



UNIVERSITÉ  
TOULOUSE 1  
CAPITOLE

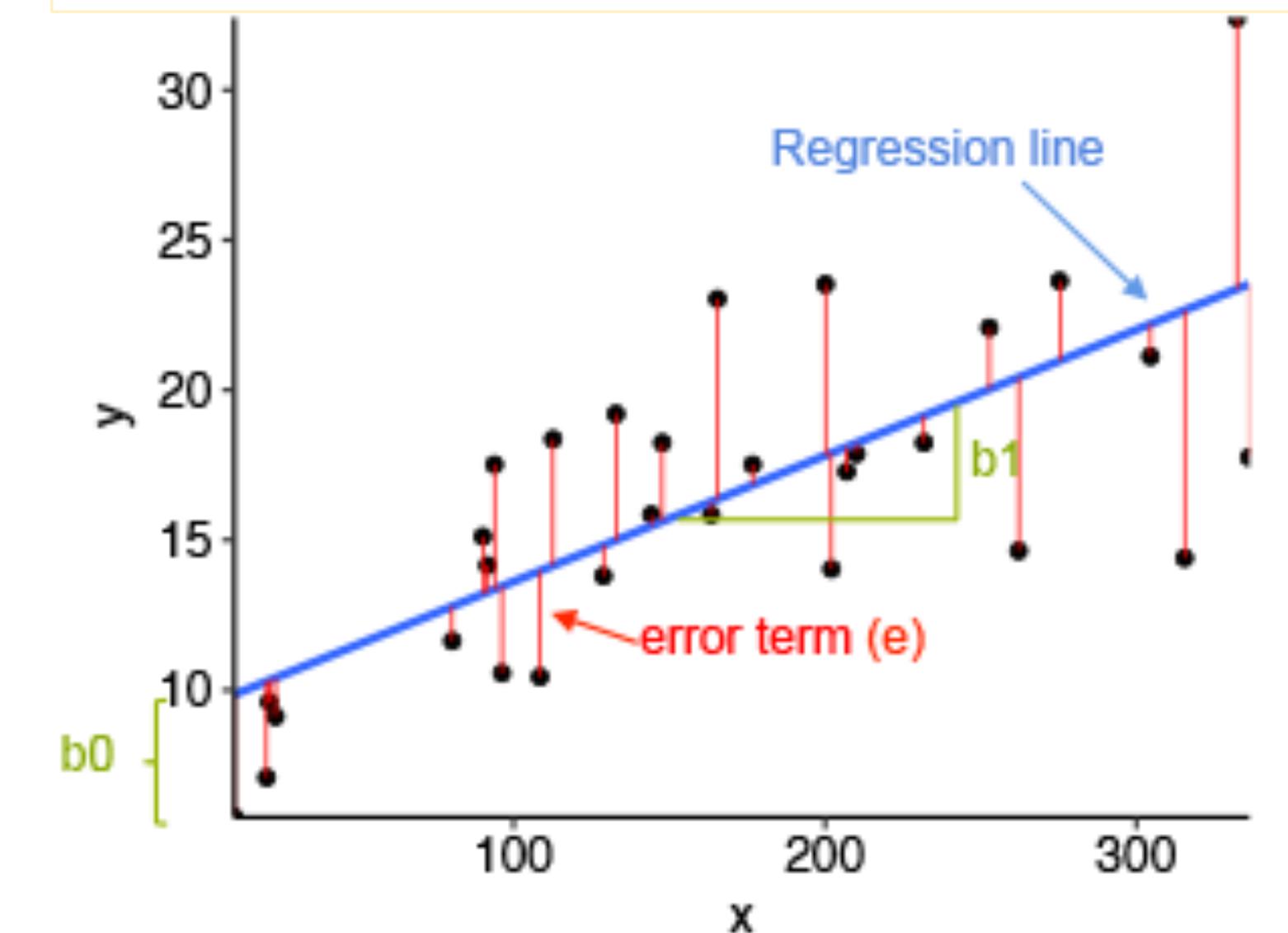
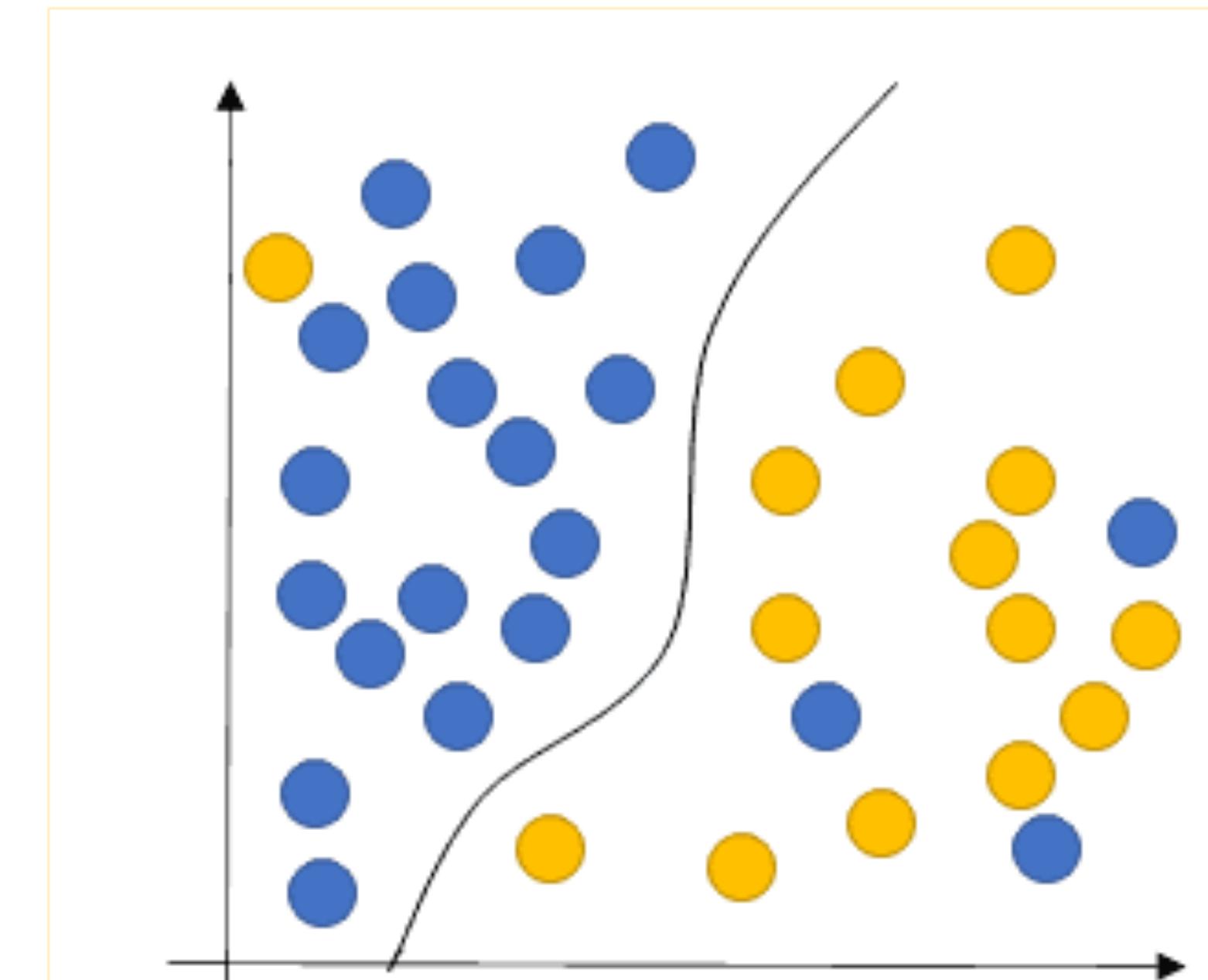
# Apprentissage supervisé

Benoit Gaudou

Sylvain Cussat-Blanc

# Apprentissage supervisé

- Classification
  - L'espace à prédire  $\mathcal{Y}$  est un ensemble discret non-ordonné
  - Classification **binaire** :  $card(\mathcal{Y}) = 2$
  - Classification **multiclasse** :  $card(\mathcal{Y}) > 2$
- Régression
  - On parle de régression quand  $\mathcal{Y}$  est un sous-ensemble de  $\mathbb{R}^d$
  - Souvent, on évalue le modèle en calculant l'erreur quadratique:  $(y - f(x))^2$



# Principe Fonction de perte (loss)

- Estime l'erreur de la prédiction  $f(x)$  par rapport à la réalité terrain  $y$

Par convention  $L(y, f(x)) = \begin{cases} 0 & \text{if } y = f(x) \\ > 0 & \text{if } y \neq f(x) \end{cases}$

- Risque réel : erreur de prédiction attendue

$$R_{true}(f) = \int_{\mathcal{X} \times \mathcal{Y}} L(y, f(x)) p(x, y) dx dy$$

- Objectif : Trouver la fonction qui minimise le risque réel

$$f^* = \arg \min_{f \in \mathcal{H}} R_{true}(f)$$

# Risque empirique

- Le risque réel est impossible à calculer car on ne connaît pas la distribution réelle des données
- Utilisation du **risque empirique** qui estime la loss sur un dataset d'entraînement

$$R_{emp} = \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i))$$

# Quelques fonctions de perte

- Erreur 0-1 (plutôt pour classification)

$$L(Y, f(X)) = \begin{cases} 0 & \text{if } y = f(x) \\ 1 & \text{otherwise} \end{cases}$$

- Erreur quadratique (plutôt pour régression)

$$L(Y, f(X)) = (Y - f(X))^2$$

- Erreur l1 (erreur absolue - régression)

$$L(Y, f(X)) = |Y - f(X)|$$

- Erreur de Hinge (plutôt pour classification)

$$L(Y, f(X)) = \max(0, 1 - yf(x))$$



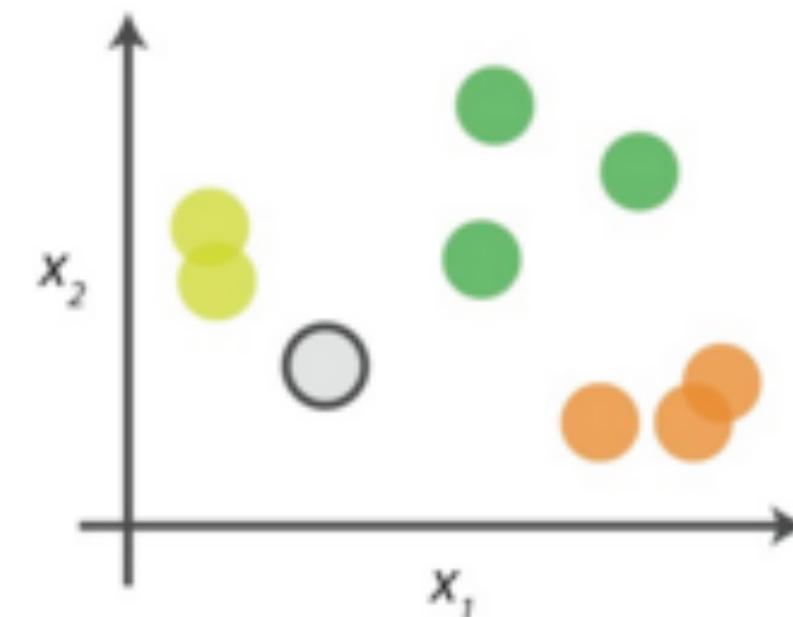
# **K Plus Proche Voisins**

## kNN Algorithm

# Principe Classification

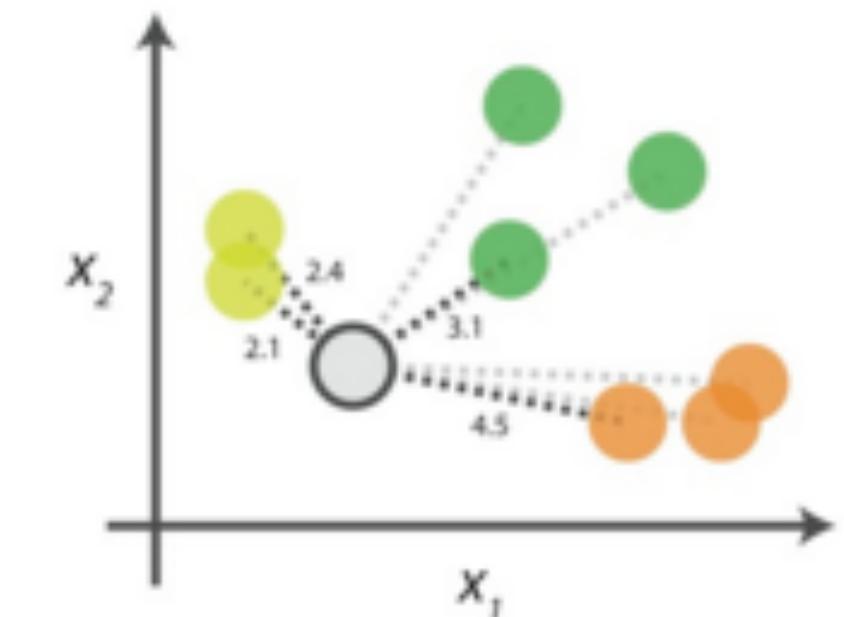
- Calculer les distances d'un point à classifier avec les points du dataset d'entraînement
- Sélectionner les k plus proches
- Compter le nombre de points dans chacune des classes
- Choisir la classe ayant le plus grand nombre de votes
- Classifier le point dans cette classe

### 0. Look at the data



Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.

### 1. Calculate distances



Start by calculating the distances between the grey point and all other points.

### 2. Find neighbours

Point	Distance	Rank
...	2.1	1st NN
...	2.4	2nd NN
...	3.1	3rd NN
...	4.5	4th NN

Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.

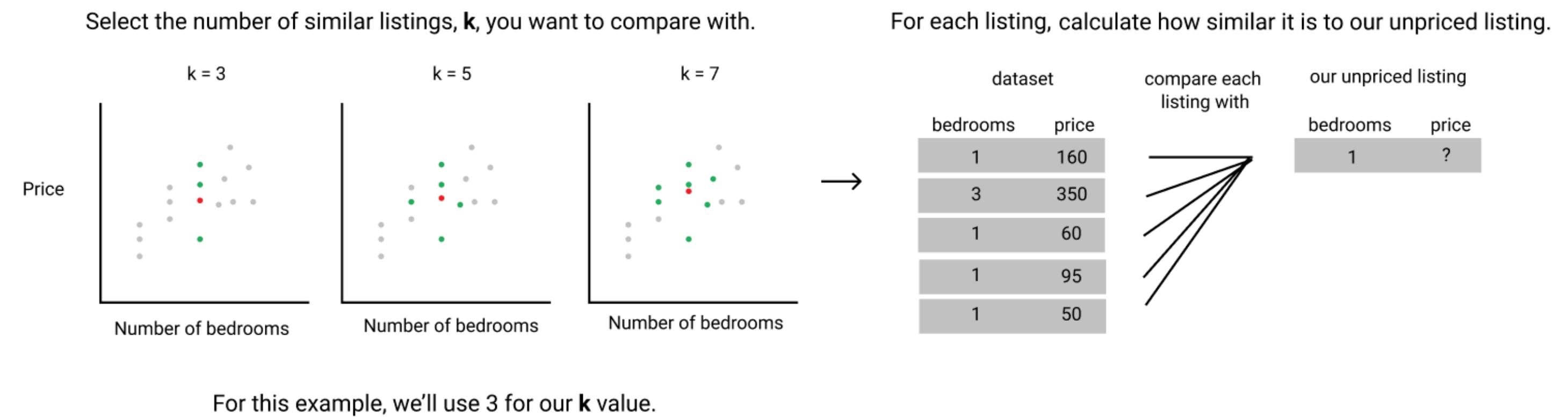
### 3. Vote on labels

Class	# of votes
lime green	2
green	1
orange	1

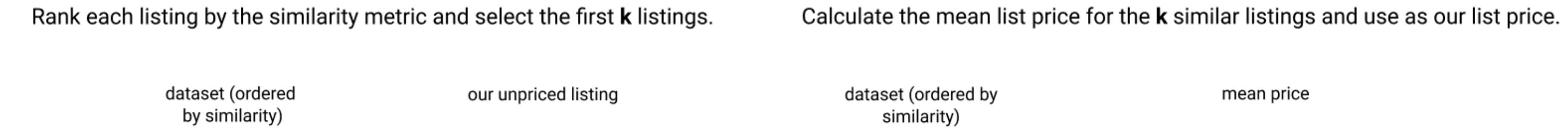
Class lime green wins the vote! Point grey is therefore predicted to be of class lime green.

Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

# Principe Régression



- Calculer les distances d'un point à classifier avec les points du dataset
- Sélectionner les **k** plus proches
- Prédiction = Moyenne des valeurs de ces **k** points



dataset (ordered by similarity)			our unpriced listing		dataset (ordered by similarity)		mean price
bedrooms	price	similarity	bedrooms	price	bedrooms	price	
1	160	0	1	?	1	160	105
1	60	0			1	60	
1	95	0			1	95	
1	50	0					
3	350	2					

[Lien vers le notebook](#)

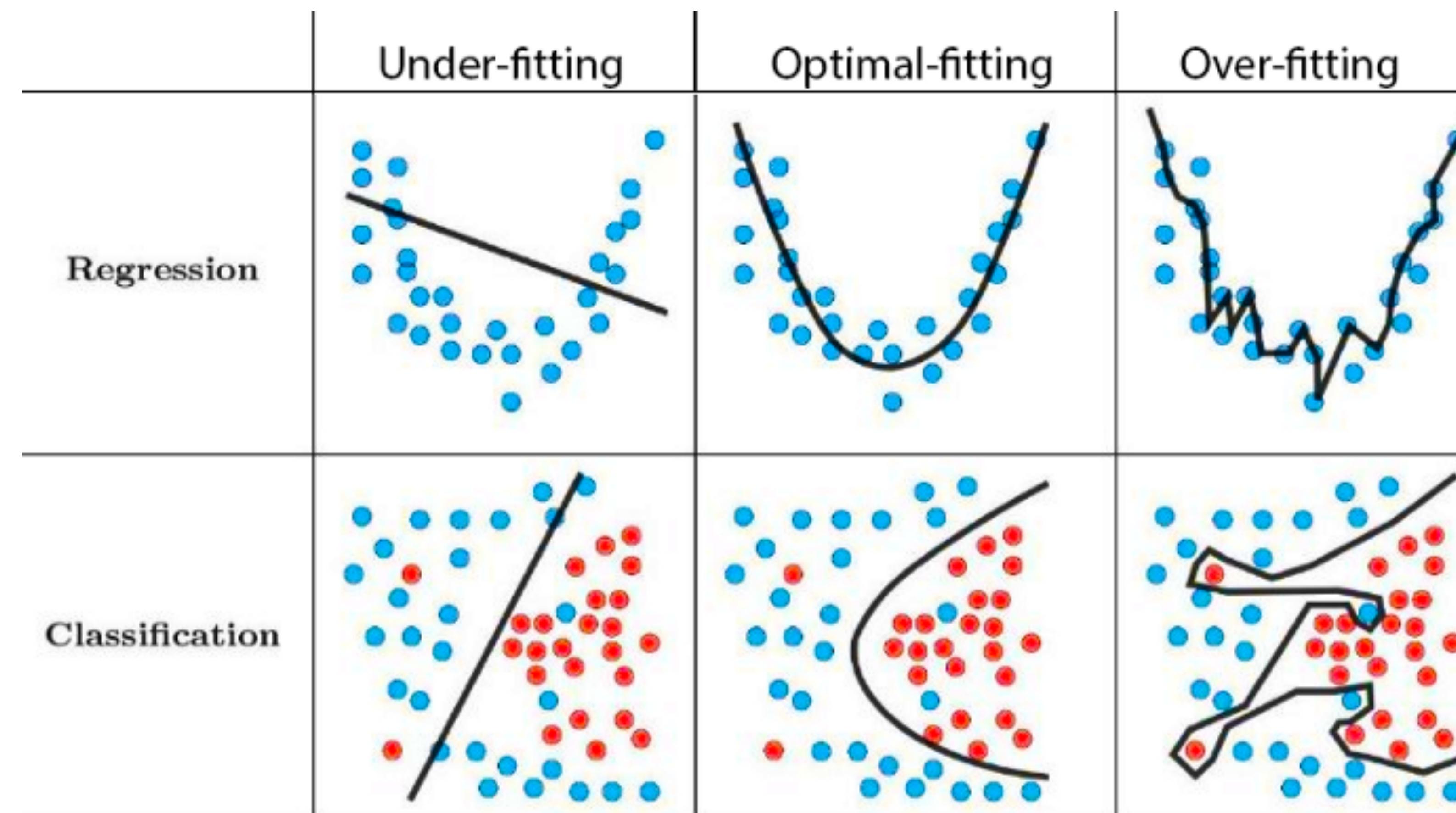
# **Selection de modèle**

**Train/test split**

**Validation croisée**

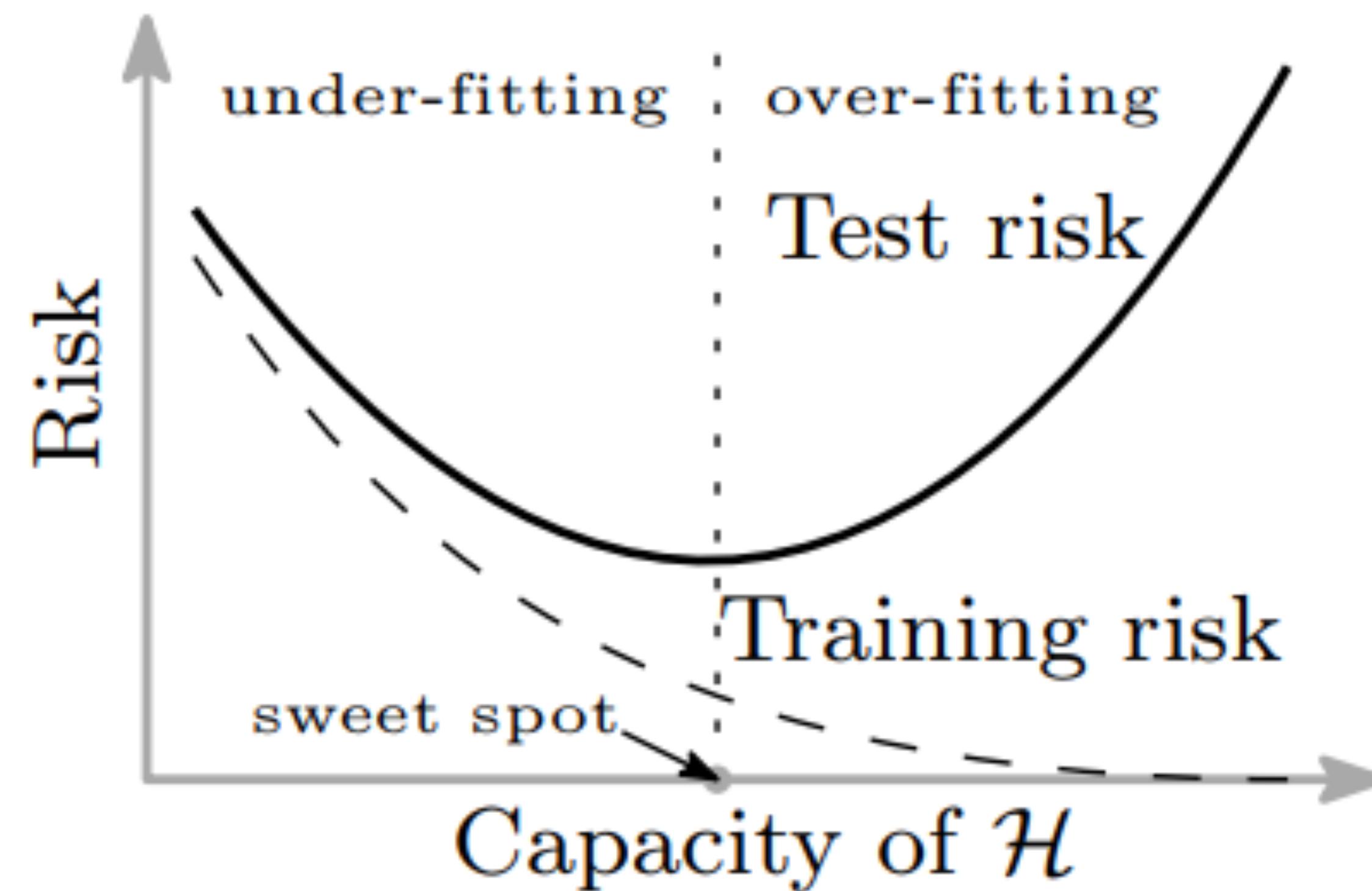
# Sur-apprentissage (overfitting)

## Illustration



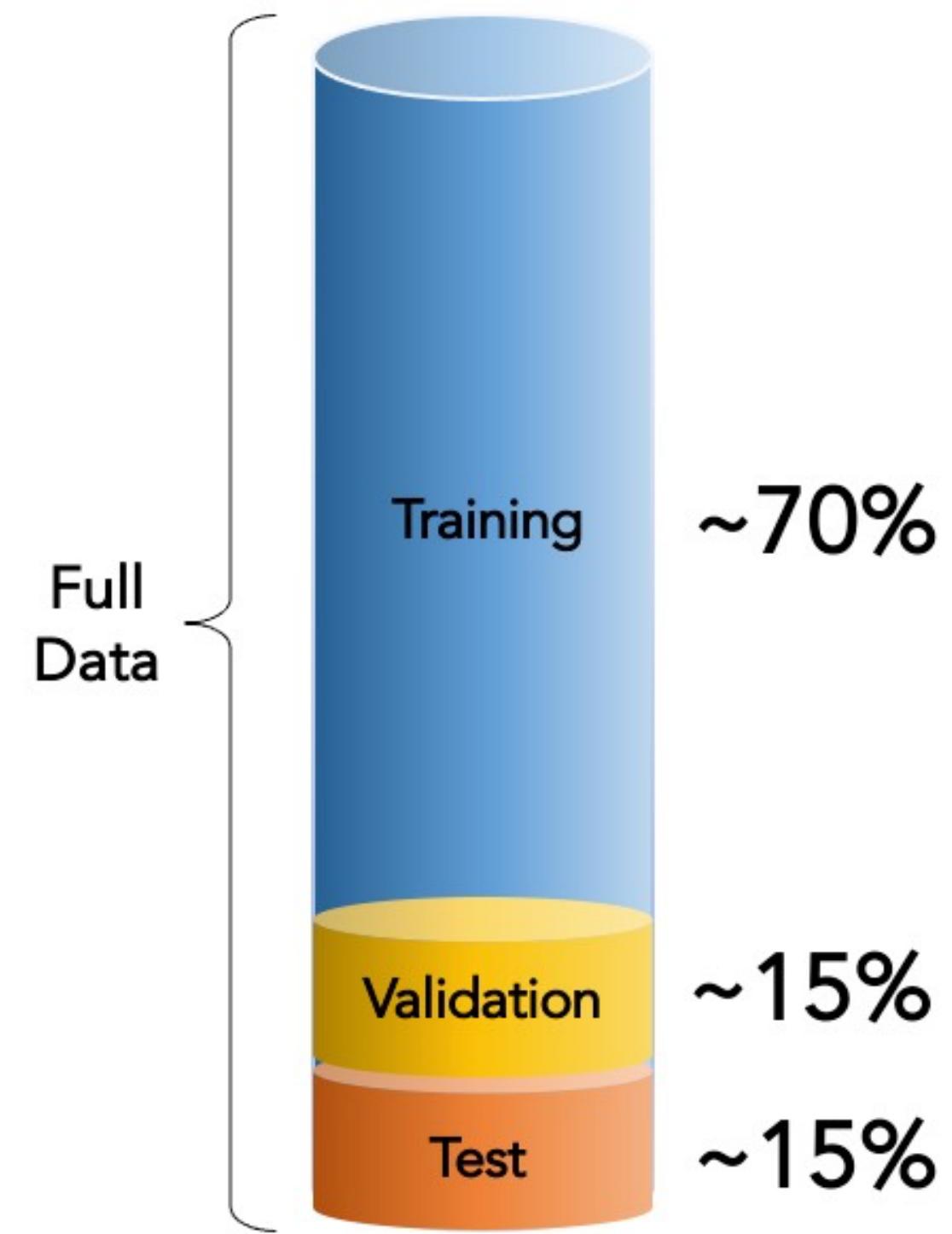
# Sur-apprentissage (overfitting)

- Le risque empirique n'est pas suffisant pour sélectionner le modèle : si  $\mathcal{H}$  est trop grand,  $R_{emp}(f) \rightarrow 0$  mais le risque réel  $R_{true}$  est grand



# Train/validation/test split

- Si le nombre d'exemples est suffisamment grand
  - Découper le dataset en 3 ensemble  $\mathcal{D}_N = \mathcal{D}_{train} \cup \mathcal{D}_{test} \cup \mathcal{D}_{val}$
  - Entrainer plusieurs modèles  $f_\alpha$  sur  $\mathcal{D}_{train}$
- Evaluer  $f_\alpha$  sur  $\mathcal{D}_{test}$  en calculant leur risque empirique  $R_{val}(f_\alpha) = \frac{1}{N_{test}} \sum_{i \in \mathcal{D}_{test}} L(y_i, f_\alpha(x_i))$
- Sélectionner le modèle qui minimise le risque empirique  $f_{best} = \arg \min_{f_\alpha} R_{test}(f_\alpha)$
- Tester la capacité de généralisation de  $f_{best}$  sur  $\mathcal{D}_{val}$



# Sélection de modèle

## Validation croisée

- Si le nombre d'exemples est plus faible, estimation de l'erreur par re-sampling
  - Découper  $\mathcal{D}_N$  en  $K$  ensembles de tailles égales
  - Pour chaque  $k = 1..K$ 
    - Entrainer un modèle sur les  $K - 1$  autres datasets
    - Evaluer le modèle sur le dataset  $\mathcal{D}_k$
  - Calculer la moyenne des  $K$  erreur obtenues

4-fold validation (k=4)



Lien vers le notebook

# **Evaluation des modèles**

# Comment savoir si nous avons obtenu un bon modèle?

- Vous avez fait apprendre beaucoup de modèles sur vos données.
- Vos modèles donnent de bons résultats sur les données d'entraînement!
  - Mais comment obtenir un bon pouvoir de généralisation ?
    - **Ex : Validation croisée**
    - Comment évaluer les performances de vos modèles ?
      - **Indicateurs de performances**
      - Comment choisir le "meilleur" ?
        - .....

