# Introduction aux méthodes ensemblistes*

## On va vous booster le gradient

**Mélina Hillion**
Unité SSP-Lab
Insee
melina.hillion@insee.fr

**Olivier Meslin**
Unité SSP-Lab
Insee
olivier.meslin@insee.fr

November 22, 2024

**Abstract**

A compléter

**Keywords**: machine learning • méthodes ensemblistes • formation

*Nous remercions Daffy Duck et Mickey Mouse pour leur contribution.

# Sommaire

# 1. Introduction

Une bien belle introduction pour le site et le DT.

# 2. Aperçu des méthodes ensemblistes

**Principe**: cette partie propose une présentation intuitive des méthodes ensemblistes à destination des lecteurs souhaitant un aperçu du fonctionnement et des cas d'utilisation de ces méthodes. Elle ne contient aucun formalisme mathématique.

## 2.1. Que sont les méthodes ensemblistes?

Les approches ensemblistes désignent un ensemble d'algorithmes de *machine learning* supervisé développés depuis le début des années 1990, c'est-à-dire des méthodes statistiques permettant de prédire une variable-cible $y$ (appelée *target*) à partir d'un ensemble de variables $\mathbf{X}$ (appelées *features*). Elles peuvent par exemple être utilisées pour prédire le salaire d'un salarié, la probabilité de réponse dans une enquête, le niveau de diplôme… Au-delà de leur diversité, ces approches se définissent par un point commun: plutôt que de tenter de construire d'emblée un unique modèle très complexe et très performant, elles visent à obtenir un modèle très performant en combinant intelligemment un ensemble de modèles peu performants, appelés « apprenants faibles » (*weak learner* ou *base learner*). Le choix de ces modèles de base (des arbres de décision dans la plupart des cas) et la manière dont leurs prédictions sont combinées sont des facteurs déterminants pour la performance de ces approches. Le présent document se concentre sur les méthodes ensemblistes à base d'arbres de décisions qui sont parmi les plus utilisées en pratique.

On distingue **deux grandes familles de méthodes ensemblistes** à base d'arbres de décisions, selon qu'elles s'appuient sur des modèles de base entraînés en parallèle indépendamment les uns des autres, ou au contraire entraînés de façon séquentielle. Lorsque les modèles sont *entrainés en parallèle, indépendamment les uns des autres*, on parle de *bagging* ou de forêt aléatoire (*random forest*). Les implémentations les plus courantes des forêts aléatoires sont les *packages* `ranger` en `R` et `scikit-learn` en Python. Lorsque les modèles de base sont *entraînés de manière séquentielle*, chaque modèle de base visant à améliorer la prédiction proposée par l'ensemble des modèles de base précédents, on parle de *boosting*. Ce document aborde essentiellement le *gradient boosting*, qui est l'approche de *boosting* la plus utilisée actuellement. Les implémentations les plus courantes du *gradient boosting* sont actuellement `XGBoost`, `CatBoost` et `LightGBM`.

## 2.2. Pourquoi utiliser des méthodes ensemblistes?

Les méthodes ensemblistes sont particulièrement bien adaptées à de nombreux cas d'usage de la statistique publique, pour deux raisons. D'une, elles sont conçues pour s'appliquer à des *données tabulaires* (enregistrements en lignes, variables en colonnes), structure de données omniprésente dans la statistique publique. D'autre part, elles

peuvent être mobilisées dans toutes les situations où le statisticien mobilise une régression linéaire ou une régression logistisque (imputation, repondération…).

