Big Data Analytics Coursework

Student No: 2273633

1 Analysis 1

Question: Perform PCA analysis in `heptathlon` data. What important information can we exclude from this method by decreasing the dimensionality of data? Can we use our results in order to perform Cluster Analysis?

1.1 Data Visualization

For `heptatlon` data, we know there are no missing (NA) values. The data dimension is 25 rows (observations) and 9 columns (variables). This data set contains results from the heptathlon at the 1988 Olympic Games in Seoul. It was Jackie Joyner-Kersee who won the heptathlon. Heptathlon is a seven-event Olympic sport, which includes:

- Hurdles, High jump, Shot, 200 meters running, Long jump, Javelin ,800 meters running

Our data includes also the scores for each athlete.

Listing (1) Summary of heptatlon data

```
    hurdles         highjump          shot          run200m         longjump
Min.   :12.69   Min.   :1.500   Min.   :10.00   Min.   :22.56   Min.   :4.880
1st Qu.:13.47   1st Qu.:1.770   1st Qu.:12.32   1st Qu.:23.92   1st Qu.:6.050
Median :13.75   Median :1.800   Median :12.88   Median :24.83   Median :6.250
Mean   :13.84   Mean   :1.782   Mean   :13.12   Mean   :24.65   Mean   :6.152
3rd Qu.:14.07   3rd Qu.:1.830   3rd Qu.:14.20   3rd Qu.:25.23   3rd Qu.:6.370
Max.   :16.42   Max.   :1.860   Max.   :16.23   Max.   :26.61   Max.   :7.270
    javelin         run800m          score
Min.   :35.68   Min.   :124.2   Min.   :4566
1st Qu.:39.06   1st Qu.:132.2   1st Qu.:5746
Median :40.28   Median :134.7   Median :6137
Mean   :41.48   Mean   :136.1   Mean   :6091
3rd Qu.:44.54   3rd Qu.:138.5   3rd Qu.:6351
Max.   :47.50   Max.   :163.4   Max.   :7291
```

Some results are measured in seconds and others in meters. We transform the time to be the difference from the slowest time. $(\max(x) - x$, for x=hurdles, run 200m, run 800m) Furthermore, we standardize the data. Also, we observe from Figure 1 that there are correlations in some variables (hurdles, high jump, shot, 200m, 800m, long jump), for example between hurdles and high jump, the correlation is 0.81, which means that there is a strong linear relationship between these two variables. The correlation matrix for the variable is a 7×7 matrix and R confirmed to us that it is a full-rank matrix (rank =7), so, all the eigenvalues are strictly positive and all the eigenvectors linearly independent. The matrix is given below:

Listing (2) Correlation matrix

	hurdles	highjump	shot	run200m	longjump	javelin	run800m
hurdles	1.000000000	0.811402536	0.6513347	0.7737205	0.91213362	0.007762549	0.77925711
highjump	0.811402536	1.000000000	0.4407861	0.4876637	0.78244227	0.002153016	0.59116282
shot	0.651334688	0.440786140	1.0000000	0.6826704	0.74307300	0.268988837	0.41961957
run200m	0.773720543	0.487663685	0.6826704	1.0000000	0.81720530	0.333042722	0.61681006
longjump	0.912133617	0.782442273	0.7430730	0.8172053	1.00000000	0.067108409	0.69951116
javelin	0.007762549	0.002153016	0.2689888	0.3330427	0.06710841	1.000000000	−0.02004909
run800m	0.779257110	0.591162823	0.4196196	0.6168101	0.69951116	−0.020049088	1.00000000

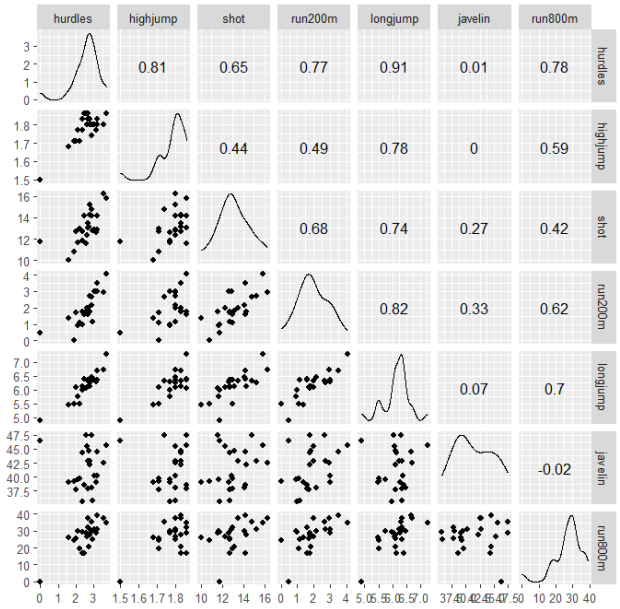

Figure (1) Correlation and density plot

1.2 PCA analysis

From the spectral theorem, any symmetric matrix is diagonalizable. Our correlation matrix is symmetric, so it is diagonalizable. Therefore, we can write our matrix (suppose the name of our correlation matrix is X) as:

$$X = P\Lambda P^T$$

P is the matrix of eigenvectors and Λ is the matrix of eigenvalues, which is diagonal.

We found that the eigenvalues and eigenvectors are:

Listing (3) PCA analysis

```
> # Eigenvalues:
> lamda <- eigen(X)$values
[1] 4.46027516 1.19432056 0.52101413 0.45716683 0.24526674 0.07295558 0.04900101
> # Eigenvectors:
```

	names	T1	T2	T3	T4	T5	T6	T7
1	hurdles	−0.4528710	0.15792058	−0.04514996	−0.02653873	0.09494792	0.78334101	−0.38024707
2	highjump	−0.3771992	0.24807386	−0.36777902	−0.67999172	−0.01879888	−0.09939981	0.43393114
3	shot	−0.3630725	−0.28940743	0.67618919	−0.12431725	−0.51165201	0.05085983	0.21762491
4	run200m	−0.4078950	−0.26038545	0.08359211	0.36106580	0.64983404	−0.02495639	0.45338483
5	longjump	−0.4562318	0.05587394	0.13931653	−0.11129249	0.18429810	−0.59020972	−0.61206388
6	javelin	−0.0754090	−0.84169212	−0.47156016	−0.12079924	−0.13510669	0.02724076	−0.17294667
7	run800m	−0.3749594	0.22448984	−0.39585671	0.60341130	−0.50432116	−0.15555520	0.09830963

Note that if we sum all the eigenvalues then the result will be 1. We will choose the right number of principal components for eigenvalues greater than the mean of eigenvalues ($\lambda_j \geq \bar{\lambda}$). But the mean of eigenvalues is $\frac{1}{p}$ where p is the number of variables. We observe that there are only two (2) eigenvalues greater than 1. Also, Figure 2 helps us to understand that choosing the first two (2) eigenvalues will lead to significant results.

