Introduction à la Science des données
Apprentissage Supervisé

V. Bertret

[1] Université de Rennes 1/UFR Mathématiques

Outline

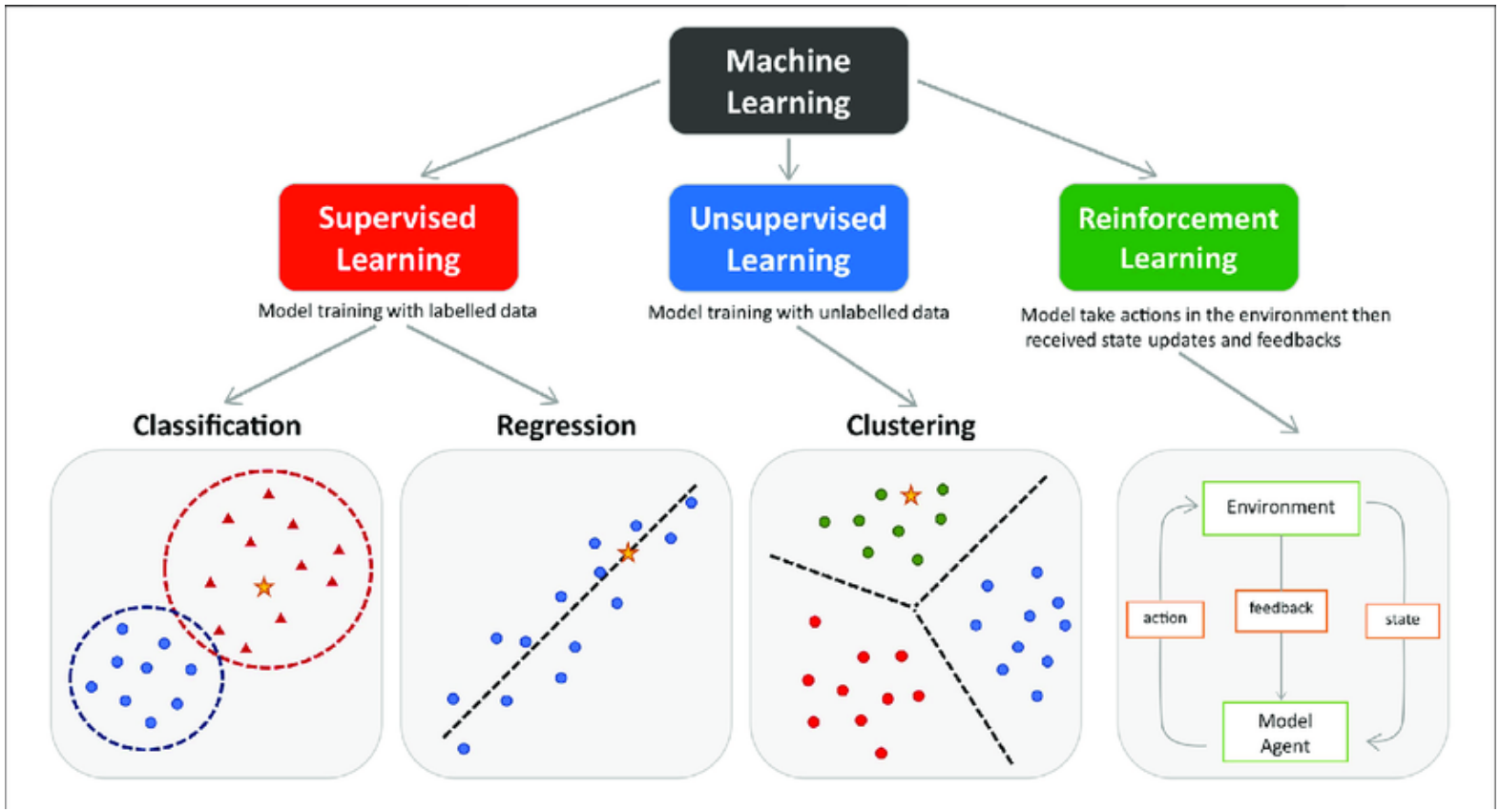
Apprentissage supervisé

Méthode 1 : k-nearest neighbors (k plus proches voisins)

Choix des hyperparamètres

Validation

Qu’est-ce que le "machine learning" (apprentissage machine) ?



Zoom sur l'apprentissage supervisé

Apprentissage supervisée : Apprendre à partir de données **labélisées/étiquetées**.

Jeu de données

- Données d'entrées (X)
- Données de sorties (Y)

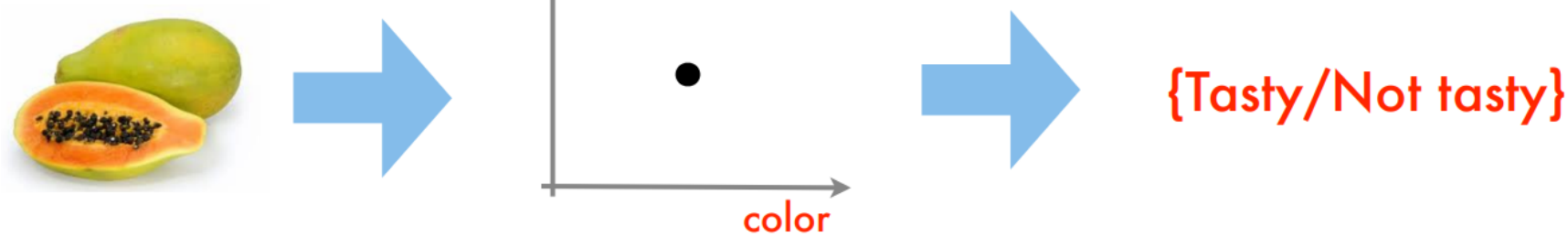
Objectif : Prédire Y à partir de X (trouver une fonction Y=h(X))

Exemples : Y=revenu, X=âge, et

- Modèle linéaire : $f(x) = ax + b$. Deux paramètres à estimer (régression linéaire)
- Modèle non-linéaire : $f(x) = ax + b\sin(cx) + d$. Quatre paramètres à estimer
- ...

Exemple simple

Objectif : Trouver une règle permettant de



à partir de



Tasty



Tasty



Not Tasty

D’autres exemples

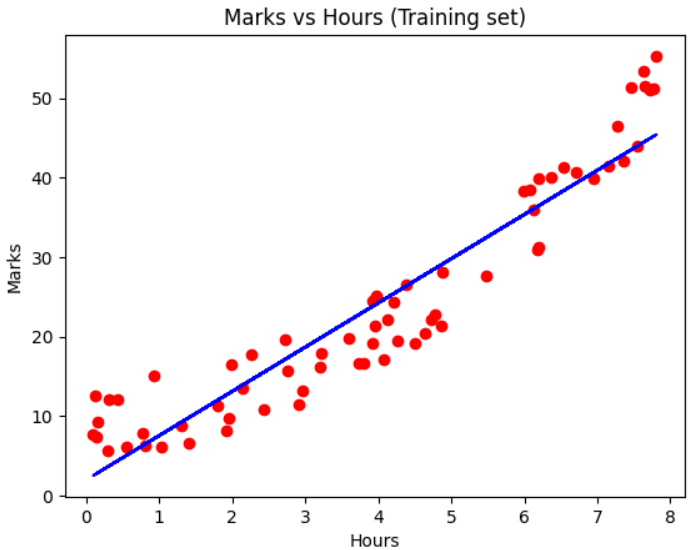


Figure – Prédiction des notes des élèves

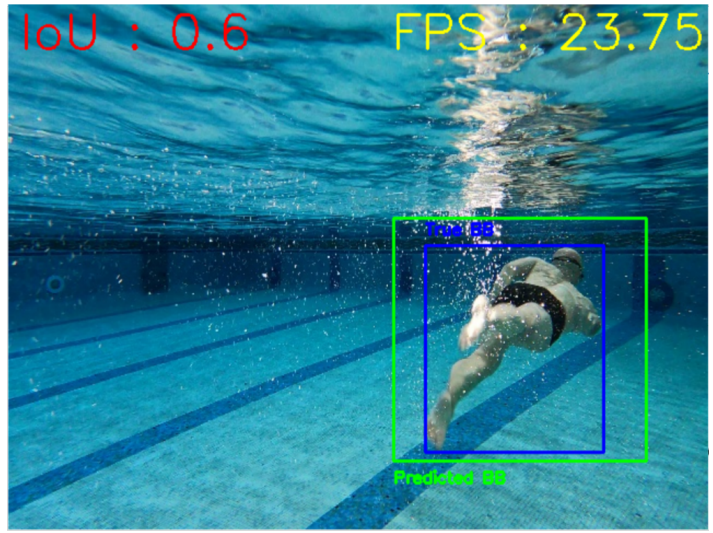


Figure – Détection d’un nageur dans une image

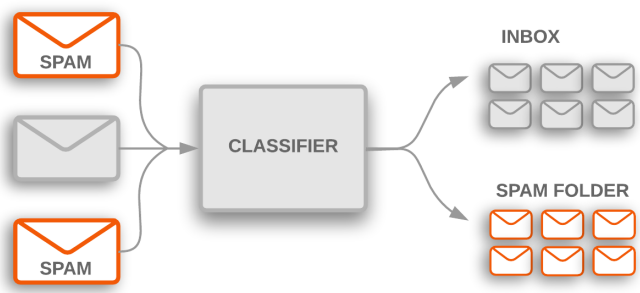


Figure – Classification de spams

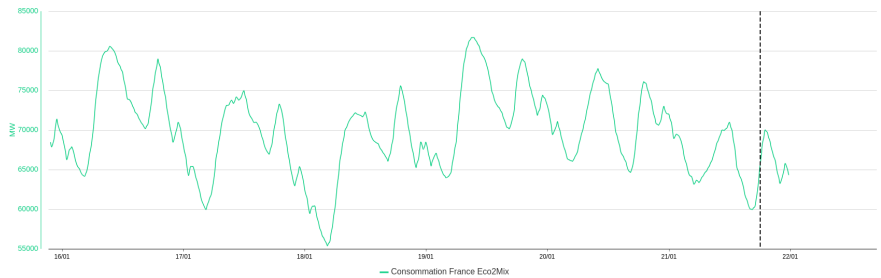


Figure – Prédire la consommation électrique en France

Théorisation de l'apprentissage supervisé

Ingrédients de l'apprentissage supervisé :

