Table of Contents

1st Objective (partitioning clustering) 3
Clustering Part 3
Optimal Number of Clusters: 7
2nd Objective (MLP) 10
Different Structures of MLP 12
MLP - NN With Two Inputs 12
MPL-NN With Three Inputs 14
MPL-NN With Four Inputs 15
Performance Indices 17
3rd Objective Support Vector Regression (SVR) 18
SVR With Two Inputs 19
Model 1 - Linear Kernel 19
Model 2 - Radial Kernel 20
SVR With Three Inputs 21
Model 3- Linear Kernel 21
SVR With Four Inputs 24
Model 5- Linear Kernel 24
Model 6 – Radial Kernel 25
Performance Indices 26
Appendix 1 27
1st Objective (partitioning clustering) 27
2nd Objective (MLP) 30
3rd Objective (SVR) 41

1st Objective (partitioning clustering)

Clustering Part

The dataset “Whitewine”, is an analysis of white wine chemical properties and their ranking by wine testers. It is easily observed that even though all variables are numerical, the last (12th) column is not an actual value but a quality ranking ranging from 3 to 9. This variable will have to be excluded from the analysis as it does not contain information on the wine chemical consistency and will cause complications in the clustering procedure.

```
str(Whitewine)
summary(Whitewine)
```

```
Classes ‘tbl_df’, ‘tbl’ and 'data.frame': 4898 obs. of  12 variables:
 $ fixed acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
 $ volatile acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
 $ citric acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
 $ residual sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
 $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0
.044 ...
 $ free sulfur dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
 $ total sulfur dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
 $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
 $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
 $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
 $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality             : num  6 6 6 6 6 6 6 6 6 6 ...
```

```
> summary(Whitewine)
 fixed acidity    volatile acidity  citric acid     residual sugar     chl
orides        free sulfur dioxide
 Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600   Min.
:0.00900   Min.   :  2.00
 1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700   1st Q
u.:0.03600   1st Qu.: 23.00
 Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200   Media
n :0.04300   Median : 34.00
 Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391   Mean
:0.04577   Mean   : 35.31
 3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900   3rd Q
u.:0.05000   3rd Qu.: 46.00
 Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800   Max.
:0.34600   Max.   :289.00
 total sulfur dioxide    density             pH          sulphates
alcohol         quality
 Min.   :  9.0        Min.   :0.9871   Min.   :2.720   Min.   :0.2200   Mi
n.   : 8.00   Min.   :3.000
 1st Qu.:108.0        1st Qu.:0.9917   1st Qu.:3.090   1st Qu.:0.4100   1s
t Qu.: 9.50   1st Qu.:5.000
 Median :134.0        Median :0.9937   Median :3.180   Median :0.4700   Me
dian :10.40   Median :6.000
 Mean   :138.4        Mean   :0.9940   Mean   :3.188   Mean   :0.4898   Me
an   :10.51   Mean   :5.878
 3rd Qu.:167.0        3rd Qu.:0.9961   3rd Qu.:3.280   3rd Qu.:0.5500   3r
d Qu.:11.40   3rd Qu.:6.000
 Max.   :440.0        Max.   :1.0390   Max.   :3.820   Max.   :1.0800   Ma
x.   :14.20   Max.   :9.000
```

Analysing the data even further, and excluding the column 12, it is observed that there are 11 attributes with 4898 inserts, however there are some extreme differences between the minimum and maximum values of each variable. These outliers can distort the analysis results, so their removal is necessary. Creating some graphs to check the distribution of each variable, will give a better ideal of the variables that contain outliers. A boxplot is a quick way of identifying any outliers and the histogram is an indication of each variable’s distribution.

The figure 1a is displaying the boxplot of each variable, where “Residual Sugar”, Free Sulfur Dioxide” and “Total Sulfur Dioxide” are appearing to have extreme values.