# **Evaluer les modèles de classification**

**Précision (Accuracy)**

# Précision (Accuracy)

- Métrique de base
- Proportion de bonnes prédictions sur l'ensemble des prédictions

$$Acc(y_{true}, y_{pred}) = \frac{1}{n} \sum_{i=0}^n \begin{cases} 1 & \text{if } y_{pred}[i] = y_{true}[i] \\ 0 & \text{if } y_{pred}[i] \neq y_{true}[i] \end{cases}$$

- Permet de comparer différents modèles entre eux
- Donne un bon aperçu de la qualité du modèle, quand le dataset est équilibré (*balanced*), c'est-à-dire autant d'éléments dans chaque classe.
- Dans le cas contraire, il peut être intéressant d'aller plus loin et de regarder le **matrice de confusion**.

# **Evaluer les modèles de classification**

**Matrice de confusion**

# Définition d'une matrice de confusion

- Une **matrice de confusion** (*confusion matrix*, ou *tableau de contingence*) met en regard des données prédites et les données observées.
- Soit un modèle de classification binaire (entre "+" et "-"), testant un dataset de test de 1000 observations.

		Observations		Total
		+	-	
Prédictions	+	250	150	400
	-	50	550	600
Total		300	700	1000

# Définition d'une matrice de confusion

- Dans la **matrice** :
  - **$250 + 550 = 800$  classifications (prédictions) sont correctes** (prédiction + et observation + OU prédiction - et observation -)
  - **$50 + 150 = 200$  classifications (prédictions) sont fausses** (prédiction + et observation - OU prédiction - et observation +)

		Observations		Total
		+	-	
Prédictions	+	250	150	400
	-	50	550	600
Total		300	700	1000

# Premières métriques simples

- **Taux d'erreur (ERR, Error Rate) =  $(50+150) / 1000 = 20\%$**   
(taux de prédictions fausses)
- **Accuracy =  $1 - \text{ERR}$**   
(taux de prédictions correctes)

		Observations		Total
		+	-	
Prédictions	+	250	150	400
	-	50	550	600
Total		300	700	1000

# Matrice de confusion - Vocabulaire

		Observations		Total
Prédictions	+	+	-	
	-	Faux positifs (FP)	Vrais négatifs (VN)	Total des négatifs prédicts (FN + VN)
	Total	Total des vrais positifs observés (VP + FN)	Total des vrais négatifs observés (FP + VN)	Taille totale de l'échantillon (N)

# Matrice de confusion - Vocabulaire

- **VP (*Vrai Positif* ; *TP, True Positives*) : prédiction positive et correcte.**

Les cas où la prédiction est positive, et où la valeur réelle est effectivement positive.

Exemple : le médecin vous annonce que vous êtes malade, et vous êtes bel et bien malade.

- **VN (*Vrai Négatif*; *TN, True Negatives*) : prédiction négative et correcte.**

Les cas où la prédiction est négative, et où la valeur réelle est effectivement négative.

Exemple : le médecin vous annonce que vous n'êtes pas malade, et vous n'êtes effectivement pas malade.

- **FP (*Faux Positif; False Positive*) : prédiction positive incorrecte.**

Les cas où la prédiction est positive, mais où la valeur réelle est négative.

Exemple : le médecin vous annonce que vous êtes malade, mais vous n'êtes pas malade.

- **FN (*Faux Négatif; False Negative*) : prédiction négative incorrecte.**

Les cas où la prédiction est négative, mais où la valeur réelle est positive.

Exemple : le médecin vous annonce que vous n'êtes pas malade, mais vous êtes malade.

# Matrice de confusion - Métriques

- **Taux d'erreur (ERR)** =  $\frac{FP + FN}{N}$

- **Taux de vrais positifs** :  $\frac{VP}{VP + FN}$  aussi appelé *rappel (recall)* ou **sensibilité** ;

- **Précision** :  $\frac{VP}{VP + FP}$

- **Specificity** :  $\frac{VN}{FP + VN}$

		Observations		Total
		+	-	
Prédictions	+	Vrais positifs (VP)	Faux positifs (FP)	Total des positifs prédits (VP + FP)
	-	Faux négatifs (FN)	Vrais négatifs (VN)	Total des négatifs prédits (FN + VN)
	Total	Total des vrais positifs observés (VP + FN)	Total des vrais négatifs observés (FP + VN)	Taille totale de l'échantillon (N)

# Matrice de confusion - Métriques

- **Taux d'erreur (ERR)** =  $\frac{FP + FN}{N}$

- **Taux de vrais positifs** :  $\frac{VP}{VP + FN}$  aussi appelé **rappel (recall)** ou **sensibilité** ;

- **Précision** :  $\frac{VP}{VP + FP}$

- Dans l'exemple précédent,
  - rappel =  $250 / (250+50) = 0.83$
  - Precision =  $250 / (250+150) = 0.63$

		Observations		Total
		+	-	
Prédictions	+	Vrais positifs (VP)	Faux positifs (FP)	Total des positifs prédis (VP + FP)
	-	Faux négatifs (FN)	Vrais négatifs (VN)	Total des négatifs prédis (FN + VN)
	Total	Total des vrais positifs observés (VP + FN)	Total des vrais négatifs observés (FP + VN)	Taille totale de l'échantillon (N)

# Score F1

- Remarque : dans tous les modèles, il y a souvent un équilibre à trouver entre précision et rappel (difficile de maximiser les deux en même temps).
- **F1 score** : métrique agrégée à partir de la précision et du rappel (moyenne harmonique de la **précision** et du **rappel**).

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{rappel}} = \frac{2 * precision * rappel}{precision + rappel}$$

# **Evaluer les modèles de classification**

**Courbe ROC**

# Principe de la courbe de ROC

- Rappels :
  - la classification repose sur un arbitrage entre rappel et précision.
  - Cet arbitrage se base sur le choix d'un seuil de décision qui va favoriser l'un ou l'autre.

$$Y = \begin{cases} '+' \text{ si } P(Y = 1) \geq s, s \in R \\ '-' \text{ sinon} \end{cases}$$

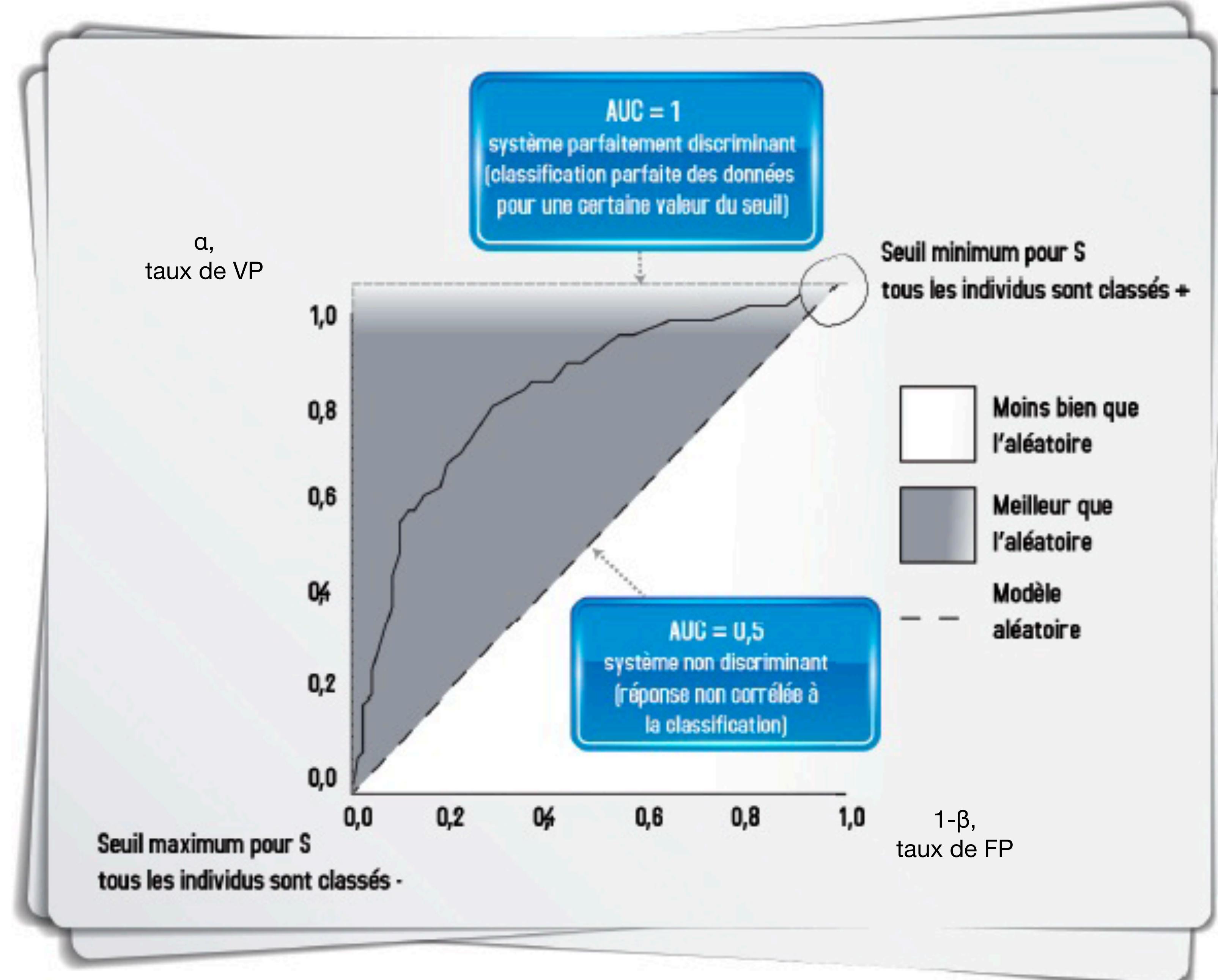
- La courbe ROC va permettre de systématiser l'analyse des résultats d'un classifieur, en fournissant une vue synthétique de sa performance pour toutes les valeurs de  $s$  possibles.

# Construction de la courbe ROC

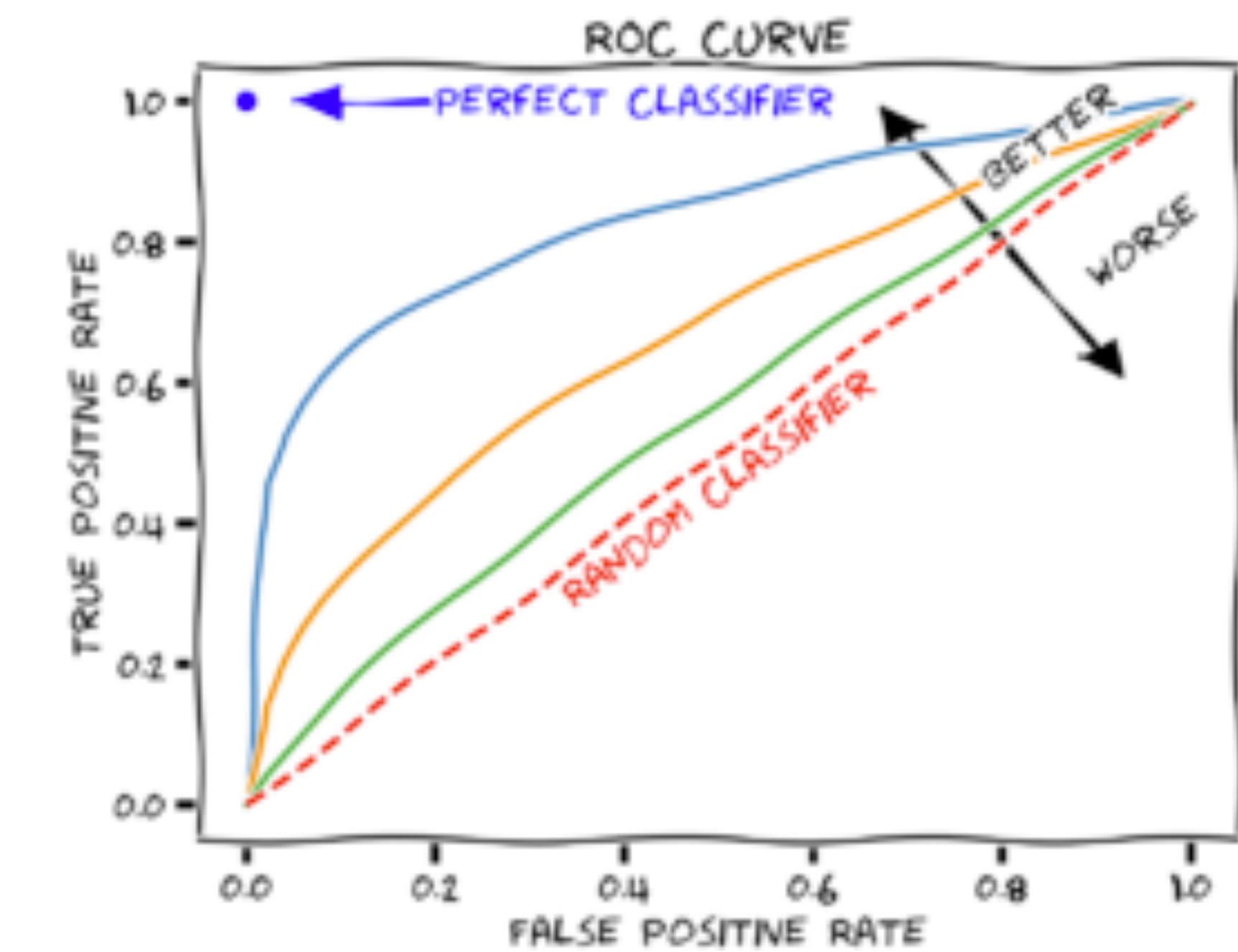
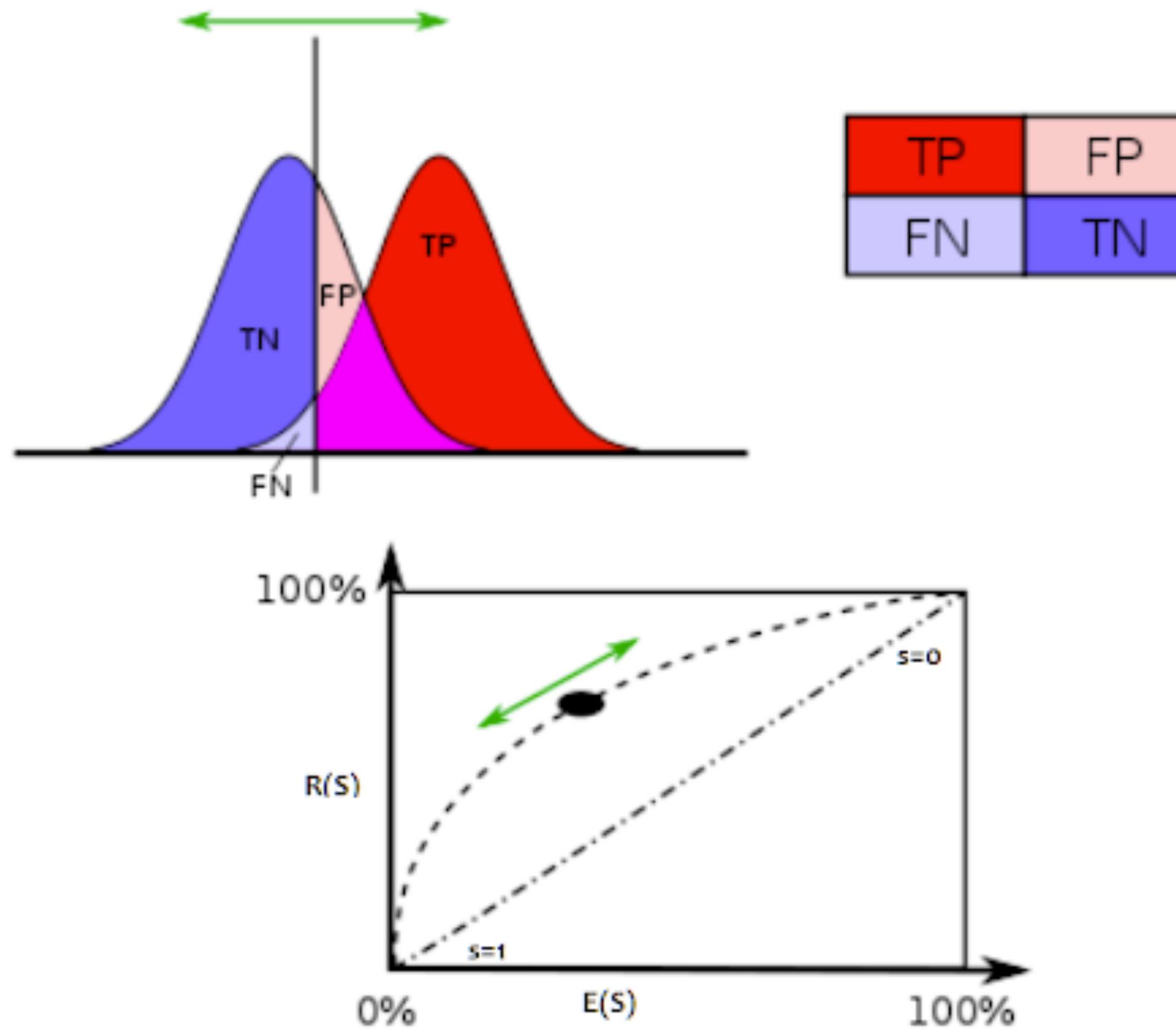
- 2 indicateurs de performance sont utilisés :
  - la **sensibilité**  $\alpha$ , c'est-à-dire le taux de vrais positifs ;
  - la **spécificité**  $\beta$ , c'est-à-dire le taux de vrais négatifs.
- La courbe ROC est tracée dans un espace de deux dimensions définies par  $\alpha$  en ordonnée et  $1-\beta$  en abscisse :  
**on trace le taux de vrais positifs en fonction du taux de faux positifs.**
- **La courbe ROC est donc le graphique  $(\alpha(s), 1-\beta(s))$  ;  $s \in \mathbb{R}$ .**

# Courbe ROC

- La courbe ROC est donc le graphique  $(\alpha(s), 1-\beta(s)) ; s \in \mathbb{R}$ .
- L'analyse de la courbe va permettre de choisir le seuil de décision optimal :
  - le seuil optimal est celui qui est au point le plus proche de l'idéal  $(1,1)$  et au plus loin de la diagonale.
- L'Aire Sous la Surface (AUC Area Under the Curve) donne une **métrique globale du classifier**.  $AUC = 1$  (système parfaitement discriminant).



# Interprétation du ROC

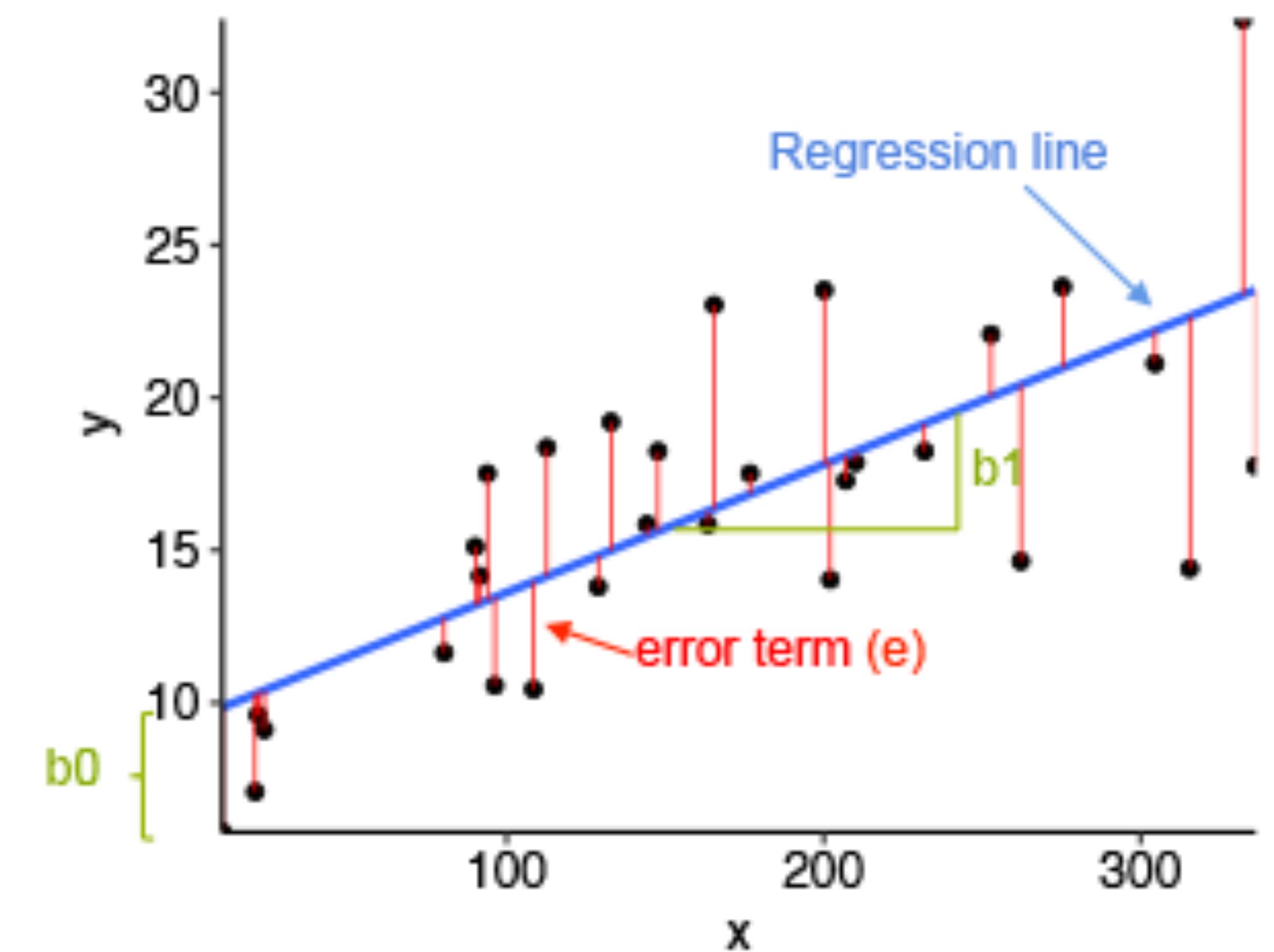


# **Métriques pour les régressions**

# Notations

- **Rappel :**

On parle de **régression** quand les valeurs à prédire sont des  **nombres réels**. Le but n'est donc pas de trouver la valeur exacte mais de s'en rapprocher le plus possible.



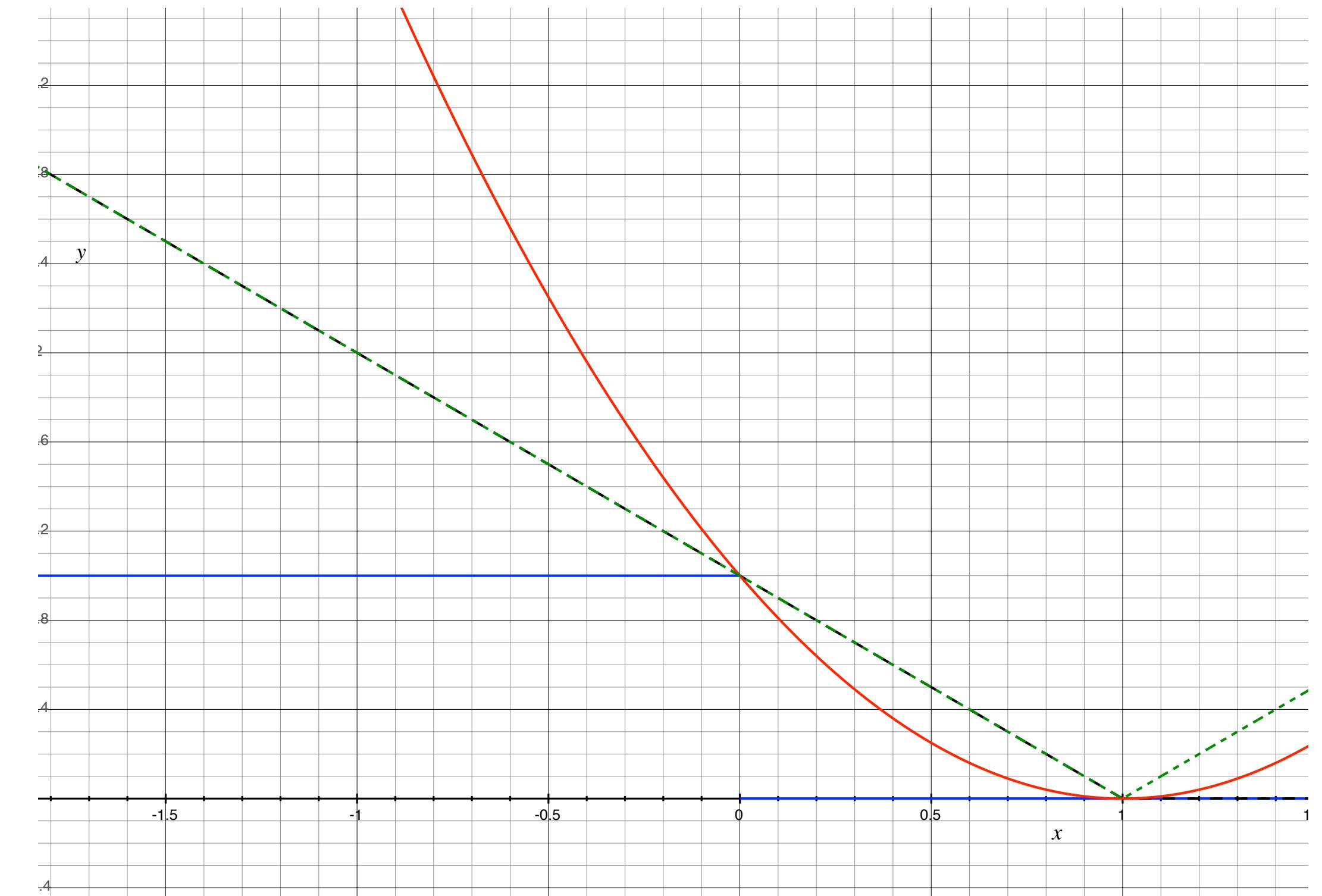
# Notations

- **Notations**
  - la **valeur observée** d'une série à prédire :  $y_i$  ;
  - la **valeur prédictive** par le modèle pour cette même valeur observée  $\widehat{y}_i$  ;
  - une **prévision naïve de référence**, qui est la moyenne de la valeur observée  $\bar{y}$  .
- Elles permettent de calculer, pour tout i des m observations :
  - l'**erreur de prédiction** du modèle :  $y_i - \widehat{y}_i$  ;
  - l'**erreur de prédiction naïve** :  $y_i - \bar{y}$ .

