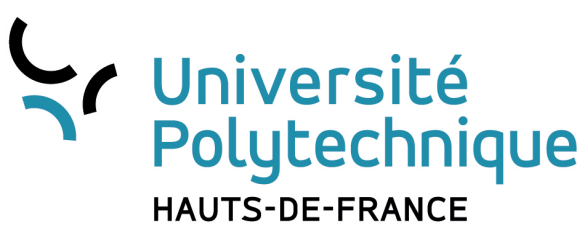


Apprentissage Statistique Automatique I

Introduction à l'apprentissage statistique automatique

Andrés F. López-Lopera
Université Polytechnique Hauts-de-France (UPHF)

Thèmes

1. Motivation

2. Apprentissage statistique automatique
   - Apprentissage supervisé
   - Apprentissage non supervisé
   - Règles de prédiction

3. Validation croisée
   - Validation croisée “*K*-fold”
   - Validation croisée “*Hold-out*”

Motivation

Problèmes d'apprentissage statistique

- **Médecine**
  - Prédiction du taux de graisse chez un patient (caractéristiques : poids, taille, ...)
  - Identification des facteurs de risque du cancer (caractéristiques : âge, sexe, ...)
- **Finance**
  - Prédiction des prix des appartements (caractéristiques : surface, année de construction, nb pièces, quartier, ...)
  - Analyse de l'influence des publicités sur les ventes d'un produit (caractéristiques : budget des publicités TV, radio, journaux, ...)
- **Biologie et science de la Terre**
  - Prédiction de la propagation du Covid (caractéristiques : nb des personnes contaminées et leur position, ...)
  - Analyse de l'évolution du changement climatique (caractéristiques : émissions de CO_2, température, ...)

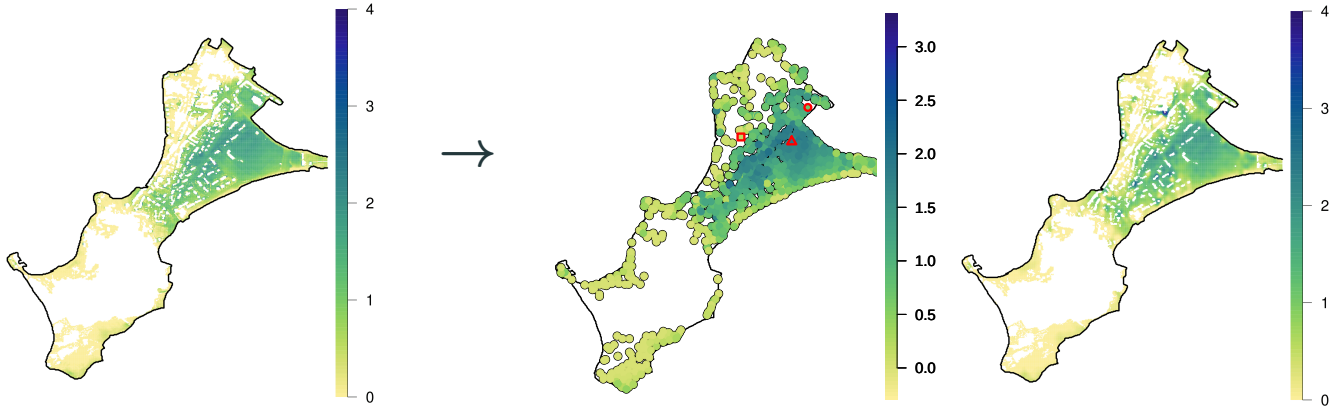
(*cf.* cours "Analyse de Données", semestre 1)

Problèmes d'apprentissage statistique

· **Ingénierie**

⋆ Prédiction de la propagation spatiale des inondations causées par une tempête (caractéristiques : puissance du vent et de la marée, hauteur de la vague, ...)

Simulation numérique — Modèle d'apprentissage statistique



∼ 3 days (∼ 65k points) → ∼ 2 seconds (∼ 1k points) ∼ 1 min (∼ 34k points)

⋆ Amélioration de la conception d'aéronefs (caractéristiques : géométrie de l'aéronef, Reynolds, Mach, ...)

· Ces problèmes se distinguent par le fait que la fonction cible $y : \mathcal{X} \to \mathcal{Y}$ est soit difficile à modéliser, soit exige une quantité importante de ressources

(⋆ expériences de post-doctorats IMT-BRGM et ONERA)

Apprentissage statistique automatique

Apprentissage statistique automatique

· On envisage alors d'utiliser des modèles capables d'apprendre des caractéristiques statistiques présentes dans les bases de données

· C'est-à-dire, on cherche à "ajuster" (entraîner) un modèle $\mathcal{M}_{D_n,\theta} : \mathcal{X} \to \mathcal{Y}$, avec des hyperparamètres $\theta \in \mathbb{R}^p$, sur un ensemble de données d'apprentissage

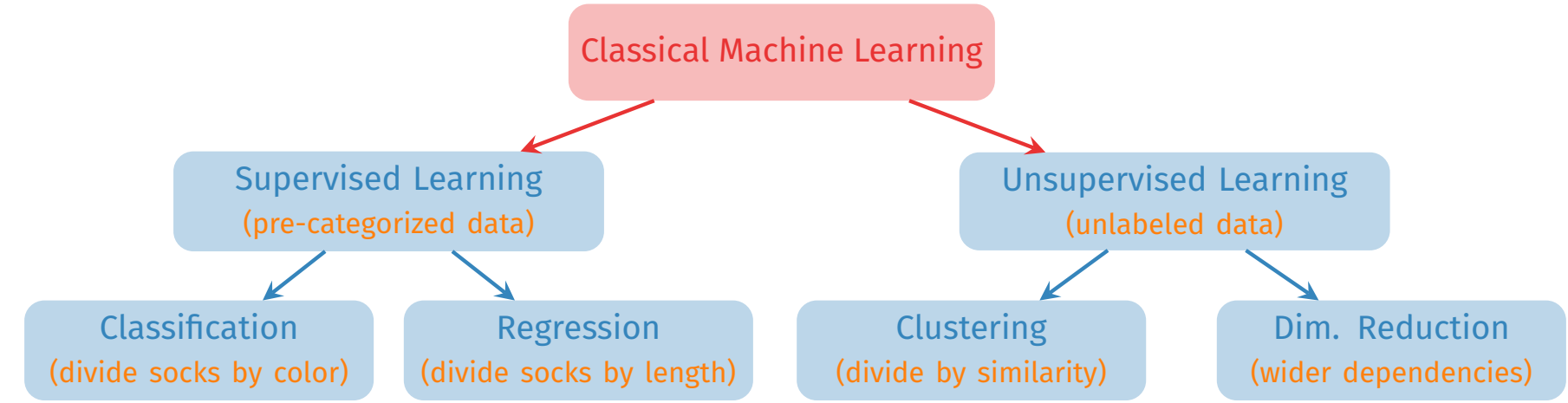
$$D_n = \{(x_1, y_1), \ldots, (x_n, y_n)\},$$

avec $(x_i \in \mathcal{X}, y_i \in \mathcal{Y})$ des paires de données correspondant aux caractéristiques d'entrée et à l'observation

· Ensuite, $\mathcal{M}_{D_n,\theta}$ est utilisé pour prédire (le plus précisément possible) y pour une nouvelle set de caractéristiques d'entrée $x_* \in \mathcal{X}$

Apprentissage statistique automatique (Machine Learning)

· Le *machine learning* (ML), basé sur l'apprentissage statistique, se concentre principalement sur [Bishop, 2006, Hastie et al., 2009]



· D'autres méthodes d'apprentissage plus récentes :

- Apprentissage par renforcement [Murphy, 2024]
- Apprentissage par transfert [Yang et al., 2020]

Apprentissage supervisé

· Soit la fonction cible (également réponse ou variable dépendante)

$$\begin{aligned} y : \quad & \mathcal{X} \to \mathcal{Y} \\ & x \mapsto y(x) \end{aligned}$$

avec $x \in \mathcal{X}$ sont les caractéristiques d'entrée (également variables explicatives ou co-variables)

- $\mathcal{X}$ est un ensemble non vide quelconque (souvent $\mathcal{X} \subseteq \mathbb{R}^d$, où $d \in \mathbb{N}$ représente la dimension de l'espace d'entrée)
- $\mathcal{Y}$ est l'ensemble correspondant aux réponses (souvent $\mathcal{Y} \subseteq \mathbb{R}$ pour la régression, et $\mathcal{Y} \subseteq \mathbb{N}$ pour la classification)

Apprentissage supervisé

· On cherche à ajuster une règle de prédiction (régression ou discrimination) du modèle $\mathcal{M}_{D_n,\theta}$:

$$\mathcal{F}_{D_n,\theta} : \mathcal{X} \to \mathcal{Y},$$

sur l'ensemble de données d'apprentissage

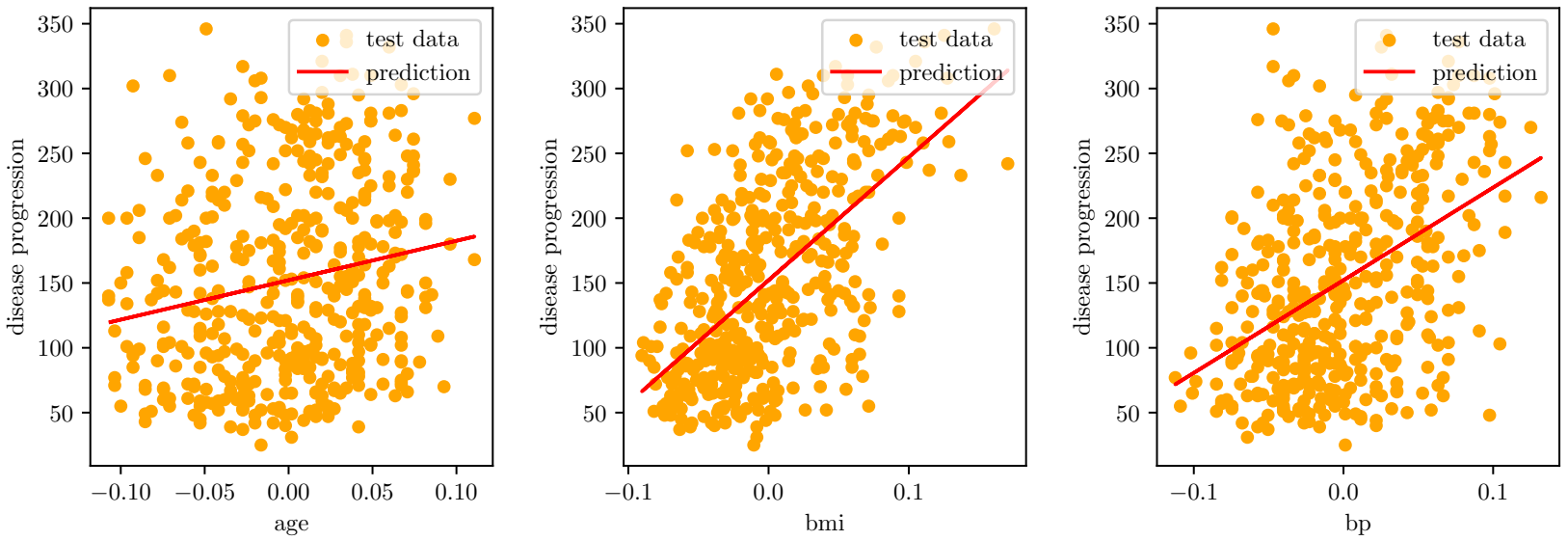
$$D_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

· Ici, $\theta \in \mathbb{R}^p$ sont les hyperparamètres du modèle (e.g., les coefficients du modèle linéaire)

Objectifs :

- Prédire (le plus précisément possible) la réponse y pour un $x \in \mathcal{X}$
- Comprendre quelles co-variables x influent sur y
- Extraire et analyser des informations clés à partir des données (but plus flou)

Régression linéaire : Cas de test - diabètes (sklearn)



bmi : indice de masse corporelle (*body mass index*)
bp : tension artérielle (*blood pressure*)

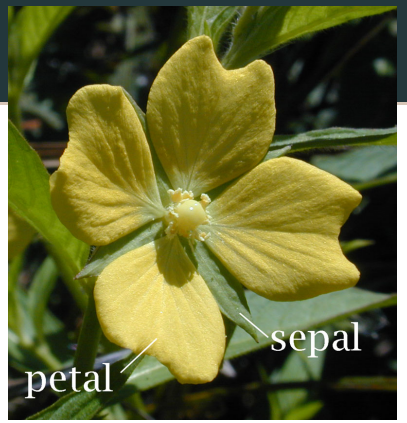
- Existence d'un lien entre le diabète avec l'âge ?
- Quelle est la force de ce lien ?
- Quelle condition (âge, bmi, bp) contribue la plus à la progression de la maladie ?
- Existe-t-il des synergies entre les conditions ?

