6BUIS001W Business Intelligence – Coursework 1

Vasileios Protopapas-W1735144

Contents

1st Objective (partitioning clustering)

Discussion of methodologies used for reducing the input dimensionality

One of the methodologies that are normally used to reduce input dimensionality are aspects such as missing values ratio. This is responsible for finding any data value that are missing from their data columns. As mentioned by Filter, L.V and Filter in regards to the missing values ratio "The goal, then, becomes to remove those data columns with too many missing values". Another method is the low variance filter which like its counterpart of MVR it's purpose is to also remove data from columns whose variance is too low, however this requires normalization. Another method would be the high correlation filter which take in values from data columns that are holding similar trends with specific pairs that hold this attribute being reduced to one. Principal Component Analysis is an important methodology as it purpose is to project all the data points of a column and only point it to a few specific components to reduce dimensionality but also ensure that no data variation is lost in the process also mentioned by (Karamizadeh, S., Abdullah, S.M., Manaf, A.A., Zamani, M. and Hooman, A., 2013.) on the overview on PCA they mentioned that its used to "convert a set of observations of possibly correlated variables into a set of values of linearly" helping to further show it's importance to lower input dimensionality.

Pre-processing tasks

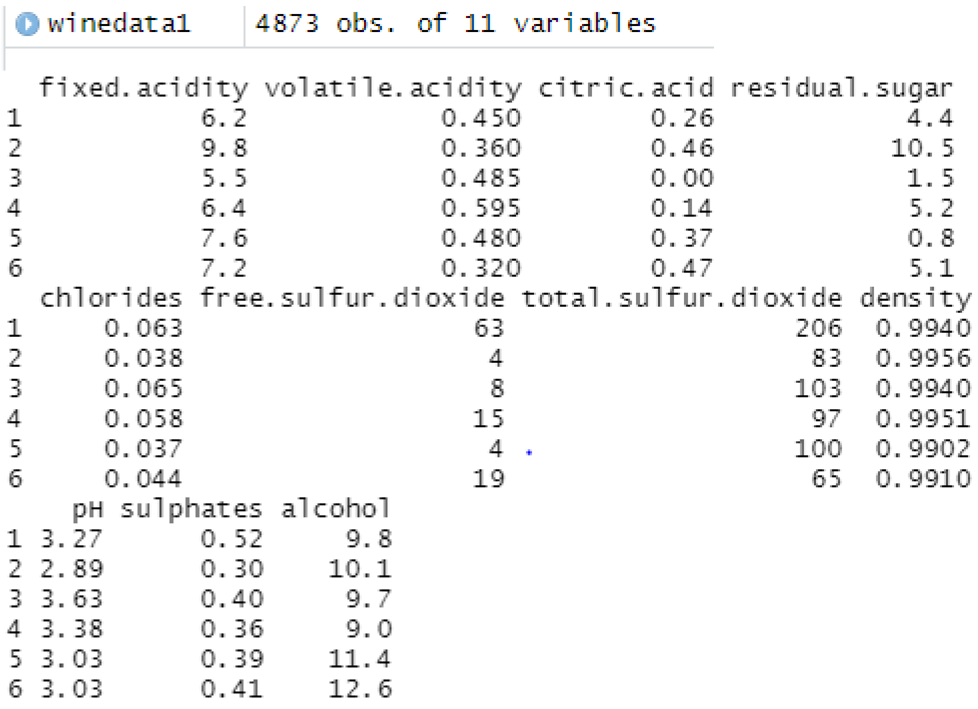
### Scaling

#Scaling and centralizing the data

winedata <- winedata

#Removes the 12th variable

winedata1 = winedata[,-12]



```
winedata1    4873 obs. of 11 variables

  fixed.acidity volatile.acidity citric.acid residual.sugar
1           6.2            0.450        0.26            4.4
2           9.8            0.360        0.46           10.5
3           5.5            0.485        0.00            1.5
4           6.4            0.595        0.14            5.2
5           7.6            0.480        0.37            0.8
6           7.2            0.320        0.47            5.1
  chlorides free.sulfur.dioxide total.sulfur.dioxide density
1     0.063                  63                  206  0.9940
2     0.038                   4                   83  0.9956
3     0.065                   8                  103  0.9940
4     0.058                  15                   97  0.9951
5     0.037                   4                  100  0.9902
6     0.044                  19                   65  0.9910
    pH sulphates alcohol
1 3.27      0.52     9.8
2 2.89      0.30    10.1
3 3.63      0.40     9.7
4 3.38      0.36     9.0
5 3.03      0.39    11.4
6 3.03      0.41    12.6
```

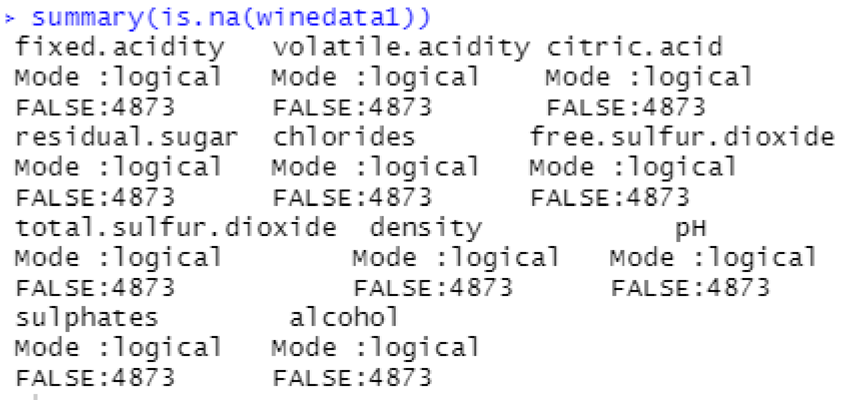
It is shown that the quality variable is no longer there as a result.