Les méthodes ensemblistes présentent trois avantages par rapport aux méthodes économétriques traditionnelles (régression linéaire et régression logistique):

- Elles ont une **puissance prédictive supérieure**: alors que les méthodes traditionnelles supposent fréquemment l'existence d'une relation linéaire ou log-linéaire entre $y$ et $\mathbf{X}$, les méthodes ensemblistes ne font quasiment aucune hypothèse sur la relation entre $y$ et $\mathbf{X}$, et se contentent d'approximer le mieux possible cette relation à partir des données disponibles. En particulier, les modèles ensemblistes peuvent facilement modéliser des **non-linéarités** de la relation entre $y$ et $\mathbf{X}$ et des **interactions** entre variables explicatives *sans avoir à les spécifier explicitement* au préalable, alors que les méthodes traditionnelles supposent fréquemment l'existence d'une relation linéaire ou log-linéaire entre $y$ et $\mathbf{X}$.
- Elles nécessitent **moins de préparation des données**: elles ne requièrent pas de normalisation des variables explicatives et peuvent s'accommoder des valeurs manquantes (selon des techniques variables selon les algorithmes).
- Elles sont généralement **moins sensibles aux valeurs extrêmes et à l'hétéroscédasticité** des variables explicatives que les approches traditionnelles.

Elles présentent par ailleurs deux inconvénients rapport aux méthodes économétriques traditionnelles. Premièrement, bien qu'il existe désormais de multiples approches permettent d'interpétrer partiellement les modèles ensemblistes, leur interprétabilité reste moindre que celle d'une régression linéaire ou logistique. Deuxièmement, les modèles ensemblistes sont plus complexes que les approches traditionnelles, et leurs hyperparamètres doivent faire l'objet d'une optimisation, par exemple au travers d'une validation croisée. Ce processus d'optimisation est généralement plus complexe et plus long que l'estimation d'une régression linéaire ou logistique. En revanche, les méthodes ensemblistes sont relativement simples à prendre en main, et ne requièrent pas nécessairement une puissance de calcul importante.

**i** Et par rapport au *deep learning*?

Si les approches de *deep learning* sont sans conteste très performantes pour le traitement du langage naturel, des images et du son, leur supériorité n'est pas établie pour les applications reposant sur des données tabulaires. Les comparaisons disponibles dans la littérature concluent en effet que les méthodes ensemblistes à base d'arbres sont soit plus performantes que les approches de *deep learning* (L. Grinsztajn, E. Oyallon, et G. Varoquaux [1], R. Shwartz-Ziv et A. Armon [2]), soit font jeu égal avec elles (D. McElfresh *et al.* [3]). Ces études ont identifié trois avantages des méthodes ensemblistes: elles sont peu sensibles aux variables explicatives non pertinentes, robustes aux valeurs extrêmes des variables explicatives, et capables d'approximer des fonctions très irrégulières. De plus, dans la pratique les méthodes ensemblistes sont souvent plus rapides à entraîner et moins gourmandes en ressources informatiques, et l'optimisation des hyperparamètres s'avère souvent moins complexe (R. Shwartz-Ziv et A. Armon [2]).

## 2.3. Comment fonctionnent les méthodes ensemblistes?

Ce paragraphe présente d'abord le modèle de base sur lesquelles sont construites les méthodes ensemblistes à base d'arbres: l'arbre de classification et de régression (Chapitre 2.3.1). Elle introduit ensuite les deux grandes familles de méthodes ensemblistes couvertes dans ce document: le *bagging* et les forêts aléatoires (Chapitre 2.3.2), puis le *gradient boosting* (Chapitre 2.3.3).

### 2.3.1. Le modèle de base: l'arbre de classification et de régression

#### 2.3.1.1. Qu'est-ce qu'un arbre CART?

Le modèle de base des méthodes ensemblistes est le plus souvent un arbre de classification et de régression (CART, L. Breiman, J. Friedman, R. Olshen, et C. Stone [4]). Un arbre CART est un algorithme prédictif assez simple avec trois caractéristiques essentielles:

- L'arbre partitionne l'espace des variables explicatives $X$ en régions (appelées feuilles ou *leaves*) les plus homogènes possible, au sens d'une certaine mesure de l'hétérogénéité;
- Chaque région est définie par un ensemble de conditions, appelées régles de décision (*splitting rules*), qui portent sur les valeurs des variables explicatives (par exemple, une région peut être définie par la condition: $age > 40$ et $statut = 'Cadre'$);
- Une fois l'arbre construit, les prédictions de l'arbre pour chaque région se déduisent des données d'entraînement de façon intuitive: il s'agira de la classe la

plus fréquente parmi les observations situées dans cette région dans le cas d'une classification, et de la moyenne des observations situées dans cette région dans le cas d'une régression.