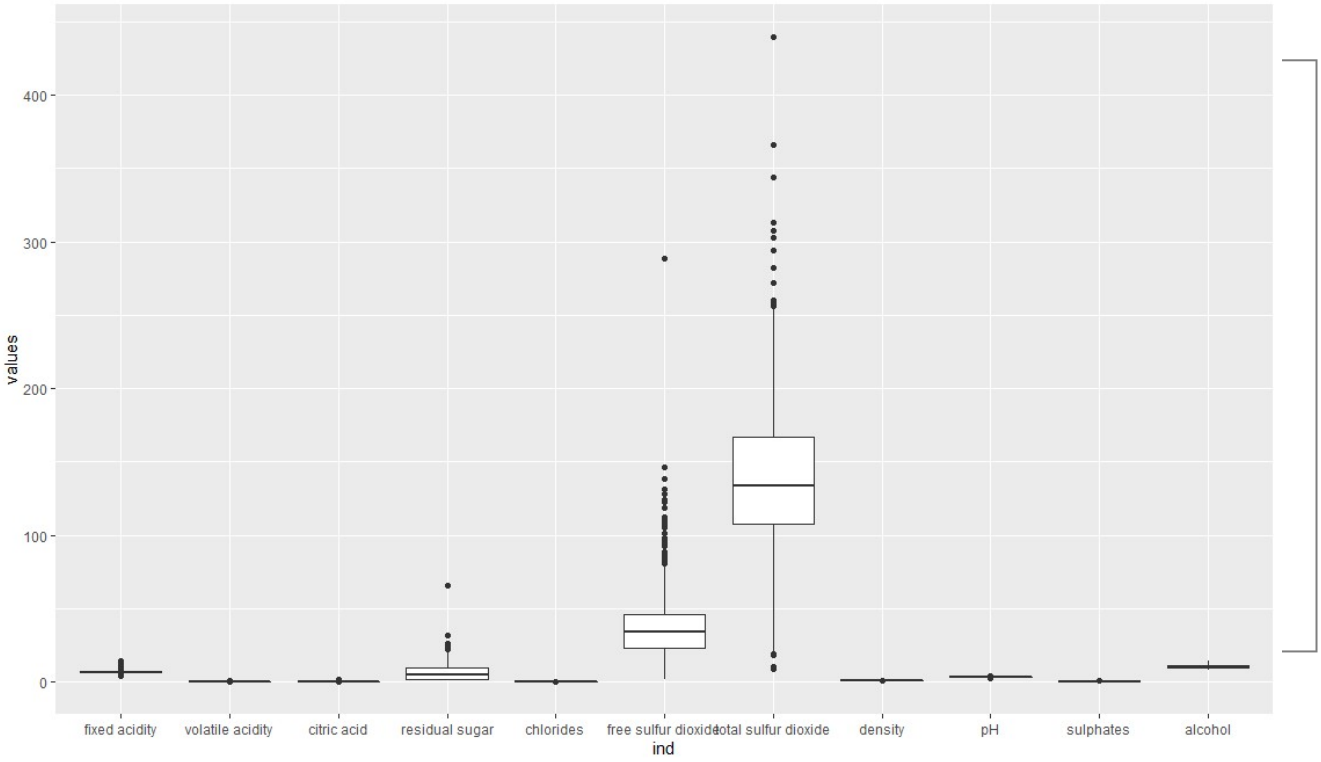

Figure 1a– White Wine Variables Boxplot

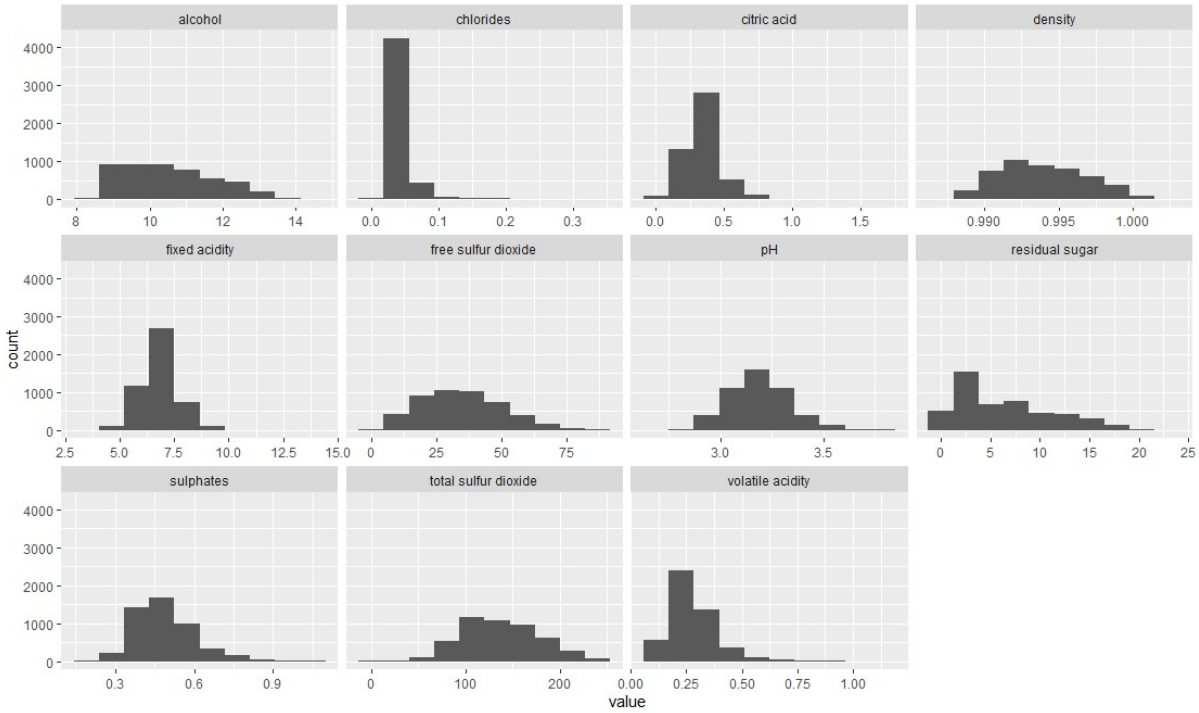

Figure 1b - White Wine Histogram per Variable

In order to find the extreme values for each variable, box plots are being plotted, individually for each variable. For each of the three variables with extreme outliers, a threshold (4th Quartile) will be set, and any data with higher values, will be removed.

```
boxplot(x$`residual sugar`, main="residual sugar")
boxplot(x$`free sulfur dioxide`, main=" free sulfur dioxide")
boxplot(x$`total sulfur dioxide`, main=" total sulfur dioxide")
```

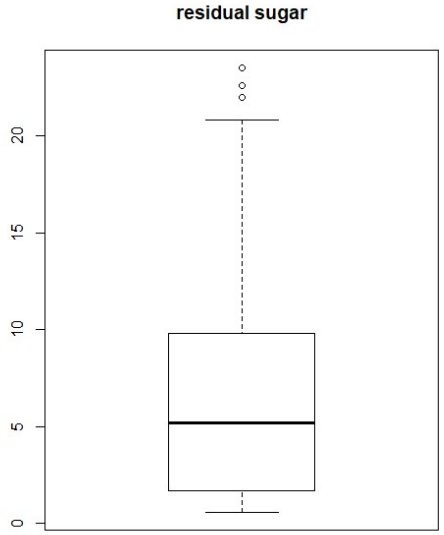

Figure 2a - Residual Sugar Boxplot

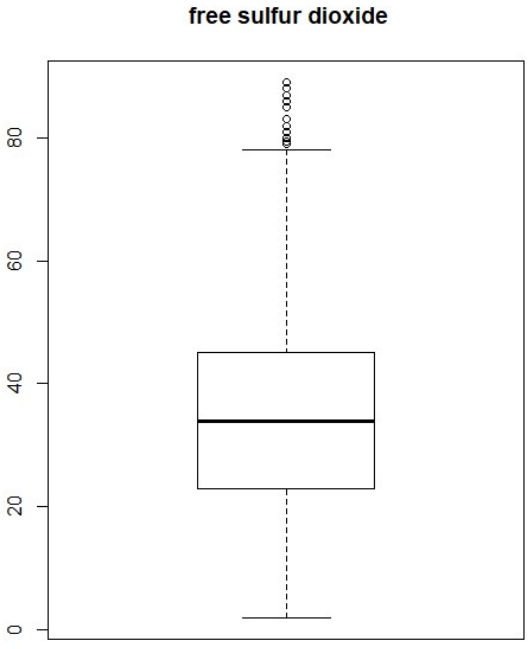

Figure 2b- Free Sulfur Boxplot

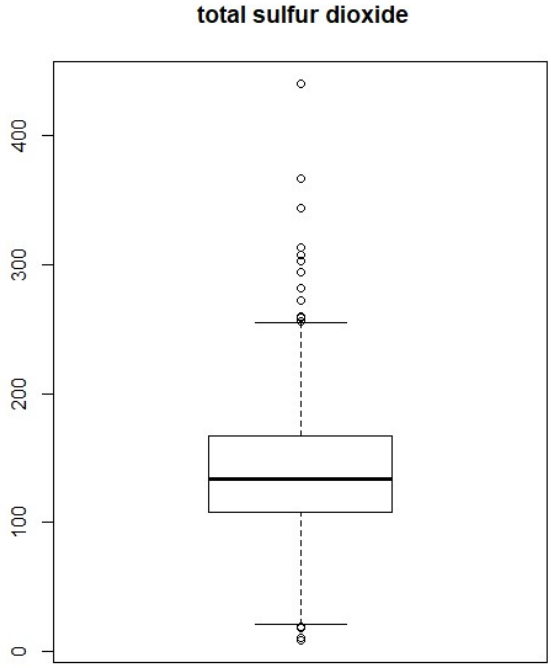

Figure 2c- Free Dioxide Boxplot

From the boxplots, it is observed that there are extreme values of residual sugar that are more than 25, free sulfur dioxide more than 90 and total sulfur dioxide more than 250. For the best results of the clustering, these values should be removed completely from the dataset.

Using Excel's function "filter", any rows containing values larger than the above mentioned for 3 variables' thresholds, have been removed. Following the removal of the extreme values of the above mentioned 3 variables, it is observed that filtering the results by quality, there are only 15 rows from quality 3 and 5 from quality 9. Since these are very low numbers and make a very small percentage of the total dataset, in order not to interfere with the accuracy of the clustering results, are also completely removed from the dataset.

In the following table, the entries (complete rows) were removed from each quality and are followed by their total data percentage. The total number of removed outliers is 75, which makes 1.53% of the entire dataset. The new dataset is consisted of 4827 observations.