Outlier Removal

# checking for missing values

sum(is.na(winedata1))

summary(is.na(winedata1))



```
> summary(is.na(winedata1))
 fixed.acidity   volatile.acidity citric.acid    
 Mode :logical   Mode :logical    Mode :logical  
 FALSE:4873      FALSE:4873       FALSE:4873     
 residual.sugar  chlorides       free.sulfur.dioxide
 Mode :logical   Mode :logical   Mode :logical      
 FALSE:4873      FALSE:4873      FALSE:4873         
 total.sulfur.dioxide  density            pH         
 Mode :logical         Mode :logical   Mode :logical  
 FALSE:4873            FALSE:4873      FALSE:4873     
 sulphates        alcohol       
 Mode :logical   Mode :logical  
 FALSE:4873      FALSE:4873     
```

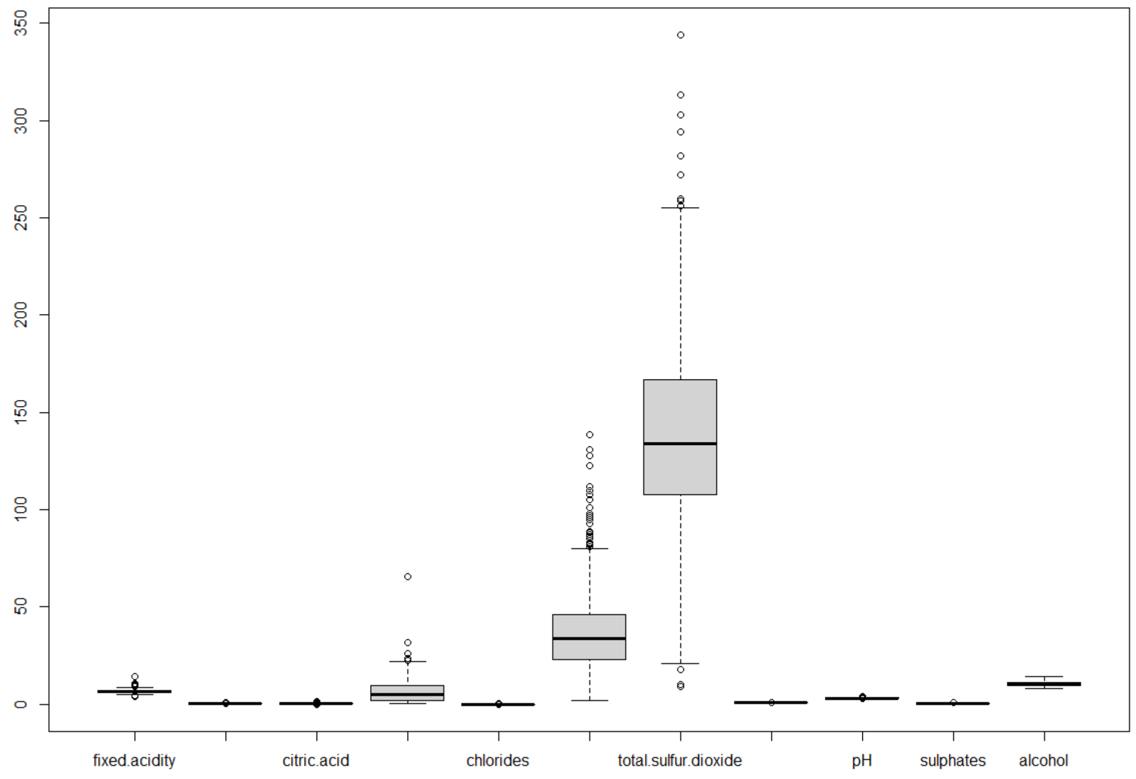
Here I have checked to see if there are any missing values with the “sum(is.na(data))” function. As shown in the image, this dataset contains no missing values

#Outlier detection by storing all numerical variables in an array structure

col = c("fixed.acidity","volatile.acidity","citric.acid" ,"residual.sugar","chlorides","free.sulfur.dioxide" ,"total.sulfur.dioxide","density","pH","sulphates", "alcohol")

#Using the BoxPlot function to detect any presence of outliers in the data columns.

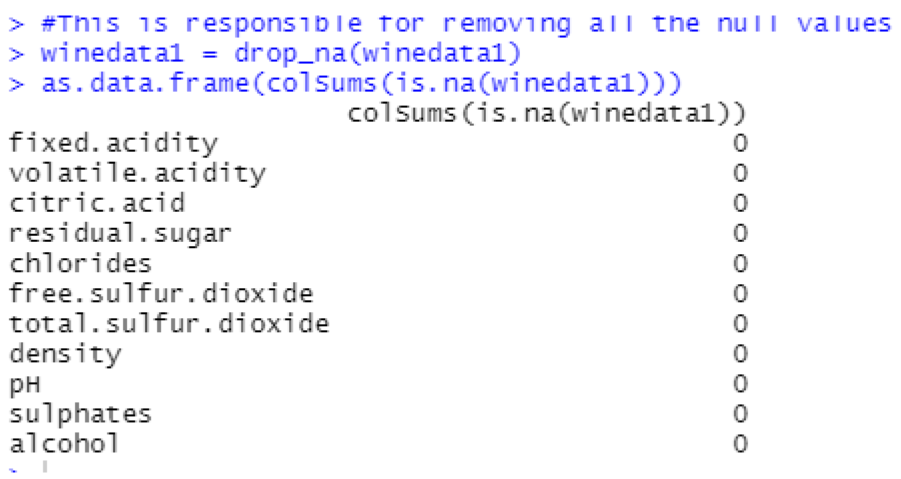
boxplot(winedata1[,c("fixed.acidity","volatile.acidity","citric.acid" ,"residual.sugar","chlorides","free.sulfur.dioxide" ,"total.sulfur.dioxide","density","pH","sulphates", "alcohol")])



From the visuals created by the boxplot this shows that all the variables contain outliers in the data values apart from the ‘alcohol’ variable

