

Apprentissage supervisé

Brigitte Gelein

bgelein@ensai.fr

Ensaï 2021-2022



1	Introduction	5
2	L'apprentissage statistique supervisé	6
●	Généralités	7
●	Formalisation du problème	16
●	Choix d'une règle de prédiction	23
●	Estimateurs du risque moyen : validation simple ou croisée	27
●	Vocabulaire anglais - français	45
●	Exemple détaillé sur l'ensemble du cours : Spotify	46
3	K plus proches voisins KNN	57
●	Algorithmes de moyennage local	58
●	K plus proches voisins	59
●	Améliorations possibles - utilisation de noyaux	66
●	Le fléau de la dimension	69
●	Exemple K plus proches voisins avec R	72

4	Classifieur bayésien naïf	78
●	Définition algorithme	79
●	Bonnes performances et optimalité	81
●	Lissage de Laplace	83
●	Ajustement d'autres hyperparamètres	85
●	Exemple Classifieur bayésien naïf avec R	86

Les autres chapitres de ce cours seront présentés dans un autre poly.

Objectifs du cours et des TD-TP :

- Présenter les aspects théoriques et pratiques des méthodes d'apprentissage supervisé
- Faciliter leur mise en oeuvre par l'utilisation d'exemples avec le logiciel R
- Interactivité pour une meilleure assimilation des concepts

Au programme :

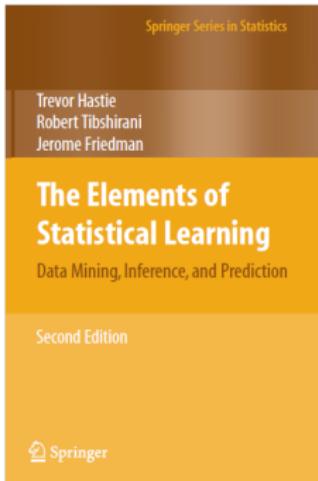
- ① L'apprentissage statistique supervisé
- ② K plus proches voisins (KNN)
- ③ Le classifieur bayésien naïf
- ④ Arbres de décision
- ⑤ Analyses discriminantes (factorielles et bayésiennes)
- ⑥ Comparaison de méthodes (performance...)

L'apprentissage statistique supervisé

Supervised learning

"Vast amounts of data are being generated in many fields, and the statistician's job is to make sense of it all : to extract important patterns and trends, and understand "what the data says." We call this learning from data."

**Trevor Hastie, Robert Tibshirani
Jerome Friedman
Stanford, California, May 2001**



<http://statweb.stanford.edu/~tibs/ElemStatLearn/>

Définition

Apprentissage supervisé

Pour chaque unité statistique i avec $i = 1, \dots, n$, on a une mesure de caractéristiques x_i et mesure d'une ou plusieurs variables d'intérêt y_i appelées aussi réponses associées. Les données sont dites annotées ou étiquetées par la (ou les) variable(s) d'intérêt.

On souhaite ajuster un modèle qui relie la ou les variables réponses Y aux caractéristiques (prédicteurs, features) X .

Deux grandes approches d'apprentissage supervisé (Cornuéjols et Miclet, 2018) :

- **approche en 2 étapes**

- ① inférence pour estimer $P(Y = y|X = x)$,
- ② décision - prédiction de \hat{y} s'appuyant sur $\hat{P}(Y = y|X = x)$.

- **approche directe** - prédiction directe de \hat{y} .

On peut distinguer d'autres types d'apprentissage.

Définition

Apprentissage non supervisé

Pour chaque unité statistique i avec $i = 1, \dots, n$ on a une mesure de caractéristiques x_i mais pas de variables réponses associées.

On peut souhaiter identifier des groupes d'unités statistiques qui se ressemblent au regard des variables caractéristiques.

Donner deux techniques qui répondent à cet objectif :

On peut aussi souhaiter identifier des groupes de variables caractéristiques qui sont liées entre elles.

Définition

Apprentissage semi supervisé : *Pour chaque unité statistique i avec $i = 1, \dots, n$ on a une mesure de caractéristiques x_i .*

Certaines unités statistiques sont étiquetées et d'autres non, (Blum et al.1998).

L'apprentissage statistique supervisé est utilisé dans de nombreux domaines comme le marketing, la finance, l'industrie, la médecine, la statistique d'enquêtes :

- Prévoir les prix d'un stock de marchandises à partir de performances de l'entreprise qui détient ce stock et des indicateurs économiques du marché.
- **Sport** : paris sportifs, optimisation des entraînements et des stratégies, prédiction de blessures.
- Détection automatique de **SPAMs** dans les mails.
- Prévoir la probabilité de **réponse à un questionnaire**.
- Évaluation des **risques-clients** (prêt bancaire).
- Prévoir si un patient qui a été hospitalisé pour un infarctus, **risque d'avoir un second infarctus** en fonction de données démographiques et de mesures cliniques faites sur ce patient.

Exemples de prévisions

Reconnaissance de chiffres manuscrits (MNIST) :

0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9



Quel chiffre ?

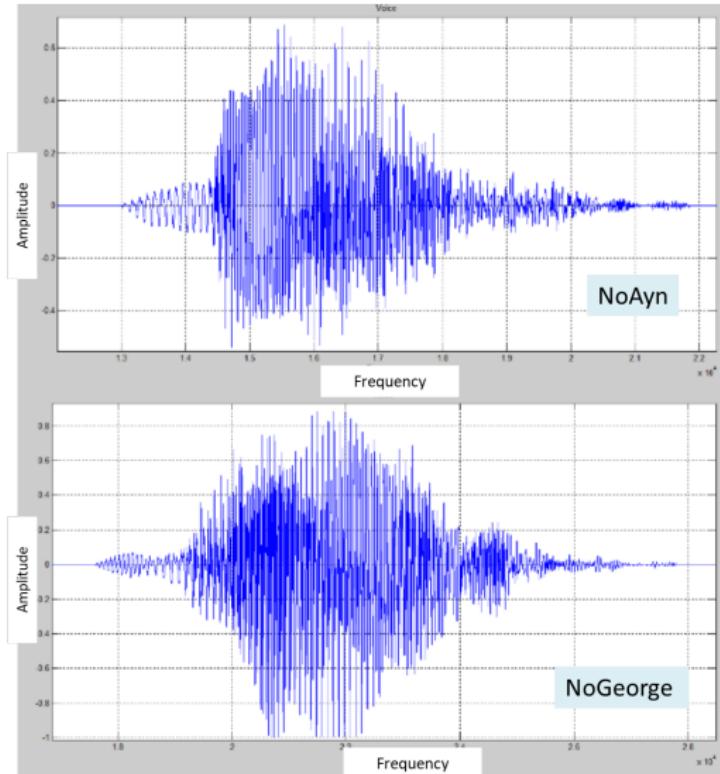
<http://yann.lecun.com/exdb/mnist/>

cité dans "Fondamentaux de l'apprentissage statistique"
par Sylvain Arlot, JES octobre 2016.

- Quelles unités statistiques ? _____
- Quels X et Y ? _____

Exemples de prévisions

Reconnaissance vocale :



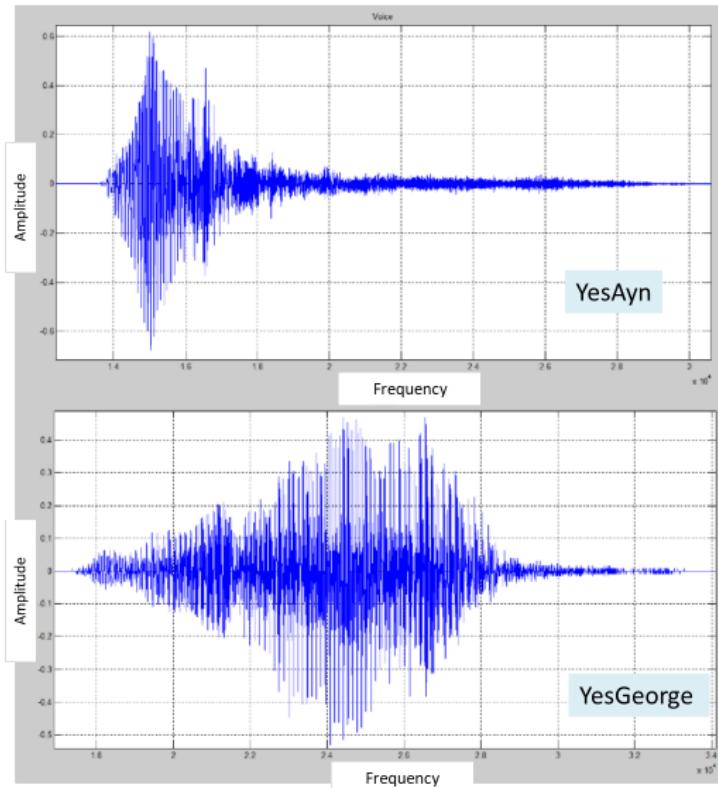
Gerome Jan Llames
"Yes or No Speech
Recognition"

Amplitude:
Intensité de la voix
Décibels – dB
Normalisée entre -1 et 1

Frequency :
Hauteur de la voix
Hertz – Hz
Son aigus ou graves

Exemples de prévisions

Reconnaissance vocale :

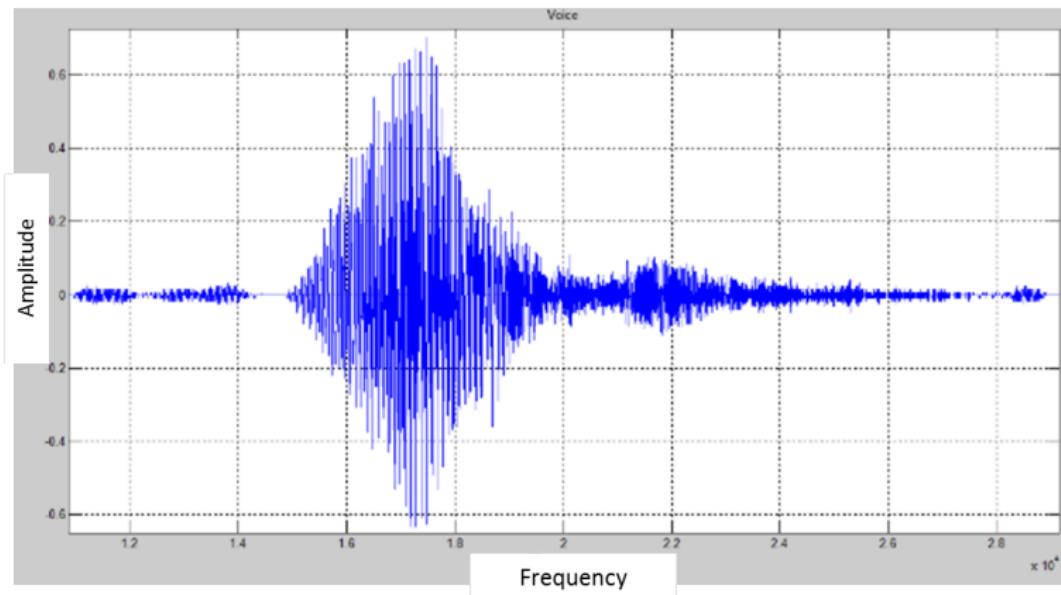


Gerome Jan Llames
"Yes or No Speech
Recognition "

Amplitude:
Intensité de la voix
Décibels – dB
Normalisée entre -1 et 1

Frequency :
Hauteur de la voix
Hertz – Hz
Son aigus ou graves

Reconnaissance vocale :



Yes or No ?

- Quelles unités statistiques ?
- Quels X et Y ?