- $\mathcal{X} \in \mathbb{R}^{n_X}$, le domaine des données d'entrée souvent représenté par des caractéristiques (features).
- $\mathcal{Y} \in \mathbb{R}^{n_Y}$, le domaine des données cible que nous souhaitons connaître.
- $S = ((x_1, y_1), \ldots, (x_n, y_n))$, l'ensemble des données d'apprentissage regroupant un ensemble de paires dans $\mathcal{X} \times \mathcal{Y}$ tirées de manière aléatoire et indépendante selon une distribution $\mathcal{D}$.
- $h : \mathcal{X} \to \mathcal{Y}$, une règle de décision à choisir et trouver pour pouvoir faire les prédictions.
- Fonction de perte $\mathcal{L}_{\mathcal{D}}(h)$ permettant de quantifier la performance d'une règle de décision h sur une distribution $\mathcal{D}$.

Catégories d'apprentissage supervisée

L'apprentissage supervisé peut se séparer en **2 catégories** :

- **Régression** :
  - Ensemble cible : $\mathcal{Y} \in \mathbb{R}^{n_Y}$
  - Fonctions de perte possibles : L2, L1,...
  - Exemples : Prédiction de production électrique, solaire, ...

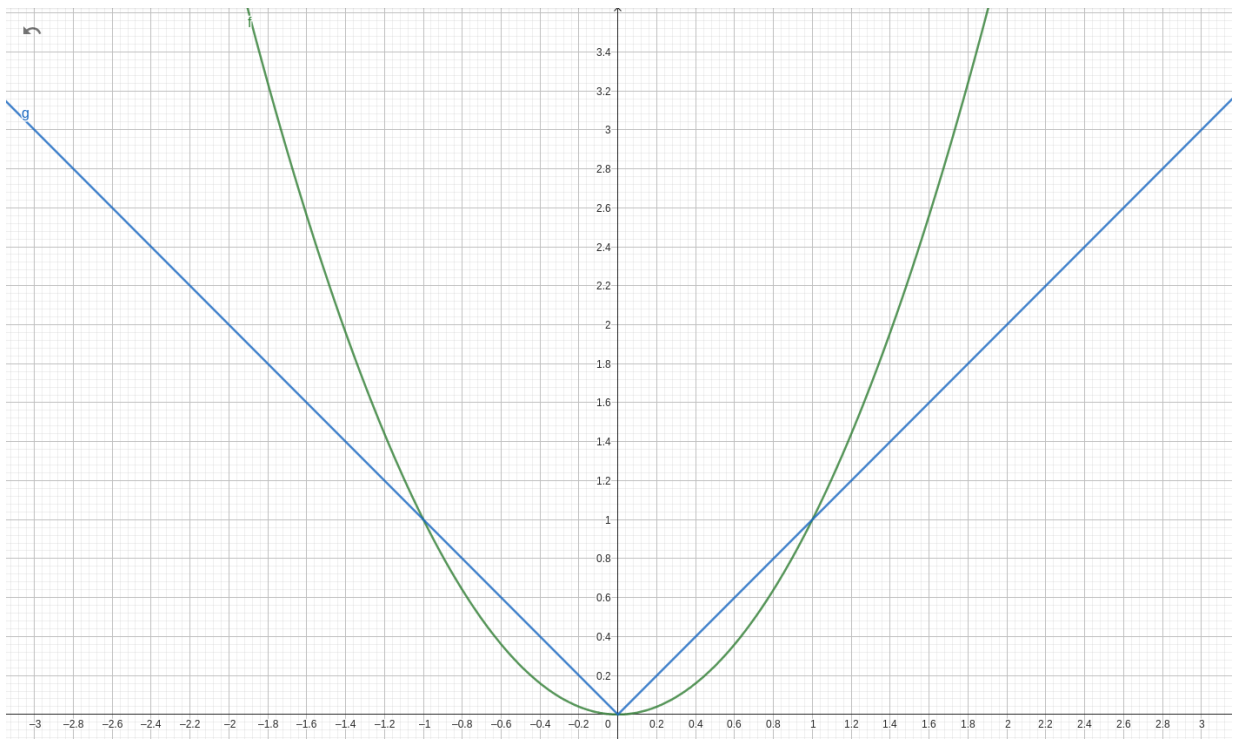
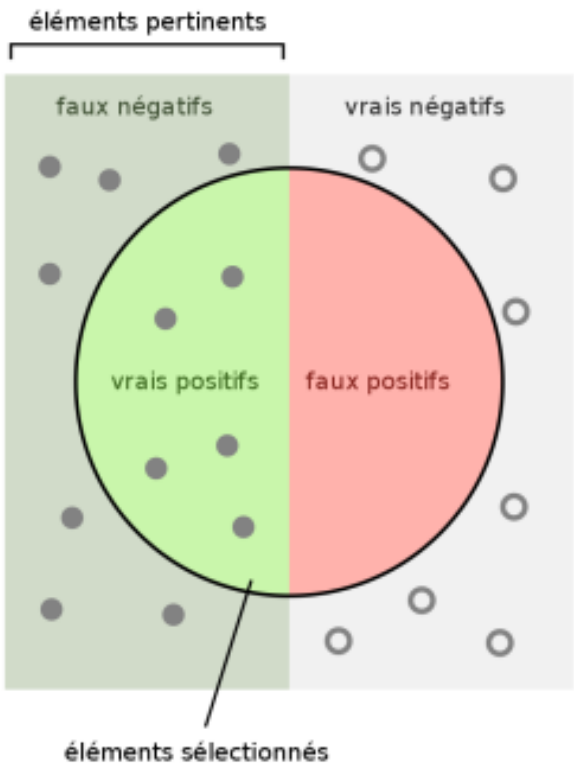


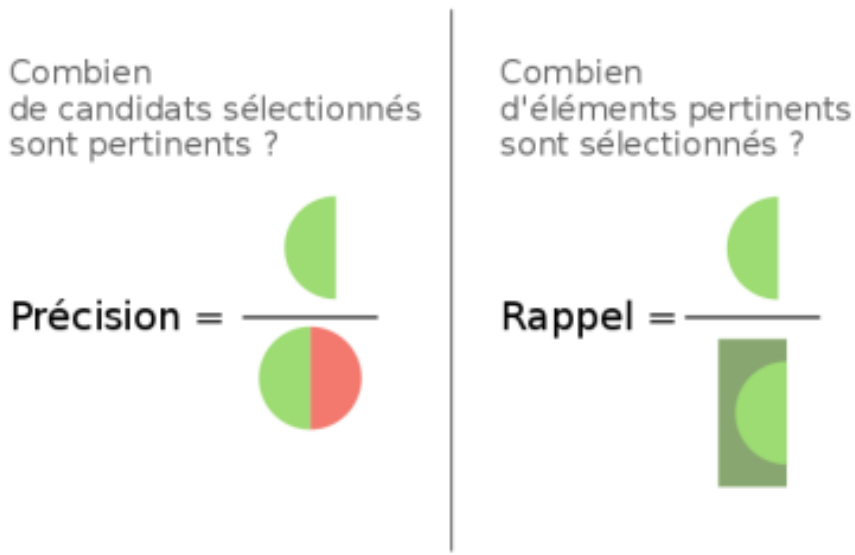
Figure – Fonctions de perte L2 et L1

Catégories d'apprentissage supervisée

L'apprentissage supervisé peut se séparer en **2 catégories** :

- **Classification** :
  - Ensemble cible : $\mathcal{Y} \in \{0,1\}^{n_Y}$, $\mathcal{Y} \in \{0,\ldots,N_{cat}\}^{n_Y}$
  - Fonctions de perte possibles : Sigmoid, Hinge, Cross-entropy, AUC, Précision
  - Exemples : Détection cyberattaque, Classification d'humeur





Comment trouver la règle de décision ?

Objectif : Trouver la règle de décision h qui minimise la fonction de perte par rapport à la distribution $\mathcal{D}$ qui est inconnue.

