Arbres

Mathieu Pigeon

UQAM

Arbres de décision

- L'objectif est de diviser l'espace des variables explicatives $\mathcal{X}_1 \times \ldots \times \mathcal{X}_p$, où $X_i \in \mathcal{X}_i$, $i = 1, \ldots, p$, en régions $\ll$ plus simples $\gg$.
- Les arbres de décision peuvent être utilisés pour des problématiques de régression et de classification.
- Ils sont simples à mettre en place et utiles pour l'interprétation.
- On combine généralement plusieurs arbres afin d'améliorer le pouvoir prédictif du modèle (prochain cours).

Exemple 1

Base de données *freaggnumber* disponible dans la librairie *CASdatasets* contient la fréquence totale de sinistres pour 12 513 classes d'assuré(e)s. Les variables sont

- **ClaimNumber** : la fréquence totale pour la classe ;
- **Exposure** : l'exposition totale pour la classe (en année-police) ;
- **VehAge** : l'âge du véhicule ;
- **LicenceAge** : l'âge auquel le conducteur a obtenu son permis de conduire ;
- **DriverAge** : l'âge du conducteur.

Exemple 1 : Préparation des données

```
library(CASdatasets)

data(freaggnumber)
dataAGG <- freaggnumber
head(dataAGG)
```

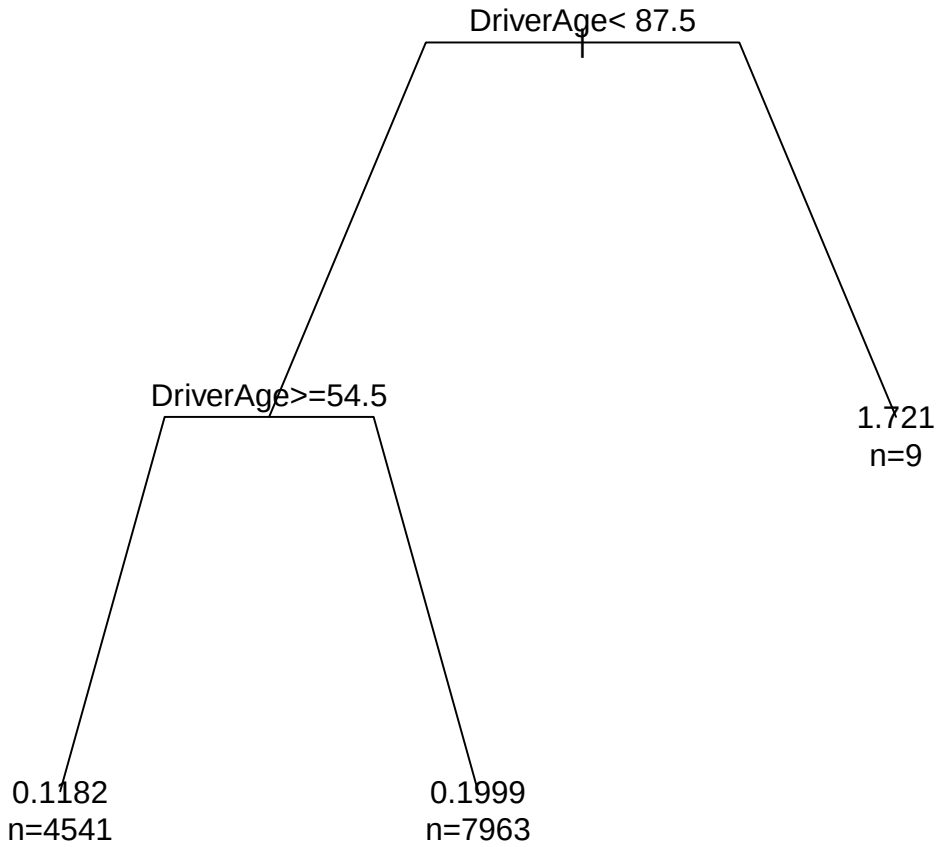
DriverAge	LicenceAge	VehAge	Exposure	ClaimNumber
39	18	3	1356.402	192
35	18	3	1243.948	172
37	18	1	1263.064	172
38	18	3	1328.589	171
39	18	2	1346.795	170
40	18	3	1263.537	165

```
dataAGG$Y <- dataAGG$ClaimNumber/dataAGG$Exposure
```

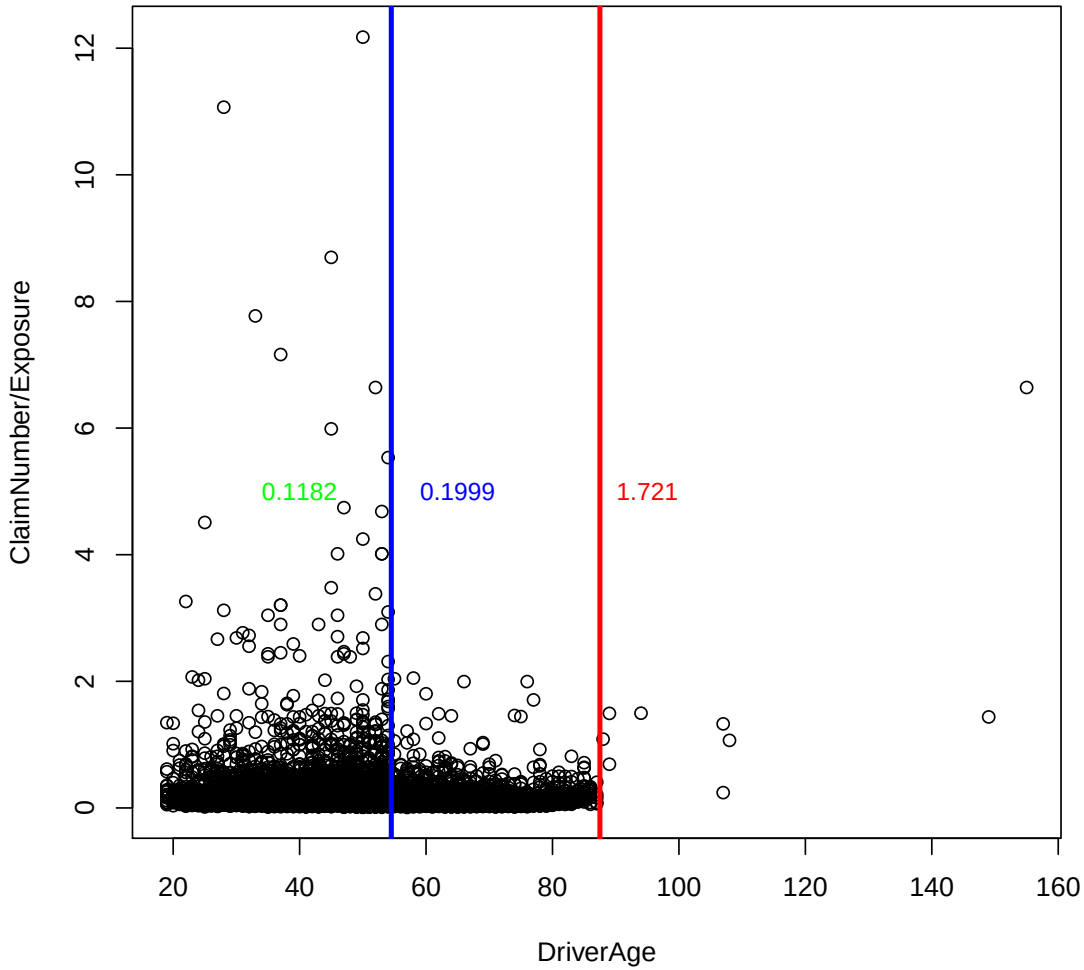
Exemple 1 : Arbre simple

```
install.packages("rpart")

library(rpart)

### Ajustement du modèle
arbre1 <- rpart(Y ~ DriverAge, data = dataAGG)

### Graphique
plot(arbre1, uniform = TRUE, branch = 0.5, margin = 0.1)
text(arbre1, all = FALSE, use.n = TRUE)
```