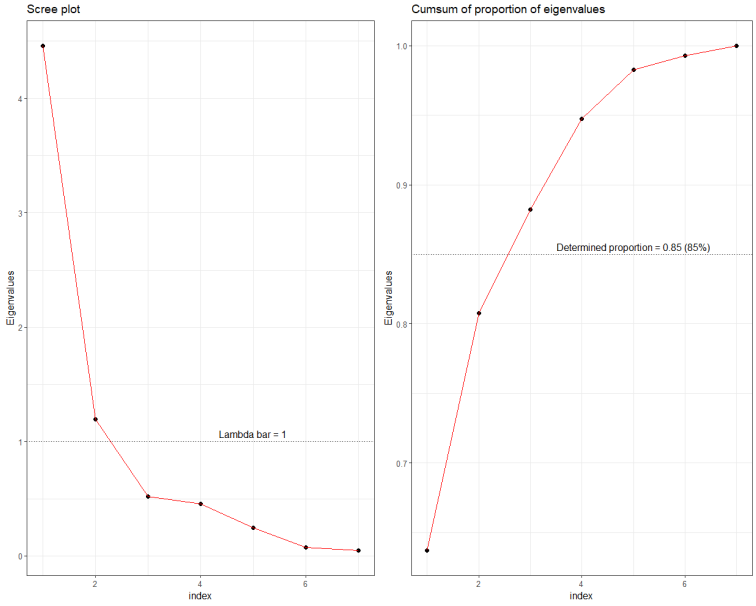

Figure (2) Scree plot (left) and cumsum plot (right)

The first two components can be considered to be the most significant since they contain 80.77994% of the total information of the data. According to theory scores of principal components can be calculated by multiplying the standardized data (25×7) with the eigenvector matrix (7×7).

$$\underbrace{Z}_{25\times 7} = \underbrace{X^*}_{25\times 7}\underbrace{T}_{7\times 7}$$

· Most loadings for the first component as shown in Listing 3 are important, except for the javelin. High jump, shot, long jump, and javelin have negative values, and hurdles, 200m run, and 800m run have positive values. Component 1, could describe athletic ability.

Component 2 has a strong negative loading on the javelin variable. So, Component 2 could describe the skill of throwing a javelin.

We demonstrate the scores (Z1, Z2).

Listing (4) Scores

	X	Z1	Z2
1	Joyner−Kersee (USA)	−4.121447626	−1.24240435
2	John (GDR)	−2.882185935	−0.52372600
3	Behmer (GDR)	−2.649633766	−0.67876243
24	Jeong−Mi (KOR)	2.970118607	0.95961101
25	Launa (PNG)	6.270021972	−2.83919926

Next, we show (Figure 3) a biplot, which is a plot of the two scores (Z1, Z2) including the loadings of variables and a plot of the two first eigenvectors.

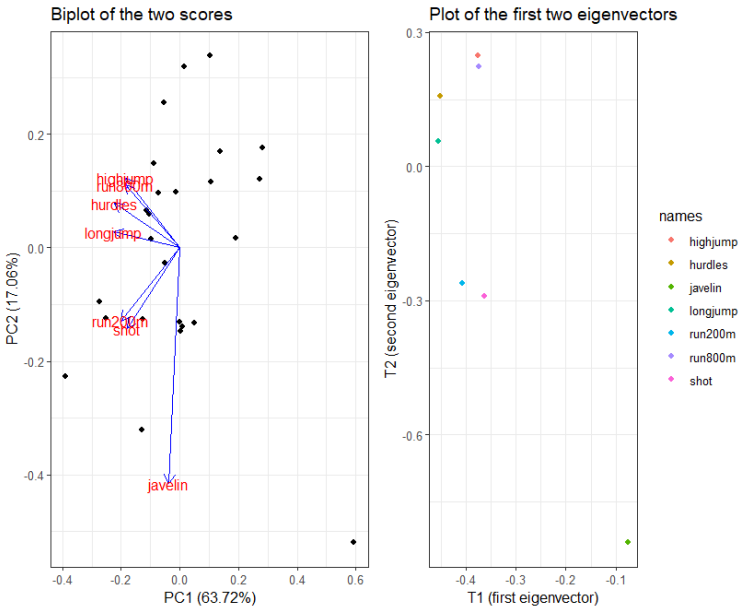

Figure (3) Left plot: Biplot, Right plot: plot of T1 and T2 eigenvectors

If we look at the Z1 score (athletic ability) Joyner-Kersee has the best score and Launa has the worst. PCA analysis stops here. We understood that PCA is a mathematical procedure that converts a data set of possibly correlated variables into a set of values of linearly uncorrelated variables, where they are called principal components.

1.3 K-means Cluster Analysis

Next, we took the two scores and tried to apply cluster analysis with the help of `cluster` and `factoextra` packages in R. Applying k-means cluster analysis we chose to work with $k = 2$ clusters, according to the Silhouette method (Figure 4)

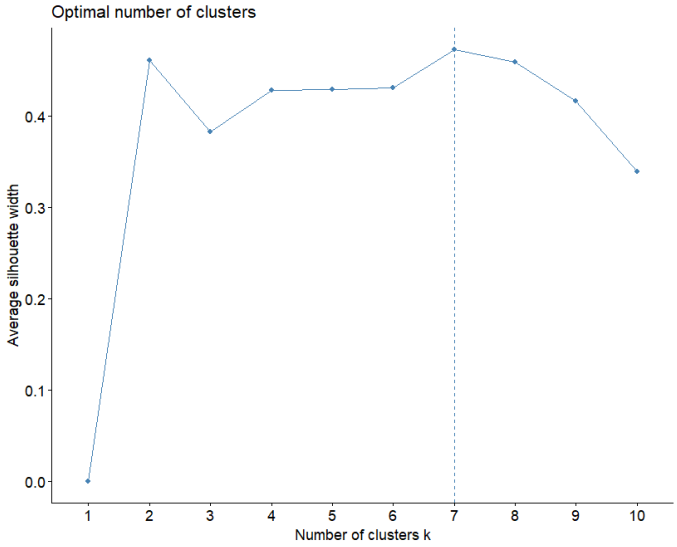

Figure (4) Silhouette method for choosing optimal k

Even if the optimal number is 7, k=2 is also a good choice, because the average silhouette width for k=2 is near k=7. In the end Figure 5 shows the clusters graphically. Note that in the left graph, there is an outlier, which is the athlete Launa. The right graph shows the clusters if we exclude athlete Launa from the analysis.

An important notation is how an outlier influences the analysis. Cluster 1 could be a cluster with very good performances and Cluster 2 could be a cluster with bad performances.

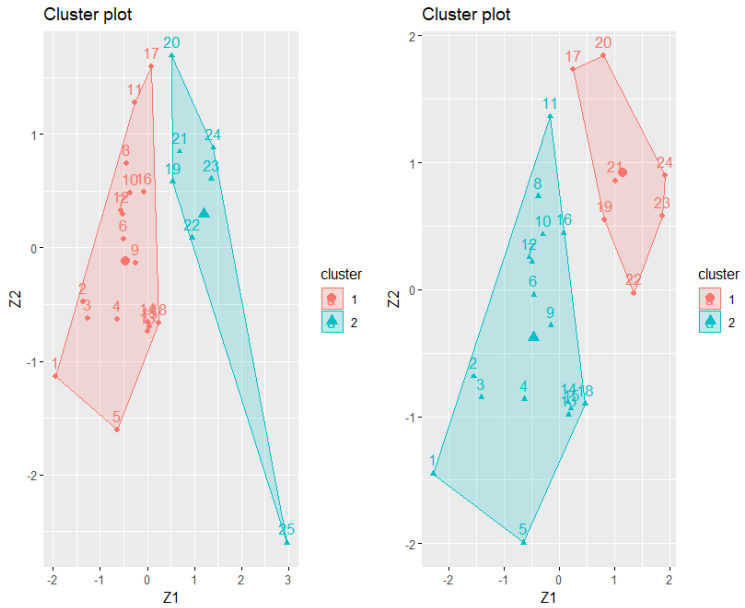

Figure (5) Cluster Analysis in graph

2 Analysis 2

Question: Can we use `protein` data and perform Cluster Analysis on them? Can we compare Agglomerative and K-means Cluster Analysis? Is there exist a method for choosing which method is better for this data set?

