Data Science

M2 Actuariat

Session 3 : Arbre de décision et méthodes ensemblistes

François HU

Responsable du pôle Intelligence Artificielle, Milliman R&D

Enseignant ISFA

2025

ISFA
Institut de Science Financière et d'Assurances
GRANDE ÉCOLE D'ACTUARIAT & GESTION DES RISQUES

Milliman

Sommaire du cours Data Science

Concept des méthodes ensemblistes

1. Arbre de décision
2. Introduction aux méthodes ensemblistes
3. Le bagging et les forêts aléatoires
4. Le boosting : apprendre de ses erreurs
5. Les champions du boosting (XGBoost, LightGBM, CatBoost)
6. Synthèse et bonnes pratiques
7. Conclusion

1. Arbre de décision

L'algorithme **CART** (*Classification And Regression Trees*) est un **arbre de décision** utilisé pour des tâches de **classification** ou de **régression**.

CART

Arbre de classification

Arbre de régression

Hyperparamètres

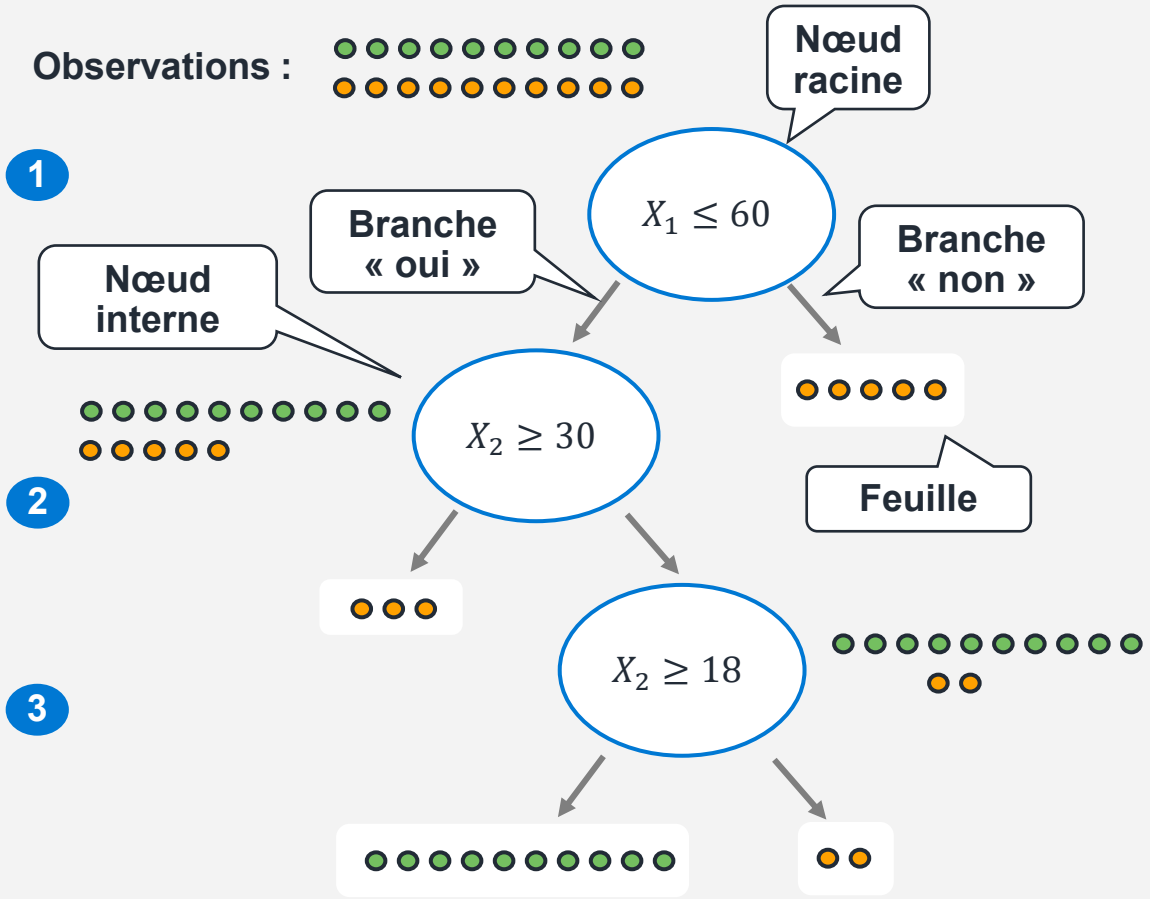
Avantages / Inconvénients

ISFA
Institut de Science Financière et d'Assurances

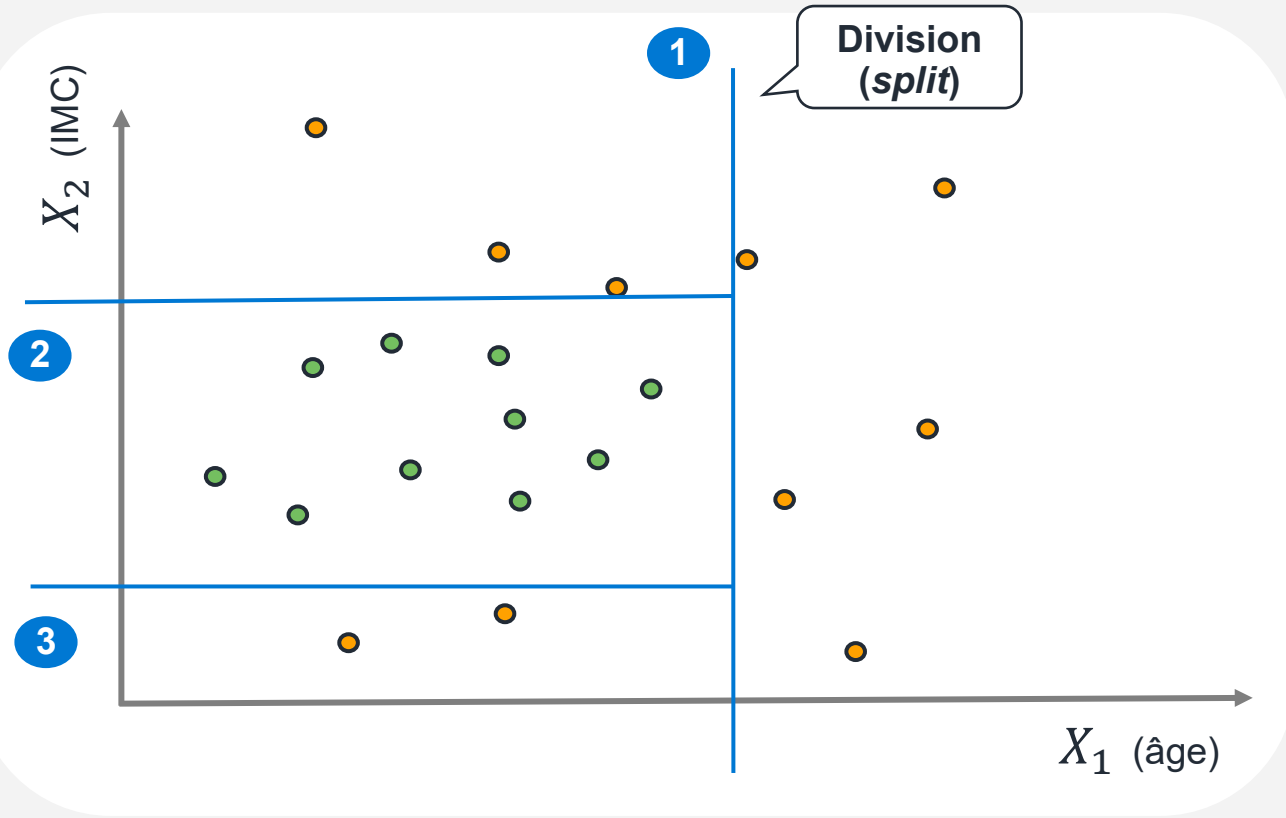
Exemple illustratif

Intuition d'un arbre de décision (pour la classification)

Modéliser une décision avec une série de questions

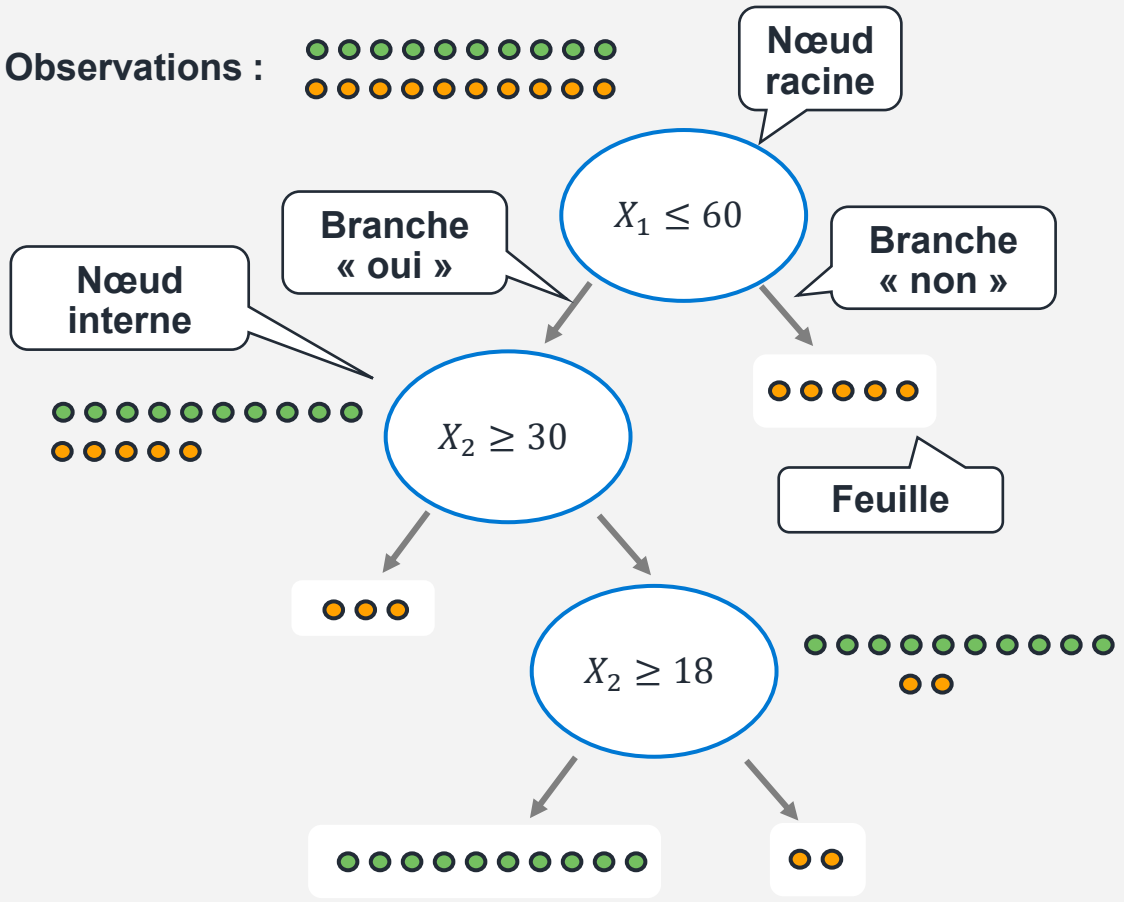


Variable cible : $Y = \{\circ, \circ\}$ (décès)



Comment construire un Arbre ?

Algorithme de Partitionnement Récursif



Algorithme de base (CART, Classification and Regression Trees) :

1. Commencer avec tous les points dans une seule région
2. Trouver le **meilleur découpage / division (split)** possible. Un découpage est défini par une variable j et un seuil s. **Comment définir le « meilleur » ?** C'est le découpage qui maximise la « pureté » des deux régions filles (nœuds enfants)
3. Appliquer ce découpage pour **créer deux nouvelles régions**
4. Répéter le processus (étapes 2-3) sur chaque nouvelle région
5. Arrêter le processus selon un critère (profondeur max, nombre min d'échantillons par feuille...).

Question : comment trouver la **meilleure division** ?