- ① Extraction des données avec ou sans échantillonnage
- ② Exploration, visualisation, nettoyage, transformations...
- ③ Données manquantes : suppression, imputation, repondération
- ④ Partition de l'échantillon : apprentissage, validation, test
- ⑤ Pour une méthode d'apprentissage :
 - choix de la complexité
 - estimation de paramètres dans le cadre paramétrique
- ⑥ Comparaison des méthodes par erreur de prévision (échantillon test...)
- ⑦ Itération éventuelle (plusieurs échantillons test par exemple)
- ⑧ Choix de la méthode (prévision, interprétabilité).
- ⑨ Ré-estimation du modèle, exploitation

Philippe Besse, *Risque et Choix de Modèles en Apprentissage*, JES octobre 2016

- **Données observées de type entrée-sortie :**
 d_n est l'observation d'un n-échantillon $\mathcal{D}_n = (X_i, Y_i)_{1 \leq i \leq n}$
 $X_i \in \mathcal{X}$: variables explicatives
 $Y_i \in \mathcal{Y}$: variable d'intérêt
- **Hypothèse** : $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d. $\sim P$
avec P une loi conjointe sur $\mathcal{X} \times \mathcal{Y}$ inconnue.
- **Objectif** : prédire la sortie y associée à une nouvelle entrée x ,
sur la base de d_n
- **Prédicteur** : fonction mesurable $f \in \mathcal{F}$
avec \mathcal{F} l'ensemble des prédicteurs
 $f : \mathcal{X} \rightarrow \mathcal{Y}$
Nouvelle observation $x_{n+1} \implies f(x_{n+1})$ qui prédit y_{n+1}

- Fonction de coût (perte) ℓ : Loss function

$$\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+$$

$$\ell(y, y) = 0$$

$$\ell(y, \hat{y}) > 0 \text{ pour } y \neq \hat{y}$$

- Risque de prévision (erreur de généralisation) :

$$\mathcal{R}_P(f) = E_{(X, Y) \sim P} [\ell(Y, f(X))]$$

$\mathcal{Y} \subset \mathbb{R}$	\Rightarrow	Régression	$\ell(y, \hat{y}) = y - \hat{y} ^p$
			perte quadratique si $p=2$

\mathcal{Y} fini	\Rightarrow	Discrimination	$\ell(y, \hat{y}) = \mathbf{1}_{y \neq \hat{y}}$
--------------------	---------------	----------------	--

On cherche un prédicteur $f \in \mathcal{F}$ tel que $\mathcal{R}_P(f)$ soit minimal
i.e. ayant une **bonne capacité de généralisation** - capacité à faire de bonnes prédictions non seulement sur les données utilisées pour le construire, mais aussi sur de nouvelles données.

La fonction de prédiction idéale :

- Risque de Bayes :

$$R_p^* = \inf_{f \in \mathcal{F}} \mathcal{R}_P(f)$$

- Prédicteur de Bayes :

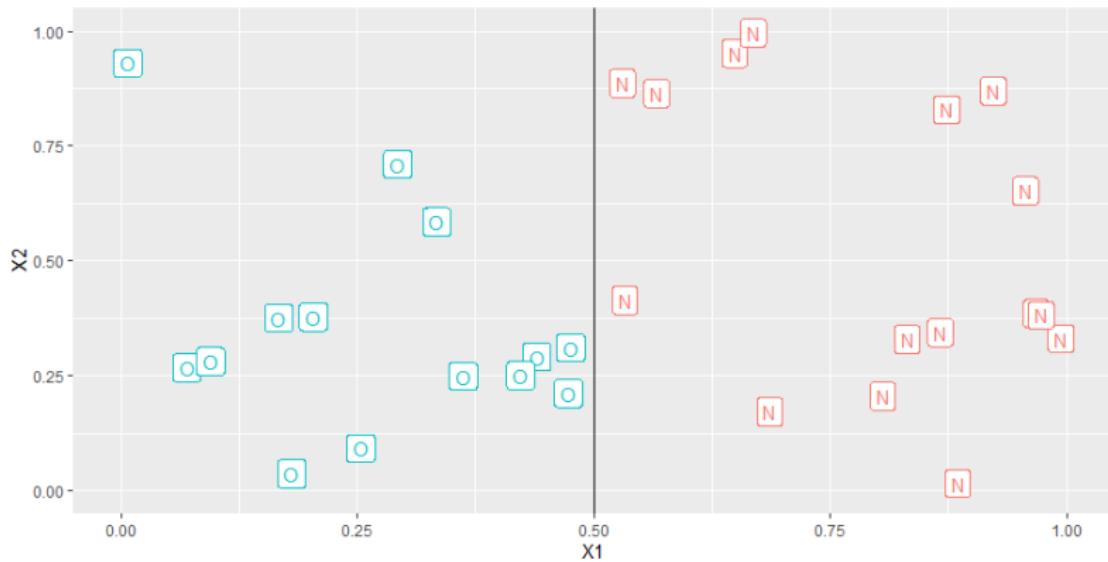
$$f^* = \operatorname{argmin}_{f \in \mathcal{F}} \mathcal{R}_P(f)$$

- Excès de risque :

$$ER(f, f^*) = R_p(f) - R_p^* \geq 0$$

Le risque minimal est obtenu pour la règle de Bayes, mais cette règle dépend de la loi P inconnue. \implies nécessité de trouver des règles de prédiction construites à partir de \mathcal{D}_n .

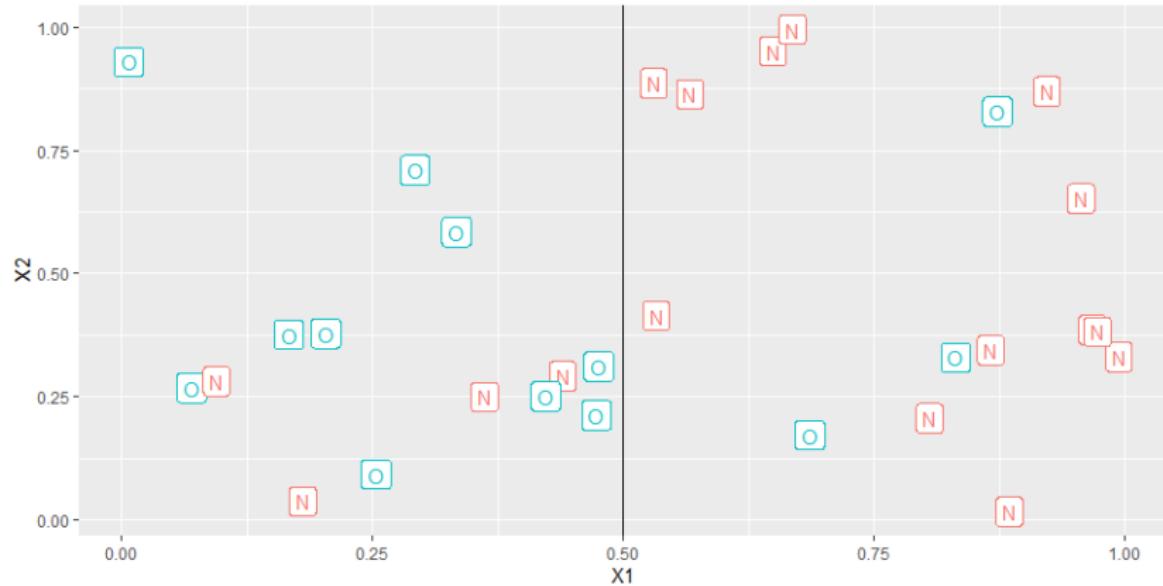
Choix de la règle de prédiction



$$X = (X_1, X_2) \sim \mathcal{U}[0, 1]^2 \quad \text{et} \quad \begin{cases} Y = O & \text{si } X_1 \leq 0.5 \\ Y = N & \text{sinon} \end{cases}$$

Règle de Bayes représentée ici par frontière verticale noire : si $x_1 \leq 0.5$ alors $f^*(x) = O$, sinon $f^*(x) = N$. Le risque de Bayes est $R_p^* = 0$

Choix de la règle de prédiction



$X = (X_1, X_2) \sim \mathcal{U}[0, 1]^2$ et $\begin{cases} Y = O \text{ si } (X_1 \leq 0.5 \text{ et } B \geq 0.25) \\ \quad \text{ou } (X_1 > 0.5 \text{ et } B < 0.25) \\ Y = N \quad \text{sinon} \end{cases}$

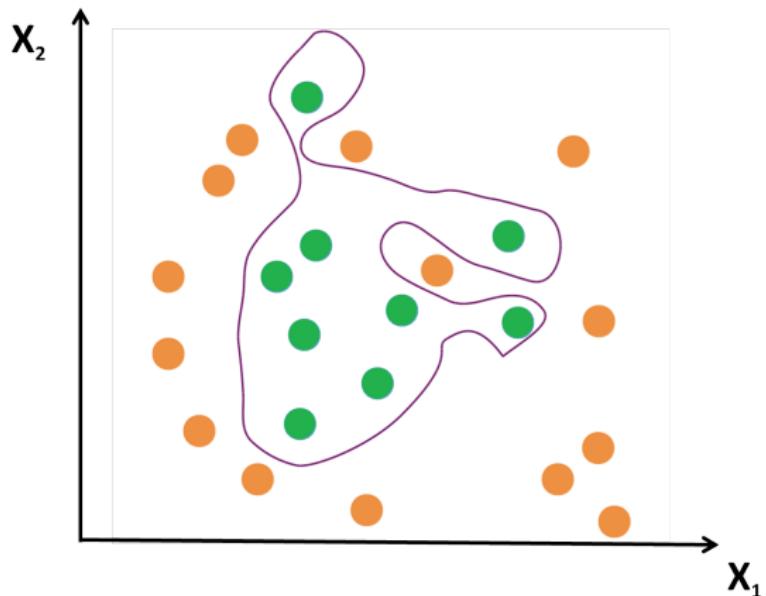
$R_p^* = ?$

Donner la règle de Bayes et calculer R_p^* dans le cas précédent

Choix de la règle de prédiction

Une règle de prédiction peut être représentée comme une **partition de l'espace des régresseurs** : définition d'une frontière.

Exemple : deux régresseurs numériques X_1 et X_2 , et une variable à expliquer à 2 modalités ("Orange", "Vert"). Ci-dessous une règle de décision représentée par la courbe violette.



- **Recherche d'une règle de prédiction :**

$$\hat{f} : \bigcup_{n \geq 1} (\mathcal{X} \times \mathcal{Y})^n \longrightarrow \mathcal{F}$$

Entrée : un échantillon \mathcal{D}_n

Sortie : un prédicteur $\hat{f}_{\mathcal{D}_n} : \mathcal{X} \longrightarrow \mathcal{Y}$

- **Risque conditionnel :**

$$R_p(\hat{f}_{\mathcal{D}_n}) = \mathbf{E}[\ell(\hat{f}_{\mathcal{D}_n}(X), Y) | \mathcal{D}_n = d_n]$$

- **Risque moyen :**

$$R_{moy}(\hat{f}_{\mathcal{D}_n}) = (\mathbf{E}[R_p(\hat{f}_{\mathcal{D}_n})]) = \mathbf{E}[\ell(\hat{f}_{\mathcal{D}_n}(X), Y)]$$

On choisit une règle de prédiction (et ses paramètres) en estimant son **risque moyen** par le risque empirique.

Le **risque empirique** d'une règle de prédiction $\hat{f} \in \mathcal{F}$ construite sur D_n est définie par :

$$\hat{R}_{emp}(\hat{f}_{D_n}) = \frac{1}{n} \sum_{i=1}^n \ell(Y_i, \hat{f}_{D_n}(X_i))$$

Attention à la sous-estimation du risque moyen par minimisation du risque empirique (apparent)

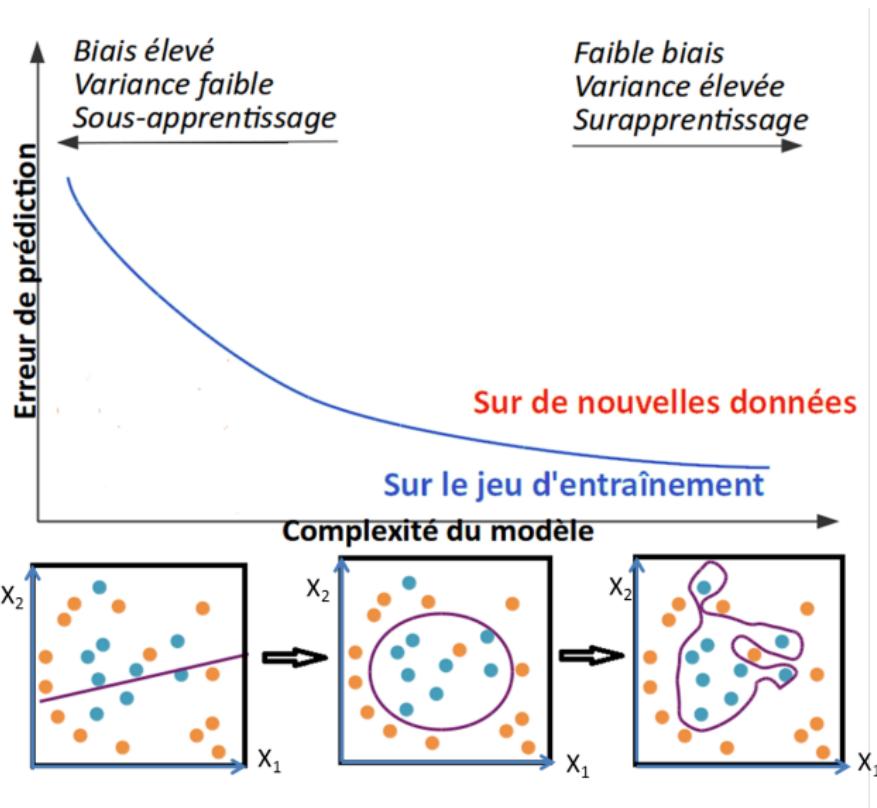
⇒ **sur-apprentissage** (overfitting en anglais).

