

# Introduction au Machine Learning en Python

Master SIREN – Parcours IRN

(1/3)

UE: Econométrie et science des données

---

**Mustapha Sali**

Maître de Conférences HDR - Université Paris Dauphine

[mustapha.Sali@dauphine.psl.eu](mailto:mustapha.Sali@dauphine.psl.eu)

# Plan

## Introduction au ML

- Introduction
- Algorithmes et tâches en ML
- Algorithmes supervisés pour la régression
- Evaluation, test et généralisation des modèles
- Outils
- Développements et enjeux actuels

## Introduction à Python

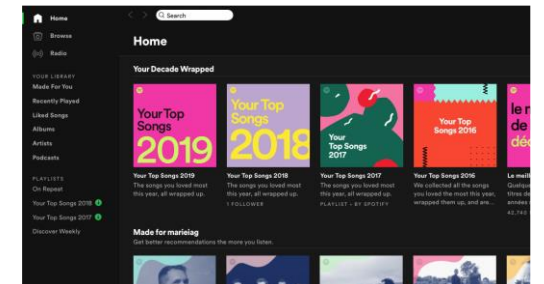
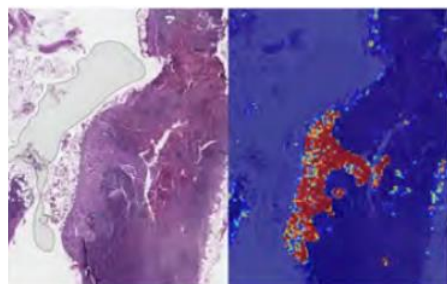
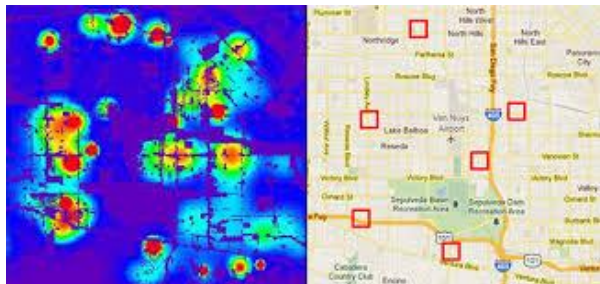
- Introduction
- Valeurs, Types, Variables
- Instructions conditionnelles
- Boucle
- Types de données séquentielles
- Fonctions

# Introduction au ML

# Introduction

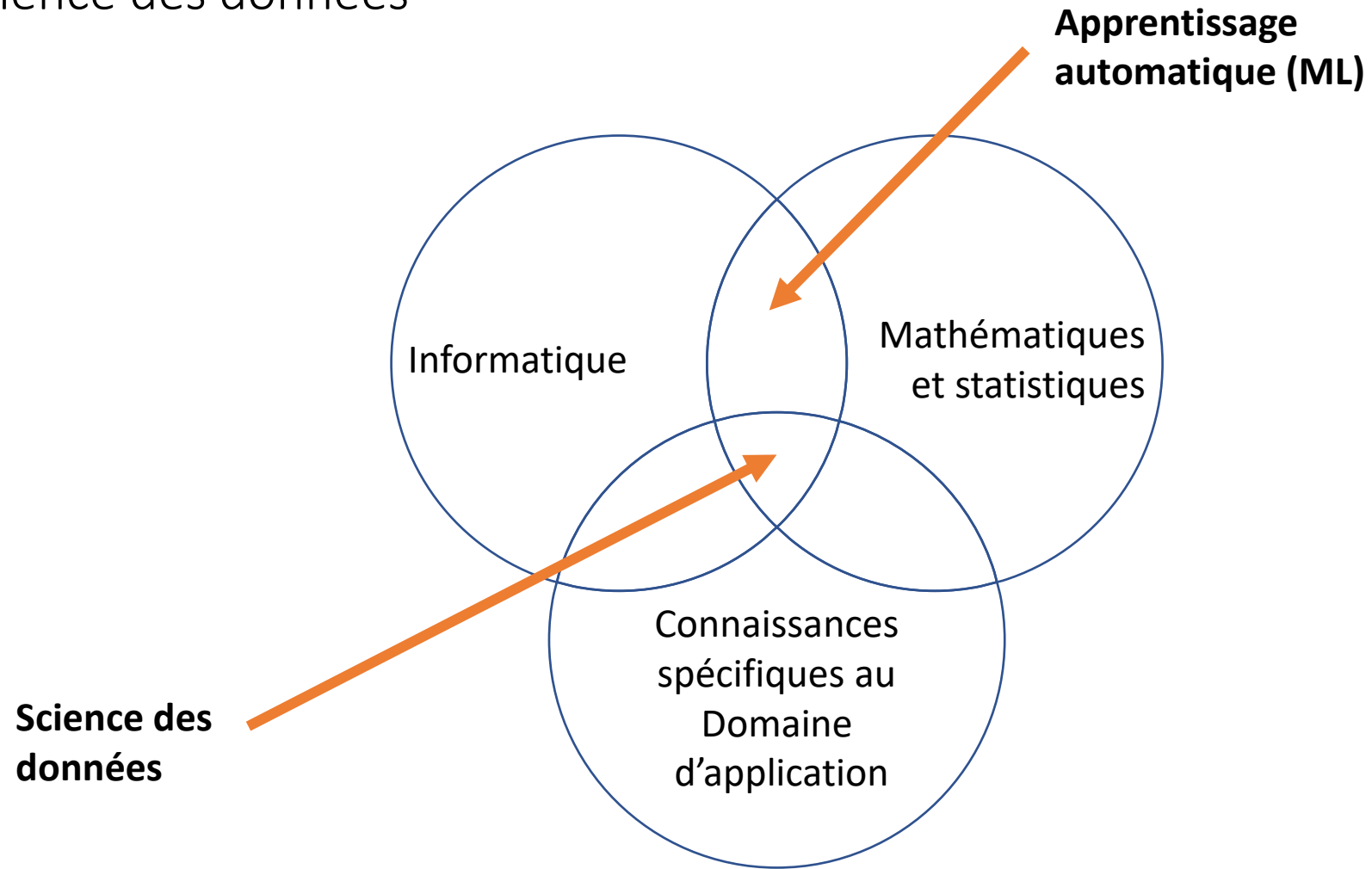
## Définition

l'apprentissage automatique (Machine Learning) fait référence à l'ensemble de techniques visant à extraire des connaissances à partir de grandes masses de données dans le but de faire des prédictions.



# Introduction

ML et science des données



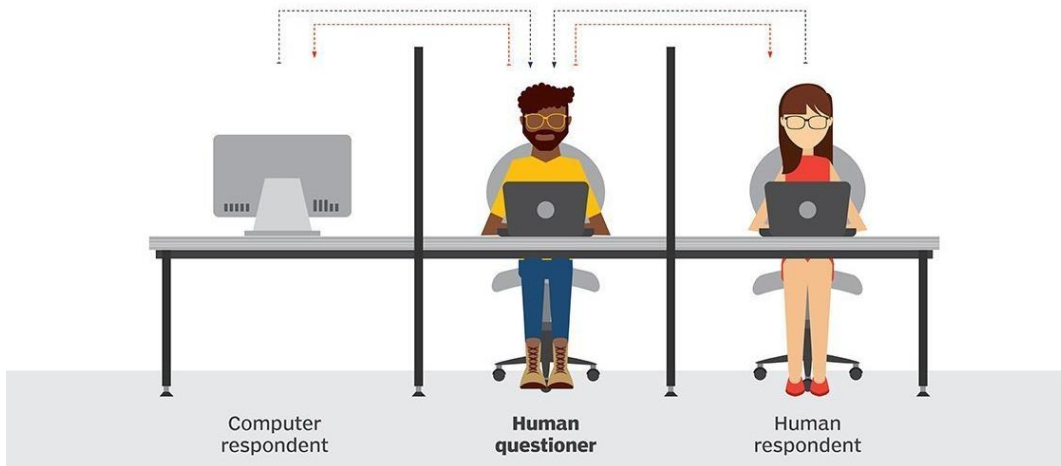
# Introduction

Des origines à nos jours

## Turing test

During the Turing test, the human questioner asks a series of questions to both respondents. After the specified time, the questioner tries to decide which terminal is operated by the human respondent and which terminal is operated by the computer.

■ QUESTION TO RESPONDENTS ■ ANSWERS TO QUESTIONER



# Introduction

## Des origines à nos jours



En 1954, (Samuel 1954) publiait un article dans la revue *IBM Journal of Research and Development* où il était question de développer un algorithme capable non seulement d'intégrer les règles du jeu

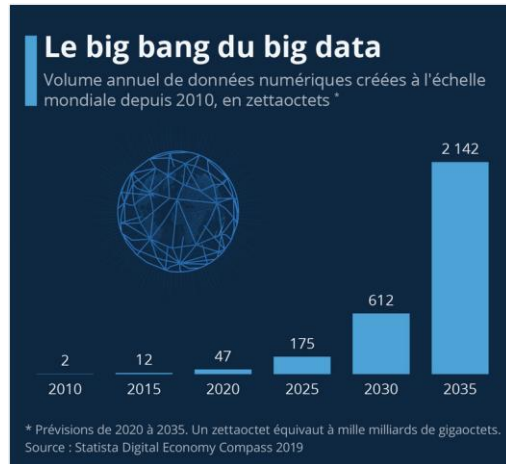
de dames mais aussi de rivaliser dans ses choix de déplacement contre un adversaire humain. L'algorithme a été implémenté sur une machine de type IBM 704 et l'auteur notait ceci à ce sujet : « *We have at our command computers with adequate data-handling ability and with sufficient computational speed to make use of machine-learning techniques, but our knowledge of the basic principles of these techniques is still rudimentary* ».





