Segmentation de l'Ensemble de Données Produit

1. Ensemble de données Produit

Nous allons durant cette séance utiliser l'ensemble de données *Data_Produit_QF.csv* qui contient des données sur l'achat d'un produit qui a été proposé à des clients. La variable *Produit* indique pour chaque client s'il a ou non acheté le produit.

Caractéristiques de l'ensemble de données :

- Instances : 600, chacune décrivant un client
- Nombre de variables : 13
- Variable de classe : *Produit*
- Valeurs manquantes : aucune

Le dictionnaire des données ci-dessous indique pour chacune des 13 variables :

- son libellé (nom),
- son type (entier, décimal, booléen, catégoriel, ordinal, date, etc.),
- sa description (sémantique),
- son domaine de valeurs (intervalle [minimum, maximum] pour les variables numériques ou liste de valeurs pour les variables modales/catégorielles).

Dictionnaire des données

Variable	Type	Description	Domaine de valeurs
ID	Entier	Numéro identifiant du client	[12101, 12400]
Age	Entier	Age en années	[18, 67]
Sexe	Catégoriel	Sexe	Homme, Femme
Habitat	Catégoriel	Type d'habitat	Centre_Ville, Petite_Ville, Rural, Banlieue
Revenus	Entier	Revenus annuels en dollars US	[60392, 505040]
Marie	Booléen	Statut marital	Oui, Non
Enfants	Entier	Nombres d'enfants	[0, 3]
Voiture	Booléen	Possède une voiture	Oui, Non
Compte_Epargne	Booléen	Possède un compte épargne	Oui, Non
Compte_Courant	Booléen	Possède un compte courant	Oui, Non
Emprunt	Booléen	Emprunt en cours	Oui, Non
Quotient_Familial	Entier	Rapport entre revenus et nombre d'enfants	[20280, 492400]
Produit	Booléen	Client acquéreur du produit Variable de classe	Oui, Non

1.1. Objectif

L'objectif de cette application est d'essayer d'identifier parmi chacune des deux classes, *Produit* = *Oui* et *Produit* = *Non*, des sous-groupes de clients, c-à-d un cluster de clients, ayant des caractéristiques communes (âge moyen, nombre d'enfants, etc.) qui leur sont spécifiques, c-à-d qui sont différentes de celles autres sous-groupes de la même classe.

Pour cela, plusieurs algorithmes de clustering (*K-means*, *Agnes*, etc.) seront appliqués, avec pour objectif de comparer la création de différents clusterings, chacun correspondant à un nombre spécifique de clusters (paramètre *K*) pour un algorithme particulier (ex : *K-means* pour *K* = 4).

1.2. Chargement des données

☛ Chargez les données du fichier *Data Produit QF.csv* dans un data frame `produit`.

- Supprimez la variable `ID` du data frame `produit` avec l'opérateur de sélection `[,]`.
- Modifiez le type de la variable *Enfants* du data frame `produit` qui est codée sous forme numérique mais représente une information de type ordinale (valeurs catégorielles ordonnée) par la commande :

```
> produit$Enfants <- factor(as.factor(produit$Enfants), ordered=TRUE)
```

  Le type ordinal permettra lors des calculs de distance entre les instances de tenir compte de l'ordre dans les valeurs.
- Affichez la classe et le mode de la variable `Enfants` à laide des fonction `class()` et `mode()`.
- Affichez les caractéristiques résumées du data frame `produit` par la fonction `str()`.
- Affichez les caractéristiques résumées des variables du data frame `produit` par la fonction `summary()`.

2. Matrice de distance

Nous allons utiliser la librairie *cluster* qui fournit un ensemble de méthodes pour le clustering.

- Installez et chargez la librairie *cluster*.

Nous allons calculer une matrice de distance pour les instances du data frame `produit` en utilisant la fonction `daisy()` du package `cluster` qui permet de traiter les données hétérogènes (numériques, catégorielles, ordinales, etc.).

- Affichez l'aide de la fonction `daisy()`.
- Calculez la matrice de distance `dmatrix` à l'aide de la fonction `daisy()` par la commande :

```
dmatrix <- daisy(produit)
```

- Affichez les informations résumées de la matrice de distance `dmatrix` à l'aide de la fonction `summary()`.

3. Clustering par partitionnement

3.1. Clustering par algorithme des K-means

La fonction `kmeans()` permet de réaliser un clustering par la méthode des *K-means* à partir d'un data frame (contenant des variables numériques uniquement) ou d'une matrice de distance.