Les critères de division pour la classification (1/2)

Mesurer le désordre : l'impureté de Gini et l'entropie

Objectif : des feuilles « pures »

- **Chaque feuille de l'arbre doit idéalement contenir des éléments d'une seule classe (parmi C classes).**
- **Pour mesurer cette pureté, on utilise une fonction d'impureté.**

L'impureté : Gini vs. Entropie

Deux mesures classiques de l'impureté d'un nœud pour la **classification (binaire)** :

- **Indice de Gini** : Probabilité de mal classer un élément (p_i) choisi au hasard si on le classe en suivant la distribution de la feuille.

$$Gini = 2p_1(1 - p_1)$$

- **Entropie** : C'est une mesure de désordre ou d'incertitude. Plus les classes sont mélangées, plus l'entropie est élevée.

$$Entropie = -\sum_{i=1,\dots,C} p_i \log p_i$$

Gain d'information :

À chaque split candidat, l'algorithme mesure combien **l'impureté totale diminue** après la division :

Gain(split) =

$$Impureté\ avant\ split - Impureté\ moyenne\ après\ split$$

ISFA Institut de Science Financière et d'Assurances

Exemple illustratif

Les critères de division pour la classification (2/2)

Mesurer le désordre : l'impureté de Gini et l'entropie

Variable cible : $Y = \{\circ, \circ\}$ (décès)

Bon split

$$Gini(X_1 > t) = 0$$

$$Gini(X_1 \leq t) = 2 \times \frac{1}{7} \times \frac{6}{7} \approx 0.24$$

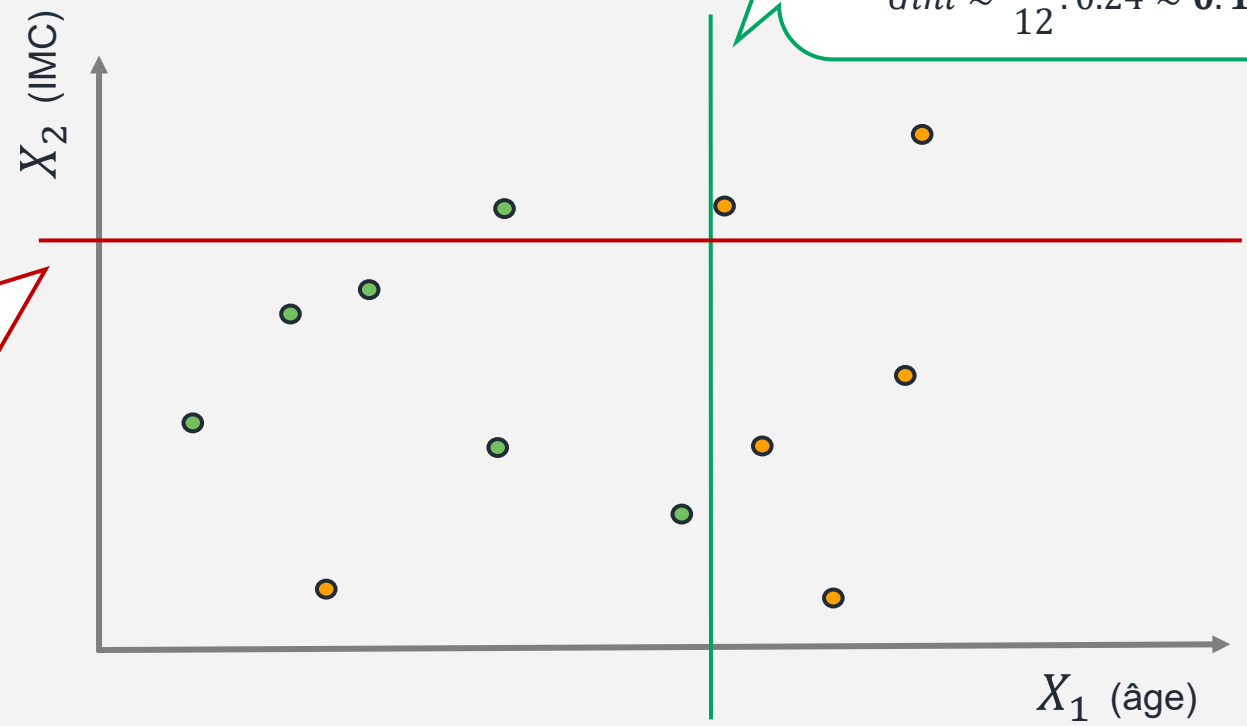
$$Gini \approx \frac{7}{12}.0.24 \approx \mathbf{0.14}$$

Mauvais split

$$Gini(X_2 > t) = 2 \times \frac{2}{3} \times \frac{1}{3} = \frac{4}{9}$$

$$Gini(X_2 \leq t) = 2 \times \frac{4}{9} \times \frac{5}{9} = \frac{40}{81}$$

$$Gini = \frac{3}{12}.\frac{4}{9} + \frac{9}{12}.\frac{40}{81} \approx \mathbf{0.48}$$



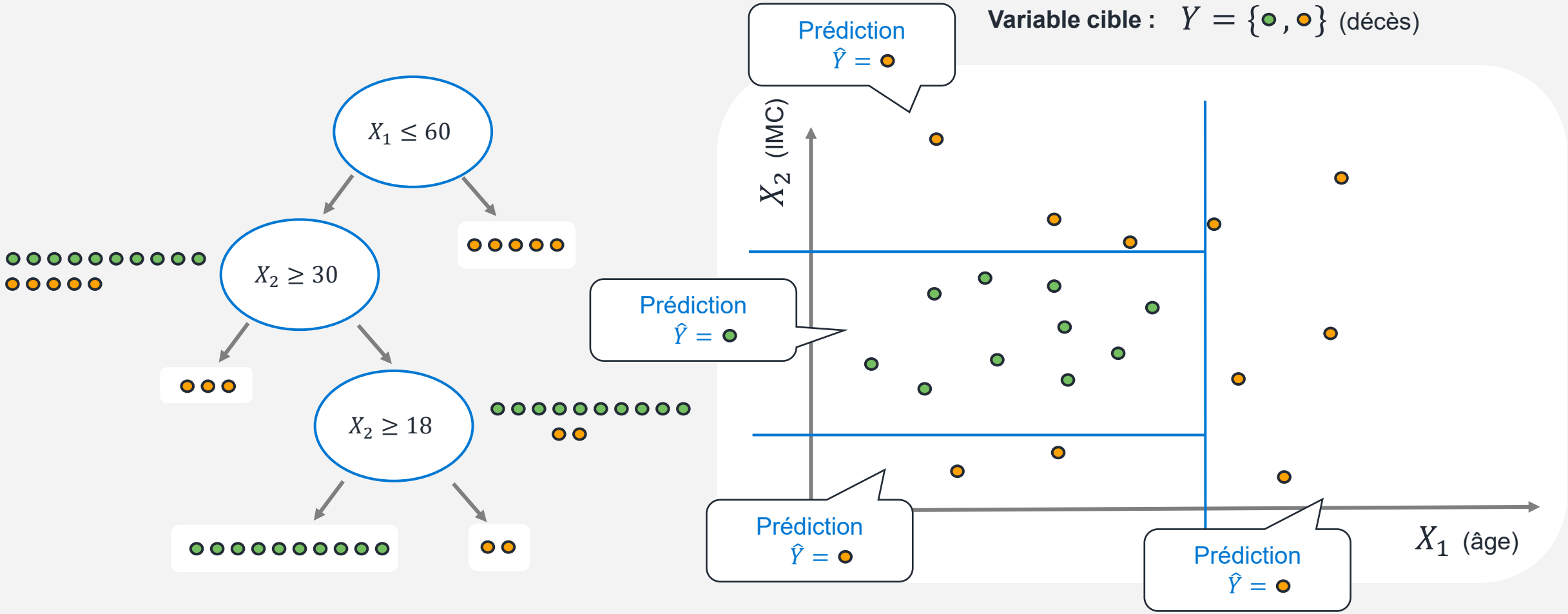
Arbre de classification

Prédiction correspondant à la classe majoritaire

Exemple illustratif

Observations :

Variable cible : $Y = \{\circ, \circ\}$ (décès)



Arbre de classification : La prédiction est la **classe majoritaire dans la feuille**.

Les critères de division pour la régression

Prédire une valeur continue, remplacer le concept d'impureté par celui de la **réduction de variance**

Un **arbre de régression** utilise la même structure hiérarchique mais pour prédire une **variable cible continue.**

Objectif : Créer des feuilles où les valeurs cibles sont les plus homogènes possible, c'est-à-dire qu'elles sont numériquement proches les unes des autres.

Critère de division

- Le critère de division n'est plus la pureté des classes, mais la **réduction de la variance** de la variable cible.
- l'algorithme cherche à chaque étape la division (variable + seuil) qui minimise **la somme des carrés des résidus (RSS)** des deux nœuds enfants.

$$\text{RSS} = \Sigma_{i \in gauche} \left(y_i - \bar{y}_{gauche}\right)^2 + \Sigma_{i \in droite} \left(y_i - \bar{y}_{droite}\right)^2$$

$\bar{y}$ *est la moyenne des* y *dans la région.*

- **Intuition :** Le meilleur split est celui qui crée deux sous-groupes où les valeurs sont le moins dispersées possible autour de leur nouvelle moyenne locale.

Prédiction finale (dans une feuille)

- La **moyenne** des valeurs cibles de tous les échantillons présents dans cette feuille.
- La **médiane** est parfois utilisée comme alternative pour être plus robuste aux valeurs aberrantes.

Arbre de régression : La prédiction est la **moyenne (ou médiane) des valeurs cibles dans la feuille**

Un panorama des algorithmes d'arbres de décision

Comprendre les distinctions pour mieux saisir le fonctionnement des outils modernes

- Le terme « **arbre de décision** » recouvre une famille d'algorithmes, chacun avec ses propres règles pour **choisir les divisions.**
- **Remarque : l**'implémentation de scikit-learn est basée sur une version optimisée de **CART**

Algorithme	Type de Cible	Variables Explicatives	Critère de Division	Structure de l'Arbre	Point Clé
ID3	Classification	Catégorielles	**Gain d'Information** (basé sur l'entropie)	Multi-branches	Le précurseur. Tendance à favoriser les variables avec beaucoup de catégories.
C4.5	Classification	Catégorielles & Continues	**Ratio de Gain** (normalise le gain d'information)	Multi-branches	Amélioration d'ID3, gère les valeurs manquantes et l'élagage.
CART	**Classification & Régression**	Catégorielles & Continues	• **Indice de Gini** (Classification) • **Réduction de Variance** (Régression)	**Strictement Binaire**	Le standard moderne, polyvalent et efficace, base des forêts aléatoires.
CHAID	Classification	Catégorielles	Test statistique du Khi-deux (χ^2)	Multi-branches	Approche statistique, cherche la division la plus statistiquement significative.

Synthèse comparative

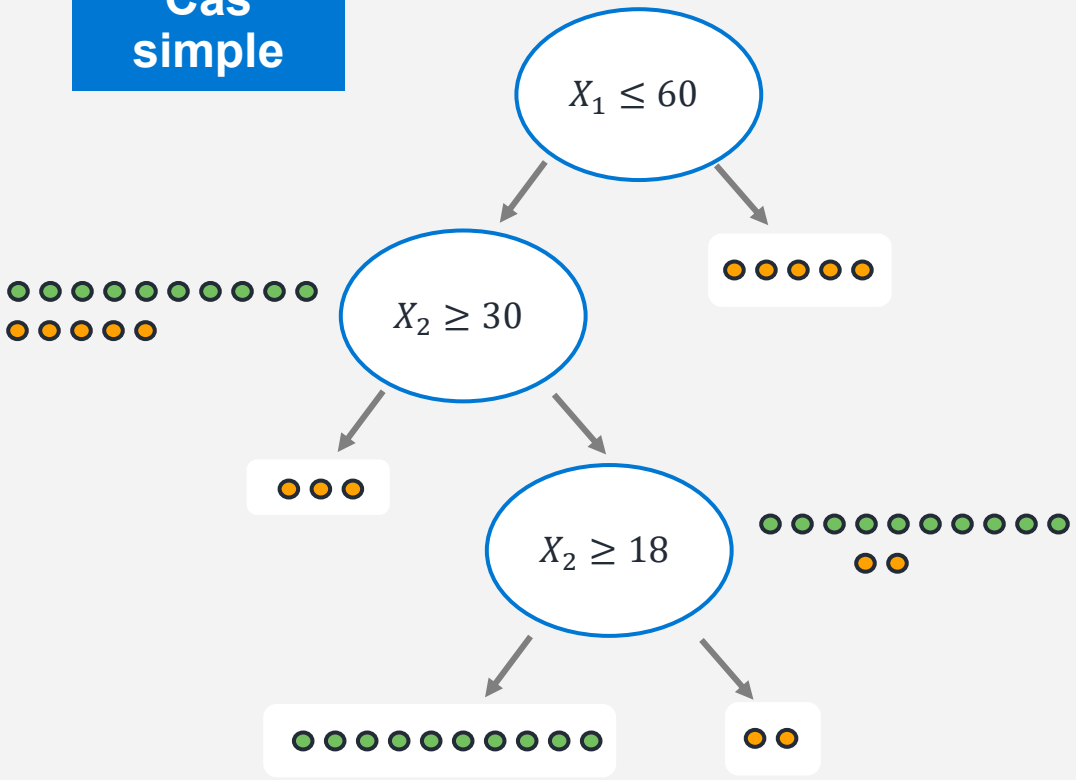
Zoom sur les algorithmes de CART

Caractéristique	Arbre de Classification	Arbre de Régression
Type de cible	Catégorielle (discrète)	Numérique (continue)
Critère d'optimisation	**Réduction de l'« impureté »** (Gini, Entropie)	**Réduction de la variance** (MSE)
Prédiction en feuille	Vote majoritaire (la classe la plus fréquente)	Moyenne (ou médiane) des valeurs cibles

