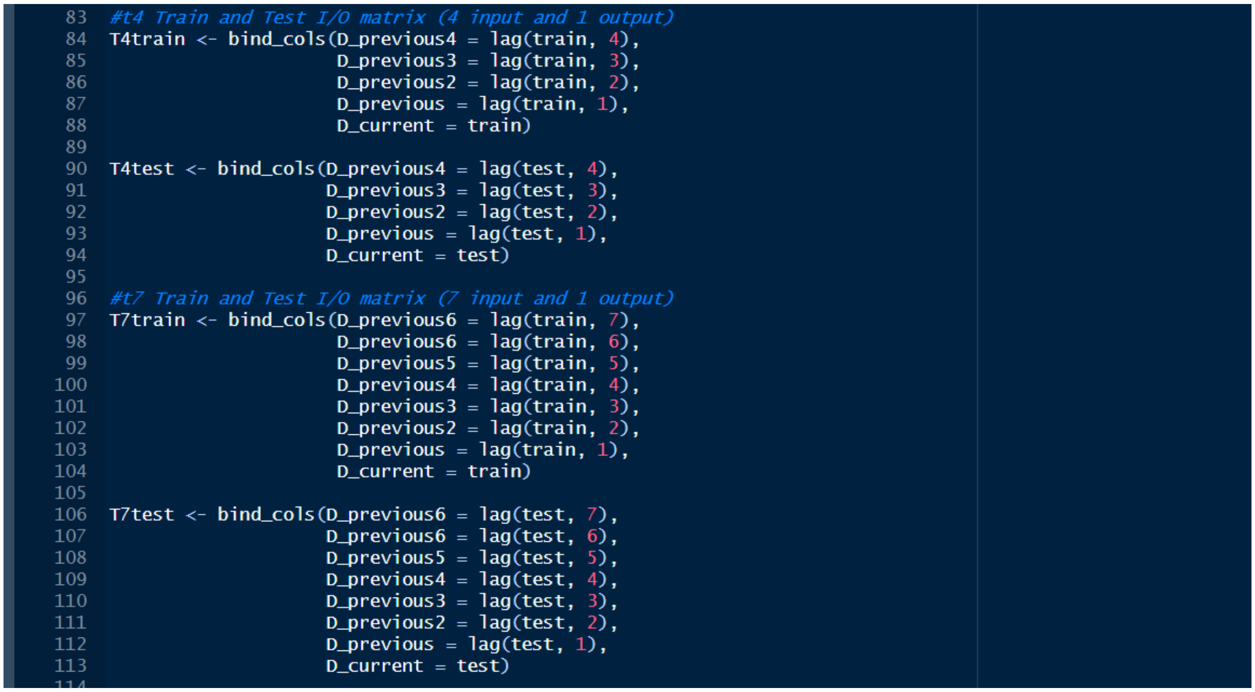


```r
#Extract the power consumption values for the 20th hour
X_label <- as.data.frame(Data[,4])

#Calculate the minimum and maximum values of the power consumption data
Type_min <- min(X_label)
Type_max <- max(X_label)

#Normalize the power consumption data
X_labelNorm <- as.data.frame(lapply(X_label, normalize))

#Split the data into train and test datasets
train <- X_labelNorm[1:380,]
test <- X_labelNorm[381:470,]
```

Below i have mentioned the code for input output matrix for T1 to T7



```r
#Define input-output matrices for t1, t2, and t3
#t1 Train and Test I/O matrix (1 input and 1 output)
T1train <- bind_cols(D_previous = lag(train,1),
                     D_current = train)

T1test <- bind_cols(D_previous = lag(test,1),
                    D_current = test)

#t2 Train and Test I/O matrix (2 input and 1 output)
T2train <- bind_cols(D_previous2 = lag(train,2),
                     D_previous = lag(train,1),
                     D_current = train)

T2test <- bind_cols(D_previous2 = lag(test,2),
                    D_previous = lag(test,1),
                    D_current = test)

#t3 Train and Test I/O matrix (3 input and 1 output)
T3train <- bind_cols(D_previous3 = lag(train,3),
                     D_previous2 = lag(train,2),
                     D_previous = lag(train,1),
                     D_current = train)

T3test <- bind_cols(D_previous3 = lag(test,3),
                    D_previous2 = lag(test,2),
                    D_previous = lag(test,1),
                    D_current = test)
```



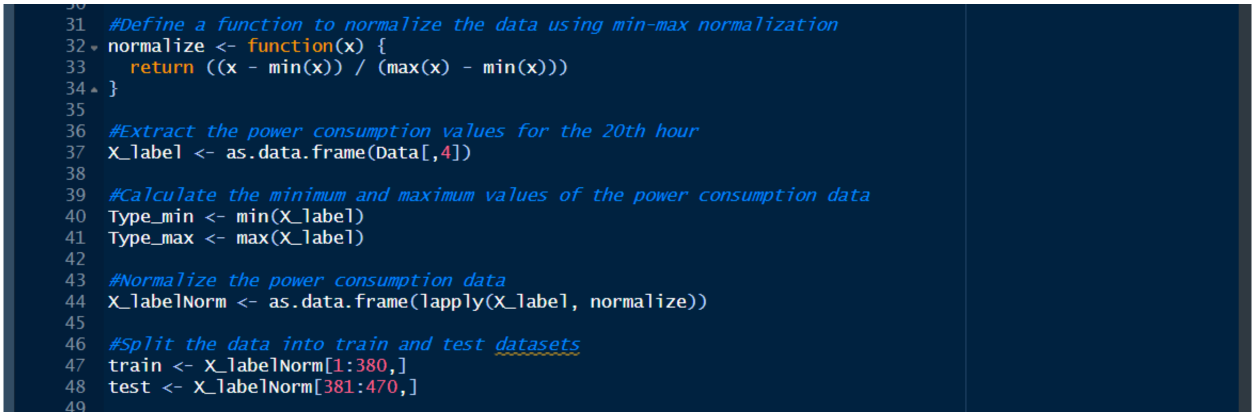
```
#t4 Train and Test I/O matrix (4 input and 1 output)
T4train <- bind_cols(D_previous4 = lag(train, 4),
                     D_previous3 = lag(train, 3),
                     D_previous2 = lag(train, 2),
                     D_previous = lag(train, 1),
                     D_current = train)

T4test <- bind_cols(D_previous4 = lag(test, 4),
                    D_previous3 = lag(test, 3),
                    D_previous2 = lag(test, 2),
                    D_previous = lag(test, 1),
                    D_current = test)

#t7 Train and Test I/O matrix (7 input and 1 output)
T7train <- bind_cols(D_previous6 = lag(train, 7),
                     D_previous6 = lag(train, 6),
                     D_previous5 = lag(train, 5),
                     D_previous4 = lag(train, 4),
                     D_previous3 = lag(train, 3),
                     D_previous2 = lag(train, 2),
                     D_previous = lag(train, 1),
                     D_current = train)

T7test <- bind_cols(D_previous6 = lag(test, 7),
                    D_previous6 = lag(test, 6),
                    D_previous5 = lag(test, 5),
                    D_previous4 = lag(test, 4),
                    D_previous3 = lag(test, 3),
                    D_previous2 = lag(test, 2),
                    D_previous = lag(test, 1),
                    D_current = test)
```

All t values, from t1 to t7, are used to create time-delayed data. The previous columns from these matrices serve as the inputs, and the current column serves as the output. The training and test sets are lagged for each t value.

C).Normalization procedure

The dataset was min-max normalized in order to prevent potential problems brought on by different feature value ranges. Without normalization, the differences in feature value ranges may favor features with larger value ranges over those with smaller ranges, which could result in squeezed data points and poor performance of machine learning models. By bringing all the features to a similar scale, normalization can spread out the data points, enabling more effective and efficient learning of machine learning models.

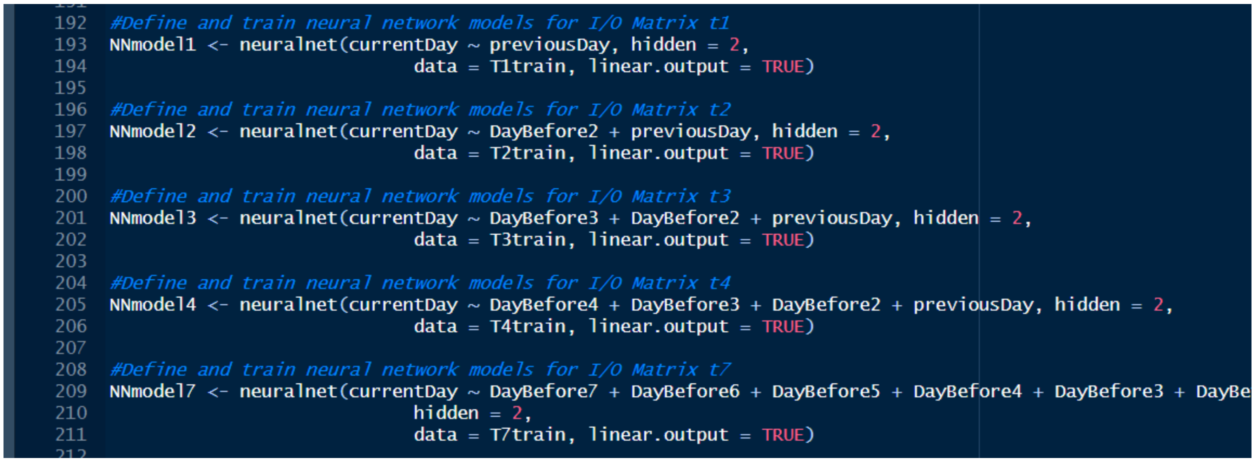
In particular, min-max normalization sets the range of all the features to 0–1, with 0 representing the smallest value and 1 representing the largest. Since unnormalized data input and incorrect activation function output can slow learning and result in prediction errors, normalizing data is especially crucial for neural networks.



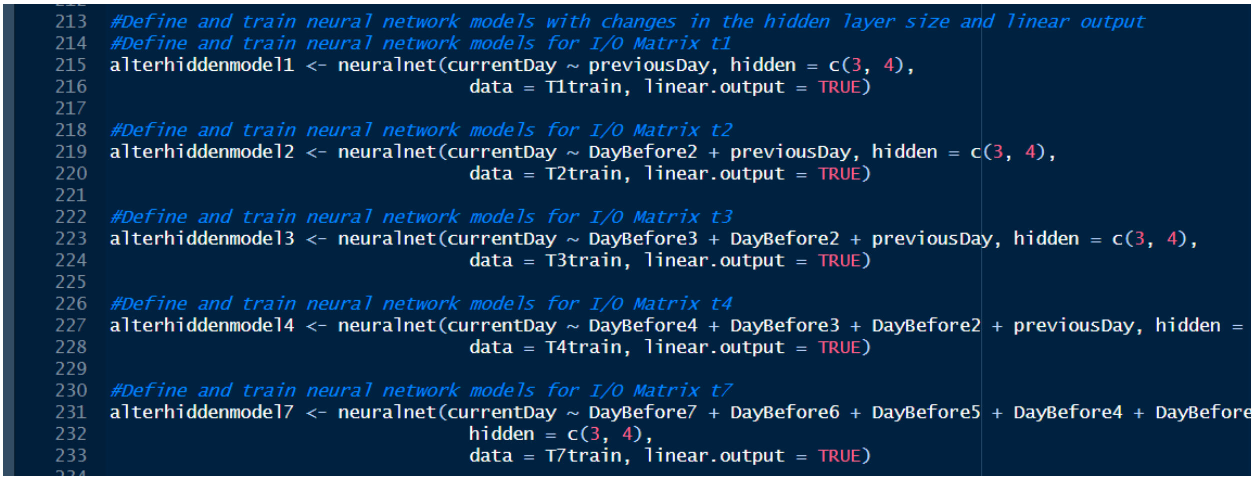
```r
#Define a function to normalize the data using min-max normalization
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

#Extract the power consumption values for the 20th hour
X_label <- as.data.frame(Data[,4])

#Calculate the minimum and maximum values of the power consumption data
Type_min <- min(X_label)
Type_max <- max(X_label)

#Normalize the power consumption data
X_labelNorm <- as.data.frame(lapply(X_label, normalize))

#Split the data into train and test datasets
train <- X_labelNorm[1:380,]
test <- X_labelNorm[381:470,]
```

D).Training phase

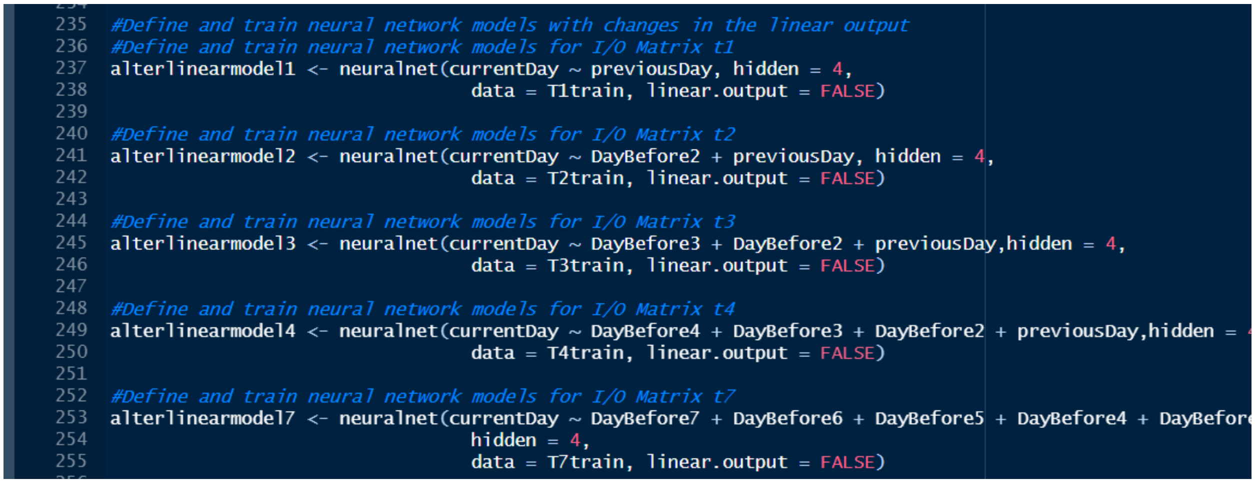
Both train and test lagged datasets were created for t1, t2, t3, t4, and t7 in order to make it easier to predict current day values using historical data. To accomplish this, five neural network models for each of these time values were built, using the previous data as input nodes and the current day value as the output node. In total 15 models were built with different hidden layers and linear output values.



```r
#Define and train neural network models for I/O Matrix t1
NNmodel1 <- neuralnet(currentDay ~ previousDay, hidden = 2,
                      data = T1train, linear.output = TRUE)

#Define and train neural network models for I/O Matrix t2
NNmodel2 <- neuralnet(currentDay ~ DayBefore2 + previousDay, hidden = 2,
                      data = T2train, linear.output = TRUE)

#Define and train neural network models for I/O Matrix t3
NNmodel3 <- neuralnet(currentDay ~ DayBefore3 + DayBefore2 + previousDay, hidden = 2,
                      data = T3train, linear.output = TRUE)

#Define and train neural network models for I/O Matrix t4
NNmodel4 <- neuralnet(currentDay ~ DayBefore4 + DayBefore3 + DayBefore2 + previousDay, hidden = 2,
                      data = T4train, linear.output = TRUE)

#Define and train neural network models for I/O Matrix t7
NNmodel7 <- neuralnet(currentDay ~ DayBefore7 + DayBefore6 + DayBefore5 + DayBefore4 + DayBefore3 + DayBe
                      hidden = 2,
                      data = T7train, linear.output = TRUE)
```

