

Introduction au Machine Learning



Séance 3 — Un premier modèle de bout en bout : l'arbre de décision

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/03-premiers-modeles.html>

Version du 11 octobre 2026



Version en ligne

Introduction au ML — Séance 3

Un premier modèle de bout en bout : l'arbre de décision

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

Cette séance construit un premier modèle de Machine Learning complet : formulation du problème, entraînement, évaluation et diagnostic d'un **arbre de décision** sur les données DataSANTÉ-221.

- Formuler un problème d'**apprentissage supervisé** : la fonction f , la matrice X , la cible y .
- Le protocole **train/test** et la mesure de la performance (accuracy, risque empirique).
- Le fonctionnement de l'arbre : régions, coupures, gain d'information.
- Le diagnostic central du ML : **sous-apprentissage** et **sur-apprentissage** ; le compromis **biais-variance**.

Les deux premières séances de ce cours sont couvertes par le cours de Programmation Python, disponible sur ce site.

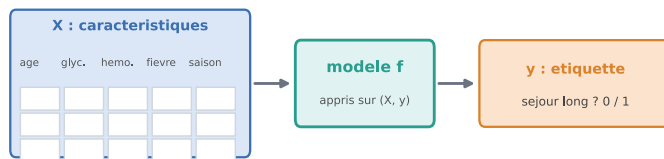
Apprendre à partir d'exemples

Le problème

La question médicale

À l'admission d'un patient, peut-on prédire si son séjour hospitalier sera **long** (au-dessus de la durée médiane) à partir de cinq mesures : âge, glycémie, hémoglobine, fièvre, saison ?

Apprentissage supervisé : trouver f telle que $f(X) \sim y$



Le registre contient 10 000 patients pour lesquels la réponse est *connue*. L'objectif est d'en extraire une règle de prédiction applicable aux patients futurs.

L'apprentissage supervisé, formellement

Définition

L'**apprentissage supervisé** consiste à construire, à partir d'exemples étiquetés (x_i, y_i) , une fonction de prédiction qui associe à toute entrée une sortie.

$$f : \mathcal{X} \longrightarrow \mathcal{Y}, \quad f(x) \approx y$$

$\mathcal{X}$ est l'espace des entrées (ici, les mesures d'un patient) et $\mathcal{Y}$ l'espace des sorties (ici, $\{0, 1\}$: séjour court ou long). Le modèle est la fonction f ; l'apprentissage consiste à la choisir à partir des données.

Les données : la matrice de design X

Organisation des données

Les n exemples s'empilent dans une matrice $X \in \mathbb{R}^{n \times d}$ — une **ligne** par exemple, une **colonne** par caractéristique — et leurs réponses dans le vecteur y . Le couple (X, y) est le jeu d'apprentissage.

Premières lignes réelles de DataSANTÉ-221

	âge	glycémie	hémoglobine	fièvre	saison	séjour long
x_1	47	9,3	11,8	38,0	0	$y_1 = 0$
x_2	18	5,1	13,0	39,5	1	$y_2 = 0$
x_3	56	7,2	9,5	37,1	1	$y_3 = 1$

Les mêmes données, en notation matricielle

X et *y* de DataSANTÉ-221

$$X = \begin{pmatrix} 47 & 9,3 & 11,8 & 38,0 & 0 \\ 18 & 5,1 & 13,0 & 39,5 & 1 \\ 56 & 7,2 & 9,5 & 37,1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \in \mathbb{R}^{10000 \times 5} \quad y = \begin{pmatrix} 0 \\ 0 \\ 1 \\ \vdots \end{pmatrix} \in \{0, 1\}^{10000}$$

Entraîner et évaluer

Le protocole scikit-learn : fit, predict, score

```
from sklearn.tree import DecisionTreeClassifier

modele = DecisionTreeClassifier(max_depth=4, random_state=0)
modele.fit(X_train, y_train)      # apprendre f a partir des exemples
y_pred = modele.predict(X_test)  # appliquer f a de nouveaux patients
modele.score(X_test, y_test)     # mesurer la qualite des predictions
```

Ce triptyque `fit` / `predict` / `score` est identique pour tous les modèles de `scikit-learn` ; seule la classe instanciée change.

Mesurer la qualité : perte et risque empirique

La perte 0/1

Pour une prédiction donnée, la **perte** vaut 0 si elle est correcte, 1 sinon. Le **risque empirique** est la moyenne des pertes sur un jeu de données.

$$\widehat{R}(f) = \frac{1}{n} \sum_{i=1}^n \mathbf{1}[f(x_i) \neq y_i] \quad \text{accuracy} = 1 - \widehat{R}(f)$$

La somme compte les erreurs ; la division par n en donne la proportion.
L'accuracy est la proportion complémentaire : celle des prédictions correctes.

