Oniris
VetAgroBio Nantes
ÉCOLE NATIONALE

ECF939 - Science des données avancées

Guillaume Franchi

Cursus Ingénieur 3ème année

Arbres de décision et forêts aléatoires

1. Contexte et problématique

Exemple (*Huile d'olive*)

Dans un souci de traçabilité alimentaire, on s'intéresse à la provenance de l'huile d'olive italienne produite dans 9 régions différentes : *Calabre, Ligurie_Est, Ligurie_Ouest, Ombrie, Pouilles_Nord, Pouilles_Sud, Sardaigne_Cote, Sardaigne_Interieur, Sicile.*

Exemple (*Huile d'olive*)

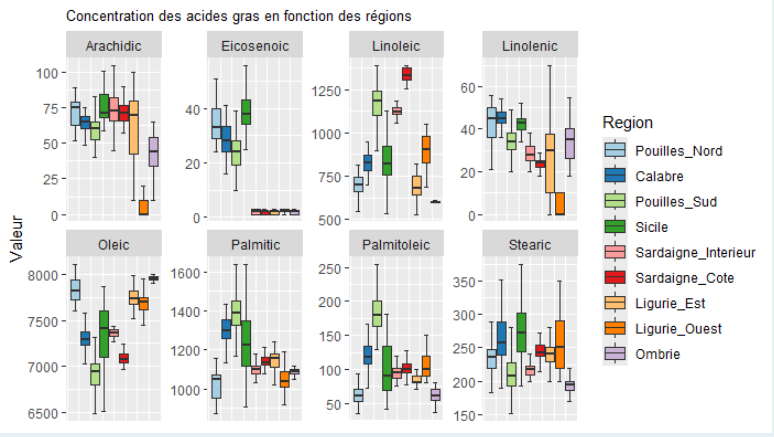
Dans un souci de traçabilité alimentaire, on s'intéresse à la provenance de l'huile d'olive italienne produite dans 9 régions différentes.

Pour ce faire, on s'intéresse à la concentration en 8 acides gras composant chacune des 572 huiles analysées.

Region	Palmitic	Palmitoleic	Stearic	Oleic	Linoleic	Linolenic	Arachidic	Eicosenoic
Ombrie	1095	55	200	7940	600	35	45	3
Pouilles_Sud	1413	202	205	6920	1165	36	46	13
Ligurie_Ouest	960	80	240	7950	740	10	20	2
Sicile	1194	135	263	7277	889	44	95	41
Sardaigne_Interieur	1060	111	231	7363	1149	20	65	1
Pouilles_Sud	1434	185	189	6771	1269	30	62	25

Exemple (*Huile d'olive*)

On cherche ici à résoudre un problème de **classification** : déterminer la région de provenance d'une huile en fonction de ses concentrations d'acides gras.



Exemple (*Iris*)

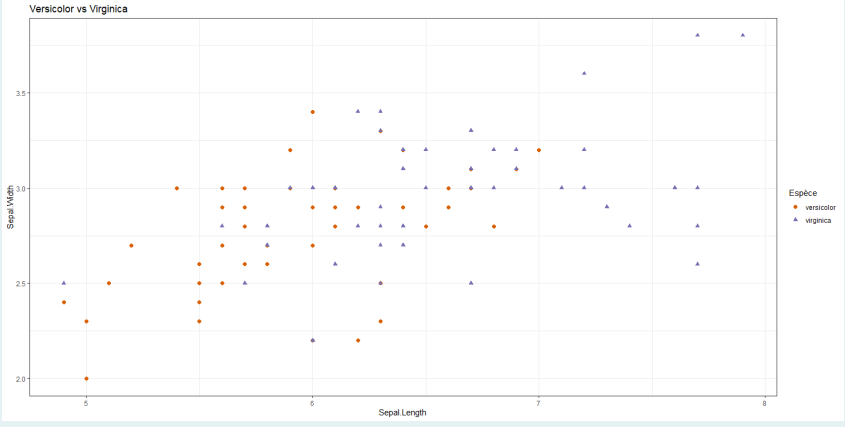
Autre problème de classification : on cherche à distinguer deux espèces d'iris - *virginica* et *versicolor* - en fonction de la longueur et de la largeur des sépales.

(a) Iris virginica. *Thomas G. Barnes.*

(b) Iris versicolor. *Charles et Diane Peirce.*

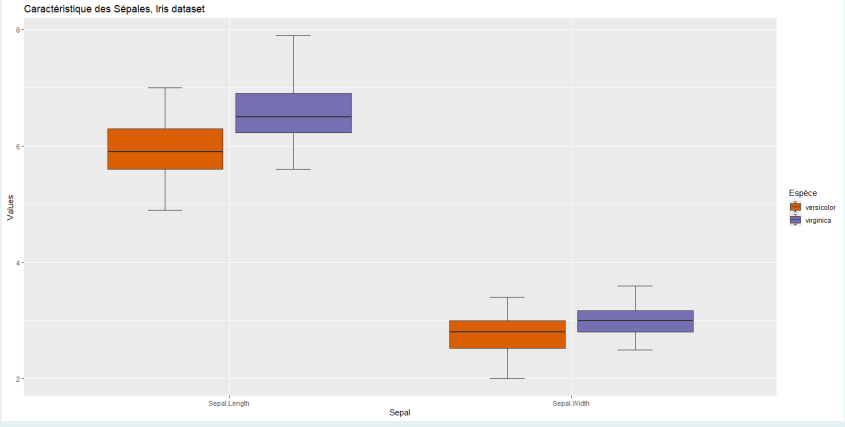
Exemple (*Iris*)

Autre problème de classification : on cherche à distinguer deux espèces d'iris - *virginica* et *versicolor* - en fonction de la longueur et de la largeur des sépales.



Exemple (*Iris*)

Autre problème de classification : on cherche à distinguer deux espèces d'iris - *virginica* et *versicolor* - en fonction de la longueur et de la largeur des sépales.



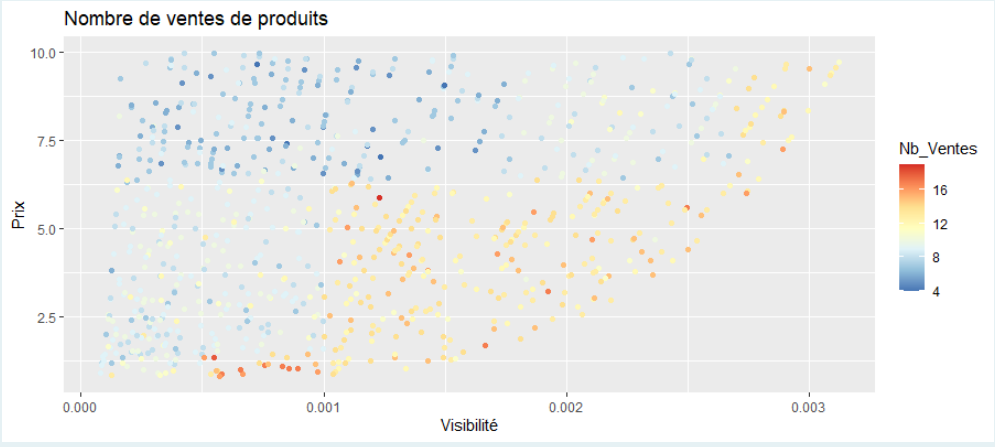
Exemple (*Prédiction de ventes*)

Un problème de régression : dans un supermarché, on cherche à prédire le nombre de ventes d'un produit en fonction de sa visibilité en magasin et de son prix.