La structure de cet algorithme a deux conséquences importantes:

- L'algorithme CART ne fait **aucune hypothèse *a priori* sur la relation entre $X$ et $y$** et se laisse au contraire guider par les données. Par exemple, on ne suppose pas qu'il existe une relation linéaire de type $y = \mathbf{X}\beta$.
- **L'arbre final est une fonction constante par morceaux**: la prédiction est identique pour toutes les observations situées dans la même région, et ne varie que d'une région à l'autre.

Illustration, et représentation graphique (sous forme d'arbre et de graphique).

#### 2.3.1.2. Avantages et limites des arbres CART

Les arbres CART présentent plusieurs avantages: leur principe est simple, ils sont aisément interprétables et peuvent faire l'objet de représentations graphiques intuitives. Par ailleurs, la flexibilité offerte par le partitionnement récursif assure que les arbres obtenus reflètent les corrélations observées dans les données d'entraînement.

Ils souffrent néanmoins de deux limites. D'une part, les arbres CART ont souvent un **pouvoir prédictif faible** qui en limite l'usage. D'autre part, ils sont **peu robustes et instables**: on dit qu'ils présentent une **variance élevée**. Ainsi, un léger changement dans les données (par exemple l'ajout ou la suppression de quelques observations) peut entraîner des modifications significatives dans la structure de l'arbre et dans la définition des feuilles. Les arbres CART sont notamment sensibles aux valeurs extrêmes, aux points aberrants et au bruit statistique. De plus, les prédictions des arbres CART sont sensibles à de petites fluctuations des données: celles-ci peuvent aboutir à ce qu'une partie des observations change brutalement de feuille et donc de valeur prédite.

Les deux familles de méthodes ensemblistes présentées ci-dessous (*bagging*, *random forests* et *gradient boosting*) combinent un grand nombre d'arbres de décision pour en surmonter les deux limites: il s'agit d'obtenir un modèle dont le pouvoir prédictif est élevé et dont les prédictions sont stables. La différence essentielle entre ces deux familles portent sur la façon dont les arbres sont entraînés.

