UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

Informatics Institute of Technology

Department of Computing (B.Eng.) in Software Engineering

Coursework

Module Name -: Machine Learning and Data Mining

Module Code -: 5DATA001C.2

Module Leader -: Mr. Achala Aponso

Student Full Name -: Warnakula Weerasuriya Mayeul Rukshan Fernando

UOW Name -: w1809821

UOW Number -: 18098217

IIT Number -: 2019784

Group -: M

Date of Submission -: 17/05/2021

Contents

List of Figures

List of Tables

Introduction

Part of Clustering

Among this project, we'll look at a collection of observations on numerous white wine types, including chemical attributes and taster rankings. As social drinking becomes more popular, the wine business has experienced a recent growth spurt. The cost of wine is determined by a very nebulous idea of wine appreciation among wine tasters. To some extent, wine pricing is influenced by such a variable component. Physicochemical tests, which are laboratory-based and include characteristics like acidity, pH level, presence of sugar, and other chemical features, are another important aspect in wine certification and quality evaluation. Again for wine market, it'd be interesting if human testing quality could be linked to chemical attributes of wine, allowing for more regulated certification, quality evaluation, and assurance. There is one dataset available (w1809821_p1_Whitewine_v2.xls) that contains 4710 kinds of white wine. Every wine comes from a specific region in Portugal. Data is gathered on 12 distinct attributes of the wines, one of which is Quality (the last column), which is based on sensory data, while the others are chemical properties such as density, acidity, and alcohol concentration. Wines have continuous chemical characteristics. Quality is a categorical variable that may be ranked from 1 (worst) to 10 (best). So every wine variety is sampled by three independent tasters, with the final rank determined by the tasters' median score.

Part of Partitioning Clustering

This white wine dataset challenge requires you to do a k-means clustering analysis. There are 11 input variables and one output variable in the dataset of 4710 wine samples (i.e. quality). There are four different levels of quality. Do not attempt to merge neighboring classes with minimal samples in this assignment. The study in this specific clustering section will begin with all beginning characteristics, as the main goal is to compare different clustering outcomes under the initial conditions. However, in the following step, principle component analysis (PCA) will be used to minimize the input dimensionality, and the newly created dataset will be clustered using the same k as the winning case from the previous phase. Perform the following pre-processing activities before running the k-means: scaling and outlier elimination, and defend your response concisely. (Suggestions: the order in which outliers are removed and the order in which they are scaled are critical.) Outlier elimination isn't addressed in tutorials, so you'll have to figure it out on your own). Define how many cluster centres there will be (via manual & automated tools). NBclust, Elbow, and a Gap statistic or silhouette approach should all be included in the automated tools. You must provide the R-outputs and an explanation of the results. Make a kmeans analysis using k=2, 3, and 4 using all input variables. Show all associated R-based kmeans outputs, including information for the centres and the ratio of between cluster sums of squares (BSS) over total sum of Squares, for each of the above k-means efforts Total Sum of Squares (TSS). Check your created cluster outcome against the information received from the 12th column for each of these k-means efforts, and give the relevant results/discussion (evidence of a "confusion-like" matrix (CM) and computation of the accuracy/recall/precision indices from it). Choose the best "winning" clustering example (justify your choice) and explain the accuracy, recall, and precision indices briefly. Because this is a typical multidimensional problem in terms of features, you must also use the PCA approach to analyse the wine dataset. We must display everything relevant to Principal Component Analysis (PCA) R-outputs. Create a new dataset that has principal components Principal Component (PC) as characteristics. Choose PCs with a cumulative score of at least 96 percent. Apply kmeans analysis to this "new altered" dataset with the same k as the previous step's winner. Show the kmeans analysis' relevant R-outputs. Calculate the related BSS, ratio BSS/TSS, and within cluster sums of squares (WSS) indices and compare them to the relevant ones from the winning model (i.e., all attributes) from the previous step to discuss the performance of this "PCA-based" kmeans model. Create R Studio code to solve all of the aforementioned concerns (codes, results, and discussion must be included in your report). Include the whole code you developed as an Appendix at the conclusion of your report. It is required to use the kmeans R function.

Part of Energy Forecasting

Buildings account for a significant portion of a country's energy usage and greenhouse gas emissions. The energy required to maintain internal building conditions accounts for a considerable share of overall energy consumption and greenhouse gas emissions. As a result, increasing building energy efficiency is critical to our overall sustainability. Many studies have been conducted over the last few decades to increase building energy efficiency using various methodologies and tactics. Demand response, fault detection and diagnosis, optimization, and energy management all need energy use forecasts in an existing facility. This is an example of a time_series application. Statistical and machine learning-based strategies are commonly used in data-driven forecasting models. The statistical strategy, which uses a pre-defined mathematical function, has demonstrated to be effective in forecasting energy over the medium to long term. Furthermore, such models have demonstrated adequate performance for estimating short-term consumption electrical demands. To offer a forecast, a machine learning technique often uses an algorithmic approach (which may non-linearly change the data). You will be working on a specific case study that incorporates a real-life organisation and a real dataset for this forecasting section of the curriculum. Most precisely, we were provided (through LG Power Station) with hourly power usage statistics (in kWh) for the University Building at 115 New Cavendish Street London for the years 2018 and 2019 in partnership with the Estates Planning & Services Department at University of Westminster. Despite the fact that we have been given all of the data, you will only utilise a fraction of it in this homework. For the 2018 and partially 2019 seasons, the given (UoW load.xlsx) file comprises daily power usage statistics for three hours (11:00, 10:00, and 09:00). (in total 500 samples). The goal of this issue is to utilise a multilayer neural network (MLP-NN) to forecast the following day's power use for the 11:00 hour scenario. Very first 430 results will be utilised for training, while the remaining samples will be used for testing.

Part of MLP

For this forecasting challenge, you'll need to build an MLP neural network. The specification of the NN input vector is a critical component of energy forecasting analysis. As a result, you must first offer a brief overview of the various schemes/methods utilised to describe this input vector in situations involving energy demand forecasting. (Add some relevant references after consulting related literature.) However, in this section of the forecasting, you will only use the "autoregressive" (AR) technique, which uses time-delayed values of the 11th hour characteristic as input variables. Because the order of this AR technique is unknown, you must experiment with various (time-delayed) input vectors and generate an input/output matrix (I/O) for Multilayer perceptron (MLP) training/testing (using "time-delayed" electrical loads) for each of these scenarios. Experiment up to the (t-4) level with various input vectors. According to the literature, the load's (t-7) value (one week prior) affects the power consumption projection. As a result, as part of your "AR" investigation, you should look into the impact of this unique time-delayed load on your NN models' predicting ability. You must additionally include other input vectors, such as information from the 10th and 9th hour qualities, in addition to this "traditional" AR technique. Your NN models could thus be classified as NARX (nonlinear autoregressive exogenous) models. This is a standard technique for this sort of NN, therefore each of these I/O matrices must be normalised. For this particular form of NN, you must briefly describe the logic for this normalisation technique. You should experiment with different MLPs during the training phase, using these input vectors and other internal network architectures (hidden layers, nodes, learning rate, activation function, and so on). The networks' testing performance (eg:evaluation) will be calculated for each instance using conventional statistical indices (RMSE, MAE, and MAPE). Make a table that compares their testing results (using these statistical metrics).

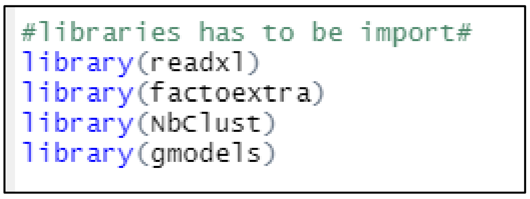
Explain the significance of these three statistical indicators in a few words. Check the "efficiency" of your best one-hidden layer and two-hidden layer networks by looking at the total number of weight parameters per network in this comparison table. Briefly explain why you choose one strategy over the other. Finally, show the outcomes of your finest MLP network graphically (prediction output vs. intended output) and using statistical indices. To meet all of these requirements, write code in R Studio. Display all of your work steps (code and results, including model comparison findings and internal structure). Because everyone's predicting results will be different, the focus in the marking scheme will be on the selected technique and the explanation/justification of various decisions you've made in order to produce an acceptable, performance-based answer. Your report must include all of your results, codes, and commentary. Include the whole code you developed as an Appendix at the conclusion of your report. MLP modelling requires the use of the neural net R function.

Objective 1: Partitioning Clustering

The data is first read into R studio from an excel file. The dataset is then cleaned up by removing any NaN values. After then, the outlier will be removed. Outliers can cause problems like erroneous data or predicted outcomes. Outliers increase data variability, lowering statistical power. By removing the outliers, we can make the data more understandable and accurate. In other cases, though, we will not need to delete outliers. In this case, outlier elimination would be desirable.

The project setup

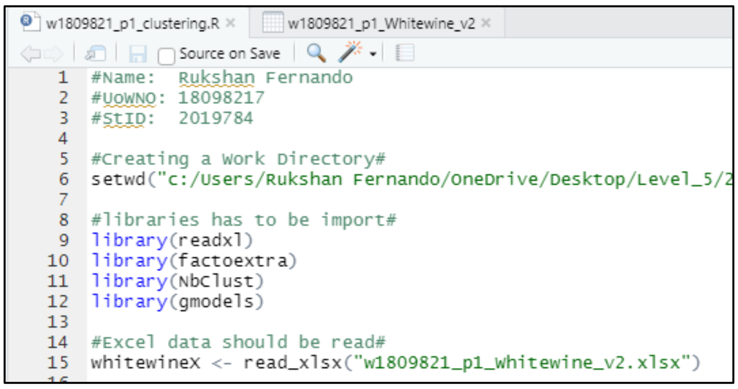
We must first import Libraries before loading the dataset. readxl, factoextra, NbClust and gmodels are among the libraries imported.



```r
#libraries has to be import#
library(readxl)
library(factoextra)
library(NbClust)
library(gmodels)
```

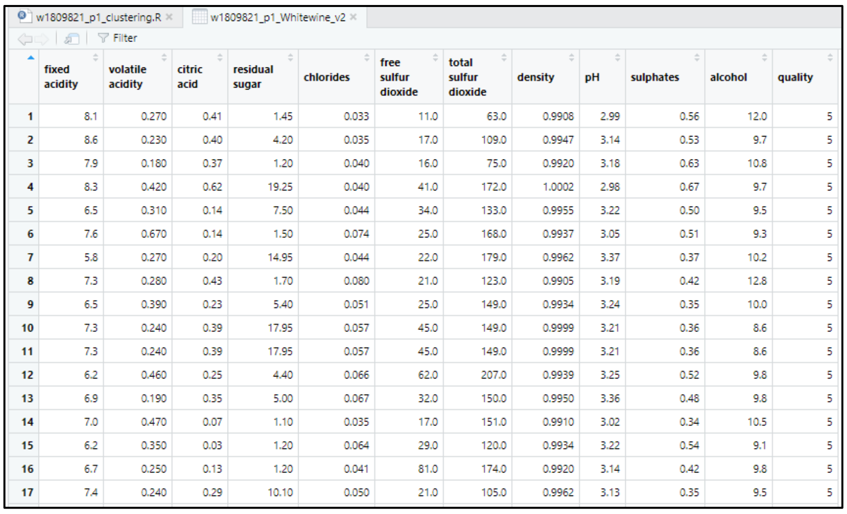
Figure 1: Import Libraries

Importing an xlsx file into a R program and setting up the working directory.



```r
#Name:  Rukshan Fernando
#UOWNO: 18098217
#StID:  2019784

#Creating a Work Directory#
setwd("C:/Users/Rukshan Fernando/OneDrive/Desktop/Level_5/2

#libraries has to be import#
library(readxl)
library(factoextra)
library(NbClust)
library(gmodels)

#Excel data should be read#
whitewineX <- read_xlsx("w1809821_p1_whitewine_v2.xlsx")
```

Figure 2: Data was read from an excel file and saved in a data frame.



w1809821_p1_clustering.R | w1809821_p1_Whitewine_v2

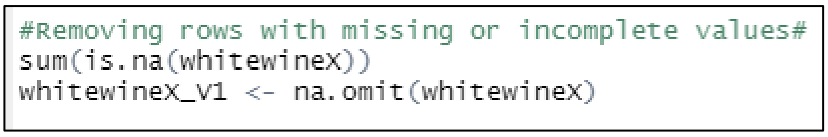
Filter

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
1	8.1	0.270	0.41	1.45	0.033	11.0	63.0	0.9908	2.99	0.56	12.0	5
2	8.6	0.230	0.40	4.20	0.035	17.0	109.0	0.9947	3.14	0.53	9.7	5
3	7.9	0.180	0.37	1.20	0.040	16.0	75.0	0.9920	3.18	0.63	10.8	5
4	8.3	0.420	0.62	19.25	0.040	41.0	172.0	1.0002	2.98	0.67	9.7	5
5	6.5	0.310	0.14	7.50	0.044	34.0	133.0	0.9955	3.22	0.50	9.5	5
6	7.6	0.670	0.14	1.50	0.074	25.0	168.0	0.9937	3.05	0.51	9.3	5
7	5.8	0.270	0.20	14.95	0.044	22.0	179.0	0.9962	3.37	0.37	10.2	5
8	7.3	0.280	0.43	1.70	0.080	21.0	123.0	0.9905	3.19	0.42	12.8	5
9	6.5	0.390	0.23	5.40	0.051	25.0	149.0	0.9934	3.24	0.35	10.0	5
10	7.3	0.240	0.39	17.95	0.057	45.0	149.0	0.9999	3.21	0.36	8.6	5
11	7.3	0.240	0.39	17.95	0.057	45.0	149.0	0.9999	3.21	0.36	8.6	5
12	6.2	0.460	0.25	4.40	0.066	62.0	207.0	0.9939	3.25	0.52	9.8	5
13	6.9	0.190	0.35	5.00	0.067	32.0	150.0	0.9950	3.36	0.48	9.8	5
14	7.0	0.470	0.07	1.10	0.035	17.0	151.0	0.9910	3.02	0.34	10.5	5
15	6.2	0.350	0.03	1.20	0.064	29.0	120.0	0.9934	3.22	0.54	9.1	5
16	6.7	0.250	0.13	1.20	0.041	81.0	174.0	0.9920	3.14	0.42	9.8	5
17	7.4	0.240	0.29	10.10	0.050	21.0	105.0	0.9962	3.13	0.35	9.5	5

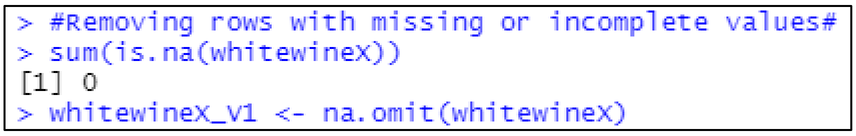
Figure 3: Dataset including white wine-related information

Pre-processing of data

Before dealing with a data collection, it must first be cleansed. Cleaning and preparing data for analysis is known as data preprocessing. After that, any missing or partial values or rows are removed from the data collection.



```
#Removing rows with missing or incomplete values#
sum(is.na(whitewineX))
whitewineX_V1 <- na.omit(whitewineX)
```



```
> #Removing rows with missing or incomplete values#
> sum(is.na(whitewineX))
[1] 0
> whitewineX_V1 <- na.omit(whitewineX)
```

Figure 4: Obtaining and deleting missing values

(A value of "[1] 0" indicates that no missing values or partial data exist.)

Create Outliers

A number that is notably different from the rest of the data is called an outlier. It has the power to directly alter the conclusion of the analysis and produce mistakes.