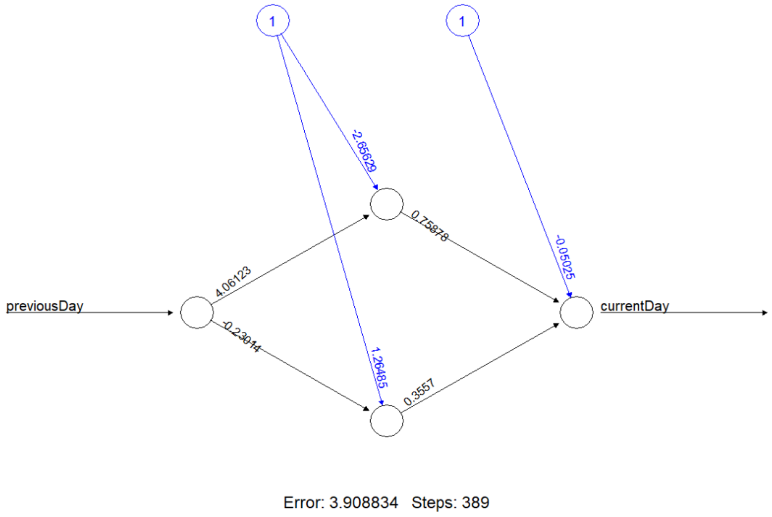


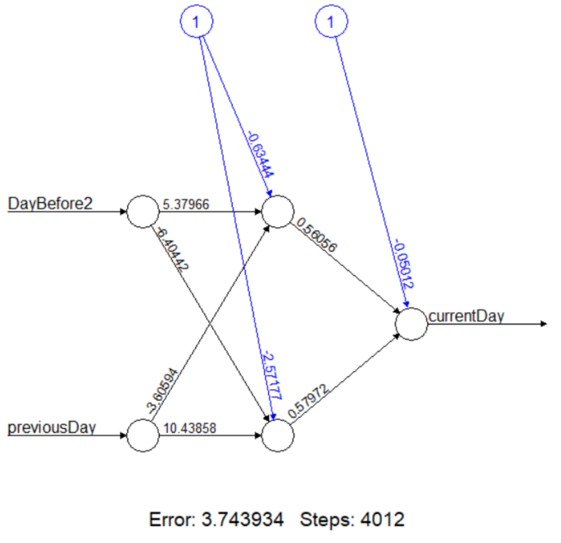
```r
#Define and train neural network models with changes in the hidden layer size and linear output
#Define and train neural network models for I/O Matrix t1
alterhiddenmodel1 <- neuralnet(currentDay ~ previousDay, hidden = c(3, 4),
                               data = T1train, linear.output = TRUE)

#Define and train neural network models for I/O Matrix t2
alterhiddenmodel2 <- neuralnet(currentDay ~ DayBefore2 + previousDay, hidden = c(3, 4),
                               data = T2train, linear.output = TRUE)

#Define and train neural network models for I/O Matrix t3
alterhiddenmodel3 <- neuralnet(currentDay ~ DayBefore3 + DayBefore2 + previousDay, hidden = c(3, 4),
                               data = T3train, linear.output = TRUE)

#Define and train neural network models for I/O Matrix t4
alterhiddenmodel4 <- neuralnet(currentDay ~ DayBefore4 + DayBefore3 + DayBefore2 + previousDay, hidden =
                               data = T4train, linear.output = TRUE)

#Define and train neural network models for I/O Matrix t7
alterhiddenmodel7 <- neuralnet(currentDay ~ DayBefore7 + DayBefore6 + DayBefore5 + DayBefore4 + DayBefore
                               hidden = c(3, 4),
                               data = T7train, linear.output = TRUE)
```

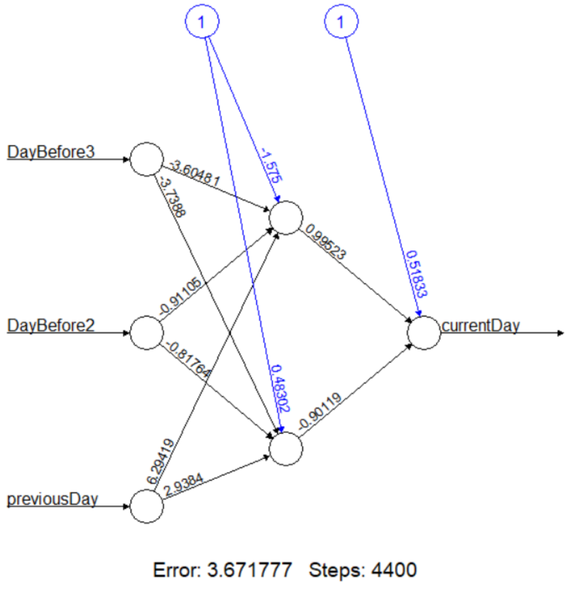


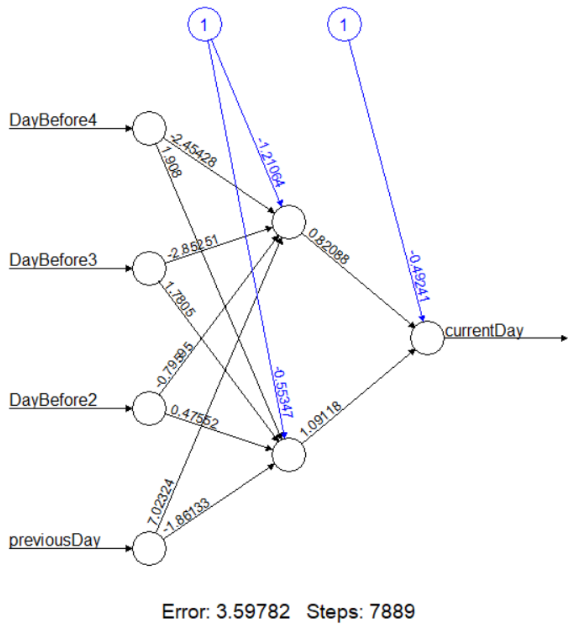
```r
#Define and train neural network models with changes in the linear output
#Define and train neural network models for I/O Matrix t1
alterlinearmodel1 <- neuralnet(currentDay ~ previousDay, hidden = 4,
                               data = T1train, linear.output = FALSE)

#Define and train neural network models for I/O Matrix t2
alterlinearmodel2 <- neuralnet(currentDay ~ DayBefore2 + previousDay, hidden = 4,
                               data = T2train, linear.output = FALSE)

#Define and train neural network models for I/O Matrix t3
alterlinearmodel3 <- neuralnet(currentDay ~ DayBefore3 + DayBefore2 + previousDay,hidden = 4,
                               data = T3train, linear.output = FALSE)

#Define and train neural network models for I/O Matrix t4
alterlinearmodel4 <- neuralnet(currentDay ~ DayBefore4 + DayBefore3 + DayBefore2 + previousDay,hidden =
                               data = T4train, linear.output = FALSE)

#Define and train neural network models for I/O Matrix t7
alterlinearmodel7 <- neuralnet(currentDay ~ DayBefore7 + DayBefore6 + DayBefore5 + DayBefore4 + DayBefore
                               hidden = 4,
                               data = T7train, linear.output = FALSE)
```

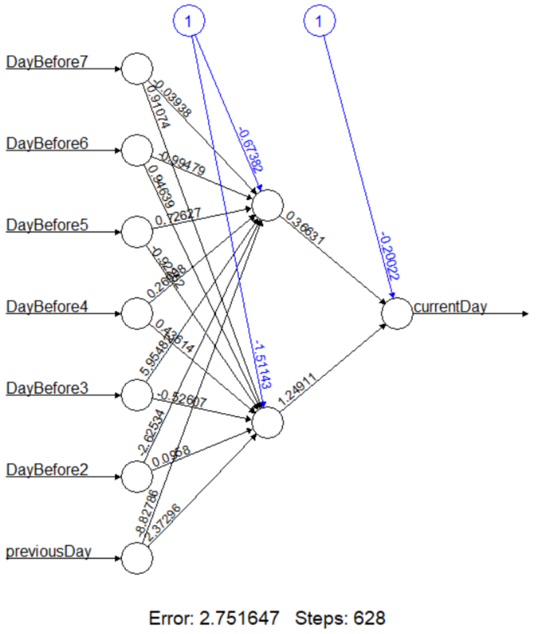
Only one hidden layer with two nodes and a true linear output was provided for the first five models.




1
1
-0.63444
DayBefore2
5.37966
-6.40442
0.56056
-0.05012
currentDay
-2.57177
-3.60594
0.57972
previousDay
10.43858
Error: 3.743934 Steps: 4012

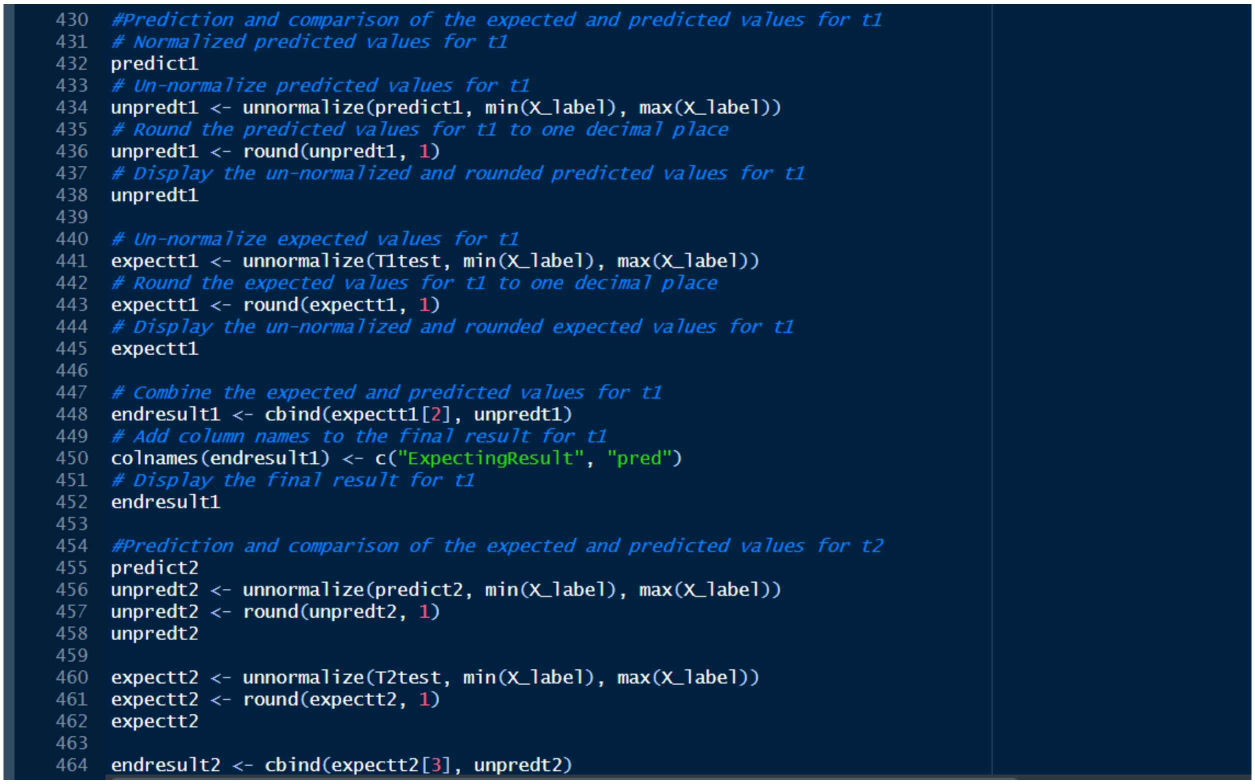

1
1
DayBefore3
-3.60481
-3.7388
-1.575
0.99523
0.51833
-0.91105
DayBefore2
currentDay
-0.81764
0.48302
-0.90119
6.29419
2.9384
previousDay
Error: 3.671777 Steps: 4400





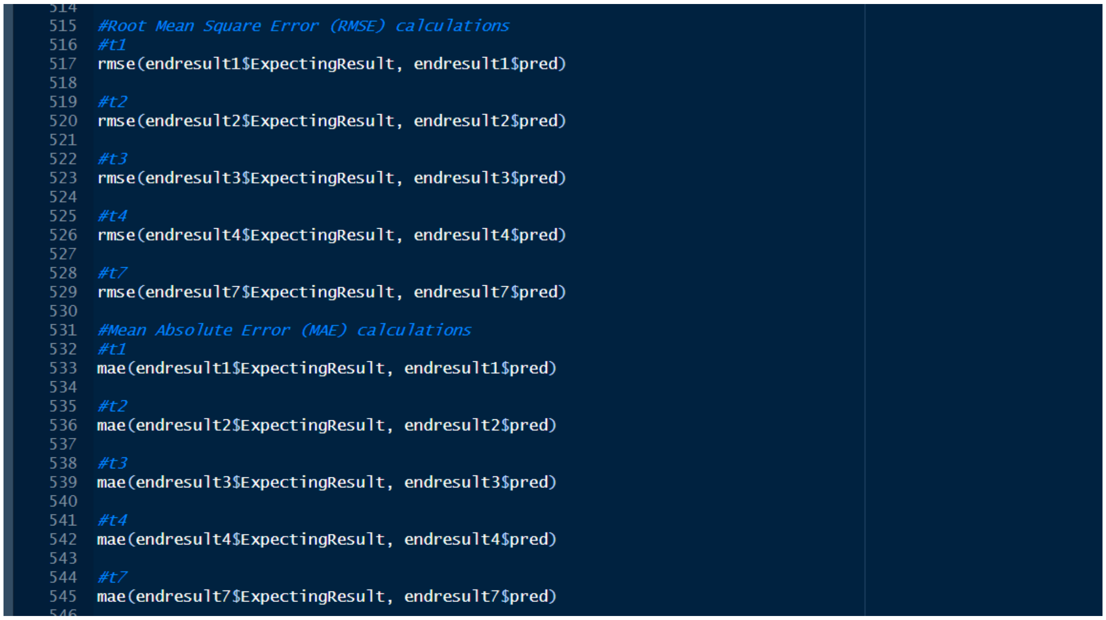
The plot shown above shows 1 input node for t1, 2 input nodes for t2, 3 input nodes for t3, 4 input nodes for t4, and 7 input nodes for t7.However, every model produces the current day as output.

Predictions are made using the lagged test sets and the aforementioned code. In order to make predictions for new values that were not used during training, the trained neural network is given input nodes from the test set in this process. To be more precise, only the input nodes from the test set are used; the output node is kept separate.