Le danger principal : Le sur-apprentissage (1/2)

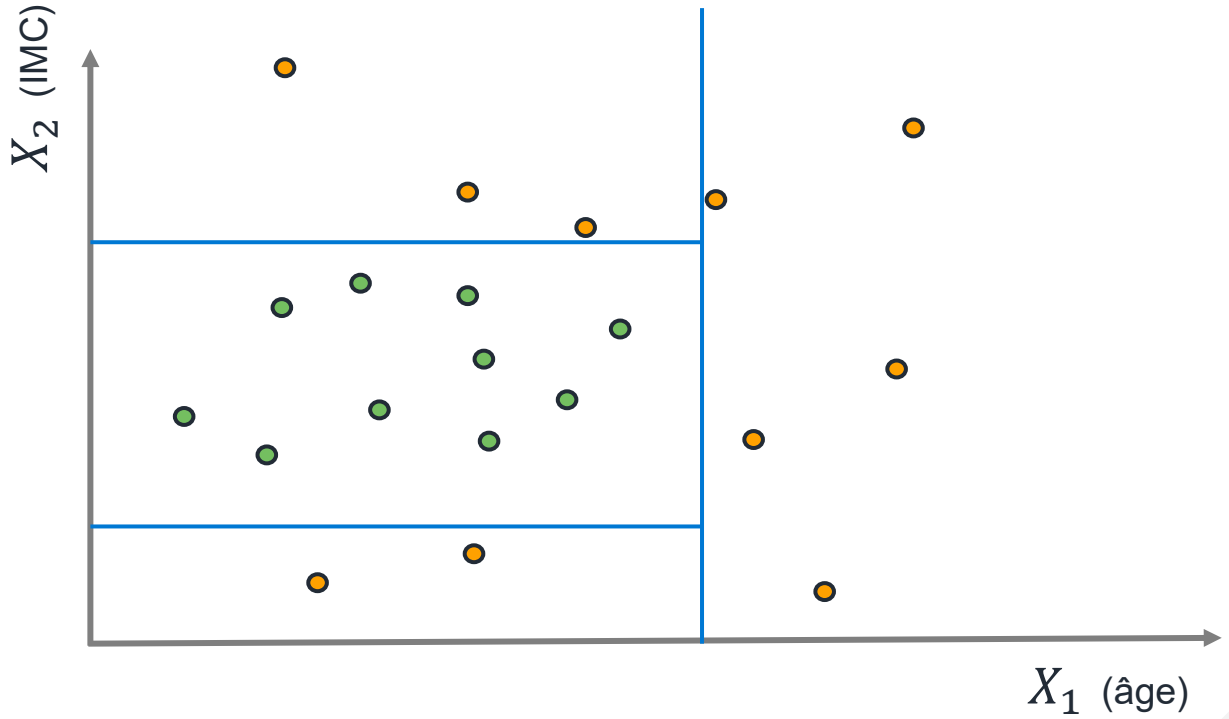
Prédiction correspondant à la classe majoritaire

Cas simple



Observations :

Variable cible : $Y = \{\circ, \circ\}$ (décès)



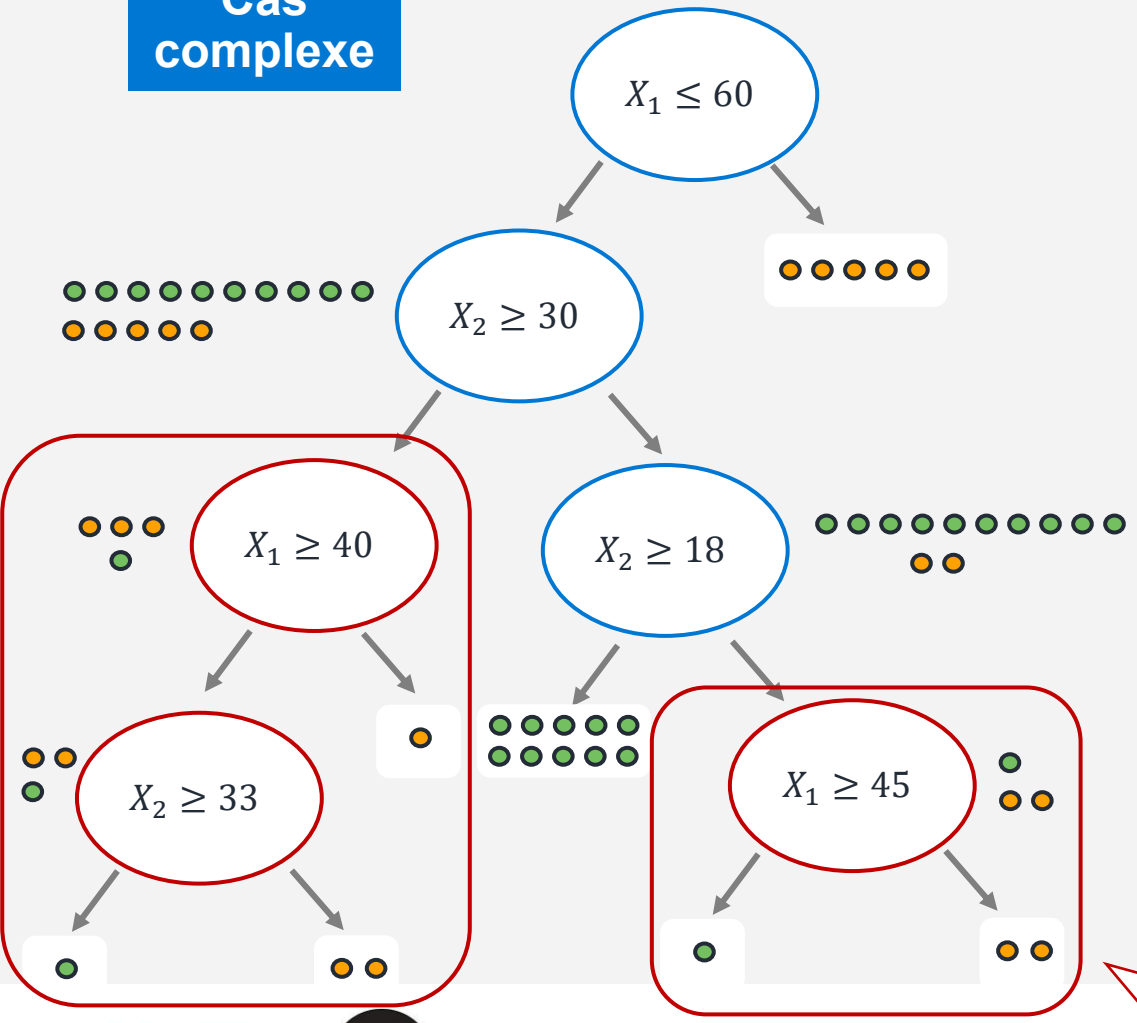
Le danger principal : Le sur-apprentissage (2/2)

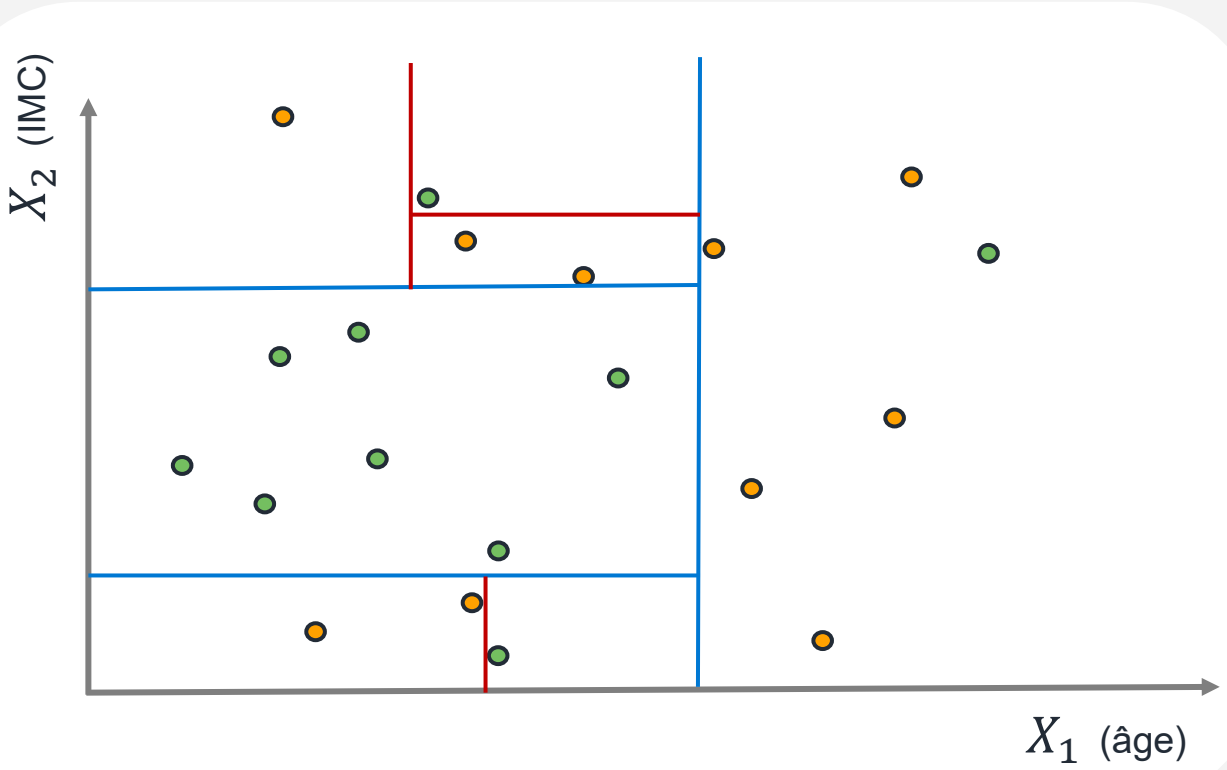
Prédiction correspondant à la classe majoritaire

Cas complexe

Observations :

Variable cible : $Y = \{\bullet, \bullet\}$ (décès)





Nœuds internes en plus avec une *profondeur max arbre (max_depth) : 3 -> 4*

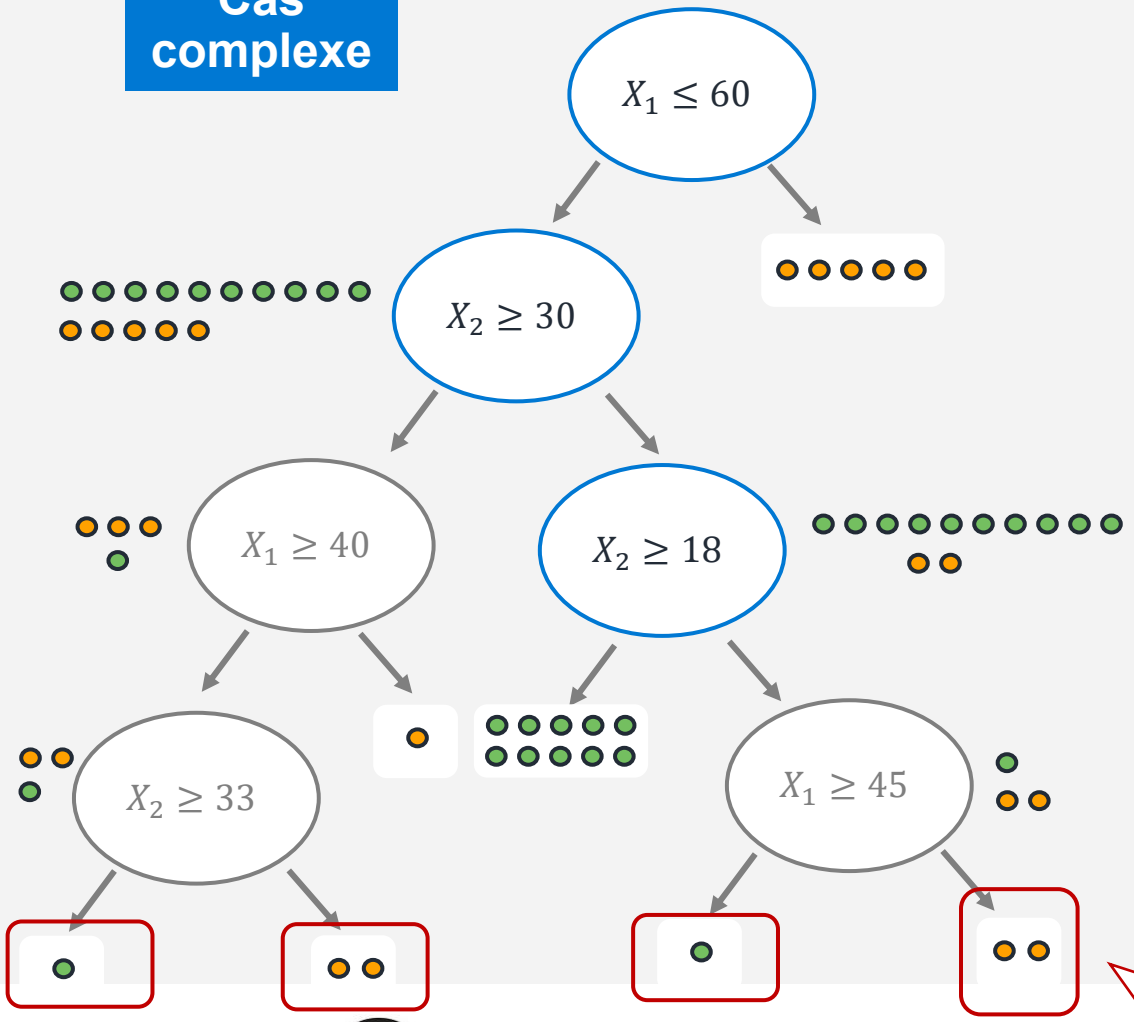
Sur-apprentissage : nous risquons de capturer les petites perturbations

