A picture containing font, typography, text, black

Description automatically generated                                                                           A red and black logo

Description automatically generated with low confidence

University of Westminster

**Informatics Institute of Technology**

**Department of Computing**

Machine Learning & Data Mining

5DATA001C

Name : Shashika Maduranga Rupasinghe

IIT Number : 20221143

UOW Number : w1953826

Table of Contents

[Table Of Figures iv](#_Toc165553141)

[Clustering 1](#_Toc165553142)

[1.1 1st Objective 1](#_Toc165553143)

[1.1.1 Pre-Processing 1](#_Toc165553144)

[1.1.2 Number of clusters calculation 4](#_Toc165553145)

[1.1.3 K-means Clustering 8](#_Toc165553146)

[1.1.3 Silhouette Analysis 10](#_Toc165553147)

[1.2 2nd Objective 12](#_Toc165553148)

[1.2.1 Apply PCA to the Dataset 12](#_Toc165553149)

[1.2.2 Find the value of K 15](#_Toc165553150)

[1.2.3 K means Clustering 19](#_Toc165553151)

[1.2.4 Silhouette analysis for K=2 21](#_Toc165553152)

[1.2.5 Calinski-Harabasz Index analysis for K=2 22](#_Toc165553153)

[Neural Network. 24](#_Toc165553154)

[2.1 Type of input variables used in MLP models for exchange rate forecasting. 24](#_Toc165553155)

[2.2 Autoregressive (AR) Approach 24](#_Toc165553156)

[Input-Output Matrices 24](#_Toc165553157)

[2.3 Normalization 27](#_Toc165553158)

[2.4 Various MLP Models 28](#_Toc165553159)

[2.4.1 AR 1 28](#_Toc165553160)

[2.4.2 AR 2 32](#_Toc165553161)

[2.4.3 AR 3 35](#_Toc165553162)

[2.4.4 AR 4 38](#_Toc165553163)

[2.5 Evaluation 41](#_Toc165553164)

[2.5.1 RMSE (Root Mean Squared Error 41](#_Toc165553165)

[2.5.2 MAE (Mean Absolute Error) 41](#_Toc165553166)

[2.5.3 MAPE (Man Absolute Percentage Error) 41](#_Toc165553167)

[2.5.4 SMAPE (Symmetric Mean Absolute Percentage Error) 41](#_Toc165553168)

[2.5.5 Comparison Table of the Models 41](#_Toc165553169)

[2.5.6 Best one hidden layer model and two hidden layers model 42](#_Toc165553170)

[APPENDIX 44](#_Toc165553171)

[Clustering subtask 1 44](#_Toc165553172)

[Clustering subtask 2 50](#_Toc165553173)

[Neural Network 56](#_Toc165553174)

[References 67](#_Toc165553175)

# Table Of Figures

[Figure 1 : Boxplot explenation 1](#_Toc165553082)

[Figure 2: Code for identify outliers and remove them 2](#_Toc165553083)

[Figure 3 : Boxplot before removing outliers 2](#_Toc165553084)

[Figure 4 : Box plot after removing outliers 3](#_Toc165553085)

[Figure 5 : Code for scalling 3](#_Toc165553086)

[Figure 6 : Code for NBClust method 4](#_Toc165553087)

[Figure 7 : NBClust result 4](#_Toc165553088)

[Figure 8 : NBClust plots 5](#_Toc165553089)

[Figure 9 : NBClust plots 5](#_Toc165553090)

[Figure 10: Elbow method result 6](#_Toc165553091)

[Figure 11: Gap statistic method result 7](#_Toc165553092)

[Figure 12: Silhouette Method 8](#_Toc165553093)

[Figure 13: Cluter Centres values 8](#_Toc165553094)

[Figure 14: Cluster Vectors 9](#_Toc165553095)

[Figure 15: BSS over TSS 9](#_Toc165553096)

[Figure 16: WSS and BSS of the clustering 9](#_Toc165553097)

[Figure 17: Cluster plots 10](#_Toc165553098)

[Figure 18: Silhouette Analysis for the clustering 11](#_Toc165553099)

[Figure 19: Cluster Silhouette widths 11](#_Toc165553100)

[Figure 20: PCA performing code 12](#_Toc165553101)

[Figure 21: PCA Result summery 12](#_Toc165553102)

[Figure 22: Eigenvalues output for 11 attributes 13](#_Toc165553103)

[Figure 23: : Eigenvectors output for 11 attributes 13](#_Toc165553104)

[Figure 24: cumulative scores for PCs 13](#_Toc165553105)

[Figure 25: code for finding cumulative score > 85 14](#_Toc165553106)

[Figure 26: cumulative scores > 85 PCs output 14](#_Toc165553107)

[Figure 27: NBClustering result 15](#_Toc165553108)

[Figure 28: NB clustering plots 16](#_Toc165553109)

[Figure 29: NB clustering plots 16](#_Toc165553110)

[Figure 30: Elbow method results for PCA 17](#_Toc165553111)

[Figure 31: Gap Statistics Method result for PCA 18](#_Toc165553112)

[Figure 32: Silhouette Method result for PCA 19](#_Toc165553113)

[Figure 33: Centres for 7 PCAs 19](#_Toc165553114)

[Figure 34: Cluster Vectors for 7 PCAs 20](#_Toc165553115)

[Figure 35: BSS over TSS for 7 PCAs 20](#_Toc165553116)

[Figure 36: WSS and BSS for clustering 20](#_Toc165553117)

[Figure 37: Clustering plot 21](#_Toc165553118)

[Figure 38: Silhouette analysis for clustering results 22](#_Toc165553119)

[Figure 39: Silhouette analysis for each clusters 22](#_Toc165553120)

[Figure 40: Calinski-Harabasz Index analysis code for clustering 23](#_Toc165553121)

[Figure 41: Calinski-Harabasz Index analysis result graph 23](#_Toc165553122)

[Figure 42: 1st matrix 26](#_Toc165553123)

[Figure 43: 2nd matrix 26](#_Toc165553124)

[Figure 44: 3rd matrix 26](#_Toc165553125)

[Figure 45: 4th matrix 26](#_Toc165553126)

[Figure 46: min-max normalization code 27](#_Toc165553127)

[Figure 47: min max normalization result 28](#_Toc165553128)

[Figure 48: model 1 29](#_Toc165553129)

[Figure 49: model 2 30](#_Toc165553130)

[Figure 50: model 3 31](#_Toc165553131)

[Figure 51: model 4 32](#_Toc165553132)

[Figure 52: model 5 33](#_Toc165553133)

[Figure 53: model 6 34](#_Toc165553134)

[Figure 54: model 7 35](#_Toc165553135)

[Figure 55: model 8 36](#_Toc165553136)

[Figure 56: model 9 37](#_Toc165553137)

[Figure 57: model 10 38](#_Toc165553138)

[Figure 58: model 11 39](#_Toc165553139)

[Figure 59: model 12 40](#_Toc165553140)

# Clustering

## 1st Objective

### 1.1.1 Pre-Processing

Given that the White wine dataset contains 2700 observations. There are 12 attributes to define the dataset. The dataset doesn’t contain any missing values. Firstly, the outlier detection process has been done. The Boxplot method was used for the outlier detection because that method is robust and doesn’t rely on any distributional assumptions.

All observations outside the following interval are considered outliers in the boxplot method.



Visualize representation of boxplot:

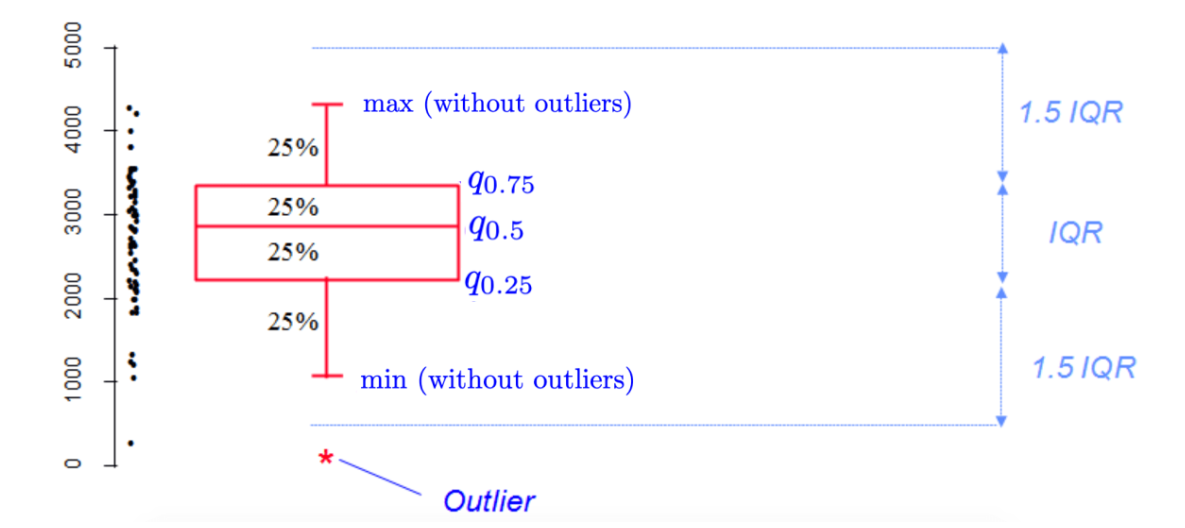


Figure 1 : Boxplot explenation

(Stats and R, 2020)

The below code is used to identify potential outliers using a boxplot and remove them.