Exemple — comparer y et $\hat{y}$

Exemple numérique

Cinq patients du test : comparaison composante par composante du vecteur des réponses y et du vecteur des prédictions $\hat{y} = f(X_{\text{test}})$.

patient	1	2	3	4	5
réponse y_i	1	0	1	1	0
prédiction $\hat{y}_i$	1	0	0	1	1
correcte ?	oui	oui	non	oui	non

Trois prédictions correctes sur cinq : $\text{accuracy} = \frac{3}{5} = 0,60$; $\text{risque} = \frac{2}{5} = 0,40$.

La mesure repose toujours sur cette comparaison des deux vecteurs — c'est elle que **score** automatise.

Exercice — calcul d'une accuracy

Énoncé

Sur 60 patients du jeu de test, le modèle commet 15 erreurs. Calculer l'accuracy et le risque empirique.

Correction

Prédictions correctes = $60 - 15 = 45$, d'où $\text{accuracy} = \frac{45}{60} = 0,75$ et $\text{risque} = \frac{15}{60} = 0,25$. Les deux quantités sont complémentaires : $\text{accuracy} = 1 - \widehat{R}$.

Le protocole train/test

Pourquoi séparer les données

La performance qui importe est celle sur des patients **jamais vus** : la **généralisation**. On réserve donc une fraction des données (le **test**, ici 20 %), exclue de l'apprentissage, pour estimer cette performance honnêtement.

train_test_split : estimer la generalisation sur un examen tenu a l'ecart



Découpage *aléatoire* et *stratifié* : 8000 patients pour apprendre, 2000 pour évaluer. Un score mesuré sur le jeu d'entraînement lui-même serait systématiquement optimiste.

Exercice — dimensions et découpage

Énoncé

DataSANTÉ-221 : $n = 10\,000$ patients, $d = 5$ caractéristiques. **(a)** Quelle est la taille de la matrice X ? **(b)** Si l'on choisissait plutôt `test_size=0.25`, combien de patients iraient dans le jeu d'entraînement et dans le jeu de test ?

Correction

(a) $X \in \mathbb{R}^{10000 \times 5}$: une ligne par patient, une colonne par caractéristique. **(b)** Test = 25 % de 10 000 = **2500** ; entraînement = **7500**.

La référence minimale : la baseline

Définition

La **baseline** est le prédicteur trivial qui répond toujours la classe majoritaire. Tout modèle digne d'intérêt doit la dépasser ; un score ne s'interprète que *relativement* à elle.

Exemple numérique

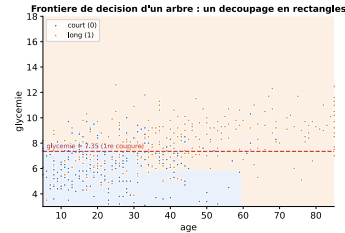
Sur DataSANTÉ-221, les classes sont quasi équilibrées (5045 contre 4955) : la baseline atteint **0,5045**. L'arbre de profondeur 4 obtient 0,7655 sur le test — soit 26 points au-dessus de la baseline : le modèle a réellement appris.

L'arbre de décision

Le principe : une suite de questions

Analogie

Aux urgences, le médecin de tri ne résout pas d'équation : il pose une question, puis une autre — « la glycémie dépasse-t-elle 7 ? », « plus de 60 ans ? » — et chaque réponse rétrécit le champ des possibles, jusqu'au pronostic. L'arbre de décision reproduit ce geste, à une différence près : l'ordre des questions et la valeur des seuils ne viennent pas de l'expérience du praticien — ils sont *calculés à partir des données*.



Chaque question trace une frontière verticale ou horizontale : l'arbre découpe le plan en **rectangles** de décision.