# Exemples de métriques

- Erreur moyenne absolue (MAE, Mean Absolute Error) :

$$\frac{1}{m} \sum_{i=1}^m |y_i - \hat{y}_i|$$



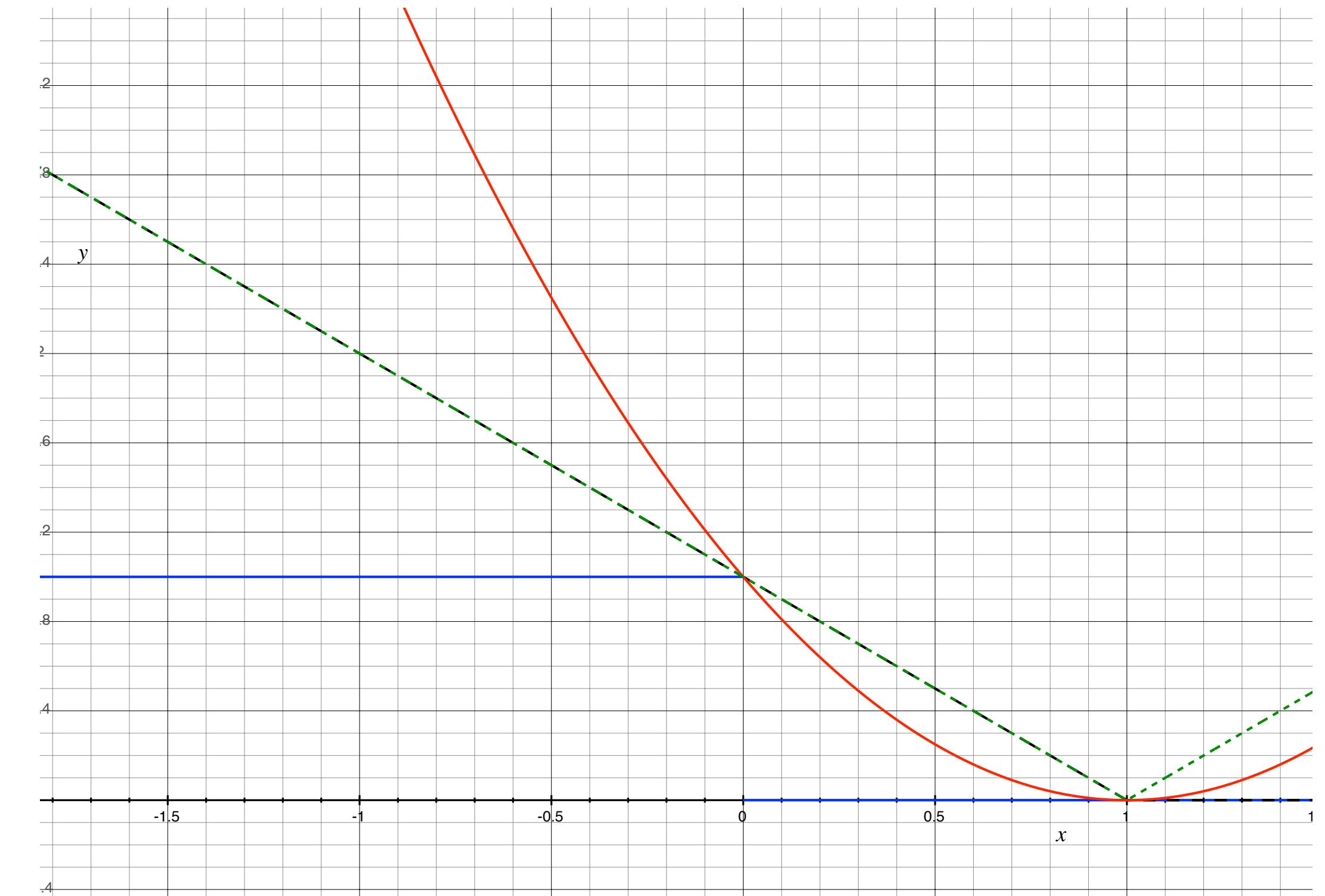
# Exemples de métriques

- Erreur quadratique moyenne (MSE, Mean Square Error) :

$$\frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$$

- Racine carrée de la moyenne du carré des erreurs (RMSE, Root Mean Squared Error)

$$\sqrt{\frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2}$$



# Exemples de métriques

- le coefficient de détermination ( $R^2$ ) :

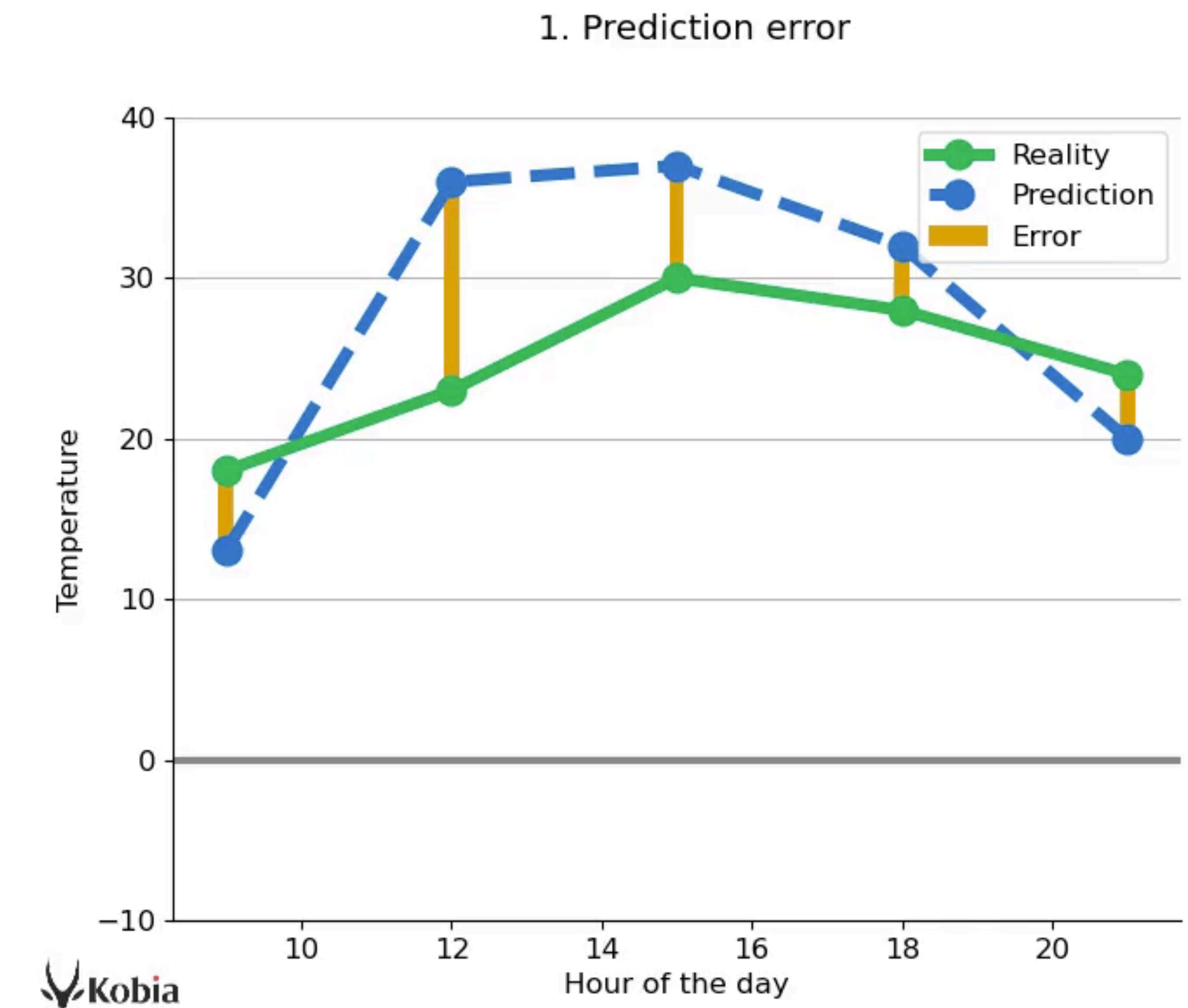
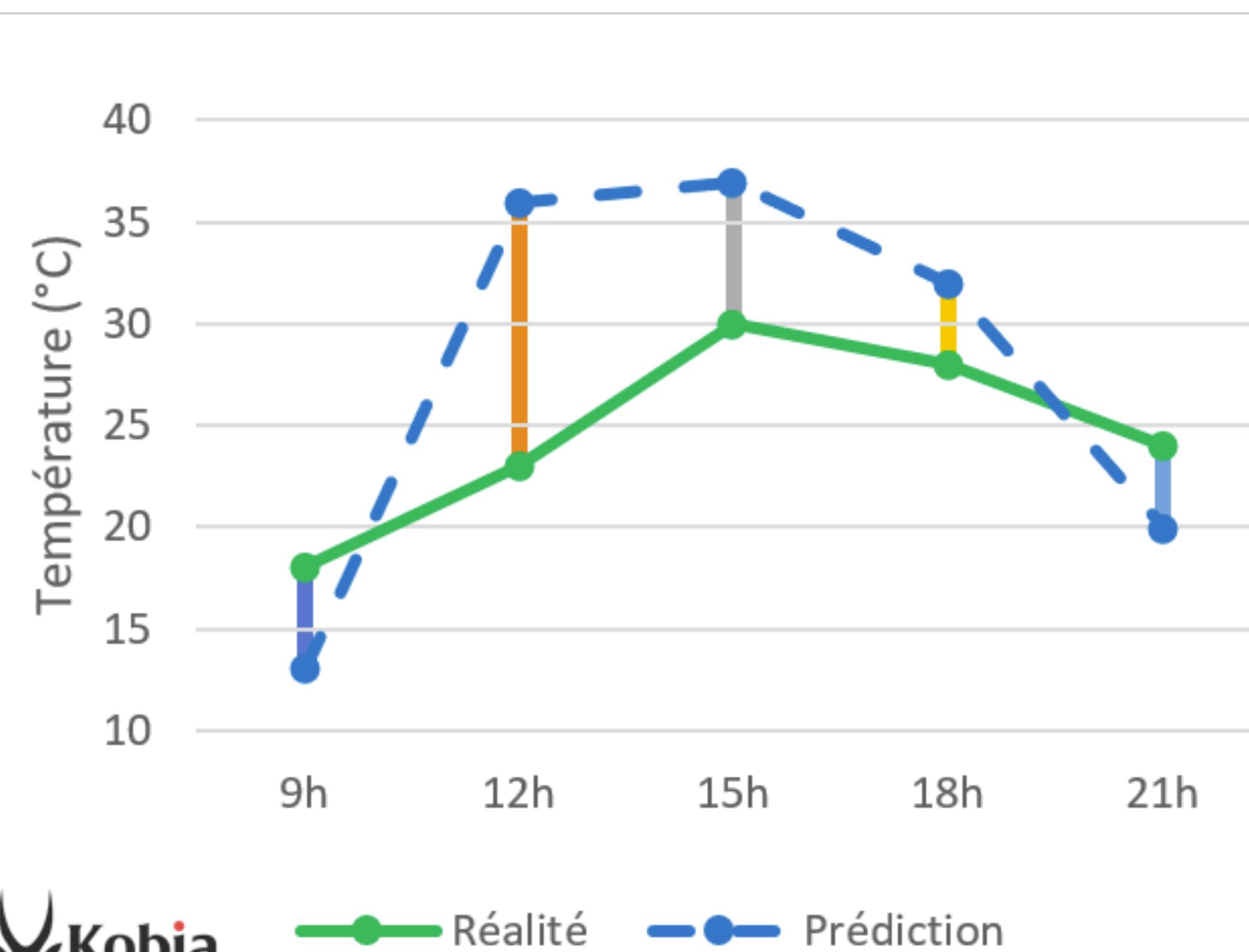
$$1 - \frac{\sum_{i=1}^m (y_i - \hat{y}_i)^2}{\sum_{i=1}^m (y_i - \bar{y})^2}$$

- On peut voir le  $R^2$  comme l'erreur du modèle divisé par l'erreur d'un modèle basique qui prédit tout le temps la moyenne de la variable à prédire.
- Utile pour **comparer 2 modèles**, mais pas pour interpréter 1 seul modèle.

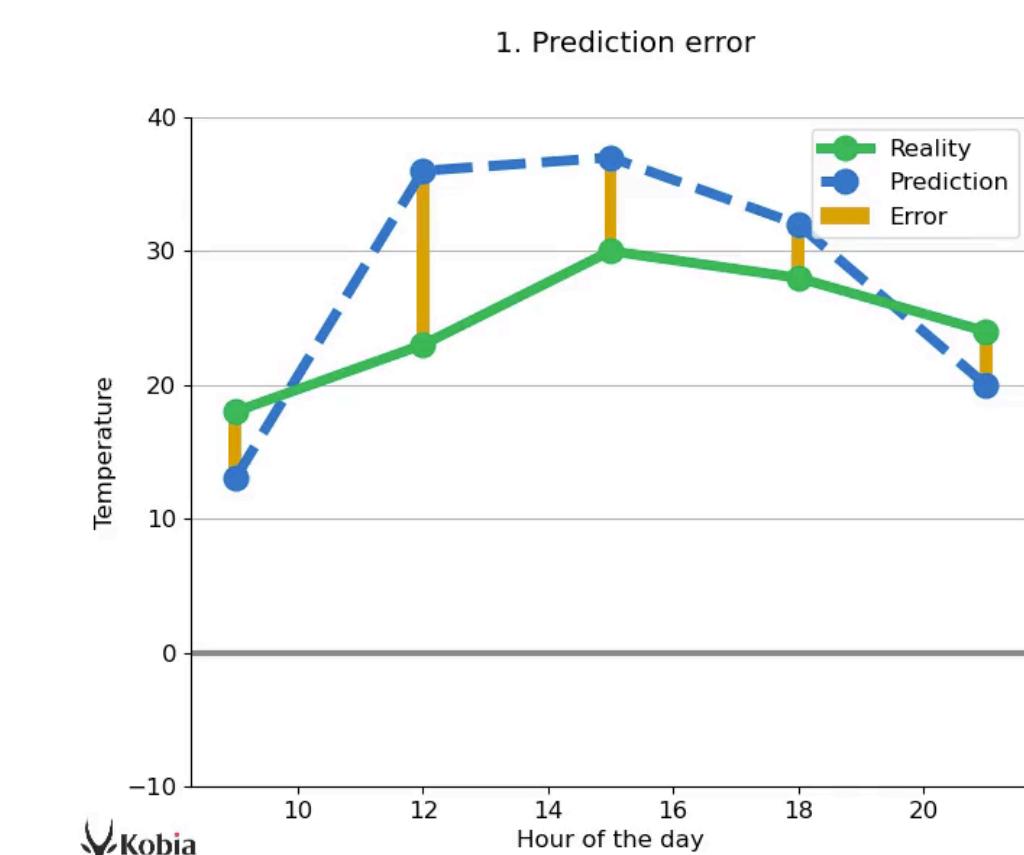
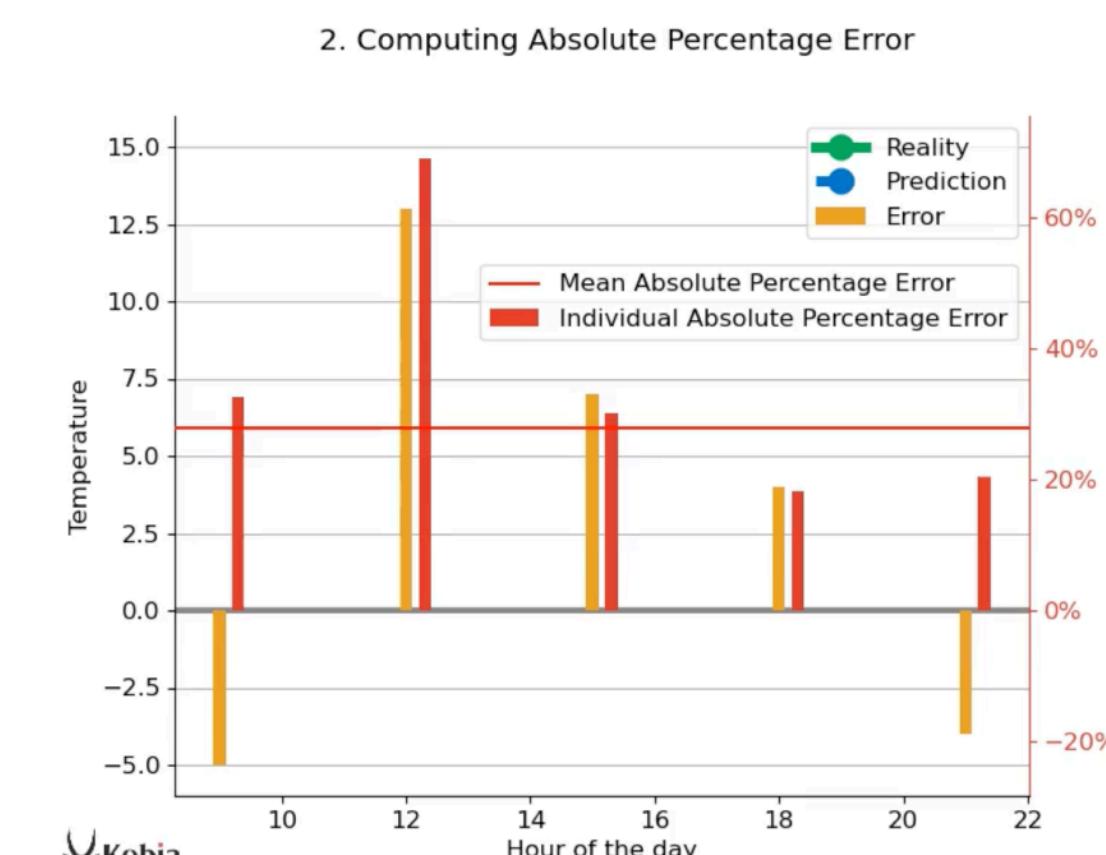
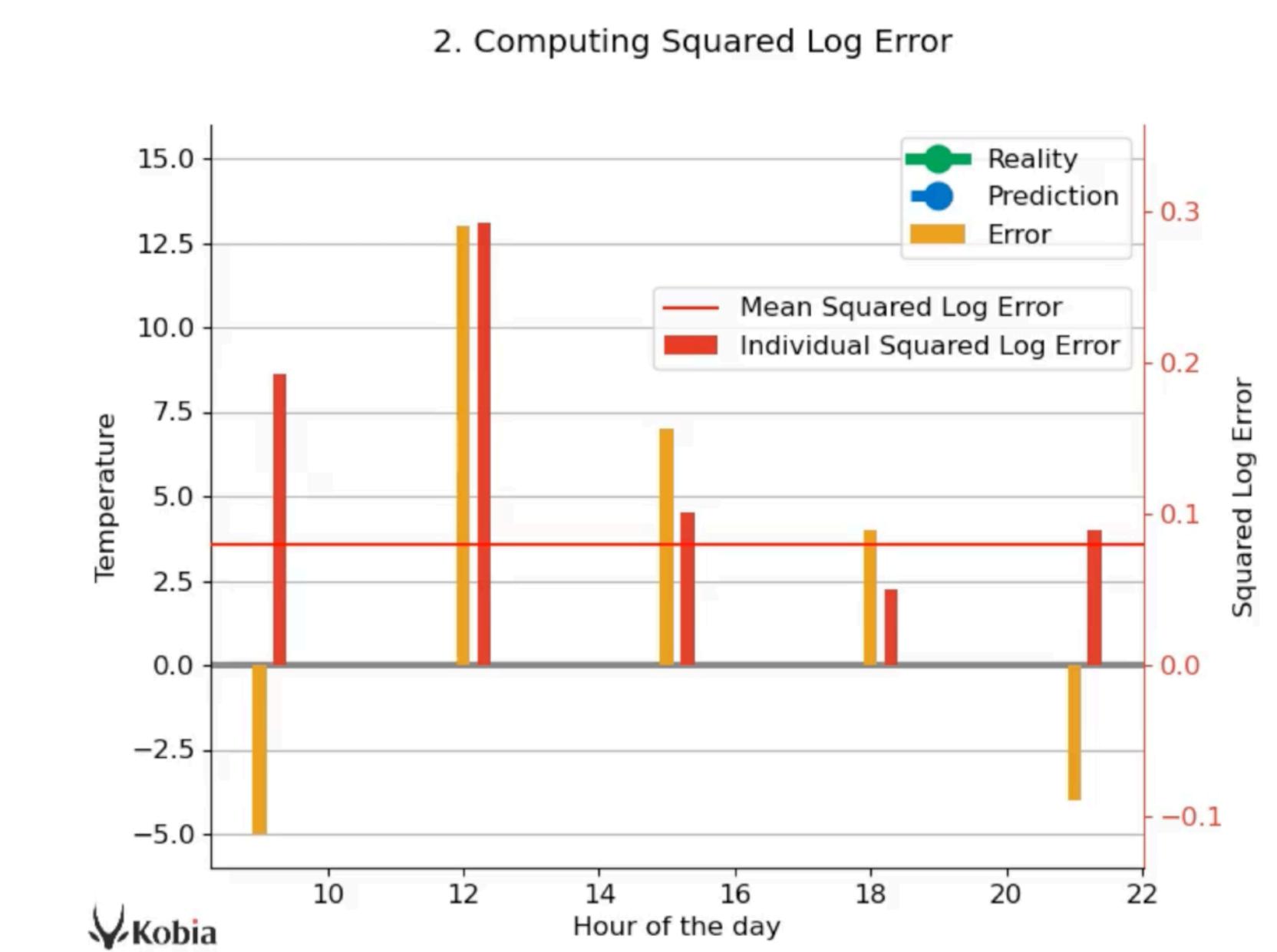
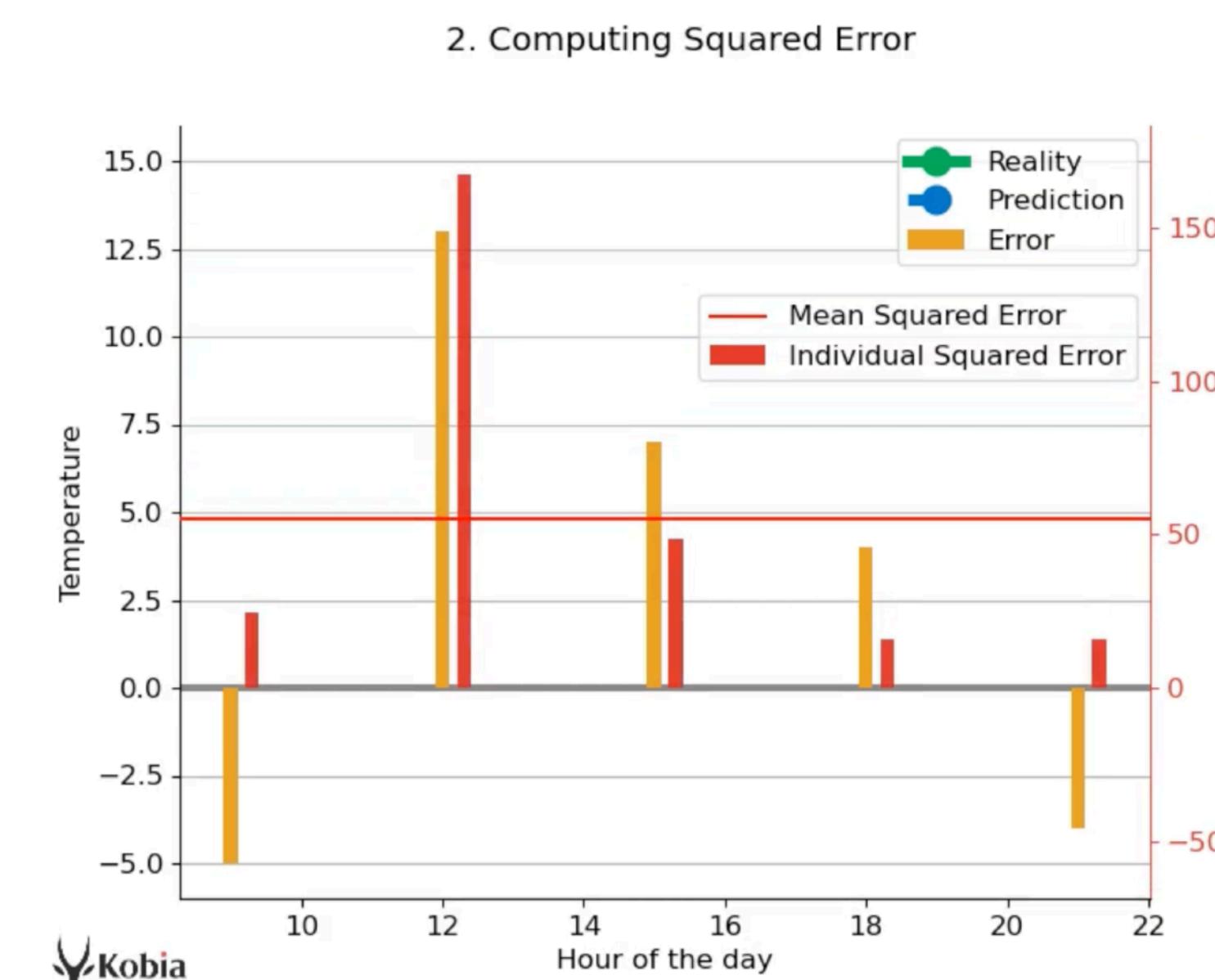
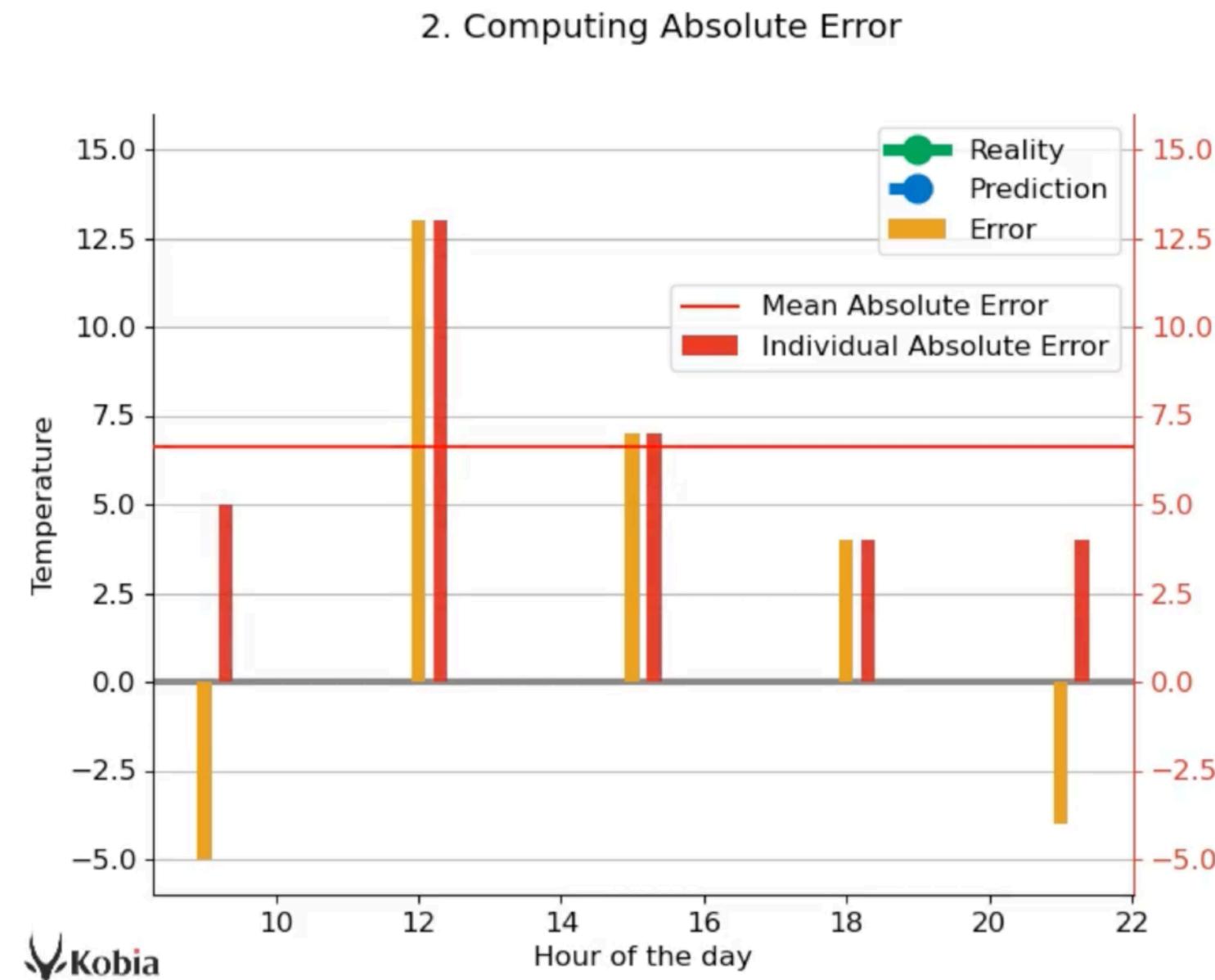
# Comparaison