C'est le cas d'un **modèle** qui colle trop aux données d'apprentissage : peut avoir de très bonnes performances sur l'échantillon d'apprentissage mais beaucoup se tromper sur de nouvelles données.

Compromis biais-variance :

éviter le **sur-apprentissage** et le **sous-apprentissage**

Choix de la règle de prédiction



Adapté de Azencott C.A. <https://openclassrooms.com>

Solutions pour trouver le meilleur compromis biais-variance :

- validation simple par apprentissage/validation (training/validation),
- validation croisée en laissant p observations de côté (leave p out),
- validation croisée à K blocs (K fold),
- validation croisée Monte Carlo (bootstrap)...etc.

Ces solutions reposent sur un **découpage de la base de données en 3 parties** :



L'utilisation de D_A et D_V diffère selon la méthode de validation retenue.

Dans tous les cas, **l'échantillon test** D_{test} sert à mesurer l'erreur commise par la meilleure règle de \mathcal{F} et la comparer à l'erreur commise par des règles de familles différentes sur des données qui n'ont pas du tout participé à la construction des règles.

- La part des observations consacrée à D_{test} dépend de la taille totale de la base de données. Avec D_{test} trop grand, on peut nuire à la qualité de l'apprentissage.
- Dans les algorithmes ci-après, on utilise les notations $D_{\mathcal{A}}$ et $D_{\mathcal{V}}$ qui permettent de construire différents estimateurs du risque moyen \Rightarrow possibilité d'étudier leur biais et leur variance (Arlot, 2017).
- En remplaçant $D_{\mathcal{A}}$ par $d_{\mathcal{A}}$ (sa réalisation) et $D_{\mathcal{V}}$ par $d_{\mathcal{V}}$ (idem), on a alors les estimations du risque moyen.

Dans le cas de la **validation simple**, on a :

- **Echantillon d'apprentissage D_A** : entraîner une seule fois chaque prédicteur possible $\hat{f}_{D_A} \in \mathcal{F}$ avec différents niveaux de complexité et différentes valeurs d'hyperparamètres.
- **Echantillon validation D_V** : mesurer l'erreur commise une seule fois par chacun des \hat{f}_{D_A} pour choisir le prédicteur dont le niveau de complexité et les valeurs des hyperparamètres minimisent le risque empirique.

On choisit le prédicteur $\hat{f}_{D_A} \in \mathcal{F}$ qui présente la plus petite erreur estimée par validation simple.

Exemple de niveaux de complexité : nombre de plus proches voisins pour les K plus proches voisins.

Exemple d'hyperparamètre : façon d'estimer une fonction de densité pour le classifieur bayésien naïf.

Algorithme : Validation simple ou "hold out"

Entrées

\mathcal{A} : l'ensemble d'apprentissage = sous-ensemble de $\{1, \dots, n\}$

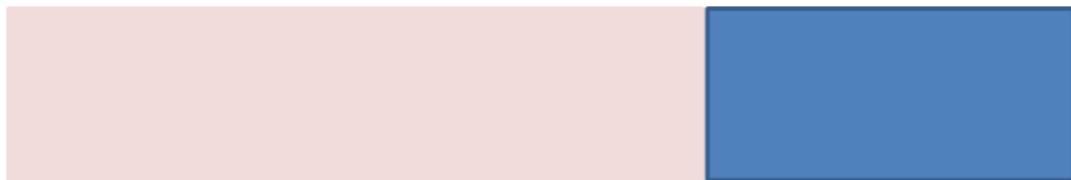
\mathcal{V} : l'ensemble de validation = sous-ensemble de $\{1, \dots, n\}$

Début

Construire la règle de prédiction $\hat{f}_{D_{\mathcal{A}}}$ sur $D_{\mathcal{A}} = \{(X_i, Y_i), i \in \mathcal{A}\}$

Retourner $\hat{R}_{Vsimp} = \frac{1}{\#\mathcal{V}} \sum_{i \in \mathcal{V}} \ell(Y_i, \hat{f}_{D_{\mathcal{A}}}(X_i))$

Fin



Echantillon d'apprentissage
Train

Echantillon
Validation

Avec les méthodes de validation croisée,

- **Echantillon $D_n = D_A \cup D_V$ entraîner une seule fois** chaque prédicteur possible $\hat{f}_{D_n} \in \mathcal{F}$ avec différents niveaux de complexité et différentes valeurs d'hyperparamètres.
- **On découpe K fois D_n en deux sous échantillons** et pour k variant de 1 à K :
 - **Echantillon d'apprentissage D_{A_k} :** entraîner le prédicteur $\hat{f}_{D_{A_k}} \in \mathcal{F}$ qui sert à approximer \hat{f}_{D_n} .
 - **Echantillon de validation D_{V_k} :** mesurer l'erreur commise par $\hat{f}_{D_{A_k}}$.
 - On fait la **moyenne des K erreurs** pour estimer l'erreur de \hat{f}_{D_n}

On choisit le prédicteur $\hat{f}_{D_A} \in \mathcal{F}$ qui présente la plus petite erreur estimée par validation croisée.

Algorithme : Validation croisée K blocs (K-folds)

Entrées

| K : entier diviseur de n

Début

| Construire une partition de $\{1, \dots, n\}$ en K parties $\mathcal{V}_1, \dots, \mathcal{V}_K$

Pour k variant de 1 à K

| Déterminer $\mathcal{A}_k = \{1 \dots n\} \setminus \mathcal{V}_k$

| Construire $f_{D_{\mathcal{A}_k}}$ sur $D_{\mathcal{A}_k} = \{(X_i, Y_i), i \in \mathcal{A}_k\}$

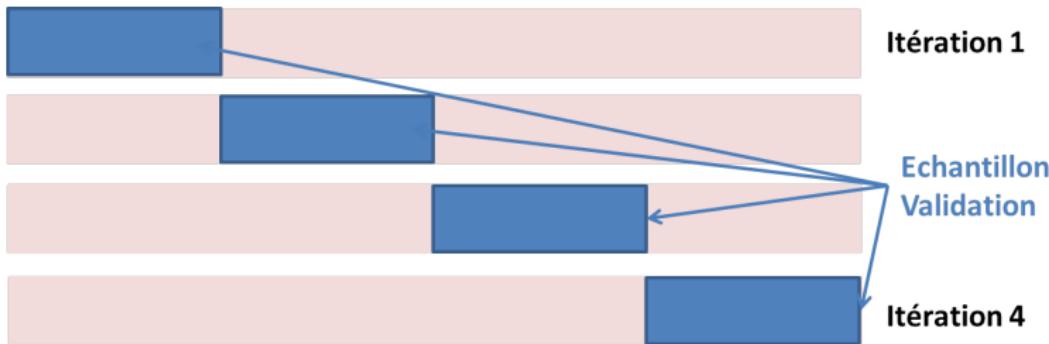
| Calculer $\hat{R}_k = \frac{1}{n/K} \sum_{i \in \mathcal{V}_k} \ell(Y_i, f_{D_{\mathcal{A}_k}}(X_i))$

FinPour

| Retourner $\hat{R}_{Kbl} = \frac{1}{K} \sum_{k=1}^K \hat{R}_k$

Fin

Illustration de la validation croisée en 4 blocs



- La valeur de K est souvent 10 par défaut dans les logiciels.
- Si K est petit (4 par exemple) la variance de cet estimateur diminue mais le biais (pessimiste) peut augmenter.
- Optimiser K correspond aussi à un (autre) compromis biais-variance mais nécessite généralement trop d'observations d'où le choix d'une valeur par défaut (P. Besse).

Algorithme : Validation croisée leave p out

Entrées

| p : entier inférieur à n

Début

| Construire $\mathcal{V}_1, \dots, \mathcal{V}_{\binom{n}{p}}$ les parties à p éléments de {1...n}

| Pour k variant de 1 à $\binom{n}{p}$

| | Déterminer $\mathcal{A}_k = \{1 \dots n\} \setminus \mathcal{V}_k$

| | Construire $\hat{f}_{D_{\mathcal{A}_k}}$ sur $D_{\mathcal{A}_k} = \{(X_i, Y_i), i \in \mathcal{A}_k\}$

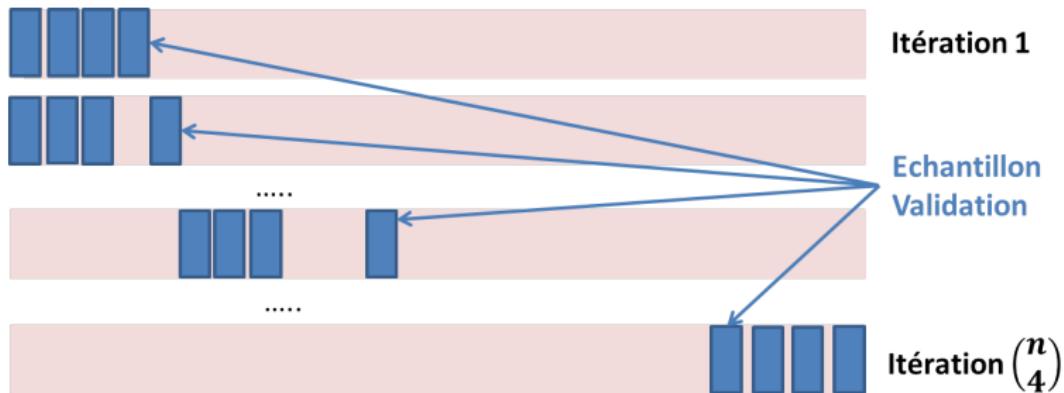
| | Calculer $\hat{R}_k = \frac{1}{p} \sum_{i \in \mathcal{V}_k} \ell(Y_i, \hat{f}_{D_{\mathcal{A}_k}}(X_i))$

| FinPour

| Retourner $\hat{R}_{LPO} = \binom{n}{p}^{-1} \sum_{k=1}^{\binom{n}{p}} \hat{R}_k$

Fin

Illustration de la validation leave 4-out



- Cas particulier L'estimateur **Leave-One-Out** du risque moyen de \hat{f}_{D_n} est défini par

$$\widehat{R_{LOO}}(\hat{f}) = \frac{1}{n} \sum_{i=1}^n \ell(Y_i, \hat{f}_{D_{(i)}}(X_i))$$

avec $D_{(i)}$ = échantillon \mathcal{D}_n privé de son i ème élément.

Les méthodes de Leave-p-Out et plus encore Leave-One-Out sont difficilement applicables si la base de données est grande (trop coûteux algorithmiquement).