Id	Price	Visibility	Nb_Ventes
DRA1	4.9400	0.0003	11.0000
DRA2	5.4200	0.0005	10.0000
DRA3	2.5800	0.0010	10.0000
DRB1	6.0500	0.0015	13.0000
DRB2	2.7400	0.0006	9.0000
DRC1	8.9700	0.0015	9.0000

Exemple (*Prédiction de ventes*)

Un problème de régression : dans un supermarché, on cherche à prédire le nombre de ventes d'un produit en fonction de sa visibilité en magasin et de son prix.



◎ Objectif : Déterminer la valeur d'une réponse y à partir de p variables $x = (x_1, \ldots, x_p)$ pouvant être de natures différentes.

- Réponse y qualitative : problème de **classification**.
- Réponse y quantitative : problème de **régression**.

⚙ Plusieurs méthodes de machine learning existent pour traiter ces problèmes.

- Classification : Régression logistique multinomiale, KNN, K-means, SVM, Réseaux de neurones,...
- Régression : Régression linéaire, GLM, GAM, Réseaux de neurones,...

💡 On va présenter ici les **arbres de décision** via la méthode **CART** *(Classification And Regression Tree, Breiman et al. (1984)*).

💡 On s'intéresse uniquement à la qualité de prévision de la réponse y, sans chercher de lien entre les variables, ni chercher d'interprétation.

2. Méthode CART

⚙ Les arbres de décision construits par la méthode CART sont des algorithmes de prédiction de type « *Instance Based* », i.e. basés sur les observations, et non sur un modèle théorique.

⚙ Etant donné jeu d'apprentissage constitué de n observations $(y_1, x_1), \ldots, (y_n, x_n)$, on crée de façon itérative une partition de l'espace des variables ($\mathbb{R}^p$), en plusieurs régions.

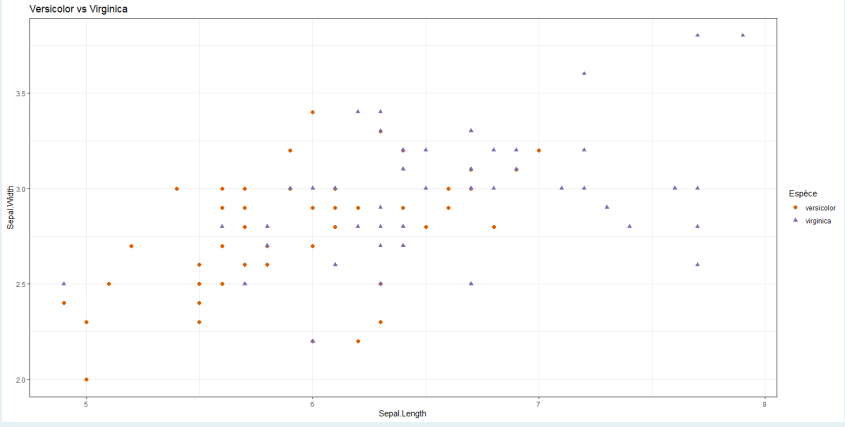
Prédiction

On prédit la valeur $\widehat{y}$ d'une nouvelle observation $\widehat{x}$ en faisant :

- un **vote à la majorité** parmi les réponses y_i associées aux variables x_i dans la même région que $\widehat{x}$ *(Classification)* ;
- la **moyenne des réponses** y_i associées aux variables x_i dans la même région que $\widehat{x}$ *(Régression)*.

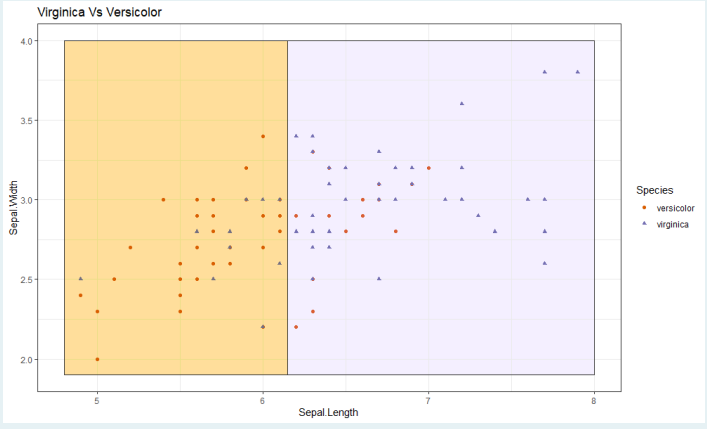
Exemple (*iris*)

On cherche à classifier les fleurs d'iris à partir de la longueur et de la largeur de leurs sépales.



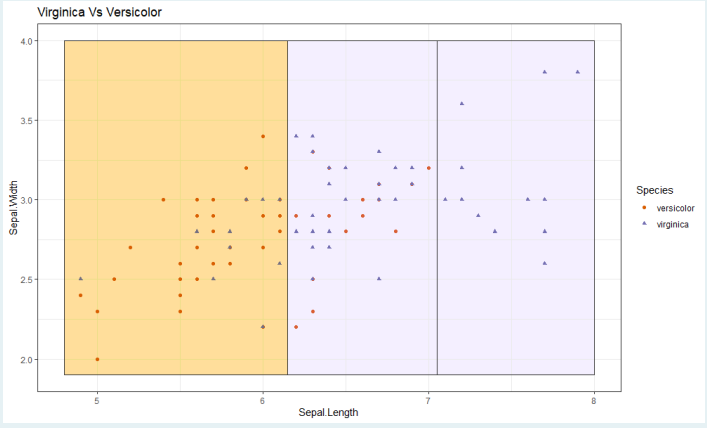
Exemple (*iris*)

On cherche à classifier les fleurs d'iris à partir de la longueur et de la largeur de leurs sépales.



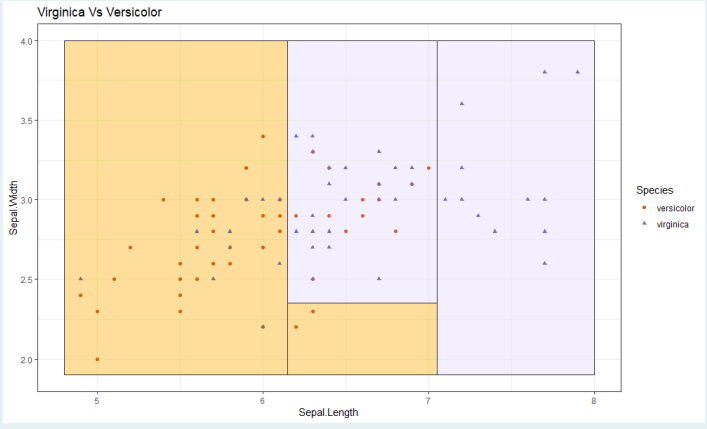
Exemple (*iris*)

On cherche à classifier les fleurs d'iris à partir de la longueur et de la largeur de leurs sépales.