Exemple 1 : Arbre simple



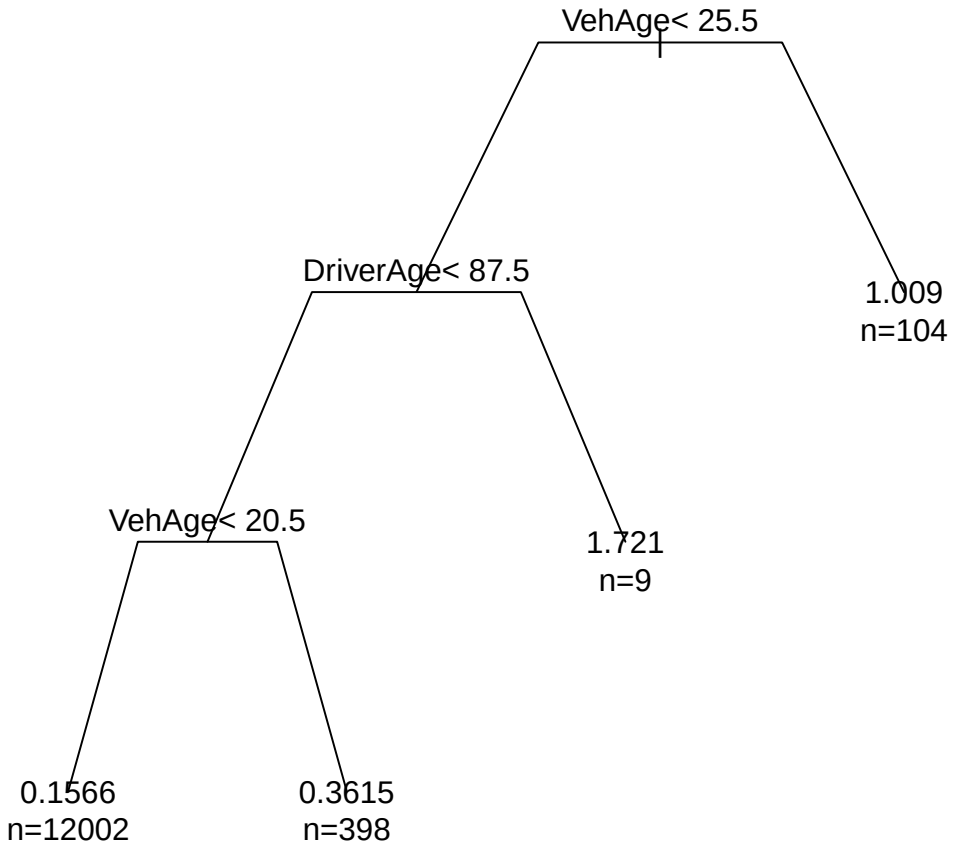
Exemple 1 : Arbre simple



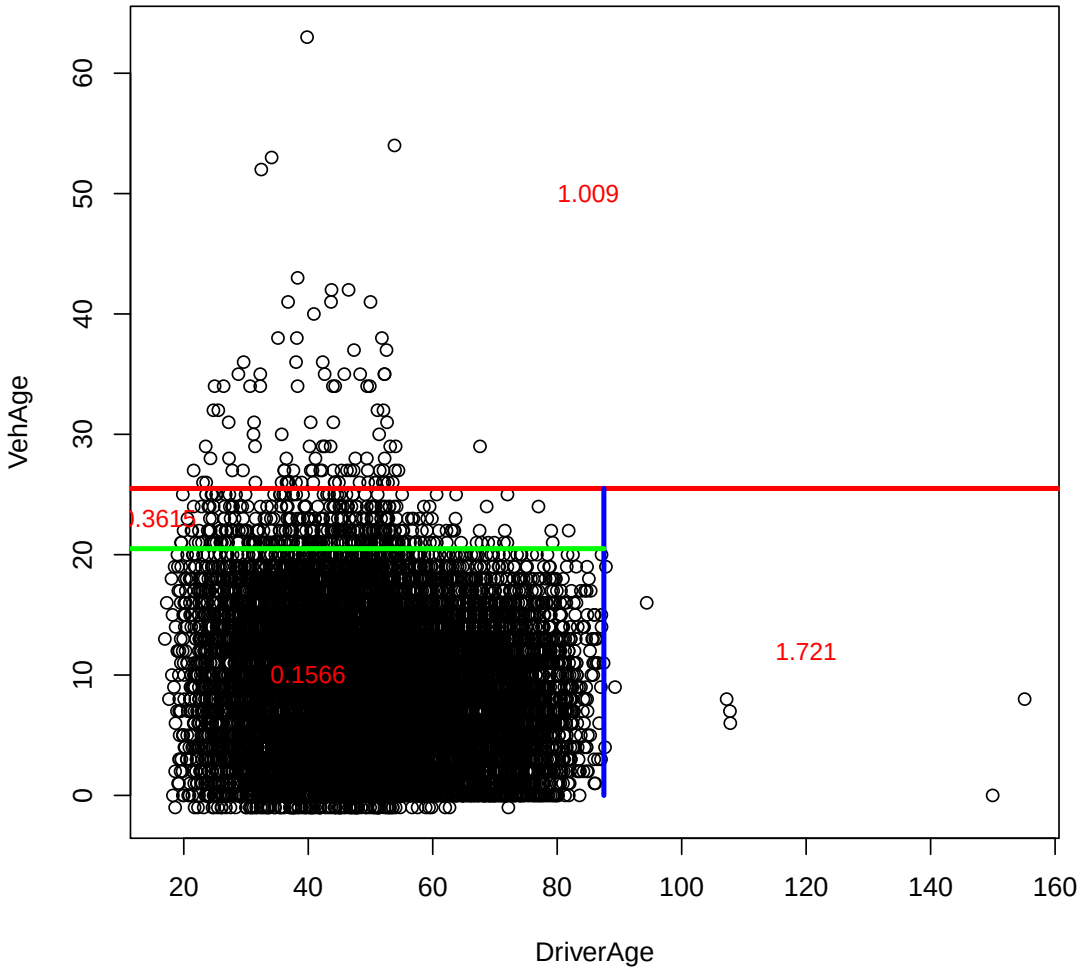
Exemple 1 : Arbre simple (2)

```
### Ajustement du modèle
arbre2 <- rpart(Y ~ DriverAge + VehAge, data = dataAGG)

### Graphique
plot(arbre2, uniform = TRUE, branch = 0.5, margin = 0.1)
text(arbre2, all = FALSE, use.n = TRUE)
```

Exemple 1 : Arbre simple (2)



Exemple 1 : Arbre simple (2)



Terminologie

- Les régions finales obtenues sont nommées **noeuds terminaux** ou, plus souvent, **feuilles** de l'arbre.
- Les autres points où se font les divisions sont des **noeuds internes** de l'arbre.
- Les segments de l'arbre qui connectent les noeuds sont des **branches**.

