

Représentations des Arbres de Décision

Nous allons durant cette séance générer différentes représentations textuelles et graphiques des arbres de décision. Nous allons ensuite comparer les résultats de l'utilisation de différents paramétrages pour l'apprentissage d'arbres de décision.

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

- ➔ Chargez les données du fichier `Data_Produit.csv` dans un data frame `produit` par la fonction `read.csv()` en utilisant les paramètres `header`, `sep`, `dec`, et `stringsAsFactors` comme lors de la séance précédente.
- ➔ 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` comme suit :
 - ➡ `produit_EA` : sélection des 400 premières lignes de `produit`.
 - ➡ `produit_ET` : sélection des 200 dernières lignes de `produit`.

3. Arbre de décision rpart

- ☛ Installez la librairie `rpart` et activez-la dans votre session R.
- ☛ Construisez un arbre de décision `tree1` avec la fonction `rpart()` selon les paramètres suivants :
 - ❖ 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`.
- ☛ Tracez la structure de l'arbre de décision `tree1` par la commande :

```
> plot(tree1)
```

- ☛ Ajoutez le texte, avec les libellés pour les variables catégorielles, par la commande :

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

- ☛ Cliquez sur le bouton *Zoom* afin d'ouvrir une fenêtre d'affichage et l'adapter à la taille de l'arbre.
L'arbre de décision obtenu possède la structure suivante :

 - Chaque nœud interne de l'arbre correspond à un test sur une variable prédictive (ex : `Revenus < 237676`).
 - Si le test est vrai, il faut suivre l'arc sur la gauche.
 - Si le test est faux, il faut suivre l'arc sur la droite.
 - Chaque nœud feuille correspond à une prédiction de classe (ex : `Produit = Non`).
 - Chaque branche (chemin de la racine à une feuille) correspond à une règle de prédiction de la classe des instances (ex : si `Revenus < 237676` et `Enfants >= 1.5` alors `Produit = Non`).

- ☛ Identifiez dans cet affichage graphique de l'arbre la classe prédite pour l'instance ci-dessous :

ID	Age	Sexe	Habitat	Revenus	Marie	Enfants	Voiture	Compte_Epargne	Compte_Courant	Emprunt
12701	23	H	Centre_Ville	150128	Oui	0.0	Oui	Oui	Non	Oui

4. Affichage textuel

- ☛ Affichez sous forme textuelle l'arbre de décision `tree1` par la commande :

```
> print(tree1)
```

Dans cet affichage textuel :

- Chaque ligne (exceptée la première) correspond à un test d'une variable prédictive. La première ligne donne des informations générales sur l'ensemble de données : nombre total d'instances, nombre d'instances de la classe majoritaire, libellé de la classe majoritaire et distribution de chaque classe.
- Les nœuds feuilles sont indiqués par le symbole * en fin de ligne.
- Pour chaque ligne, qui décrit un nœud de l'arbre, sont indiqués dans l'ordre :
 - le nombre d'instances correspondant à ce nœud,
 - le nombre d'instances de la classe non majoritaire pour ce nœud,
 - la classe majoritaire (valeur Oui ou Non de la variable `Produit` à prédire) pour ce nœud,
 - la proportion de la classe majoritaire pour ce nœud,
 - la proportion de la classe minoritaire pour ce nœud.
- L'indentation (décalage) d'une ligne, et le numéro associé à la ligne, indiquent le niveau du nœud dans l'arbre. Deux lignes au même niveau d'indentation représentent deux nœuds frères, c-à-d deux nœuds de même nœud père dans l'arbre.