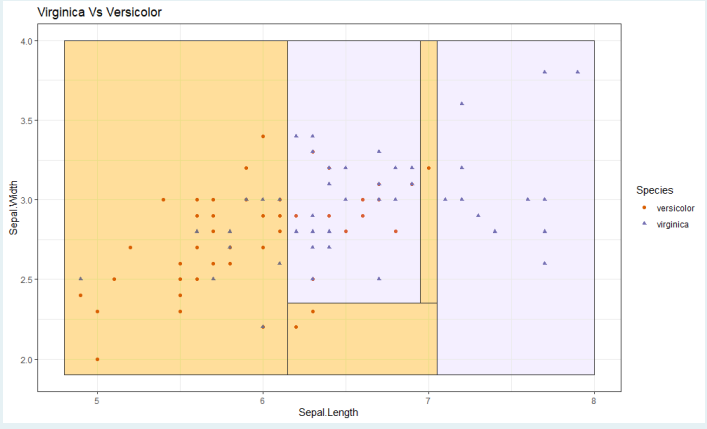
Exemple (*iris*)

On cherche à classifier les fleurs d'iris à partir de la longueur et de la largeur de leurs sépales.



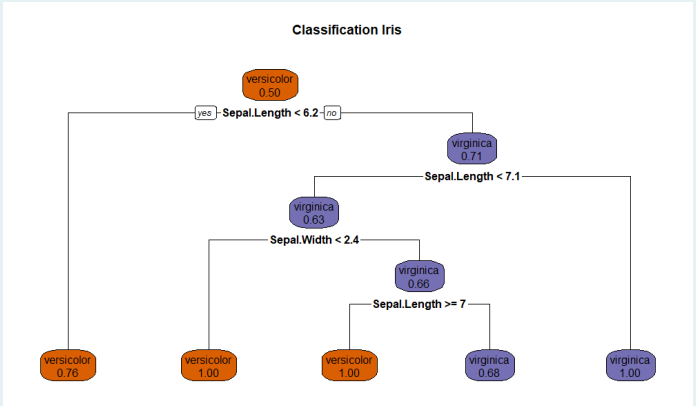
Exemple (*iris*)

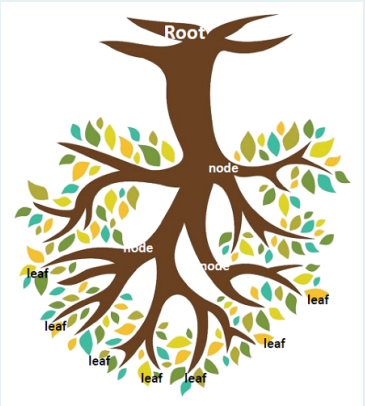
On cherche à classifier les fleurs d'iris à partir de la longueur et de la largeur de leurs sépales.



Exemple (*iris*)

Le partitionnement précédent donne lieu à la règle de classification suivante.

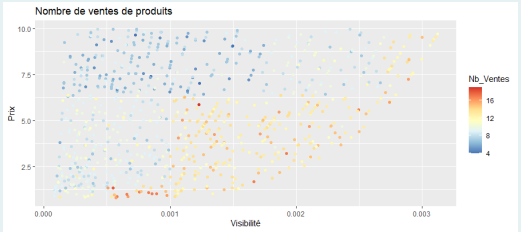


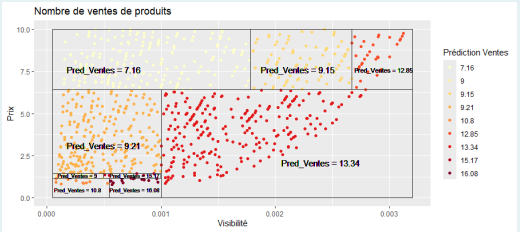


On a ici un **arbre de classification**.

Exemple (*Prédiction de ventes*)

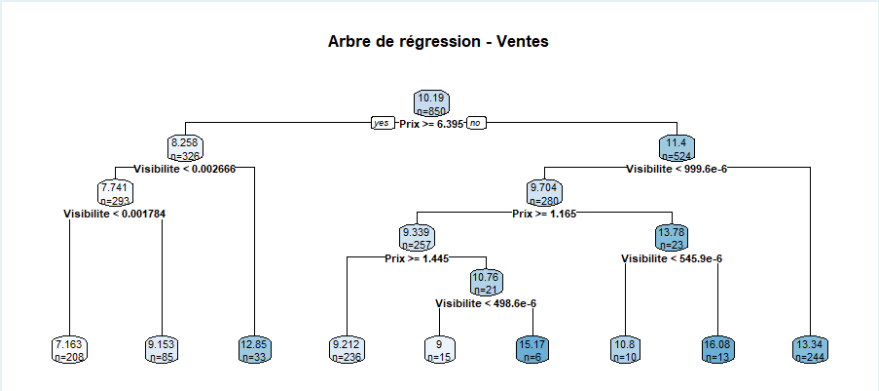
Dans le cas où l'on cherchait à déterminer le nombre d'articles vendus dans un supermarché, la méthode CART nous donne le partitionnement ci-dessous.





Exemple (*Prédiction de ventes*)

Ce partitionnement correspond à l'**arbre de régression** :



⚙ A chaque noeud, le jeu d'apprentissage est coupé en deux en choisissant :

- une variable $j \in \{1, \ldots, p\}$;
- un seuil de coupure s.

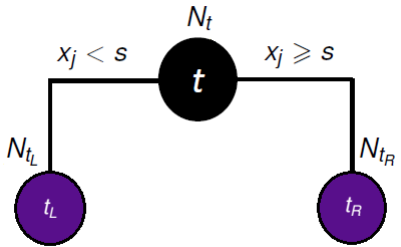
⚙ Comment choisir la variable et le seuil de coupure ?

💡 Maximiser la diminution globale de l'**impureté**, indicateur mesurant l'hétérogénéité des variables réponses pour une région de $\mathbb{R}^p$.

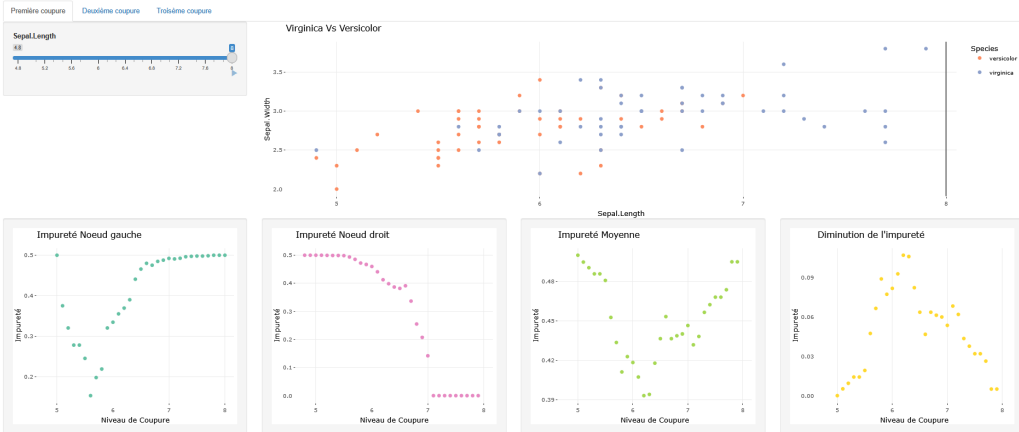
Critère de coupure

- Soit $i(t)$ une mesure de l'impureté pour un noeud t.
- A chaque noeud t, on détermine les paramètres j et s de manière à maximiser la diminution de l'impureté pour le noeud t

$$\Delta_t(j,s) = i(t) - \frac{N_{t_L}}{N_t} i(t_L) - \frac{N_{t_R}}{N_t} i(t_R).$$



Iris dataset - Classification



Mesures d'impureté

Arbre de Classification

Réponse qualitative y, ayant K modalités.