Procédure cadre

Pour réaliser des prédictions à partir d'un arbre il faut

1. diviser l'espace des variables explicatives en J régions, $R_1, \ldots, R_J$, telles que
   - $R_i \cap R_j = \emptyset$, $i \neq j$ et
   - $R_1 \cup R_2 \cup \ldots \cup R_J = \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_J$.
2. Pour chaque observation qui tombe dans la région R_j, une prédiction identique sera faite en utilisant, par exemple, la **moyenne** des observations de cette classe : $\widehat{Y}_{R_j} = \frac{1}{|R_j|} \sum_{i:\mathbf{x}_i \in R_j} Y_i$.

Comment diviser l'espace des variables explicatives ?

- La procédure cadre ne donne aucune indication sur la forme des régions : toutes les possibilités peuvent être tentées $\rightarrow$ une infinité de choix.
- On limite la forme des régions à des (hyper-)rectangles (des rectangles dans $\mathbb{R}^p$) afin de simplifier la construction de l'arbre et de permettre une interprétation des résultats (sous la forme d'un graphique).
- Chacun des noeuds se construit en optimisant une fonction objectif.

Fonction objectif

Pour une problématique de régression, on travaillera souvent avec la somme des erreurs au carré (SSE) ou l'erreur quadratique moyenne (MSE)

$$\text{SSE} = \sum_{j=1}^{J} \sum_{i:\mathbf{X}_i \in R_j} \left(Y_i - \widehat{Y}_{R_j} \right)^2,$$

où $\widehat{Y}_{R_j}$ est la valeur prédite dans la région R_j. On cherche alors une partition $R_1, \ldots, R_J$ qui **minimise** le SSE.

Fonction objectif

- À nouveau, si on considère toutes les partitions possibles de l'espace des variables explicatives, on devra considérer un trop grand nombre de possibilités.
- On va plutôt utiliser un algorithme glouton descendant (*top-down greedy approach*) par division binaire récursive (*recursive binary splitting*).
- Algorithme **descendant** : on commence avec toutes les observations dans une même classe (racine de l'arbre) et on divise l'espace en régions de plus en plus petites.
- Algorithme **glouton** : l'optimisation se fait à chaque étape sans regard vers le passé ou vers l'avenir.

Division binaire récursive

Première étape : on sélectionne la variable explicative X_j et le point c tels que la division en deux régions

$$R_1(j,c) = \{X_1, \ldots, X_J | X_j < c\}$$
$$R_2(j,c) = \{X_1, \ldots, X_J | X_j \geq c\}$$

conduise à la plus grande réduction possible de la fonction objectif. Si la somme des erreurs au carré a été choisie, on cherche alors à minimiser

$$\sum_{i:\mathbf{X}_i \in R_1(j,c)} \left(Y_i - \widehat{Y}_{R_1(j,c)}\right)^2 + \sum_{i:\mathbf{X}_i \in R_2(j,c)} \left(Y_i - \widehat{Y}_{R_2(j,c)}\right)^2.$$

Division binaire récursive

Étapes suivantes : à l'étape k, on répète la procédure décrite à la première étape pour chacune des régions créées à l'étape $k-1$, c'est-à-dire que pour chacune des régions r, on doit identifier la variable explicative X_j et le point c qui vont minimiser la fonction objectif après la division donnée par

$$\{\mathbf{X} \in r | X_j < c\} \text{ et } \{\mathbf{X} \in r | X_j \geq c\}.$$

La procédure est arrêtée lorsqu'un certain critère est atteint.
En l'absence de critère d'arrêt, l'arbre obtenu aura n feuilles (n n'étant pas nécessairement le nombre d'observations dans la base de données) $\rightarrow$ surapprentissage.

Critère d'arrêt

- On utilise souvent comme critère d'arrêt le fait d'avoir un nombre minimal d'observations dans chacune des régions.
- Utiliser comme critère d'arrêt le fait d'avoir une diminution de la fonction objectif qui dépasse un certail seuil n'est généralement pas une bonne idée.
- Même si un critère d'arrêt approprié est sélectionné, il y a fort à parier que l'arbre conduise à du surajustement, c'est-à-dire qu'il performera (trop) bien sur une base de données d'entrainement mais sera mauvais sur une base de données d'évaluation.

Élagage de l'arbre

- Une meilleure stratégie consiste en (1) construire un très gros arbre de départ $\mathcal{T}_0$ et (2) couper certaines branches de l'arbre.
- On cherchera à identifier le sous-arbre $\mathcal{T} \subset \mathcal{T}_0$ qui minimise