Quality	No of entries	Total %
3	22	0.45
4	3	0.06
5	19	0.39
6	20	0.41
7	2	0.04
8	4	0.08
9	5	0.10
Total	**75**	**1.53**

The new dataset is named Whitewine_nooutliers. Based on this, will be performed the k-means clustering with Euclidean distance, after the data has been scaled for uniformity and best accuracy of the results.

```
x<- Whitewine_nooutliers[,-12]

data.train<-scale(x)

y<- Whitewine_nooutliers$quality
```

Optimal Number of Clusters:

The quality score that each wine tester gave to the wines in the dataset were 7, however 2 have been removed as above mentioned (quality 3 and 9). However it is not quite clear how many categories of wines are there. In order to separate the wined into categories, based on their chemical properties, the unsupervised clustering method k-means will be used. Using R's package NbClust, the optimal number of clusters for this dataset can be determined. The results displayed below, show that the optimal number of clusters is 2 but close to this result, is also clustering with 3 clusters.

```
library(factoextra)
library(NbClust)

set.seed(123)
Clustno <- NbClust(x, distance = "euclidean", min.nc = 2, max.nc = 15,
                   method = "kmeans")

barplot(x(nc$Best.n[1,]),
        xlab="Numer of Clusters", ylab="Number of Criteria",
        main="Number of Clusters Chosen by 26 Criteria").
```

```
* Among all indices:
* 10 proposed 2 as the best number of clusters
* 8 proposed 3 as the best number of clusters
* 2 proposed 10 as the best number of clusters
* 2 proposed 12 as the best number of clusters
* 1 proposed 14 as the best number of clusters
* 1 proposed 15 as the best number of clusters

                   ***** Conclusion *****

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

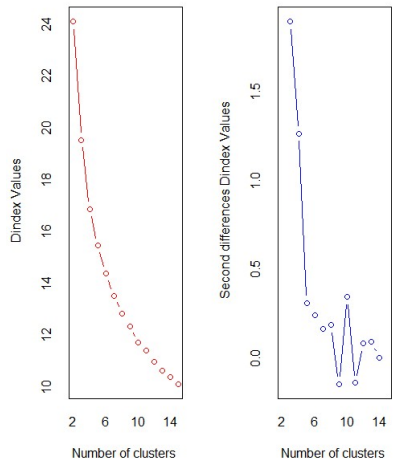

Figure 3 - NbClust Graphical Display

Trying to validate this result, values for “k” have been manually assigned, such as k= 2,3,4,5 and 6. If Plotting a table against the results of the last column (quality), the following tables are being obtained.

```
kc2<-kmeans(x,2)
kc2
table(y, kc2$cluster)

kc3<-kmeans(x,3)
kc3
table(y, kc3$cluster)

kc4<-kmeans(x,4)
kc4
table(y, kc4$cluster)

kc<-kmeans(x,5)
table(y, kc$cluster)

kc<-kmeans(x,6)
table(y, kc$cluster)
```

Table 1 - K-Means Clustering With k=2

K=2	Cluster 1	Cluster 2	Total
4	**106**	55	**161**
5	614	**825**	**1439**
6	**1250**	928	**2178**
7	**627**	251	**878**
8	**120**	51	**171**
Total	**2717**	**2110**	**4827**

Correct	*2928*
Accuracy	***0.6066***

Table 2 - K-Means Clustering With k=3

K=3	Cluster 1	Cluster 2	Cluster 3	Total
4	37	33	**91**	**161**
5	522	**561**	356	**1439**
6	488	**866**	824	**2178**
7	79	**395**	404	**878**
8	16	**79**	76	**171**
Total	**1142**	**1934**	**1751**	**4827**

Correct	*1953*
Accuracy	***0.4046***

Table 3- K-Means Clustering With k=4

K=4	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Total
4	30	39	63	29	161
5	489	386	198	366	1439
6	626	771	441	340	2178
7	209	395	216	58	878
8	41	80	39	11	171
Total	1395	1671	957	804	4827

Correct	*1530*
Accuracy	***0.317***

Table 4 - K-Means Clustering With k=5

K=5	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Total
4	20	49	19	31	42	161
5	279	133	348	437	242	1439
6	278	226	541	495	638	2178
7	38	107	277	143	313	878
8	7	20	53	30	61	171
Total	622	535	1238	1136	1296	4827

Correct	*1189*
Accuracy	***0.2463***

Table 5 - K-Means Clustering With k=6

K=6	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Total
4	27	6	15	34	45	34	161
5	302	166	327	180	113	351	1439
6	534	162	477	512	181	312	2178
7	279	19	188	259	74	59	878
8	54	6	36	50	15	10	171
Total	1196	359	1043	1035	428	766	4827

Correct	*1186*
Accuracy	***0.2457***

The red values cells are the “dominant”, largest values of each row. By taking the largest values of each row, it is observed that the compact results and the more “distinctive dominant values” are in clustering with cluster numbers 2 and 3. Also, the total number of items per cluster looks more equally distributed as well.

In table 2, where k=3, there is no dominant value in cluster 1. However since the amount of data points from quality 5 are close for cluster 1 and 2, and since in cluster 2 there are more dominant values (as well as the most dominant, 866), it is assumed that for cluster 1, the dominant is 522, marked with grey. Respectively, with grey marking are the chosen or assigned dominant numbers for the rest of tables, for the same reason.

The larger the number of clusters, the more difficult it is for the system to distribute them as there is no distinctive dominant and when k>3, the total number of datapoints per cluster varies unequally

and becomes more difficult to manually assign a dominant value. Especially when k=6, cluster 2 has no dominant (since the qualities are 5 and the clusters 6).

By calculating their accuracy, it is confirmed that k=2 clusters and following closely k=3 clusters is the optimal clustering result. The accuracy calculation is obtained by adding number of all the "correctly" assigned datapoints to a cluster divided by the total number of datapoints. To the models where the k-means has not performed well and there is no distinctive dominant, we calculate the assigned larger values marked in green.

The results show that the ideal no of clusters is 2, where the datapoints are as equally distributed as possible and this model has the highest accuracy.

The means of clustering with k= 2 are as follow:

```
K-means clustering with 2 clusters


Cluster means:
  fixed acidity volatile acidity citric acid residual sugar  chlorides
1      6.783272        0.2728874   0.3214943        4.75749 0.04221642
free sulfur dioxide
26.95620

2      6.937251        0.2842938   0.3501137        8.39372 0.05028294
45.07227
  total sulfur dioxide   density       pH sulphates   alcohol
1             107.8515 0.9927547 3.188281 0.4799264 10.967519
2             175.6391 0.9955928 3.188303 0.5017583  9.938687
```

To conclude with, the Whitewine data set, can be clustered in 2 clusters of 2717 and 2110 data points respectively. The means of each attribute may help the wine experts in the wine market into understanding the relationship between human testing and the chemical consistency of wine varieties.

2nd Objective (MLP)

For this objective the dataset ExchangeUSD is being analysed. The file contains the exchange rate between the USD and EUR daily for a period of almost 3 years, therefore this is a time-series analysis. Long term prediction of time series, has been proven a rather difficult task for machine learning, especially in regards to the number of inputs required in order to obtain the forecasting result.

One of the most accurate methods in order to forecast long term time series results, is the use of neural networks. In the neural networks, the use of hidden layers and nodes play a significant role in the results, however the number if inputs can be also crucial for accurate prediction. However, there is no known rule on how to select the best number of inputs for each problem, as timeseries can have seasonality, odd results that are not dependant on the previous results and other factors that can affect the accuracy of the prediction. The artificial neural networks, is a supervised technique that even though can bring accurate results, it is considered a black box where their calculation is not understandable for each result, therefore there is no clear way, one can define the best number of inputs.

It is advised, that a combination of different input variables is being tested for each prediction. Then, the best model and combination is selected based on their performance on the testing set. Even though it may be perceived as the more inputs, the better training, therefore the best result, this is not usually the case since the model may perform well in training but cause overfitting in the testing process. Another issue that may be addressed, is when the inputs are highly correlated, the training process may not be able to recognise clear patterns and the results may not be as accurate.

Selecting multiple inputs and neural network structures, can be time consuming, however for the most accurate results, a multilevel perceptron (MLP) with the optimal combination of inputs and internal structure can perform quite well.

In the Exchange data set, various combinations of inputs and internal structures will be used in order to test the results and select the model with the highest performance.

By running the commands summary and plotting a histogram of the data, it is obtained the first impression of what the data set is consisted of.