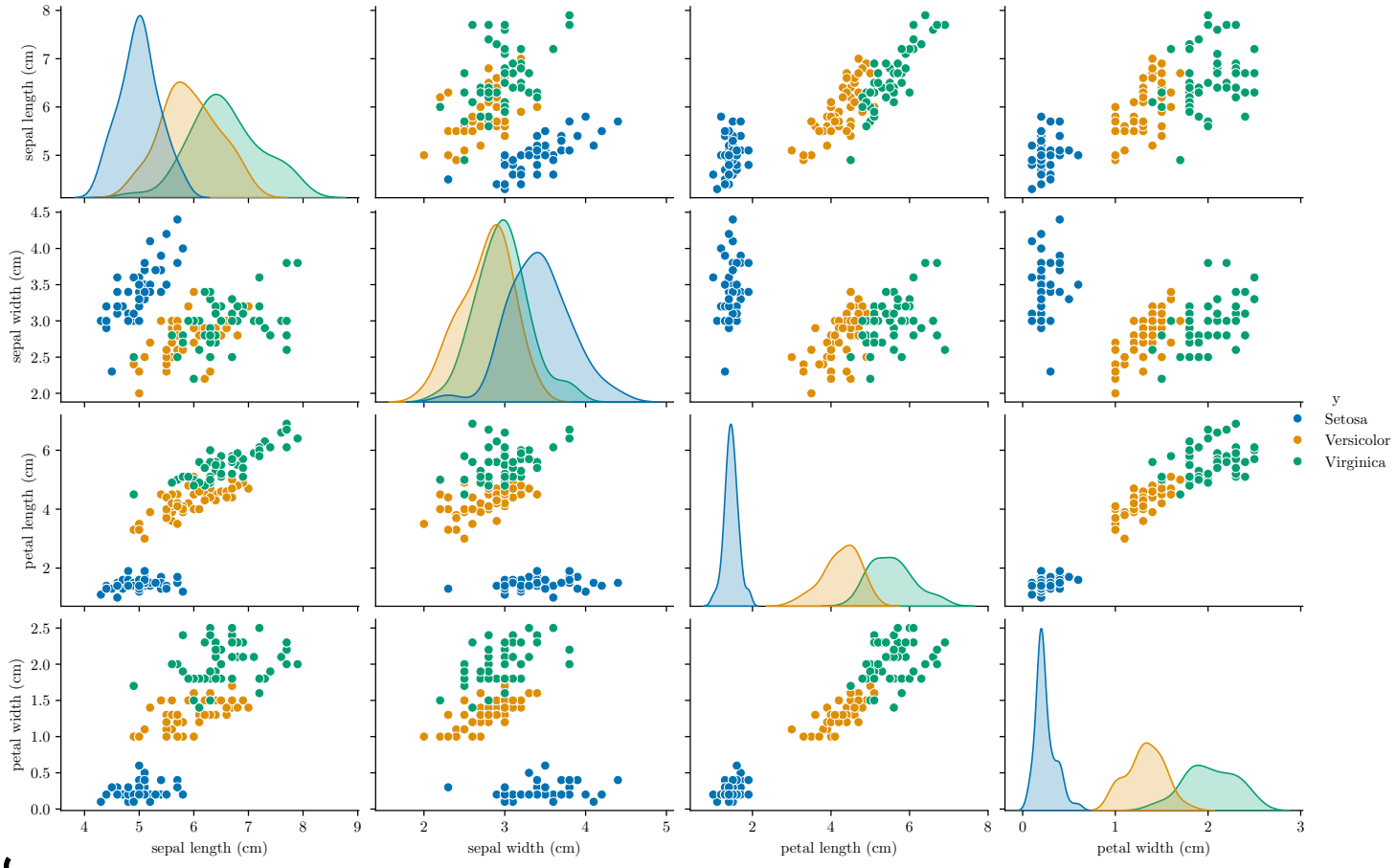
Classification : Cas de test - Iris (sklearn)

- 3 classes :

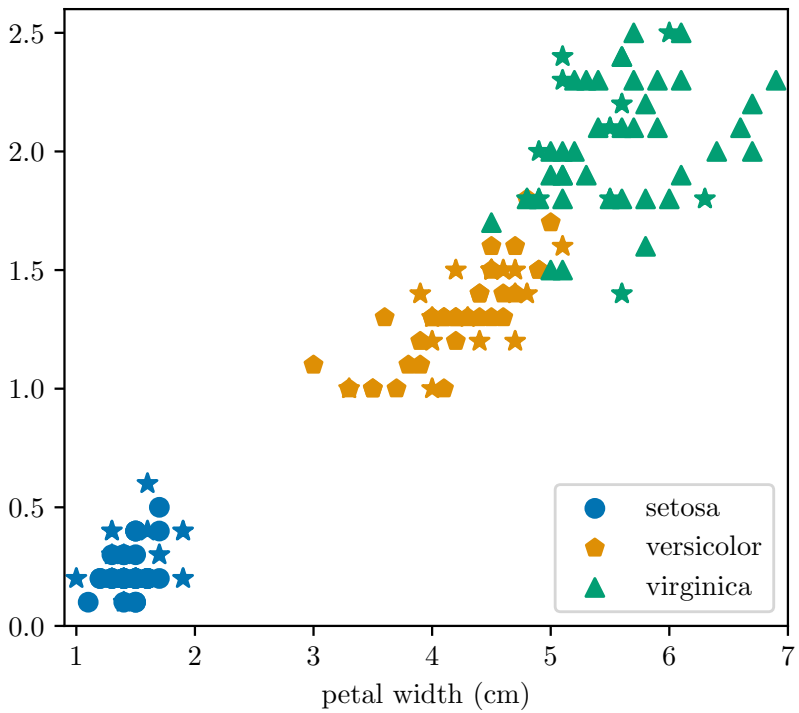
$$\kappa \in \{\texttt{setosa}, \texttt{versicolor}, \texttt{virginica}\}$$

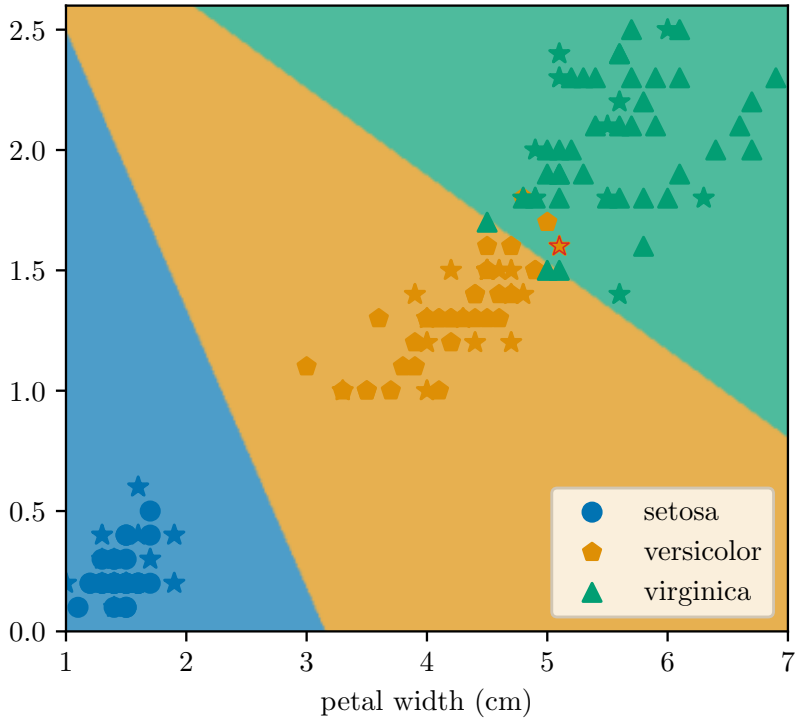
- 4 caractéristiques (patterns) : la longueur et la largeur (cm) du sépale et du pétale





Régression logistique : Cas de test - Iris (sklearn)





● : Données d'apprentissage

⋆ : Données à prédire

Apprentissage non supervisé

· Dans de nombreuses applications, il n'est pas possible d'accéder aux réponses $y_1, \ldots, y_n$ associées aux entrées $x_1, \ldots, x_n$

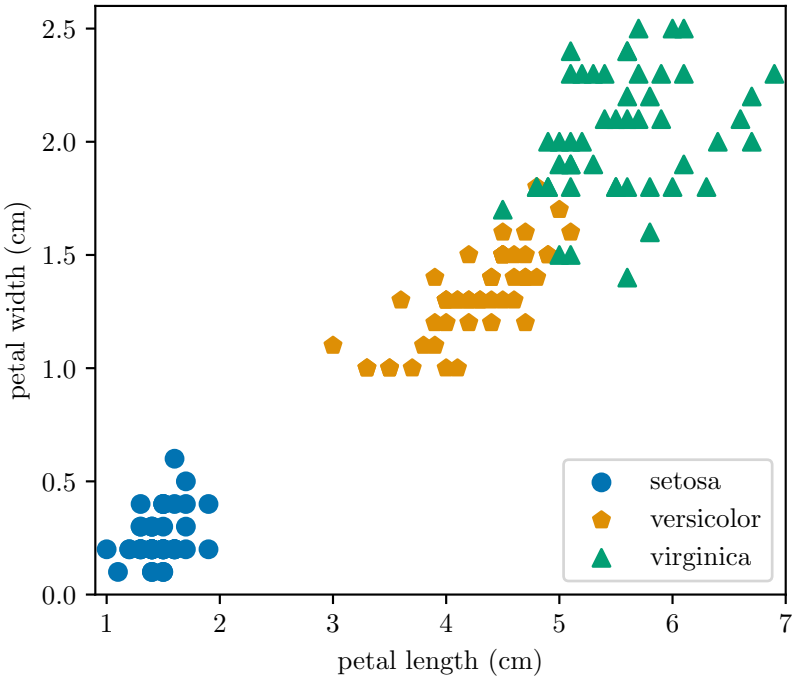
· Par exemple, dans le contexte de la recommandation musicale sur des services de streaming, il n'est pas possible de connaître à priori la "classe" associée à chaque client

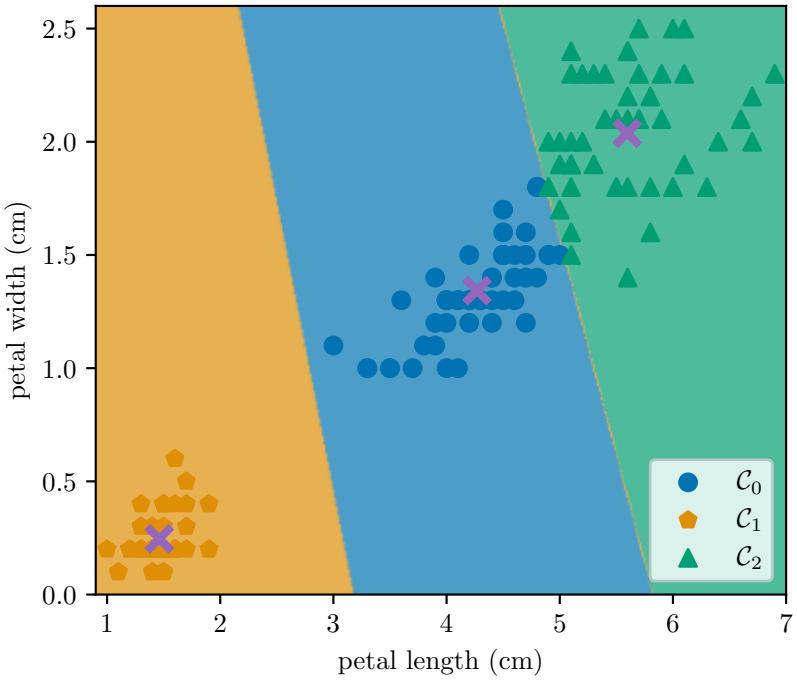
· Par contre, il est possible de l'associer à d'autres clients en fonction de ses affinités (*e.g.*, style de musique et artistes préférés)