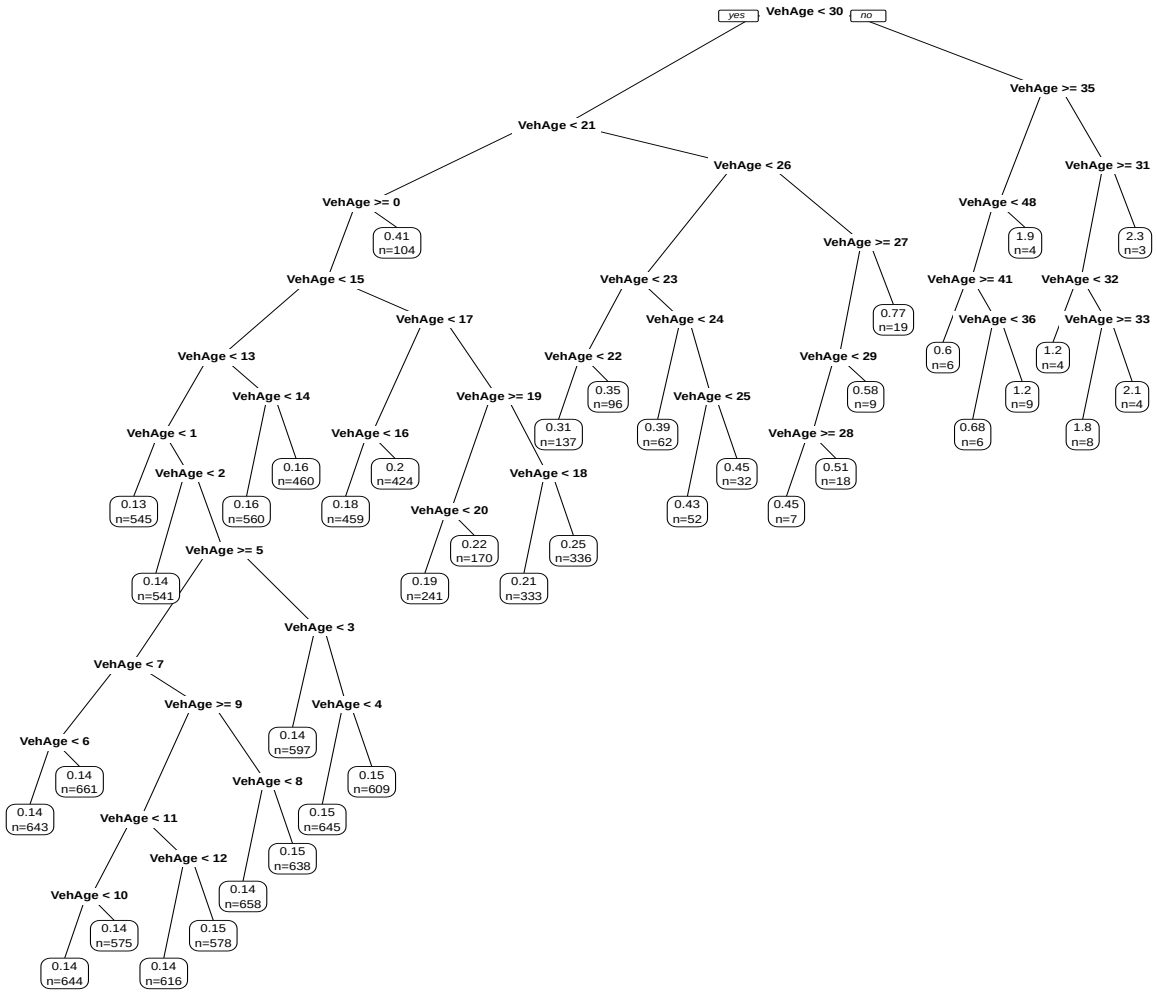
$$\sum_{m=1}^{|\mathcal{T}|} \sum_{i:\mathbf{X}_i \in R_m} \left(Y_i - \widehat{Y}_{R_m}\right)^2 + \alpha \left|\mathcal{T}\right|,$$

  où α est un (hyper-)paramètre de complexité et $|\mathcal{T}|$ représente le nombre de feuilles du sous-arbre $\mathcal{T}$.
- Similaire à la régression Ridge/Lasso : compromis entre le biais et la variance, entre la précision et la parcimonie.
- Le paramètre de complexité peut être estimé par validation croisée.

Exemple 1 : Arbre complet

```
### Ajustement du modèle
arbre3 <- rpart(Y ~ VehAge, data = dataTRAIN,
          control = rpart.control(minsplit = 10, cp = 0))

### Graphique (j'en parle plus loin de cette fonction)
prp(arbre3, extra = 1)
```

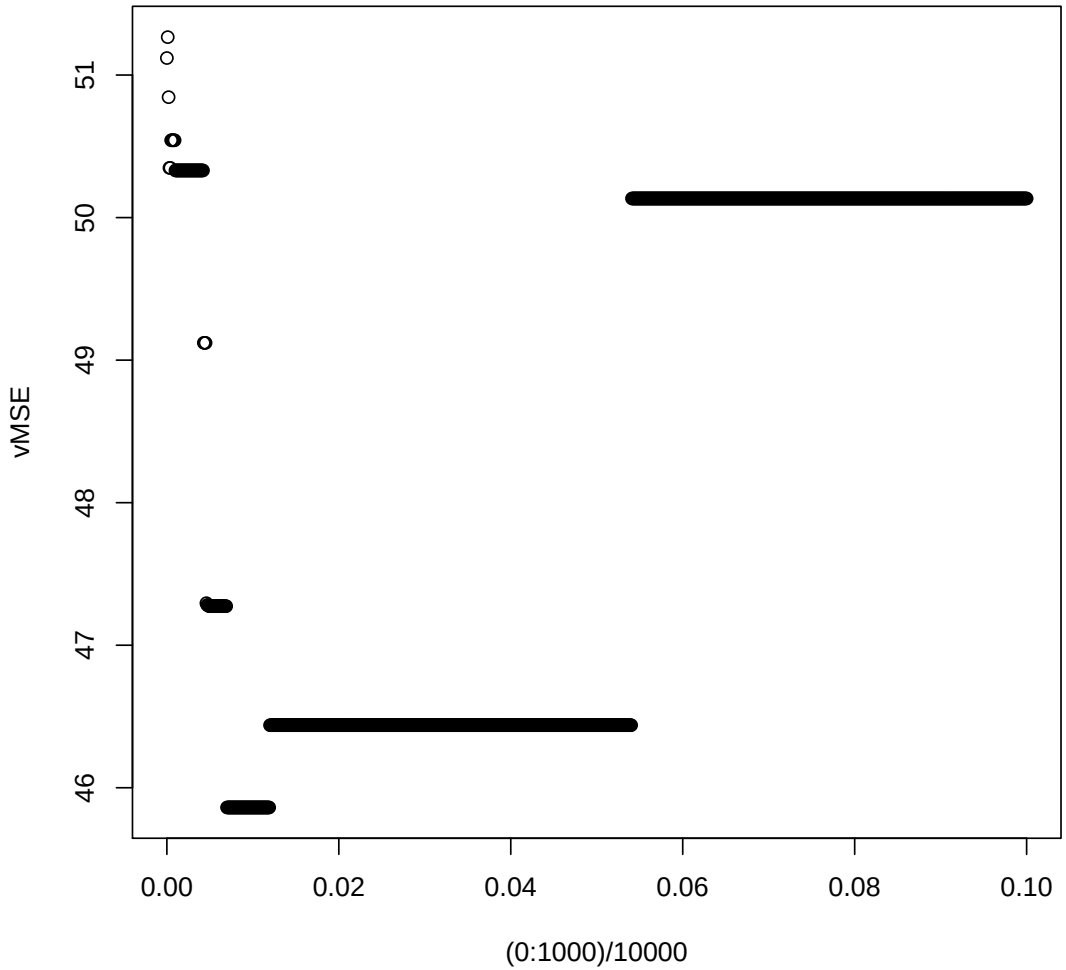
Exemple 1 : Arbre complet



Exemple 1 : Paramètre de complexité

```
FUN <- function(x)
{
AAA <- rpart(Y ~ VehAge, data = dataTRAIN,
             control=rpart.control(cp = x))
sum((predict(AAA, newdata = dataVALID) - dataVALID$Y)^2)
}
vMSE <- sapply((0:1000)/10000, function(x) FUN(x))

plot((0:1000)/10000, vMSE)
```