```
View(ExchangeUSD)
summary(ExchangeUSD)
str(ExchangeUSD)
qplot(ExchangeUSD$`USD/EUR`, geom="histogram")
```

```
USD/EUR
 Min.   :1.206
 1st Qu.:1.287
 Median :1.308
 Mean   :1.305
 3rd Qu.:1.327
 Max.   :1.417


Classes ‘tbl_df’, ‘tbl’ and 'data.frame':     500 obs. of  3 variables:
 $ YYYY/MM/DD: POSIXct, format: "2011-10-13" "2011-10-14" "2011-10-17" ...
 $ Wdy       : chr  "Thu" "Fri" "Mon" "Tue" ...
 $ USD/EUR   : num  1.37 1.39 1.38 1.37 1.38 ...
```

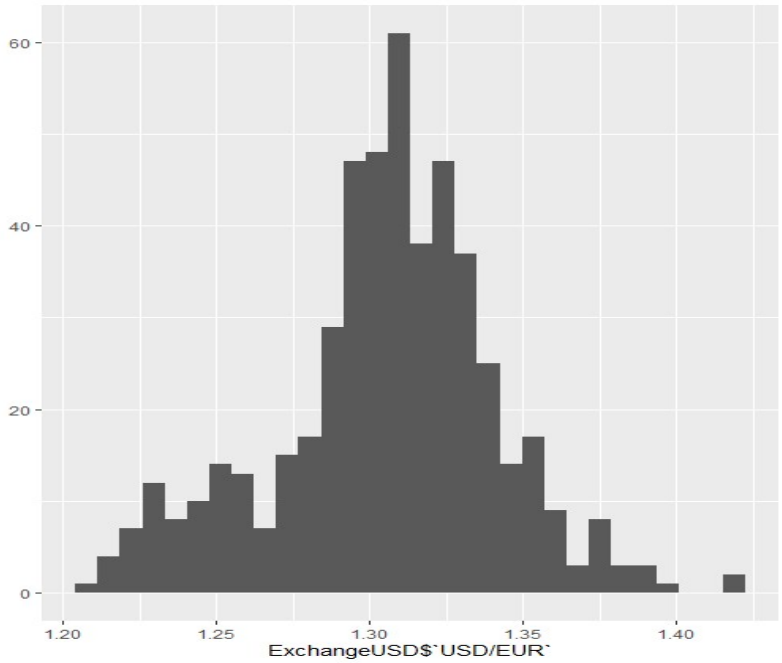

Figure 4 - Histogram of USD / EUR

The goal is to predict the exchange rate of the following dates. For time series prediction and regression-based models, is common to use historical data as training input.

One of the prediction methods that can be used is the MPL neural network, that will be trained with the past values in order to be able to predict the future values, as accurately as possible, very effective for time series data.

As in an MLP, the input variables are combined linearly, then may be considered necessary to normalise the inputs.

The data set is defined as time series in R and only the last column is used, where the exchange rates

```
ExchangeUSD <- ExchangeUSD[,3]
rate <- ts (ExchangeUSD)
norm_data <- sapply(ExchangeUSD, function(x) (x - min(x))/(max(x) - min(x)))
class(norm_data)
norm_data <- data.frame(norm_data)
head(norm_data)
norm_data
write.table(norm_data, file = "newdata.csv", sep=",", row.names = F)
```

are displayed. After the normalisation function is being applied, the data frame is exported to excel, by using the command “write.table”.

```
> head(norm_data)
    USD.EUR
1 0.7909953
2 0.8526066
3 0.8090047
4 0.7853081
5 0.8118483
6 0.7635071
```

An excel fie with a data frame of one column and 500 observations is created. However, in order to perform any prediction technique, it is essential to define the inputs and outputs. For this, it is used lagging for as many inputs required and one output.

Different Structures of MLP

MLP - NN With Two Inputs

For the first model of neural network, 2 inputs will be used. In this case, the first rows and the general outline of the normalised and lagged dataset are as follow (Mydata1):

	Input1	Input2	Output
1	0.7909953	0.8526066	0.8090047
2	0.8526066	0.8090047	0.7853081
3	0.8090047	0.7853081	0.8118483
4	0.7853081	0.8118483	0.7635071
5	0.8118483	0.7635071	0.8582938
6	0.7635071	0.8582938	0.8867299
7	0.8582938	0.8867299	0.8767773
8	0.8867299	0.8767773	0.8421801
9	0.8767773	0.8421801	1.0000000
10	0.8421801	1.0000000	0.9966825

The new dataset is defined as the normalised and lagged dataset with 2 inputs and 1 output. Approximately 70% of the data will be used for training and the remaining 30% will be used for testing. During the lagging process, for every input, the data are being moved to the next day, so there one row less at the end of the dataset. For 2 inputs, the dataset of total 500 rows will have 498. Training a model using the neuralnet package.

```
str(Mydata1)
summary(Mydata1)
train1 <- Mydata1[1:400,]
test1 <- Mydata1[401:498,]
```

Various models with 2 inputs will be tested, in order to gather index matrices regarding their performance. A sample of the code used is as follows:

```
##1 HIDDEN LAYER, 3 NODES, LINEAR OUTPUT, LOGISSTIC ACTIVATION FUNCTION
set.seed(1234)
NN1 <- neuralnet(Output ~ Input1 + Input2, data = train1, hidden=3,
                 act.fct= "logistic", threshold = 0.01, linear.output = T)
plot(NN1)

## Evaluating Model Performance
model1_results <- compute(NN1, test1[1:2])
predicted1 <- model1_results$net.result
cor(predicted1, test1$Output)
```

The correlation results below.