Algorithme : Validation croisée Monte Carlo

Entrées

K : entier

pr : pr un réel tel que $0 < pr < 1$

Début

Tirer aléatoirement sans remise dans \mathcal{D}_n ,

K échantillons de taille $nxpr$: $\mathcal{V}_1, \dots, \mathcal{V}_K$

Pour k variant de 1 à K

Déterminer $\mathcal{A}_k = \{1 \dots n\} \setminus \mathcal{V}_k$

Construire $f_{D_{\mathcal{A}_k}}$ sur $D_{\mathcal{A}_k} = \{(X_i, Y_i), i \in \mathcal{A}_k\}$

Calculer $\hat{R}_k = \frac{1}{n.pr} \sum_{i \in \mathcal{V}_k} \ell(Y_i, f_{D_{\mathcal{A}_k}}(X_i))$

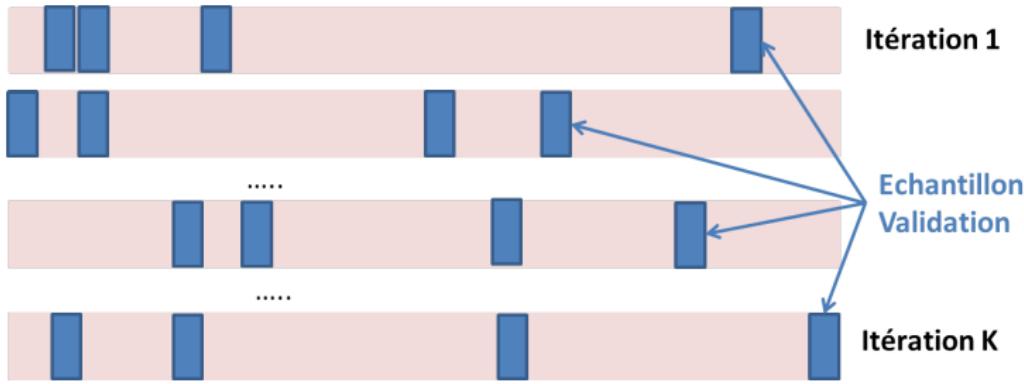
FinPour

Retourner $\hat{R}_{VMC} = \frac{1}{K} \sum_{k=1}^K \hat{R}_k$

Fin

On choisit K assez grand, par exemple 200.
 pr comme proportion.

Illustration de la validation croisée Monte Carlo



Chaque rectangle bleu vertical représente une observation

- La variance de cet estimateur est une fonction décroissante de K (S. Arlot, 2018)
- Plus la base de données est petite et moins les évaluations des erreurs sont "indépendantes" → moindre réduction de variance de l'estimateur du risque moyen obtenue à l'issue de la moyenne (P. Besse).

K-blocs ou Monte-Carlo ? (S. Arlot, 2018)

- La validation croisée K-blocs présente l'avantage de faire un usage « équilibré » des données : chaque observation est utilisée exactement $K - 1$ fois pour l'apprentissage et une fois pour la validation.
- Ce n'est pas forcément le cas avec la validation croisée Monte-Carlo.
- La validation croisée K-blocs présente l'inconvénient de toujours utiliser ensemble (soit pour l'apprentissage, soit pour la validation) les observations d'un même bloc.
- L'approche « Monte-Carlo », par son caractère aléatoire, permet d'éviter d'éventuels biais induits par ce lien entre observations.

Algorithme : Bootstrap "naïf"

Entrées

K : entier

n_B : entier tel que $n_B \leq n$

Début

Tirer aléatoirement avec remise dans \mathcal{D}_n ,

K échantillons de taille n_B : $\mathcal{V}_1, \dots, \mathcal{V}_K$

Pour k variant de 1 à K

Déterminer $\mathcal{A}_k = \{1 \dots n\} \setminus \mathcal{V}_k$

Construire $f_{D_{\mathcal{A}_k}}$ sur $D_{\mathcal{A}_k} = \{(X_i, Y_i), i \in \mathcal{A}_k\}$

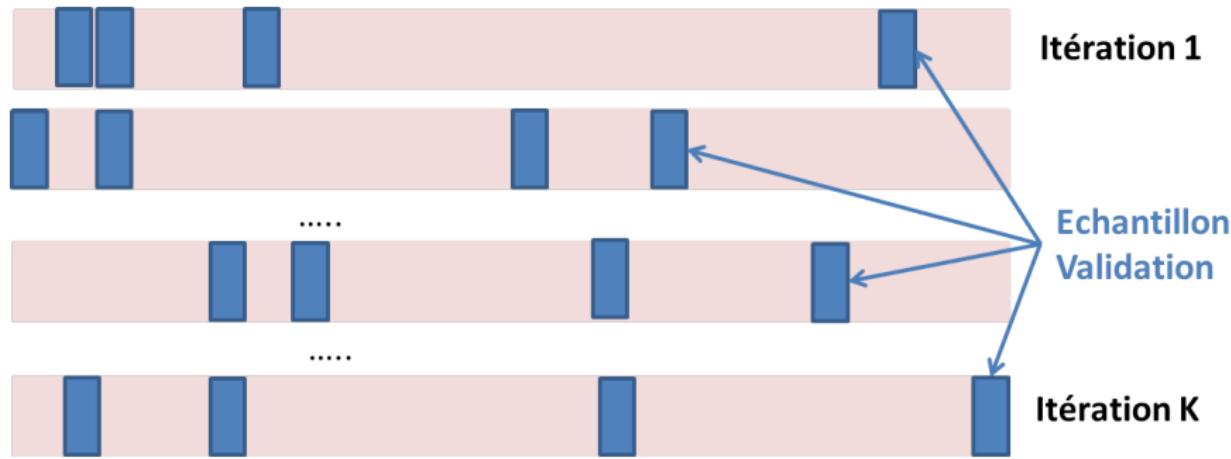
Calculer $\hat{R}_k = \frac{1}{n_B} \sum_{i \in \mathcal{V}_k} \ell(Y_i, f_{D_{\mathcal{A}_k}}(X_i))$

FinPour

Retourner $\hat{R}_{Boot} = \frac{1}{K} \sum_{k=1}^K \hat{R}_k$

Fin

Illustration de la validation croisée Bootstrap



Chaque rectangle bleu vertical représente une observation

Que proposez-vous comme modification du graphique pour illustrer correctement l'algorithme Bootstrap ?

L'estimateur \hat{R}_{Boot} présente généralement un biais (optimiste) car, d'une simulation à l'autre, on peut retrouver les mêmes observations (x_i, y_i) à la fois :

- dans l'apprentissage du modèle,
- dans l'estimateur de l'erreur du modèle.

Efron et Tibshirani (1997) ont proposé deux autres approches pour corriger ce biais :

- le Leave One Out Bootstrap,
- le Leave One Out Bootstrap .632 et variantes.

La probabilité qu'une observation i soit tirée dans un échantillon bootstrap \mathcal{D}_b de taille n est :

$$P(i \in \mathcal{D}_b) = 1 - (1 - \frac{1}{n})^n \xrightarrow{n \rightarrow \infty} 1 - e^{-1} \simeq 0.632$$

Algorithme : Leave One Out Bootstrap

Entrées

| B : entier assez grand

Début

Pour b variant de 1 à B

| Générer un échantillon bootstrap \mathcal{D}_b de taille n,
| tiré aléatoirement avec remise dans \mathcal{D}_n .

| Calculer $I_i^b = \mathbb{1}_{i \notin \mathcal{D}_b}$, pour $i = 1 \dots n$.

FinPour

Retourner $\hat{R}_{LOOB} = \frac{1}{n} \sum_{i=1}^n \frac{\sum_{b=1}^B I_i^b \ell(Y_i, f_{\mathcal{D}_b}(X_i))}{\sum_{b=1}^B I_i^b}$

Fin

\hat{R}_{LOOB} résoud le problème du biais rencontré par \hat{R}_{Boot} mais subsiste encore un biais d'où la variante $\hat{R}_{.632}$

$$\hat{R}_{.632} = 0.368 \times \hat{R}_{emp}(\hat{f}_{\mathcal{D}_n}) + 0.632 \times \hat{R}_{LOOB}$$

Ces estimateurs du risque moyen (validation croisée, bootstrap) sont asymptotiquement équivalents mais il est difficile de savoir lequel sera le plus précis à n fini.

Principes à retenir :

- utiliser un même estimateur du risque moyen pour comparer 2 modèles,
- si le temps de calcul est limité, prendre en compte la complexité algorithmique des procédures de validation croisée (souvent proportionnelle au nombre de découpages entre d_A et d_V),
- si on utilise la validation croisée K-blocs prendre K entre 5 et 10 (Arlot 2018),
- une certaine expérience nécessaire.

Anglais	Clustering	Classification
Français "Plutôt" Informatique	Classification non supervisée	Classification supervisée
Français "Plutôt" Statistique	Classification ex : CAH	Discrimination Classement ex : Analyse discriminante bayésienne

- **Les prédicteurs :**
Input variables = features
- **Les variables d'intérêt à prédire :**
Output variables = targets = response variables

Exemple d'apprentissage supervisée : Spotify

Exemple d'apprentissage supervisé pour illustrer les différentes méthodes présentées avec R

Prédire et analyser les déterminants des "likes" d'un client donné pour différents morceaux de musique disponibles sur Spotify en fonction leurs caractéristiques.

Source des données :

<https://www.kaggle.com/geomack/spotifyclassification>.

On utilisera la variable à expliquer qualitative binaire "*like*", définie comme facteur et avec comme codage :

- 1 Like,
- 0 Don't like.

A quoi ça peut servir ?



- ***acousticness*** : A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
- ***danceability*** : Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
- ***duration*** : The duration of the track in milliseconds
- ***energy*** : Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.
- ***instrumentalness*** : Predicts whether a track contains no vocals. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content.

- **key** : The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation . E.g. 0 = C, 1 = C_# / D_b, 2 = D, and so on. If no key was detected, the value is -1.
- **liveness** : Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.
- **loudness** : The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks
- **mode** : Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
- **speechiness** : Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value.

Exemple d'apprentissage supervisée : Spotify

- ***tempo*** : The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
- ***time_signature*** : An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).
- ***valence*** : A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).



Exemple d'apprentissage supervisée : Spotify

```
> library(skimr)
> skim(spotify)

> skim(spotify[,c(-15,-16)])
Skim summary statistics
n obs: 2017
n variables: 14

-- Variable type:factor --
variable missing complete n n_unique top_counts ordered
  key      0     2017 2017      12 1: 257, 0: 216, 7: 212, 9: 191 FALSE
  like     0     2017 2017       2      1: 1020, 0: 997, NA: 0 FALSE
  mode     0     2017 2017       2      1: 1235, 0: 782, NA: 0 FALSE
time_signature 0     2017 2017       4    4: 1891, 3: 93, 5: 32, 1: 1 FALSE

-- Variable type:numeric --
variable missing complete n mean sd p0 p25 p50 p75 p100 hist
acousticness   0     2017 2017  0.19 0.26 2.8e-06 0.0096 0.063 0.26 0.99 
danceability   0     2017 2017  0.62 0.16 0.12 0.51 0.63 0.74 0.98 
duration       0     2017 2017 246306.2 81981.81 16042 2e+05 229261 270333 1e+06 
energy         0     2017 2017  0.68 0.21 0.015 0.56 0.71 0.85 1 
instrumentalness 0     2017 2017  0.13 0.27 0 0 7.6e-05 0.054 0.98 
liveness        0     2017 2017  0.19 0.16 0.019 0.092 0.13 0.25 0.97 
loudness        0     2017 2017 -7.09 3.76 -33.1 -8.39 -6.25 -4.75 -0.31 
speechiness     0     2017 2017  0.093 0.09 0.023 0.037 0.055 0.11 0.82 
tempo           0     2017 2017 121.6 26.69 47.86 100.19 121.43 137.85 219.33 
valence         0     2017 2017  0.5 0.25 0.035 0.29 0.49 0.69 0.99 
```

>

Exemple d'apprentissage supervisée : Spotify

```
> library(corrplot)  
> matcorr=cor(spotify[,c(1 :5,7 :8,10 :11,13)])  
> corrplot.mixed(cor=matcorr,upper = "ellipse", tl.cex=.8, tl.pos  
= 'lt', number.cex = .8)
```