2.1 Data Visualization

Cluster analysis doesn't need to group data points into any predefined groups, which means that it is an unsupervised learning method. So, `protein` data is available for performing Cluster Analysis, using the help of `cluster` and `factoextra` packages in R. This data set contains 25 observations and 9 (numerical) variables (Red Meat, White Meat, Eggs, Milk, Fish, Cereals, Starch, Nuts, Fr.Veg). In the end, there are no missing values (NA).

Listing (5) Summary of protein data

```
> # No missing values:
  RedMeat WhiteMeat      Eggs      Milk      Fish   Cereals   Starch      Nuts   Fr.Veg
        0         0         0         0         0         0         0         0         0
> summary(df)
    RedMeat         WhiteMeat          Eggs             Milk            Fish      
 Min.   : 4.400   Min.   : 1.400   Min.   :0.500   Min.   : 4.90   Min.   : 0.200  
 1st Qu.: 7.800   1st Qu.: 4.900   1st Qu.:2.700   1st Qu.:11.10   1st Qu.: 2.100  
 Median : 9.500   Median : 7.800   Median :2.900   Median :17.60   Median : 3.400  
 Mean   : 9.828   Mean   : 7.896   Mean   :2.936   Mean   :17.11   Mean   : 4.284  
 3rd Qu.:10.600   3rd Qu.:10.800   3rd Qu.:3.700   3rd Qu.:23.30   3rd Qu.: 5.800  
 Max.   :18.000   Max.   :14.000   Max.   :4.700   Max.   :33.70   Max.   :14.200  
    Cereals          Starch           Nuts            Fr.Veg      
 Min.   :18.60   Min.   :0.600   Min.   :0.700   Min.   :1.400  
 1st Qu.:24.30   1st Qu.:3.100   1st Qu.:1.500   1st Qu.:2.900  
 Median :28.00   Median :4.700   Median :2.400   Median :3.800  
 Mean   :32.25   Mean   :4.276   Mean   :3.072   Mean   :4.136  
 3rd Qu.:40.10   3rd Qu.:5.700   3rd Qu.:4.700   3rd Qu.:4.900  
 Max.   :56.70   Max.   :6.500   Max.   :7.800   Max.   :7.900  
```

Next, we show a correlation-density plot (Figure 6), in which we observe significant correlations, between the variables, except for Fr. Veg and Starch, Eggs and Fr. Veg, Fish and Eggs, Fr. Veg and Eggs, Fr. Veg and White meat, Fr. Veg and Red meat, Fish and Red meat and Fr. Veg and Red meat.

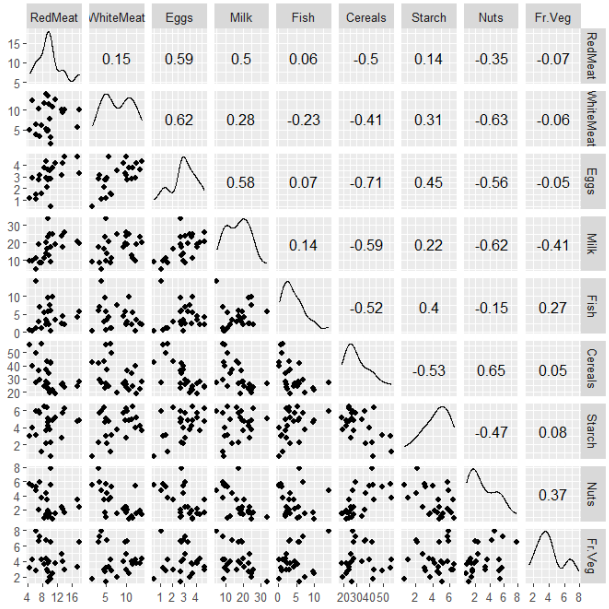

Figure (6) Correlation-density plot

Also, Figure 7 shows a box plot, in which we detect outliers. Our aim is to exclude them.

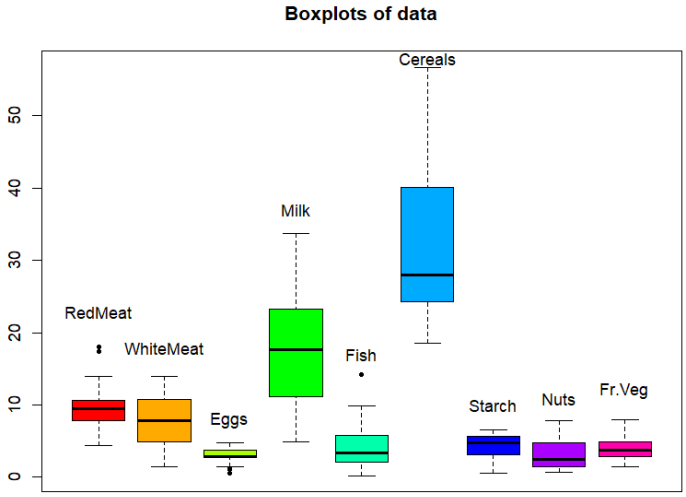

Figure (7) Box plot

Our goal is to compare Agglomerative Cluster Analysis (using the complete method) and the K-means Cluster Analysis method.

2.2 Agglomerative Complete method

We chose to work with a complete method. For the distances, we use the Euclidean distance.

Listing (6) Agglomerative Cluster Analysis of protein data

```
> clusters_4
     Austria      Belgium     Bulgaria Czechoslovakia      Denmark    E Germany
           1            1            2            3            1            3
     Finland       Greece      Hungary      Ireland        Italy  Netherlands
           1            4            2            1            4            1
      Norway       Poland      Romania        Spain       Sweden  Switzerland
           1            3            2            4            1            1
        USSR   W Germany
           3            1
```

The Cluster Dendrogram is shown in Figure 8.

