# Bagging et forêts aléatoires

Laurène DAVID

Mai 2021

## Contents

# 1 Introduction

Le bagging est une méthode d'apprentissage statistique qui permet d'améliorer la performance d'un estimateur faible, soit en anglais d'un *week learner*.

Il se base sur une principe essentiel, celui de la *Sagesse de la Foule* qui explique que pour apporter une solution à un problème, il est toujours plus intéressant de rassembler les opinions d'un ensemble d'individus que de n'utiliser l'opinion que d'une personne.

Appliqué à l'inférence statistique, cette théorie stipule qu'une collection de *week learners* est toujours plus performant qu'un des ses estimateurs étudié seul.

Pour implementer ce principe, le bagging utilise un procédé de bootstrap, combiné à une méthode d'aggrégation de modèles, pour rassembler des estimateurs instables en un estimateur performant.

Nous pouvons noter dés à présent que le bagging est réellement efficace que sur des estimateurs à forte variance mais à faible biais.

Les algorithmes de forêts alétoires proposent d'améliorer les résultats du bagging, pour les arbres de décision CART qui sont des procédés à forte variance.

Pour bien comprendre le fonctionnement du bagging et des forêts alétoires, nous allons commencer par présenter les méthodes de ré-échantillonage bootstrap, puis nous allons nous intéresser au bagging lui-même.

# 2 Le Bootstrap

Le bootstrap est une méthode d'estimation générale de la performance d'un estimateur.

Le but du bootstrap est d'approcher la distribution d'un échantillon afin de prédire une caractéristique de la distribution de l'estimateur. Cette caractéristique peut être, par exemple, le biais, la variance, l'erreur quadratique moyen, ou les quartiles de l'estimateur.

Il se base sur les simulations de Monte Carlo, sur le principe du *plug-in* et sur un procédé de ré-échantillonage.

Nécessistant que de peu d'hypothèses sur l'échantillon de départ, le bootstrap peut ainsi d'adapter à un grand nombre de modélisation, ce qui en fait sa force. C'est une méthode utile lorsqu'il n'existe pas de formules simples pour décrire un estimateur où lorsque sa formule s'avère trop complexe pour calculer directement l'une de ses caractéristiques.

Nous allons commencer par présenter le principe du plug-in et les estimateurs qu'il permet de construire. Ensuite, nous étudierons le principe du bootstrap et la perfomance de ses estimateurs. Enfin, nous utiliserons une simulation d'un jeu de données sur R pour présenter un exemple d'une estimation par bootstrap.

## 2.1 La méthode du *plug-in*

Soit $x = \{x_1, ....., x_n\}$ un échantillon i.i.d d'observations réelles où chaque $x_i$ correspond à la réalisation d'une variable aléatoire $X_i$ de loi $F$

Soit $\theta = t(F)$ le paramètre que l'on cherche à estimer.

La méthode de *plug-in* consiste à remplacer $F$, dans la formule du paramètre, par une approximiation empirique de cette distribution.

Le *plug-in* est la méthode d'estimation privilégiée pour des estimateurs dont on ne dipose d'aucune informations, à part sur le caractère i.i.d des observations.

Dans le cas où la distribution de l'échantillon est supposé paramètrique, il est préférable de chercher à estimer son paramètre afin de construire une approximation paramètrique de $F$ pour ensuite estimer $\theta$.

Nous allons, dans un premier temps, présenter la fonction de distribution empirique ainsi que ses propriétés. Puis, nous expliciterons son rôle dans la méthode d'estimation par du*plug-in*.

### 2.1.1 La fonction de répartition empirique $\hat{F}_n$

La fonction de répartition empirique est la fonction de répartition d'une loi discrère qui est définie par :

$$\hat{F}_n(x_i) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{]-\infty,x]}(X_i) = \frac{\#\{i : X \leq x_i\}}{n}$$

Prenons une variable aléatoire $X$ de loi $\hat{F}_n$. La fonction de distribution empirique associe à X, la probabilité suivante :

$$P(X = x_i) = \frac{\#\{k : x_k = x_i\}}{n}$$

Si on suppose que toutes les observations de notre échantillon sont deux à deux distincts, alors on obtient :

$$P(X = x_i) = \frac{1}{n}$$

Dans ce cas là, $\hat{F}_n$ est une loi uniforme discrète à valeurs dans $\{x_1, ..., x_n\}$ et qui attribue à chaque observation $x_i$ le poinds $\frac{1}{n}$.

On remarque que $\hat{F}_n$ associe à chaque observation $x_i$ de l'échantillon, sa fréquence d'apparition dans l'échantillon. $\hat{F}_n$ est donc un approximation empirique de la distribution $F$ de l'échantillon, qui associe à chaque observation $x_i$ sa probabilité empirique.

Nous verrons dans la partie de ce chapitre dédiée au principe du bootstrap que l'on peut simuler un échantillon de cette distribution par à des tirages avec remise.

Maintenant interessons-nous aux propriétés de cette estimateur :

Soit $x \in \mathbb{R}$ avec $X_1, ..., X_n$ une suite de variables aléatoires i.i.d de distribution $F(x)$ et $\hat{F}_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{]-\infty,x]}(X_i)$ un estimateur de $F(x)$.

**Propriétés de la fonction de répartition empirique $\hat{F}_n$**

- $E[\hat{F}_n(x)] = \frac{1}{n}\sum_{i=1}^{n} E[\mathbb{1}_{]-\infty,x]}(X_i)] = \frac{1}{n}\sum_{i=1}^{n} P(X_i \leq x) = F(x)$

  $\hat{F}_n$ est donc un estimateur sans biais de $F$

- $\hat{F}_n(x) \underset{n\to+\infty}{\overset{p.s}{\longrightarrow}} F(x)$

  Donc, $\hat{F}_n(x)$ est un estimateur consistant de $F(x)$

- $\sqrt{n}(\hat{F}_n(x)_{F(x)}) \overset{Loi}{\longrightarrow} \mathbb{N}(0, F(x)(1-F(x)))$

  d'après le Théorème de la limite central où $Var(\mathbb{1}_{X_i\leq x}) = F(x)(1-F(x))$

- D'après le Théorème de Glivenko-Cantelli, on a : $\|\hat{F}_n - F\|_\infty \overset{p.s}{\longrightarrow} 0$

  $\hat{F}_n$ est donc un estimateur fortement consistant de $F$

Pour présenter les résulats asymptotiques de $\hat{F}_n$, nous allons simuler sur R la fonction de distribution empirique d'un échantillon de loi normale centrée réduire.