- Pas de meilleure métrique, elles permettent d'évaluer des modèles selon différentes dimensions !
- **Interprétation :**
  - MAE, MSE et RMSE ont une erreur dans l'unité et l'ordre de grandeur de la valeur observée (facilement interprétable)
  - MAPE interprétation en pourcentage (pourcentage moyen d'écart entre la valeur prédite et la réalité), MSLE difficilement interprétable
- **Outliers :**
  - MSE et RMSE a tendance à plus pénaliser les grandes erreurs  
=> très sensibles aux outliers
  - MAE pénalise autant les petites que les grandes erreurs
  - MSLE peu sensible aux outliers

# Comparaison de métriques



# Comparaison de métriques



# Régression logistique

# Régression linéaire

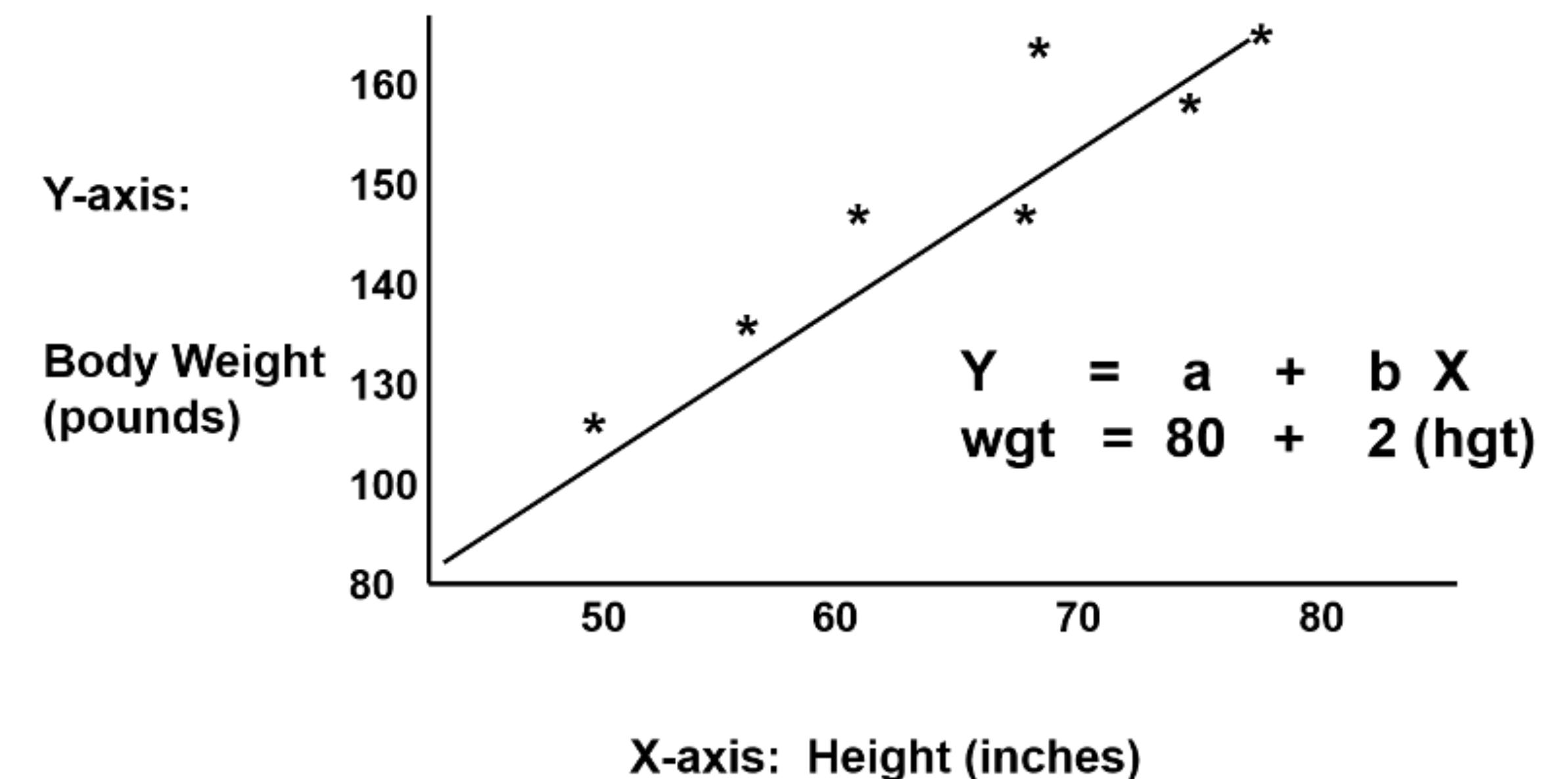
## Principe

- Objectifs : trouver une corrélation linéaire entre une target à prédire et un ensemble de features
- Hypothèses :
  - Les targets doivent être une combinaison linéaire des features
  - Les features doivent être normalement distribuées
- 2 types de régressions linéaires :
  - Simple : 1 seule feature
  - Multiple : plusieurs features

# Régression linéaire simple linéaire

- Trouver les coefficients  $a$  et  $b$  qui minimisent l'erreur entre une target  $Y \in \mathbb{R}^N$  dépendant linéairement d'une observation  $X \in \mathbb{R}^{N \times 1}$  indépendante

$$Y = aX + b$$



# Régression linéaire multiple

- Généralisation de la régression linéaire simple à plusieurs features  $X \in \mathbb{R}^{N \times d}$

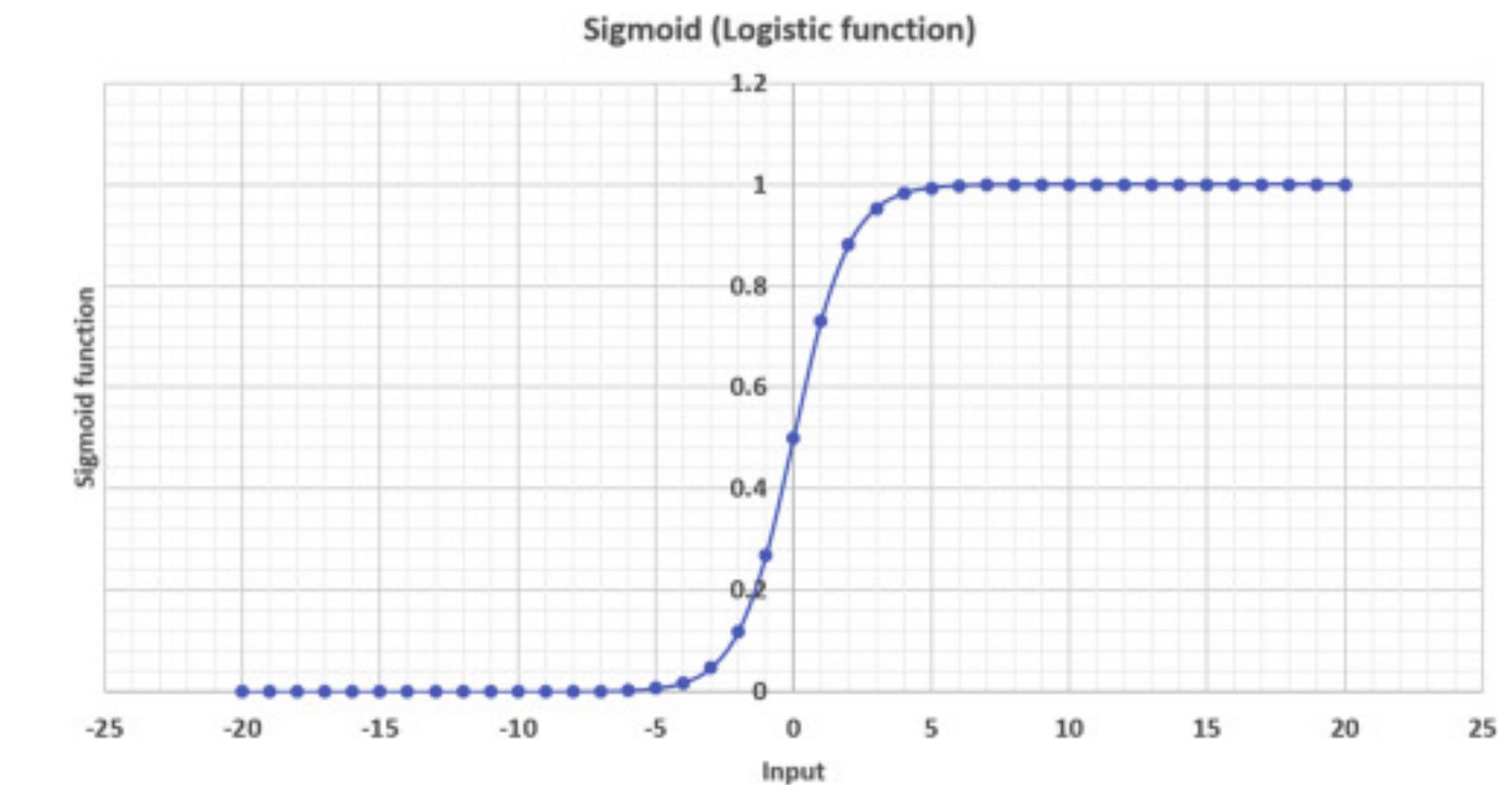
$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_d X_d + \epsilon$$

- Les coefficients  $\beta_i$  sont obtenus par descente de gradient (vu avec M. Berro) ou par résolution des équations.

# Régression logistique

## Principe

- Application de la régression linéaire à la classification binaire
  - Utilisation d'une fonction sigmoid (logistic) pour prédire la probabilité d'appartenance à une classe
- Fonction logistique :  $y = \frac{1}{1 + e^{-x}} \in [0,1]$
- Si la probabilité est  $>0.5$ , on prédit la classe 1
  - Sinon, on prédit la classe 0



# Régression logistique

- La régression logistique est une forme paramétrique d'une distribution de probabilité  $P(Y|X)$  où  $Y \in \mathbb{R}^N$  est l'espace à prédire et  $X \in \mathbb{R}^{N \times d}$  l'espace des features
- Le modèle paramétrique peut être écrit :

$$P(Y = 1 | X) = \frac{1}{1 + \exp\left(\beta_0 + \sum_{i=1}^d \beta_i X_i\right)}$$

et

$$P(Y = 0 | X) = \frac{\exp\left(\beta_0 + \sum_{i=1}^d \beta_i X_i\right)}{1 + \exp\left(\beta_0 + \sum_{i=1}^d \beta_i X_i\right)}$$

- Les coefficients  $\beta_i$  sont optimisés pour maximiser la vraisemblance conditionnelle des données, c'est à dire la probabilité d'apparition d'une observation Y dans le dataset

# Régression logistique

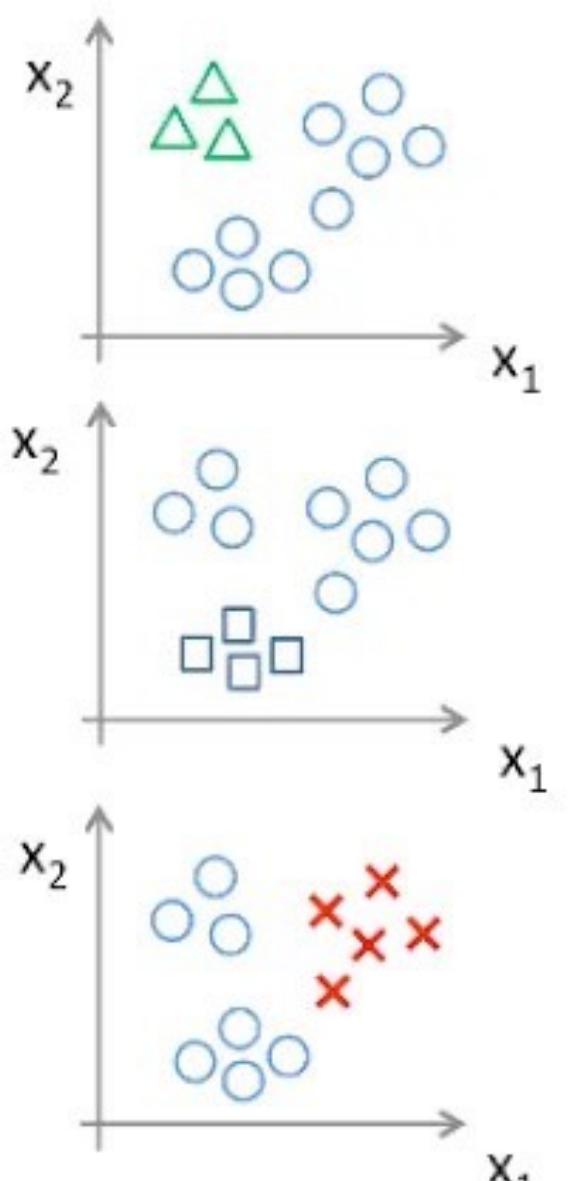
## Cas multiclasse

- Stratégie du “One-vs-rest”
  - On cherche un modèle qui sépare 1 classe de toutes les autres classes du dataset
  - Séparation de 3 classes [rouge, vert, bleu]
    - 1 modèle logistique de “rouge” vs [vert+bleu]
    - 1 modèle logistique de “vert” vs [rouge+bleu]
    - 1 modèle logistique de “bleu” vs [rouge+vert]
  - On obtient une segmentation de l'espace en assemblant les différents modèles

One-vs-all (one-vs-rest):

Class 1: **Green**  
Class 2: **Blue**  
Class 3: **Red**

<https://antonhaugen.medium.com/introducing-mllibs-one-vs-rest-classifier-402eeab22493>



# Regression logistique: Avantages et inconvénients

## Avantages

- Facile à interpréter
- Efficacité computationnelle
- Peu sensible au sur-apprentissage
- Pas besoin de normalisation

## Inconvénients

- Hypothèse de linéarité
- Sensibilité au déséquilibre de classe
- Performance limitée sur les données complexes
- Sensibilité aux données aberrantes

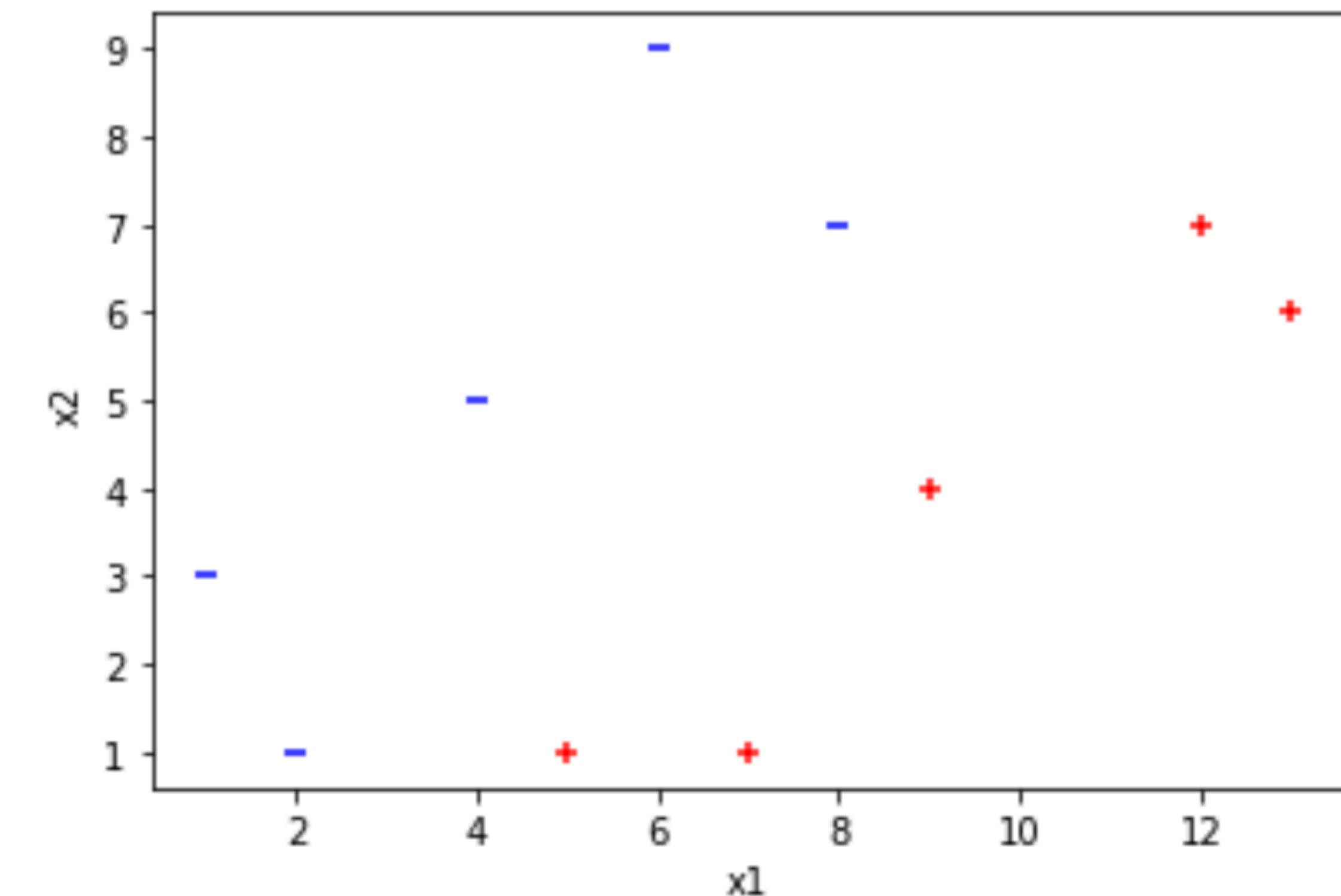
# **Support Vector Machine**

# Introduction

- Le **Support Vector Machine** ou SVM est un puissant algorithme capable de trouver des patterns fortement non linéaires. Il repose sur deux idées essentielles :
  - la **maximisation de la marge** entre la frontière de décision et les exemples les plus proches, qu'on appelle les **vecteurs de support** ;
  - et le choix d'un **hyperplan séparateur dans un nouvel espace de combinaisons non linéaires entre les variables**, dans lequel une séparation linéaire des individus sera possible.

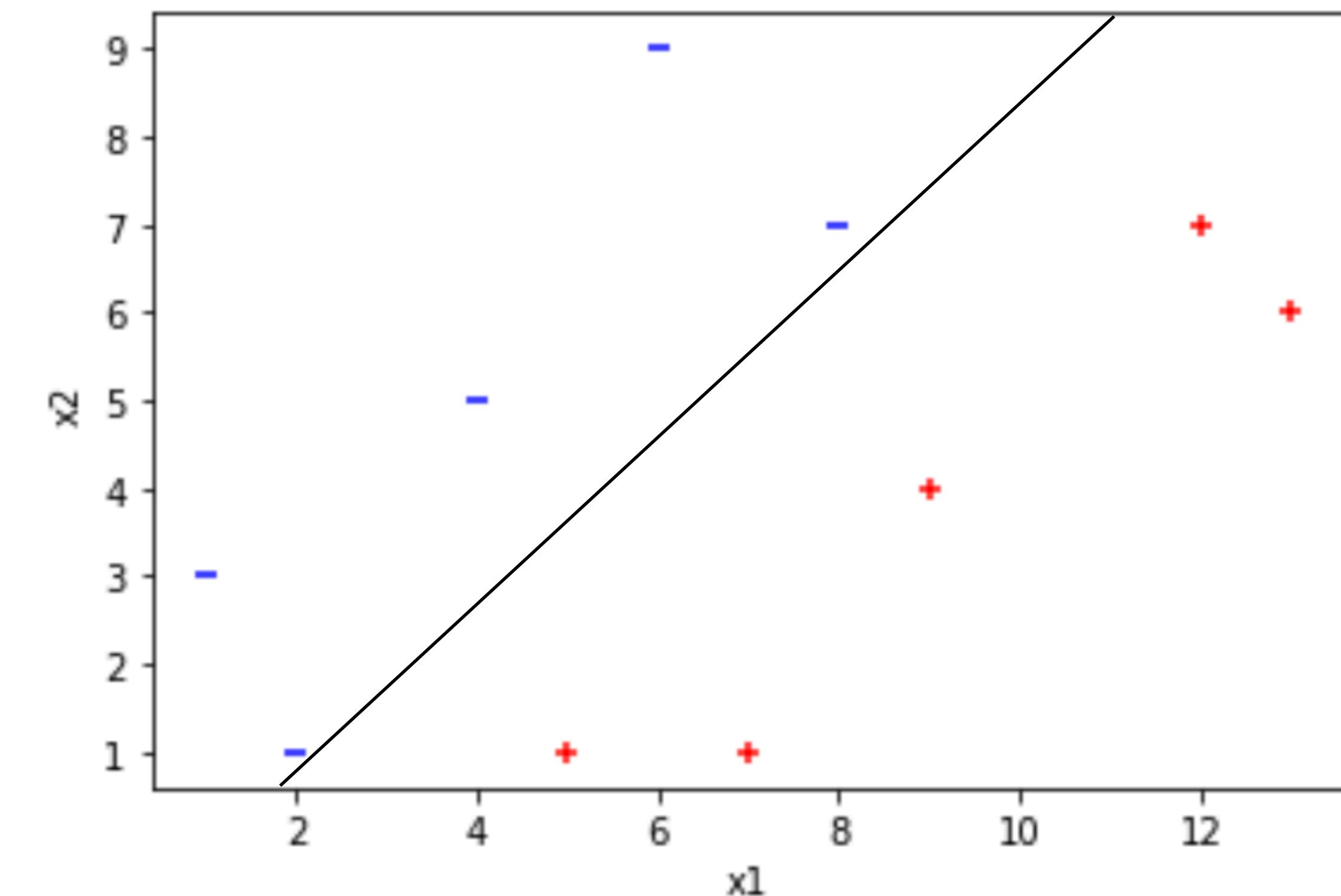
# Introduction : discrimination linéaire

- Soit le cas le plus simple : classification **binaire** de  $m$  observations (en **2 dimensions**) linéairement séparables.



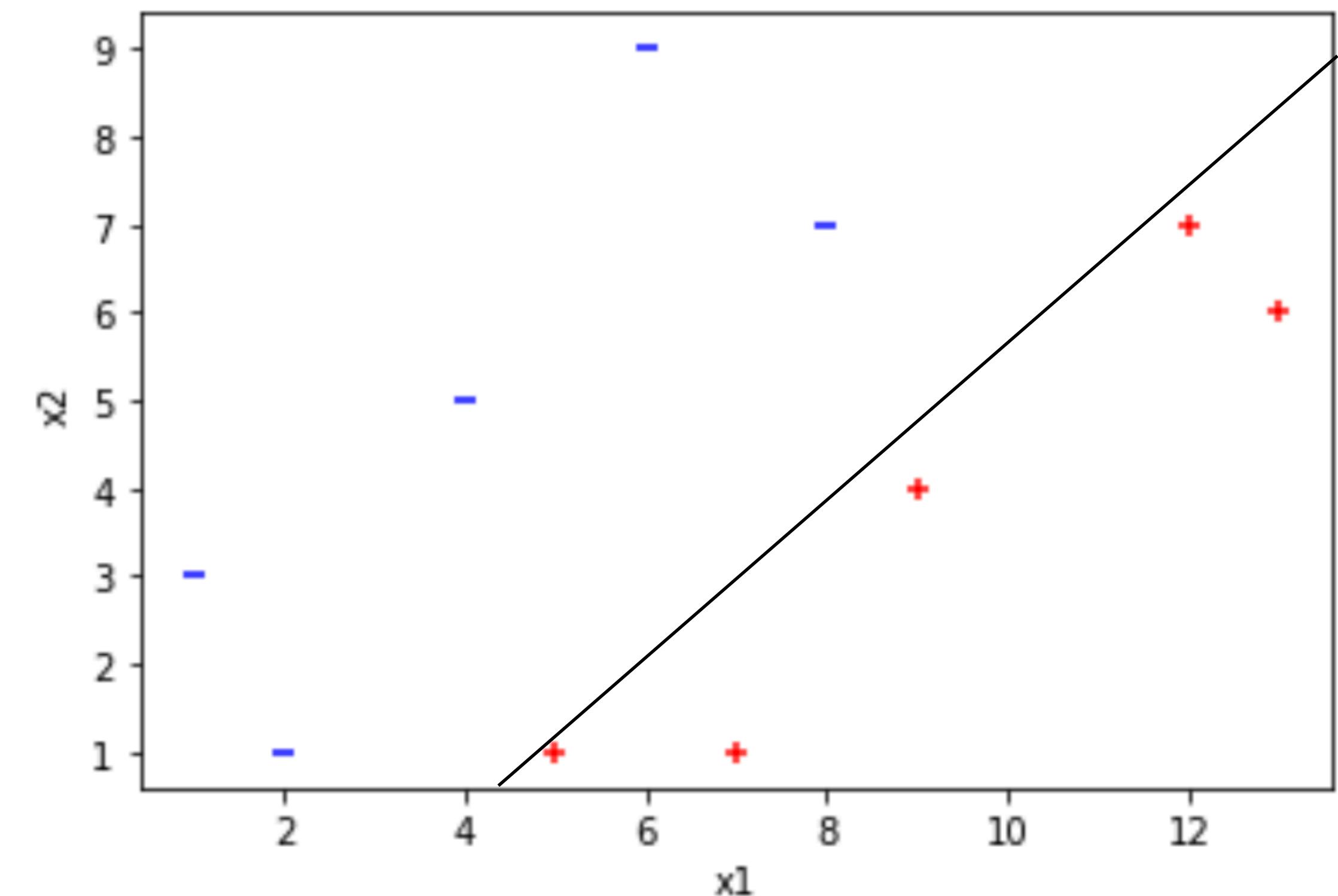
# Introduction : discrimination linéaire

- Soit le cas le plus simple : classification **binaire** de  $m$  observations (en **2 dimensions**) linéairement séparables.