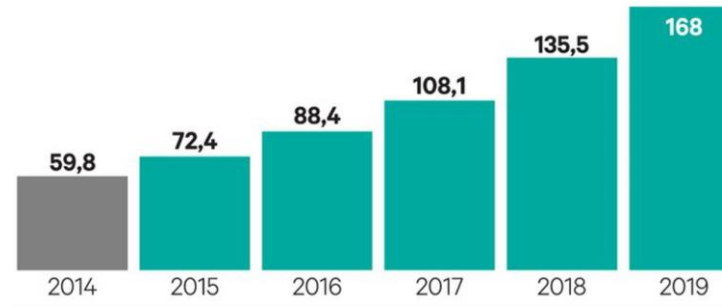
# Introduction

## Les raisons du succès

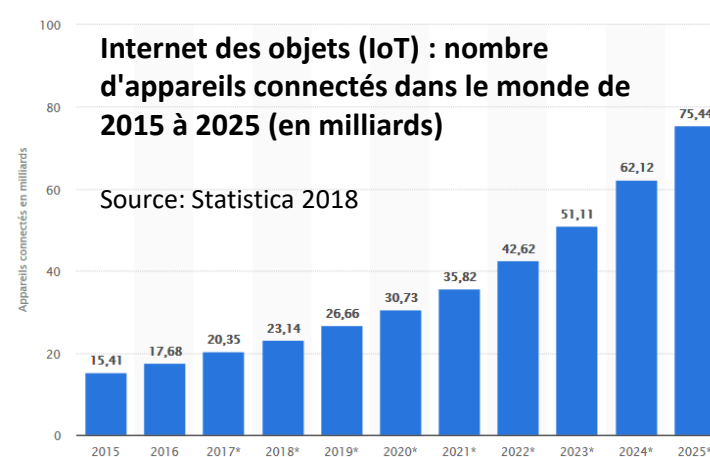
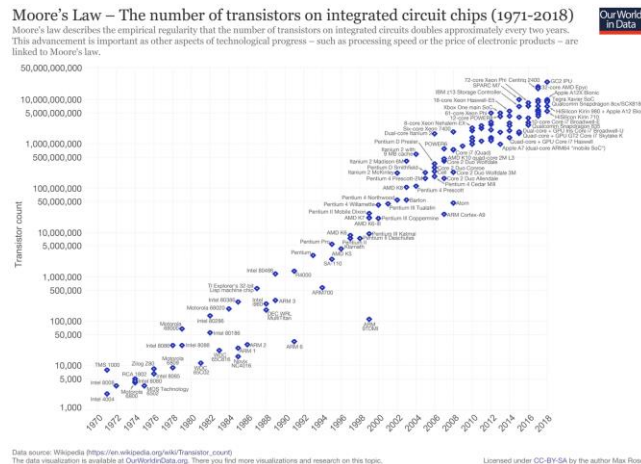
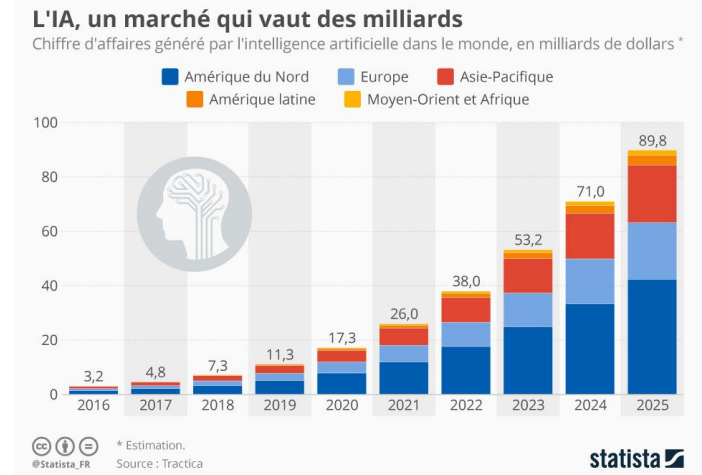


### La croissance du trafic Internet mondial

En exabytes (  $10^{18}$  bytes soit 1 milliard de gigabytes) par mois

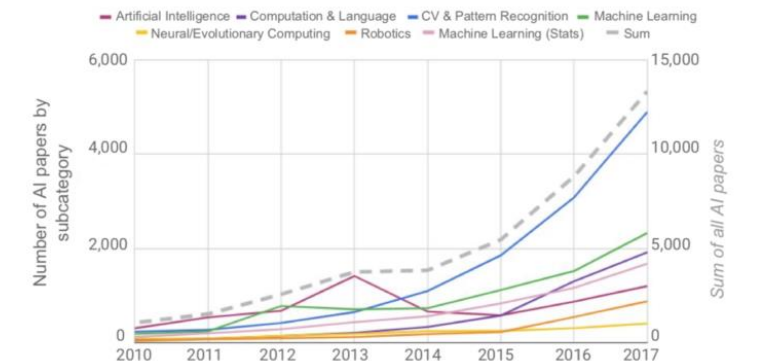


Source : les échos juillet 2015



### Number of AI papers on arXiv by subcategory (2010–2017)

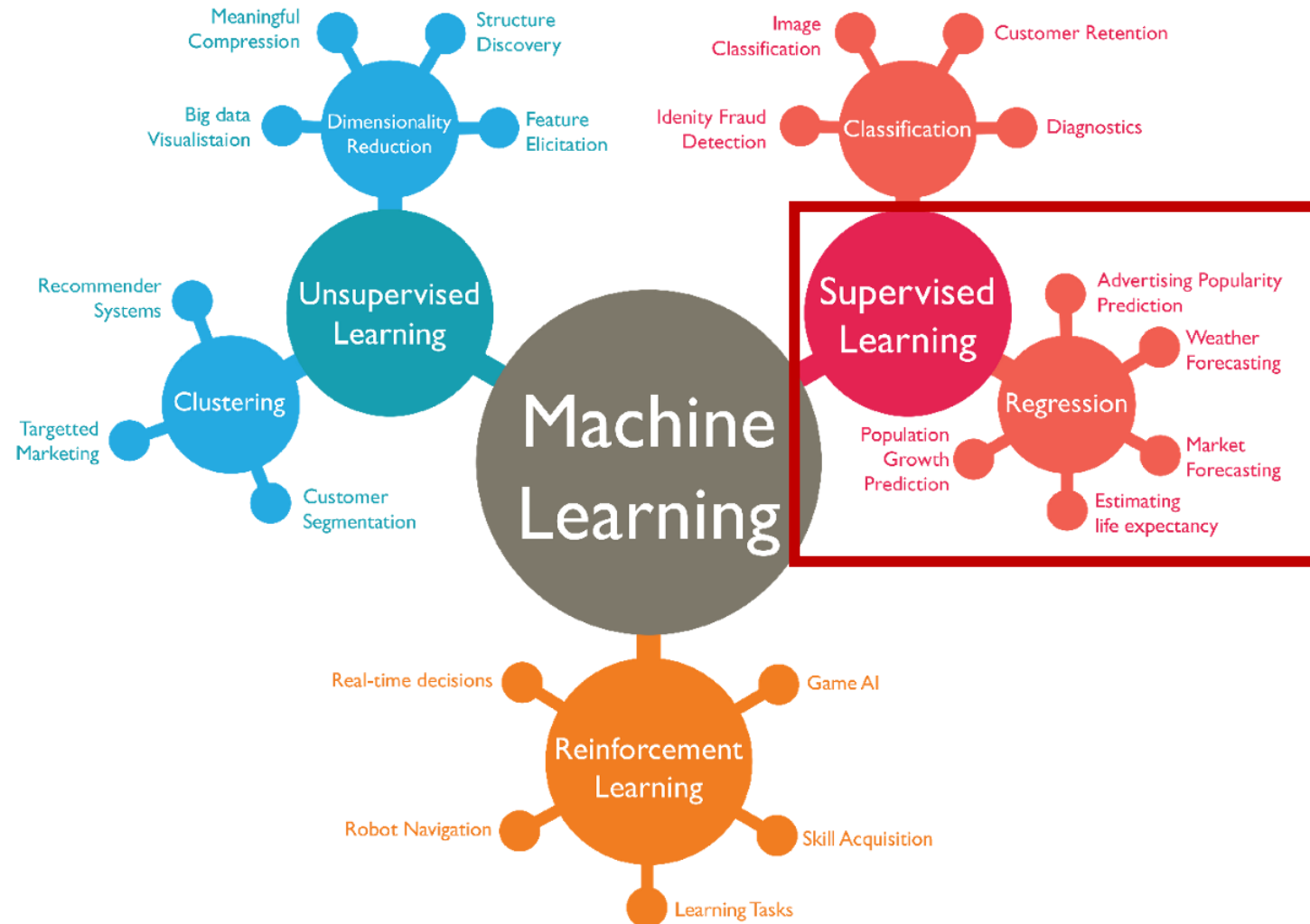
Source: arXiv





# Algorithmes et des tâches en ML

## Typologie



# Algorithmes supervisés pour la régression

## Principes de base

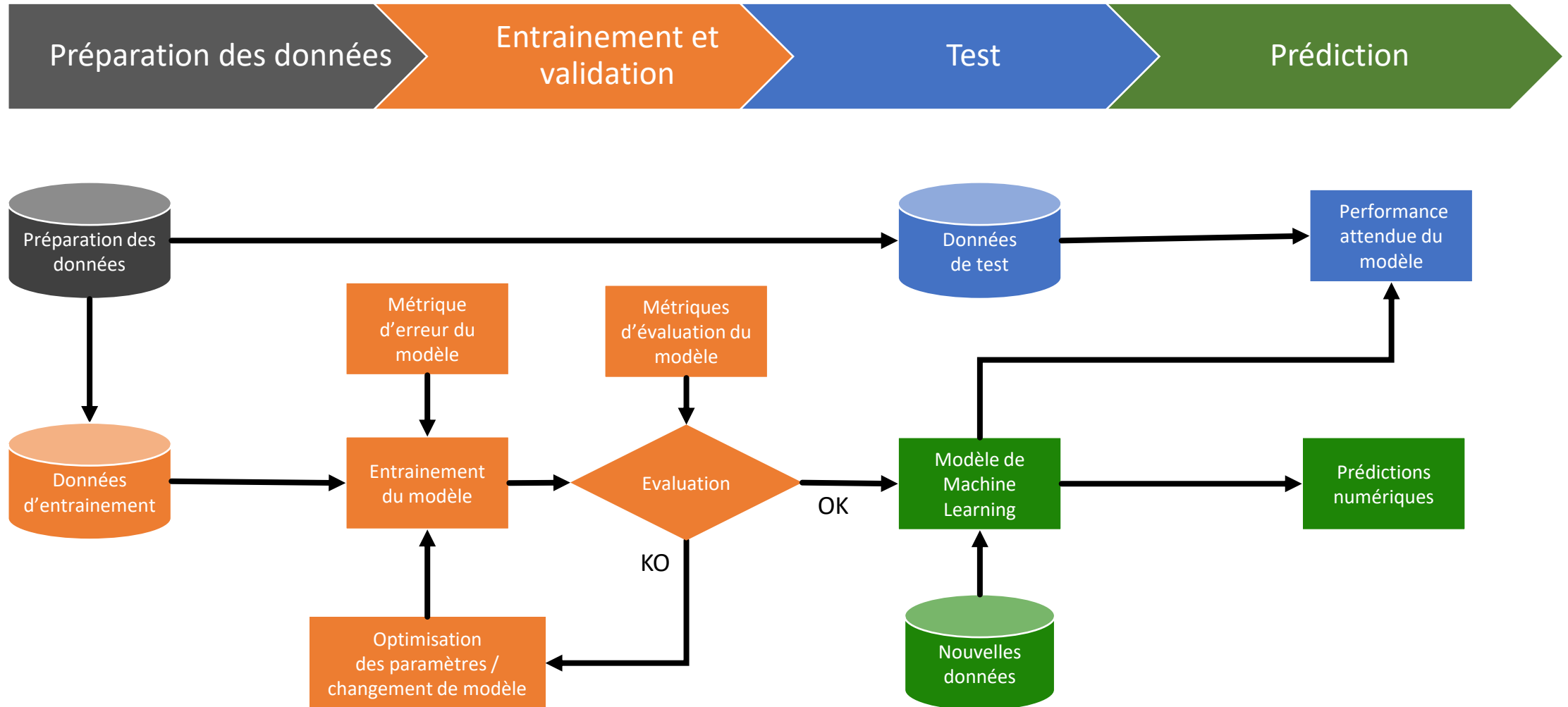
On parle d'apprentissage supervisé pour les algorithmes qui utilisent des données d'entraînement **étiquetées** ou autrement dit qui comportent un **label dont on cherche à approximer la valeur à travers les caractéristiques ou *attributs* intrinsèques des observations.**

On parle de **régression** lorsque le label correspond à une **valeur continue**.

- Prédire le cours d'une actions à travers des données boursières
- Prédire le prix d'un bien à partir de ses caractéristiques
- Prédire l'accidentologie routière à partir d'informations concernant les conducteurs
- ...