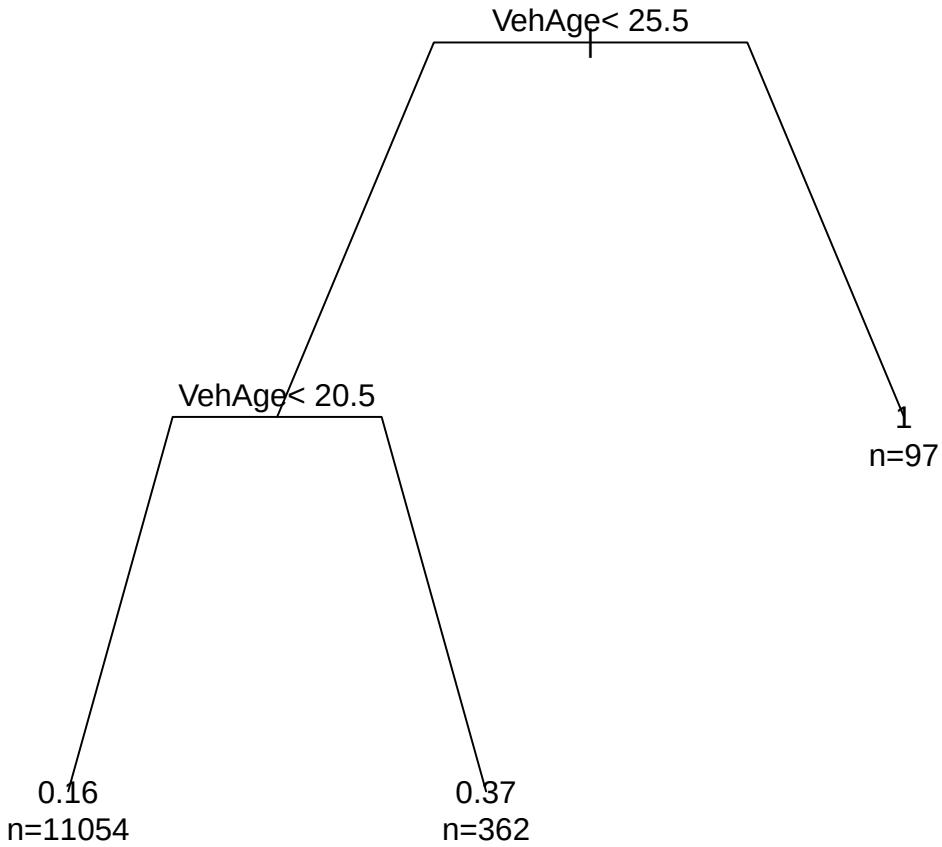
Exemple 1 : Paramètre de complexité



Exemple 1 : Arbre final

```
arbre3a <- rpart(Y ~ VehAge, data = dataTRAIN,
                 control=rpart.control(cp = 0.01))

plot(arbre3a, uniform = TRUE, branch = 0.5, margin = 0.1)
text(arbre3a, all = FALSE, use.n = TRUE,
     splits = TRUE, digits = 2)
```

Exemple 1 : Arbre final



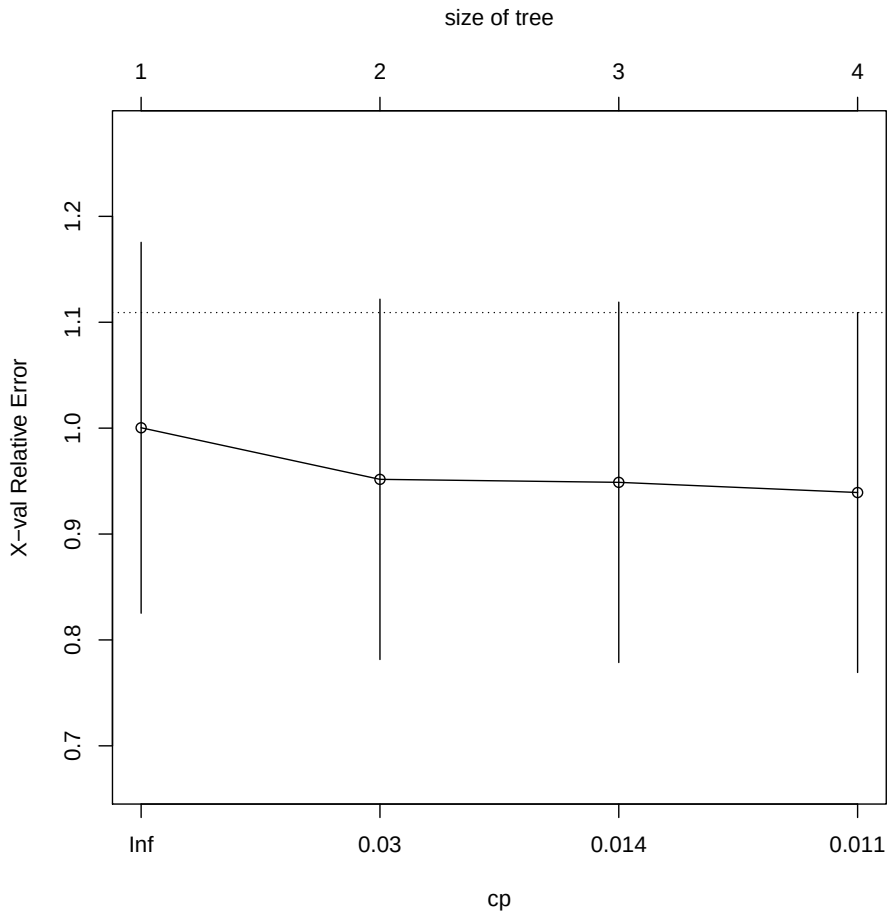
Exemple 1 : Arbre final (automatique)

```
### Le choix se fait par 10-validation croisée
arbre4 <- rpart(Y ~ VehAge + DriverAge, data = dataAGG)
summary(arbre4)

          CP nsplit rel error    xerror      xstd
1 0.05481642      0 1.0000000 1.0003560 0.1751165
2 0.01624138      1 0.9451836 0.9516489 0.1701225
3 0.01204489      2 0.9289422 0.9488334 0.1701539
4 0.01000000      3 0.9168973 0.9392039 0.1699391

Variable importance
   VehAge DriverAge
       80        20
```