ISFA
Institut de Science Financière et d'Assurances

Le danger principal : Le sur-apprentissage (2/2)

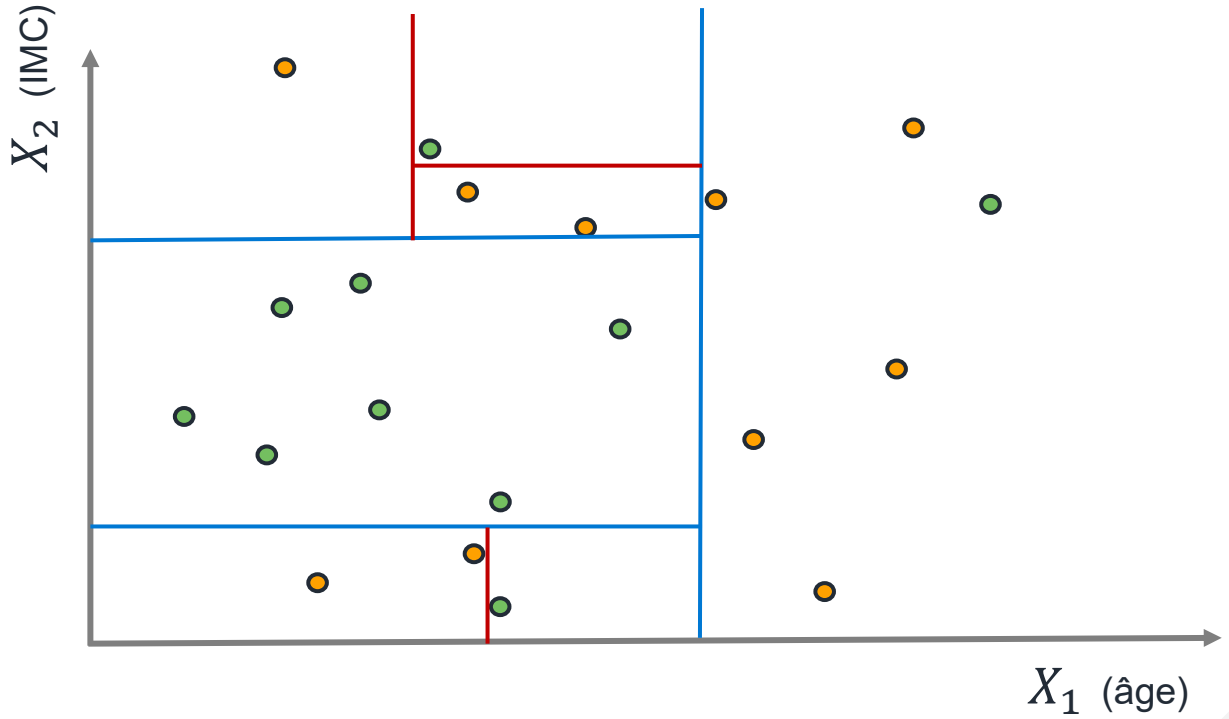
Prédiction correspondant à la classe majoritaire

Cas complexe



Observations :

Variable cible : $Y = \{\bullet, \bullet\}$ (décès)



Taille de l'échantillon dans une feuille *(min_samples_leaf)* **bien trop petite !**

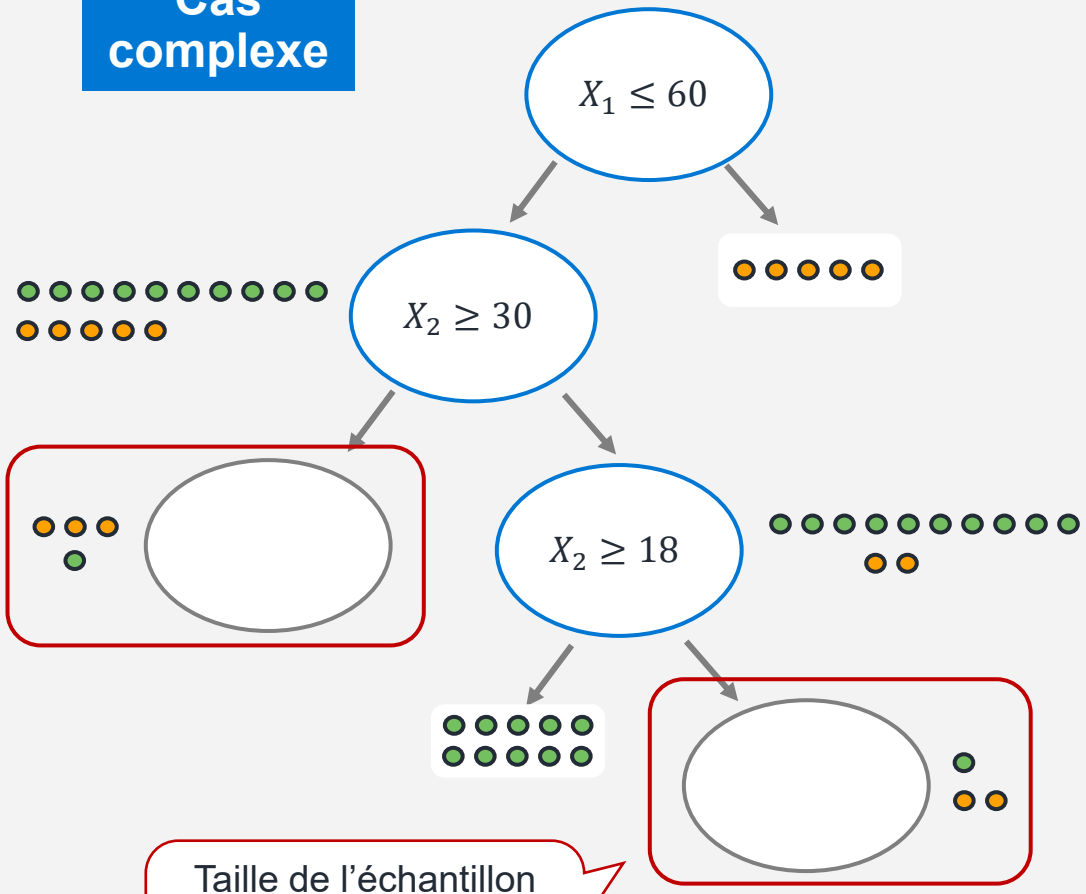
Sur-apprentissage : nous risquons de capturer les petites perturbations

ISFA
Institut de Science Financière et d'Assurances

Le danger principal : Le sur-apprentissage (2/2)

Prédiction correspondant à la classe majoritaire

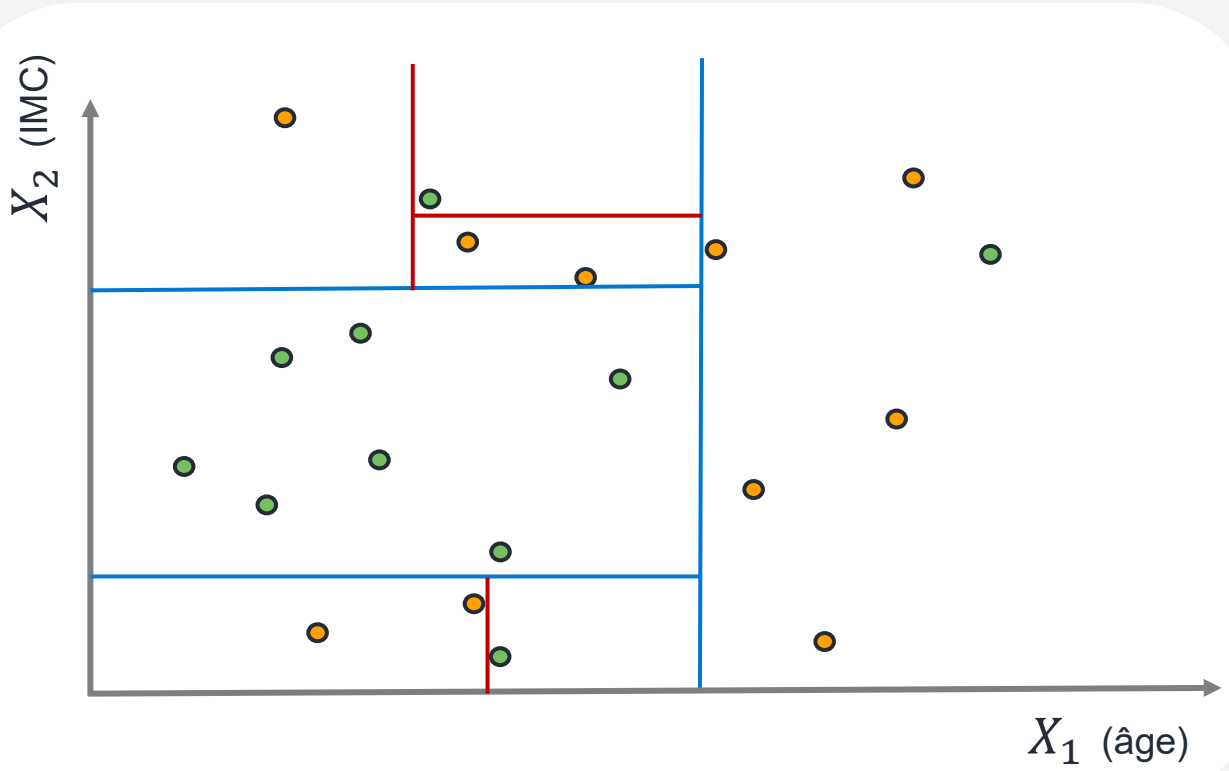
Cas complexe



Taille de l'échantillon après la division *(min_samples_split)* **très petite !**

Observations :

Variable cible : $Y = \{\bullet, \bullet\}$ (décès)



Sur-apprentissage : nous risquons de capturer les petites perturbations

ISFA Institut de Science Financière et d'Assurances

Les techniques pour maîtriser la complexité (Hyperparamètres)

Régulariser l'arbre avec les hyperparamètres

Hyperparamètre	Rôle
max_depth	Profondeur maximale de l'arbre. **Limite directe du surapprentissage** (overfitting).
min_samples_split	Nombre minimal d'échantillons requis pour diviser un nœud.
min_samples_leaf	Nombre minimal d'échantillons dans **une feuille**. **Empêche les feuilles trop spécifiques.**
max_features	Nombre de variables considérées à chaque split. Introduit **de l'aléa / diversité** (utile en ensemble).