L'arbre, formellement

Modèle

Un arbre partitionne l'espace des entrées en M régions disjointes $R_1, \dots, R_M$ (les feuilles) et prédit une valeur constante dans chacune.

$$f(x) = \sum_{m=1}^M \hat{c}_m \mathbf{1}[x \in R_m]$$

L'indicatrice $\mathbf{1}[\cdot]$ sélectionne la région contenant x ; $\hat{c}_m$ est la classe majoritaire des exemples d'entraînement tombés dans R_m . Chaque nœud interne applique une **coupure** (j, s) : « la caractéristique j dépasse-t-elle le seuil s ? »

Exemple — évaluer $f(x)$ terme à terme

Un arbre à trois régions

Un petit arbre découpe le plan (glycémie, âge) en $M = 3$ régions, avec leurs classes majoritaires :

$$R_1 = \{\text{glycémie} \leq 7\}, \hat{c}_1 = 0;$$

$$R_2 = \{\text{glycémie} > 7, \text{âge} \leq 50\}, \hat{c}_2 = 0;$$

$$R_3 = \{\text{glycémie} > 7, \text{âge} > 50\}, \hat{c}_3 = 1.$$

Patient $x = (\text{glycémie } 9,0, \text{âge } 62)$: il n'appartient qu'à R_3 , donc $\mathbf{1}[x \in R_1] = 0, \mathbf{1}[x \in R_2] = 0, \mathbf{1}[x \in R_3] = 1$.

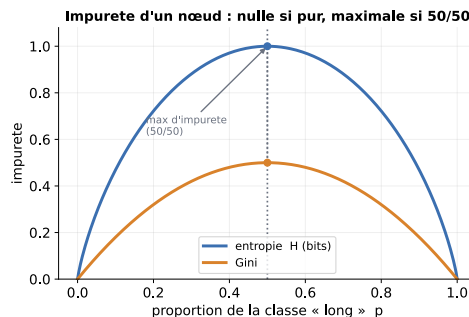
$$f(x) = \hat{c}_1 \times 0 + \hat{c}_2 \times 0 + \hat{c}_3 \times 1 = 0 + 0 + 1 = \mathbf{1}(\text{séjour long}).$$

Les régions étant disjointes, une seule indicatrice vaut 1 : la somme *sélectionne* la prédiction de la région où tombe x . Autre patient, $x' = (5,2, 70)$: indicatrices $(1, 0, 0)$, d'où $f(x') = \hat{c}_1 = 0$.

La pureté d'un nœud : l'idée

Les chiffres d'abord

Un groupe est **pur** lorsqu'il ne contient qu'une seule classe. L'**impureté** mesure son désordre : 0 pour un groupe pur, 0,88 bit pour un mélange 70/30, 1 bit pour un mélange 50/50 — le désordre maximal.



L'**entropie** H et l'**indice de Gini** G selon la proportion p : nulles aux extrêmes (pureté), maximales à $p = 0,5$.

L'entropie, formellement

L'idée en une phrase

L'entropie mesure le **désordre** d'un nœud — la surprise moyenne de son contenu : nulle quand l'issue est jouée d'avance (nœud pur), maximale quand les deux classes sont à égalité.

$$H(p) = -p \log_2 p - (1 - p) \log_2 (1 - p)$$

p est la proportion d'une classe dans le nœud ; l'unité est le **bit**. Rappel : $\log_2 x$ est l'exposant à donner à 2 pour obtenir x ($\log_2 4 = 2$, $\log_2 0,25 = -2$) — négatif pour $0 < x < 1$, d'où les signes moins.

Exemple numérique

Nœud à 25 % de séjours longs ($p = 0,25$) : $H = -0,25 \log_2 0,25 - 0,75 \log_2 0,75 = 0,25 \times 2 + 0,75 \times 0,415 = 0,5 + 0,311 = \mathbf{0,811}$ bit.

Exercice — calcul d'une entropie

Énoncé

Un nœud contient 90 % de séjours longs ($p = 0,9$). Calculer son entropie, sachant que $\log_2 0,9 = -0,152$ et $\log_2 0,1 = -3,322$. Commenter le résultat.

Correction

$H = -0,9 \times (-0,152) - 0,1 \times (-3,322) = 0,137 + 0,332 = 0,469$ bit. Le nœud est déjà bien ordonné (90/10) : son impureté est moins de la moitié du maximum — il reste peu de désordre à réduire.

L'indice de Gini

L'idée en une phrase

L'indice de Gini mesure le même désordre par une question simple : si l'on classait un patient **au hasard**, selon les proportions du nœud, quelle serait la probabilité de se tromper ?

$$G(p) = 1 - p^2 - (1 - p)^2 = 2p(1 - p)$$

Maximum 0,5 lorsque les classes sont à égalité ($p = 0,5$) ; nul si le nœud est pur — il est alors impossible de se tromper.