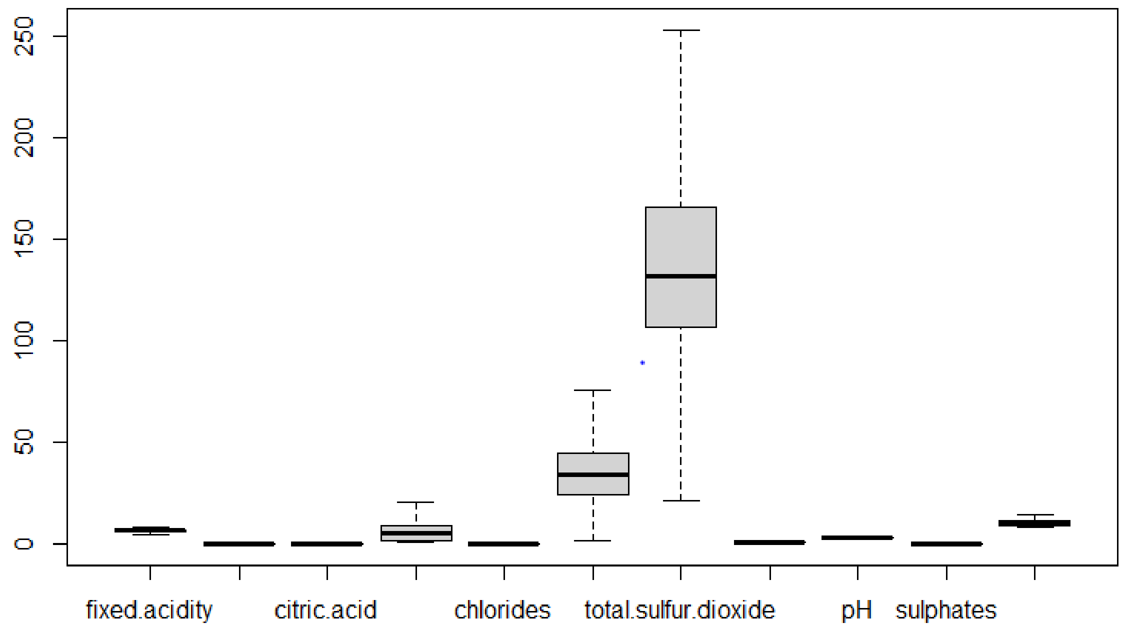
```
> #Checking whether the outliers have been replaced by null d
a or not
> as.data.frame(colSums(is.na(winedata1)))
                     colSums(is.na(winedata1))
fixed.acidity                              113
volatile.acidity                           182
citric.acid                                270
residual.sugar                               7
chlorides                                  206
free.sulfur.dioxide                         46
total.sulfur.dioxide                        15
density                                      5
pH                                          75
sulphates                                  124
alcohol                                      0
```

Here we can see that all the variables apart from alcohol that have outlier data



```
> #This is responsible for removing all the null values
> winedata1 = drop_na(winedata1)
> as.data.frame(colSums(is.na(winedata1)))
                     colSums(is.na(winedata1))
fixed.acidity                                0
volatile.acidity                             0
citric.acid                                  0
residual.sugar                               0
chlorides                                    0
free.sulfur.dioxide                          0
total.sulfur.dioxide                         0
density                                      0
pH                                           0
sulphates                                    0
alcohol                                      0
```

Here all the data has been turned to NULL and has been removed



After this operation all the outliers present on the first boxplot are now all gone

Defining all cluster centres

#Manually finding clusters 1

wss <- 0

for (i in 1:12){

 wss[i] <-

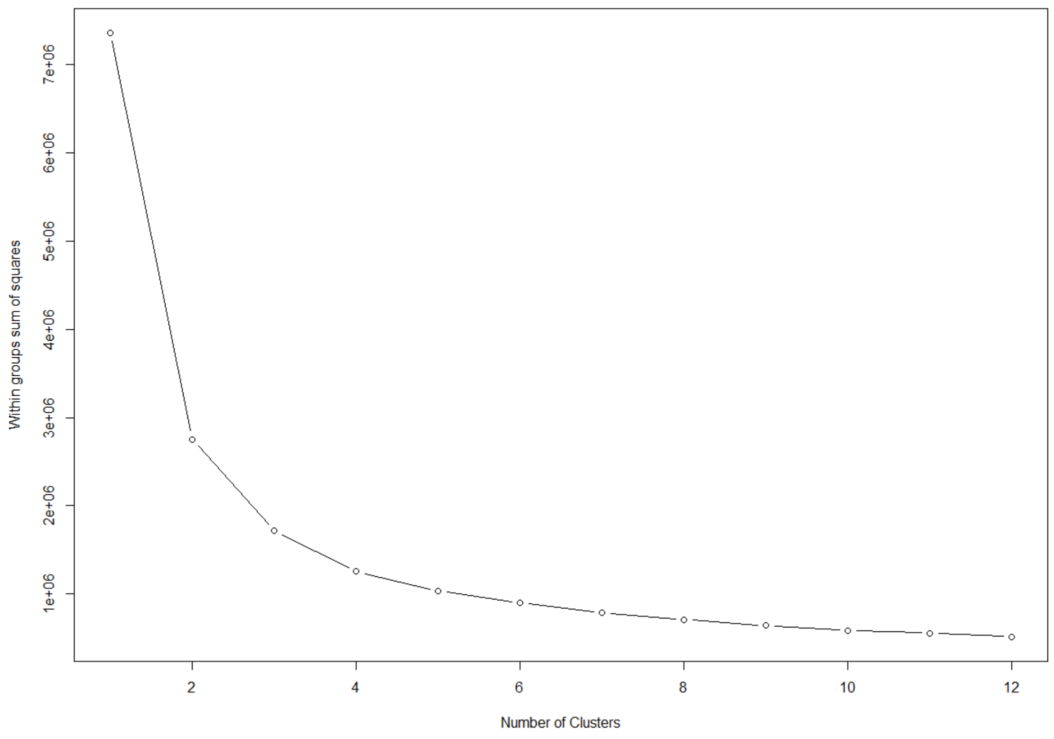
   sum(kmeans(winedata1, centers=i)$withinss)

}

plot(1:12,

    wss,

```
    type="b",     ### "b" for both####
    xlab="Number of Clusters",
    ylab="Within groups sum of squares")
```