· Dans l'apprentissage non supervisé, on s'intéresse à ce type de problèmes où $\mathcal{M}_{D_n,\theta}$ doit être ajusté sur l'ensemble de données d'apprentissage

$$D_n = \{(x_1), \ldots, (x_n)\}$$

Clustering (k-means) : Cas de test - Iris (sklearn)



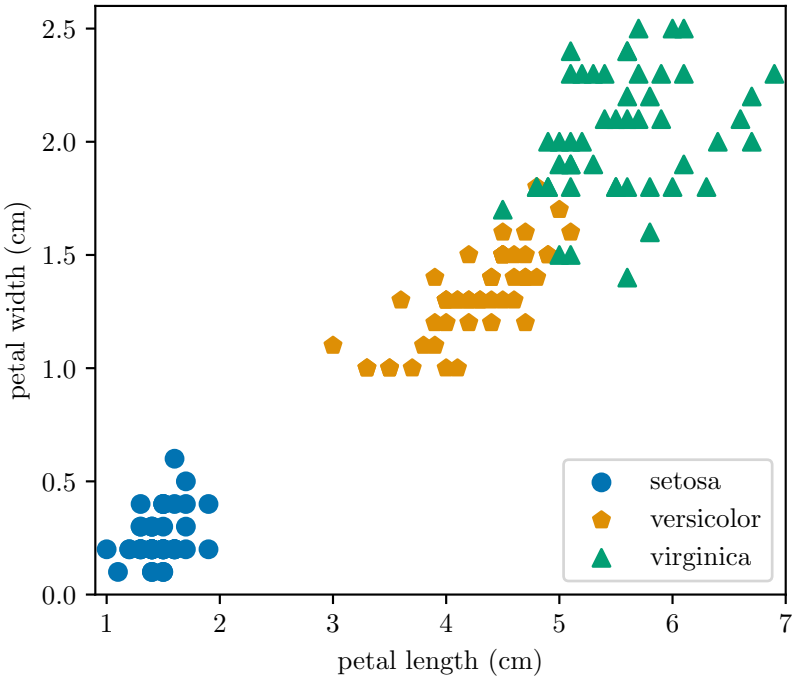


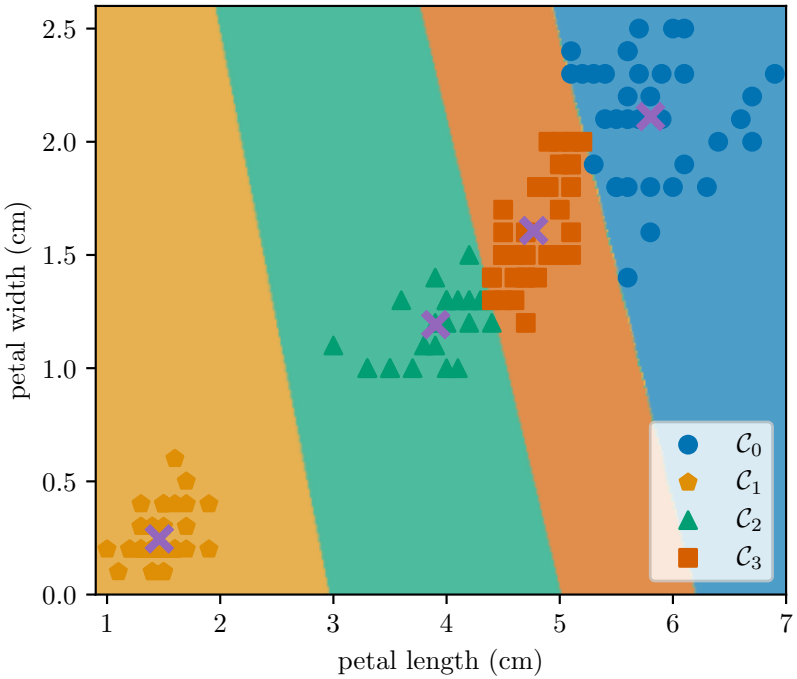
● : Données d'apprentissage

× : centroïdes

· Dans l'apprentissage non supervisé, l'évaluation du résultat est plus difficile

Clustering (k-means) : Cas de test - Iris (sklearn)





● : Données d'apprentissage

× : centroïdes

· Dans l'apprentissage non supervisé, l'évaluation du résultat est plus difficile

Apprentissage non supervisé

Objectifs :

- ~~Prédire (le plus précisément possible) la réponse y pour un $x \in \mathcal{X}$~~
- ~~Comprendre quelles co-variables x influent sur y~~

On ne dispose pas des réponses y !

- Extraire et analyser des informations clés à partir des données (but plus flou)

· L'apprentissage non supervisé peut être une étape préliminaire à un apprentissage supervisé

- Par exemple, pour regrouper des observations / clients / patients par des critères d'affinités

Règles de prédiction (régression ou discrimination)

· Suppose que $D_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ est un n-échantillon qui suit une loi P sur $\mathcal{X} \times \mathcal{Y}$ inconnue

· On suppose que X est une nouvelle observation, (X, Y) étant un couple aléatoire de loi conjointe P indépendante de D_n

· Une règle de prédiction est une fonction (mesurable) $f : \mathcal{X} \to \mathcal{Y}$ qui associe la sortie $f(x)$ à l'entrée $x \in \mathcal{X}$

· Pour l'instant, on suppose que f ne dépend des paramètres (modèle non paramétrique)

Rem : Dans la communauté du machine learning, la distinction entre y (élément déterministe) et Y (variable aléatoire) appartenant à $\mathcal{Y}$ est rarement explicitée. Il faut toujours faire référence au contexte

Qualité de prédiction

· Pour l'ajustement d'un modèle ML, on cherche souvent à minimiser une fonction de perte

· Une fonction $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ est une fonction de perte si

$$\ell(y, y') = \begin{cases} 0, & \text{si } y = y', \\ e, & \text{sinon,} \end{cases}$$

avec $e > 0$.

Exemple.

- $\ell(y, y') = |y - y'|^q$ en régression (perte absolue si $q = 1$, perte quadratique si $q = 2$)
- $\ell(y, y') = \mathbb{1}_{y \neq y'}$ en discrimination binaire

· Le risque (ou l'erreur de généralisation) d'une règle de prédiction f est défini par

$$R_P(f) = \mathbb{E}_{(X,Y)\sim P}[\ell(Y, f(X))]$$

Règles de prédiction - régression réelle

· $\mathcal{Y} = \mathbb{R}$ et $\ell(y, y') = |y - y'|^q$ pour tout $q \in \mathbb{N}$

· On appelle fonction de régression la fonction $\eta^* : \mathcal{X} \to \mathcal{Y}$ définie par

$$\eta^*(x_o) = \mathbb{E}_{Y \sim P_Y}[Y|X = x_o]$$

· Soit $\mathcal{F}$ l'ensemble des règles de prédiction possibles. La fonction η^* est une règle de prédiction optimale au sens de la minimisation du risque :

$$R_P(\eta^*) = \inf_{f \in \mathcal{F}} R_p(f)$$

· La règle de régression $\nu^*(x_o) = \text{mediane}[Y|X = x_o]$ vérifie également

$$R_P(\nu^*) = \inf_{f \in \mathcal{F}} R_p(f)$$

Règles de prédiction - discrimination binaire

· $\mathcal{Y} = \{-1, 1\}$ et $\ell(y, y') = \mathbb{1}_{y \neq y'}$

· On appelle règle de Bayes toute fonction ϕ^* de $\mathcal{F}$ telle que pour tout $x_0 \in \mathcal{X}$,

$$\mathbb{P}(Y = \phi^*(x_0)|X = x_0) = \max_{y' \in \mathcal{Y}} \mathbb{P}(Y = y'|X = x_0)$$

· Si ϕ^* est une règle de Bayes, alors $R_P(\phi^*) = \inf_{f \in \mathcal{F}} R_p(f)$

· La règle de discrimination plug-in définie par

$$\phi_{\eta^*}(x) = \text{sign}(\eta^*(x)) = \mathbb{1}_{\eta^*(x) \geq 0} - \mathbb{1}_{\eta^*(x) < 0},$$

est une règle de Bayes

Risque moyen

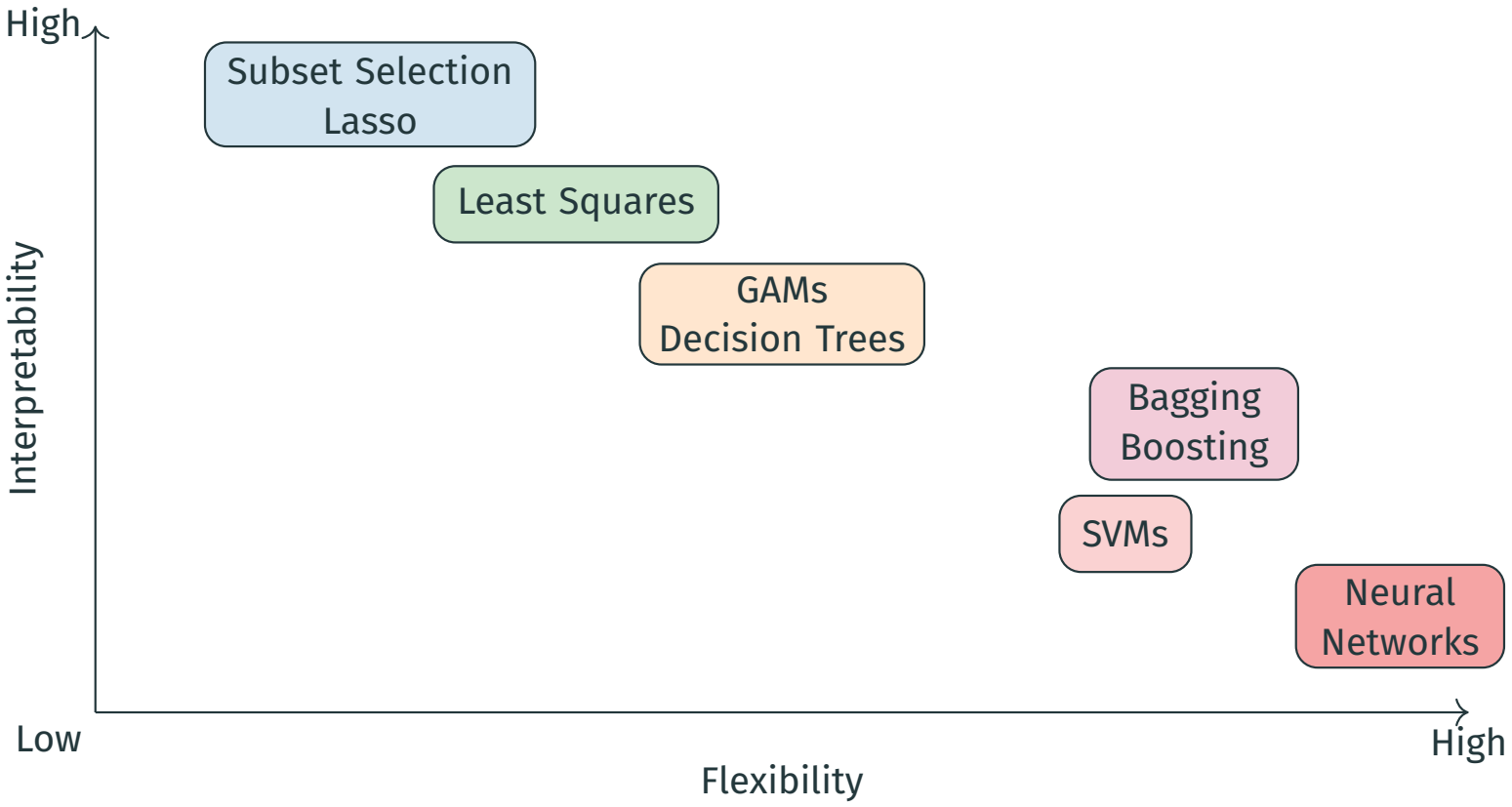
· Le risque moyen de $\widehat{y}$, construit avec D_n, est

$$\begin{aligned}\mathcal{R}_P(\widehat{f}) &= \mathbb{E}_{D_n \sim P^{\otimes n}}[R_P(\widehat{f})] \\ &= \mathbb{E}_{D_n \sim P^{\otimes n}}[\mathbb{E}_{(X,Y)\sim P}[\ell(Y, \widehat{f}(X))]]\end{aligned}$$

· Dans l'ajustement du modèle ML, on cherche $\widehat{f}$ t.q. $\mathcal{R}_P(\widehat{f})$ soit le plus petit possible

· Le risque moyen $\mathcal{R}_P(\widehat{f})$ dépend de P inconnu. Comment faire en pratique ?