### 2.3.2. Le *bagging* (Bootstrap Aggregating) et les forêts aléatoires

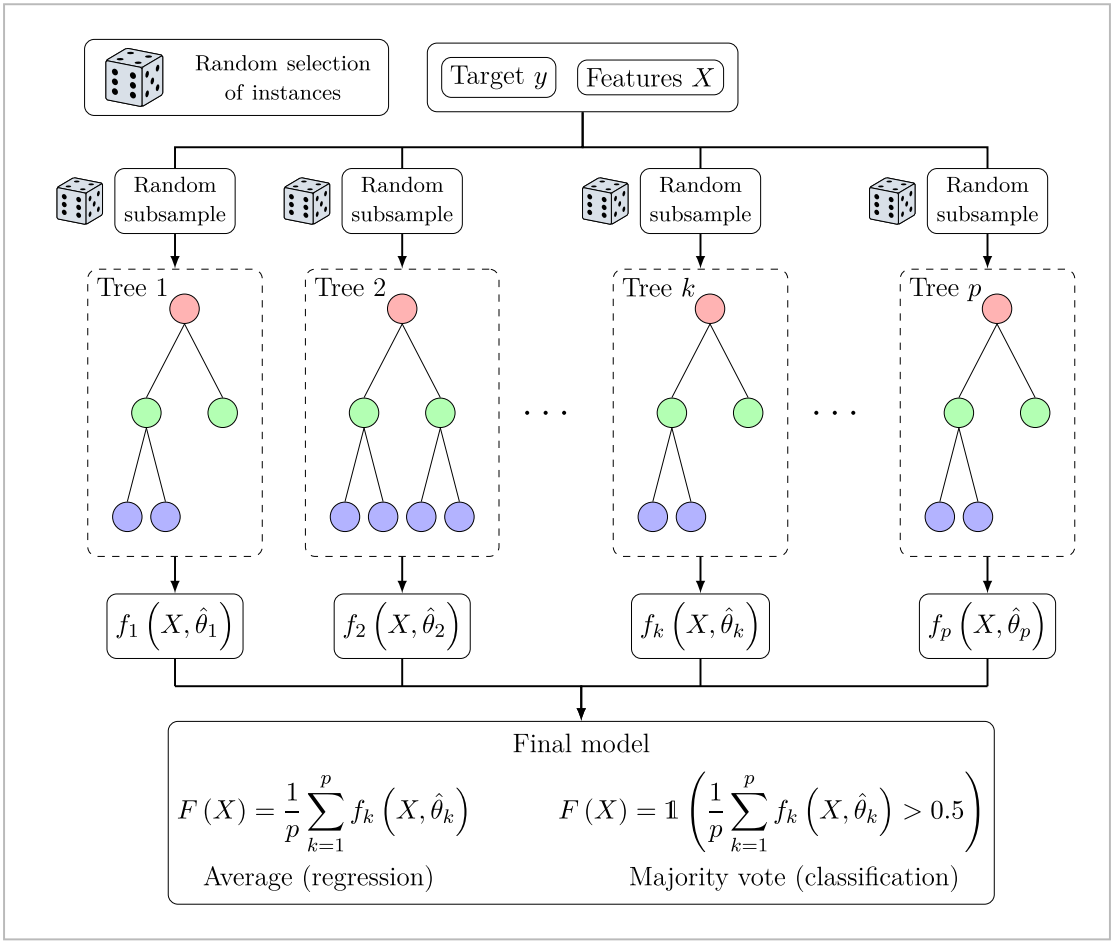
Le *bagging* (Bootstrap Aggregating) et les forêts aléatoires constituent une famille de méthodes ensemblistes dont le point commun est de combiner des modèles de bases qui ont été entraînés indépendamment les uns des autres.

#### 2.3.2.1. Le *bagging*

Le *bagging*, ou *Bootstrap Aggregating* (L. Breiman [5]), est une méthode ensembliste qui comporte trois étapes principales:

- **Tirage de sous-échantillons aléatoires**: À partir du jeu de données initial, plusieurs sous-échantillons sont générés par échantillonnage aléatoire avec remise (*bootstrapping*). Chaque sous-échantillon a la même taille que le jeu de données original, mais peut contenir des observations répétées, tandis que d'autres peuvent être omises.
- **Entraînement parallèle**: Un arbre est entraîné sur chaque sous-échantillon de manière indépendante. Ces arbres sont habituellement assez complexes et profonds.
- **Agrégation des prédictions**: Les prédictions des modèles sont combinées pour produire le résultat final. En classification, la prédiction finale est souvent déterminée par un vote majoritaire, tandis qu'en régression, elle correspond généralement à la moyenne des prédictions.

**Figure 1 : Représentation schématique d'un algorithme de *bagging***



La figure Figure 1 propose une représentation schématique du *bagging*: tout d'abord, on tire des sous-échantillons aléatoires des données d'entraînement. Ensuite, un arbre est entraîné sur chaque sous-échantillon. Enfin, les arbres sont combinés de façon à obtenir la prédiction finale.

- Illustration avec un cas d'usage de classification en deux dimensions.