Exemple numérique

$$G(0,5) = 2 \times 0,5 \times 0,5 = \mathbf{0,5} ; G(0,25) = 2 \times 0,25 \times 0,75 = \mathbf{0,375} ; G(1) = \mathbf{0}.$$

Entropie et Gini classent les coupures presque toujours dans le *même ordre* ; Gini, sans logarithme, est moins coûteux — c'est le critère par défaut de [scikit-learn](#). La suite utilise l'entropie, plus interprétable (en bits).

La coupure et le gain d'information

Le principe de sélection

Une **coupure** (j, s) scinde un nœud selon « la caractéristique j dépasse-t-elle le seuil s ? ».
Une bonne coupure produit des enfants *plus purs* que le parent : le **gain** mesure cette baisse d'impureté.

$$\text{Gain}(j, s) = H_{\text{parent}} - (\pi_g H_g + \pi_d H_d) \quad (j^*, s^*) = \arg \max_{(j, s)} \text{Gain}(j, s)$$

π_g et π_d sont les proportions d'exemples envoyées à gauche et à droite :
l'impureté des enfants est une *moyenne pondérée*. L'algorithme évalue **toutes** les coupures possibles et retient celle du gain maximal — puis recommence dans chaque enfant, récursivement.

Construire un arbre à la main — les huit patients

Un exemple de bout en bout

Pour observer l'algorithme à l'œuvre, on construit *intégralement* un arbre sur huit patients et deux caractéristiques — une **ligne** par patient, comme dans la matrice X .

Les huit patients

patient	glycémie	fièvre	séjour long
x_1	5,0	37,3	$y_1 = 1$
x_2	6,0	37,0	$y_2 = 0$
x_3	6,5	37,6	$y_3 = 0$
x_4	7,0	38,2	$y_4 = 0$
x_5	8,0	37,9	$y_5 = 1$
x_6	9,0	38,5	$y_6 = 1$
x_7	10,0	39,1	$y_7 = 1$
x_8	12,0	38,8	$y_8 = 0$

La racine : quatre longs, quatre courts

Quatre séjours longs, quatre courts : la racine est au désordre maximal, $H_{\text{racine}} = 1,000$ bit ($G = 0,5$).

Étape 1 — le duel des coupures candidates

Trois candidates évaluées

A. glycémie ≤ 7 : gauche = $\{1, 0, 0, 0\}$, droite = $\{1, 1, 1, 0\}$ $\rightarrow$ gain = **0,189** bit (détail page suivante).

B. fièvre ≤ 38 : gauche = $\{1, 0, 0, 1\}$ ($p = 0,5, H = 1$), droite = $\{0, 1, 1, 0\}$ ($p = 0,5, H = 1$) $\rightarrow$ gain = $1 - \frac{4}{8} \cdot 1 - \frac{4}{8} \cdot 1 = \mathbf{0}$ bit.

C. glycémie $\leq 9,5$: gauche = $\{1, 0, 0, 0, 1, 1\}$ ($p = 0,5, H = 1$), droite = $\{1, 0\}$ ($p = 0,5, H = 1$) $\rightarrow$ gain = **0** bit.

La coupure B scinde sans trier : ses enfants restent au désordre maximal — un gain *nul* malgré la séparation. L'argmax (sur *toutes* les coupures des deux caractéristiques) retient **A : glycémie** ≤ 7 .

Étape 1 — le calcul détaillé du gain retenu

Coupure glycémie ≤ 7 pas à pas

Gauche (patients 1-4) : étiquettes $\{1, 0, 0, 0\}$, $p_g = \frac{1}{4} = 0,25$, $H_g = 0,811$ (calculé plus haut).

Droite (patients 5-8) : étiquettes $\{1, 1, 1, 0\}$, $p_d = 0,75$, $H_d = 0,811$ (symétrie de H).

$$\text{Gain} = 1,000 - \left(\frac{4}{8} \times 0,811 + \frac{4}{8} \times 0,811 \right) = 1,000 - 0,811 = 0,189 \text{ bit.}$$

Le même calcul avec Gini : $0,5 - 0,375 = 0,125$ — un autre chiffre, le *même* verdict. Les deux enfants restent impurs (0,811 bit) : la récursion continue.

Étape 2 — purifier le nœud gauche

Nœud gauche : glycémies 5 ; 6 ; 6{,}5 ; 7 — étiquettes 1 ; 0 ; 0 ; 0

Coupure **glycémie** $\leq 5,5$: gauche = {1} (pur, $H = 0$), droite = {0, 0, 0} (pur, $H = 0$).

Gain = $0,811 - \left(\frac{1}{4} \times 0 + \frac{3}{4} \times 0 \right) = 0,811$ bit — tout le désordre restant est absorbé.

