Four Case Studies in Data Science

Table of contents

Introduction

The purpose of this project is to showcase the way statistical methods are used to solve problems regarding datasets, as well as provide analysis and concrete conclusions in the four following case studies. The programming language that will be used is R.

Case I

- Introduction:

Use the data in the file dataA1.txt and run 3 algorithms (each of a different philosophy) for clustering. Choose the particular charecteristic of each method/technique. In case you have to use a clustering method via some criteria, explain your selection. Comment on the results. How many teams will you keep? Between 3 groupings, which would you recommend as the most appropriate?

- Analysis:

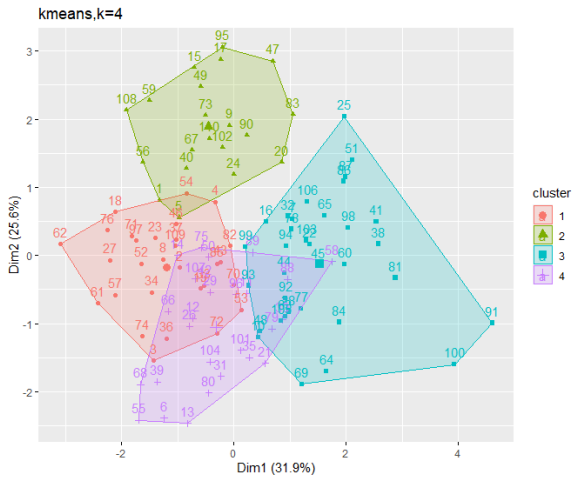
We will start by importing our data in RStudio and naming them as dataA1. The first method we will use is k-Means and in order to do so we'll implement the following command:

kmeans(data,centers), data will be replaced by dataA1 and centres by the number of clusters we want to create.

At first we will attempt to visualize our data so that we can get a better perception of how many teams we want to create. We'll use the command:

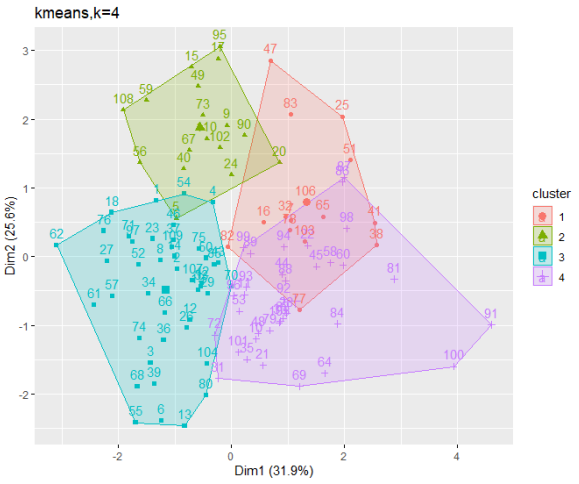
fviz_cluster(kc,dataA1,main ='kmeans,k=6'), kc = kmeans(dataA1,6)

We have the following diagram:

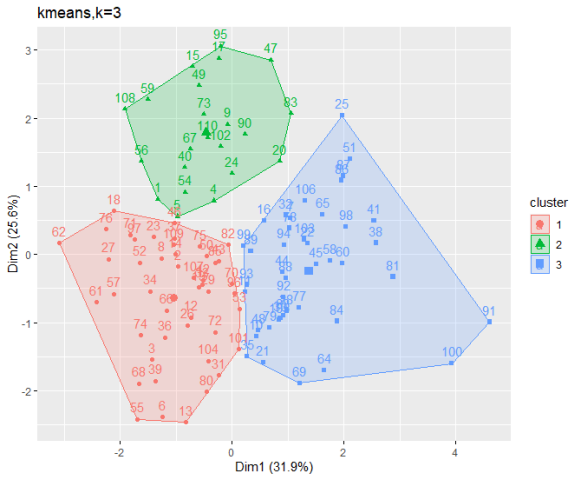


We'll implement the same command in order to visualize the results for 4, 3 and 2 clusters respectively. We have the following results:

- fviz_cluster(kc,dataA1,main = 'kmeans,k=4'), όπου kc=kmeans(dataA1,4)



- fviz_cluster(kc,dataA1,main = 'kmeans,k=3'), όπου kc=kmeans(dataA1,3)

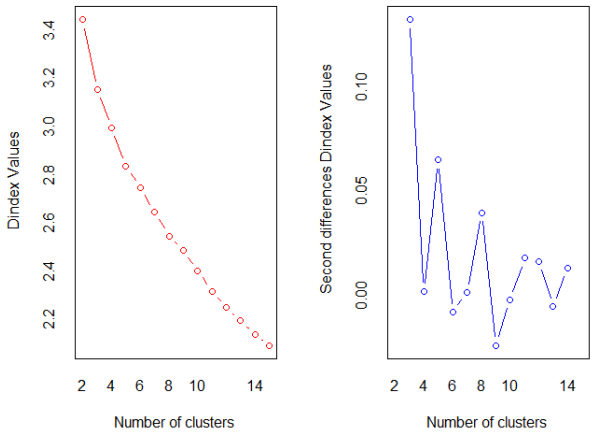


- fviz_cluster(kc,dataA1,main = 'kmeans,k=2'), όπου kc=kmeans(dataA1,2)

We notice that by decreasing the number of groups we get a better indication of how many

