

Introduction au Machine Learning



Séance 8 — La force du nombre : les méthodes d'ensemble

Dr. El Hadji Bassirou TOURÉ

Département de Mathématiques et Informatique, Faculté des Sciences et
Techniques, Université Cheikh Anta Diop de Dakar

<https://ebtoure.sn/cours/introduction-machine-learning/chapitres/08-methodes-ensemble.html>

Version du 11 octobre 2026



Version en ligne

Introduction au ML — Séance 8

La force du nombre : les méthodes d'ensemble

Dr. El Hadji Bassirou TOURÉ

Département de Mathématiques et Informatique, Faculté des Sciences et Techniques, Université Cheikh Anta Diop de Dakar

Objectifs de la séance

Contenu

La Séance 7 a réglé un arbre du mieux possible : bien dosé, il atteignait AUC 0,73 sur le paludisme — puis butait. Cette séance dépasse ce plafond non pas en perfectionnant *un* modèle, mais en en combinant *beaucoup*, selon deux philosophies opposées.

- La **sagesse de la foule** : pourquoi, et à quelle condition, agréger des modèles médiocres bat un expert unique.
- Le **bagging** : le rééchantillonnage *bootstrap*, l'agrégation, et l'évaluation *out-of-bag* gratuite.
- La **forêt aléatoire** : décorréler les arbres (et pourquoi cela compte *quantitativement*), l'importance des variables et son piège.
- Le **boosting** : des arbres séquentiels qui corrigent leurs erreurs ; le gradient boosting comme descente de gradient.
- **Bagging contre boosting**, et la leçon centrale : le modèle le plus puissant n'est pas toujours le meilleur.

Le plafond de l'arbre unique

Le dilemme de l'arbre seul

Souple ou stable : il faut choisir

Un arbre profond mémorise l'entraînement — **variance élevée** : déplacez quelques patients, et l'arbre change du tout au tout. Un arbre bridé généralise mais manque de finesse — **biais élevé**. La validation croisée (Séance 7) trouvait le meilleur compromis, mais ce compromis est un *plafond* : un seul arbre ne peut être à la fois souple et stable.

Rappel chiffré de la Séance 7, sur la durée d'hospitalisation : l'arbre de profondeur 10 obtenait $R^2 = 0,726$ en entraînement mais 0,461 en test — l'écart signe la variance. Le bridant à la profondeur 6, on stabilisait à $R^2 \approx 0,55$, sans jamais aller plus haut.

L'idée de la séance : contourner le dilemme

L'idée en une phrase

Plutôt que de chercher l'arbre parfaitement dosé, on laisse chaque arbre être imparfait — puis on en combine un grand nombre, de sorte que leurs défauts se compensent.

Deux façons de combiner, qui structurent toute la séance :

- laisser les arbres **instables** (profonds, à variance élevée) et **moyenner** pour effacer la variance — c'est le *bagging* et les *forêts* ;
- prendre des arbres **trop simples** (à biais élevé) et les **enchaîner** pour effacer le biais — c'est le *boosting*.

Réduire la variance, ou réduire le biais : deux familles, deux mécaniques, qu'il faut comprendre séparément avant de les opposer.

La sagesse de la foule

L'idée : un jury vaut mieux qu'un juge

Analogie

Un jury de quartier juge une affaire. Chaque juré pris isolément se trompe souvent — disons une fois sur trois. Mais si leurs erreurs sont *indépendantes* (ils ne se copient pas), la **majorité** se trompe bien plus rarement que chacun : les fautes des uns sont noyées par les votes corrects des autres. Un collège de votants médiocres mais variés surpasse n'importe lequel d'entre eux.

L'idée en une phrase

Agréger les prédictions de nombreux modèles **décorrélés** fait tomber l'erreur de l'ensemble en dessous de l'erreur de chaque membre — à la stricte condition que leurs erreurs ne soient pas les mêmes.

La foule, formellement

Définition

Pour M votants indépendants ayant chacun raison avec probabilité $p > 0,5$, la probabilité que la *majorité* ait raison suit une loi binomiale, et croît vers 1 avec M .

$$P(\text{majorité correcte}) = \sum_{k > M/2} \binom{M}{k} p^k (1-p)^{M-k}$$

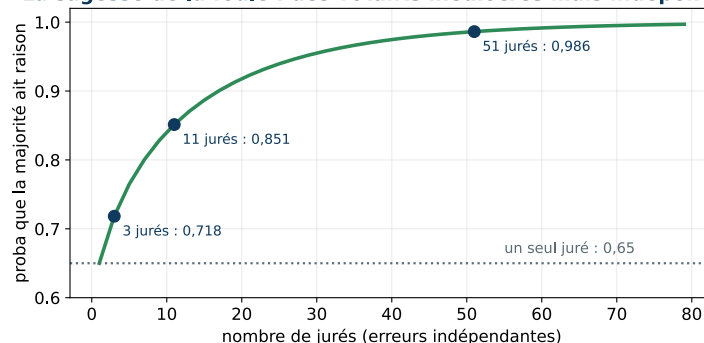
Pour $p > 0,5$, cette somme tend vers 1 quand M grandit (théorème du jury de Condorcet, 1785). La condition $p > 0,5$ est indispensable : si chaque votant fait *pire* que le hasard, la majorité empire elle aussi. Et tout repose sur **l'indépendance** — la formule binomiale n'est valable que si les erreurs ne sont pas corrélées.