Les deux enfants sont **purs** : ce sont des **feuilles**. La branche gauche est terminée — aucune coupure supplémentaire n'apporterait de gain.

Étape 2 — purifier le nœud droit

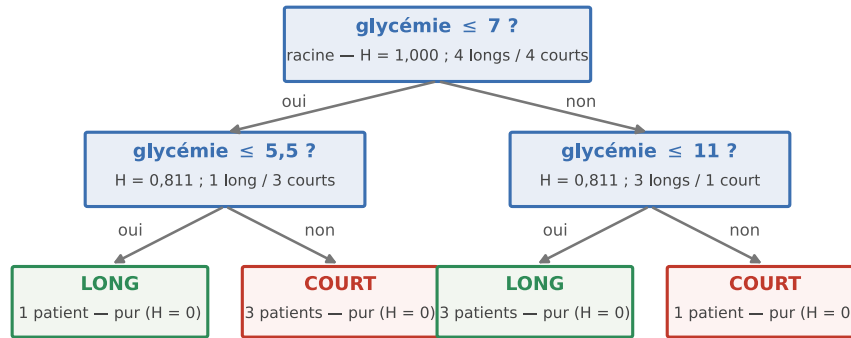
Nœud droit : glycémies 8 ; 9 ; 10 ; 12 — étiquettes 1 ; 1 ; 1 ; 0

Coupure **glycémie** ≤ 11 : gauche = $\{1, 1, 1\}$ (pur), droite = $\{0\}$ (pur).

Gain = $0,811 - 0 = 0,811$ bit. Deux feuilles pures : la branche droite est terminée.

L'algorithme s'arrête : toutes les feuilles sont pures. L'arbre complet a été obtenu par trois applications du *même* principe — l'argmax du gain, appliqué récursivement.

L'arbre final, et sa vérification



`DecisionTreeClassifier(criterion="entropy")` entraîné sur ces huit patients reproduit **exactement** cet arbre (seuils aux points médians : 7,5 ; 5,5 ; 11) — score 8/8. Feuilles pures sur huit patients : l'arbre a *mémorisé* l'échantillon ; ce constat annonce la question du sur-apprentissage, traitée plus loin.

Exercice — un gain à la main

Énoncé

Sur le nœud droit (étiquettes $\{1, 1, 1, 0\}$, $H = 0,811$), calculer le gain de la coupure **glycémie** $\leq 9,5$ (gauche = $\{1, 1\}$, droite = $\{1, 0\}$) et le comparer au gain de la coupure retenue (0,811).

Correction

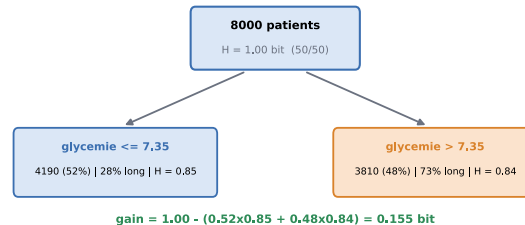
$H_g = H(1) = 0$ (pur) ; $H_d = H(0,5) = 1$. Gain = $0,811 - \left(\frac{2}{4} \times 0 + \frac{2}{4} \times 1 \right) = 0,811 - 0,5 = 0,311$ bit — inférieur à 0,811 : l'argmax écarte cette candidate, qui laisse un enfant au désordre maximal.

La première coupure réelle

Exemple numérique

Sur les 8000 patients d'entraînement, la coupure optimale trouvée par l'algorithme est **glycémie $\leq 7,35$** , avec un gain de **0,155 bit** — la meilleure parmi toutes les coupures possibles sur les cinq caractéristiques.

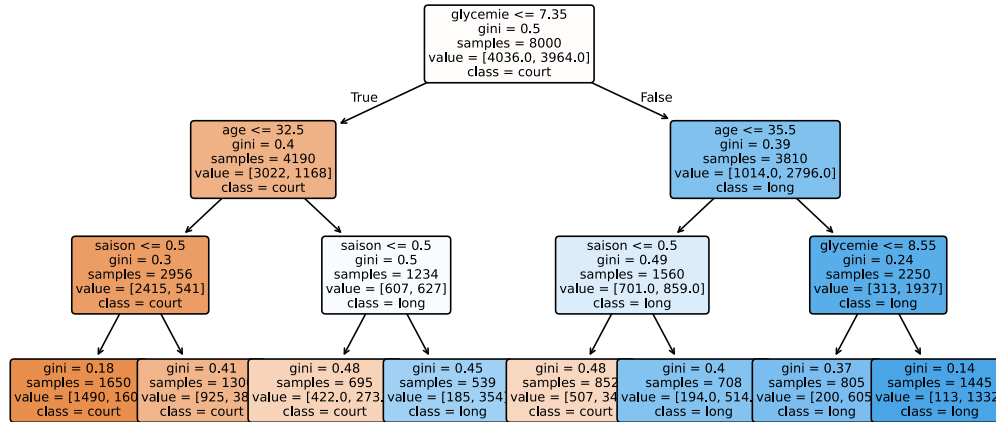
Gain d'information : la coupure qui rend les groupes plus purs