To guarantee that anomalous data does not impact the outcome, outliers must be removed from the data set. There are two approaches to spot outliers, Interquartile range and Z-scores.

Between these two, the Interquartile range (IQR) approach is used in this partitioning study. In a dataset, the IQR is the difference between the third (Q3) and first quarter (Q1), and it indicates the difference between the third (Q3) and first quarter (Q1) (Q1). This is necessary in order to determine the dispersion of values in the middle 50%. The boxplot R tool may be used to determine this. The numbers outside of the Q1 and Q3 range are displayed. The outcomes of removing outliers are presented following.

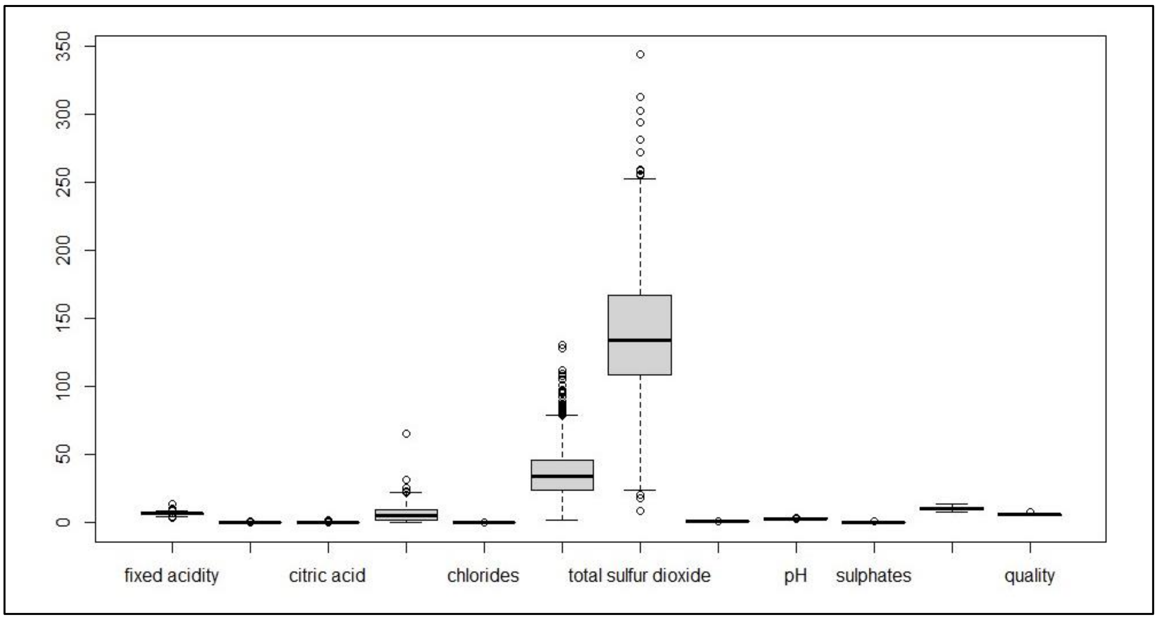
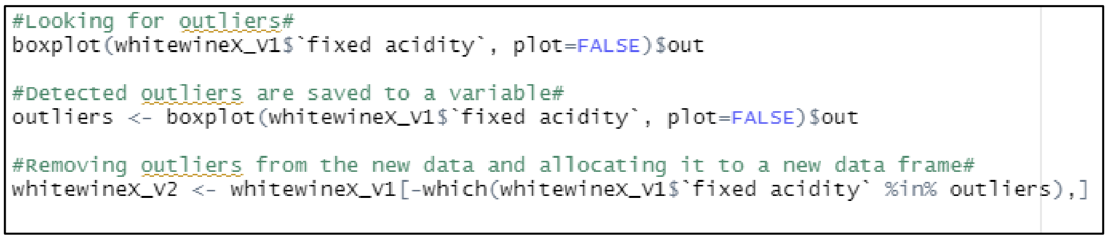


Figure 5: Outliers before removing - All of

Outliers Removing

A column's outliers are examined and a new data frame is created. Delete the rows containing records from the discovered outliers, then allocate the outliers removed data set to a new data frame.



```
#Looking for outliers#
boxplot(whitewinex_V1$`fixed acidity`, plot=FALSE)$out

#Detected outliers are saved to a variable#
outliers <- boxplot(whitewinex_V1$`fixed acidity`, plot=FALSE)$out

#Removing outliers from the new data and allocating it to a new data frame#
whitewinex_V2 <- whitewinex_V1[-which(whitewinex_V1$`fixed acidity` %in% outliers),]
```

Figure 6: Outliers removing

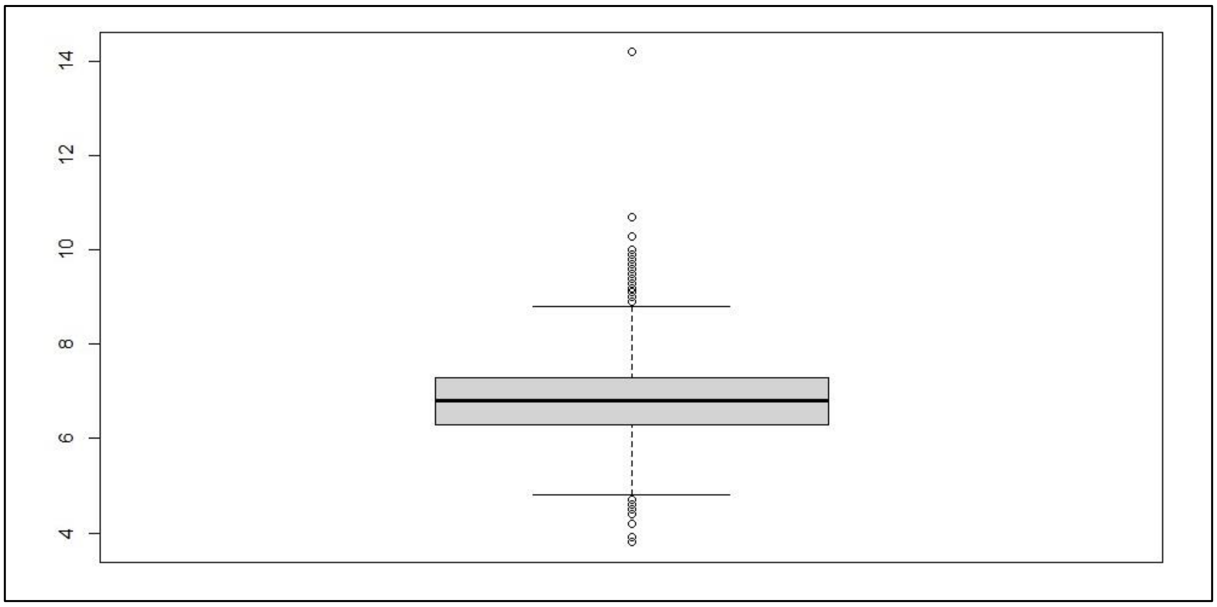


Figure 7: Outliers before removing

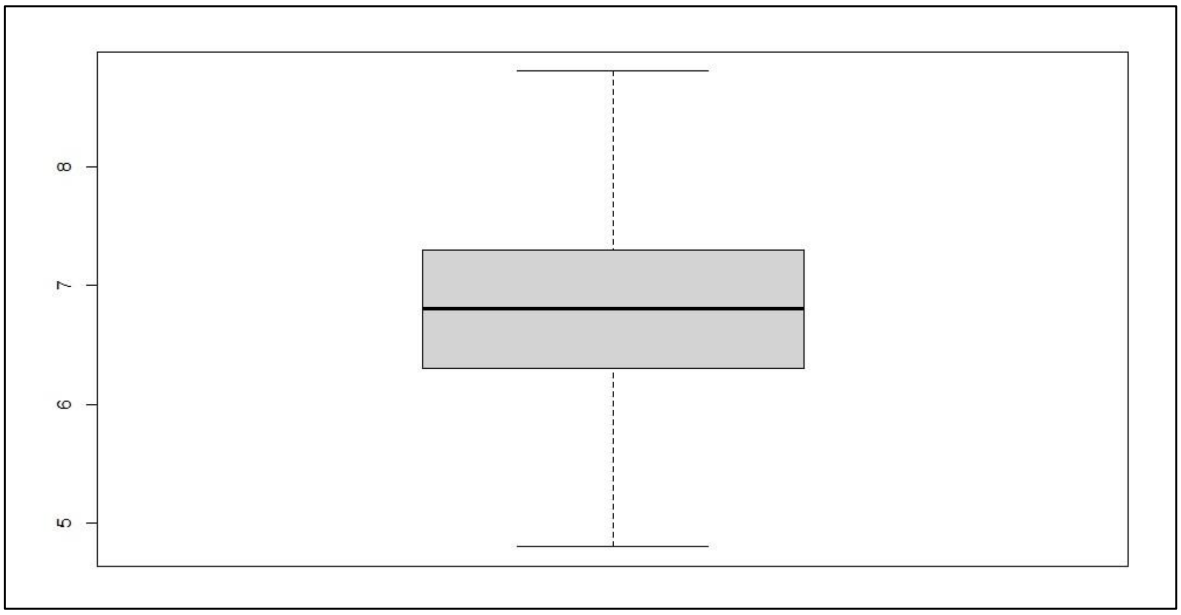
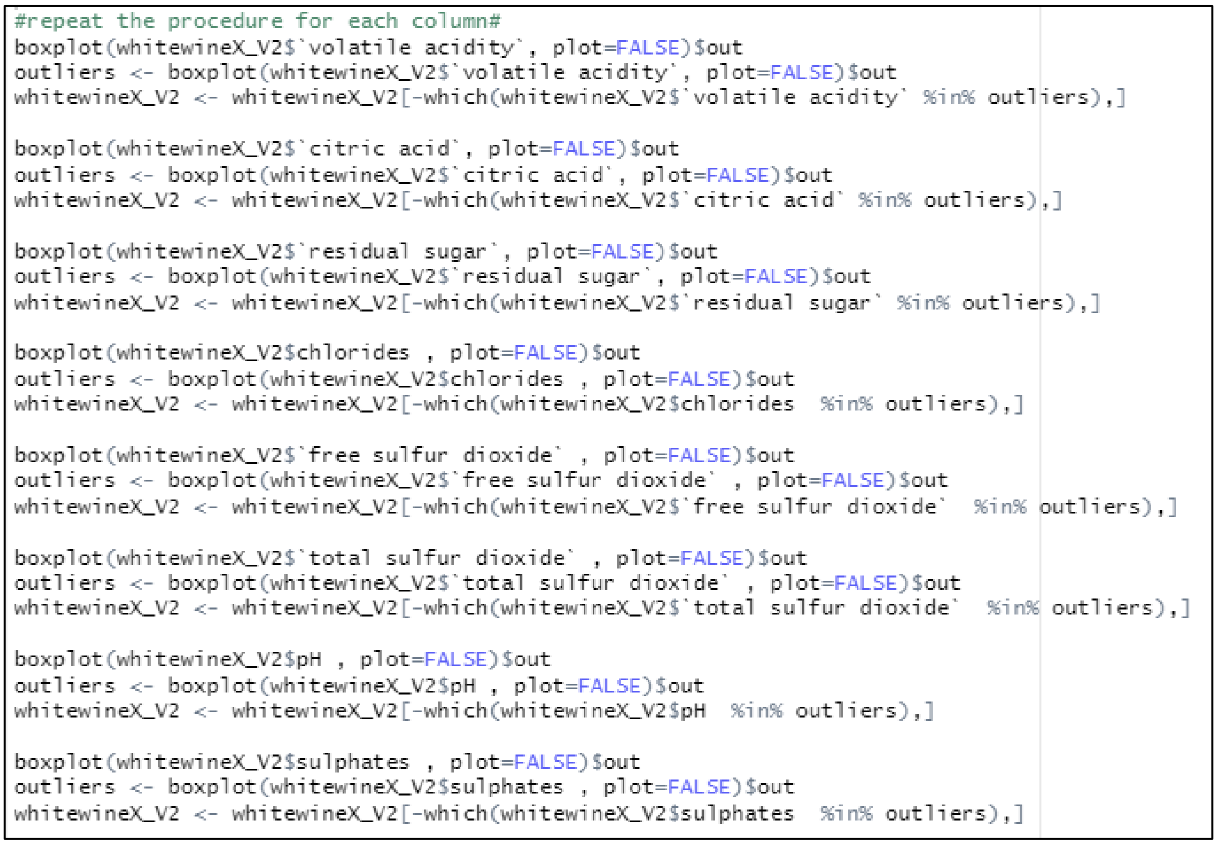


Figure 8: Outliers after removing

Next, one by one, the identical code repeats the operation for all the columns that include outliers. This procedure might even be executed using a loop, but I'm not familiar with R and don't have much expertise with it. As a result, rather than coding ethics and general simplicity, I concentrated on the data's functional elements.



```
#repeat the procedure for each column#
boxplot(whitewineX_V2$`volatile acidity`, plot=FALSE)$out
outliers <- boxplot(whitewineX_V2$`volatile acidity`, plot=FALSE)$out
whitewineX_V2 <- whitewineX_V2[-which(whitewineX_V2$`volatile acidity` %in% outliers),]

boxplot(whitewineX_V2$`citric acid`, plot=FALSE)$out
outliers <- boxplot(whitewineX_V2$`citric acid`, plot=FALSE)$out
whitewineX_V2 <- whitewineX_V2[-which(whitewineX_V2$`citric acid` %in% outliers),]

boxplot(whitewineX_V2$`residual sugar`, plot=FALSE)$out
outliers <- boxplot(whitewineX_V2$`residual sugar`, plot=FALSE)$out
whitewineX_V2 <- whitewineX_V2[-which(whitewineX_V2$`residual sugar` %in% outliers),]

boxplot(whitewineX_V2$chlorides , plot=FALSE)$out
outliers <- boxplot(whitewineX_V2$chlorides , plot=FALSE)$out
whitewineX_V2 <- whitewineX_V2[-which(whitewineX_V2$chlorides  %in% outliers),]

boxplot(whitewineX_V2$`free sulfur dioxide` , plot=FALSE)$out
outliers <- boxplot(whitewineX_V2$`free sulfur dioxide` , plot=FALSE)$out
whitewineX_V2 <- whitewineX_V2[-which(whitewineX_V2$`free sulfur dioxide`  %in% outliers),]

boxplot(whitewineX_V2$`total sulfur dioxide` , plot=FALSE)$out
outliers <- boxplot(whitewineX_V2$`total sulfur dioxide` , plot=FALSE)$out
whitewineX_V2 <- whitewineX_V2[-which(whitewineX_V2$`total sulfur dioxide`  %in% outliers),]

boxplot(whitewineX_V2$pH , plot=FALSE)$out
outliers <- boxplot(whitewineX_V2$pH , plot=FALSE)$out
whitewineX_V2 <- whitewineX_V2[-which(whitewineX_V2$pH  %in% outliers),]

boxplot(whitewineX_V2$sulphates , plot=FALSE)$out
outliers <- boxplot(whitewineX_V2$sulphates , plot=FALSE)$out
whitewineX_V2 <- whitewineX_V2[-which(whitewineX_V2$sulphates  %in% outliers),]
```