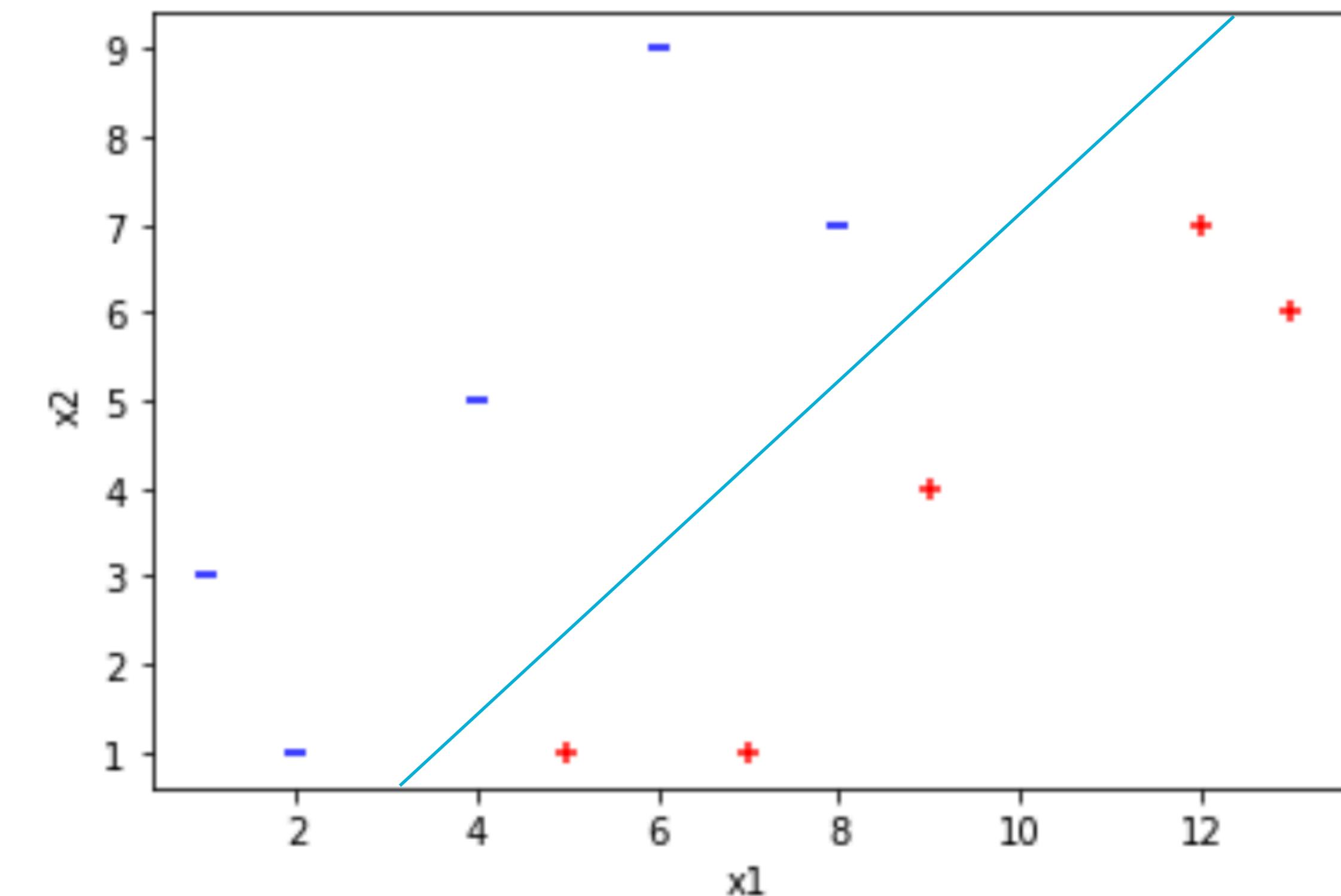
# Introduction : discrimination linéaire

- Soit le cas le plus simple : classification **binaire** de  $m$  observations (en **2 dimensions**) linéairement séparables.



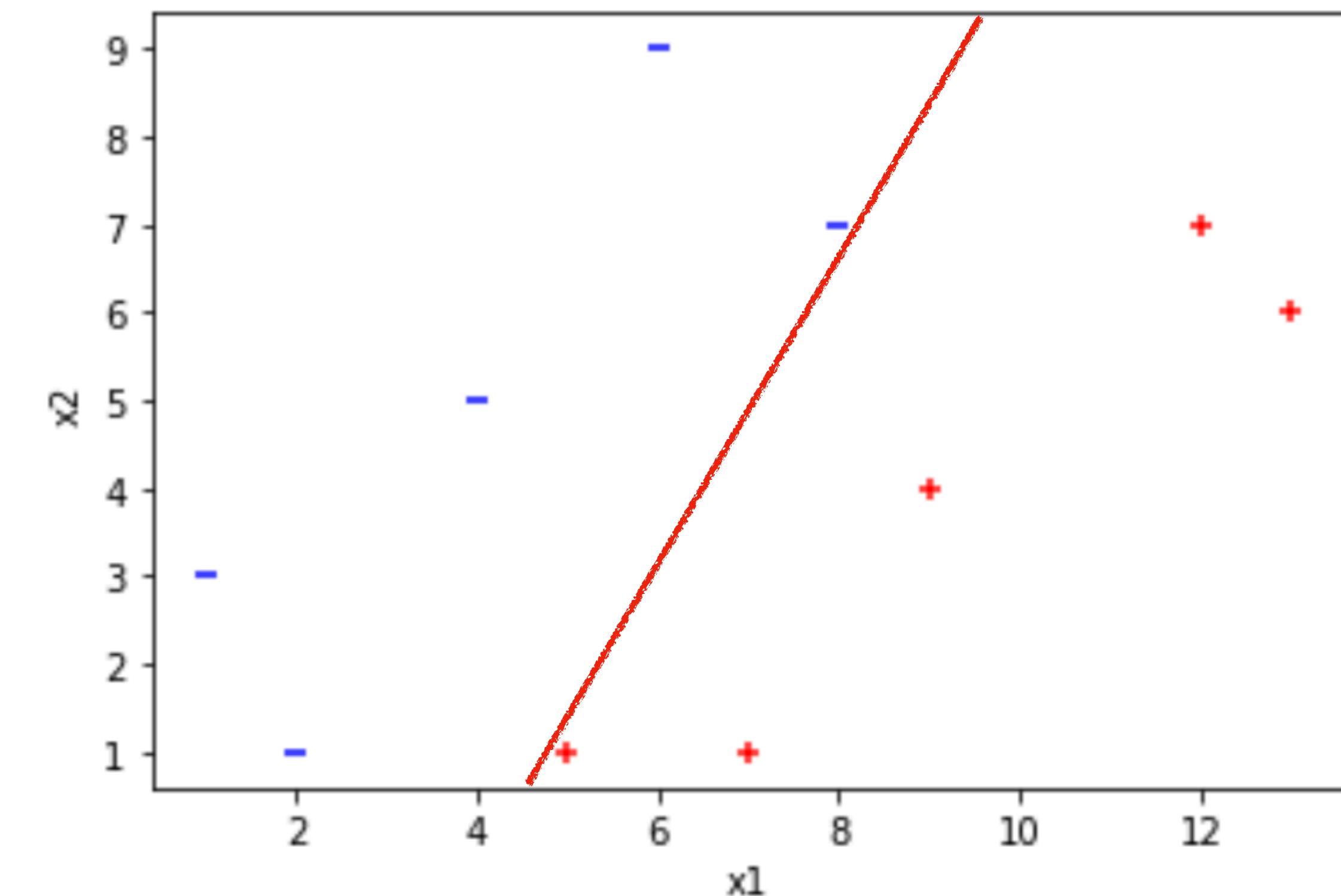
# Introduction : discrimination linéaire

- Soit le cas le plus simple : classification **binaire** de  $m$  observations (en **2 dimensions**) linéairement séparables.



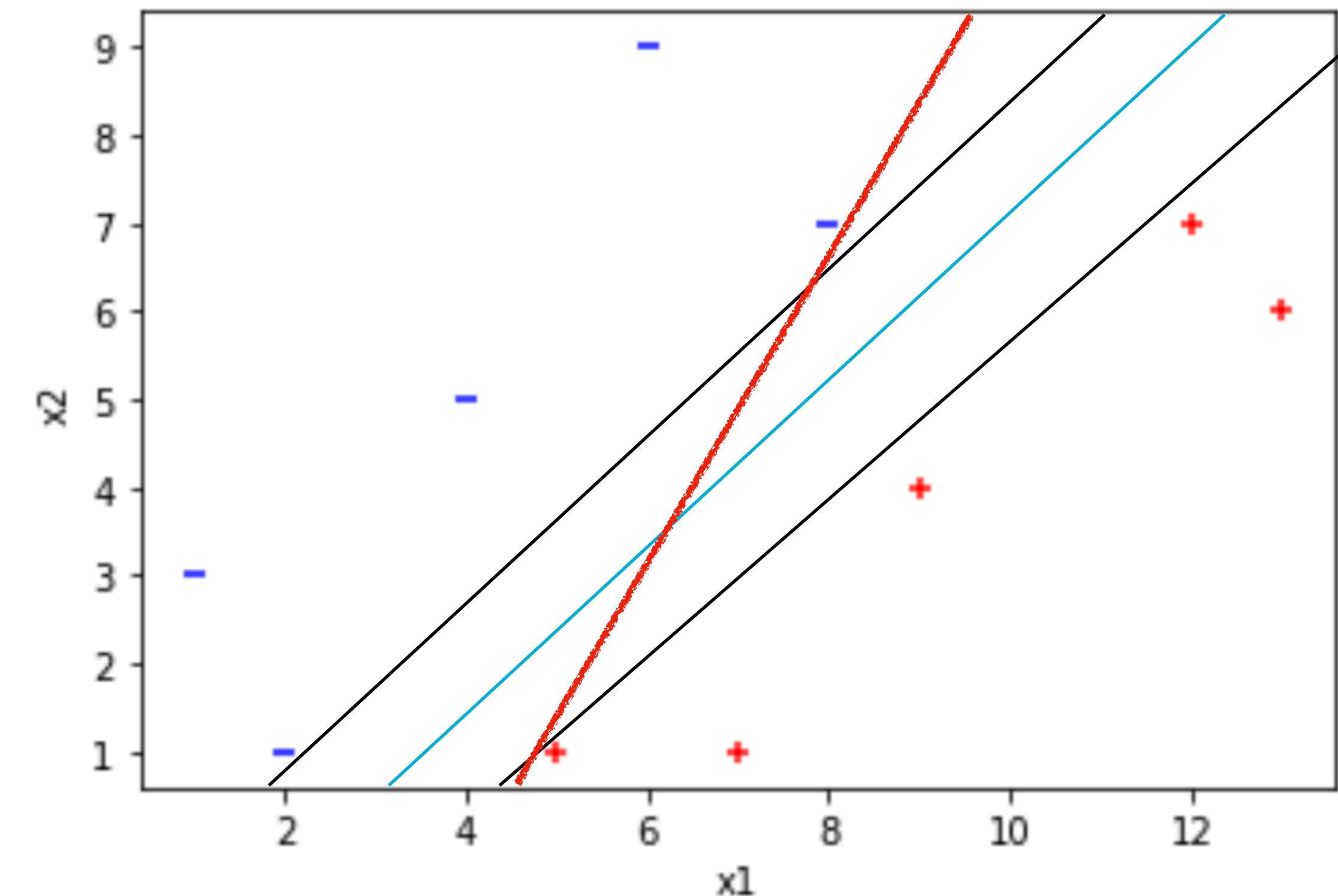
# Introduction : discrimination linéaire

- Soit le cas le plus simple : classification **binaire** de  $m$  observations (en **2 dimensions**) linéairement séparables.



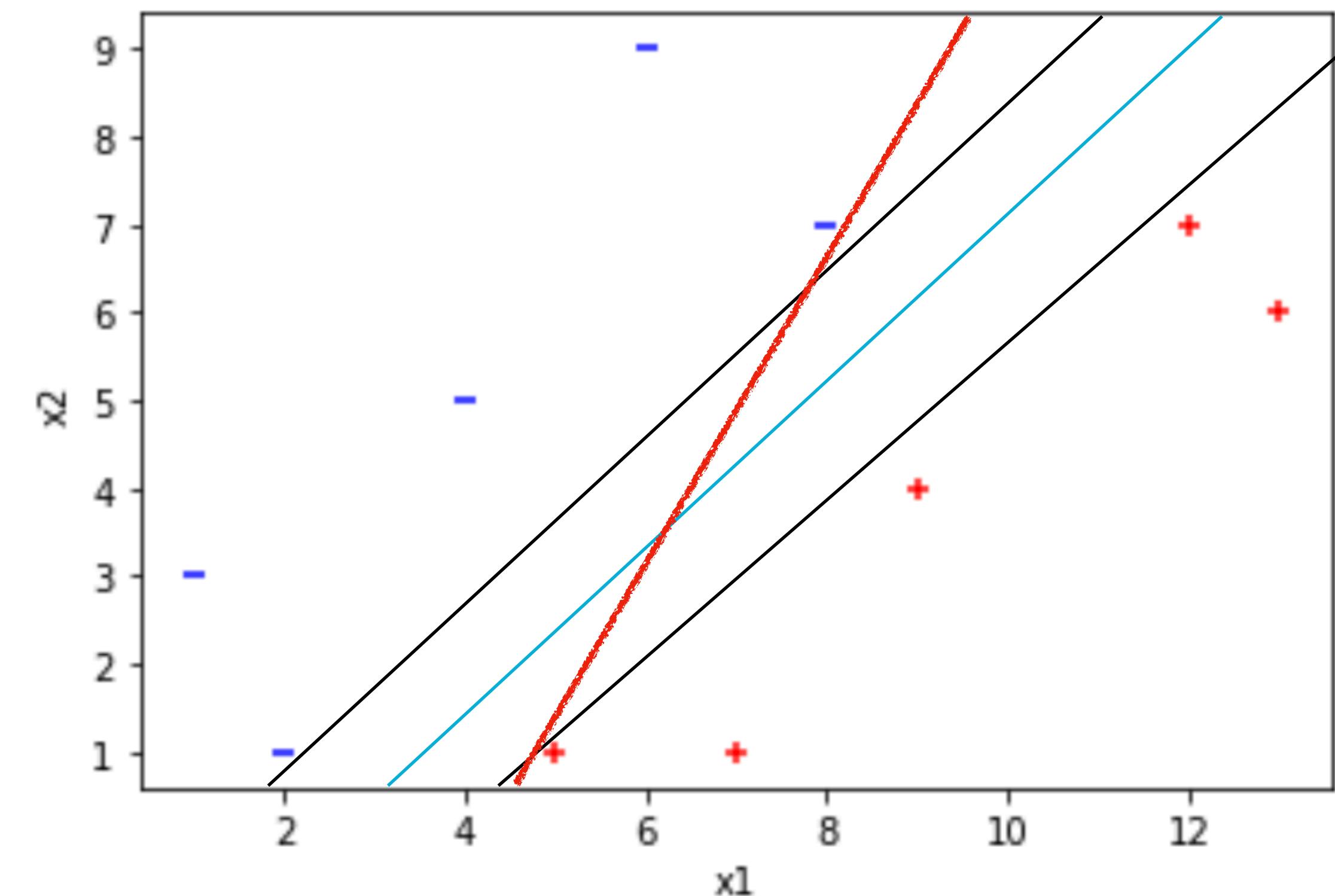
# Introduction : discrimination linéaire

- Soit le cas le plus simple : classification **binaire** de  $m$  observations (en **2 dimensions**) **linéairement séparables**.
- **Que dire de ces 4 modèles ?**
- Note :
  - en 2 dimensions, le séparateur est une droite ;
  - en 3 dimensions, un plan ;
  - en  $N$  dimensions, un **hyperplan** (espace de  $N-1$  dimensions)



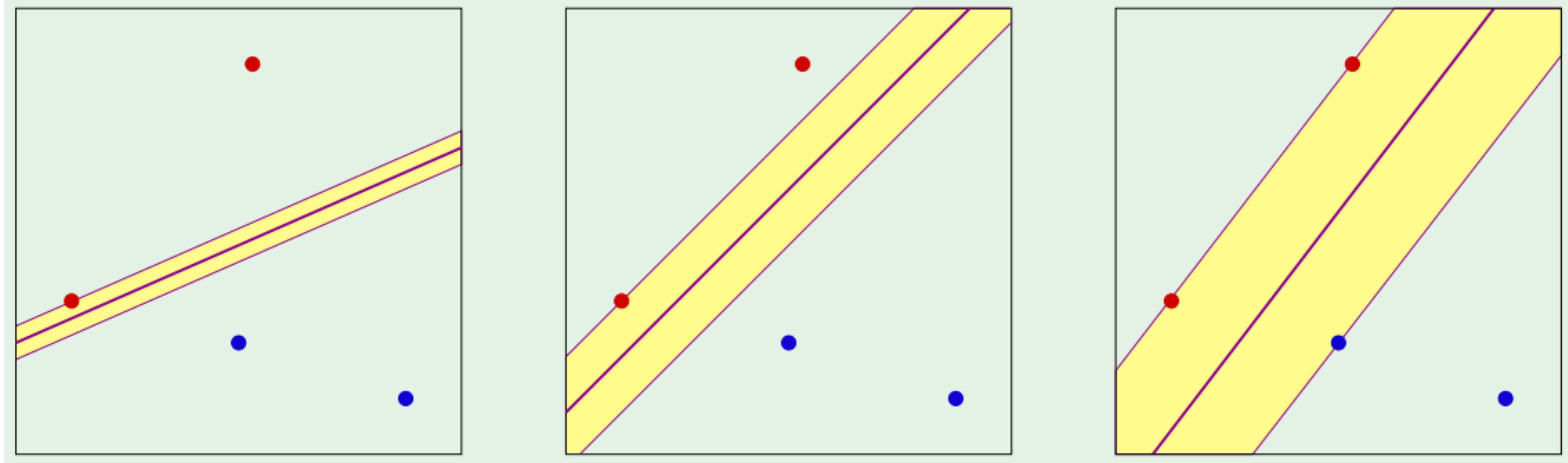
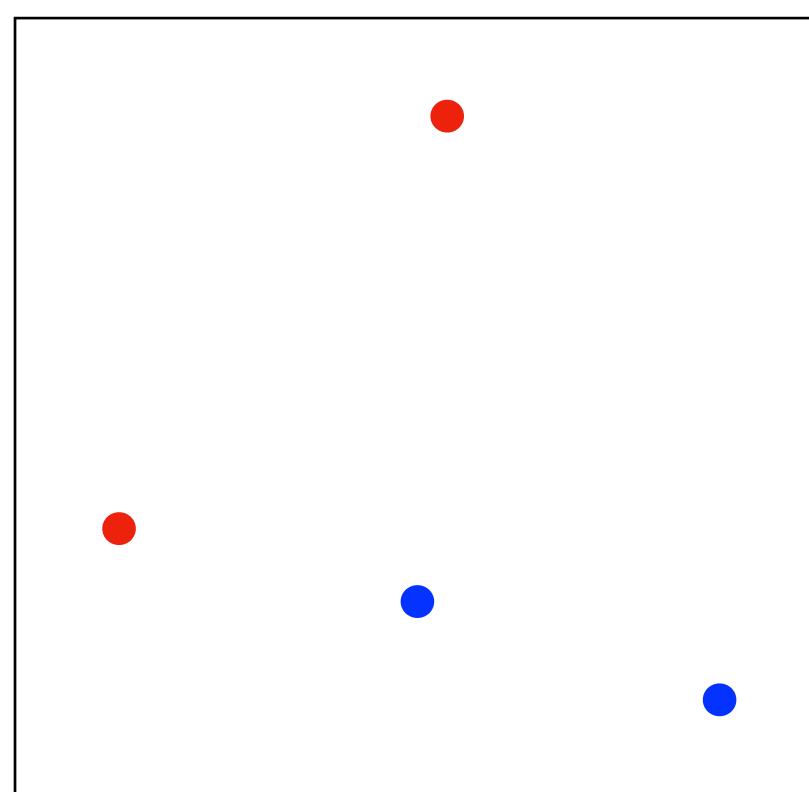
# Introduction : discrimination linéaire

- Ces 4 modèles qui offrent le même score de classification.
- Il en existe une infinité d'autres possibles.
- Ce qui les différencie, c'est leur **capacité à généraliser** : si le nombre d'observations augmente, intuitivement le modèle bleu devrait produire moins de faux négatifs ou positifs.
- Ce qui les différencie est leur **marge**.



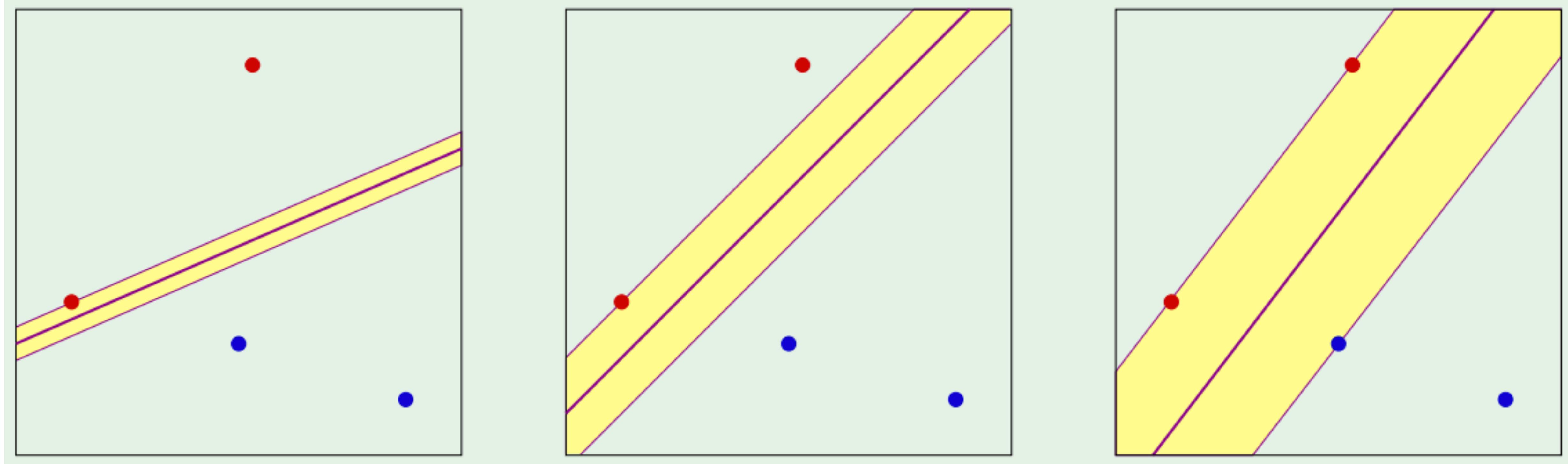
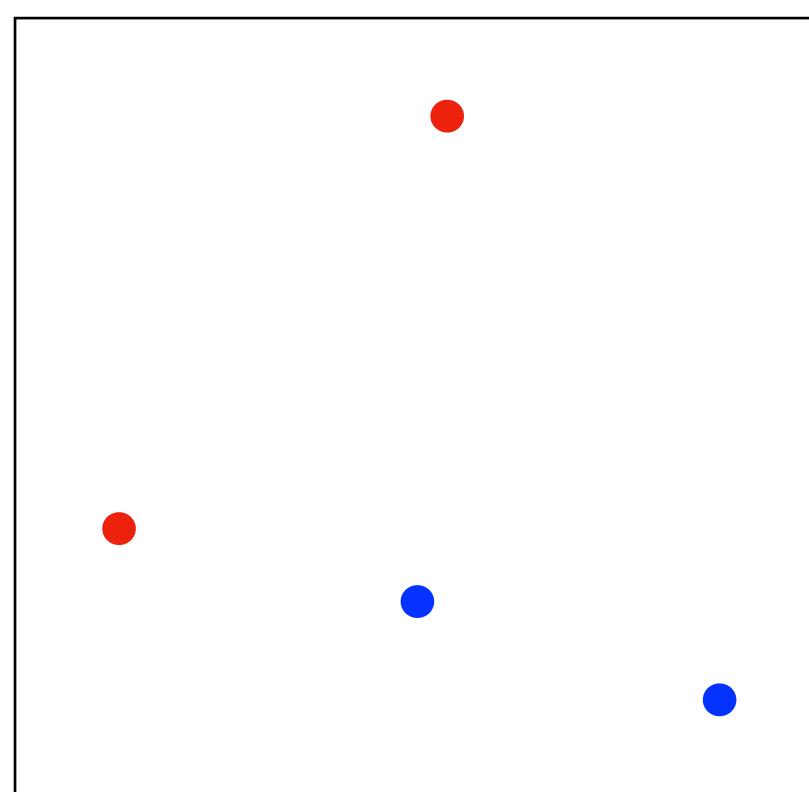
# Introduction à la notion de marge

- Considérons différents séparateurs



# Introduction à la notion de marge

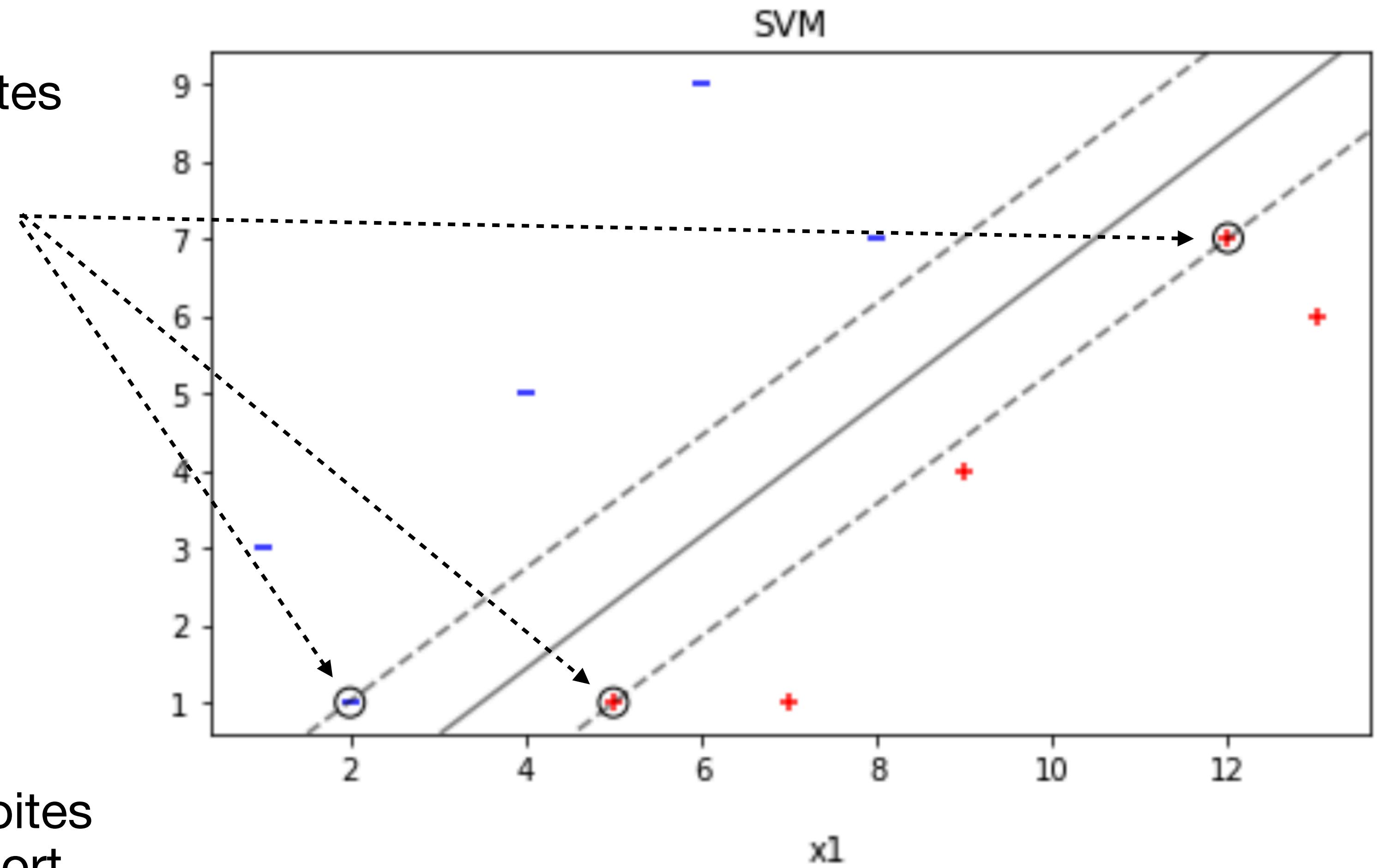
- Considérons différents séparateurs



- La droite (ou plus généralement l'Hyperplan) optimal doit **maximiser la distance** entre la frontière de séparation et les points de chaque classe qui lui sont le plus proche [HTF, page 132]