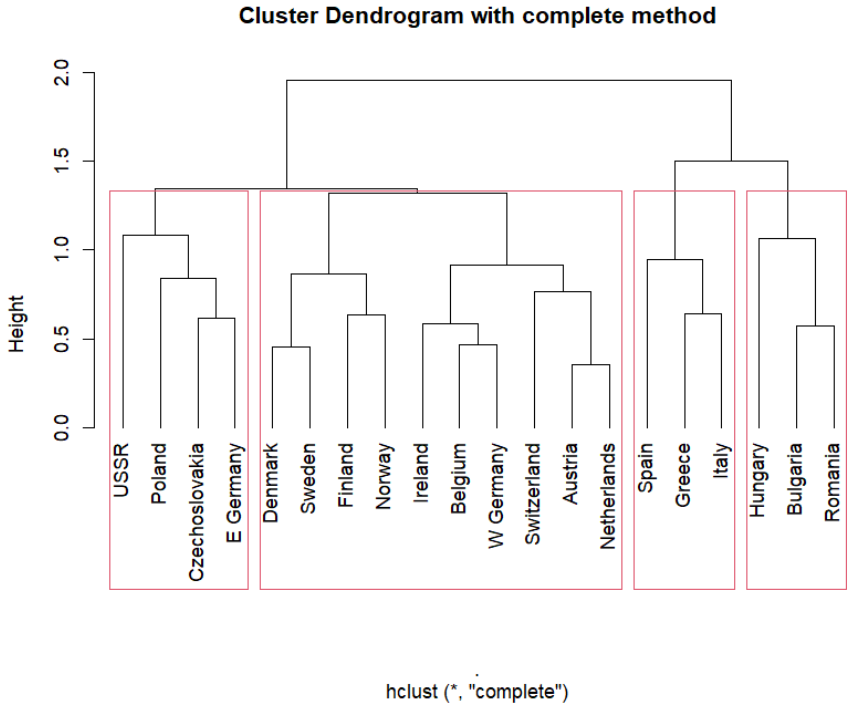

Figure (8) Cluster Dendrogram for k=4

2.3 K-means Cluster Analysis

We performed K-means Cluster Analysis with $k = 4$. The results are shown below:

Listing (7) K-means Cluster Analysis of protein data

```
> kmean_cluster_4
     Austria      Belgium     Bulgaria Czechoslovakia      Denmark    E Germany
           3            3            2            3            4            3
     Finland       Greece      Hungary      Ireland        Italy  Netherlands
           4            1            2            3            1            3
      Norway       Poland      Romania        Spain       Sweden  Switzerland
           4            3            2            1            4            3
        USSR   W Germany
           2            3
```

2.4 Evaluation of Clustering results

We use the Silhouette method in order to choose which method (agglomerative or k-means) is better.

According to Figure 9 the Silhouette scores are the same.

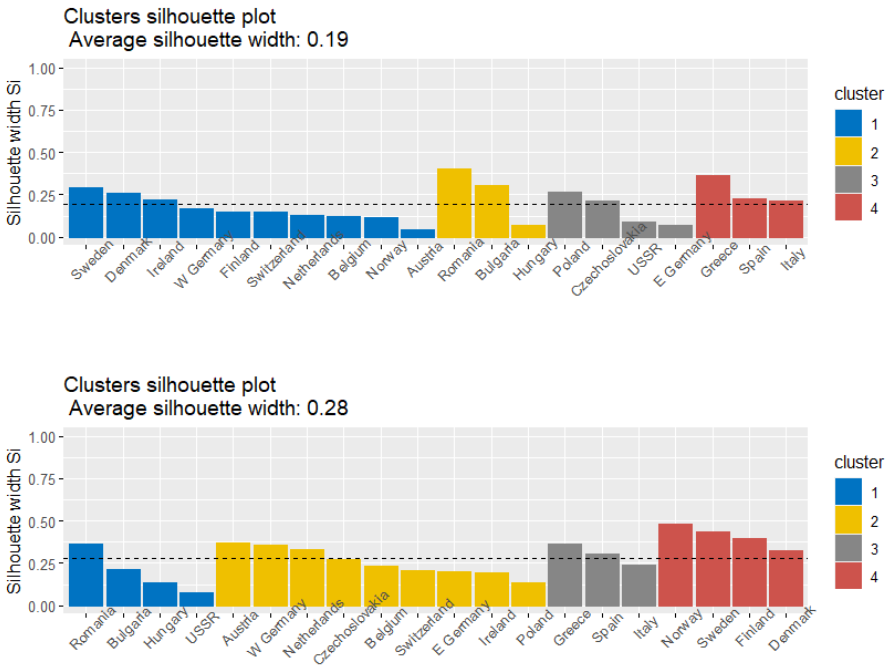

Figure (9) Silhouette plots, First plot: Agglomerative method, Second plot: K-Means method

We choose the method with the maximum silhouette score. So, K-Means method (0.28) is better method than Agglomerative Clustering (0.19) (complete method)

At the end, we would like to find an optimal number of clusters using the K-Means method. Silhouette method (Figure 10) suggests that $k = 3$.

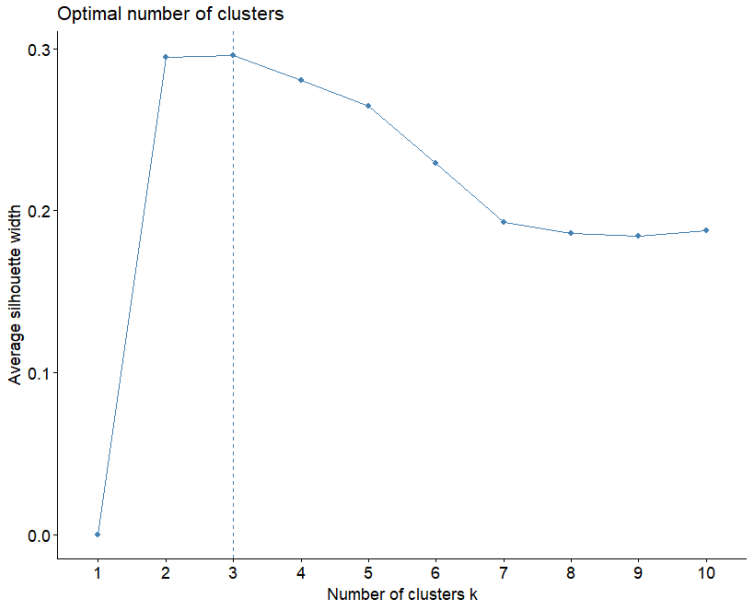

Figure (10) Silhouette method for choosing optimal number of clusters

Also, plotting for different k values (Figure 11) we conduct that k = 3 is an optimal number of partitions.

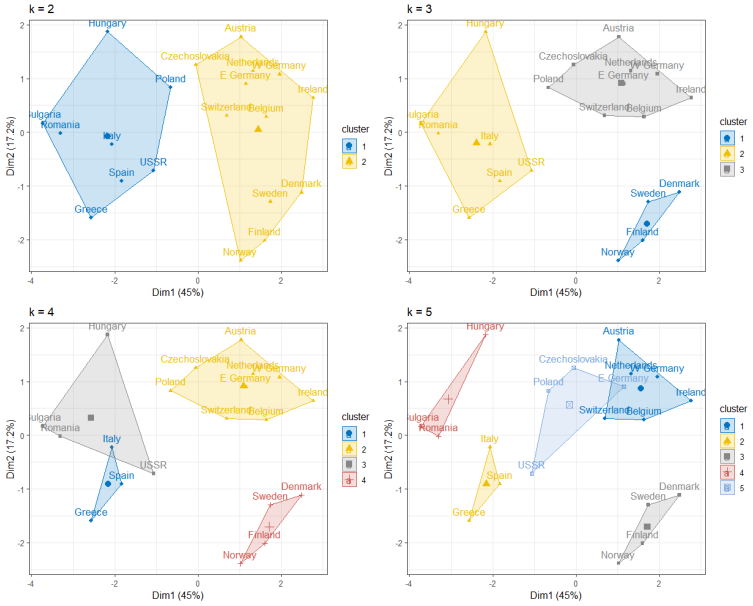

Figure (11) Plots for different values of k

3 Analysis 3

Question: Classification methods are very important in Machine Learning. Can we use a Decision tree and a Random Forest algorithm to do predictions? Which supervised method is more accurate?