À gauche de la coupure, les séjours courts dominent ; à droite, les séjours longs. Même principe que sur les huit patients — appliqué à 8000. Au TP : ce gain de 0,155 bit est recalculé à la main, ainsi que la construction complète de l'arbre jouet.

Lire une prédiction : du chemin à la feuille

L'arbre appris (profondeur 3) : seuils, impurete (gini), repartition



Pour prédire, on suit le chemin des réponses de la racine jusqu'à une feuille.
Exemple : un patient de 60 ans, glycémie 9,0 — la branche droite conduit à une feuille à 95 % « long », d'où $f(x) = 1$ avec une confiance de 0,95.

Exercice — suivre un chemin

Énoncé

Arbre simplifié : racine « glycémie $\leq 7,35$? » — **oui** → feuille à 88 % « court » ; **non** → « âge ≤ 50 ? » — **oui** → feuille à 60 % « long », **non** → feuille à 95 % « long ». Déterminer la prédiction et la confiance pour un patient de 45 ans dont la glycémie vaut 8,2.

Correction

$8,2 > 7,35 \rightarrow$ branche droite ; $45 \leq 50 \rightarrow$ feuille « long » à 60 %. Prédiction $f(x) = 1$, confiance **0,60** — une prédiction peu assurée : la feuille est presque partagée.

Sous-apprentissage et sur-apprentissage

La profondeur contrôle la complexité

Un réglage central

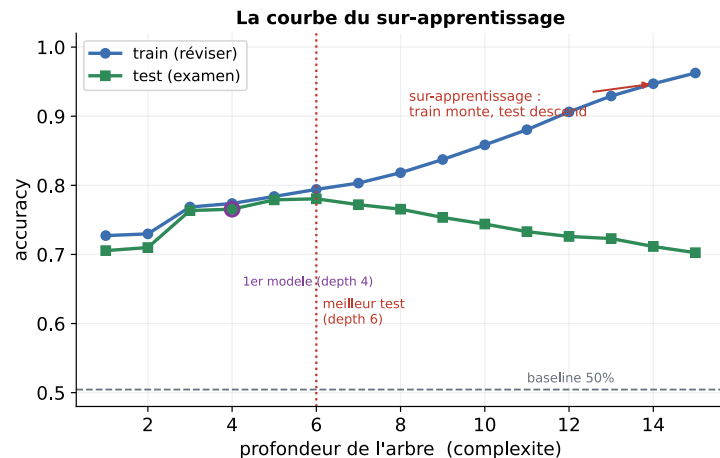
La profondeur maximale (`max_depth`) borne le nombre de questions successives. Elle détermine la **complexité** du modèle — et son comportement face aux données.

Exemple numérique

	score train	score test
profondeur 4	0,7738	0,7655
profondeur illimitée	0,9999	0,6995

L'arbre illimité mémorise presque parfaitement le jeu d'entraînement — et prédit *moins bien* les patients nouveaux. Mémoriser n'est pas généraliser.

La courbe de complexité



Le score d'entraînement croît avec la profondeur ; le score de test culmine puis **redescend**. À gauche du sommet, le modèle est trop simple (**sous-apprentissage**) ; à droite, il apprend les particularités du jeu d'entraînement, bruit compris (**sur-apprentissage**).

Exercice — diagnostiquer un modèle

Énoncé

Trois modèles, trois couples (score train, score test) : **A** (0,71; 0,70) ; **B** (0,78; 0,76) ; **C** (0,99; 0,68). Diagnostiquer chacun.

Correction

A : scores faibles et proches — **sous-apprentissage** (modèle trop simple). **B** : scores corrects et proches — bon compromis. **C** : écart majeur entre train et test — **sur-apprentissage** caractérisé. Le diagnostic repose toujours sur la *paire* de scores, jamais sur un seul.