# Algorithmes supervisés pour la régression

## Processus d'apprentissage



# Algorithmes supervisés pour la régression

## Algorithmes

Algorithme	Principe
Régression linéaire	Établir une relation linéaire entre une variable $y$ (label) et plusieurs variables $x_i$ (caractéristiques)
Régression polynomiale	Établir une relation polynomiale entre une variable $y$ (label) et plusieurs variables $x_i$ (caractéristiques)
K plus proches voisins (KNN)	Estimer la valeur du label à partir des $k$ observations les plus proches compte tenu d'une distance appliquée aux caractéristiques
Machines à vecteur de support (SVM)	Identifier un ajustement linéaire ou non linéaire au voisinage duquel le nombre d'observations est maximisé.
Arbres de décision	Représenter les exemples avec un arbre où chaque nœud est étiqueté avec un attribut ou une question quelconque, et dans lequel les branches de ce nœud correspondent aux valeurs possibles de l'attribut, ou des réponses à la question.
Forêts aléatoires	Construire des ensembles d'arbres de décision fonctionnant en parallèle dans le but d'éviter les problèmes de sur-apprentissage.
Réseaux de neurones (ANN)	Imiter le fonctionnement des neurones humains par la création d'une collection de neurones artificiels simples reliés par des connexions pondérées dirigées, et actionnées selon des fonctions d'activation.
Méthodes de Boosting	Le but des méthodes de Boosting est de combiner les prédictions de plusieurs estimateurs de base afin d'améliorer la robustesse. Le modèle le plus connu est le AdaBoost qui utilise une séquence de plusieurs estimateurs simples (arbres de décision, SVMs, KNNs) entraînés avec des ensembles différents des données.

**Certains algorithmes peuvent avoir une double fonction de régression et de classification !**

# Algorithmes supervisés pour la régression

## Formulation du problème

A partir d'un jeu de données d'entraînement  $S=\{(X_i, Y_i)\}_{i=1 \rightarrow m}$  il s'agit de trouver la fonction  $h$  qui approxime le plus possible la fonction continue  $f$  avec :

$$f: R^n \rightarrow R$$

$$X \rightarrow Y=f(X)$$

La qualité de l'approximation se mesure à travers une fonction loss / qu'il s'agit de minimiser pour trouver les meilleurs paramètres de  $h$ .

$$\text{Min } l(h(X), Y)$$

# Algorithmes supervisés pour la régression

## Formulation du problème

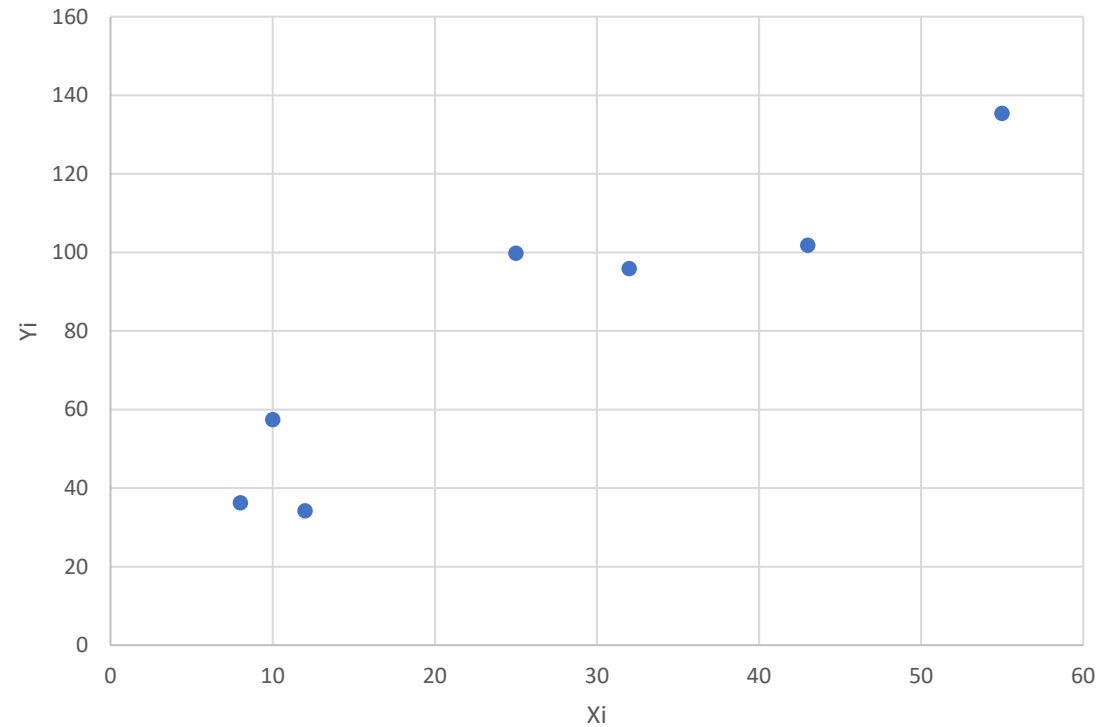
Dans le cas de la régression, la fonction loss  $J$  correspond généralement à une mesure de :

- *MSE : Mean Square Error*
- *RMSE : Root Mean Square Error*
- *MAE : Mean Absolute Error*

L'entraînement consiste à identifier les paramètres de la fonction  $h$  qui minimisent la fonction  $J$ . La forme générale de  $h$  (linéaire, polynomiale,...) est une hypothèse d'entrée du modèle.

# Evaluation, test et généralisation des modèles

Choix du modèle et notion de complexité



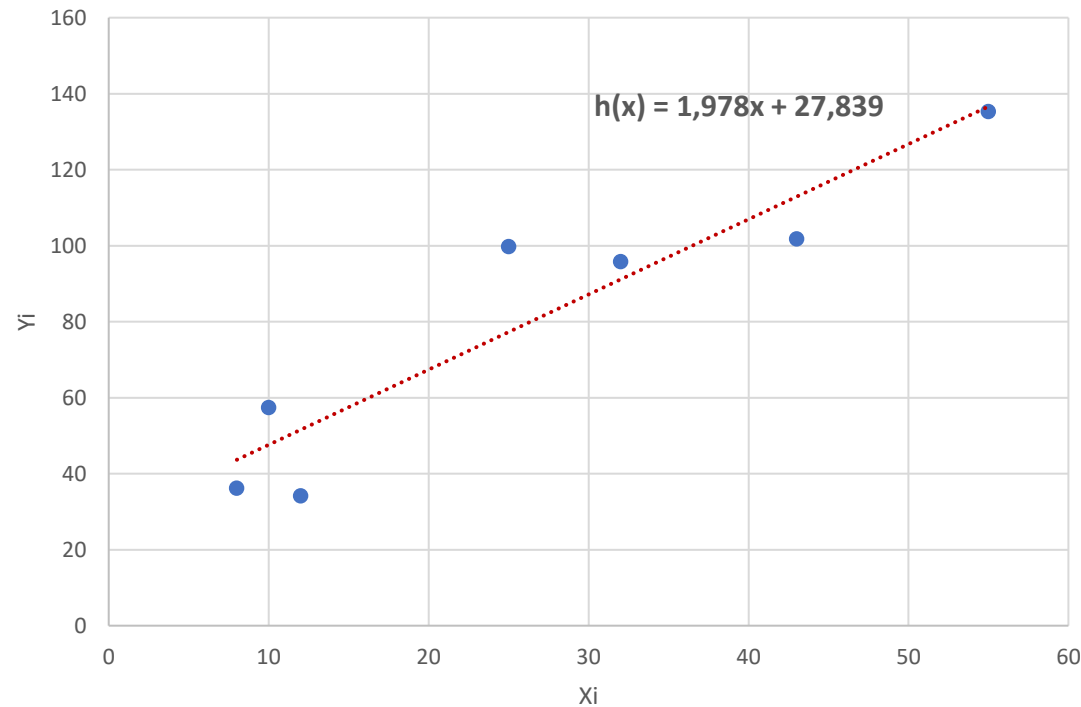
Quelle hypothèse pour la forme générale de  $h$  ?

Quelle fonction loss / ?



# Evaluation, test et généralisation des modèles

## Choix du modèle et notion de complexité



Quelle hypothèse pour la forme générale de  $h$  ?

$$h(X) = \theta_0 + \theta_1 \times X$$

Quelle fonction loss  $l$  ?

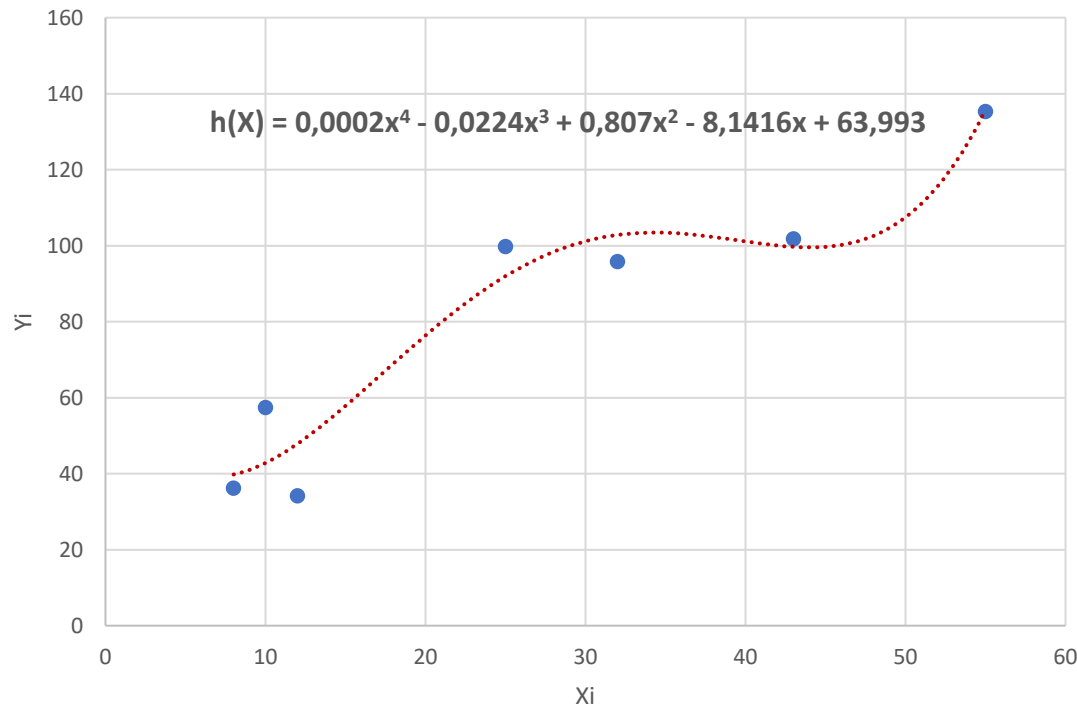
$$\text{Min}_{\theta_j} l(h(X), Y) = \frac{1}{m} \times \sum_{i=1}^m (h(X_i) - Y_i)^2$$

Après optimisation des paramètres  $\theta_j$ :

$$RMSE = 12463$$

# Evaluation, test et généralisation des modèles

## Choix du modèle et notion de complexité



Quelle hypothèse pour la forme générale de  $h$  ?

$$h(X) = \theta_0 + \theta_1 \times X + \theta_2 \times X^2 + \theta_3 \times X^3 + \theta_4 \times X^4$$

Quelle fonction loss / ?