Première idée : Calculer l'erreur en utilisant l'ensemble des données d'apprentissage S tiré selon $\mathcal{D}$

$$\min_h \mathcal{L}_S(h) = \min_h \sum_{i=1}^{n} ||h(x_i) - y_i||_2^2 \qquad \text{(Régression L2)}$$

- $\mathcal{L}_S(h)$: Erreur/Risque d'entraînement
- **Empirical Risk Minimization (ERM)**
- "Logique" car S représente l'information sur la distribution $\mathcal{D}$

Outline

Apprentissage supervisé

Méthode 1 : k-nearest neighbors (k plus proches voisins)

Choix des hyperparamètres

Validation

Présentation visuelle k plus proches voisins

Sachant un jeu de données S, quels sont les **K** exemples qui sont les plus proches ?





Présentation visuelle k plus proches voisins

Sachant un jeu de données S, quels sont les **3** exemples qui sont les plus proches ?





- 3 chats $\rightarrow$ 100% de probabilité chat

Présentation visuelle k plus proches voisins

Sachant un jeu de données S, quels sont les **6** exemples qui sont les plus proches ?





- 3 chats et 3 lions $\rightarrow$ 50% de probabilité chat et 50% de probabilité lion

Formalisation des k plus proches voisins

Configuration/Entrée :

- Une distance $\rho(x_1, x_2)$ entre deux exemples x_1 et x_2
  - On pose $\forall x \in \mathcal{X}, \pi_1(x), \ldots, \pi_n(x)$ une réorganisation des entiers de $\{1, \ldots, n\}$ selon la distance ρ
- Un nombre de voisins à choisir k
- Jeu de données S

Sortie : Pour chaque individu $x_{test} \in \mathcal{X}$

- $\{y_{\pi_i(x)} : i \leq k\}$: k plus proches voisins de x_{test} dans S par rapport la distance ρ.
- Règle de décision générale

$$h_S(x) = \Phi\Big((x_{\pi_1(x)}, y_{\pi_1(x)}), \ldots, (x_{\pi_k(x)}, y_{\pi_k(x)})\Big)$$

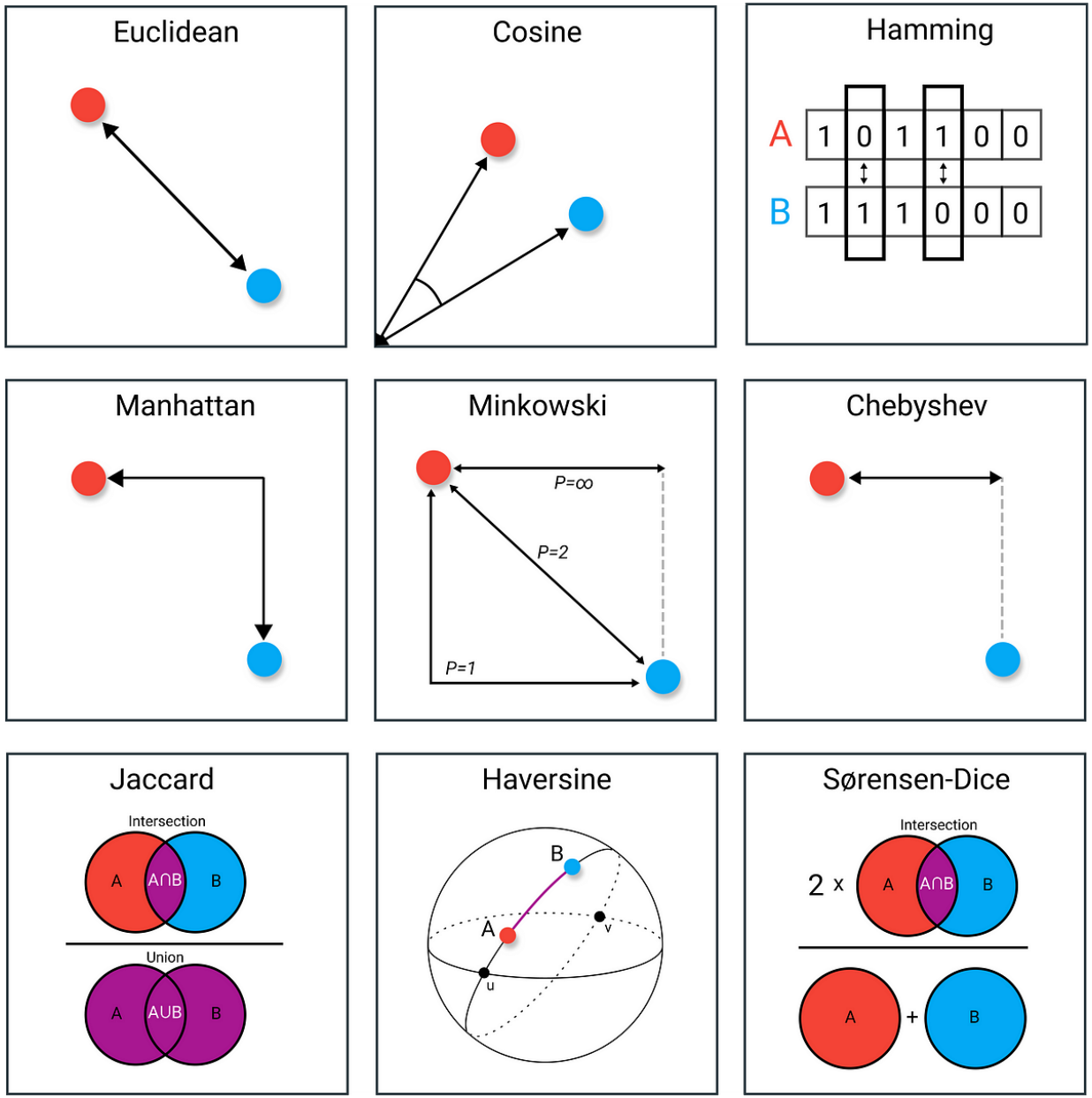
- Exemples :
  - Vote à la majorité (classification)
  - Moyenne (régression)

$$h_S(x_{test}) = \frac{1}{k} \sum_{i=1}^{k} y_{\pi_i(x_{test})}$$

  - Moyenne pondéré par rapport à la distance (régression)

$$h_S(x_{test}) = \frac{\sum_{i=1}^{k} \rho(x_{test}, x_{\pi_i(x_{test})}) y_{\pi_i(x_{test})}}{\sum_{i=1}^{k} \rho(x_{test}, x_{\pi_i(x_{test})})}$$

Différentes distances



Exemple classification/régression

- X = Nombre de sites photovoltaïques
- Y = Énergie produite annuelle (MWh)

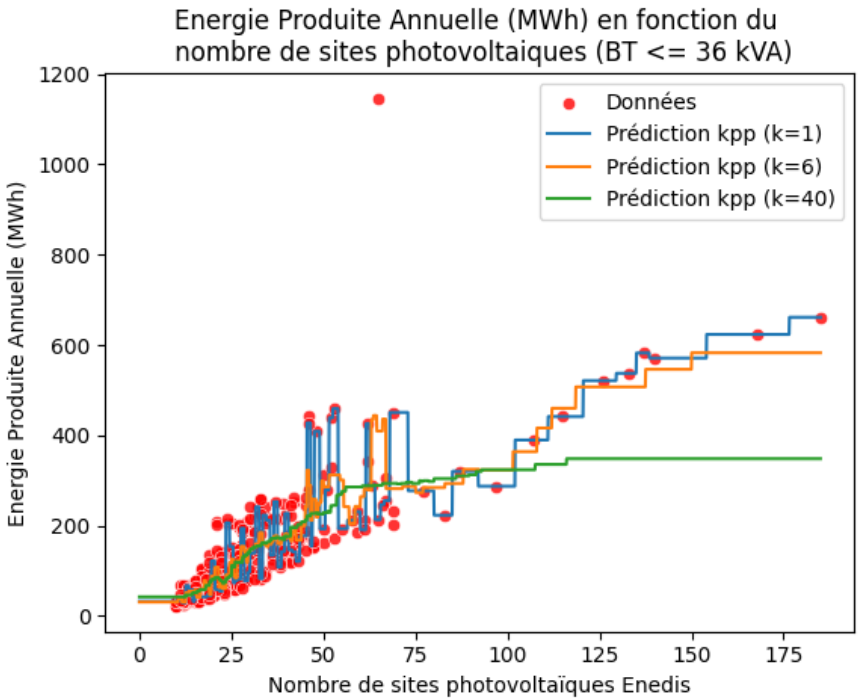


Figure – Régression avec k plus proches voisins.