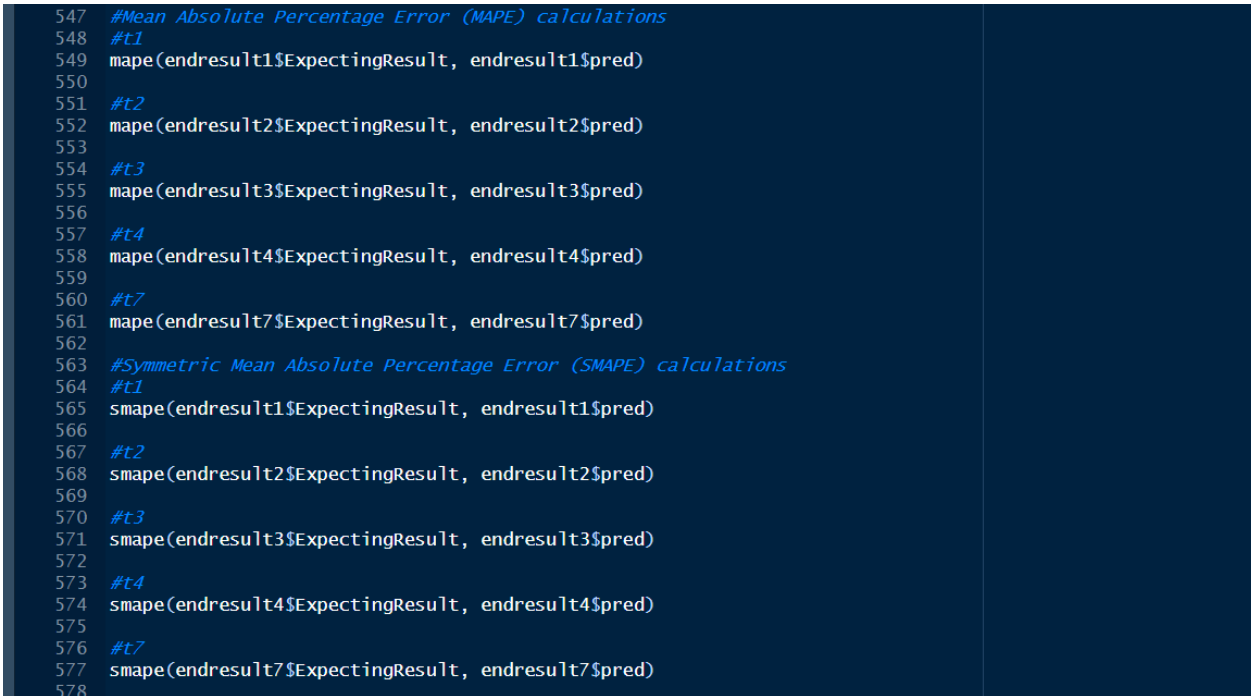


```
#Prediction and comparison of the expected and predicted values for t1
# Normalized predicted values for t1
predict1
# Un-normalize predicted values for t1
unpredt1 <- unnormalize(predict1, min(X_label), max(X_label))
# Round the predicted values for t1 to one decimal place
unpredt1 <- round(unpredt1, 1)
# Display the un-normalized and rounded predicted values for t1
unpredt1

# Un-normalize expected values for t1
expectt1 <- unnormalize(T1test, min(X_label), max(X_label))
# Round the expected values for t1 to one decimal place
expectt1 <- round(expectt1, 1)
# Display the un-normalized and rounded expected values for t1
expectt1

# Combine the expected and predicted values for t1
endresult1 <- cbind(expectt1[2], unpredt1)
# Add column names to the final result for t1
colnames(endresult1) <- c("ExpectingResult", "pred")
# Display the final result for t1
endresult1

#Prediction and comparison of the expected and predicted values for t2
predict2
unpredt2 <- unnormalize(predict2, min(X_label), max(X_label))
unpredt2 <- round(unpredt2, 1)
unpredt2

expectt2 <- unnormalize(T2test, min(X_label), max(X_label))
expectt2 <- round(expectt2, 1)
expectt2

endresult2 <- cbind(expectt2[3], unpredt2)
```

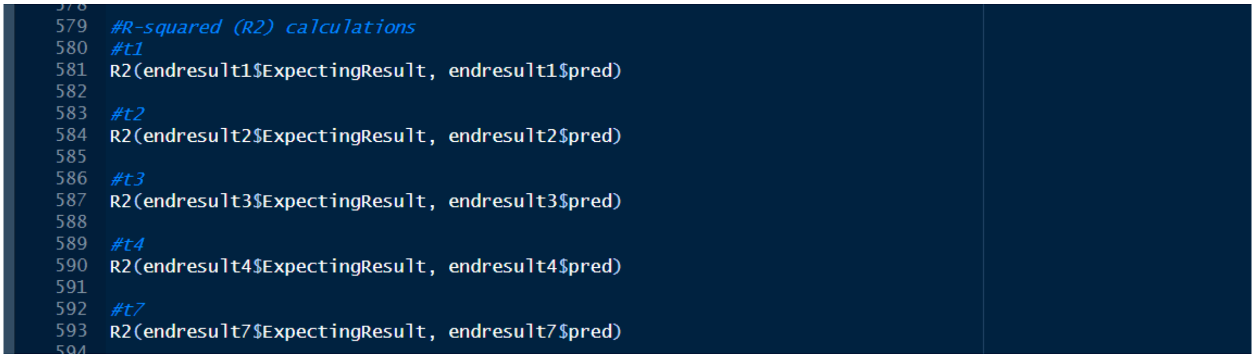
The prediction and testing set were first normalized, and then they were un-normalized to make them clearer and simpler to read. A comparison between the predicted and actual test set output was possible using the unnormalized data. To make comparisons easier and to improve readability, the data was denormalized. Following the comparison and prediction, the various t values were subjected to the application of RMSE, MAE, MAPE, and sMAPE evaluation metrics.



```
#Root Mean Square Error (RMSE) calculations
#t1
rmse(endresult1$ExpectingResult, endresult1$pred)

#t2
rmse(endresult2$ExpectingResult, endresult2$pred)

#t3
rmse(endresult3$ExpectingResult, endresult3$pred)

#t4
rmse(endresult4$ExpectingResult, endresult4$pred)

#t7
rmse(endresult7$ExpectingResult, endresult7$pred)

#Mean Absolute Error (MAE) calculations
#t1
mae(endresult1$ExpectingResult, endresult1$pred)

#t2
mae(endresult2$ExpectingResult, endresult2$pred)

#t3
mae(endresult3$ExpectingResult, endresult3$pred)

#t4
mae(endresult4$ExpectingResult, endresult4$pred)

#t7
mae(endresult7$ExpectingResult, endresult7$pred)
```



```
#Mean Absolute Percentage Error (MAPE) calculations
#t1
mape(endresult1$ExpectingResult, endresult1$pred)

#t2
mape(endresult2$ExpectingResult, endresult2$pred)

#t3
mape(endresult3$ExpectingResult, endresult3$pred)

#t4
mape(endresult4$ExpectingResult, endresult4$pred)

#t7
mape(endresult7$ExpectingResult, endresult7$pred)

#Symmetric Mean Absolute Percentage Error (SMAPE) calculations
#t1
smape(endresult1$ExpectingResult, endresult1$pred)

#t2
smape(endresult2$ExpectingResult, endresult2$pred)

#t3
smape(endresult3$ExpectingResult, endresult3$pred)

#t4
smape(endresult4$ExpectingResult, endresult4$pred)

#t7
smape(endresult7$ExpectingResult, endresult7$pred)
```



```
#R-squared (R2) calculations
#t1
R2(endresult1$ExpectingResult, endresult1$pred)

#t2
R2(endresult2$ExpectingResult, endresult2$pred)

#t3
R2(endresult3$ExpectingResult, endresult3$pred)

#t4
R2(endresult4$ExpectingResult, endresult4$pred)

#t7
R2(endresult7$ExpectingResult, endresult7$pred)
```

A total of 15 neural network models were created using the same procedure as described above. These models included five (t1 to t4 and t7) with a hidden node of 4, five (t1 to t4 and t7) with two hidden layers containing three and four nodes, and five (t1 to t4 and t7) with a false linear output and hidden nodes set at four.

E).Four statistical indices

Root Mean Squared Error (RMSE), which evaluates the average difference between the predicted and actual values, was one of the evaluation metrics used in the analysis. An increased RMSE suggests that the regression analysis might not have been carried out to its full potential.

Mean Absolute Error, which is frequently used to gauge the accuracy of continuous variables, was another metric used in the analysis. In order to calculate the overall accuracy, it calculates the absolute difference between the predicted and actual values and averaging them.

To determine the average number of incorrect predictions, the analysis also used Mean Absolute Percentage Error (MAPE). To provide a comprehensive assessment of accuracy, this metric computes the percentage difference between the expected and observed values and averages them.

The percentage difference between the predicted and actual values was lastly measured symmetrically using Symmetric Mean Absolute Percentage Error (sMAPE). A useful metric of accuracy is provided by calculating the average of the percentage difference between the predicted and actual values.

F).Comparison table AR approach

Model no	Structure description	RMSE	MAE	MAPE	sMAPE
T1	1 input 1 output 1 hidden layer 2 hidden nodes linear output = True	3.6271	2.902	0.0742	0.0747
T1	1 input 1 output 2 hidden layer 3 and 4 hidden nodes linear output = True	3.622	2.902	0.0758	0.0747
T1	1 input 1 output 1 hidden layer 4 hidden nodes linear output = False	3.625	2.934	0.0764	0.0755
T2	2 input 1 output 1 hidden layer 2 hidden nodes linear output = True	3.6354	2.95	0.0756	0.0759

T2	2 input 1 output 2 hidden layer 3 and 4 hidden nodes linear output = True	1.723	1.355	0.0343	0.3424
T2	2 input 1 output 1 hidden layer 4 hidden nodes linear output = False	1.566	1.2477	0.0316	0.0315
T3	3 input 1 output 1 hidden layer 2 hidden nodes linear output = True	3.6642	2.941	0.0751	0.0757
T3	3 input 1 output 2 hidden layer 3 and 4 hidden nodes linear output = True	3.845	3.096	0.0806	0.0796
T3	3 input 1 output 1 hidden layer 4 hidden nodes linear output = False	3.902	3.128	0.0814	0.0804
T4	4 input 1 output 1 hidden layer 2 hidden nodes linear output = True	3.5761	2.866	0.0727	0.0735
T4	4 input 1 output 2 hidden layer 3 and 4 hidden nodes linear output = True	4.488	3.595	0.0920	0 .0919
T4	4 input 1 output 1 hidden layer 4 hidden nodes linear output = False	4.280	3.445	0.0886	0.0885
T7	7 input 1 output 1 hidden layer	3.0512	2.301	0.0588	0.0583

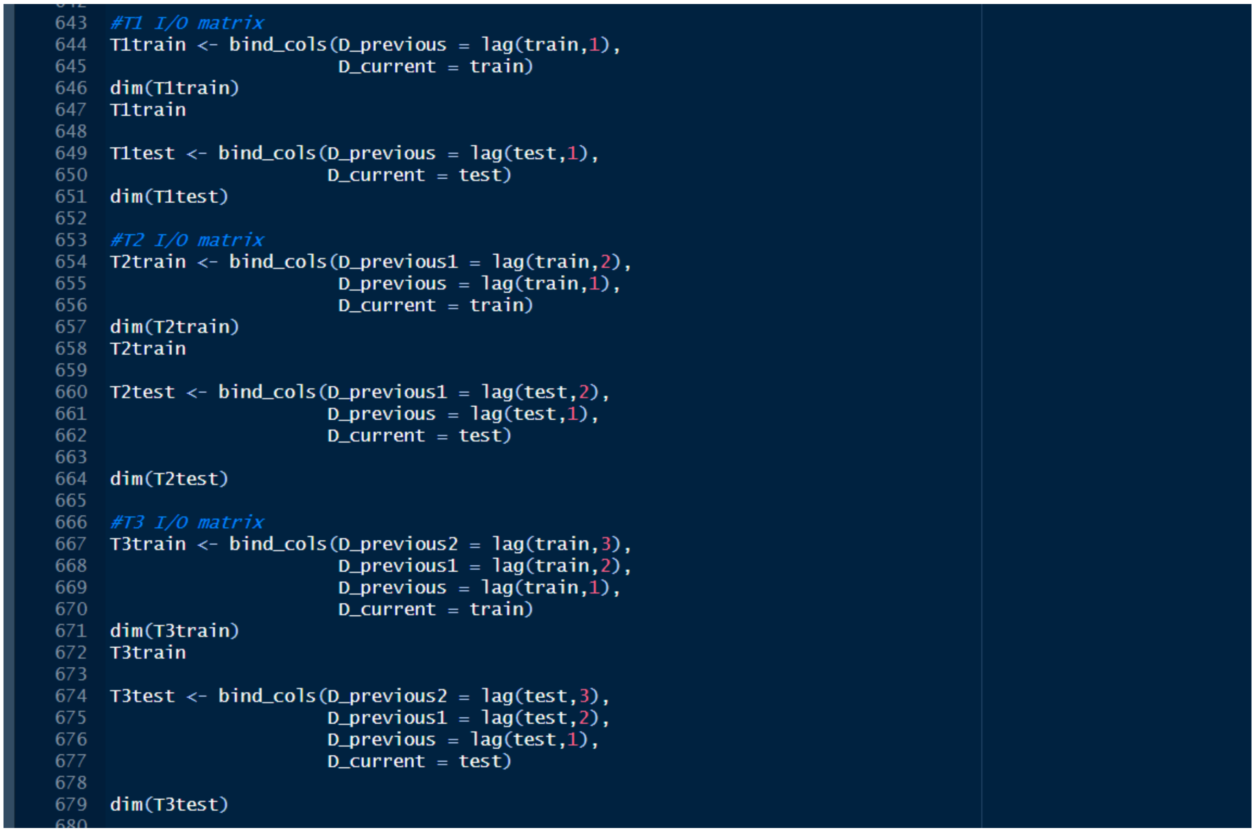
	2 hidden nodes linear output = True				
T7	7 input 1 output 2 hidden layer 3 and 4 hidden nodes linear output = True	3.015	3.639	0.0772	0.0771
T7	7 input 1 output 1 hidden layer 4 hidden nodes linear output = False	3.696	3.116	0.0801	0.0796