3.1 Data Visualization

In this chapter, we will do a classification analysis. We use a decision tree and random forest algorithm. We will examine which method is more accurate, by using the `carseats_sale.csv` available from Brightspace.

Listing (8) Summary of data

```
> # Dimension of data:
> dim(df)
[1] 400  11
> # No missing values:
> fun <- function(x) length(which(is.na(x)==TRUE))
> df %>% apply(2, FUN=fun)
       Sales    CompPrice       Income  Advertising   Population        Price    ShelveLoc
           0            0            0            0            0            0            0
         Age    Education        Urban           US
           0            0            0            0
> # Summary of data:
> summary(df)
   Sales       CompPrice       Income        Advertising      Population      Price
 High:164   Min.   : 77   Min.   : 21.00   Min.   : 0.000   Min.   : 10.0   Min.   : 24.0
 Low :236   1st Qu.:115   1st Qu.: 42.75   1st Qu.: 0.000   1st Qu.:139.0   1st Qu.:100.0
            Median :125   Median : 69.00   Median : 5.000   Median :272.0   Median :117.0
            Mean   :125   Mean   : 68.66   Mean   : 6.635   Mean   :264.8   Mean   :115.8
            3rd Qu.:135   3rd Qu.: 91.00   3rd Qu.:12.000   3rd Qu.:398.5   3rd Qu.:131.0
            Max.   :175   Max.   :120.00   Max.   :29.000   Max.   :509.0   Max.   :191.0
  ShelveLoc        Age          Education    Urban       US
 Bad    : 96   Min.   :25.00   Min.   :10.0   No :118   No :142
 Good   : 85   1st Qu.:39.75   1st Qu.:12.0   Yes:282   Yes:258
 Medium:219   Median :54.50   Median :14.0
              Mean   :53.32   Mean   :13.9
              3rd Qu.:66.00   3rd Qu.:16.0
              Max.   :80.00   Max.   :18.0
```

The response variable is Sales. The other seven (7) variables are numerical (CompPrice, Income, Advertising, Population, Price, Age, Education), and three (3) variables are factors (ShelveLoc, Urban, US). Also, there are no missing values (NA). The dimension of the data set is 400 rows and 11 columns.

3.2 Decision tree, Random forest

In this chapter, we performed a Decision tree algorithm. Also, we performed a cross-validation method to prune the initial tree, if it needs. We will do that by selecting the nodes with the minimum error.

Listing (9) Summary of data

```
> # Cross validation:
> cv_table <- data.frame(
+   size  = cv_output.tree$size, error = cv_output.tree$dev)
> cv_table
   size error
1    25    70
2    24    68
3    16    74
4    14    75
5    13    77
6     9    79
7     6    81
8     4    90
9     3    91
10    2   108
11    1   121
```

We conduct that we have to prune the tree into 24 nodes. The plot (Figure 12) of the unpruned-pruned is shown below:

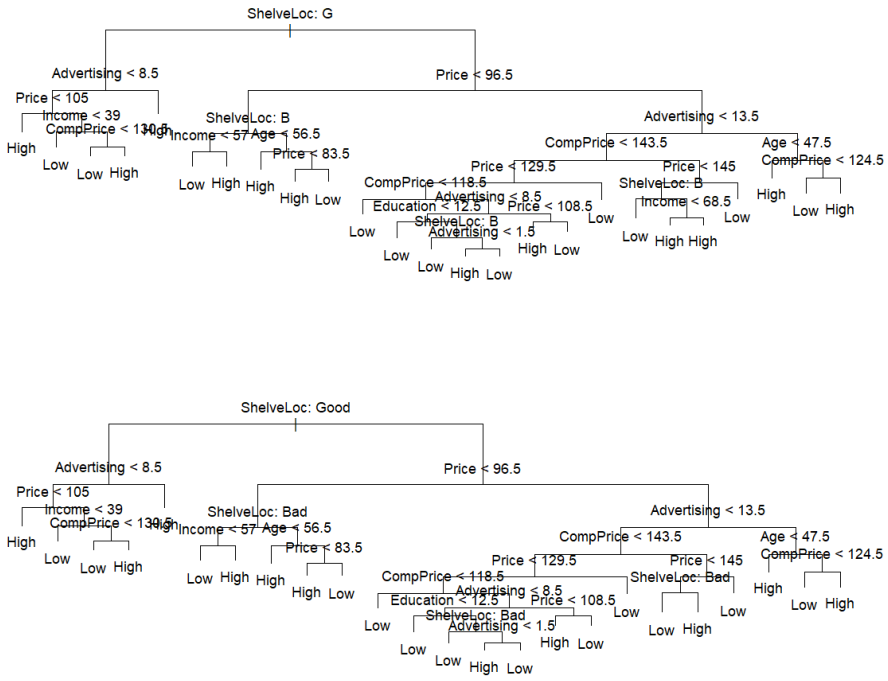

Figure (12) Unprunned / Pruned plots

We trained our model. Now, it is time to do predictions. The confusion matrix for the decision tree (left) and for the random forest (right) algorithm is shown below:

Listing (10) Confusion Matrices for Decision trees and Random forest algorithms

```
Confusion Matrix and Statistics
        (Decision Tree)
          Reference
Prediction High Low
      High   28  12
      Low    16  44

               Accuracy : 0.72
                 95% CI : (0.6213, 0.8052)
    No Information Rate : 0.56
    P-Value [Acc > NIR] : 0.0007243

                  Kappa : 0.4262

            Sensitivity : 0.6364
            Specificity : 0.7857
         Pos Pred Value : 0.7000
         Neg Pred Value : 0.7333
             Prevalence : 0.4400
         Detection Rate : 0.2800
   Detection Prevalence : 0.4000
      Balanced Accuracy : 0.7110
'Positive' Class : High
```

```
Confusion Matrix and Statistics
        (Random Forest)
          Reference
Prediction High Low
      High   31   4
      Low    13  52

               Accuracy : 0.83
                 95% CI : (0.7418, 0.8977)
    No Information Rate : 0.56
    P-Value [Acc > NIR] : 9.707e-09

                  Kappa : 0.6473

            Sensitivity : 0.7045
            Specificity : 0.9286
         Pos Pred Value : 0.8857
         Neg Pred Value : 0.8000
             Prevalence : 0.4400
         Detection Rate : 0.3100
   Detection Prevalence : 0.3500
      Balanced Accuracy : 0.8166
'Positive' Class : High
```

We observe that the accuracy $\left(\frac{TP+TN}{TP+TN+FP+FN}\right)$ of the **decision tree** algorithm is 72%, the sensitivity (True Positive rate) which is the number of correct positive predictions (High sales) divided by the total number of positives $\left(\frac{TP}{TP+FN}\right)$ is 63.64% and the specificity (True Negative rate), which is the number of correct negative predictions (Low Sales) divided by the total number of negatives $\left(\frac{TN}{TN+FP}\right)$ is 78.57% and for the **random forest**, the accuracy is 83%, the sensitivity is 70.45% and specificity 92.86%. We conduct that accuracy, sensitivity, and specificity are better in the random forest algorithm.