Code

- X = Année de construction et Surface Thermique
- Y = DPE de classe A, B ou C

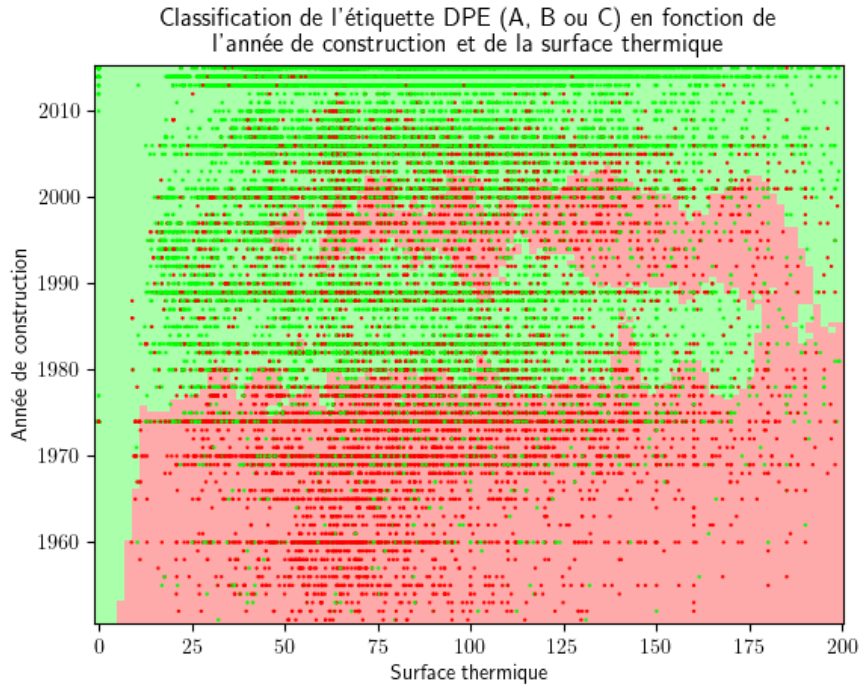


Figure – Classification avec k plus proches voisins.

Outline

Apprentissage supervisé

Méthode 1 : k-nearest neighbors (k plus proches voisins)

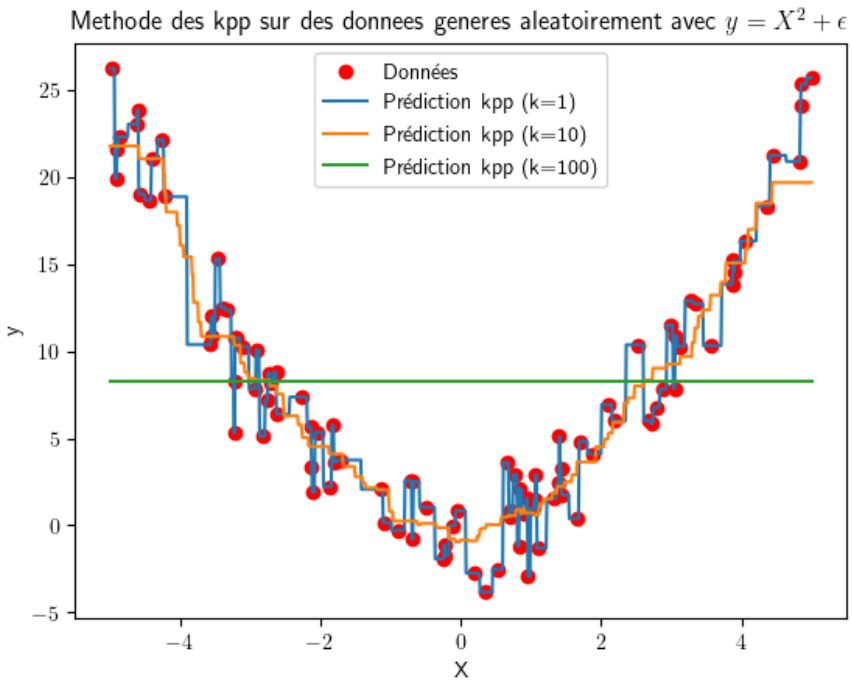
Choix des hyperparamètres

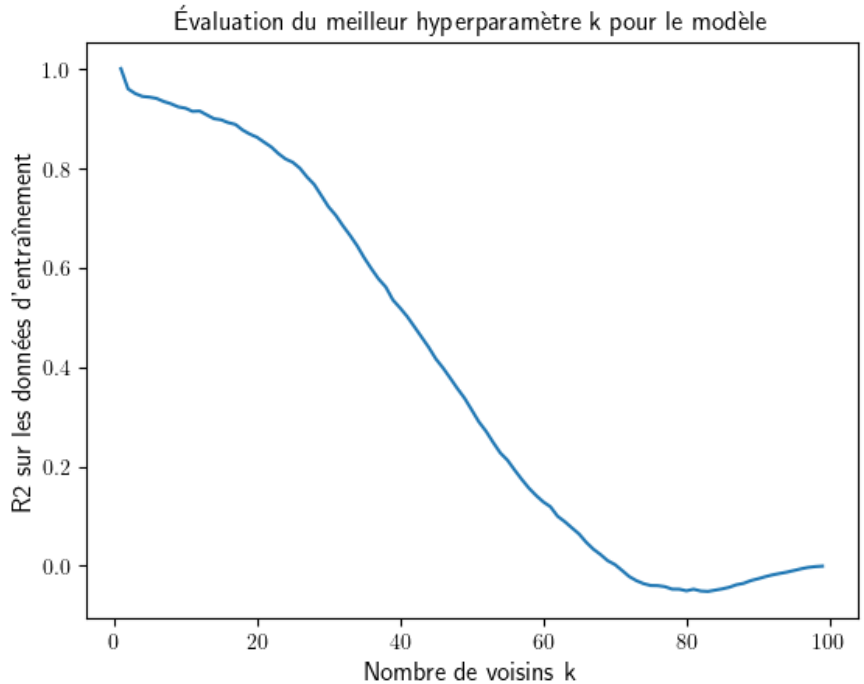
Validation

Comment choisir *k* ?

Exemple précédent : Test de différents hyperparamètres *k*.

Première idée : Minimiser l'erreur sur le jeu de données d'entraînement (ERM).





Résultat : k=1 voisin nous donne le meilleur score.

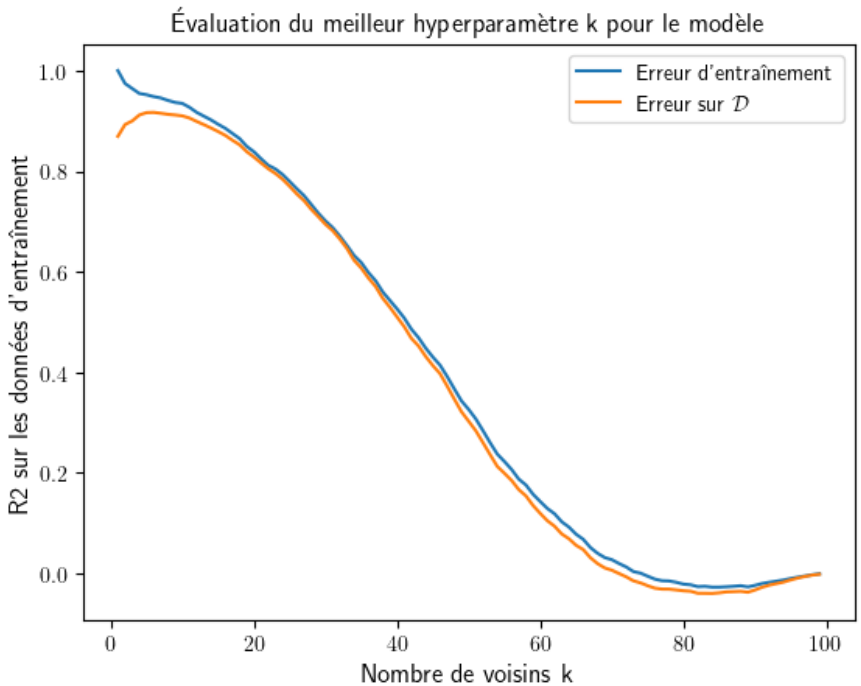
- Étrange au vu des prédictions sur le graphique de gauche.
- k=10 voisins paraît plus naturelle.

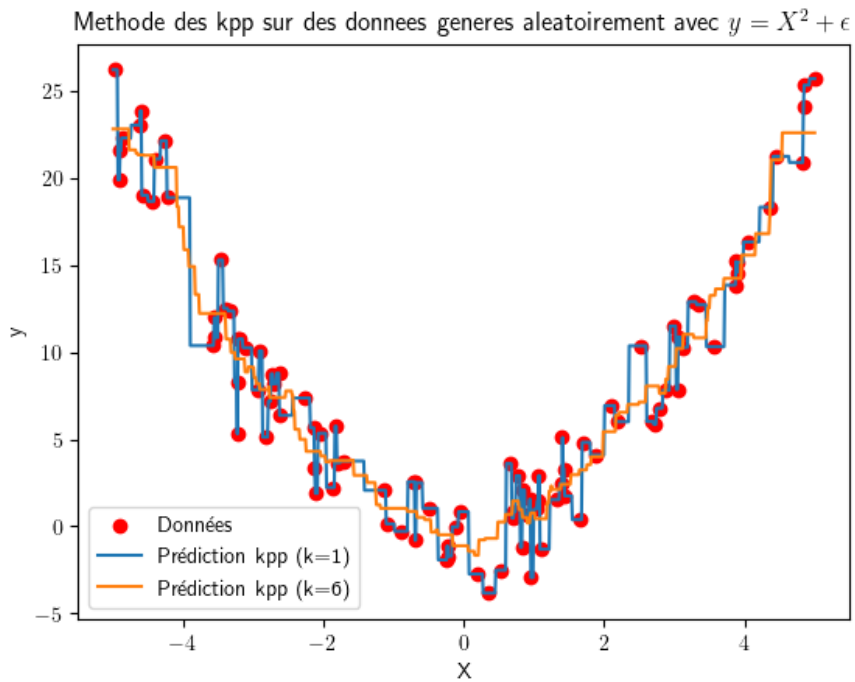
Code

Comment choisir k ?