G).Efficiency

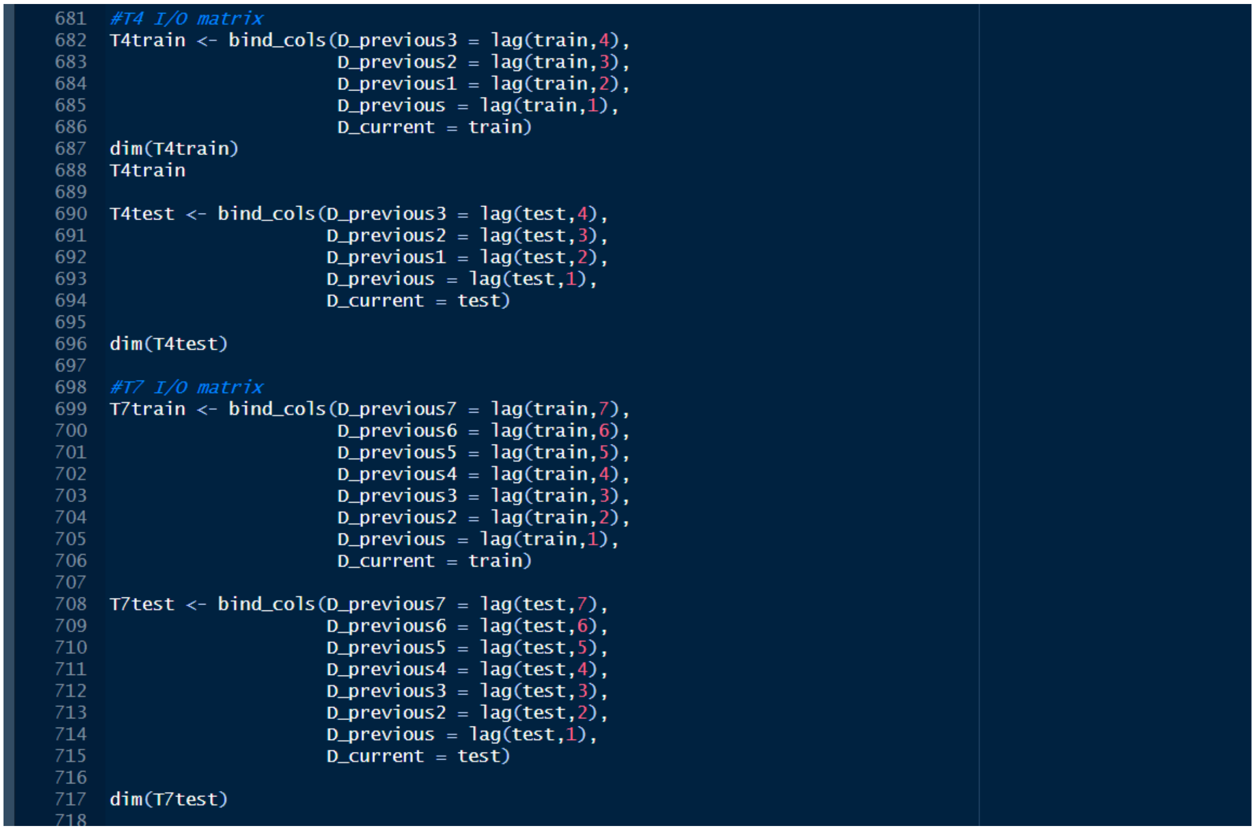
In terms of all four statistical indices, the t-7 model with a single hidden layer performs better than other t-values. Inferring that it is more accurate than other models, the t-7 model with a single hidden node. To determine why t-7 performs better than other t-values and whether this method has any drawbacks, more research is required. To make sure the results are reliable and effectively generalize, it is crucial to assess the model's performance on a larger dataset. Additionally, it might be beneficial to investigate various neural network architectures and evaluate how well they perform in comparison to the t-7 model.

H).NARX approach analysis

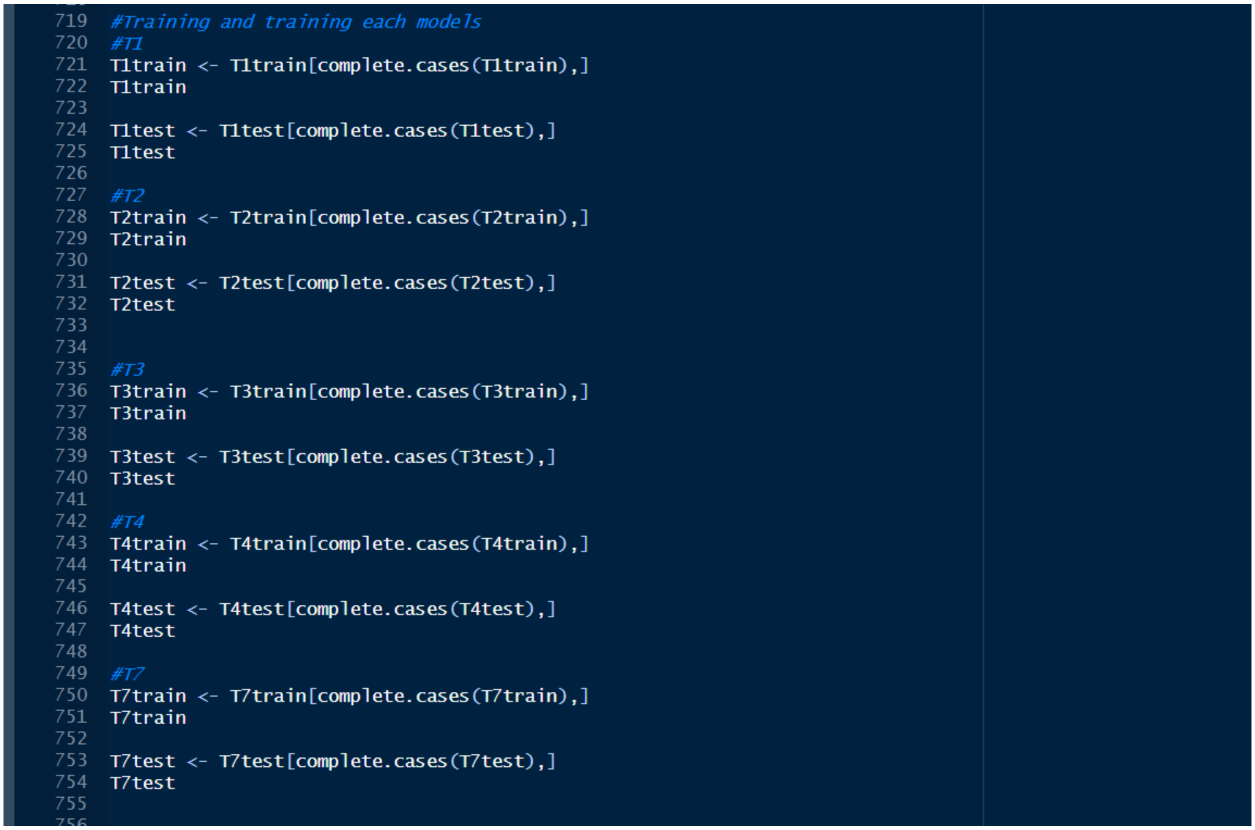
Using the same procedure as in the previous section, the dataset was loaded, but this time, the "18th hour" and "19th hour" columns were also included to train the neural network. The min-max normalization method was used to standardize the dataset. The dataset was then split into a training set and a test set, and each of these sets was given a lagged value to produce a lagged time series dataset.



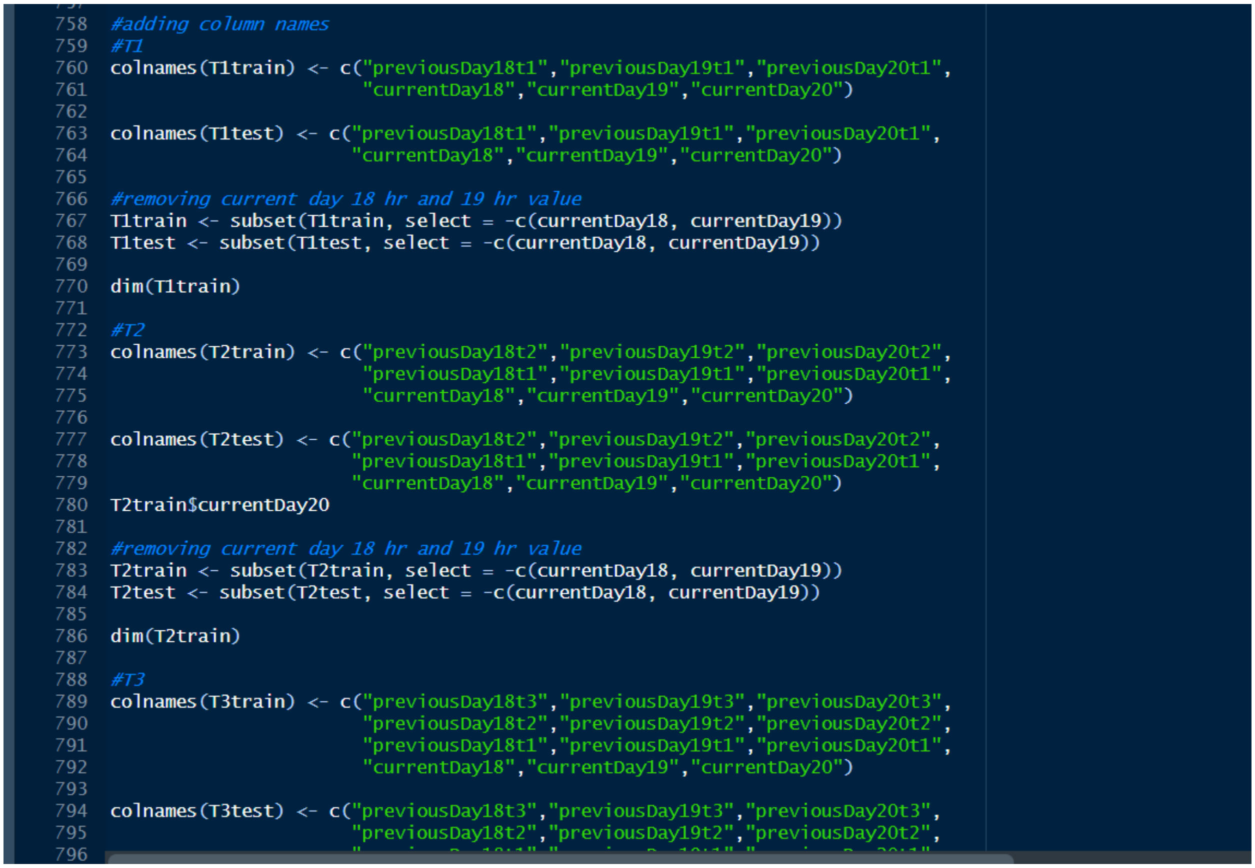
```r
#T1 I/O matrix
T1train <- bind_cols(D_previous = lag(train,1),
                     D_current = train)
dim(T1train)
T1train

T1test <- bind_cols(D_previous = lag(test,1),
                    D_current = test)
dim(T1test)

#T2 I/O matrix
T2train <- bind_cols(D_previous1 = lag(train,2),
                     D_previous = lag(train,1),
                     D_current = train)
dim(T2train)
T2train

T2test <- bind_cols(D_previous1 = lag(test,2),
                    D_previous = lag(test,1),
                    D_current = test)

dim(T2test)

#T3 I/O matrix
T3train <- bind_cols(D_previous2 = lag(train,3),
                     D_previous1 = lag(train,2),
                     D_previous = lag(train,1),
                     D_current = train)
dim(T3train)
T3train

T3test <- bind_cols(D_previous2 = lag(test,3),
                    D_previous1 = lag(test,2),
                    D_previous = lag(test,1),
                    D_current = test)

dim(T3test)
```



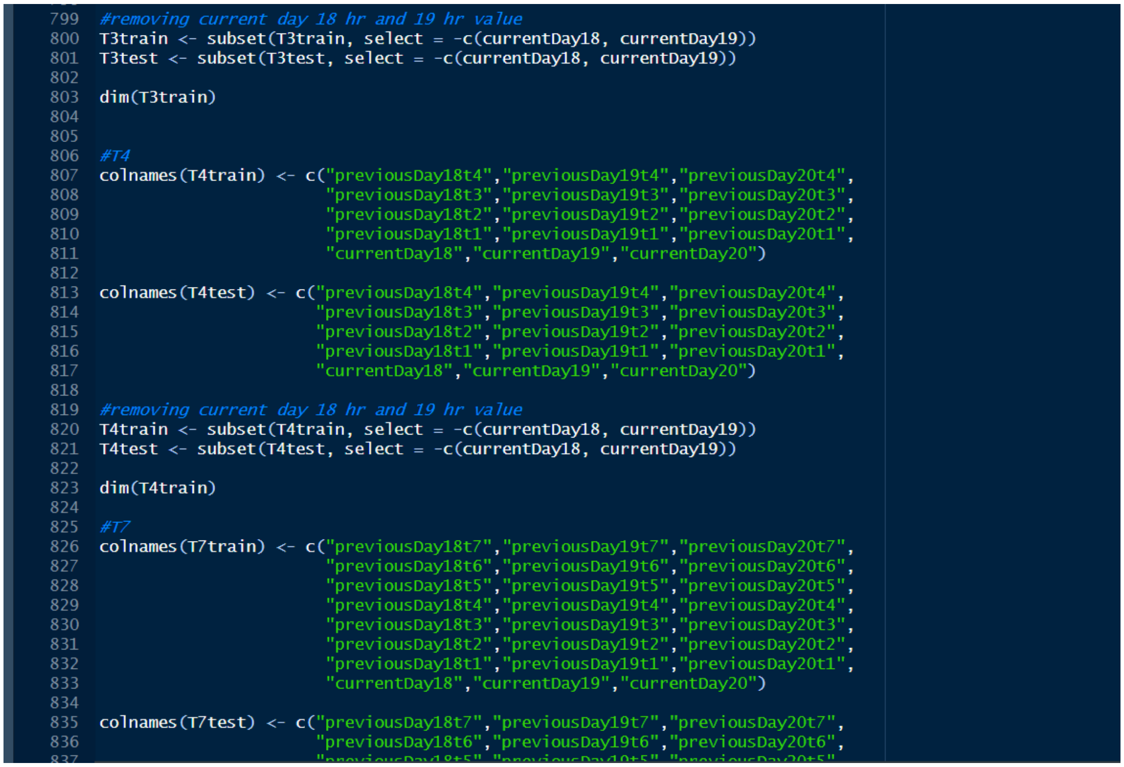
```r
#T4 I/O matrix
T4train <- bind_cols(D_previous3 = lag(train,4),
                     D_previous2 = lag(train,3),
                     D_previous1 = lag(train,2),
                     D_previous = lag(train,1),
                     D_current = train)
dim(T4train)
T4train

T4test <- bind_cols(D_previous3 = lag(test,4),
                    D_previous2 = lag(test,3),
                    D_previous1 = lag(test,2),
                    D_previous = lag(test,1),
                    D_current = test)

dim(T4test)

#T7 I/O matrix
T7train <- bind_cols(D_previous7 = lag(train,7),
                     D_previous6 = lag(train,6),
                     D_previous5 = lag(train,5),
                     D_previous4 = lag(train,4),
                     D_previous3 = lag(train,3),
                     D_previous2 = lag(train,2),
                     D_previous = lag(train,1),
                     D_current = train)

T7test <- bind_cols(D_previous7 = lag(test,7),
                    D_previous6 = lag(test,6),
                    D_previous5 = lag(test,5),
                    D_previous4 = lag(test,4),
                    D_previous3 = lag(test,3),
                    D_previous2 = lag(test,2),
                    D_previous = lag(test,1),
                    D_current = test)

dim(T7test)
```



```r
#Training and training each models
#T1
T1train <- T1train[complete.cases(T1train),]
T1train

T1test <- T1test[complete.cases(T1test),]
T1test

#T2
T2train <- T2train[complete.cases(T2train),]
T2train

T2test <- T2test[complete.cases(T2test),]
T2test


#T3
T3train <- T3train[complete.cases(T3train),]
T3train

T3test <- T3test[complete.cases(T3test),]
T3test

#T4
T4train <- T4train[complete.cases(T4train),]
T4train

T4test <- T4test[complete.cases(T4test),]
T4test

#T7
T7train <- T7train[complete.cases(T7train),]
T7train

T7test <- T7test[complete.cases(T7test),]
T7test