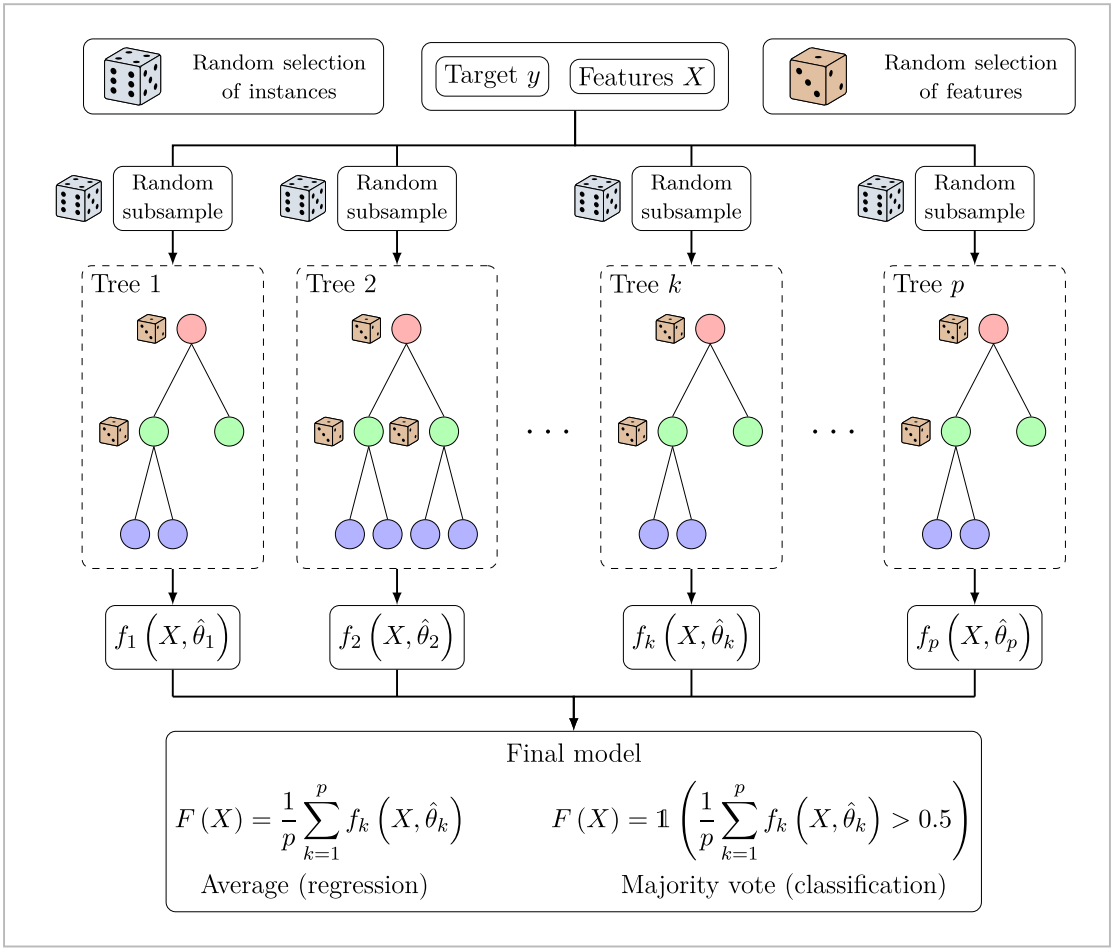
L'intuition qui explique l'efficacité du *bagging* est la suivante: en diversifiant les données d'entraînement par le tirage d'échantillons aléatoires, on obtient un grand nombre d'arbres différents les uns des autres et qui, pris dans leur ensemble, constituent un modèle plus prédictif et plus stable que chaque arbre pris isolément. Une image fréquemment employée pour décrire le *bagging* est celle d'un collège de juges. Chaque juge a sa propre façon de juger, qui est imparfaite et qui dépend des cas qu'il a déjà rencontrés. Il peut donc rendre une décision complètement erronée dans telle ou telle situation, rendant son verdict instable et peu fiable. Mais si le verdict repose sur l'opinion majoritaire d'un ensemble de juges différents les uns des autres, il est probable que le jugement sera plus robuste et plus fiable.

Le *bagging* présente donc deux avantages par rapport aux arbres CART: un pouvoir prédictif plus élevé et des prédictions plus stables. L'inconvénient du *bagging* réside dans la corrélation des arbres entre eux: malgré l'échantillonnage des données, les arbres ont souvent une structure similaire car les relations entre variables restent à peu près les mêmes dans les différents sous-échantillons. Ce phénomène de corrélation entre arbres est le principal frein à la puissance prédictive du *bagging*, et c'est pour surmonter (ou au moins minimiser) ce problème que les forêts aléatoires ont été mises au point. Le pouvoir prédictif plus élevé des forêts aléatoires explique pourquoi le *bagging* est très peu utilisé en pratique aujourd'hui.