Exemple — la foule en chiffres

Des jurés indépendants, chacun correct à 65 %, décision à la majorité :

nombre de jurés	majorité correcte
1	0,650
3	0,718
11	0,851
51	0,986

La sagesse de la foule : des votants médiocres mais indépendants



Onze votants à 65 % atteignent 85 % ; cinquante et un, près de 99 %. **Mais** l'indépendance est tout : cinquante et un jurés qui copient le même se trompent encore 35 % du temps. Tout l'art du métier sera de fabriquer des arbres qui *ne se ressemblent pas*.

Le lien avec biais et variance

Pourquoi moyenner aide — et ce que ça ne corrige pas

Moyenner plusieurs modèles ne change pas leur *biais* commun : si tous se trompent dans le même sens, la moyenne aussi. En revanche, cela réduit leur *variance* — les écarts individuels dus au hasard du tirage se compensent. C'est exactement le profil d'un arbre profond : beaucoup de variance, peu de biais.

De là, les deux stratégies de la séance, désormais nommées précisément : **(1)** prendre des modèles à *variance* élevée et la faire tomber en moyennant — le bagging et les forêts ; **(2)** prendre des modèles à *biais* élevé et le réduire en les enchaînant — le boosting. La prochaine section quantifie exactement de combien moyenner réduit la variance, et pourquoi la décorrélation y est décisive.

Exercice — la foule est-elle toujours sage ?

Énoncé

- a. Onze médecins indépendants diagnostiquent juste à 70 % ; la majorité fait-elle mieux ou moins bien que chacun ? (b) Onze médecins qui se concertent et rendent tous le *même* avis (corrélation parfaite) : que vaut leur « majorité » ? (c) Onze médecins indépendants justes à seulement 40 % : la foule aide-t-elle ?

Correction

(a) Mieux : avec $p = 0,70 > 0,5$ et l'indépendance, la majorité dépasse 0,70 (et grimpe vers 1 avec le nombre). **(b)** Exactement 0,70 — onze avis identiques valent un seul avis ; la corrélation parfaite annule tout le bénéfice de la foule. **(c)** Non, elle *aggrave* : $p = 0,40 < 0,5$, donc la majorité fait *pire* que 40 % — une foule de votants sous le hasard converge vers l'erreur. Conditions vitales : $p > 0,5$ et indépendance.

Le bagging

L'idée : varier les données pour varier les arbres

L'idée en une phrase

Pour fabriquer des arbres différents à partir d'un *seul* jeu de données, on entraîne chacun sur un **échantillon bootstrap** — un tirage avec remise de même taille — puis on agrège : vote majoritaire en classification, moyenne en régression. C'est le **bagging** (*bootstrap aggregating*).

Chaque arbre voit une version légèrement déformée des données : certains patients en double, d'autres absents. Les arbres diffèrent donc, leurs erreurs se décorrèlent *un peu*, et la moyenne gagne en stabilité. Reste à comprendre ce qu'est précisément ce tirage, et combien il laisse de patients de côté.

Le mécanisme — le bootstrap, à la main

Un tirage avec remise sur six patients

Six patients numérotés 1 à 6. Un échantillon bootstrap tire 6 fois *avec remise* — un même patient peut sortir plusieurs fois, un autre pas du tout :

tirage : {2, 3, 4, 5, 5, 6} $\Rightarrow$ vus : {2, 3, 4, 5, 6}, absent : {1}.

Le patient 5 compte double (il pèsera plus dans cet arbre) ; le patient 1 n'a pas servi — il est **out-of-bag** (OOB) pour cet arbre. Un autre arbre tirera un autre échantillon, laissera d'autres patients de côté : c'est cette diversité qui décorrèle les arbres.

Sur un grand jeu, la proportion de patients *vus* se stabilise à une valeur remarquable — que le calcul suivant établit.

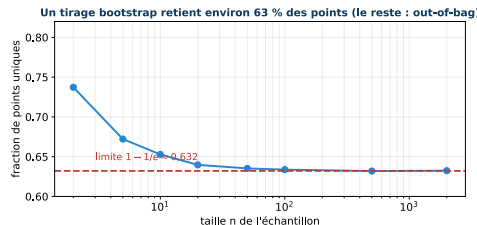
Pourquoi 63 % ? le calcul détaillé

La limite $1 - 1/e$

À chaque tirage, un patient donné a une chance $\frac{1}{n}$ d'être choisi, donc $1 - \frac{1}{n}$ de *ne pas* l'être. Les n tirages étant indépendants, la probabilité qu'il soit absent des n tirages est le produit.

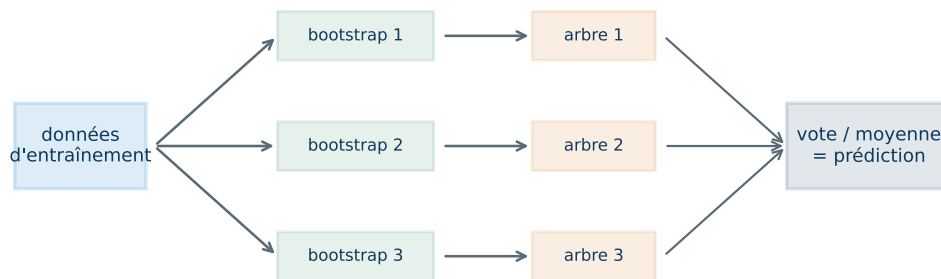
$$P(\text{absent}) = \left(1 - \frac{1}{n}\right)^n \xrightarrow{n \rightarrow \infty} \frac{1}{e} \approx 0,368$$

(Rappel : $\left(1 + \frac{x}{n}\right)^n \rightarrow e^x$; ici $x = -1$, d'où e^{-1} .) Donc un patient a $\approx 37\%$ de chances d'être out-of-bag, et $\approx 63\%$ de figurer dans l'échantillon. Vérification : $n = 6 \rightarrow 0,335$; $n = 100 \rightarrow 0,366$; $n = 10\,000 \rightarrow 0,368$. La convergence est rapide.