Exemple utilisé : Données simulées

- La distribution $\mathcal{D}$ est connue.
- Il est possible de calculer l'erreur globale $\mathcal{L}_{\mathcal{D}}(h)$.





Conclusion : k=6 voisins nous donne le meilleur score sur $\mathcal{D}$.

- La méthode ERM (minimisation sur S) ne fonctionne pas bien.

Code

Problème générale du ERM

Règle de décision :

$$h_s(x) = \begin{cases} Y_i & \text{si } i \in \{1, \ldots, n\} \text{ et } x = x_i \\ C, \text{une constance} & \text{sinon} \end{cases}$$

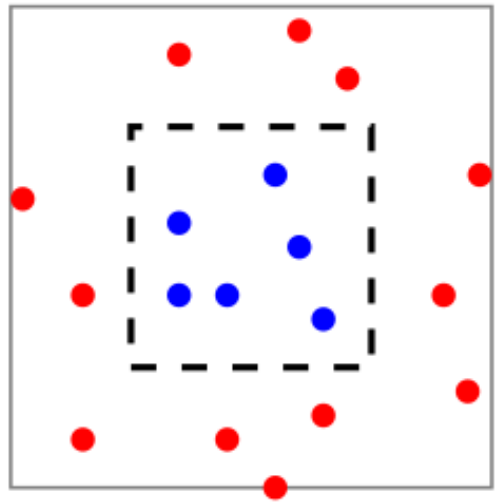


Figure – Jeu d'entraînement (rouge : not tasty, bleu : tasty)

- La fonction h_s minimise l'erreur d'entraînement ($\mathcal{L}_S(h_S) = 0$).
- La fonction h_s a une erreur importante sur la distribution $\mathcal{D}$ ($\mathcal{L}_{\mathcal{D}}(h_S) = \frac{1}{2}$).
- **Overfitting (Surajustement)** ! : faible erreur d'entraînement mais erreur totale élevée (faible généralisation).

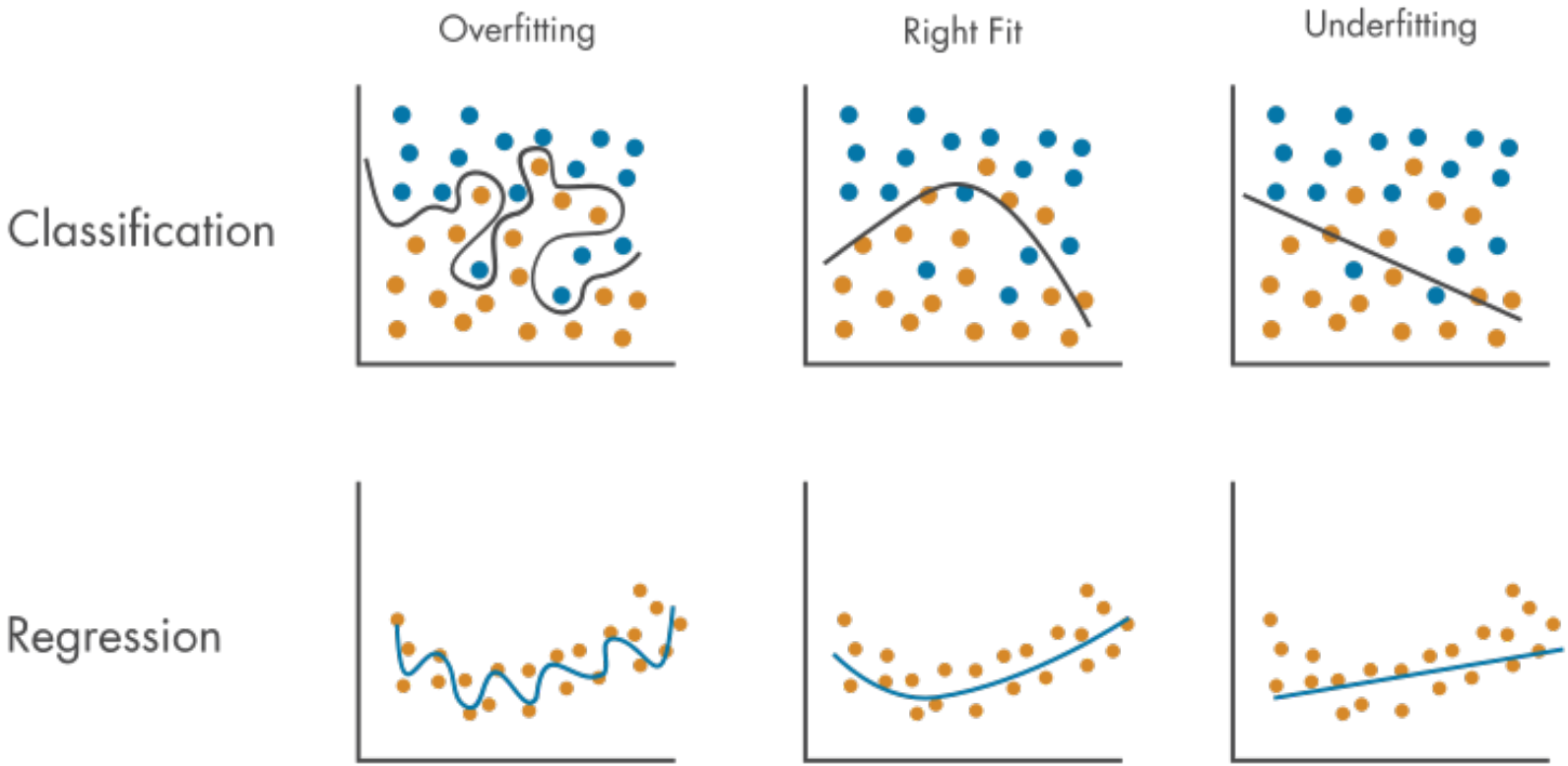
Solution : Restreindre la recherche de la fonction h dans une classe d'hypothèse

- Biais inductif (recherche sur un espace restreint)
  - Exemple 1 : classe des fonctions linéaires (recherche du meilleur a et b)
  - Exemple 2 : classe des fonctions de type k plus proches voisins (recherche du meilleur k)
- Garantie de non surajustement en fonction des classes choisies.

Compromis biais variance

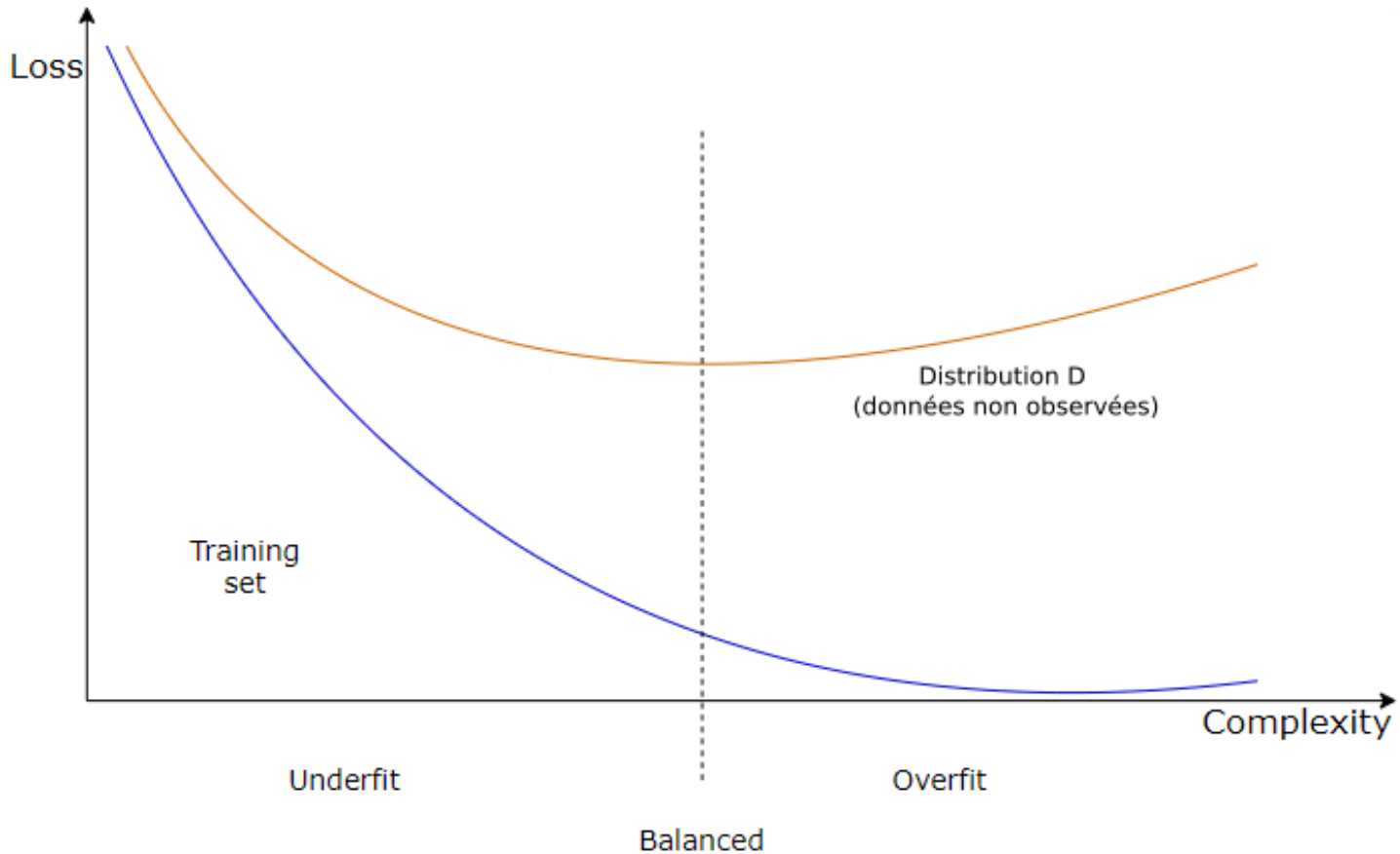
Difficulté : Trouver la bonne classe d'hypothèse (fonction)