#### 2.3.2.2. Les *random forests*

Les forêts aléatoires (*random forests*, L. Breiman [6]) sont une variante du *bagging* qui vise à produire des modèles très performants en conciliant deux objectifs: maximiser le pouvoir prédictif des arbres pris isolément, et minimiser la corrélation entre ces arbres (le problème inhérent au *bagging*). Pour atteindre ce second objectif, la forêt aléatoire introduit une nouvelle source de variation aléatoire dans la construction des arbres: au moment de choisir une règle de décision pour diviser une région en deux sous-régions, la procédure d'entraînement ne considère qu'**un sous-ensemble de variables sélectionnées aléatoirement**, et non toutes les variables. Cette randomisation supplémentaire a pour effet mécanique d'aboutir à des arbres plus diversifiés (parce que des arbres différents ne peuvent pas mobiliser les mêmes variables au même moment) et donc de **réduire la corrélation entre arbres**, ce qui permet d'améliorer la performance et la stabilité du modèle agrégé. Un enjeu important de l'entraînement d'une forêt aléatoire est l'arbitrage entre puissance prédictive des arbres et corrélation entre arbres.

**Figure 2 : Représentation schématique d'un algorithme de forêt aléatoire**




La figure Figure 2 propose une représentation schématique d'une forêt aléatoire. La logique d'ensemble est identique à celle du *bagging*: combinés de façon à obtenir la prédiction finale. La seule différence est que la liste des variables utilisables pour construire des règles de décision varie à chaque étape de l'entraînement. Cette restriction de la liste des variables considérées permet de réduire l'utilisation des variables les plus prédictives et de mieux mobiliser l'information disponible dans les variables peu corrélées avec $y$.

Contrairement au *bagging*, les forêts aléatoires sont un algorithme qui est très largement employé pour plusieurs raisons: les forêts aléatoires ont un faible nombre d'hyperparamètres, sont généralement peu sensibles aux valeurs de ces hyperparamètres et proposent de bonnes performances avec les valeurs par défaut. Les forêts aléatoires sont toutefois sujettes au problème de surapprentissage (voir encadré), bien que dans une mesure moindre que le *gradient boosting*.

Les forêts aléatoires présentent également un avantage de taille: **il est possible d'évaluer la qualité d'une forêt aléatoire en utilisant les données sur les-**