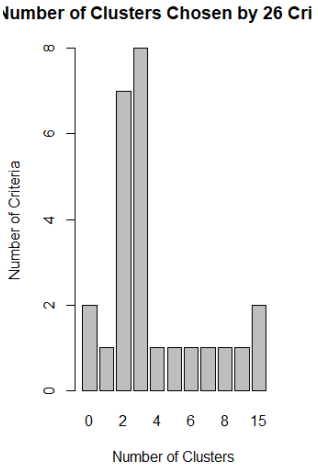
clusters we should use. However we have to be more precise, therefore we'll use the following command, which gives us the best possible number of groups by combining all possible clusters and distance measurement methods. We will get the following diagram:

- nc=NbClust(dataA1,min.nc = 2,max.nc = 15,method = "kmeans")



Based on the chart above, the elbow method indicates that the 3 team selection is pretty good. The following code will help us come up with the number of groups to use:

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



We notice that the optimal number of groups is 3. The command kc=kmeans(dataA1,3) will

give us the centers of the groups and show us in which group each observation belongs to. The distribution of the observations in the groups is 46, 42 and 22. The second method will showcase the use of hierarchical algorithms in order to determine the number of clusters. First we will use Ward's method to quantify similarity and use the following command to get the best number of groups:

- nn=NbClust(data=dataA1,diss = NULL,distance = "euclidean",min.nc =2,max.nc = 15,method = "ward.D2",index = "all",alphaBeale = 0.1)

Ward's method suggest the we should create 3 clusters. Our next step is to present the dendrogram and divide it into 3 groups, so that we can visualize the number of observations in each cluster.

- hc=hclust(dist(dataA1),method = "ward.D2")
- plot(hc)
- rect.hclust(hc,3)
- groupsW=cutree(hc,k=3)
- table(groupsW)

The observations are sorted into clusters with 21, 42, 47 each. We'll carry out the same steps for the simple linkage method:

- nn=NbClust(data=dataA1,diss = NULL,distance = "euclidean",min.nc =2,max.nc = 15,method = "single",index = "all",alphaBeale = 0.1)

This method suggests that we divide the observations into 2 groups containing 108 and 2 respectively.

- hc2=hclust(dist(dataA1),method = "single")
- plot(hc2)
- rect.hclust(hc2,2)
- groupsS=cutree(hc2,k=2)
- table(groupsS)

It is obvious that this method does not produce compact clusters, since one is too small and one too large. Our third method will be the complete linkage. Once again we will use the following code:

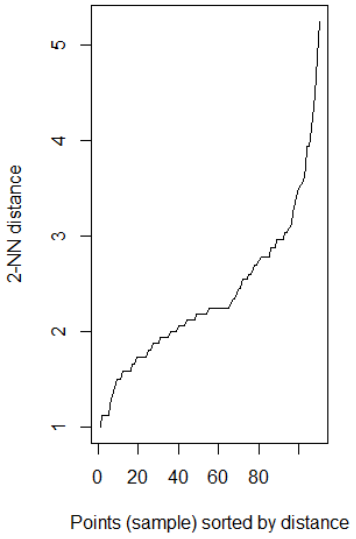
- nn=NbClust(data=dataA1,diss = NULL,distance = "euclidean",min.nc =2,max.nc = 15,method = "complete",index = "all",alphaBeale = 0.1)

It turns out that the best combinations of clusters are between 2 and 3 with 76 and 34 and 46,30,34 observations respectively. We'll use the 3 clusters since they are more compact. In order to determine which method works best we will check the ARIs. Για την Ward και πλήρη μέθοδο έχουμε την εντολή randIndex(groupsW,groupsC), με ARI = 0.3449, για την απλή μέθοδο και την Ward έχουμε randIndex(groupsW,groupsS) με ARI = 0.0031 και για την πλήρη μέθοδο και την απλή έχουμε randIndex(groupsC,groupsS) με ARI = 0.0002. According to the results above the simple linkage method exhibits random selection when in agreement with both complete linkage and Ward's method. The same conclusion can be

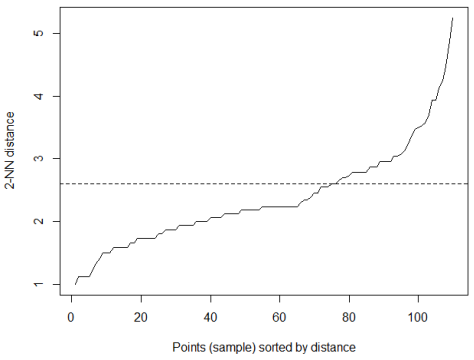
drawn for the agreement between Ward's method and complete linkage. Therefore Ward's method is our best option. The third and final method of clustering will be the DBSCAN algorithm. At first we will present the graph of ordered distances for 2 and 3 nearest neighbors respectively, with the following code:

- kNNdistplot(dataA1,k=2)
- kNNdistplot(dataA1,k=3)

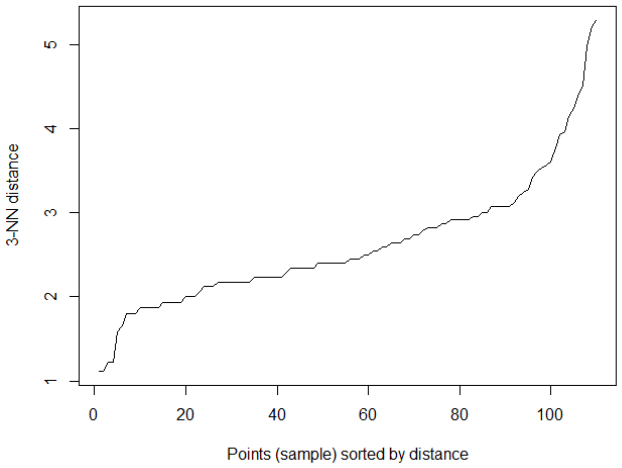
The first command gives us the following graph:



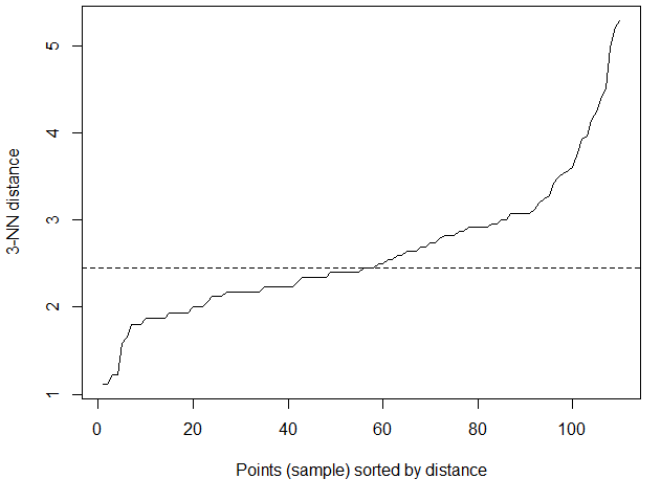
We are looking for the "knee" that is created in the chart, i.e. the point from which the graph resembles a straight line. In this case this point equals to 2.6 and by using the command abline(h=2.6,lty=2) the graph is as follows:



The same process is applied for 3 nearest neighbors.



- abline(h=2.6,lty=2)



For 3 nearest neighbors eps equals 2.45. Now we have to examine how many clusters the DBSCAN algorithm suggests. For eps=2.6 we have the following command:

- dbscan(dataA1,eps = 2.6, MinPts = 4)

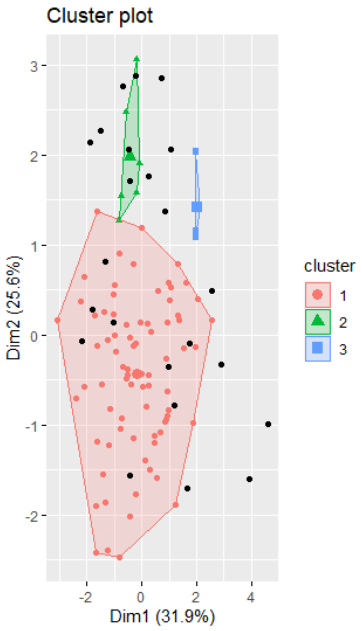
We have the following output:

dbscan Pts=110 MinPts=4 eps=2.6
0 1 2 3

border 23 19 2 2
seed 0 58 4 2
total 23 77 6 4

Now we will attempt to create a scatter plot in order to visualize the clusters.

- fviz_cluster(z,dataA1,geom="point"), z= dbscan(dataA1,eps = 2.6, MinPts = 4)



Aside from presenting the clusters, the plot above presents us with the noise as well. The same steps will be implemented for eps=2.45.

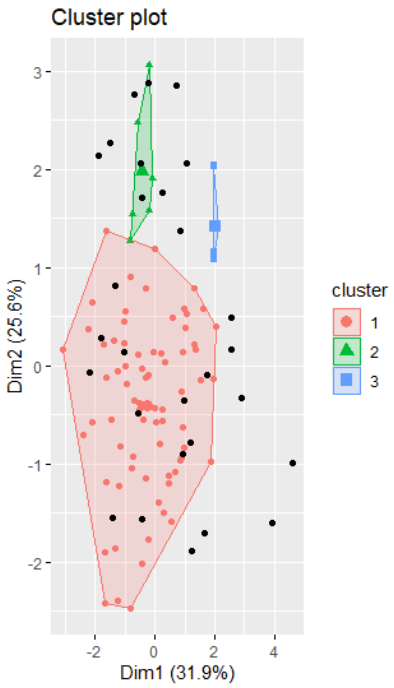
- dbscan(dataA1,eps = 2.45, MinPts = 4)

We have the following output:

dbscan Pts=110 MinPts=4 eps=2.45
0 1 2 3
border 28 20 2 2
seed 0 52 4 2
total 28 72 6 4

Now we will attempt to create a scatter plot in order to visualize the clusters.

- fviz_cluster(d,dataA1,geom="point"), d= dbscan(dataA1,eps = 2.45, MinPts = 4)



We conclude that we will choose the analysis with 2 nearest neighbors and 3 clusters. Among the 3 algorithms we applied, we will go with the DBSCAN algorithm since, as it turned out, it is more sensitive to noise and we did not have to determine the number of clusters in advance.

Case II

- Introduction:

Use the data in the file dataA2.txt and answer the questions below. The first 3 variables (respectively) fuel, repair and capital are related to the transport of milk by truck from the farms to the production plant for 84 companies. 50 trucks use gasoline and 34 use diesel-powered trucks. Note that the grouping variable is the 4th variable (type) with gasoline and diesel prices. This variable must be registered as a factor. When you import the data, create a training data (data.train) and a test data (data.test) randomly.Data.train should contain 35 out of 50 cases of trucks running on gasoline and 24 out of 34 cases diesel powered trucks. The rest to make up the data.test.Then apply at least 2 classification algorithms and compare their results. One of the two algorithms must be logistic regression. Choose the hyper-parameters of each algorithm (if such exist) so as to maximize its performance. After training the model, then apply it to data.test and evaluate the performance of the methods.Calculate evaluation measures such as sensitivity, specialty and correctness.