```



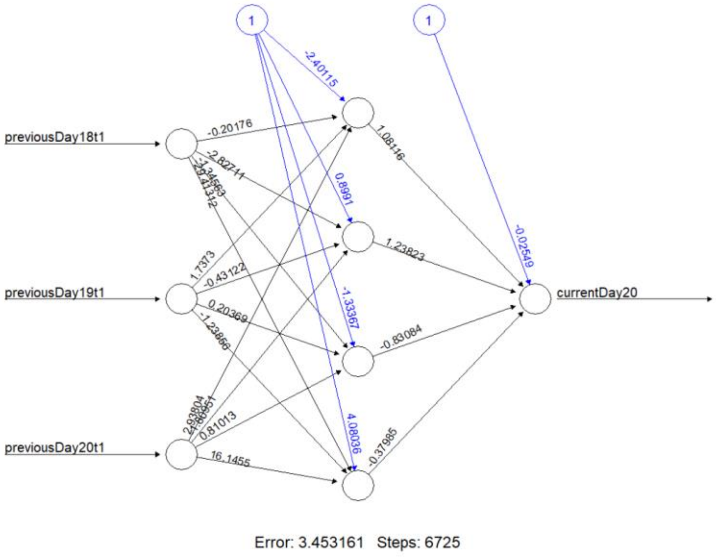
```r
#adding column names
#T1
colnames(T1train) <- c("previousDay18t1","previousDay19t1","previousDay20t1",
                       "currentDay18","currentDay19","currentDay20")

colnames(T1test) <- c("previousDay18t1","previousDay19t1","previousDay20t1",
                      "currentDay18","currentDay19","currentDay20")

#removing current day 18 hr and 19 hr value
T1train <- subset(T1train, select = -c(currentDay18, currentDay19))
T1test <- subset(T1test, select = -c(currentDay18, currentDay19))

dim(T1train)

#T2
colnames(T2train) <- c("previousDay18t2","previousDay19t2","previousDay20t2",
                       "previousDay18t1","previousDay19t1","previousDay20t1",
                       "currentDay18","currentDay19","currentDay20")

colnames(T2test) <- c("previousDay18t2","previousDay19t2","previousDay20t2",
                      "previousDay18t1","previousDay19t1","previousDay20t1",
                      "currentDay18","currentDay19","currentDay20")
T2train$currentDay20

#removing current day 18 hr and 19 hr value
T2train <- subset(T2train, select = -c(currentDay18, currentDay19))
T2test <- subset(T2test, select = -c(currentDay18, currentDay19))

dim(T2train)

#T3
colnames(T3train) <- c("previousDay18t3","previousDay19t3","previousDay20t3",
                       "previousDay18t2","previousDay19t2","previousDay20t2",
                       "previousDay18t1","previousDay19t1","previousDay20t1",
                       "currentDay18","currentDay19","currentDay20")

colnames(T3test) <- c("previousDay18t3","previousDay19t3","previousDay20t3",
                      "previousDay18t2","previousDay19t2","previousDay20t2",
```

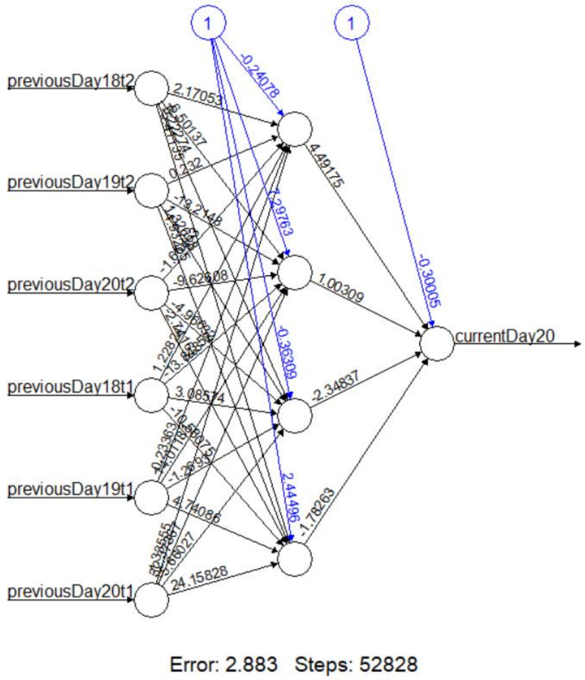


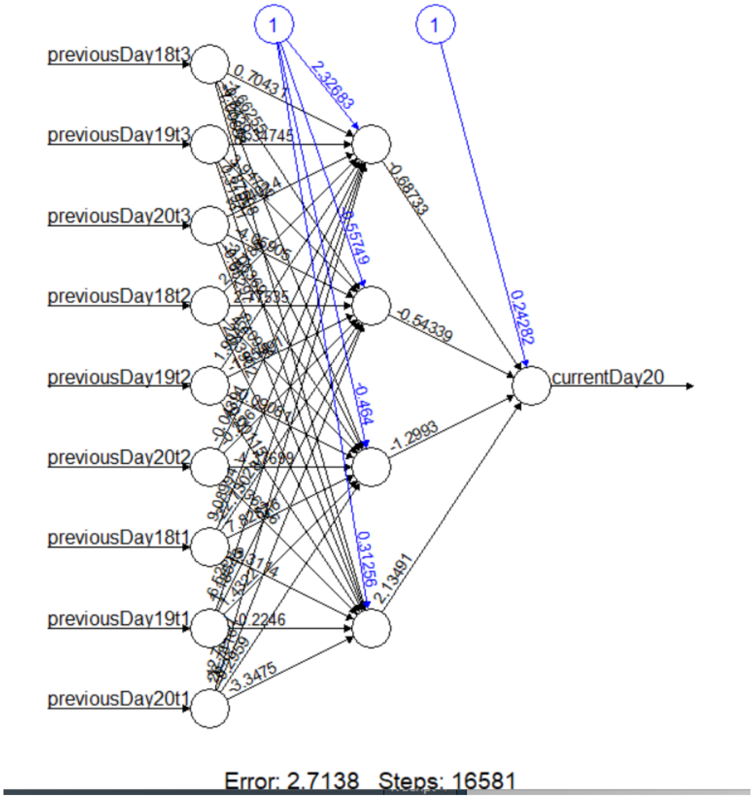
```
#removing current day 18 hr and 19 hr value
T3train <- subset(T3train, select = -c(currentDay18, currentDay19))
T3test <- subset(T3test, select = -c(currentDay18, currentDay19))

dim(T3train)


#T4
colnames(T4train) <- c("previousDay18t4","previousDay19t4","previousDay20t4",
                       "previousDay18t3","previousDay19t3","previousDay20t3",
                       "previousDay18t2","previousDay19t2","previousDay20t2",
                       "previousDay18t1","previousDay19t1","previousDay20t1",
                       "currentDay18","currentDay19","currentDay20")

colnames(T4test) <- c("previousDay18t4","previousDay19t4","previousDay20t4",
                      "previousDay18t3","previousDay19t3","previousDay20t3",
                      "previousDay18t2","previousDay19t2","previousDay20t2",
                      "previousDay18t1","previousDay19t1","previousDay20t1",
                      "currentDay18","currentDay19","currentDay20")

#removing current day 18 hr and 19 hr value
T4train <- subset(T4train, select = -c(currentDay18, currentDay19))
T4test <- subset(T4test, select = -c(currentDay18, currentDay19))

dim(T4train)

#T7
colnames(T7train) <- c("previousDay18t7","previousDay19t7","previousDay20t7",
                       "previousDay18t6","previousDay19t6","previousDay20t6",
                       "previousDay18t5","previousDay19t5","previousDay20t5",
                       "previousDay18t4","previousDay19t4","previousDay20t4",
                       "previousDay18t3","previousDay19t3","previousDay20t3",
                       "previousDay18t2","previousDay19t2","previousDay20t2",
                       "previousDay18t1","previousDay19t1","previousDay20t1",
                       "currentDay18","currentDay19","currentDay20")

colnames(T7test) <- c("previousDay18t7","previousDay19t7","previousDay20t7",
                      "previousDay18t6","previousDay19t6","previousDay20t6",
                      "previousDay18t5","previousDay19t5","previousDay20t5"
```

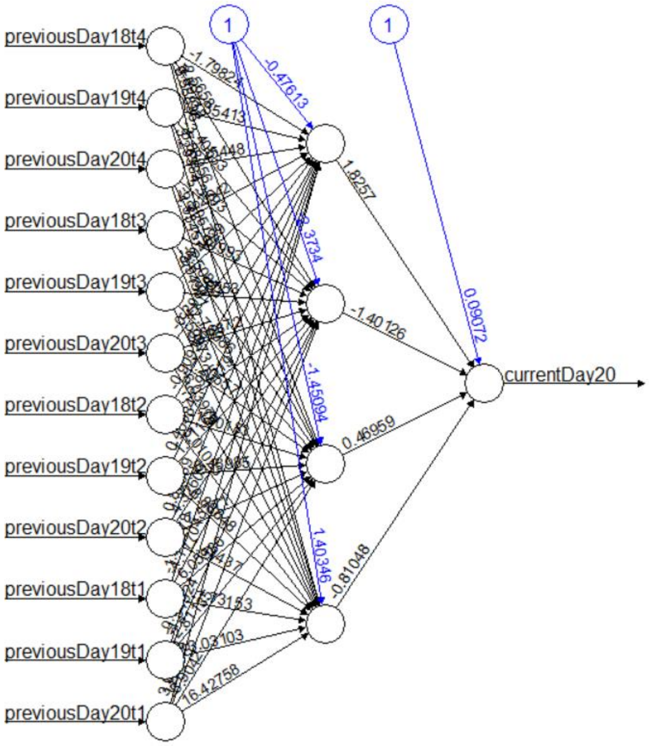
Input-output matrices for times t-1 to t-4 and t-7 were made specifically for this task. These matrices produce the 20th hour at the moment and take as inputs the 20th, 19th, and 18th hours in the past. Similar techniques were employed for the earlier task. For various t-values, five neural network models were trained.

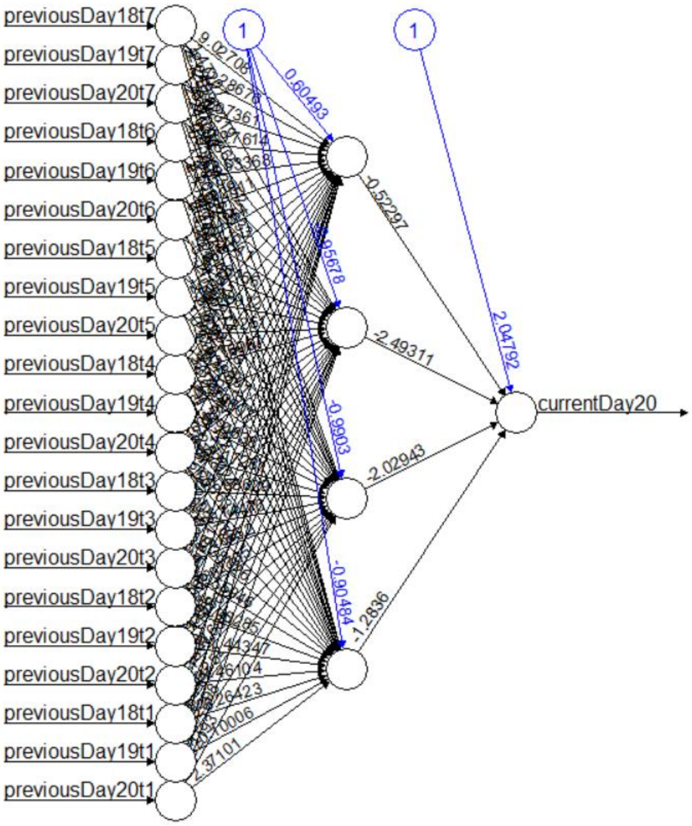
The neural network model is depicted in the above image. As evident The 18th, 19th, and 20th hours of previous days served as the inputs for the output (current day).




1
1
previousDay18t2
previousDay19t2
previousDay20t2
previousDay18t1
previousDay19t1
previousDay20t1
currentDay20
2.17053
0.232
-9.62608
3.08574
4.74086
24.15828
-0.24078
-0.29763
-0.36309
2.44496
-0.30005
4.49175
1.00309
-2.34837
-1.78263
Error: 2.883   Steps: 52828


1
1
previousDay18t3
previousDay19t3
previousDay20t3
previousDay18t2
previousDay19t2
previousDay20t2
previousDay18t1
previousDay19t1
previousDay20t1
currentDay20
0.70437
2.32683
0.55749
-0.464
0.31256
0.24282
-0.68733
-0.54339
-1.2993
2.13491
-0.2246
3.3475
Error: 2.7138   Steps: 16581





All the NARX models from T1 to T4 and Last model T7 is displayed in order above.
Similar to the previous task, the neural network computation used the lagged test dataset inputs to produce predictions. The test set values and predictions were then unnormalized for

comparison. For the NARX models, four statistical indices—RMSE, MAE, MAPE, and sMAPE—were also computed.

