Apprentissage d'Arbres de Décision

Nous allons durant cette séance comparer différentes fonctions d'apprentissage d'arbres de décision. L'objectif est de construire plusieurs modèles de prédiction, de comparer leur efficacité et de sélectionner le plus performant pour l'appliquer ensuite à de nouveaux exemples de classe inconnue.

Rappel : afin de réaliser les exercices, référez-vous si besoin 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

Afin que lors du chargement du fichier de données `Data Produit.csv`, la variable de classe Produit soit enregistrée dans de la data frame `produit` comme étant de type Factor, c-à-d de type catégorielle (i.e. avec une liste finie de valeurs possibles), le paramètre `stringsAsFactors` de la fonction `read.csv()` sera défini à la valeur `TRUE`.

Note : la définition de la variable de classe en type Factor est requise par certains algorithmes d'apprentissage de classifieurs (e.g. C5.0) et permet d'optimiser certains calculs tels que les mesures de distances.

☛ Chargez les données du fichier `Data Produit.csv` dans un data frame `produit` par la commande :

```
> produit <- read.csv("Data Produit.csv", header = TRUE, sep = ",", dec = ".",
```

`stringsAsFactors = T)`

☛ Vérifiez le chargement des données dans le data frame `produit` en affichant la liste des variables et leur type (mode) à l'aide de la fonction `str()`.

☛ Construisez l'ensemble d'apprentissage `produit_EA` et l'ensemble de test `produit_ET` par la méthode du partitionnement (*percentage split*) avec les 2 premiers tiers des exemples de `produit` pour `produit_EA` et le dernier tiers de `produit` des exemples pour `produit_ET` comme précédemment :

☞`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 trois librairies R suivantes pour la construction d'arbres de décision :

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

Rappel : afin d'installer et charger des librairies, vous pouvez utiliser :

• soit l'interface de RStudio (menu Tools pour installer et onglet Packages pour activer),

• soit les fonctions `install.packages("`*`nom_librairie`*`")` et `library(`*`nom_librairie`*`)` en ligne de commande.

☛ 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.

☛ Consultez les documentations de ces trois librairies soit via l'onglet Packages dans la zone bas-droite de RStudio, soit par la fonction `help()`, afin d'avoir un aperçu des fonctions fournies par chacune de ces librairies.

Rappel : si lors de l'installation d'une librairie vous obtenez un message d'erreur « `Installation of package '`*`nom_librairie`*`' had` **`non zero exit status`** », faites une mise à jour de R à partir de RStudio ou en ligne de commandes en :

• Installant et activant la librairie *installr*.

• Exécutant la fonction `updateR()` de cette librairie.

4. Apprentissage des arbres de décision

Nous allons utiliser les fonctions `rpart()`, `C5.0()` et `tree()` de création d'arbres de décision.

Pour ces trois fonctions, nous allons utiliser la même forme de commande (ci-dessous) qui permet de créer un arbre nommé *`nom_arbre`* de prédiction de la variable `Produit` à partir du data frame `produit_EA` par la fonction *`nom_fonction`* :

```
> nom_arbre <- nom_fonction(Produit ~ ., produit_EA)
```

Le paramètre « `Produit ~ .` » indique que la variable à prédire est `Produit` et le terme « `.` » indique que toutes les autres variables du data frame (Age, Sexe, …, Emprunt) sont les variables prédictives.

Les autres paramètres de la fonction n'étant pas spécifiés, leur valeur par défaut (définie dans la fonction) sera utilisée.

Ces arbres de décision seront construits avec pour paramètres :

• 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`.

Note : 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 utilisez cette formule pour le paramètre.

☛ Construisez un arbre de décision `tree1` avec la fonction `rpart()` selon les paramètres ci-dessus.

☛ Construisez un arbre de décision `tree2` avec la fonction `C5.0()` selon les paramètres ci-dessus.

☛ Construisez un arbre de décision `tree3` avec la fonction `tree()` selon les paramètres ci-dessus.