- Indice de Gini :

$$i(t) = \sum_{k=1}^{K} p_{t,k}(1 - p_{t,k}).$$

- Entropie de Shannon :

$$i(t) = -\sum_{k=1}^{K} p_{t,k} \log\left(p_{t,k}\right).$$

$p_{t,k}$ est la proportion de réponses k dans le noeud t.

Arbre de Régression

Réponse quantitative y.

- Variance :

$$i(t) = \frac{1}{N_t} \sum_{i=1}^{N_t} \left(y_{i,t} - \overline{y}(t)\right)^2$$

$y_{1,y}, \ldots, y_{N_t,t}$ sont les réponses du noeud t.

$\overline{y}(t)$ est la moyenne des réponses du noeud t.

Remarques

- Pour l'indice de Gini

$$i(t) = \sum_{k=1}^{K} p_{t,k}(1 - p_{t,k})$$

ou l'entropie de Shannon

$$i(t) = -\sum_{k=1}^{K} p_{t,k} \log\left(p_{t,k}\right),$$

plus les $p_{t,k}$ sont proches de 0 ou 1, plus la mesure d'impureté est proche de 0.

- Dans le cas d'un arbre de régression, maximiser la diminution de l'impureté revient à minimiser la variance intra-groupes.

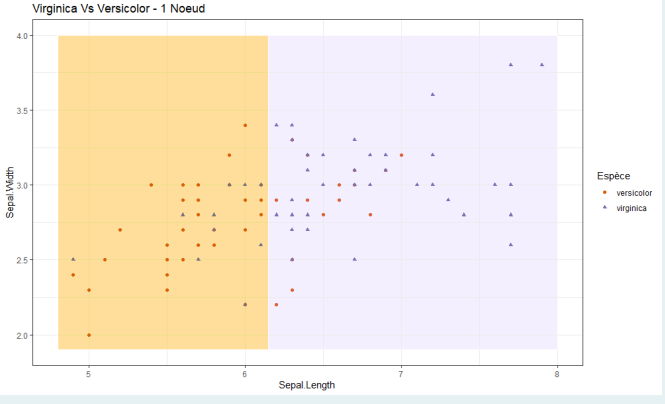
⚙ Jusqu'où aller ?

- Tous les noeuds terminaux sont purs.
- Fixer un nombre minimal d'observations pour pouvoir couper un noeud.
- Fixer un seuil minimal pour le paramètre de complexité :

  Il s'agit de s'arrêter lorsque la diminution de l'impureté attendue pour un noeud est trop faible.

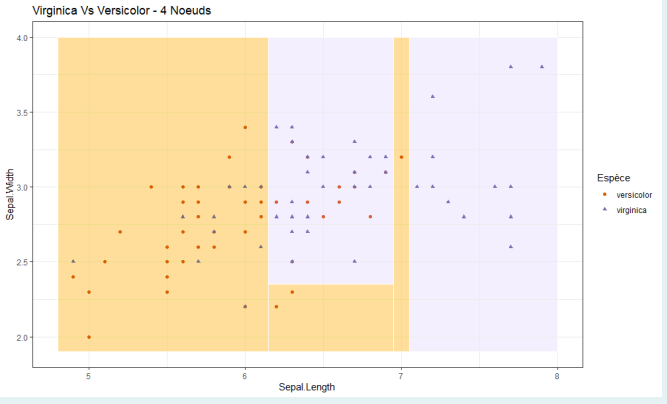
Risque de sur-ajustement

Si on fixe un critère d'arrêt trop faible *(noeuds terminaux purs, 2 observations pour couper un noeud, paramètre de complexité faible)*, on prend le risque du **sur-apprentissage**.



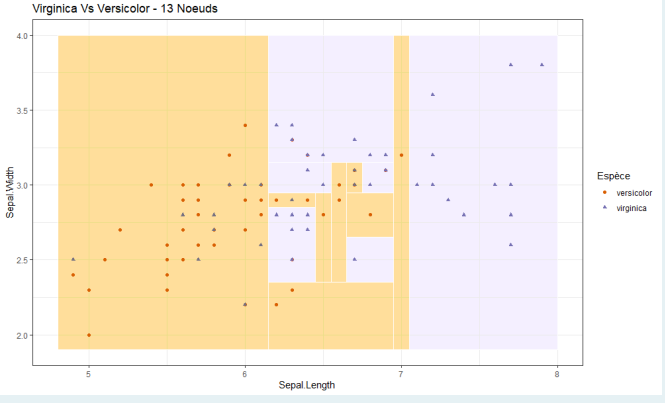
Risque de sur-ajustement

Si on fixe un critère d'arrêt trop faible *(noeuds terminaux purs, 2 observations pour couper un noeud, paramètre de complexité faible)*, on prend le risque du **sur-apprentissage**.



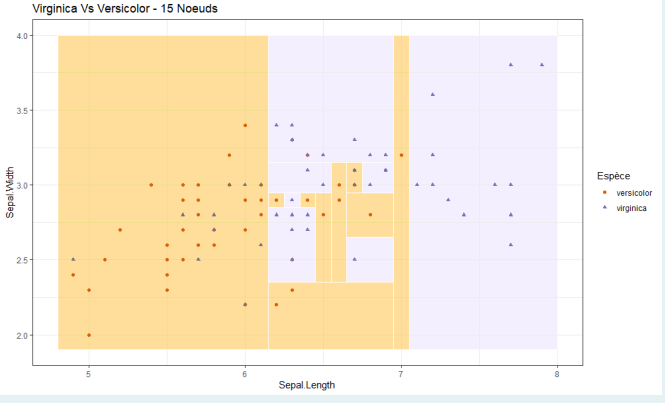
Risque de sur-ajustement

Si on fixe un critère d'arrêt trop faible *(noeuds terminaux purs, 2 observations pour couper un noeud, paramètre de complexité faible)*, on prend le risque du **sur-apprentissage**.



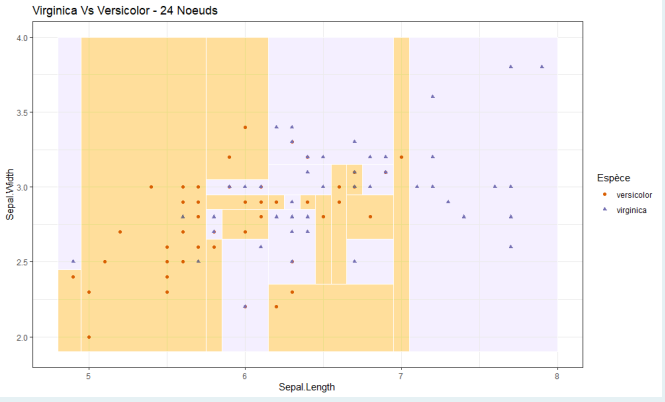
Risque de sur-ajustement

Si on fixe un critère d'arrêt trop faible *(noeuds terminaux purs, 2 observations pour couper un noeud, paramètre de complexité faible)*, on prend le risque du **sur-apprentissage**.



