Paramétrage de l'Apprentissage

Nous allons durant cette séance générer différents arbres de décision correspondants à différents paramétrages de l'apprentissage de différents algorithmes et comparer les résultats de ces différents paramétrages obtenus lors de la phase de test.

Rappel : afin de réaliser les exercices plus rapidement, référez-vous aux corrections des travaux dirigés précédents disponibles sous forme de scripts R sur la page du cours. Si la commande ou fonction à utiliser pour réaliser un exercice ne vous est pas indiquée dans l'énoncé, cela signifie qu'elle a déjà été utilisée.

1. Ensemble de données *Produit*

Caractéristiques de l'ensemble de données :

- Instances : 600 clients
- Nombre de variables : 12
- Valeurs manquantes : aucune
- Séparateur de colonnes : virgule
- Séparateur de décimales : point
- Variable de classe : Produit
- Variables prédictives : Age, Sexe, Habitat, Revenus, Marie, Enfants, Voiture, Compte_Epargne, Compte_Courant, Emprunt

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
Produit	Booléen	Client acquéreur du produit Variable de classe	Oui, Non

2. Chargement des données

- Chargez les données du fichier `Data Produit.csv` dans un data frame `produit`.
- Vérifiez le chargement des données dans le data frame `produit` en affichant la liste des variables et leur mode à l'aide de la fonction `str()`.
- Construisez l'ensemble d'apprentissage `produit_EA` et l'ensemble de test `produit_ET` comme suit :
  - `produit_EA` : sélection des 400 premières lignes de `produit`.
  - `produit_ET` : sélection des 200 dernières lignes de `produit`.

3. Installation des librairies

Nous allons utiliser les fonctions `rpart()`, `C5.0()` et `tree()` de création d'arbres de décision fournies par les trois librairies R suivantes :

Librairie	Fonction
`rpart` (Recursive Partitioning and Regression Trees)	`rpart()`
`C50` (Decision Trees and Rule-Based Models)	`C5.0()`
`tree` (Classification and Regression Trees)	`tree()`

☛ Installez les librairies `rpart`, `C50` et `tree` (si celles-ci sont déjà installées, cela les mettra-à-jour si c'est nécessaire) et activez-les dans R.

4. Informations sur les paramètres en entrée des algorithmes

Les arbres de décision seront construits avec pour paramètres communs :

• Ensemble de données : `produit_EA`.

• Variable de classe (à prédire) : `Produit`.

• Variables prédictives : `Age`, `Sexe`, `Habitat`, `Revenus`, `Marie`, `Enfants`, `Voiture`, `Compte_Epargne`, `Compte_Courant`, `Emprunt`.

• Variables non utilisées : `ID`.

Rappel : le paramètre « `Produit ~ .` » indiquant à la fonction d'utiliser toutes les variables autres que la variable `Produit` comme variables prédictives, il est **nécessaire de supprimer la variable `ID`** de `produit_EA` pour qu'elle ne soit pas prise en compte si vous utiliser ce paramètre.

Afin d'optimiser la qualité des classifieurs générés, nous allons tester différents autres paramètres des fonctions d'apprentissage `rpart()`, `C5.0()` et `tree()`. Les différents paramétrages donneront lieu chacun à un appel à la fonction d'apprentissage de l'arbre..

5. Test de paramètres de l'apprentissage d'arbres de décision *rpart()*

Les paramètres principaux suivants seront utilisés lors des appels à la fonction `rpart()` :

• `split` : méthode de sélection d'attribut utilisée parmi les 2 possibles qui sont

◦ Coefficient de Gini : paramétrage `parms = list(split="gini")`

◦ Information Gain : paramétrage `parms = list(split="information")`

• `minbucket` : nombre minimal d'exemples dans un nœud feuille (défini l'effectif minimal requis comme support d'une prédiction). Par exemple pour un effectif minimal de 10 le paramétrage est : `control = rpart.control(minbucket = 10)`.