Les techniques pour maîtriser la complexité (Pruning)

Régulariser l'arbre avec l'élagage (pruning), avec deux types d'égalage

	Type d'élagage	Principe
Avant construction	**Pré-élagage (Pre-pruning)**	Stopper la croissance selon des **règles simples** (ex. max_depth, min_samples_split)
Après construction	**Post-élagage (Post-pruning)**	Supprimer des branches **a posteriori** si elles **n'améliorent pas la performance**

Objectif : Trouver un **compromis entre complexité de l'arbre et erreur de généralisation**. On cherche à minimiser la fonction :

$$R_\alpha(T) = R(T) + \alpha \cdot |T|$$

- $R(T)$: erreur de l'arbre T
- $|T|$: nombre de feuilles de l'arbre T
- α : paramètre de régularisation (pénalise les arbres complexes)

Synthèse sur l'arbre de décision

Avantages et inconvénients du modèle

✅ Avantages	❌ Inconvénients
Interprétable (*white-box model*)	**Instable** : un petit changement peut modifier toute la structure de l'arbre
Facile à visualiser (arbre lisible, règles explicites)	**Risque élevé d'overfitting** (arbre profond = surapprentissage)
Pas besoin de normalisation des données (se base sur des seuils)	**Biais possibles** si une classe est dominante (bordures de décision injustes)
Gère naturellement les variables **numériques et catégorielles**	**Moins performant seul** que des modèles plus complexes (ex: forêts, boosting)

2. Introduction aux méthodes ensemblistes

ISFA
Institut de Science Financière et d'Assurances

Fondements des Méthodes Ensemblistes

Combiner plusieurs modèles pour améliorer la performance prédictive globale

Objectif :

- **Améliorer la robustesse : moins sensible aux variations des données d'entraînement.**
- **Renforcer la généralisation : meilleure performance sur les données non vues.**
- **Réduire le surapprentissage (overfitting) et/ou le sous-apprentissage (underfitting), selon la méthode utilisée.**

Limitations d'un modèle seul :

- Un modèle seul, **comme un arbre de décision, peut être instable** : une petite variation dans les données peut entraîner un changement radical dans la structure du modèle.
- Il peut aussi être biaisé ou trop spécialisé, ce qui nuit à sa capacité à généraliser sur de nouvelles données.

Intelligence collective :

- Plutôt que de s'appuyer sur un seul modèle « expert », dont les prédictions peuvent être biaisées ou instables, on **combine plusieurs modèles pour obtenir un résultat plus fiable**.
- C'est le concept de **« sagesse des foules »** appliqué au domaine du machine learning (James Surowiecki, 2004) : en agrégeant des points de vue diversifiés, on tend à réduire les erreurs individuelles.

Rappel compromis Biais-Variance (1/2)

Comprendre les sources d'erreur pour mieux les mitiger

Compromis Biais-Variance :

Erreur de généralisation (erreur sur données test)

= **Biais** + **Variance** + **Erreur irréductible**

Biais : erreur systématique, due à des hypothèses trop simplistes. Le modèle « rate » la cible
→ **sous-apprentissage**.

Variance : erreur liée à la sensibilité aux fluctuations des données d'entraînement. Le modèle « s'éparpille » → **sur-apprentissage**.

À retenir :

Le **Bagging** vise à **réduire la variance**,

Le **Boosting** vise à **réduire le biais**.

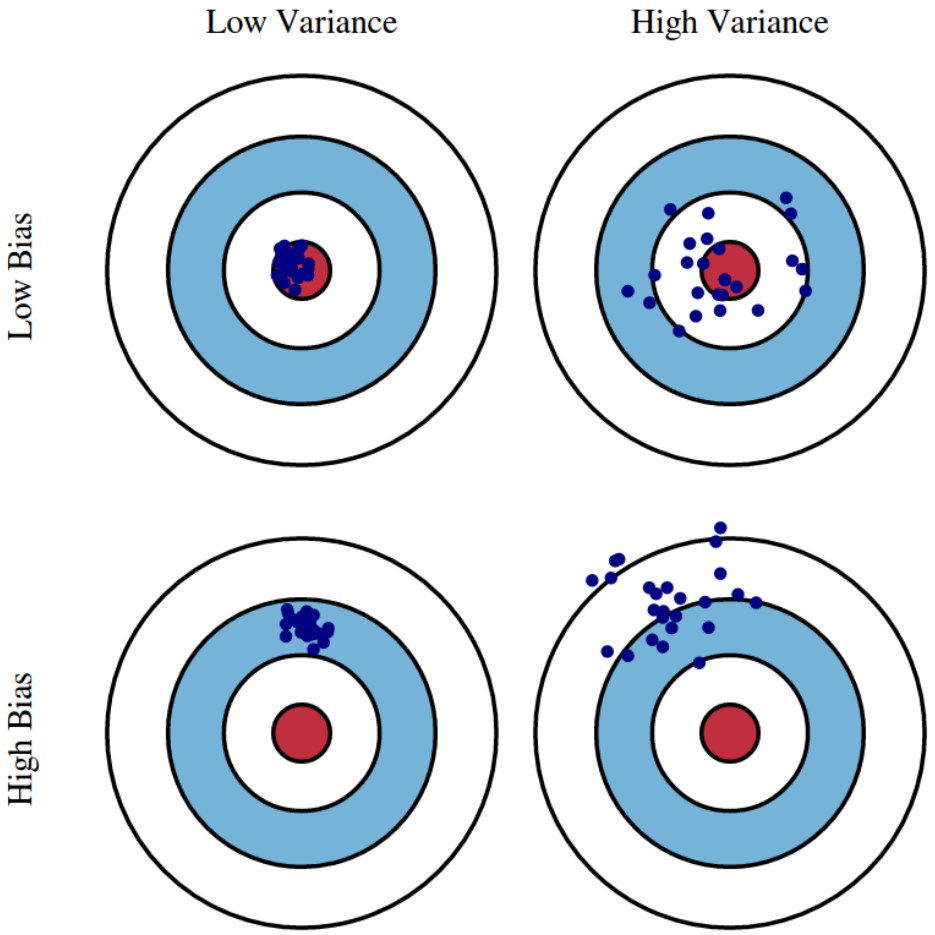
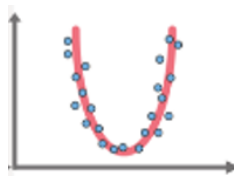


Image Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

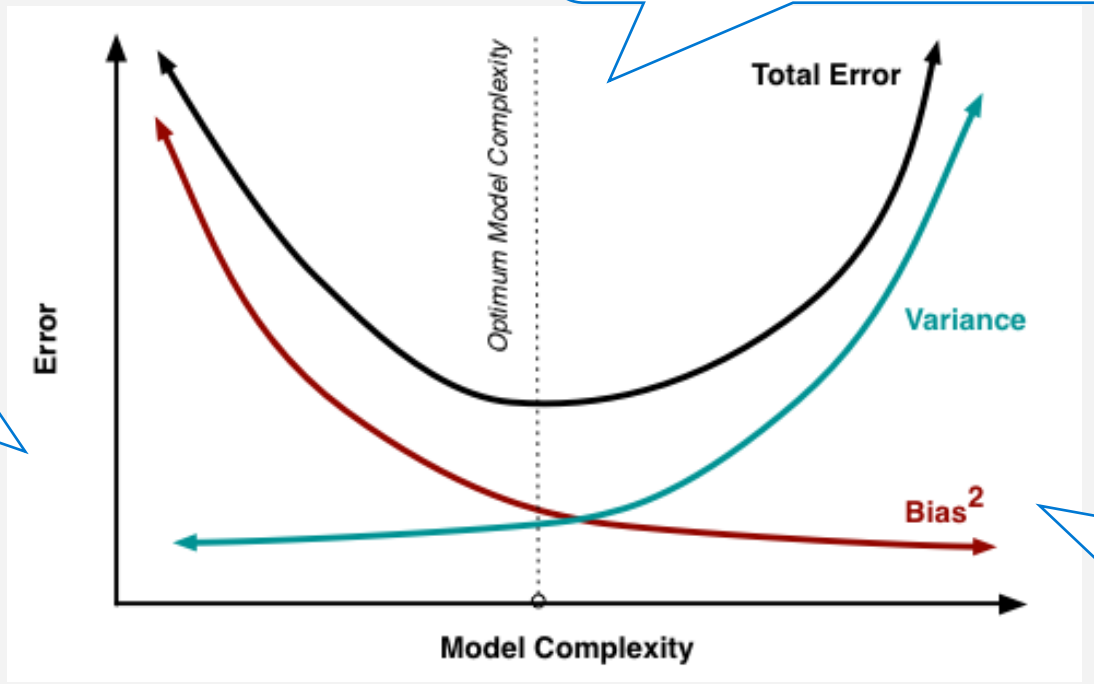
Rappel compromis Biais-Variance (2/2)

Comprendre les sources d'erreur pour mieux les mitiger

Notre objectif



Sous-apprentissage



Sur-apprentissage

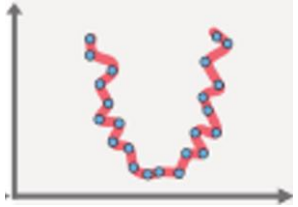


Image Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

3. Le bagging et les forêts aléatoires

Principe du Bagging

Forêts aléatoires

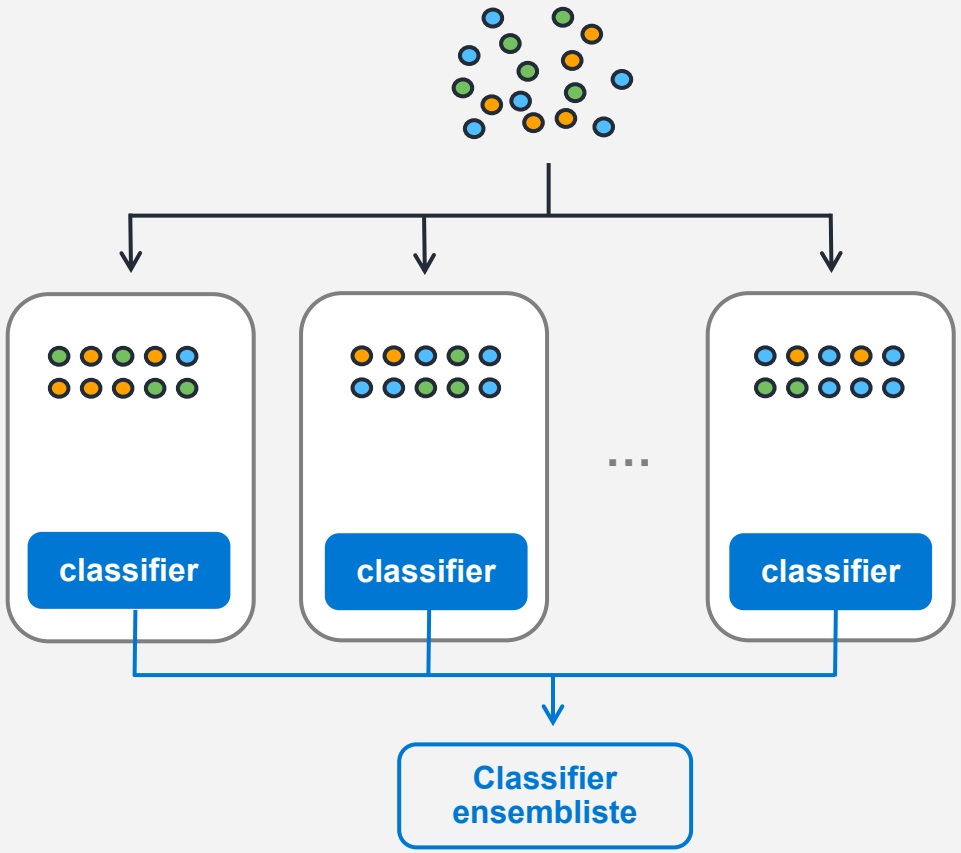
Hyperparamètres

Avantages / Inconvénients

ISFA
Institut de Science Financière et d'Assurances

Le principe du bagging (bootstrap aggregating)

Réduire l'instabilité en agrégeant des modèles



Observations : ce sont généralement les données d'entrainement

Bootstraper : créer N nouveaux datasets en tirant avec remise depuis le dataset original.