A screenshot of a computer program

Description automatically generated

Figure 2: Code for identify outliers and remove them

Visualise boxplot representation of the dataset before removing the outliers:

A graph with text on it

Description automatically generated

Figure 3 : Boxplot before removing outliers

Visualize boxplot representation of the dataset after removing the outliers:

A graph with text on it

Description automatically generated

Figure 4 : Box plot after removing outliers

The boxplot function is used to identify potential outliers in each dataset feature inside a for loop. 490 outlier data rows were detected and removed. The remaining data row count is 2210.

Now, the dataset should be scaled. For that, the Scale () function is used. In this function, z-score normalization is used for scaling. Using that function, only the first 11 features were scaled.

Code for scaling:

A black and white text

Description automatically generated

Figure 5 : Code for scalling

### 1.1.2 Number of clusters calculation

Cluster count can be find in 4 method

1. NBClust Method
2. Elbow Method
3. Gap statistics Method
4. Silhouette Method

#### 1.1.2.1 NBClust Method

It provides 30 indices which determine the number of clusters in a data set and it offers also the best clustering scheme from different results to the user. In addition, it provides a function to perform k-means and hierarchical clustering with different distance measures and aggregation methods. (Charrad, 2014)

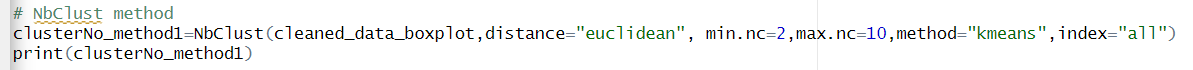
The following code is used to find Clusters using NBClustering.  
  


Figure 6 : Code for NBClust method

Got the following output for the number of clusters.

A screenshot of a computer

Description automatically generated

Figure 7 : NBClust result

A graph of numbers and a number of clusters

Description automatically generatedA graph of numbers and a number of clusters

Description automatically generated

Figure 8 : NBClust plots

Figure 9 : NBClust plots

These results determine, according to the NBClustering method, 2 clusters are better for this.

#### 1.1.2.2 Elbow Method

The elbow method was used to specify the number of clusters on a set of data. The graph obtained the Within-Cluster-Sum of Squared Errors (WSS) calculation. The cluster number is calculated by looking at the point position on the “elbow” arm. **(Hestry Humaira, 2020, Determining The Appropiate Cluster Number Using Elbow Method for KMeans Algorithm)**

Below is the graph that represents the Elbow method output

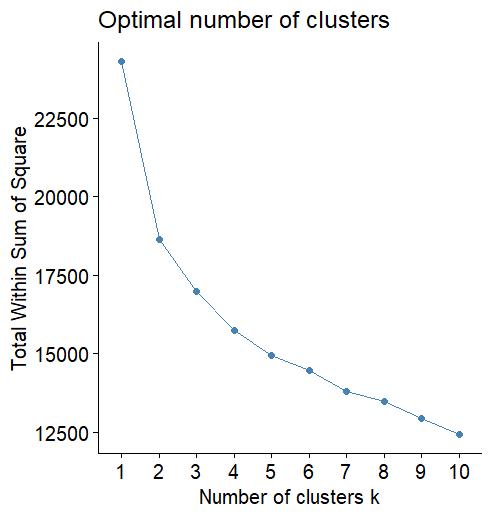


Figure 10: Elbow method result

According to this graph, the elbow position is on 2. So, according to the Elbow method, the optimal clusters are 2.

#### 1.1.2.3 Gap Statistics Method

This will use k mean clustering to determine the number of clustering and then calculate the sum of the distance of all objects from cluster means. Every gap is described as a logarithmic difference between the mean dispersion of the dataset. The gap is maximised when applying the minimum value of k.

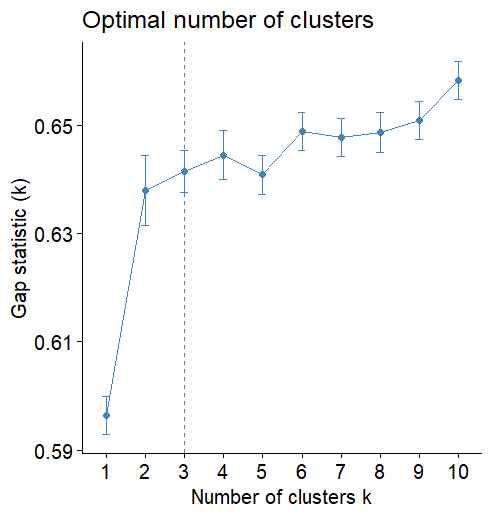


Figure 11: Gap statistic method result

According to the output Gap Statistic graph, best cluster amount is 3.

#### 1.1.2.4 Silhouette Method

This method calculates the separation distance between resulting clusters and display measure of how close each point in one cluster is to point in neighboring clusters. Good clustering indicate by high average silhouette width.

A graph with blue lines and white text

Description automatically generated

Figure 12: Silhouette Method

The output graph shows that the best k value is 2 for the silhouette method.

According to all 4 results K=2 is the cluster count to run k means clustering.

### 1.1.3 K-means Clustering

Using K=2, K-means clustering is performed on the white wine outlier removed, scaled dataset. The clustering result is presented below.  
  
1. Cluster means (Centres)

A close-up of a number

Description automatically generated

Figure 13: Cluter Centres values

2. Cluster Vectors

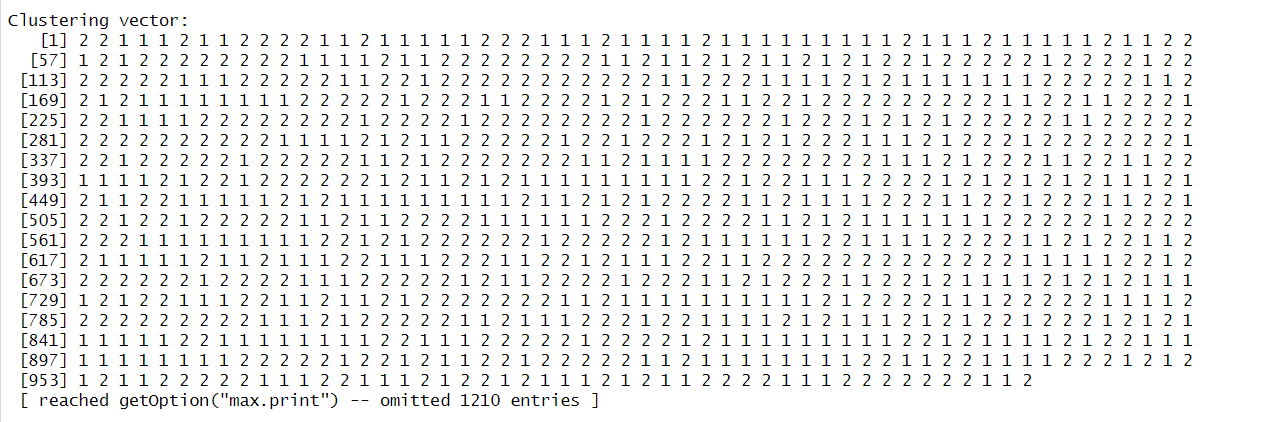


Figure 14: Cluster Vectors

3. Ratio of between\_cluster\_sums\_of\_squares (BSS) over total\_sum\_of\_Squares (TSS)

A black text on a white background

Description automatically generated

Figure 15: BSS over TSS

4. BSS and WSS

BSS (Between-cluster sum of Squares) and WSS (Within-cluster sum of Squares) are used to evaluate the performances of clustering algorithms. Below are the performed clustering BSS and WSS values.