- Analysis:

After Importing the data into RStudio, we will register the fourth variable as a factor by

using the command below:

- dataA2$type=factor(dataA2$type,levels = c(0,1),labels = c("gasoline","sustainable"))

We will use the command set.seed(17238) and “shuffle” the dataset.

- dataA2=dataA2%>%slice(sample(1:n()))

The code below will provide us with the training dataset:

- dataA2sample=sort(sample(nrow(dataA2),nrow(dataA2)*.7))
- train=dataA2[dataA2sample,]
- test=dataA2[-dataA2sample,]

First we will implement the multiple logistic regression model and we’ll apply the following commands in R:

- model=glm(type~.,family=binomial(link='logit'),data=dataA2)
- summary(model)

The output is as follows:

```
Call:
glm(formula = type ~ ., family = binomial(link = "logit"), data = dataA2)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.33380  -0.16128   0.08987   0.36323   1.91786

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   2.9091     1.8391   1.582  0.11369
fuel          0.6220     0.1902   3.271  0.00107 **
repair       -0.2137     0.1077  -1.985  0.04715 *
capital      -0.5722     0.1417  -4.038 5.39e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 113.38  on 83  degrees of freedom
Residual deviance:  40.55  on 80  degrees of freedom
AIC: 48.55

Number of Fisher Scoring iterations: 7
```

The logistic regression model is: logit(type)=2.9091 + 0.6220(fuel) + (-0.2137)(repair) + (-0.57220)(capital). In order to estimate the model we will use the code below:

- train$type=predict(model,newdata = train,type = "response")
- train=cbind(train,predict(model,newdata = train,type = "link",se=TRUE))
- PredictedProb=plogis(train$fit)
- train$type=predict(model,newdata = train,type = "response")
- train$type

- train=cbind(train,predict(model,newdata = train,type = "link",se=TRUE))
- PredictedProb=plogis(train$fit)
- predResults=ifelse(train$type>0.6264,"diesel","gasoline")
- table(train$type,predResults)
- predResults=factor(PredResults,levels = c("diesel","gasoline"))
- train$type=factor(train$type,levels = c("diesel","gasoline"))
- confusionMatrix(predResults,train$type)

The command confusionMatrix(predResults,train$type) gives us the following output:

```
Confusion Matrix and Statistics

          Reference
Prediction diesel gasoline
  diesel        0        0
  gasoline      0        0

               Accuracy : NaN
                 95% CI : (NA, NA)
    No Information Rate : NA
    P-Value [Acc > NIR] : NA

                  Kappa : NaN

 Mcnemar's Test P-Value : NA

            Sensitivity :  NA
            Specificity :  NA
         Pos Pred Value :  NA
         Neg Pred Value :  NA
             Prevalence : NaN
         Detection Rate : NaN
   Detection Prevalence : NaN
      Balanced Accuracy :  NA

       'Positive' Class : diesel
```

Apparently something went wrong (!and I couldn't fix it). As a second classification method we will use knn.

- knn1<-knn(train=train[,1:3],test=test[,1:3],cl=train[,4],k=3)
- table(test[,4],knn1)

We have the following output:

```
knn1
            gasoline  diesel
  gasoline      11     5
  diesel         2     8
```

With 3 nearest neighbors the correct classification rate is (11+8)/26 = 0.7307 or 73.07%.
With 6 nearest neighbors we have the following:

- knn1<- knn(train=train[,1:3],test=test[,1:3],cl=train[,4],k=6)
- table(test[,4],knn1)

```
knn1
          gasoline diesel
gasoline       15      1
diesel          2      8
```

With 6 nearest neighbors the correct classification rate is (15+8)/26 = 0.8846 or 88.46%.
With 12 nearest neighbors we have the following:

- knn1<-knn(train=train[,1:3],test=test[,1:3],cl=train[,4],k=12)
- table(test[,4],knn1)

```
knn1
          gasoline diesel
gasoline       14      2
diesel          2      8
```

With 6 nearest neighbors the correct classification rate is (14+8)/26 = 0.8461 or 84.61%.
With 15 nearest neighbors we have the following:

- knn1<- knn(train=train[,1:3],test=test[,1:3],cl=train[,4],k=15)
- table(test[,4],knn1)

```
knn1
          gasoline diesel
gasoline       15      1
diesel          2      8
```

With 15 nearest neighbors the correct classification rate is (15+8)/26 = 0.8846 or 88.46%. We can assume that the percentage of correct classification differs less and less as the nearest neighbors increase and in fact as we continue the process we have indications that the percentage returns to the same prediction as with 6 neighbors. We can therefore assume that the algorithm's performance peaks for 6 nearest neighbors.

Case III

- Introduction:

Use the data in the file dataA3.txt and derive correlation rules. The data refers to 1000 transactions in a supermarket. Use the apriori algorithm. The maximum length of the rules should be 10, while the support and confidence level should be such that the number of rules that will arise does not exceed 100. The value that you set for support should not be higher than 0.45. Then identify the cases with the highest confidences and higher lifts. Comment on your results. Make a (basic) outline statistical analysis regarding the distribution of the number of goods that customers buy from the supermarket.

- Analysis:

We will start by importing the dataset to RStudio via the following command:

- datat<-read.transactions("C:/Users/user/Desktop/datascience/dataA3.txt", format = 'basket',header = FALSE)

Then we will select the values for support and confidence. In order to achieve a number of rules less than 100, we will choose extremely high values which will probably give us both a low lift and maybe a lift closer to 1. The consequence of this will be that the pattern that will be generated may not happen by chance. The above hypothesis will be examined by the coverage and fishersExactTest measures. We'll begin by presenting some descriptive measures of our data.

