INTRODUCTION À LA SCIENCE DES DONNÉES

Apprentissage Supervisé

Validation & Arbres de Décision

Victor Bertret

- Docteur en Mathématiques
- Ingénieur IA chez **Purecontrol**

16 février 2026

Flashback : Les ingrédients de l'Apprentissage Supervisé

Rappel S1 : Notre but est de trouver une fonction mathématique capable de généraliser.

1. L'Expérience (S)

On observe un échantillon fini de couples :

$$S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

- $\mathbf{x}$: **Entrées** (Features).
- y : **Réponse** (Vérité terrain).

Hypothèse : Données i.i.d.

2. Le Modèle (h)

On cherche une fonction h dans un espace d'hypothèses $\mathcal{H}$:

$$\hat{y} = h(\mathbf{x})$$

- **Objectif :** $\hat{y} \approx y$ (Minimiser l'erreur).
- **Risque :** Apprendre par cœur (Overfitting).

Flashback : Les ingrédients de l'Apprentissage Supervisé

Rappel S1 : Notre but est de trouver une fonction mathématique capable de généraliser.

1. L'Expérience (S)

On observe un échantillon fini de couples :

$$S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

- $\mathbf{x}$: **Entrées** (Features).
- y : **Réponse** (Vérité terrain).

Hypothèse : Données i.i.d.

2. Le Modèle (h)

On cherche une fonction h dans un espace d'hypothèses $\mathcal{H}$:

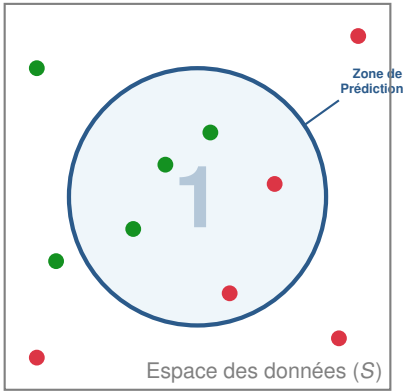
$$\hat{y} = h(\mathbf{x})$$

- **Objectif :** $\hat{y} \approx y$ (Minimiser l'erreur).
- **Risque :** Apprendre par cœur (Overfitting).

Test Mémoire : On a vu comment noter un modèle.
Quelles sont les métriques qu'on peut utiliser pour évaluer la qualité d'un modèle de régression ?

Rappel : Le Cercle de Classification

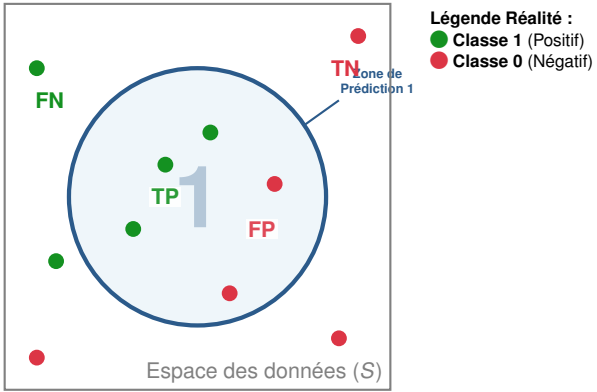
Quiz : Sachant que ce modèle prédit **1** à l'intérieur du cercle...
Retrouvez les 4 zones (TP, FP, TN, FN).



Légende Réalité :
Classe 1 (Positif)
Classe 0 (Négatif)

Rappel : Le Cercle de Classification

Quiz : Sachant que ce modèle prédit **1** à l'intérieur du cercle...
Retrouvez les 4 zones (TP, FP, TN, FN).



Légende Réalité :
- **Classe 1** (Positif)
- **Classe 0** (Négatif)

Rappel : Vert = Classe 1 (Réelle), Rouge = Classe 0 (Réelle).

Rappel S1 : Le Compromis Éternel

Précision (Qualité)

$$\frac{TP}{TP + FP}$$

"Quand je dis OUI, est-ce que c'est vrai ?"

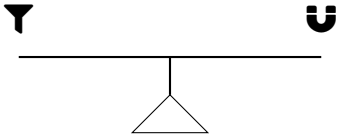
Peur des Fausses Alarmes.

Rappel (Quantité)

$$\frac{TP}{TP + FN}$$

"Est-ce que j'ai trouvé tout le monde ?"

Peur de louper un truc.



On ne peut pas maximiser les deux à la fois.

Le Piège de la Moyenne Arithmétique

Cas extrême : J'ai un modèle très prudent.

- **Précision = 100%** (Il ne se trompe jamais quand il parle).
- **Rappel = 1%** (Il ne parle presque jamais, il loupe 99% des cas).

Le Piège de la Moyenne Arithmétique

❓ **Cas extrême :** J'ai un modèle très prudent.

- **Précision = 100%** (Il ne se trompe jamais quand il parle).
- **Rappel = 1%** (Il ne parle presque jamais, il loupe 99% des cas).

Si on fait la moyenne classique (Arithmétique) :

$$\text{Score} = \frac{100 + 1}{2} = \mathbf{50.5}\%$$

C'est trompeur !

Un modèle qui rate 99% des cibles est **inutile**, il ne vaut pas la moyenne.
Il nous faut une méthode de calcul qui sanctionne le déséquilibre.

La Solution : Le F1-Score (Moyenne Harmonique)

⚖ Le F1-Score

$$F1 = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

L'Intuition Physique :

- C'est comme la **résistance d'une chaîne**.
- La solidité globale dépend du **maillon le plus faible**.

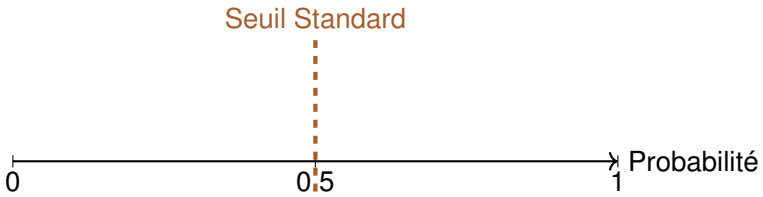
Reprenons l'exemple :

$$F1 = 2 \times \frac{100 \times 1}{100 + 1} \approx \mathbf{1.98\%}$$

✔ La sanction est tombée.

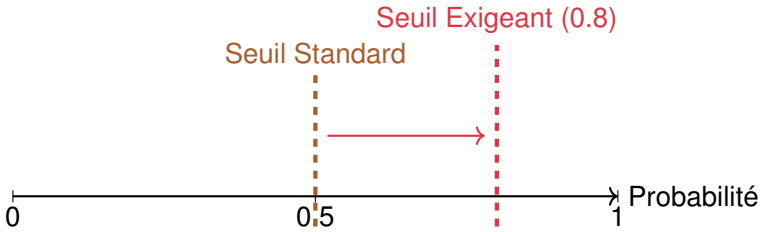
La limite des métriques figées : Le Seuil

⚙ Rappel : Le modèle sort une probabilité (ex : 0.7).
Pour calculer (Précision, Rappel, F1), on doit fixer un seuil (ex : 0.5).



La limite des métriques figées : Le Seuil

Rappel : Le modèle sort une probabilité (ex : 0.7).
Pour calculer (Précision, Rappel, F1), on doit fixer un seuil (ex : 0.5).

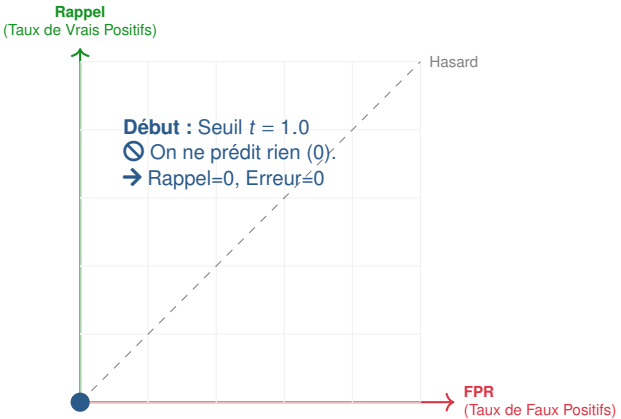


Le Problème :

- Si je change le seuil, mes métriques changent !
- Comment savoir si mon modèle est "intrinsèquement" bon, indépendamment de mon réglage ?

La Solution Globale : La Courbe ROC

Receiver Operating Characteristic : L'histoire de tous les seuils possibles.



↑ Axe Y : Le Gain (Rappel)

Combien de "1" ai-je trouvés ?

$$TPR = \frac{\text{Vrais Positifs (TP)}}{\text{Total Positifs Réels}}$$

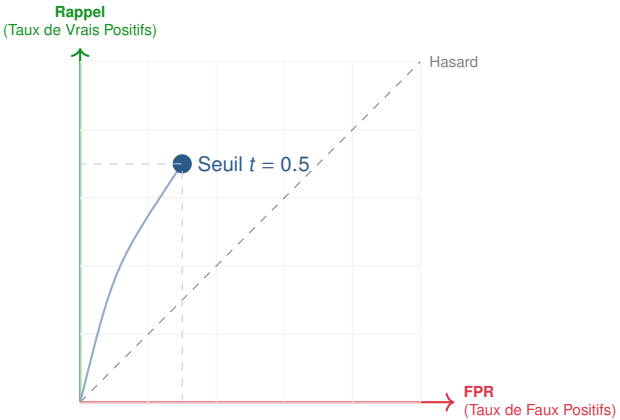
→ Axe X : Le Coût (FPR)

Taux de fausses alarmes.

$$FPR = \frac{\text{Faux Positifs (FP)}}{\text{Total Négatifs Réels}}$$

La Solution Globale : La Courbe ROC

Receiver Operating Characteristic : L'histoire de tous les seuils possibles.



↑ Axe Y : Le Gain (Rappel)

Combien de "1" ai-je trouvés ?

$$TPR = \frac{\text{Vrais Positifs (TP)}}{\text{Total Positifs Réels}}$$

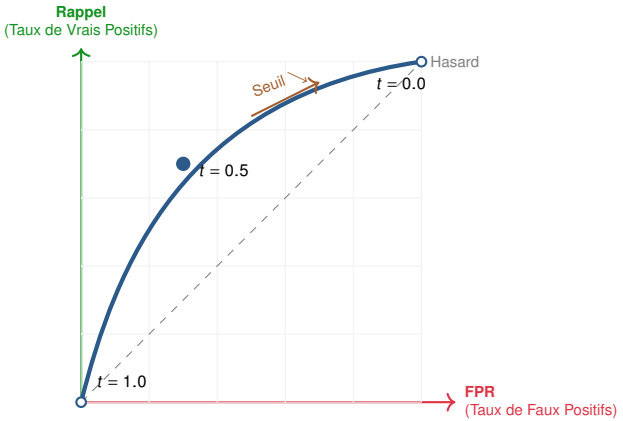
→ Axe X : Le Coût (FPR)

Taux de fausses alarmes.

$$FPR = \frac{\text{Faux Positifs (FP)}}{\text{Total Négatifs Réels}}$$

La Solution Globale : La Courbe ROC

Receiver Operating Characteristic : L'histoire de tous les seuils possibles.



↑ Axe Y : Le Gain (Rappel)

Combien de "1" ai-je trouvés ?

$$TPR = \frac{\text{Vrais Positifs (TP)}}{\text{Total Positifs Réels}}$$

→ Axe X : Le Coût (FPR)

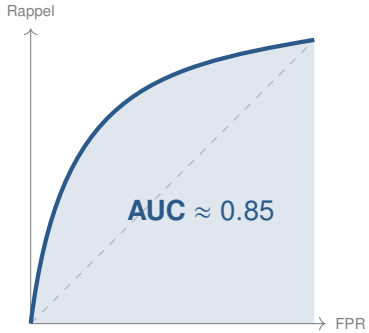
Taux de fausses alarmes.

$$FPR = \frac{\text{Faux Positifs (FP)}}{\text{Total Négatifs Réels}}$$

🏆 Objectif : Monter vite (Gain) sans aller à droite (Coût).

Le Score Unique : AUC (Area Under Curve)

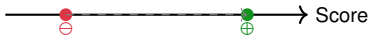
Problème : Comment comparer deux courbes ? → On calcule l'aire en dessous.



Le "Jeu" de l'AUC

On tire au hasard **un Positif** (⊕) et **un Négatif** (⊖).
On compare leur score donné par le modèle.

Bon Ordre ✔



$P(\text{Score}_{\oplus} > \text{Score}_{\ominus}) = \text{AUC}$

Interprétation :

- **AUC = 0.5** : Le modèle range au hasard (1 fois sur 2).
- **AUC = 1.0** : Le modèle range toujours le positif devant.

Synthèse : Quelle métrique choisir ? (Bilan Global)

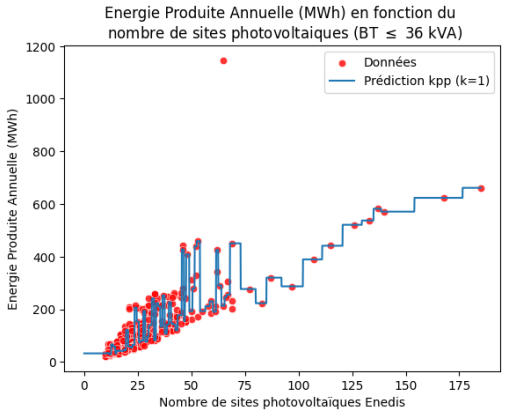
Contexte / Objectif	Métrique	Pourquoi ?
CLASSIFICATION (Discret)		
Classes Équilibrées	**Accuracy**	Simple et intuitif (% de réussite).
Classes Déséquilibrées	**F1-Score**	Sanctionne si on ignore la classe rare.
Priorité : Qualité (Peu de Faux Pos.)	**Précision**	Éviter les fausses alarmes (ex : Spam).
Priorité : Sécurité (Peu de Faux Neg.)	**Rappel**	Ne rien rater (ex : Cancer, Fraude).
Comparaison Globale	**AUC-ROC**	Indépendant du seuil de décision.
RÉGRESSION (Continu)		
Standard (Sensible aux erreurs)	**RMSE (L_2)**	Punit fortement les grosses erreurs (au carré).
Robuste (Présence d'Outliers)	**MAE (L_1)**	Punit linéairement (ignorance des extrêmes).
Interprétation / Communication	R^2	Lisible (0 à 1). Compare au modèle naïf (moyenne).