Commençons par construire un échantillon de 10 variables aléatoires de loi normale centrée réduite

### 2.1.2 Un estimateur "plug-in"

Un estimateur *plug-in* est un estimateur qui utilise la fonction de répartition empirique pour estimer un paramètre inconnu.

Il est construit de façon à remplacer la distribution de l'échantillon $F$ par la fonction de distribution empiriques $\hat{F}_n$ et est défini par : $\hat{\theta} = t(\hat{F}_n)$.

Il dispose de bonnes propriétés asymptotiques que nous allons à présent présenter.

**Convergence de l'estimateur *plug-in* :**

- L'estimateur plug-in converge, grâce à des hypothèses assez générales sur $t$, presque sûrement vers le paramètre estimé.

  On a : $t(\hat{F}_n) \underset{n\to+\infty}{\overset{p.s}{\longrightarrow}} t(F)$

- Puisque la convergence presque sûre implique la convergence en probabilité, on a : $t(\hat{F}_n) \underset{n\to+\infty}{\overset{P}{\longrightarrow}} t(F)$

  L'estimateur plug-in est donc un estimateur consistant.

En général, l'estimateur **plug-in** n'est pas un estimateur sans biais du paramètre. Il est tout de même interessant car il est construit en fonction des observations de l'échantillon et non par des résultats dont les hypothèses ne peuvent pas toujours être vérifiées.

De plus, puisqu'il est obtenu par une approximation empirique, il est facile à générer informatiquement.

Pour récapituler, la méthode de plug-in permet de construire des estimateurs empiriques consistants grâce à une approximation empirique de la distribution de l'échantillon.

Nous verrons dans ce prochain chapitre comment construire un estimateur *plug-in* par ré-échantillonage de l'échantillon de départ.

## 2.2 Le principe du bootstrap

Le bootstrap est une méthode algorithmique qui permet d'estimer une caractéristique de la distribution d'un estimateur $q(F)$, selon les principes des simulations de Monte Carlo et de la méthode du *plug-in*.

Dans les simulations de Monte Carlo, des nouveaux échantillons sont simulés suivant la distribution de l'échantillon afin de construire des répliques de l'estimateur et à partir de ses répliques, déterminer une caractéristique de sa distribution.

Dans les méthodes de bootstraps, on suppose que la distribution de l'échantillon n'est pas connue. On simule alors des nouveaux échantillons selon la fonction de distribution empirique, une approxmiation empirique de la vrai distribution.

Nous verrons comment les méthodes de bootstrap permettent de construire des estimateurs *plug-in* $\hat{q}(\hat{\theta})$ grâce à un processus entièrement informatisé.

Soit $z = \{z_1, ..., z_n\}$ un échantillon d'observations qui correspond à la réalisation de variables aléatoires i.i.d et $\hat{\theta}$ un estimateur estimé au préalable.

Nous allons détailler le procédé d'une estimation par bootstrap:

1. **Générer $B$ échantillons *bootstraps* par ré-échantillonage.**

   Un *échantillon bootstrap* est un échantillon obtenu par une simulation de $x$ en fonction de $\hat{F}_n$. Dans la pratique, pour générer $B$ échantillons *bootstraps*, on utilise le réechantillonage.

- On tire aléatoirement et avec remise une observation de l'échantillon de départ.
- On répète ce tirage $n$ fois jusqu'à obtenir un nouvel échantillon $x^* = \{z_1^*, ..., z_n^*\}$, de taille n
- On recommence plusieurs fois les deux premières étapes jusqu'à obtenir $B$ échantillons *bootstraps* :

$$\mathbf{z}^{*1} = \{z_1^{*1}, ..., z_n^{*1}\}$$

$$\mathbf{z}^{*2} = \{z_1^{*2}, ..., z_n^{*2}\}$$

$$.$$

$$\mathbf{z}^{*B} = \{z_1^{*B}, ..., z_n^{*B}\}$$

Sachant que $\hat{F}_n$ correspond à une loi uniforme discrète, les échantillons bootstraps sont donc construits par tirage avec remise, pour simuler un échantillon de distribution $\hat{F}_n$.

2. **Evaluer $B$ répliques bootstraps de l'estimateur $\hat{\theta}$**

- Pour chaque échantillon bootstrap $x^{*b}$, on calcule la réplication bootstrap associée $\hat{\theta}^{*b} = \hat{\theta}(\mathbf{z}^{*b})$
- On obtient ainsi $B$ répliques bootstraps de l'estimateur $\hat{\theta}$ : $\hat{\theta}^{*1}, ..., \hat{\theta}^{*B}$
- Ces $B$ répliques bootstraps sont, par construction avec tirages aléatoires et avec remise des échantillons bootstraps, i.i.d

  Nous verrons, dans le prochain chapitre, en quoi ce résultat est important pour les méthodes de bagging

3. **Calculer un estimateur de la caractéristique $q(F)$ de la distribution de l'estimateur, à partir des répliques bootstraps :**

- On construit, à partir des $B$ répliques bootstraps, une caractéristique de $\hat{\theta}$.
- On peut estimer des estimateus *bootstraps* du biais $\beta_B^*$, de la variance $\mathbb{V}_B^*$, ou même du risque quadratique $R_B^*(q, \theta)$ de $\hat{\theta}$.
- Pour estimer le biais de l'estimateur, on cherche à reconstruire l'estimateur *plug-in* du biais en calculant : $\bar{q} = \frac{1}{B}\sum_{b=1}^{B} \hat{\theta}^{*b}$

  Pour enfin obtenir : $\beta_B^* = \bar{q} - \hat{\theta}$

On note que les estimateurs *bootstraps* construits dans l'algorithme ci-dessus sont des approximations, par simulations, des **estimateurs bootstrap ideaux**.