Le schéma du bagging

Le bagging : des arbres entraînés sur des tirages différents, puis agrégés



Un seul jeu de données engendre plusieurs échantillons bootstrap ; un arbre par échantillon, entraîné indépendamment ; les prédictions sont agrégées en une seule (moyenne ou vote). Chaque arbre est individuellement instable — mais leurs instabilités se compensent à l'agrégation. Comme les arbres ne dépendent pas les uns des autres, on peut les entraîner *en parallèle*.

Le bagging, formellement

Définition

Soit B échantillons bootstrap, et $\hat{f}_b$ l'arbre entraîné sur le b -ième. La prédiction du bagging agrège les B arbres : par moyenne des probabilités (classification, ou régression), ou par vote majoritaire des classes.

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

Les arbres baggés sont délibérément *profonds* (non élagués) : leur biais est faible, et le bagging se charge d'écraser leur variance. C'est l'inverse du réflexe « bridons l'arbre » de la Séance 7 — ici, on *veut* des arbres instables, car on dispose du moyen de les stabiliser collectivement.

L'évaluation out-of-bag : une validation gratuite

L'idée en une phrase

Puisque chaque patient est OOB pour environ un tiers des arbres, on peut le faire prédire *par ces arbres-là seulement* — qui ne l'ont jamais vu. En agrégeant ces prédictions OOB sur tous les patients, on obtient une estimation de performance **sans réserver de jeu de validation**.

Le mécanisme reproduit exactement l'esprit de la validation croisée (Séance 7) — juger chaque donnée par des modèles qui l'ignorent — mais il est offert par le bagging sans calcul supplémentaire : les arbres qui n'ont pas vu un patient existent déjà, il suffit de les interroger. C'est un avantage propre aux méthodes baggées.

Exemple — l'out-of-bag sur le paludisme

Un score gratuit

Un bagging de 100 arbres sur le paludisme affiche un **score OOB** de 0,869 (accuracy), calculé sans toucher au test : chaque patient d'entraînement a été prédit par les ≈ 37 arbres qui ne l'avaient pas dans leur échantillon. C'est un diagnostic immédiat de la qualité de l'ensemble, disponible à la fin du [fit](#).

Deux nuances. L'OOB est une accuracy : sur cette classe rare (11,8 % de positifs), 0,869 doit se lire *en regard de la baseline* 0,882 (Séance 6) — ce bagging ne la bat pas en accuracy, et il faudra le juger à l'AUC. Et le test reste **scellé** : l'OOB sert au développement, pas au verdict final.

Exercice — bootstrap et out-of-bag

Énoncé

Un échantillon bootstrap est tiré sur cinq patients $\{1, 2, 3, 4, 5\}$; le tirage donne $\{2, 2, 4, 5, 5\}$. (a) Quels patients sont vus, lesquels sont out-of-bag ? (b) Le patient 3 peut-il servir à évaluer l'arbre entraîné sur ce tirage, et pourquoi ? (c) Sur un très grand jeu, quelle fraction de patients un échantillon bootstrap laisse-t-il en moyenne de côté ?

Correction

(a) Vus : $\{2, 4, 5\}$ (le 2 et le 5 comptent double) ; out-of-bag : $\{1, 3\}$. **(b)** Oui — le patient 3 n'a pas servi à entraîner cet arbre, il en est OOB ; le faire prédire par cet arbre est une évaluation honnête, car l'arbre ne l'a jamais mémorisé. **(c)** Environ $1/e \approx 37\%$ sont laissés de côté (et $\approx 63\%$ retenus) — la limite de $\left(1 - 1/n\right)^n$.

La forêt aléatoire

Pourquoi le bagging ne suffit pas toujours

Des arbres baggés peuvent rester semblables

Si une variable est très prédictive, *tous* les arbres baggés la choisissent en haut de leur structure — ils se ressemblent malgré le bootstrap, et leurs erreurs restent fortement corrélées. Or moyenner des modèles corrélés ne réduit presque pas la variance : le bénéfice de la foule s'effondre quand les votants se ressemblent.

Pour mesurer l'enjeu, il faut quantifier exactement comment la *corrélation* entre arbres limite la réduction de variance — c'est l'objet de la formule suivante, et elle justifiera à elle seule l'ingrédient des forêts.

Le cœur quantitatif : la variance d'une moyenne corrélée

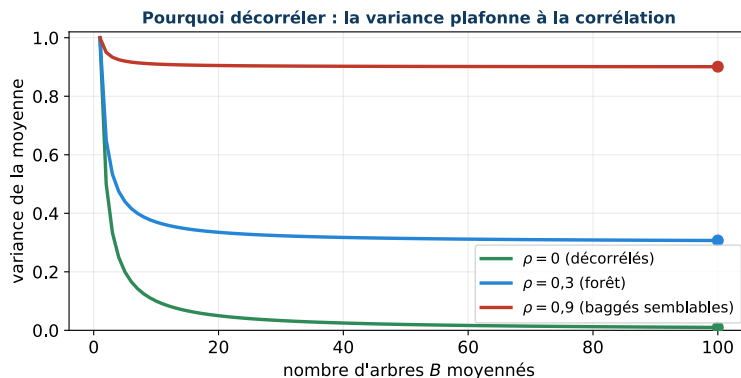
Définition

La variance de la moyenne de B modèles, chacun de variance σ^2 et deux à deux corrélés au coefficient ρ , n'est *pas* σ^2/B : un terme irréductible subsiste, gouverné par ρ .

$$\text{Var}\left(\frac{1}{B} \sum_b \hat{f}_b\right) = \rho \sigma^2 + \frac{1-\rho}{B} \sigma^2 \xrightarrow{B \rightarrow \infty} \rho \sigma^2$$

Lecture décisive : ajouter des arbres ($B \rightarrow \infty$) fait disparaître le second terme, mais **laisse** $\rho \sigma^2$. La variance de l'ensemble ne descend jamais sous la *corrélation* entre arbres. Donc : pour gagner, il ne suffit pas d'ajouter des arbres — il faut *baisser* ρ . C'est toute la raison d'être de la forêt aléatoire.