```
[,1]
[1,] 0.9531168
```

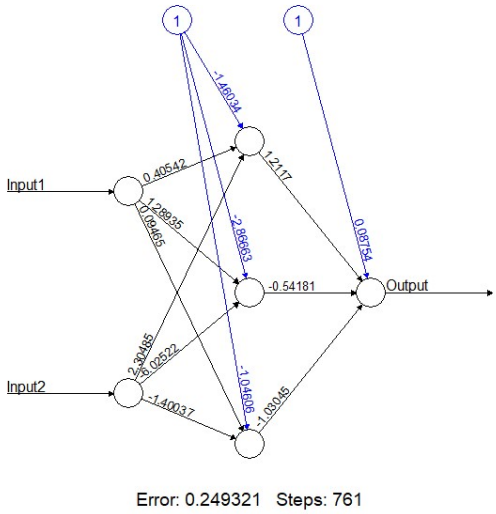

Figure 5 - NN1 Neural Network 1

Accordingly, are created and plotted the following combinations for 2 inputs neural networks, changing the number of hidden layers, nodes, activation function and output (Appendix2).

- NN1 with 1 hidden layer, 3 nodes, linear output, logistic activation function
- NN2 with 1 hidden layer, 7 nodes, non - linear output, tanh activation function
- NN3 with 2 hidden layers - 4 and 2 nodes, non - linear output, logistic activation function
- NN4 with 2 hidden layers - 10 and 6 nodes, linear output, logistic activation function

MPL-NN With Three Inputs

A new dataset with lagging for three inputs has been created as Mydata2 and looks as below:

	Input1	Input2	Input3	Output
1	0.7909953	0.8526066	0.8090047	0.7853081
2	0.8526066	0.8090047	0.7853081	0.8118483
3	0.8090047	0.7853081	0.8118483	0.7635071
4	0.7853081	0.8118483	0.7635071	0.8582938
5	0.8118483	0.7635071	0.8582938	0.8867299
6	0.7635071	0.8582938	0.8867299	0.8767773
7	0.8582938	0.8867299	0.8767773	0.8421801
8	0.8867299	0.8767773	0.8421801	1.0000000
9	0.8767773	0.8421801	1.0000000	0.9966825
10	0.8421801	1.0000000	0.9966825	0.8938389
11	1.0000000	0.9966825	0.8938389	0.7649289
12	0.9966825	0.8938389	0.7649289	0.8246445

Separating Mydata2 into train and test set, since there are 3 inputs, the total number of observations will be 497. Several models will be creating trying different combinations of hidden layers, nodes, activation function and output (Appendix2).

```
train2 <- Mydata2[1:400,]
test2 <- Mydata2[401:497,]
set.seed(1234)
NN5 <- neuralnet(Output ~ Input1 + Input2 + Input3, train2, hidden=4,
                 act.fct= "logistic", threshold = 0.01, linear.output = T)

## Evaluating Model Performance
model5_results <- compute(NN5, test2[1:3])
predicted5 <- model5_results$net.result
cor(predicted5, test2$Output)
```

The correlation results below.

```
[,1]
[1,] 0.9522907
```

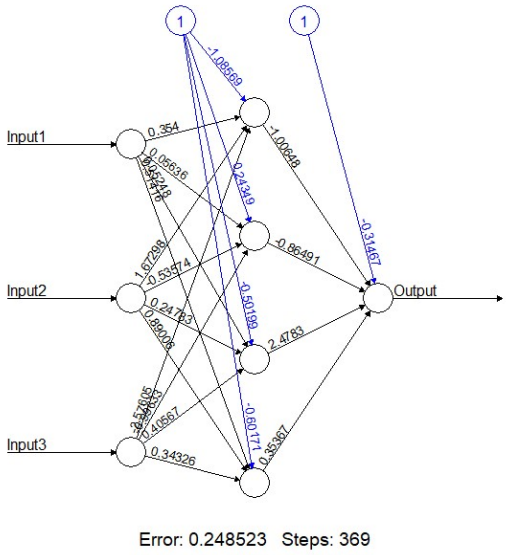

Figure 6 - NN5 Neural Network 5

- NN5 1 hidden layer - 4 nodes, linear output, logistic activation function
- NN6 1 hidden layer - 6 nodes, non - linear output, tanh activation function
- NN7 2 hidden layers - 3 and 6 nodes, linear output, tanh activation function
- NN8 2 hidden layers - 10 and 4 nodes, linear output, logistic activation function
- NN9 2 hidden layers - 7 and 4 nodes, linear output, tanh activation function

MPL-NN With Four Inputs

A new dataset with lagging for three inputs has been created as Mydata3 and looks as below:

	Input1	Input2	Input3	Input4	Output
1	0.7909953	0.8526066	0.8090047	0.7853081	0.8118483
2	0.8526066	0.8090047	0.7853081	0.8118483	0.7635071
3	0.8090047	0.7853081	0.8118483	0.7635071	0.8582938
4	0.7853081	0.8118483	0.7635071	0.8582938	0.8867299
5	0.8118483	0.7635071	0.8582938	0.8867299	0.8767773
6	0.7635071	0.8582938	0.8867299	0.8767773	0.8421801
7	0.8582938	0.8867299	0.8767773	0.8421801	1.0000000
8	0.8867299	0.8767773	0.8421801	1.0000000	0.9966825
9	0.8767773	0.8421801	1.0000000	0.9966825	0.8938389
10	0.8421801	1.0000000	0.9966825	0.8938389	0.7649289
11	1.0000000	0.9966825	0.8938389	0.7649289	0.8246445

Separating Mydata3 into train and test set, since there are 4 inputs, the total number of observations will be 496. Several models will be creating trying different combinations of hidden layers, nodes, activation function and output (Appendix2).

```
train3 <- Mydata3[1:400,]
test3 <- Mydata3[401:496,]

set.seed(1234)
NN10 <- neuralnet(Output ~ Input1 + Input2 + Input3 + Input4, train3,
                  hidden=5, act.fct= "tanh", threshold = 0.01,
                  linear.output = F)

## Evaluating Model Performance
model10_results <- compute(NN10, test3[1:4])
predicted10 <- model10_results$net.result
cor(predicted10, test3$Output)
```

The correlation result below.

```
[,1]
[1,] 0.9522907
```

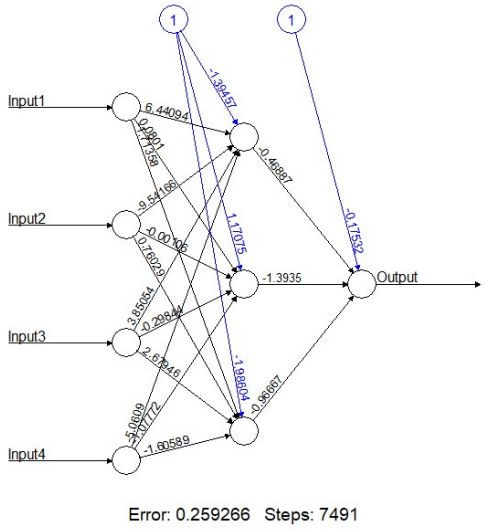

Figure 7- NN10 Neural Network 10

- NN10 1 hidden layer - 3 nodes, non- linear output, tanh activation function
- NN11 1 hidden layer - 5 nodes, non- linear output, logistic activation function
- NN12 1 hidden layer - 10 nodes, linear output, logistic activation function
- NN13 2 hidden layers - 9 and 4 nodes, linear output, tanh activation function
- NN14 2 hidden layers - 6 and 3 nodes, non- linear output, logistic activation function

Performance Indices

Having tried the different structures in neural networks, evaluation of their performance is required in order to select the optimal structure. There are multiple indices that measure the accuracy and performance of a neural network, usually these are the error indices as MAPE, MAE and RMSE.

The outputs of the models tried above, are normalised. In order to have the best comparison with the original dataset, the values should be de-normalised. Once de-normalised, the error for each model will be calculated based on the de-normalised predictions and the original data set.