Des méthodes, des compromis



- Précision vs. interprétation
- Parcimonie vs. boîte noire (modèles simples impliquent peu de paramètres)
- Ajustement vs. sur-ajustement vs. sous-ajustement (comment savoir ?)

Des méthodes, des compromis

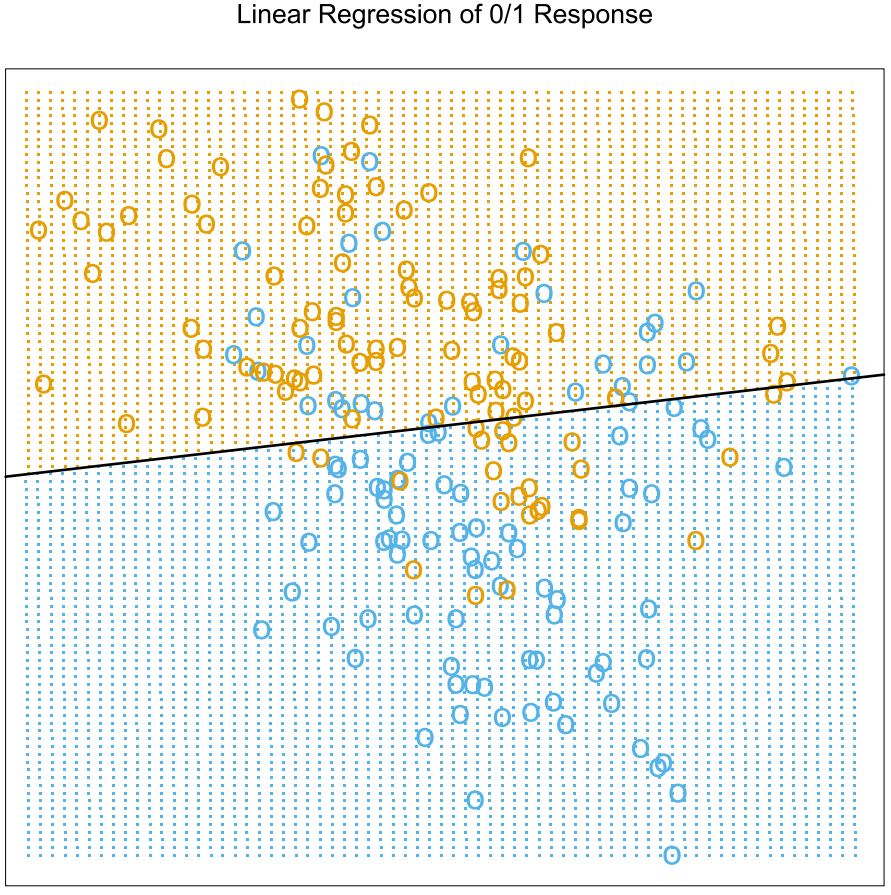


FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (*BLUE $= 0$, ORANGE $= 1$*), and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta} = 0.5$*. The orange shaded region denotes that part of input space classified as* ORANGE*, while the blue region is classified as* BLUE*.*

[Hastie et al., 2009] (The Elements of Statistical Learning)

Des méthodes, des compromis

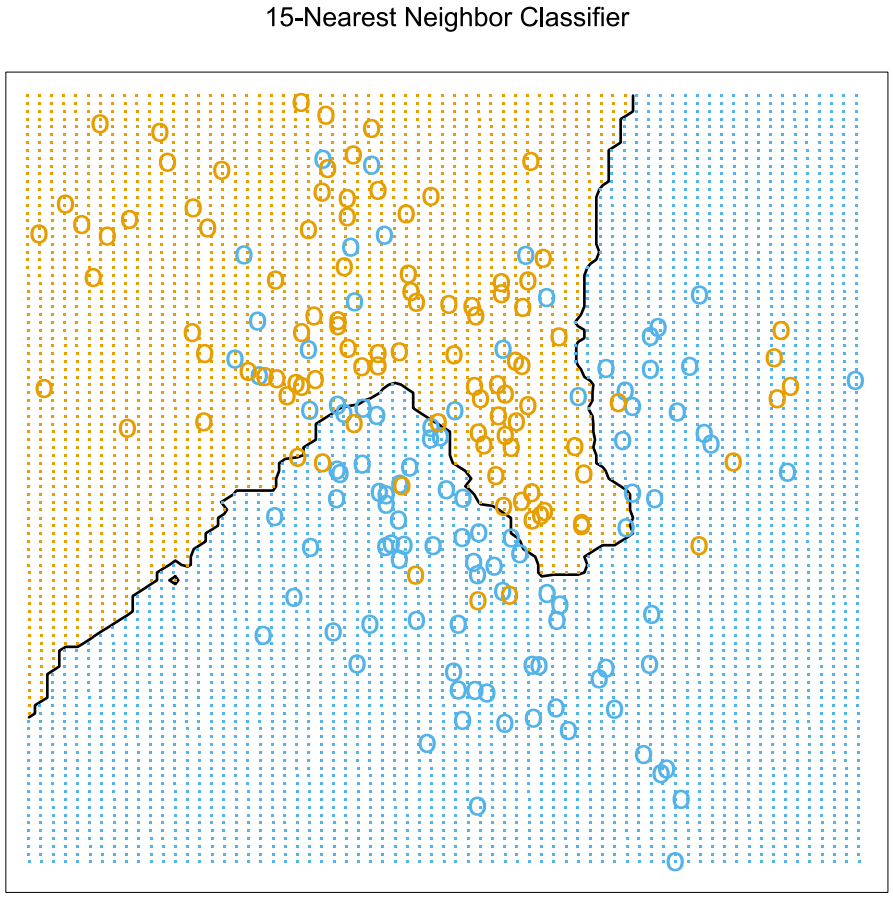


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

[Hastie et al., 2009] (The Elements of Statistical Learning)

Des méthodes, des compromis

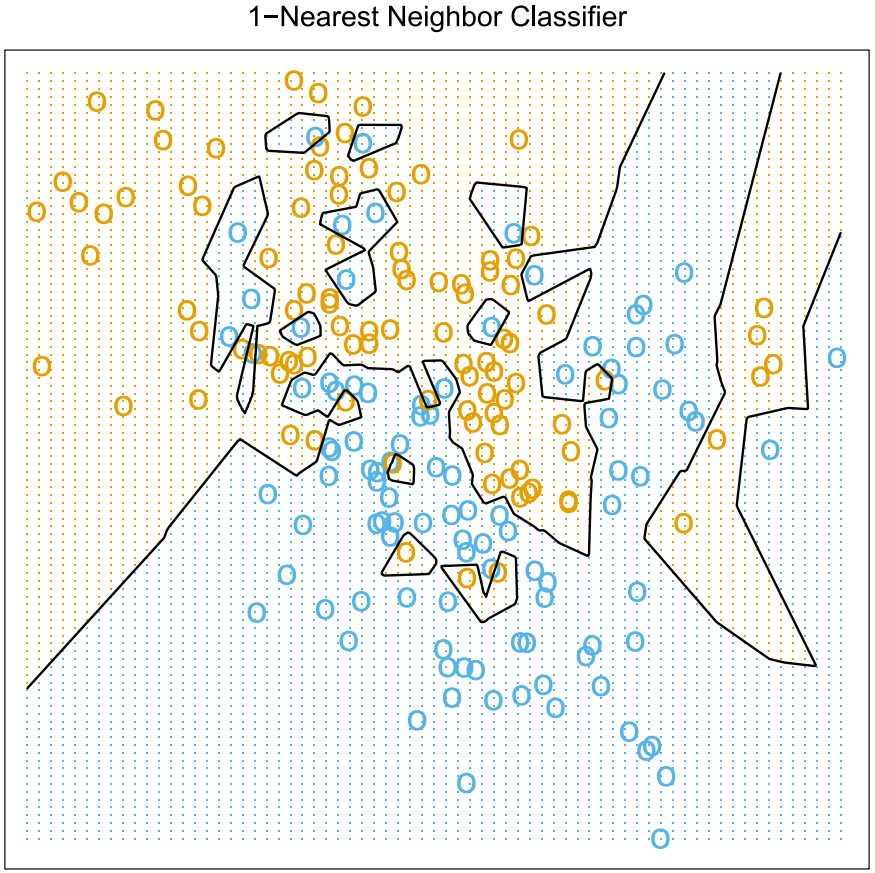


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by* 1*-nearest-neighbor classification.*

[Hastie et al., 2009] (The Elements of Statistical Learning)

Des méthodes, des compromis

- Il n'existe pas de méthode universellement meilleure que les autres
- Sélectionner une approche nécessite de savoir les comparer, *i.e.*, estimer le risque de plusieurs règles
- Le choix de la méthode dépend aussi des objectifs du statisticien / ingénieur ML (interprétation)
- Pour chaque approche : de nouveaux paramètres à ajuster

Validation croisée

Validation croisée (*cross-validation, CV*)

· Considérons un modèle non paramétrique $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$, avec règle de prédiction $\widehat{f}$, et un ensemble de données d'apprentissage

$$D_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

· Le risque empirique (ou risque apparent) du modèle $\mathcal{M}_n$ est défini par

$$\widehat{R}_n(\widehat{f}) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \widehat{y}_i),$$

où $\widehat{y}_i := \widehat{f}(x_i)$ pour tout $i \in \{1, \ldots, n\}$

Validation croisée (*cross-validation, CV*)

· Pour construire de bons modèles, nous souhaitons donc utiliser autant de données disponibles que possible pour l'ajustement

· Cependant, dans de nombreuses applications, la quantité de données disponibles pour l'apprentissage (entraînement) et la validation (test) est limitée

· Il est souvent difficile de disposer d'un ensemble de données dédié exclusivement au test, ce qui limite la comparaison des modèles

· La validation croisée cherche à évaluer la performance des modèles qu'à partir de données d'apprentissage D_n

· Elle peut également être utilisée pour estimer les paramètres des modèles paramétriques (*e.g.*, le paramètre de régularisation de la régression Ridge)

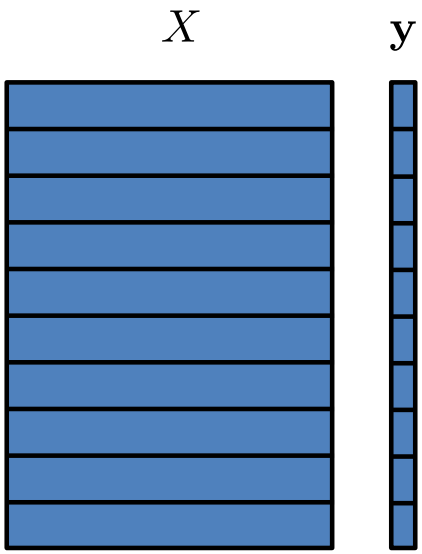
Validation croisée "K-fold" ($K = 10$)

· C'est la méthode d'apprentissage la plus couramment utilisée pour estimer l'erreur de test

· L'idée est de partitionner les données en K groupes de même taille

$$D_n = \bigsqcup_{\kappa \in \{1,\dots,K\}} D_{n,\kappa} \quad \text{(union des ensembles disjoints)}$$

· Ensuite, on prend la i-ème partition, pour tout $i = \{1, \dots, K\}$, pour tester un modèle $\mathcal{M}_i$