A number with numbers on it

Description automatically generated with medium confidence

Figure 16: WSS and BSS of the clustering

Using this clustering, the dataset is split into two clusters where data points in the same cluster are similar, and data points in different clusters are distinct. So for that, WSS should be smaller, and BSS should be higher.

A graph of a cluster plot

Description automatically generated

Figure 17: Cluster plots

### 1.1.3 Silhouette Analysis

This evaluation method is used to check how well the clustering is performed. Average silhouette value is calculated for each points and ranged from -1 to 1. If the value is negative that indicate maybe value is assigned to a wrong cluster. Positive value indicate it is assigned correctly.

The graph represent the clustered k-means algorithm silhouette plot.

A graph with different colored lines

Description automatically generated

Figure 18: Silhouette Analysis for the clustering

This graph shows that data points are well distributed among two clusters.

A black text on a white background

Description automatically generated

Figure 19: Cluster Silhouette widths

The average silhouette is 0.21, and each cluster's average silhouette width is 0.20 and 0.23, indicating that the data points have been well distributed among the two clusters. But the cluster 1 average is less than 0.21. That indicates 1st cluster is less well-defined.

## 2nd Objective

### 1.2.1 Apply PCA to the Dataset

PCA method is used to reduce the dimension of a dataset while keeping the maximum amount of information. This is an unsupervised technique. The below code snippet is used to perform PCA to the scaled and outlier detected dataset (used the same method of objective 1 to remove outliers and scale dataset).

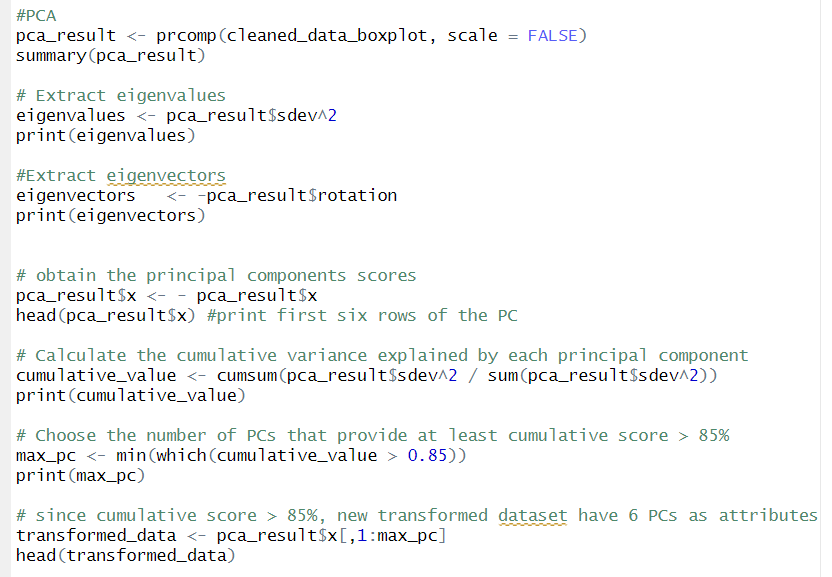


Figure 20: PCA performing code

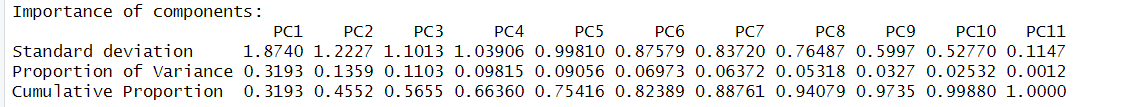


Figure 21: PCA Result summery

1. **Eigenvalues/Eigenvectors**

Eigenvalues represent the total variance a given principal component (PC) can explain. Eigenvectors represent the weight of each eigenvalue.

Eigenvalues can be calculated by squaring the standard deviations.

Eigenvalues output for 11 attributes:  
  


Figure 22: Eigenvalues output for 11 attributes

Eigenvectors output for 11 attributes:

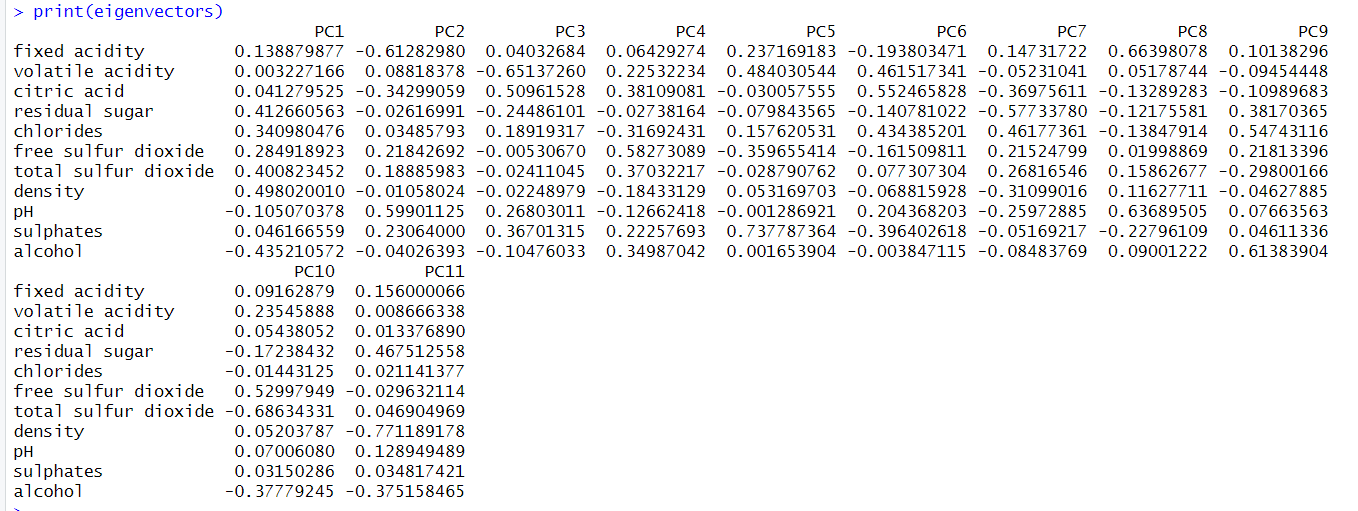


Figure 23: : Eigenvectors output for 11 attributes

1. **Cumulative Score per principal Component (PC)**

The Cumulative Score represents how much of the information is explained by the certain umber of PCs.

For the 11 PCs, the cumulative scores are below.



Figure 24: cumulative scores for PCs

1. **Cumulative Score > 85% PC count**

For the given clustering, Instructions says PCs must be selected up to 85% of the Cumulative Score.

Code block for that and the output is in below

A screenshot of a computer code

Description automatically generated

Figure 25: code for finding cumulative score > 85

A screenshot of a computer

Description automatically generated

Figure 26: cumulative scores > 85 PCs output

According to this, the first 7 PCA represent 85% of cumulative scores. So, the PCA count is 7.

### Find the value of K

As the methods used in the previous step, here the same 4 methods have been used with same codes

**NBClustering**

NBClustering suggests 2 clusters are the best number of clusters for the K-mans algorithm.