- datatsize=size(datat)
- quantile(datatsize,probs=c(0.01,0.05,0.10,0.25,0.50,0.75,0.90,0.95,0.99,0.999,0.9999,0.99999,0.999999))

We have the following outputs:

- Min. 1st Qu. Median Mean 3rd Qu. Max.
  4.00 8.00 12.00 12.06 16.00 20.00

- 1% 5% 10% 25% 50% 75% 90%
  4 4 5 8 12 16 19
  95% 99% 99.9% 99.99% 99.999% 99.9999%
  20 20 20 20 20 20

We observe that 50% of consumers buy 16 products. To see which are the 10 products that have the most demand, we apply the code below:

- datatcount<-itemFrequency(datat,'absolute')
- datatgoods<-sort(datatcount,decreasing=TRUE)

- datatgoods[1:10]

Which gives us the following output:

- Psomi Makaronia Tyri Zaxari Voutiro Mpananes
  853 819 793 741 728 683

  Ryzi Fakes Mpiskota Zampon
  675 671 662 634

It's obvious that that bread (Psomi) is the most frequent purchase by consumers. To extract association rules, we use the code below:

- rules=apriori(datat,parameter = list(maxlen=10,supp=0.45,conf=0.93))
- rules.sorted=sort(rules,by='lift')
- summary(rules.sorted)

The output:

```
set of 82 rules

rule length distribution (lhs + rhs):sizes
 3  4  5
37 43  2

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.000   3.000   4.000   3.573   4.000   5.000

summary of quality measures:
    support         confidence        coverage         lift            count
 Min.   :0.4500   Min.   :0.9301   Min.   :0.4720   Min.   :1.090   Min.   :450.0
 1st Qu.:0.4580   1st Qu.:0.9349   1st Qu.:0.4863   1st Qu.:1.100   1st Qu.:458.0
 Median :0.4705   Median :0.9413   Median :0.5015   Median :1.109   Median :470.5
 Mean   :0.4762   Mean   :0.9417   Mean   :0.5058   Mean   :1.112   Mean   :476.2
 3rd Qu.:0.4855   3rd Qu.:0.9473   3rd Qu.:0.5160   3rd Qu.:1.117   3rd Qu.:485.5
 Max.   :0.5430   Max.   :0.9619   Max.   :0.5770   Max.   :1.150   Max.   :543.0

mining info:
  data ntransactions support confidence                                                call
 datat         1000    0.45       0.93
```

The output indicates that the rules are 82.

- apriori(data = datat, parameter = list(maxlen = 10, supp = 0.45, conf = 0.93))

The number of rules does not exceed 100, however the lift we observed is very close to 1 and equal to 1,090, an indication of possible randomness. To evaluate the rules we will apply:

- measures=interestMeasure(rules.sorted,measure=c('coverage','fishersExactTest'),tra nsactions=datat)

- summary(measures)

We have the following output:

```
    coverage        fishersExactTest
    Min.   :0.4720   Min.   :0.000e+00
    1st Qu.:0.4863   1st Qu.:0.000e+00
    Median :0.5015   Median :1.900e-17
    Mean   :0.5058   Mean   :3.657e-14
    3rd Qu.:0.5160   3rd Qu.:6.678e-16
    Max.   :0.5770   Max.   :2.365e-12
```

We can see that the Min value for fishersExactTest is almost 0, while the Max value is less than 1. Therefore, we can assume that the rules describe actual customer behavior. For the 10 highest confidences and lifts we have the corresponding codes:

- inspect(head(sort(rules.sorted,by='confidence'),n=10))

Output:

```
    lhs                                rhs     support confidence coverage   lift     count
[1] {Mpananes, Mpiskota, Tyri} => {Psomi} 0.454   0.9618644  0.472    1.1276
[2] {Tyri, Zampon, Zaxari => {Psomi} 0.459   0.9602510  0.478            1.1257
[3] {Makaronia, Mpananes, Mpiskota} => {Psomi} 0.452   0.9576271  0.472    1.1226
[4] {Mpiskota, Zampon}             => {Psomi} 0.465   0.9567901  0.486    1.1216
[5] {Fakes, Makaronia, Ryzi}       => {Psomi} 0.452   0.9556025  0.473    1.1202
[6] {Mpiskota, Tyri, Zaxari}       => {Psomi} 0.470   0.9533469  0.493    1.1176
[7] {Fakes, Tyri, Voutiro}         => {Psomi} 0.464   0.9527721  0.487    1.1169
[8] {Makaronia, Tyri, Zampon}      => {Psomi} 0.481   0.9524752  0.505    1.1166
[9] {Mpiskota, Tyri, Voutiro}      => {Psomi} 0.458   0.9521830  0.481    1.1162
[10] {Makaronia, Mpananes, Ryzi}   => {Psomi} 0.452   0.9515789  0.475    1.1155
```

The output above, validates the fact that bread (Psomi) is the most common purchase. As for the 10 highest lifts we have the following:

- inspect(head(sort(rules.sorted,by='lift'),n=10))

Output:

```
    lhs                          rhs        support confidence coverage lift     count
[1] {Fakes, Mpananes, Psomi}     => {Makaronia} 0.452   0.9416667  0.480
1.149776

[2] {Mpananes, Tyri, Voutiro}    => {Makaronia} 0.467   0.9415323  0.496    1.149612

[3] {Psomi, Tyri, Voutiro, Zaxari} => {Makaronia} 0.457   0.9403292  0.486    1.148143

[4] {Fakes, Tyri, Voutiro}       => {Makaronia} 0.457   0.9383984  0.487    1.145786

[5] {Ryzi, Tyri, Zaxari}         => {Makaronia} 0.452   0.9377593  0.482    1.145005
```