Comparison table AR approach:

Model no	Structure Description	RMSE	MAE	MAPE	sMAPE
T1	3 input 1 output 1 hidden layer 4 hidden nodes linear output = TRUE.	3.859	3.128	0.075	0.076
T2	6 input 1 output 1 hidden layer 4 hidden nodes linear output = TRUE.	3.888	3.134	0.076	0.076
T3	9 input 1 output 1 hidden layer 4 hidden nodes linear output = TRUE.	3.681	3.036	0.073	0.073
T4	12 input 1 output 1 hidden layer 4 hidden nodes linear output = TRUE.	3.356	2.717	0.065	0.065
T7	21 input 1 output 1 hidden layer 4 hidden nodes linear output = TRUE.	3.337	2.467	0.059	0.058

Appendix

Clustering code with task 1 and task 2

```
####task 1####
```

```
##subtask 1##

#Loading the required packages
library("readxl")
library("tidyverse")
library("cluster")

#Reading the xlsx file data into a variable
#Package installation to read xlsx file
install.packages("readxl")
Data <- readxl::read_excel("vehicles.xlsx")

#Removing the column -> Samples
Altered_Data <- subset(Data, select = -c(Samples))
head(Altered_Data)
names(Altered_Data)

#Checking the data type of the data in columns
glimpse(Altered_Data)

#The classes available
unique(Altered_Data$Class)

#Removing the class label "y" and keeping only the x labels in the first 18 columns
Labelled_X <- Altered_Data[,-19]

#Dimensions of x label
dim(Labelled_X)

#Checking for any null value in any position in the dataset
which(is.na.data.frame(Labelled_X))
names(Labelled_X)

#Creating boxplots and summary statistics for x label
boxplot(Labelled_X)
summary(Labelled_X)

#Defining functions to detect outliers from multiple columns in a dataframe
detect_outlier <- function(x) {

  #Calculate first quantile
  firstQuantile <- quantile(x, probs=.25)

  #Calculate third quantile
  thirdQuantile <- quantile(x, probs=.75)

  #Calculate interquartile range
  IQR = thirdQuantile - firstQuantile

  #Return true or false
  x > thirdQuantile + (IQR*1.5) | x < firstQuantile - (IQR*1.5)
}

##Defining functions to remove outliers from multiple columns in a dataframe
remove_outlier <- function(dataframe, columns) {

  #Traverse through columns vector using a for loop
  for (col in columns) {
    # Remove observation if it satisfies outlier function
    dataframe <- dataframe[!detect_outlier(dataframe[[col]]),]
  }
  return(dataframe)
}

#Removing outliers from x label data
dataframe <- remove_outlier(Labelled_X, names(Labelled_X))
boxplot(dataframe)

#Removing final outlier
alterData <- remove_outlier(dataframe, names(dataframe))
boxplot(alterData)

dim(alterData)

#Normalizing the x labels using Z-Score standardization as it is useful for finding outliers
alterDataNorm <- as.data.frame(scale(alterData[1:18]))
boxplot(alterDataNorm)

#Installing NbCluster package
install.packages("NbClust")
library("NbClust")

#Setting seed for reproducibility
set.seed(3)

#Using NbClust to find the optimal number of clusters
noOfClusters <- NbClust(alterDataNorm, distance = "euclidean", min.nc = 2, max.nc = 10,
method = "kmeans", index = "all")
```

```
noOfClusters <- NbClust(alterDataNorm, distance = "manhattan", min.nc = 2, max.nc = 10,
method = "kmeans", index = "all")
noOfClusters <- NbClust(alterDataNorm, distance = "maximum", min.nc = 2, max.nc = 10,
method = "kmeans", index = "all")

# Load the factoextra package for clustering analysis visualization
library("factoextra")

# Implement Elbow method to find optimal k using WSS (within-cluster sum of squares)
fviz_nbclust(alterDataNorm, kmeans, method = 'wss')

# Implement Gap Statistic method to find optimal k
fviz_nbclust(alterDataNorm, kmeans, method = 'gap_stat')

# Implement Silhouette method to find optimal k
fviz_nbclust(alterDataNorm, kmeans, method = 'silhouette')

# Apply K-means clustering with k = 3
kmeanfinalOutcome <- kmeans(alterDataNorm, 3)

# Assign cluster label for each data point
kmeanfinalOutcome$cluster

# Compute and show the cluster centers
kmeanfinalOutcome$centers

# Compute the within-cluster sum of squares
wss = kmeanfinalOutcome$tot.withinss
wss

# Compute the between-cluster sum of squares
bss = kmeanfinalOutcome$betweenss
bss

# Compute the ratio of between-cluster sum of squares and total sum of squares
bss/kmeanfinalOutcome$totss

# Compute silhouette for the k-means clustering
sil <- silhouette(kmeanfinalOutcome$cluster, dist(alterDataNorm))
fviz_silhouette(sil)

# Load the fpc package for cluster analysis
install.packages("fpc")
library(fpc)
```

```
# Visualize the clustering result
fviz_cluster(kmeanfinalOutcome, data = alterDataNorm)


##subtask 2##

#Load required libraries
library("readxl")
library("tidyverse")
library("NbClust")
library("gridExtra")
library("ggplot2")
library("factoextra")
library("fpc")

#Reading the xlsx file data into a variable
Data <- readxl::read_excel("vehicles.xlsx")

#Remove the column named "Samples"
Altered_Data <- subset(Data, select = -c(Samples))

#Print the first few rows of the modified dataset
head(Altered_Data)

#Print the column names of the modified dataset
names(Altered_Data)

#Print the data type of each column in the modified dataset
glimpse(Altered_Data)

#Print the unique classes available in the "Class" column of the modified dataset
unique(Altered_Data$Class)

#Remove the "Class" label and keep the first 18 columns as the x labels
Labelled_X <- Altered_Data[,-19]

#Print the dimensions of the modified dataset
dim(Labelled_X)

#Check for any null values in the modified dataset
which(is.na.data.frame(Labelled_X))

#Print the column names of the modified dataset
```

```
names(Labelled_X)

#Calculate the variances of each column in the modified dataset
apply(Labelled_X,2,var)

#Print the first few rows of the modified dataset
head(Labelled_X)

#Create a boxplot to visualize the distribution of each column in the modified dataset
boxplot(Labelled_X)

#Normalize the modified dataset
Labelled_X_scaled <-scale(Labelled_X)

#Print the first few rows of the normalized dataset
head(Labelled_X_scaled)

#Calculate the correlation matrix of the normalized dataset
cor(Labelled_X_scaled)

#Calculate the mean correlation of the normalized dataset
mean(cor(Labelled_X_scaled))

#Calculate the covariance matrix of the normalized dataset
Labelled_X_cov <- cov(Labelled_X_scaled)

#Print the first few rows of the covariance matrix
head(Labelled_X_cov)

#Calculate the eigenvalues and eigenvectors of the covariance matrix
Labelled_X_eigen <- eigen(Labelled_X_cov)

#Print the eigenvalues and eigenvectors
Labelled_X_eigen
str(Labelled_X_eigen)
Labelled_X_eigen$vectors

#Calculate the proportion of variance explained by each principal component
PVE <- Labelled_X_eigen$values / sum(Labelled_X_eigen$values)
round(PVE, 2)

#visualize a scree plot, proportion of variance explained by each principal component
PVEplot <- qplot(c(1:18), PVE) +
  geom_line() +
  xlab("Principal Component") +
  ylab("PVE") +
  ggtitle("Scree Plot") +
  ylim(0, 1)

#visualize a cumulative plot, cumulative proportion of variance explained by each principal
component
cumPVE <- qplot(c(1:18), cumsum(PVE)) +
  geom_line() +
  xlab("Principal Component") +
  ylab(NULL) +
  ggtitle("Cumulative Scree Plot") +
  ylim(0,1)

#Display both the scree plot and the cumulative scree plot side by side
grid.arrange(PVEplot, cumPVE, ncol = 2)

# Convert PVE to percentage and create a barplot to visualize the variance explained by each
PC
varPercent <- PVE*100
barplot(varPercent, xlab='PC', ylab='Percent Variance',
        names.arg=1:length(varPercent), las=1, ylim=c(0, max(varPercent)), col='gray')

# Add a horizontal line representing the percentage of variance explained by a random variable
abline(h=1/ncol(Labelled_X)*100, col='red')

# Extract the first 6 eigenvectors from Labelled_X_eigen and store them in phi
phi <- Labelled_X_eigen$vectors[,1:6]

# Invert the signs of the eigenvectors in phi
phi <- -phi

# Extract the first principal component (PC1) of Labelled_X_scaled
PC1 <- as.matrix(Labelled_X_scaled) %*% phi[,1]

# Extract the second principal component (PC2) of Labelled_X_scaled
PC2 <- as.matrix(Labelled_X_scaled) %*% phi[,2]

# Extract the third principal component (PC3) of Labelled_X_scaled
PC3 <- as.matrix(Labelled_X_scaled) %*% phi[,3]

# Extract the fourth principal component (PC4) of Labelled_X_scaled
PC4 <- as.matrix(Labelled_X_scaled) %*% phi[,4]
```

```
# Extract the fifth principal component (PC5) of Labelled_X_scaled
PC5 <- as.matrix(Labelled_X_scaled) %*% phi[,5]

# Extract the sixth principal component (PC6) of Labelled_X_scaled
PC6 <- as.matrix(Labelled_X_scaled) %*% phi[,6]

# Combine the principal components into a new dataset
dataset <- cbind(PC1,PC2,PC3,PC4,PC5,PC6)

# Create a boxplot of the dataset
boxplot(dataset)

# Display the first few rows of the dataset
head(dataset)

# Set the random seed to 3 for reproducibility
set.seed(3)

# Use the NbClust function to determine the optimal number of clusters using different distance
metrics and clustering methods
noOfClusters = NbClust(dataset,distance="euclidean",
min.nc=2,max.nc=10,method="kmeans",index="all")
noOfClusters = NbClust(dataset,distance="manhattan",
min.nc=2,max.nc=10,method="kmeans",index="all")
noOfClusters = NbClust(dataset,distance="maximum",
min.nc=2,max.nc=10,method="kmeans",index="all")

#visualize the results of the NbClust function using the within-cluster sum of squares method
fviz_nbclust(dataset, kmeans, method = 'wss')

#visualize the results of the NbClust function using the gap statistic method
fviz_nbclust(dataset, kmeans, method = 'gap_stat')

#visualize the results of the NbClust function using the silhouette method
fviz_nbclust(dataset, kmeans, method = 'silhouette')

# Use the kmeans function to perform k-means clustering on the dataset with k=2
kmeanfinalOutcome <- kmeans(dataset,2)

# Display the cluster assignments for each data point
kmeanfinalOutcome$cluster

# Display the center coordinates of each cluster
kmeanfinalOutcome$centers
```

```
# Calculate the within-cluster sum of squares
wss = kmeanfinalOutcome$tot.withinss
wss

#Calculate the between-cluster sum of square
bss = kmeanfinalOutcome$betweenss
bss

#Calculate the ratio between the between-cluster sum of square and the total sum of square
bss/kmeanfinalOutcome$totss

#Calculate the silhouette index
sil <- silhouette(kmeanfinalOutcome$cluster, dist(dataset))
fviz_silhouette(sil)

#Calculate the Calinski-Harabasz index
calinhara(dataset,kmeanfinalOutcome$cluster,cn=max(kmeanfinalOutcome$cluster))

#Visualize the clustering results using fviz_cluster
fviz_cluster(kmeanfinalOutcome, data = dataset)
```