Les estimateurs bootstraps idéaux sont des estimateurs *plug-in* des caractéristiques de la distribution de $\hat{\theta}$. Leur distribution dépend de la fonction de distribution empirique $\hat{F}_n$. Ils sont difficiles à obtenir par des calculs théoriques et sont donc approcher par un processus informatisé.

**Performance de l'estimateur bootstrap :**

Un estimateur *bootstrap* $q_B^*$ est donc une approximation d'un estimateur *plug-in* $q(\hat{F}_n)$, qui est lui même une approximation du vrai paramètre $q(F)$.

Nous pouvons donc nous demander en quoi l'estimateur bootstrap est un bon estimateur de l'estimateur bootstrap idéal et un bon estimateur de la caractéristique $q(F)$.

D'apres le théorème de la limite central, on a que :

$$q_B^* \underset{B\to+\infty}{\xrightarrow{P}} q(\hat{F}_n)$$

Il suffit donc de prendre un $B$ assez grand pour obtenir un bon estimateur de l'estimateur *plug-in*.

Nous avons vu, dans la partie précédente sur la méthode du *plug-in* que $q(\hat{F}_n)$ est un estimateur consistant de $q(F)$, soit :

$$q(\hat{F}_n) \underset{n\to+\infty}{\xrightarrow{P}} q(F)$$

On obtient donc :

$$q_B^* \underset{B\to+\infty}{\xrightarrow{P}} q(\hat{F}_n) \underset{n\to+\infty}{\xrightarrow{P}} q(F)$$

Ainsi, $q_B^*$ est un bon estimateur d'une caractéristique de la distribution de $F$ tant que la taille de l'échantillon $n$ et le nombre d'échantillons *bootstrap* $B$ sont relativement grand.

Dans la pratique, nous n'avons aucun contrôle sur la taille $n$ de l'échantillon mais le nombre d'échantillons *bootstrap* $B$ peut lui être choisi suivant la caractéristique que l'on cherche à estimer.

# 3 Le Bagging

Introduit par le mathématicien américain Leo Breiman, le bagging, ou *Bootstrap Aggregation* est un procédé permettant d'améliorer la performance d'un estimateur.

D'où son nom de *Bootstrap Aggregation*, le bagging combine les méthodes de bootstrap pour générer des répliques de l'estimateur de base à un procédé d'aggrègation de modèles.

Contrairement au boostrap dont le but est d'estimer les erreurs de prédiction d'un estimateur, la méthode de bagging cherche à améliorer la performance de ses prédictions et de réduire ses erreurs de prédictions.

Selon le type d'estimateur traité, l'aggregation s'opère soit en calculant la moyenne empirique de ces estimateurs, dans le cadre d'une régression, soit en opérant à un vote à la majorité, dans le cadre des modèles de classification.

L'intéret du bagging est qu'il permet de réduire la variance de l'estimateur étudié et ainsi de réduire ses erreurs de prédiction. Nous verrons plus tard en quoi l'utilisation du bootstrap est essentiel pour réduire cette variance.

Le bagging peut être utilisé à fin d'améliorer des estimateurs ou des modèles non-linéaires. Il est même plus efficace d'appliquer le bagging sur ce type de modélisation puisque un procédé d'aggrégation d'estimateurs n'améliore pas forcément la prédiction de modèles linéaires

Cette méthode est particulièrement efficace lorsqu'on travaille sur des estimateurs instables, c'est-à-dire sur des modèles à haute variance. Nous verrons dans le chapitre sur les forêts aléatoires que lorsqu'on applique la méthode de bagging aux arbres de décision CART, un modèle de prédiction particulièrement instable, on obtient de bon résultats.

## 3.1 Le fonctionnement du bagging :

Le bagging est une méthode d'aggregation qui fonctionne en partie grâce aux **procédés bootstrap**. Le but étant de construire des estimateurs $\hat{f}_1(\mathbf{x}), ..., \hat{f}_B(\mathbf{x})$ i.i.d pour ensuite les agrégés par estimation de leur moyenne ou par vote à la majorité.

Nous allons détailler le procédé d'estimation par bootstrap :

1. **Générer $B$ échantillons bootstrap**.

- Pour construire des répliques bootstraps i.i.d de l'estimateur, on ré-échantillone par tirage avec remise notre échantillon de départ.
- Le tirage de l'échantillon bootstrap peut s'effectuer de deux façons différentes :

  Soit par un tirage avec remise de $n$ observations sur l'échantillon $x$. On obtient un échantillon bootstrap $\mathbf{z}_b^*$ de taille $n$

  Soit par un tirage avec remise de $m$ observations sur l'échantillon $x$ avec $m < n$. On obtient un échantillon bootstrap $\mathbf{z}_b^*$ de taille $m$

- Dans les deux cas, on répète les tirages jusqu'à obtenir $B$ échantillons *bootstraps* : $\mathbf{z}_1^*, \mathbf{z}_2^*, ..., \mathbf{z}_B^*$

2. **Calculer les $B$ estimateurs bootstrap**

   - On calcule un nouvel estimateur $\hat{f}_{\mathbf{z}_b^*}$ à partir de chaque échantillon *bootstrap* $\mathbf{z}_b^*$,
   - On répète cette procédure $B$ fois jusqu'à obtenir $B$ répliques bootstrap i.i.d : $\hat{f}\mathbf{z}_1^*, \hat{f}\mathbf{z}_2^*, ..., \hat{f}\mathbf{z}_B^*$

3. **Prévision par agrégation de modèles**

   - Si on cherche à prédire une valeur quantitive, on calcule la moyenne des $B$ estimateurs bootstraps : $\hat{f}_{bag}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_{\mathbf{z}_b^*}(\mathbf{x})$
   - Si on cherche à prédire une class, on opère à un vote à la majorité $\hat{f}_{bag}(\mathbf{x}) = \text{argmax}_j \; card \; \{ \text{ b } | \hat{f}_{\mathbf{z}_b} = j\}$

     On associe à $y$ la class qui est la plus souvent prédite par les estimateurs *bootstrap*.