5. Représentation des arbres de décision

Nous allons afficher les arbres `tree1`, `tree2` et `tree3` à l'aide des fonctions `plot()` et `text()`.

Les fonctions et paramètres à utiliser pour afficher graphiquement un arbre de décision dépendent de son type : les arbres `rpart()` et `tree()` nécessitent des appels aux fonctions `plot()` et `text()` alors que les arbres `C5.0()` nécessitent un appel à la fonction `plot()` seulement.

Les paramètres d'application des fonctions `plot()` et `text()` à utiliser sont décrits dans l'aide de la version de la fonction définie dans la librairie du type d'arbre correspondant :

• Fonctions `plot.rpart()` et `text.rpart()` de la librairie `rpart`

• Fonctions `plot.tree()` et `text.tree()` de la librairie `tree`

• Fonction `plot.C5.0()` de la librairie `C50`

Lors de l'appel à la fonction, R détermine la version de la fonction à utiliser en identifiant le type d'objet passé en paramètre. Par exemple lors de l'appel `plot(tree1)` la fonction `plot.rpart()` est utilisée car l'arbre `tree1` est de type `rpart`.

Note : dans chacune des représentations graphiques des arbres de décision que vous allez construire, chaque nœud interne de l'arbre correspond à un test d'une variable prédictive. Si le test est vrai, il faut suivre l'arc sur la gauche et si le test est faux, il faut suivre l'arc sur la droite.

☛ Affichez l'arbre `tree1` sous forme graphique à l'aide des fonctions `plot.rpart()` et `text.rpart()` en traçant la structure de l'arbre par la commande :

```
> plot(tree1)
```

et en ajoutant le texte à la structure par la commande :

```
> text(tree1, pretty = 0)
```

☛ Affichez l'arbre `tree2` sous forme graphique à l'aide de la fonction `plot.C5.0()` par la commande :

```
> plot(tree2, type="simple")
```

☛ Affichez l'arbre `tree3` sous forme graphique à l'aide des fonctions `plot.tree()` et `text.tree()` en traçant la structure de l'arbre par la commande :

```
> plot(tree3)
```

et en ajoutant le texte à la structure par la commande :

```
> text(tree3, pretty = 0)
```

6. Test des arbres et calcul des taux de succès

Afin d'évaluer la précision des prédictions pour chacun des trois arbres sur l'ensemble de test `produit_ET`, nous allons utiliser la fonction `predict()` qui permet l'application d'un classifieur à un ensemble de données.

Les paramètres d'application de la fonction `predict()` peuvent varier selon le type de classifieur utilisé (i.e. arbre `rpart()`, `C5.0()` ou `tree()`).

Afin de générer la classe prédite, c-à-d la valeur « `Oui` » ou la valeur « `Non` », pour chaque exemple de test, la commande à utiliser pour les arbres `rpart()`, `C5.0()` et `tree()` est la suivante :

```
> nom_resultat <- predict(nom_arbre, nom_data_frame, type="class")
```

Note : si vous omettez le paramètre `type="class"` pour l'arbre `tree()`, vous obtiendrez pour chaque exemple de test la probabilité de prédiction (pourcentage de chances) pour chacune des classes (i.e. une colonne de probabilités pour `Oui` et une colonne de probabilités pour `Non`) et non la classe prédite (i.e. une unique colonne contenant la prédiction `Oui` ou `Non` la plus probable).

☛ Appliquez le classifieur `tree1` à l'ensemble de test `produit_ET` en stockant le résultat dans un objet nommé `test_tree1` en générant la classe prédite par la commande :

```
> test_tree1 <- predict(tree1, produit_ET, type="class")
```

☛ Affichez le contenu du vecteur `test_tree1` avec la fonction `print()`. Ce vecteur contient pour

chaque exemple de test la classe prédite pour cet exemple.