Figure 9: All the outliers removing

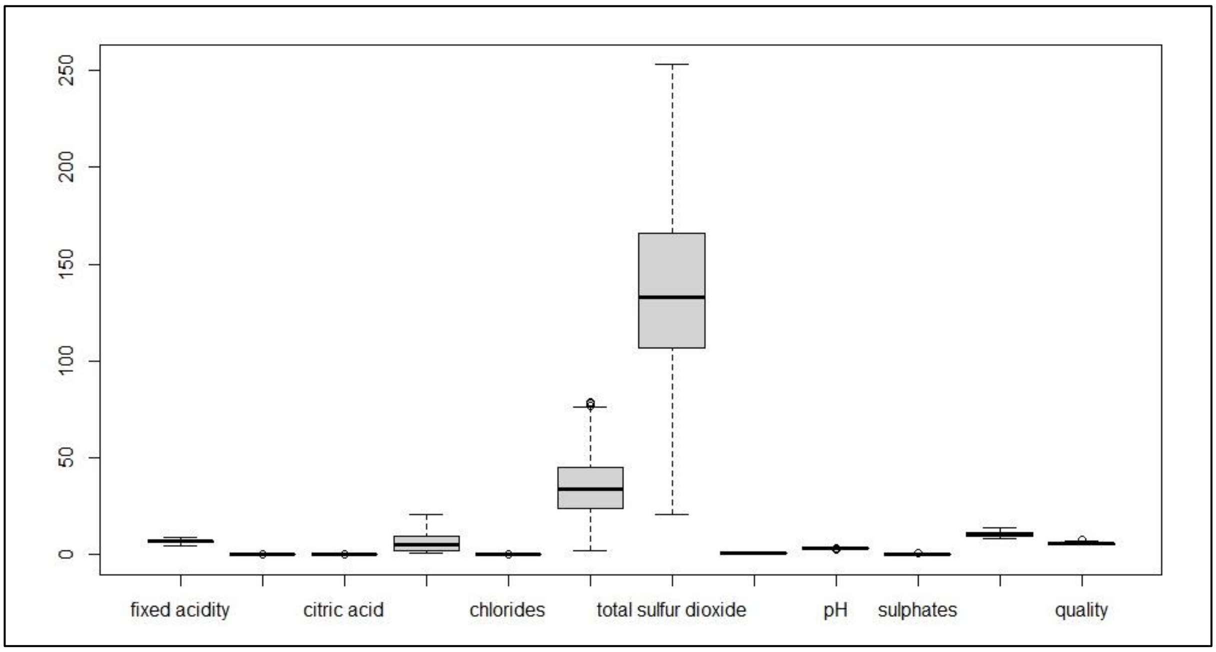


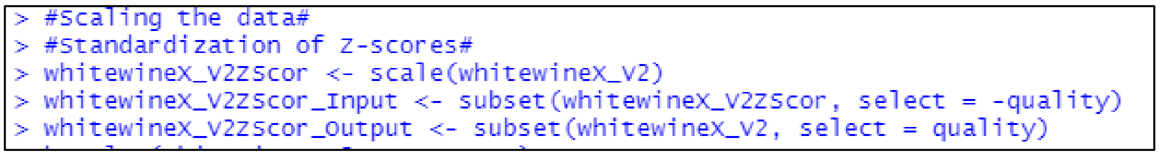
Figure 10: All the outliers after removing

Even after eliminating the outliers, some data points fall outside the (Q1) and (Q3) categories. It's because when outliers are removed, the data spread's 50 percent point, or mean, is revised. There may be new outliers as a result of the enhanced 50% mean. Because the additional outliers would be in the (Q1) and (Q3) ranges in the original data set, they had no impact on the analysis.

Data Standardisation

The process of transforming the values of diverse columns in a data gathering to a similar scale is known as standardisation. In Z – score standardisation, the mean of a column is 0, and the results go up and down appropriately. Because standardisation utilises the data's mean as the middle point, deleting outliers after standardisation might modify the mean data point, resulting in erroneous data points and influencing clustering.

<u>Using Z – score Standardization to scale data</u>



```
> #Scaling the data#
> #Standardization of Z-scores#
> whitewineX_v2ZScor <- scale(whitewineX_v2)
> whitewineX_v2ZScor_Input <- subset(whitewineX_v2ZScor, select = -quality)
> whitewineX_v2ZScor_Output <- subset(whitewineX_v2, select = quality)
```

Figure 11: Using Z–score Standardization to scale data

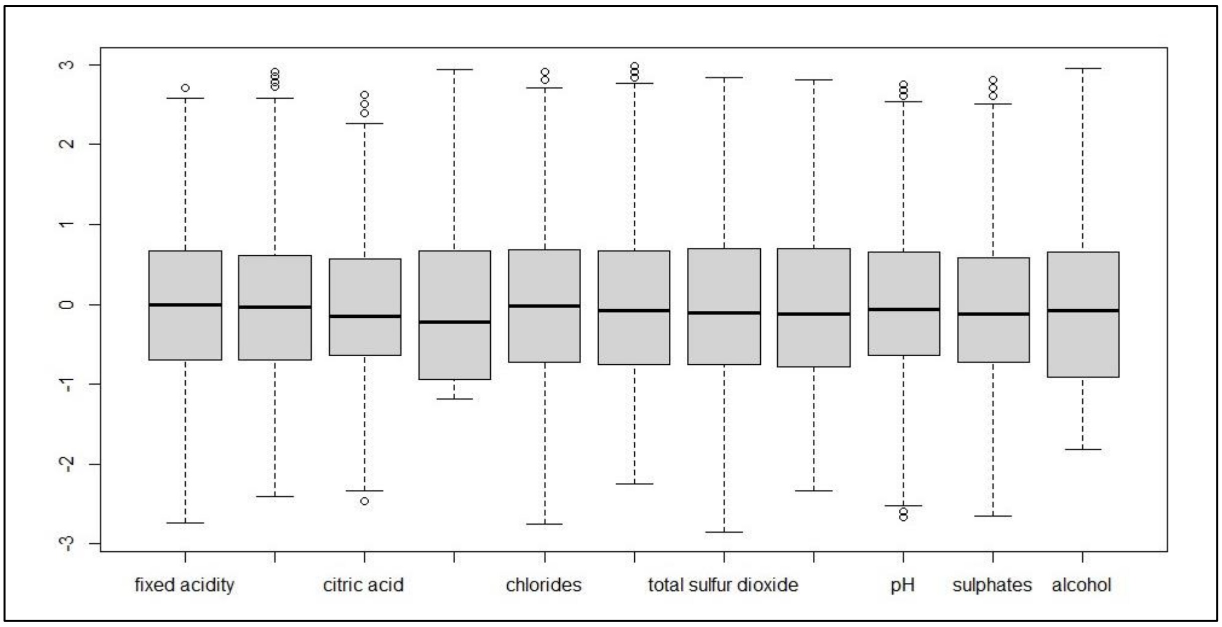
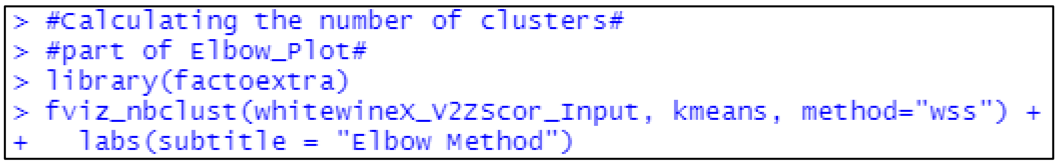


Figure 12: Data Standaradization

Choosing the Right Number of Clusters

The number of clusters must be specified when using K-means clustering. There are a few things you may perform in this case before determining how many clusters are required. Some of these techniques are listed down.

Functionality of Elbow Plot



```
> #Calculating the number of clusters#
> #part of Elbow_Plot#
> library(factoextra)
> fviz_nbclust(whitewineX_V2ZScor_Input, kmeans, method="wss") +
+   labs(subtitle = "Elbow Method")
```

Figure 13: Functionality of Elbow Plot

Result:

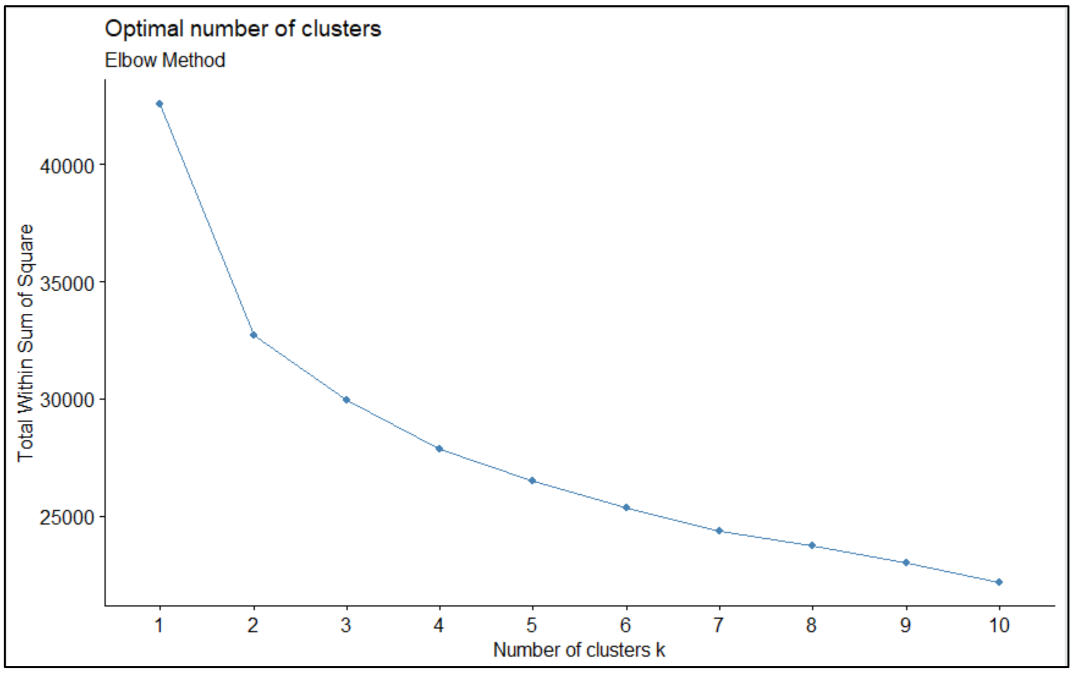


Figure 14: Elbow Plot Diagram

Method of Silhouette

```
> #Method of Silhouette#
> fviz_nbclust(whitewineX_V2ZScor_Input, kmeans, method="silhouette") +
+   labs(subtitle = "Silhouette Method")
```

Figure 15: Method of Silhouette

Result:

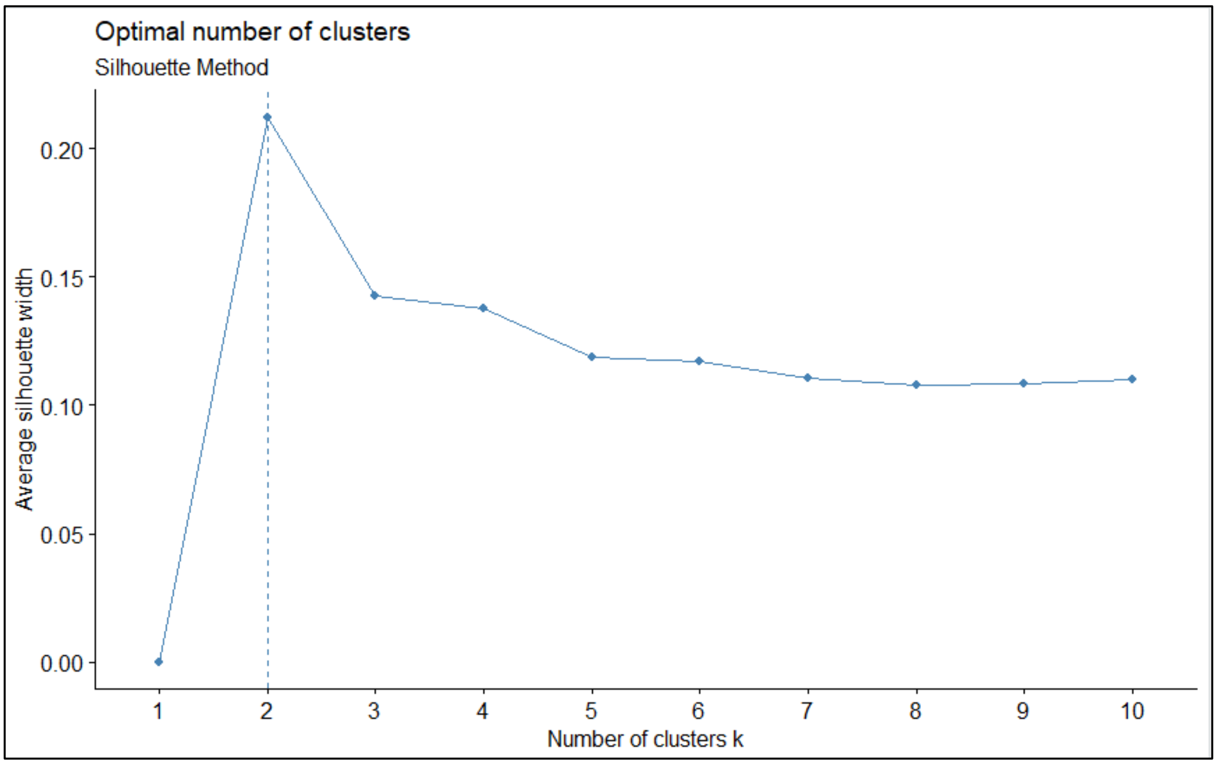


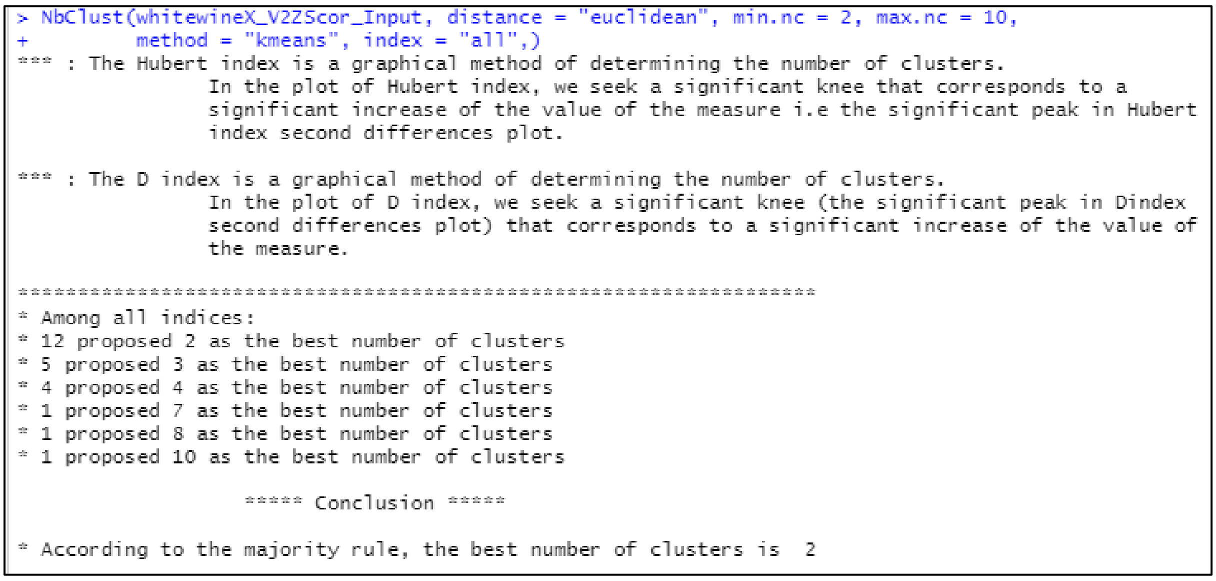
Figure 16: Method of Silhouette Diagram

Method of Nb Clust

```
#Part of NBclust#
NbClust(whitewineX_V2ZScor_Input, distance = "euclidean", min.nc = 2, max.nc = 10,
        method = "kmeans", index = "all",)
```