En pratique, obtenir plusieurs répliques i.i.d de l'estimateur étudié, à partir d'un même échantillon, $D$ n'est pas faisable car ils seraient tous dépendants des observations de $D$.

Dans le bagging, l'utilisation de procédés bootstrap permet de contourner ce problème en produisant des répliques i.i.d de l'estimateur à partir d'un même échantillon.

Aggréger des estimateurs i.i.d permet d'optimiser la performance de l'estimateur par bagging.

Nous allons voir, dans cette prochaîne partie, en quoi d'aggréger des estimateurs i.i.d permet d'optimiser la performance du bagging.

## 3.2 La performance du Bagging et l'importance de construire des estimateurs i.i.d

Soit $\hat{f}_1(\mathbf{x}), \hat{f}_2(\mathbf{x}), ..., \hat{f}_B(\mathbf{x})$ des estimateurs identiquements distribués avec $\rho(\mathbf{x})$ le coefficient de corrélation entre deux estimateurs *bootstrap*

Soit $\mathbf{V}_{bag}(\mathbf{x})$ la variance d'un estimateur de bagging $\hat{f}_{bag}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_{\mathbf{z}_b^*}(\mathbf{x})$

$$\mathbf{V}_{bag}(\mathbf{x}) = \mathbf{V}(\frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(\mathbf{x})) = \frac{1}{B^2}\mathbf{V}(\sum_{b=1}^{B} \hat{f}_b(\mathbf{x}))$$

$$\mathbf{V}_{bag}(\mathbf{x}) = \frac{1}{B^2}[\sum_{b=1}^{B}(\mathbf{V}(\hat{f}_b(\mathbf{x})) + \sum_{k\neq k'} \mathbf{Cov}(\hat{f}_k(\mathbf{x}), \hat{f}_{k'}(\mathbf{x})]$$

$$= \frac{1}{B}\mathbf{V}(\hat{f}_b(\mathbf{x})) + \frac{1}{B^2}(B^2 - B)\rho(\mathbf{x})\mathbf{V}(\hat{f}_b(\mathbf{x}))$$

Donc : $\mathbf{V}_{bag}(\mathbf{x}) = \rho(\mathbf{x})\mathbf{V}(\hat{f}_{\mathbf{z}_b^*}(\mathbf{x}) + \frac{1-\rho(\mathbf{x})}{B}\mathbf{V}(\hat{f}_{\mathbf{z}_b^*}(\mathbf{x}))$

On remarque que la variance de l'estimateur agrégé est une fonction croissante de la corrélation $\rho(\mathbf{x})$ des estimateurs que l'on agrège. Plus la dépendance entre les estimateurs que l'on agrège est importante, plus la variance de l'estimateur de bagging est élevé.

Lorsque les estimateurs $\hat{f}_{\mathbf{z}_b^*}(\mathbf{x})$ sont i.i.d, $\rho(\mathbf{x}) = 0$ et donc, pour $B > 1$ :

$$\mathbf{V}_{bag}(\mathbf{x}) = \frac{1}{B}\mathbf{V}(\hat{f}_{\mathbf{z}_b^*}(\mathbf{x})) < \mathbf{V}(\hat{f}_{\mathbf{z}_b^*}(\mathbf{x}))$$

Construire des répliques i.i.d de l'estimateur permet ainsi de minimiser la variance de l'estimateur de bagging et ainsi de maximiser le gain de sa procédure.

Le bagging ne permet pas d'améliorer le biais d'un estimateur.

$$\mathbf{E}[\hat{f}_{bag}(\mathbf{x})] = \mathbf{E}[\frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(\mathbf{x})] = \frac{1}{B}\sum_{b=1}^{B} \mathbf{E}[\hat{f}_b(\mathbf{x})] = \mathbf{E}[\hat{f}_1(\mathbf{x})]$$

Le bagging est donc une procédure performante pour des estimateurs peu biaisés mais à forte variance.

On peut aussi noter que le bagging améliore la performance d'un estimateur non-linéaire, comme ceux des procédés d'arbres CART mais n'a aucun impact sur des estiamteurs linéaires. Le procédé d'aggrégation de modèles n'apporte aucune amélioration à des prédicateurs linéaires.

Malheuresement, dans la pratique, les estimateurs obtenus par bootstrap ne sont pas entièrement indépendants. Ils sont construits sur des échantillons ré-échantillonées à partir du même jeu de données. Il existe donc une certaine dépendance entre les observations des échantillons *bootstrap* qui se traduit par une dépendance entre les répliques *bootstrap* de l'estimateur.

Nous verrons comment l'algorithme de forêts aléatoires, dans le cadre d'arbres de décision CART, permet de réduire la dépendance des échantillons bootstraps et donc d'améliorer la procédure de bagging.

# 4 Les Arbres de décision binaires

Dans ce chapitre, nous allons nous intéresser aux arbres de décision binaires, aussi appelés arbres de décision CART (*Classification and Regression Tree*).

Les arbres de décision CART sont des méthodes de sélection de modèles qui découpent un échantillon initiale en plusieurs régions de prédiction homogènes afin de réduire l'hétérogénéité des données.

Les arbres de décision binaires sont représentés par un ensemble de règles de décision binaires qui découpent récursivement et de façon binaire l'échantillon d'étude initiale, selon une approche déscendante.

On dit que ce sont des procédés couteux car ils prennent en compte la meilleure découpe possible à partir de l'étape du procédé et non la meilleure découpe possible pour l'arbre tout entier.

Les arbres de décision CART se concentrent sur deux types d'arbres, les arbres de régression et les arbres de classification. Nous verrons que les critères de division de l'arbre et sa prédiction finale diffèrent selon le type d'arbre étudié.

Ce sont des processus d'apprentissages faciles à comprendre, à visualiser et à interpréter puisque chaque étape de son processus est représenté par un arbre. D'où l'intérêt d'utiliser ce type de méthode. Ce sont pourtant des procédures hautement instables. Une légère modification de l'échantillon de départ provoque un grand changement sur l'arbre obtenu. Nous verrons dans le prochain chapitre comment réduire l'instabilité de ces modèles par des algorithmes de Forêts Aléatoires.