Matrice des
corrélations



Exemple d'apprentissage supervisée : Spotify

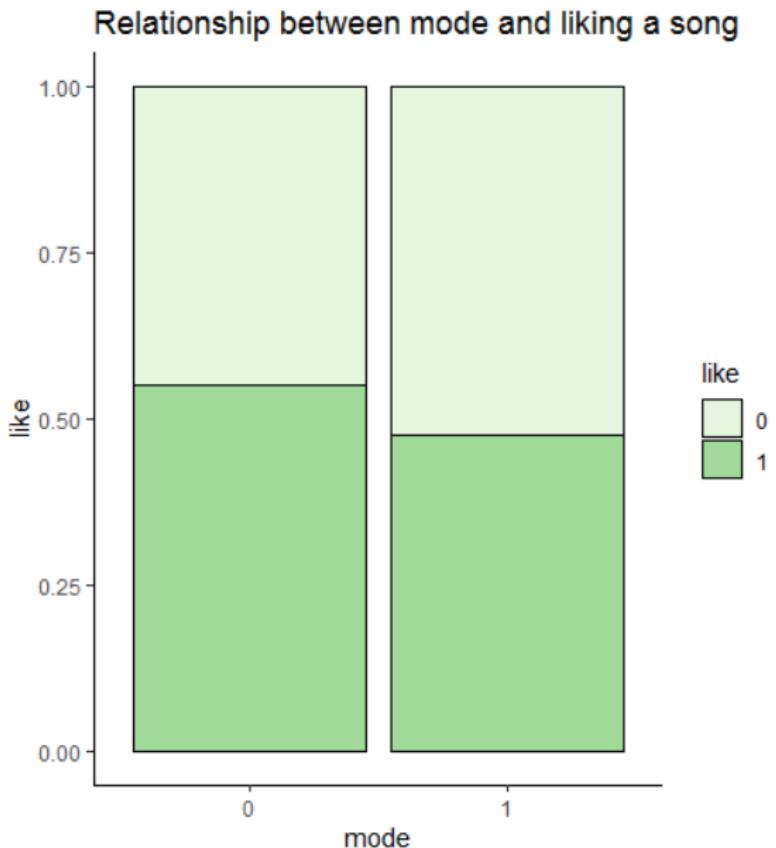
```
library(ggplot2)

# Tableau croisé entre mode et like
mode.like <- as.data.frame(prop.table(table(spotify$mode,spotify$like),1))

# Barplot
ggplot(mode.like, aes(x = Var1, y = Freq, fill = Var2)) +
  geom_bar(stat = "identity", position = "fill", color = "black") +
  theme_bw() + scale_fill_brewer(palette = "Dark1") +
  labs( x = "mode", y = "like", fill = "like",
       title = "Relationship between mode and liking a song")
```

Quelle autre analyse , non graphique, du lien entre **like** et **mode** proposez-vous ?

Exemple d'apprentissage supervisée : Spotify

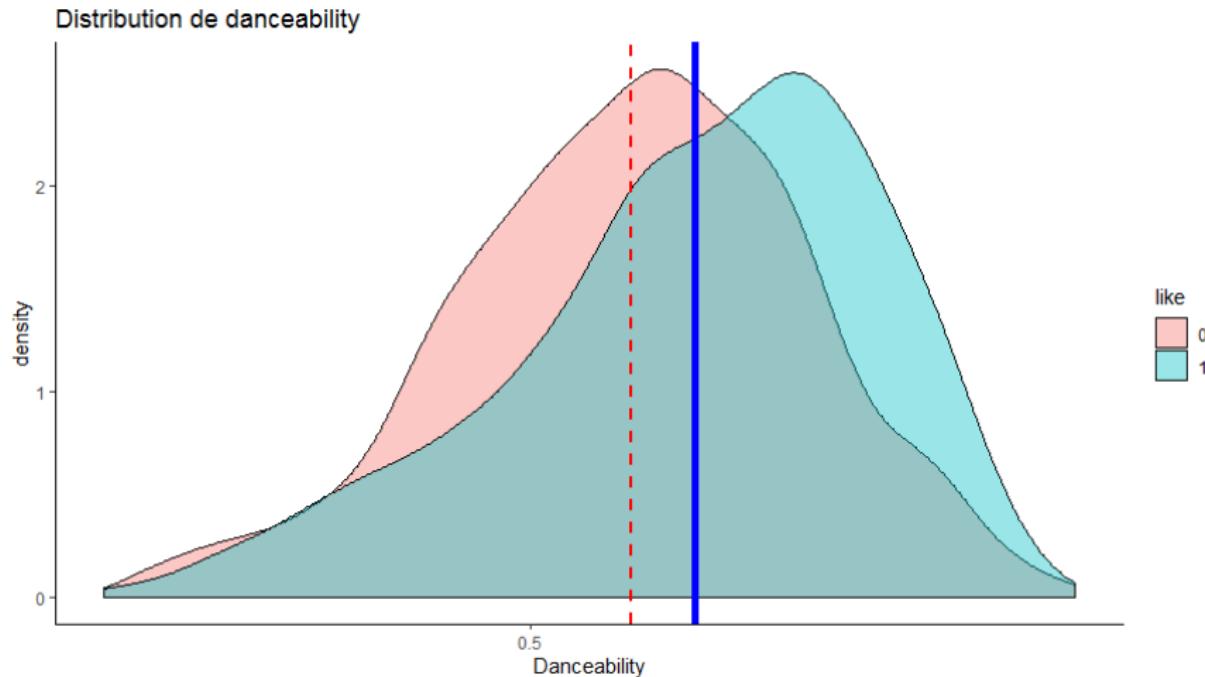


Exemple d'apprentissage supervisée : Spotify

```
library(ggplot2)
ggplot(spotify,aes(x=danceability,fill=like))+geom_density(alpha=0.4)+  
geom_vline(aes(xintercept=mean(danceability[like==0])),  
color="red",linetype="dashed",lwd=1)+  
geom_vline(aes(xintercept=mean(danceability[like==1])),  
color="blue",lwd=2)+  
scale_x_continuous(breaks = seq(0.5,1))+  
xlab(label = "Danceability")+
ggtitle("Distribution de danceability")+
theme_classic()
```

Quelle autre analyse, non graphique, du lien entre **like** et **danceability** proposez-vous ?

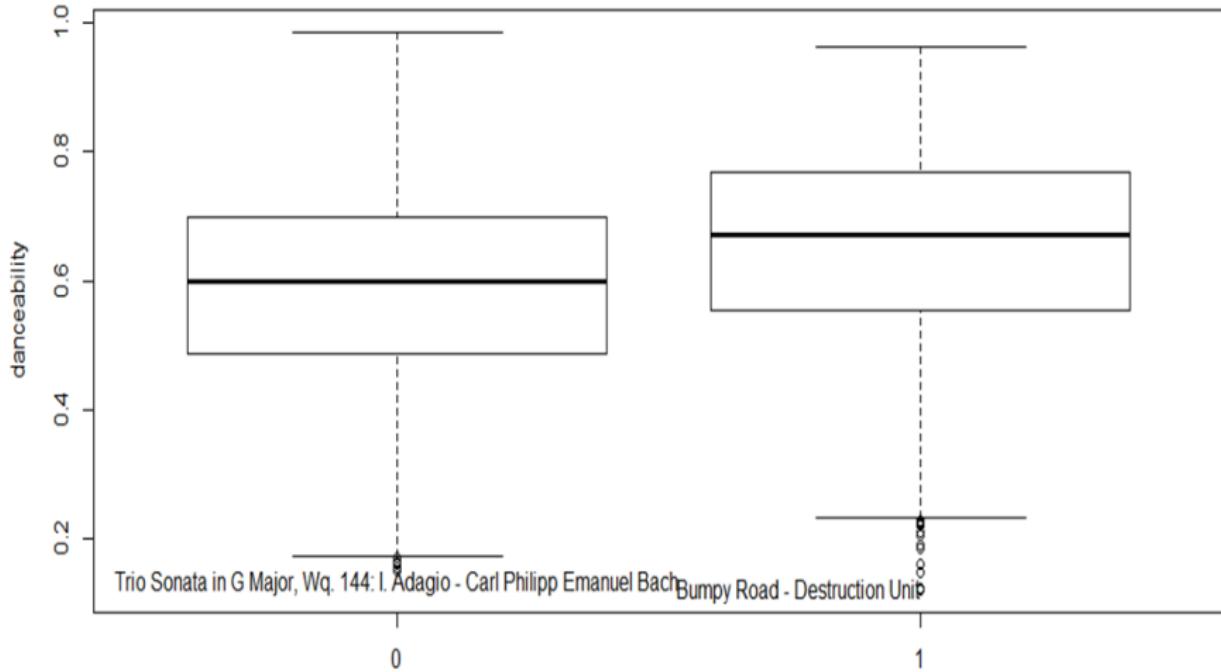
Exemple d'apprentissage supervisée : Spotify



Exemple d'apprentissage supervisée : Spotify

```
library(car)
```

```
Boxplot(danceability ~ like, spotify, id = list(n = 1, location = "r",  
labels = as.character(paste(spotify$song, " - ", spotify$band))))
```



K plus proches voisins

K Nearest Neighbors (KNN)

Rappel : pour tout nouveau x on veut prédire y

Définition

Un algorithme de moyennage local est défini pour

$D_n = (X_1, Y_1), \dots, (X_n, Y_n)$ par :

- $\hat{f}(X) = \sum_{i=1}^n W_i(X)Y_i$ si $\mathcal{Y} = \mathbb{R}$ donc en **régression**,
- $\hat{f}(X) = \text{signe}(\sum_{i=1}^n W_i(X)Y_i)$
si $\mathcal{Y} = \{-1; 1\}$ donc en **discrimination binaire**.

avec

$\{W_i, 1 \leq i \leq n\}$ une famille de poids positifs tels que
pour tout $n \geq 1$, $\sum_{i=1}^n W_i(X) = 1$

Définition

L'algorithme des K plus proches voisins est un algorithme par moyennage local, dont les poids vérifient :

$$W_i(X) =$$

$$\begin{cases} 1/K & \text{si } X_i \text{ fait partie des } K \text{ ppv de } X \text{ dans } \{X_1, \dots, X_n\} \\ 0 & \text{sinon.} \end{cases}$$

En cas d'égalité : tirage aléatoire possible.

- Quelle distance utiliser dans \mathcal{X} ?
- Comment choisir K ?

Soient x_1 et $x_2 \in \mathcal{X}$, il existe un grand nombre de distances possibles.

- Distance de **Hamming**

$$\sum_{j=1}^J \mathbb{1}_{x_{1j} \neq x_{2j}}$$

Seulement si $\mathcal{X} = \mathbb{R}^J$:

- Les distances de **Minkowski**, L_p

$$d(x_1, x_2) = \left(\sum_{j=1}^J |x_{1j} - x_{2j}|^p \right)^{1/p}$$

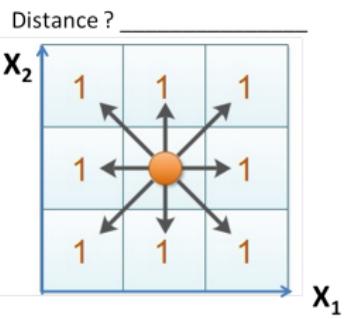
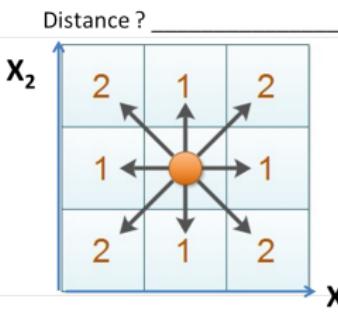
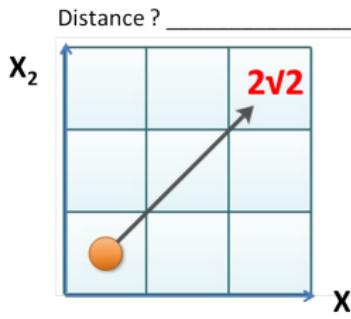
- Distance de **Manhattan**, L_1 : $p=1$
- Distance **euclidienne**, L_2 : $p=2$
- Distance de **Tchebychev**, distance $p \rightarrow \infty$

$$\lim_{p \rightarrow \infty} \left(\sum_{j=1}^J |x_{1j} - x_{2j}|^p \right)^{\frac{1}{p}} = \max_{j=1}^J |x_{1j} - x_{2j}|.$$

...etc.