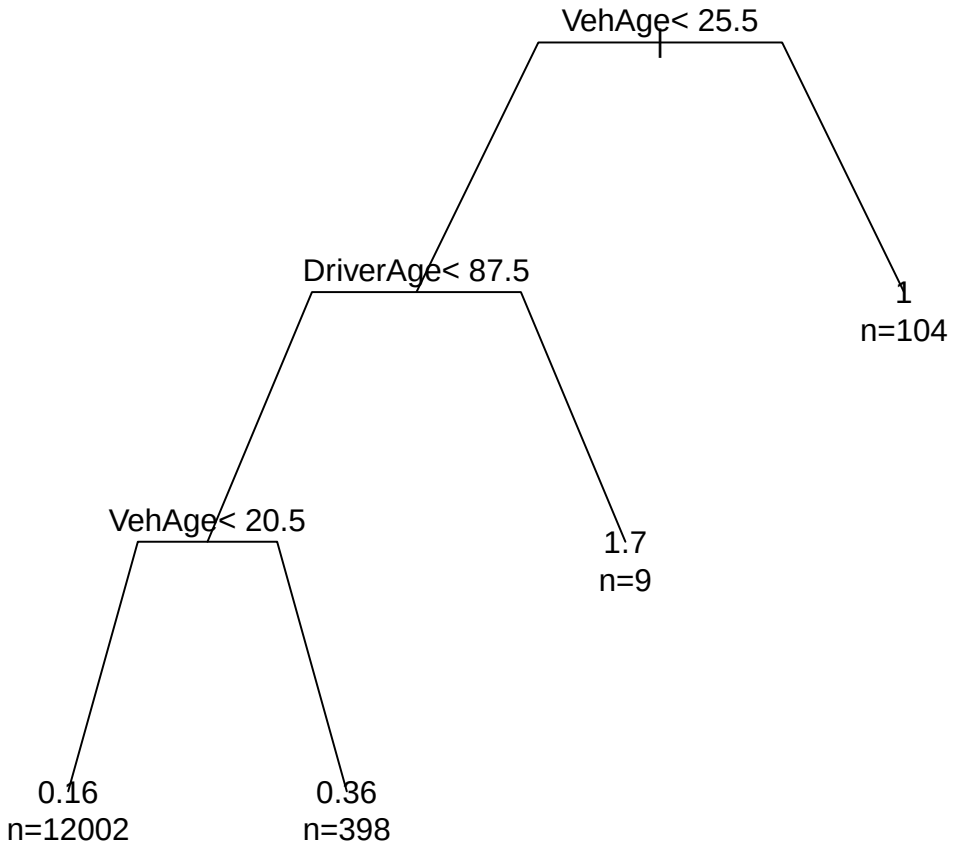
Exemple 1 : Arbre final (automatique)

```
Node number 1: 12513 obs, complexity param=0.05481642
  mean=0.1713315, MSE=0.1073434
  left son=2 (12409 obs) right son=3 (104 obs)
  Primary splits:
      VehAge    < 25.5 to the left,  improve=0.05481642,
      DriverAge < 87.5 to the left,  improve=0.01609504,

Node number 2: 12409 obs, complexity param=0.01624138
  mean=0.164309, MSE=0.08627632
  left son=4 (12400 obs) right son=5 (9 obs)
  Primary splits:
      DriverAge < 87.5 to the left,  improve=0.02037659,
      VehAge    < 20.5 to the left,  improve=0.01493965,
```

Exemple 1 : Arbre final (automatique)



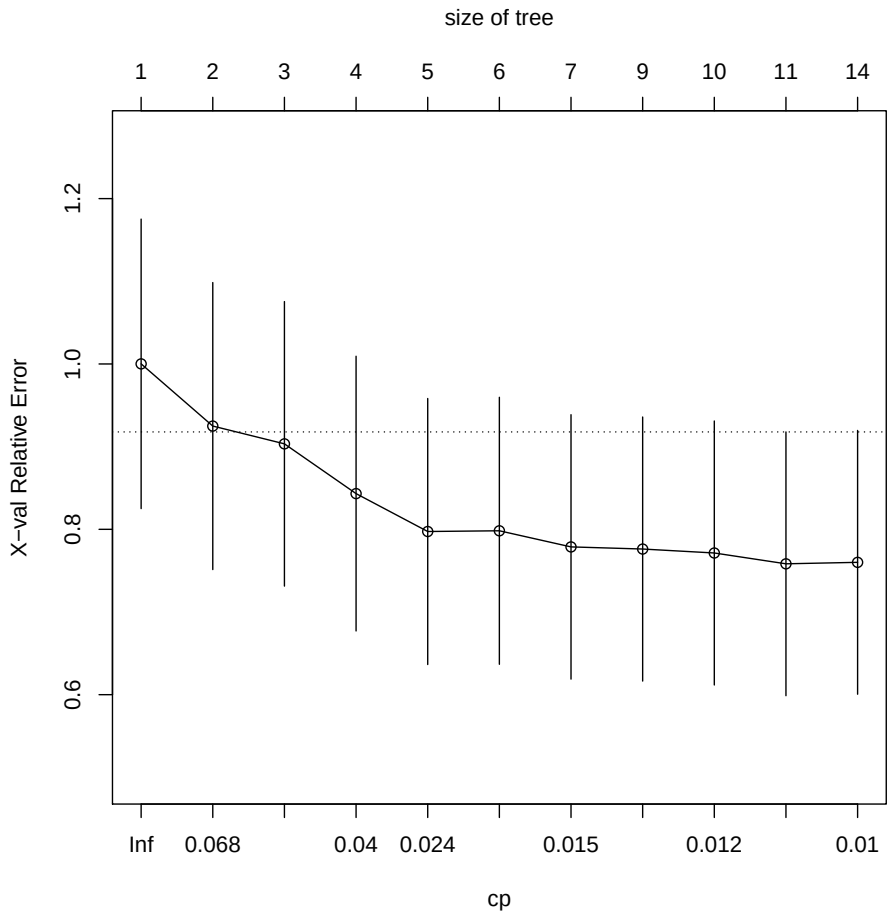
Exemple 1 : Arbre final (automatique)



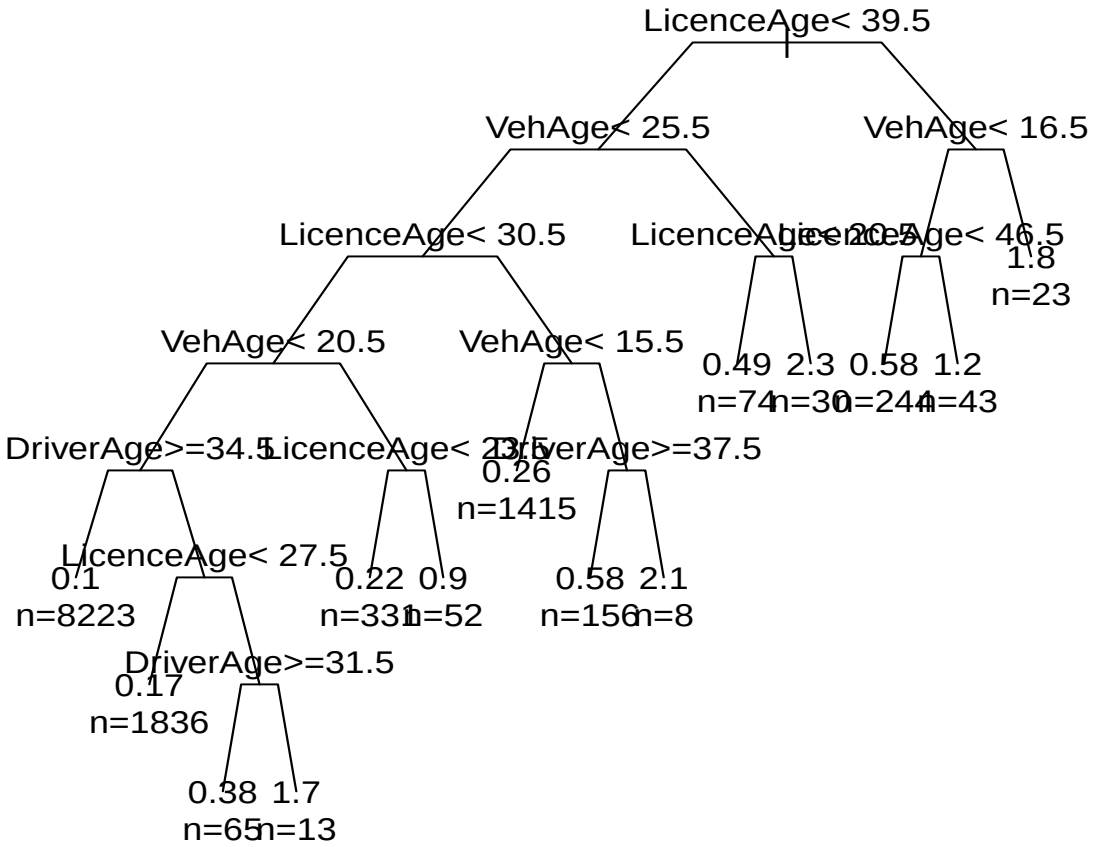
Exemple 1 : Arbre final

```
arbre5 <- rpart(Y ~ VehAge + DriverAge + LicenceAge,
                data = dataAGG)
summary(arbre5)
Variable importance
LicenceAge     VehAge  DriverAge
        58         31         11

Node number 1: 12513 obs, complexity param=0.08106918
  mean=0.1713315, MSE=0.1073434
  left son=2 (12203 obs) right son=3 (310 obs)
  Primary splits:
      LicenceAge < 39.5  to the left,  improve=0.08106918,
      VehAge     < 25.5  to the left,  improve=0.05481642,
      DriverAge  < 87.5  to the left,  improve=0.01609504,
```