[6] {Fakes, Psomi, Zaxari} => {Makaronia} 0.469 0.9361277 0.501 1.143013

[7] {Mpiskota, Zampon} => {Makaronia} 0.454 0.9341564 0.486 1.140606

[8] {Fakes, Psomi, Ryzi} => {Makaronia} 0.452 0.9338843 0.484 1.140274

[9] {Mpananes, Tyri, Zaxari} => {Makaronia} 0.463 0.9334677 0.496 1.139765

[10] {Psomi, Ryzi, Zaxari} => {Makaronia} 0.474 0.9330709 0.508 1.139281

Contrary to confidence, the 10 highest lifts suggest that in reality, the first 10 consumers are quite likely to combine spaghetti(Makaronia) instead of bread(Psomi) with the rest of the products.

Case IV

- Introduction:

Use the data in the dataA4.txt file. The variables that are given below contain information regarding 76 young bulls, aged <2 years. We have the following variables:

- Price of their sale (SalePr)
- Breed = Categorical variable with values 1, 5 and 8
- YrHgt = Yearling height at shoulder (inches)
- FrFrBody = Fat free body (pounds)
- PrctFFB = Percent fat-free body
- Frame = Scale from 1 (small) to 8 (large)
- BkFat = Back fat (inches)
- SaleHt = Sale height at shoulder (inches)
- SaleWt = Sale weight (pounds)

1)

Fit an appropriate linear regression model in which the dependent variable is SalePr and all the rest are independent except the variable Breed. Interpret the coefficients of the model in terms of whether the independent variables are correlated with the dependent variable(and in which direction). Explain the reason you chose a particular method to construct your model. Then provide forecasts for the selling prices of the following bulls.

AA	YrHgt	FtFrBody	PrctFFB	Frame	BkFat	SaleHt	SaleWt
1	51.3	990	72.4	5	0.15	52.5	1620
2	52.5	1110	66.3	6	0.10	52.3	1580
3	50.6	1230	69.5	8	0.15	51.1	1430
4	48.4	1040	74.3	7	0.20	55.9	1210
5	47.6	988	70.5	6	0.15	54.8	1370

If the corresponding selling prices are 1425, 1650, 1800, 1725, 1250 (ie the new prices of SalePr) calculate the MSE.

2)

Use the PLS method and try to explain how different measurements categorize bulls into 3 Breed categories. Provide the related diagram in two dimensions and interpret the results. What's your conclusion?

3)

Use the Decision Tree technique and construct a model that can predict the Breed category for each new bull. Use the Decision Tree you made and place the 5 new cases of bulls (see question (2)).

- Analysis:

1)

We'll begin by checking the correlation of our variables using the correlation matrix. First we will import our dataset.

- read.csv("C:/Users/Lefteris/Desktop/dataA4.txt", sep="", stringsAsFactors=TRUE)
- rcorr(as.matrix(dataA4),type = 'pearson')

We have the following correlation matrix:

```
Breed SalePr YrHgt FtFrBody PrctFFB Frame BkFat SaleHt SaleWt
Breed    1.00  -0.22  0.52     0.41    0.47  0.43 -0.62   0.49   0.12
SalePr  -0.22   1.00  0.42     0.10   -0.11  0.48  0.28   0.39   0.32
YrHgt    0.52   0.42  1.00     0.62    0.52  0.94 -0.34   0.86   0.37
FtFrBody 0.41   0.10  0.62     1.00    0.69  0.60 -0.17   0.70   0.56
```

PrctFFB	0.47	-0.11	0.52	0.69	1.00	0.48	-0.49	0.52	0.20
Frame	0.43	0.48	0.94	0.60	0.48	1.00	-0.26	0.80	0.37
BkFat	-0.62	0.28	-0.34	-0.17	-0.49	-0.26	1.00	-0.28	0.21
SaleHt	0.49	0.39	0.86	0.70	0.52	0.80	-0.28	1.00	0.57
SaleWt	0.12	0.32	0.37	0.56	0.20	0.37	0.21	0.57	1.00

We notice from the table above that there is a high correlation between the variables YrHgt and Frame (r=0.940), YrHgt and SaleHt (r=0.8595) while there is a moderate correlation between YrHgt and PrcFFB (r=0.5228). A moderate positive correlation is observed between SalePr and the YrHgt,Frame variables, a low positive correlation between SalePr and FtFrBody,BkFat,SaleHt,SaleWt, while the PrctFFB variable is negatively correlated with SalePr. Generally, we observe that almost all variables are correlated with each other. Then we'll implement the multiple regression model via the commands:

- modellm=lm(SalePr~ YrHgt + PrctFFB + Frame + BkFat + SaleHt + SaleWt + FtFrBody,data = dataA4)
- summary(modellm)

We have the following output:

```
Call:
lm(formula = SalePr ~ YrHgt + PrctFFB + Frame + BkFat + SaleHt +
   SaleWt + FtFrBody, data = dataA4)

Residuals:
   Min     1Q  Median     3Q    Max
-927.31 -279.95  -43.97  220.15 1612.87

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -3156.8944  4105.7192  -0.769  0.44461
YrHgt          25.9314   111.4029   0.233  0.81664
PrctFFB       -30.0976    26.7168  -1.127  0.26390
Frame         360.3168   172.9033   2.084  0.04093 *
BkFat        2571.5500   790.7881   3.252  0.00179 **
SaleHt         84.3969    64.0910   1.317  0.19232
SaleWt          0.3634     0.6085   0.597  0.55233
FtFrBody       -2.2012     1.0707  -2.056  0.04365 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 460.9 on 68 degrees of freedom
Multiple R-squared:  0.5037,       Adjusted R-squared:  0.4526
F-statistic: 9.859 on 7 and 68 DF,  p-value: 2.008e-08
```