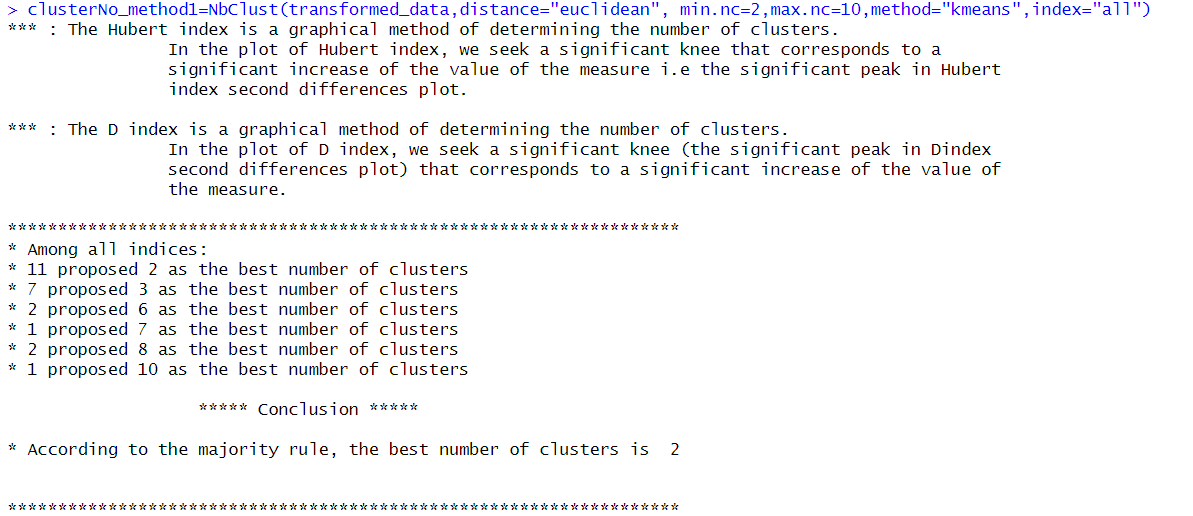


Figure 27: NBClustering result

A graph of numbers and a number of clusters

Description automatically generated

Figure 28: NB clustering plots

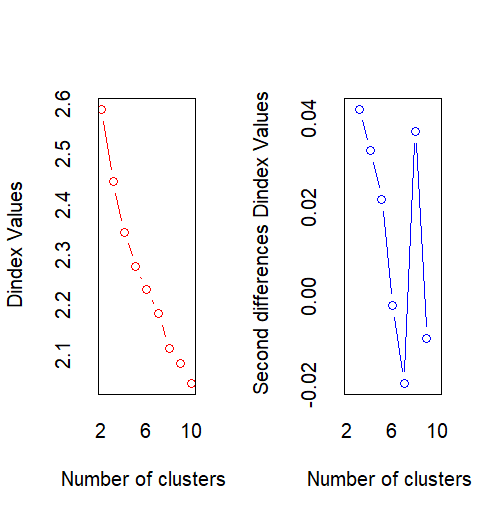


Figure 29: NB clustering plots

**Elbow Method**

The WSS value decreases with the increasing cluster count. The arm elbow shape is gotten in the cluster count 2. So that is the suggested cluster count using the elbow method.

A graph with numbers and lines

Description automatically generated

Figure 30: Elbow method results for PCA

**Gap Statistics Method**

Gap statistics method gave cluster count as 2.

A graph of a number of clusters

Description automatically generated

Figure 31: Gap Statistics Method result for PCA

**Silhouette Method**

Silhouette method gave cluster count as 2.

A graph with a line

Description automatically generated

Figure 32: Silhouette Method result for PCA

According to the outputs the best cluster count is 2.

### K means Clustering

Clustering is performed using the same methods in the previous objective and below results are given by the clustering.

1. Cluster means (Centres)

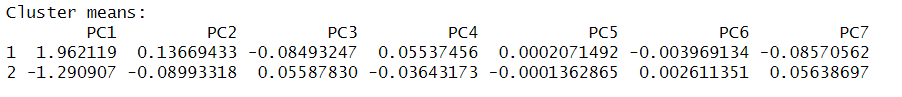


Figure 33: Centres for 7 PCAs

1. Cluster Vectors

A grid of numbers

Description automatically generated

Figure 34: Cluster Vectors for 7 PCAs

1. Ratio of between\_cluster\_sums\_of\_squares (BSS) over total\_sum\_of\_Squares (TSS)

A black text with black letters

Description automatically generated with medium confidence

Figure 35: BSS over TSS for 7 PCAs

1. BSS and WSS

A number with numbers on it

Description automatically generated with medium confidence

Figure 36: WSS and BSS for clustering

In below the visualize representation of the clustering is presented.

A graph with numbers and a diagram

Description automatically generated with medium confidence

Figure 37: Clustering plot

### Silhouette analysis for K=2

As the earlier description, the cluster silhouette plot also shows how well the clustering is happening. The average silhouette width is 0.21, and according to the results below, cluster 1 width is 0.23, and cluster 2 width is 0.20. According to that, cluster 2 width is less than the average length. That indicates there should be a better way to cluster this in more distinct clusters.

A graph with different colored lines

Description automatically generated

Figure 38: Silhouette analysis for clustering results

A black text on a white background

Description automatically generated

Figure 39: Silhouette analysis for each clusters

### Calinski-Harabasz Index analysis for K=2

This index is an internal evaluation for measuring the ratio between cluster dispersion and between cluster dispersion. The higher this value indicates that it is the best clustering value.

The code snippet below is used to calculate the calinski-harabasz index.

A close-up of a computer code

Description automatically generated

Figure 40: Calinski-Harabasz Index analysis code for clustering

The resulting graph is below.

A graph of a number of clusters

Description automatically generated

Figure 41: Calinski-Harabasz Index analysis result graph

This graph indicates that 2 clusters are the best for clustering this dataset. It has the maximum CH value of 783.8244.

# Neural Network.

## Type of input variables used in MLP models for exchange rate forecasting.

Input vectors are crucial for accurately forecasting future exchange rates. There are various methods used in it.

* Lagged Values: This approach uses past observations as input to predict future values. Like, exchange rate from the past few days to predict the exchange rate for tomorrow.
* Economic Indicators: Indicators like GDP, inflation rate, and interest rates can be used to predict future exchange rates.
* Multivariate Time Series: This approach uses multiple related time series of data.

## Autoregressive (AR) Approach

### Input-Output Matrices

The data set includes 500 observations. There are 3 attributes in the dataset, but only the exchange rate is considered when using the AR approach. Input vectors contain the data the data that is going to be fed into the neural network. The autoregressive approach uses past data to predict future data.

If the neural network is predicting the Y[t], Let’s check upto Y[t-4]:

AR1 inputs: Y[t-1]

AR2 inputs: Y[t-1], Y[t-2]

AR3 inputs: Y[t-1], Y[t-2], Y[t-3]

AR4 inputs: Y[t-1], Y[t-2], Y[t-3], Y[t-4]

4 I/O Matrices were created (using time-delayed forecasting). The following figures represent 4 I/O matrices. Also, missing values have been removed from matrices. The first 13 observations of each matrix are presented below.

A screenshot of a graph

Description automatically generatedA screenshot of a table

Description automatically generated A table of numbers with numbers on it

Description automatically generated A screenshot of a graph

Description automatically generated

Figure 42: 1st matrix

Figure 43: 2nd matrix

Figure 44: 3rd matrix

Figure 45: 4th matrix

## Normalization

Normalization can be used to make data more suitable for training a neural network. Normalization is use to make get the data into a same scale. Different attribute data can be in different scale. When training a model, high scale data is giving higher impact and low scale data gives low impact to the results. For make that right, all data should be in the same scale. In here min-max normalization is used to perform the normalization for these data.

Code:

A screen shot of a computer code

Description automatically generated

Figure 46: min-max normalization code

The below output summery indicate that all data has been scale between 0 and 1

A screenshot of a computer

Description automatically generated

Figure 47: min max normalization result

## Various MLP Models

Now lets build different MLP models using these 4 I/O matrices. For this, different layer counts, node counts, linear and nonlinear outputs and different activation functions can be used.

### 2.4.1 AR 1

Let's try 1 input and one output model using the M1 matric.