# Maximisation de la marge

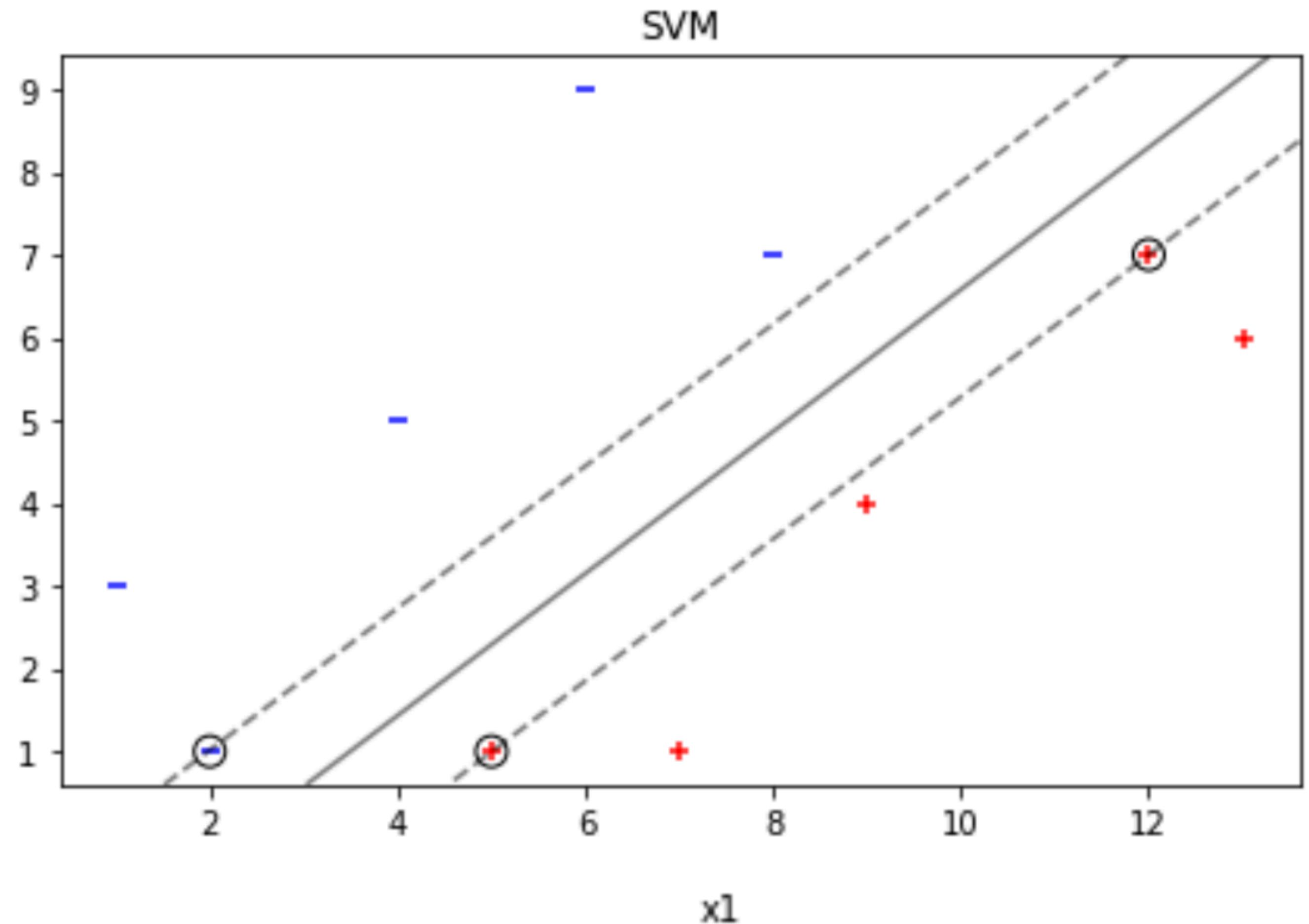
- Les points où «s'appuient» les droites «marges» sont les **«vecteurs de support»**.  
Si on les retire de l'échantillon, la solution est modifiée.
- Plusieurs zones sont définies dans l'espace de représentation
  - $f(x) = 0$ , on est sur la frontière
  - $f(x) > 0$ , on classe «+»
  - $f(x) < 0$ , on classe «-»
  - $f(x) = +1$  ou  $-1$ , on est sur les droites délimitant des vecteurs de support



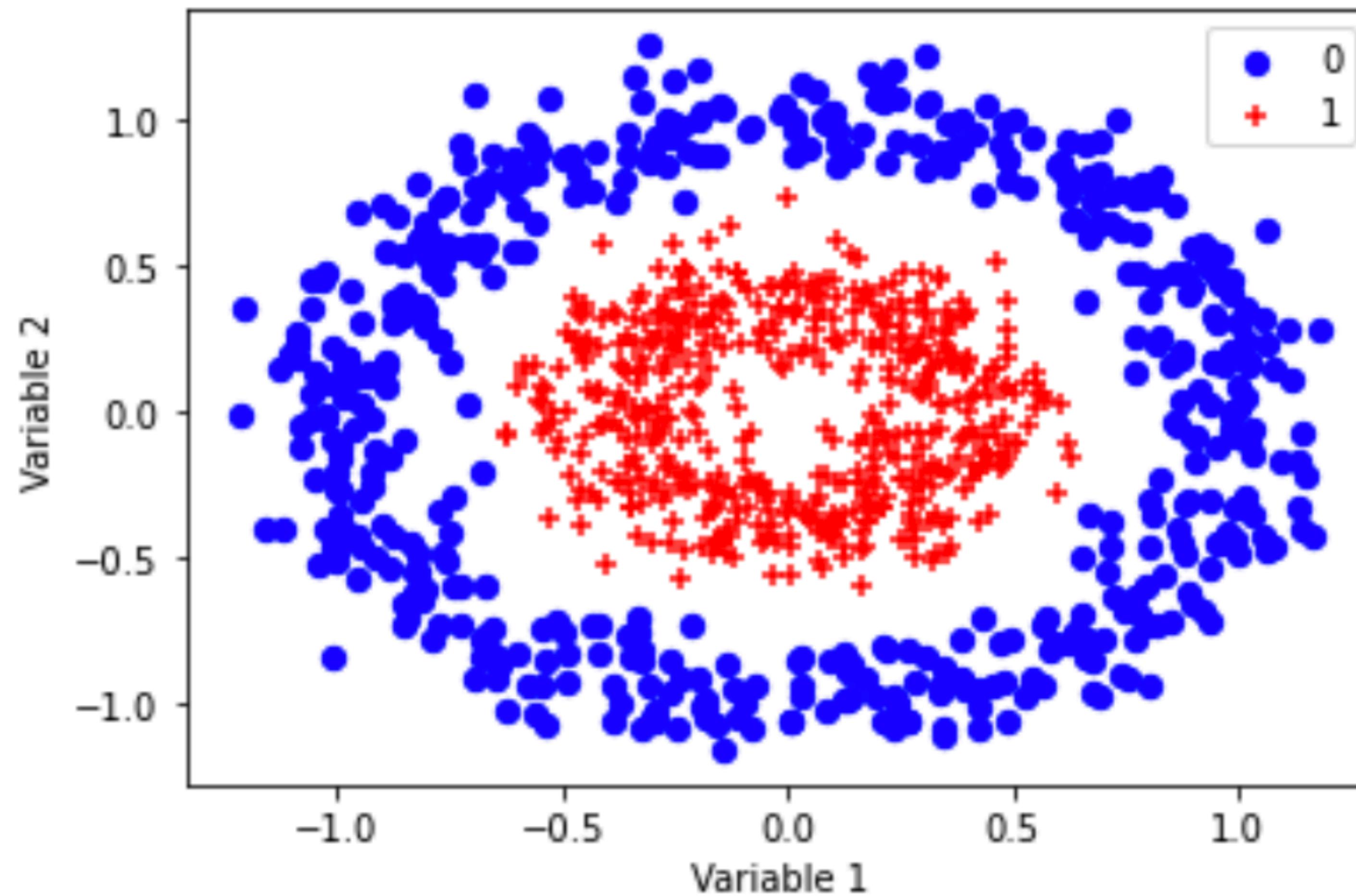
# Maximisation de la marge

$$d = \frac{|a^T \omega + \omega_0|}{\|\omega\|}$$

- Plusieurs zones sont définies dans l'espace de représentation
  - $f(x) = +1$  ou  $-1$ , on est sur les droites délimitant des vecteurs de support
- On en déduit donc la marge maximale vaut :  $\frac{2}{\|\omega\|}$

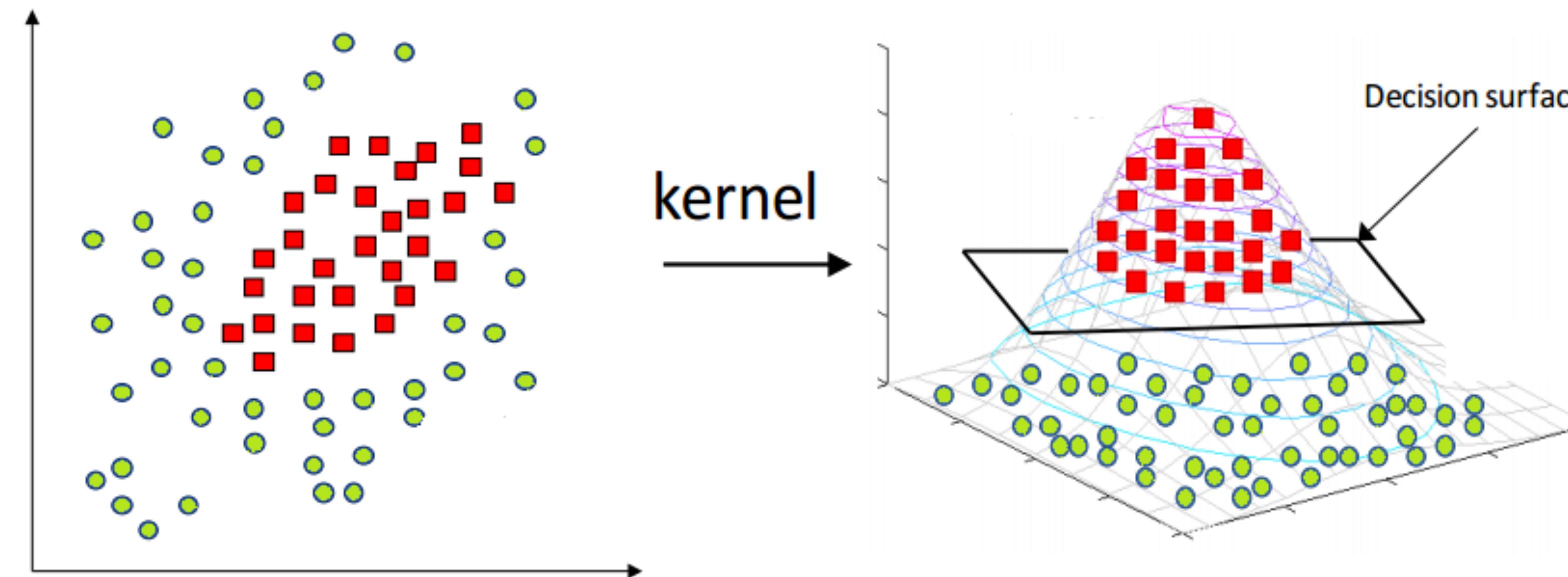


# Classification non-linéaire



# Kernel trick

- Le kernel trick permet de passer dans **un espace de dimension supérieur**, mais **sans avoir à transformer explicitement nos observations** dans cet espace de redescription (trop coûteux) par des observations mathématiques ...



# Kernel trick : fonctions noyau usuelles

- Il existe différentes formes de kernel, les plus populaires étant :
  - le kernel polynomial :

$$K(x, x') = (1 + x^T x')^q$$

- le kernel RBF (pour Radial Basis Function) ou Kernel Gaussien

$$K(x, x') = e^{-\gamma x - x'^2}$$

- le kernel sigmoid

$$K(x, x') = \tanh(kx x' - \delta)$$

# SVM: Avantages et inconvénients

## Avantages

- Efficace dans les espaces de haute dimension
- Robustesse face au sur-apprentissage
- Flexibilité avec les fonctions de noyau
- Solution unique et optimale

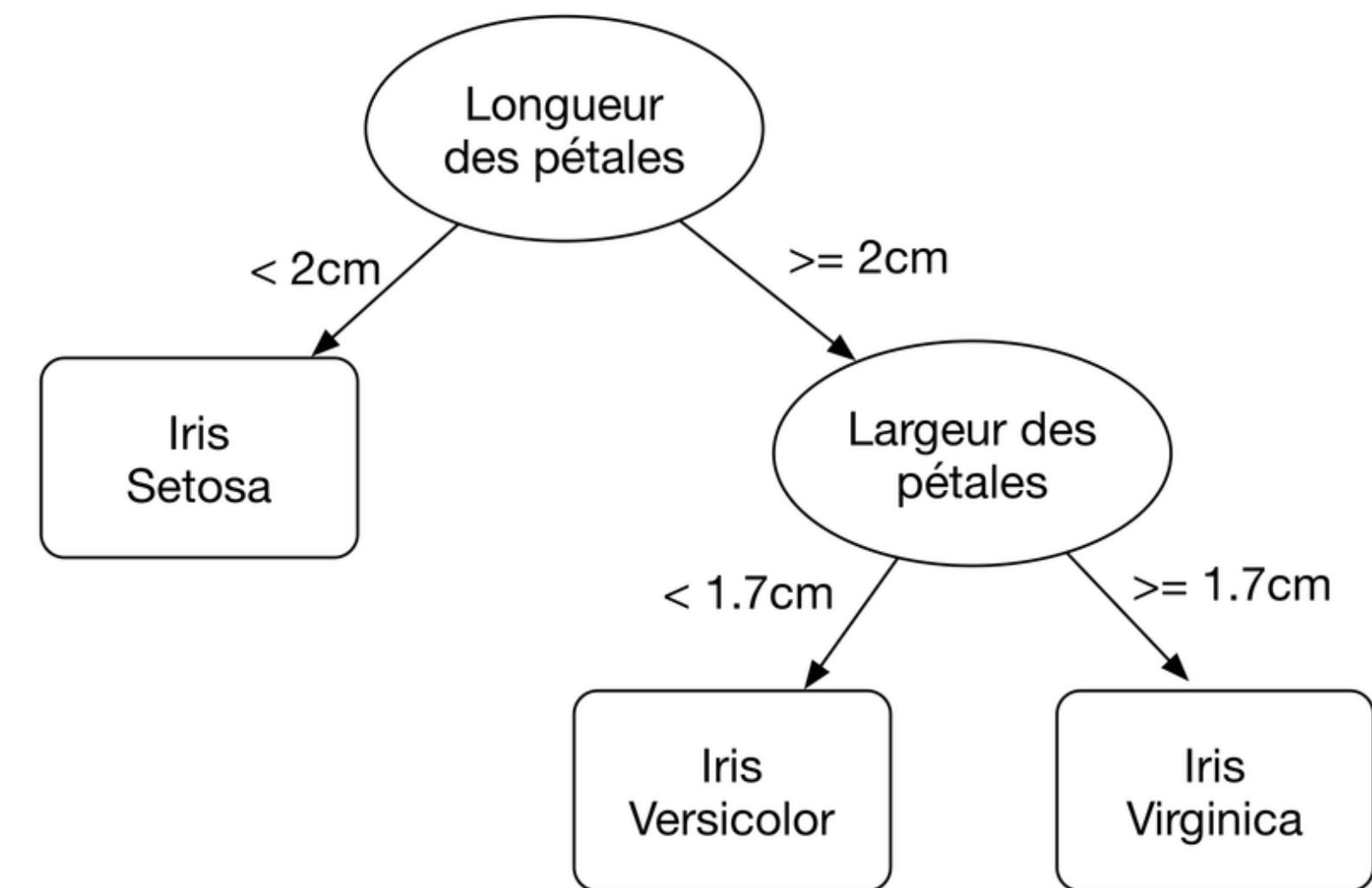
## Inconvénients

- Choix du noyau et des paramètres
- Complexité computationnelle
- Moins efficace sur de très grands ensembles de données
- Interprétabilité pour les kernels non linéaires
- Sensibilité aux paramètres
- Classification multiclasse

# **Decision Tree**

# Exemple de Classification - Apprentissage supervisé

- Le but général d'un arbre de décision est **d'expliquer une valeur à partir d'une série de variables discrètes ou continues.**
- On est donc dans un cas très classique de matrice X avec m observations et n variables, associée à un vecteur Y à expliquer. Les valeurs de Y peuvent être de deux sortes :
  - continues : on parle alors **d'arbre de régression**
  - qualitatives : on parlera **d'arbre de classification**



# Exemple de Classification - Apprentissage supervisé

- Le dataset d'exemple est **iris**.

	Longueur des sépales	Largeur des sépales	Longueur des pétales	Largeur des pétales	Cible
0	5.1	3.5	1.4	0.2	Iris-setosa
1	7.0	3.2	4.7	1.4	Iris-versicolor
2	6.3	3.3	6.0	2.5	Iris-virginica
3	...	...	...	...	...

Attribut cible : classes que l'on cherche à prédire.

Nouvelle plante      8.0      6.4      1.2      2.3      Quelle est sa classe ?

Dataset Iris (4 attributs, 150 instances)  
<https://archive.ics.uci.edu/ml/datasets/Iris>

# Arbre de décision

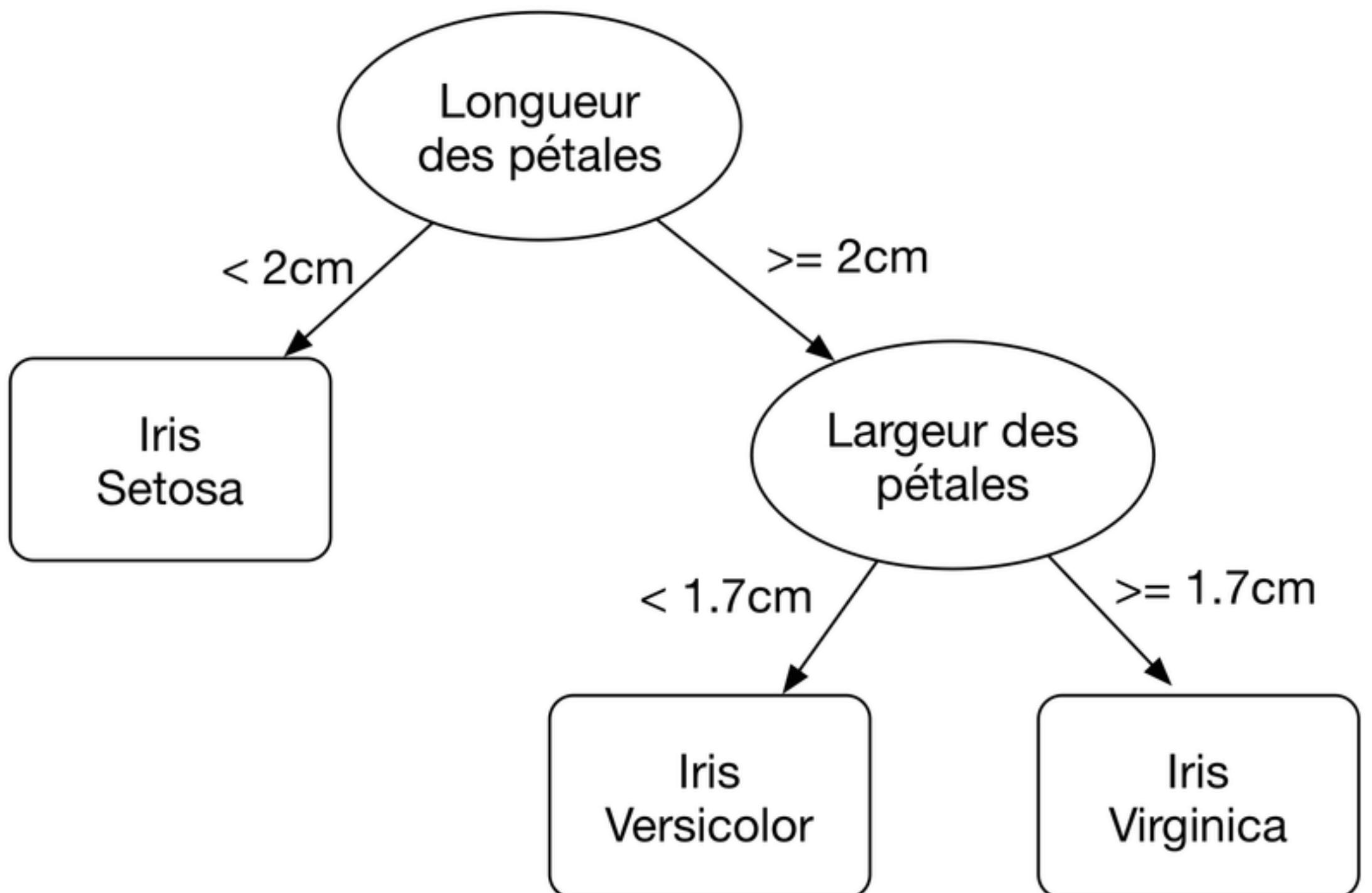
- Le dataset d'exemple est **iris**.

	Longueur des sépales	Largeur des sépales	Longueur des pétales	Largeur des pétales	Cible
0	5.1	3.5	1.4	0.2	Iris-setosa
1	7.0	3.2	4.7	1.4	Iris-versicolor
2	6.3	3.3	6.0	2.5	Iris-virginica
3	...	...	...	...	...

Nouvelle plante

8.0      6.4      1.2      2.3

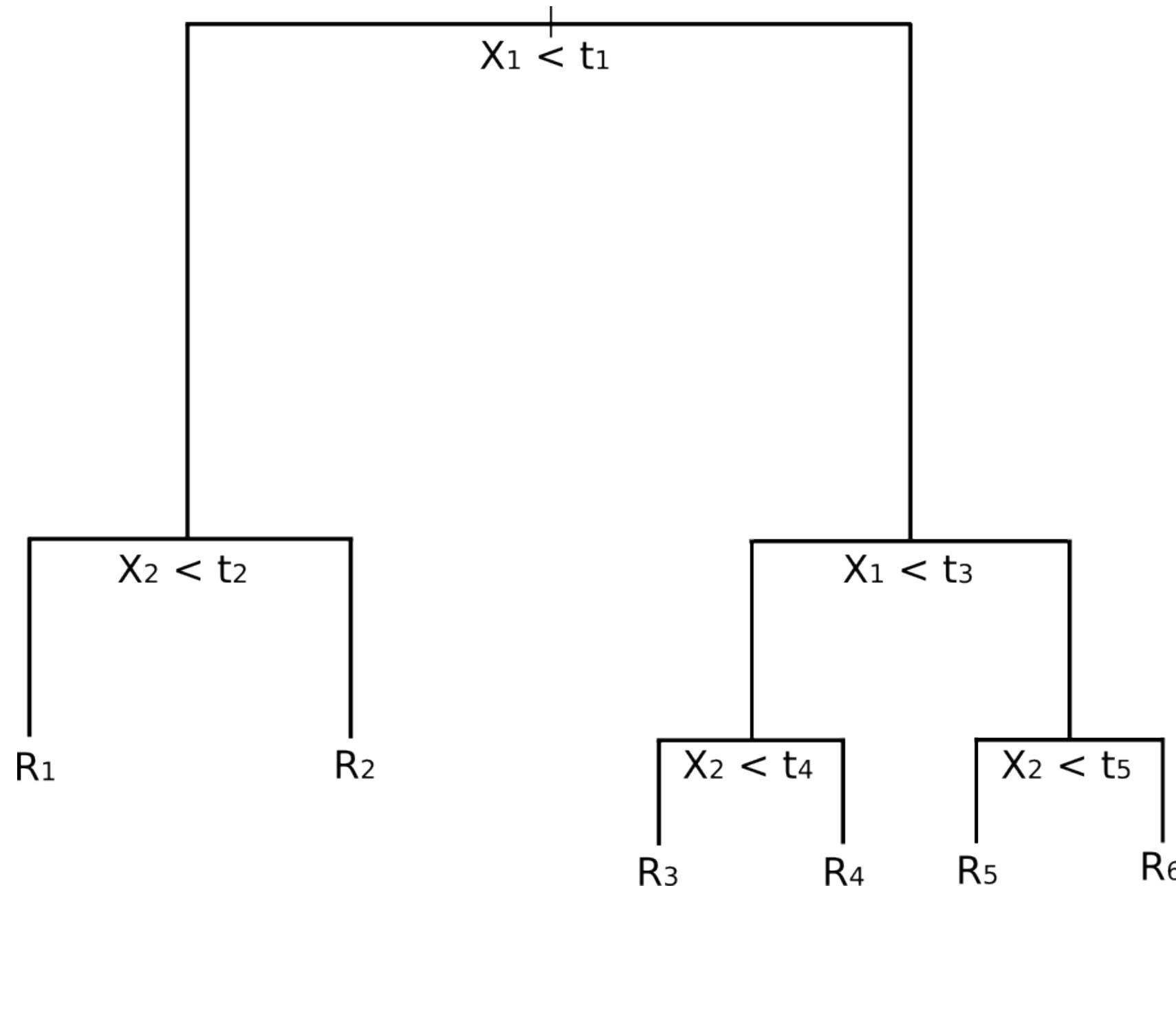
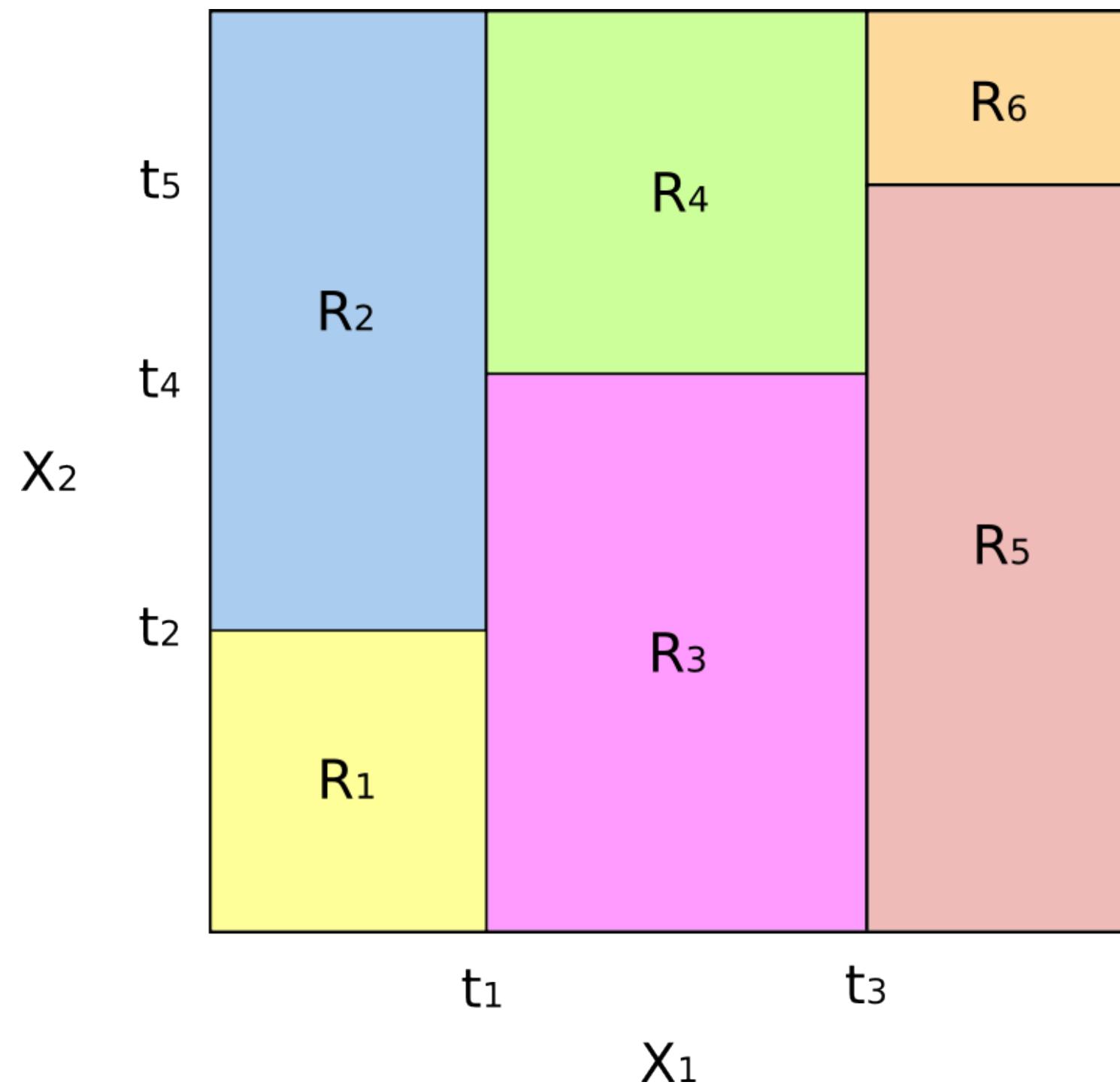
Quelle est sa classe ?