```
Original_train <- ExchangeUSD[1:400,3]
Original_test <- (ExchangeUSD[401:500,3])

Output_min <- min(Original_train)
Output_max <- max(Original_train)
Original_test <- apply(Original_test, 2, as.numeric))
unnormalize <- function(x, min, max) {
  return( (max - min)*x + min )}
unnorm_data1 <- unnormalize(predicted1, Output_min, Output_max)

library(metrics)
rmse(Original_test[1:98,], unnorm_data1)
mape(Original_test[1:98,], unnorm_data1)
mae(Original_test[1:98,], unnorm_data1)
```

```
> rmse(Original_test[1:98,], unnorm_data1)
[1] 0.005444849
> mape(Original_test[1:98,], unnorm_data1)
[1] 0.003096893
> mae(Original_test[1:98,], unnorm_data1)
[1] 0.004080777
```

The results of de-normalising each prediction of each model and perform the performance indices, are shown in the table below.

NN1	RSME	MAPE	MAE	COR
NN2	0.00520	0.00293	0.00385	0.95251
NN3	0.00549	0.00308	0.00406	0.95305
NN4	0.00574	0.00323	0.00425	0.95266
NN5	0.00816	0.00460	0.00606	0.95229
NN6	0.00854	0.00484	0.00638	0.95159
NN7	0.00840	0.00474	0.00626	0.95259
NN8	0.00840	0.00474	0.00626	0.95055
NN9	0.00843	0.00461	0.00607	0.94554
NN10	0.01034	0.00618	0.00815	0.95028
NN11	0.01061	0.00633	0.00835	0.95160
NN12	0.00996	0.00585	0.00771	0.95002
NN13	0.01026	0.00596	0.00785	0.95110
NN14	0.01043	0.00617	0.00813	0.95213

From the above table, several indices are taken into consideration, however for a time series prediction evaluation, the best index to evaluate the performance is MAPE. It can also be considered as the accuracy of the model, since multiplied by 100, it gives the accuracy percentage of the model.

it is observed that NN2 and NN3 have the lowest MAPE, therefore the highest accuracy. Their structure is as follows:

NN2 with 1 hidden layer, 7 nodes, non - linear output, tanh activation function
NN3 with 2 hidden layers - 4 and 2 nodes, non - linear output, logistic activation function

Comparing the two models, it is observed that NN2 has the highest accuracy with 99.997% while NN3 performs with 99.996% accuracy. The performance is very close; however, the simpler structure is preferred as it is more cost-effective, requires less parameters and steps.

For the time-series forecast graph, the predicted data of model 2, are being exported to excel.

```
finalresults <- cbind.data.frame(Original_test[1:99,], unnorm_data1)
data.frame(finalresults)


write.table(finalresults, file =
"C:/Users/Vicky/Desktop/DM&ML/CW1/finalresults.csv", sep=",", row.names = F)
```

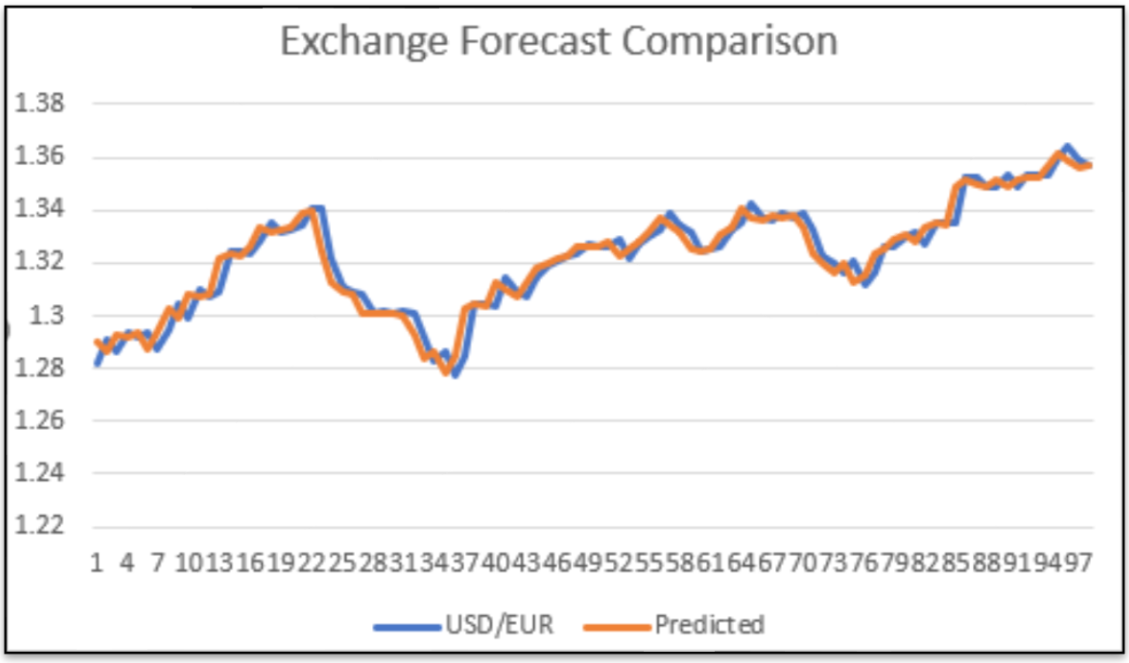

Figure 8 - Exchange Forecast Comparison

3rd Objective Support Vector Regression (SVR)

Another method to predict values, also very fit for time-series is the Support Vector Regression (SVR). By the utilisation of support vectors machine (SVM), the model will predict the future currency exchange, again based on a training and test set.

The number of inputs in this method is also a problem that needs to be solved, as there is no easy way to indicate the optimal number of inputs and the best combination of parameters, in a dataset as large as the Exchange one.

Similarly with the above, a various combination if inputs will be used, as well as a combination of parameters, such as the kernel, gamma, epsilon and cost. The kernel will define the dimension in which the data will be transformed in high dimensional, in order be classified. Gamma is a parameter of the non-linear kernel that defines how spread the datapoints will be from the decision area. The parameter epsilon it the ε-intensive loss function that defines the number of support vectors and the cost is the function that controls the training errors and margins. Using the same normalised dataset as the neural network models, the various models and different combinations are as follow.

SVR With Two Inputs

Model 1 - Linear Kernel

```
library(e1071)
train1 <- Mydata1[1:400,]
test1 <- Mydata1[401:498,]
model1 <- svm(Output ~ Input1 + Input2,data=train1,kernel='linear',
              epsilon=1,cost=0.1)
preds1 <- predict(model1,test1)
summary(model1)
```

```
Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  linear
       cost:  0.1
      gamma:  0.5
    epsilon:  1


Number of Support Vectors:  2
```

The accuracy indices for each model are the following, comparing the errors of the predicted test results and the original test data.

```
library(metrics)
rmse(Original_test[1:98,], preds1)
mape(Original_test[1:98,], preds1)
mae(Original_test[1:98,], preds1)
```

```
> rmse(Original_test[1:98,], preds1)
[1] 0.7512834
> mape(Original_test[1:98,], preds1)
[1] 0.5673592
> mae(Original_test[1:98,], preds1)
[1] 0.749138
```

Several combinations of epsilon and cost parameters are being used. The results are displayed in the table below.

Epsilon	Cost	MAPE	SVM
0	1	0.5830430	400
0.1	1	0.5844466	221
0.2	1	0.5873147	117
0.5	1	0.5822327	9
1	1	0.5673592	2

Epsilon	Cost	MAPE	SVM
1	0.01	0.6212902	14
1	0.1	0.5673592	2
1	1	0.5673592	2
1	10	0.5673592	2
1	100	0.5673592	2
1	500	0.5673592	2

Table 6 - Model 1 SVM Parameters Table

Based on the above table, by keeping cost with value 1 and assigning different values for epsilon, the best performance (based on the smallest MAPE - the mean absolute percentage error) is the value 1.0 for epsilon. Keeping epsilon in 1.0 and changing the parameters of cost, it is observed that the MAPE and number of SVM does not change at all except for the value 0.01. By selecting the smallest value for cost that gives the smallest error (0.1), once again the value of epsilon has been changed in order to find the optimal combination. The observations have not been recorded, since no other combination was giving a better result than epsilon = 1 and cost = 0.1.

Model 2 - Radial Kernel

```
model2 <- svm(Output ~ Input1 +
Input2,data=train1,kernel='radial',epsilon=0.01,
              gamma=0.05,cost=10)
preds2 <- predict(model2,test1)
summary(model2)
```

```
Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  10
      gamma:  0.05
    epsilon:  0.01


Number of Support Vectors:  387
```

The accuracy indices for each model are the following, comparing the errors of the predicted test results and the original test data.