MLP code with task 1 and task 2

```
####task 2####

##subtask 1##
#Load required libraries
library("readxl")
library("tidyverse")
library("neuralnet")
library("Metrics")
library("rsq")

#Read the Excel file containing power consumption data into a variable named 'Data'
Data <- readxl::read_excel("uow_consumption.xlsx")

#Rename the column names of Data
colnames(Data) <- c("date","18:00","19:00","20:00")

#Display the Data dataset
Data
```

```
#Display the first few rows of the Data dataset
head(Data)

#Check the data types of the columns in the Data dataset
glimpse(Data)

#Get the dimensions of the Data dataset
dim(Data)

#Check for any null values in the Data dataset
which(is.na.data.frame(Data))

#Define a function to normalize the data using min-max normalization
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

#Extract the power consumption values for the 20th hour
X_label <- as.data.frame(Data[,4])

#Calculate the minimum and maximum values of the power consumption data
Type_min <- min(X_label)
Type_max <- max(X_label)

#Normalize the power consumption data
X_labelNorm <- as.data.frame(lapply(X_label, normalize))

#Split the data into train and test datasets
train <- X_labelNorm[1:380,]
test <- X_labelNorm[381:470,]

#Define a function to unnormalize the data using the min-max normalization formula
unnormalize <- function(x, min, max) {
  return( (max - min)*x + min )
}

#Define input-output matrices for t1, t2, and t3
#t1 Train and Test I/O matrix (1 input and 1 output)
T1train <- bind_cols(D_previous = lag(train,1),
                     D_current = train)

T1test <- bind_cols(D_previous = lag(test,1),
                    D_current = test)
```

```
#t2 Train and Test I/O matrix (2 input and 1 output)
T2train <- bind_cols(D_previous2 = lag(train,2),
                     D_previous = lag(train,1),
                     D_current = train)

T2test <- bind_cols(D_previous2 = lag(test,2),
                    D_previous = lag(test,1),
                    D_current = test)

#t3 Train and Test I/O matrix (3 input and 1 output)
T3train <- bind_cols(D_previous3 = lag(train,3),
                     D_previous2 = lag(train,2),
                     D_previous = lag(train,1),
                     D_current = train)

T3test <- bind_cols(D_previous3 = lag(test,3),
                    D_previous2 = lag(test,2),
                    D_previous = lag(test,1),
                    D_current = test)

#t4 Train and Test I/O matrix (4 input and 1 output)
T4train <- bind_cols(D_previous4 = lag(train, 4),
                     D_previous3 = lag(train, 3),
                     D_previous2 = lag(train, 2),
                     D_previous = lag(train, 1),
                     D_current = train)

T4test <- bind_cols(D_previous4 = lag(test, 4),
                    D_previous3 = lag(test, 3),
                    D_previous2 = lag(test, 2),
                    D_previous = lag(test, 1),
                    D_current = test)

#t7 Train and Test I/O matrix (7 input and 1 output)
T7train <- bind_cols(D_previous6 = lag(train, 7),
                     D_previous6 = lag(train, 6),
                     D_previous5 = lag(train, 5),
                     D_previous4 = lag(train, 4),
                     D_previous3 = lag(train, 3),
                     D_previous2 = lag(train, 2),
                     D_previous = lag(train, 1),
                     D_current = train)
```

```
T7test <- bind_cols(D_previous6 = lag(test, 7),
             D_previous6 = lag(test, 6),
             D_previous5 = lag(test, 5),
             D_previous4 = lag(test, 4),
             D_previous3 = lag(test, 3),
             D_previous2 = lag(test, 2),
             D_previous = lag(test, 1),
             D_current = test)

#Remove missing values from T1train
T1train <- T1train[complete.cases(T1train),]

#Check the dimensions of T1train
dim(T1train)

#Remove missing values from T1test
T1test <- T1test[complete.cases(T1test),]

#Check the dimensions of T1test
dim(T1test)

#Remove missing values from T2train
T2train <- T2train[complete.cases(T2train),]

#Remove missing values from T2test
T2test <- T2test[complete.cases(T2test),]

#Remove missing values from T3train
T3train <- T3train[complete.cases(T3train),]

#Remove missing values from T3test
T3test <- T3test[complete.cases(T3test),]

#Check the contents of T3test
T3test

#Remove missing values from T4train
T4train <- T4train[complete.cases(T4train),]

#Remove missing values from T4test
T4test <- T4test[complete.cases(T4test),]

#Remove missing values from T7train
T7train <- T7train[complete.cases(T7train),]
```

```
#Remove missing values from T7test
T7test <- T7test[complete.cases(T7test),]

#Check the contents of T7test
T7test

#Rename columns for T1train and T1test
colnames(T1train) <- c("previousDay","currentDay")
colnames(T1test) <- c("previousDay","currentDay")

#Print the currentDay column of T1train
T1train$currentDay

#Rename columns for T2train and T2test
colnames(T2train) <- c("DayBefore2","previousDay","currentDay")
colnames(T2test) <- c("DayBefore2","previousDay","currentDay")

#Print the currentDay column of T2train
T2train$currentDay

#Rename columns for T3train and T3test
colnames(T3train) <- c("DayBefore3","DayBefore2","previousDay","currentDay")
colnames(T3test) <- c("DayBefore3","DayBefore2","previousDay","currentDay")

#Print the currentDay column of T3test
T3test$currentDay

#Rename columns for T4train and T4test
colnames(T4train) <- c("DayBefore4","DayBefore3","DayBefore2","previousDay","currentDay")
colnames(T4test) <- c("DayBefore4","DayBefore3","DayBefore2","previousDay","currentDay")

#Print the currentDay column of T4test
T4test$currentDay

#Rename columns for T7train and T7test
colnames(T7train) <- c("DayBefore7","DayBefore6",
"DayBefore5","DayBefore4","DayBefore3","DayBefore2","previousDay","currentDay")
colnames(T7test) <- c("DayBefore7","DayBefore6",
"DayBefore5","DayBefore4","DayBefore3","DayBefore2","previousDay","currentDay")

#Print the DayBefore6 column of T7test
T7test$DayBefore6
```

```
#Define and train neural network models for I/O Matrix t1
NNmodel1 <- neuralnet(currentDay ~ previousDay, hidden = 2,
                      data = T1train, linear.output = TRUE)

#Define and train neural network models for I/O Matrix t2
NNmodel2 <- neuralnet(currentDay ~ DayBefore2 + previousDay, hidden = 2,
                      data = T2train, linear.output = TRUE)

#Define and train neural network models for I/O Matrix t3
NNmodel3 <- neuralnet(currentDay ~ DayBefore3 + DayBefore2 + previousDay, hidden = 2,
                      data = T3train, linear.output = TRUE)

#Define and train neural network models for I/O Matrix t4
NNmodel4 <- neuralnet(currentDay ~ DayBefore4 + DayBefore3 + DayBefore2 + previousDay,
hidden = 2,
                      data = T4train, linear.output = TRUE)

#Define and train neural network models for I/O Matrix t7
NNmodel7 <- neuralnet(currentDay ~ DayBefore7 + DayBefore6 + DayBefore5 + DayBefore4 +
DayBefore3 + DayBefore2 + previousDay,
                      hidden = 2,
                      data = T7train, linear.output = TRUE)

#Define and train neural network models with changes in the hidden layer size and linear output
#Define and train neural network models for I/O Matrix t1
alterhiddenmodel1 <- neuralnet(currentDay ~ previousDay, hidden = c(3, 4),
                          data = T1train, linear.output = TRUE)

#Define and train neural network models for I/O Matrix t2
alterhiddenmodel2 <- neuralnet(currentDay ~ DayBefore2 + previousDay, hidden = c(3, 4),
                          data = T2train, linear.output = TRUE)

#Define and train neural network models for I/O Matrix t3
alterhiddenmodel3 <- neuralnet(currentDay ~ DayBefore3 + DayBefore2 + previousDay, hidden
= c(3, 4),
                          data = T3train, linear.output = TRUE)

#Define and train neural network models for I/O Matrix t4
alterhiddenmodel4 <- neuralnet(currentDay ~ DayBefore4 + DayBefore3 + DayBefore2 +
previousDay, hidden = c(3, 4),
                          data = T4train, linear.output = TRUE)

#Define and train neural network models for I/O Matrix t7
```

```
alterhiddenmodel7 <- neuralnet(currentDay ~ DayBefore7 + DayBefore6 + DayBefore5 +
DayBefore4 + DayBefore3 + DayBefore2 + previousDay,
                     hidden = c(3, 4),
                     data = T7train, linear.output = TRUE)

#Define and train neural network models with changes in the linear output
#Define and train neural network models for I/O Matrix t1
alterlinearmodel1 <- neuralnet(currentDay ~ previousDay, hidden = 4,
                     data = T1train, linear.output = FALSE)

#Define and train neural network models for I/O Matrix t2
alterlinearmodel2 <- neuralnet(currentDay ~ DayBefore2 + previousDay, hidden = 4,
                     data = T2train, linear.output = FALSE)

#Define and train neural network models for I/O Matrix t3
alterlinearmodel3 <- neuralnet(currentDay ~ DayBefore3 + DayBefore2 + previousDay,hidden =
4,
                     data = T3train, linear.output = FALSE)

#Define and train neural network models for I/O Matrix t4
alterlinearmodel4 <- neuralnet(currentDay ~ DayBefore4 + DayBefore3 + DayBefore2 +
previousDay,hidden = 4,
                     data = T4train, linear.output = FALSE)

#Define and train neural network models for I/O Matrix t7
alterlinearmodel7 <- neuralnet(currentDay ~ DayBefore7 + DayBefore6 + DayBefore5 +
DayBefore4 + DayBefore3 + DayBefore2 + previousDay,
                     hidden = 4,
                     data = T7train, linear.output = FALSE)

#Model for the input/output matrix of time period 1
plot(NNmodel1)

#Model for the input/output matrix of time period 2
plot(NNmodel2)

#Model for the input/output matrix of time period 3
plot(NNmodel3)

#Model for the input/output matrix of time period 4
plot(NNmodel4)

#Model for the input/output matrix of time period 7
plot(NNmodel7)
```

```
#Plot the changed models with modified hidden layers
plot(alterhiddenmodel1)
plot(alterhiddenmodel2)
plot(alterhiddenmodel3)
plot(alterhiddenmodel4)
plot(alterhiddenmodel7)

#Plot the models with linear output set to false
plot(alterlinearmodel1)
plot(alterlinearmodel2)
plot(alterlinearmodel3)
plot(alterlinearmodel4)
plot(alterlinearmodel7)

#Output the result matrix for time period 1
NNmodel1$result.matrix

#Output the result matrix for time period 2
NNmodel2$result.matrix

#Output the result matrix for time period 3
NNmodel3$result.matrix

#Output the result matrix for time period 4
NNmodel4$result.matrix

#Output the result matrix for time period 7
NNmodel7$result.matrix

#t1: Compute and store the prediction results using NNmodel1 on T1test data
dim(T1test)
T1test[1:2]
modifiedT1 <- compute(NNmodel1, T1test[1])
predict1 <- modifiedT1$net.result

#t2: Compute and store the prediction results using NNmodel2 on T2test data
dim(T2test)
T2test[1:3]
modifiedT2 <- compute(NNmodel2, T2test[1:2])
predict2 <- modifiedT2$net.result

#t3: Compute and store the prediction results using NNmodel3 on T3test data
dim(T3test)
```

```
T3test[1:4]
modifiedT3 <- compute(NNmodel3, T3test[1:3])
predict3 <- modifiedT3$net.result

#t4: Compute and store the prediction results using NNmodel4 on T4test data
dim(T4test)
T4test[1:5]
modifiedT4 <- compute(NNmodel4, T4test[1:4])
predict4 <- modifiedT4$net.result

#t7: Compute and store the prediction results using NNmodel7 on T7test data
dim(T7test)
T7test[1:8]
modifiedT7 <- compute(NNmodel7, T7test[1:7])
predict7 <- modifiedT7$net.result

#Define a function to evaluate the model predictions
evaluation <- function(pred, test, Type_min, Type_max){

  #Un-normalize the predictions
  prediction <- unnormalize(pred, Type_min, Type_max)
  prediction <- round(prediction, 1)

  #Un-normalize the test data for comparison
  expected <- unnormalize(test, Type_min, Type_max)
  expected <- round(expected, 1)

  #Combine the predicted and expected results in a table
  result <- cbind(prediction, expected[-1])
  colnames(result) <- c("Prediction", "Expected")

  #Evaluation
  rmse_result <- rmse(result$Expected, result$Prediction)
  mae_result <- mae(result$Expected, result$Prediction)
  mape_result <- mape(result$Expected, result$Prediction)
  smape_result <- smape(result$Expected, result$Prediction)


  cat("MAE",mae_result,",")
  cat("RMSE", rmse_result,",")
  cat("MAPE", mape_result,",")
  cat("SMAPE",smape_result,",")

}
```

```
#Evaluating results of the changed model
# Get the dimensions of the test set
dim(T1test)
# Print out the first two rows of the test set
T1test[1:2]
# Compute the model result using the changed hidden model and the first row of the test set
modifiedT1 <- compute(alterhiddenmodel1, T1test[1])
# Get the predicted normalized result from the computed model result
predict1 <- modifiedT1$net.result
# Evaluate the predicted result against the test set using the evaluation function
evaluation(predict1, T1test, Type_min, Type_max)

dim(T2test)
T2test[1:3]
modifiedT2 <- compute(alterhiddenmodel2, T2test[1:2])
predict2 <- modifiedT2$net.result
evaluation(predict2, T2test, Type_min, Type_max)

dim(T3test)
T3test[1:4]
modifiedT3 <- compute(alterhiddenmodel3, T3test[1:3])
predict3 <- modifiedT3$net.result
evaluation(predict3, T3test, Type_min, Type_max)

dim(T4test)
T4test[1:5]
modifiedT4 <- compute(alterhiddenmodel4, T4test[1:4])
predict4 <- modifiedT4$net.result
evaluation(predict4, T4test, Type_min, Type_max)

dim(T7test)
T7test[1:8]
modifiedT7 <- compute(alterhiddenmodel7, T7test[1:7])
predict7 <- modifiedT7$net.result
evaluation(predict7, T7test, Type_min, Type_max)

#Evaluating results of the changed linear output model
dim(T1test)
T1test[1:2]
modifiedT1 <- compute(alterlinearmodel1, T1test[1])
predict1 <- modifiedT1$net.result
evaluation(predict1, T1test, Type_min, Type_max)
```

```
dim(T2test)
T2test[1:3]
modifiedT2 <- compute(alterlinearmodel2, T2test[1:2])
predict2 <- modifiedT2$net.result
evaluation(predict2, T2test, Type_min, Type_max)

dim(T3test)
T3test[1:4]
modifiedT3 <- compute(alterlinearmodel3, T3test[1:3])
predict3 <- modifiedT3$net.result
evaluation(predict3, T3test, Type_min, Type_max)

dim(T4test)
T4test[1:5]
modifiedT4 <- compute(alterlinearmodel4, T4test[1:4])
predict4 <- modifiedT4$net.result
evaluation(predict4, T4test, Type_min, Type_max)

dim(T7test)
T7test[1:8]
modifiedT7 <- compute(alterlinearmodel7, T7test[1:7])
predict7 <- modifiedT7$net.result
evaluation(predict7, T7test, Type_min, Type_max)


#Prediction and comparison of the expected and predicted values for t1
# Normalized predicted values for t1
predict1
# Un-normalize predicted values for t1
unpredt1 <- unnormalize(predict1, min(X_label), max(X_label))
# Round the predicted values for t1 to one decimal place
unpredt1 <- round(unpredt1, 1)
# Display the un-normalized and rounded predicted values for t1
unpredt1

# Un-normalize expected values for t1
expectt1 <- unnormalize(T1test, min(X_label), max(X_label))
# Round the expected values for t1 to one decimal place
expectt1 <- round(expectt1, 1)
# Display the un-normalized and rounded expected values for t1
expectt1

# Combine the expected and predicted values for t1
endresult1 <- cbind(expectt1[2], unpredt1)
```

```
# Add column names to the final result for t1
colnames(endresult1) <- c("ExpectingResult", "pred")
# Display the final result for t1
endresult1

#Prediction and comparison of the expected and predicted values for t2
predict2
unpredt2 <- unnormalize(predict2, min(X_label), max(X_label))
unpredt2 <- round(unpredt2, 1)
unpredt2

expectt2 <- unnormalize(T2test, min(X_label), max(X_label))
expectt2 <- round(expectt2, 1)
expectt2

endresult2 <- cbind(expectt2[3], unpredt2)
colnames(endresult2) <- c("ExpectingResult", "pred")
endresult2

#Prediction and comparison of the expected and predicted values for t3
predict3
unpredt3 <- unnormalize(predict3, min(X_label), max(X_label))
unpredt3 <- round(unpredt3, 1)
unpredt3

expectt3 <- unnormalize(T3test, min(X_label), max(X_label))
expectt3 <- round(expectt3, 1)
expectt3

endresult3 <- cbind(expectt3[4], unpredt3)