Exemple d'arbre de décision

# Construction d'un arbre : apprentissage par partitionnement

- **Objectif :** on veut construire des sous-groupes les plus « homogènes » du point de vue de la variable à prédire



- Le sous-groupe  $R_i$  est complètement pur : il ne contient que des éléments +.
- L'idée est de trouver le plus rapidement possible (avec le moins de variables) des groupes où  $P(Y=+) = 1$  ou proche.

# Construction d'un arbre : principales questions

- **Comment choisir la variable de division ?**

On doit en effet pouvoir définir l'arborescence de l'arbre en sélectionnant les variables, de la plus discriminante à la moins discriminante.

- **Comment traiter les variables continues ?**

Pour définir des noeuds à partir d'une variable continue, l'on doit savoir "couper" la variable continue, pour que ses valeurs inférieures et supérieures à cette coupe puissent caractériser des noeuds distincts.

- **Comment définir la taille de l'arbre ?**

L'objectif est de situer le niveau de noeuds optimal, pour trouver le juste équilibre entre sur-apprentissage et arbre trivial. Processus de post-élagage.

- Réponse fonction de l'algorithme CHAID (Chi-2), CART et C4.5 sur l'index (on parle aussi de critère de split) de Gini (indice de concentration) et sur la notion d'entropie (théorie de l'information).

# Construction avec l'indice Gini

Revision time	Break time	Success
High	Low	No
High	Medium	Yes
Medium	Low	No
Medium	High	Yes
Low	Low	No
Low	High	Yes
High	High	Yes
Medium	Medium	Yes
Low	Medium	No
High	Low	No

## I. Calcul de l'indice Gini sur l'ensemble complet:

$$Gini(D) = 1 - \sum_{i=1}^K P_i^2$$

$P_i = \frac{Nb_{classe_i}}{Nb_{lignes}}$  : probabilité d'avoir la classe i

L'indice Gini mesure donc l'impureté/l'incertitude dans un ensemble de données:

- Si il est nul, l'ensemble est pur (une seule classe)
- Plus il est élevé, plus l'ensemble est diversifié en classes

# Construction avec l'indice Gini

Revision time	Break time	Success
High	Low	No
High	Medium	Yes
Medium	Low	No
Medium	High	Yes
Low	Low	No
Low	High	Yes
High	High	Yes
Medium	Medium	Yes
Low	Medium	No
High	Low	No

I. Calcul de l'indice Gini sur l'ensemble complet:

$$P_{oui} = ???$$

$$P_{non} = ???$$

$$Gini(D) = 1 - \frac{5^2}{10} - \frac{5^2}{10} = 0.5$$

# Construction avec l'indice Gini

Revision time	Break time	Success
High	Low	No
High	Medium	Yes
Medium	Low	No
Medium	High	Yes
Low	Low	No
Low	High	Yes
High	High	Yes
Medium	Medium	Yes
Low	Medium	No
High	Low	No

2. Sélection de l'attribut pour la première division:

I. Evaluation de l'indice Gini pondéré pour chaque attribut:

$$Gini_{pondere} = \sum_{j=1}^k \frac{N_j}{N} Gini(j)$$

avec

$k$  : le nombre de noeuds enfants résultant du split

$N_j$  : le nombre d'échantillons dans le noeud enfant  $j$

$N$  : le nombre total d'échantillons dans le noeud parent

$Gini(j)$  : l'indice Gini du noeud enfant  $j$

2. Sélection de l'attribut avec l'indice le plus faible

# Construction avec l'indice Gini

Revision time	Break time	Success
High	Low	No
High	Medium	Yes
Medium	Low	No
Medium	High	Yes
Low	Low	No
Low	High	Yes
High	High	Yes
Medium	Medium	Yes
Low	Medium	No
High	Low	No

## 2. Sélection de l'attribut pour la première division

Pour l'attribution « Revision time »

$$\text{High (4 éléments)} : Gini_{High} = 1 - \frac{2^2}{4} - \frac{2^2}{4} = 0.5$$

$$\text{Medium (3 éléments)} : Gini_{Medium} = 1 - \frac{2^2}{3} - \frac{1^2}{3} = 0.444$$

$$\text{Low (3 éléments)} : Gini_{Low} = 1 - \frac{1^2}{3} - \frac{2^2}{3} = 0.444$$

$$Gini_{pondere}(RevisionTime) = \frac{4}{10} \times 0.5 + \frac{3}{10} \times 0.444 + \frac{3}{10} \times 0.444 \\ = 0.4664$$

# Construction avec l'indice Gini

Revision time	Break time	Success
High	Low	No
High	Medium	Yes
Medium	Low	No
Medium	High	Yes
Low	Low	No
Low	High	Yes
High	High	Yes
Medium	Medium	Yes
Low	Medium	No
High	Low	No

## 2. Sélection de l'attribut pour la première division

Pour l'attribution « Break time »

$$\text{High (3 éléments)} : Gini_{High} = 1 - \frac{3^2}{3} - \frac{0^2}{3} = 0.0$$

$$\text{Medium (3 éléments)} : Gini_{Medium} = 1 - \frac{2^2}{3} - \frac{1^2}{3} = 0.444$$

$$\text{Low (4 éléments)} : Gini_{Low} = 1 - \frac{0^2}{4} - \frac{4^2}{4} = 0.0$$

$$Gini_{pondere}(BreakTime) = \frac{3}{10} \times 0.0 + \frac{3}{10} \times 0.444 + \frac{4}{10} \times 0.0 \\ = 0.1332$$

# Construction avec l'indice Gini

Revision time	Break time	Success
High	Low	No
High	Medium	Yes
Medium	Low	No
Medium	High	Yes
Low	Low	No
Low	High	Yes
High	High	Yes
Medium	Medium	Yes
Low	Medium	No
High	Low	No

2. Sélection de l'attribut pour la première division

Comparaison des indices Gini:

$$Gini_{pondere}(RevisionTime) = 0.4664$$

$$Gini_{pondere}(BreakTime) = 0.1332$$

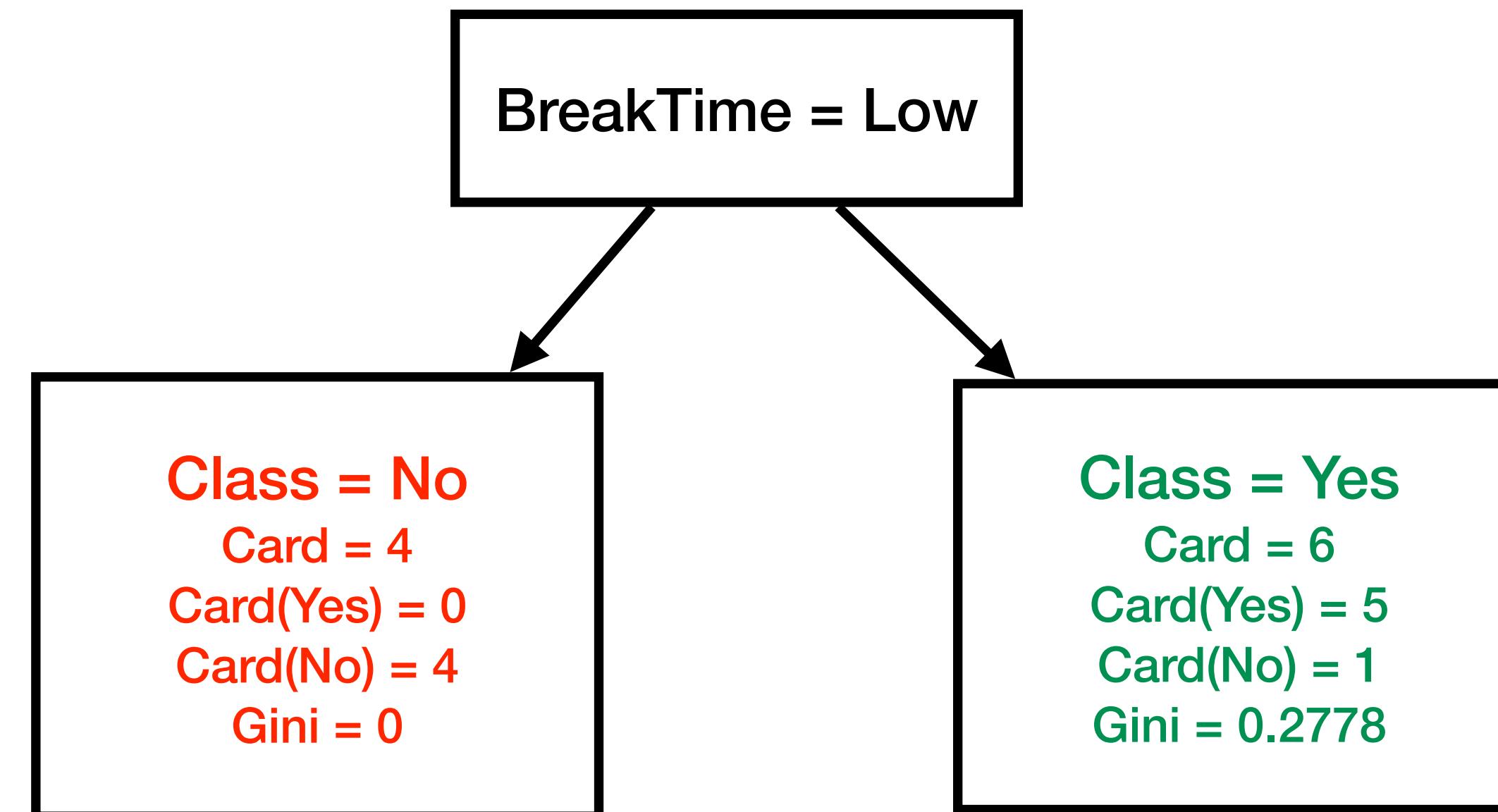
On sélectionne l'attribut *BreakTime* qui a l'indice le plus faible

# Construction avec l'indice Gini

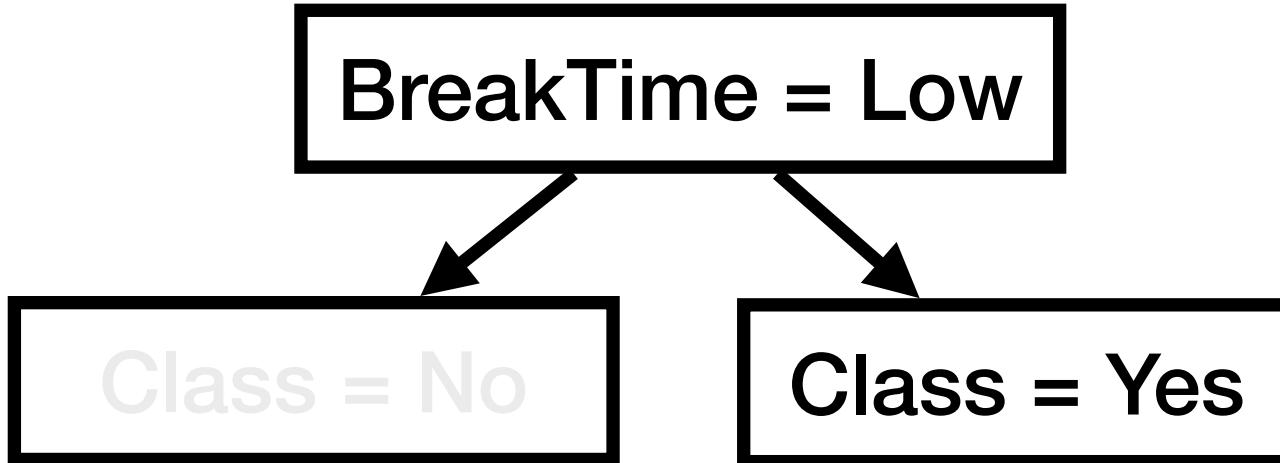
Revision time	Break time	Success
High	Low	No
High	Medium	Yes
Medium	Low	No
Medium	High	Yes
Low	Low	No
Low	High	Yes
High	High	Yes
Medium	Medium	Yes
Low	Medium	No
High	Low	No

2. Sélection de l'attribut pour la première division

Attribution des classes par probabilité de valeur



# Construction avec l'indice Gini



Revision time	Break time	Success
High	Low	No
High	Medium	Yes
Medium	Low	No
Medium	High	Yes
Low	Low	No
Low	High	Yes
High	High	Yes
Medium	Medium	Yes
Low	Medium	No
High	Low	No

2. On recommence sur chacune des feuilles non pures

Calcul de l'indice Gini pour RevisionTime

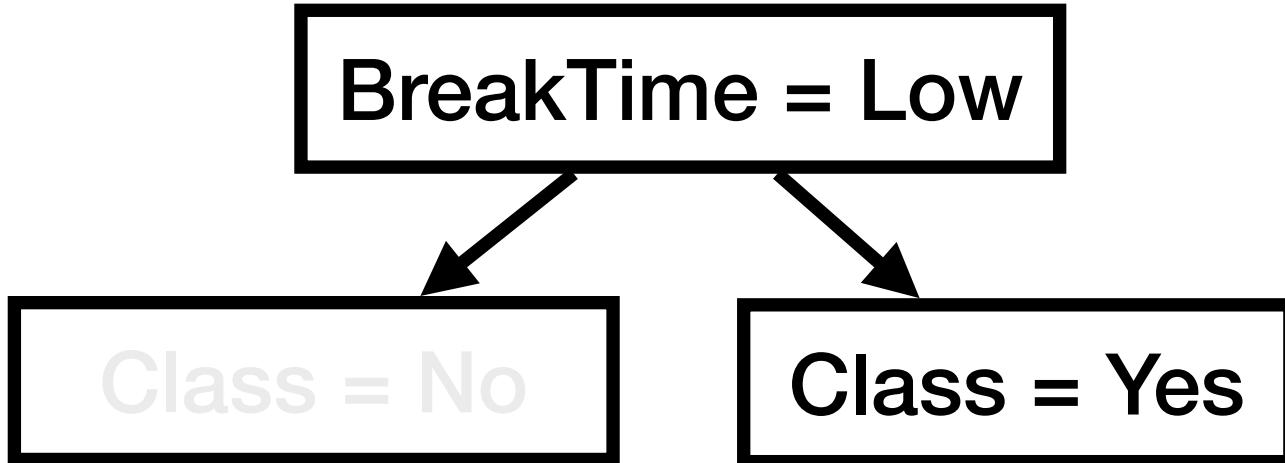
$$Gini(High) = 1 - \frac{2^2}{2} - \frac{0^2}{2} = 0.0$$

$$Gini(Medium) = 1 - \frac{2^2}{2} - \frac{0^2}{2} = 0.0$$

$$Gini(Low) = 1 - \frac{1^2}{2} - \frac{1^2}{2} = 0.5$$

$$Gini_{pondere}(RevisionTime) = \frac{2}{6} \times 0.0 + \frac{2}{6} \times 0.0 + \frac{2}{6} \times 0.5 \\ = 0.16667$$

# Construction avec l'indice Gini



Revision time	Break time	Success
High	Low	No
High	Medium	Yes
Medium	Low	No
Medium	High	Yes
Low	Low	No
Low	High	Yes
High	High	Yes
Medium	Medium	Yes
Low	Medium	No
High	Low	No

2. On recommence sur chacune des feuilles non pures

Calcul de l'indice Gini pour BreakTime

$$Gini(High) = 1 - \frac{3^2}{3} - \frac{0^2}{3} = 0.0$$

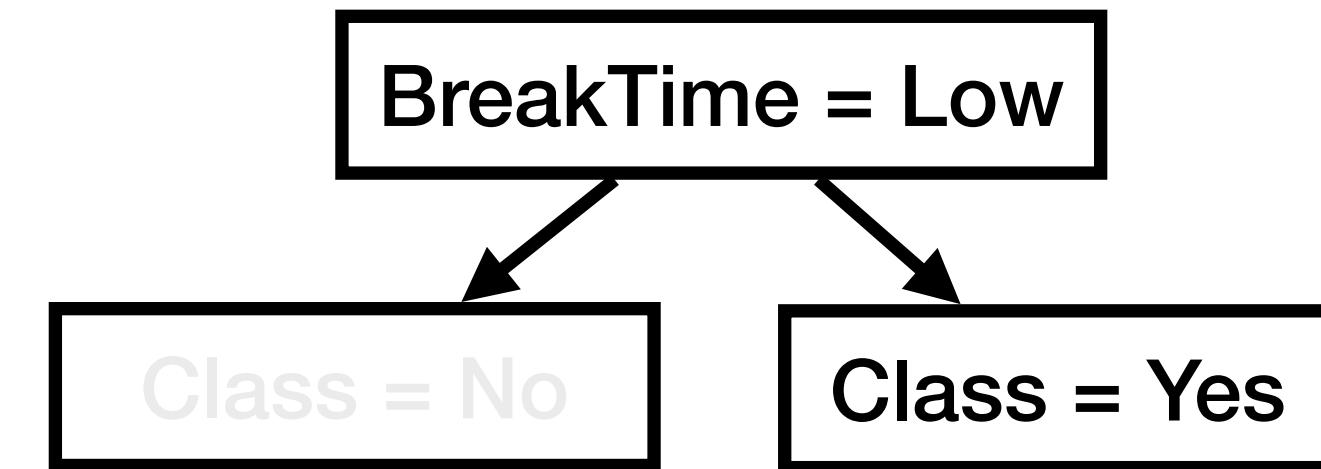
$$Gini(Medium) = 1 - \frac{2^2}{3} - \frac{1^2}{3} = 0.444$$

$$Gini(Low) = 0.0$$

$$Gini_{pondere}(BreakTime) = \frac{3}{6} \times 0.0 + \frac{3}{6} \times 0.444 + \frac{0}{6} \times 0.0 \\ = 0.2222$$

# Construction avec l'indice Gini

Revision time	Break time	Success
High	Low	No
High	Medium	Yes
Medium	Low	No
Medium	High	Yes
Low	Low	No
Low	High	Yes
High	High	Yes
Medium	Medium	Yes
Low	Medium	No
High	Low	No



2. Sélection de l'attribut pour la première division

Comparaison des indices Gini:

$$Gini_{pondere}(RevisionTime) = 0.16667$$

$$Gini_{pondere}(BreakTime) = 0.2222$$

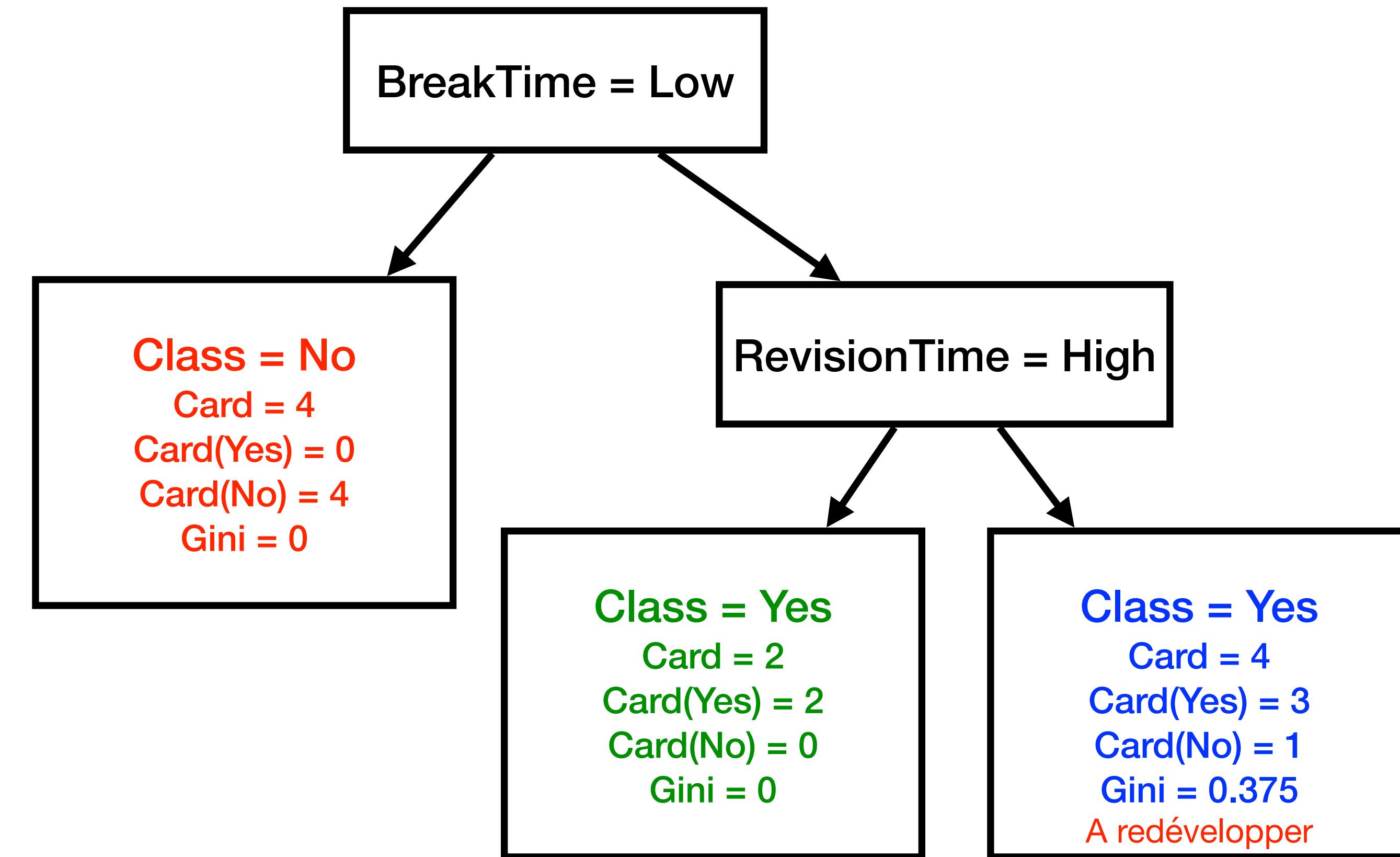
On sélectionne l'attribut *RevisionTime* qui a l'indice le plus faible

# Construction avec l'indice Gini

Revision time	Break time	Success
High	Low	No
High	Medium	Yes
Medium	Low	No
Medium	High	Yes
Low	Low	No
Low	High	Yes
High	High	Yes
Medium	Medium	Yes
Low	Medium	No
High	Low	No

2. Sélection de l'attribut pour la deuxième division

Attribution des classes par probabilité de valeur



# Arbre de décision: Avantages et inconvénients

## Avantages

- Facilité d'interprétation
- Pas de normalisation nécessaire
- Gestion des valeurs manquantes
- Flexibilité avec les types de données
- Robustesse aux outliers

## Inconvénients

- Forte tendance à sur-apprendre
- Instabilité à la variabilité des données
- Complexité computationnelle
- Sensible au bruit

# **Random Forest**

# Introduction aux méthodes ensemblistes

- **Principe**
  - si un médecin (et un seul) vous annonce que vous avez **probablement** une maladie grave qui nécessite une intervention chirurgicale, que ferez-vous ?

# Introduction aux méthodes ensemblistes

- **Principe**
  - si un médecin (et un seul) vous annonce que vous avez **probablement** une maladie grave qui nécessite une intervention chirurgicale, que ferez-vous ?
  - Fort probable : **demander à d'autres personnes/experts leur avis.**

# Introduction aux méthodes ensemblistes

- **Principe**
  - si un médecin (et un seul) vous annonce que vous avez **probablement** une maladie grave qui nécessite une intervention chirurgicale, que ferez-vous ?
  - Fort probable : **demander à d'autres personnes/experts leur avis.**
- Méthodes ensemblistes fonctionnent de même :
  - Construction de plusieurs estimateurs (même de qualité moyenne), avec une vision parcellaire du problème
  - La décision finale vient de l'agrégation des avis de tous ces estimateurs.