```
rmse(Original_test[1:98,], preds2)
mape(Original_test[1:98,], preds2)
mae(Original_test[1:98,], preds2)
```

```
> rmse(Original_test[1:98,], preds2)
[1] 0.7718748
> mape(Original_test[1:98,], preds2)
[1] 0.5821041
> mae(Original_test[1:98,], preds2)
[1] 0.7684351
```

Several combinations of epsilon, gamma and cost parameters are being used. The results are displayed in the table below.

Epsilon	Gamma	Cost	MAPE	SVM
0	1	1	0.5839471	400
0.01	1	1	0.5835952	385
0.05	1	1	0.5839510	311
0.5	1	1	0.5876474	17
1	1	1	0.6078077	9
2	1	1	0.6058228	4
Epsilon	Gamma	Cost	MAPE	SVM
0.01	0	1	0.6486430	398
0.01	0.01	1	0.5849444	378
0.01	0.05	1	0.5824911	382
0.01	0.5	1	0.5838183	382
0.01	1	1	0.5835952	385
0.01	10	1	0.5839569	384
Epsilon	Gamma	Cost	MAPE	SVM
0.01	0.05	0.5	0.5832379	385
0.01	0.05	1	0.5824911	382
0.01	0.05	5	0.5823137	385
0.01	0.05	10	0.5821041	387
0.01	0.05	100	0.5824426	382
0.01	0.05	500	0.5826282	380

Table 7 -- Model 2 SVM Parameters Table

Based on the above table, by keeping cost and gamma with value 1 and assigning different values for epsilon, the best performance (based on the smallest MAPE - the mean absolute percentage error) is the value 0.01 for epsilon. Keeping epsilon in 0.01 and changing the parameters of gamma the best performance is when gamma has the value 0.05. Respectively, the cost has taken several values and the best performance is when cost is 10. By trying to re-assign values on gamma and epsilon keeping the cost at 10, no other combination was giving a better result than epsilon 0.01, gamma 0.05 and cost 10. The observations have not been recorded, since the selection of the best combination of parameters did not change.

SVR With Three Inputs

For a model with 3 inputs, the dataset that will be used will be Mydata2.

Model 3- Linear Kernel

```
train2 <- Mydata2[1:400,]
test2 <- Mydata2[401:497,]

model3 <- svm(formula = Output ~ Input1 + Input2 + Input3, data = train2,
              kernel = "linear", epsilon = 1, cost = 0.1)
preds3 <- predict(model3,test2)
summary(model3)
```

```
Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  linear
       cost:  0.1
      gamma:  0.3333333
    epsilon:  1

Number of Support Vectors:  2
```

The accuracy indices for each model are the following, comparing the errors of the predicted test results and the original test data.

```
rmse(Original_test[1:97,], preds3)
mape(Original_test[1:97,], preds3)
mae(Original_test[1:97,], preds3)
```

```
> rmse(Original_test[1:97,], preds3)
[1] 0.7604905
> mape(Original_test[1:97,], preds3)
[1] 0.5744554
> mae(Original_test[1:97,], preds3)
[1] 0.7583465
```

Several combinations of epsilon and cost parameters are being used. The results are displayed in the table below.

Epsilon	Cost	MAPE	SVM
0	1	0.5810664	400
0.1	1	0.5831393	223
0.2	1	0.5861171	121
0.5	1	0.5808276	9
1	1	0.5744554	2
2	1	0.5973345	2
Epsilon	Cost	MAPE	SVM
1	0.1	0.5744554	2
1	1	0.5744554	2
1	10	0.5744554	2
1	50	0.5744554	2
1	100	0.5744554	2
1	500	0.5744554	2

Table 8 - Model 3 SVM Parameters Table

Based on the above table, by keeping cost with value 1 and assigning different values for epsilon, the best performance (based on the smallest MAPE - the mean absolute percentage error) is the value 1.0 for epsilon. Keeping epsilon in 1.0 and changing the parameters of cost, it is observed that the MAPE and number of SVM does not change at all except for the value 0.01. By selecting the smallest value for cost that gives the smallest error (0.1), once again the value of epsilon has been changed in order to find the optimal combination. The observations have not been recorded, since no other combination was giving a better result than epsilon 1 and cost 0.1.

Model 4 – Radial Kernel

```
model4 <- svm(Output ~ Input1 + Input2 + Input3,data=train2,kernel='radial',
              epsilon=0.5, gamma=0.05,cost=3)
preds4 <- predict(model4,test2)
summary(model4)
```

```
Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  3
      gamma:  0.05
    epsilon:  0.5


Number of Support Vectors:  15
```

The accuracy indices for each model are the following, comparing the errors of the predicted test results and the original test data.

```
rmse(Original_test[1:97,], preds4)
mape(Original_test[1:97,], preds4)
mae(Original_test[1:97,], preds4)
```

```
rmse(Original_test[1:97,], preds4)
[1] 0.7689869
> mape(Original_test[1:97,], preds4)
[1] 0.5797752
> mae(Original_test[1:97,], preds4)
[1] 0.7651281
```

Several combinations of epsilon and cost parameters are being used. The results are displayed in the table below.

Epsilon	Gamma	Cost	MAPE	SVM
0	1	1	0.5835041	400
0.01	1	1	0.5829804	386
0.05	1	1	0.5841113	329
0.5	1	1	0.5847100	25
1	1	1	0.6063438	12
2	1	1	0.6136529	4
Epsilon	Gamma	Cost	MAPE	SVM
0.01	0	1	0.6493737	396
0.01	0.01	1	0.5835059	382
0.01	0.05	1	0.5812704	380
0.01	0.5	1	0.5823397	387
0.01	1	1	0.5829804	396
0.01	10	1	0.5853538	72
Epsilon	Gamma	Cost	MAPE	SVM
0.01	0.05	0.5	0.5825022	378
0.01	0.05	1	0.5809901	21
0.01	0.05	5	0.5801341	14
0.01	0.05	10	0.5821299	13
0.01	0.05	100	0.5848474	13
0.01	0.05	3	0.5797752	15

Table 9 - - Model 4 SVM Parameters Table

Based on the above table, by keeping cost and gamma with value 1 and assigning different values for epsilon, the best performance (based on the smallest MAPE - the mean absolute percentage error) is the value 0.01 for epsilon. Keeping epsilon in 0.01 and changing the parameters of gamma the best performance is when gamma has the value 0.05. Respectively, the cost has taken several values and the best performance is when cost is 5. By trying to re-assign values on gamma and epsilon keeping the cost at 5, no other combination was giving a better result than epsilon 0.01, gamma 0.05 and cost 5. However, by changing the cost to 3, an even smaller MAPE is being observed, hence the combination epsilon 0.01, gamma 0.05 and cost 3, is considered optimal for this model.

SVR With Four Inputs

For a model with 4 inputs, the dataset that will be used will be Mydata3.

Model 5- Linear Kernel

```
train3 <- Mydata3[1:400,]
test3 <- Mydata3[401:496,]
str(train3)
```

```
                    epsilon = 1, cost = 0.1)
preds5 <- predict(model5,test3)
```

```
SVM-Type:  eps-regression
 SVM-Kernel:  linear
       cost:  0.1
      gamma:  0.25
    epsilon:  1


Number of Support Vectors:  2
```

The accuracy indices for each model are the following, comparing the errors of the predicted test results and the original test data.