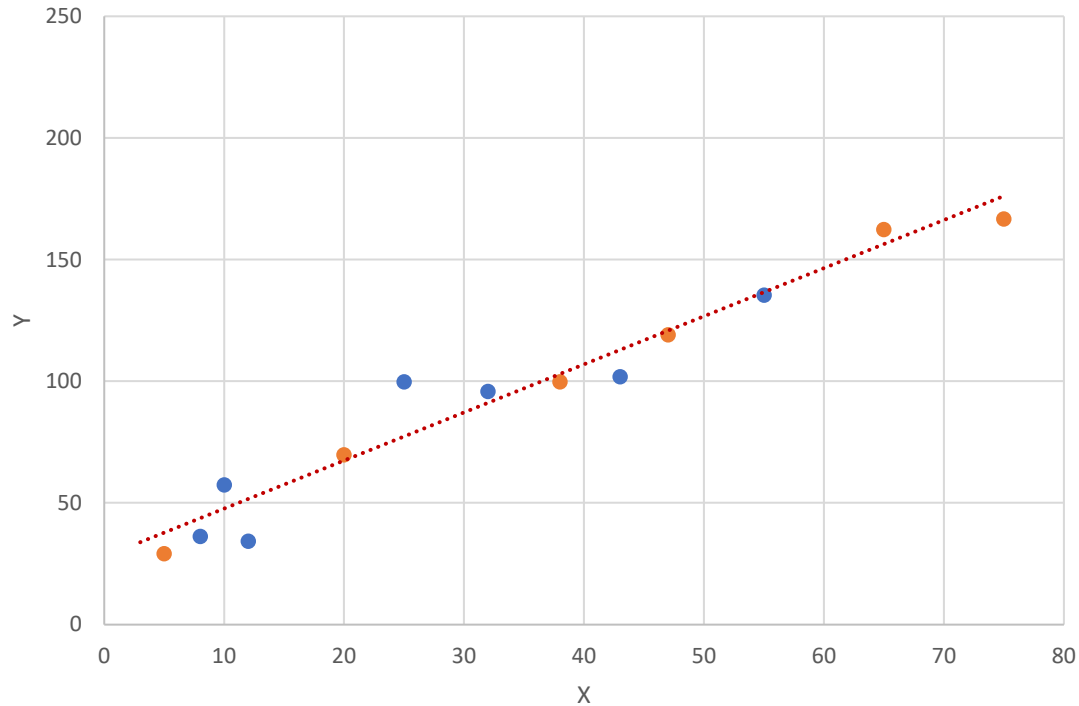
$$\text{Min}_{\theta_j} l(h(X), Y) = \frac{1}{m} \times \sum_{i=1}^m (h(X_i) - Y_i)^2$$

Après optimisation des paramètres  $\theta_j$ :

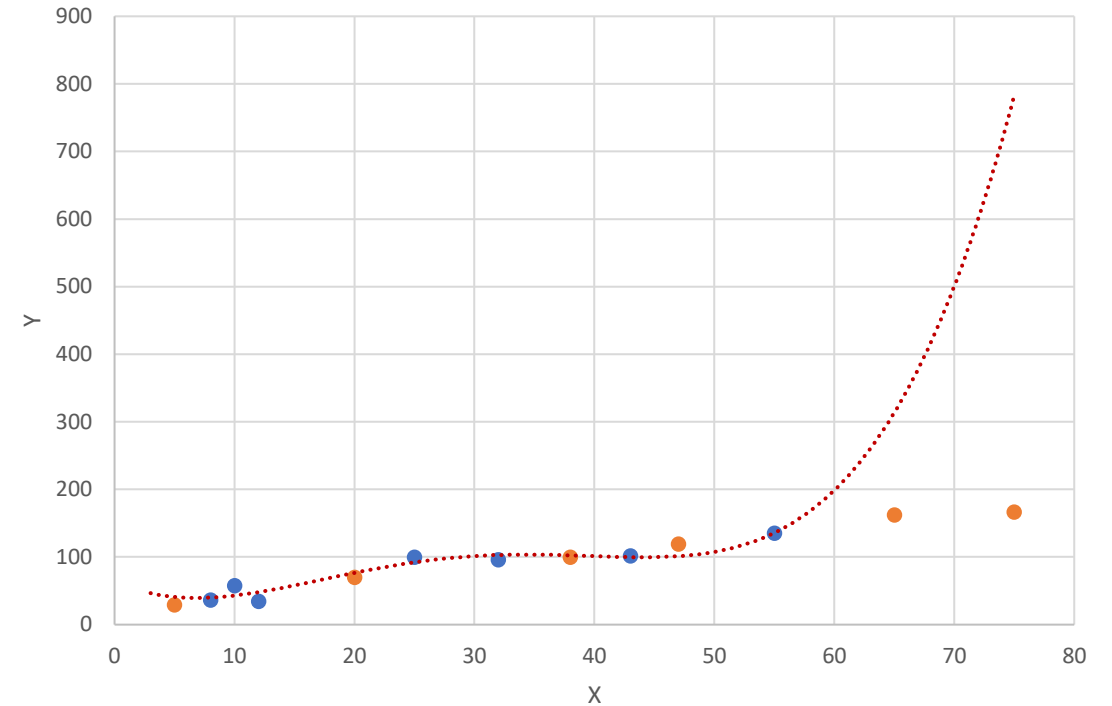
$$RMSE = 177$$

# Evaluation, test et généralisation des modèles

Choix du modèle et notion de complexité



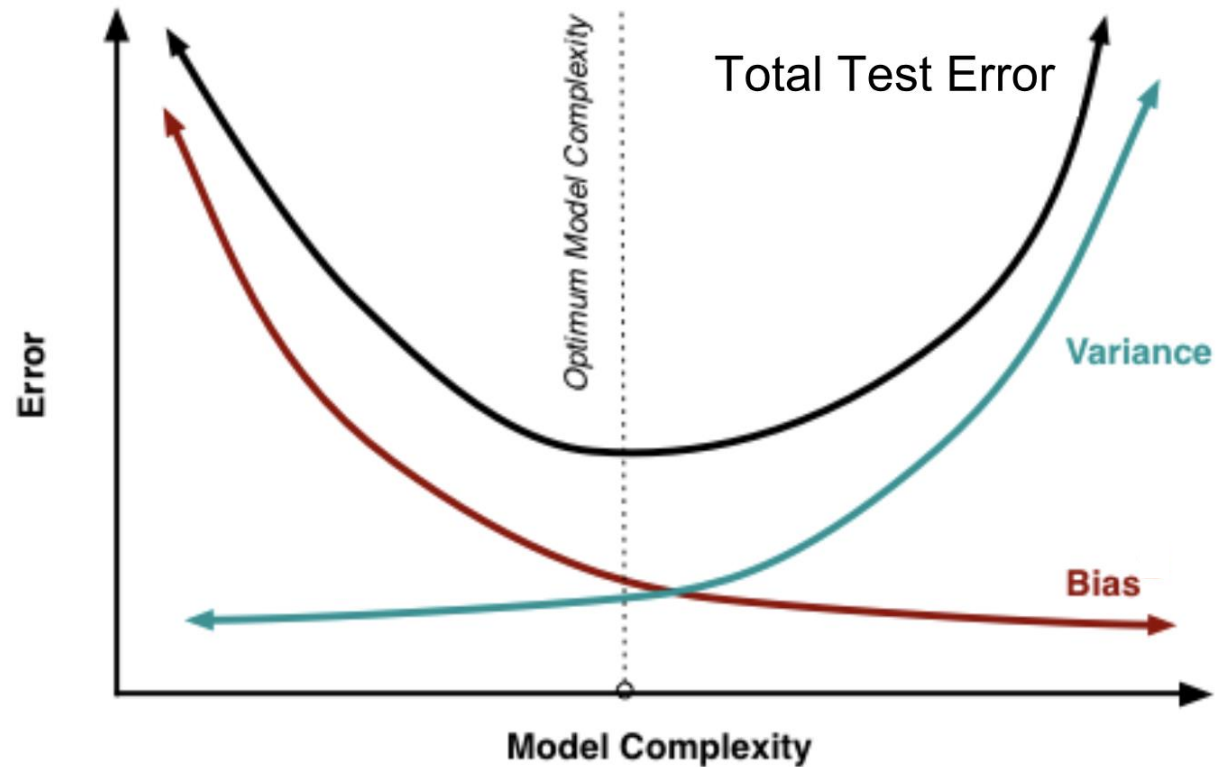
Modèle linéaire (complexité faible)



Modèle polynomiale (complexité forte)

# Evaluation, test et généralisation des modèles

Arbitrage biais variance



Bias: error due to the choice of  $H$ .

Variance: error due to the (weak) representativity of the sample

# Evaluation, test et généralisation des modèles

Arbitrage biais variance

*Principe du rasoir d'ockaham « les hypothèses suffisantes les plus simples doivent être préférées »*




# Evaluation, test et généralisation des modèles

## Données

Descripteurs  
(features)

Label



longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity	median_house_value
-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	NEAR BAY	452600.0
-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	NEAR BAY	358500.0
-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	NEAR BAY	352100.0
-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	NEAR BAY	341300.0
-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	NEAR BAY	342200.0
-122.25	37.85	52.0	919.0	213.0	413.0	193.0	4.0368	NEAR BAY	269700.0
-122.25	37.84	52.0	2535.0	489.0	1094.0	514.0	3.6591	NEAR BAY	299200.0
-122.25	37.84	52.0	3104.0	687.0	1157.0	647.0	3.12	NEAR BAY	241400.0
-122.26	37.84	42.0	2555.0	665.0	1206.0	595.0	2.0804	NEAR BAY	226700.0
-122.25	37.84	52.0	3549.0	707.0	1551.0	714.0	3.6912	NEAR BAY	261100.0
-122.26	37.85	52.0	2202.0	434.0	910.0	402.0	3.2031	NEAR BAY	281500.0
-122.26	37.85	52.0	3503.0	752.0	1504.0	734.0	3.2705	NEAR BAY	241800.0
-122.26	37.85	52.0	2491.0	474.0	1098.0	468.0	3.075	NEAR BAY	213500.0
-122.26	37.84	52.0	696.0	191.0	345.0	174.0	2.6736	NEAR BAY	191300.0
-122.26	37.85	52.0	2643.0	626.0	1212.0	620.0	1.9167	NEAR BAY	159200.0
-122.26	37.85	50.0	1120.0	283.0	697.0	264.0	2.125	NEAR BAY	140000.0
-122.27	37.85	52.0	1966.0	347.0	793.0	331.0	2.775	NEAR BAY	152500.0
-122.27	37.85	52.0	1228.0	293.0	648.0	303.0	2.1202	NEAR BAY	155500.0
-122.26	37.84	50.0	2239.0	455.0	990.0	419.0	1.9911	NEAR BAY	158700.0
-122.27	37.84	52.0	1503.0	298.0	690.0	275.0	2.6033	NEAR BAY	162900.0

Exemple  
(instance  $X_i$ )