☛ Affichez l'aide le fonction `rpart()` par la commande :

```
> ? rpart()
```

☛ En vous aidant des informations et exemples fournis dans l'aide de la fonction `rpart()`, construisez les 4 arbres de décision `rpart()` correspondant aux 4 paramétrages suivants :

☞ `split = "gini"` et `minbucket = 10`

☞ `split = "gini"` et `minbucket = 5`

☞ `split = "information"` et `minbucket = 10`

☞ `split = "information"` et `minbucket = 5`

☛ Affichez les 4 arbres de décision à l'aide des fonctions `plot.rpart()` et `text.rpart()` de la librairie `rpart`.

Vous devez constatez que plusieurs de ces arbres sont identiques, c-à-d que vous obtenez en fait 2 arbres de décision différents seulement parmi les 4.

Cela signifie que le changement du paramètre concerné (le paramètre changé pour lequel 2 arbres sont identiques) n'a pas eu d'impact sur l'apprentissage.

☛ Identifiez le paramètre qui n'a pas eu d'impact sur l'apprentissage.

☛ Pour les deux différents arbres de décision obtenus :

☞Appliquez l'arbre à l'ensemble de test `produit_ET`.

☞Calculez le taux de succès (mesure de *Classification Accuracy*) de l'arbre.

6. Test de paramètres de l'apprentissage d'arbres de décision *C5.0()*

Les paramètres principaux suivants seront utilisés lors des appels à la fonction `C5.0()` :

- `minCases` : nombre entier définissant le nombre minimal d'exemples pour diviser un nœud en nœuds fils. Par exemple pour définir un nombre minimal de 20 le paramétrage est : `control = C5.0Control(minCases = 20)`.
- `noGlobalPruning` : une valeur booléenne indiquant si un élagage global doit être appliqué en fin de processus ou non pour simplifier l'arbre. Par exemple pour activer l'élagage global le paramétrage est : `control = C5.0Control(noGlobalPruning = TRUE)`.

☛ Affichez l'aide le fonction `C5.0()` par la commande :

```
> ? C5.0()
```

☛ En vous aidant des informations et exemples fournis dans l'aide de la fonction `C5.0()`, construisez les 4 arbres de décision `C5.0()` correspondant aux 4 paramétrages suivants :

☞`minCases = 10` et `noGlobalPruning = FALSE`

☞`minCases = 10` et `noGlobalPruning = TRUE`

☞`minCases = 5` et `noGlobalPruning = FALSE`

☞`minCases = 5` et `noGlobalPruning = TRUE`

☛ Affichez les 4 arbres de décision à l'aide de la fonction `plot.C5.0()` de la librairie `C50`.

☛ Pour les différents arbres de décision `C5.0()` obtenus :

☞Appliquez l'arbre à l'ensemble de test `produit_ET`.

☞Calculez le taux de succès (mesure de *Classification Accuracy*) de l'arbre.

7. Test de paramètres de l'apprentissage d'arbres de décision *tree()*

Les paramètres principaux suivants seront utilisés lors des appels à la fonction `tree()` :

- `split` : méthode de sélection d'attribut utilisée parmi les 2 possibles qui sont
  - Mesure de Deviance (cf. « *C4.5: Programs for Machine Learning* » - J.R. Quinlan, 1993) : paramétrage `split = "deviance"`
  - Coefficient de Gini : paramétrage `split = "gini"`
- `mincut` : nombre minimal d'exemples dans un nœud feuille (défini l'effectif minimal requis comme support d'une prédiction). Par exemple pour un effectif minimal de 10 le paramétrage est : `control = tree.control(nrow(produit_EA), mincut = 10)`.

☛ Affichez l'aide le fonction `tree()` par la commande :

```
> ? tree()
```

☛ En vous aidant des informations et exemples fournis dans l'aide de la fonction `tree()`, construisez les 4 arbres de décision `tree()` correspondant aux 4 paramétrages suivants :

☞`split = "deviance"` et `mincut = 10`

☞`split = "deviance"` et `mincut = 5`

☞`split = "gini"` et `mincut = 10`

☞`split = "gini"` et `mincut = 5`

☛ Affichez les 4 arbres de décision à l'aide des fonctions `plot.tree()` et `text.tree()` de la librairie `tree`.

☛ Pour les différents arbres de décision `tree()` obtenus :

☞Appliquez l'arbre à l'ensemble de test `produit_ET`.

☞Calculez le taux de succès (mesure de *Classification Accuracy*) de l'arbre.