# Random Forest ??

- **Forest**
  - On utilise un ensemble d'arbres de décision, formant une forêt...
- **Random**
  - Pour éviter d'avoir plusieurs fois le même arbre, ils sont entraînés sur un sous-ensemble d'observations (*tree bagging*) et de features tirés au hasard (*feature sampling*).
    - *Random forest* = *tree bagging* + *feature sampling*
- **Fonction d'agrégation**
  - En classification, **vote**
  - En regression, **moyenne**

# Exemple de construction d'une forêt aléatoire

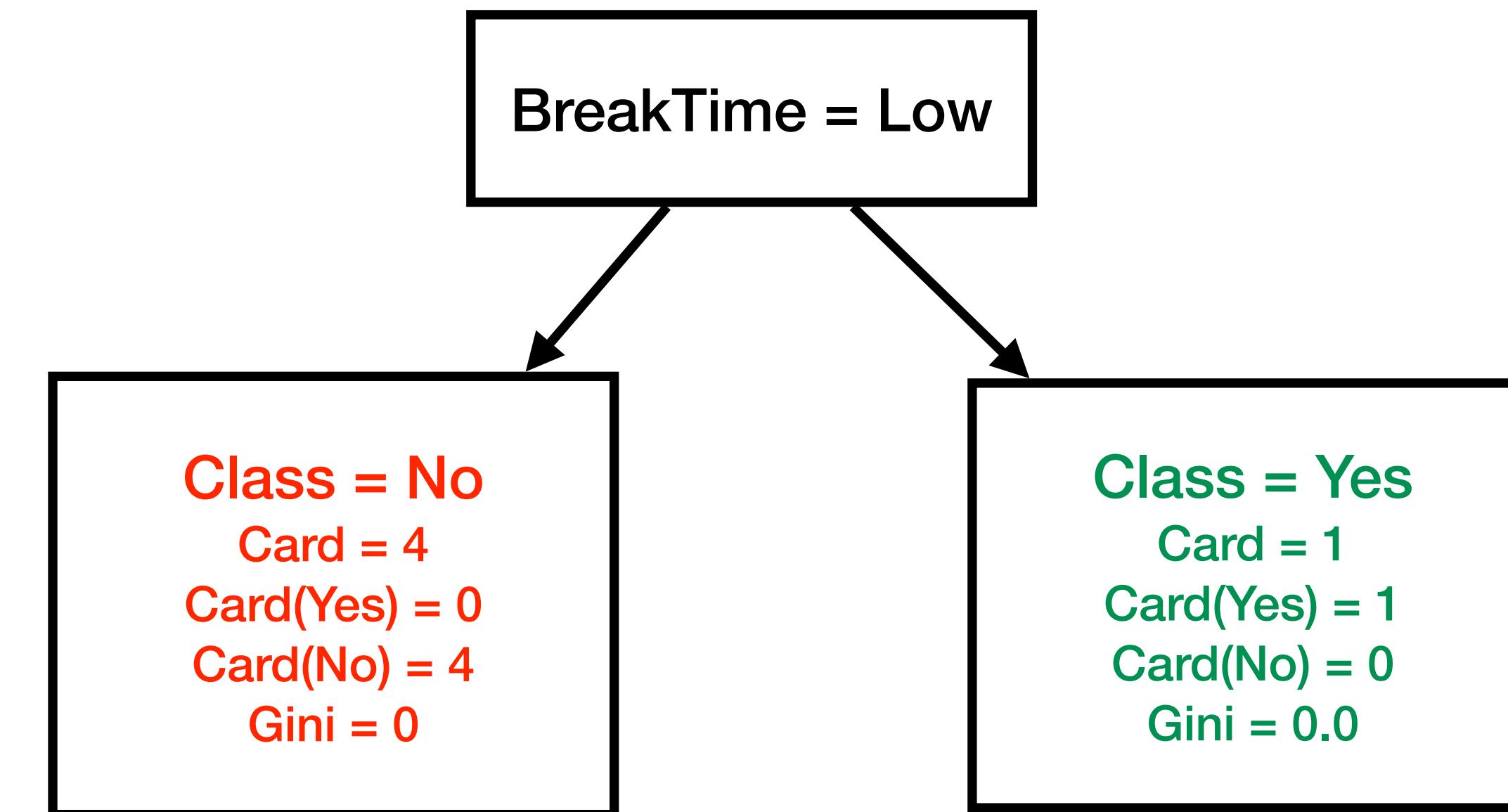
Revision time	Break time	Success
High	Low	No
High	Medium	Yes
Medium	Low	No
Medium	High	Yes
Low	Low	No
Low	High	Yes
High	High	Yes
Medium	Medium	Yes
Low	Medium	No
High	Low	No

- Sélection d'un premier sous-ensemble aléatoire

# Exemple de construction d'une forêt aléatoire

Revision time	Break time	Success
High	Low	No
High	Medium	Yes
Medium	Low	No
Medium	High	Yes
Low	Low	No
Low	High	Yes
High	High	Yes
Medium	Medium	Yes
Low	Medium	No
High	Low	No

- Sélection d'un premier sous-ensemble aléatoire
- Construction du premier arbre



# Exemple de construction d'une forêt aléatoire

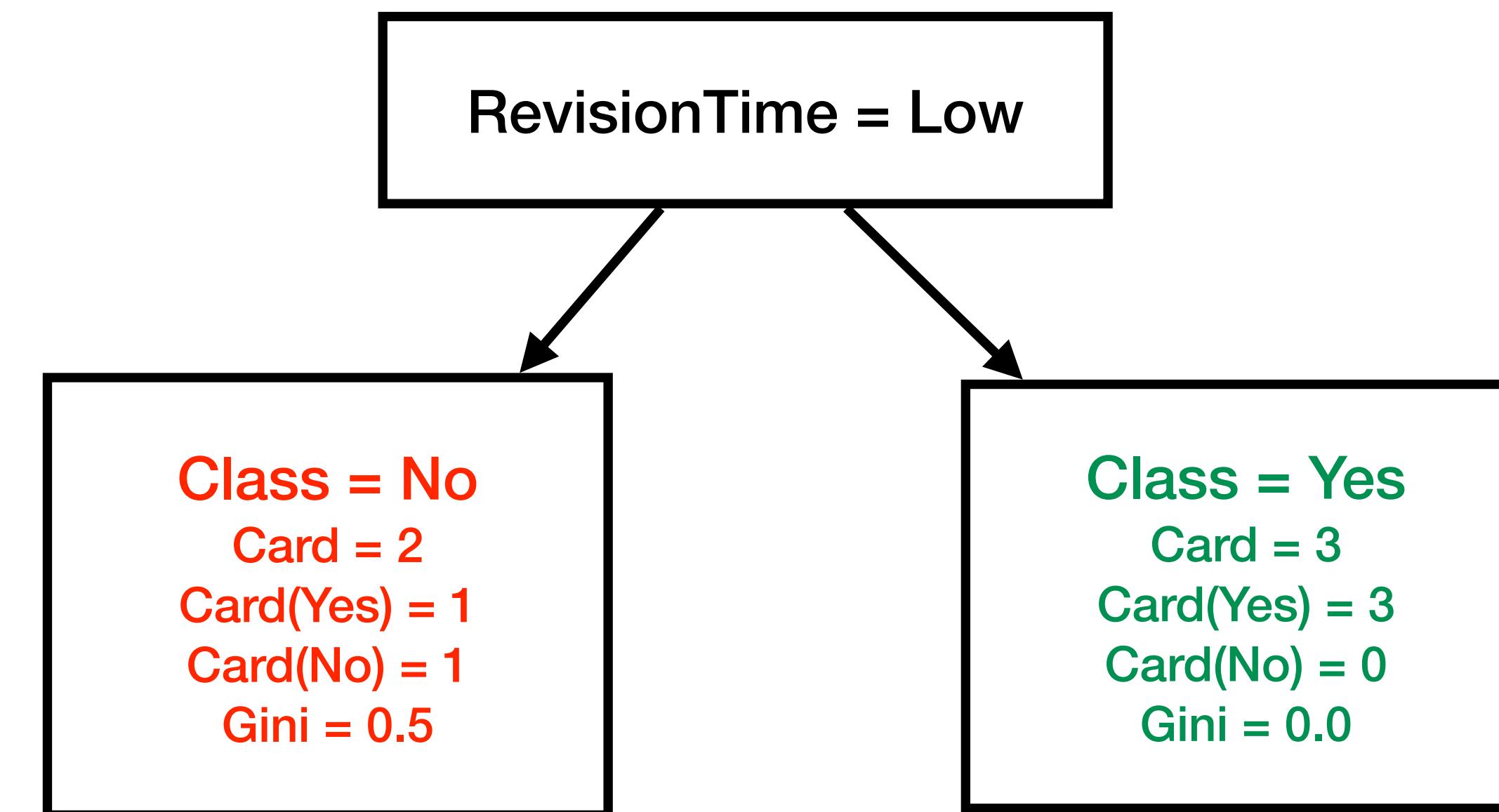
Revision time	Break time	Success
High	Low	No
High	Medium	Yes
Medium	Low	No
Medium	High	Yes
Low	Low	No
Low	High	Yes
High	High	Yes
Medium	Medium	Yes
Low	Medium	No
High	Low	No

- Sélection d'un premier sous-ensemble aléatoire
- Construction du premier arbre

# Exemple de construction d'une forêt aléatoire

Revision time	Break time	Success
High	Low	No
High	Medium	Yes
Medium	Low	No
Medium	High	Yes
Low	Low	No
Low	High	Yes
High	High	Yes
Medium	Medium	Yes
Low	Medium	No
High	Low	No

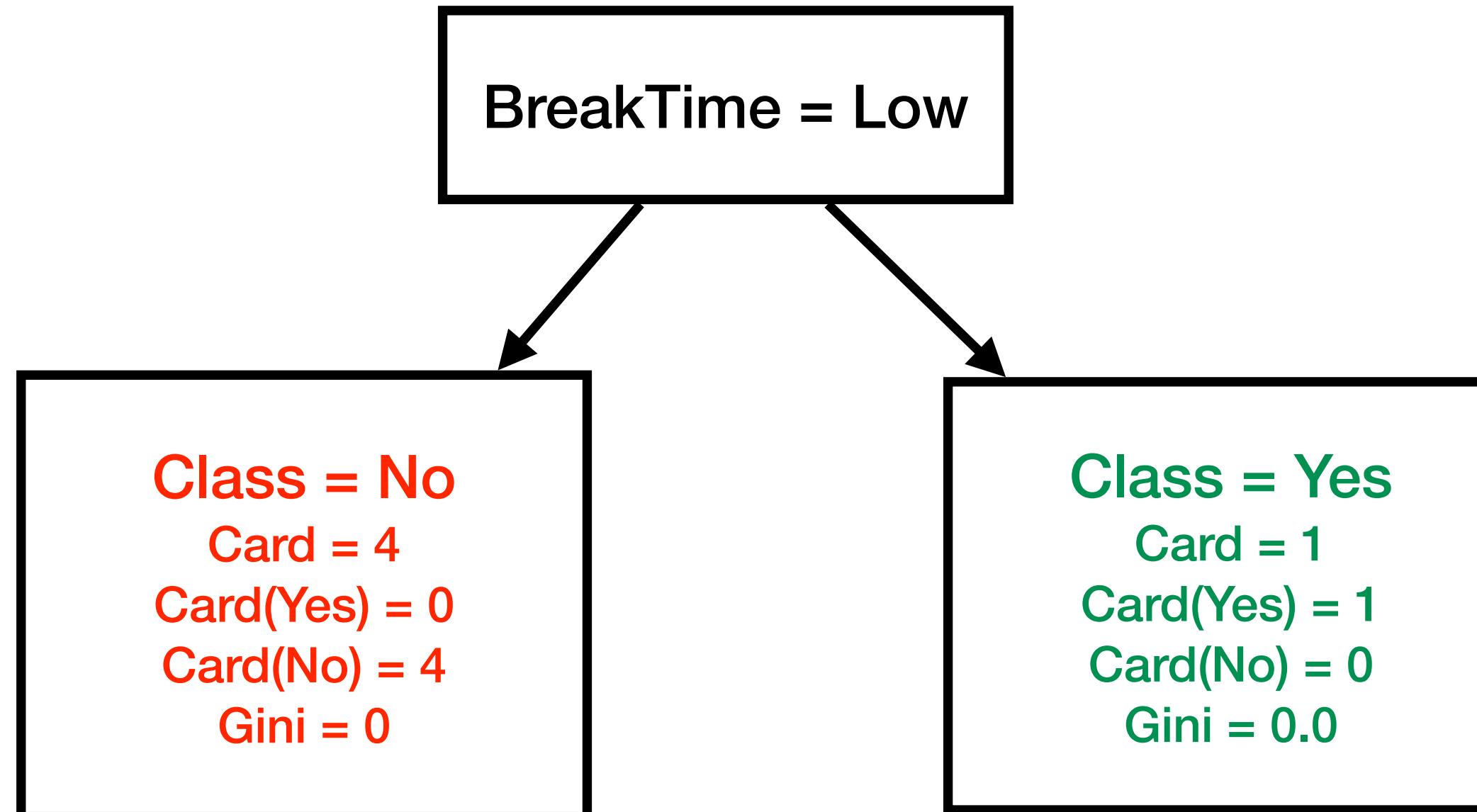
- Sélection d'un premier sous-ensemble aléatoire
- Construction du premier arbre
- Sélection d'un second sous-ensemble aléatoire
- Construction du second arbre



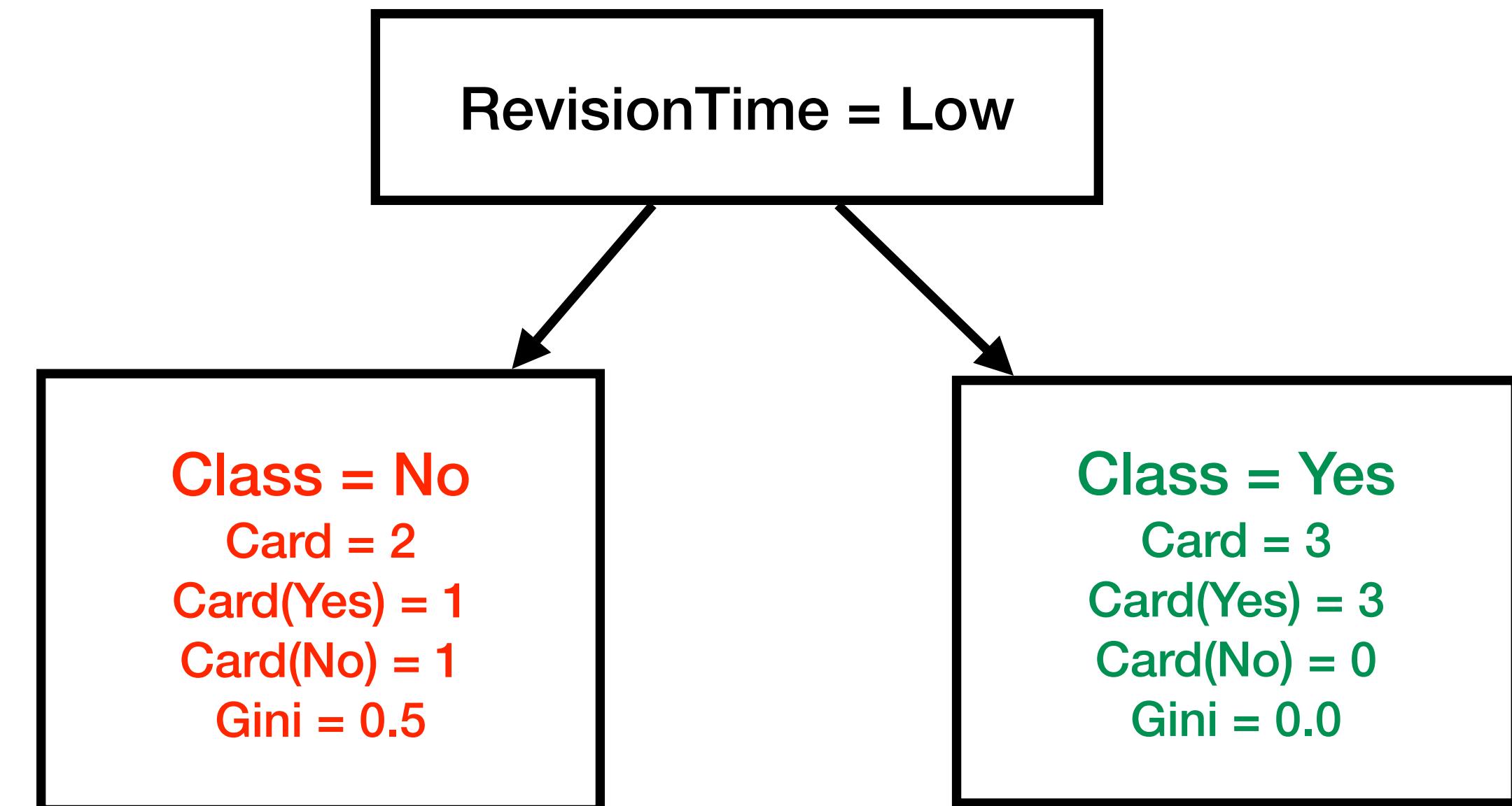
# Exemple de construction d'une forêt aléatoire

- Prédiction: vote entre les différents arbres

- Soit un nouvel étudiant  
RevisionTime = Medium  
BreakTime = Medium



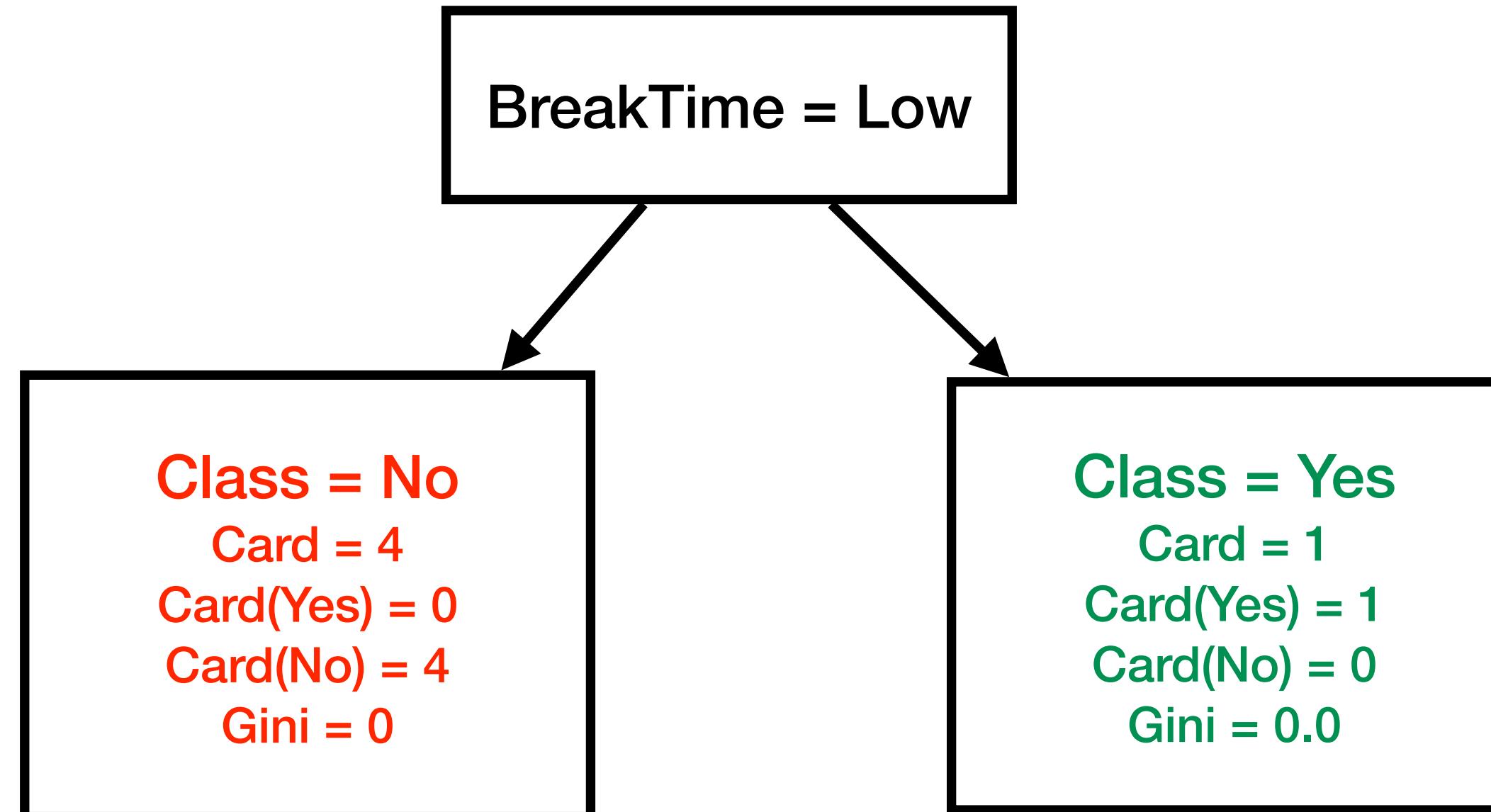
On est d'accord, on classifie YES!



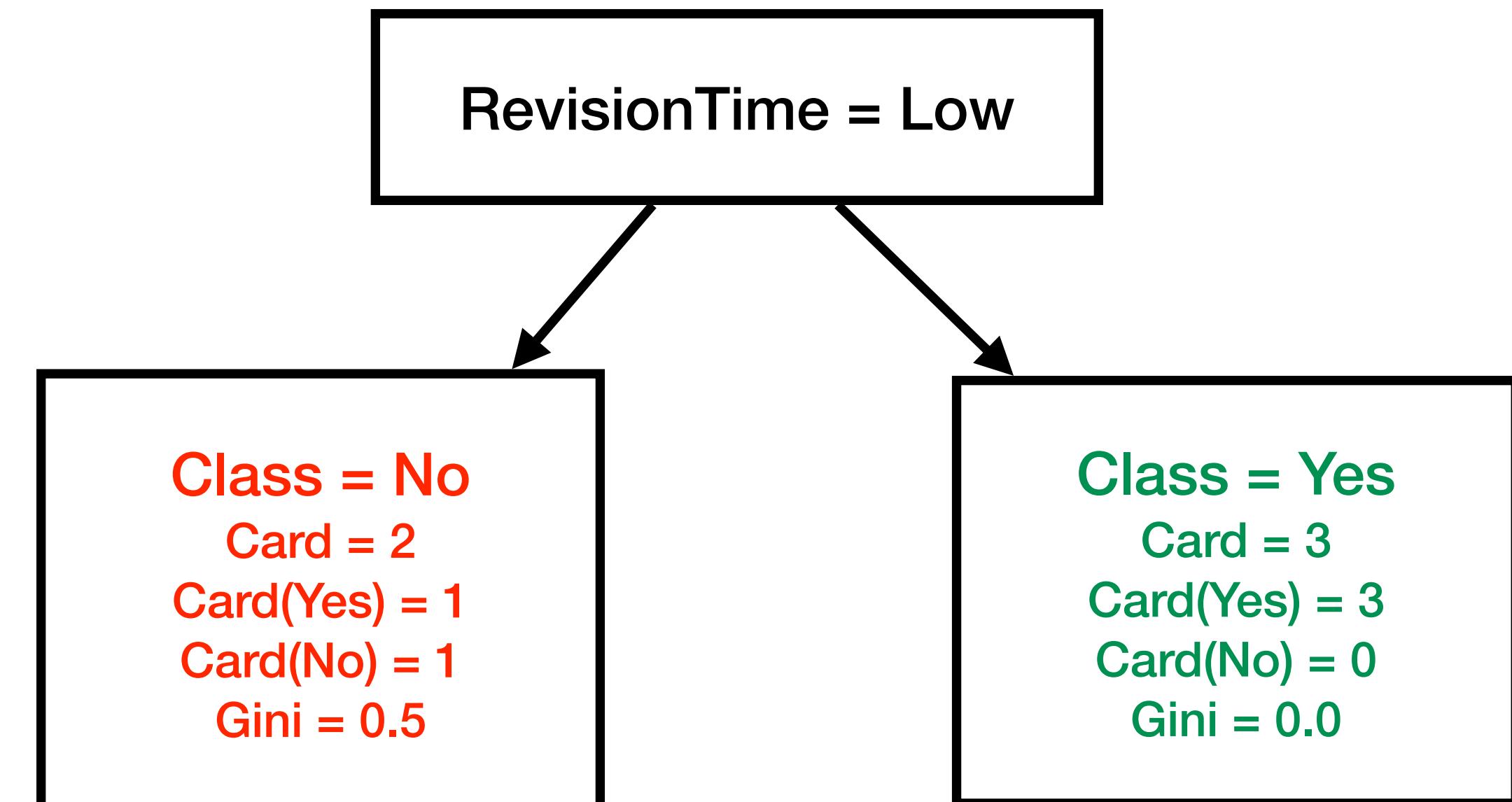
# Exemple de construction d'une forêt aléatoire

- Prédiction: vote entre les différents arbres

- Soit un nouvel étudiant  
RevisionTime = High  
BreakTime = Low



On n'est pas d'accord, on choisit au hasard!



# Random Forest : à retenir

- Apprentissage supervisé : régression ou classification
- Le *random forest* construit plusieurs arbres de décision indépendants.
- Chaque arbre construit dispose d'une vision parcellaire du problème, conditionnée par un double tirage aléatoire :
  - *tree bagging* : tirage aléatoire avec remplacement sur les lignes (les observations) ;
  - *feature sampling* : tirage aléatoire sur les colonnes (les variables).
- La construction de chaque arbre se base (en classification) sur l'un des deux critères :
  - *Gini*, qui se focalise sur la séparation de la classe la plus représentée ;
  - *entropie*, qui vise à maximiser le gain d'information à chaque étape de réalisation de l'algorithme.
- Les arbres créés sont ensuite assemblés. La prévision pour de nouvelles données est une moyenne (en régression) ou **un vote (en classification)**.

# Random forest: Avantages et inconvénients

## Avantages

- Sur-apprentissage réduit par rapport aux arbres de décision
- Meilleure précision
- Gestion des valeurs manquantes
- Flexibilité avec les types de données
- Capable de gérer de très grands volumes de données

## Inconvénients

- Complexité computationnelle
- Non interprétable
- Beaucoup d'hyper-paramètres
- Sur-apprentissage potentiel

# **Optimisation paramétrique**

# Introduction

- Les modèles de machine learning nécessitent le réglage d'un certain nombre de paramètres afin :
  - D'augmenter l'efficacité de l'apprentissage (score / performance)
  - Réduire le temps de l'apprentissage
- Ces paramètres sont appelés des hyperparamètres, paramètres qui contrôlent / orientent le processus d'apprentissage.

# Amélioration d'un modèle

- Améliorer un modèle, c'est rechercher le jeu de paramètres qui obtiendra la meilleure moyenne sur l'ensemble des validations croisées.
  - A la main...
  - Cela peut être long si plusieurs paramètres et de nombreuses valeurs associées.
  - SkLearn : validation\_curve
- Automatiquement par l'algorithme Grid Search
  - Recherche exhaustive de toutes les combinaisons de paramètres
  - SkLearn : GridSearchCV

# Optimisation d'hyper-paramètres

- Grid Search (GridSearchCV)
  - Recherche exhaustive du meilleur jeu de paramètres
- Random Search (RandomizedSearchCV)
  - Recherche aléatoire du meilleur jeu de paramètres
  - Le nombre d'itérations est donné, ainsi le temps de calcul ne dépend pas des paramètres
  - Exploration plus diversifiée
- Ces 2 méthodes sont naïves
  - Elles n'exploitent pas les résultats pour améliorer leur exploration des paramètres

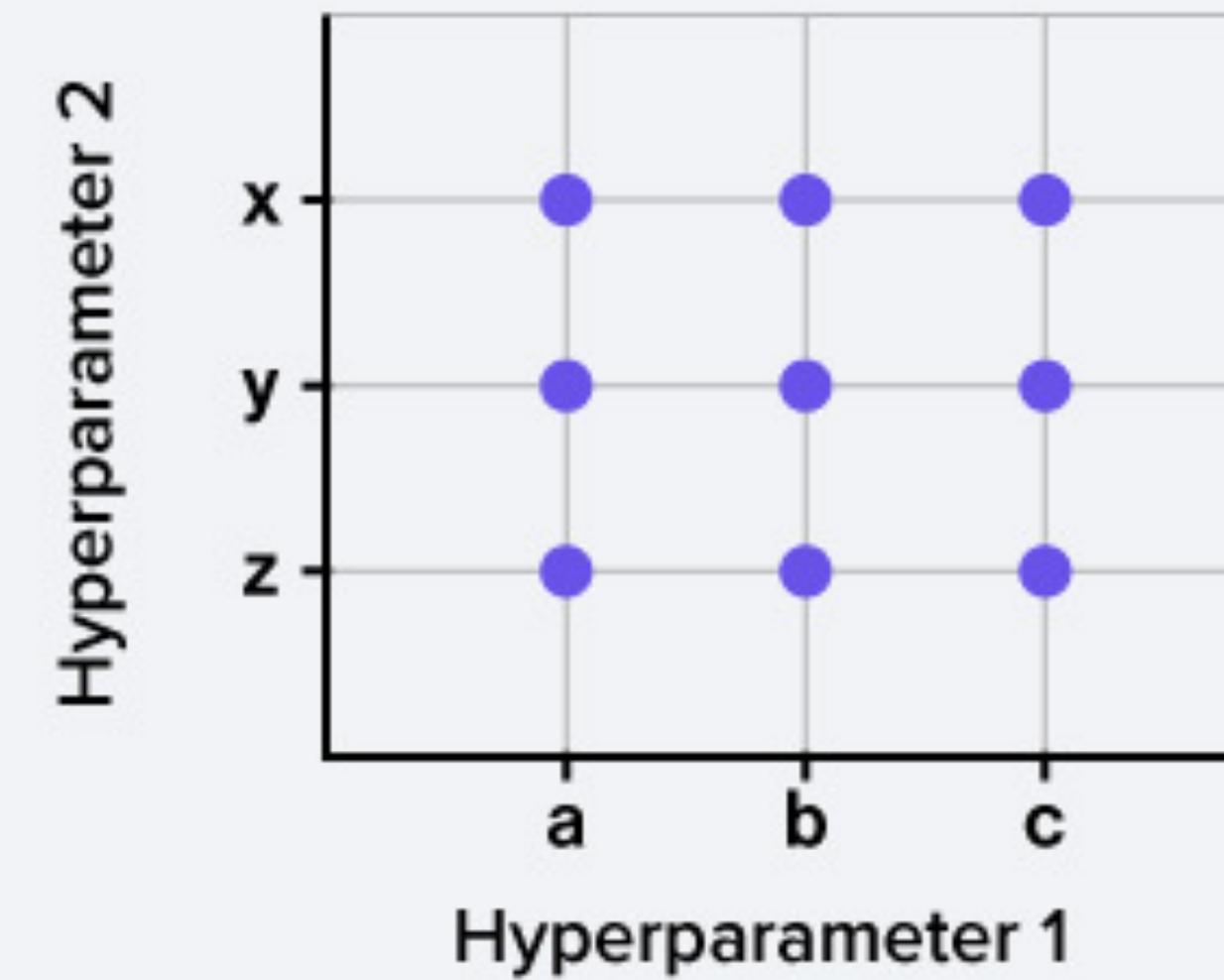
# Grid vs Random search

## Grid Search

Hyperparameter\_One = [ a, b, c ]

Hyperparameter\_Two = [ x, y, z ]

Hyperparameter\_X = [ i, j, k ]



## Random Search

Hyperparameter\_One = random.num (range)

Hyperparameter\_Two = random.num (range)

Hyperparameter\_X = random.num (range)