- Peut dépendre d'hyperparamètres (k pour knn) pour choisir la complexité



- Classe **trop** restrictive : **biais important**, **faible variance**
- Classe **peu** restrictive : **faible biais**, **variance importante**

Surajustement/Sous-Ajustement



- Trouver la bonne complexité permettant de minimiser l'erreur sur $\mathcal{D}$ et non sur S !
- **Comment faire ?** (pas d'informations sur $\mathcal{D}$)

Outline

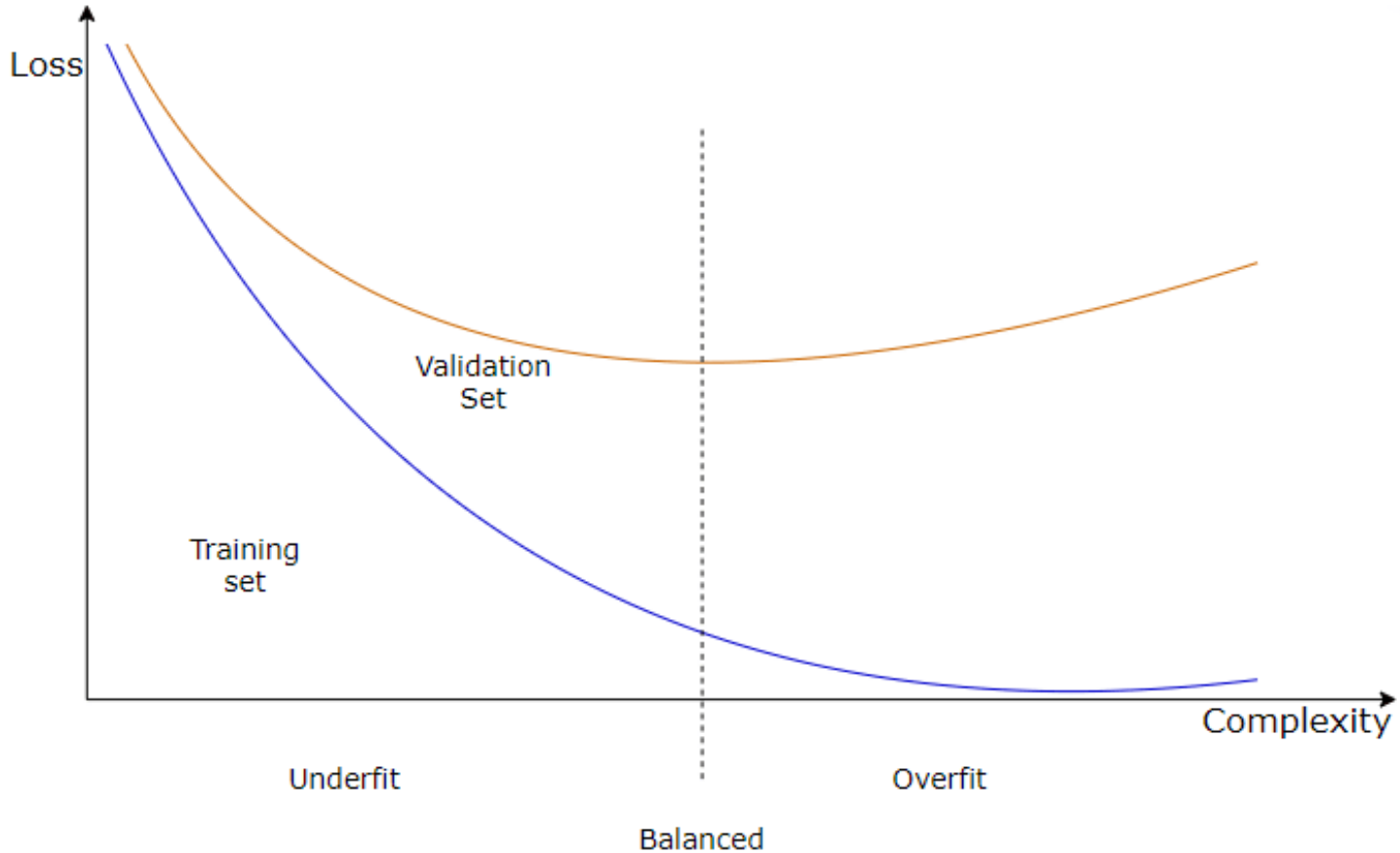
Apprentissage supervisé

Méthode 1 : k-nearest neighbors (k plus proches voisins)

Choix des hyperparamètres

Validation

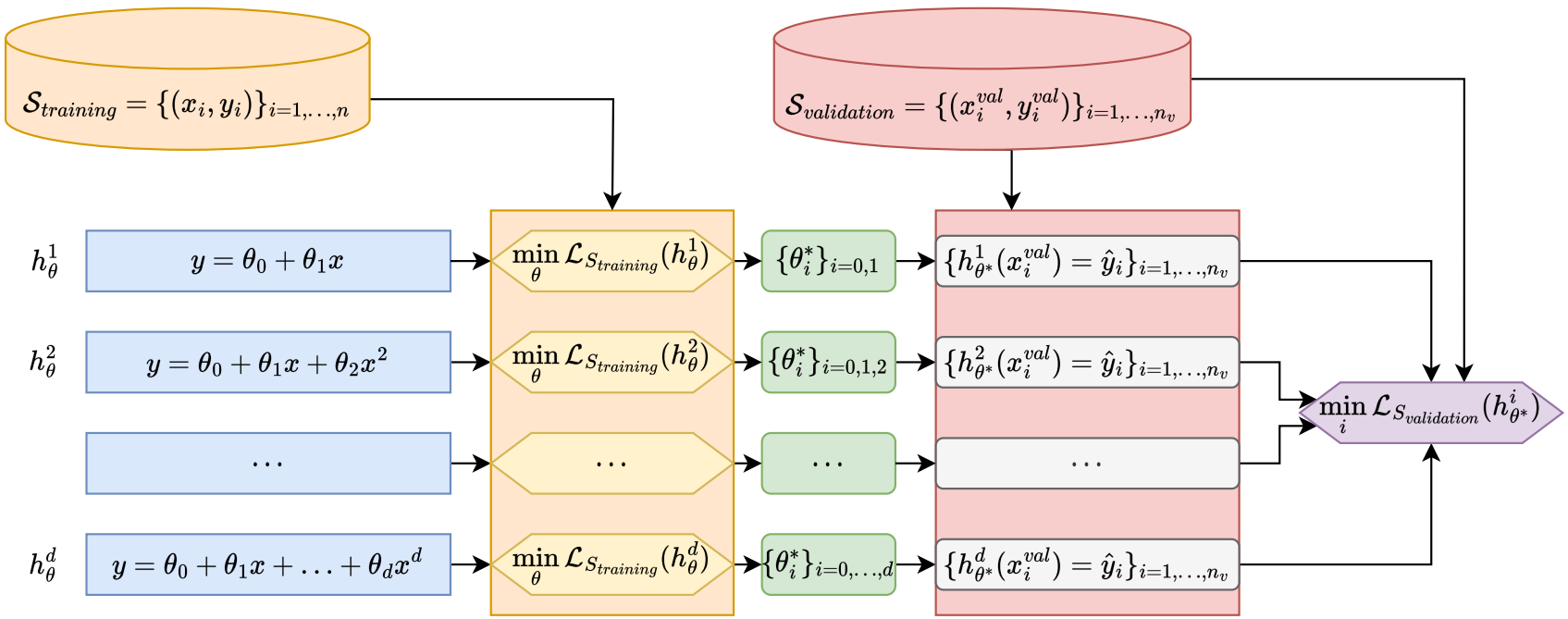
Solution : Utilisation d'un ensemble de validation



- **Deux** jeux de données : $S_{training}$ et $S_{validation}$
- $S_{training}$ est utilisé pour trouver les **meilleurs paramètres** (θ^*) des règles de décision à un niveau de complexité choisi (classe h_i) : $min_\theta \quad \mathcal{L}_{S_{training}}(h_i(\theta))$
- $S_{validation}$ est utilisé pour choisir la **meilleure règle de décision** à partir de celles entraînés avec $S_{training}$: $min_{h_i} \quad \mathcal{L}_{S_{validation}}(h_i(\theta^*))$

Schéma du déroulement de la validation

- **Entraînement individuel** des **différents modèles** (hyperparamètres et/ou structures différents) sur $S_{training}$.
- Calcul des **performances** de chaque **modèle optimisé** sur $S_{validation}$.
- **Sélection** du **meilleur** modèle.