L'effet de la corrélation, en image



À corrélation $\rho = 0,9$ (arbres baggés qui se ressemblent), cent arbres ne font tomber la variance que de 1,0 à 0,90 — presque rien. À $\rho = 0,3$ (arbres décorrélés), de 1,0 à 0,31. À $\rho = 0$ (idéal inatteignable), à 0,01. Le message est sans appel : **la décorrélation, pas le nombre, est le vrai levier**. Reste à savoir comment décorréler — et la forêt a une idée simple.

L'idée de la forêt : décorréler à chaque coupure

L'idée en une phrase

La **forêt aléatoire** ajoute au bagging un second hasard : à *chaque coupure* de chaque arbre, seules `max_features` variables tirées au sort sont candidates. Aucun arbre ne peut alors s'appuyer systématiquement sur la même variable dominante — leur corrélation ρ chute, et la moyenne devient bien plus efficace.

L'astuce est subtile : on *dégrade volontairement* chaque arbre (en lui cachant des variables) pour *améliorer* l'ensemble. Un arbre pris seul devient un peu moins bon ; mais comme les arbres se ressemblent moins, leur moyenne — la seule chose qui compte — gagne sur le tableau précédent.

La forêt aléatoire, formellement

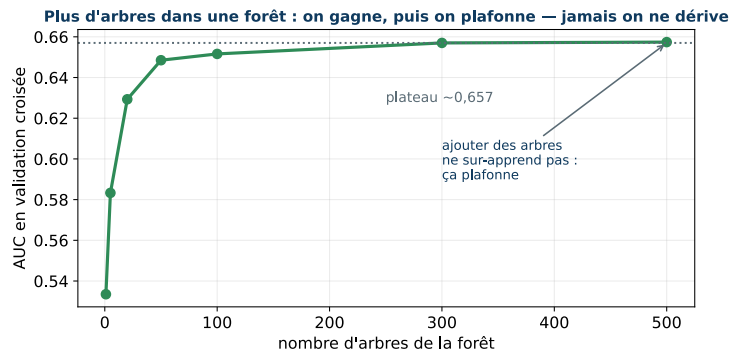
Définition

Une forêt aléatoire est un bagging d'arbres profonds où chaque nœud ne choisit sa coupure que parmi `max_features` variables tirées au hasard. La prédiction est la moyenne des probabilités des B arbres ainsi décorrélés.

$$\hat{p}(x) = \frac{1}{B} \sum_{b=1}^B \hat{p}_b(x) \quad (B = \text{n_estimators}, \text{ coupures sur max_features variables})$$

Deux réglages principaux : `n_estimators` (le nombre d'arbres) et `max_features` (le degré de décorrélation). Le défaut de `max_features` est $\sqrt{d}$ variables par coupure en classification — un point de départ éprouvé. Une variante, les *extra-trees*, pousse le hasard plus loin en tirant aussi les *seuils* au sort : encore plus de décorrélation, des arbres encore plus faibles.

Plus d'arbres : on gagne, puis on plafonne



AUC de validation croisée selon le nombre d'arbres : $1 \rightarrow 0,53$; $20 \rightarrow 0,63$; $100 \rightarrow 0,65$; $500 \rightarrow 0,66$. C'est la formule de variance en action : le terme $\frac{1-\rho}{B} \sigma^2$ s'éteint, on bute sur le plancher $\rho \sigma^2$. Propriété capitale qui en découle : ajouter des arbres à une forêt ne fait **jamais sur-apprendre** — la courbe monte puis plafonne. `n_estimators` n'est donc pas à régler finement : on en met « assez » (quelques centaines), le surcoût est en temps de calcul, pas en qualité.

Le réglage de la décorrélation : `max_features`

Le seul réglage vraiment sensible de la forêt

`max_features` dose directement la corrélation ρ : *petit* $\Rightarrow$ arbres très décorrélés (mais chacun plus faible) ; *grand* (toutes les variables) $\Rightarrow$ on retombe sur le bagging, ρ élevé. C'est un compromis entre décorrélation et force individuelle.

Sur le paludisme, AUC de validation croisée

<code>max_features</code>	1	2	3	$\sqrt{d}$	toutes
AUC CV	0,659	0,657	0,649	0,657	0,647

La valeur la plus restrictive (1 variable par coupure) donne ici la meilleure AUC, et « toutes les variables » (le bagging pur) la plus faible — confirmation directe : sur ces données, plus on décorrèle, mieux c'est. L'écart reste modeste car le problème ne compte que 5 variables.