K plus proches voisins - KNN (K Nearest Neighbors)

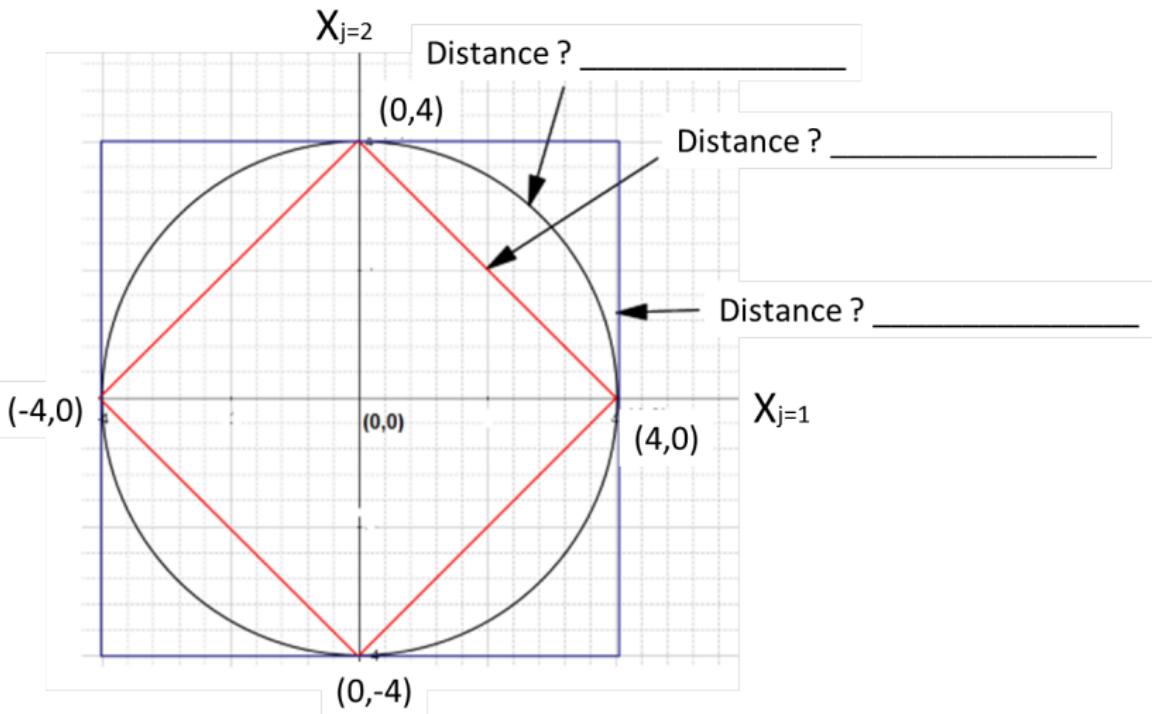
Distance de Hamming, Manhattan, euclidienne ou Tchebychev ?



Chaque flèche représente la distance qu'on veut mesurer entre le point orange placé au centre d'un carré et un autre centre de carré. Les carrés mesurent 1 de côté.

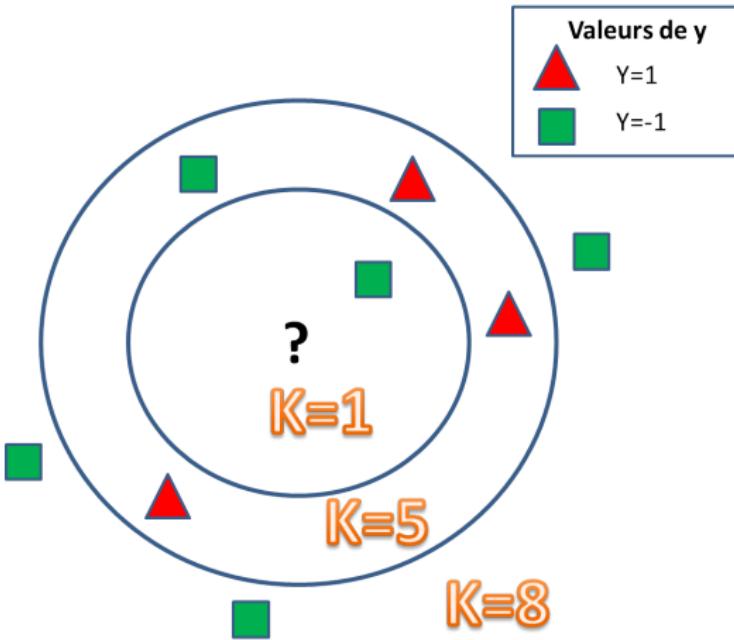
<https://gamedev.stackexchange.com>

K plus proches voisins - KNN (K Nearest Neighbors)



Courbes d'isodistance (distances = 4) par rapport à $(0,0)$

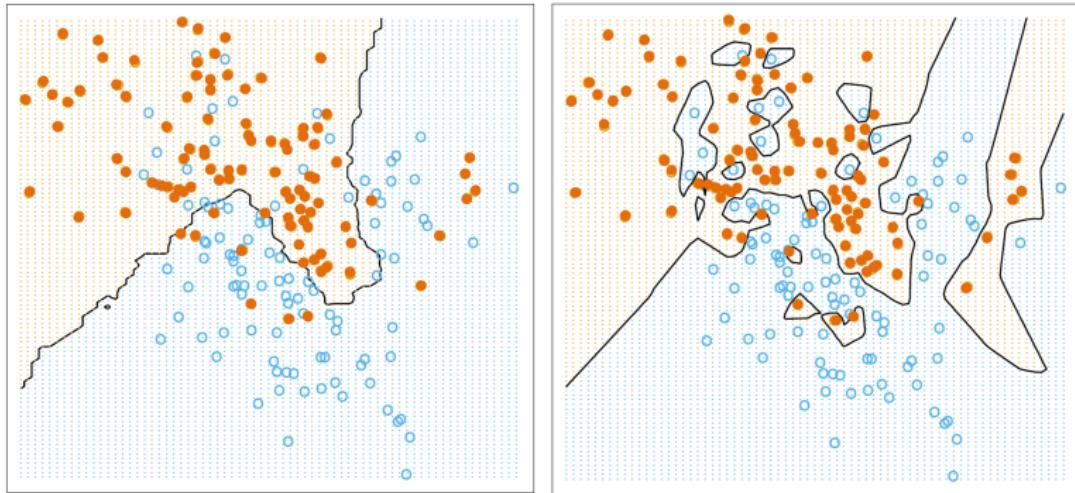
K plus proches voisins - KNN (K Nearest Neighbors)



Exemple KNN discrimination binaire =
quelle prédition pour l'observation au centre du graphique ?

Comment choisir K ? Utiliser des critères de performances.

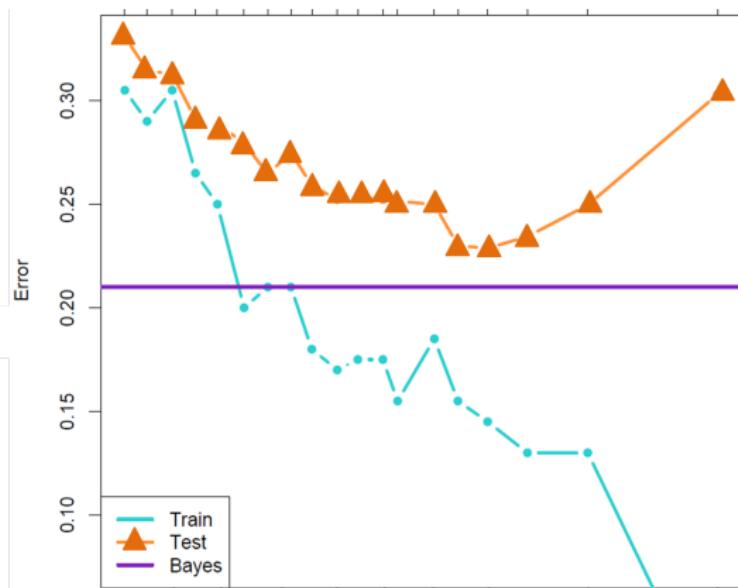
K plus proches voisins - KNN (K Nearest Neighbors)



Elements of Statistical Learning (2nd Ed.) Hastie, Tibshirani et Friedman, 2009

- Lequel de ces 2 graphiques correspond à une application de 1-plus proche voisin et lequel aux 15-plus proches voisins ?
- Voyez-vous un risque de sur-apprentissage sur un des deux graphiques ?

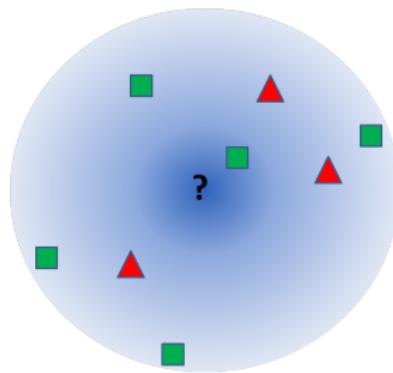
K plus proches voisins - KNN (K Nearest Neighbors)



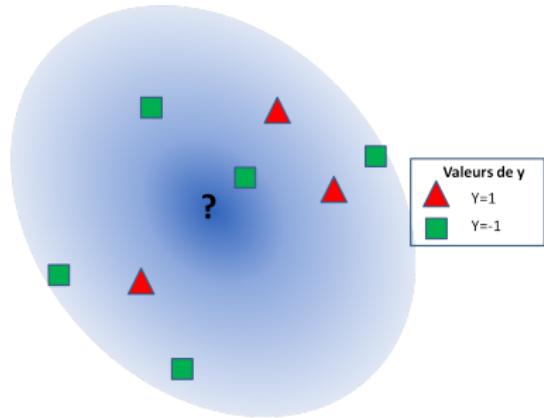
Elements of Statistical Learning (2nd Ed.) Hastie, Tibshirani et Friedman, 2009

- L'échelle supérieure correspond à différentes valeurs de K. Où placer K=1 et K=151 ?
- Comment est calculée "Error" ?

- ① Utiliser des méthodes à noyau avec des poids décroissant régulièrement avec la distance au point à prédire
- ② En grandes dimensions la distance calculée par noyau est modifiée pour mettre l'accent sur certaines variables



Solution 1



Solution 2

Algorithme par noyau d'approximation : méthode de Nadaraya-Watson

Les noyaux de convolution ont d'abord été utilisés par Parzen et Rosenblatt (1962) pour faire de l'estimation de densité (méthode des fenêtres de Parzen).

Définition

*Un algorithme par **noyau d'approximation** est un algorithme de moyennage local dont les poids prennent la forme :*

$$W_i(x) = \mathfrak{K}\left(\frac{X_i - x}{h}\right) / \sum_{i=1}^n \mathfrak{K}\left(\frac{X_i - x}{h}\right)$$

avec

$(X_i, Y_i) \in D_n$ l'échantillon d'apprentissage,

\mathfrak{K} le noyau - une fonction à valeurs dans \mathbb{R}_+ ,

$h \in \mathbb{R}_+^*$ la largeur de bande du noyau,

par convention $0/0 = 0$.

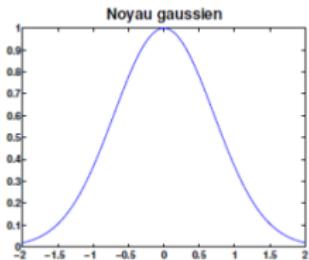
Algorithme par noyau d'approximation : méthode de Nadaraya-Watson

On note $(q)_+ = \max(0; q)$ la fonction *partie positive*.

Exemples de noyaux pour $t \in \mathbb{R}^d$:

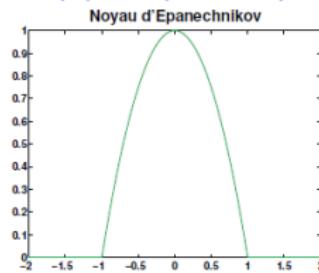
Gaussien ou radial

$$\mathfrak{K}(t) = e^{-\|t\|^2}$$



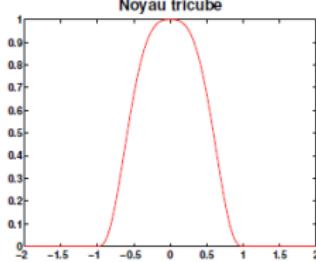
Epanechnikov

$$\mathfrak{K}(t) = (1 - t^2)_+$$



Tricube

$$\mathfrak{K}(t) = (1 - \|t\|^3)^3_+$$



S. Lacoste-Julien

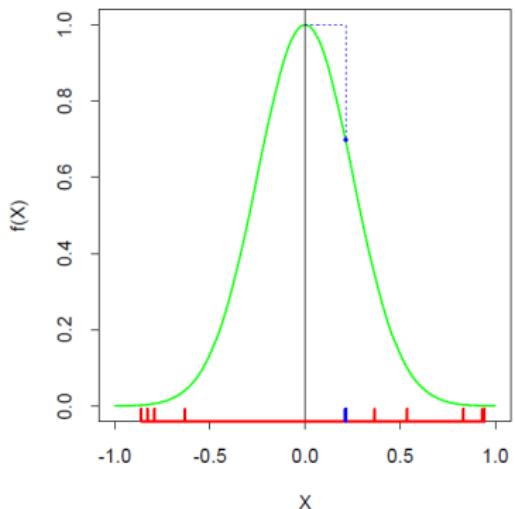
<https://www.di.ens.fr/~slacoste>

L'algorithme des K plus proches voisins peut être assez bon pour un faible nombre J de variables explicatives, i.e. $J \leq 4$ et un grand nombre d'observations dans d_n (Friedman et al., 2009).