This graph shows that k=3 is the best possible cluster for the data

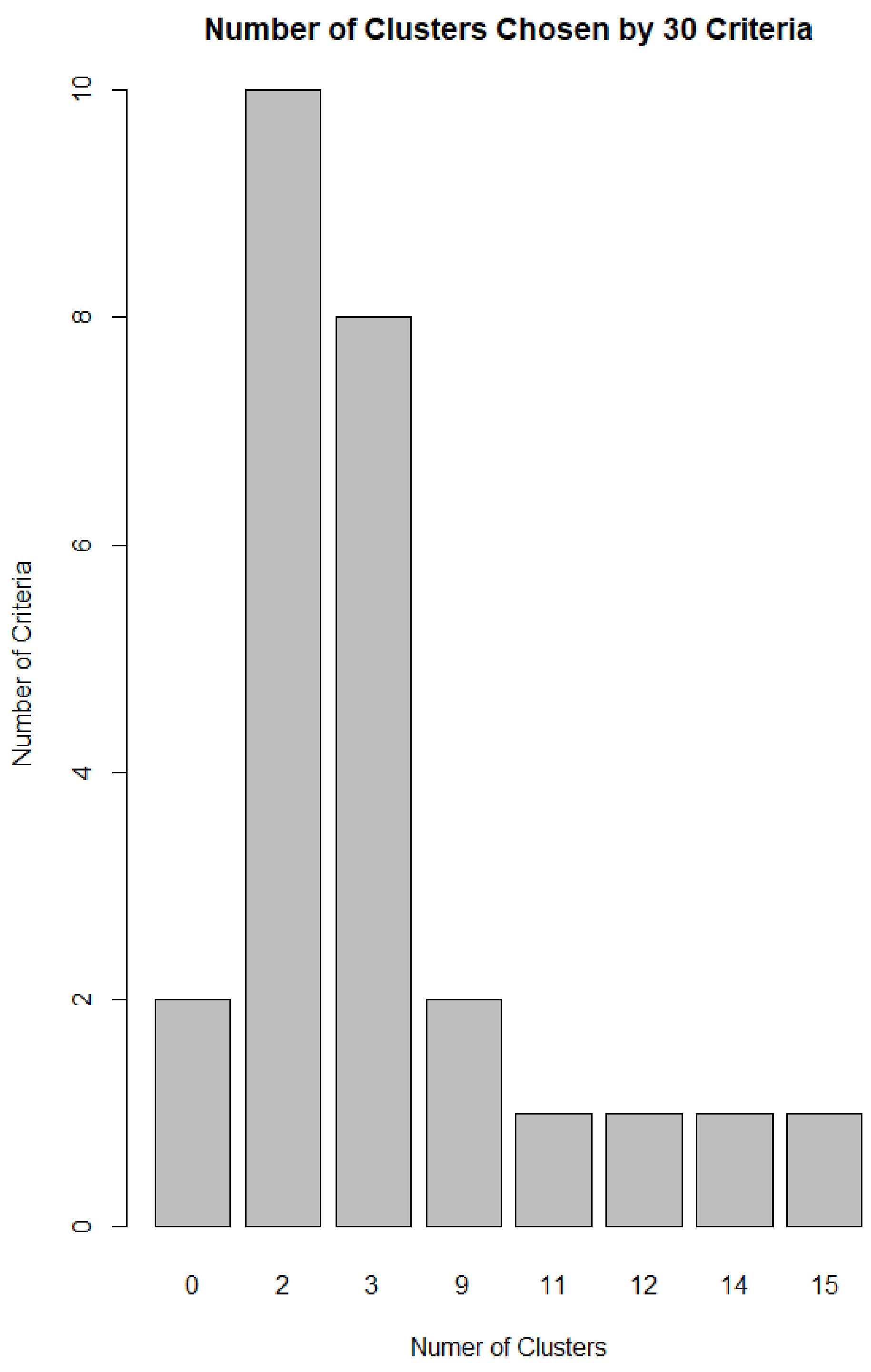
```
#Manually finding clusters in a larger criteria
set.seed(1234)

nc <- NbClust(winedata1,
          min.nc=2, max.nc=15,
          method="kmeans")

table(nc$Best.n[1,])

barplot(table(nc$Best.n[1,]),    # provide bar charts####
      xlab="Numer of Clusters",
      ylab="Number of Criteria",

      main="Number of Clusters Chosen by 30 Criteria")
```