1. **Model 1**

Layers – 1

Layer 1 nodes – 8

Activation function – logistic

Output – linear

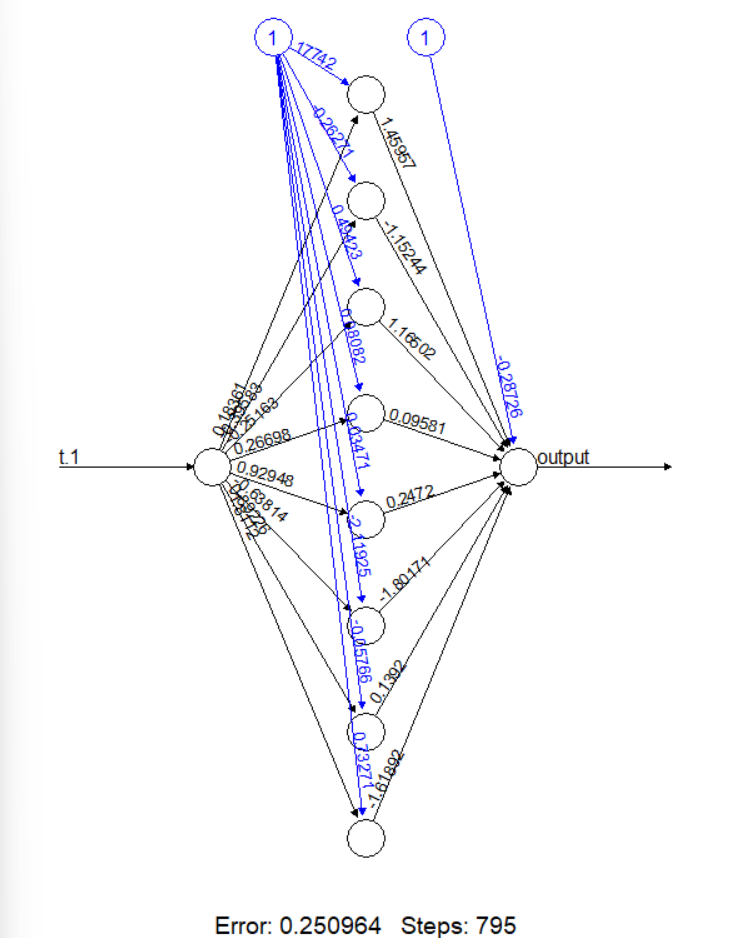


Figure 48: model 1

1. **Model 2**

Layers – 2

Layer 1 nodes – 8

Layer 2 nodes - 5

Activation function – tanh

Output – linear

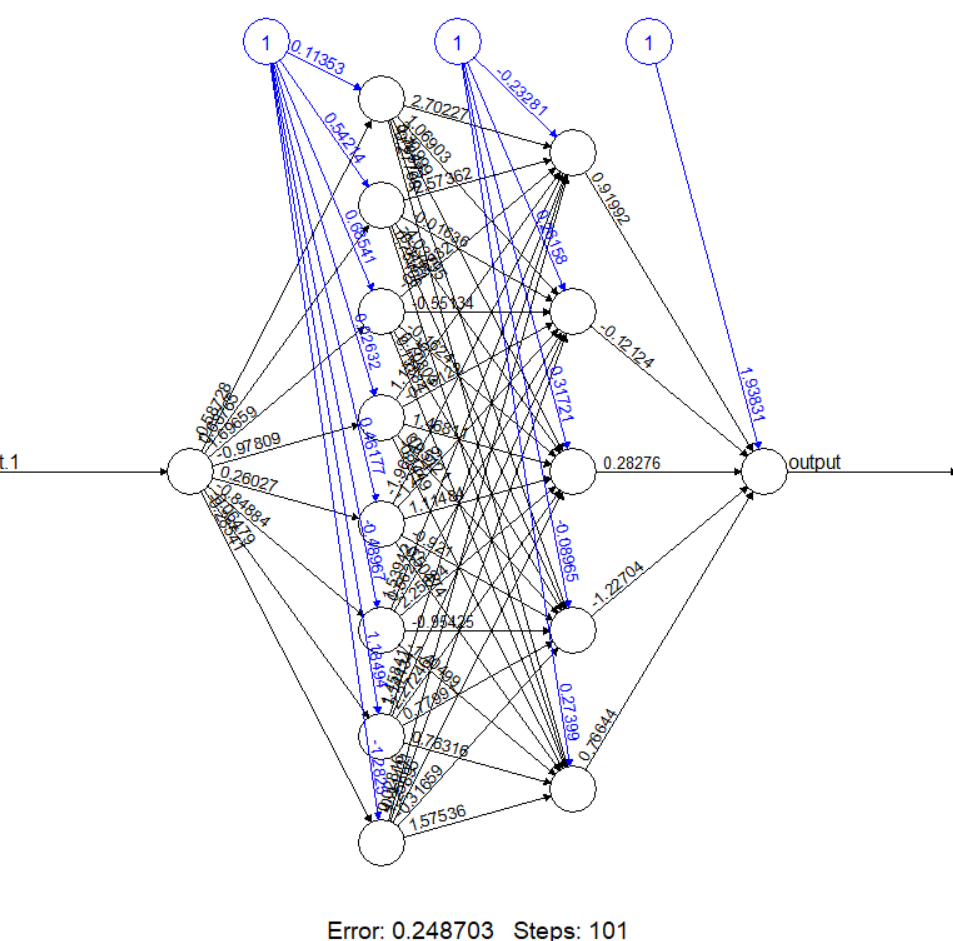
****

Figure 49: model 2

1. **Model 3**

Layers – 2

Layer 1 nodes – 10

Layer 2 nodes – 6

Activation function – logistic

Output – non-linear

A diagram of a network

Description automatically generated

Figure 50: model 3

### 2.4.2 AR 2

Let's try 2 input and one output model using the M2 matric.

1. **Model 4**

Layers – 1

Layer 1 nodes – 12

Activation function – logistic

Output – linear

A diagram of a network

Description automatically generated

Figure 51: model 4

1. **Model 5**

Layers – 2

Layer 1 nodes – 7

Layer 2 nodes - 5

Activation function – tanh

Output – linear

A diagram of a network

Description automatically generated

Figure 52: model 5

1. **Model 6**

Layers – 2

Layer 1 nodes – 12

Layer 2 nodes – 7

Activation function – logistic

Output – non-linear

A diagram of a network

Description automatically generated

Figure 53: model 6

### 2.4.3 AR 3

Let's try 3 input and one output model using the M3 matric.

1. **Model 7**

Layers – 1

Layer 1 nodes – 10

Activation function – logistic

Output – linear

A diagram of a network

Description automatically generated

Figure 54: model 7

1. **Model 8**

Layers – 2

Layer 1 nodes – 10

Layer 2 nodes - 5

Activation function – tanh

Output – linear

A diagram of a network

Description automatically generated

Figure 55: model 8

1. **Model 9**

Layers – 2

Layer 1 nodes – 10

Layer 2 nodes – 8

Activation function – logistic

Output – non-linear

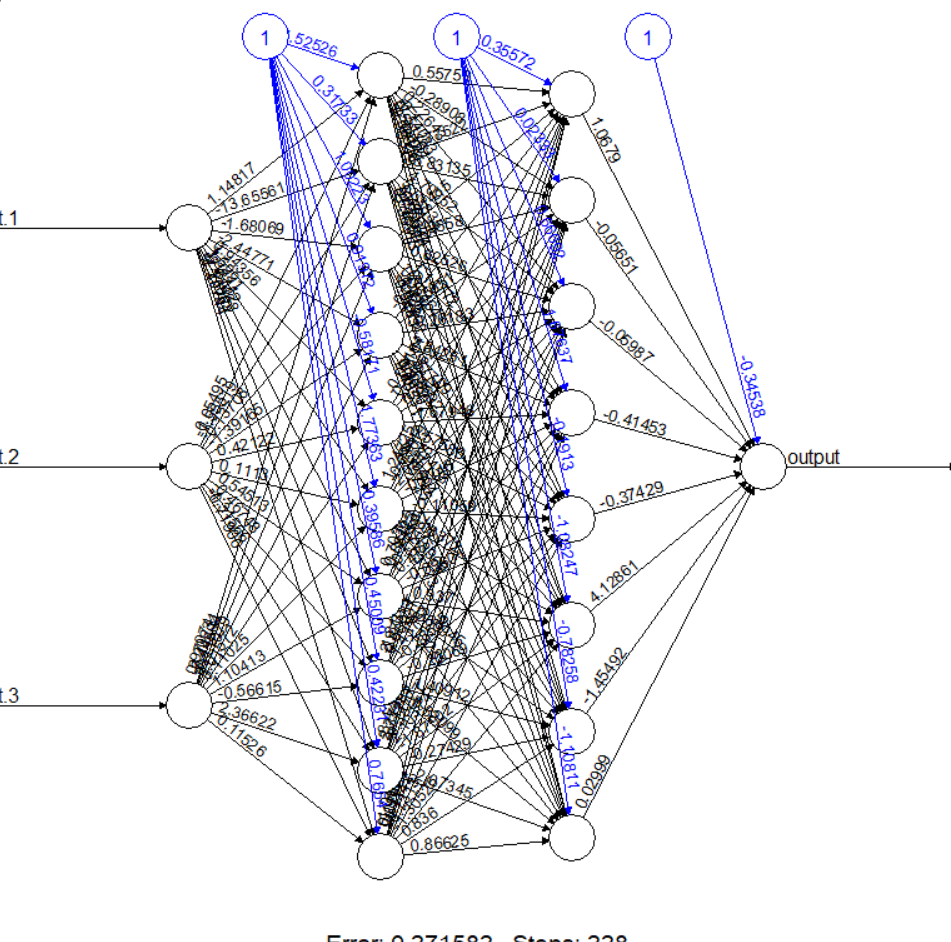


Figure 56: model 9

### 2.4.4 AR 4

Let's try 4 input and one output model using the M4 matric.