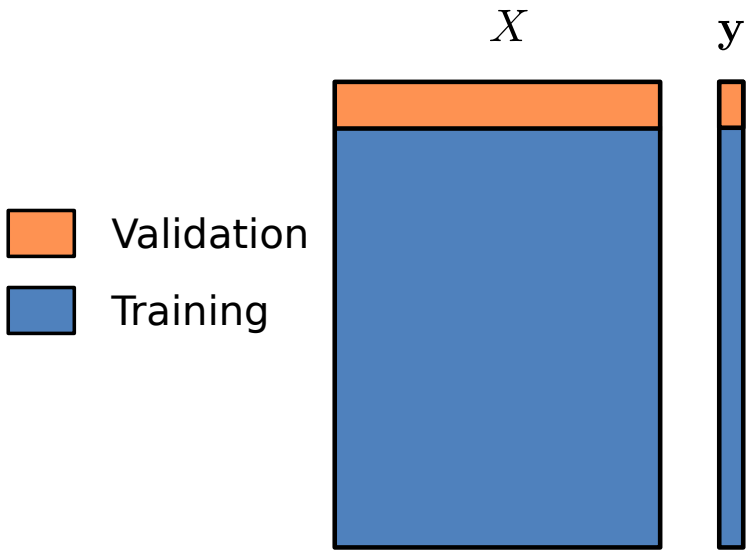
Validation croisée "K-fold" ($K = 10$)

· C'est la méthode d'apprentissage la plus couramment utilisée pour estimer l'erreur de test

· L'idée est de partitionner les données en K groupes de même taille

$$D_n = \bigsqcup_{\kappa \in \{1,\dots,K\}} D_{n,\kappa} \quad \text{(union des ensembles disjoints)}$$

· Ensuite, on prend la i-ème partition, pour tout $i = \{1, \dots, K\}$, pour tester un modèle $\mathcal{M}_i$



$$i = 1$$

1. A partir des données d'apprentissage

$$D_{n,-i} = \bigsqcup_{\substack{\kappa \in \{1,\dots,K\} \\ \kappa \neq i}} D_{n,\kappa},$$

définir la règle $\widehat{f}_i$

2. Calculer $\widehat{R}_{n_i}(\widehat{f}_i)$ sur les données de test (validation) $D_{n,i}$

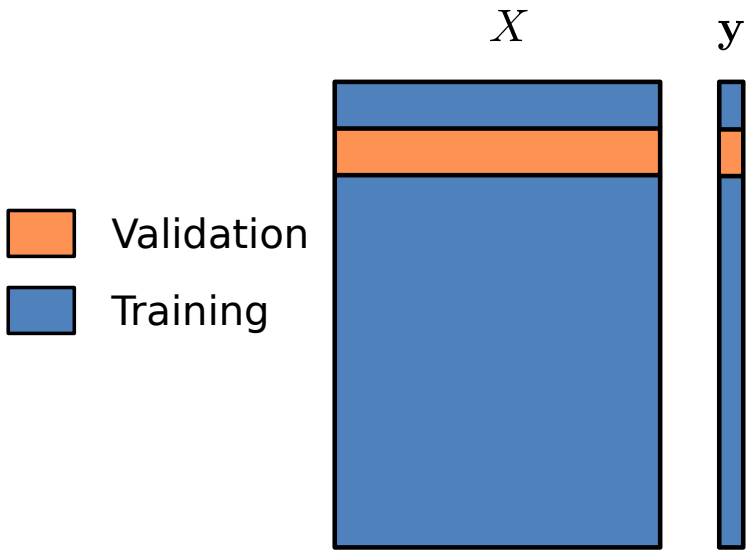
Validation croisée "K-fold" ($K = 10$)

· C'est la méthode d'apprentissage la plus couramment utilisée pour estimer l'erreur de test

· L'idée est de partitionner les données en K groupes de même taille

$$D_n = \bigsqcup_{\kappa \in \{1,\dots,K\}} D_{n,\kappa} \quad \text{(union des ensembles disjoints)}$$

· Ensuite, on prend la i-ème partition, pour tout $i = \{1, \dots, K\}$, pour tester un modèle $\mathcal{M}_i$



$$i = 2$$

1. A partir des données d'apprentissage

$$D_{n,-i} = \bigsqcup_{\substack{\kappa \in \{1,\dots,K\} \\ \kappa \neq i}} D_{n,\kappa},$$

définir la règle $\widehat{f}_i$

2. Calculer $\widehat{R}_{n_i}(\widehat{f}_i)$ sur les données de test (validation) $D_{n,i}$

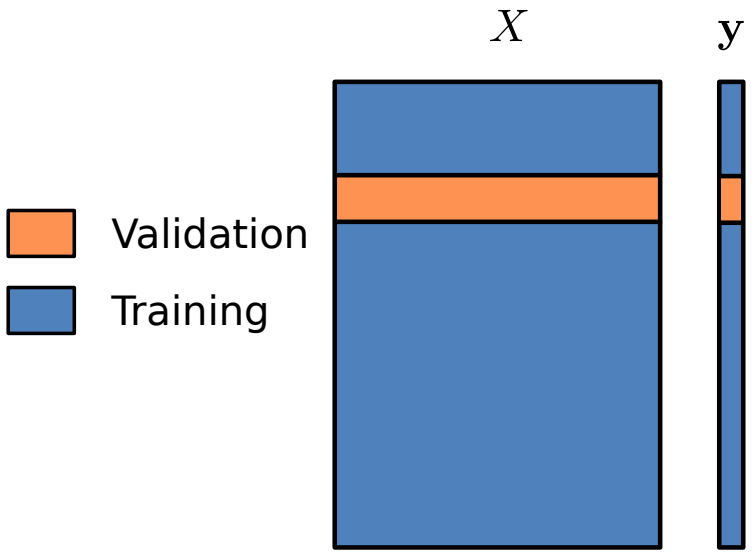
Validation croisée "K-fold" ($K = 10$)

· C'est la méthode d'apprentissage la plus couramment utilisée pour estimer l'erreur de test

· L'idée est de partitionner les données en K groupes de même taille

$$D_n = \bigsqcup_{\kappa \in \{1,\dots,K\}} D_{n,\kappa} \quad \text{(union des ensembles disjoints)}$$

· Ensuite, on prend la i-ème partition, pour tout $i = \{1, \dots, K\}$, pour tester un modèle $\mathcal{M}_i$



$$i = 3$$

1. A partir des données d'apprentissage

$$D_{n,-i} = \bigsqcup_{\substack{\kappa \in \{1,\dots,K\} \\ \kappa \neq i}} D_{n,\kappa},$$

définir la règle $\widehat{f}_i$

2. Calculer $\widehat{R}_{n_i}(\widehat{f}_i)$ sur les données de test (validation) $D_{n,i}$

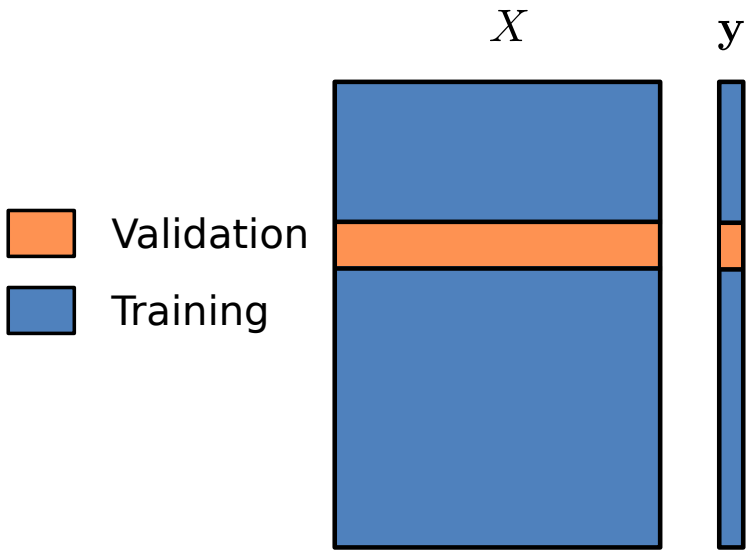
Validation croisée "K-fold" ($K = 10$)

· C'est la méthode d'apprentissage la plus couramment utilisée pour estimer l'erreur de test

· L'idée est de partitionner les données en K groupes de même taille

$$D_n = \bigsqcup_{\kappa \in \{1,\dots,K\}} D_{n,\kappa} \quad \text{(union des ensembles disjoints)}$$

· Ensuite, on prend la i-ème partition, pour tout $i = \{1, \dots, K\}$, pour tester un modèle $\mathcal{M}_i$



$$i = 4$$

1. A partir des données d'apprentissage

$$D_{n,-i} = \bigsqcup_{\substack{\kappa \in \{1,\dots,K\} \\ \kappa \neq i}} D_{n,\kappa},$$

définir la règle $\widehat{f}_i$

2. Calculer $\widehat{R}_{n_i}(\widehat{f}_i)$ sur les données de test (validation) $D_{n,i}$

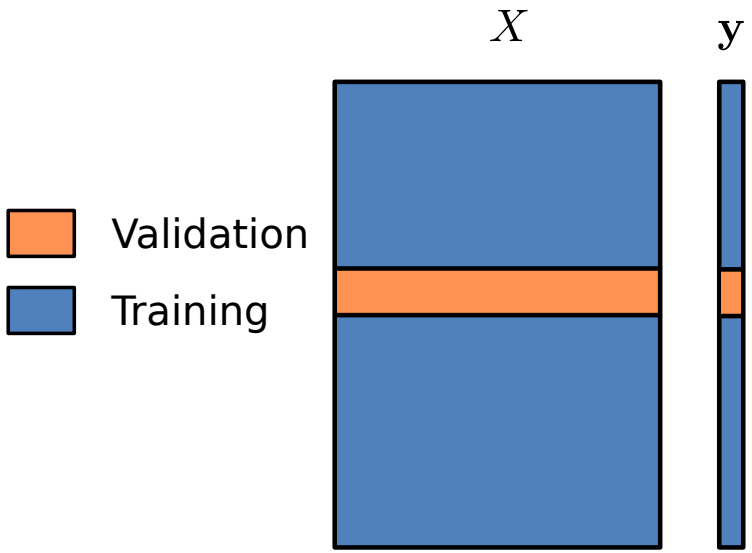
Validation croisée "K-fold" ($K = 10$)

· C'est la méthode d'apprentissage la plus couramment utilisée pour estimer l'erreur de test

· L'idée est de partitionner les données en K groupes de même taille

$$D_n = \bigsqcup_{\kappa \in \{1,\dots,K\}} D_{n,\kappa} \quad \text{(union des ensembles disjoints)}$$

· Ensuite, on prend la i-ème partition, pour tout $i = \{1, \dots, K\}$, pour tester un modèle $\mathcal{M}_i$



$$i = 5$$

1. A partir des données d'apprentissage

$$D_{n,-i} = \bigsqcup_{\substack{\kappa \in \{1,\dots,K\} \\ \kappa \neq i}} D_{n,\kappa},$$

définir la règle $\widehat{f}_i$

2. Calculer $\widehat{R}_{n_i}(\widehat{f}_i)$ sur les données de test (validation) $D_{n,i}$

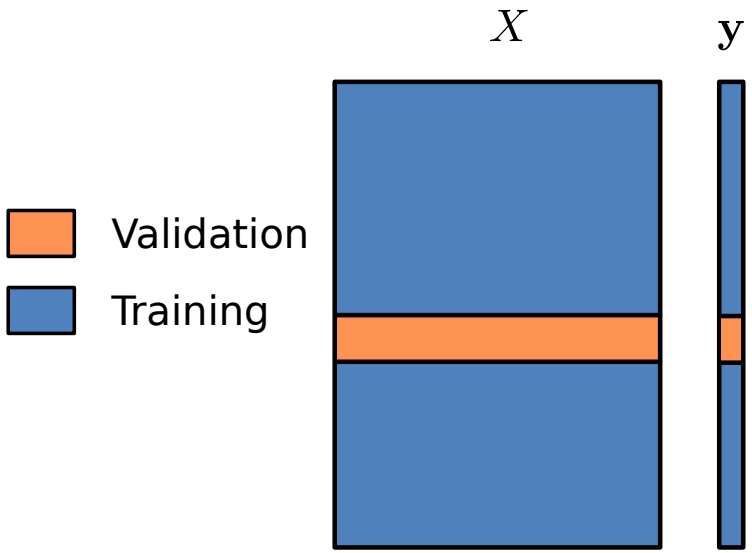
Validation croisée "K-fold" ($K = 10$)

· C'est la méthode d'apprentissage la plus couramment utilisée pour estimer l'erreur de test

· L'idée est de partitionner les données en K groupes de même taille

$$D_n = \bigsqcup_{\kappa \in \{1,\dots,K\}} D_{n,\kappa} \quad \text{(union des ensembles disjoints)}$$