The False Positive rate (FPR) $Pr(Positive|Negative)$ is the number of the falsely predicted negative class (Low Sales) as positive divided by the number of false positives and true negatives. $FPR = \frac{FP}{FP+TN}$. For the **decision tree** algorithm $FPR = \frac{16}{16+44} = 0.27$, and for the **random forest** algorithm $FPR = \frac{13}{13+52} = 0.2$. So the random forest provides a lower FPR than the decision tree algorithm.

The False Negative rate (FNR) $Pr(Negative|Positive)$ is the number of the falsely predicted positive class label (High Sales) as negative divided by the number of true positives and false negatives. $FNR = \frac{FN}{TP+FN}$. For the **decision tree** algorithm $FNR = \frac{FN}{FN+TP} = \frac{12}{28+12} = 0.3$ and for the **random forest** is $\frac{4}{31+4} = 0.1142857$. So the random forest provides a lower FNR than the decision tree algorithm.

Furthermore, we cover important plots for the random forest algorithm. In the Left plot of Figure 13, we observe that increasing the number of trees decreases the error rate. Also, the error rate for High sales is greater than the error rate for Low and out-of-bag (OOB). The error rate for Low Sales contains the lowest values, as a result, the prediction for Low Sales to be more accurate. The out-of-bag is approximately 0.2 for $ntrees > 100$, which is very good. The Right plot of Figure 13 suggests that the Price and Shelveloc variables are the most important. The population is a less critical variable.

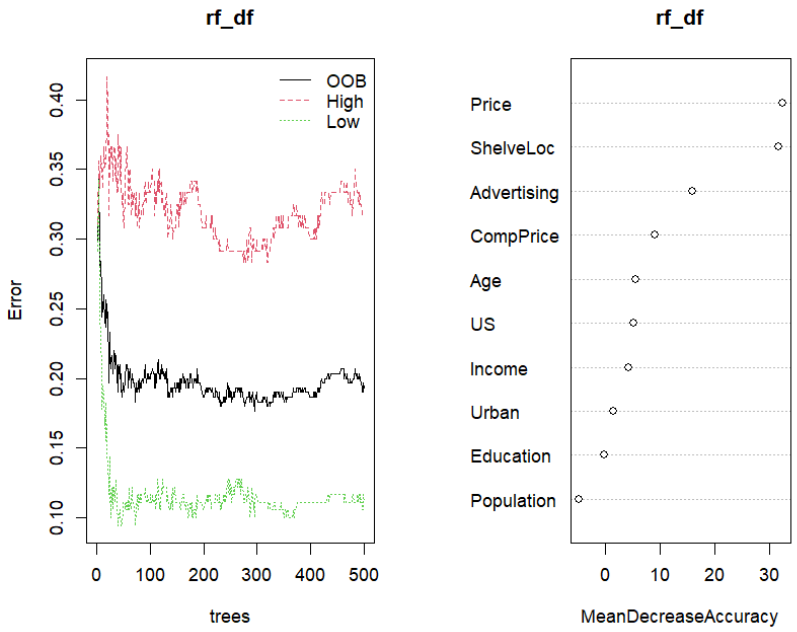

Figure (13) Left plot: Error plot for prediction, Right plot: Variable Importance plot

3.3 Receiver Operating Characteristic (ROC) curve

We select the best model by using the help of the ROC curve, as shown in Figure 14. It is a plot of the True Positive Rate (TPR) and False Positive Rate (FPR). We conduct that the Random Forest algorithm gives better predictions than the Decision tree algorithm. So, we conclude that for this data set, we choose the random forest algorithm to do predictions in classes (High and Low sales).