Figure 17: Method of NB Clust

Result:



```
> NbClust(whitewineX_V2ZScor_Input, 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:
* 12 proposed 2 as the best number of clusters
* 5 proposed 3 as the best number of clusters
* 4 proposed 4 as the best number of clusters
* 1 proposed 7 as the best number of clusters
* 1 proposed 8 as the best number of clusters
* 1 proposed 10 as the best number of clusters

                   ***** Conclusion *****

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

Figure 18: Method of NB Clust (Output)

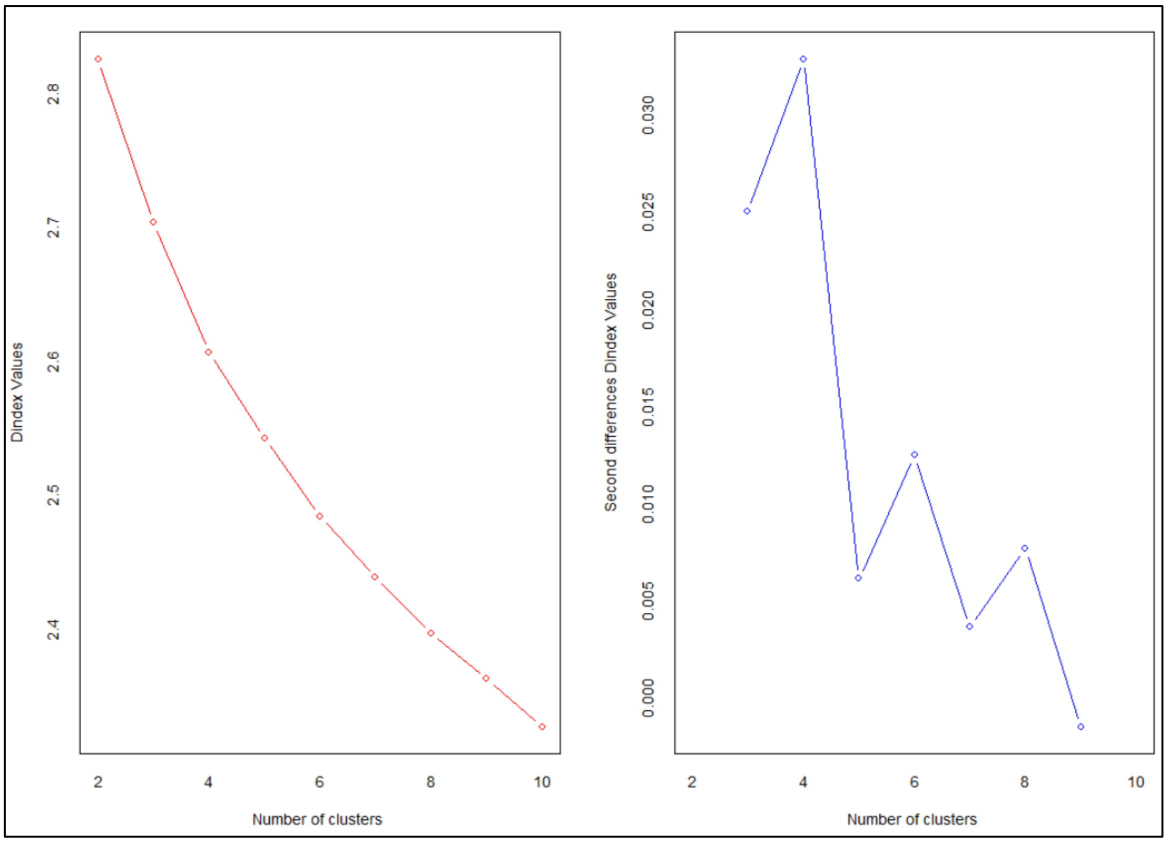


Figure 19: Method of NB Clust Diagram

Part of K-Means Clustering

K-means Clustering is a type of unsupervised learning. It's used to look for patterns and correlations in unlabeled data before identifying them using the provided cluster (K).

<u>If K = 2;</u>

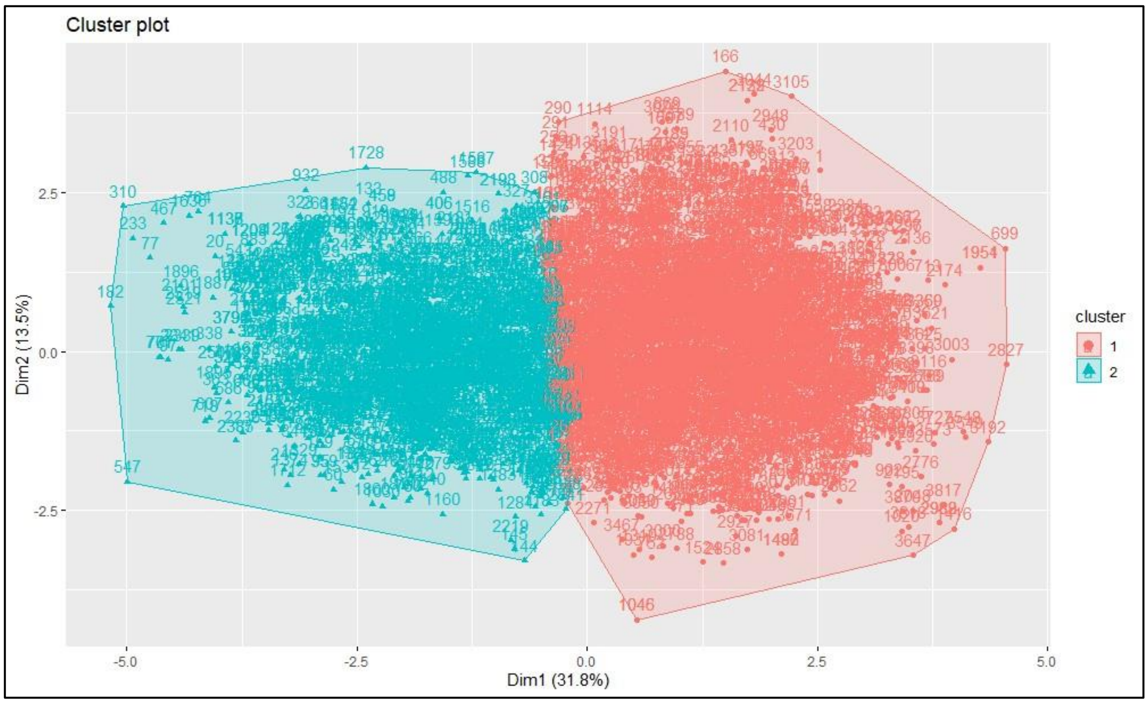


Figure 20: If K = 2

Between Sum of Squares(BSS) = 9855.06

Total Sum of Squares (TSS) = 42581

$$\frac{(BSS)}{(TSS)} = 0.2314 \text{ //}$$

<u>If K=3;</u>

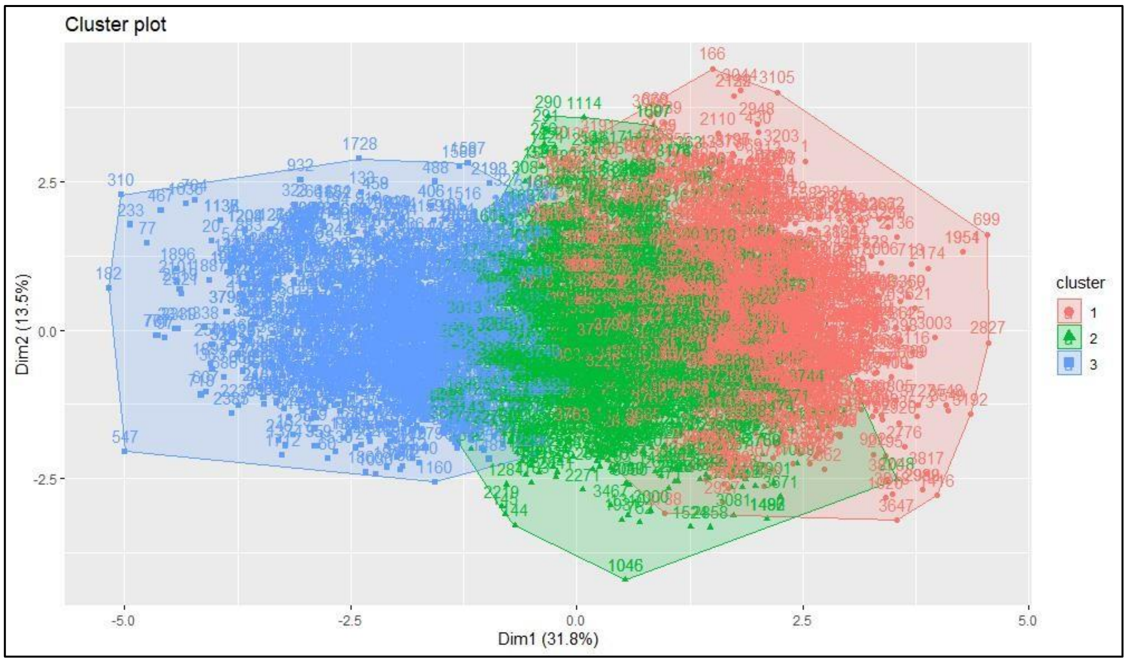


Figure 21: If K = 3

Between Sum of Squares(BSS) = 12647.62

Total Sum of Squares (TSS) = 42581

$$\frac{(BSS)}{(TSS)} = 0.2970 \; //$$

<u>If K=4;</u>

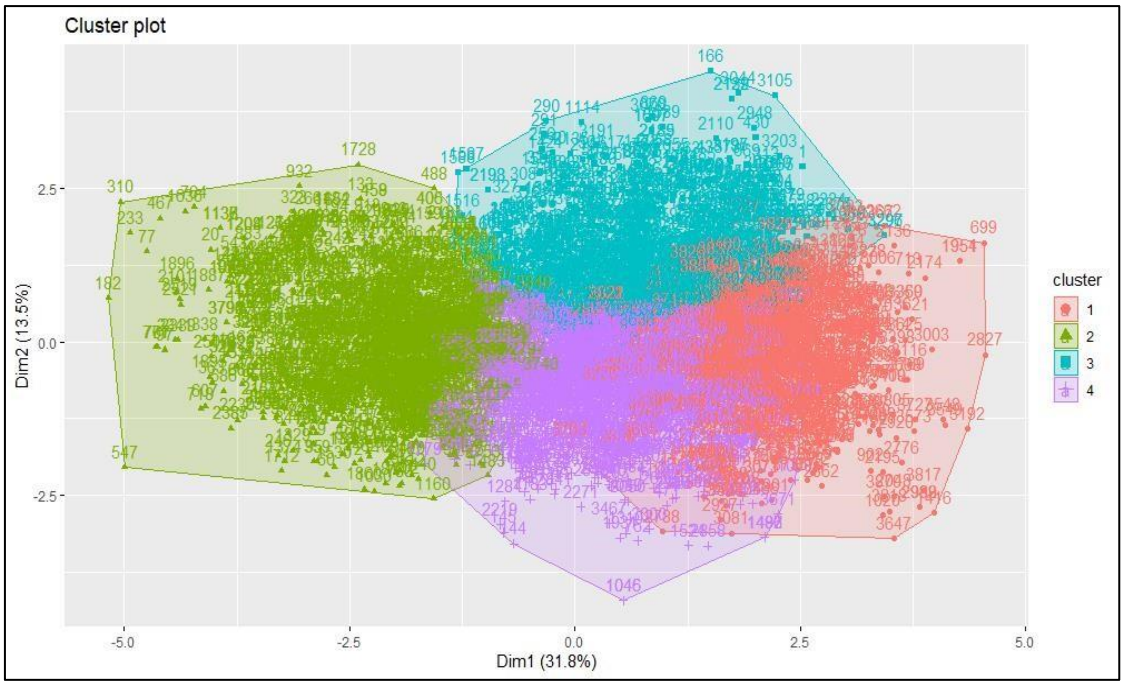


Figure 22: If K = 4

Between Sum of Squares(BSS) = 14746.06

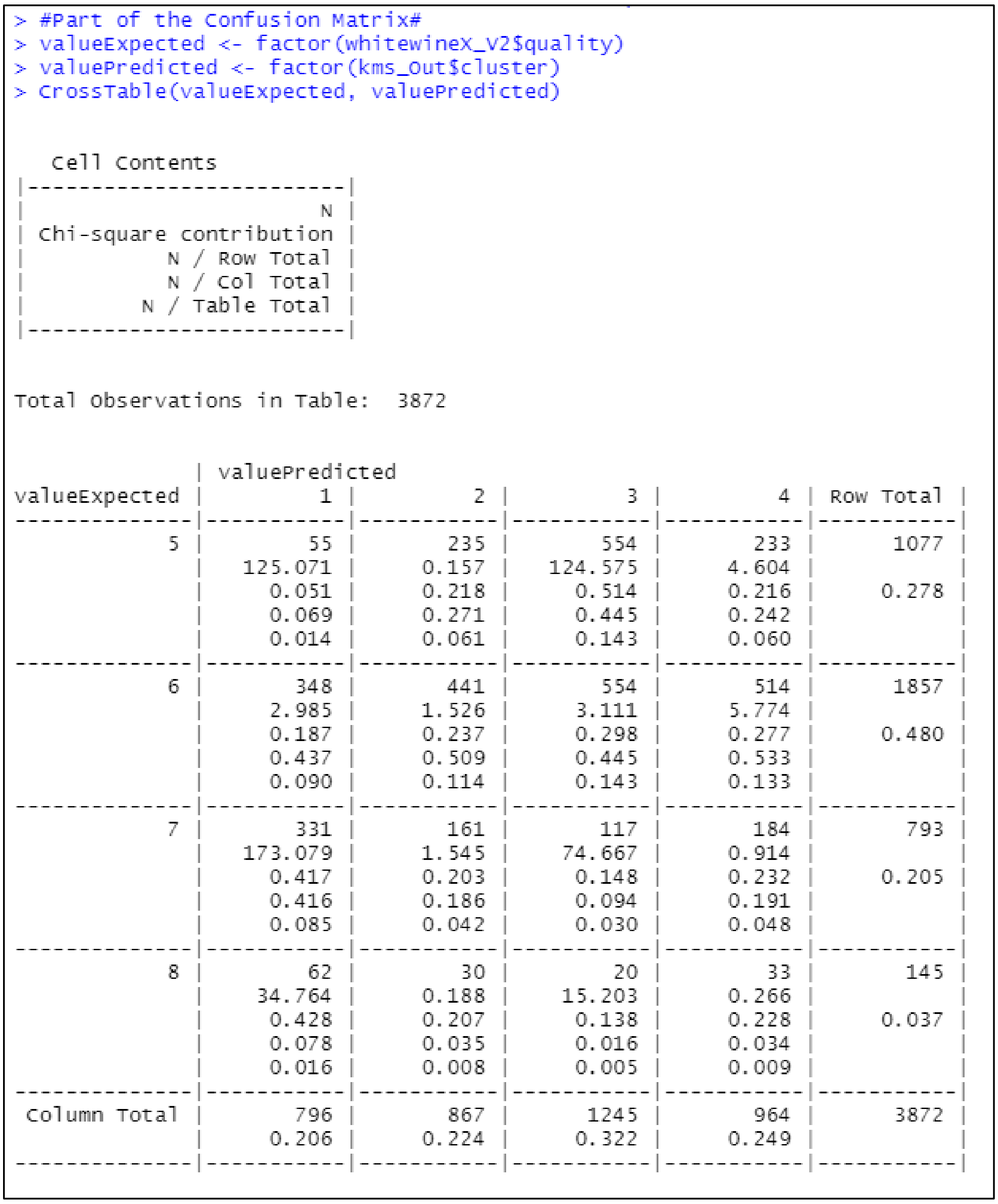
Total Sum of Squares (TSS) = 42581

$\frac{(BSS)}{(TSS)}$ = 0.3463 //

The results were assessed towards the 12th column

Each K cluster is evaluated using the confusion matrix.

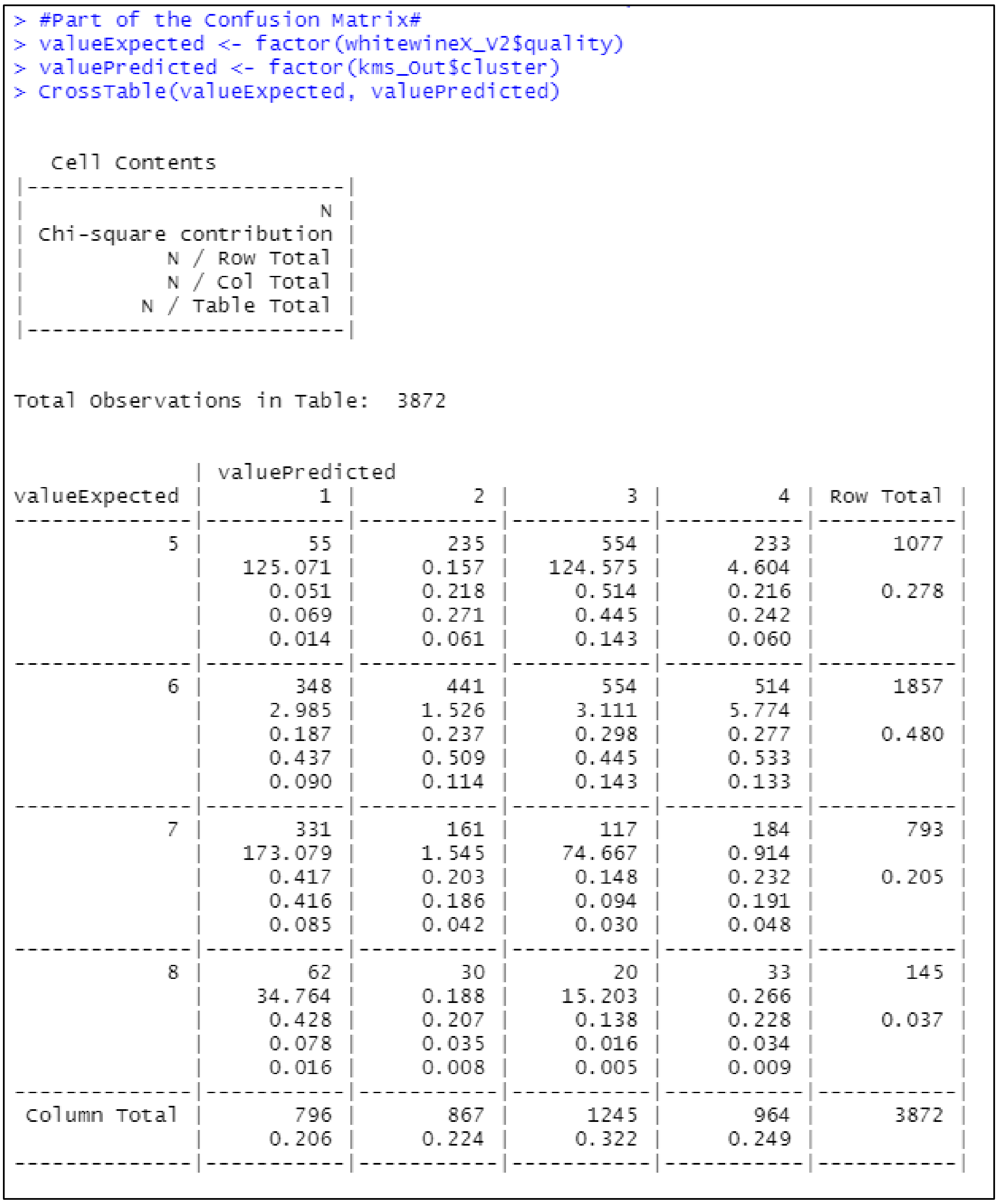
<u>If Confusion Matrix for K=2;</u>