# Evaluation, test et généralisation des modèles

## Données

longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity	median_house_value
-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	NEAR BAY	452600.0
-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	NEAR BAY	358500.0
-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	NEAR BAY	352100.0
-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	NEAR BAY	341300.0
-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	NEAR BAY	342200.0
-122.25	37.85	52.0	919.0	213.0	413.0	193.0	4.0368	NEAR BAY	269700.0
-122.25	37.84	52.0	2535.0	489.0	1094.0	514.0	3.6591	NEAR BAY	299200.0
-122.25	37.84	52.0	3104.0	687.0	1157.0	647.0	3.12	NEAR BAY	241400.0
-122.26	37.84	42.0	2555.0	665.0	1206.0	595.0	2.0804	NEAR BAY	226700.0
-122.25	37.84	52.0	3549.0	707.0	1551.0	714.0	3.6912	NEAR BAY	261100.0
-122.26	37.85	52.0	2202.0	434.0	910.0	402.0	3.2031	NEAR BAY	281500.0
-122.26	37.85	52.0	3503.0	752.0	1504.0	734.0	3.2705	NEAR BAY	241800.0
-122.26	37.85	52.0	2491.0	474.0	1098.0	468.0	3.075	NEAR BAY	213500.0
-122.26	37.84	52.0	696.0	191.0	345.0	174.0	2.6736	NEAR BAY	191300.0
-122.26	37.85	52.0	2643.0	626.0	1212.0	620.0	1.9167	NEAR BAY	159200.0
-122.26	37.85	50.0	1120.0	283.0	697.0	264.0	2.125	NEAR BAY	140000.0
-122.27	37.85	52.0	1966.0	347.0	793.0	331.0	2.775	NEAR BAY	152500.0
-122.27	37.85	52.0	1228.0	293.0	648.0	303.0	2.1202	NEAR BAY	155500.0
-122.26	37.84	50.0	2239.0	455.0	990.0	419.0	1.9911	NEAR BAY	158700.0
-122.27	37.84	52.0	1503.0	298.0	690.0	275.0	2.6033	NEAR BAY	162900.0

Training Set : utilisé pour minimiser le risque empirique (en pratique 60 à 80% des données)

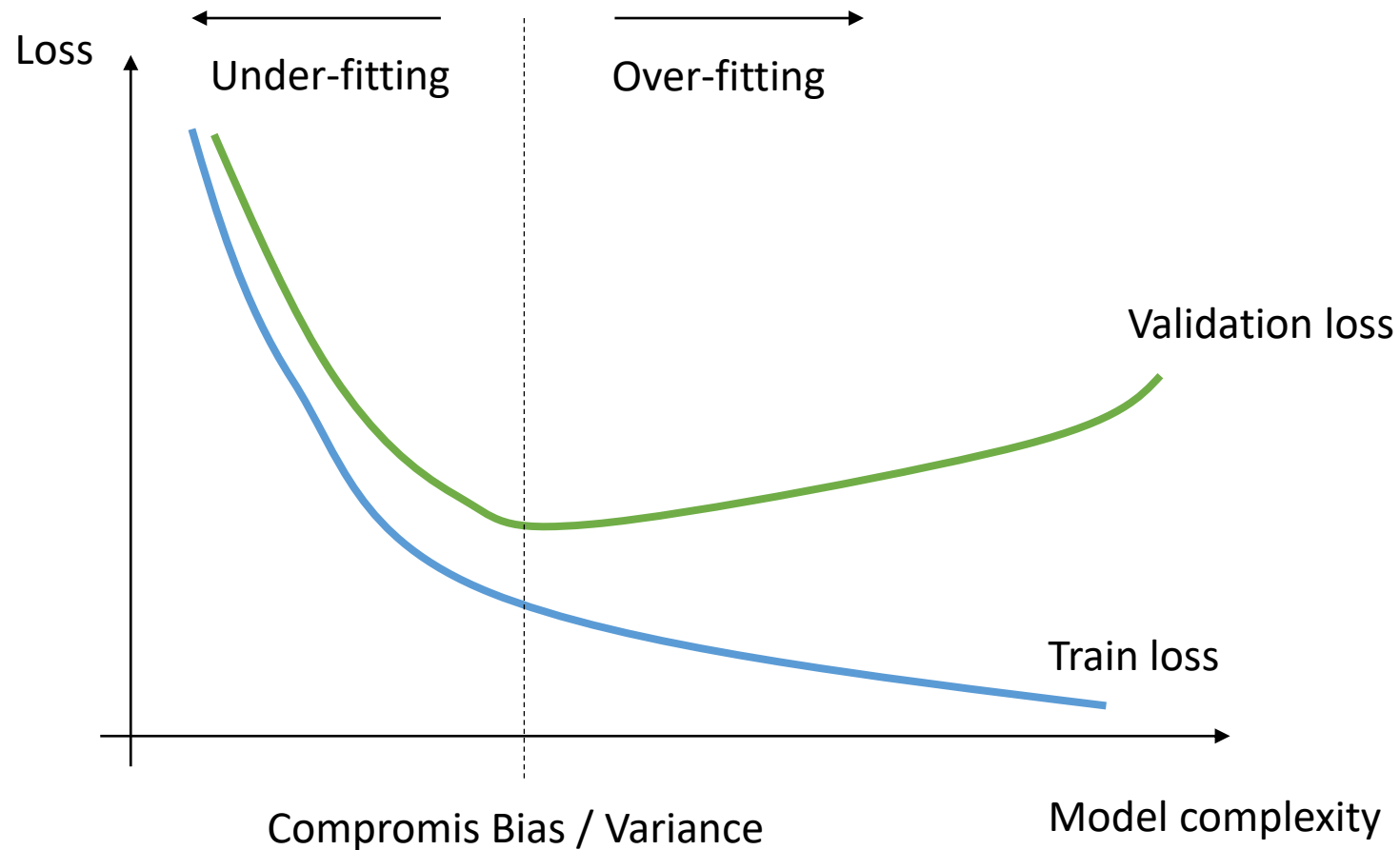
Validation Set : utilisé pour estimer le risque en généralisation d'une hypothèse sur la forme générale de  $h$  (en pratique 20% des données ou cross-validation)

Test Set : utilisé pour estimer la performance du modèle retenu



# Evaluation, test et généralisation des modèles

Sur et sous entraînement



# Outils

- Langages de programmation



- Bibliothèques

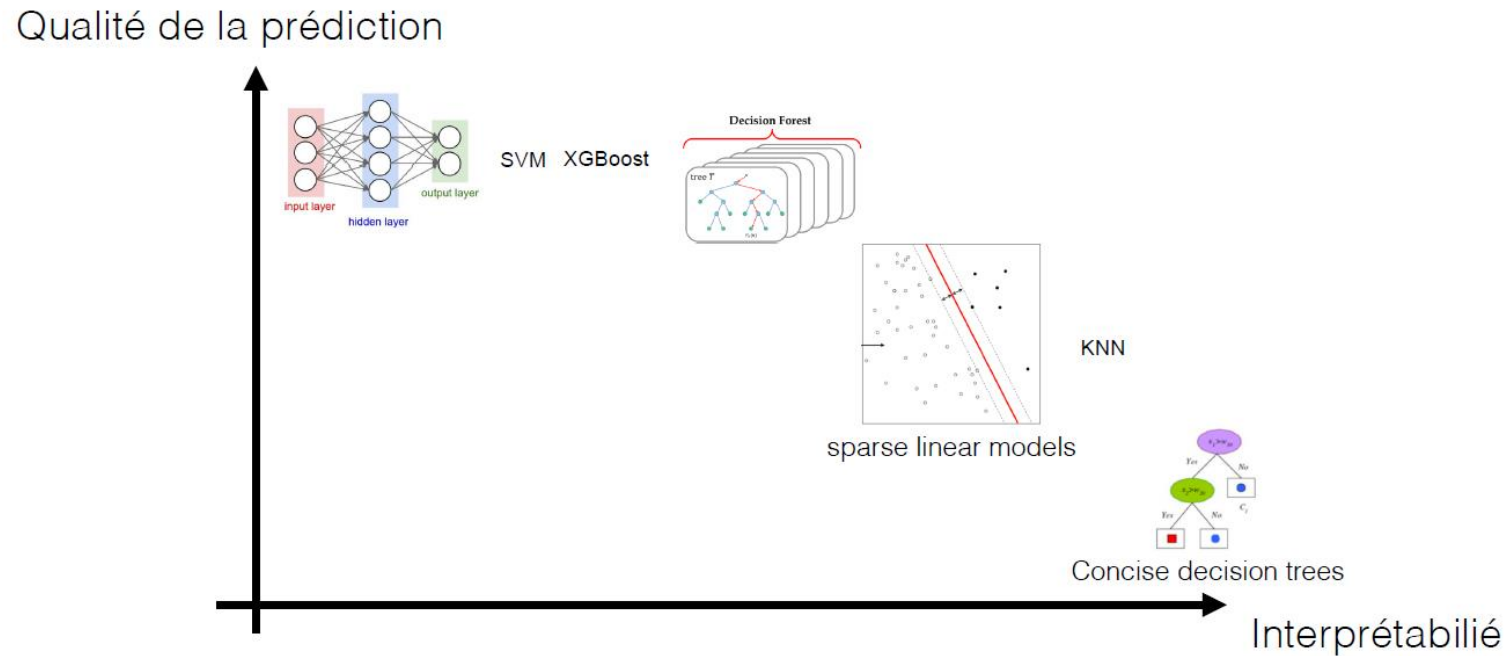


- Interfaces utilisateur



# Développements et enjeux actuels

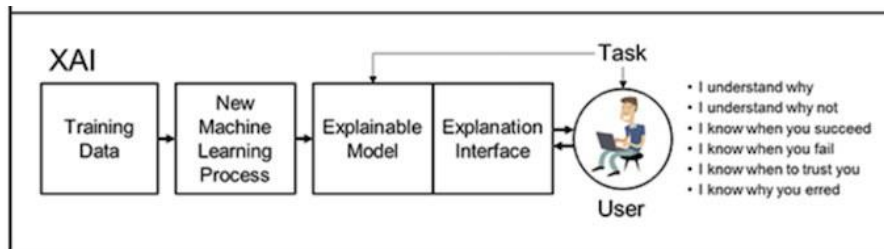
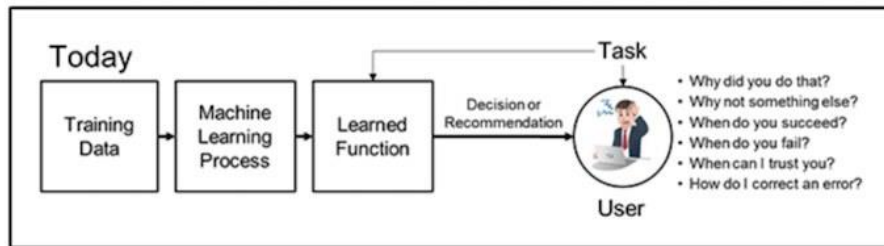
Interprétabilité, robustesse et confidentialité



**Interprétabilité : comment rendre interprétables les prédictions faites par un modèle ML ?**

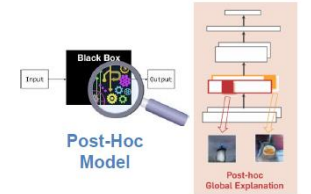
# Développements et enjeux actuels

Interprétabilité, robustesse et confidentialité



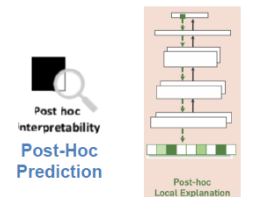
## Explicabilité type A : Interpréter les modèles black-box.