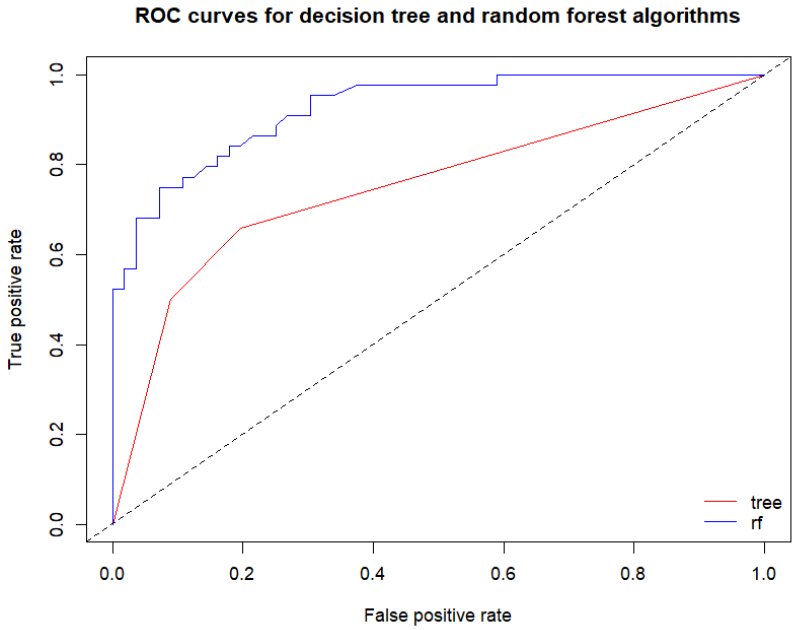

Figure (14) ROC curves

4 Appendix (R code)

```
# PCA for big data analytics
library(ggplot2)
library(corrplot)
library(GGally)
library(Matrix)
library(ggfortify)
library(dplyr)
library(ggpubr)
library(cluster)
library(factoextra)
library(gridExtra)
library(rpart)
library(RColorBrewer)
library(party)
library(randomForest)
library(tree)
library(caret)
library(ROCR)

# Import the dataset:
df <- read.csv("heptatlon.csv",header = TRUE)
df

if(is.data.frame(df)==FALSE) {
  df = as.data.frame(df)
} else {df}

df$hurdles <- max(df$hurdles) - df$hurdles
df$run200m <- max(df$run200m) - df$run200m
df$run800m <- max(df$run800m) - df$run800m

# Class of variables:
sapply(df,class) # 1 variable is character, 7 numerics and 1 integer

# Dimension of data:
dim(df) # 25 rows and 9 columns
25*9 # 225 saved data points in heptatlon

# Names of variables:
data.frame('Names of Variables' = colnames(df[,-1])) # Names of variables
```

```r
# Check for missing values:
which(is.na(df)==TRUE) # 0 NAs in data

# Summary of data:
df %>% select(-X) %>% summary(df)

# Data frame of sports variables
df.1 <- df %>% select(-X,-score)
dim(df.1)

# Correlation and density plots:
ggscatmat(df.1,columns = 1:ncol(df.1))

res <- round(cor(df.1),2)
corrplot(res, type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45)

# We standardize the data in order to transfer all the
# variables in the same units of measure

St <- matrix(0,25,7)

M_i <- apply(df.1,2,mean) # mean at each column
s_i <- apply(df.1,2,sd)   # std at each column

for(i in 1:7){
  St[,i]  = (df.1[,i] - M_i[i])/s_i[i]
}

St # Standardized data

X <- cov(St) # note the cov() gives the same outcome with cor()
             # because we have standardized the data.
cor(St)

colnames(X)  <- colnames(df %>% select(-X,-score))
row.names(X) <- colnames(df %>% select(-X,-score))
X
rankMatrix(X)[1] # The rank of the matrix is 7 = variables. (full ranked)

#----------------------------------------#
#------------------- PCA ----------------#
#----------------------------------------#
```

```
# We start the PCA

# We find the eigenvalues and eigenvectors with the command (eigen):

# Eigenvalues:
lamda <- eigen(X)$values
lamda

# We find the mean of eigenvalues
lamda.bar <- mean(lamda)
lamda.bar

which(lamda>lamda.bar) # We choose to work with the first two eigenvalues:
prop <- lamda/sum(lamda)
cumsum(prop) # Cumulative proportion of eigenvalues

# Scree plot:

index <- 1:length(lamda)
lamda

df.lamda <- data.frame(index,lamda,'cdf' =  cumsum(prop))

g1 <- df.lamda %>%
    ggplot(aes(x=index,y=lamda))+
  geom_point(size=2,colour='black') +
  geom_line(colour='red') +
  geom_hline(yintercept=1,size=0.5,linetype='dotted') +
  annotate("text", x = 5, y=1,label = "Lambda bar = 1", vjust = -0.5)+
  labs(y = 'Eigenvalues')+
  ggtitle('Scree plot') +
  theme_bw()

g2 <- df.lamda %>%
  ggplot(aes(x=index,y=cdf))+
  geom_point(size=2,colour='black') +
  geom_line(colour='red') +
  geom_hline(yintercept=0.85,size=0.5,linetype='dotted') +
  annotate("text", x = 5, y=0.85,
  label = "Determined proportion = 0.85 (85%)", vjust = -0.5)+
  labs(y = 'Eigenvalues')+
  ggtitle('Cumsum of proportion of eigenvalues') +
```

```
  theme_bw()

ggarrange(g1,g2,nrow = 1,ncol=2)

# Eigenvectors:
T <- eigen(X)$vectors
# Eigenvectors with the names of variables:
vec <- data.frame(colnames(df)[-c(1,9,10,11)],T)
colnames(vec) <- c('names','T1','T2','T3','T4','T5','T6','T7')
vec

# Scores:
# From theory we know that Z = X*T

Z <- St%*%T

Z1 <- Z[,1] # The score of the first principal
Z2 <- Z[,2] # The score of the second principal
Z1
Z2

plot(Z1,Z2)
plot(T[,1],T[,2])

cov(Z)
sum(diag(cov(Z)))
sum(diag(X))
sum(lamda)

# The first two eigenvectors:
vec[,1:3]

# Adding the scores in data:
df[,10] = Z1
df[,11] = Z2
colnames(df)[10:11] = c('Z1','Z2')

cor(df$score,df$Z1) # Correlation between scores and Z1
df %>% select(X,Z1,Z2) # Scores for the first two components

# Graph of scores:
df %>%
  ggplot(aes(x=Z1,y=Z2,group = X)) +
```

```
  geom_point() +
  labs(x = 'PC1 (63.72%)', y = 'PC2 (17.06%)',
       title = 'Scatter-plot of the two scores') +
  theme_bw()

# Biplot:

df.2 <- df %>% select(-score,-X,-Z1,-Z2)
pca_res <- prcomp(df.2, scale. = TRUE)
g3 <-autoplot(pca_res,df,loadings = TRUE, loadings.colour = 'blue',
         loadings.label = TRUE)+
  labs(title = 'Biplot of the two scores') +
  theme_bw()

# Graph of eigenvectors:
g4 <- ggplot(vec,aes(x = T1, y = T2, group = names,color = names)) +
        geom_point() +
          labs(x = 'T1 (first eigenvector)', y = 'T2 (second eigenvector)',
            title = 'Plot of the first two eigenvectors')+
              theme_bw()
ggarrange(g3,g4)
#----------------------------------------#
#----------- K-mean Clustering -----------#
#----------------------------------------#

scores <- df %>% select(Z1,Z2)

kmeans(scores,3)$cluster
library(cluster)
install.packages('factoextra')
library(factoextra)

g5 <- fviz_nbclust(scores, kmeans, method = 'silhouette')
g5

k = 2
kmeans_df = kmeans(scores,iter.max=50,centers = k,nstart=50)
g6 <- fviz_cluster(kmeans_df, data = scores)

# Index 25 is an outlier, we exclude from the data and we have 3 clusters
df[25,]
kmeans_df = kmeans(scores[-25,],iter.max=50,centers = k,nstart=50)
g7 <- fviz_cluster(kmeans_df, data = scores[-25,])
```

```
ggarrange(g6,g7)
#-----------------------------------------#
#------------ Cluster Analysis ------------#
#-----------------------------------------#

df <- read.csv("protein.csv",header = TRUE)

# We exclude the column of Countries and we make a data frame without countries
# with row names the countries.

row.names(df) <- df$Country
df <- data.frame(df %>% select(-Country))
df

dim(df)
head(df,5)

# Structure of data:
str(df)

# No missing values:
fun <- function(x) length(which(is.na(x)==TRUE))
df %>% apply(2, FUN=fun)

# Summary of data:
summary(df)

# Correlation and Density plot:
ggscatmat(df,columns = 1:ncol(df))

# Boxplots of data:
df %>%  boxplot(col=rainbow(9),main='Boxplots of data',xaxt='n',pch=20)
          text(1,23,'RedMeat')
            text(2,18,'WhiteMeat')
              text(3,8,'Eggs')
                text(4,37,'Milk')
                  text(5,17,'Fish')
                    text(6,58,'Cereals')
                      text(7,10,'Starch')
                        text(8,11,'Nuts')
                          text(9,12,'Fr.Veg')

# From the boxplots we detect some outliers.
```

```
## Detecting Outliers:

# Make a loop for detecting the index of outliers:

v = integer(0)
for(i in 1:ncol(df)){
vs = which(df[,i] %in% boxplot(df[,i],plot = FALSE)$out)
v = unique(c(vs,v))
}
v

# Now we exclude them from the data:
df <- df[-v,]

# Normalize the data:

standard = function(x) {
  if(is.numeric(x) == TRUE) {
   return((x - min(x))/(max(x)-min(x)))
                              }
  else if(is.character(x) == TRUE){return(x)}
                         }

df_st <- data.frame(df %>% apply(2,standard))

#-----------------------------------#
#-----Agglomerative Clustering------#
#-----------------------------------#

# Complete method:

df_st  %>%
   dist(method = 'euclidean') %>%
     hclust(method = 'complete') %>%