```
> #Part of the Confusion Matrix#
> valueExpected <- factor(whitewineX_V2$quality)
> valuePredicted <- factor(kms_Out$cluster)
> CrossTable(valueExpected, valuePredicted)


   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  3872


              | valuePredicted
valueExpected |           1 |           2 |           3 |           4 | Row Total |
--------------|-------------|-------------|-------------|-------------|-----------|
            5 |          55 |         235 |         554 |         233 |      1077 |
              |     125.071 |       0.157 |     124.575 |       4.604 |           |
              |       0.051 |       0.218 |       0.514 |       0.216 |     0.278 |
              |       0.069 |       0.271 |       0.445 |       0.242 |           |
              |       0.014 |       0.061 |       0.143 |       0.060 |           |
--------------|-------------|-------------|-------------|-------------|-----------|
            6 |         348 |         441 |         554 |         514 |      1857 |
              |       2.985 |       1.526 |       3.111 |       5.774 |           |
              |       0.187 |       0.237 |       0.298 |       0.277 |     0.480 |
              |       0.437 |       0.509 |       0.445 |       0.533 |           |
              |       0.090 |       0.114 |       0.143 |       0.133 |           |
--------------|-------------|-------------|-------------|-------------|-----------|
            7 |         331 |         161 |         117 |         184 |       793 |
              |     173.079 |       1.545 |      74.667 |       0.914 |           |
              |       0.417 |       0.203 |       0.148 |       0.232 |     0.205 |
              |       0.416 |       0.186 |       0.094 |       0.191 |           |
              |       0.085 |       0.042 |       0.030 |       0.048 |           |
--------------|-------------|-------------|-------------|-------------|-----------|
            8 |          62 |          30 |          20 |          33 |       145 |
              |      34.764 |       0.188 |      15.203 |       0.266 |           |
              |       0.428 |       0.207 |       0.138 |       0.228 |     0.037 |
              |       0.078 |       0.035 |       0.016 |       0.034 |           |
              |       0.016 |       0.008 |       0.005 |       0.009 |           |
--------------|-------------|-------------|-------------|-------------|-----------|
 Column Total |         796 |         867 |        1245 |         964 |      3872 |
              |       0.206 |       0.224 |       0.322 |       0.249 |           |
--------------|-------------|-------------|-------------|-------------|-----------|
```

Figure 23: Confusion Matrix for K=2

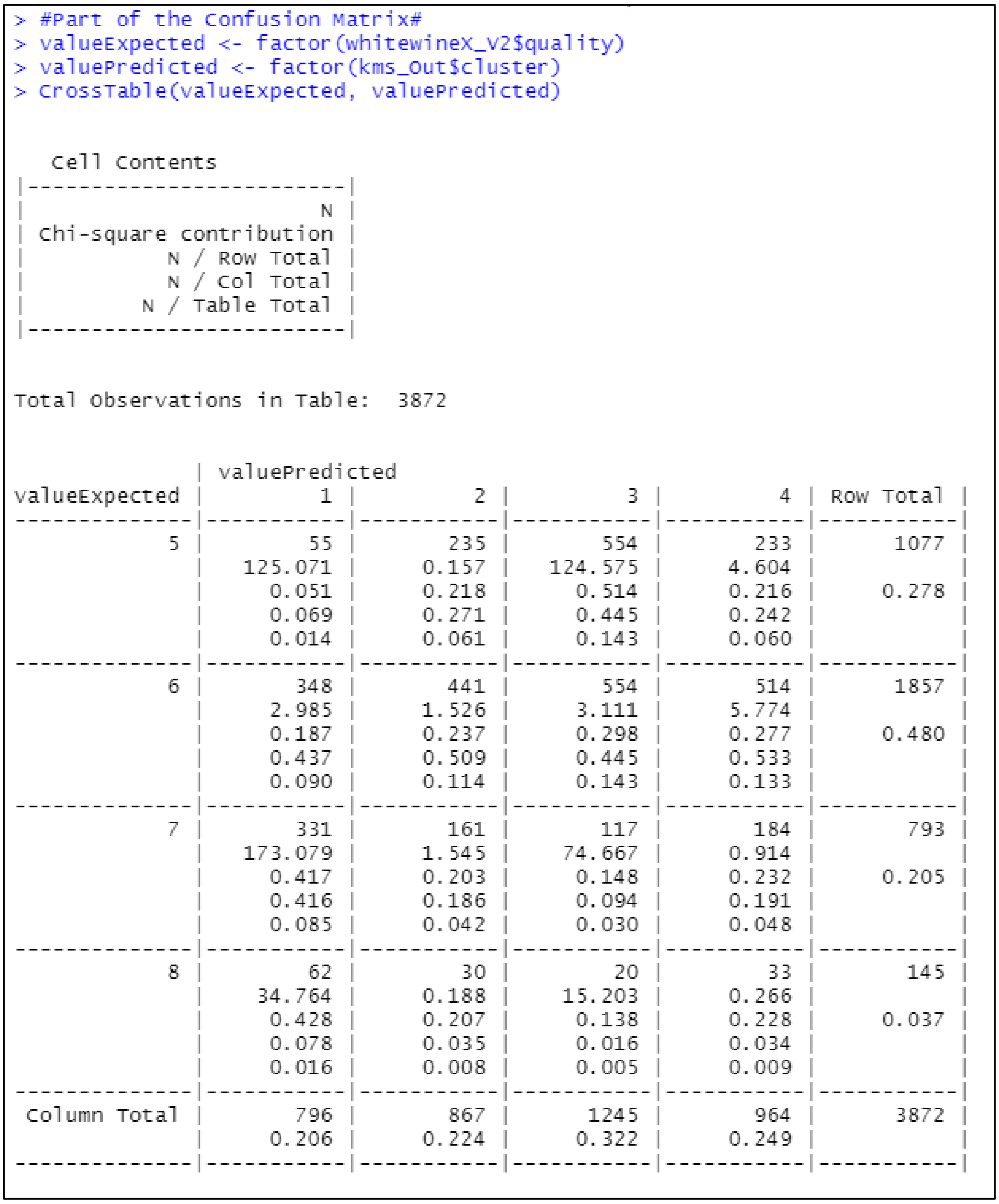
<u>If Confusion Matrix for K=3;</u>



```
> #Part of the Confusion Matrix#
> valueExpected <- factor(whitewineX_V2$quality)
> valuePredicted <- factor(kms_Out$cluster)
> CrossTable(valueExpected, valuePredicted)


   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  3872


              | valuePredicted
valueExpected |           1 |           2 |           3 |           4 | Row Total |
--------------|-------------|-------------|-------------|-------------|-----------|
            5 |          55 |         235 |         554 |         233 |      1077 |
              |     125.071 |       0.157 |     124.575 |       4.604 |           |
              |       0.051 |       0.218 |       0.514 |       0.216 |     0.278 |
              |       0.069 |       0.271 |       0.445 |       0.242 |           |
              |       0.014 |       0.061 |       0.143 |       0.060 |           |
--------------|-------------|-------------|-------------|-------------|-----------|
            6 |         348 |         441 |         554 |         514 |      1857 |
              |       2.985 |       1.526 |       3.111 |       5.774 |           |
              |       0.187 |       0.237 |       0.298 |       0.277 |     0.480 |
              |       0.437 |       0.509 |       0.445 |       0.533 |           |
              |       0.090 |       0.114 |       0.143 |       0.133 |           |
--------------|-------------|-------------|-------------|-------------|-----------|
            7 |         331 |         161 |         117 |         184 |       793 |
              |     173.079 |       1.545 |      74.667 |       0.914 |           |
              |       0.417 |       0.203 |       0.148 |       0.232 |     0.205 |
              |       0.416 |       0.186 |       0.094 |       0.191 |           |
              |       0.085 |       0.042 |       0.030 |       0.048 |           |
--------------|-------------|-------------|-------------|-------------|-----------|
            8 |          62 |          30 |          20 |          33 |       145 |
              |      34.764 |       0.188 |      15.203 |       0.266 |           |
              |       0.428 |       0.207 |       0.138 |       0.228 |     0.037 |
              |       0.078 |       0.035 |       0.016 |       0.034 |           |
              |       0.016 |       0.008 |       0.005 |       0.009 |           |
--------------|-------------|-------------|-------------|-------------|-----------|
 Column Total |         796 |         867 |        1245 |         964 |      3872 |
              |       0.206 |       0.224 |       0.322 |       0.249 |           |
--------------|-------------|-------------|-------------|-------------|-----------|
```

Figure 24: Confusion Matrix for K=3

If Confusion Matrix for K=4;



```
> #Part of the Confusion Matrix#
> valueExpected <- factor(whitewinex_v2$quality)
> valuePredicted <- factor(kms_out$cluster)
> CrossTable(valueExpected, valuePredicted)