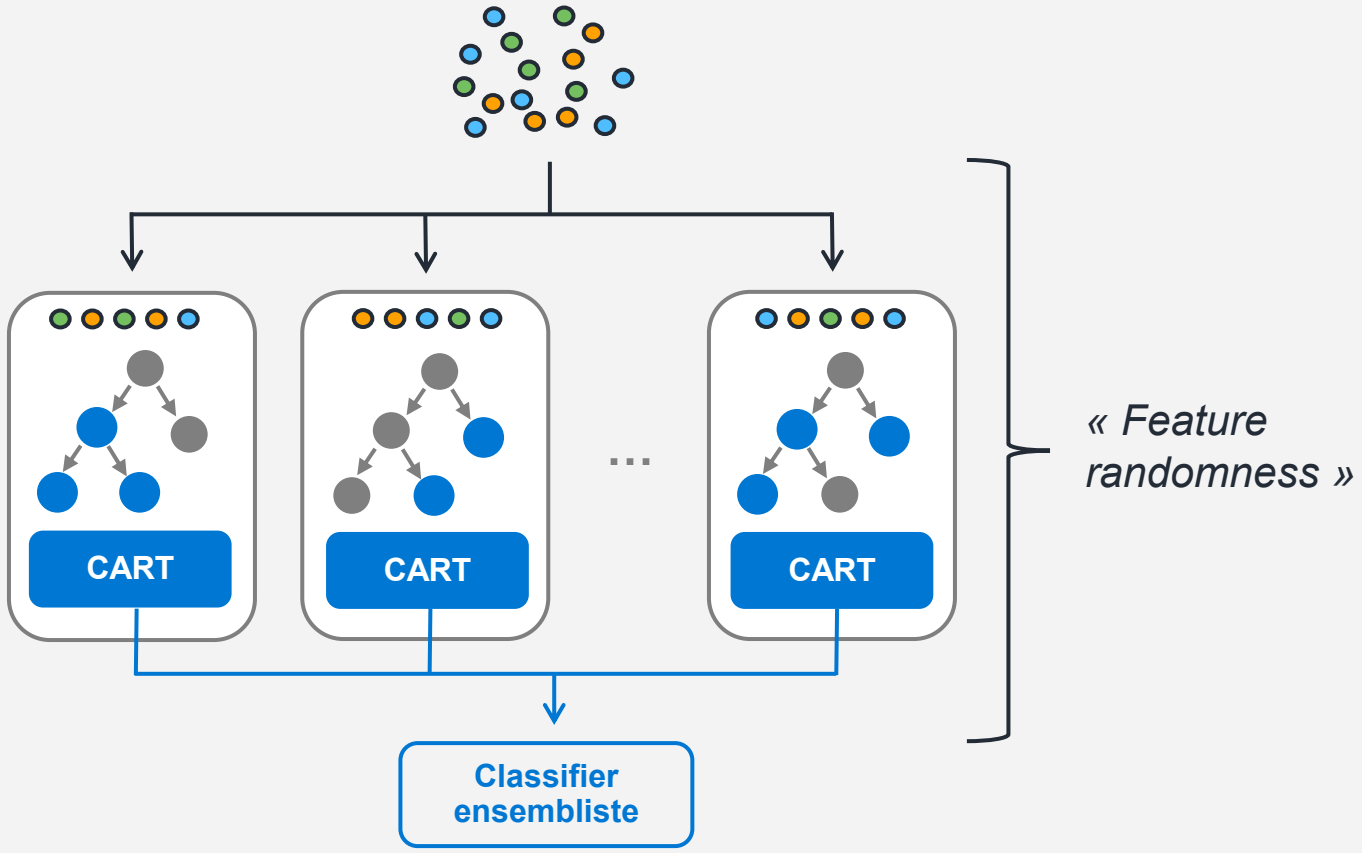
Entraînement : entraîner un arbre de décision sur chaque dataset « bootstrapé »

Bagging / agréger : réduit la variance en agrégeant les résultats. Faire la **moyenne (régression)** ou **voter (classification)**.

Les forêts aléatoires : l'amélioration clé

Comment s'assurer que les arbres sont suffisamment différents ?

- C'est du **Bagging + une astuce supplémentaire**.
- **L'astuce « Random » :** À chaque split d'un arbre, on ne **considère qu'un sous-ensemble aléatoire de variables**.
- Pourquoi ? Pour **décorréler les arbres**. Si une variable est très corrélée avec la variable cible, tous les arbres du Bagging voudront l'utiliser en premier, ce qui les rendrait très similaires. Le Random Forest les force à explorer d'autres variables.

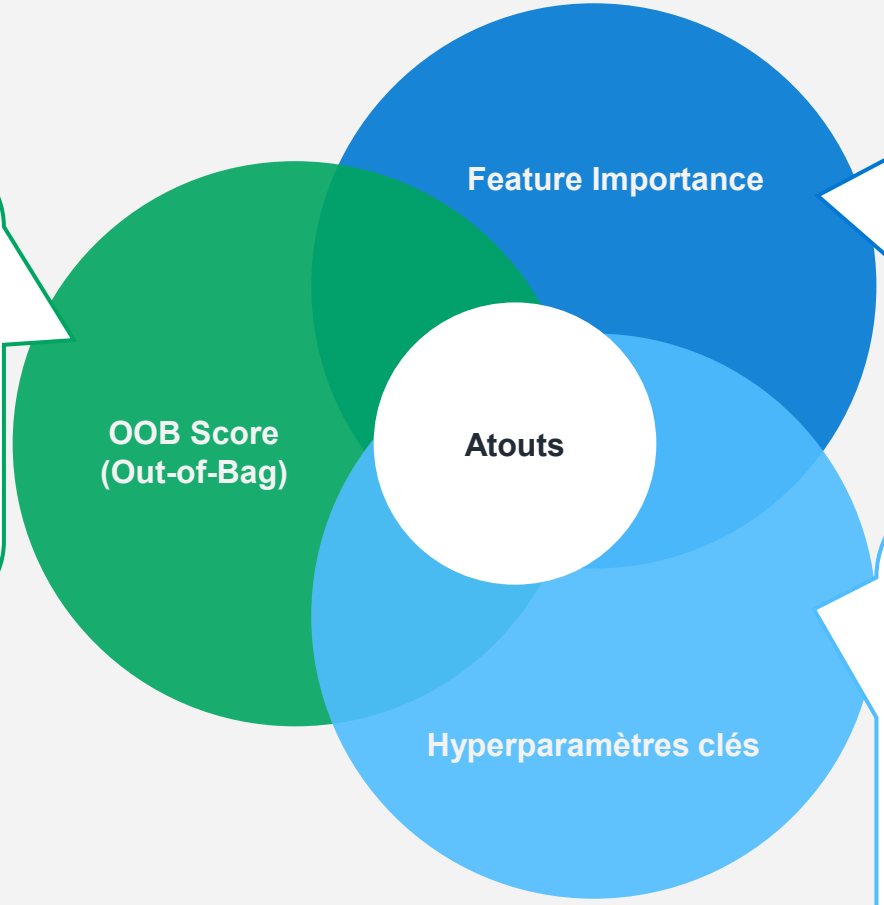


Les atouts pratiques des forêts aléatoires

Le score « out-of-bag » et l'importance des variables

Principe : chaque arbre est entraîné sur un échantillon bootstrap (avec remise). Les **données non utilisées** par l'arbre servent à **tester sa performance**

Intérêt pratique : évite de créer un jeu de validation : on obtient une **évaluation du modèle "gratuite"**



Principe : On mesure, pour chaque variable, **la réduction moyenne de l'impureté** (Gini ou Entropie) qu'elle provoque lorsqu'elle est utilisée pour un split

Intérêt pratique : Permet de **prioriser les variables** les plus utiles au modèle → utile en sélection de variables

Hyperparamètres clés à connaître (voir aussi CART) **:**

Paramètre	Définition	Impact sur le modèle
n_estimators	Nombre total d'arbres à entraîner dans la forêt	Plus il est élevé, plus le modèle est **stable et performant** (jusqu'à un certain seuil)
max_features	Nombre de variables sélectionnées **au hasard** à chaque split	Réduit la corrélation entre les arbres → **augmente la diversité et la robustesse globale**

ISFA
Institut de Science Financière et d'Assurances

Bilan des forêts aléatoires

Avantages et inconvénients de la méthode

✅ Avantages	❌ Inconvénients
Robuste aux variations des données (réduction de la variance)	**Moins interprétable** qu'un arbre unique (effet boîte noire)
Performant sur de nombreux jeux de données, sans tuning complexe	**Entraînement lent** si la forêt est très grande
Réduction massive de l'overfitting par rapport à un seul arbre	**Gourmand en mémoire** (stocke de nombreux arbres en RAM)
Facile à utiliser : peu d'hyperparamètres critiques à régler	**Feature importance parfois biaisée** (variables très corrélées ou catégorielles)

- Un excellent **compromis robustesse/performance/simplicité**
- Mais attention à la **perte de transparence** si l'explicabilité est critique

4. Le **boosting** : apprendre de ses erreurs

Principe du boosting

Gradient de boosting

Lien avec la descente de gradient

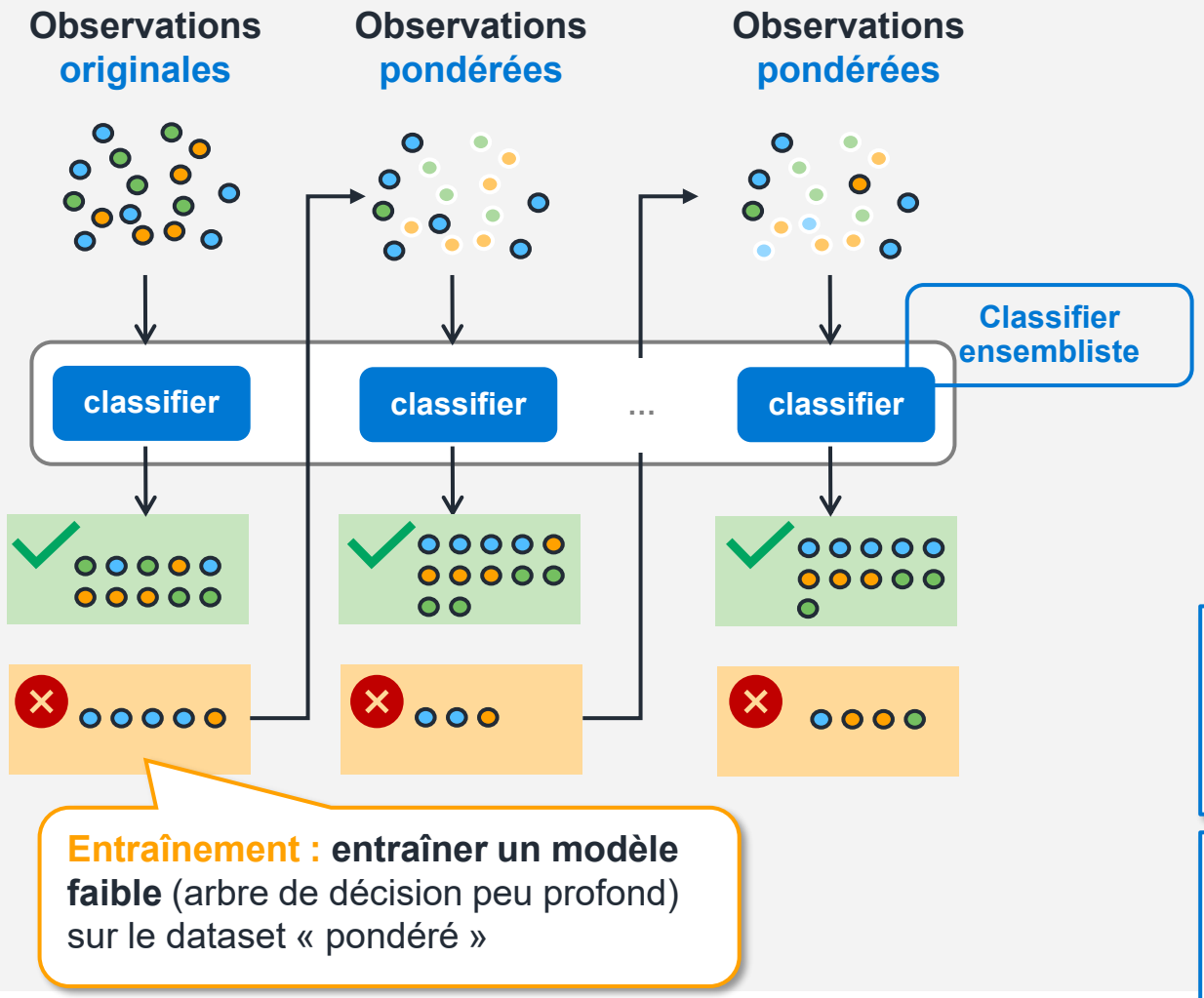
Hyperparamètres

Avantages / Inconvénients

ISFA
Institut de Science Financière et d'Assurances

Le boosting, une approche séquentielle

Apprendre de ses erreurs, une itération à la fois



- **Construire un modèle fort** (*strong* learner) en additionnant des modèles faibles (*weak* learners, comme les petits arbres de décision).
- **Contraste avec le bagging** (parallèle vs. séquentiel).
- L'idée est de se concentrer sur les **exemples mal classés par les itérations précédentes**.
- **Objectif principal : la réduction du biais.**

AdaBoost : chaque modèle est **pondéré** selon sa performance, et les données mal classées reçoivent **plus de poids**

AdaBoost réagit aux erreurs **en réajustant les poids des données**, alors que **Gradient Boosting (GBM)** les corrige **en suivant la pente de l'erreur**

Le Boosting vu comme un modèle additif

La **construction additive d'un modèle**.