       plot(hang = -0.1,
        main = 'Cluster Dendrogram with complete method')
          rect.hclust(df_st %>%
              dist(method = 'euclidean') %>%
                hclust(method = 'complete'),
                  k=4)

### Select a partition containing k=5 groups
```

```
clusters_4 <- df_st  %>%
                dist(method = 'euclidean') %>%
                  hclust(method = 'complete') %>%
                    cutree(k = 4)

clusters_4

#---------------------#
#------ K-Means ------#
#---------------------#

# We choose k = 5 for kmeans clustering.

kmean_cluster_4 <- (df_st %>% kmeans(centers=4,nstart = 25))$cluster
kmean_cluster_4
# Evaluation of cluster results:
# Here we will use the Silhouette method in order to examine which cluster
# method is better.

# Silhouette method for Hierarchical Clustering with 5 partitions:

sil_hc_4      <- silhouette(clusters_4,
                     df_st %>% dist(method = 'euclidean'))
row.names(sil_hc_4) <- row.names(df_st)
sil_hc_4

# Silhouette method for K-Means Clustering with 5 partitions:

sil_k_means_4 <- silhouette(kmean_cluster_4,
                              df_st %>% dist(method = 'euclidean'))

row.names(sil_k_means_4) <- row.names(df_st)
sil_k_means_4

# Silhouette ggplots:

s1 <- fviz_silhouette(sil_hc_4,label = TRUE,rect = TRUE, palette = "jco")
s1$layers[[2]]$aes_params$colour <- "black"
s1

s2 <- fviz_silhouette(sil_k_means_4,label = TRUE,rect = TRUE, palette = "jco")
s2$layers[[2]]$aes_params$colour <- "black"
```

```
s2

ggarrange(s1,s2,nrow = 2,ncol = 1)
# Plotting the clusters with kmeans method with the help of PCA:

p1 <-    fviz_cluster(df_st %>%
                        kmeans(centers=2,nstart = 25) ,
                         data = df_st , palette = "jco" ) +
        ggtitle("k = 2") + theme_bw()

p2 <-    fviz_cluster(df_st  %>%
                         kmeans(centers=3,nstart = 25) ,
                          data = df_st   , palette = "jco") +
        ggtitle("k = 3") + theme_bw()

p3 <-    fviz_cluster(df_st %>%
                        kmeans(centers=4,nstart = 25) ,
                         data = df_st   ,  palette = "jco") +
        ggtitle("k = 4") + theme_bw()

p4 <-    fviz_cluster(df_st %>%
                        kmeans(centers=5,nstart = 25) ,
                         data = df_st , palette = "jco") +
        ggtitle("k = 5") + theme_bw()

grid.arrange(p1, p2, p3, p4, nrow = 2)

# We observe that k = 3 is a good choice.

# Optimal number of clusters for kmeans method:

fviz_nbclust(df_st, kmeans, method = "silhouette")

# We conduct that k = 3 is an optima number of partitions.

#--------------------------------
# Data: Carseats
#--------------------------------
df <- read.csv("carseats_sale.csv",stringsAsFactors = T)

#--------------------------------#
#-------- Decision Trees --------#
#--------------------------------#
```

```
# Structure of data:
str(df)
# Dimension of data:
dim(df)
# No missing values:
fun <- function(x) length(which(is.na(x)==TRUE))
df %>% apply(2, FUN=fun)
# Summary of data:
summary(df)

# Split the data into training and testing (75%-25%):
set.seed(52)
ind <- sample(1:dim(df)[1],dim(df)[1]*(0.75))
ind

training.df <- df[ind,]
testing.df  <- df[-ind,]

# Decision trees:
formula     <- reformulate(names(training.df[, -1]), response = 'Sales')
output.tree <- tree(formula = formula, data = training.df)

# Plot of the tree:
plot(output.tree)
text(output.tree, pretty = 1)

summary(output.tree)

# Cross validation:
cv_output.tree <- cv.tree(output.tree, FUN=prune.misclass)
cv_table <- data.frame(
  size  = cv_output.tree$size,
  error = cv_output.tree$dev
)
cv_table

pruned_tree_size <- cv_table[which(cv_table$error == min(cv_table$error)),'size']

# prune the tree to the required size:
pruned_tree_df <- prune.misclass(output.tree,best=pruned_tree_size)

# plot
plot(pruned_tree_df)
```

```
text(pruned_tree_df,pretty=0)

# Comparing unpruned and pruned trees:
par(mfrow=c(2,1))
plot(output.tree)
text(output.tree, pretty = 1)
plot(pruned_tree_df)
text(pruned_tree_df,pretty=0)
par(mfrow = c(1,1))

######################################################################
# 4. Decision tree prediction

predict_output.tree <- predict(output.tree,testing.df[,-1],type='class')
predict_output.tree

pruned_predict_output.tree <- predict(pruned_tree_df,testing.df[,-1],type='class')
pruned_predict_output.tree

predict_tree_table <- data.frame(actual   = testing.df$Sales,
                                 unpruned = predict_output.tree,
                                 pruned   = pruned_predict_output.tree)

predict_tree_table

unpruned_tree_table <- table(predict_tree_table[,c('actual','unpruned')])
unpruned_tree_table

pruned_tree_table <-  table(predict_tree_table[,c('actual','pruned')])
pruned_tree_table

#Evaluation:
#Confusion matrix for predictions of decision tree algorithm:
tree_conf_matrix <-
  confusionMatrix(data = predict_output.tree,reference = testing.df$Sales)
tree_conf_matrix

#------------------------------
# Random Forests
#------------------------------

# apply random forest to data:
rf_df <- randomForest(formula,ntree=500,importance = TRUE,data=training.df)
```

```
par(mfrow = c(1,2))
# Plot of random forest:
plot(rf_df)
legend('topright', colnames(rf_df$err.rate), bty = 'n',
lty = c(1,2,3), col = c(1:3))

# variable importance:
varImpPlot(rf_df, type = 1)
par(mfrow = c(1,1))
# Predict:

rf_df_predict <- predict(rf_df,testing.df[,-1],type = 'class')

rf_predict_results <- data.frame(actual = testing.df$Sales,predict=rf_df_predict)
rf_predict_results

rf_df_predict_table <- table(rf_predict_results)
rf_df_predict_table

# Confusion matrix:

rf_conf_matrix <-
  confusionMatrix(data = rf_df_predict,reference = testing.df$Sales)
rf_conf_matrix

# ROC curve for decision trees and random forest:

# set the parameters for tuning to 10-fold CV
ctrl_parameters <- trainControl(method = 'CV', number = 10)

# Train a tree:
train_tree <- train(formula,data=training.df,method='rpart',
trControl = ctrl_parameters)
train_tree

# Train a forest:
train_rf  <- train(formula,data=training.df,method='rf',
trControl = ctrl_parameters)
train_rf

pred_tree <- predict(train_tree,testing.df[,-1],type='prob')$High
pred_rf   <- predict(train_rf,testing.df[,-1],type='prob')$High
```

```
# data frame containing probability values for the prediction 'yes' and
# data frame for which a yes is true and no is false.

models_prob <- data.frame(tree = pred_tree, rf = pred_rf)
models_label <- data.frame(tree = testing.df$Sales =='High',
                           rf = testing.df$Sales == 'High')

# ROC curves:
ROC_pred <- prediction(models_prob,models_label)
ROC_perf <- performance(ROC_pred,'tpr','fpr')

plot(ROC_perf,col=as.list(c('red','blue')),
     main='ROC curves for decision tree and random forest algorithms')
abline(a = 0, b = 1, lty = 2, col = 'black')
legend(
  "bottomright",
  names(models_prob),
  col = c("red", "blue"),
  lty = rep(1,2),
  bty = 'n')

# Random forest algorithm is better classification algorithm than decision tree
# for this data set according to ROC curve.
#-------------------------------
# END
#-------------------------------
```