Le compromis biais-variance

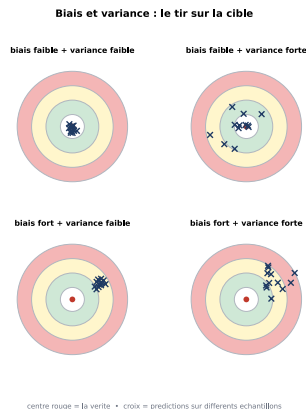
Deux sources d'erreur

Analogie

Un tailleur doit livrer un boubou à un client, après plusieurs essayages. Premier excès : il coud sur un **patron unique**, taille moyenne, sans toucher aux mesures — quel que soit l'essayage, la manche est trop courte des *mêmes* trois centimètres. L'erreur est systématique et se répète à l'identique : c'est le **biais**, l'excès de rigidité. Second excès, sur le *même* boubou : à chaque essayage, il reprend toute la coupe pour épouser les mesures du jour — y compris le pull épais que le client portait ce jour-là. Chaque version diffère de la précédente : c'est la **variance**, l'excès de sensibilité aux données du jour.

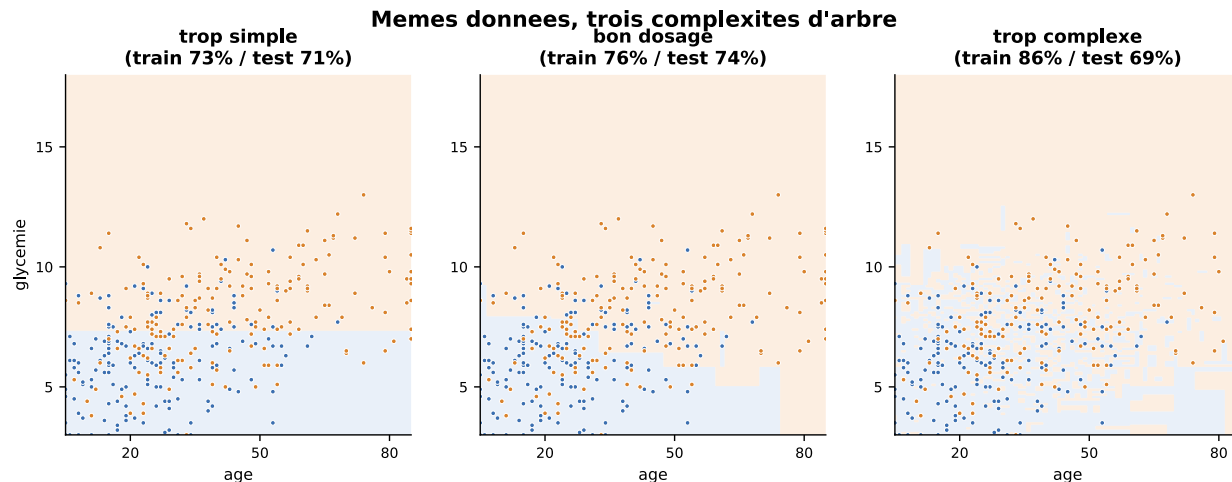
Le sous-apprentissage est un excès de biais ; le sur-apprentissage, un excès de variance. Toute la pratique du ML consiste à doser entre les deux.

La cible : visualiser biais et variance



Chaque impact est un modèle entraîné sur un échantillon différent. **Biais** : impacts décalés du centre, systématiquement — le boubou au patron unique, toujours faux de la même manière. **Variance** : impacts dispersés — le boubou recousu à chaque essayage, différent à chaque fois. Le cas favorable — groupés au centre — exige de réduire les deux.

Trois ajustements, une même donnée



Profondeur 1 : frontière trop grossière, biais dominant (test 0,71). Profondeur 4 : compromis (test 0,77). Profondeur illimitée : frontière déchiquetée épousant chaque point, variance dominante (test 0,70).

La décomposition de l'erreur

$$\mathbb{E}[\text{erreur}] = \underbrace{\text{biais}^2}_{\text{rigidité}} + \underbrace{\text{variance}}_{\text{instabilité}} + \underbrace{\sigma^2}_{\text{bruit irréductible}}$$

L'erreur attendue d'un modèle se décompose en trois termes : l'erreur systématique (biais), la sensibilité à l'échantillon (variance), et le bruit propre aux données, qu'aucun modèle ne peut éliminer. Augmenter la complexité réduit le biais mais accroît la variance — d'où l'existence d'un optimum intermédiaire.