1. **Model 10**

Layers – 1

Layer 1 nodes – 12

Activation function – logistic

Output – linear

A diagram of a network

Description automatically generated

Figure 57: model 10

1. **Model 11**

Layers – 2

Layer 1 nodes – 10

Layer 2 nodes - 9

Activation function – tanh

Output – linear

A diagram of a network

Description automatically generated

Figure 58: model 11

1. **Model 12**

Layers – 2

Layer 1 nodes – 12

Layer 2 nodes – 8

Activation function – logistic

Output – non-linear

A diagram of a network

Description automatically generated

Figure 59: model 12

## Evaluation

### 2.5.1 RMSE (Root Mean Squared Error

This indicates the average distance between actual and predicted values. This is calculated using the average square root of the difference between the above values.

### 2.5.2 MAE (Mean Absolute Error)

This measures the average absolute value between predicted and actual values. The average absolute value difference between these results will be calculated for this calculation.

### 2.5.3 MAPE (Man Absolute Percentage Error)

This measures the percentage difference between actual and predicted values. The average of the absolute percentages is used to calculate this.

### 2.5.4 SMAPE (Symmetric Mean Absolute Percentage Error)

This is the same as the MAPE but uses a symmetric percentage difference to predict the value.

### 2.5.5 Comparison Table of the Models

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model number | Input count | Hidden layers and nodes | Activation function | output | RMSE | MAE | MAPE | SMAPE |
| 1 | 1 | 8 | Logistic | Linear | 0.006135154 | 0.004630462 | 0.003506548 | 0.003509670 |
| 2 | 1 | 8,5 | Tanh | Linear | 0.006175844 | 0.004705545 | 0.003562129 | 0.003565892 |
| 3 | 1 | 10,6 | Logistic | Non-linear | 0.006159105 | 0.004597229 | 0.003482128 | 0.003484919 |
| 4 | 2 | 12 | Logistic | Linear | 0.006197981 | 0.004687889 | 0.003545808 | 0.003549774 |
| 5 | 2 | 7,5 | Tanh | Linear | 0.006209826 | 0.004710004 | 0.003563056 | 0.003566789 |
| 6 | 2 | 12,7 | Logistic | Non-linear | 0.006150074 | 0.004566822 | 0.003456428 | 0.003459035 |
| 7 | 3 | 10 | Logistic | Linear | 0.006190405 | 0.004554071 | 0.003446480 | 0.003449122 |
| 8 | 3 | 10,5 | Tanh | Linear | 0.006259427 | 0.004737453 | 0.003583110 | 0.003586897 |
| 9 | 3 | 10,8 | Logistic | Non-linear | 0.006356810 | 0.004662389 | 0.003527903 | 0.003529956 |
| 10 | 4 | 12 | Logistic | Linear | 0.006150482 | 0.004643765 | 0.003512089 | 0.003515317 |
| 11 | 4 | 10,9 | Tanh | Linear | 0.006181717 | 0.004729088 | 0.003574409 | 0.003578125 |
| 12 | 4 | 12,8 | Logistic | Non-linear | 0.006142728 | 0.004576780 | 0.003463124 | 0.003465118 |

### 2.5.6 Best one hidden layer model and two hidden layers model

According to the above table, model 7 is the best for one hidden layer model. Because it has low scores more than any other hidden model. RMSE value indicates it is 1, but the other 3 values indicate it is 7.

The best 2-hidden layer model is 6. RMSE indicates it is 12, but the other 3 values indicate it is 6.

Now, let's check the best efficiency model between the best one hidden layer model and the best two hidden layer models.

For that, let’s calculate the adjusting parameters count for each model.

Model 7:  
input – 3

Nodes – 10

Output – 1

Parameters = (3+1)10 + (10+1)\*1 = 51

Model 6:

Inputs – 2

1st layer nodes – 12

2nd layer nodes – 7

Outputs – 1

Parameters = (2+1)\*12 + (12+1)\*7 + (7+1)\*1 = 135

According to this comparison, the 2 hidden layer model is highly complex. Also, according to the error rate comparison table, 1 hidden layer model (model 7) has better accuracy than model 7 (2 hidden layer model). According to all of these comparisons, model 6 ( 1 hidden layer model) is less complex and has better accuracy. Because of that, model 6 is more suitable for this task.

# APPENDIX

## Clustering subtask 1

library(readxl) #for read xlsx file

library(factoextra) #for determining the relevant number of clusters

library(NbClust)

# Load the dataset

wine\_data <- read\_excel("D:/E driver/IIT/Level\_5\_year\_2/Semester 2/Machine learning/Assessment/Git-implimenatations/Machine-Learning-CW/Partitioning Clustering/sub task 1/Whitewine\_v6.xlsx")

head(wine\_data)

# Extract the first 11 attributes

wine\_features <- wine\_data[, 1:11]

head(wine\_features)

#The structure of data set

str(wine\_data)

#Checking missing values

sum(is.na(wine\_data))

#checking number of rows and columns (before handling outliers)

dim(wine\_data)

#boxplot

#View the outliers using using boxplot

boxplot(wine\_data[1:11])

boxplot(wine\_features)

#Generating box plot for each variable to identify outliers in the data set and remove them

for (vari in 1:11) {

#Detect outliers using boxplot

outliers <- boxplot(wine\_data[[vari]], plot = FALSE)$out

#Remove outliers

wine\_data <- wine\_data[!wine\_data[[vari]] %in% outliers, , drop = FALSE]

}

#Checking outliers removal

boxplot(wine\_data[1:11])

dim(wine\_data)

# Scale the features

scaled\_features <- scale(wine\_features)

#Scaling data

cleaned\_data\_boxplot <- scale(wine\_data[, -c(12)])

dim(cleaned\_data\_boxplot)

#####################additional try#####################################

# Detect outliers using Z-score

z\_scores <- scale(wine\_features)

# Define a threshold for outliers

threshold <- 3

# Find indices of outliers

outlier\_indices <- which(abs(z\_scores) > threshold, arr.ind = TRUE)

# Print outlier rows

outlier\_rows <- wine\_data[outlier\_indices[,1], ]

print(outlier\_rows)

# Remove outliers

cleaned\_data <- wine\_features[-outlier\_indices[,1], ]

# Check dimensions of original dataset

dim(wine\_data)

# Check dimensions of scaled features

dim(scaled\_features)

# Check dimensions of original dataset

dim(wine\_data)

# Check dimensions of cleaned data

dim(cleaned\_data)

#############################################

#Determine the number of cluster centres

set.seed(26)

# NbClust method

clusterNo\_method1=NbClust(cleaned\_data\_boxplot,distance="euclidean", min.nc=2,max.nc=10,method="kmeans",index="all")

print(clusterNo\_method1)

# Elbow Method

fviz\_nbclust(cleaned\_data\_boxplot, kmeans, method = 'wss')

# Gap Statistic method

fviz\_nbclust(cleaned\_data\_boxplot, kmeans, method = 'gap\_stat')

# Silhouette Method

fviz\_nbclust(cleaned\_data\_boxplot, kmeans, method = 'silhouette')

# kmeans clustering K=2

install.packages("cluster")

library(cluster)

k = 2

kmeans\_wine\_2 = kmeans(cleaned\_data\_boxplot, centers = k, nstart = 10)

kmeans\_wine\_2

#Illustration of the clusters

fviz\_cluster(kmeans\_wine\_2, data = cleaned\_data\_boxplot)

# Internal evaluation

wss\_2 = kmeans\_wine\_2$tot.withinss

bss\_2 = kmeans\_wine\_2$betweenss

wss\_2

bss\_2

# kmeans clustering K = 3

k = 3

kmeans\_wines\_3 = kmeans(cleaned\_data\_boxplot, centers = k, nstart = 10)

kmeans\_wines\_3

#Illustration of the clusters

fviz\_cluster(kmeans\_wines\_3, data = cleaned\_data\_boxplot)

# Internal evaluation

wss\_3 = kmeans\_wines\_3$tot.withinss

bss\_3 = kmeans\_wines\_3$betweenss

wss\_3

bss\_3

# The silhouette analysis

sil <- silhouette(kmeans\_wine\_2$cluster, dist(cleaned\_data\_boxplot))

fviz\_silhouette(sil)

# The silhouette analysis

sil <- silhouette(kmeans\_wines\_3$cluster, dist(cleaned\_data\_boxplot))

fviz\_silhouette(sil)

## Clustering subtask 2

library(readxl) #for read xlsx file

library(factoextra) #for determining the relevant number of clusters

library(NbClust)

library(cluster)