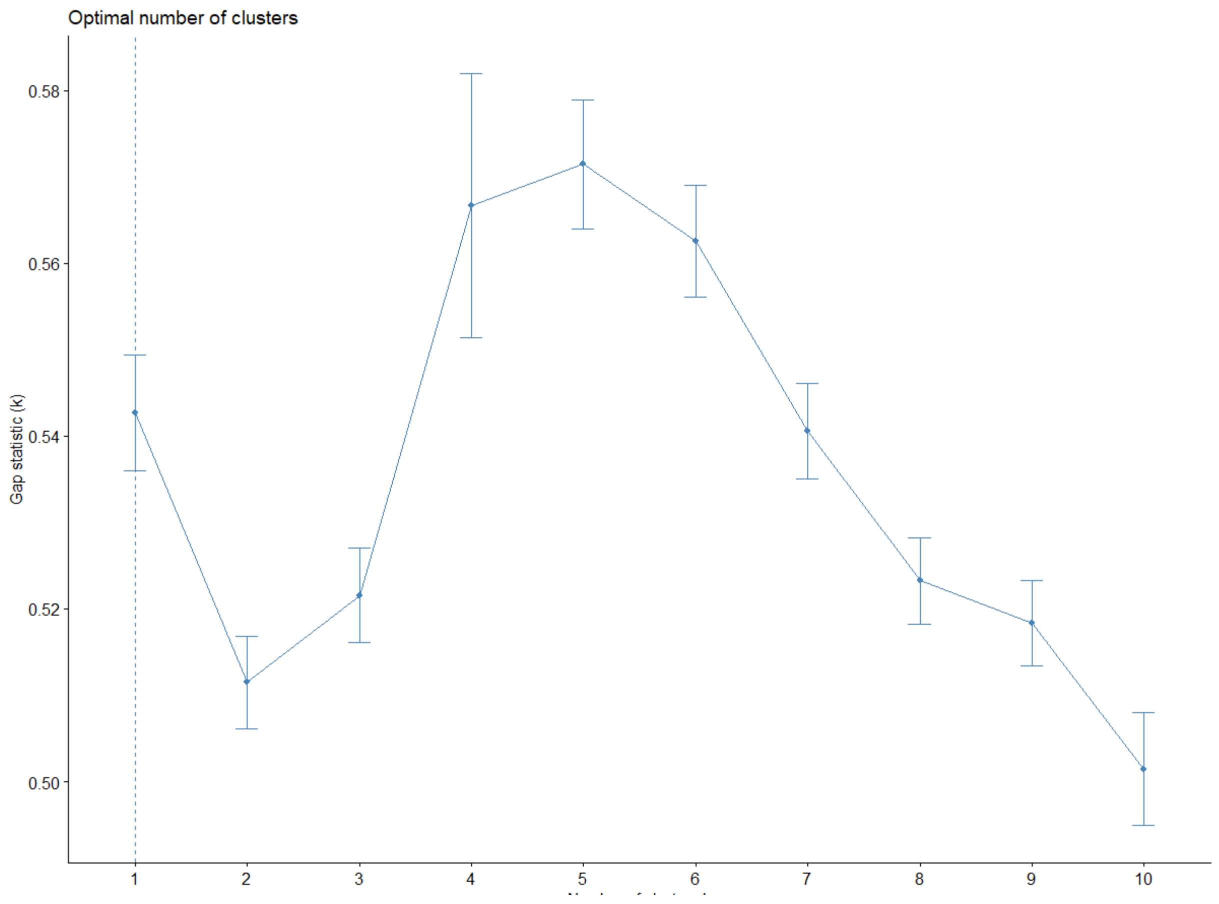


Here we can see that K=2 is the best number

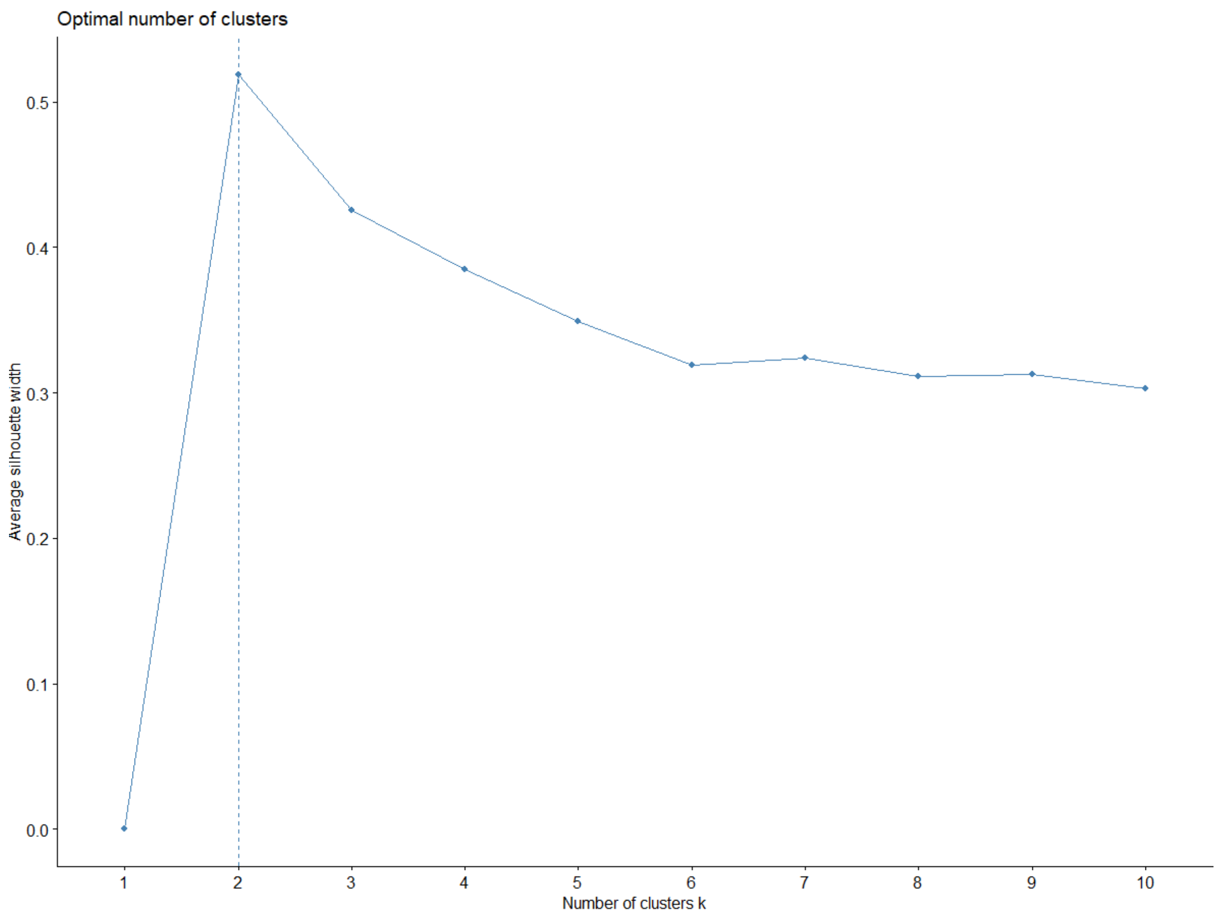
Here is the dominant winner for the 2 clusters

```
# Automatically finding clusters from 3 different methods
```