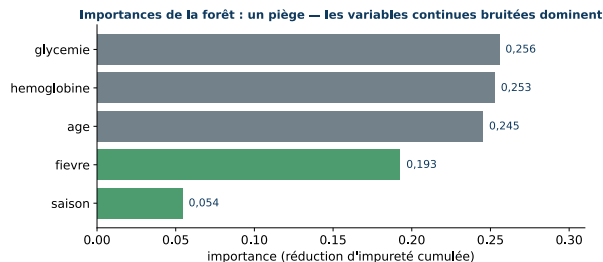
L'importance des variables

L'idée en une phrase

La forêt peut mesurer combien chaque variable a fait *baisser l'impureté* (Gini, Séance 3), cumulé sur tous les nœuds qui l'utilisent et tous les arbres — donnant un classement de l'utilité apparente des variables.

C'est l'un des attraits majeurs des forêts : là où la logistique offrait des poids (Séance 6), la forêt offre un score d'importance par variable, applicable même quand les relations sont non linéaires. Séduisant — mais cette mesure recèle un piège sérieux, que le cas du paludisme va révéler crûment.

Le piège de l'importance par impureté



Lecture prudente

Sur le paludisme, glycémie, hémoglobine et âge ressortent « importantes » — alors que, *par construction des données*, la maladie ne dépend que de la **saison** et de la **fièvre**. Pourquoi cette trahison ? Ces variables **continues** offrent une multitude de seuils de coupure possibles : les arbres s'en servent abondamment pour fragmenter le bruit, gonflant leur impureté cumulée. L'importance par impureté **favorise mécaniquement les variables à forte cardinalité**, même non causales.

À retenir : l'importance *suggère* des pistes, elle ne *prouve* aucune causalité — et se méfie tout particulièrement des variables continues. Une mesure plus fiable (l'importance par permutation) existe, mais dépasse le cadre de la séance.

En pratique : RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier

foret = RandomForestClassifier(n_estimators=300, # assez d'arbres (ca plafonne)
                              max_features="sqrt", # decorrelation (default)
                              oob_score=True, # validation gratuite
                              random_state=0, n_jobs=-1)