Par exemple, la ligne :

```
2) Revenus< 237676 251 86 Non (0.65737052 0.34262948)
```

indique que :

- instances possèdent la propriété `Revenus < 237676`,
- de ces instances sont de la classe non majoritaire dans le nœud,

- la classe majoritaire dans le nœud est *Produit=Non*,
- les proportions des classes *Produit=Non* et *Produit=Oui* sont de 65.73 % et 34.26 % respectivement.
- ☛ Identifiez dans l'affichage textuel de `tree1` le nœud frère du nœud représenté par la ligne 2). Quelle est la classe majoritaire de ce nœud ?
- ☛ Identifiez dans cet affichage textuel de l'arbre la classe prédictive pour l'instance ci-dessous :

ID	Age	Sexe	Habitat	Revenus	Marie	Enfants	Voiture	Compte_Epargne	Compte_Courant	Emprunt
12702	30	H	Rural	79320	Non	1.0	Non	Oui	Non	Oui

5. Affichage graphique par la librairie *rpart.plot*

La librairie *rpart.plot* fournit les fonctions `rpart.plot()` et `prp()` qui permettent un affichage plus détaillé, avec davantage d'options, d'un arbre de décision `rpart()`.

- ☛ Installez la librairie `rpart.plot` et activez-la dans votre session R.
 - ☛ Construisez une représentation graphique simple de l'arbre `tree1` par la commande :
- ```
> prp(tree1)
```
- ☛ Cliquez sur le bouton *Zoom* de l'onglet *Plots* dans l'interface de RStudio afin d'ouvrir une fenêtre d'affichage et l'adapter à la taille de l'arbre (mettre la fenêtre en plein écran).

**Note** : 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.

Les possibilités de la fonction `prp()` sont présentées de manière synthétique dans le document « Plotting rpart trees with prp » accessible depuis la documentation de la librairie `rpart.plot` en suivant le lien **User guides, package vignettes and other documentation**.

Les différentes possibilités offertes par les paramètres `type` et `extra` que nous allons manipuler sont décrites succinctement sur les pages 4 et 5 respectivement de ce document.

- ☛ Identifiez dans ce document les valeurs des paramètres `type` et `extra` de la fonction `prp()` permettant d'afficher les éléments ci-dessous pour l'arbre `tree1` :
  - ☞ Les libellés (valeur de la variable testée) sur chaque arc de l'arbre (paramètre `type`).
  - ☞ Le libellé (*Oui* ou *Non*) de la classe majoritaire (paramètre `type`).
  - ☞ La proportion d'instances de la classe majoritaire pour chaque nœud de l'arbre (paramètre `extra`).

**Note** : afin de colorer les nœuds étiquetés *Non* en rouge clair et les nœuds étiquetés *Oui* en turquoise, ajoutez le paramètre `box.col=c("tomato", "darkturquoise") [tree1$frame$yval]` dans votre appel à la fonction `prp()`. Vous pouvez sinon utiliser à la place le paramètre `box.palette = "auto"` qui définit automatiquement les couleurs des nœuds avec pour chaque nœud une intensité proportionnelle (dégradés) à la distribution des classes.

- ☛ Modifiez le paramètre `extra` pour que soit affiché pour chaque nœud le nombre d'instances de chaque classe au lieu de la proportion d'instances de la classe majoritaire.

Cette proportion représente la probabilité que la prédiction soit correcte dans l'ensemble d'apprentissage (taux de véracité). Elle pourra ensuite être associée à chaque prédiction faite par ce nœud lors de l'application de la fonction `predict()` afin d'évaluer la fiabilité de cette prédiction.

## 6. Précision des prédictions

Lors de l'application des classificateurs à l'ensemble de test par la fonction `predict()`, nous avons jusqu'à présent généré comme résultat un vecteur contenant la valeur de la classe prédictive (i.e. valeur « *Oui* » ou « *Non* ») pour chaque exemple de test.

Il est également possible lors de l'application d'un classificateur à l'ensemble de test par la fonction `predict()` d'obtenir comme résultat les probabilités de prédiction dans chaque classe (i.e. probabilités de prédiction *Produit=Oui* et probabilités de prédiction *Produit=Non*) pour chaque exemple de test.