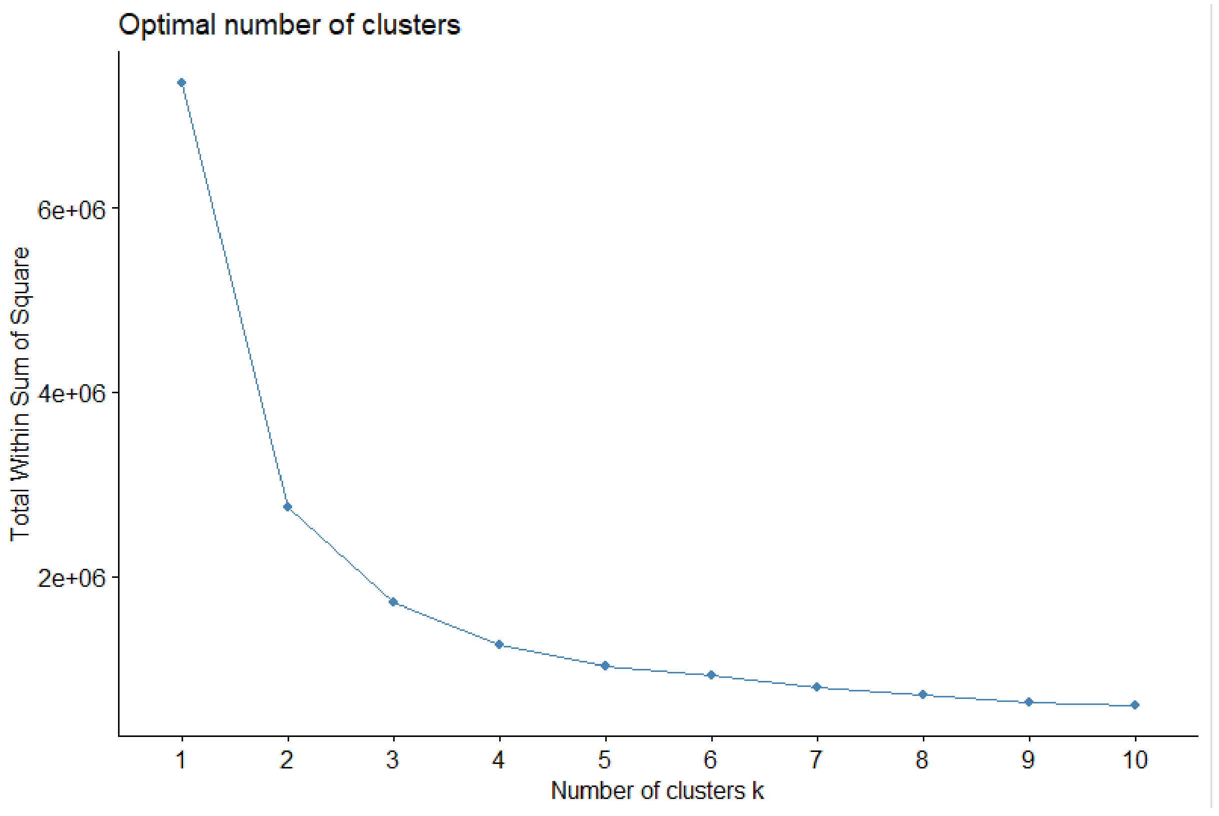
```
fviz_nbclust(winedata1[,-ncol(winedata1)], FUNcluster = kmeans, method = 'wss')
fviz_nbclust(winedata1[,-ncol(winedata1)], FUNcluster = kmeans, method = 'silhouette')
fviz_nbclust(winedata1[,-ncol(winedata1)], FUNcluster = kmeans, method = 'gap_stat')
#From all of the cluster analysis the best are 2,3 and 4
```



This graph shows that k=4 is the best possible cluster for the data



This graph shows that K=2 is the best choice in clustering



This graph shows that the optimal number of clusters would be k=2

Overall, from my findings I found out that 2,3 and 4 have all come up positively in all the manual and automatic tools.

K-Means Analysis & final "winner" cluster case with providing brief explanation

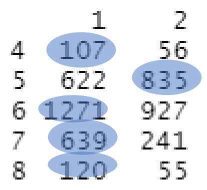
#We now fit wine data to K-Means with k = 2

fit.km2 <- kmeans(winedata, 2)

#The original data set will be used to be compared with the 19th column

confuseTable.km <- table(winedata$quality, fit.km2$cluster)

confuseTable.km



```
       1     2
  4  107    56
  5  622   835
  6 1271   927
  7  639   241
  8  120    55
```

The recall of sensitivity per class

R4	107/163 = 65.64%
R5	835/1457 = 57.30%
R6	1271/2198 = 57.82%
R7	639/880 = 72.61%
R8	120/175 = 68.57%

Precision per cluster

Cluster 1 2137/2759 = 77.45%

Cluster 2 835/2114 = 39.49%

Accuracy of Clustering

2972/4873 = 60.98%

#We now fit wine data to K-Means with k = 3

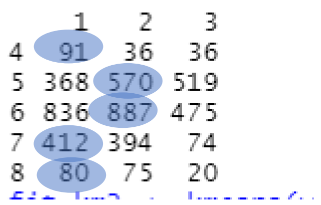
fit.km <- kmeans(winedata, 3)

#The original data set will be used to be compared with the 19th column

confuseTable.km <- table(winedata$quality, fit.km$cluster)

confuseTable.km

Here is the dominant winner for the 3 clusters



```
    1   2   3
4  91  36  36
5 368 570 519
6 836 887 475
7 412 394  74
8  80  75  20
```

The recall of sensitivity per class

R4	91/163 = 55.82%
R5	570/1457 = 39.12%
R6	887/2198 = 40.35%
R7	412/880 = 46.81%
R8	80/175 = 45.71%

Precision per cluster

Cluster 1 583/1787 = 32.62%

Cluster 2 1457/1962 = 74.26%

Cluster 3 0/1124 = 0%

Accuracy of Clustering

2040/4873 = 41.86%

#We now fit wine data to K-Means with k = 4

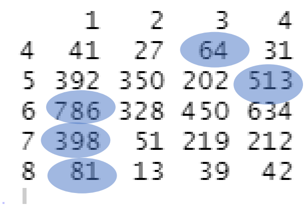
fit.km3 <- kmeans(winedata, 4)

#The original data set will be used to be compared with the 19th column

confuseTable.km <- table(winedata$quality, fit.km3$cluster)

confuseTable.km

Here is the dominant winner for the 4 clusters



```
     1   2   3   4
4   41  27  64  31
5  392 350 202 513
6  786 328 450 634
7  398  51 219 212
8   81  13  39  42
```

The recall of sensitivity per class

R4	64/163 = 39.26%
R5	513/1457 = 35.20%
R6	786/2198 = 35.75%
R7	398/880 = 45.22%
R8	81/175 = 46.28%

Precision per cluster

Cluster 1 1247/1680 = 74.22%

Cluster 2 0/769 = 0%

Cluster3 64/974 = 6.5%

Cluster 4 513/1432 = 35.82%

Accuracy of clustering

1824/4855 = 37.56%

Overall, through all the calculations, I found out that cluster group 2 has the most accuracy out of all with cluster 1 being the winner of all the previous calculations. Against the 12th column the produced outputs which usually ended up on turning to 1% means that the data having little agreement between the wine quality and the cluster solution only being 0.01 and as calculated by the Kmeans analysis.