# Load the dataset

wine\_data <- read\_excel("D:/E driver/IIT/Level\_5\_year\_2/Semester 2/Machine learning/Assessment/Git-implimenatations/Machine-Learning-CW/Partitioning Clustering/sub task 1/Whitewine\_v6.xlsx")

head(wine\_data)

# Extract the first 11 attributes

wine\_features <- wine\_data[, 1:11]

head(wine\_features)

#The structure of data set

str(wine\_data)

#Checking missing values

sum(is.na(wine\_data))

#checking number of rows and columns (before handling outliers)

dim(wine\_data)

#boxplot

#View the outliers using using boxplot

boxplot(wine\_data[1:11])

boxplot(wine\_features)

#Generating box plot for each variable to identify outliers in the data set and remove them

for (vari in 1:11) {

#Detect outliers using boxplot

outliers <- boxplot(wine\_data[[vari]], plot = FALSE)$out

#Remove outliers

wine\_data <- wine\_data[!wine\_data[[vari]] %in% outliers, , drop = FALSE]

}

#Checking outliers removal

boxplot(wine\_data[1:11])

dim(wine\_data)

# Scale the features

scaled\_features <- scale(wine\_features)

#Scaling data

cleaned\_data\_boxplot <- scale(wine\_data[, -c(12)])

dim(cleaned\_data\_boxplot)

# Scale the features

scaled\_features <- scale(wine\_features)

#Scaling data

cleaned\_data\_boxplot <- scale(wine\_data[, -c(12)])

#PCA

pca\_result <- prcomp(cleaned\_data\_boxplot, scale = FALSE)

summary(pca\_result)

# Extract eigenvalues

eigenvalues <- pca\_result$sdev^2

print(eigenvalues)

#Extract eigenvectors

eigenvectors <- -pca\_result$rotation

print(eigenvectors)

# obtain the principal components scores

pca\_result$x <- - pca\_result$x

head(pca\_result$x) #print first six rows of the PC

# Calculate the cumulative variance explained by each principal component

cumulative\_value <- cumsum(pca\_result$sdev^2 / sum(pca\_result$sdev^2))

print(cumulative\_value)

# Choose the number of PCs that provide at least cumulative score > 85%

max\_pc <- min(which(cumulative\_value > 0.85))

print(max\_pc)

# since cumulative score > 85%, new transformed dataset have 7 PCs as attributes

transformed\_data <- pca\_result$x[,1:max\_pc]

head(transformed\_data)

#Determine the number of cluster centres

set.seed(26)

# NbClust method

clusterNo\_method1=NbClust(transformed\_data,distance="euclidean", min.nc=2,max.nc=10,method="kmeans",index="all")

print(clusterNo\_method1)

# Elbow Method

fviz\_nbclust(transformed\_data, kmeans, method = 'wss')

# Gap Statistic method

fviz\_nbclust(transformed\_data, kmeans, method = 'gap\_stat')

# Silhouette Method

fviz\_nbclust(transformed\_data, kmeans, method = 'silhouette')

# kmeans clustering K=2

k = 2

kmeans\_wines = kmeans(transformed\_data, centers = k, nstart = 10)

kmeans\_wines

#Illustration of the clusters

fviz\_cluster(kmeans\_wines, data = transformed\_data)

# Internal evaluation

wss = kmeans\_wines$tot.withinss

bss = kmeans\_wines$betweenss

wss

bss

# The silhouette analysis

sil <- silhouette(kmeans\_wines$cluster, dist(cleaned\_data\_boxplot))

fviz\_silhouette(sil)

# Calinski-Harabasz Index

ch\_values <- NULL

for (i in 1:10) {

kmeans\_Wines\_cslinski = kmeans(transformed\_data, centers = i, nstart = 10)

ch <- calinhara(transformed\_data,kmeans\_Wines\_cslinski$cluster)

ch\_values <- c(ch\_values,ch)

}

# plot the Calinski-Harabasz Index values for different k values.

plot(1:10, ch\_values, type = "b", xlab = "Number of Clusters", ylab = "Calinski-Harabasz Index")

max(ch\_values[2])

## Neural Network

library(readxl) #for read xlsx file

library(neuralnet)

library(Metrics) # to use RMSE function

# Load the dataset

exchange\_data <- read\_excel("D:/E driver/IIT/Level\_5\_year\_2/Semester 2/Machine learning/Assessment/Git-implimenatations/Machine-Learning-CW/Financial Forecasting/ExchangeUSD.xlsx")

colnames(exchange\_data) <- c("YYYY/MM/DD", "Wdy", "USD\_EUR")

# Extract exchange rates column

exchange\_rates <- as.data.frame(exchange\_data$"USD\_EUR")

#Checking missing values

sum(is.na(exchange\_rates))

# Convert data frame to time series object

#ts\_data <- ts(exchange\_rates)

# Remove missing values

exchange\_rates <- na.omit(exchange\_rates)

# Calculate lagged values

t\_1 <- lag(exchange\_rates, 1)

t\_2 <- lag(exchange\_rates, 2)

t\_3 <- lag(exchange\_rates, 3)

t\_4 <- lag(exchange\_rates, 4)

# Creating I/O Matrixes

M1 <- cbind(t\_1,exchange\_rates)

M2 <- cbind(t\_1,t\_2,exchange\_rates)

M3 <- cbind(t\_1,t\_2,t\_3,exchange\_rates)

M4 <- cbind(t\_1,t\_2,t\_3,t\_4,exchange\_rates)

# Rename column names in I/O Matrixes

colnames(M1)<-c("t-1","output")

colnames(M2)<-c("t-1","t-2","output")

colnames(M3)<-c("t-1","t-2","t-3","output")

colnames(M4)<-c("t-1","t-2","t-3","t-4","output")

# Remove missing values

M1 <- M1[complete.cases(M1),]

M2 <- M2[complete.cases(M2),]

M3 <- M3[complete.cases(M3),]

M4 <- M4[complete.cases(M4),]

# min max normalization

min\_max\_normalization <- function(x) {

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

}

# Apply normalization

M1\_norm <- as.data.frame(lapply(M1, min\_max\_normalization))

M2\_norm <- as.data.frame(lapply(M2, min\_max\_normalization))

M3\_norm <- as.data.frame(lapply(M3, min\_max\_normalization))

M4\_norm <- as.data.frame(lapply(M4, min\_max\_normalization))

# Checking the range of normalized data

summary(M1\_norm)

summary(M2\_norm)

summary(M3\_norm)

summary(M4\_norm)

#Creating training and testing data

M1\_train\_norm <- M1\_norm[1:400, ]

M2\_train\_norm <- M2\_norm[1:400, ]

M3\_train\_norm <- M3\_norm[1:400, ]

M4\_train\_norm <- M4\_norm[1:400, ]

M1\_test\_norm <- M1\_norm[401:499, ]

M2\_test\_norm <- M2\_norm[401:498, ]

M3\_test\_norm <- M3\_norm[401:497, ]

M4\_test\_norm <- M4\_norm[401:496, ]

# Evaluation function

evaluate <- function(actual,prediction){

rmse <- rmse(actual = actual,predicted = prediction)

mae <- mae(actual = actual, predicted = prediction)

mape <- mape(actual = actual, predicted = prediction)

smape <- smape(actual = actual, predicted = prediction)

df <- data.frame(stat.indices = c("RMSE","MAE","MAPE","sMAPE"),values = c(rmse,mae,mape,smape))

return(df)

}

# De-Normalization

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

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

}

set.seed(20)

#M1

original\_train\_outputs\_M1 <- M1[1:400,"output"]

original\_test\_outputs\_M1 <- M1[401:499,"output"]

min\_output\_M1 <- min(original\_train\_outputs\_M1)

max\_output\_M1 <- max(original\_train\_outputs\_M1)