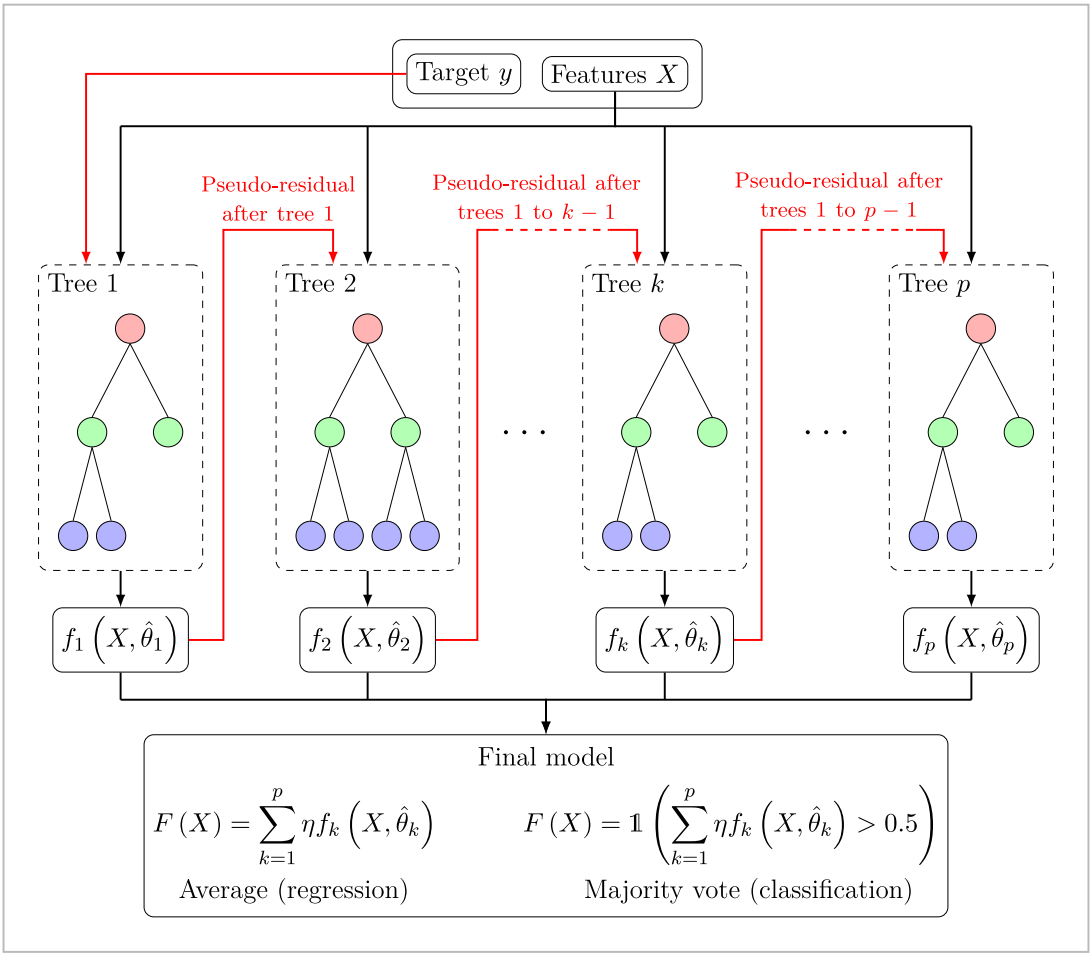
**quelles elle a été entraînée**, sans avoir besoin d'un jeu de test séparé. En effet, lors de la construction de chaque arbre, l'échantillonnage aléatoire implique que certaines observations ne sont pas utilisées pour entraîner cet arbre; ces observations sont dites *out-of-bag*. On peut donc construire pour chaque observation une prédiction qui agrège uniquement les arbres pour lesquels cette observation est *out-of-bag*; cette prédiction n'est pas affectée par le surapprentissage. De cette façon, il est possible d'évaluer correctement la performance de la forêt aléatoire.

> **i** Qu'est-ce que le surapprentissage?
>
> Le surapprentissage (*overfitting*) est un phénomène fréquent en *machine learning* où un modèle apprend non seulement les relations sous-jacentes entre la variable cible et les variables explicatives, mais également le bruit présent dans les données d'entraînement. En capturant ces fluctuations aléatoires plutôt que les tendances générales, le modèle affiche une performance excellente mais trompeuse sur les données d'entraînement, et s'avère médiocre sur des données nouvelles ou de test, car il ne parvient pas à généraliser efficacement.

### 2.3.3. Le *gradient boosting*

Alors que les forêts aléatoires construisent un ensemble d'arbres complexes et indépendants les uns des autres, le *gradient boosting* adopte une autre approche, dans laquelle les arbres sont peu complexes et entraînés de façon séquentielle, chaque arbre essayant d'améliorer la prédiction proposée par l'ensemble des arbres précédents. Bien qu'elles ressemblent fortement aux forêts aléatoires en apparence, il est important de noter que les approches de *boosting* reposent sur des fondements théoriques très différents. La logique du *gradient boosting* est illustrée par la figure Figure 3:

**Figure 3 : Représentation schématique d'un algorithme de *gradient boosting***




- Un premier modèle simple et peu performant est entraîné sur les données.
- Un deuxième modèle est entraîné de façon à corriger les erreurs du premier modèle (par exemple en pondérant davantage les observations mal prédites);
- Ce processus est répété en ajoutant des modèles simples, chaque modèle corrigeant les erreurs commises par l'ensemble des modèles précédents;
- Tous ces modèles sont finalement combinés (souvent par une somme pondérée) pour obtenir un modèle complexe et performant.