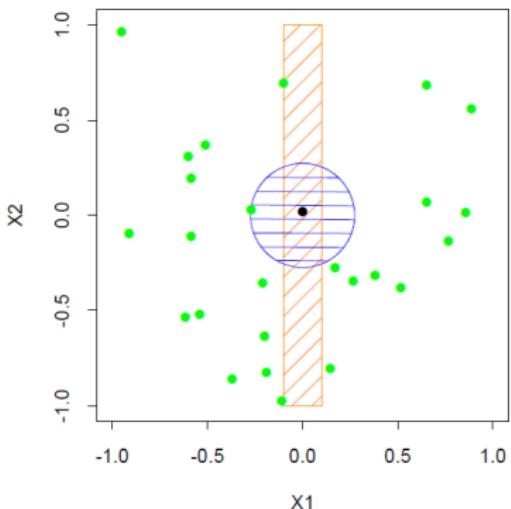
- L'algorithme des K plus proches voisins est moins performant si J est grand en raison du "fléau" de la dimension (curse of dimensionality, Bellman 1961). Plus le nombre de dimensions augmente et plus les plus proches voisins ont tendance à être éloignés.
- On a besoin d'une proportion raisonnable des observations de Y à moyenner pour réduire la variance de l'estimateur, par exemple 10%.
- Avec un voisinage de 10% des n observations, en grande dimension les plus proches voisins sélectionnés ne sont plus si proches et ça ne correspond plus à l'idée d'estimer $E(Y|X = x)$ par moyennage *local*.

Le fléau de la dimension

1-NN in One Dimension



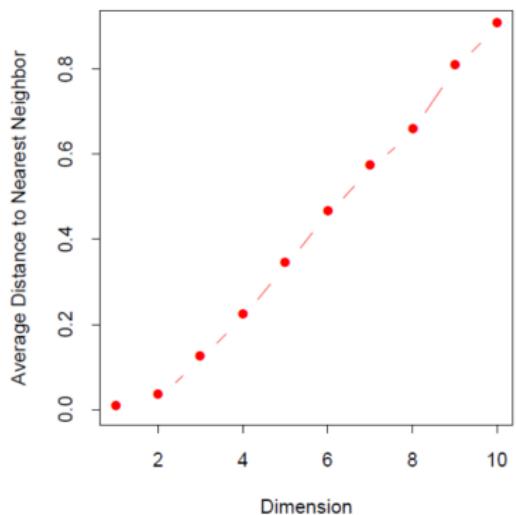
1-NN in One vs. Two Dimensions



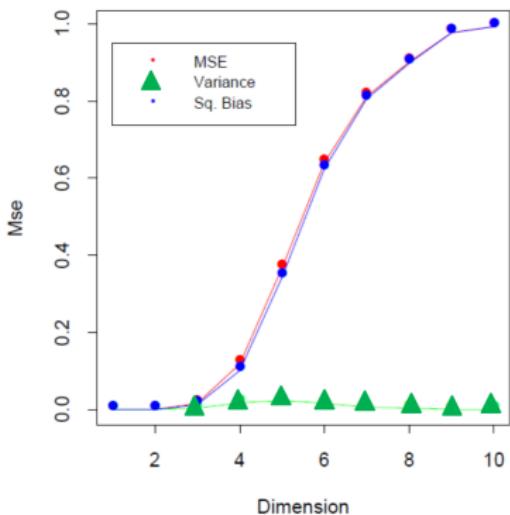
Elements of Statistical Learning (2nd Ed.) Hastie, Tibshirani et Friedman, 2009

Le fléau de la dimension

Distance to 1-NN vs. Dimension



MSE vs. Dimension



Elements of Statistical Learning (2nd Ed.) Hastie, Tibshirani et Friedman, 2009

Exemple d'apprentissage supervisée : Spotify

Plusieurs package R permettent de mettre en oeuvre les K plus proches voisins : caret, e1071, FNN...etc.

```
> library(e1071)

# regresseurs numeriques
> xnum<- (spotify[,c(2:6,8:9,11:12,14)])

> knn.cross <- tune.knn(x = scale(xnum),
y = spotify$like, k = 1:50,
tunecontrol=tune.control(sampling = "cross"),
cross=10)

> summary(knn.cross)

# Plot the error rate
> plot(knn.cross)
```

Exemple d'apprentissage supervisée : Spotify

Parameter tuning of 'knn.wrapper':

- sampling method: 10-fold cross validation

- best parameters:

k

19

- best performance: 0.278107

- Detailed performance results:

k error dispersion

1 1 0.3207601 0.01972663

2 2 0.3475248 0.03284410

3 3 0.3039087 0.02861677

...

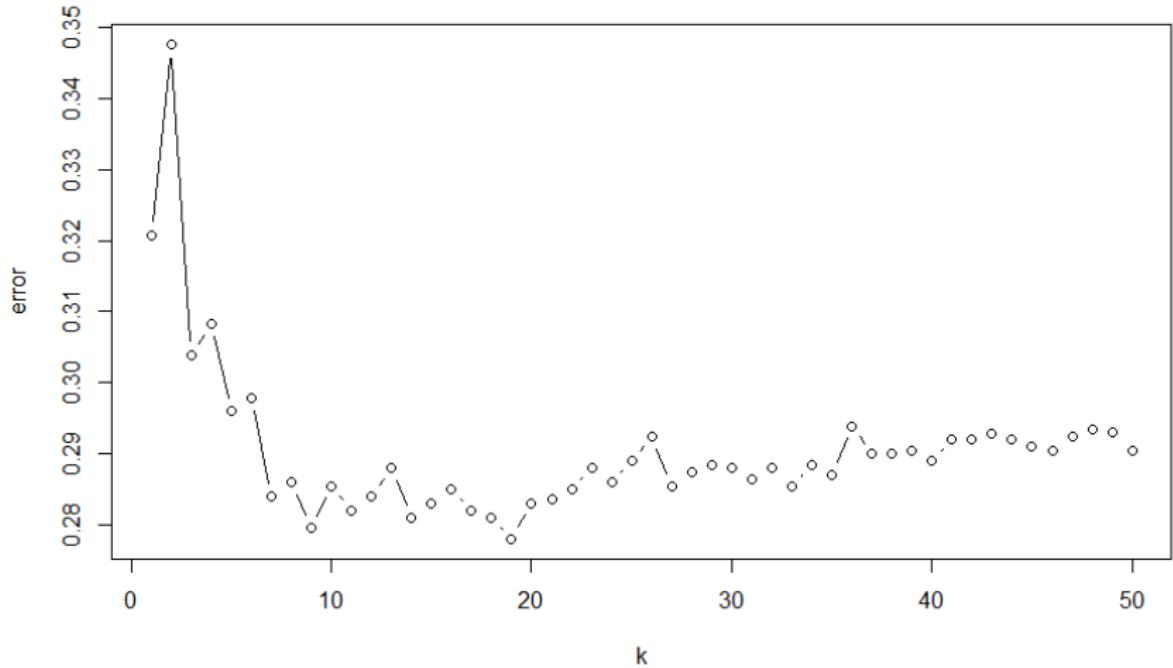
18 18 0.2810896 0.02808754

19 19 0.2781070 0.02872688

20 20 0.2830723 0.03360802

Cross Validation

Performance of 'knn.wrapper'



Exemple d'apprentissage supervisée : Spotify

```
> knn.boot <- tune.knn(x = scale(xnum),  
y = spotify$like,k = 1:50,  
tunecontrol=tune.control(sampling = "boot") )  
  
> # Summarize the resampling results  
  
> summary(knn.boot)  
  
> # Plot the error rate  
  
> plot(knn.boot )
```

Exemple d'apprentissage supervisée : Spotify

Parameter tuning of 'knn.wrapper':

- sampling method: bootstrapping

- best parameters:

k

19

- best performance: 0.2951492

- Detailed performance results:

k	error	dispersion
---	-------	------------

1	1	0.3333729 0.011006313
---	---	-----------------------

2	2	0.3389703 0.022214456
---	---	-----------------------

3	3	0.3334528 0.016401730
---	---	-----------------------

...

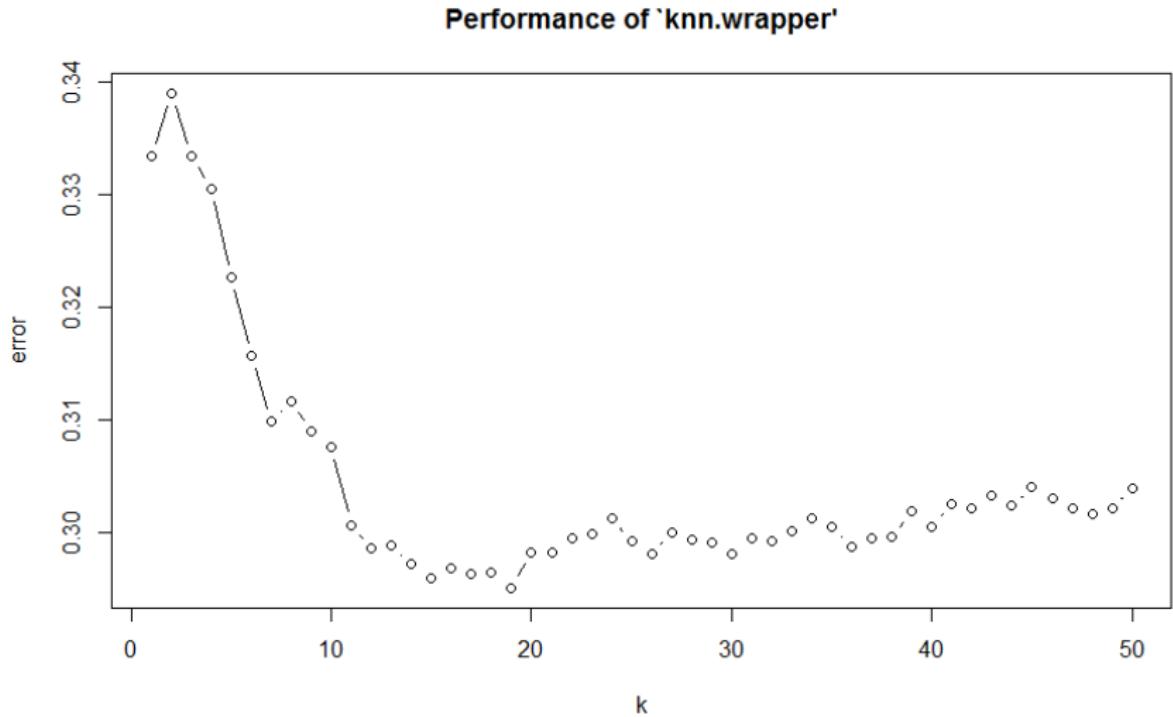
18	18	0.2965136 0.017095282
----	----	-----------------------

19	19	0.2951492 0.013519124
----	----	-----------------------

20	20	0.2983083 0.013378575
----	----	-----------------------

Exemple d'apprentissage supervisée : Spotify

Bootstrap Cross Validation



Classifieur bayésien naïf

Naive Bayes classifier

On considère un problème de classification supervisée.

- \mathcal{Y} fini.

Par exemple, $\mathcal{Y} = \{G_1, G_2, \dots, G_R\}$ - Y a R modalités.

- $X \in \mathcal{X}$ avec \mathcal{X} de dimension J .

On cherche à apprendre $P(Y = y|X = x)$.

Le classifieur naïf **bayésien** est basé sur
la formule de Bayes :

$$P(Y = y|X = x) = \frac{P(Y = y)P(X = x|Y = y)}{P(X = x)}$$

.