Qu'est ce qui a été appris par le modèle et qu'est ce qui s'y cache ?



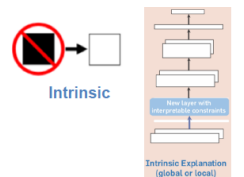
## Explicabilité type B: Interpréter les prédictions des modèles black-box

Pourquoi telle valeur de label a-t-elle été générée par le modèle pour tel individu ?



## Explicabilité type C: Apprentissage automatique à partir de modèles interprétables

Comment expliquer intrinsèquement le modèle ?



**Interprétabilité : comment rendre interprétables les prédictions faites par un modèle ML ?**

# Développements et enjeux actuels

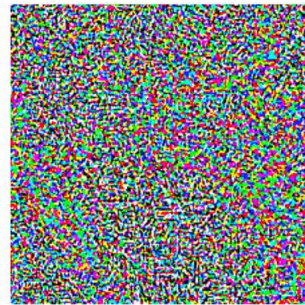
Interprétabilité, robustesse et confidentialité



“panda”

57.7% confidence

+ .007 ×



noise

=



“gibbon”

99.3% confidence



Panneau STOP

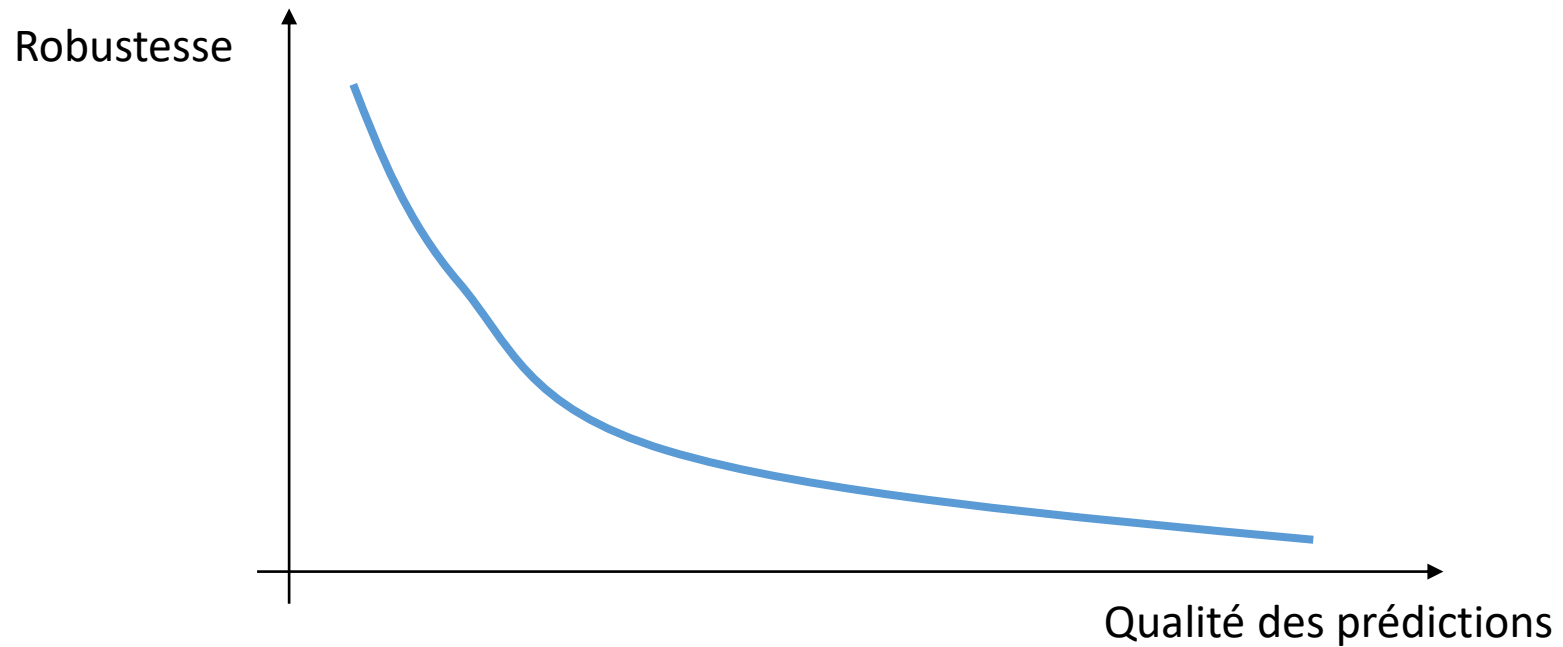


Panneau limite  
50 km/h

**Robustesse : comment rendre un modèle robuste face à des attaques adversariales ?**

# Développements et enjeux actuels

Interprétabilité, robustesse et confidentialité



**Robustesse : comment rendre un modèle robuste face à des attaques adversariales ?**

# Développements et enjeux actuels

Interprétabilité, robustesse et confidentialité

“87% of the population in the United States had reported characteristics that likely made them unique based only on {5-digit ZIP, gender, date of birth}”

Latanya Sweeney 2002 “K-anonymity: A model for protecting privacy”

“few movie ratings suffice to uniquely identify anonymized users from Netflix prize dataset by linking with IMDB publicly available database. This was a subject of lawsuits and had a major impact on Netflix’s privacy policy.”

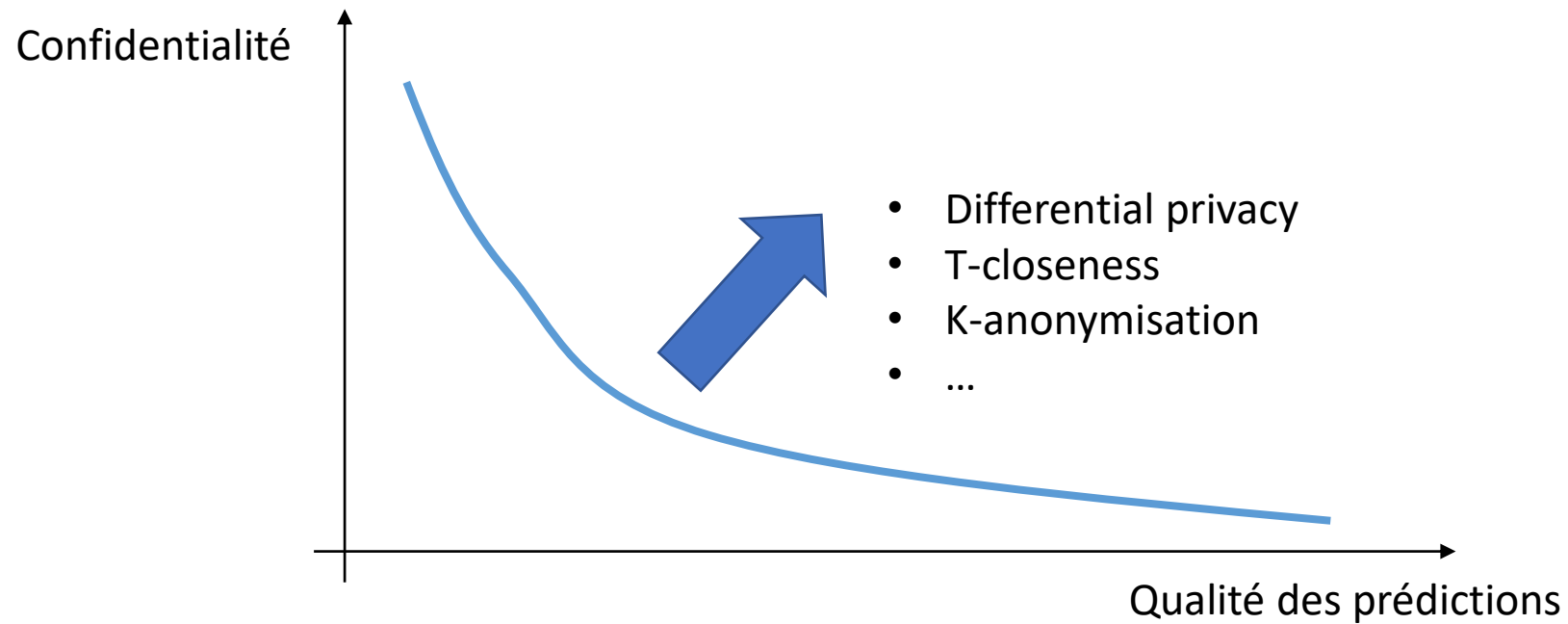
Arvind Narayanan et al. 2008 “ Robust de-anonymization of large sparse datasets “

**Confidentialité : comment garantir la confidentialité des données utilisées en ML ?**



# Développements et enjeux actuels

Interprétabilité, robustesse et confidentialité



**Confidentialité : comment garantir la confidentialité des données utilisées en ML ?**

# Introduction à Python

# Introduction

## Paradigmes de programmation

- **Programmation impérative** : décrit les opérations en séquences d'instructions exécutées par l'ordinateur après compilation ou interprétation
  - Structurée : organisation hiérarchique du code (Pascal, C, C++, Java, **Python**,...)
  - Procédurale : décomposition du code en routines et sub-routines (Pascal, C, C++, Java, **Python**,...)
- **Programmation orientée objet** : définition et assemblage de briques logicielles élémentaires dotées d'attributs (caractéristiques) et de méthodes (comportements) (C++, Java, **Python**,...)
- **Programmation déclarative**
  - Descriptive : description de structures de données (HTML, XML,...)
  - Fonctionnelle: décomposition en fonctions mathématiques (Lisp, **Python**,...)
  - Logique : expression sous forme de prédicat (Prolog, SQL,...)

# Introduction python™

- Né en 1989
- Actuellement version 3.8 (Distribution anaconda, <https://www.anaconda.com/products/individual>)
- Langage interprété orienté objet
- Typage dynamique
- Adapté au ML grâce aux bibliothèques: ScikitLearn, Pandas, Numpy