- Affichez le nombre de prédictions pour chaque classe dans `test_tree1` à l'aide de la fonction `table()` par la commande :

```
> table(test_tree1)
```

- Appliquez le classifieur `tree2` à l'ensemble de test `produit_ET` en stockant le résultat dans un objet nommé `test_tree2`.
- Affichez le contenu du vecteur `test_tree2` avec la fonction `print()`.
- Affichez le nombre de prédictions pour chaque classe dans `test_tree2` à l'aide de la fonction `table()`.
- Appliquez le classifieur `tree3` à l'ensemble de test `produit_ET` en stockant le résultat dans un objet nommé `test_tree3`.
- Affichez le contenu du vecteur `test_tree3` avec la fonction `print()`.
- Affichez le nombre de prédictions pour chaque classe dans `test_tree3` à l'aide de la fonction `table()`.
- Ajoutez le vecteur `test_tree1` comme nouvelle colonne `Tree1` dans le data frame `produit_ET` par la commande :

```
> produit_ET$Tree1 <- test_tree1
```

- Ajoutez le vecteur `test_tree2` comme nouvelle colonne `Tree2` dans le data frame `produit_ET`.
- Ajoutez le vecteur `test_tree3` comme nouvelle colonne `Tree3` dans le data frame `produit_ET`.
- Affichez les colonnes correspondant à la classe réelle (`Produit`) et aux classes prédites par les classifieurs (`Tree1`, `Tree2` et `Tree3`) à l'aide de la fonction `View()` par la commande :

```
> View(produit_ET[,c("Produit", "Tree1", "Tree2", "Tree3")])
```

- Calculez le taux de succès de l'arbre `tree1` en divisant le nombre de succès (valeurs des colonnes `Produit` et `Tree1` égales) par le nombre total d'exemples de test (nombre d'exemple dans `produit_ET`) par la commande :

```
> taux_succes1 <- length(produit_ET[produit_ET$Produit==produit_ET$Tree1,"ID"]) /
  nrow(produit_ET)
```

- Calculez le taux de succès de l'arbre `tree2`.
- Calculez le taux de succès de l'arbre `tree3`.
- À partir des taux de succès obtenus, identifiez l'arbre de décision `tree`*N* le plus performant (arbre `tree1`, `tree2` ou `tree3`).

7. Application de l'arbre sélectionné

- Nous allons maintenant appliquer l'arbre de décision sélectionné précédemment pour prédire la classe des instances de l'ensemble de données `Data Produit Prospects.csv` qui ne contient pas la variable Produit.
- Chargez les données du fichier `Data Produit Prospects.csv` dans un data frame `produit_PR`.
- Vérifiez le chargement des données dans le data frame `produit_PR` en affichant la liste des variables et leur type à l'aide de la fonction `str()`.
- Appliquez l'arbre sélectionné (`tree`*1*, `tree`*2* ou `tree`*3*) à l'ensemble à prédire `produit_PR` à l'aide de la fonction `predict()` en générant la classe prédite pour chaque instance et en stockant le résultat dans un objet `class_tree`*N*.
- Affichez le nombre de prédictions pour chacune des deux classes en appliquant la fonction `table()` à l'objet `class_tree`*N*.
- Ajoutez au data frame `produit_PR` une nouvelle colonne `Prediction` contenant la prédiction (classe `Oui` ou classe `Non`) faite pour chaque exemple prospect dans `class_tree`*N*.

8. Enregistrement des prédictions dans un fichier

Afin de déterminer quel paramétrage de la fonction `write.table()` permet d'enregistrer ce résultat dans un fichier au format `csv`, consultez la documentation de la fonction `write.table()`.

- Enregistrez le data frame `produit_PR` dans un fichier `resultats.csv` à l'aide de la fonction `write.table()` en spécifiant :
  - le séparateur de colonnes comme la tabulation (paramètre `sep = "\t"`),
  - le séparateur de décimales comme le point (paramètre `dec = "."`),
  - de na pas créer une colonne contenant le numéro de ligne (paramètre `row.names = F`).
- Vérifiez le résultat obtenu en ouvrant le fichier `resultats.csv` généré avec un éditeur de texte (e.g. Notepad++) ou un tableur (e.g. Excel ou LibreOffice).