À garder sous le coude : La "Boîte à Outils" complète.

Sommaire

Le Problème Fondamental : L'Illusion de l'ERM

⚠ **Rappel S1 :** Nous cherchons à minimiser l'erreur réelle $\mathcal{R}(h)$ (sur $\mathcal{D}$), mais nous ne voyons que l'erreur empirique $\hat{\mathcal{R}}_S$ (sur S).



Résultat de l'optimisation :

- L'algorithme a testé tous les k.
- Pour **k=1**, l'erreur est de **0** (Score parfait).
- → **Il a donc choisi** $k = 1$**.**

C'est mathématiquement "juste" sur S...
... mais c'est du **OVERFITTING** en réalité.

La Solution ?

Question

Nous en avons parlé oralement la dernière fois...

Comment piéger le modèle pour mesurer sa vraie performance ?

La Solution ?

Question

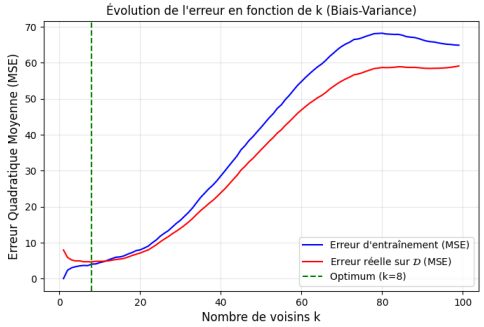
Nous en avons parlé oralement la dernière fois...

Comment piéger le modèle pour mesurer sa vraie performance ?

L'idée : Il faut lui cacher des données !
On simule l'examen en gardant une partie des données secrètes pendant l'entraînement.

Rappel S1 : L'Expérience "Oracle" ($y = x^2 + \epsilon$)

Souvenez-vous : Nous avions simulé un cas où nous connaissions la **Vérité Terrain** (la courbe rouge).



Ce que l'Oracle nous a montré :

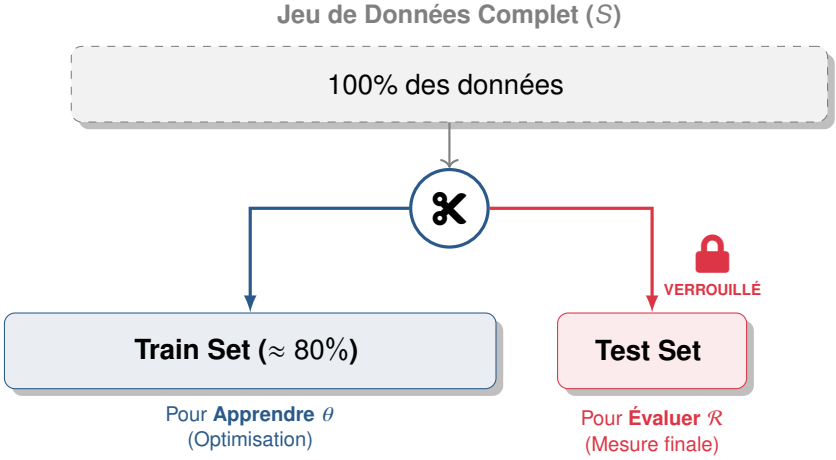
- Le modèle $k = 1$ (très complexe) avait une erreur nulle sur les points bleus...
- ... mais une erreur **gigantesque** sur la courbe rouge (Réalité).

Conclusion

Pour évaluer honnêtement, il faut tester sur des données que le modèle **n'a jamais vues**.

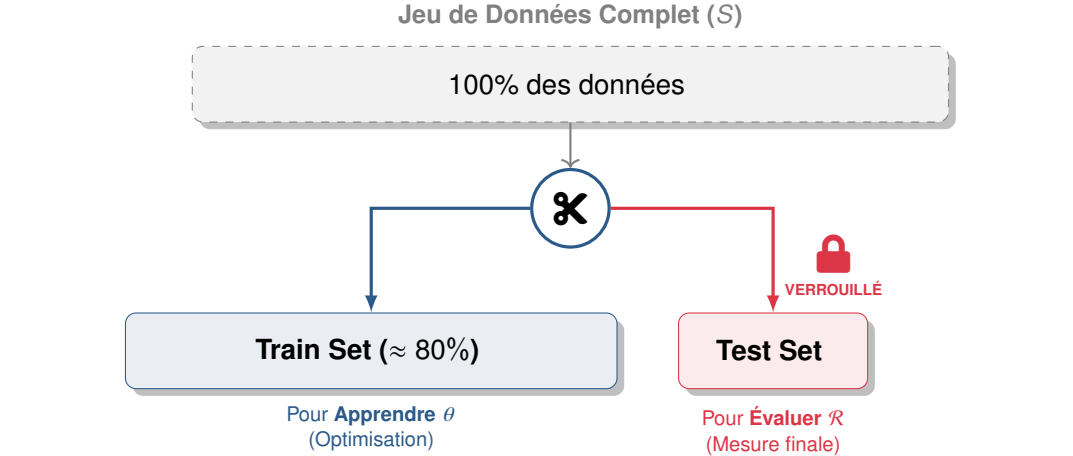
Le Standard Absolu : Séparation Train / Test

✂ **Principe :** Avant tout travail, on isole une partie des données pour l'évaluation finale.



Le Standard Absolu : Séparation Train / Test

✂ **Principe :** Avant tout travail, on isole une partie des données pour l'évaluation finale.



⚠ **Règle d'Or :** Le modèle ne doit **JAMAIS** voir le Test Set pendant son entraînement.

Le Dilemme : Comment régler le curseur k ?

Nous avons notre découpage standard : **Train Set** (80%) et **Test Set** (20%).
Mais le modèle k-NN a besoin qu'on lui fixe k (1, 5, 10, 50 ?).

Protocole proposé :

- **Entraîner** plusieurs versions du modèle ($k = 1, k = 2, \ldots k = 50$) sur le **Train Set**.
- **Calculer** l'erreur de chaque modèle sur le **Test Set**.
- **Sélectionner** le k qui donne la plus petite erreur (le "Champion").
- **Livrer** ce modèle au client avec cette erreur comme garantie.

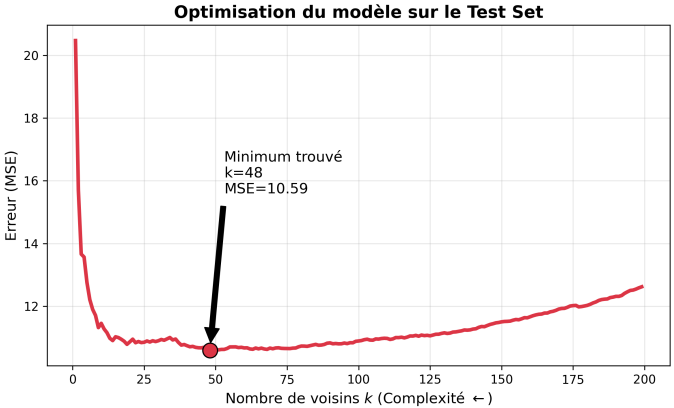
Cette méthode vous semble-t-elle correcte ?

OUI

NON

Expérience : Recherche du meilleur modèle ($y = x^2 + \epsilon$)

Objectif : Trouver l'hyperparamètre k optimal pour notre jeu de données.
Méthode : On teste tous les k et on sélectionne celui qui minimise l'erreur sur le Test Set.

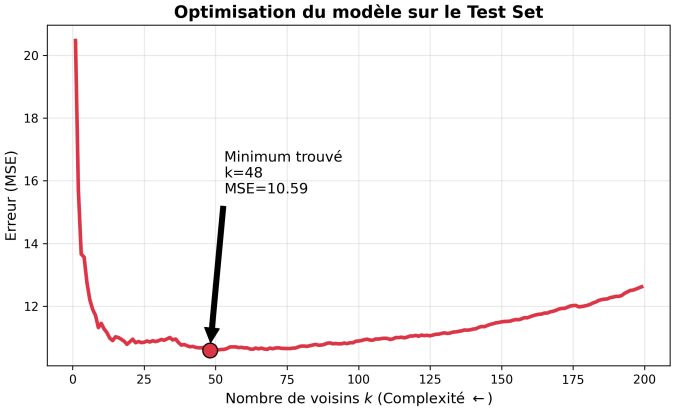


Résultats de la boucle :

- On observe une courbe en U typique.
- **Minimum atteint :** Pour $k = 48$ (par exemple).
- **Erreur minimale :** 10.59.

Expérience : Recherche du meilleur modèle ($y = x^2 + \epsilon$)

Objectif : Trouver l'hyperparamètre k optimal pour notre jeu de données.
Méthode : On teste tous les k et on sélectionne celui qui minimise l'erreur sur le Test Set.



Résultats de la boucle :

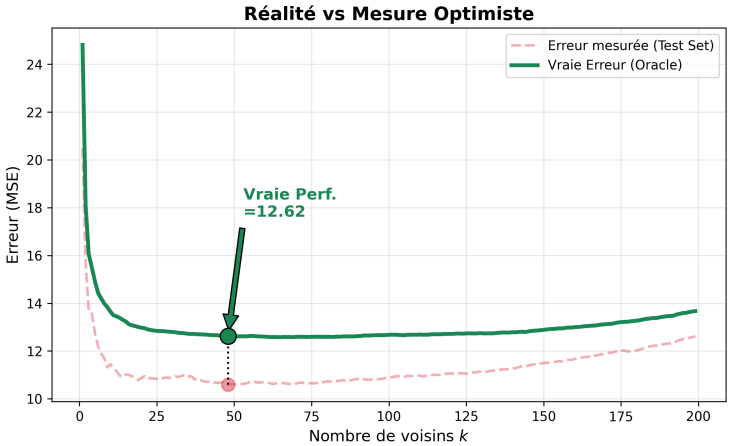
- On observe une courbe en U typique.
- **Minimum atteint :** Pour $k = 48$ (par exemple).
- **Erreur minimale :** 10.59.

✔ **Conclusion apparente** : Nous avons trouvé le meilleur modèle (Erreur garantie : **10.59**).

Le Verdict de l'Oracle : Biais d'Optimisation

Réalité vs Mesure

Comparons le score optimisé avec la **Vraie Erreur** (Oracle).



Analyse de l'échec :

- **Biais Massif :**

  L'écart entre la courbe rose et verte est énorme. Votre score est **faux**.

- **Variance :**

  Le k trouvé (48) n'est même pas le vrai optimal (≈ 65).

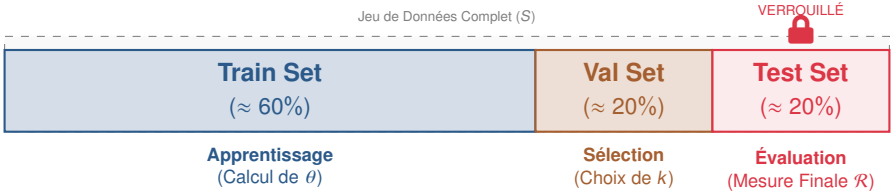
OVERFITTING SUR LE TEST

Le Test Set a servi à choisir k : il n'est plus neutre.

Sommaire

Solution 1 : Le "Hold-Out" (Train-Validation-Test Split)

✂ **Principe :** On divise le jeu de données en **trois** parties disjointes pour protéger le Test Set.



Le Workflow Rigoureux :

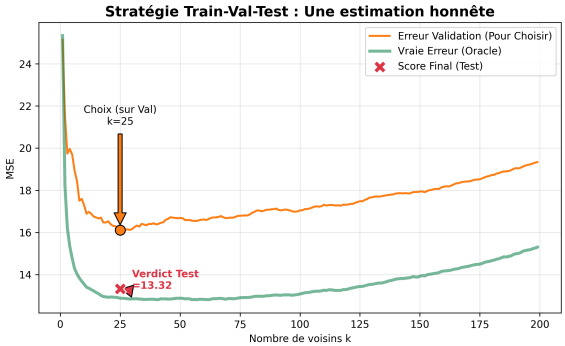
- **Entraîner** plusieurs modèles ($k = 1, k = 5...$) sur le **Train**.
- **Comparer** leurs performances sur la **Validation** et garder le meilleur (k^*).
- **Estimer** l'erreur réelle de ce champion sur le **Test** (une seule fois !).

Le Test Set ne sert jamais à choisir. Il ne sert qu'à constater.

La Preuve par l'Oracle : Enfin une mesure honnête !