colnames(endresult3) <- c("ExpectingResult", "pred")
endresult3

#Prediction and comparison of the expected and predicted values for t4
predict4
unpredt4 <- unnormalize(predict4,min(X_label),max(X_label))
unpredt4 <- round(unpredt4,1)
unpredt4

expectt4 <- unnormalize(T4test,min(X_label),max(X_label))
expectt4 <- round(expectt4,1)
expectt4

endresult4 <- cbind(expectt4[5],unpredt4)
```

```
colnames(endresult4) <- c("ExpectingResult","pred")
endresult4

#Prediction and comparison of the expected and predicted values for t7
predict7
unpredt7 <- unnormalize(predict7,min(X_label),max(X_label))
unpredt7 <- round(unpredt7,1)
unpredt7

expectt7 <- unnormalize(T7test,min(X_label),max(X_label))
expectt7 <- round(expectt7,1)
expectt7

endresult7 <- cbind(expectt7[8],unpredt7)
colnames(endresult7) <- c("ExpectingResult","pred")
endresult7
summary(NNmodel7$result.matrix)

#summarizing data frames
library(dplyr)
sqrt(mean((endresult4$ExpectingResult - endresult4$pred)^2))

#Root Mean Square Error (RMSE) calculations
#t1
rmse(endresult1$ExpectingResult, endresult1$pred)

#t2
rmse(endresult2$ExpectingResult, endresult2$pred)

#t3
rmse(endresult3$ExpectingResult, endresult3$pred)

#t4
rmse(endresult4$ExpectingResult, endresult4$pred)

#t7
rmse(endresult7$ExpectingResult, endresult7$pred)

#Mean Absolute Error (MAE) calculations
#t1
mae(endresult1$ExpectingResult, endresult1$pred)

#t2
mae(endresult2$ExpectingResult, endresult2$pred)
```

```
#t3
mae(endresult3$ExpectingResult, endresult3$pred)

#t4
mae(endresult4$ExpectingResult, endresult4$pred)

#t7
mae(endresult7$ExpectingResult, endresult7$pred)

#Mean Absolute Percentage Error (MAPE) calculations
#t1
mape(endresult1$ExpectingResult, endresult1$pred)

#t2
mape(endresult2$ExpectingResult, endresult2$pred)

#t3
mape(endresult3$ExpectingResult, endresult3$pred)

#t4
mape(endresult4$ExpectingResult, endresult4$pred)

#t7
mape(endresult7$ExpectingResult, endresult7$pred)

#Symmetric Mean Absolute Percentage Error (SMAPE) calculations
#t1
smape(endresult1$ExpectingResult, endresult1$pred)

#t2
smape(endresult2$ExpectingResult, endresult2$pred)

#t3
smape(endresult3$ExpectingResult, endresult3$pred)

#t4
smape(endresult4$ExpectingResult, endresult4$pred)

#t7
smape(endresult7$ExpectingResult, endresult7$pred)

library("caret")
#R-squared (R2) calculations
```

```
#t1
R2(endresult1$ExpectingResult, endresult1$pred)

#t2
R2(endresult2$ExpectingResult, endresult2$pred)

#t3
R2(endresult3$ExpectingResult, endresult3$pred)

#t4
R2(endresult4$ExpectingResult, endresult4$pred)

#t7
R2(endresult7$ExpectingResult, endresult7$pred)

#create a scatter plot to visualize predicted vs. actual values
ggplot(endresult7, aes(x = ExpectingResult, y = pred)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
  xlab("Actual Values") +
  ylab("Predicted Values") +
  ggtitle("Predicted vs. Actual Values")



##subtask 2##

#Loading necessary packages
library("readxl") # For reading Excel files
library("tidyverse") # For data manipulation and visualization
library(neuralnet) # For building neural network models
library(dplyr)

#Storing the vehicle data from an Excel file
Data <- readxl::read_excel("uow_consumption.xlsx")
colnames(Data) <- c("date","18:00","19:00","20:00")

head(Data)

#Function for normalizing data using min-max normalization
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
```

```
#Function for un-normalizing data using min-max normalization
unnormalize <- function(x, min, max) {
  return( (max - min)*x + min )
}

#Selecting the relevant features for input data
X_Label <- as.data.frame(Data[2:4])
X_Label

#Normalizing the input data
X_LabelNorm <- as.data.frame(lapply(X_Label, normalize))

#Splitting data into training and testing sets
train <- X_LabelNorm[1:380,]
train

test <- X_LabelNorm[381:470,]
test

#T1 I/O matrix
T1train <- bind_cols(D_previous = lag(train,1),
                     D_current = train)
dim(T1train)
T1train

T1test <- bind_cols(D_previous = lag(test,1),
                    D_current = test)
dim(T1test)

#T2 I/O matrix
T2train <- bind_cols(D_previous1 = lag(train,2),
                     D_previous = lag(train,1),
                     D_current = train)
dim(T2train)
T2train

T2test <- bind_cols(D_previous1 = lag(test,2),
                    D_previous = lag(test,1),
                    D_current = test)

dim(T2test)

#T3 I/O matrix
T3train <- bind_cols(D_previous2 = lag(train,3),
                     D_previous1 = lag(train,2),
                     D_previous = lag(train,1),
                     D_current = train)
dim(T3train)
T3train

T3test <- bind_cols(D_previous2 = lag(test,3),
                    D_previous1 = lag(test,2),
                    D_previous = lag(test,1),
                    D_current = test)

dim(T3test)

#T4 I/O matrix
T4train <- bind_cols(D_previous3 = lag(train,4),
                     D_previous2 = lag(train,3),
                     D_previous1 = lag(train,2),
                     D_previous = lag(train,1),
                     D_current = train)
dim(T4train)
T4train

T4test <- bind_cols(D_previous3 = lag(test,4),
                    D_previous2 = lag(test,3),
                    D_previous1 = lag(test,2),
                    D_previous = lag(test,1),
                    D_current = test)

dim(T4test)

#T7 I/O matrix
T7train <- bind_cols(D_previous7 = lag(train,7),
                     D_previous6 = lag(train,6),
                     D_previous5 = lag(train,5),
                     D_previous4 = lag(train,4),
                     D_previous3 = lag(train,3),
                     D_previous2 = lag(train,2),
                     D_previous = lag(train,1),
                     D_current = train)

T7test <- bind_cols(D_previous7 = lag(test,7),
                    D_previous6 = lag(test,6),
                    D_previous5 = lag(test,5),
                    D_previous4 = lag(test,4),
                    D_previous3 = lag(test,3),
                    D_previous2 = lag(test,2),
                    D_previous = lag(test,1),
                    D_current = test)

dim(T7test)

#Training and training each models
#T1
T1train <- T1train[complete.cases(T1train),]
T1train

T1test <- T1test[complete.cases(T1test),]
T1test

#T2
T2train <- T2train[complete.cases(T2train),]
T2train

T2test <- T2test[complete.cases(T2test),]
T2test


#T3
T3train <- T3train[complete.cases(T3train),]
T3train

T3test <- T3test[complete.cases(T3test),]
T3test

#T4
T4train <- T4train[complete.cases(T4train),]
T4train

T4test <- T4test[complete.cases(T4test),]
T4test

#T7
T7train <- T7train[complete.cases(T7train),]
T7train

T7test <- T7test[complete.cases(T7test),]
T7test
```

```
#adding column names
#T1
colnames(T1train) <- c("previousDay18t1","previousDay19t1","previousDay20t1",
                       "currentDay18","currentDay19","currentDay20")

colnames(T1test) <- c("previousDay18t1","previousDay19t1","previousDay20t1",
                      "currentDay18","currentDay19","currentDay20")

#removing current day 18 hr and 19 hr value
T1train <- subset(T1train, select = -c(currentDay18, currentDay19))
T1test <- subset(T1test, select = -c(currentDay18, currentDay19))

dim(T1train)

#T2
colnames(T2train) <- c("previousDay18t2","previousDay19t2","previousDay20t2",
                       "previousDay18t1","previousDay19t1","previousDay20t1",
                       "currentDay18","currentDay19","currentDay20")

colnames(T2test) <- c("previousDay18t2","previousDay19t2","previousDay20t2",
                      "previousDay18t1","previousDay19t1","previousDay20t1",
                      "currentDay18","currentDay19","currentDay20")
T2train$currentDay20

#removing current day 18 hr and 19 hr value
T2train <- subset(T2train, select = -c(currentDay18, currentDay19))
T2test <- subset(T2test, select = -c(currentDay18, currentDay19))

dim(T2train)

#T3
colnames(T3train) <- c("previousDay18t3","previousDay19t3","previousDay20t3",
                       "previousDay18t2","previousDay19t2","previousDay20t2",
                       "previousDay18t1","previousDay19t1","previousDay20t1",
                       "currentDay18","currentDay19","currentDay20")

colnames(T3test) <- c("previousDay18t3","previousDay19t3","previousDay20t3",
                      "previousDay18t2","previousDay19t2","previousDay20t2",
                      "previousDay18t1","previousDay19t1","previousDay20t1",
                      "currentDay18","currentDay19","currentDay20")

#removing current day 18 hr and 19 hr value
```

```
T3train <- subset(T3train, select = -c(currentDay18, currentDay19))
T3test <- subset(T3test, select = -c(currentDay18, currentDay19))

dim(T3train)


#T4
colnames(T4train) <- c("previousDay18t4","previousDay19t4","previousDay20t4",
                   "previousDay18t3","previousDay19t3","previousDay20t3",
                   "previousDay18t2","previousDay19t2","previousDay20t2",
                   "previousDay18t1","previousDay19t1","previousDay20t1",
                   "currentDay18","currentDay19","currentDay20")

colnames(T4test) <- c("previousDay18t4","previousDay19t4","previousDay20t4",
                  "previousDay18t3","previousDay19t3","previousDay20t3",
                  "previousDay18t2","previousDay19t2","previousDay20t2",
                  "previousDay18t1","previousDay19t1","previousDay20t1",
                  "currentDay18","currentDay19","currentDay20")

#removing current day 18 hr and 19 hr value
T4train <- subset(T4train, select = -c(currentDay18, currentDay19))
T4test <- subset(T4test, select = -c(currentDay18, currentDay19))

dim(T4train)

#T7
colnames(T7train) <- c("previousDay18t7","previousDay19t7","previousDay20t7",
                   "previousDay18t6","previousDay19t6","previousDay20t6",
                   "previousDay18t5","previousDay19t5","previousDay20t5",
                   "previousDay18t4","previousDay19t4","previousDay20t4",
                   "previousDay18t3","previousDay19t3","previousDay20t3",
                   "previousDay18t2","previousDay19t2","previousDay20t2",
                   "previousDay18t1","previousDay19t1","previousDay20t1",
                   "currentDay18","currentDay19","currentDay20")

colnames(T7test) <- c("previousDay18t7","previousDay19t7","previousDay20t7",
                  "previousDay18t6","previousDay19t6","previousDay20t6",
                  "previousDay18t5","previousDay19t5","previousDay20t5",
                  "previousDay18t4","previousDay19t4","previousDay20t4",
                  "previousDay18t3","previousDay19t3","previousDay20t3",
                  "previousDay18t2","previousDay19t2","previousDay20t2",
                  "previousDay18t1","previousDay19t1","previousDay20t1",
                  "currentDay18","currentDay19","currentDay20")
```

```
#removing current day 18 hr and 19 hr value
T7train <- subset(T7train, select = -c(currentDay18, currentDay19))
T7test <- subset(T7test, select = -c(currentDay18, currentDay19))

dim(T7train)

install.packages("neuralnet")
library(neuralnet)
#plotting NN
#NN for T1
NNmodel1 <- neuralnet(currentDay20 ~ previousDay18t1 + previousDay19t1+
previousDay20t1,
                    hidden = 4, data = T1train, linear.output = TRUE)

plot(NNmodel1)

#NN for T2
NNmodel2 <- neuralnet(currentDay20 ~ previousDay18t2+ previousDay19t2 + previousDay20t2
                    + previousDay18t1 + previousDay19t1+ previousDay20t1,
                    hidden = 4, data = T2train, linear.output = TRUE)

plot(NNmodel2)

#NN for T3
NNmodel3 <- neuralnet(currentDay20 ~ previousDay18t3 + previousDay19t3 +
previousDay20t3 +
                     previousDay18t2+ previousDay19t2 + previousDay20t2 +
                     previousDay18t1 + previousDay19t1+ previousDay20t1,
                    hidden = 4, data = T3train, linear.output = TRUE)

plot(NNmodel3)

#NN for T4
NNmodel4 <- neuralnet(currentDay20 ~ previousDay18t4 + previousDay19t4 +
previousDay20t4 +
                     previousDay18t3 + previousDay19t3 + previousDay20t3 +
                     previousDay18t2+ previousDay19t2 + previousDay20t2 +
                     previousDay18t1 + previousDay19t1+ previousDay20t1,
                    hidden = 4, data = T4train, linear.output = TRUE)

plot(NNmodel4)

#NN for T7
```

```
NNmodel7 <- neuralnet(currentDay20 ~ previousDay18t7 + previousDay19t7 +
previousDay20t7 +
                  previousDay18t6 + previousDay19t6 + previousDay20t6 +
                  previousDay18t5 + previousDay19t5 + previousDay20t5 +
                  previousDay18t4 + previousDay19t4 + previousDay20t4 +
                  previousDay18t3 + previousDay19t3 + previousDay20t3 +
                  previousDay18t2 + previousDay19t2 + previousDay20t2 +
                  previousDay18t1 + previousDay19t1 + previousDay20t1,hidden = 4,
                 data = T7train, linear.output = TRUE)

plot(NNmodel7)

#create predicted results
#T1
modifiedT1 <- compute(NNmodel1,T1test[1:3])
predict1 <- modifiedT1$net.result

predict1

#T2
modifiedT2 <- compute(NNmodel2,T2test[1:6])
predict2 <- modifiedT2$net.result

predict2

#T3
modifiedT3 <- compute(NNmodel3,T3test[1:9])
predict3 <- modifiedT3$net.result

predict3

#T4
modifiedT4 <- compute(NNmodel4,T4test[1:12])
predict4 <- modifiedT4$net.result

predict4

#T7
modifiedT7 <- compute(NNmodel7,T7test[1:21])
predict7 <- modifiedT7$net.result

predict7
```

```
#Un-normalize
#T1
unpredt1 <- unnormalize(predict1,min(X_Label),max(X_Label))
unpredt1 <- round(unpredt1,1)
unpredt1

expectt1 <- unnormalize(T1test,min(X_Label),max(X_Label))
expectt1 <- round(expectt1,1)
expectt1

#T2
unpredt2 <- unnormalize(predict2,min(X_Label),max(X_Label))
unpredt2 <- round(unpredt2,1)
unpredt2

expectt2 <- unnormalize(T2test,min(X_Label),max(X_Label))
expectt2 <- round(expectt2,1)
expectt2

#T3
unpredt3 <- unnormalize(predict3,min(X_Label),max(X_Label))
unpredt3 <- round(unpredt3,1)
unpredt3

expectt3 <- unnormalize(T3test,min(X_Label),max(X_Label))
expectt3 <- round(expectt3,1)
expectt3

#T4
unpredt4 <- unnormalize(predict4,min(X_Label),max(X_Label))
unpredt4 <- round(unpredt4,1)
unpredt4

expectt4 <- unnormalize(T4test,min(X_Label),max(X_Label))
expectt4 <- round(expectt4,1)
expectt4

#T7
unpredt7 <- unnormalize(predict7,min(X_Label),max(X_Label))
unpredt7 <- round(unpredt7,1)
unpredt7

expectt7 <- unnormalize(T7test,min(X_Label),max(X_Label))
expectt7 <- round(expectt7,1)
```

```
expectt7


#Final result
#T1
endresult1 <- cbind(expectt1[4],unpredt1)
colnames(endresult1) <- c("ExpectingResult","pred")
endresult1

#T2
endresult2 <- cbind(expectt2[7],unpredt2)
colnames(endresult2) <- c("ExpectingResult","pred")
endresult2

#T3
endresult3 <- cbind(expectt3[10],unpredt3)
colnames(endresult3) <- c("ExpectingResult","pred")
endresult3

#T4
endresult4 <- cbind(expectt4[13],unpredt4)
colnames(endresult4) <- c("ExpectingResult","pred")
endresult4

#T7
endresult7 <- cbind(expectt7[22],unpredt7)
colnames(endresult7) <- c("ExpectingResult","pred")
endresult7



library(Metrics)

#RMSE
#T1
rmse(endresult1$ExpectingResult, endresult1$pred)
#T2
rmse(endresult2$ExpectingResult, endresult2$pred)
#T3
rmse(endresult3$ExpectingResult, endresult3$pred)
#T4
rmse(endresult4$ExpectingResult, endresult4$pred)
#T7
rmse(endresult7$ExpectingResult, endresult7$pred)
```

```
#MAE
#T1
mae(endresult1$ExpectingResult, endresult1$pred)
#T2
mae(endresult2$ExpectingResult, endresult2$pred)
#T3
mae(endresult3$ExpectingResult, endresult3$pred)
#T4
mae(endresult4$ExpectingResult, endresult4$pred)
#T7
mae(endresult7$ExpectingResult, endresult7$pred)

#MAPE
#T1
mape(endresult1$ExpectingResult, endresult1$pred)
#T2
mape(endresult2$ExpectingResult, endresult2$pred)
#T3
mape(endresult3$ExpectingResult, endresult3$pred)
#T4
mape(endresult4$ExpectingResult, endresult4$pred)
#T7
mape(endresult7$ExpectingResult, endresult7$pred)

#sMAPE
#T1
smape(endresult1$ExpectingResult, endresult1$pred)
#T2
smape(endresult2$ExpectingResult, endresult2$pred)
#T3
smape(endresult3$ExpectingResult, endresult3$pred)
#T4
smape(endresult4$ExpectingResult, endresult4$pred)
#T7
smape(endresult7$ExpectingResult, endresult7$pred)
```