Comment obtenir les deux jeux de données ?

Train-Validation-Test Split : Séparation du jeu de données trois parties

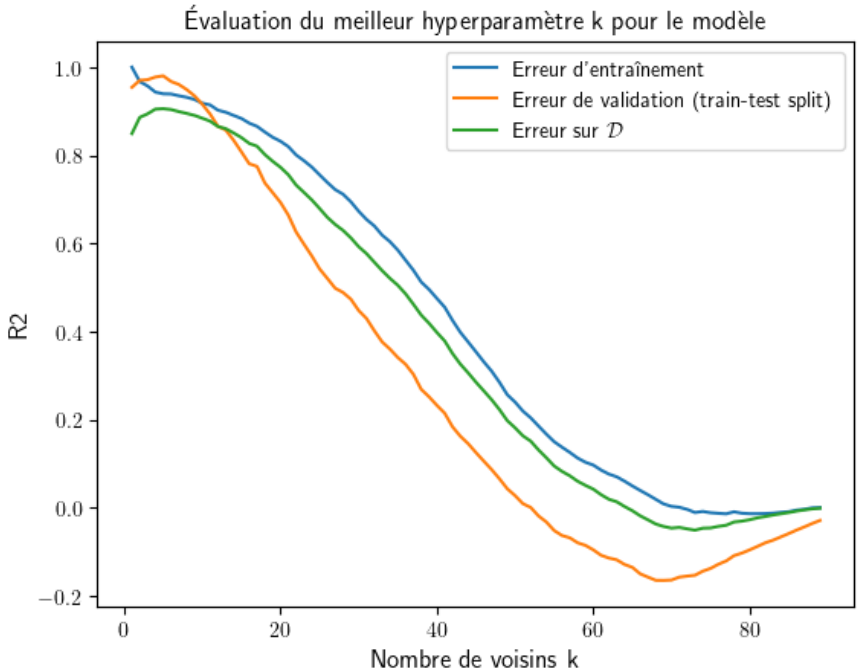
- 70 % pour l'entraînement, 20 % pour la validation, 10 % pour le test
- **Avantage** : simplicité
- **Inconvénients** : perte de données pour l'entraînement

k-Fold Cross Validation (Validation croisée) : Séparation en k groupes

- **Répéter** k fois (itération i)
  - **Regroupement** de $k - 1$ groupes (**suppression** du groupe i) pour l'**apprentissage**
  - Calculer l'**erreur** sur le **groupe i**
- Faire la **moyenne** des erreurs obtenues
- **Avantage** : peu de perte de données pour l'entraînement
- **Inconvénients** : k apprentissages (complexité)
- Cas particuliers : k=n, Leave-one-out

Train-Validation-Test Split



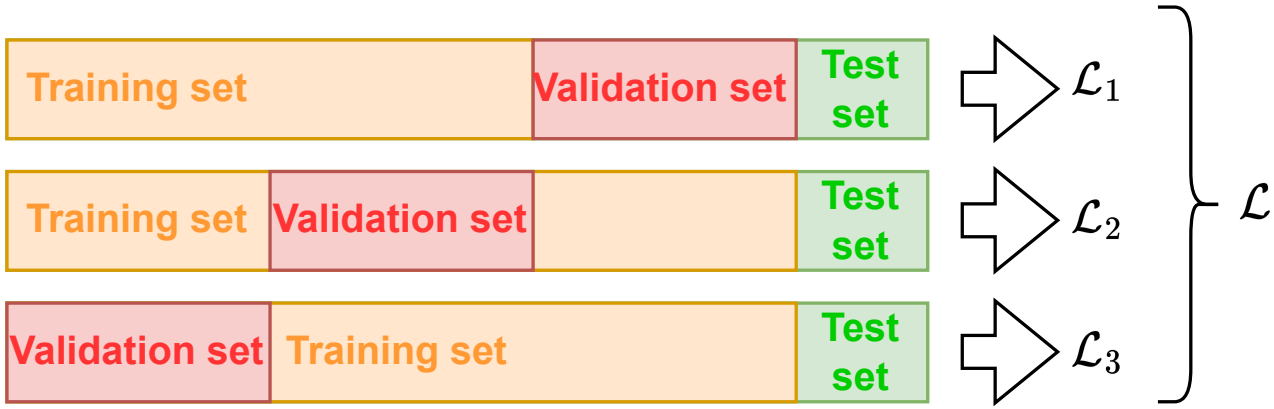


- 20% du jeu de données est utilisé pour la validation.
- k=5 voisins nous donne le meilleur score par validation.
- Proche de l'hyperparamètre k optimal.

Figure – Validation par Train-Validation-Test Split.

Code

k-Fold Cross Validation (Validation croisée)



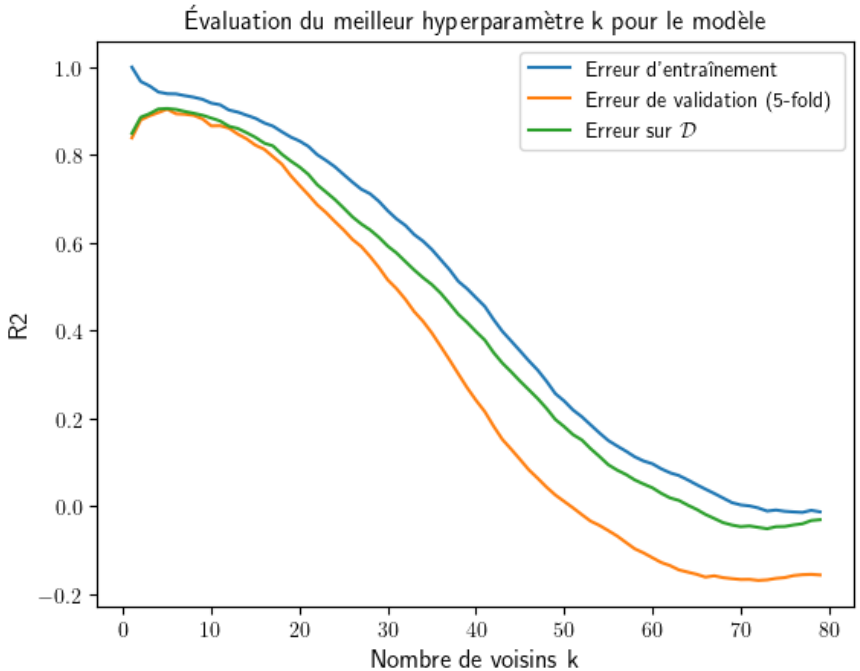
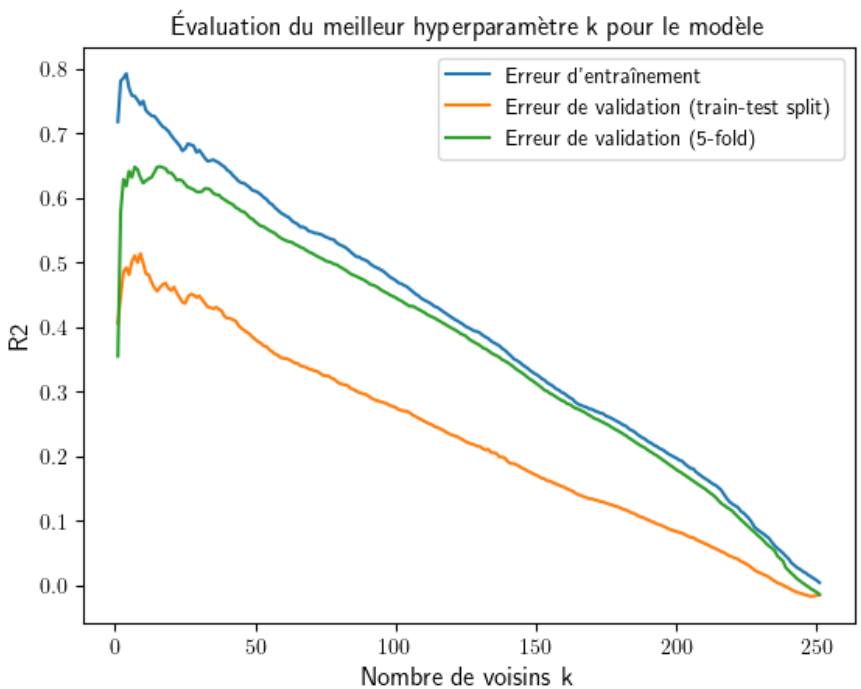


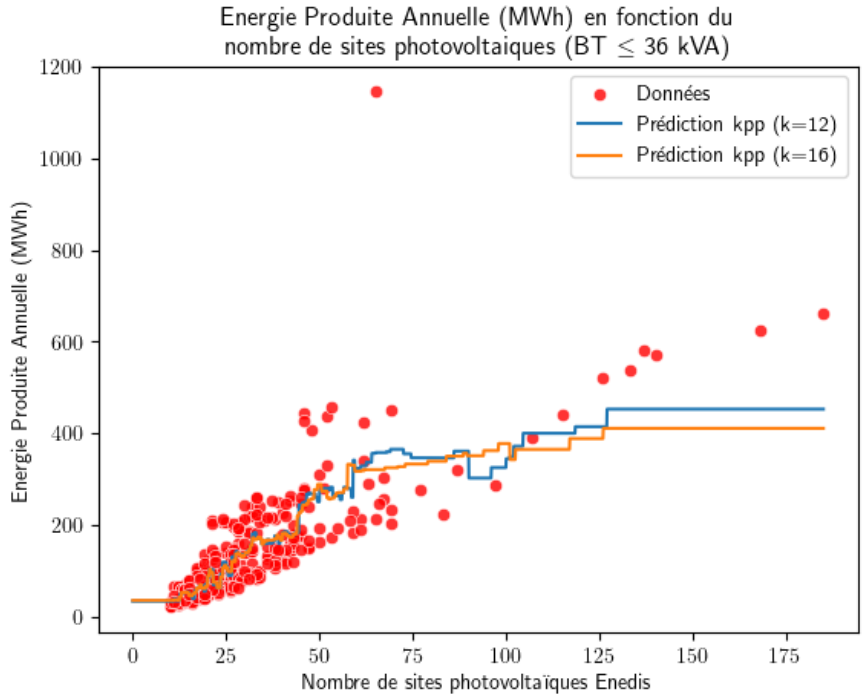
Figure – Validation par 5-Fold.