## 4.1 Construction d'un arbre de décision CART

Construire un arbre de décision CART nécessite d'avoir estimer au préalable :

1. **Les noeuds de l'arbre :**
   - Un noeud de l'arbre $C_i$ correspond à une étape du partitionnement récursif des données.
   - A chaque noeud $C_i$, on associe une région de prédiction $R_i$
   - La région de prédiction $R_i$ correspond à un sous-ensemble de l'échantillon de départ, construit selon un *critère de division*
2. **Les critères de division :**
   - Un critère de division est composé d'une variable explicative $X_i$ et d'un réel $d$

- Le critère de divison permet de diviser la région de prédiction d'un noeud en deux régions plus homogènes en choisissant la meilleure division *admissible* possible, c'est-à-dire la meilleure division qui génère deux régions de prédiction non vides.
- Le choix de la variable $X_i$ et le réel $d$ dépend d'une fonction d'hétérogénéité, qui permet de calculer les erreurs de précision des deux régions $R_c$ et $R_d$ que l'on cherche à créer.
- Le but est de choisir $X_i$ et $d$ de façon à minimiser la fonction d'homogénéité des deux régions
- Le fonction d'hétérogénéité et la forme du critère de division diffèrent selon le type d'arbre étudié

3. **La règle d'arrêt :**
    - La règle d'arrêt dicte la taille minimale accepté pour une région de prédiction.
    - Elle permet à l'arbre de s'arrêter lorsque la dimension des régions de prédictions créer deviennent suffis celle-ci n'est pas vérifié alors les divisions s'arrètent.
    - En théorie, l'arbre devrait s'arrêter lorsque la fonction d'hétérogénéité de l'ensemble des régions de prédiction devient nulle. En pratique, cette situation est, soit impossible, soit produit des arbres beaucoup trop complexes.
    - Cette règle permet ainsi d'éviter une sur-compléxification de l'arbre, ce qui le rendrait difficile à interpréter.
4. **Les régions de prédiction final ou "feuille" de l'arbre:**
    - Les noeuds qui se situent au bout d'une branche de l'arbre sont des noeuds finaux.
    - On associe à chaque région de prédiction final, une prédiction (qualitative ou quantitative en fonction du type d'arbre) construite en fonction des observations de la région.
    - Toutes les observations qui attérissent dans une région de prédiction va être prédite par la même valeur.

Une règle d'arrêt supplémentaire peut être à ajouter à un arbres de décision binaire. Cette règle consiste à ne garder que les divisons qui produisent une décroissance suffisante de l'hétérogénéité de l'ensemble.

Cette règle permet, en théorie, d'empecher à un arbre de trop se complexifier. Son gros problème est qu'elle peut forcer un arbre à s'arrêter trop tôt et donc de peut-être rater une division pertinente qui réduierait significativement l'hétérogénéité des donnés.

Nous verrons plus tard qu'il existe d'autres moyens de réduire la complexité d'un arbre, comme par exemple par élagage de ses noeuds.

Nous allons à présent détailler le processus de création d'arbres de *régression* et de *classification*.

### 4.1.1 Les Arbres de Régression

Soit $R_0 = \{(y_1, \mathbf{x}_1), ..., (y_n, \mathbf{x}_p)\}$ un échantillon d'observations réelles.

On suppose que $R_0$ soit une réalisation de $n$ couples $(Y, \mathbf{X})$ avec $\mathbf{X} \in \mathbb{R}^p$ et $Y$ une variable aléatoire quantitative.

Dans un arbre de régression, le *critère de division* correspond à deux inégalités qui divisent l'échantillon du noeud en deux régions $R_1(j,d) = \{X|X_j < d\}$ et $R_2(j,d) = \{X|X_j \geq d\}$ où $X_j$ une variable explicative et $d \in \mathbb{R}$ une *division*.

Dans le cas général, les variables $X_j$ et $d$ sont choisit pour générer la plus forte décroissance de l'hétérogénéité des observations $y_j$ l'échantillon. Elles sont donc choisit pour minimiser la fonction d'hétérogénéité des deux sous-échantillons définis par $R_1(j,d)$ et $R_2(j,d)$.

Pour un arbre de régression, la fonction d'hétérogénéité à minimiser est définie par :

$$\sum_{i:\ x_i \in R_1(j,d)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\ x_i \in R_2(j,d)} (y_i - \hat{y}_{R_2})^2$$

où $\hat{y}_{R_1}$ et $\hat{y}_{R_2}$ correspond à la moyenne des observations $y$ calculé sur les régions $R_1(j,d)$ et $R_2(j,d)$ e

Pour construire un critère de division, l'algorithme d'arbre de décision va calculer toutes les fonctions d'hétérogénéités de chaque combinaison possibles de $X_j$ et $d$ puis sortira les deux $X_j$ et $d$ qui auront au mieux minimiser cette fonction. Notons que les combinaisons retenus par l'algorithme sont des combinaisons qui génèrent une division *admissible*.

Une fois la variable $X_i$ et la division $d$ choisis, la région de prédiction est divisée en deux nouvelles régions de prédictions $R_1(j,d) = \{X|X_j < d\}$ et $R_2(j,d) = \{X|X_j \geq d\}$ maintenant associées à deux nouveaux noeuds ainsi créés.

On continue alors de diviser en deux sous-échantillons les régions $R_1(j,d) = \{X|X_j < d\}$ et $R_2(j,d) = \{X|X_j \geq d\}$ en fonction de nouveaux critères de division propres à chaque régions $R_1$ et $R_2$ Le partitionnemenet binaire des

deux régions $R_1$ et $R_2$ produit quatre nouvelles régions de précidiction et quatre nouveaux *noeuds* de l'arbre.

La création de nouvelles régions de prédiction et de nouveaux noeuds s'enchaînent jusqu'à ce que la règle de décision de l'arbre ne soit plus vérifié. En général, cette règle est fixé à 5 observations.

