Data Clustering

1. The Product Sales Dataset

During this session, we will use the Data_Product.csv dataset that contains data on the sales of a product that has been proposed to customers. The `Product` variable indicates for each customer whether he has purchased the product or not.

Characteristics of the dataset:

- Instances: 600 customers
- Number of variables: 13
- Class variable: Product
- Missing values: none
- Column separator: comma
- Separator of decimals: point

The data dictionary below gives for each of the 13 variables its:

- label (name),
- type (integer, boolean or categorical),
- description (semantics),
- value domain (minimum and maximum values for a numeric variable, and list of possible values for a binary or categorical variable).

Data Dictionary

Variable	Type	Description	Value domain
ID	Integer	Identification number of the customer	[12101, 12400]
Age	Integer	Age in year	[18, 67]
Gender	Categorical	Gender of the customer	Male, Female
Habitat	Categorical	Places of residence	City_center, Small_town, Rural, Suburban
Income	Integer	Annual incomes in US dollars	[60392, 505040]
Married	Boolean	Marital status	Yes, No
Children	Integer	Number of children	[0, 3]
Car	Boolean	Owns a car?	Yes, No
Savings_Account	Boolean	Owns a savings account?	Yes, No
Current_Account	Boolean	Owns a current account?	Yes, No
Loan	Boolean	Outstanding loan?	Yes, No
Family_Quotient	Integer	Ratio between income and number of children	[20280, 492400]
Product	Boolean	Customer acquired the product? Class variable	Yes, No

1.1. Loading Data

☛ Load the data from the Data_Product.csv file into a data frame named `product`.

☛ Delete the `ID` variable from the `product` data frame with the selection operator `[,]`.

☛ Change the type of the numeric `Children` variable in the `product` data frame to the categorical type by the command:

```
> product$Children <- as.factor(product $ Children)
```

☛ Display the summarized characteristics of the `product` data frame with the `summary()` function.

2. Partitioning based Clustering

2.1. Distance Matrix

We will use the `cluster` library that provides a set of methods for clustering.

☛ Install/update and load into R the `cluster` library.

We will compute a distance matrix for the instances of the `product` data frame using the `daisy()` function of the `cluster` package that can handle heterogeneous data (numeric, categorical, etc.).

☛ Display the help of the `daisy()` function.

☛ Calculate the `dmatrix` distance matrix using the `daisy()` function with the command:

```
> dmatrix <- daisy(product)
```

☛ Display summary information from the `dmatrix` distance matrix using the `summary()` function.

2.2. K-means Clustering

The function `kmeans()` allows clustering by the k-means method from a data frame (containing numeric variables only) or a distance matrix.

☛ Run the K-means clustering algorithm, setting the number of clusters to 4 and storing the result in a `km4` object, with the command:

```
> km4 <- kmeans(dmatrix, 4)
```

The result is represented in the `km4$cluster` vector that provides for each instance the number of the cluster to which it is assigned.

☛ Add the cluster number given in the `km4$cluster` clustering result for each instance of the `product` data frame by the command:

```
> product <- data.frame(product, km4$cluster)
```

The assignmed cluster number (1, 2, 3, or 4) appears now as a new `km4$cluster` variable in the `product` data frame.

☛ Display the `product` data frame with the `View()` function.

2.3. Proportion of Classes by Cluster

In order to evaluate the correspondence between the clusters and the class of instances, we will compare the distribution of each class in each cluster.

☛ Display a contingency table showing for each cluster the number of instances in each class by the command:

```
> table (km4$cluster, Product)
```

☛ Using the `qplot()` function of the `ggplot2` library, display a population histogram for the `km4$cluster` variable in the `product` data frame with the `Product` variable (class variable) represented in color:

```
> qplot(km4$cluster, data = product, fill = product$Product)
```

☛ Display a population histogram for the `km5` to `km8` variables in the `product` data frame with the `Product` variable (class variable) represented in color:

☛ Using the `qplot()` function, display a scatter plot from the `product` data frame with:

- ☞ for the abscissa axis, the `children` variable,
- ☞ for the ordinate axis, the variable `km4$cluster`,
- ☞ for color of the points the variable `Product` (instance class).