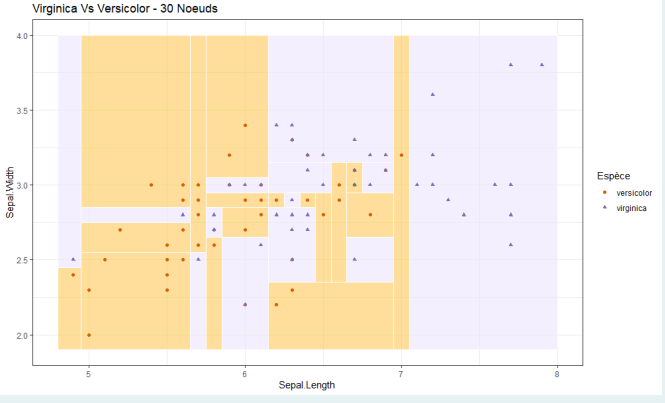
Risque de sur-ajustement

Si on fixe un critère d'arrêt trop faible *(noeuds terminaux purs, 2 observations pour couper un noeud, paramètre de complexité faible)*, on prend le risque du **sur-apprentissage**.



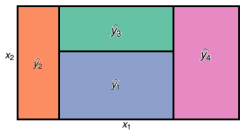
Risque de sur-ajustement

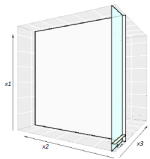
Si on fixe un critère d'arrêt trop faible *(noeuds terminaux purs, 2 observations pour couper un noeud, paramètre de complexité faible)*, on prend le risque du **sur-apprentissage**.



Prédiction

- Un arbre de décision CART permet un découpage en régions *(hyper-cubes)* de l'espace des prédicteurs.
- Chaque région est associée à une fonction de prédiction constante.





Conséquences

- Un arbre peu profond *(peu de noeuds terminaux)*, simple, n'offrira qu'un nombre limité de prédictions possibles $\Longrightarrow$ *Faible capacité prédictive*.
- Un arbre profond, complexe, permettra de prédire parfaitement les observations du jeu d'apprentissage $\Longrightarrow$ *Risque de sur-ajustement*.

Solution : Elagage

- On construit d'abord un arbre T_{max} complexe (*profond*), de taille $|T_{max}|$ noeuds terminaux (*feuilles*).
- Pour un noeud $t \in \{1, \dots, |T_{max}|\}$, on définit le risque d'ajustement dans ce noeud par
  - **Régression :** $R_t(T_{max}) = \frac{1}{N_t} \sum_{i=1}^{N_t} (y_{i,t} - \overline{y}(t))^2$.
  - **Classification :** $R_t(T_{max}) = \frac{1}{N_t} \sum_{i=1}^{N_t} \mathbb{1}_{y_{i,t} \neq \widehat{y}(t)}$.

Fonction de coût

On définit la fonction de coût d'un sous-arbre $T \subset T_{max}$:

$$C_\alpha(T) = \sum_{t=1}^{|T|} N_t R_t(T) + \alpha |T|,$$

où $\alpha > 0$ est un paramètre à déterminer (*tuning parameter*).

Remarques

- L'idée est de trouver un sous-arbre $T \subset T_{max}$ minimisant la fonction de coût :

$$C_\alpha(T) = \sum_{t=1}^{|T|} N_t R_t(T) + \alpha |T|.$$

- Cela revient à trouver un compromis entre la capacité prédictive d'un arbre et sa complexité.

Arbre optimal

Le choix du paramètre α et de l'arbre optimal minimisant la fonction de coût se fait en général par **validation croisée**.

Exemple (*Huile d'olive*)

On revient à l'exemple où l'on cherche à déterminer la région de provenance d'une huile en fonction de ses concentrations en acides gras.

- On construit d'abord un arbre profond avec la fonction `rpart` de R.

```
library(rpart)
oil_tree <-rpart(data=df,
                 formula = Region~.,
                 control = rpart.control(minsplit = 2,cp=0.001))
```

Exemple (*Huile d'olive*)

- On peut ensuite afficher la collection de sous-arbres construits.

```
printcp(oil_tree)
```

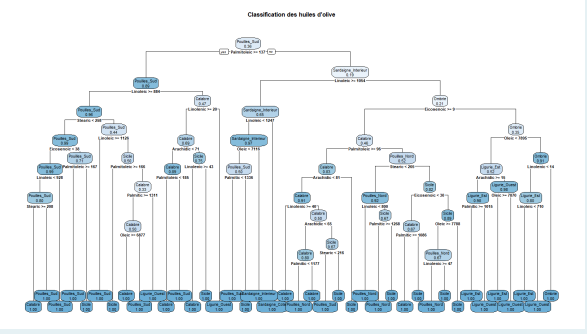
```
Classification tree:
rpart(formula = Region ~ ., data = df, control = rpart.control(minsplit = 2,cp = 0.001))

Variables actually used in tree construction:
[1] Arachidic Eicosenoic Linoleic Linolenic Oleic Palmitic Palmitoleic Stearic

Root node error: 366/572 = 0.63986

n= 572
```

	CP	nsplit	rel error	xerror	xstd		CP	nsplit	rel error	xerror	xstd
1	0.1721311	0	1.0000000	1.00000	0.03136	9	0.0163934	9	0.1147541	0.18306	0.02101
2	0.1393443	1	0.8278689	0.84153	0.03257	10	0.0136612	10	0.0983607	0.15574	0.01957
3	0.1243169	2	0.6885246	0.73497	0.03261	11	0.0109290	11	0.0846995	0.15574	0.01957
4	0.1092896	4	0.4398907	0.52732	0.03089	12	0.0054645	12	0.0737705	0.13115	0.01811
5	0.0901639	5	0.3306011	0.35246	0.02731	13	0.0027322	13	0.0683060	0.14754	0.01910
6	0.0601093	6	0.2404372	0.27322	0.02481	14	0.0013661	37	0.0027322	0.16120	0.01987
7	0.0437158	7	0.1803279	0.24590	0.02379	15	0.0010000	39	0.0000000	0.16393	0.02002
8	0.0218579	8	0.1366120	0.21311	0.02242						

Exemple (*Huile d'olive*)

- On peut alors afficher l'arbre crée, (trop ?) profond et complexe.

```
library(rpart.plot)
rpart.plot(oil_tree,type=2,
           extra=8,
           cex=0.5,
           box.palette = "Blues",
           main="Classification des huiles d'olive")
```



Exemple (*Huile d'olive*)

- On peut éviter cet écueil en élaguant notre arbre. On choisit d'abord un arbre optimal *(par validation croisée)*.

```
printcp(oil_tree)
Classification tree:
rpart(formula = Region ~ ., data = df, control = rpart.control(minsplit = 2,cp = 0.001))

Variables actually used in tree construction:
[1] Arachidic Eicosenoic Linoleic Linolenic Oleic Palmitic Palmitoleic Stearic

Root node error: 366/572 = 0.63986

n= 572
```