According to the output the model is:

y = -3156.8944+25.9314(YrHgt)+(-30.0976)(PrctFFB)+360.3168(Frame)+2571.55(BkFat)+84.3969(SaleHt)+0.3634(SaleWt)+(-2.2012)(FtFrbody)

We will then evaluate the model in terms of its multicollinearity with the command:

- vif(modellm)

The output is as follows:

```
   YrHgt         PrctFFB    Frame        BkFat         SaleHt    SaleWt  FtFrBody
13.134475  2.693996  9.064782  1.770954  5.826196  2.202311  3.478312
```

The high value of the vif measure for both the Frame variable and the YrHgt variable, 9.0647 and 13.1344 respectively, suggests that the model is not good and indicates the strong presence of multicollinearity. We will now implement Ridge Regression.

- dataA4sc=scale(dataA4)
- dataA4sc=as.data.frame(dataA4sc)
- modelridge=lm.ridge(SalePr~ YrHgt + PrctFFB + Frame + BkFat + SaleHt + SaleWt + FtFrBody,data = dataA4sc,lambda = seq(0,100,1))

In order to get the best value for lambda we'll use the code below:

- select(lm.ridge(SalePr~ YrHgt + PrctFFB + Frame + BkFat + SaleHt + SaleWt + FtFrBody,data = dataA4sc,lambda = seq(0,100,1)))

Output:

```
modified HKB estimator is 4.32744
modified L-W estimator is 5.506111
smallest value of GCV  at 10
```

With lambda = 10 we will get the new coefficients:

- modelridgeOtp=lm.ridge(SalePr~ YrHgt + PrctFFB + Frame + BkFat + SaleHt + SaleWt + FtFrBody,data = dataA4sc,lambda = 10)

- coef(modelridgeOtp)

Output:

```
      YrHgt              PrctFFB            Frame           BkFat              SaleHt
 2.638963e-16  2.031222e-01  -1.839481e-01  3.553646e-01   3.096060e-01

     SaleWt            FtFrBody
 1.899778e-01   8.228046e-02


 -2.110630e-01
```

From the above output we notice that the dependent variable has a negative correlation

with the variables FtFrBody, PrctFFB and a positive correlation with the variables YrHgt, Frame, BkFat, SaleHt, SaleWt. We now have the following model:

y = -3156.8944+(25.9314)*(YrHgt)+(-30.0976)*(PrctFFB)+(360.3168)*(Frame)+(2571.55)*(BkFat)+(84.3969)*(SaleHt)+(0.3634)*(SaleWt)+(-2.2012)*(FtFrbody)

Now we'll estimate the selling prices of the 5 new bulls:

- y1 = -3156.8944+(25.9314)*(51.3)+(-30.0976)*(72.4)+(360.3168)*(5)+(2571.55)*(0.15)+(84.3969)*(52.5)+(0.3634)*(1620)+(-2.2012)*(990)= 1021.994

- y2= -3156.8944+(25.9314)*(52.5)+(-30.0976)*(66.3)+(360.3168)*(6)+(2571.55)*(0.10)+(84.3969)*(52.3)+(0.3634)*(1580)+(-2.2012)*(1110)= 1172.8

- y3 = -3156.8944+(25.9314)*(50.6)+(-30.0976)*(69.5)+(360.3168)*(8)+(2571.55)*(0.15)+(84.3969)*(51.1)+(0.3634)*(1430)+(-2.2012)*(1230)=1456.586

- y4 = -3156.8944+(25.9314)*(48.4)+(-30.0976)*(74.3)+(360.3168)*(7)+(2571.55)*(0.2)+(84.3969)*(55.9)+(0.3634)*(1210)+(-2.2012)*(1040)=1766.714

- y5 = -3156.8944+(25.9314)*(47.6)+(-30.0976)*(70.5)+(360.3168)*(6)+(2571.55)*(0.15)+(84.3969)*(54.8)+(0.3634)*(1370)+(-2.2012)*(988)=1451.215

To calculate the MSE we will make 2 vectors that will contain the predicted values and the actual values of the 5 bulls.