   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  3872


              | valuePredicted
valueExpected |          1 |          2 |          3 |          4 | Row Total |
--------------|------------|------------|------------|------------|-----------|
            5 |         55 |        235 |        554 |        233 |      1077 |
              |    125.071 |      0.157 |    124.575 |      4.604 |           |
              |      0.051 |      0.218 |      0.514 |      0.216 |     0.278 |
              |      0.069 |      0.271 |      0.445 |      0.242 |           |
              |      0.014 |      0.061 |      0.143 |      0.060 |           |
--------------|------------|------------|------------|------------|-----------|
            6 |        348 |        441 |        554 |        514 |      1857 |
              |      2.985 |      1.526 |      3.111 |      5.774 |           |
              |      0.187 |      0.237 |      0.298 |      0.277 |     0.480 |
              |      0.437 |      0.509 |      0.445 |      0.533 |           |
              |      0.090 |      0.114 |      0.143 |      0.133 |           |
--------------|------------|------------|------------|------------|-----------|
            7 |        331 |        161 |        117 |        184 |       793 |
              |    173.079 |      1.545 |     74.667 |      0.914 |           |
              |      0.417 |      0.203 |      0.148 |      0.232 |     0.205 |
              |      0.416 |      0.186 |      0.094 |      0.191 |           |
              |      0.085 |      0.042 |      0.030 |      0.048 |           |
--------------|------------|------------|------------|------------|-----------|
            8 |         62 |         30 |         20 |         33 |       145 |
              |     34.764 |      0.188 |     15.203 |      0.266 |           |
              |      0.428 |      0.207 |      0.138 |      0.228 |     0.037 |
              |      0.078 |      0.035 |      0.016 |      0.034 |           |
              |      0.016 |      0.008 |      0.005 |      0.009 |           |
--------------|------------|------------|------------|------------|-----------|
 Column Total |        796 |        867 |       1245 |        964 |      3872 |
              |      0.206 |      0.224 |      0.322 |      0.249 |           |
--------------|------------|------------|------------|------------|-----------|
```

Figure 25: Confusion Matrix for K=4

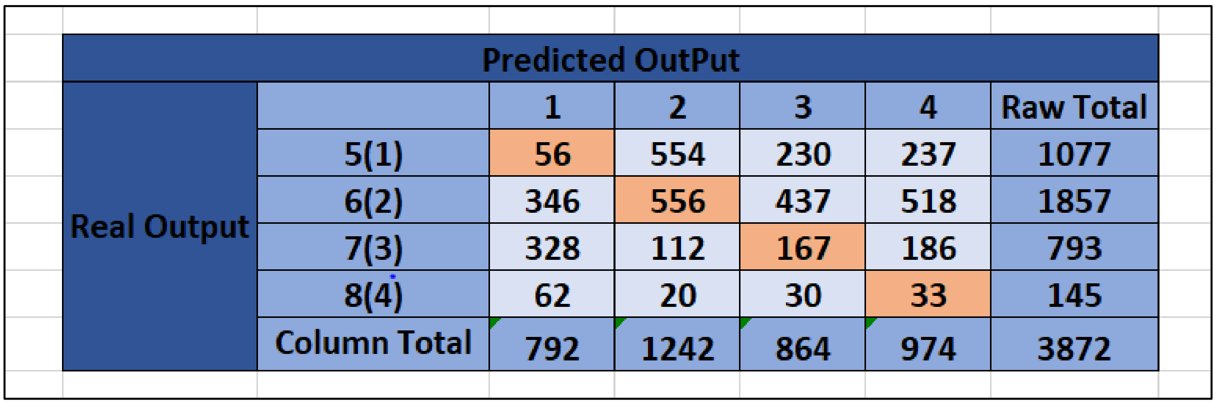
Accuracy of Part A

The amount of accurate classifications is divided by the total number of datasets to determine accuracy. The greatest accuracy level is (1.0), while the poorest is (0.0). In layman's terms, accuracy is the proportion of correctly predicted instances to all examples.

Accuracy = Number of correct predictions / Total number of predictions

By terms of the Confusion Matrix, it is as follows;

Accuracy = TP + TN / TP + TN + FP + FN



		Predicted OutPut				
		1	2	3	4	Raw Total
Real Output	5(1)	56	554	230	237	1077
	6(2)	346	556	437	518	1857
	7(3)	328	112	167	186	793
	8(4)	62	20	30	33	145
	Column Total	792	1242	864	974	3872

Figure 26: Accuracy Rate Table

Total accuracy= $\dfrac{(56+556+167+33)}{3872}$

$= \dfrac{812}{3872}$

$= 0.209$

$= 20.9\%$ //

Part of Principal Component Analysis (PCA) Techniques

"PCA" is a dimensional reduction approach that lowers a dataset's features to a lesser number of features known as main components while keeping as much information in the original data as feasible.

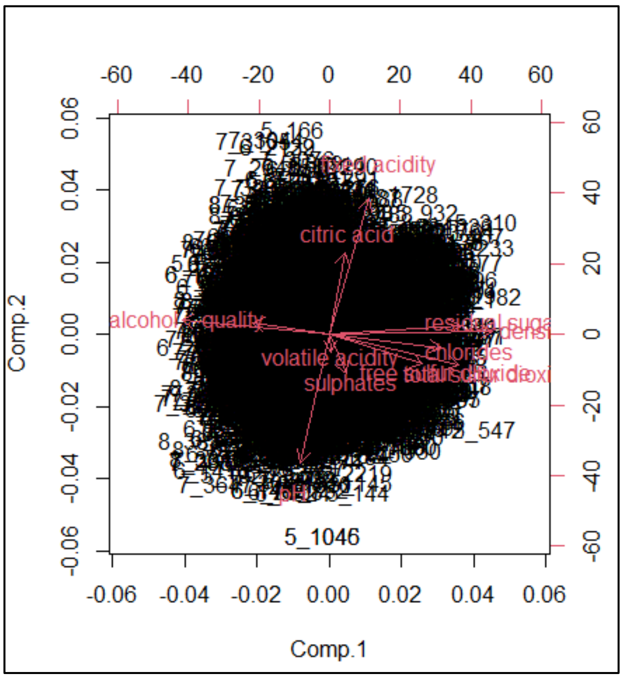


Figure 27: Representation of the Part of Principal Component Analysis (PCA)

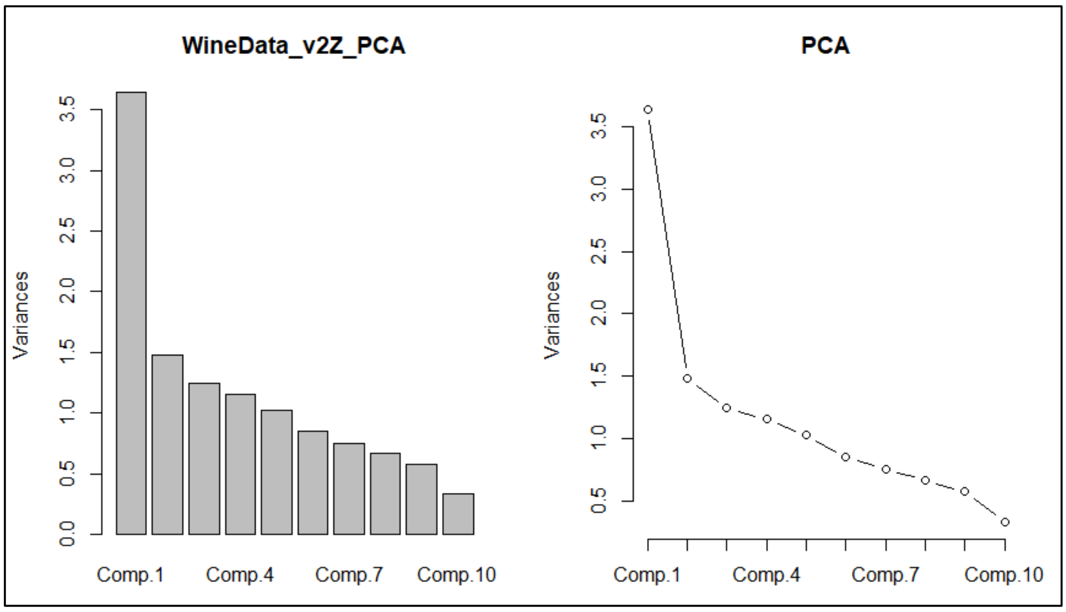
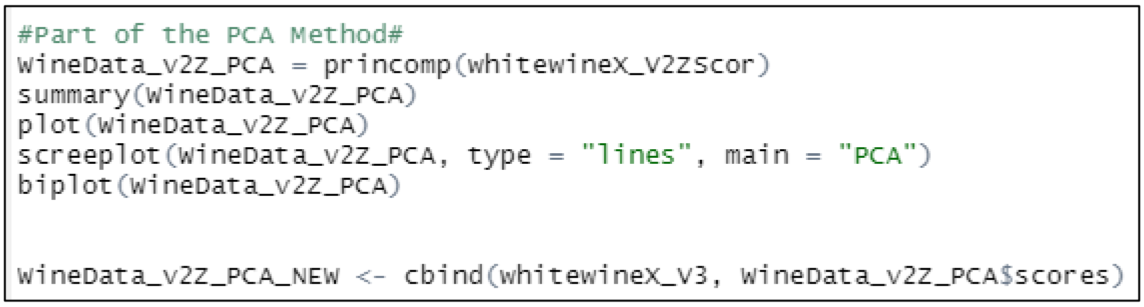


Figure 28: The distribution of variance among PCs

New collection of data

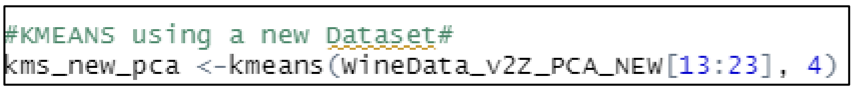
The task directs students to create a new dataset among those PCs that have a final score of higher than 96 percent; in this case, all the nine components should be used with.



```
#Part of the PCA Method#
WineData_v2Z_PCA = princomp(whitewineX_v2ZScor)
summary(WineData_v2Z_PCA)
plot(WineData_v2Z_PCA)
screeplot(WineData_v2Z_PCA, type = "lines", main = "PCA")
biplot(WineData_v2Z_PCA)


WineData_v2Z_PCA_NEW <- cbind(whitewineX_V3, WineData_v2Z_PCA$scores)
```

Figure 29: PCA technique code snippet

Using K-means to analyse a dataset



```
#KMEANS using a new Dataset#
kms_new_pca <-kmeans(WineData_v2Z_PCA_NEW[13:23], 4)
```

Figure 30: Using K-means to analyse a PCA-based dataset

New K-means have been developed

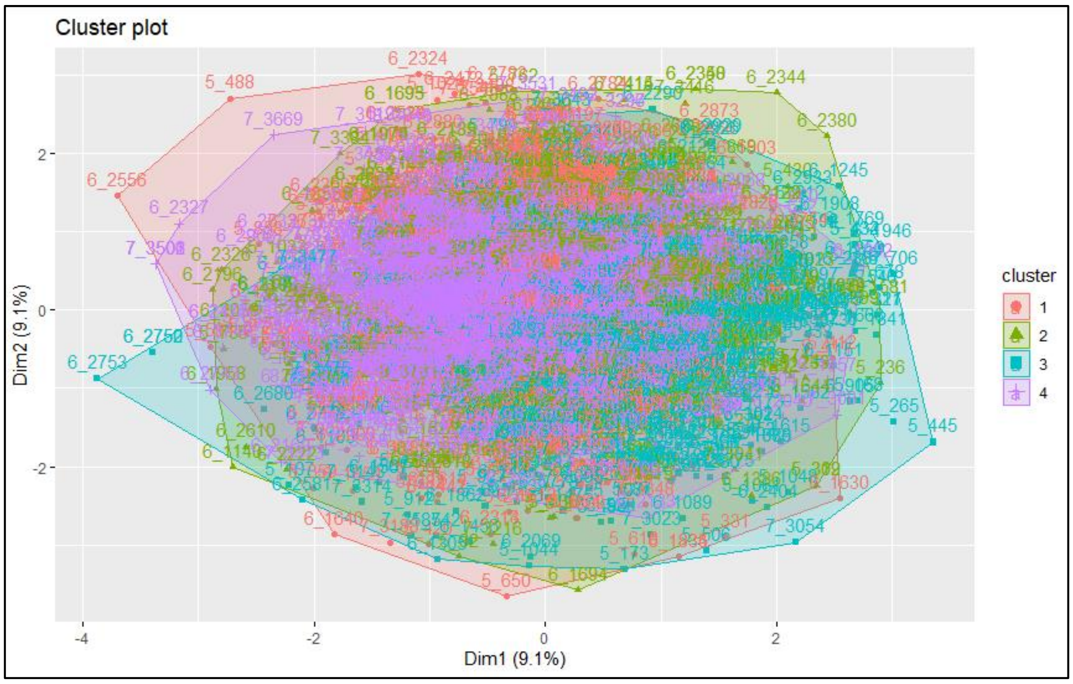


Figure 31: Final K-means Output

Between Sum of Squares(BSS) = 15421.50

Total Sum of Squares (TSS) = 46404.25

$\frac{(BSS)}{(TSS)}$ = 0.3323 //

Objective 2: Multilayer Perceptron (MLP)

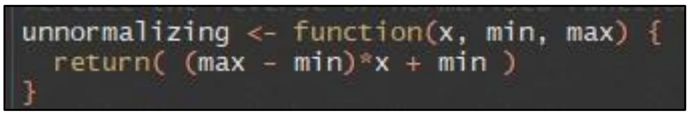
A multilayer perceptron is employed in this case. MLP stands for multilayer perceptron (ANN). An input layer, a hidden layer, and an output layer are the three layers of nodes in the simplest MLP.

Forecasting electricity load has been identified as a critical concern in the implementation of smart energy systems. All smart grid participants can utilise load forecasting to achieve their objectives. Load forecasting is used by users for consumption planning and scheduling, while grid operators use it to ensure safe and secure energy supply. Load forecasting can be classified as succinct term (VSTLF) ranging from a few minutes to an hour, short-term (STLF) ranging from an hour to a week, standard size (MTLF) ranging from a week to a year, and longer - term load forecasting (LTLF) for predictions more than a year ahead of time.

LTLS,MTLF, and STLF might all be utilised to answer the question.

For predicting difficulties, several methods such as auto-regressive (AR), autoregressive-exogenous (ARX), and auto - regressive exogenous (NARX) can be used. Auto-regression (AR) is a time series model that uses observations from previous time steps as input to a regression equation to predict the value at the next time step. The model for the AR technique must be built with the year with the input and the 11th hour value as the output. The ARX model (Auto - regressive with Additional Inputting) is distinguished from the AR model by the presence of an input component. This input term is the exogenous variable, and ARX stands for Auto - regressive with Exogenous Variables. In the ARX technique, we may also feed the model 9th and 10th hour values. NARX stands for "nonlinear ARX."

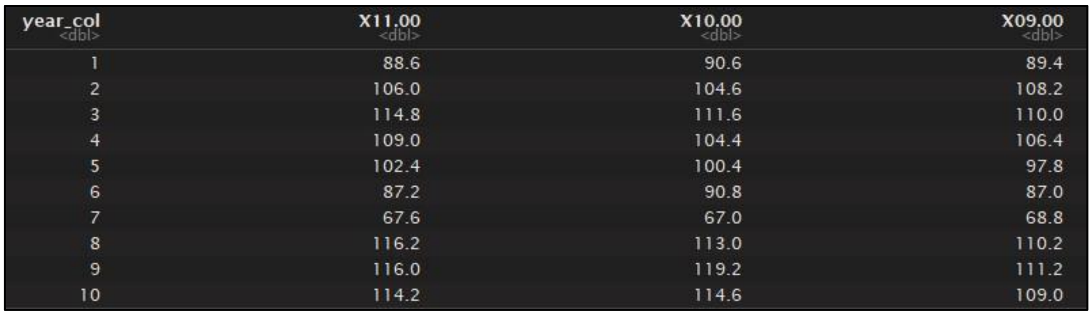
I was using the min max normalisation to normalise the data. The work of normalisation is vital. If the data is not normalised, variables with a greater scale will dominate the machine learning experience, reducing model efficiency. As a result, it is crucial to normalise the data.



```
unnormalizing <- function(x, min, max) {
  return( (max - min)*x + min )
}
```

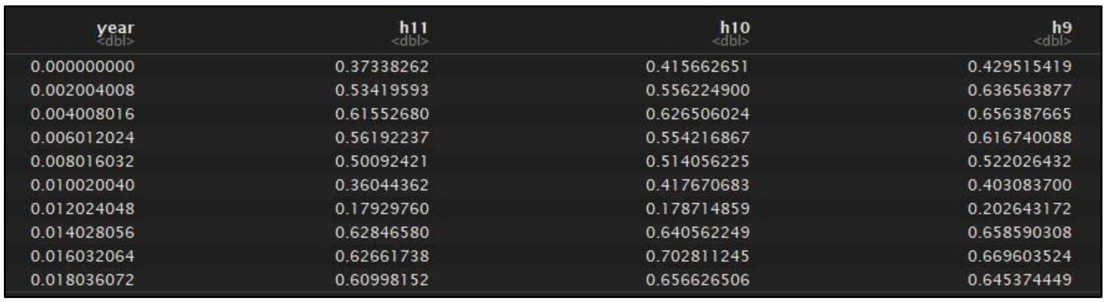
Figure 32: Normalization

The following process is used to normalise the model's input data, and the pictures below demonstrate just how well the dataset seems before and after normalisation.



year_col <dbl>	X11.00 <dbl>	X10.00 <dbl>	X09.00 <dbl>
1	88.6	90.6	89.4
2	106.0	104.6	108.2
3	114.8	111.6	110.0
4	109.0	104.4	106.4
5	102.4	100.4	97.8
6	87.2	90.8	87.0
7	67.6	67.0	68.8
8	116.2	113.0	110.2
9	116.0	119.2	111.2
10	114.2	114.6	109.0

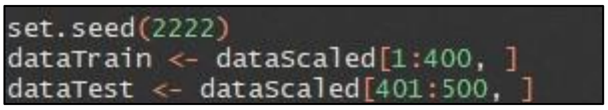
Figure 33: Before the Normalization



year <dbl>	h11 <dbl>	h10 <dbl>	h9 <dbl>
0.000000000	0.37338262	0.415662651	0.429515419
0.002004008	0.53419593	0.556224900	0.636563877
0.004008016	0.61552680	0.626506024	0.656387665
0.006012024	0.56192237	0.554216867	0.616740088
0.008016032	0.50092421	0.514056225	0.522026432
0.010020040	0.36044362	0.417670683	0.403083700
0.012024048	0.17929760	0.178714859	0.202643172
0.014028056	0.62846580	0.640562249	0.658590308
0.016032064	0.62661738	0.702811245	0.669603524
0.018036072	0.60998152	0.656626506	0.645374449

Figure 34: After the Normalization

The dataset is separated into training and testing after the normalisation procedure, as indicated in the graphic below.