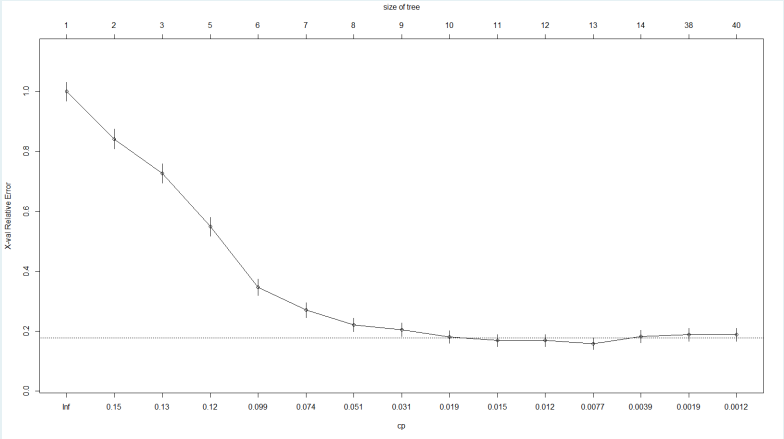
Erreur CV

	CP	nsplit	rel error	xerror	xstd		CP	nsplit	rel error	xerror	xstd
1	0.1721311	0	1.0000000	1.00000	0.03136	9	0.0163934	9	0.1147541	0.18306	0.02101
2	0.1393443	1	0.8278689	0.84153	0.03257	10	0.0136612	10	0.0983607	0.15574	0.01957
3	0.1243169	2	0.6885246	0.73497	0.03261	11	0.0109290	11	0.0846995	0.15574	0.01957
4	0.1092896	4	0.4398907	0.52732	0.03089	12	0.0054645	12	0.0737705	0.13115	0.01811
5	0.0901639	5	0.3306011	0.35246	0.02731	13	0.0027322	13	0.0683060	0.14754	0.01910
6	0.0601093	6	0.2404372	0.27322	0.02481	14	0.0013661	37	0.0027322	0.16120	0.01987
7	0.0437158	7	0.1803279	0.24590	0.02379	15	0.0010000	39	0.0000000	0.16393	0.02002
8	0.0218579	8	0.1366120	0.21311	0.02242						

Exemple (*Huile d'olive*)

- On peut aussi afficher le résultat de cette validation croisée.

```
plotcp(oil_tree)
```



Exemple (*Huile d'olive*)

- On choisit ensuite le coefficient de complexité optimal.

```
library(dplyr)
cp_optim <- oil_tree$cptable %>%
  as.data.frame() %>%
  arrange(xerror) %>%
  slice(1) %>%
  select(CP) %>%
  as.numeric()
# $cptable pour extraire la table des coefficients de complexité
```

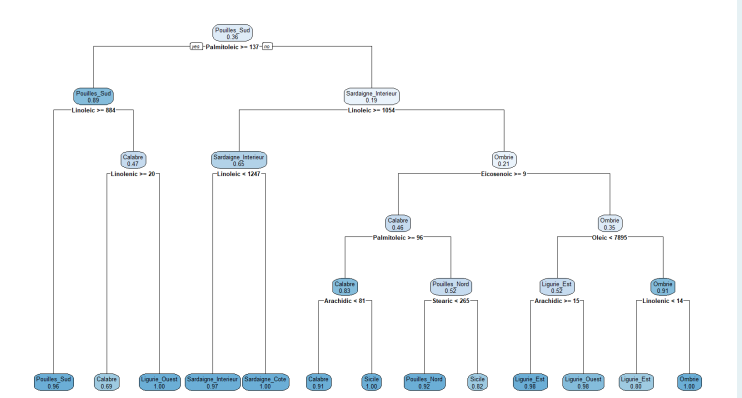
- Et on élague l'arbre à partir de ce coefficient.

```
oil_tree_optim <- prune(oil_tree,cp=cp_optim)
```

Exemple (*Huile d'olive*)

- On peut afficher l'arbre obtenu.

```
rpart.plot(oil_tree_optim,type=2,extra = 8,
           box.palette = "Blues",cex=0.8)
```



Exemple (*Huile d'olive*)

- Les prédictions du modèle peuvent être obtenues avec la fonction `predict()`.

```
df_pred <- df %>%
  mutate(Prediction = predict(oil_tree_optim,
  newdata = df,type = "class"))

error_pred <- df_pred %>%
  reframe(error_classif = Prediction !=Region) %>%
  sum()/nrow(df_pred)

error_pred

[1] 0.04370629
```

- Ici, 4,37% des observations sont mal classées par l'arbre sur les données d'entraînement.

Résumé méthode CART

Avantages

- Méthode simple, facile à mettre en oeuvre.
- Fonctionne en régression ET en classification.
- Résultats facilement interprétables *(si arbre peu profond)*.
- Peut prendre en compte les effets non linéaires et effets d'interaction.
- Pas d'hypothèse sur la distribution des variables.

Inconvénients

- Performances prédictives limitées.
- Méthode instable, sensible aux perturbations de l'échantillon.

3. Bagging

Cadre :

On cherche toujours à prédire une variable Y (*quantitative ou qualitative*) en fonction de p variables prédictives $X = (X_1, \ldots, X_p)$.

Dans la suite, on se place dans le cas de la régression (Y *quantitatif*) pour simplifier, mais tout s'adapte immédiatement au cas de la classification (Y *qualitatif*).

Agrégation :

- Si on dispose de $T_1(\cdots), \ldots, T_B(\cdot)$ algorithmes de prédictions identiquement distribués, on peut considérer l'algorithme d'agrégation

$$T_{agg}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x).$$

- **Idée :** l'agrégation permet d'améliorer la qualité des prédictions.

Problème :

On ne dispose que d'un échantillon de n observations $(y_1, x_1), \ldots, (y_n, x_n)$ pour entraîner nos B algorithmes.

- Ajuster le même algorithme B fois sur les même données n'est d'aucun intérêt...
- Ajuster le même algorithme sur B sous-échantillons disjoints est d'un intérêt limité...

Bootstrap :

- On va construire B algorithmes sur des **échantillons bootstrap**, et les agréger.
- C'est le principe du **Bagging** (*Bootstrap Aggregating*), qui désigne un ensemble de méthodes introduit par Léo Breiman (*Breiman L. (1996)*).

Bootstrap

Considérons un échantillon $(y_1, x_1), \ldots, (y_n, x_n)$.

Un **échantillon bootstrap** consiste en un tirage de n observations avec remise de cet échantillon.

Exemple

- Echantillon initial :

1	2	3	4	5	6	7	8	9	10

- Echantillons bootstrap :

4	3	8	6	7	8	1	3	5	1	T_1
2	1	3	8	6	8	10	2	2	5	T_2
5	9	4	7	7	1	6	1	8	6	T_3
$\vdots$									$\vdots$	$\vdots$
1	5	3	5	5	4	8	8	2	4	T_B

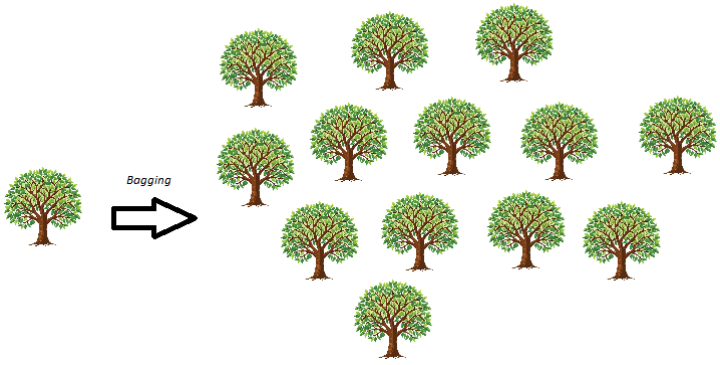
A la fin, on agrège : $T_{Bagg}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x)$.