foret.fit(X_train, y_train)
print(foret.oob_score_) # estimation OOB, sans toucher au test
print(foret.feature_importances_) # a lire avec prudence
proba = foret.predict_proba(X_test)[: , 1]
```

Aucun **StandardScaler** : les arbres travaillent par seuils, insensibles à l'échelle — contrairement à la logistique de la Séance 6 qui l'exigeait. `n_jobs=-1` entraîne les arbres en parallèle : le bagging est trivialement parallélisable, chaque arbre étant indépendant des autres.

Exercice — diagnostiquer une forêt

Énoncé

Deux forêts sont entraînées sur le même problème. Forêt A : `max_features=1`, AUC CV 0,74. Forêt B : `max_features` = toutes les variables, AUC CV 0,70. (a) Laquelle a les arbres les plus décorrélés ? (b) Pourquoi B, qui a des arbres individuellement *meilleurs*, obtient-elle un ensemble *moins* bon ? (c) Que vaudrait B, conceptuellement, par rapport à un simple bagging ?

Correction

(a) A : moins de variables candidates par coupure $\Rightarrow$ arbres plus variés $\Rightarrow \rho$ plus faible. **(b)** Parce que la performance de l'ensemble dépend de $\rho \sigma^2$, pas de la qualité d'un arbre seul : les arbres de B se ressemblent (tous prennent la variable dominante), leur moyenne réduit peu la variance. **(c)** B est un bagging : « toutes les variables candidates à chaque coupure » supprime le second hasard de la forêt — d'où sa corrélation élevée et son moindre rendement.

Le boosting

L'idée : apprendre de ses erreurs, en série

Analogie

Un élève qui révise : il traite une série d'exercices, repère ceux qu'il a ratés, et concentre la révision suivante *sur ses fautes* — pas sur ce qu'il maîtrise déjà. Le boosting fait pareil : chaque nouvel arbre se concentre sur ce que les précédents ont mal prédit. Les arbres ne votent plus en parallèle (comme la forêt) ; ils se **succèdent**, chacun corrigeant le cumul de tous ceux d'avant.

L'idée en une phrase

Le **boosting** construit les arbres *séquentiellement* : chacun est entraîné à rattraper l'**erreur résiduelle** de la somme des arbres déjà construits. On réduit ainsi le *biais* — l'exact opposé de la forêt, qui réduisait la variance.

Le mécanisme — le gradient boosting sur les résidus (1/3)

Quatre points, premier arbre

Jouet de régression : $x = (1, 2, 3, 4)$, $y = (2, 3, 5, 9)$. Le gradient boosting part d'une prédiction constante, puis ajoute des arbres (ici des *souches* : profondeur 1) ajustés sur les résidus.

- **Départ** $F_0 = \bar{y} = 4,75$ (la meilleure constante au sens de la MSE, Séance 5).
- **Résidus** $r_1 = y - F_0 = (-2,75; -1,75; 0,25; 4,25)$: ce qu'il reste à expliquer.
- **Souche 1** ajustée *sur ces résidus* : coupure optimale en $x = 3,5$, prédit $-1,42$ à gauche (moyenne des trois premiers résidus) et $+4,25$ à droite.

MSE de départ : 7,19.

Le mécanisme — le gradient boosting sur les résidus (2/3)

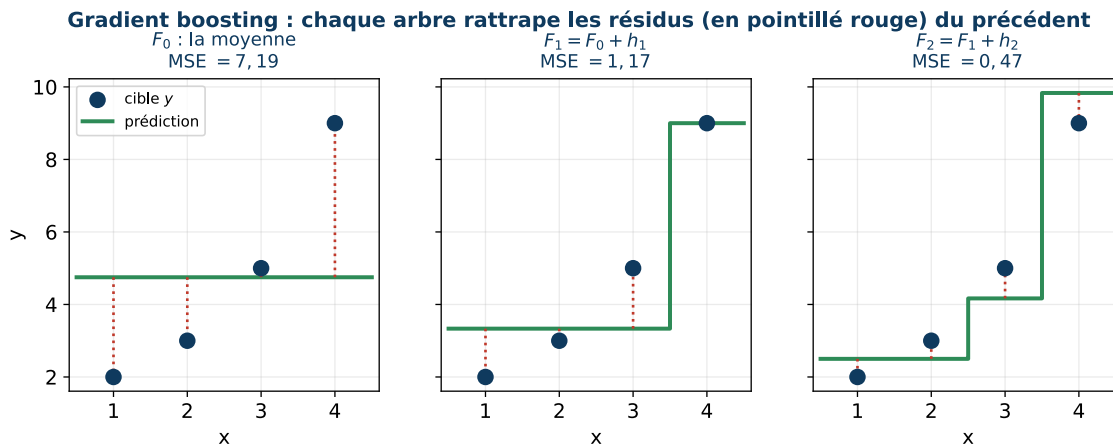
Mise à jour et deuxième arbre

- **Mise à jour** $F_1 = F_0 + h_1 = (3,33; 3,33; 3,33; 9,00)$: la prédiction s'est rapprochée de y .
- **Nouveaux résidus** $r_2 = y - F_1 = (-1,33; -0,33; 1,67; 0,00)$: plus petits que r_1 — l'erreur a fondu.
- **Souche 2** ajustée sur r_2 , puis $F_2 = F_1 + h_2 = (2,5; 2,5; 4,17; 9,83)$.

La MSE chute étape par étape : $7,19 \rightarrow 1,17 \rightarrow 0,47$. Chaque arbre ne corrige que le *reste* laissé par les précédents — jamais la cible entière.

La prédiction finale est la **somme** de tous les arbres, $F_0 + h_1 + h_2 + \dots$, et non leur moyenne : différence structurelle avec le bagging, où l'on divisait par B .

Le mécanisme — le gradient boosting sur les résidus (3/3)



Les pointillés rouges — les résidus que chaque arbre doit rattraper — raccourcissent visiblement à chaque étape : la prédiction (en vert) épouse de mieux en mieux la cible. C'est exactement le comportement « apprendre de ses fautes » de l'analogie, rendu quantitatif.

Pourquoi « gradient » ? le lien avec la Séance 5

Boosting et descente de gradient sont une même idée

Pour la perte MSE $\frac{1}{2}(y - F)^2$, la dérivée par rapport à la prédiction F est $-(y - F) = -\text{résidu}$. Ajuster un arbre sur les résidus, c'est donc le faire pointer dans la direction $-\nabla(\text{perte})$: un *pas de descente de gradient*, mais dans l'espace des fonctions au lieu de l'espace des poids.

$$r_i = y_i - F(x_i) = -\frac{\partial}{\partial F} \left[\frac{1}{2} (y_i - F(x_i))^2 \right]$$

La descente de gradient de la Séance 5 déplaçait des poids $(b, \mathbf{w})$; le gradient boosting ajoute des *fonctions* (des arbres), chacune approximant le gradient négatif de la perte. C'est la même mécanique d'optimisation, et c'est pourquoi il fonctionne avec n'importe quelle perte dérivable — dont la log-loss de la Séance 6 pour la classification.

Le taux d'apprentissage : ne pas tout corriger d'un coup

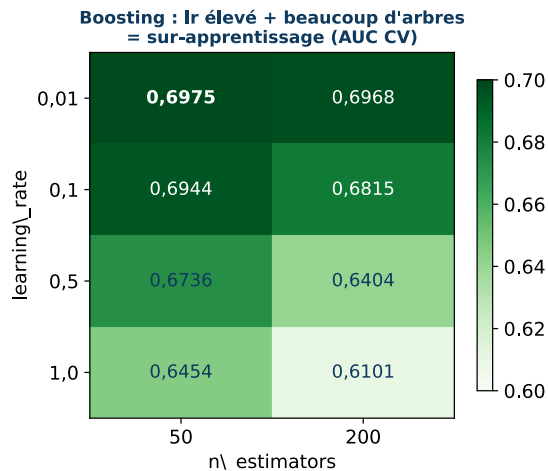
L'idée en une phrase

Ajouter chaque arbre *en entier* corrige vite mais brutalement — et finit par rattraper le bruit. On multiplie donc chaque arbre par un petit **taux d'apprentissage** `learning_rate` ν : des pas plus petits, plus nombreux, plus prudents.

$$F_m(x) = F_{m-1}(x) + \nu \cdot h_m(x) \quad (\nu = \text{learning_rate} \in]0, 1])$$

C'est le même ν que le pas de la descente de gradient (Séance 5) : trop grand, on oscille et on rattrape le bruit ; trop petit, on progresse lentement. `learning_rate` et `n_estimators` se règlent donc *ensemble* — un ν petit exige plus d'arbres pour le même progrès, mais généralise mieux.

Le boosting peut sur-apprendre



AUC de validation croisée sur le paludisme, selon le couple $(\nu, n_estimators)$. Le meilleur coin est en haut à gauche — $\nu = 0,01$, peu d'arbres : 0,698. En bas à droite — $\nu = 1$, beaucoup d'arbres : 0,610, le modèle a rattrapé le bruit de l'entraînement. **À l'exact opposé de la forêt** : là où ajouter des arbres faisait plafonner sans risque, ici « plus d'arbres » à grand pas *dégrade*. Le boosting est plus puissant, mais exige le réglage soigneux de la Séance 7.

En pratique : GradientBoostingClassifier