# Environnement de travail

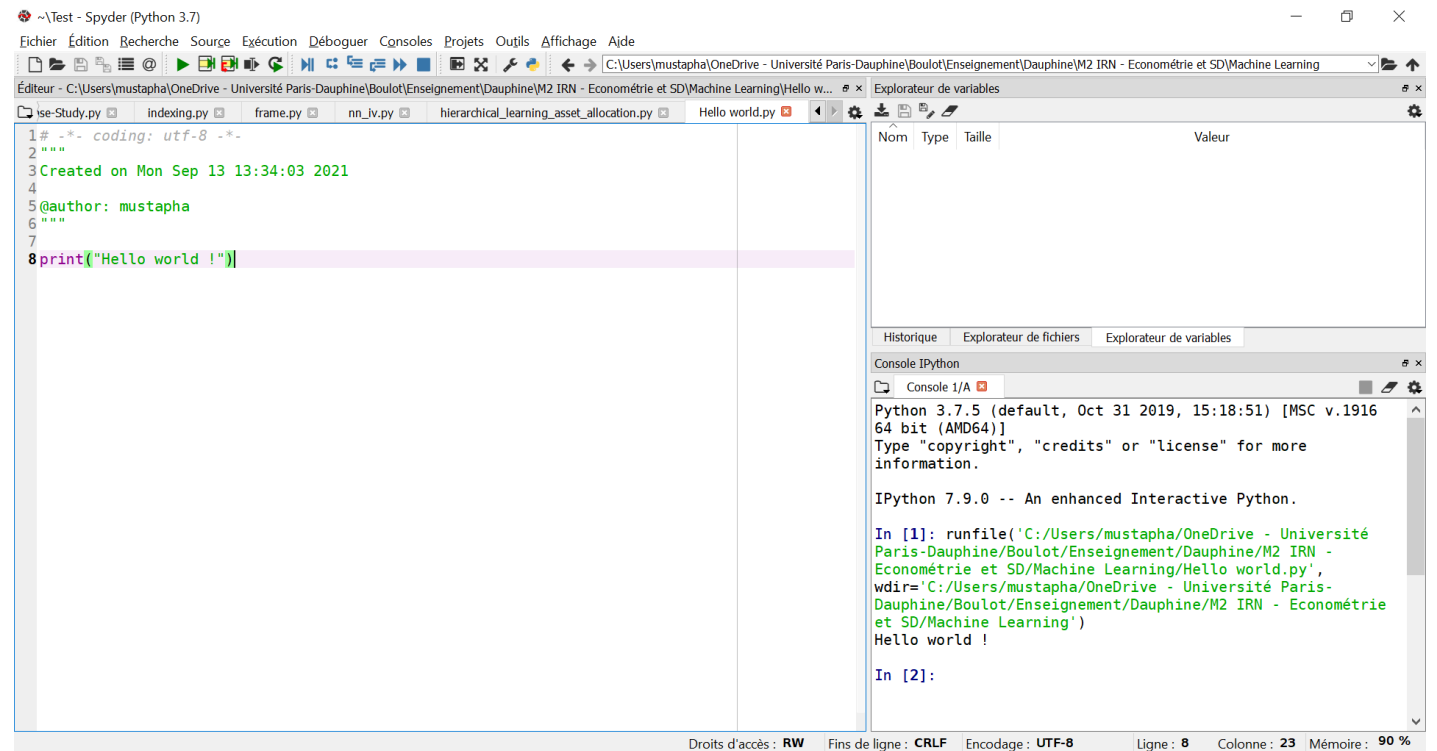
## Spyder



Un **IDE** (integrated development environment) est un logiciel de création d'applications, qui rassemble des outils de développement fréquemment utilisés dans une seule interface utilisateur.

Ces outils sont dédiés à :

- L'édition de code
- Le débogage
- L'exécution
- L'affichage



# Environnement de travail

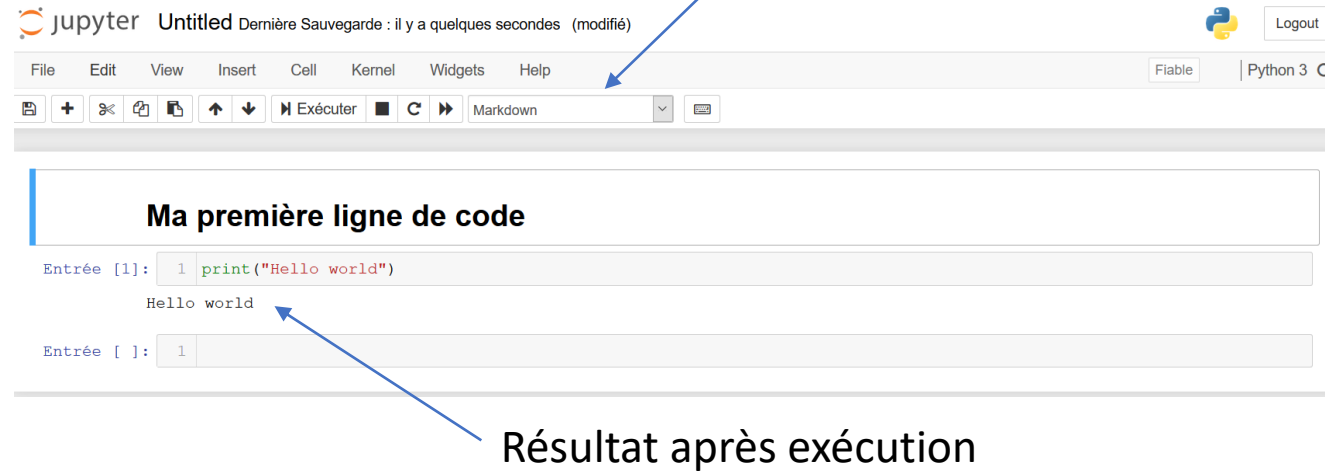
Jupyter



Type de cellule  
(texte, code)

Menu  
principal

Raccourcis  
(enregistrement  
, exécution,...)



Noyau actif

Cellule de texte  
(markdown)

Cellule de code

Résultat après exécution

**Jupyter** est une application web qui permet de réaliser des **notebooks**, c'est-à-dire des fichiers organisés en cellules contenant du texte ou du code.

Le notebook Jupyter est un **environnement de programmation interactif simple d'utilisation** mais limité sur certaines fonctionnalités comme le débogage par exemple.

# Introduction au Machine Learning en Python

Master SIREN – Parcours IRN

(2/3)

UE: Econométrie et science des données

---

**Mustapha Sali**

Maître de Conférences HDR - Université Paris Dauphine

[mustapha.Sali@dauphine.psl.eu](mailto:mustapha.Sali@dauphine.psl.eu)



# Plan

## Introduction à Pandas et Numpy

- Séries et Dataframes
- Manipulation de données avec Pandas
- Numpy arrays
- Calculs avec Numpy

## Régression linéaire et polynomiale

- Modèle de régression linéaire
- Equation normale
- Descente de gradient
- Régularisation
- Modèle de régression polynomiale

# Introduction à Pandas et Numpy

## Notebook Jupyter

# Régression linéaire et polynomiale

# Modèle de régression linéaire

## Notations

**Variables**

**Exemple ou  
instance**

**Vecteur de  
variables**

$X_i(x_{i1}, x_{i2}, x_{i3})$

Variable 1 (x1)	Variable 2 (x2)	Variable 3 (x3)	Phénomène à prédire (y)
68	242	13	978
83	365	7	1370
84	384	4	1495
76	391	9	1511
71	458	2	1792
78	392	5	1518
68	397	4	1580
85	265	12	1007
83	466	14	1829
83	381	14	1442

...

...

...

...

82	346	16	1345
73	390	9	1526
75	215	11	831
78	391	3	1526
79	392	10	1498

**Label**

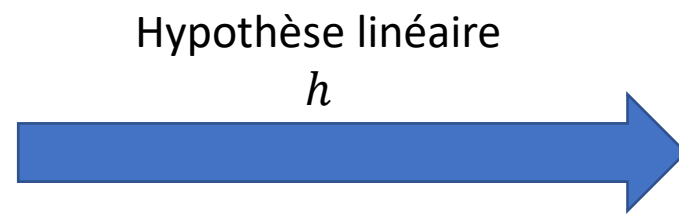
$y_i$

# Modèle de régression linéaire

Forme générale

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1j} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x_{i1} & \dots & \dots & x_{ij} & \dots & x_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x_{m1} & \dots & \dots & x_{mj} & \dots & x_{mn} \end{bmatrix}$$

Matrice  $X(m, n)$



$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_i \\ \dots \\ y_m \end{bmatrix}$$

Matrice  $Y(m, 1)$

# Modèle de régression linéaire

Forme générale

$$\begin{bmatrix} \mathbf{1} & x_{11} & x_{12} & \dots & x_{1j} & \dots & x_{1n} \\ \mathbf{1} & x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \mathbf{1} & x_{i1} & \dots & \dots & x_{ij} & \dots & x_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \mathbf{1} & x_{m1} & \dots & \dots & x_{mj} & \dots & x_{mn} \end{bmatrix} \times \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \dots \\ \theta_j \\ \dots \\ \theta_n \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \dots \\ \hat{y}_i \\ \dots \\ \hat{y}_m \end{bmatrix} \longleftrightarrow \hat{Y} = X \times \theta$$

Matrice  $X(m, n + 1)$       Matrice  $\theta \ (n + 1, 1)$       Matrice  $\hat{Y}(m, 1)$

Pour un exemple  $i$  donné l'hypothèse linéaire  $h$  se traduit comme suit :

$$\hat{y}_i = \mathbf{h}_\theta(\mathbf{X}_i) = \mathbf{X}_i \cdot \boldsymbol{\theta} = \theta_0 + \theta_1 x_{i1} + \dots + \theta_j x_{ij} + \dots + \theta_n x_{in}$$

Avec :

$\hat{y}_i$ : valeur prédite

$n$ : nombre de variables

$\theta_j$ :  $j^{\text{ème}}$  paramètre du modèle (avec  $\theta_0$  terme constant ou intercept)

# Equation normale

Fonction Loss

Concevoir un modèle linéaire  $h_\theta$  revient à trouver les valeur de  $\theta_j$  qui minimisent un écart mesuré sur le jeu d'entraînement entre les valeurs réelles  $y_i$  et les valeurs prédites  $\hat{y}_i$

La fonction qui mesure l'écart à minimiser s'appelle la Loss et se note  $\mathcal{L}$ . Dans le cas de la régression, une façon de mesurer la Loss est la MSE (Mean Squared Error). Appliqué à la régression linéaire on a :

$$\mathcal{L}(\theta) = \frac{1}{m} \sum_{i=1}^m (\hat{y}_i - y_i)^2 = \frac{1}{m} \sum_{i=1}^m (h_\theta(X_i) - y_i)^2$$

# Equation normale

Résolution exacte

La fonction à minimiser  $\mathcal{L}(\theta) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(X_i) - y_i)^2$

$$\mathcal{L}(\theta) = \frac{1}{m} \sum_{i=1}^m (X_i \cdot \theta - y_i)^2$$

$$\mathcal{L}(\theta) = \frac{1}{m} (X \cdot \theta - Y)^T \cdot (X \cdot \theta - Y)$$



## Equation normale

Résolution exacte

$$\begin{aligned}\frac{d\mathcal{L}(\theta)}{d\theta} &= \frac{d}{d\theta} \left( (X \times \theta - Y)^T \cdot (X \times \theta - Y) \right) \\ &= \frac{d}{d\theta} \left( ((X \times \theta)^T - Y^T) \cdot (X \times \theta - Y) \right) \\ &= \frac{d}{d\theta} \left( (X \times \theta)^T \cdot (X \times \theta) - (X \times \theta)^T \cdot Y - Y^T \cdot (X \times \theta) + Y^T \cdot Y \right) \\ &= \frac{d}{d\theta} \left( (X \times \theta)^T \cdot (X \times \theta) - 2 * (X \times \theta)^T \cdot Y + Y^T \cdot Y \right) \\ &= \frac{d}{d\theta} \left( (\theta^T \times X^T) \cdot (X \times \theta) - 2 * (X \times \theta)^T \cdot Y + Y^T \cdot Y \right) \\ &= 2 * (X^T \times X \times \theta - X^T \times Y)\end{aligned}$$

# Equation normale

Résolution exacte

$$\frac{d\mathcal{L}(\theta)}{d\theta} = 0 \Rightarrow X^T \times X \times \theta - X^T \times Y = 0$$

$$\Rightarrow \theta = \underbrace{(X^T \times X)^{-1} \times X^T \times Y}_{\text{Equation Normale}}$$

Equation Normale

L'inversion de la matrice  $(X^T \times X)$  peut être très coûteuse lorsque le nombre de variable  $n$  est grand !