Exemple 1 : Arbre final



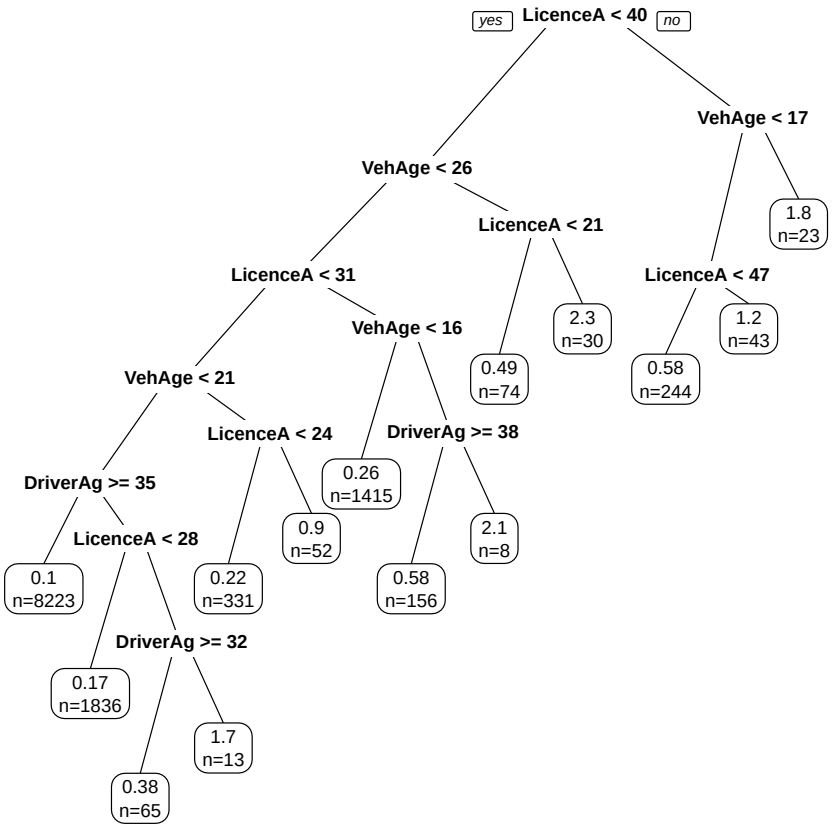
Exemple 1 : Arbre final



Exemple 1 : Arbre final

```
### Pour faire des graphiques plus "clean"

install.packages("rpart.plot")
library(rpart.plot)

prp(arbre5,extra=1)
```

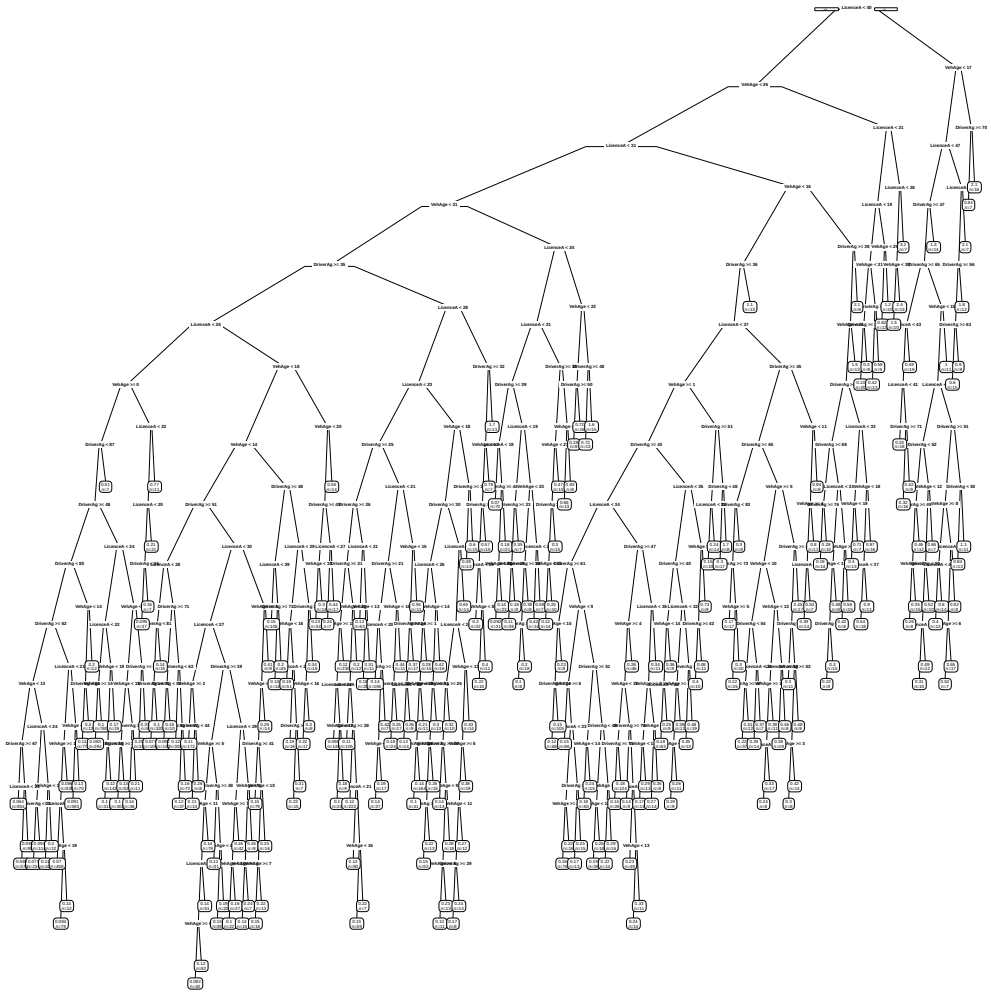
Exemple 1 : Arbre final



Exemple 1 : Arbre final

```
### Approche alternative
arbre5 <- rpart(Y ~ VehAge + DriverAge + LicenceAge,
                data = dataAGG,
                control = rpart.control(cp = 0))

CP <- arbre5$cptable[which.min(arbre5$cptable[,4]),1]
CP

1.899117e-05
arbre6 <- prune(arbre5,cp=CP)
prp(arbre6, extra = 1)
```