Il s'avère que le *gradient boosting* offre des performances prédictives particulièrement élevées. Toutefois, cet avantage incontestable ne doit pas masquer les sérieux inconvénients de cette approche: les algorithmes de *gradient boosting* comprennent un nombre élevé d'hyperparamètres et sont plus sensibles que les forêts aléatoires aux valeurs de ces hyperparamètres. Par ailleurs, ces algorithmes se caractérisent par un risque élevé de surapprentissage, et sont assez sensibles au bruit statistique et aux éventuelles erreurs

sur $y$. Par conséquent, l'usage de ces algorithmes est plus délicat, et l'optimisation de leurs hyperparamètres est une étape importante qui peut prendre un certain temps et demande une bonne connaissance des algorithmes.

## 2.4. Comparaison entre forêts aléatoires et *gradient boosting*

Les forêts aléatoires et le *gradient boosting* paraissent très similaires au premier abord: il s'agit de deux approches ensemblistes, qui construisent des modèles très prédictifs performants en combinant un grand nombre d'arbres de décision. Mais en réalité, ces deux approches présentent plusieurs différences fondamentales:

- Les deux approches reposent sur des **fondements théoriques différents**: la loi des grands nombres pour les forêts aléatoires, la théorie de l'apprentissage statistique pour le *boosting*.
- **Les arbres n'ont pas le même statut dans les deux approches**. Dans une forêt aléatoire, les arbres sont entraînés indépendamment les uns des autres et constituent chacun un modèle à part entière, qui peut être utilisé, représenté et interprété isolément. Dans un modèle de *boosting*, les arbres sont entraînés séquentiellement, ce qui implique que chaque arbre n'a pas de sens indépendamment de l'ensemble des arbres qui l'ont précédé dans l'entraînement. Par ailleurs, les arbres d'une forêt aléatoire sont relativement complexes et profonds (car ce sont des modèles à part entière), alors que dans le *boosting* les arbres sont plus souvent simples et peu profonds.
- Les **points d'attention lors de l'entraînement** des algorithmes sont différents: l'enjeu principal de l'entraînement d'une forêt aléatoire est trouver le bon arbitrage entre puissance prédictive des arbres et corrélation entre arbres, tandis que l'entraînement d'un algorithme de *gradient boosting* porte davantage sur la lutte contre le surapprentissage.
- **Complexité d'usage**: les forêts aléatoires s'avèrent plus faciles à prendre en main que le *gradient boosting*, car elles comprennent moins d'hyperparamètres dont l'optimisation est moins complexe.
- **Conditions d'utilisation**: il est possible d'évaluer la qualité d'une forêt aléatoire en utilisant les données sur lesquelles elle a été entraînée grâce à l'approche *out-of-bag*, alors que c'est impossible avec le *gradient boosting*, pour lequel il faut impérativement conserver un ensemble de test. Cette différence peut sembler purement technique en apparence, mais elle s'avère importante en pratique dans de nombreuses situations, par exemple lorsque les données disponibles sont de taille restreinte ou lorsque les ressources informatiques disponibles ne sont pas suffisantes pour mener un exercice de validation croisée.

#### 2.4.1. Quelle approche choisir?

**2.5. Le point de départ recommandé est de commencer par entraîner une forêt aléatoire avec les hyperparamètres par défaut.**

# References

[1] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?," *Advances in neural information processing systems*, vol. 35, pp. 507–520, 2022.

[2] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Information Fusion*, vol. 81, pp. 84–90, 2022.

[3] D. McElfresh *et al.*, "When do neural nets outperform boosted trees on tabular data?," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[4] L. Breiman, J. Friedman, R. Olshen, and C. Stone, "Cart," *Classification and regression trees*, 1984.

[5] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, pp. 123–140, 1996.

[6] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.