✔ Ce qui marche

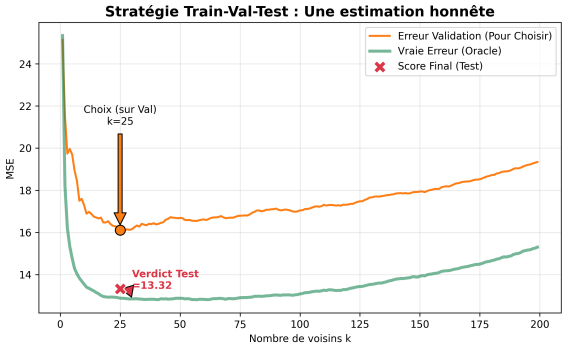
Regardez la croix rouge (**Test**) : Elle est très proche de la ligne verte (**Oracle**).
➔ Le score que nous annonçons au client (13.32) est **VRAI**. Plus de mensonge.



La Preuve par l'Oracle : Enfin une mesure honnête !

✔ Ce qui marche

Regardez la croix rouge (**Test**) : Elle est très proche de la ligne verte (**Oracle**).
➔ Le score que nous annonçons au client (13.32) est **VRAI**. Plus de mensonge.



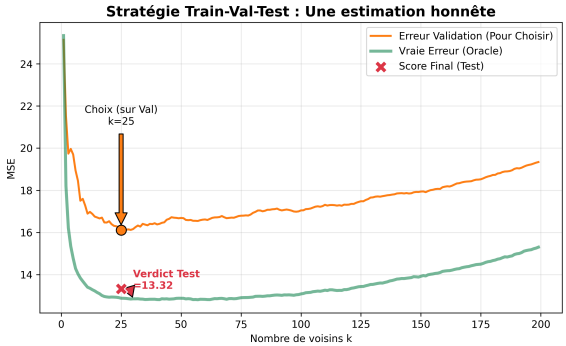
Mais observez bien...

- Le modèle choisi a $k = 25$.
- Or, le vrai optimum de la courbe verte (Oracle) semble être vers $k \approx 65$.

La Preuve par l'Oracle : Enfin une mesure honnête !

✔ Ce qui marche

Regardez la croix rouge (**Test**) : Elle est très proche de la ligne verte (**Oracle**).
➔ Le score que nous annonçons au client (13.32) est **VRAI**. Plus de mensonge.



Mais observez bien...

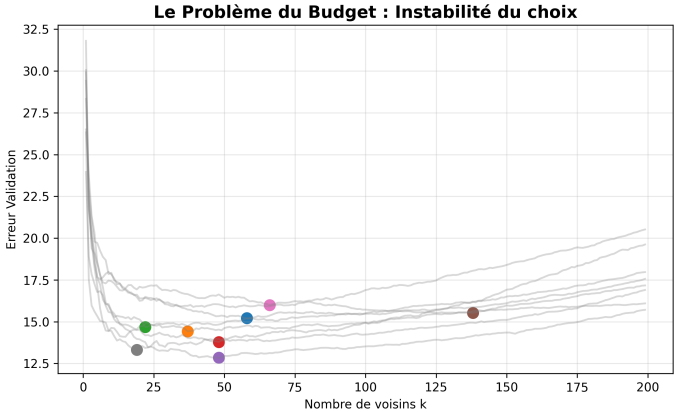
- Le modèle choisi a $k = 25$.
- Or, le vrai optimum de la courbe verte (Oracle) semble être vers $k \approx 65$.

Pourquoi ce décalage ?

En coupant en 3, nous n'avons gardé que **60%** des données pour apprendre.

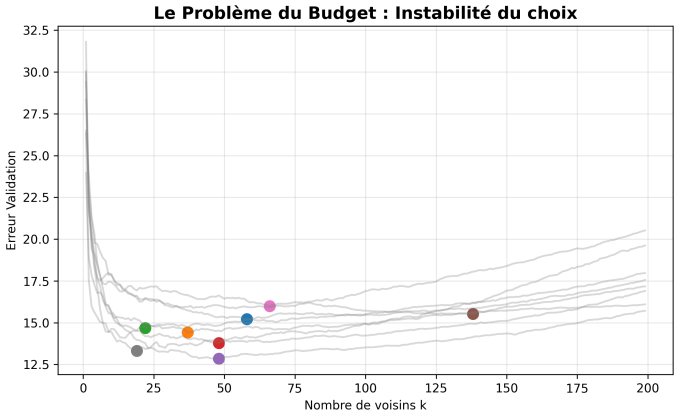
Le Problème du "Petit Budget" : La Loterie

🔀 **Expérience :** Relançons le découpage 8 fois avec des "graines" différentes.
Rappel : Le dataset est petit ($N = 1000$), donc le Validation Set ne fait que **200 points**.



Le Problème du "Petit Budget" : La Loterie

🔀 **Expérience :** Relançons le découpage 8 fois avec des "graines" différentes.
Rappel : Le dataset est petit ($N = 1000$), donc le Validation Set ne fait que **200 points**.

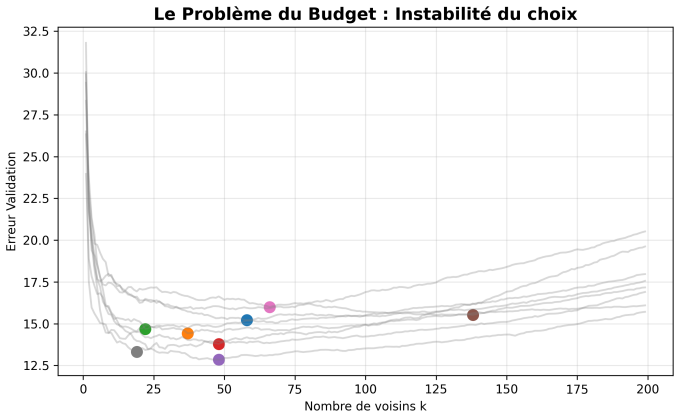


Constat Alarmant :

- Les courbes grises (Validation) partent dans tous les sens.
- **Le choix de k est chaotique :** Tantôt $k = 20$, tantôt $k = 140$...

Le Problème du "Petit Budget" : La Loterie

🔀 **Expérience :** Relançons le découpage 8 fois avec des "graines" différentes.
Rappel : Le dataset est petit ($N = 1000$), donc le Validation Set ne fait que **200 points**.



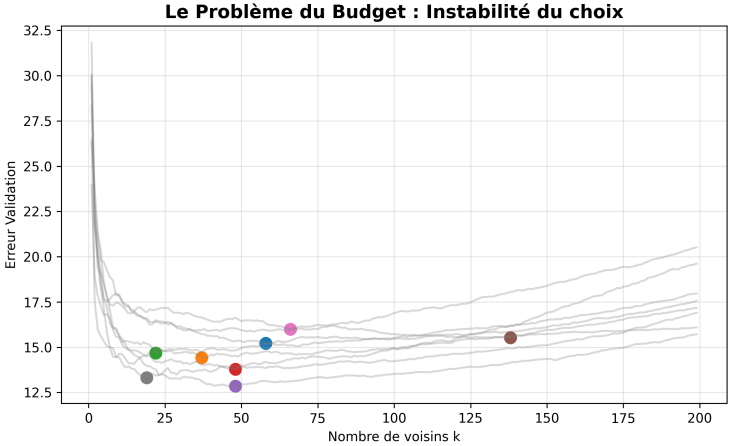
Constat Alarmant :

- Les courbes grises (Validation) partent dans tous les sens.
- **Le choix de k est chaotique :** Tantôt $k = 20$, tantôt $k = 140$...

C'est la "Variance d'Échantillonnage".

Si vous n'avez qu'un seul essai (vie réelle), vous jouez à la roulette russe.

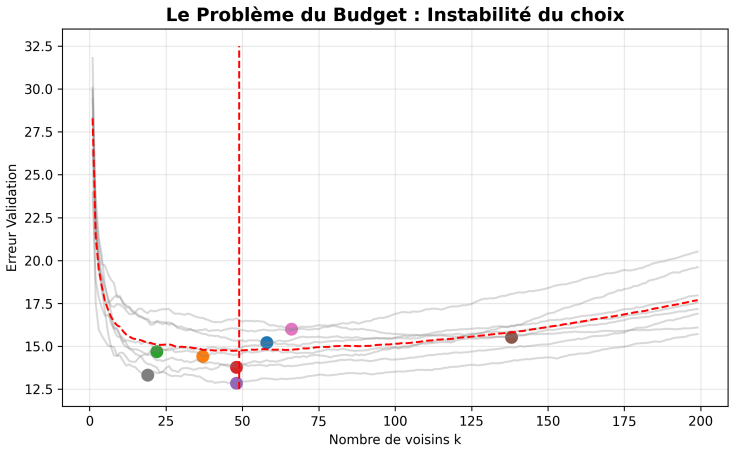
L’Intuition Visuelle : Trouver le Signal dans le Bruit



1 Seul Essai = Hasard

Regardez les lignes grises.
Chaque découpage donne un résultat différent. C’est risqué.

L'Intuition Visuelle : Trouver le Signal dans le Bruit



1 Seul Essai = Hasard

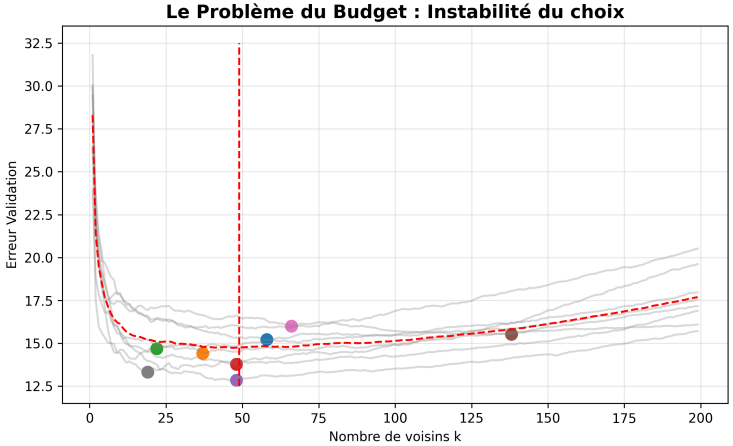
Regardez les lignes grises.
Chaque découpage donne un résultat différent. C'est risqué.

La Solution

Si on fait la **MOYENNE** de tous ces essais...

Le bruit s'annule.
La courbe rouge est stable.

L'Intuition Visuelle : Trouver le Signal dans le Bruit



1 Seul Essai = Hasard

Regardez les lignes grises.
Chaque découpage donne un résultat différent. C'est risqué.

La Solution

Si on fait la **MOYENNE** de tous ces essais...

Le bruit s'annule.
La courbe rouge est stable.

C'est le concept exact de la Validation Croisée.

Solution 2 : La Validation Croisée (L'Algorithme)

L'Idée : On ne gaspille plus de données. Tout le monde passe au tableau !

1. Découper

On divise les 80% de données d'apprentissage en K **blocs** (Folds).

2. Tourner

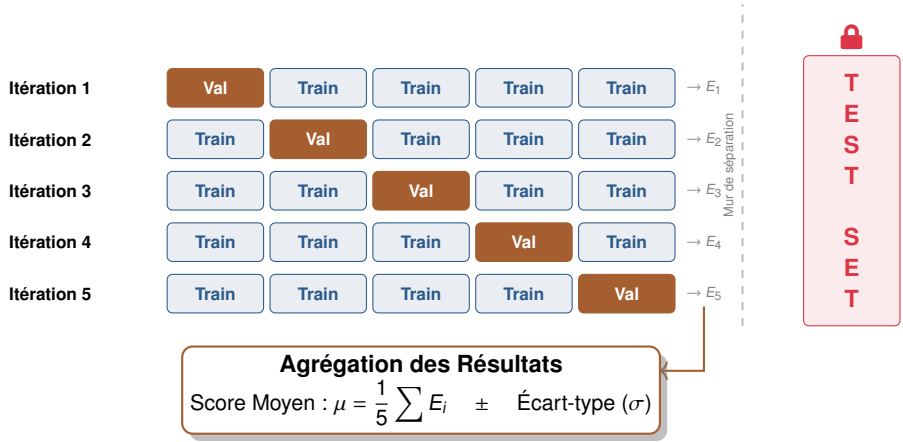
Chacun son tour, un bloc sert de **Validation** et les autres d'**Entraînement**.

3. Moyenner

On calcule la moyenne des K scores pour avoir une note **stable**.

Analogie : C'est comme faire passer 5 examens différents à un élève et faire la moyenne pour connaître son vrai niveau.

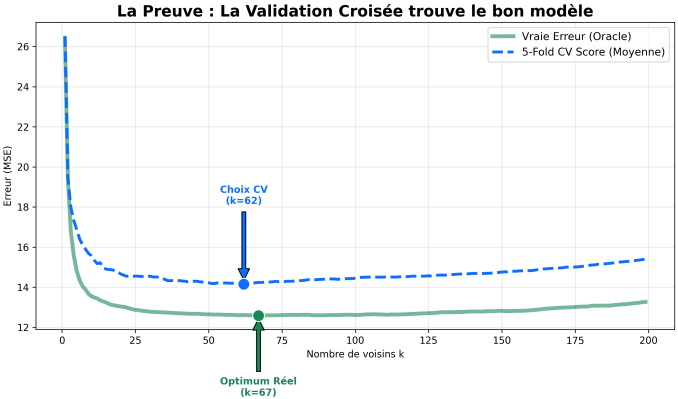
Visualisation : La Mécanique du K-Fold (ex : $K = 5$)



Note : On regarde principalement la **Moyenne** (μ) pour la performance, mais l'**Écart-type** (σ) est crucial pour juger la **stabilité** du modèle (si σ est grand, le modèle est instable).