Algorithme Bagging

- **Entrées :**
  - Un échantillon $(y_1, x_1), \ldots, (y_n, x_n)$.
  - B un entier positif.
  - Un algorithme de prévision T.
- **Pour b allant de 1 à B :**
  - Faire un tirage aléatoire avec remise de taille n dans $\{1, \ldots, n\}$.
    On note θ_b l'ensemble des indices tirés et $D_b = \{(y_i, x_i) : i \in \theta_b\}$ l'échantillon bootstrap associé.
  - Entraîner l'algorithme T sur $D_b \Longrightarrow T_b(\cdot) = T(\cdot, D_b)$.
- **Retourner :**
  - $T_{Bagg}(\cdot) = \dfrac{1}{B} \displaystyle\sum_{b=1}^{B} T_b(\cdot)$ dans le cas de la régression.
  - $T_{Bagg}(\cdot) = \underset{k=1,\ldots,K}{argmax} \displaystyle\sum_{b=1}^{B} \mathbb{1}_{T_b(\cdot)=k}$ dans le cas de la classification (K *modalités*).

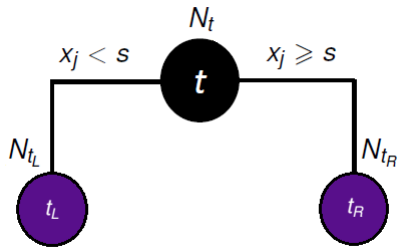
4. Forêts aléatoires

⚙ Une forêt aléatoire, ou **Random Forest**, consiste en l'agrégation d'une collection d'arbres construits sur des échantillons bootstrap (*Bagging*).



⚙ Les forêts aléatoires les plus utilisées sont celles proposées par Léo Breiman au début des années 2000.

Afin de diminuer la corrélation entre les arbres que l'on agrège, Breiman propose d'effectuer chaque coupure dans un arbre de façon « aléatoire ».



On sélectionne la meilleure variable selon laquelle couper (au sens de la diminution de l'impureté), parmi un ensemble composé uniquement de ***mtry*** variables, choisies de façon aléatoire parmi les p variables initiales.

Algorithme Random Forest

- **Entrées :**
  - Un échantillon $(y_1, x_1), \ldots, (y_n, x_n)$.
  - B un entier positif.
  - *mtry* un entier entre 1 et p.
  - *min.node.size* un entier entre 1 et n.
- **Pour b allant de 1 à B :**
  - Faire un tirage aléatoire avec remise de taille n dans $\{1, \ldots, n\}$. On note θ_b l'ensemble des indices tirés et $D_b = \{(y_i, x_i) : i \in \theta_b\}$ l'échantillon bootstrap associé.
  - Construire un arbre CART en découpant chaque noeud de la façon suivante :
    - Choisir *mtry* variables parmi les p variables explicatives.
    - Sélectionner la meilleur coupure $x_j < s$ en ne considérant que les *mtry* variables sélectionnées.
    - Ne pas découper un noeud s'il contient moins de *min.node.size* observations.
  - On note $T_b(\cdot)$ l'arbre obtenu.
- **Retourner :**
  - Régression : $F(\cdot) = \frac{1}{B} \sum_{b=1}^{B} T_b(\cdot)$.
  - Classification (K *modalités*) : $F(\cdot) = \underset{k=1,\ldots,K}{argmax} \sum_{b=1}^{B} \mathbb{1}_{T_b(\cdot)=k}$.

Choix des paramètres

- Plus *min.node.size* est petit, plus le biais diminue.

  Par défaut, *min.node.size* $= 5$ dans le problème de régression, et *min.node.size* $= 1$ dans le problème de classification.

- *mtry* est lié à la corrélation des arbres, et a une influence sur le compromis biais/variance de la forêt.

  Par défaut, *mtry* $= \lfloor \frac{p}{3} \rfloor$ en régression, et *mtry* $= \lfloor \sqrt{p} \rfloor$ en classification.

Remarque

En `R`, on pourra faire appel aux fonctions `randomForest()` ou `ranger()` (*plus rapide, codée en C++*) des packages des mêmes noms.

Exemple (*Huile d'olive*)

On reprend l'exemple de classification des huiles d'olive.

On cherche à déterminer la région de provenance d'une huile en fonction de ses concentrations en acides gras.

Region	Palmitic	Palmitoleic	Stearic	Oleic	Linoleic	Linolenic	Arachidic	Eicosenoic
Ombrie	1095	55	200	7940	600	35	45	3
Pouilles_Sud	1413	202	205	6920	1165	36	46	13
Ligurie_Ouest	960	80	240	7950	740	10	20	2
Sicile	1194	135	263	7277	889	44	95	41
Sardaigne_Interieur	1060	111	231	7363	1149	20	65	1
Pouilles_Sud	1434	185	189	6771	1269	30	62	25

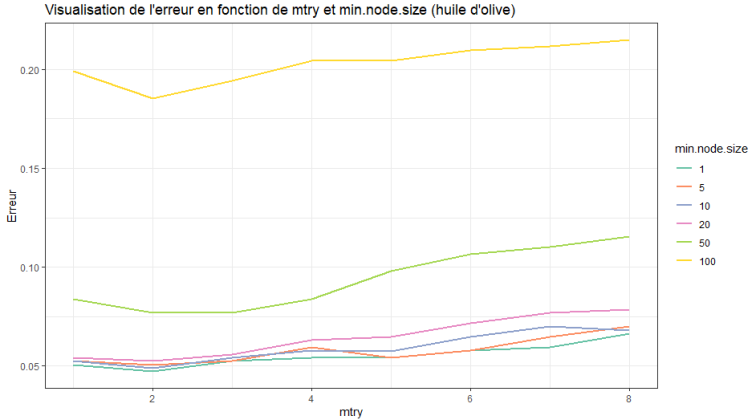
Exemple (*Huile d'olive*)

On utilise une forêt aléatoire :

```
library(ranger)
oil_forest <- ranger(data = df,formula = Region~.)
oil_forest
```

```
Ranger result
Call:
 ranger(data = df, formula = Region ~ .)
Type:                             Classification
Number of trees:                  500
Sample size:                      572
Number of independent variables:  8
Mtry:                             2
Target node size:                 1
Variable importance mode:         none
Splitrule:                        gini
OOB prediction error:             4.90 %
```

On peut calibrer *min.node.size* et *mtry* via les méthodes usuelles (*validation croisée*)...



Mais les valeurs par défaut sont souvent performantes !

⚙ La performance d'une forêt aléatoire peut être mesurée en estimant son risque d'erreur, et peut se faire par les méthodes classiques de ré-échantillonnage (*validation croisée*).

💡 Les tirages bootstrap permettent une alternative, beaucoup moins coûteuse.

💡 On utilise les observations non sélectionnées dans les échantillons bootstraps (*pour la construction de chaque arbre*) afin d'estimer le risque.