Les noeuds qui se trouvent ainsi en bout d'arbres deviennent des "feuilles" ou des noeuds terminaux et les régions de prédictions associés à ces noeuds deviennent des régions de prédictions finales. Pour chacune de ses régions finales, l'arbre va prédire une valeur quantitaive, calculée en fonction des observations présentes dans cette région.

Pour les arbres de régression, la prédiction associé à une région finale $R_J^*$ correspond à la moyenne des observations $y_j$ de cette région. Cela veut dire que l'arbre va prédire à toute observation qui "attérit" dans cette région, la moyenne des observations $y_j$ de la région.

$$\hat{y} = \frac{1}{|R_J^*|} \sum_{j:x_j \in R_J^*} y_j$$

### 4.1.2 Les Arbres de Classification :

Soit $R_0 = \{(y_1, \mathbf{x}_1), ..., (y_n, \mathbf{x}_p)\}$ un échantillon d'observations réelles.

On suppose que $R_0$ soit une réalisation de $n$ couples $(Y, \mathbf{X})$ avec $\mathbf{X} \in \mathbb{R}^p$ et $Y$ une variable aléatoire qualitative à $q$ modalités $\{1, ..., q\}$.

La forme d'un critère de division reste la même pour les deux types de régression, donc on peut définir deux régions de prédictions par : $R_1(j, d) = \{X | X_j < d\}$ et $R_2(j, d) = \{X | X_j \geq d\}$

On note tout de même que si les variables explicatives sont elles-aussi qualitative, on obtient des régions de prédictions légèremment différents : $R_1(j, d) = \{X | X_j = d\}$ et $R_2(j, d) = \{X | X_j \neq d\}$ où $d$ n'est plus un réel quelconque mais l'une des class possibles de $X_j$.

Puisque les observations $y$ sont qualitatives, la fonction d'hétérogénéité de l'arbre de régression ne peut plus convenir, On doit donc utiliser de nouvelles fonctions d'hétérogénité que l'on va détailler ci-dessous :

Il existe plusieurs fonctions qui permette de calculer l'hétérogénéité d'une région de prédiction pour des variables $y_i$ qualitative.

La première est l'indice de Gini d'une région $R_m$ qui est définie par :

$$G = \sum_{k=1}^{K} \hat{p}_m(k)(1 - \hat{p}_m(k))$$

où $p_m(k) = \frac{1}{|R_m|} \sum_{x_i \in R_m} \mathbb{1}_{\{y_i = k\}}$ correspond à la proportion, dans la région de prédiction $R_m$, d'observations $y_i$ de la class $k$.

Si $p_m(k)$ est proche de 1 alors on peut en déduire que la class $k$ est majoritaire dans la région de prédiction $R_m$.

L'indice de Gini de la région $R_m$ est faible lorsque la valeurs de $p_m(k)$, pour chaque class $k$, se rapproche de 0 ou de 1. On peut donc dire qu'une région de prédiction est homogène si son indice de Gini est faible, et elevé si son indice de Gini est important. L'indice de Gini permet ainsi de déterminer l'hétérogénéité d'une région.

Pour obtenir la variable explicative $X_i$ et la division $d$ qui réduit au mieux l'hétérogénéité des deux régions $R_1(j, d)$ et $R_1(j, d)$, on doit minimiser :

$$\sum_{k=1}^{K} \hat{p}_{R_1}(k)(1 - \hat{p}_{R_1}(k)) + \sum_{k=1}^{K} \hat{p}_{R_2}(1 - \hat{p}_{R_2}(k)$$

La deuxième est la fonction d'entroprie-croisée, qui est définie par :

$$-\sum_{k=1}^{K} \hat{p}_m(k) log(\hat{p}_m(k))$$

On peut montrer, comme dans le cas de l'indice de Gini, que cette fonction sort un résultat proche de zéro lorsque $\hat{p}_m(k)$ est proche de 0 ou de 1.

La troisième est une fonction qui calcule l'erreur de classification de la région $R_m$.

Elle est définie par :

$$E_m(k^*) = 1 - \hat{p}_m(k^*)$$

où $k^* = arg\ max_k(\hat{p}_m(k))$ est la class majoritaire de la région $R_m$

En général, les arbres de classification utilise l'indice de Gini ou l'entropie-croisée car elles sont plus sensibles à la pureté d'une région, c'est-à-dire à son homogénéité.

L'arbre de classification associe, à chaque région de prédiction finale, la class qui apparait le plus souvent dans les observations $y_i$ de cette région. Cela veut dire que pour toute observation qui attérit dans cette "feuille", l'arbre lui prédit la class majoritaire de cette région.

$$\hat{y} = arg\ max_k(\hat{p}_m(k)) = k^*$$

## 4.2 Elagage d'un arbre CART

Le problème des ces arbres CART est qu'ils sont souvent trop complexes et donc sur-ajustent les données. Ces arbres découpent l'échantillon initiale en une importante séquence de noeuds qui s'avèrent trop complexes pour produire une bonne estimation.

Puisque la complexité d'un arbre CART est dictée par le nombre de noeuds qu'ils possèdent, une solution à la sur-complexité de ses arbres peut être de construire, à partir d'un arbre complexe, un sous-arbre optimal. Cela correspond à un procédé d'élagage ou de *pruning* d'arbres de décision CART.

Nous allons nous intéresser à l'élagage d'un arbre de régression. Ce procédé peut aussi être appliqué à des arbres de classfication où l'erreur de l'estimation est déterminé par la proportion d'erreur de classification de chaque noeud.

Le but de ce procédé est de construire un arbre CART de taille maximale $T_0$ que l'on va ensuite segmenté *noeuds* par *noeuds* afin d'obtenir une séquence emboîtée de sous-arbres $T$. Chaque noeud finale d'un sous-arbres est indexé par la variable $m$.

Pour choisir le sous-arbre optimal, on introduit un paramètre de complexité $\alpha$. Il permet d'indiquer l'importance que l'on accorde à la taille d'un arbre, soit au nombre de *noeuds* $|T|$ qu'il possède.