$P(Y = y|X = x)$ = probabilité a posteriori

$P(Y = y)$ = probabilité a priori (*priors*)

Le classifieur **naïf** bayésien repose sur l'hypothèse forte que les variables explicatives X_j sont indépendantes conditionnellement à la classe de Y d'où :

$$P(X = x | Y = y) = \prod_{j=1}^J P(X_j = x_j | Y = y)$$

On aura une approche **MAP (Maximum a Posteriori)**

$$\begin{aligned}\hat{f}(X) &= \arg \max_{y \in \mathcal{Y}} \hat{P}(Y = y | X = x) \\ &= \arg \max_{y \in \mathcal{Y}} \hat{P}(Y = y) \prod_{j=1}^J \hat{P}(X_j = x_j | Y = y)\end{aligned}$$

On peut estimer $P(Y = y)$ où faire des hypothèses sur sa valeur.

Malgré l'hypothèse simpliste d'indépendance entre les régresseur conditionnellement aux valeurs prises par Y , **le classifieur bayésien naïf s'avère souvent plus performant que des modèles plus sophistiqués.**

Si J est grand, cette technique est particulièrement appropriée. En grande dimension, l'estimation des fonctions de densité devient très compliquée.

Avec le classifieur bayésien naïf, chaque loi de probabilité des X_j conditionnellement aux valeurs de Y peut être estimée indépendamment en tant que loi de probabilité à une dimension.

Ce classifieur classe correctement du moment que la classe adéquate est plus probable que toutes les autres. Les probabilités de classe n'ont pas à être estimées de façon très précises, tant que les probabilités a posteriori ne sont pas trop touchées surtout autour de la zone de décision.

- Les bonnes performances du classifieur bayésien naïf malgré l'hypothèse d'indépendance entre les régresseurs étant surprenantes, plusieurs auteurs en ont recherché les raisons (Hand et Yu 2001, Zhang 2004).
- Zhang montre que **le classifieur bayésien naïf est optimal si les dépendances entre les régresseurs sont les mêmes d'une classe à l'autre** (une classe d'individus statistiques est définie par une modalité de Y) **ou si ces dépendances "s'annulent" les unes les autres.**
- Zhang donne également, dans le cas gaussien, les conditions suffisantes sous lesquelles le classifieur bayésien naïf est optimal, même si l'hypothèse d'indépendance conditionnelle n'est pas respectée.

Le classifieur bayésien naïf utilise le produit de probabilités conditionnelles $P(X_{ij} = x_{ij} | Y = y_i)$ avec $i = 1, \dots, n$ et $j = 1, \dots, J$.

Si une nouvelle observation i' présente une valeur sur un régresseur X_j qui n'apparaît jamais pour un ou plusieurs niveaux de Y dans la table d'apprentissage alors $P(X_{i'j} = x_{i'j} | Y = y_{i'}) = 0$.

Cette valeur nulle sur un seul régresseur va annuler tout le produit des probabilités conditionnelles d'où

$$\hat{P}(Y = y_{i'} | X = x_{i'}) = \frac{\hat{P}(Y = y_{i'}) \prod_{j=1}^J \hat{P}(X_j = x_{i'j} | Y = y_{i'})}{P(X = x_{i'})} = 0$$

Solution : le lissage de Laplace (Laplace smoother)

- Le lissage de Laplace ajoute un petit nombre à chaque effectifs dans les croisements régresseurs x variable à expliquer.
- On s'assure ainsi de ne pas avoir des probabilités à zéro pour chacune des valeurs prises par la variable à expliquer.
- Ce nombre à rajouter aux effectifs est un hyperparamètre qu'on optimise par exemple par validation croisée (tuning parameter). En général, 1 ou 2 est suffisant.

Classifieur bayésien naïf - Ajustement d'autres hyperparamètres

usekernel : permet d'utiliser un estimateur de densité à noyau versus un estimateur de densité gaussienne

adjust : permet de choisir la largeur de bande du noyau (un nombre plus grand donne plus de flexibilité à l'estimateur),

fL : permet d'utiliser le lissage de Laplace.

Classifieur bayésien naïf - Exemple avec R

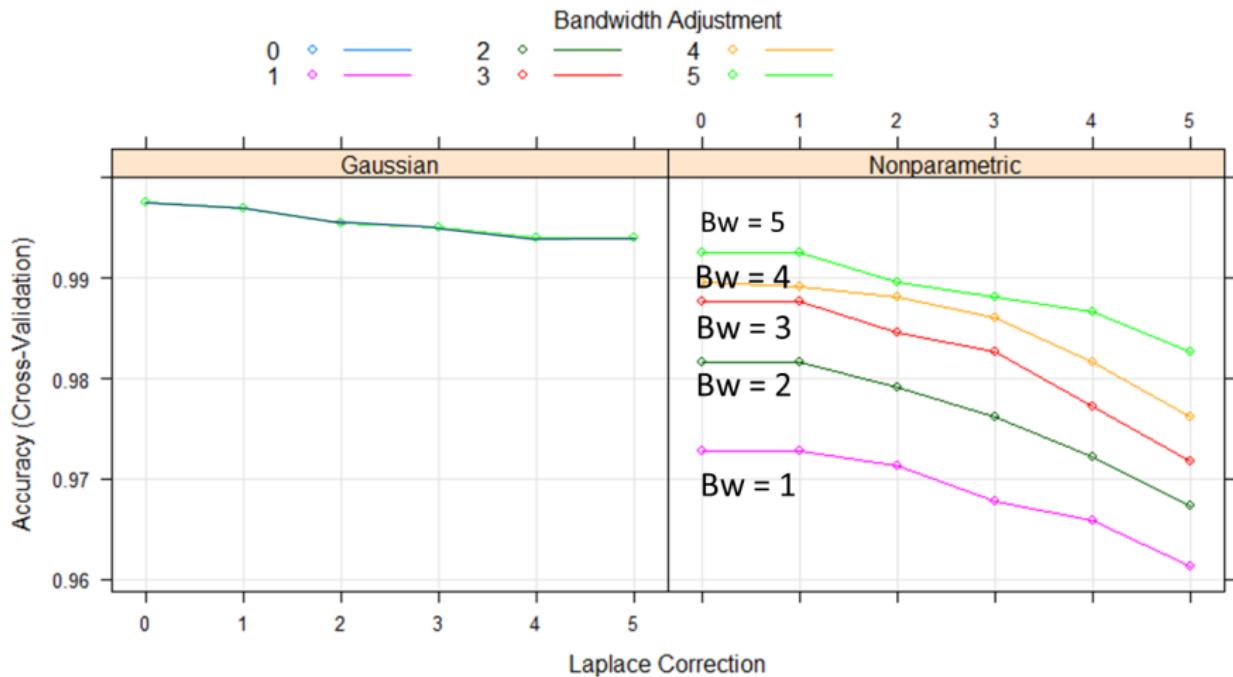
```
> library(caret)
> xnum<- (spotify[,c(2:6,8:9,11:12,14)])
> grid <- expand.grid(usekernel = c(TRUE, FALSE),
fL = 0:5,adjust = seq(0, 5, by = 1))

control <- trainControl(method = "cv", number = 10)

# train model

> naye.bayes1 <- train(x = xnum, y = spotify$like,
method = "nb", trControl = control,
tuneGrid = grid)
# plot search grid results
> library(ggplot2)
> plot(naye.bayes1)
```

Performance en fonction des valeurs des hyperparamètres



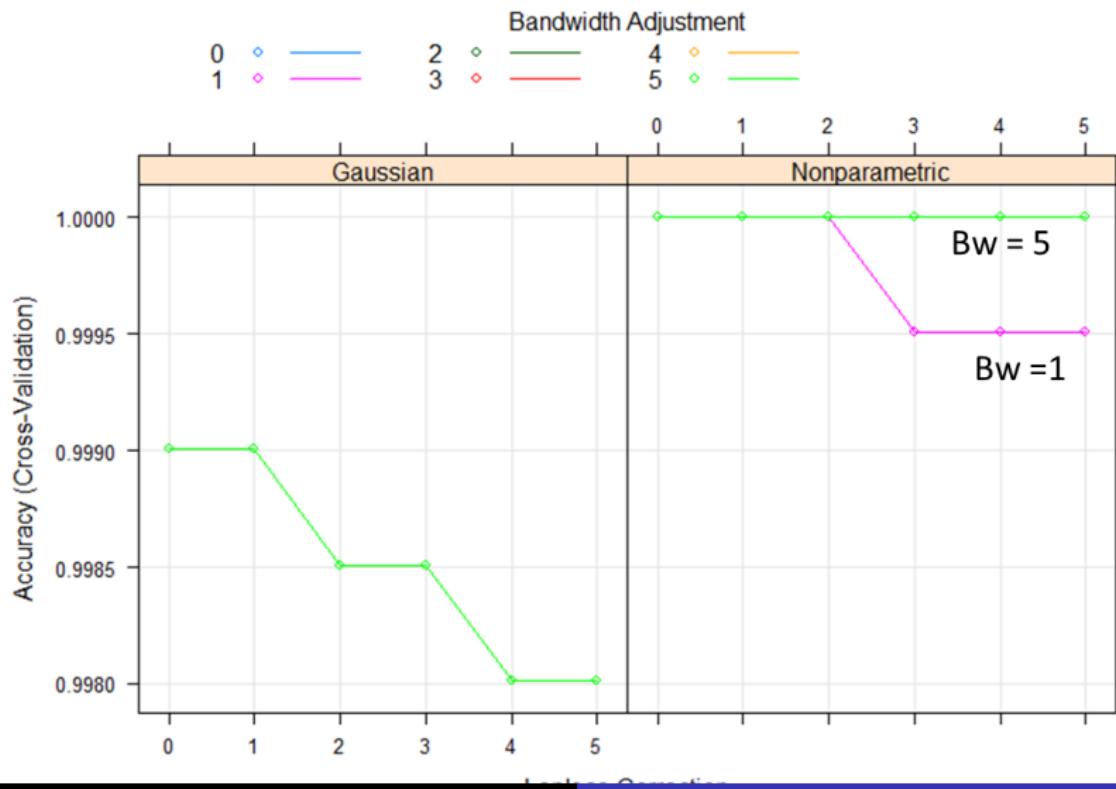
Il y a aussi une possibilité de pré traitement des variables :

- "normalize with Box Cox,
- standardize with center-scaling,
- reducing with PCA".

Classifieur bayésien naïf - Exemple avec R

```
> grid <- expand.grid(usekernel = c(TRUE, FALSE),  
fL = 0:5,adjust = seq(0, 5, by = 1))  
  
control <- trainControl(method = "cv", number = 10)  
  
# train model  
  
> naye.bayes2 <- train(x = xnum, y = spotify$like,  
method = "nb", trControl = control,  
preProc = c("BoxCox", "center", "scale", "pca")  
tuneGrid = grid)  
# plot search grid results  
> library(ggplot2)  
> plot(naye.bayes2)
```

Performance en fonction des valeurs des hyperparamètres



Dans les deux cas, modèles 1 et 2, des avertissements ont été fournis par R. A quoi correspondent-ils (voir ci-dessous) ?

```
> warnings()
```

Messages d'avertissement :

1 : model fit failed for Fold01 : usekernel= TRUE, fL=0, adjust=0 Error
in density.default(xx, ...) : 'bw' is not positive.

2 : In FUN(X[[i]], ...) :

Numerical 0 probability for all classes with observation 72

3 : In FUN(X[[i]], ...) :

Numerical 0 probability for all classes with observation 74

...etc.

Messages d'avis suite et fin ! Que constatez-vous ?

...

47 : In $\text{FUN}(\mathbf{X}[[i]], \dots)$:

Numerical 0 probability for all classes with observation 72

48 : In $\text{FUN}(\mathbf{X}[[i]], \dots)$:

Numerical 0 probability for all classes with observation 74

49 : In $\text{FUN}(\mathbf{X}[[i]], \dots)$:

Numerical 0 probability for all classes with observation 72

50 : In $\text{FUN}(\mathbf{X}[[i]], \dots)$:

Numerical 0 probability for all classes with observation 72

Pour info :

obs 72 = "Sabali" du groupe Amadou Mariam

obs 74 = "Pick Up The Pieces" du groupe Average White Band