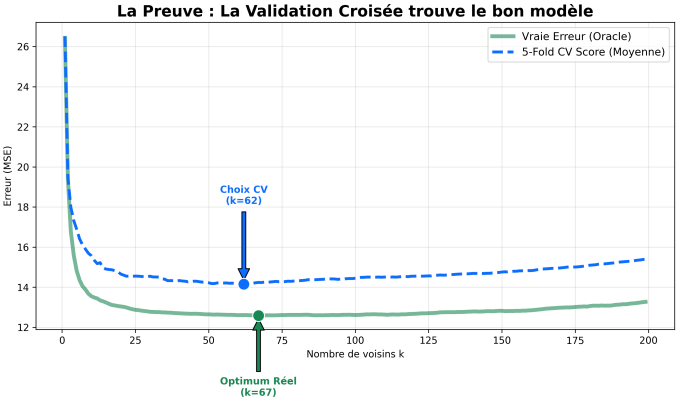
La Preuve : La Validation Croisée dit la Vérité

✔ **Test Final :** Comparons la courbe CV (Bleue) avec la Vraie courbe Oracle (Verte).



La Preuve : La Validation Croisée dit la Vérité

✔ **Test Final :** Comparons la courbe CV (Bleue) avec la Vraie courbe Oracle (Verte).

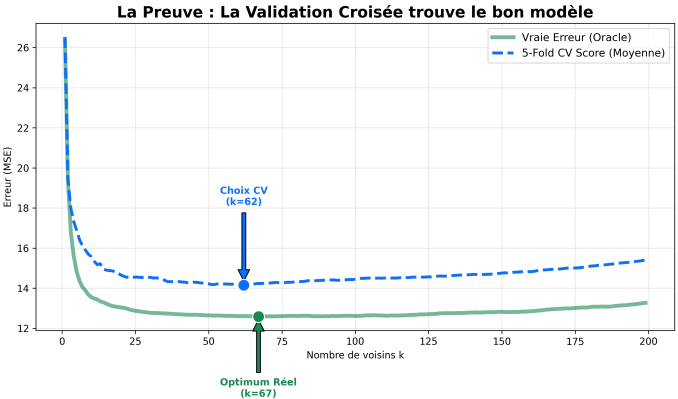


Pourquoi on est contents ?

- **Moins de Bruit :**
  La courbe bleue est **lisse**. En faisant la moyenne, on a éliminé l'instabilité des essais individuels.
- **Proche de l'Optimum :**
  Le minimum trouvé ($k = 62$) est quasi-identique au vrai idéal ($k = 67$).

La Preuve : La Validation Croisée dit la Vérité

✔ **Test Final :** Comparons la courbe CV (Bleue) avec la Vraie courbe Oracle (Verte).



Pourquoi on est contents ?

- **Moins de Bruit :**
  La courbe bleue est **lisse**. En faisant la moyenne, on a éliminé l'instabilité des essais individuels.
- **Proche de l'Optimum :**
  Le minimum trouvé ($k = 62$) est quasi-identique au vrai idéal ($k = 67$).

🏆 **Victoire ! La CV nous a guidés vers le bon modèle sans avoir besoin de l'Oracle.**

Pourquoi la CV est-elle supérieure ? (Et comment régler K)

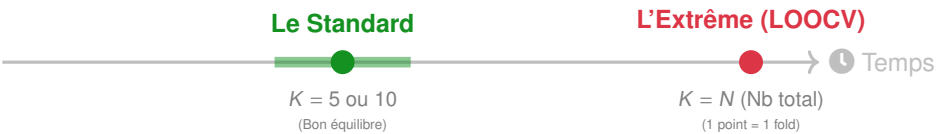
⚖ Stabilité

- La moyenne lisse le bruit.
- *"L'union fait la force."*

♻ Efficience

- 100% des données sont utilisées.
- Idéal pour les petits datasets.

Le Réglage du curseur K (Compromis Temps / Variance)



Règle du pouce : Commencez toujours par $K = 5$.
Si vous avez moins de 50 données, tentez le **Leave-One-Out** ($K = N$).

Bilan : Quelle stratégie d'évaluation choisir ?

Validation Croisée (*K*-Fold)

Le "Standard Industriel"

✔ **À privilégier si :**

- Données < 100k lignes (PME, Tabulaire).
- Vous voulez une mesure **robuste**.
- Le temps de calcul est supportable.

Hold-Out (Train-Val-Test Split)

L'Option "Big Data"

✔ **À privilégier si :**

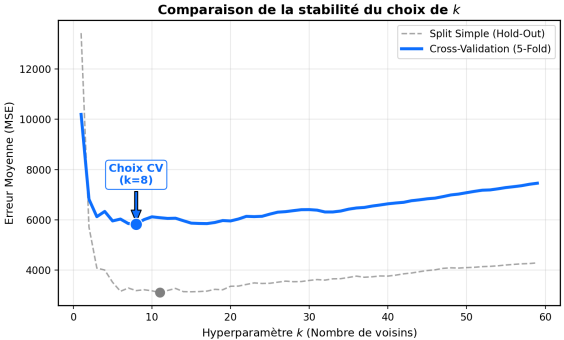
- Données > 1M lignes (Big Data).
- **Deep Learning** (Entraînement très long).
- Vous avez tellement de données que la variance est négligeable.

La Règle d'Or en Entreprise :
Commencez **toujours** par une Validation Croisée (5-Fold).
Ne passez au *Train-Val-Test Split* simple que si votre ordinateur fume !

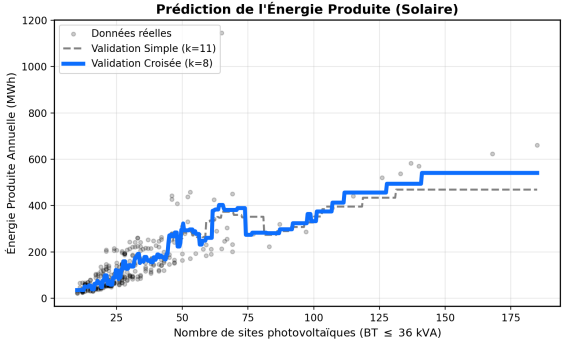
Application 1 : Prédire l'Énergie Solaire ($N = 316$)

Données : 316 installations. On cherche à prédire la production annuelle.

1. Choix de la complexité k



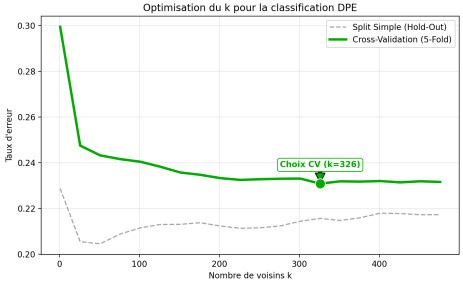
2. Comparaison des Modèles

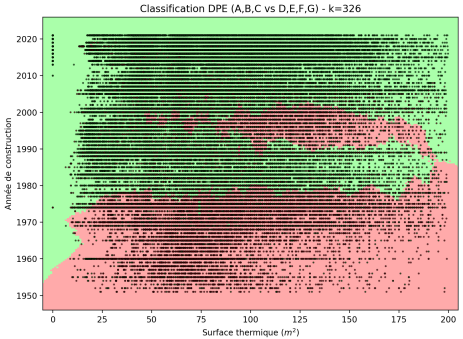


Conclusion : Les deux méthodes valident une zone de complexité similaire ($k \approx 10$).

Application 2 : Classification DPE ($N = 139\,084$)

Données Réelles : Diagnostic de Performance Énergétique (140k logements).





Écart d'estimation

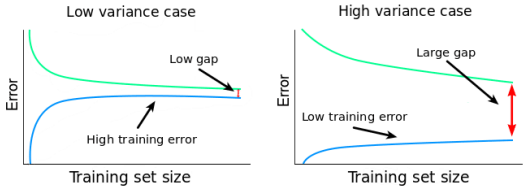
Le **Simple Split** sous-estime l'erreur ($\approx 21\%$). La **CV** remonte l'estimation à $\approx 23.5\%$.
La CV est ici plus conservatrice.

Choix du modèle

Stabilité : La CV impose $k = 326$ (contre $k \approx 50$ pour le split). On privilégie un modèle plus "lisse" pour mieux généraliser sur 140k points.

Que faire si l'entraînement ne fonctionne pas ?

1. Analyser le "Gap"



Diagnostic visuel :

→ Large Gap = Haute Variance (Overfitting)

→ Erreur haute partout = Haut Biais (Underfitting)

2. Choisir le bon levier

Plus de données

Effet : **Diminution de la variance**.

Aide à réduire l'écart entre Train et Val.

Ajuster k

↑ k (Simplifier) → **Diminue la variance**, augmente le biais.

↓ k (Complexifier) → **Diminue le biais**, augmente la variance.

Représentation

Ajouter des *features* ou changer de classe de fonction.

Diminue le biais.

Problème : Haute Variance → Plus de données, Simplifier le modèle (↑ k).

Problème : Haut Biais → Features complexes, Complexifier le modèle (↓ k).

Le Workflow en Code : Maitriser l'optimisation

1. Validation sur bloc fixe (Hold-Out)

```python
# 1. Decoupage en 3 blocs (60/20/20)
X_dev, X_test, y_dev, y_test = \
    train_test_split(X, y, test_size=0.2)
X_train, X_val, y_train, y_val = \
    train_test_split(X_dev, y_dev, test_size=0.25)

# 2. Boucle manuelle sur k
scores = []
for k in range(1, 51):
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train, y_train)
    # Score sur le bloc VAL unique
    scores.append(model.score(X_val, y_val))

best_k = range(1, 51)[np.argmax(scores)]

```

Inconvenient : Le choix de k depend du hasard du decoupage du set de Validation.

2. Validation Croisee (Rotation)

```python
# A. Boucle manuelle avec cross_val_score
scores_cv = []
for k in range(1, 51):
    knn = KNeighborsClassifier(n_neighbors=k)
    # Score moyen sur 5 rotations (Folds)
    sc = cross_val_score(knn, X_dev, y_dev, cv=5)
    scores_cv.append(sc.mean())

# B. Version automatique (GridSearchCV)
params = {'n_neighbors': range(1, 50)}
grid = GridSearchCV(KNeighborsClassifier(),
                    params, cv=5)
grid.fit(X_dev, y_dev)
best_k = grid.best_params_['n_neighbors']

```

Avantage : Le choix de k est beaucoup plus stable et robuste.

Livraison Finale : Le Workflow de l'Ingenieur

1. RE-ENTRAINEMENT

Donnees : Dev Set (100%)
(Train + Validation)

Action : Re-entrainer le modele avec le **k-optimal** sur tout le set de Developpement.

But : Maximiser le signal
On ne "gâche" aucune donnee avant la prod.

2. LE VERDICT

Donnees : Test Set
(Verouillé)

Action : Evaluation **unique** du modele final sur les donnees jamais vues.

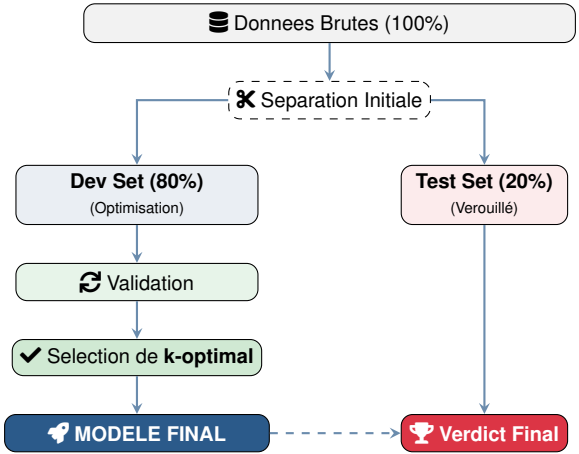
But : Performance Reelle
C'est la mesure de generalisation finale.

LA REGLE D'OR

Interdiction de modifier le k-optimal apres le test.

Si vous changez le modele pour ameliorer ce score → **Triche (Data Leakage)**.

Synthese : Le Pipeline de Validation Ideal



Dev Set : Apprendre et Choisir | **Test Set** : Une seule mesure finale.

Transition : De l'outil a la reflexion

Ou en sommes-nous ?

- Nous avons un modèle et une **pipeline automatique**.
- Elle trouve les **meilleurs hyperparamètres et paramètres** toute seule.

Transition : De l'outil a la reflexion

Ou en sommes-nous ?

- Nous avons un modèle et une **pipeline automatique**.
- Elle trouve les **meilleurs hyperparamètres et paramètres** toute seule.

⚙ La methode est prete... mais le modele est-il le bon ?

Maintenant qu'on sait **comment** valider, on peut tester **n'importe quel** moteur.

Transition : De l'outil a la reflexion

Ou en sommes-nous ?

- Nous avons un modèle et une **pipeline automatique**.
- Elle trouve les **meilleurs hyperparamètres et paramètres** toute seule.

La methode est prete... mais le modele est-il le bon ?

Maintenant qu'on sait **comment** valider, on peut tester **n'importe quel** moteur.

Comment prenez-vous une decision complexe ?

(Ex : Choisir d'aller courir ou non selon la meteo)

"Est-ce qu'il pleut ?"

Transition : De l'outil a la reflexion

Ou en sommes-nous ?

- Nous avons un modèle et une **pipeline automatique**.
- Elle trouve les **meilleurs hyperparamètres et paramètres** toute seule.

La methode est prete... mais le modele est-il le bon ?

Maintenant qu'on sait **comment** valider, on peut tester **n'importe quel** moteur.

Comment prenez-vous une decision complexe ?

(Ex : Choisir d'aller courir ou non selon la meteo)

"Est-ce qu'il pleut ?" → "Ai-je un ami motive ?"

Transition : De l'outil a la reflexion

Ou en sommes-nous ?