Sont alors générés en sortie deux vecteurs numériques : un vecteur pour chaque classe.

Les paramètres d'application de la fonction `predict()` pour générer la classe prédictive ou bien les proba-

bilités de chaque classe dépendent de la fonction qui a été utilisée pour créer le classifieur appliqué (fonction `rpart()` pour `tree1`, fonction `C5.0()` pour `tree2` et fonction `tree()` pour `tree3`).

Afin de générer pour chaque exemple de test, les probabilités de prédiction dans chaque classe, la commande à utiliser pour un arbre généré par les fonctions `rpart()` ou `C5.0()` est la suivante :

```
> nom_result <- predict(nom_arbre, nom_data_frame, type = "prob")
```

Afin de générer les probabilités pour chaque classe pour un arbre généré par la fonction `tree()`, la commande à utiliser est :

```
> nom_result <- predict(nom_arbre, nom_data_frame, type = "vector")
```

Rappel : les paramètres d'application de la fonction `predict()` à utiliser sont décrits dans l'aide de la version de la fonction `predict()` fournie par la librairie de la fonction correspondante :

- Fonction `predict.rpart()` de la librairie `rpart`.
- Fonction `predict.C5.0()` de la librairie `C50`.
- Fonction `predict.tree()` de la librairie `tree`.

☞ Générez pour chaque instance de l'ensemble de test les probabilités de prédiction des deux classes pour l'arbre `tree1` à l'aide de la fonction `predict()` en stockant les résultats obtenus dans un objet `prob_tree1`.

☞ Affichez l'objet `prob_tree1` obtenu à l'aide de la fonction `print()`. Cet affichage vous permettra de déterminer quel vecteur contient les probabilités de prédictions *Produit=Oui* et quel vecteur contient probabilités de prédiction *Produit=Non*, l'un ou l'autre pouvant être la 1<sup>ère</sup> ou la 2<sup>ème</sup> colonne du résultat.

☞ Affichez la liste des probabilités de prédiction *Produit=Oui* stockées dans le 2<sup>ème</sup> vecteur de l'objet `prob_tree1` par la commande :

```
> prob_tree1[,2]
```

☞ Affichez la liste des probabilités de prédiction *Produit=Non* stockées dans le 1<sup>er</sup> vecteur de l'objet `prob_tree1`.

Afin d'afficher des statistiques concernant ces probabilités de prédictions, nous allons construire un data frame contenant pour chaque exemple de l'ensemble de test :

- La classe réelle de l'exemple (vecteur `produit_ET$Produit`).

- La classe prédite pour l'exemple (vecteur `test_tree1`).

- La probabilité de prédiction de l'exemple dans la classe *Produit=Oui* (vecteur `prob_tree1[,2]`).

- La probabilité de prédiction de l'exemple dans la classe *Produit=Non* (vecteur `prob_tree1[,1]`).

☞ Construisez un data frame `df_result1` contenant les quatre vecteurs ci-dessus à l'aide de la fonction `data.frame()` dont la forme est la suivante :

```
> nom_data_frame <- data.frame(vecteur1, vecteur2, ..., vecteurN) .
```

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

☞ Renommez les colonnes dans le data frame `df_result1` par les libellés « *Classe* », « *Prediction* », « *P(Oui)* », « *P(Non)* » à l'aide la fonction `colnames()` dont la structure est la suivante :

```
> colnames(nom_data_frame) = list("colonne1", "colonne2", ..., "colonneN")
```

☞ Réaffichez le data frame créé à l'aide de la fonction `View()`.

☞ Utilisez la fonction `summary()` afin d'afficher les quartiles et la moyenne des probabilités des prédictions pour la classe *Produit=Oui* par la commande :

```
> summary(df_result1[df_result1$Prediction=="Oui", "P(Oui)"])
```

Affichez les quartiles et la moyenne des probabilités des prédictions pour la classe *Produit=Non* à l'aide de la fonction `summary()`.