- Séparation du jeu de données en 5 groupes de tailles équivalentes.
- k=5 voisins nous donne le meilleur score par validation.
- Proche de l'hyperparamètre k optimal.

Code

Exemple de régression : Choix de k

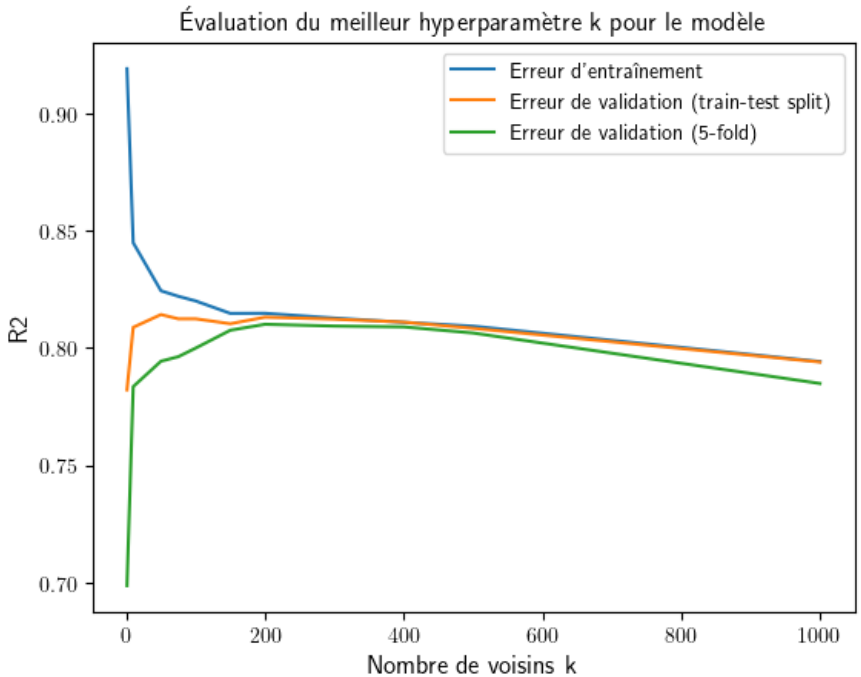


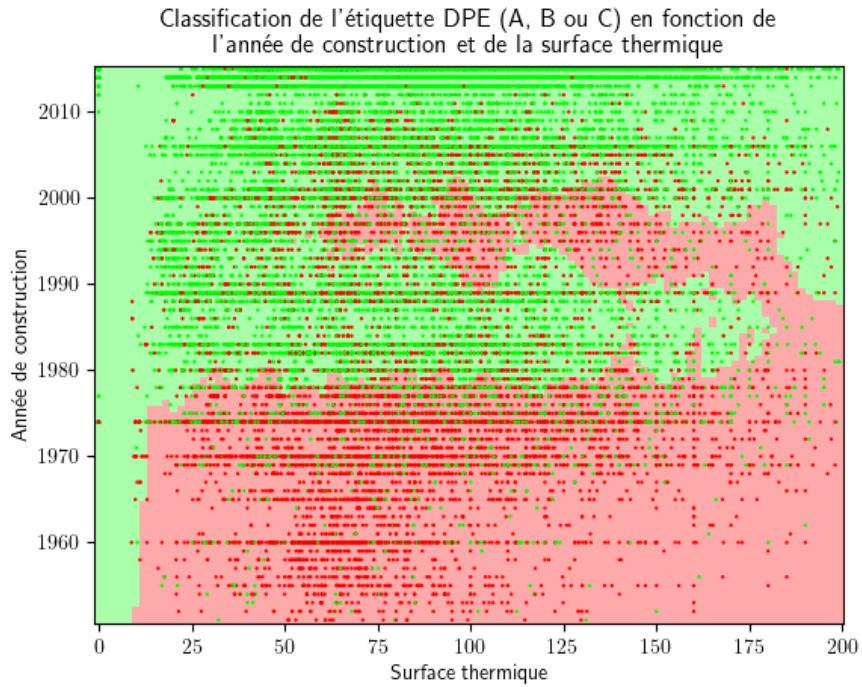


- k=12 voisins nous donne le meilleur score par validation croisée.
- k=16 voisins nous donne le meilleur score par validation train-test split.
- Bonne complexité et pas de surajustement.

Code

Exemple de classification : Choix de k





- k=50 voisins nous donne le meilleur score par validation croisée.
- k=200 voisins nous donne le meilleur score par validation train-test split.
- Temps d'exécution long avec un grand jeu de données.

Code

Que faire si l'entraînement ne fonctionne pas ?

Collecter plus de données : Diminution de la variance

Changer de classe de fonction

- **Agrandissement** : augmentation de la variance, diminution du biais
- **Réduction** : diminution de la variance, augmentation du biais
- **Changement**

Changer la représentation des données : Créer des caractéristiques supplémentaires sur les données (augmentation de la variance, diminution du biais)

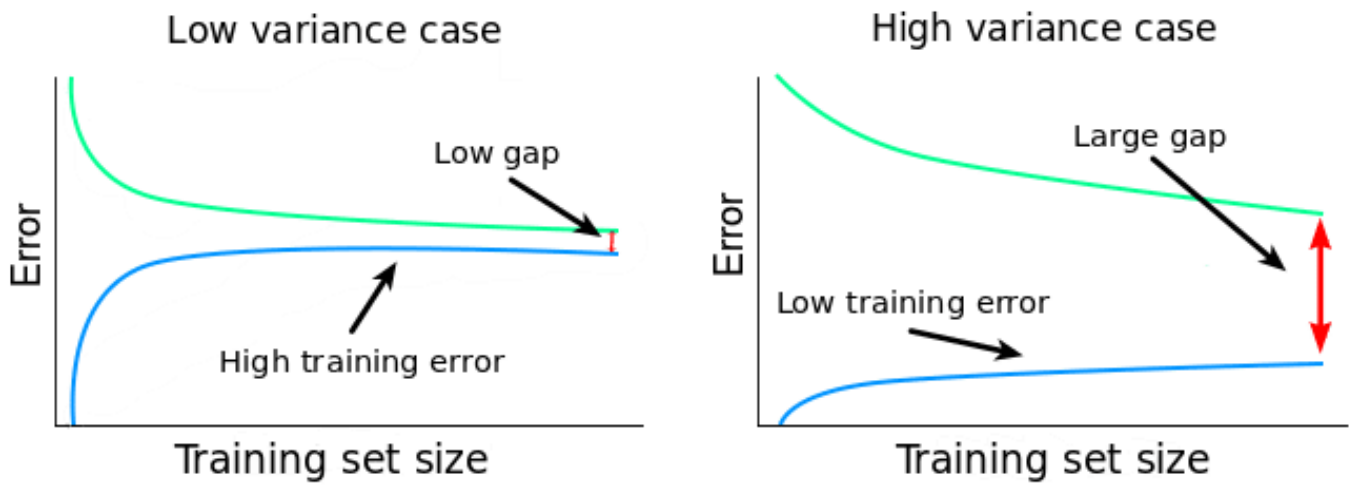


Figure – Outil pour détecter si le problème vient du biais ou de la variance.

À retenir

Apprentissage supervisé

- Données **labélisées**.
- Trouver la **meilleure fonction** de prédiction par rapport à une **mesure** de **succès**.
- Attention au surajustement.
- Choix de la bonne **complexité** du modèle par **validation**.

Méthode des k plus proches voisins

- Les **données** sont le **modèle**.
  - Pas besoin de phase d'apprentissage.
  - Bons résultats sur application réelle.
  - Modèle boîte noire (pas de compréhension des relations).
  - Fléau de la dimension (si $\mathcal{X}$ est grand).
  - Besoin de stocker les données en mémoire.
- Choix **important**
  - Un seul paramètre **k** important à régler (compromis biais-variance).
  - Choix de la **métrique** et de la **fonction de poids**.