Exemple 1 : Arbre final



Construction

La procédure est similaire à celle utilisée pour une problématique de régression. Quelques différences :

- pour chaque observation qui tombe dans la région R_j, une prédiction identique sera faite en utilisant, par exemple, le **mode** des observations de cette classe ;
- les variables réponses n'étant pas numériques, l'erreur quadratique moyenne (ou la somme des erreurs au carré) ne peut par être utilisée comme fonction objectif.

Fonction objectif

- Pour une région r, si le mode est utilisée comme prédiction, on peut utiliser le **taux d'erreur de classification** donné par

$$E_r = 1 - \max_{g \in \mathcal{G}} \widehat{p}_{rg},$$

où $\widehat{p}_{rg}$ est la proportion d'observations i pour lesquelles $\mathbf{X}_i \in r$ et dont la variable réponse est g.

- Simple à mettre en oeuvre mais plus difficile à interpréter.

Fonction objectif

- L'**index Gini** associé à la région r est donné par

$$G_r = \sum_{g \in \mathcal{G}} \widehat{p}_{rg}(1 - \widehat{p}_{rg}).$$

- On remarque que l'index Gini prend des petites valeurs si $\widehat{p}_{rg}$ est soit près de 0, soit près de 1 : mesure de l'homogénéité (ou de la pureté) du noeud.
- Petites valeurs $\rightarrow$ le noeud contient essentiellement des observations provenant d'une seule classe.

Fonction objectif

- L'**entropie** associée à la région r est donnée par

$$D_r = -\sum_{g \in \mathcal{G}} \widehat{p}_{rg} \ln(\widehat{p}_{rg}) \mathbb{I}_{\{\widehat{p}_{rg} \neq 0\}},$$

où

$$\ln(\widehat{p}_{rg}) \mathbb{I}_{\{\widehat{p}_{rg} \neq 0\}} = \begin{cases} \ln(\widehat{p}_{rg}), & \widehat{p}_{rg} \neq 0 \\ 0, & \text{sinon.} \end{cases}$$

- À nouveau, on remarque que l'entropie prend des petites valeurs si $\widehat{p}_{rg}$ est soit près de 0, soit près de 1 : mesure de l'homogénéité (ou de la pureté) du noeud.

Division binaire récursive

Première étape : on sélectionne la variable explicative X_j et le point c tels que la division en deux régions

$$R_1(j, c) = \{X_1, \ldots, X_J | X_j < c\}$$
$$R_2(j, c) = \{X_1, \ldots, X_J | X_j \geq c\}$$

conduise à la plus grande réduction possible de l'entropie (ou de l'index Gini) agrégée.

Division binaire récursive

- L'entropie (ou l'index Gini) agrégée est simplement la somme pondérée des entropies des deux régions :

$$D_r^* = \pi_1 D_{R_1} + \pi_2 D_{R_2},$$

où D_{R_j} est l'entropie de la région j et

$$\pi_j = \frac{|R_j|}{|R_1| + |R_2|}, \quad j = 1, 2.$$

Division binaire récursive

- Lors de la **construction** de l'arbre initial $\mathcal{T}_0$, on utilise presque toujours soit l'index Gini, soit l'entropie.
- Lors de l'élagage de l'arbre, on utilise
  - le taux d'erreur de classification si on vise principalement la précision des prédictions (plutôt que l'interprétation des résultats) ;
  - l'index Gini ou l'entropie sinon.

Exemple 2

On considère un noeud avec 10 observations :

$$\{A, A, A, A, A, A, A, A, B, B\}$$

et pour lequel la classe prédite est A. Si on considère les sous-groupes

$$R_1 = \{A, A, A, A, A\} \text{ et } R_2 = \{A, A, A, B, B\},$$

évaluer l'impact de cette division sur la pureté du noeud.

Exemple 3

Base de données *credit* disponible dans la librairie *CASdatasets* contient de l'information sur 1 000 dossiers de crédits. La base contient une variable réponse binaire et une vingtaine de variables explicatives :

- **class** : 0 pour bon crédit et 1 pour mauvais crédit (variable réponse) ;
- **credit_amount** : le montant accordé du crédit ;
- **savings** : le montant des économies : **A61** pour moins de 100, **A62** pour entre 100 et 500, **A63** pour entre 500 et 1 000, **A64** pour plus de 1 000 et **A65** pour inconnu ;
- **age** : l'âge de l'emprunteur ;
- ...

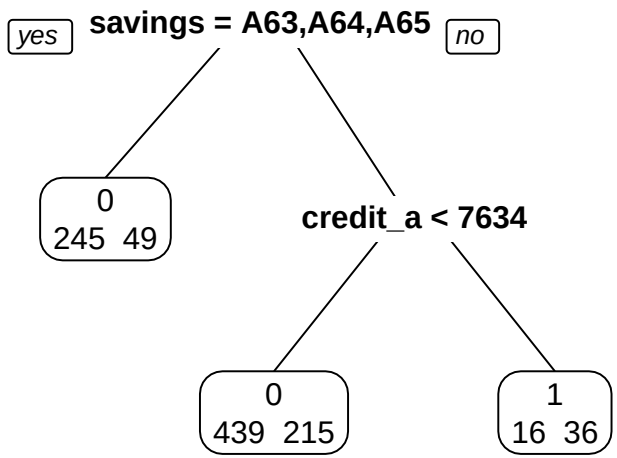
Exemple 3 : Préparation des données

```
library(CASdatasets)

data(credit)
dataC <- credit
```

Exemple 3 : Construction de l'arbre

```
arbre1 <- rpart(as.factor(class) ~ credit_amount + savings
                + age, data = dataC,
                control = rpart.control(cp = 0))
CP <- arbre1$cptable[which.min(arbre1$cptable[,4]),1]
CP

0.008333333

arbre2 <- prune(arbre1, cp = CP)
prp(arbre2, extra = 1)
```

Exemple 3 : Arbre



Exemple 3 : Prédictions

```
predict(arbre2, dataC[1:5,])

          0         1
1 0.8333333 0.1666667
2 0.6712538 0.3287462
3 0.6712538 0.3287462
4 0.3076923 0.6923077
5 0.6712538 0.3287462

predict(arbre2, dataC[1:5,], type = "class")

1 2 3 4 5
0 0 0 1 0
```

Exemple 3 : Matrice de confusion

```
table(dataC$class, predict(arbre2, dataC, type="class"))
```

```
     0   1
0  684  16
1  264  36
```