```
from sklearn.ensemble import GradientBoostingClassifier

gb = GradientBoostingClassifier(n_estimators=100,    # arbres sequentiels
                               learning_rate=0.1,  # pas (a regler avec n_estimators)
                               max_depth=3,        # arbres FAIBLES (peu profonds)
                               random_state=0)

gb.fit(X_train, y_train)
proba = gb.predict_proba(X_test)[:, 1]
```

Trois différences de fond avec la forêt : les arbres sont *peu profonds* (des « apprenants faibles » — le biais sera réduit par le *nombre* d'étapes, pas par la profondeur de chacune) ; l'entraînement est **séquentiel**, donc non parallélisable (chaque arbre attend la prédiction du précédent) ; et le couple `learning_rate/n_estimators` doit être réglé. Les bibliothèques **XGBoost** et **LightGBM** en sont des versions optimisées, omniprésentes dans les compétitions et l'industrie.

Exercice — une étape de boosting à la main

Énoncé

Sur trois points, la prédiction courante est $F_1 = (3; 5; 8)$ et la cible $y = (4; 4; 10)$. (a) Calculer les résidus $r_2 = y - F_1$. (b) Une souche prédit ces résidus par $(+0,5; +0,5; +2)$; avec $\nu = 0,5$, donner la nouvelle prédiction F_2 . (c) La somme des carrés des résidus a-t-elle diminué ? (d) Un point a-t-il été *sur-correct* ?

Correction

(a) $r_2 = (4 - 3; 4 - 5; 10 - 8) = (+1; -1; +2)$. **(b)** $F_2 = F_1 + 0,5 (0,5; 0,5; 2) = (3,25; 5,25; 9,00)$. **(c)** Avant : $1 + 1 + 4 = 6$; après : $0,5625 + 1,5625 + 1 = \mathbf{3,125}$ — oui, l'erreur a diminué de moitié. **(d)** Le point 2 : son résidu était -1 , la souche a ajouté $+0,25$, l'éloignant légèrement (de 5 à 5,25, alors que la cible est 4). Le pas prudent corrige globalement mais peut sur-corriger localement ; d'autres arbres rattraperont.

Bagging contre boosting

Deux familles, deux mécaniques

Le tableau qui tranche

	forêt aléatoire (bagging)	gradient boosting
arbres	profonds, forts	peu profonds, faibles
construits	en parallèle (indépendants)	en série (chacun corrige)
agrégation	moyenne / vote ($\div B$)	somme pondérée ($+\nu h_m$)
réduit surtout	la variance	le biais
trop d'arbres ?	plafonne (sans risque)	sur-apprend (dérive)
réglage clé	<code>max_features</code> (peu sensible)	ν et <code>n_estimators</code> (délicat)
parallélisable	oui	non

Lequel choisir ?

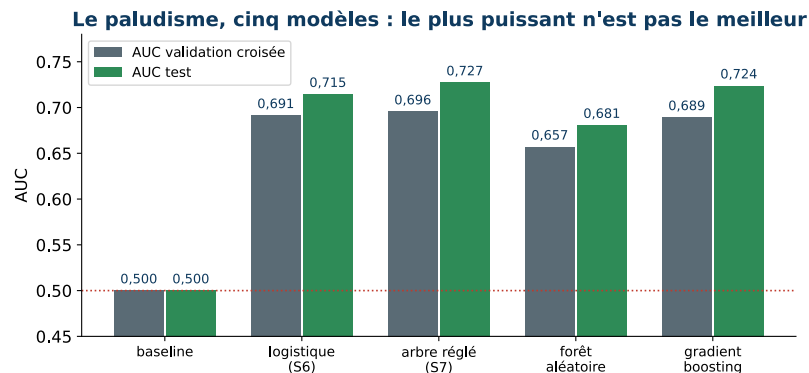
Une règle de pouce, à nuancer toujours par la validation croisée :

- la **forêt aléatoire** est le *premier réflexe* robuste : peu de réglage, difficile à rater, OOB gratuit, parallélisable — idéale pour un résultat solide rapidement ;
- le **gradient boosting** bien réglé est souvent *plus précis*, surtout quand le signal contient des interactions fines — au prix d'un effort de réglage (ν , `n_estimators`) et d'un risque réel de sur-apprentissage ;
- dans les deux cas, **aucune standardisation** n'est nécessaire (ce sont des arbres), et le verdict se prend en validation croisée puis sur le test scellé (Séance 7).

Mais « plus précis en général » ne veut pas dire « plus précis ici » : le cas du paludisme va le rappeler avec force.

Le verdict sur le paludisme

Le grand comparatif



Cinq modèles sur le paludisme, jugés à l'AUC en validation croisée et sur le test (méthode de la Séance 7) : baseline 0,50 ; logistique 0,71 ; arbre réglé 0,73 ; forêt 0,68 ; gradient boosting 0,72. Surprise : les ensembles *ne dominant pas*. L'arbre réglé et la logistique font aussi bien, voire mieux.

La leçon : plus puissant n'est pas meilleur

Pourquoi les ensembles ne gagnent pas ici

Le risque de paludisme, dans ces données, dépend essentiellement de deux variables de façon *simple* (saison, fièvre — c'est ainsi que le jeu est construit). Un modèle linéaire ou un petit arbre capture déjà ce signal. Les forêts et le boosting, en cherchant des interactions complexes dans des variables continues **non causales** (glycémie, âge), se laissent distraire par le bruit — leur puissance se retourne contre elles.

Ce n'est pas un défaut des ensembles : c'est une *inadéquation* entre la complexité du modèle et celle du problème. La même forêt, sur un jeu riche en interactions, écraserait la logistique.

Le principe de parcimonie

La morale, généralisée

Aucun modèle n'est supérieur dans l'absolu (c'est le théorème dit du « no free lunch »). La complexité d'un modèle doit être *justifiée par les données* : on n'emploie une forêt ou un boosting que si le gain en validation croisée le démontre.