```
set.seed(2222)
dataTrain <- dataScaled[1:400, ]
dataTest <- dataScaled[401:500, ]
```

Figure 35: Normalisation procedure

Multilayer Perceptron (MLP) Structure

There are three primary pieces to an MLP. The input layer, hidden layers, and output layer are the three layers. A visual depiction of an MLP is shown in the figure below.

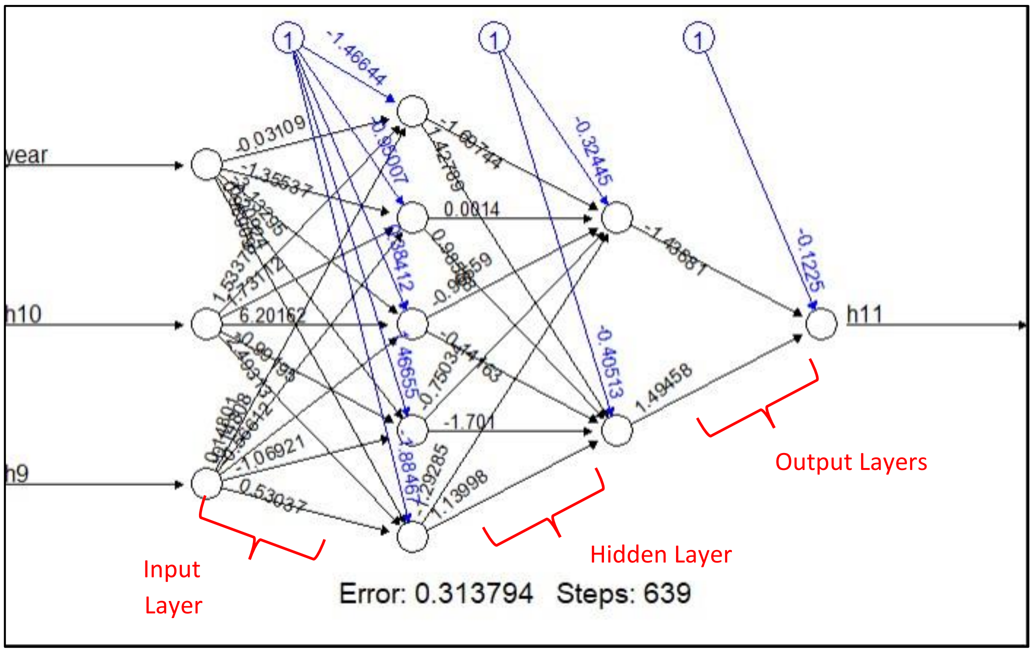


Figure 36: Multilayer Perceptron (MLP) Structure

Input Layer

In our example, the important data for forecasting the output is included in the year, 10th, and 9th hour values. When entering these numbers into the model, they must be normalised.

Hidden Layer

Hidden layers are the layers that exist between the input and output layers. They use data more effectively and adjust to increase the model's predictive accuracy. Neural Networks are said to be always efficient if the hidden layers are 2/3 the length of the input layer plus the output layer. As a result, this scenario employs four concealed layers. Various sizes of hidden layers were used to calculate the prediction levels, and the most effective one was chosen. An examination of several hidden layer values and their forecast outcomes is included in the paper.

Output Layer

This layer will output the full MLP's output. The output of our MLP is the 11th hour result.

The data is then unnormalized to use the following function after receiving the output from the MLP model.

The performance/accuracy of the model will then be measured using the unnormalized values. MAPE(Mean Absolute Percentage Error), MAE(Mean Absolute Error), and RMSE are used to assess the system's correctness and performance (Root Mean Squared Error). The following is a basic explanation of these indexes.

<u>RMSE</u>

The RMSE is the point difference of the errors occur while making a prediction on a dataset. The root of the integer is taken into consideration while determining the model's accuracy, similar to MSE (Mean Squared Error).

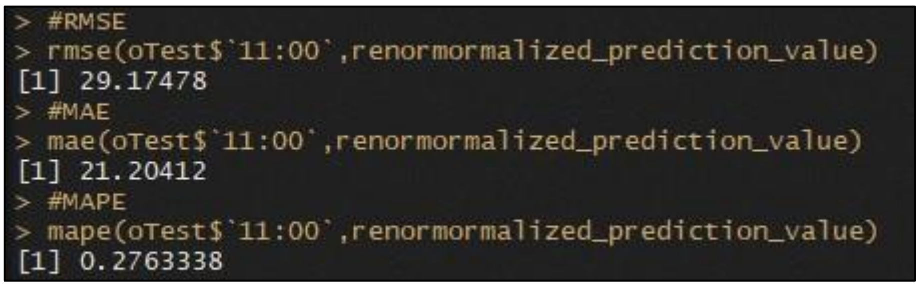
<u>MAE</u>

Absolute error in machine learning defined as the difference between the prediction of an observation and the real value of that observation.

<u>MAPE</u>

MAPE stands for mean or average of expected absolute percentage errors. Error is defined as the difference between the actual or observed value and the predicted value. MAPE is calculated by adding percentage mistakes without regard to sign.

The RMSE, MAE, and MAPE are then computed using the functions following.



```
> #RMSE
> rmse(oTest$`11:00`,renormormalized_prediction_value)
[1] 29.17478
> #MAE
> mae(oTest$`11:00`,renormormalized_prediction_value)
[1] 21.20412
> #MAPE
> mape(oTest$`11:00`,renormormalized_prediction_value)
[1] 0.2763338
```

Figure 37: RMSE, MAE, and MAPE

Various AR MLP Approaches Testing

Table 1: Various AR MLP Approaches Testing Table

Approaches No.	Nodes in hidden Layers	Total Number of Nodes	Learning Rate	Input vectors	Output vectors	Activation function	RMSE	MAE	MAPE
No.1	5 -> 3	8	0.01	Year	11nth hour	Logistic	30.1	23.7	0.26
No.2	10 -> 5	15	0.01	Year	11nth hour	Logistic	29.1	21.2	0.27
No.3	20->10 -> 5	35	0.01	Year	11nth hour	Logistic	32.2	22.2	0.29
No.4	10 -> 5	15	0.0001	Year	11nth hour	Logistic	29.1	21.2	0.27
No.5	10 -> 5	15	0.01	Year	11nth hour	Tanh	28.3	20.3	0.26
No.6	128->9	137	0.01	Year	11nth hour	Logistic	30.1	21.7	0.27
No.7	20->10 -> 5	35	0.01	Year	11nth hour	Tanh	30.4	21.6	0.28

Various NARX MLP Approaches Testing

Table 2: Various NARX MLP Approaches Testing Table

Approaches No.	Nodes in Hidden Layers	Total Number of Nodes	Learning Rate	Input Vectors	Output Vectors	Activation Function	RMSE	MSE	MAPE
No.1	10->5	15	0.01	Year+10nth hour+9nth hour	11nth hour	Logistic	5.1	3.9	0.04
No.2	20>10 -> 5	35	0.01	Year+10nth hour+9nth hour	11nth hour	Logistic	4.7	3.5	0.03
No.3	64->3	67	0.01	Year+10nth hour+9nth hour	11nth hour	Logistic	4.7	3.5	0.03
No.4	16->3	19	0.01	Year+10nth hour+9nth hour	11nth hour	Logistic	4.8	3.6	0.03

We have two models in the NARX MLP methods with the same highest accuracy but distinct topologies. Let's examine which of the two is the most efficient. The first model has (20->10->5) nodes for a total of (35) nodes, while the second model has (64->3) nodes for total of (67) nodes. Let's figure out how many parameters each model has in total.

Because both models have three inputs and one output, the 1st model has [298 ((2*20)+(20*10)+(10*5)+(5*1)+3] parameters, whereas the 2nd model has [389 ((3*64)+(64*3)+(3*1)+2] parameters. A Although the 1st model has three hidden layers and the 2nd model has two layers with the same accuracy, the second model has around 90 more components than the first. As a result, we may regard the 1st model to be the best in this scenario since it is the most productive.

The most efficient AR model

With a (0.01) training rate and Tanh activation function, the most efficient AR model has two hidden layers having 10 nodes during first hidden layer and 5 nodes in the second hidden layer. In comparison to the other techniques, it had the best RMSE and MAE.

The RMSE of this model was (28.3) and the MAE was (20.3).

The construction of the MLP and its values are shown in the picture below.

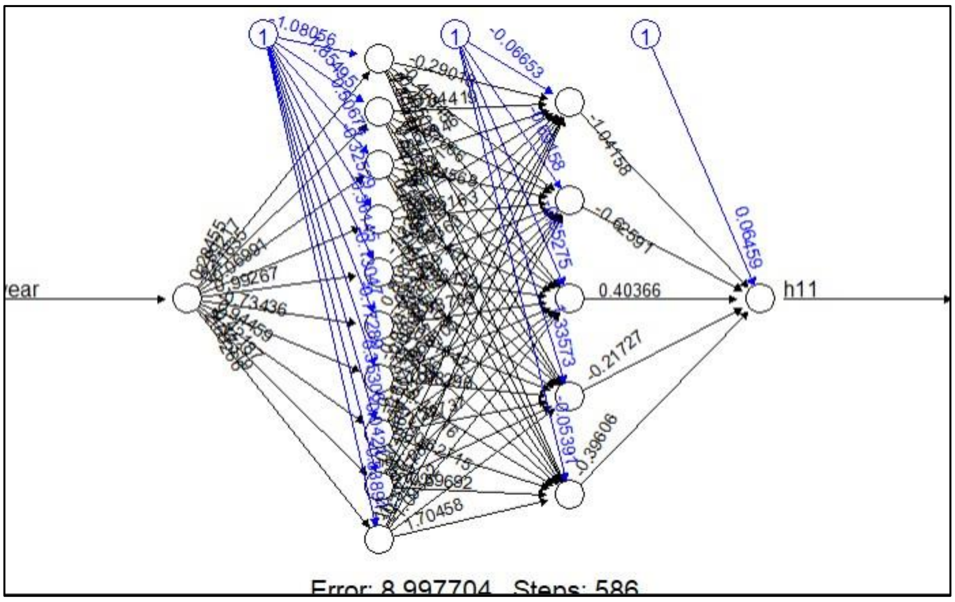


Figure 38: Construction of the MLP and the values (AR Model)

The predicted and actual results for the test set to use this model are shown in the figure below.

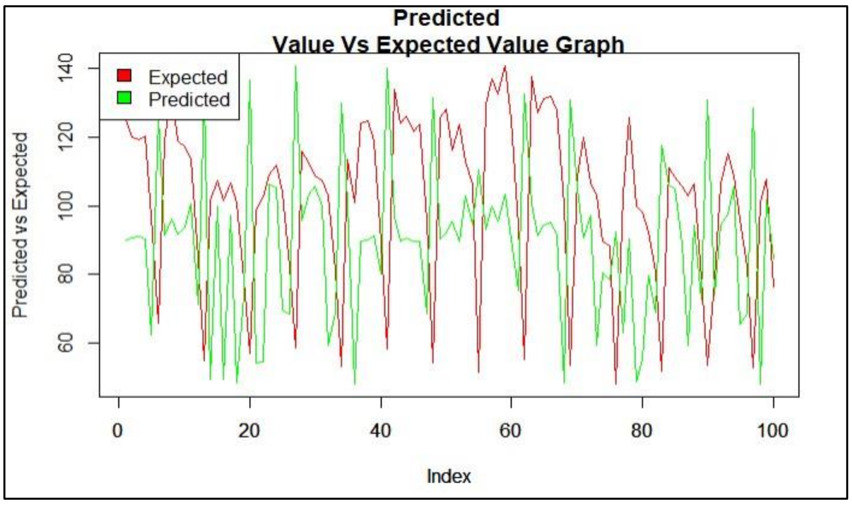


Figure 39: Predicted and Actual results (AR Model)

The most efficient NARX model

The efficient NARX model has three hidden layers: (20) nodes in the first, (10) nodes in the second, and (5) nodes in the third hidden layer, all with a 0.01 training rate and Tanh activation function. When compared to all other ways, two approaches had the best RMSE and MAE, however as previously indicated, we will choose the most efficient of the two.

The RMSE of this model was (4.7), while the MAE was (3.5).

The construction of the MLP and its values are shown in the picture below.

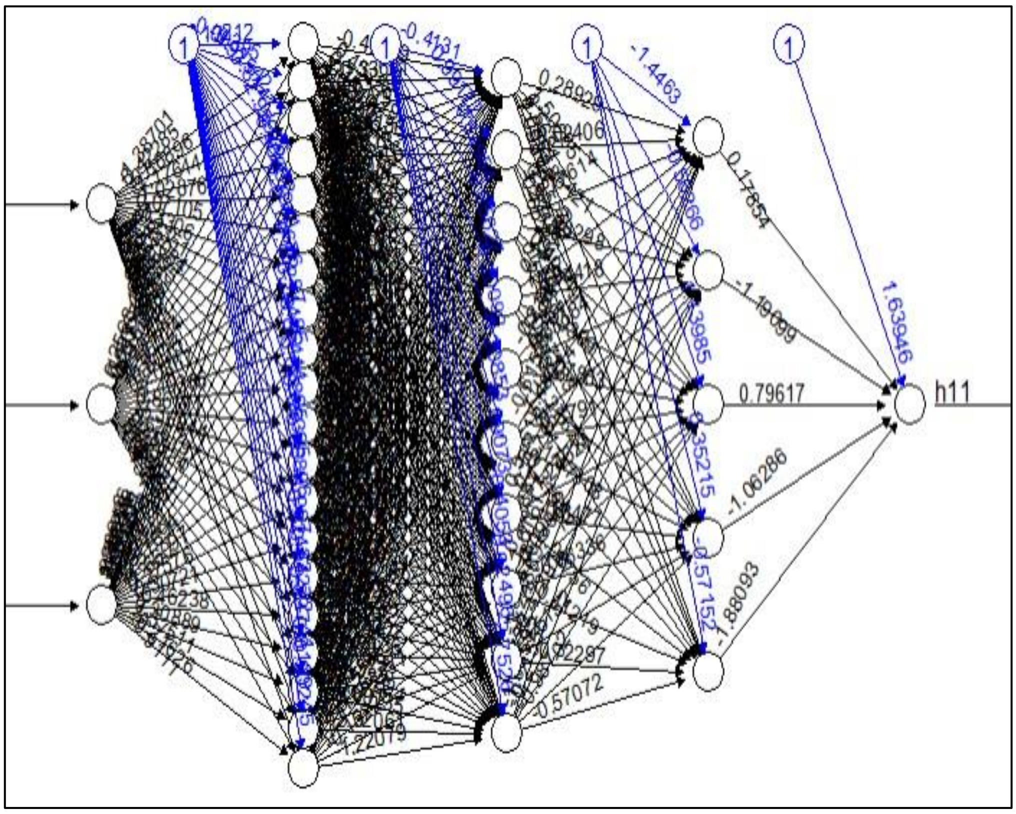


Figure 40: Construction of the MLP and the values (NARX Model)

The predicted and actual results for the test set to use this model are shown in the figure below.

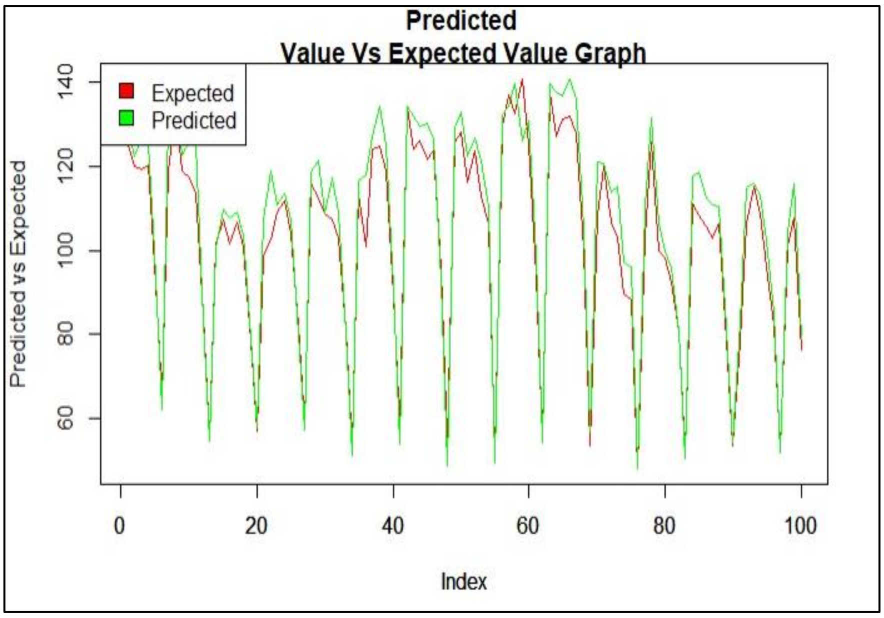


Figure 41: Predicted and Actual results (NARX Model)

Appendix

Objective 1: Clustering

```
#Name:  Rukshan Fernando                                    .