Erreur fréquente

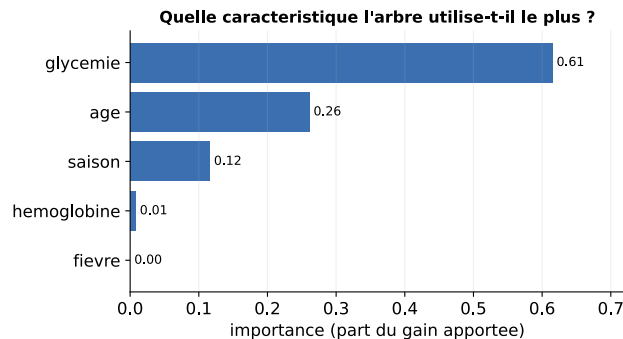
Viser un score d'entraînement parfait. Le terme de bruit σ^2 rend cet objectif illusoire : un score train de 100 % signale presque toujours que la variance a explosé.

Interpréter le modèle

L'importance des caractéristiques

Définition

L'**importance** d'une caractéristique est la part du gain total d'impureté apportée par les coupures qui l'utilisent. Les importances sont normalisées : leur somme vaut 1.



Sur DataSANTÉ-221 : glycémie 0,615, âge 0,261, saison 0,116, hémoglobine 0,008, fièvre 0,000. La glycémie domine ; la fièvre n'est utilisée par aucune coupure.

Exercice — lire des importances

Énoncé

Importances de l'arbre : glycémie 0,615, âge 0,261, saison 0,116, hémoglobine 0,008, fièvre 0,000. **(a)** Vérifier leur somme. **(b)** Que conclure du 0,000 de la fièvre ?

Correction

(a) $0,615 + 0,261 + 0,116 + 0,008 + 0,000 = 1,000$: les importances sont des parts du gain total. **(b)** Aucune coupure n'utilise la fièvre : *pour cette cible*, elle n'apporte aucun gain. Une variable peut être médicalement pertinente et inutile pour un problème de prédiction donné.

Synthèse

Le protocole complet d'une étude

De bout en bout

1. Formuler le problème : (X, y) , espaces d'entrée et de sortie. ; **2.** Découper : `train_test_split`, le test mis de côté. ; **3.** Entraîner : `fit` sur le train. ; **4.** Évaluer : `score` sur le test, comparé à la **baseline**. ; **5.** Diagnostiquer : paire (train, test) — biais ou variance. ; **6.** Interpréter : importances, lecture de l'arbre.

Ce protocole reste valable pour tous les modèles des séances suivantes ; seule l'étape 3 change de classe `scikit-learn`.

À retenir — Séance 3

L'essentiel

- Un modèle est une fonction $f : \mathcal{X} \rightarrow \mathcal{Y}$ apprise des données ; celles-ci s'organisent en $X \in \mathbb{R}^{n \times d}$ et y .
- La performance se mesure sur des données **jamais vues** (test) et se compare à la **baseline**.
- L'arbre partitionne l'espace en régions par coupures successives, choisies par **gain d'impureté**.
- Le couple (score train, score test) diagnostique **sous-** et **sur-apprentissage**.
- L'erreur se décompose en **biais**, **variance** et bruit ; la complexité arbitre entre les deux premiers.

Pièges fréquents

- Évaluer un modèle sur ses propres données d'entraînement : score systématiquement optimiste.
- Annoncer une accuracy sans la comparer à la **baseline**.
- Viser un score d'entraînement parfait : signature du sur-apprentissage, pas de la qualité.
- Conclure qu'une variable « ne sert à rien » dans l'absolu parce que son importance est nulle *pour cette cible*.
- Diagnostiquer sur un seul score : seul le *couple* (train, test) est informatif.

Prochaine séance

Séance 4 — Des données propres aux distances

Les données réelles comportent des valeurs manquantes, des catégories et des échelles hétérogènes. La Séance 4 introduit leur préparation rigoureuse (**Pipeline**, fuite de données) et un second modèle, les k **plus proches voisins**, pour lequel la mise à l'échelle est une condition de validité.

Références

- G. James, D. Witten, T. Hastie, R. Tibshirani, *An Introduction to Statistical Learning*, 2^e éd., Springer, 2021.
- A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*, 3^e éd., O'Reilly, 2022.
- A. C. Müller, S. Guido, *Introduction to Machine Learning with Python*, O'Reilly, 2016.
- Cours : UW CSE 446 ; Stanford CS229.

Ressources de la séance

- **Notebook** · [TP 3 — Premier modèle](#) (347 Ko)
- **TD** · [Fiche d'exercices 3](#) (228 Ko)