# neural network code/training

model1 <- neuralnet(output ~ t.1, data = M1\_train\_norm, hidden = 8, act.fct = 'logistic', linear.output = T)

plot(model1)

model1\_results <- neuralnet::compute(model1,M1\_test\_norm[1])

predicted\_output\_model1 <- unnormalize(model1\_results$net.result,min\_output\_M1,max\_output\_M1) #obtain predicted output

evaluate(original\_test\_outputs\_M1,predicted\_output\_model1) # evaluation

# neural network code/training

model2 <- neuralnet(output ~ t.1, data = M1\_train\_norm, hidden = c(8,5), act.fct = 'tanh', linear.output = T)

plot(model2)

model2\_results <- neuralnet::compute(model2,M1\_test\_norm[1])

predicted\_output\_model2 <- unnormalize(model2\_results$net.result,min\_output\_M1,max\_output\_M1) #obtain predicted output

evaluate(original\_test\_outputs\_M1,predicted\_output\_model2) # evaluation

# neural network code/training

model3 <- neuralnet(output ~ t.1, data = M1\_train\_norm, hidden = c(10,6), act.fct = 'logistic', linear.output = F)

plot(model3)

model3\_results <- neuralnet::compute(model3,M1\_test\_norm[1])

predicted\_output\_model3 <- unnormalize(model3\_results$net.result,min\_output\_M1,max\_output\_M1) #obtain predicted output

evaluate(original\_test\_outputs\_M1,predicted\_output\_model3) # evaluation

#M2

original\_train\_outputs\_M2 <- M2[1:400,"output"]

original\_test\_outputs\_M2 <- M2[401:498,"output"]

min\_output\_M2 <- min(original\_train\_outputs\_M2)

max\_output\_M2 <- max(original\_train\_outputs\_M2)

# neural network code/training

model4 <- neuralnet(output ~ t.1 + t.2, data = M2\_train\_norm, hidden = 12, act.fct = 'logistic', linear.output = T)

plot(model4)

model4\_results <- neuralnet::compute(model4,M2\_test\_norm[1:2])

predicted\_output\_model4 <- unnormalize(model4\_results$net.result,min\_output\_M2,max\_output\_M2) #obtain predicted output

evaluate(original\_test\_outputs\_M2,predicted\_output\_model4) # evaluation

# neural network code/training

model5 <- neuralnet(output ~ t.1 + t.2, data = M2\_train\_norm, hidden = c(7,5), act.fct = 'tanh', linear.output = T)

plot(model5)

model5\_results <- neuralnet::compute(model5,M2\_test\_norm[1:2])

predicted\_output\_model5 <- unnormalize(model5\_results$net.result,min\_output\_M2,max\_output\_M2) #obtain predicted output

evaluate(original\_test\_outputs\_M2,predicted\_output\_model5) # evaluation

# neural network code/training

model6 <- neuralnet(output ~ t.1 + t.2, data = M2\_train\_norm, hidden = c(12,7), act.fct = 'logistic', linear.output = F)

plot(model6)

model6\_results <- neuralnet::compute(model6,M2\_test\_norm[1:2])

predicted\_output\_model6 <- unnormalize(model6\_results$net.result,min\_output\_M2,max\_output\_M2) #obtain predicted output

evaluate(original\_test\_outputs\_M2,predicted\_output\_model6) # evaluation

#M3

original\_train\_outputs\_M3 <- M3[1:400,"output"]

original\_test\_outputs\_M3 <- M3[401:497,"output"]

min\_output\_M3 <- min(original\_train\_outputs\_M3)

max\_output\_M3 <- max(original\_train\_outputs\_M3)

# neural network code/training

model7 <- neuralnet(output ~ t.1 + t.2 + t.3, data = M3\_train\_norm, hidden = 10, act.fct = 'logistic', linear.output = T)

plot(model7)

model7\_results <- neuralnet::compute(model7,M3\_test\_norm[1:3])

predicted\_output\_model7 <- unnormalize(model7\_results$net.result,min\_output\_M3,max\_output\_M3) #obtain predicted output

evaluate(original\_test\_outputs\_M3,predicted\_output\_model7) # evaluation

# neural network code/training

model8 <- neuralnet(output ~ t.1 + t.2 + t.3, data = M3\_train\_norm, hidden = c(10,5), act.fct = 'tanh', linear.output = T)

plot(model8)

model8\_results <- neuralnet::compute(model8,M3\_test\_norm[1:3])

predicted\_output\_model8 <- unnormalize(model8\_results$net.result,min\_output\_M3,max\_output\_M3) #obtain predicted output

evaluate(original\_test\_outputs\_M3,predicted\_output\_model8) # evaluation

# neural network code/training

model9 <- neuralnet(output ~ t.1 + t.2 + t.3, data = M3\_train\_norm, hidden = c(10,8), act.fct = 'logistic', linear.output = F)

plot(model9)

model9\_results <- neuralnet::compute(model9,M3\_test\_norm[1:3])

predicted\_output\_model9 <- unnormalize(model9\_results$net.result,min\_output\_M3,max\_output\_M3) #obtain predicted output

evaluate(original\_test\_outputs\_M3,predicted\_output\_model9) # evaluation

#M4

original\_train\_outputs\_M4 <- M4[1:400,"output"]

original\_test\_outputs\_M4 <- M4[401:496,"output"]

min\_output\_M4 <- min(original\_train\_outputs\_M4)

max\_output\_M4 <- max(original\_train\_outputs\_M4)

# neural network code/training

model10 <- neuralnet(output ~ t.1 + t.2 + t.3 + t.4, data = M4\_train\_norm, hidden = 12, act.fct = 'logistic', linear.output = T)

plot(model10)

model10\_results <- neuralnet::compute(model10,M4\_test\_norm[1:4])

predicted\_output\_model10 <- unnormalize(model10\_results$net.result,min\_output\_M4,max\_output\_M4) #obtain predicted output

evaluate(original\_test\_outputs\_M4,predicted\_output\_model10) # evaluation

# neural network code/training

model11 <- neuralnet(output ~ t.1 + t.2 + t.3 + t.4, data = M4\_train\_norm, hidden = c(10,9), act.fct = 'tanh', linear.output = T)

plot(model11)

model11\_results <- neuralnet::compute(model11,M4\_test\_norm[1:4])

predicted\_output\_model11 <- unnormalize(model11\_results$net.result,min\_output\_M4,max\_output\_M4) #obtain predicted output

evaluate(original\_test\_outputs\_M4,predicted\_output\_model11) # evaluation

# neural network code/training

model12 <- neuralnet(output ~ t.1 + t.2 + t.3 + t.4, data = M4\_train\_norm, hidden = c(12,8), act.fct = 'logistic', linear.output = F)

plot(model12)

model12\_results <- neuralnet::compute(model12,M4\_test\_norm[1:4])

predicted\_output\_model12 <- unnormalize(model12\_results$net.result,min\_output\_M4,max\_output\_M4) #obtain predicted output

evaluate(original\_test\_outputs\_M4,predicted\_output\_model12) # evaluation

# Evaluate model 1

evaluate(original\_test\_outputs\_M1, predicted\_output\_model1)

# Evaluate model 2

evaluate(original\_test\_outputs\_M1, predicted\_output\_model2)

# Evaluate model 3

evaluate(original\_test\_outputs\_M1, predicted\_output\_model3)

# Evaluate model 4

evaluate(original\_test\_outputs\_M2, predicted\_output\_model4)

# Evaluate model 5

evaluate(original\_test\_outputs\_M2, predicted\_output\_model5)

# Evaluate model 6

evaluate(original\_test\_outputs\_M2, predicted\_output\_model6)

# Evaluate model 7

evaluate(original\_test\_outputs\_M3, predicted\_output\_model7)

# Evaluate model 8

evaluate(original\_test\_outputs\_M3, predicted\_output\_model8)

# Evaluate model 9

evaluate(original\_test\_outputs\_M3, predicted\_output\_model9)

# Evaluate model 10

evaluate(original\_test\_outputs\_M4, predicted\_output\_model10)

# Evaluate model 11

evaluate(original\_test\_outputs\_M4, predicted\_output\_model11)

# Evaluate model 12

evaluate(original\_test\_outputs\_M4, predicted\_output\_model12)

#graphical representation of models

par(mfrow=c(1,1))

plot(original\_test\_outputs\_M4, predicted\_output\_model12 ,col='red',main='Real vs predicted NN',pch=18,cex=0.7)

abline(a=0, b=1, h=90, v=90)

# References

[1] M. Charrad, N. Ghazzali, and A. Niknafs, “NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set”, Journal of Statistical Software 61(6), pp.1-36,2014.

[2] H. Humaira and R. Rasyidah, “Determining The Appropiate Cluster Number Using Elbow Method for K-Means Algorithm”, Conference: Proceedings of the 2nd Workshop on Multidisciplinary and Applications (WMA), PP 24-25. 2018.

[3] A. M. El-Mandouh, Laila A.,A. Hamdi, and H. Mohamed, “Optimized K-Means Clustering Model based on Gap Statistic”, International Journal of Advanced Computer Science and Applications 10(1), 2019.

[4] A. Soetewey, Stats and R, Outlier Detection in R, 2020-08-11, https://statsandr.com/blog/outliers-detection-in-r/.