Exemple

- Echantillon initial :

1	2	3	4	5	6	7	8	9	10

- Echantillons bootstrap :

4	3	8	6	7	8	1	3	5	1	T_1
2	6	3	8	6	8	10	2	2	5	T_2
5	9	4	7	10	2	6	5	8	6	T_3
6	5	4	3	1	2	10	5	1	4	T_4
4	3	8	8	7	10	2	2	10	5	T_5

- Les échantillons 2,3 et 5 ne contiennent pas l'observation 1, on estime donc

$$\widehat{y}_1 = \frac{1}{3}(T_2(x_1) + T_3(x_2) + T_5(x_3)).$$

- On fait de même pour toutes les observations, et on obtient $\widehat{y}_2, \ldots, \widehat{y}_n$.
- On calcule ensuite l'erreur :

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2 \quad \text{ou} \quad \frac{1}{n}\sum_{i=1}^{n}\mathbb{1}_{y_i \neq \widehat{y}_i}.$$

Erreur OOB (*Out Of Bag*)

- Pour chaque observation $i \in \{1, \dots, n\}$, on note $OOB(i) = \{b \in \{1, \dots, B\} : i \notin \theta_b\}$ l'ensemble des tirages bootstraps ne contenant pas i.
- La prévision de la forêt pour l'observation i, et pour les arbres ne contenant pas cette observation est

$$F_{OOB(i)} = \frac{1}{|OOB(i)|} \sum_{b \in OOB(i)} T_b(x_i).$$

- L'**erreur Out Of Bag** est alors donnée par

$$\frac{1}{n} \sum_{i=1}^{n} (y_i - F_{OOB(i)})^2 \quad \text{ou} \quad \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{y_i \neq F_{OOB(i)}}.$$

Remarque

- Si une observation appartient à **tous** les échantillons bootstraps, l'erreur OOB est calculée sans tenir compte de cette observation.
- La probabilité qu'une observation appartienne à un échantillon bootstrap tend cependant vers $1 - \frac{1}{e} \approx 0.642$ lorsque $n \to +\infty$.
- Si le nombre d'observations est assez grand, et le nombre d'arbres également, il est assez peu probable que ce cas de figure arrive...

Exemple (*Huile d'olive*)

Dans notre forêt aléatoire pour la classification des huiles d'olive, on avait une erreur OOB de 4,90%.

```
library(ranger)
oil_forest <- ranger(data = df,formula = Region~.)
oil_forest
```

```
Ranger result
Call:
 ranger(data = df, formula = Region ~ .)
Type:                             Classification
Number of trees:                  500
Sample size:                      572
Number of independent variables:  8
Mtry:                             2
Target node size:                 1
Variable importance mode:         none
Splitrule:                        gini
OOB prediction error:             4.90 %
```

⚙ L'erreur OOB permet par ailleurs de mesurer l'importance des variables dans le problème de régression ou de classification.

Importance par permutation

- L'idée est, pour chaque arbre de la forêt, de comparer les erreurs :
  - Erreur OOB de l'arbre ;
  - Erreur OOB de l'arbre en permutant les valeurs de la variable $j \in \{1, \dots, p\}$.
- Si la variable j est importante, ces erreurs doivent être très différentes.

⚙ On note pour chaque arbre $b \in \{1, \ldots, B\}$ son erreur sur les données OOB :

$$Err(OOB_b) = \frac{1}{|OOB_b|} \sum_{i \in OOB_b} (y_i - T_b(x_i))^2 \quad (\textit{Régression})$$

ou

$$Err(OOB_b) = \frac{1}{|OOB_b|} \sum_{i \in OOB_b} \mathbb{1}_{y_i \neq T_b(x_i)} \quad (\textit{Classification}),$$

avec **l'échantillon OOB** : $OOB_b = \{i \in \{1, \ldots, n\} : i \notin \theta_b\}$.

⚙ On recalcule ensuite cette erreur, mais en ayant permuté les valeurs de la variable j dans l'échantillon OOB.

Exemple

$$\begin{bmatrix} x_{1,1} & \dots & x_{1,j} & \dots & x_{1,p} \\ x_{2,1} & \dots & x_{2,j} & \dots & x_{2,p} \\ x_{3,1} & \dots & x_{3,j} & \dots & x_{3,p} \\ x_{4,1} & \dots & x_{4,j} & \dots & x_{4,p} \\ x_{5,1} & \dots & x_{5,j} & \dots & x_{5,p} \end{bmatrix} \Longrightarrow \begin{bmatrix} x_{1,1} & \dots & x_{3,j} & \dots & x_{1,p} \\ x_{2,1} & \dots & x_{5,j} & \dots & x_{2,p} \\ x_{3,1} & \dots & x_{1,j} & \dots & x_{3,p} \\ x_{4,1} & \dots & x_{2,j} & \dots & x_{4,p} \\ x_{5,1} & \dots & x_{4,j} & \dots & x_{5,p} \end{bmatrix}$$

On note $\widetilde{x_i}^j$ les individus de l'échantillon OOB permuté, et on note

$$Err(OOB_b^j) = \frac{1}{|OOB_b|} \sum_{i \in OOB_b} \left(y_i - T_b(\widetilde{x_i}^j) \right)^2 \quad (\textit{Régression})$$

ou

$$Err(OOB_b^j) = \frac{1}{|OOB_b|} \sum_{i \in OOB_b} \mathbb{1}_{y_i \neq T_b(\widetilde{x_i}^j)} \quad (\textit{Classification}).$$

Importance par permutation

L'**importance par permutation** de la variable j est donnée par

$$\mathcal{I}^j_{perm} = \frac{1}{B}\sum_{b=1}^{B}\left(Err(OOB^j_b) - Err(OOB_b)\right).$$

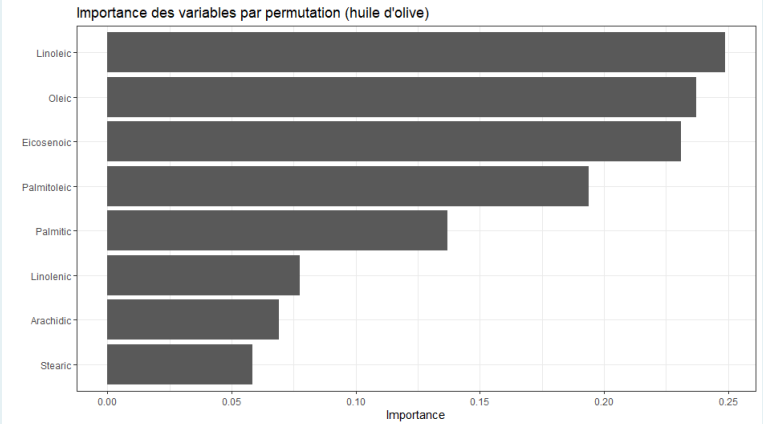
Exemple (*Huile d'olive*)

On peut visualiser facilement l'importance des variables avec le package `vip` et la fonction du même nom.

Dans l'exemple de classification des huiles d'olive, on a :

```
library(ranger)
library(vip)
oil_forest <- ranger(data = df,formula = Region~.,
                     importance = "permutation")
vip(oil_forest)+
  ggtitle("Importance des variables par permutation (huile d'olive)")+
  theme_bw()
```

Exemple (*Huile d'olive*)



Résumé Random Forest

Avantages

- Bonnes performances prédictives.
- Facile à calibrer.
- Mesure d'importance des variables $\Longrightarrow$ Possibilité de sélectionner les variables explicatives d'un modèle (*package* `vsurf`*, Genuer et al. (2010 et 2015)*).

Inconvénients

- Côté boîte noire...
- Mais pas plus que d'autres méthodes.