```
rmse(Original_test[1:96,], preds5)
mape(Original_test[1:96,], preds5)
mae(Original_test[1:96,], preds5)
```

```
> rmse(Original_test[1:96,], preds5)
[1] 0.7661127
> mape(Original_test[1:96,], preds5)
[1] 0.5787831
> mae(Original_test[1:96,], preds5)
[1] 0.763865
```

Several combinations of epsilon and cost parameters are being used. The results are displayed in the table below.

Epsilon	Cost	MAPE	SVM
0.01	1	0.5793710	384
0.1	1	0.5810969	223
0.2	1	0.5843734	120
0.5	1	0.5829555	9
1	1	0.5787831	2
2	1	0.5974469	2

Epsilon	Cost	MAPE	SVM
1	0.05	0.5842379	4
1	0.1	0.5787831	2
1	1	0.5787831	2
1	10	0.5787831	2
1	100	0.5787831	2
1	500	0.5787831	2

Table 10 - - Model 5 SVM Parameters Table

Based on the above table, by keeping cost with value 1 and assigning different values for epsilon, the best performance (based on the smallest MAPE - the mean absolute percentage error) is the value 1.0 for epsilon. Keeping epsilon in 1.0 and changing the parameters of cost, it is observed that the MAPE and number of SVM does not change at all except for the value 0.01. By selecting the smallest value for cost (0.1) that gives the smallest error, once again the value of epsilon has been changed in order to find the optimal combination. The observations have not been recorded, since no other combination was giving a better result than epsilon = 1 and cost = 0.1.

Model 6 – Radial Kernel

```
model6 <- svm(formula = Output ~ Input1 + Input2 + Input3 + Input4,
              data = train3, kernel = "radial",
              epsilon = 0.05, gamma =0.01, cost =3)
preds6 <- predict(model6,test3)
summary(model6)
```

```
SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  3
      gamma:  0.01
    epsilon:  0.05


Number of Support Vectors:  308
```

The accuracy indices for each model are the following, comparing the errors of the predicted test results and the original test data.

```
rmse(Original_test[1:96,], preds6)
mape(Original_test[1:96,], preds6)
mae(Original_test[1:96,], preds6)
```

```
> rmse(Original_test[1:96,], preds6)
[1] 0.7684881
> mape(Original_test[1:96,], preds6)
[1] 0.5796216
> mae(Original_test[1:96,], preds6)
[1] 0.764869
```

Several combinations of epsilon and cost parameters are being used. The results are displayed in the table below.

Epsilon	Gamma	Cost	MAPE	SVM
0	1	1	0.58279030	400
0.01	1	1	0.58228590	382
0.05	1	1	0.58218600	329
0.5	1	1	0.58393950	28
1	1	1	0.60856150	17
2	1	1	0.61759390	6
Epsilon	Gamma	Cost	MAPE	SVM
0.05	0	1	0.6509615	380
0.05	0.01	1	0.5817130	316
0.05	0.05	1	0.5822464	308
0.05	0.5	1	0.5823715	320
0.05	1	1	0.5821860	329
0.05	10	1	0.5851546	336
Epsilon	Gamma	Cost	MAPE	SVM
0.05	0.01	0.01	0.6381686	370
0.05	0.01	0.5	0.5839252	324
0.05	0.01	1	0.5817130	316
0.05	0.01	5	0.5799461	311
0.05	0.01	10	0.5803757	308
0.05	0.01	100	0.5806640	314
0.05	0.01	3	0.5796216	308

Table 11 - - Model 6 SVM Parameters Table

Based on the above table, by keeping cost and gamma with value 1 and assigning different values for epsilon, the best performance (based on the smallest MAPE - the mean absolute percentage error) is the value 0.01 for epsilon. Keeping epsilon in 0.05 and changing the parameters of gamma the best performance is when gamma has the value 0.01. Respectively, the cost has taken several values and the best performance is when cost is 5. By trying to re-assign values on gamma and epsilon keeping the cost at 5, no other combination was giving a better result than epsilon 0.05, gamma 0.01 and cost 5. However, by changing the cost to 3, an even smaller MAPE is being observed, hence the combination epsilon 0.05, gamma 0.01 and cost 3, is considered optimal for this model.

Performance Indices

One of the general observations based on the above parameter testing is that at a non-linear kernel, the cost does not change a lot the performance of the model. Also, the radial kernels, were creating a lot of SVM that caused overfitting. Having the optimal combination of parameters from all 6 models, the performance evaluation indices are being applied in order to select the best performing model of all. The results are shown in the table below.

Stats	RSME	MAPE	MAE	SVM
Model 1	0.7512834	0.5673592	0.7491380	2
Model 2	0.7718748	0.5821041	0.7684351	387
Model 3	0.7604905	0.5744554	0.7583465	2
Model 4	0.7689869	0.5797752	0.7651281	15
Model 5	0.7661127	0.5787831	0.7638650	2
Model 6	0.7684881	0.5796216	0.7648690	308

Table 11 - Performance Indices of SVR Models

From the above table, based on the less error and best accuracy, the model that brings the best results is model 1. The marked in red results are the smallest in the table. The model with the smallest errors and less SVM is model 1. In the following scatterplot, is the graphic illustration of model 1.

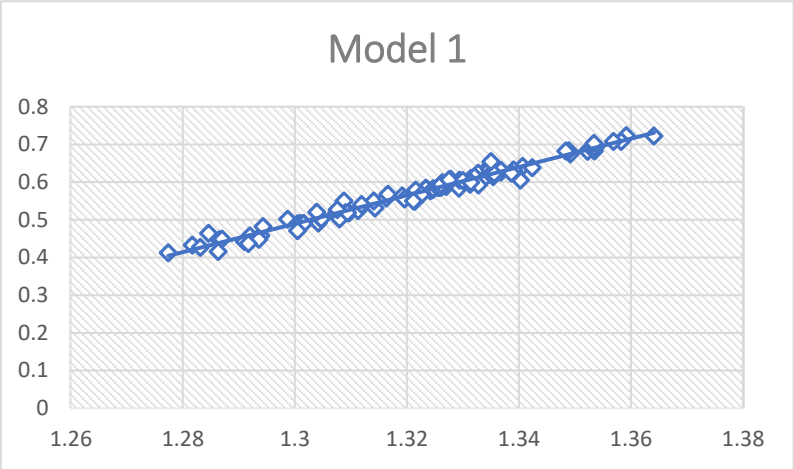

Figure 7 - Scatterplot of Model 1

Appendix 1

1st Objective (partitioning clustering)

```
# WHITEWINE COURSEWORK NO. 1


library(readxl)
Whitewine <- read_excel("C:/Users/Vicky/Desktop/DM&ML/CW1/Whitewine.xlsx")
View(Whitewine)
head(Whitewine)
str(Whitewine)
summary(Whitewine)


## Once we view the structure of our date, we need to detect potential outliers.
#  We need to remove the 12th column as it consist of


x=Whitewine[,-12]
summary(x)
boxplot(x)


install.packages("tidyr")
```