- Exécutez le clustering par *K-means* en fixant le nombre de clusters à 4 et en stockant le résultat dans un objet `km4` par la commande :

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

Le résultat permet d'associer à chaque instance le numéro du cluster auquel cette instance est affectée.

- Créez un nouveau data frame `produit_km4` en ajoutant le numéro de cluster, stocké dans la variable `km4$cluster` du résultat `km4`, à chaque instance du data frame `produit` par la commande :

```
> produit_km4 <- data.frame(produit, km4$cluster)
```

Le numéro du cluster d'affectation (1, 2, 3 ou 4) apparaît comme une nouvelle colonne `km4.cluster` dans le data frame `produit_km4`.

- Affichez le data frame `produit_km4` à l'aide de la fonction `View()`.

3.2. Proportion des classes par clusters

Afin d'évaluer la correspondance entre les regroupements des instances (clusters) et la classe des instances, nous allons comparer les proportions de chaque classe pour chaque cluster.

- Affichez une table de contingence affichant pour chaque cluster le nombre d'instances de chaque classe par la commande :

```
> table(km4$cluster, produit$Produit)
```

- A l'aide de la fonction `qplot()` de la librairie *ggplot2*, affichez un histogramme d'effectifs pour chaque cluster (variable `km4$cluster`) avec la proportion de chaque classe en couleur (variable `produit$Produit`) :

```
> qplot(km4$cluster, data=produit, fill=produit$Produit)
```

- A l'aide de la fonction `qplot()`, affichez un nuage de points à partir du data frame `produit` avec :
  - pour axe des abscisses la variable `Enfants`,
  - pour axe des ordonnées la variable `km4$cluster`,
  - pour couleur des points la variable `Produit` (classe).

Note : L'instruction `+ geom_jitter(width = val1, height = val2)`, à la suite de l'instruction `qplot()`, permet de déplacer d'une distance aléatoire les points d'un nuage de points afin de distinguer les points qui sinon sont superposés (indistinguables dans le nuage).

- A l'aide de la fonction `qplot()`, affichez un nuage de points à partir du data frame `produit` avec :
  - pour axe des abscisses la variable `Quotient_Familial`,
  - pour axe des ordonnées la variable `km4$cluster`,
  - pour couleur des points la variable `Produit` (classe).

3.3. Création de scripts avec R Studio

Il est possible de créer directement dans R Studio des scripts comportant une suite de commandes R qui pourront être exécutées automatiquement.

Les scripts permettent de facilement automatiser les exécutions répétées d'une séquence de commandes.

- Créez un nouveau script R (`Ctrl+Shift+N`) et enregistrez-le dans un fichier `TD_Clust1.R`.

Il est possible soit d'écrire directement les commandes dans la fenêtre R Studio du script, soit de les copier depuis l'historique de la session R courante (par le bouton `To Source`).

- Copiez dans le script `TD_Clust1` les commandes que vous avez réalisées pour :
  - le chargement des données,
  - la création de la matrice de distance,
  - le clustering par *K-means*,
  - l'affichage de la table de contingence des numéros de clusters et des classes.
- Exécutez le script `TD_Clust1` et comparez les résultats obtenus avec ceux obtenus précédemment (tables de contingence).

Remarque : l'initialisation aléatoire des centres des clusters de la méthode des *K-means* peut conduire à des résultats différents d'une exécution sur l'autre.

3.4. Variation du nombre de clusters

Nous allons comparer les résultats obtenus pour des nombres de clusters de 4 à 7 avec la méthode des *K-means*. Nous allons Utiliser pour cela une boucle for dont l'indice va aller de 4 à 7.

Afin d'obtenir davantage d'informations sur les boucles en R, vous pouvez consulter la page https://www.datacamp.com/community/tutorials/tutorial-on-loops-in-r.

Voici par exemple une boucle `for` qui applique successivement la fonction `summary()` aux colonnes 1 à 12 du data frame `produit` :

```
for (i in 1:12) {
  print(summary(produit[,i]))
}
```

Rappel : il est nécessaire de faire appel à la fonction `print()` afin d'afficher un résultat, l'affichage implicite n'étant pas actif dans les boucles et les fonctions en R.

- Créez une boucle `for()` qui permette pour un nombre *K* de clusters variant de 4 à 7 de :
  - Exécuter les *K-means* pour le nombre *K*.
  - Créez la table de contingence affichant pour chaque cluster le nombre d'instances dans chaque classe.
  - Afficher l'histogramme d'effectifs pour chaque cluster avec la proportion de chaque classe en couleur.
- Quel est le nombre de clusters pour lequel la proportion d'instances de même classe dans chaque cluster est maximale ?