- y=c(y1,y2,y3,y4,y5)#pred

- x=c(1425,1650,1800,1725,1250)#actual

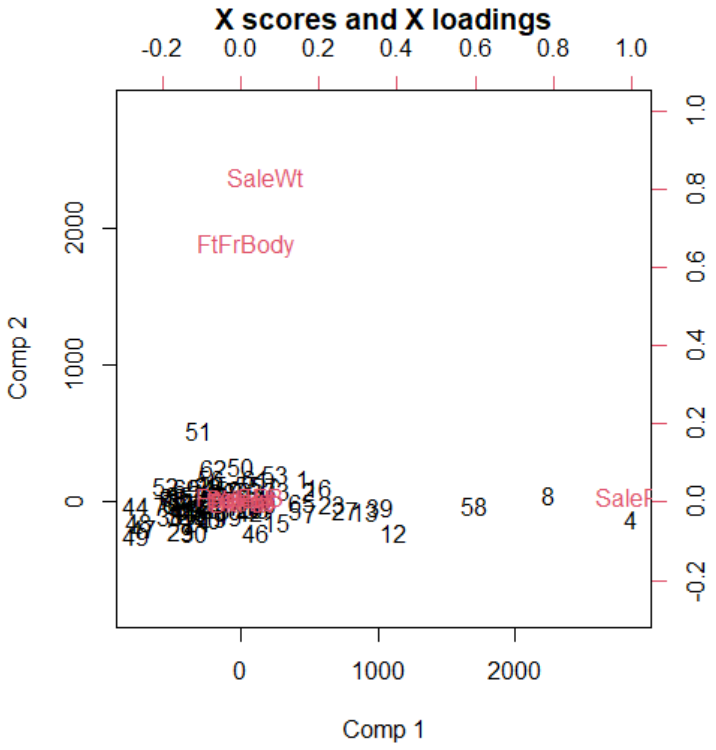
Through the command MSE( ), we have the MSE(y,x) = 110042.4. We can also calculate MSE by using the following code:

- data= data.frame(pred = y,actual = x)
- data
- mean((data$actual-data$pred)^2)

2)

In order to explain how the Breed variable is affected by the other variables, we will apply the following code:

- set.seed(17238)
- Xmatrix=dataA4[,-1]
- Xmatrix
- Yfactor=as.factor(dataA4[,1])
- PLSDA.model= caret::plsda(x=Xmatrix,y=Yfactor,ncomp=8)
- biplot(PLSDA.model)



We will first interpret the 1st dimension. On the far right we have the variable SalePr, while on the far left we have all the others except SaleWt,FtFrBody. So, we have an indication that the variables with the greatest influence on the categorization of the data are YrHgt, PrctFFB, Frame, BkFat, SaleHt, moreover it is easonable to assume that the frame of the bulls will influence their distribution in the genus. The same applies to the elbow height. So, based on the 1st dimension, the above variables contain the most valuable information. In the 2nd dimension, there is a strong separation between the variables SaleWt, FtFrBody and YrHgt, PrctFFB, Frame, BkFat, SaleHt. However we cannot claim that the variables located above and to the left significantly affect the categorization of our data. Therefore

the most important variables are those at the bottom left.

3)

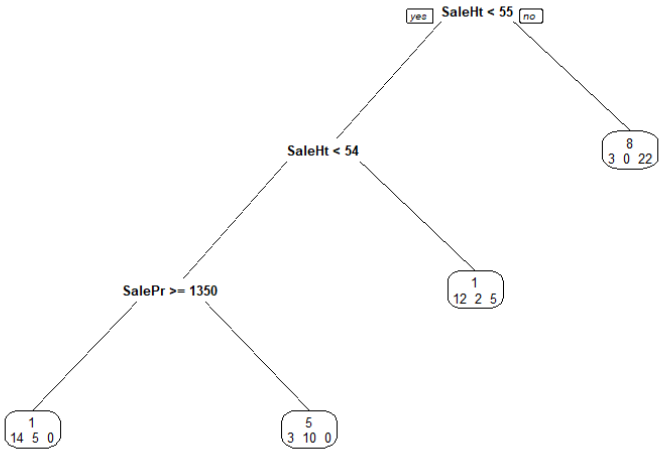
To build a decision tree we will use the following code:

- mtree=rpart(
  Breed~.,
  data = dataA4,
  method = "class",
  parms = list(split='information'),

)

- prp(mtree,faclen=0,cex = 0.8, extra = 1)
- mtree

We have the following output:



n= 76

node), split, n, loss, yval, (yprob)
* denotes terminal node

1) root 76 44 1 (0.42105263 0.22368421 0.35526316)
2) SaleHt< 55.05 51 22 1 (0.56862745 0.33333333 0.09803922)
4) SaleHt< 53.65 32 15 1 (0.53125000 0.46875000 0.00000000)
8) SalePr>=1350 19 5 1 (0.73684211 0.26315789 0.00000000) *
9) SalePr< 1350 13 3 5 (0.23076923 0.76923077 0.00000000) *
5) SaleHt>=53.65 19 7 1 (0.63157895 0.10526316 0.26315789) *
3) SaleHt>=55.05 25 3 8 (0.12000000 0.00000000 0.88000000) *

We notice that the initial node has 76 observations, skipping 44 objects we have as a dominant class the first one. In the next node with SaleHt < 55.05 we have 51 observations of which 22 will be wrongly distributed. The node splits once more and stops. Next, we will check if our model classifies the observations correctly, through the classification matrix.

- datapred=predict(mtree, newdata = dataA4, type = "class")
- conf_matrix=table(dataA4$Breed,datapred)
- conf_matrix

Output:

```
datapred
     1  5  8
1 26  3  3
5  7 10  0
8  5  0 22
```

We will measure the percentage of correct classification, which is equal to 0.7631 or 76.31%. This means that our model classifies 76.31% of the data correctly.

- sum(diag(conf_matrix))/sum(conf_matrix)

Based on the classification matrix and given the fact that the 1st new observation has SaleHt < 55, SaleHt < 54, SalePr = 1425 > 1350, it will be classified into group 1. The 2nd observation has SaleHt < 55, SaleHt < 54, SalePr = 1650 > 1350, therefore it will be classified into group 1 as well.The 3rd observation has SaleHt < 55, SaleHt < 54, SalePr = 1800 > 1350,it will also be classified into group 1. The 4th observation has SaleHt > 55, and will be classified in group 8. Finally the 5th observation has SaleHt < 55, SaleHt > 54, so it will be classified into group 1.