- Nous avons un modèle et une **pipeline automatique**.
- Elle trouve les **meilleurs hyperparamètres et paramètres** toute seule.

La methode est prete... mais le modele est-il le bon ?

Maintenant qu'on sait **comment** valider, on peut tester **n'importe quel** moteur.

Comment prenez-vous une decision complexe ?

(Ex : Choisir d'aller courir ou non selon la meteo)

"Est-ce qu'il pleut ?" → "Ai-je un ami motive ?" → "Je reste au chaud."

Transition : De l'outil a la reflexion

Ou en sommes-nous ?

- Nous avons un modèle et une **pipeline automatique**.
- Elle trouve les **meilleurs hyperparamètres et paramètres** toute seule.

La methode est prete... mais le modele est-il le bon ?

Maintenant qu'on sait **comment** valider, on peut tester **n'importe quel** moteur.

Comment prenez-vous une decision complexe ?

(Ex : Choisir d'aller courir ou non selon la meteo)

"Est-ce qu'il pleut ?" → "Ai-je un ami motive ?" → "Je reste au chaud."

Direction : Les Arbres de Decision

Un modele qui nous explique enfin ses choix.

Sommaire

Pourquoi changer de modèle ? (Le besoin d'explication)

Rappel : Le k-NN est performant mais c'est une **"Boîte Noire"**.
Il est difficile d'expliquer à un client pourquoi le modèle a pris telle décision ("C'est la faute des voisins").

Notre nouvel objectif :

- Trouver une fonction $h(\mathbf{x})$ qui soit **lisible** par un humain.
- Obtenir des règles explicites de type :

```
SI (Age > 25) ET (Salaire > 2000)
    ALORS (Crédit = Accordé)
```

L'Arbre de Décision mime le raisonnement humain.

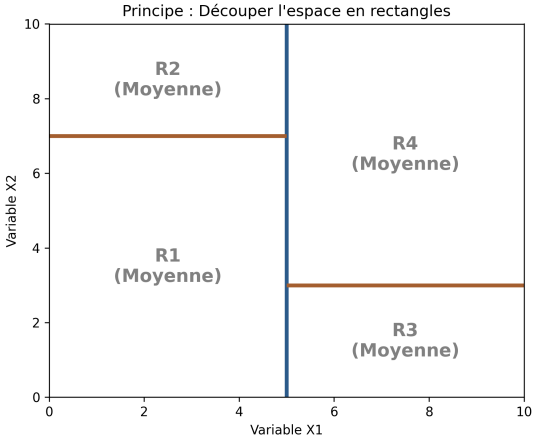
Le Concept Clé : "Diviser pour Régner"

✂ Le Principe

L'idée est de **partitionner** (découper) l'espace des données en **rectangles**.

- On cherche des régions **homogènes** (où les y se ressemblent).
- Dans chaque rectangle R_m, le modèle sera **très simple** (souvent juste la moyenne).

On découpe l'espace complexe en problèmes simples.

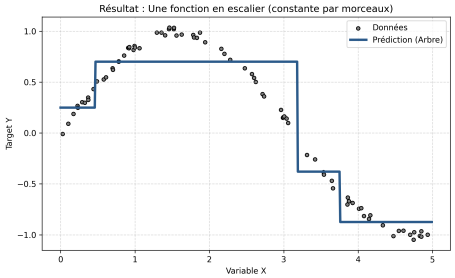


Résultat : Une fonction "en escalier"

Mathématiquement, la fonction de prédiction $\hat{f}(x)$ s'écrit :

$$\hat{f}(x) = \sum_{m=1}^{M} c_m \cdot \mathbb{I}_{R_m}(x)$$

*Où c_m est la **constante** (moyenne) prédite dans la région R_m.*



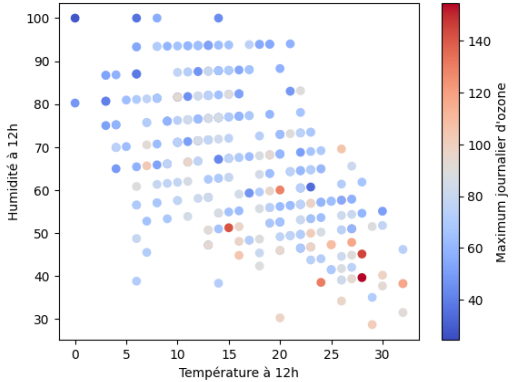
Propriétés du modèle :

- **Discontinu :** La prédiction saute d'une valeur à l'autre aux frontières.
- **Non-linéaire :** Capable de modéliser des comportements complexes par morceaux.
- **Interprétable :** On sait exactement dans quelle "marche" de l'escalier on se trouve.

Sommaire

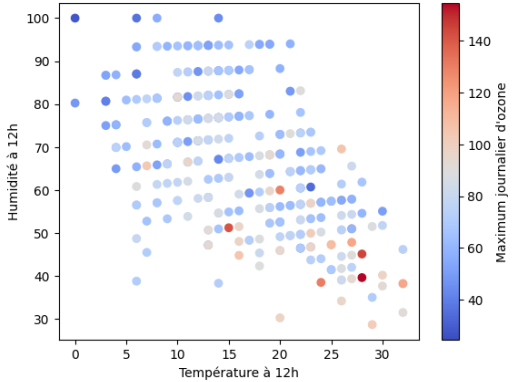
Cas Concret : Prédire la pollution à l'Ozone (Y)

Données : Relevés météo à Mordelles (2023).
But : Prédire le pic d'Ozone journalier (Y) en fonction de Température (X_1) et Humidité (X_2).



Cas Concret : Prédire la pollution à l'Ozone (Y)

Données : Relevés météo à Mordelles (2023).
But : Prédire le pic d'Ozone journalier (Y) en fonction de Température (X_1) et Humidité (X_2).

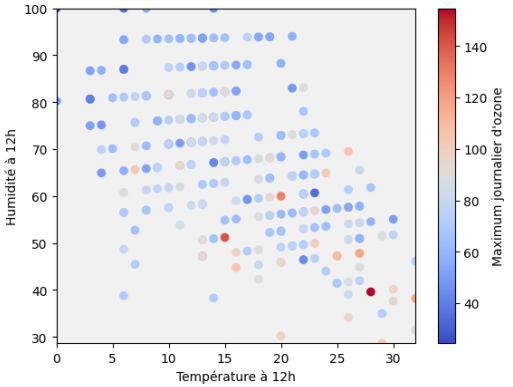


Observez...

- **Rouge** = Forte pollution.
- **Bleu** = Faible pollution.

Question :
Si je ne vous donne aucune info météo, que prédisez-vous ?

Niveau 0 : La "Baseline" (Racine de l'arbre)



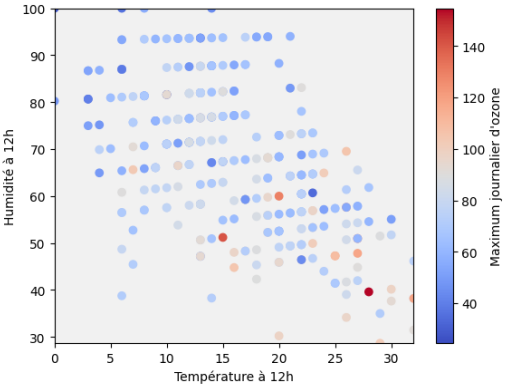
Aucune frontière de décision = Un seul bloc.

Prédiction Unique

$$\hat{y} = \bar{y} = 72.78$$

Moyenne globale
(Minimise l'erreur quadratique)

Niveau 0 : La "Baseline" (Racine de l'arbre)



Aucune frontière de décision = Un seul bloc.

Prédiction Unique

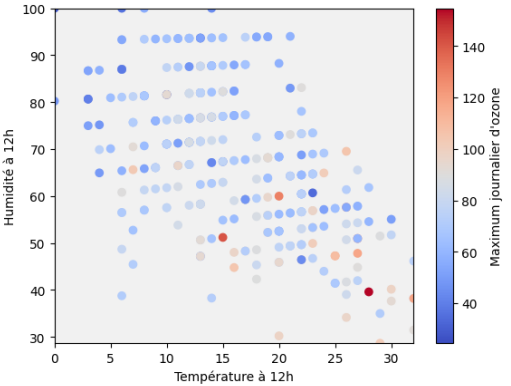
$$\hat{y} = \bar{y} = 72.78$$

Moyenne globale
(Minimise l'erreur quadratique)

Le Problème :

- ✗ Forte diversité (Variance).
- ✗ On mélange les **Rouges** (Pollué) et les **Bleus** (Sain).

Niveau 0 : La "Baseline" (Racine de l'arbre)



Aucune frontière de décision = Un seul bloc.

Prédiction Unique

$$\hat{y} = \bar{y} = 72.78$$

Moyenne globale
(Minimise l'erreur quadratique)

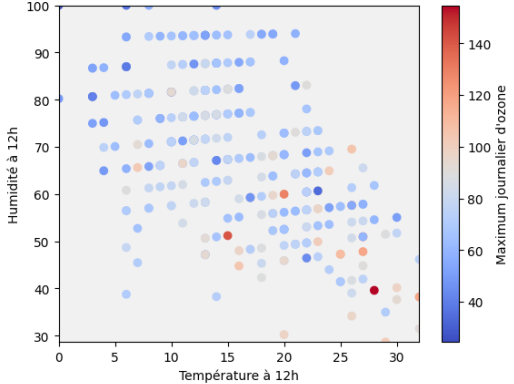
Le Problème :

- ✗ Forte diversité (Variance).
- ✗ On mélange les **Rouges** (Pollué) et les **Bleus** (Sain).

Idée : *Et si on coupait le nuage en deux ?*

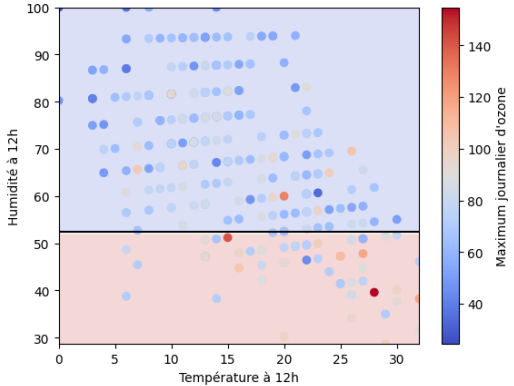
Niveau 1 : La Première Fracture

À vous de jouer : Tracez une ligne (verticale ou horizontale) pour séparer au mieux les Rouges des Bleus.



Niveau 1 : La Première Fracture

À vous de jouer : Tracez une ligne (verticale ou horizontale) pour séparer au mieux les Rouges des Bleus.

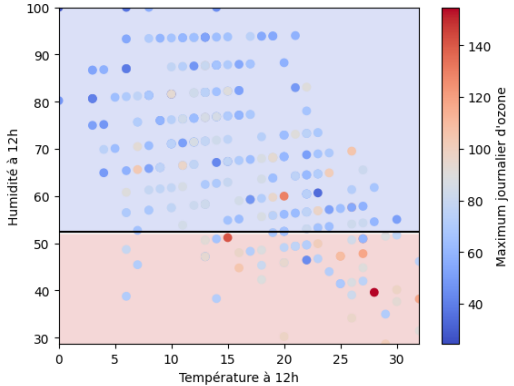


Le Choix de l'Algorithme :

- **Variable :** Humidité (X_2).
- **Seuil :** ≤ 52.4.

Niveau 1 : La Première Fracture

À vous de jouer : Tracez une ligne (verticale ou horizontale) pour séparer au mieux les Rouges des Bleus.



Le Choix de l'Algorithme :

- **Variable :** Humidité (X_2).
- **Seuil :** ≤ 52.4.

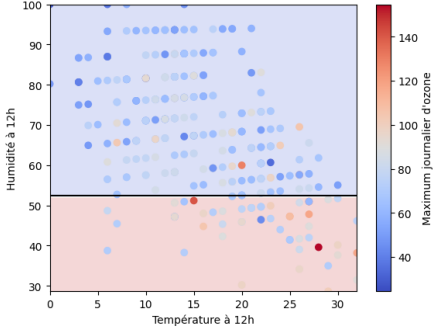
Résultat

Zone du BAS (Sec) :
$\hat{y} = 86.5$ **(Pollué)**

Zone du HAUT (Humide) :
$\hat{y} = 67.0$ **(Sain)**

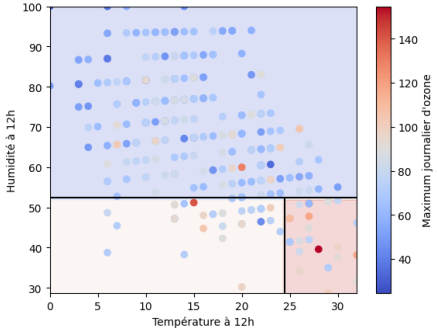
Niveau 2 : Diviser pour mieux régner

Focus : Regardez la zone du **BAS** (Zone "Sèche").
Défi : Quelle autre variable (axe horizontal) permettrait de les isoler ?



Niveau 2 : Diviser pour mieux régner

Focus : Regardez la zone du **BAS** (Zone "Sèche").
Défi : Quelle autre variable (axe horizontal) permettrait de les isoler ?

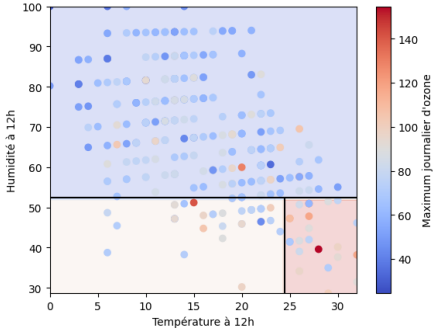


La Solution Algorithmique :

- **Nouvelle Variable :** Température (X_1).
- **Nouveau Seuil :** $\leq 24.5°C$.