Conséquence pratique, qui prolonge la discipline de la Séance 7 : comparer honnêtement *tous* les candidats — du plus simple au plus puissant — par validation croisée, puis, **à performance égale, préférer le plus simple**. Le plus simple est plus rapide à entraîner, plus facile à déployer, plus interprétable, et moins sujet au sur-apprentissage. La puissance ne se paie que lorsqu'elle rapporte.

Exercice — choisir un modèle

Énoncé

Sur un problème, on mesure en validation croisée : logistique AUC 0,71, forêt 0,72, gradient boosting réglé 0,80. Sur un *autre* problème : logistique 0,90, forêt 0,88, boosting 0,90. Pour chaque problème, quel modèle retenir, et que peut-on deviner sur la nature du signal ?

Correction

Problème 1 : le boosting (0,80) domine nettement — le signal contient probablement des *interactions complexes* que le linéaire et la forêt simple ne captent pas ; la puissance est ici justifiée, on retient le boosting. **Problème 2** : les trois se valent ($\approx 0,90$) — le signal est largement *linéaire*, et l'on choisit la **logistique** : même performance, modèle plus simple, plus rapide, plus interprétable. À performance égale, la simplicité l'emporte.

Synthèse

Le protocole d'ensemble

De bout en bout

1. Établir une baseline et un modèle simple (logistique, arbre réglé — Séances 6 et 7) → **2.** essayer une **forêt aléatoire** (quelques centaines d'arbres, défauts) : robuste, peu de réglage, OOB pour un premier diagnostic → **3.** si le signal semble riche, essayer un **gradient boosting** et régler $\nu/n_estimators$ par validation croisée → **4.** comparer *tous* les candidats à l'AUC en CV, jamais sur le test → **5.** à performance égale, choisir le plus simple → **6.** verdict final sur le test scellé, une seule fois.

À retenir — Séance 8

L'essentiel

- Agréger des modèles **décorrélés** bat chacun d'eux — à condition $p > 0,5$ et erreurs non corrélées (la sagesse de la foule, Condorcet).
- La variance d'une moyenne de B arbres tend vers $\rho \sigma^2$: **décorrélér ($\rho \downarrow$)**, **pas seulement multiplier les arbres**.
- **Bagging** : arbres profonds sur des échantillons *bootstrap* ($\approx 63\%$ des points), agrégés ; réduit la **variance** ; l'**OOB** offre une validation gratuite.
- **Forêt** : bagging + `max_features` variables par coupure (décorrélation) ; `n_estimators` plafonne (jamais de sur-apprentissage).
- L'**importance des variables** suggère, ne prouve pas — et favorise les variables continues.
- **Boosting** : arbres faibles *séquentiels* rattrapant les résidus ($= -\nabla$ perte) ; réduit le **biais** ; $\nu \times$ `n_estimators` décisif ; **peut** sur-apprendre.
- Leçon centrale : **le modèle le plus puissant n'est pas toujours le meilleur** — la complexité se justifie par les données (no free lunch).

Pièges fréquents (1/2)

- Croire qu'un ensemble bat toujours un modèle simple — le comparatif paludisme prouve le contraire.
- Penser que multiplier les arbres suffit : sans décorrélation, la variance plafonne à $\rho \sigma^2$, élevé.
- Régler finement `n_estimators` d'une forêt : inutile, ça plafonne — en mettre « assez » suffit.
- Lire l'importance des variables comme une preuve de causalité, et oublier son biais vers les variables continues.

Pièges fréquents (2/2)

- Confondre les deux familles : la forêt réduit la variance (arbres profonds, parallèles), le boosting le biais (arbres faibles, séquentiels).
- Pousser `n_estimators` d'un boosting à grand `learning_rate` : sur-apprentissage garanti.
- Standardiser pour une forêt ou un boosting : inutile, les arbres sont insensibles à l'échelle (à la différence de la logistique).
- Juger ces modèles sur le test plutôt qu'en validation croisée : la discipline de la Séance 7 reste entière.

Prochaine séance

Séance 9 — Apprendre sans étiquettes

Jusqu'ici, chaque patient portait une étiquette à prédire (durée, paludisme) : c'était l'apprentissage *supervisé*. La Séance 9 lâche les étiquettes : comment *regrouper* des patients similaires sans savoir d'avance en quels groupes — le clustering (*k*-moyennes) — et comment *résumer* des données à beaucoup de variables en en gardant l'essentiel — la réduction de dimension (PCA).

Références

- James, Witten, Hastie, Tibshirani — *An Introduction to Statistical Learning*, 2^e éd., Springer, 2021.
- Géron — *Hands-On Machine Learning*, 3^e éd., O'Reilly, 2022.
- Müller, Guido — *Introduction to Machine Learning with Python*, O'Reilly, 2016.
- Hastie, Tibshirani, Friedman — *The Elements of Statistical Learning*, 2^e éd., Springer, 2009.
- Cours : University of Washington CSE 446 ; Stanford CS229.

Ressources de la séance

- **Notebook** · [TP 8 — Méthodes d'ensemble](#) (45 Ko)
- **TD** · [Fiche d'exercices 8](#) (183 Ko)