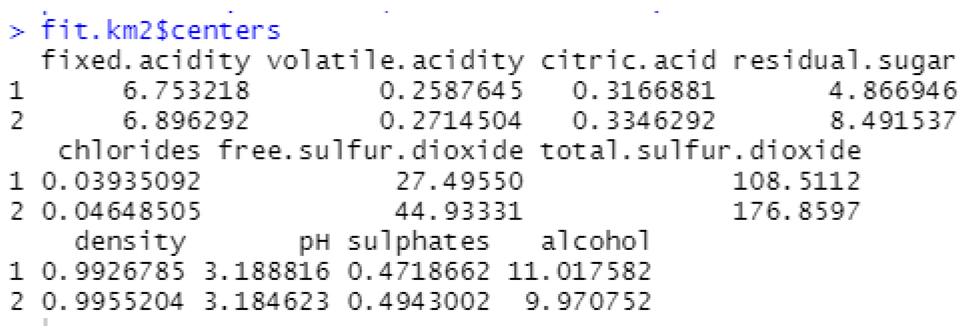
ARI

0.01126822

Coordinates for each centre clustering group

Here we have the coordinates for the centre clustering of k=2 group

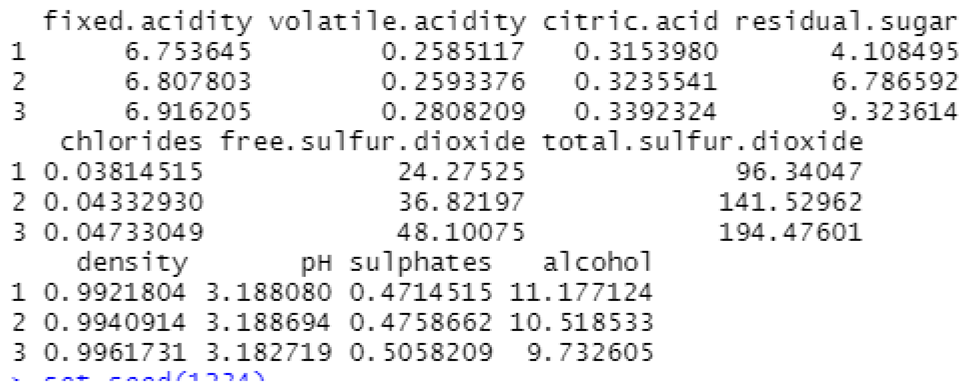
fit.km2$centers #finds the Kmeans value and centralizes the data



```
> fit.km2$centers
  fixed.acidity volatile.acidity citric.acid residual.sugar
1      6.753218        0.2587645   0.3166881       4.866946
2      6.896292        0.2714504   0.3346292       8.491537
   chlorides free.sulfur.dioxide total.sulfur.dioxide
1 0.03935092            27.49550             108.5112
2 0.04648505            44.93331             176.8597
    density       pH sulphates   alcohol
1 0.9926785 3.188816 0.4718662 11.017582
2 0.9955204 3.184623 0.4943002  9.970752
```

Here we have the coordinates for the centre clustering of k=3 group

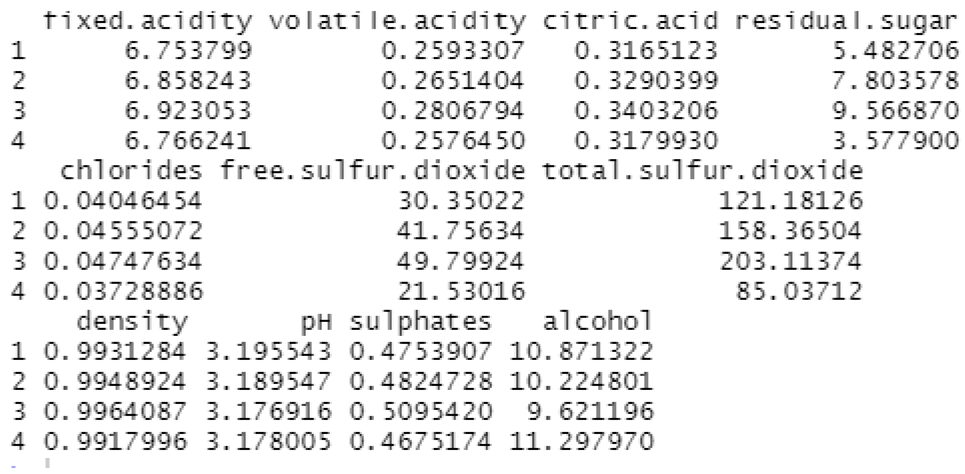
fit.km$centers #finds the Kmeans value and centralizes the data



```
  fixed.acidity volatile.acidity citric.acid residual.sugar
1      6.753645        0.2585117   0.3153980       4.108495
2      6.807803        0.2593376   0.3235541       6.786592
3      6.916205        0.2808209   0.3392324       9.323614
   chlorides free.sulfur.dioxide total.sulfur.dioxide
1 0.03814515            24.27525             96.34047
2 0.04332930            36.82197            141.52962
3 0.04733049            48.10075            194.47601
    density       pH sulphates   alcohol
1 0.9921804 3.188080 0.4714515 11.177124
2 0.9940914 3.188694 0.4758662 10.518533
3 0.9961731 3.182719 0.5058209  9.732605
```

Here we have the coordinates for the centre clustering of k=4 group

fit.km3$centers #finds the Kmeans value and centralizes the data



```
  fixed.acidity volatile.acidity citric.acid residual.sugar
1      6.753799        0.2593307   0.3165123       5.482706
2      6.858243        0.2651404   0.3290399       7.803578
3      6.923053        0.2806794   0.3403206       9.566870
4      6.766241        0.2576450   0.3179930       3.577900
   chlorides free.sulfur.dioxide total.sulfur.dioxide
1 0.04046454            30.35022            121.18126
2 0.04555072            41.75634            158.36504
3 0.04747634            49.79924            203.11374
4 0.03728886            21.53016             85.03712
    density       pH sulphates   alcohol
1 0.9931284 3.195543 0.4753907 10.871322
2 0.9948924 3.189547 0.4824728 10.224801
3 0.9964087 3.176916 0.5095420  9.621196
4 0.9917996 3.178005 0.4675174 11.297970
```