Weak learners

Strong learner

$$\hat{f}(x) = \hat{f}_0(x) + \hat{f}_1(x) + \ldots + \hat{f}_M(x)$$

Le **Gradient Boosting** (Friedman, 2001) généralise le boosting en le considérant comme un **modèle additif construit de manière séquentielle**

1. On commence avec un modèle initial simple, $\hat{f}_0(x)$. C'est **souvent une constante** : la moyenne de Y pour la régression, ou le logit de la probabilité moyenne pour la classification.

$$\hat{f}_0(x) = \arg min_\gamma \; \Sigma \, L(y_i, \gamma)$$

2. À chaque étape $m = 1, \ldots, M$, on cherche à améliorer le modèle courant $\hat{f}_{m-1}(x)$ en y ajoutant une nouvelle fonction $h_m(x)$ (un « weak learner », typiquement un arbre de décision) :

$$\hat{f}_m(x) = \hat{f}_{m-1}(x) + h_m(x)$$

La question centrale : Comment choisir la « meilleure » fonction $h_m(x)$ à chaque étape pour améliorer au maximum notre modèle ?

Idée clé : ajuster les résidus (cas de la perte quadratique)

- Considérons le cas de la **régression** avec la **perte quadratique**

$$L(y, \hat{f}) = \frac{1}{2}(y - \hat{f})^2$$

- À l'étape m, nous voulons **minimiser la perte totale** :

$$\Sigma_i \, L\left(y_i, \hat{f}_{m-1}(x_i) + h_m(x_i)\right) = \frac{1}{2} \Sigma_i (y_i - \hat{f}_{m-1}(x_i) - h_m(x_i))^2$$

- Notons $r_{im} = y_i - \hat{f}_{m-1}(x_i)$ le **résidu** de l'étape précédente pour l'observation i.

- La fonction à minimiser devient :

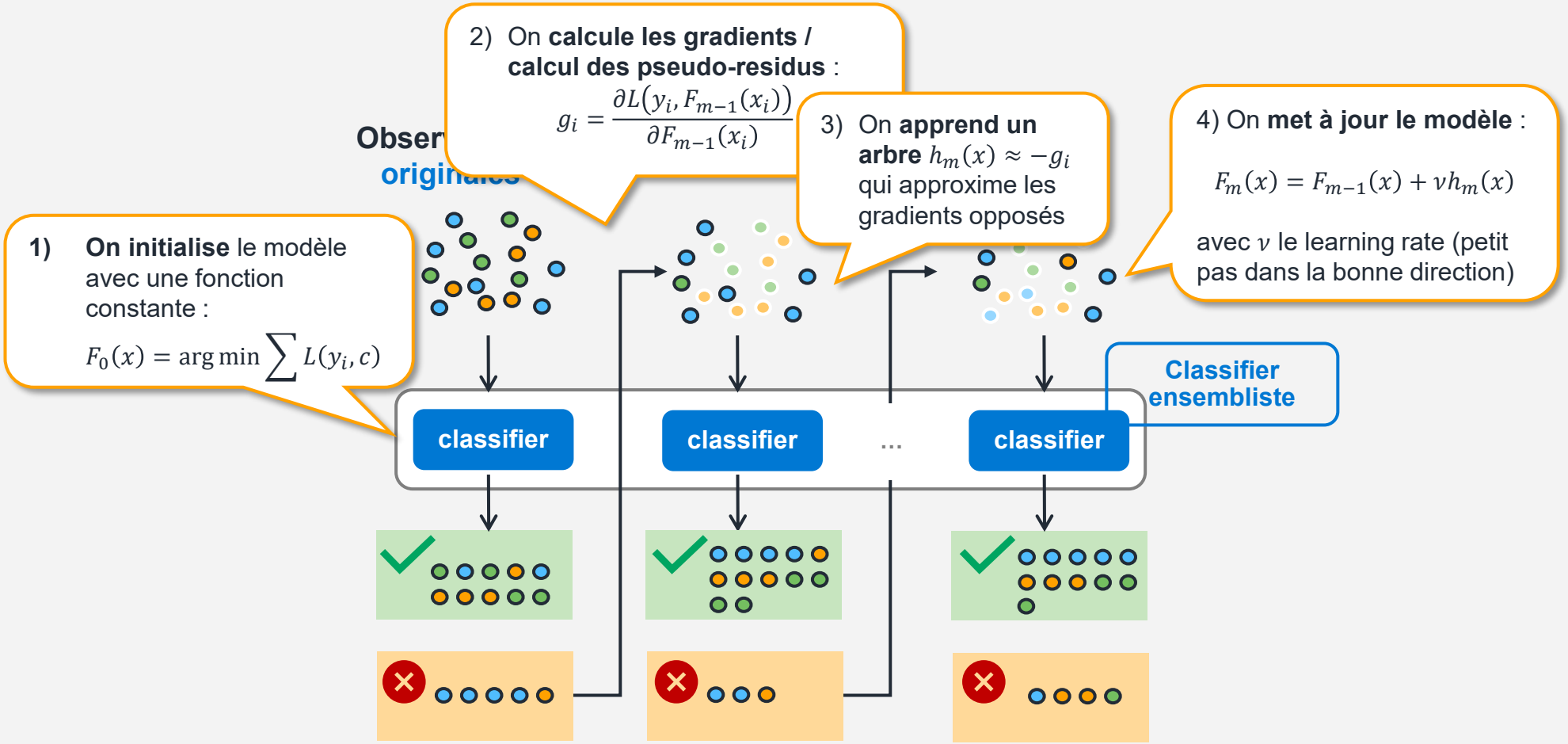
$$\frac{1}{2} \Sigma_i \, (r_{im} - h_m(x_i))^2$$

- **Remarque :** Pour minimiser la perte globale, **la meilleure fonction $h_m(x)$** est simplement un modèle (un arbre de régression) entraîné à prédire les **résidus** de l'étape précédente !

Pour étendre cette méthode au-delà de la perte quadratique (ex: pour la classification), on remplace le résidu (y - f̂) par son équivalent mathématique plus général : le **pseudo-résidu**, qui est le gradient négatif de la fonction de perte.

L'intuition du Gradient Boosting (Machine) (1/2)

Comment chaque nouvel arbre améliore le modèle global



1) On cherche à **minimiser une fonction de perte** $L(y, F(x))$, où $F(x)$ est le modèle actuel.
2) On optimise **directement une fonction** (la prédiction elle-même).
3) À chaque étape, on ajoute une nouvelle fonction (typiquement un **arbre de décision**) pour **corriger l'erreur** du modèle actuel.

Gradient Boosting (GBM) est une optimisation par descente de gradient dans l'espace des fonctions

L'intuition du Gradient Boosting (Machine) (2/2)

Comment chaque nouvel arbre améliore le modèle global

	AdaBoost	Gradient Boosting (GBM)
Mécanisme	**Augmente le poids** des erreurs	**Suit le gradient** de la fonction de perte
Ce que le nouveau modèle apprend	À corriger **les exemples mal classés**	À corriger **les erreurs numériques** (résidus ou gradients)
Poids des données	Modifiés à chaque itération	Ne change pas : le gradient guide
Fonction de perte	**Fixe :** souvent log-loss (exponentielle)	**Flexible** : MSE, log-loss, Huber, custom
Sensibilité au bruit	Plus sensible (fortes erreurs sur pondérations)	Plus robuste (corrige « en douceur »)
Modèle de base typique	**Stump** (arbre profondeur 1)	Arbre peu profond (mais + complexe que stump)

Gradient Boosting (GBM) est une optimisation par descente de gradient dans l'espace des fonctions

Hyperparamètres clés du Gradient Boosting

Piloter la performance : learning_rate et n_estimators

Hyperparamètre	Rôle	Effet sur le modèle
n_estimators	Nombre total d'arbres (itérations de boosting)	Trop grand → overfitting ; trop petit → sous-apprentissage
learning_rate (ou eta)	Contrôle la contribution de chaque nouvel arbre	Plus petit = plus lent, mais souvent **plus précis** (à coupler avec n_estimators élevé)
Hyperparamètres des CARTs : *max_depth, min_samples_split, min_samples_leaf, …*		
subsample	Proportion de données utilisées à chaque étape	< 1 → introduit du **hasard** → aide à la généralisation
max_features	Nb de variables testées à chaque split	Diminue la **corrélation entre arbres**, réduit variance
loss	Fonction de perte à minimiser	Dépend de la tâche (log-loss, MSE, etc.)

Avantages et Inconvénients du GBM « classique »

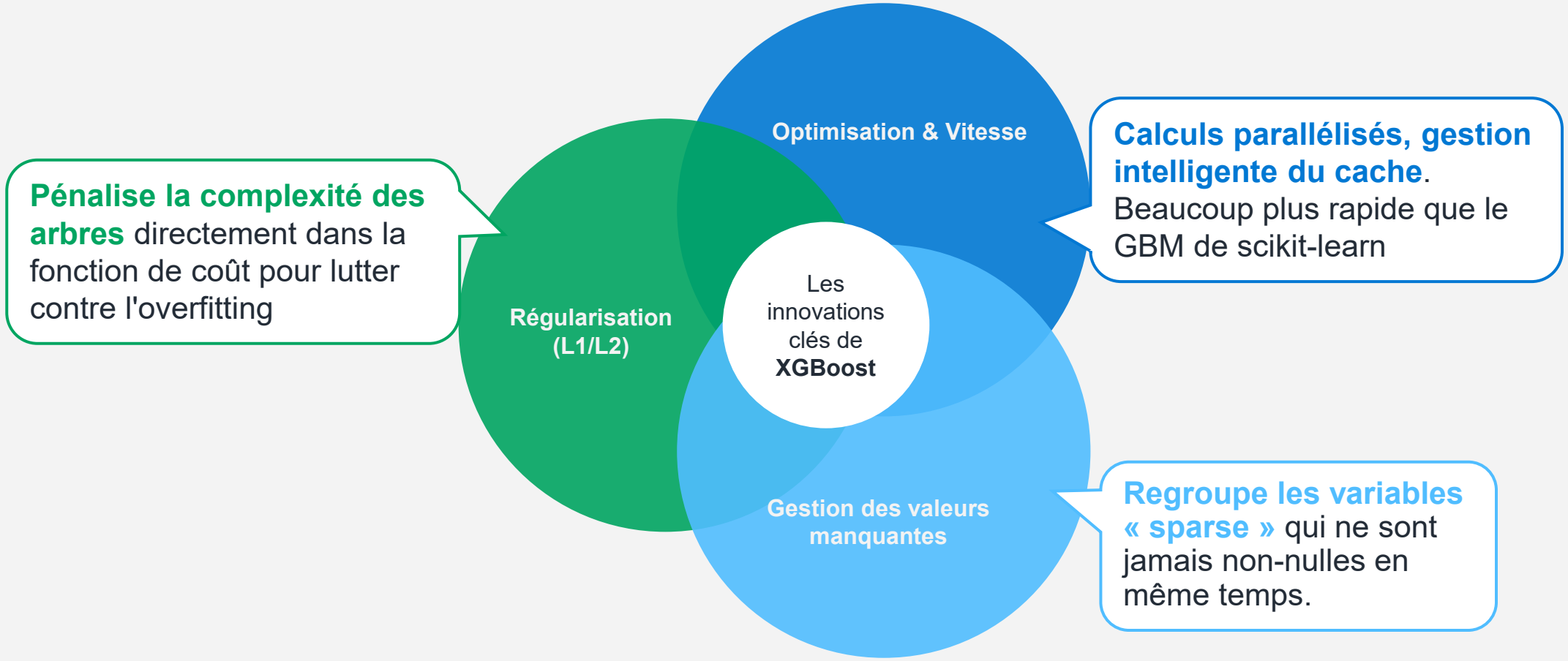
Critère	Arbre de décision	Forêts aléatoires	Gradient Boosting (GBM)
Principe	Arbre unique, divise les données	Ensemble d'arbres **entraînés en parallèle**	Ensemble d'arbres **entraînés en séquence**
Performance	Moyenne	Bonne (réduction de la variance)	**Excellente** (réduction du biais)
Overfitting	Risque élevé	Moins de sur-apprentissage	Risque si mal régulé, mais **très précis**
Interprétabilité	✅ Très bonne	❌ Faible	❌❌ Très faible
Vitesse d'entraînement	Très rapide	Parallélisable (rapide)	Lent (séquentiel, en général pas parallélisable)
Tuning nécessaire	Très peu	Peu	Beaucoup (learning_rate, n_estimators…)
Importance des variables	Lisible (simple)	Moyenne (mais biais possible)	Moyenne (via réduction d'impureté cumulée)