Niveau 2 : Diviser pour mieux régner

Focus : Regardez la zone du **BAS** (Zone "Sèche").
Défi : Quelle autre variable (axe horizontal) permettrait de les isoler ?



La Solution Algorithmique :

- **Nouvelle Variable :** Température (X_1).
- **Nouveau Seuil :** $\leq 24.5°C$.

Interprétation Physique

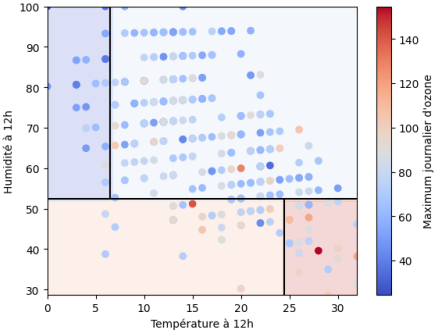
Même si l'air est sec (Condition favorable à la pollution)...
S'il fait froid (< 24.5°C) → La pollution ne se forme pas (Zone Bleue récupérée à gauche).

La Révélation : De l'Espace 2D au Graphe

Une **Zone** du graphique (Rectangle) = Une **"Feuille"** de l'arbre.

*(Ici, visualisons le résultat après une **3ème coupure** supplémentaire).*

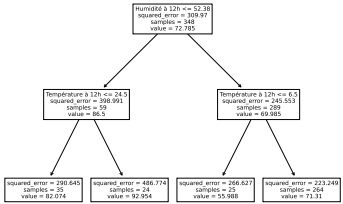
Vision Géométrique



4 rectangles distincts.

⟺

Vision Algorithmique



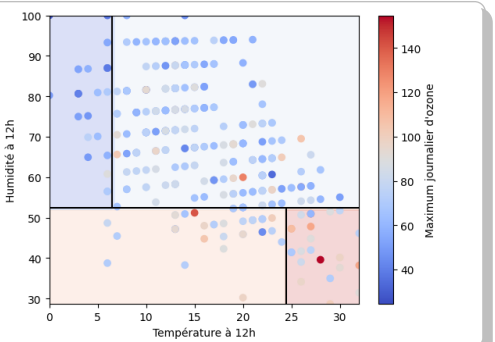
4 feuilles finales.

La Révélation : De l'Espace 2D au Graphe

Une **Zone** du graphique (Rectangle) = Une **"Feuille"** de l'arbre.

*(Ici, visualisons le résultat après une **3ème coupure** supplémentaire).*

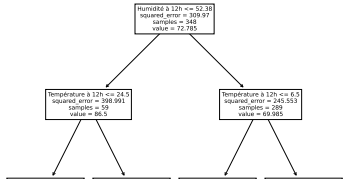
Vision Géométrique



4 rectangles distincts.

⟺

Vision Algorithmique



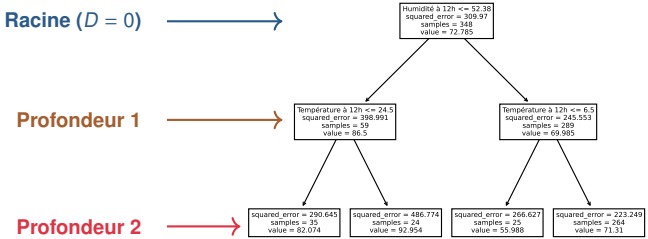
4 feuilles finales.

▶ **Lecture :** Pour prédire un nouveau point, on descend l'arbre en répondant aux questions jusqu'à tomber dans une feuille unique.

Concept Clé : La Profondeur de l'Arbre

Définition

La **Profondeur** est le nombre maximum de questions (splits) posées pour arriver à une décision finale.



Géométrie :

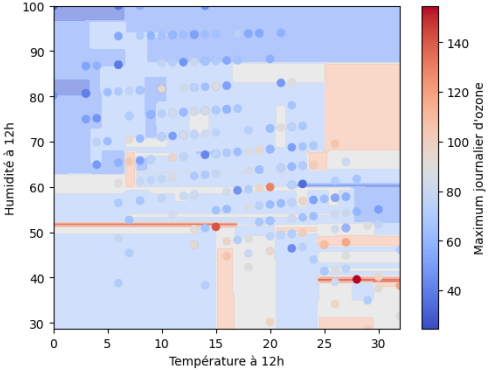
P.1 On coupe le monde en 2.

P.2 On recoupe les morceaux.

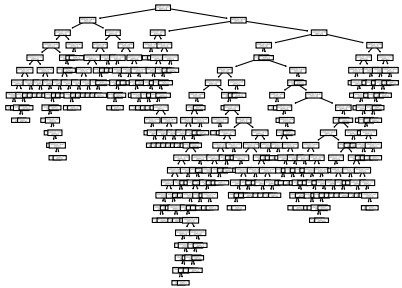
Le Piège : Faut-il continuer indéfiniment ?

⚠ **Question :** Si on laisse l'arbre grandir sans limite, que se passe-t-il ?

Partition "Parfaite" (Profondeur 22)



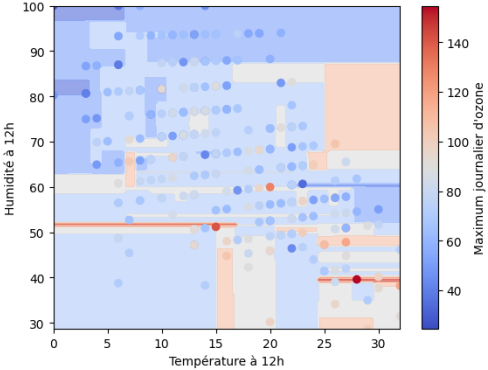
Arbre correspondant



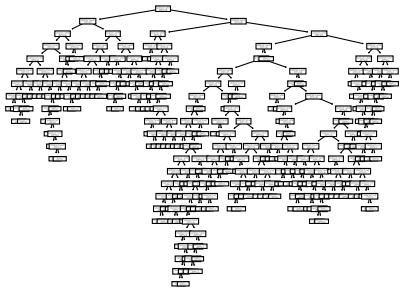
Le Piège : Faut-il continuer indéfiniment ?

⚠ **Question :** Si on laisse l'arbre grandir sans limite, que se passe-t-il ?

Partition "Parfaite" (Profondeur 22)



Arbre correspondant



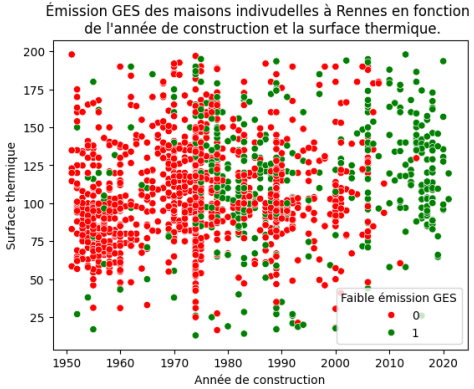
OVERFITTING !

L'arbre a appris le "bruit" des données (chaque point est isolé). Il ne généralisera pas.

➔ *Il faudra l'élaguer (le couper) plus tard.*

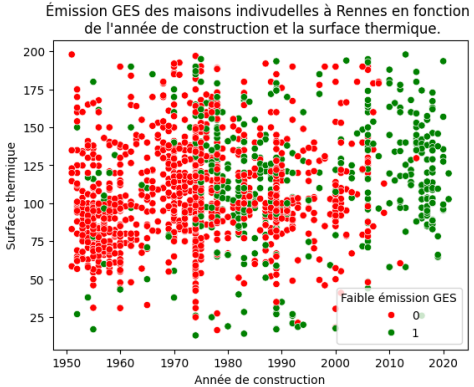
Nouveau Défi : Les émissions de GES à Rennes

Données : Maisons individuelles à Rennes.
Objectif : Classer les maisons en **Faible Émission (Vert)** ou **Forte Émission (Rouge)**.



Nouveau Défi : Les émissions de GES à Rennes

Données : Maisons individuelles à Rennes.
Objectif : Classer les maisons en **Faible Émission (Vert)** ou **Forte Émission (Rouge)**.



Observation

Il y a un mélange de points.
Codage :

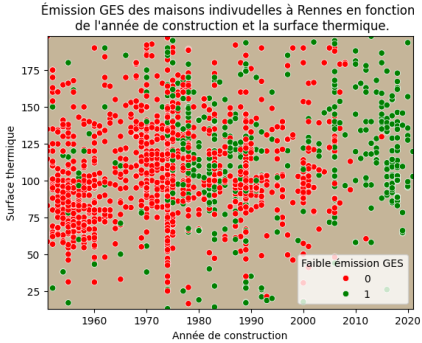
- $y = 1$: Faible (Vert)
- $y = 0$: Forte (Rouge)

Question : Quelle est la probabilité globale d'être "Vert" ?

Étape 0 : L'Arbre est un modèle Probabiliste

Concept Clé

En classification, un arbre ne prédit pas directement une classe.
Il prédit une probabilité (la proportion de la classe majoritaire dans la feuille).



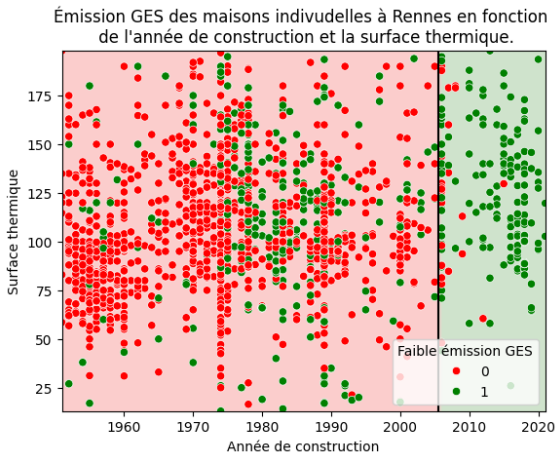
À la Racine (Tout le monde) :

- On compte les Verts (N_1).
- On divise par le total (N).

$$\hat{P}(Y = \text{Vert}) = \frac{N_1}{N} = 0.43$$

→ Si on devait parier, on dirait **Rouge** (car $0.43 < 0.5$).

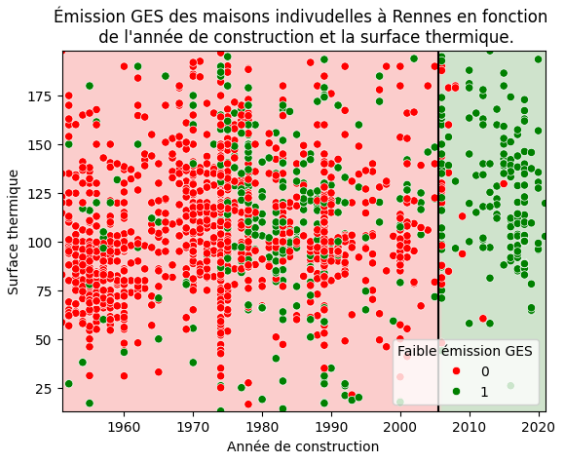
Étape 1 : Affiner la Probabilité par découpage



✂ **Critère de Split :** Année de construction ≤ 2005

Conséquence du Split :
On recalcule la probabilité **localement**.

Étape 1 : Affiner la Probabilité par découpage



✂ **Critère de Split :** Année de construction $\leq$ 2005

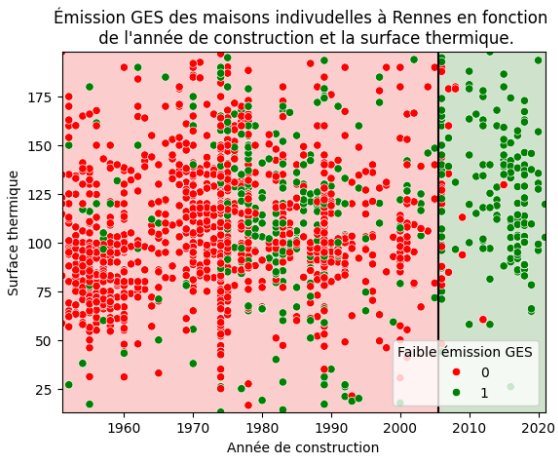
Conséquence du Split :
On recalcule la probabilité **localement**.

← **Zone** $\leq$ 2005

Beaucoup de **Rouges**.

$\hat{P}$(Vert) = **0.23**

Étape 1 : Affiner la Probabilité par découpage



Conséquence du Split :
On recalcule la probabilité **localement**.

← Zone ≤ 2005

Beaucoup de **Rouges**.

$\hat{P}(\text{Vert}) = \mathbf{0.23}$

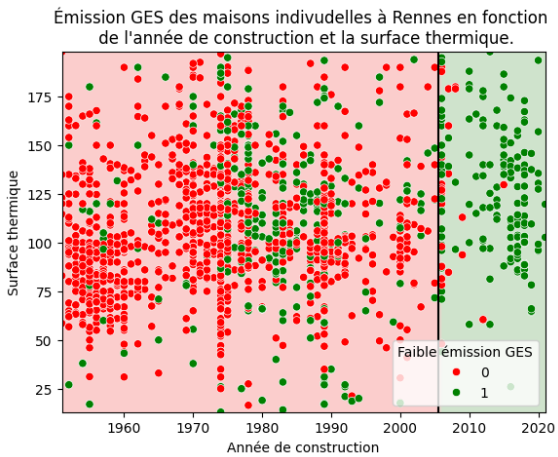
→ Zone > 2005

Beaucoup de **Verts**.