2nd Objective (MLP)

The various methods that can be used for defining input vectors in time-series problems can be using an autoregressive approach as it's main purpose is related to being observing previous data and forecast any future data based on "a linear model, where current period values are a sum of past outcomes multiplied by a numeric factor" (Mehandzhiyski, V.,2021). This means that the autoregressive approach as its use isn't just limited one the day before but like a moving average it can go back to the last 7 days.There are also moving averages which can be used to calculate any amount of time periods specified as long as they are in an average for example a 7 day week would be considered a moving average and in time series problems it also assists in the time series data as smooths out any shorter term fluctuations in order to predict any future numerical data.

References

- Zebari, R., Abdulazeez, A., Zeebaree, D., Zebari, D. and Saeed, J., 2020. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. *Journal of Applied Science and Technology Trends*, *1*(2), pp.56-70.
- Filter, L.V. and Filter, P.C.A., 2014. Seven Techniques for Dimensionality Reduction.
- KDnuggets. 2021. *Seven Techniques for Data Dimensionality Reduction - KDnuggets*. [online] Available at: <https://www.kdnuggets.com/2015/05/7-methods-data-dimensionality-reduction.html> [Accessed 15 November 2021].
- Karamizadeh, S., Abdullah, S.M., Manaf, A.A., Zamani, M. and Hooman, A., 2013. An overview of principal component analysis. Journal of Signal and Information Processing, 4(3B), p.173.
- Mehandzhiyski, V., 2021. What Is an Autoregressive Model? | 365 Data Science. [online] 365 Data Science. Available at: <https://365datascience.com/tutorials/time-series-analysis-tutorials/autoregressive-model/> [Accessed 10 November 2021].

Appendix

Part A

```
#Main r directory setup

setwd("~/Business intelligence cwk")

# Loading required packages

library(xlsx)

library(factoextra)

library(flexclust)

library(NbClust)

library(tidyr)

#Reading the winedata

winedata <- read.xlsx("Whitewine_v1.xlsx", sheetIndex = 1)
```

```
#Scaling and centralizing the data

winedata <- winedata

#Removes the 12th variable

winedata1 = winedata[,-12]


head(winedata1)


# checking for missing values

sum(is.na(winedata1))

summary(is.na(winedata1))



#Outlier detection by storing all numerical variables in an array structure

col = c("fixed.acidity","volatile.acidity","citric.acid" ,"residual.sugar","chlorides","free.sulfur.dioxide"
,"total.sulfur.dioxide","density","pH","sulphates", "alcohol")

#Using the BoxPlot function to detect any presence of outliers in the data columns.

boxplot(winedata1[,c("fixed.acidity","volatile.acidity","citric.acid"
,"residual.sugar","chlorides","free.sulfur.dioxide" ,"total.sulfur.dioxide","density","pH","sulphates",
"alcohol")])


# The box plot identifies all present outliers

# All the outlier data from the numeric variables will now be replaced by NULL values

for (x in c("fixed.acidity","volatile.acidity","citric.acid"
,"residual.sugar","chlorides","free.sulfur.dioxide"
,"total.sulfur.dioxide","density","pH","sulphates","alcohol"))

{

  value = winedata1[,x][winedata1[,x] %in% boxplot.stats(winedata1[,x])$out]

  winedata1[,x][winedata1[,x] %in% value] = NA

}


#Checking whether the outliers have been replaced by null data or not

as.data.frame(colSums(is.na(winedata1)))
```

```
#This is responsible for removing all the null values
winedata1 = drop_na(winedata1)
as.data.frame(colSums(is.na(winedata1)))

#Output a newly created boxplot without outliers
boxplot(winedata1)

#Manually finding clusters 1
wss <- 0
for (i in 1:12){
  wss[i] <-
    sum(kmeans(winedata1, centers=i)$withinss)
}
plot(1:12,
     wss,
     type="b",     ### "b" for both####
     xlab="Number of Clusters",
     ylab="Within groups sum of squares")



#Manually finding clusters in a larger criteria
set.seed(1234)

nc <- NbClust(winedata1,
              min.nc=2, max.nc=15,
              method="kmeans")

table(nc$Best.n[1,])
```

```
barplot(table(nc$Best.n[1,]),    # provide bar charts####
        xlab="Numer of Clusters",
        ylab="Number of Criteria",
        main="Number of Clusters Chosen by 30 Criteria")




# Automatically finding clusters from 3 different methods
fviz_nbclust(winedata1[,-ncol(winedata1)], FUNcluster = kmeans, method = 'wss')
fviz_nbclust(winedata1[,-ncol(winedata1)], FUNcluster = kmeans, method = 'silhouette')
fviz_nbclust(winedata1[,-ncol(winedata1)], FUNcluster = kmeans, method = 'gap_stat')
#From all of the cluster analysis the best are 2,3 and 4

#We now fit wine data to K-Means with k = 3
fit.km <- kmeans(winedata, 3)
#The original data set will be used to be compared with the 19th column
confuseTable.km <- table(winedata$quality, fit.km$cluster)
confuseTable.km
#Checking specific information relating to the centre clustering
fit.km$centers

#We now fit wine data to K-Means with k = 2
fit.km2 <- kmeans(winedata, 2)
#The original data set will be used to be compared with the 19th column
confuseTable.km <- table(winedata$quality, fit.km2$cluster)
confuseTable.km
#Checking specific information relating to the centre clustering
fit.km2$centers

#We now fit wine data to K-Means with k = 4
fit.km3 <- kmeans(winedata, 4)
```

```
#The original data set will be used to be compared with the 19th column
confuseTable.km <- table(winedata$quality, fit.km3$cluster)
confuseTable.km
#Checking specific information relating to the centre clustering
fit.km3$centers
```

Part B