Après avoir estimer le paramètre $\alpha$, on selectionne le sous arbre qui minimise :

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|$$

avec :

$$N_m = Card\{i : x_i \in R_m\}$$

$$Q_m(T) = \frac{1}{N_m} \sum_{x_i in R_m} (y_i - \hat{y}_i)^2$$

On peut considérer $Q_m(T)$ comme un indicateur de la performance des estimations d'une région finale $R_m$

L'estimation du paramètre $\alpha$ peut être effectuée grâce à un procédé de validation croisée à 5 ou 10 blocs, appelé en anglais *Five-Fold Cross-Validation* ou *Ten-Fold Cross-Validation*

## 4.3 L'instabilité des arbres CART

Les estimations des arbres de décision CART sont des estimations peu biaisés puisque le but des arbres de décision CART est de prédire une valeur qualitative ou quantitative en s'adaptant aux observations étudiées.

Cependant, les arbres CART ont une forte variance car ce ce sont des procédés instables, très sensible à un changement des observations étudiés. Une lègère modification des observations étudiées produit un arbre significativement différent.

Cette instabilité est dû à la constuction hierarchique de ses estimations, qui donnent une forte importance aux divisons effectuées en haut de l'arbre. Une erreur à cette étape de l'arbre engendre donc de lourdes conséquences sur les prédictions finales de l'arbre.

Nous verrons, dans ce prochain chapitre, une possible solution à la forte instabilité de ces arbres, apportée par les procédés de *Forêts Alétaoires*.

# 5 Forêts Alétoires

Le bagging est une méthode d'aggregation qui permet d'améliorer la performance d'un estimateur peu biaisé et à forte variance, grâce à des répliques *bootstraps* de l'estimateur.

L'un des problèmes majeurs de cette méthode est que sa performance dépend essentiellement de la dépendance des estimateurs que l'on agrège et que la simple utilisation des réplications *bootstrap* ne permet pas toujours de diminuer cette dépendance.

Les Forêts Aléatoires proposent une amélioration des méthodes de bagging, dans le cadre des arbres de décision CART, en rajoutant une étape au bagging qui permet de réduire la dépendance des estimateurs *bootstrap*.

Les méthodes de bagging sont déjà relativement efficaces sur les arbres de décision CART, car comme nous l'avons dit dans le chapitre précédent, les arbres de décision sont des procédés hautements instables qui produisent des estimations qui varient énormement selon l'échantillon étudié.

Les Forêts Aléatoires permettent d'améliorer les résultats du bagging sur les arbres de décision CART en ajoutant une étape de décorrélations des arbres.

## 5.1 Construction d'un algorithme de forêts aléatoires :

Soit $(Y, \mathbf{X})$ un vecteur de variables alétoires avec $Y$ une variable à expliquer et $\mathbf{X}$ un vecteur de $p$ variables explicatives $X^1, ..., X^p$.

Soit $\mathbf{z} = \{(y_1, x_1), ..., (y_n, x_n)\}$ un échantillon où $(y_i, x_i)$ est une réalisation du vecteur $(Y, \mathbf{X})$.

Soit $A(\mathbf{z})$ un arbre de décision CART construit à partir de l'échantillon $\mathbf{z}$.

Les algorithmes de Forêts Alétoires commencent par construire des échantillons *bootstrap* de l'échantillon $\mathbf{z}$, ce qui nous donne : $\mathbf{z}_1^*, ..., \mathbf{z}_1^B$.

Les échantillons *bootstrap* sont construits par tirage avec remise sur l'échantillon $z$, comme dans les procédés de bagging

Chaque échantillon bootstrap est ensuite adapté à un même arbre de décision, ce qui permet d'obtenir $B$ répliques *bootstrap* d'un même arbre : $A(\mathbf{z}_1^*), ..., A(\mathbf{z}_B^*)$

Dans le cadre du bagging, les arbres sont construits tel sorte à prendre en compte toutes les $p$ variables explicatives $X_1, ..., X_p$ pour le choix d'un critère de décision.

Les procédés de forêts alétoires n'utilisent seulement qu'une partie des variables explicatives, pour construire les critères de décision de ses arbres. Les variables $X_1, ..., X_m$ avec $m \leq p$ considéréés sont tirées aléatoirement.

Cette modification par rapport au bagging permet de réduire la dépendance des arbres créés.

Les arbres de décision sont des méthodes de selection de modèles hierarchisées. Le choix d'une variable explicative à une étape de l'arbre va donc influencer toutes les régions de prédictions qui vont suivre cette étape.

Lorsqu'une variable est particulièrement performante par rapport à d'autres variables c'est-à-dire que son choix produit une bien plus forte décroissance de l'hétérogénéité des observations, un arbre de décision aura tendance à lui donner de l'importance. Cette variable va donc être rapidement choisi par un critère de division et aura un grand impact sur les prédictions faites par l'arbre.

Si on utilise pour chaque arbre *bootstrap* les mêmes variables explicatives, la plupart des arbres vont donc choisir de donner de l'importance à la variable performante et ainsi produire les mêmes estimations.

Pour obtenir des arbres décorrélés, il est donc important de les construire à partir d'une selection de variables explicatives et non pas la totalité.

Une fois avoir construit les répliques *bootstrap* de l'arbre $A(\mathbf{z}_1^*), ..., A(\mathbf{z}_B^*)$, la forêt aléatoire prédit la valeur de $y_i$, en aggrégant l'ensemble des ces arbres.

A la différence du bagging, les forêts alétoires estiment $y_i$ en aggrégeant des arbres crées à partir d'échantillons *bootstrap* $z_i$ ne contenant pas $y_i$. Les forêts aléatoires utilisent donc des échantillons *bootstrap* out-of-bag.

Soit $\hat{f}_{A(\mathbf{z}_b^*)}(x)$ la valeur prédite par un arbre $A(\mathbf{z}_b^*)$ où $z_b^*$ est un échantillon *out-of-bag*.