$\hat{P}(\text{Vert}) = \mathbf{0.88}$

⌘ Critère de Split : Année de construction ≤ 2005

Étape 1 : Affiner la Probabilité par découpage



Conséquence du Split :
On recalcule la probabilité **localement**.

← **Zone** ≤ 2005

Beaucoup de **Rouges**.

$\hat{P}(\text{Vert}) = \mathbf{0.23}$

→ **Zone** > 2005

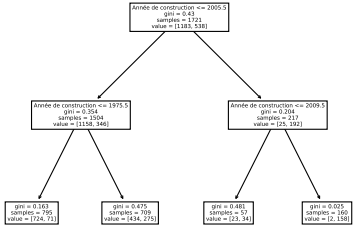
Beaucoup de **Verts**.

$\hat{P}(\text{Vert}) = \mathbf{0.88}$

⌘ **Critère de Split :** Année de construction ≤ 2005

✔ Le modèle gagne en **confiance**.

Lecture Pratique : La "Fiche d'Identité" d'un Nœud



Population (Qui est là ?)

Total

709

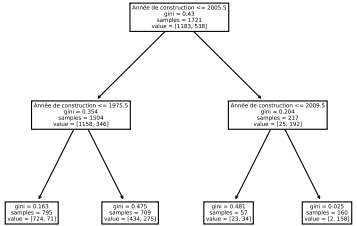
(samples)

434 Rouges

275 Verts

(C'est le vecteur "value")

Lecture Pratique : La "Fiche d'Identité" d'un Nœud



Population (Qui est là ?)

Total

709

(samples)

434 Rouges

275 Verts

(C'est le vecteur "value")

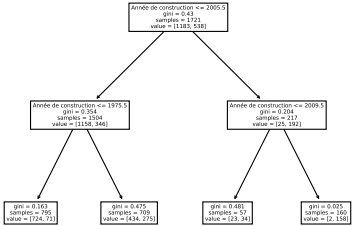
Décision (Qui gagne ?)

On calcule la majorité :

$$\frac{434}{709} = \mathbf{61\%} \quad \text{vs} \quad \frac{275}{709} = 39\%$$

→ **GAGNANT : ROUGE**

Lecture Pratique : La "Fiche d'Identité" d'un Nœud



Population (Qui est là ?)

Total

709

(samples)

434 Rouges

275 Verts

(C'est le vecteur "value")

Décision (Qui gagne ?)

On calcule la majorité :

$$\frac{434}{709} = \mathbf{61\%} \quad \text{vs} \quad \frac{275}{709} = 39\%$$

→ GAGNANT : ROUGE

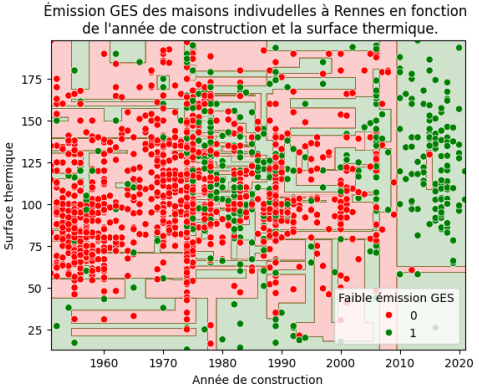
Qualité du nœud : Gini = **0.475**

(0.0 = Parfaitement Pur ↔ Plus c'est haut, plus c'est mélangé).

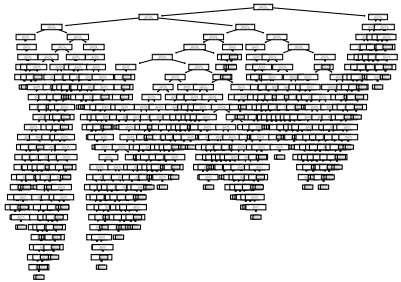
Le Danger : La quête de la pureté absolue

⚠ **Risque :** Si on laisse l'arbre grandir jusqu'à ce que **Gini = 0** partout.

Partition (Prof. 27)



Arbre résultant



Sommaire

Le Défi Algorithmique : Passer de l'Oeil au Code

◎ Rappel de l'objectif

Nous voulons trouver la partition (les rectangles) où les données sont les plus **pures** possibles.
Visuellement, c'était facile. Mais pour l'ordinateur ?

Le Défi Algorithmique : Passer de l'Oeil au Code

◎ Rappel de l'objectif

Nous voulons trouver la partition (les rectangles) où les données sont les plus **pures** possibles.
Visuellement, c'était facile. Mais pour l'ordinateur ?

Idée Naïve (Brute Force) :

"Testons *toutes* les découpes possibles et gardons la meilleure."

Le Défi Algorithmique : Passer de l'Oeil au Code

◎ Rappel de l'objectif

Nous voulons trouver la partition (les rectangles) où les données sont les plus **pures** possibles.
Visuellement, c'était facile. Mais pour l'ordinateur ?

Idée Naïve (Brute Force) :
"Testons *toutes* les découpes possibles et gardons la meilleure."

Problème NP-Complet

Si on a 100 variables et 1000 valeurs...
Le nombre d'arbres possibles dépasse le nombre d'atomes dans l'univers.
Impossible de tout tester.

Le Défi Algorithmique : Passer de l'Oeil au Code

◎ Rappel de l'objectif

Nous voulons trouver la partition (les rectangles) où les données sont les plus **pures** possibles.
Visuellement, c'était facile. Mais pour l'ordinateur ?

Idée Naïve (Brute Force) :

"Testons *toutes* les découpes possibles et gardons la meilleure."

Problème NP-Complet

Si on a 100 variables et 1000 valeurs...
Le nombre d'arbres possibles dépasse le nombre d'atomes dans l'univers.
Impossible de tout tester.

Solution : Glouton (Greedy)

On ne cherche pas l'arbre parfait absolu.
On fait le meilleur choix immédiat (étape par étape), sans revenir en arrière.

L'Algorithme CART : Le "Casting" des Coupures

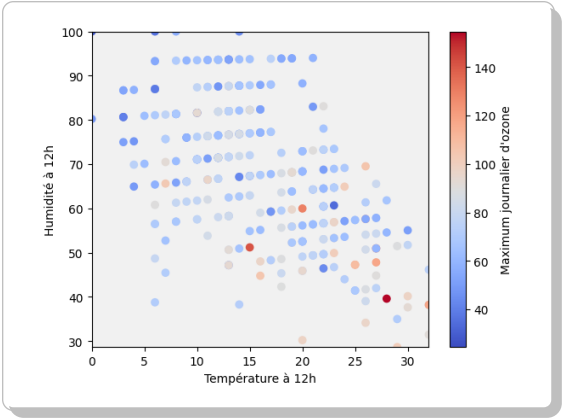
(Classification And Regression Trees)

</> La Boucle Gloutonne

Pour chaque nœud, l'algo teste **TOUT** :

- **Variable j :**
  Est-ce que je prends la Température ou l'Humidité ?
- **Seuil s :**
  Est-ce que je coupe à 10, 20, 24.5 ... ?
- **Score :**
  Calcul du Gain pour ce couple (j, s).

→ On garde le **Champion**.



Phase 1 : L'algorithme scanne ce nuage de points à la recherche de la meilleure coupure.

L'Algorithme CART : Le "Casting" des Coupures

(Classification And Regression Trees)

</> La Boucle Gloutonne

Pour chaque nœud, l'algo teste **TOUT** :

- **Variable j :**
  Est-ce que je prends la Température ou l'Humidité ?
- **Seuil s :**
  Est-ce que je coupe à 10, 20, 24.5 ... ?
- **Score :**
  Calcul du Gain pour ce couple (j, s).

→ On garde le **Champion**.

Le Grand Tournoi (j, s)

Candidat 1

Couple : (Température, 10°C)
→ Gain = 0.05 (Bof, ça sépare mal)

Candidat 2

Couple : (Température, 24.5°C)
→ Gain = 0.18 (Mieux, mais pas top)

Candidat 3 (Vainqueur)

Couple : (Humidité, 52.4%)
Gain = 0.45 (Maximal !)

L'ordi fait ce test pour **toutes** les variables et **tous** les seuils possibles (des milliers de fois).

Le Critère de Choix : Le Gain d'Information

Question : C'est quoi une "bonne" coupure ?
→ C'est une coupure qui **réduit l'Impureté**.

$$\text{Gain} = \underbrace{\text{Impureté}(\textit{Parent})}_{\text{Chaos avant}} - \underbrace{\left(\frac{N_G}{N}\text{Imp}(G) + \frac{N_D}{N}\text{Imp}(D)\right)}_{\text{Chaos moyen après}}$$

Avant (Parent)

Imp = 0.5

→

Après (Enfants)

Imp = 0.1 Imp = 0.2

On cherche la coupure qui maximise cette différence (Le "Nettoyage").

Formalisation (1/3) : L'Arbre est une Fonction

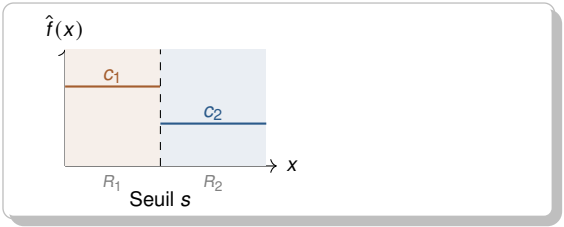
Le Modèle Mathématique $\hat{f}(x)$

L'arbre découpe l'espace en M régions $R_1, \ldots, R_M$.
Le modèle est une **somme pondérée d'indicatrices** :

$$\hat{f}(x) = \sum_{m=1}^{M} c_m \cdot \mathbb{1}(x \in R_m)$$

Légende :

- R_m : La zone (feuille/rectangle).
- $\mathbb{1}(\ldots)$: Vaut 1 si x est dans la zone, 0 sinon.
- c_m : La valeur prédite (constante).



Fonction constante par morceaux.

Formalisation (2/3) : Que valent les constantes c_m ?

Question : Une fois dans le rectangle R_m, quelle valeur prédire ?

Cas Régression

On calcule la **Moyenne** des points :

$$c_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$$

(Minimise l'erreur quadratique).

Cas Classification

On fait un **Vote à la Majorité** :

$$c_m = \arg\max_k \left(\sum_{x_i \in R_m} \mathbb{1}(y_i = k) \right)$$

(Classe la plus fréquente).

Formalisation (2/3) : Que valent les constantes c_m ?

Question : Une fois dans le rectangle R_m, quelle valeur prédire ?

Cas Régression

On calcule la **Moyenne** des points :

$$c_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$$

(Minimise l'erreur quadratique).

Cas Classification

On fait un **Vote à la Majorité** :

$$c_m = \arg\max_k \left(\sum_{x_i \in R_m} \mathbb{1}(y_i = k) \right)$$

(Classe la plus fréquente).

L'Objectif de l'Algorithme

Trouver la partition $\{R_m\}$ qui minimise l'impureté globale :

$$\mathcal{E} = \sum_{m=1}^{M} \text{Impureté}(R_m)$$

Formalisation (3/3) : Le Critère de Division (Le Gain)

L'équation du Gain d'Information Δ

$$\Delta(j, s) = \underbrace{I(R_p)}_{\text{Chaos Avant}} - \left(\underbrace{\frac{N_G}{N_p} I(R_G) + \frac{N_D}{N_p} I(R_D)}_{\text{Chaos Moyen Après}} \right)$$

Notations :

- N_p : Total points (Parent).
- N_G, N_D : Points à Gauche / Droite.
- $I(R)$: Fonction **inconnue** (le désordre).

Objectif

Maximiser $\Delta(j, s)$

(Trouver le plus grand "nettoyage").

Formalisation (3/3) : Le Critère de Division (Le Gain)

L'équation du Gain d'Information Δ

$$\Delta(j, s) = \underbrace{I(R_p)}_{\text{Chaos Avant}} - \underbrace{\left(\frac{N_G}{N_p} I(R_G) + \frac{N_D}{N_p} I(R_D) \right)}_{\text{Chaos Moyen Après}}$$

Notations :

- N_p : Total points (Parent).
- N_G, N_D : Points à Gauche / Droite.
- $I(R)$: Fonction **inconnue** (le désordre).

Objectif

Maximiser $\Delta(j, s)$
(Trouver le plus grand "nettoyage").

À vous de jouer :

Si vous deviez inventer une formule pour mesurer le "Désordre" ($I(R)$)... que proposeriez-vous ?
(Indice : Cela dépend si on fait de la Régression ou de la Classification...)

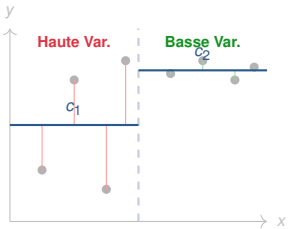
Mesurer le Désordre (A) : La Variance

Usage : Régression (Variables continues). On cherche à minimiser la dispersion autour de la moyenne $\bar{y}_m$.

Formule de la Variance

$$I(R_m) = \frac{1}{N_m} \sum_{i \in R_m} (y_i - c_m)^2$$

Rappel : c_m est la moyenne du nœud.



Intuition :
Plus les points sont "serrés" dans le rectangle, plus la prédiction est stable.