· Ensuite, on prend la i-ème partition, pour tout $i = \{1, \dots, K\}$, pour tester un modèle $\mathcal{M}_i$



$$i = 6$$

1. A partir des données d'apprentissage

$$D_{n,-i} = \bigsqcup_{\substack{\kappa \in \{1,\dots,K\} \\ \kappa \neq i}} D_{n,\kappa},$$

définir la règle $\widehat{f}_i$

2. Calculer $\widehat{R}_{n_i}(\widehat{f}_i)$ sur les données de test (validation) $D_{n,i}$

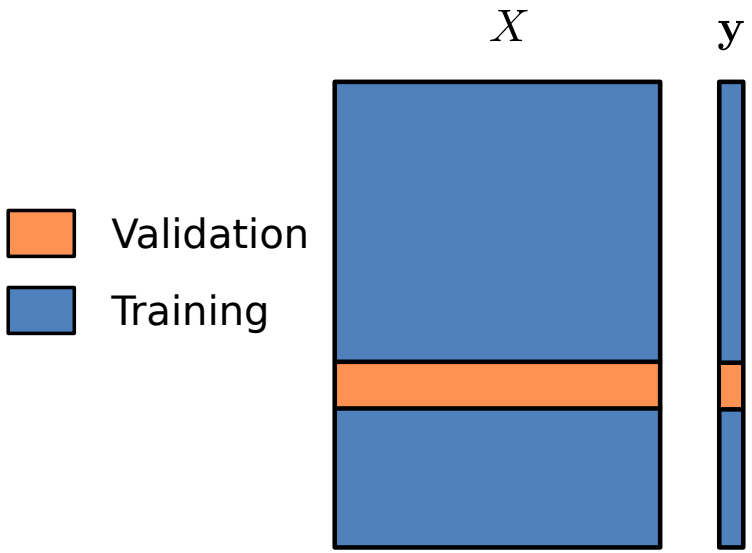
Validation croisée "K-fold" ($K = 10$)

· C'est la méthode d'apprentissage la plus couramment utilisée pour estimer l'erreur de test

· L'idée est de partitionner les données en K groupes de même taille

$$D_n = \bigsqcup_{\kappa \in \{1,\dots,K\}} D_{n,\kappa} \quad \text{(union des ensembles disjoints)}$$

· Ensuite, on prend la i-ème partition, pour tout $i = \{1, \dots, K\}$, pour tester un modèle $\mathcal{M}_i$



$$i = 7$$

1. A partir des données d'apprentissage

$$D_{n,-i} = \bigsqcup_{\substack{\kappa \in \{1,\dots,K\} \\ \kappa \neq i}} D_{n,\kappa},$$

définir la règle $\widehat{f}_i$

2. Calculer $\widehat{R}_{n_i}(\widehat{f}_i)$ sur les données de test (validation) $D_{n,i}$

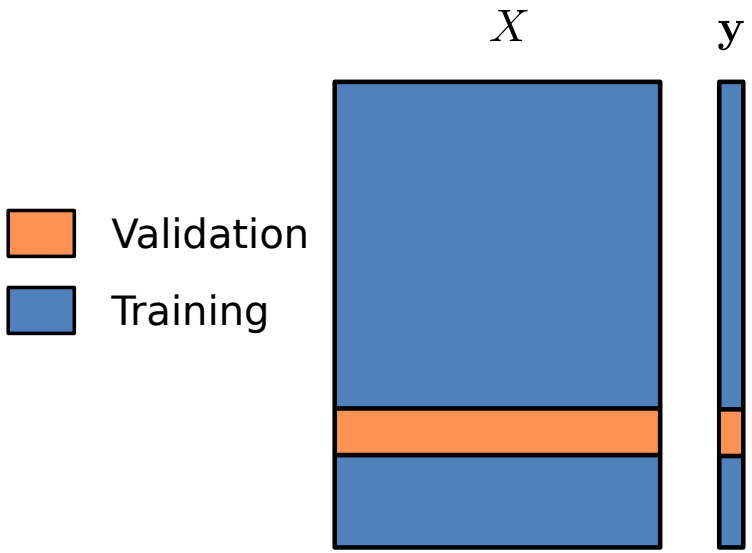
Validation croisée "K-fold" ($K = 10$)

· C'est la méthode d'apprentissage la plus couramment utilisée pour estimer l'erreur de test

· L'idée est de partitionner les données en K groupes de même taille

$$D_n = \bigsqcup_{\kappa \in \{1,\dots,K\}} D_{n,\kappa} \quad \text{(union des ensembles disjoints)}$$

· Ensuite, on prend la i-ème partition, pour tout $i = \{1, \dots, K\}$, pour tester un modèle $\mathcal{M}_i$



$$i = 8$$

1. A partir des données d'apprentissage

$$D_{n,-i} = \bigsqcup_{\substack{\kappa \in \{1,\dots,K\} \\ \kappa \neq i}} D_{n,\kappa},$$

définir la règle $\widehat{f}_i$

2. Calculer $\widehat{R}_{n_i}(\widehat{f}_i)$ sur les données de test (validation) $D_{n,i}$

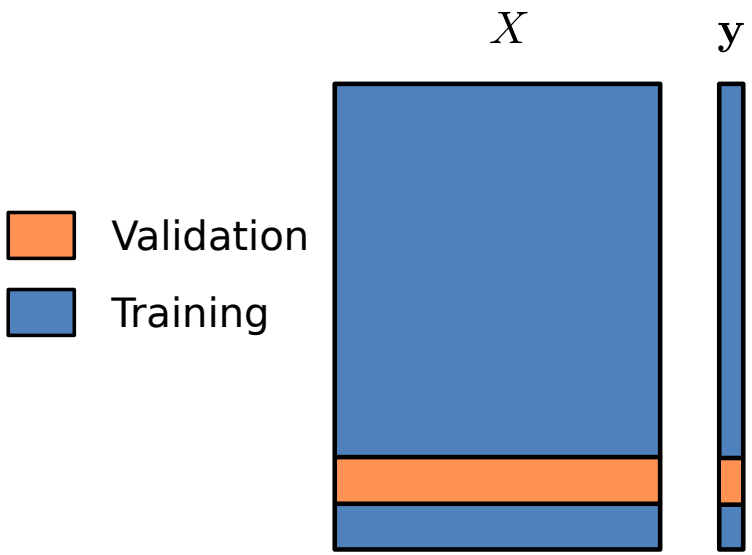
Validation croisée "K-fold" ($K = 10$)

· C'est la méthode d'apprentissage la plus couramment utilisée pour estimer l'erreur de test

· L'idée est de partitionner les données en K groupes de même taille

$$D_n = \bigsqcup_{\kappa \in \{1,\dots,K\}} D_{n,\kappa} \quad \text{(union des ensembles disjoints)}$$

· Ensuite, on prend la i-ème partition, pour tout $i = \{1, \dots, K\}$, pour tester un modèle $\mathcal{M}_i$



$$i = 9$$

1. A partir des données d'apprentissage

$$D_{n,-i} = \bigsqcup_{\substack{\kappa \in \{1,\dots,K\} \\ \kappa \neq i}} D_{n,\kappa},$$

définir la règle $\widehat{f}_i$

2. Calculer $\widehat{R}_{n_i}(\widehat{f}_i)$ sur les données de test (validation) $D_{n,i}$

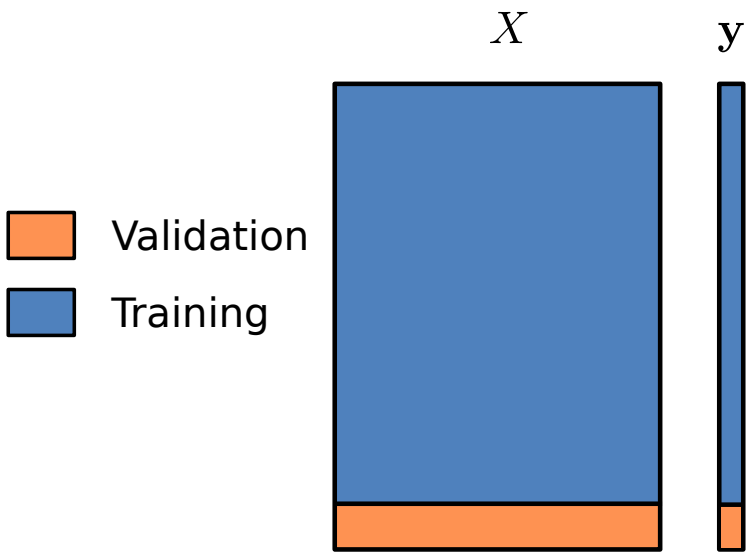
Validation croisée "K-fold" ($K = 10$)

· C'est la méthode d'apprentissage la plus couramment utilisée pour estimer l'erreur de test

· L'idée est de partitionner les données en K groupes de même taille

$$D_n = \bigsqcup_{\kappa \in \{1,\dots,K\}} D_{n,\kappa} \quad \text{(union des ensembles disjoints)}$$

· Ensuite, on prend la i-ème partition, pour tout $i = \{1, \dots, K\}$, pour tester un modèle $\mathcal{M}_i$



$$i = 10$$

1. A partir des données d'apprentissage

$$D_{n,-i} = \bigsqcup_{\substack{\kappa \in \{1,\dots,K\} \\ \kappa \neq i}} D_{n,\kappa},$$

définir la règle $\widehat{f}_i$

2. Calculer $\widehat{R}_{n_i}(\widehat{f}_i)$ sur les données de test (validation) $D_{n,i}$

Risque CV. $\widehat{R}_{n,CV} = \frac{1}{K} \sum_{i=1}^{K} \widehat{R}_{n_i}(\widehat{f}_i)$

Validation croisée "K-fold"

· Cas extrême de validation croisée

- $K = 1$: impossible car un seul bloc
- $K = n$ (stratégie *leave-one-out*, *cf. Jackknife*) : autant de blocs que de variables (trop de calculs mais possible de les paralléliser)

Conseils pratiques

- Choix habituels : $K = 5, 10$
- Lorsque qu'on a peu de données : $K = 1$ (*leave-one-out*)
- "Randomiser les observations" : observations dans un ordre aléatoire afin éviter des blocs de données trop similaires (chaque sous-bloc doit être représentatif de l'ensemble)

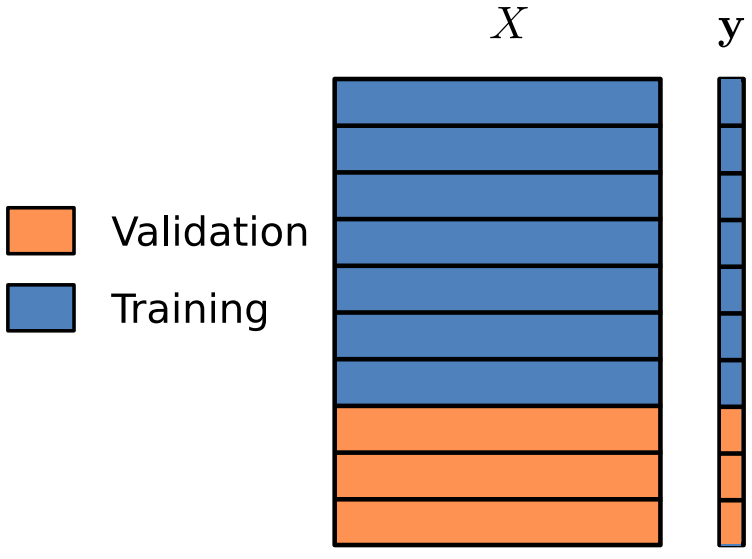
Validation croisée "*Hold-out*"

· La méthode consiste à partitionner (aléatoirement) D_n en 2 groupes : données d'apprentissage $D_{p,\text{train}}$ (taille p) et de test $D_{n-p,\text{test}}$ (taille $n-p$)

$$D_n = D_{p,\text{train}} \sqcup D_{n-p,\text{test}}$$

· Ensuite, on prend $D_{p,\text{train}}$ pour ajuster un modèle $\mathcal{M}_{D_{p,\text{train}}}$

· Cette procédure se répète N_{reps} (*replicates*) fois en utilisant des partitions aléatoires



$i = 1$

1. $\mathcal{I}_i \sim \text{sample}(n, p)$ (permutation aléatoire)
2. $D^{(i)}_{p,\text{train}} = D_{n,\mathcal{I}_i}$, et $D^{(i)}_{n-p,\text{test}} = D_{n,-\mathcal{I}_i}$
3. A partir des données d'apprentissage $D^{(i)}_{p,\text{train}}$, définir la règle $\widehat{f}_i$
4. Calculer $\widehat{R}_{n-p}(\widehat{f}_i)$ sur l'ensemble de test $D^{(i)}_{n-p,\text{test}}$