Si l'arbre étudié est un arbre de régression, on prédit $y_i$ en fonction de la moyenne des prédictions des répliques de l'arbre crée

$$\hat{f}_{RF}^B(y_i) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_{A(\mathbf{z}_b^*)}(y_i)$$

Pour les arbres de classification, la prédiction de la class de $y_i$ s'opère selon un vote à la majorité :

$$\hat{f}_{RF}^B(y_i) = \arg\max_j \text{card}\{b|\ \hat{f}_{A(\mathbf{z}_b^*)}(y_i) = j\}$$

L'estimateur du *Random Forests* associe à $x$ la class qui domine parmi celle prédite par les arbres $A(\mathbf{z}_1^*), ..., A(\mathbf{z}_B^*)$.

Nous pouvons maintenant nous demander comment fixer le nombre $m$ de variables aléatoires tirées pour un critère de division :

En général, pour les arbres de classification, on fixe $m$ à $\sqrt{(}p)$.

Pour les arbres de régression, on fixe $m$ à $\frac{p}{3}$.

On aura tendance à vouloir selectionner un $m$ relativement petit car on cherche à réduire au maximum la dépendance des arbres et donc travailler sur des petits échantillons de variables explicatives permet de réduire cette dépendance.

Cependant, lorsqu'on travaille avec un grand nombre de variables explicatives mais que seulement une petite partie est significative pour le modèle, prendre un petit $m$ n'est pas optimal.

Dans ce cas, à chaque tirage des variables explicatives, la probabilité de tirer une variable non-significative est importante. Puisqu'on tire peu de variables, la chance de tirer une variable significative sont donc assez faibles et on construit alors des arbres peu performants.

Récapitulons les étapes des procédés de forêts aléatoires :

1. Simuler $B$ échantillons bootstrap par tirage avec remise : $\mathbf{z}_1^*, ..., \mathbf{z}_1^B$
2. Construire à partir de ces échantillons *bootstrap*, $B$ arbres de décision en suivant ces étapes :
   - Tirer aléatoirement $m$ variables explicatives parmi les $p$ disponibles
   - Choisir parmi ces varibles, celle qui, associé à un diviseur $d$, minimise la fonction d'hétérogénéité des deux nouvelles régions $R_1(j, s)$ et $R_2(j, s)$
   - Diviser le noeud en deux et recommencer jusqu'à ce que la règle d'arrêt ne soit plus vérifiée.
3. Prédire $y_i$ par aggregation des arbres *bootstrap* qui n'ont pas été construit avec $y_i$
   - Si $y_i$ est quantitative, on calcule : $\hat{f}_{RF}^B(y_i) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_{A(\mathbf{z}_b^*)}(y_i)$
   - Si $y_i$ est qualitative, on calcule :
     $\hat{f}_{RF}^B(y_i) = \arg\max_j \text{card}\{b|\ \hat{f}_{A(\mathbf{z}_b^*)}(y_i) = j\}$

## 5.2 Erreur out-of-bag

Pour avoir une idée de la performance des forêts aléatoires, on peut calculer ses erreurs out-of-bag. L'erreur out-of-bag permet d'estimer les erreurs de prédictions d'une forêt aléatoire.

Pour lesvarbres CART de régression, elle est prédite par :

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

où $\hat{y}_i$ est une prédiction out-of-bag de $y_i$, c'est-à-dire une prédiction de $y_i$ construite sur des répliques *bootstrap* de l'arbre ne travaillant pas sur l'observation $y_i$.

Pour les arbres CART de classification , elle est prédite par :

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}_{\{\hat{y}_i \neq y_i\}}$$

Pour des problèmes de classification, elle moyenne l'erreur de classification de chaque prédiction $\hat{y}_i$

L'avantage de l'erreur out-of-bag est qu'elle se construit directement grâce à l'algorithme de forêts aléatoires.

Elle se démarque ainsi des méthodes de validation-coirsé qui nécessite de construire deux échantillons, un échantillon d'apprentissage et un échantillon test, pour estimer l'erreur de prédiction d'un modèle.

# 6 Conclusion

Le bagging et les forêts alétoires fonctionnement selon le même principe : générér des répliques *bootstrap* i.i.d de l'estimateur étudié, pour ensuite les aggrégés.

Contrairement aux forêts aléatoires, la performance du bagging est limité par sa construction de répliques i.i.d , entièrement dictée par le bootstrap, qui en pratique ne permet pas de construire des répliques entièrement décorrélées.

La performance du bagging est entièrement lié à sa capacité à réduire la corrélation des répliques de l'estimation qu'il construit.

Les forêts alétoires améliorent, pour les arbres de décision CART, les procédés de bagging, en ajoutant une étape aléatoire au choix d'une variable explicative pour un critère de division. Cette étape permet de décorréler les repliques *bootstrap* des arbres.

Nous pouvons tout de même souligner que pour réduire la variance d'un estimateur, le bagging reste une procédure particulièrement efficace. Cependant, le bagging ne permet pas de diminuer le biais d'un estimateur et ne permet donc pas d'améliorer des prédictions peu variantes mais fortement biaisées.

Il existe un procédé d'aggrégation efficace sur des modèles peu variants mais fortement biaisées, ce procédé est nommé Boosting. L'intérêt du Boosting est qu'il construit des *échantillons pondérés* qui s'adapte aux erreurs de prédiction du modèle.

Le choix de l'utilisation du Bagging ou du Boosting pour un modèle dépend donc de la qualité de ses estimations. Si le modèle est fortement variants, il est plus utile d'utiliser le Bagging. Au contraire si le modèle est fortement variants, le Boosting est à privilégier.

# 7 Bibliographie

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani - *An introduction to Statistical Learning* (2013)

Trevor Hastie, Robert Tibshirani, Jerome Friedman - *The Elements of Statistical Learning* (2009)

Robin Genuer, Jean-Michel Poggi - *Arbres CART et Forêts aléatoires, Importance et sélection de variables.* (2017) ffhal-01387654v2f

Introduction au bootstrap (wikistat.fr) : https://www.math.univ-toulouse.fr/ besse/Wikistat/pdf/st-m-app-bootstrap.pdf

Arbres de décision CART (wikistat.fr) : https://www.math.univ-toulouse.fr/ besse/Wikistat/pdf/st-m-app-cart.pdf