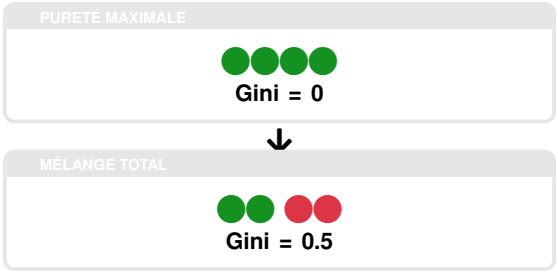
Mesurer le Désordre (B) : L'Indice de Gini

Origine : Mesure d'inégalité (revenus). 0 = Égalité totale (pureté) → 1 = Inégalité maximale (mélange).

Formule de Gini

$$I(R_m) = 1 - \sum_{k=1}^{K} p_{mk}^2$$

Avec $p_{mk} = \frac{1}{N_m} \sum_{i | x_i \in R_m} \mathbf{1}_{\{y_i = k\}}$



Intuition : Le Gini est maximal quand les classes sont à 50/50. L'algorithme cherche à "vider" les classes minoritaires pour s'approcher de 0.

L'Alternative : L'Entropie de Shannon

Mesure du Désordre

$$H(R_m) = -\sum_{k=1}^{K} p_{mk} \log_2(p_{mk})$$

Avec la proportion de classe k :

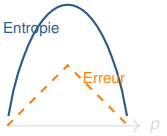
$$p_{mk} = \frac{1}{N_m} \sum_{i:x_i \in R_m} \mathbf{1}(y_i = k)$$

- **H = 0** : Homogénéité totale (Ordre).
- **H = 1** : Chaos (Mélange parfait 50/50).

Pourquoi pas l'erreur ?

On n'utilise pas le simple **Taux d'Erreur** car il est moins **sensible**.

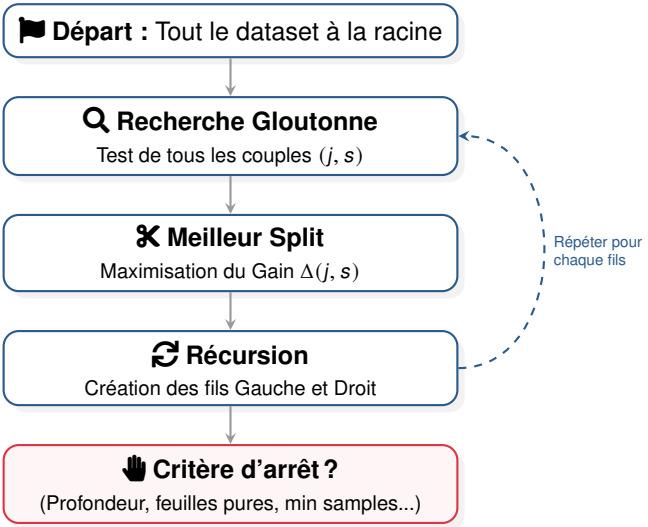
Gini et l'Entropie favorisent les divisions qui créent des **nœuds purs**, même si le taux d'erreur global ne change pas immédiatement.



L'entropie est "plus stricte" (courbe bombée) que l'erreur.

En résumé : On cherche des critères "différentiables" et sensibles qui poussent l'algorithme à isoler des groupes 100% purs le plus vite possible.

Synthèse : L'Algorithme CART en 5 étapes



Sommaire

Comment arrêter la croissance ? (Pré-Élagage)

Problème : Sans limites, l'algorithme CART continue jusqu'à ce que chaque feuille soit pure ($Gini = 0$). C'est le chemin direct vers l'overfitting.

Les Paramètres de Contrôle :

- **Profondeur Max :** Limite le nombre de questions successives.
- **Min Samples Split :** Nb minimum d'individus pour autoriser une coupure.
- **Min Samples Leaf :** Nb minimum d'individus par feuille finale.

Pourquoi limiter ?

Si une feuille ne contient que 2 ou 3 individus :

- L'estimation statistique est **bruitée**.
- La variance de la prédiction explose.
- On capture des particularités du dataset, pas la loi générale.

Alternative : Utiliser un **ensemble de validation**. On arrête de couper quand l'erreur de validation ne baisse plus.

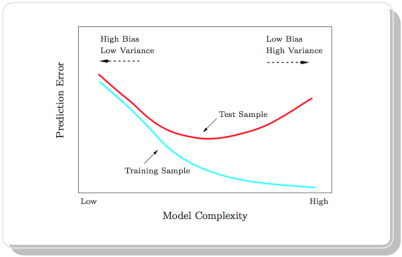
Le Cœur du Problème : Le Compromis Biais-Variance

⚖️ *L'enjeu est de trouver le curseur idéal entre simplicité et complexité.*

❄️ Arbre Court

(Sous-apprentissage)

+ **Biais Élevé**
  Trop simpliste.
− **Variance Faible**
  Très stable.



🔥 Arbre Profond

(Sur-apprentissage)

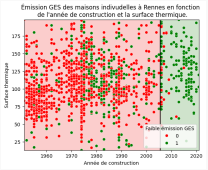
− **Biais Faible**
  Très précis.
+ **Variance Forte**
  Instable.

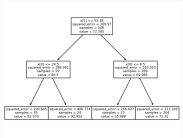
Zone Optimale

💡 **Conclusion :** La profondeur de l'arbre est le levier principal pour **minimiser l'erreur totale** (Biais^2 + Variance).

Illustration : L'Arbre "Simple" vs "Le Monstre"

Profondeur 1 (Stable)

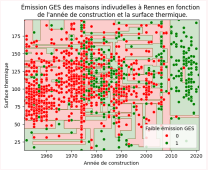




Sous-apprentissage

La règle est trop rudimentaire.

Profondeur 27 (Instable)



Sur-apprentissage

L'arbre apprend le bruit par cœur.

Optimiser la taille : L'Élagage Coût-Complexité

🌲 Phase 1 : Expansion

On laisse l'arbre grandir au maximum ($\mathbb{T}_0$) jusqu'à pureté totale.

→

✂ Phase 2 : Élagage

On coupe les branches "coûteuses" pour minimiser le critère C_α.

Optimiser la taille : L'Élagage Coût-Complexité

Phase 1 : Expansion

On laisse l'arbre grandir au maximum ($\mathbb{T}_0$) jusqu'à pureté totale.

→

Phase 2 : Élagage

On coupe les branches "coûteuses" pour minimiser le critère C_α.

Le Critère à Minimiser

$$C_\alpha(\mathbb{T}) = \underbrace{\text{Erreur}(\mathbb{T})}_{\text{Performance}} + \underbrace{\alpha}_{\text{Pénalité}} \cdot \underbrace{|\mathbb{T}|}_{\text{Taille}}$$

Le paramètre de réglage α :

C'est un **curseur** qui contrôle l'agressivité de la taille.

- Si $\alpha = 0$: On garde l'arbre géant (Risque de sur-apprentissage).
- Si $\alpha \to \infty$: On coupe tout jusqu'à la racine (Risque de sous-apprentissage).

Optimiser la taille : L'Élagage Coût-Complexité

Phase 1 : Expansion

On laisse l'arbre grandir au maximum ($\mathbb{T}_0$) jusqu'à pureté totale.

→

Phase 2 : Élagage

On coupe les branches "coûteuses" pour minimiser le critère C_α.

Le Critère à Minimiser

$$C_\alpha(\mathbb{T}) = \underbrace{\text{Erreur}(\mathbb{T})}_{\text{Performance}} + \underbrace{\alpha}_{\text{Pénalité}} \cdot \underbrace{|\mathbb{T}|}_{\text{Taille}}$$

Le paramètre de réglage α :

C'est un **curseur** qui contrôle l'agressivité de la taille.

- Si $\alpha = 0$: On garde l'arbre géant (Risque de sur-apprentissage).
- Si $\alpha \to \infty$: On coupe tout jusqu'à la racine (Risque de sous-apprentissage).

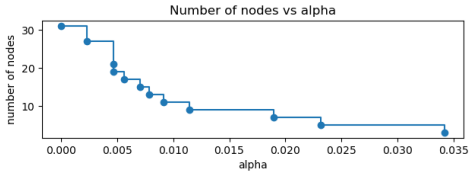
L'OBJECTIF

Trouver le α optimal par **Validation Croisée** pour avoir le meilleur compromis.

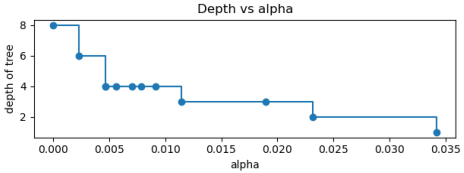
Impact de α sur la structure de l'arbre

Plus α augmente, plus la pénalité est forte, plus l'arbre est "raboté".

Évolution du nombre de nœuds



Évolution de la profondeur



- **À gauche :** On voit que le nombre de nœuds chute brutalement dès que α s'éloigne de 0.
- **À droite :** La profondeur diminue par paliers.
- **Conclusion :** α permet de naviguer de façon continue dans l'espace des modèles, de l'arbre "monstre" à l'arbre racine.

Exemples d'élagage par validation croisée

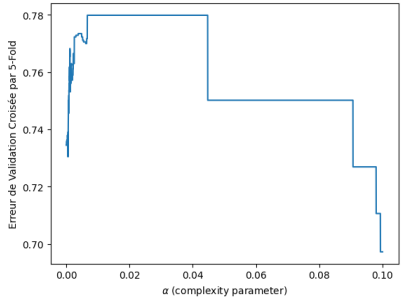


Figure – Évolution de l'erreur par validation croisée 5-Fold en fonction de α

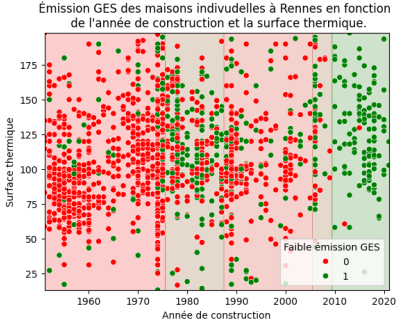


Figure – Visualisation de la règle de décision du meilleur arbre obtenu.

- Arbre **peu profond** et décision sur une **seule variable**.
- Pas de **surajustement** mais la deuxième variable est inutile. Il serait intéressant de créer/trouver de nouvelles **variables** plus **complexes**/**corrélés** avec la réponse.

Exemples d'élagage par validation croisée

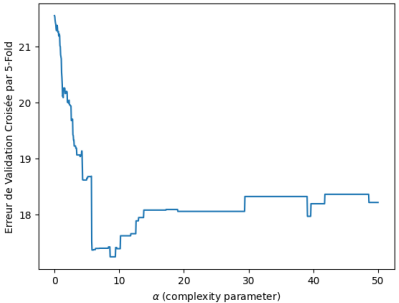


Figure – Évolution de l'erreur par validation croisée 5-Fold en fonction de α

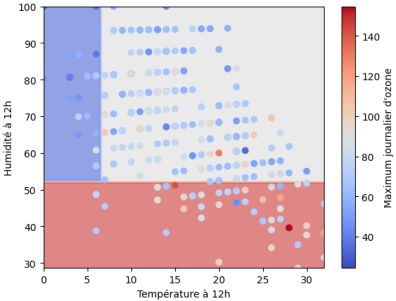


Figure – Visualisation de la règle de décision du meilleur arbre obtenu.

- Arbre **peu profond** mais décision sur les 2 **variables**.
- Pas de **surajustement** mais il faudrait sûrement ici **rajouter des variables** pour diminuer le biais.

Synthèse : Pourquoi utiliser les Arbres ?

Avantages (Modèle "Boîte Blanche")

- **Non-paramétrique :** Capture des relations **non linéaires** (proche des PPV).
- **Intuitif :** Mimique la prise de décision humaine.
- **Robuste :**
  - Pas besoin de normalisation ou de transformations marginales.
  - Peu sensible aux **outliers**.
- **Flexible :** Gère nativement les variables **catégorielles** (> 2 modalités) et les **valeurs manquantes** (catégorie "missing").

Limites et Défis

- **Performance :** Souvent inférieure aux algos complexes (règles trop simples, décisions "non lisses").
- **Instabilité (Variance) :** Une petite modification des données peut changer tout l'arbre.
- **Le Dilemme :**
  - Profond = Forte Variance.
  - Peu profond = Fort Biais.

Synthèse : Pourquoi utiliser les Arbres ?

Avantages (Modèle "Boîte Blanche")

- **Non-paramétrique :** Capture des relations **non linéaires** (proche des PPV).
- **Intuitif :** Mimique la prise de décision humaine.
- **Robuste :**
  - Pas besoin de normalisation ou de transformations marginales.
  - Peu sensible aux **outliers**.
- **Flexible :** Gère nativement les variables **catégorielles** (> 2 modalités) et les **valeurs manquantes** (catégorie "missing").

Limites et Défis

- **Performance :** Souvent inférieure aux algos complexes (règles trop simples, décisions "non lisses").
- **Instabilité (Variance) :** Une petite modification des données peut changer tout l'arbre.
- **Le Dilemme :**
  - Profond = Forte Variance.
  - Peu profond = Fort Biais.

Vers la suite : Comment garder la puissance des arbres sans leur instabilité ?

Voir le Code Complet du Cours

Références Bibliographiques I

T. Hastie, R. Tibshirani, J. Friedman (2009).
The Elements of Statistical Learning : Data Mining, Inference, and Prediction.
Springer Series in Statistics. 2nd Edition.

S. Shalev-Shwartz, S. Ben-David (2014).
Understanding Machine Learning : From Theory to Algorithms.
Cambridge University Press.

Scikit-Learn Developers.
User Guide : Machine Learning in Python.
Documentation officielle : `https://scikit-learn.org`

L. Breiman, J. Friedman, R. Olshen, C. Stone (1984).
Classification and Regression Trees.
Wadsworth International Group.