Note: The `+ geom_jitter()` statement, following the `qplot()` statement, moves the points of a scatter plot of a random distance to distinguish the points that coincide.

☛ Using the `qplot()` function, display a scatter plot from the data frame produced with:

- ☞ for the abscissa axis, the variable `Family_Quotient`,
- ☞ for the ordinate axis, the variable `km4$cluster`,
- ☞ for color of the points the variable `Product` (instance class).

2.4. Creating R Scripts

It is possible to create R scripts directly in R Studio. An R script is a sequence of R commands that can

be executed automatically. Scripts make it easy to automate repeated executions of a sequence of commands.

- Create a new R script in R Studio (Ctrl + Shift + N) and save it to a `TD_Clust1.R` file.

It is possible either to write the commands directly in the R Studio window of the script, or to copy them from the history of the current R session using the `To Source` button.

- Copy and paste into the `TD_Clust1` script the commands you have made for:
  - loading the data,
  - creating the distance matrix,
  - clustering the data with K-means algorithm,
  - displaying the contingency table of cluster numbers and classes.
- Run the `TD_Clust1` script and compare the results obtained with those obtained previously (contingency tables).

Note: The random initialization of cluster centers of the K-means method can lead to different results from one execution to another.

2.5. Variation of the Number of Clusters

We will compare the results obtained for different numbers of clusters with the K-means method.

- Modify the `TD_Clust1` script by adding the commands to:
  - execute the K-means for a number of clusters ranging from 4 to 8 using a `for` loop,
  - display the contingency table of the cluster numbers and classes for each of the five clusterings obtained.
- Display the histogram of the number of clusters, with for color points the class of the instance, for each of the five partitions obtained.
- What is the number of clusters for which the proportion of instances of the same class in each cluster is maximal?

3. Hierarchical Clustering

3.1. Construction of a Dendrogram

The `hclust()` function is used to perform hierarchical clustering, the result of which is a dendrogram, from a distance matrix.

- Run hierarchical clustering with the `hclust()` function using the `ward.D2` aggregation method and store the result in an `hc` object with the command:

```
> hc <- hclust(dmatrix, method = "ward.D2")
```

- Display the resulting dendrogram `hc` using the `plot()` function with the command:

```
> plot(hc)
```

- In order to delimit with a red border the groups corresponding to 4 clusters in the displayed dendrogram, execute the command:

```
> rect.hclust(hc, k = 4, border = "red")
```

3.2. Result for a Given Number of Clusters

The result obtained for a given number of clusters can be obtained by the function `cutree()`.

- Calculate the result for 4 clusters, by storing it in a `ghc4` object, from the `hc` object by the command:

```
> ghc4 <- cutree(hc1, k = 4)
```

The `ghc4` object generated makes it possible to associate with each instance the number of the cluster to which this instance is assigned.

- Add the cluster number in `ghc4` for each instance of the data frame produced by the command:

```
> product <- data.frame(product, ghc4)
```

The assignment cluster number (1, 2, 3, or 4) appears as a new `ghc4` variable in the `product` data frame.

- Display a contingency table showing for each cluster the number of instances of each class by the command:

```
> table(ghc4, Product)
```

3.3. Variation of the Number of Clusters

We will compare the results obtained for different numbers of clusters from the dendrogram.

- Modify the `TD_Clust1` script by adding the commands to:
  - calculate the result for a number of clusters varying from 4 to 8 from the `hc` object,
  - to display the contingency table of the cluster numbers and classes for each of the five partitions obtained,
  - to delimit by a colored border in the dendrogram the groupings corresponding to the clusters obtained for each of the five partitions obtained:
    - red border for 4 clusters,
    - blue border for 5 clusters,
    - green border for 6 clusters,
    - gold border for 7 clusters,
    - sky blue border for 8 clusters.

Note: The list of predefined color codes in the R language can be found on the web page: http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf.

- Using the `qplot()` function of the `ggplot2` library, display a population histogram for the `ghc4` to `ghc8` variables in the `product` data frame with the `Product` variable (class variable) represented in color:
- What is the number of clusters for which the proportion of instances of the same class in each cluster is maximal?