#UoWNO: 18098217                                     .

#StID:  2019784


#Creating a Work Directory#

setwd("c:/Users/Rukshan         Fernando/OneDrive/Desktop/Level_5/2ndSem/5DATA001C.2
Machine Learning and Data Mining/Assesment")


#libraries has to be import#

library(readxl)

library(factoextra)

library(NbClust)

library(gmodels)


#Excel data should be read#

whitewineX <- read_xlsx("w1809821_p1_Whitewine_v2.xlsx")


#Removing rows with missing or incomplete values#

sum(is.na(whitewineX))

whitewineX_V1 <- na.omit(whitewineX)


#Looking for outliers#

boxplot(whitewineX_V1$`fixed acidity`, plot=FALSE)$out


#Detected outliers are saved to a variable#
```

```
outliers <- boxplot(whitewineX_V1$`fixed acidity`, plot=FALSE)$out

#Removing outliers from the new data and allocating it to a new data frame#
whitewineX_V2 <- whitewineX_V1[-which(whitewineX_V1$`fixed acidity` %in% outliers),]



#repeat the procedure for each column#
boxplot(whitewineX_V2$`volatile acidity`, plot=FALSE)$out
outliers <- boxplot(whitewineX_V2$`volatile acidity`, plot=FALSE)$out
whitewineX_V2 <- whitewineX_V2[-which(whitewineX_V2$`volatile acidity` %in% outliers),]

boxplot(whitewineX_V2$`citric acid`, plot=FALSE)$out
outliers <- boxplot(whitewineX_V2$`citric acid`, plot=FALSE)$out
whitewineX_V2 <- whitewineX_V2[-which(whitewineX_V2$`citric acid` %in% outliers),]

boxplot(whitewineX_V2$`residual sugar`, plot=FALSE)$out
outliers <- boxplot(whitewineX_V2$`residual sugar`, plot=FALSE)$out
whitewineX_V2 <- whitewineX_V2[-which(whitewineX_V2$`residual sugar` %in% outliers),]

boxplot(whitewineX_V2$chlorides , plot=FALSE)$out
outliers <- boxplot(whitewineX_V2$chlorides , plot=FALSE)$out
whitewineX_V2 <- whitewineX_V2[-which(whitewineX_V2$chlorides  %in% outliers),]

boxplot(whitewineX_V2$`free sulfur dioxide` , plot=FALSE)$out
outliers <- boxplot(whitewineX_V2$`free sulfur dioxide` , plot=FALSE)$out
whitewineX_V2 <- whitewineX_V2[-which(whitewineX_V2$`free sulfur dioxide` %in%
outliers),]
```

```
boxplot(whitewineX_V2$`total sulfur dioxide` , plot=FALSE)$out

outliers <- boxplot(whitewineX_V2$`total sulfur dioxide` , plot=FALSE)$out

whitewineX_V2 <- whitewineX_V2[-which(whitewineX_V2$`total sulfur dioxide`  %in%
outliers),]


boxplot(whitewineX_V2$pH , plot=FALSE)$out

outliers <- boxplot(whitewineX_V2$pH , plot=FALSE)$out

whitewineX_V2 <- whitewineX_V2[-which(whitewineX_V2$pH  %in% outliers),]


boxplot(whitewineX_V2$sulphates , plot=FALSE)$out

outliers <- boxplot(whitewineX_V2$sulphates , plot=FALSE)$out

whitewineX_V2 <- whitewineX_V2[-which(whitewineX_V2$sulphates  %in% outliers),]


#Outliers have been removed#



#Scaling the data#

#Standardization of Z-scores#

whitewineX_V2ZScor <- scale(whitewineX_V2)

whitewineX_V2ZScor_Input <- subset(whitewineX_V2ZScor, select = -quality)

whitewineX_V2ZScor_Output <- subset(whitewineX_V2, select = quality)

boxplot(whitewineX_V2ZScor_Input)

whitewineX_V3 <- cbind(whitewineX_V2ZScor_Input, whitewineX_V2ZScor_Output)


#Calculating the number of clusters#

#part of Elbow_Plot#

library(factoextra)
```

```
fviz_nbclust(whitewineX_V2ZScor_Input, kmeans, method="wss") +
  labs(subtitle = "Elbow Method")

#Method of Silhouette#
fviz_nbclust(whitewineX_V2ZScor_Input, kmeans, method="silhouette") +
  labs(subtitle = "Silhouette Method")

#Part of NBclust#
NbClust(whitewineX_V2ZScor_Input, distance = "euclidean", min.nc = 2, max.nc = 10,
        method = "kmeans", index = "all",)

#Part of the Kmeans Clustering#

kms_Out <- kmeans(whitewineX_V2ZScor_Input, 4)
print(kms_Out$betweenss)
print(kms_Out$totss)

#Make the clusters visible#
kms_Cluster <- kms_Out$cluster
rownames(whitewineX_V2ZScor)                    <-                    paste(whitewineX_V2$quality,
1:dim(whitewineX_V2)[1], sep = "_")
fviz_cluster(list(data = whitewineX_V2ZScor_Input, cluster = kms_Cluster))

#Part of the Confusion Matrix#
valueExpected <- factor(whitewineX_V2$quality)
valuePredicted <- factor(kms_Out$cluster)

CrossTable(valueExpected, valuePredicted)
```

```
#Part of the PCA Method#

WineData_v2Z_PCA = princomp(whitewineX_V2ZScor)

summary(WineData_v2Z_PCA)

plot(WineData_v2Z_PCA)

screeplot(WineData_v2Z_PCA, type = "lines", main = "PCA")

biplot(WineData_v2Z_PCA)



WineData_v2Z_PCA_NEW <- cbind(whitewineX_V3, WineData_v2Z_PCA$scores)


#KMEANS using a new Dataset#

kms_new_pca <-kmeans(WineData_v2Z_PCA_NEW[13:23], 4)

fviz_cluster(kms_new_pca, WineData_v2Z_PCA_NEW[13:23], cluster = kms_new_pca$cluster)


kms_new_pca$betweenss

kms_new_pca$tot.withinss

kms_new_pca$totss
```

Objective 2: Multilayer Perceptron (MLP)

~~~
---

title: "R Notebook"

output: html_notebook

---



```{r}

#Name:  Rukshan Fernando                                  .

#UoWNO: 18098217                                   .

#StID:  2019784

~~~

```{r}

library(readxl)


UoW_dataX=read_excel("c:/Users/Rukshan Fernando/OneDrive/Desktop/Level_5/2ndSem/5DATA001C.2 Machine Learning and Data Mining/Assesment/w1809821_p2_UoW_load.xlsx")


UoW_dataX

```

```{r}

x <- UoW_dataX[,4] x normalization <- function(x)


  {  return ((x - min(x)) / (max(x) - min(x)))

}


str(UoW_dataX)

```
```
```

```{r}

UoW_dataX columnOfYear<-factor(UoW_dataX$Dates)

columnOfYear<-as.numeric(columnOfYear) columnOfYear

UoW_data_frame<-
UoW_dataX.frame(columnOfYear,UoW_dataX['11:00'],UoW_dataX['10:00'],UoW_dataX['09:
00'])

UoW_data_frame names(UoW_data_frame)[1] <- "year"

names(UoW_data_frame)[2] <- "h11"

names(UoW_data_frame)[3] <- "h10"

names(UoW_data_frame)[4] <- "h9"

str(UoW_data_frame)

UoW_data_Scaled <- as.UoW_dataX.frame(lapply(UoW_data_frame, normalization))

UoW_data_Scaled
```

```{r}

#Training a NN model library("neuralnet")
```

set.seed(2222)

UoW_data_Train <- UoW_data_Scaled[1:400,] UoW_data_Test <- UoW_data_Scaled[401:500,]

#nn <- neuralnet(h11 ~ year,hidden=c(10,5) , UoW_dataX = UoW_data_Train ,linear.output=TRUE)

#nn <- neuralnet(h11 ~ year,hidden=c(128,9) , UoW_dataX = UoW_data_Train ,linear.output=TRUE)

#best

#nn <- neuralnet(h11 ~ year,hidden=c(64,3) , UoW_dataX = UoW_data_Train ,linear.output=TRUE)

#also best

nn <- neuralnet(h11 ~ year,hidden=c(5,2) , UoW_dataX = UoW_data_Train ,linear.output=TRUE, learningrate = 0.01) plot(nn)

```{r}

UoW_data_Test[2] ResultOfModel <- predict(nn, newdata = UoW_data_Test[2])

#ResultOfModel

#model1Result1 = compute(nn, UoW_data_Test[1])

#model1Result1

```

~~~
```{r}
#Obtaining the original (non-normalized) training and testing the required Output

TrainOutput <- x[1:400,"11:00"]

#The 400 initial rows

TestOutput <- x[401:500,"11:00"]

#Find the maximum and least values in the remaining rows.

min_1 <- min(TrainOutput)

max_1 <- max(TrainOutput)

#Present its contents

#Create a renormalized version of a normalised function.

no_normalizing <- function(x, min, max) {

  return( (max - min)*x + min )
}

renormormalized_prediction_value <- no_normalizing(ResultOfModel, min_1, max_1)

#renormormalized_prediction_value
```
~~~

~~~
```{r}

UoW_dataX

~~~

```{r}

library(Metrics)


#The RMSE


rmse(TestOutput$`11:00`,renormormalized_prediction_value)


#The MAE


mae(TestOutput$`11:00`,renormormalized_prediction_value)


#The MAPE


mape(TestOutput$`11:00`,renormormalized_prediction_value)


#Investigate the relationship between expected and actual values


cor(renormormalized_prediction_value,TestOutput$`11:00`)


#Plot for exchange_model

```

```{r}

par(row_mf=c(1,1))
```

plot(TestOutput$`11:00`, renormormalized_prediction_value ,col='green',main='Renormalized Prediction Graph',pch=18,cex=0.7)

abline(0,1,lwd=2)

legend('bottomright',legend='NN', pch=18,col='green', bty='n') fin_result1_1 <- cbind(TestOutput, renormormalized_prediction_value)

fin_result1_1 plot(TestOutput$`11:00` , ylab = "Predicted vs Expected", type="l", col="red")

par(new=TRUE)

plot(renormormalized_prediction_value, ylab = " ", yaxt="n", type="l", col="green" ,main='Predicted Value Vs Expected Value Graph')

legend("topleft", c("Expected","Predicted"), fill=c("red","green"))

```{r}

#in_nnnarx <- neuralnet(h11 ~ year+h10+h9,hidden=c(64,32,3), UoW_dataX = UoW_data_Train, linear.output=TRUE)

in_nnnarx <- neuralnet(h11 ~ year+h10+h9,hidden=c(5,2),UoW_dataX = UoW_data_Train ,linear.output=TRUE)

plot(in_nnnarx)
```

```{r}

UoW_data_Test[2]


result_nnnarx_model1 <- predict(in_nnnarx, newdata = UoW_data_Test[2:4])


#result_nnnarx_model1 TestOutput
```

```{r}

nnnarx_renorm_pre_val <- no_normalizing(result_nnnarx_model1, min_1, max_1)


library("Metrics") rmse(TestOutput$`11:00`,nnnarx_renorm_pre_val)


# The MSE


mae(TestOutput$`11:00`,nnnarx_renorm_pre_val)


#The MAPE


mape(TestOutput$`11:00`,nnnarx_renorm_pre_val)


#Investigate the relationship between expected and actual values


cor(TestOutput$`11:00`,nnnarx_renorm_pre_val)


#Exchange model plot
```

~~~
```{r}

par(row_mf=c(1,1))


plot(TestOutput$`11:00`,        nnnarx_renorm_pre_val        ,col='green',main='Renormalized
Prediction Graph',pch=18,cex=0.7)


abline(0,1,lwd=2)


legend('bottomright',legend='NN', pch=18,col='green', bty='n')


fin_result1_1 <- cbind(TestOutput, nnnarx_renorm_pre_val)


fin_result1_1 plot(TestOutput$`11:00` , ylab = "Predicted vs Expected", type="l", col="red" )


par(new=TRUE)


plot(nnnarx_renorm_pre_val, ylab    =         "          ",         yaxt="n",                    type="l",
        col="green" ,main='Predicted


Value Vs Expected Value Graph')


legend("topleft", c("Expected","Predicted"), fill=c("red","green") )
```
~~~

Reference

Alamaniotis, Ikonomopoulos, Tsoukalas, Bourbakis, (2012). Evolutionary multiobjective optimization of kernel-based veryshort-term load forecasting. IEEE Trans Power Syst 27(3):1477–1484 [Accessed April 27 2022].

Ahmed Sallam, A. and Malik, (2018). *Load Forecasting*. 2nd ed. pp.(pp.41-71).

Medium. 2022. *K-means Clustering of Wine Data*. [online] Available at:

https://towardsdatascience.com/k-means-clustering-of-wine-data-95bac074baae
[Accessed 02 May 2022].

Finnstats, (2021). *R-Bloggers*. [Online]

Available at: https://www.r-bloggers.com/2021/12/how-to-use-the-scale-function-in-r/

[Accessed 02 May 2022].

Alamaniotis, Tsoukalas, Bourbakis, (2014) Virtual cost approach: electricity consumption scheduling for smartgrids/cities in price-directed electricity markets. In: Proceedings of the 5th international conference on information,intelligence, systems and applications, IISA 2014, pp 38–43 [Accessed May 12 2022].

Finnstats, (2021). *R-Bloggers*. [Online]

Available at: https://www.r-bloggers.com/2021/09/how-to-remove-outliers-in-r-3/

[Accessed 13 May 2022].

-------------------------------------(END)-------------------------------------