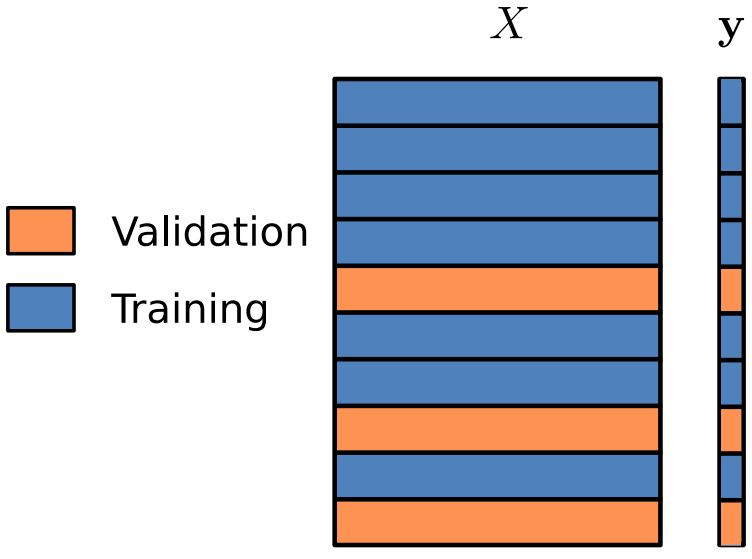
Validation croisée "*Hold-out*"

· La méthode consiste à partitionner (aléatoirement) D_n en 2 groupes : données d'apprentissage $D_{p,\text{train}}$ (taille p) et de test $D_{n-p,\text{test}}$ (taille $n-p$)

$$D_n = D_{p,\text{train}} \sqcup D_{n-p,\text{test}}$$

· Ensuite, on prend $D_{p,\text{train}}$ pour ajuster un modèle $\mathcal{M}_{D_{p,\text{train}}}$

· Cette procédure se répète N_{reps} (*replicates*) fois en utilisant des partitions aléatoires



$$i = 2$$

1. $\mathcal{I}_i \sim \text{sample}(n, p)$ (permutation aléatoire)
2. $D^{(i)}_{p,\text{train}} = D_{n,\mathcal{I}_i}$, et $D^{(i)}_{n-p,\text{test}} = D_{n,-\mathcal{I}_i}$
3. A partir des données d'apprentissage $D^{(i)}_{p,\text{train}}$, définir la règle $\widehat{f}_i$
4. Calculer $\widehat{R}_{n-p}(\widehat{f}_i)$ sur l'ensemble de test $D^{(i)}_{n-p,\text{test}}$

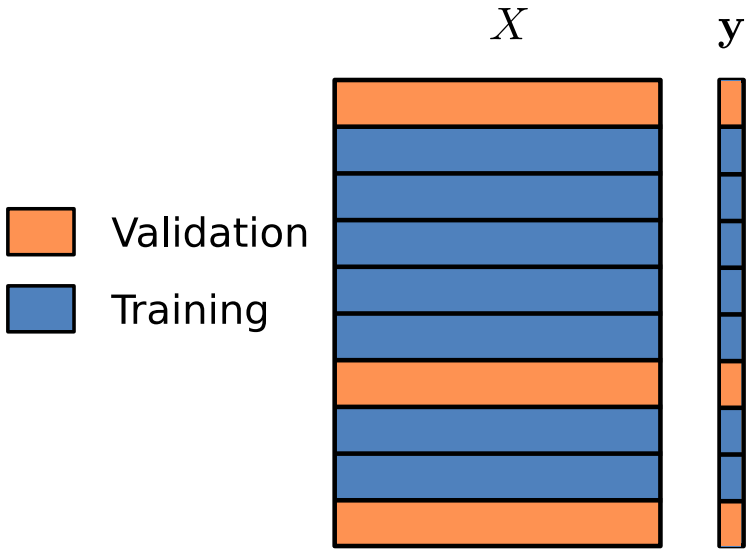
Validation croisée "*Hold-out*"

· La méthode consiste à partitionner (aléatoirement) D_n en 2 groupes : données d'apprentissage $D_{p,\text{train}}$ (taille p) et de test $D_{n-p,\text{test}}$ (taille $n-p$)

$$D_n = D_{p,\text{train}} \sqcup D_{n-p,\text{test}}$$

· Ensuite, on prend $D_{p,\text{train}}$ pour ajuster un modèle $\mathcal{M}_{D_{p,\text{train}}}$

· Cette procédure se répète N_{reps} (*replicates*) fois en utilisant des partitions aléatoires



$$i = 3$$

1. $\mathcal{I}_i \sim \text{sample}(n, p)$ (permutation aléatoire)
2. $D^{(i)}_{p,\text{train}} = D_{n,\mathcal{I}_i}$, et $D^{(i)}_{n-p,\text{test}} = D_{n,-\mathcal{I}_i}$
3. A partir des données d'apprentissage $D^{(i)}_{p,\text{train}}$, définir la règle $\widehat{f}_i$
4. Calculer $\widehat{R}_{n-p}(\widehat{f}_i)$ sur l'ensemble de test $D^{(i)}_{n-p,\text{test}}$

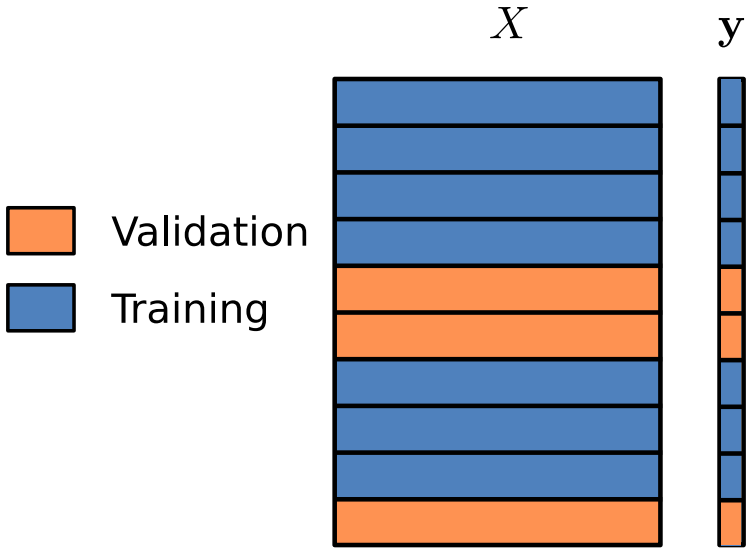
Validation croisée "*Hold-out*"

· La méthode consiste à partitionner (aléatoirement) D_n en 2 groupes : données d'apprentissage $D_{p,\text{train}}$ (taille p) et de test $D_{n-p,\text{test}}$ (taille $n-p$)

$$D_n = D_{p,\text{train}} \sqcup D_{n-p,\text{test}}$$

· Ensuite, on prend $D_{p,\text{train}}$ pour ajuster un modèle $\mathcal{M}_{D_{p,\text{train}}}$

· Cette procédure se répète N_{reps} (*replicates*) fois en utilisant des partitions aléatoires



$i = 4$

1. $\mathcal{I}_i \sim \text{sample}(n, p)$ (permutation aléatoire)
2. $D^{(i)}_{p,\text{train}} = D_{n,\mathcal{I}_i}$, et $D^{(i)}_{n-p,\text{test}} = D_{n,-\mathcal{I}_i}$
3. A partir des données d'apprentissage $D^{(i)}_{p,\text{train}}$, définir la règle $\widehat{f}_i$
4. Calculer $\widehat{R}_{n-p}(\widehat{f}_i)$ sur l'ensemble de test $D^{(i)}_{n-p,\text{test}}$

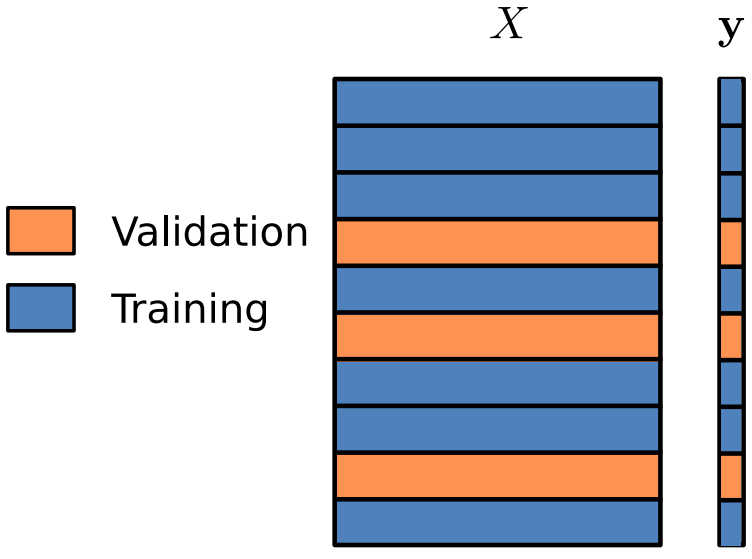
Validation croisée "*Hold-out*"

· La méthode consiste à partitionner (aléatoirement) D_n en 2 groupes : données d'apprentissage $D_{p,\text{train}}$ (taille p) et de test $D_{n-p,\text{test}}$ (taille $n-p$)

$$D_n = D_{p,\text{train}} \sqcup D_{n-p,\text{test}}$$

· Ensuite, on prend $D_{p,\text{train}}$ pour ajuster un modèle $\mathcal{M}_{D_{p,\text{train}}}$

· Cette procédure se répète N_{reps} (*replicates*) fois en utilisant des partitions aléatoires



$$i = 5$$

1. $\mathcal{I}_i \sim \text{sample}(n, p)$ (permutation aléatoire)
2. $D^{(i)}_{p,\text{train}} = D_{n,\mathcal{I}_i}$, et $D^{(i)}_{n-p,\text{test}} = D_{n,-\mathcal{I}_i}$
3. A partir des données d'apprentissage $D^{(i)}_{p,\text{train}}$, définir la règle $\widehat{f}_i$
4. Calculer $\widehat{R}_{n-p}(\widehat{f}_i)$ sur l'ensemble de test $D^{(i)}_{n-p,\text{test}}$

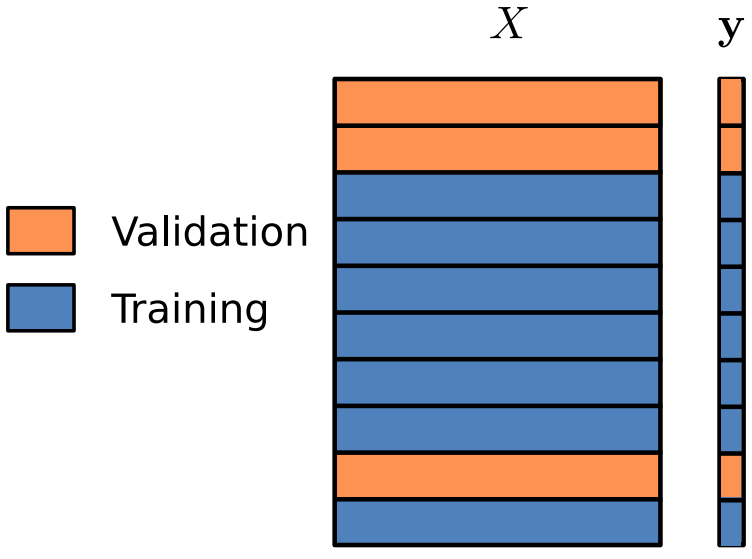
Validation croisée "*Hold-out*"

· La méthode consiste à partitionner (aléatoirement) D_n en 2 groupes : données d'apprentissage $D_{p,\text{train}}$ (taille p) et de test $D_{n-p,\text{test}}$ (taille $n-p$)

$$D_n = D_{p,\text{train}} \sqcup D_{n-p,\text{test}}$$

· Ensuite, on prend $D_{p,\text{train}}$ pour ajuster un modèle $\mathcal{M}_{D_{p,\text{train}}}$

· Cette procédure se répète N_{reps} (*replicates*) fois en utilisant des partitions aléatoires