5. Les champions du boosting (XGBoost, LightGBM, CatBoost)

ISFA
Institut de Science Financière et d'Assurances

XGBoost (eXtreme Gradient Boosting), la référence en compétition

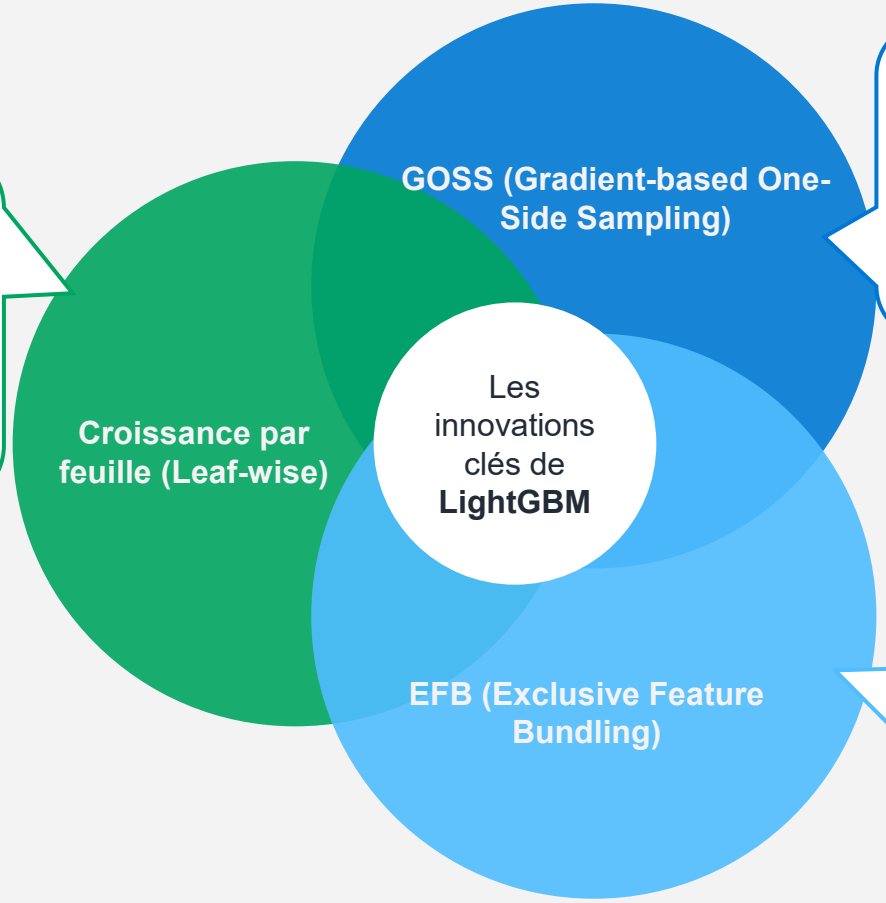
Performance et robustesse grâce à la régularisation



LightGBM (Light Gradient Boosting Machine), le choix de la vitesse

Optimisé pour les très grands volumes de données

Au lieu de construire l'arbre niveau par niveau, il **développe la feuille qui promet le plus grand gain d'information**. C'est plus rapide mais peut overfitter sur de petits datasets.

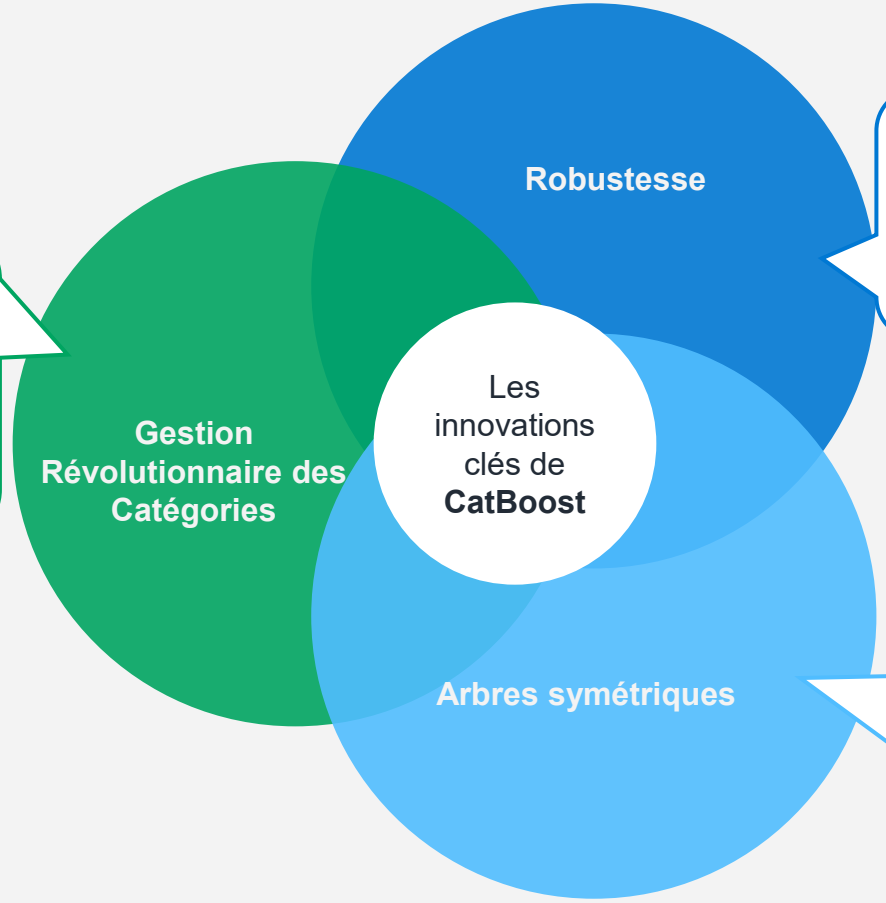


Se concentre sur les **observations avec de grands gradients** (les erreurs les plus difficiles à corriger)

L'algorithme apprend lui-même le meilleur chemin pour les NaN

CatBoost (Categorical Boosting), le spécialiste des données catégorielles

Gérer nativement les variables non-numériques



Utilise une variation du **« Target Encoding » (Ordered TS)** qui évite la fuite de données (data leakage) et la nécessité de faire du One-Hot Encoding massif.

Souvent performant avec les paramètres par défaut, moins de tuning nécessaire.

Utilise des **arbres « oblivious »** (tous les nœuds d'un même niveau utilisent le même critère de split), ce qui aide à combattre l'overfitting et accélère les prédictions.

Tableau récapitulatif des méthodes de Boosting

Gérer nativement les variables non-numériques

Méthode	Construction	Ce que le modèle apprend	Particularités clés
Boosting (générique)	Modèles ajoutés séquentiellement, chaque nouveau modèle corrige le précédent	Les **erreurs** ou **résidus**	**Cadre général** : base de toutes les méthodes suivantes
AdaBoost	Chaque modèle est **pondéré** selon sa performance, et les données mal classées reçoivent **plus de poids**	Les **erreurs pondérées** (échantillons difficiles)	**pondère les erreurs** → bon pour données simples.
GBM	À chaque étape, le modèle suit le **gradient de la fonction de perte**	Le **gradient** (direction de l'erreur)	**suit les gradients** → très flexible.
XGBoost	Améliore GBM avec **régularisation, parallélisation, gestion du surapprentissage**	Gradient + **régularisation L1/L2**	GBM **+ vitesse + régularisation**.
LightGBM	Utilise des **histogrammes** et une croissance **leaf-wise** (feuilles les plus utiles d'abord)	Gradient (comme GBM), mais **optimisé pour la vitesse**	GBM **ultra rapide** avec feuilles optimisées.
CatBoost	Gère **nativement les variables catégorielles** sans encodage manuel	Gradient + **ordonnancement intelligent des données**	**catégorielles natives**, très robuste.

6. Synthèse et bonnes pratiques

ISFA
Institut de Science Financière et d'Assurances

Tableau comparatif des algorithmes

Une grille d'analyse pour guider la décision

Algorithme	Performance	Vitesse d'entraînement	Gestion des catégories	Facilité de tuning	Robustesse à l'overfitting	Interprétabilité
Arbre de Décision	Faible / Moyenne	Très rapide	Manuelle (encodage requis)	Très facile	Faible (overfitting fréquent)	Excellente
Random Forest	Bonne	Rapide à Moyenne (parallélisable)	Manuelle	Facile	Bonne (moyennage entre arbres)	Moyenne
GBM (classique)	Très bonne	Lente (séquentielle)	Manuelle	Sensible (learning rate etc…)	Bonne avec régularisation manuelle	Faible
XGBoost	Excellente	Moyenne	Automatique (simple)	Moyennement complexe	Bonne (régularisation L1/L2)	Faible
LightGBM	Excellente	Rapide	Encodage requis	Moyenne	Bonne (mais peut overfitter vite)	Faible
CatBoost	Excellente	Moyenne	Native (très efficace)	Facile	Très bonne	Faible

Tableau comparatif des algorithmes

Une grille d'analyse pour guider la décision

Algorithme	Performance	Vitesse d'entraînement	Gestion des catégories	Facilité de tuning	Robustesse / l'overfitting	Interprétabilité
Arbre de Décision	Faible / Moyenne	Très rapide	Manuelle (encodage requis)	Très facile	Faible (overfitting fréquent)	Excellente
Random Forest	Bonne	Rapide à Moyenne (parallélisable)	Manuelle	Facile	Bonne (moyennage entre arbres)	Moyenne
GBM (classique)	Très bonne	Lente (séquentielle)	Manuelle	Sensible (learning rate etc…)	Bonne avec régularisation manuelle	
XGBoost	Excellente	Moyenne	Automatique (simple)	Moyennement complexe	Bonne (régularisation L1/L2)	Faible
LightGBM	Excell		uis	Moyenne	Bonne (mais peut overfitter vite)	Faible
CatBoost	Excellente	Moyenne	Native (très efficace)	Facile	Très bonne	Faible

Besoin ***d'interprétabilité*** *?*
Arbre de Décision

Première approche ***robuste*** *?*
Random Forest

Standard de ***compétition****, dataset de* ***taille moyenne*** *?* XGBoost

Dataset très large *avec* ***beaucoup de catégories*** *?*
Essayer CatBoost ou LightGBM

ISFA
Institut de Science Financière et d'Assurances

Méthodologie : validation et optimisation

De la validation croisée à l'optimisation des hyperparamètres

Validation Croisée : toujours l'utiliser pour **évaluer et tuner vos modèles**.

Tuning d'hyperparamètres :

Méthode	Description simple	Avantages / Limites
Grid Search	Teste **toutes les combinaisons** d'un ensemble de paramètres	Exhaustif mais … ❌ Très lent, peu scalable
Random Search	Tire **au hasard des combinaisons** dans l'espace de recherche	**Rapide et efficace** si certaines dimensions sont inutiles
Optimisation bayésienne (ex : Optuna, Hyperopt)	Utilise **les résultats précédents** pour choisir intelligemment les combinaisons suivantes	**État de l'art :** Moins de tests pour meilleurs résultats

Méthodes principales d’évaluation de l’importance des variables

De la validation croisée à l'optimisation des hyperparamètres

Méthode d'importance	Description	Algorithmes concernés
Gain (ou Gini Gain)	Réduction de l'impureté (Gini ou Entropie) ou du loss à chaque split.	Tous (Random Forest, GBM, XGBoost, LGBM, CatBoost)
Split/Count	Nombre de fois qu’une variable est utilisée pour un split.	Tous, mais souvent en complément du gain
Permutation Importance	Diminution de performance quand on permute les valeurs d’une variable.	Modèle-agnostique (mais disponible dans scikit-learn et certains packages pour tous ces modèles)

Interprétabilité des boîtes noires : Même si le modèle est complexe (GBM, XGBoost, CatBoost…), il est possible de comprendre ses décisions. Utilisez des outils comme **SHAP (SHapley Additive exPlanations)** pour :

- **Visualiser l’impact des variables sur chaque prédiction**
- **Construire des analyses locales ou globales sur l’importance des features**

7. Conclusion

Les points essentiels à retenir

Synthèse des concepts clés de la session

- Les **arbres** sont les briques.
- Le **Bagging** (Random Forest) **réduit la variance** en parallélisant des experts indépendants.
- Le **Boosting** **réduit le biais** (et la variance) en faisant collaborer des experts en série.
- **XGBoost, LightGBM, CatBoost** sont des **implémentations optimisées** de Boosting pour des cas d'usage spécifiques (performance, vitesse, catégories).

Milliman

ISFA
Institut de Science Financière et d'Assurances
GRANDE ÉCOLE D'ACTUARIAT & GESTION DES RISQUES

Merci

François HU

Francois.hu@milliman.com