$i = 6$

1. $\mathcal{I}_i \sim \text{sample}(n, p)$ (permutation aléatoire)
2. $D^{(i)}_{p,\text{train}} = D_{n,\mathcal{I}_i}$, et $D^{(i)}_{n-p,\text{test}} = D_{n,-\mathcal{I}_i}$
3. A partir des données d'apprentissage $D^{(i)}_{p,\text{train}}$, définir la règle $\widehat{f}_i$
4. Calculer $\widehat{R}_{n-p}(\widehat{f}_i)$ sur l'ensemble de test $D^{(i)}_{n-p,\text{test}}$

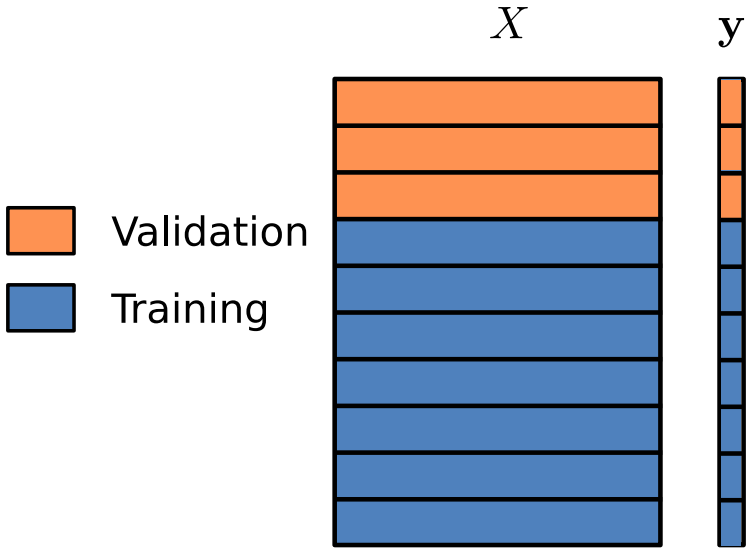
Validation croisée "*Hold-out*"

· La méthode consiste à partitionner (aléatoirement) D_n en 2 groupes : données d'apprentissage $D_{p,\text{train}}$ (taille p) et de test $D_{n-p,\text{test}}$ (taille $n-p$)

$$D_n = D_{p,\text{train}} \sqcup D_{n-p,\text{test}}$$

· Ensuite, on prend $D_{p,\text{train}}$ pour ajuster un modèle $\mathcal{M}_{D_{p,\text{train}}}$

· Cette procédure se répète N_{reps} (*replicates*) fois en utilisant des partitions aléatoires



$$i = 7$$

1. $\mathcal{I}_i \sim \text{sample}(n, p)$ (permutation aléatoire)
2. $D^{(i)}_{p,\text{train}} = D_{n,\mathcal{I}_i}$, et $D^{(i)}_{n-p,\text{test}} = D_{n,-\mathcal{I}_i}$
3. A partir des données d'apprentissage $D^{(i)}_{p,\text{train}}$, définir la règle $\widehat{f}_i$
4. Calculer $\widehat{R}_{n-p}(\widehat{f}_i)$ sur l'ensemble de test $D^{(i)}_{n-p,\text{test}}$

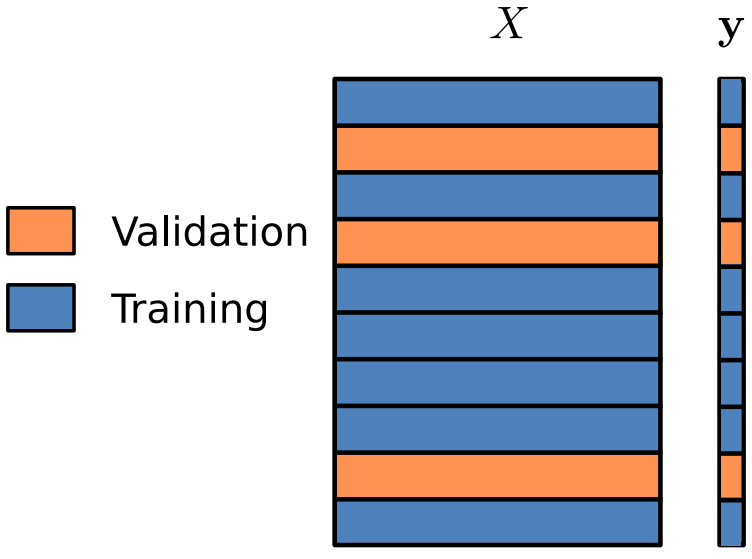
Validation croisée "*Hold-out*"

· La méthode consiste à partitionner (aléatoirement) D_n en 2 groupes : données d'apprentissage $D_{p,\text{train}}$ (taille p) et de test $D_{n-p,\text{test}}$ (taille $n-p$)

$$D_n = D_{p,\text{train}} \sqcup D_{n-p,\text{test}}$$

· Ensuite, on prend $D_{p,\text{train}}$ pour ajuster un modèle $\mathcal{M}_{D_{p,\text{train}}}$

· Cette procédure se répète N_{reps} (*replicates*) fois en utilisant des partitions aléatoires



$$i = 8$$

1. $\mathcal{I}_i \sim \text{sample}(n, p)$ (permutation aléatoire)
2. $D^{(i)}_{p,\text{train}} = D_{n,\mathcal{I}_i}$, et $D^{(i)}_{n-p,\text{test}} = D_{n,-\mathcal{I}_i}$
3. A partir des données d'apprentissage $D^{(i)}_{p,\text{train}}$, définir la règle $\widehat{f}_i$
4. Calculer $\widehat{R}_{n-p}(\widehat{f}_i)$ sur l'ensemble de test $D^{(i)}_{n-p,\text{test}}$

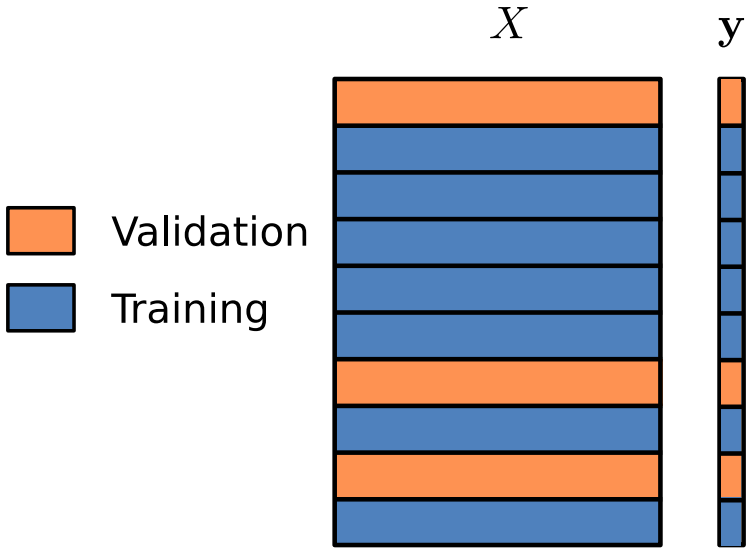
Validation croisée "*Hold-out*"

· La méthode consiste à partitionner (aléatoirement) D_n en 2 groupes : données d'apprentissage $D_{p,\text{train}}$ (taille p) et de test $D_{n-p,\text{test}}$ (taille $n-p$)

$$D_n = D_{p,\text{train}} \sqcup D_{n-p,\text{test}}$$

· Ensuite, on prend $D_{p,\text{train}}$ pour ajuster un modèle $\mathcal{M}_{D_{p,\text{train}}}$

· Cette procédure se répète N_{reps} (*replicates*) fois en utilisant des partitions aléatoires



$i = 9$

1. $\mathcal{I}_i \sim \text{sample}(n, p)$ (permutation aléatoire)
2. $D^{(i)}_{p,\text{train}} = D_{n,\mathcal{I}_i}$, et $D^{(i)}_{n-p,\text{test}} = D_{n,-\mathcal{I}_i}$
3. A partir des données d'apprentissage $D^{(i)}_{p,\text{train}}$, définir la règle $\widehat{f}_i$
4. Calculer $\widehat{R}_{n-p}(\widehat{f}_i)$ sur l'ensemble de test $D^{(i)}_{n-p,\text{test}}$

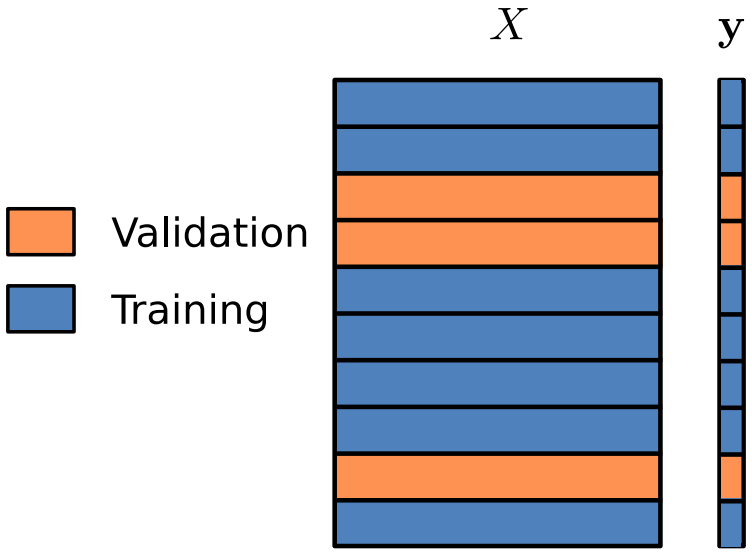
Validation croisée "*Hold-out*"

· La méthode consiste à partitionner (aléatoirement) D_n en 2 groupes : données d'apprentissage $D_{p,\text{train}}$ (taille p) et de test $D_{n-p,\text{test}}$ (taille $n-p$)

$$D_n = D_{p,\text{train}} \sqcup D_{n-p,\text{test}}$$

· Ensuite, on prend $D_{p,\text{train}}$ pour ajuster un modèle $\mathcal{M}_{D_{p,\text{train}}}$

· Cette procédure se répète N_{reps} (*replicates*) fois en utilisant des partitions aléatoires



$i = 10$

1. $\mathcal{I}_i \sim \text{sample}(n, p)$ (permutation aléatoire)
2. $D^{(i)}_{p,\text{train}} = D_{n,\mathcal{I}_i}$, et $D^{(i)}_{n-p,\text{test}} = D_{n,-\mathcal{I}_i}$
3. A partir des données d'apprentissage $D^{(i)}_{p,\text{train}}$, définir la règle $\widehat{f}_i$
4. Calculer $\widehat{R}_{n-p}(\widehat{f}_i)$ sur l'ensemble de test $D^{(i)}_{n-p,\text{test}}$

Risque hold-out. $\widehat{R}_{n,\text{hold-out}} = \dfrac{1}{N_{reps}} \displaystyle\sum_{i=1}^{N_{reps}} \widehat{R}_{n-p}(\widehat{f}_i)$

Validation croisée “*Hold-out*”

· L’estimation du risque via hold-out peut être très variable, et dépend de la chance (ou malchance) dans la construction de $D_{p,\text{train}}$ et $D_{n-p,\text{test}}$

· Si p est petit, l’erreur calculée peut surestimer l’erreur de test d’un modèle ajusté sur l’ensemble des données

Conseils pratiques

- Choix habituels : $p = \lceil 0.6 \times n \rceil, \lceil 0.7 \times n \rceil, \ldots$
- Le choix $N_{reps} > 1$ doit garantir que les résultats obtenus soient “stables” (*e.g.*, variabilité faible du risque empirique $\widehat{R}_{n,\textit{hold-out}}$)
- Lorsque qu’on a peu de données : privilégier la validation de type *leave-one-out*

Méthodes CV classiques

- *K*-fold : KFold
- Hold-out : train_test_split
- Variante pour séries temporelles : TimeSeriesSplit
- Variante pour la classification et pour les cas avec classes déséquilibrées : StratifiedKFold

Plus de détails :
http://scikit-learn.org/stable/modules/cross_validation.html

Pour les données catégorielles. Si on enlève toute les occurrences d'une modalité de la partie apprentissage, on crée une colonne de zéro

- Séparation apprentissage/test adaptée pour équilibrer les *folds*

Références

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, New York, second edition, 2009.

Kevin Murphy. Reinforcement learning: An overview, 2024. URL https://arxiv.org/abs/2412.05265.

Qiang Yang, Yu Zhang, Wenyuan Dai, and Sinno Jialin Pan. *Transfer Learning*. Cambridge University Press, 2020.