# Descente de gradient

## Principe général

- Minimisation de la Loss par actualisation itératives des paramètres du modèle dans la direction inverse du gradient
- Le gradient mesure la contribution marginale d'un paramètre à la fonction qu'on cherche à optimiser. Il s'obtient par dérivées partielles.
- La descente de gradient se fait selon un pas d'apprentissage fixé à priori noté  $\eta$ .

# Descente de gradient

## Principe général

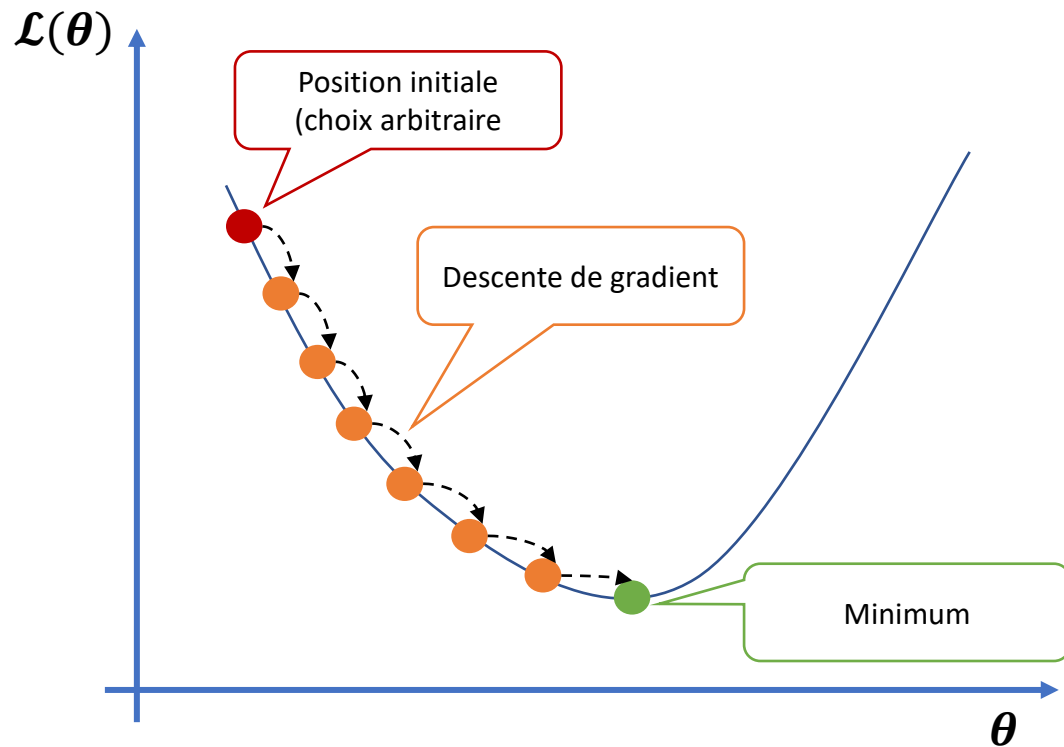


Illustration avec un seul paramètre ( $\theta$ )

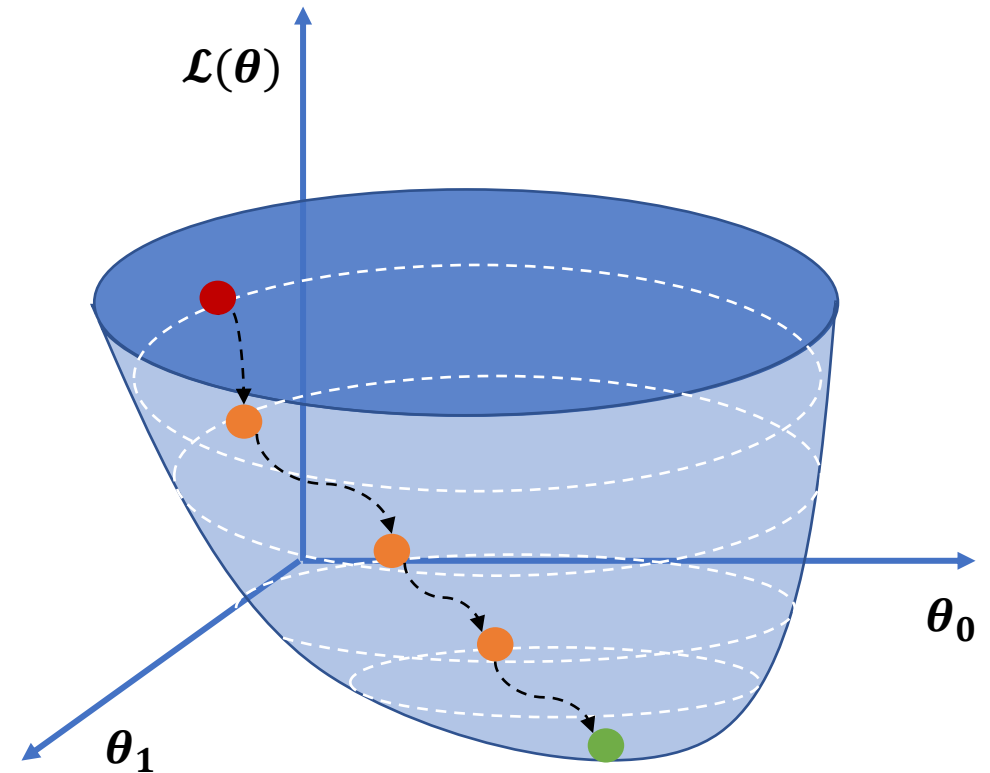
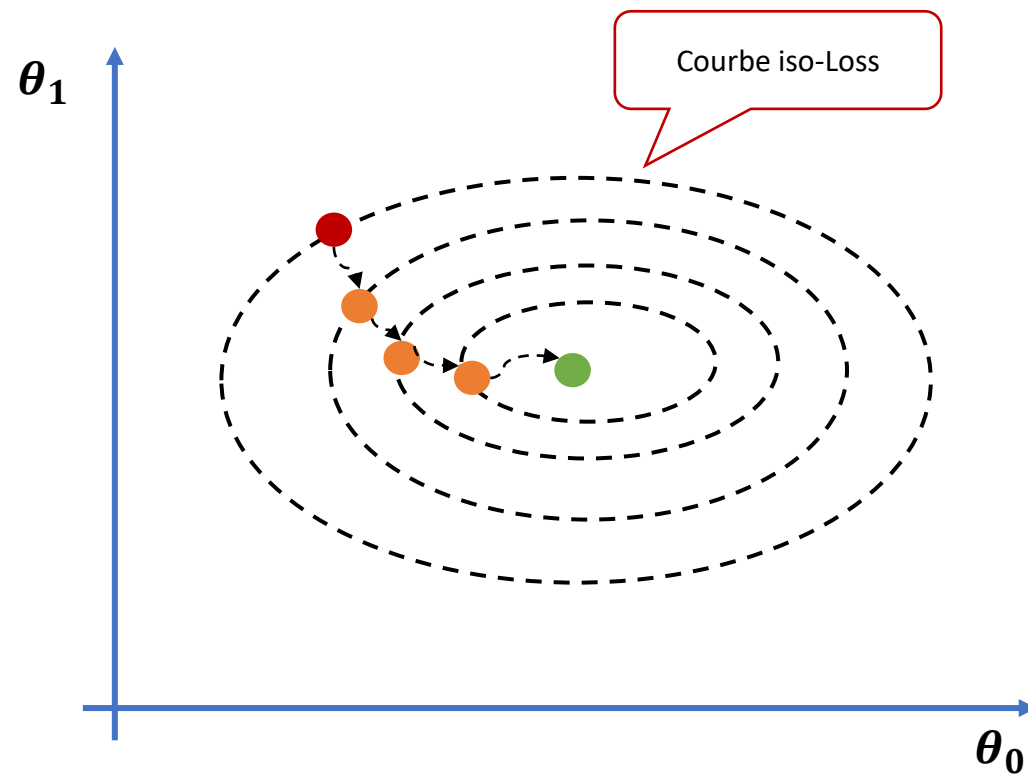
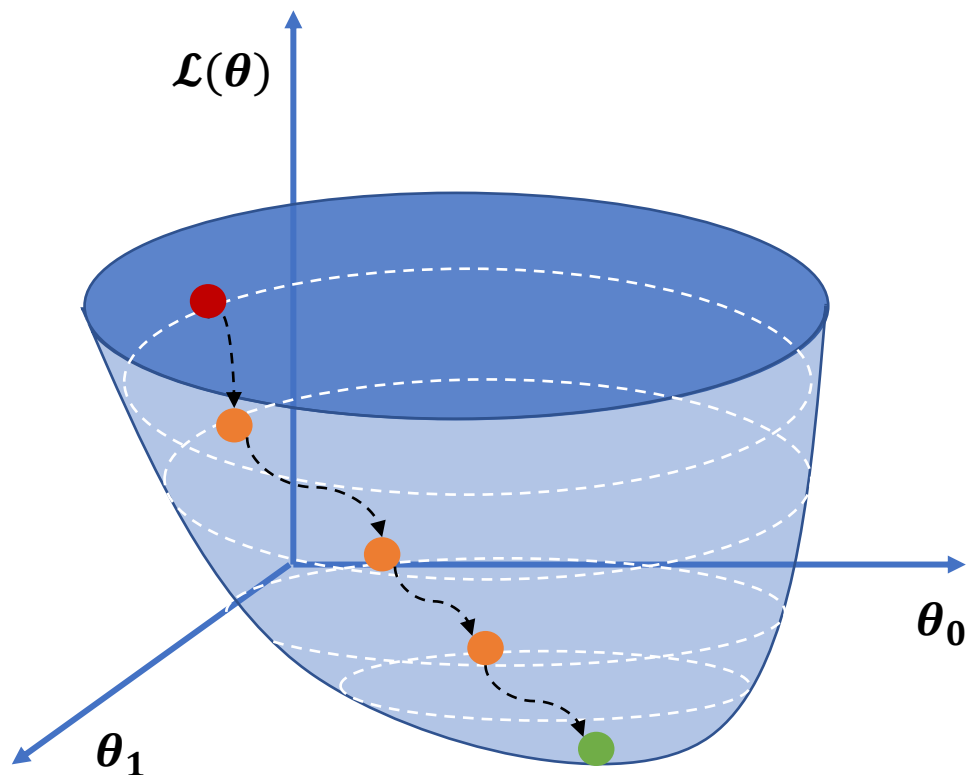


Illustration avec deux paramètres ( $\theta_0, \theta_1$ )

# Descente de gradient

Principe général



# Descente de gradient

Approche ordinaire

L'approche ordinaire consiste à utiliser l'ensemble du jeu de données pour calculer le gradient et mettre à jour les valeurs des paramètres.

Dans le cas de la régression linéaire:

- La Loss calculée à partir de MSE a une forme convexe (garantie de trouver les paramètres optimaux)

- La Loss s'exprime comme étant  $\mathcal{L}(\theta) = MSE(\theta) = \frac{1}{m} \sum_{i=1}^m (X_i \cdot \theta - y_i)^2$

- La dérivée partielle de la Loss par rapport à un paramètre  $\theta_j$  est donc:

$$\frac{\partial MSE(\theta)}{\partial \theta_j} = \frac{2}{m} \sum_{i=1}^m x_{ij} (X_i \cdot \theta - y_i)$$

# Descente de gradient

Approche ordinaire

Le vecteur gradient de la Loss s'écrit comme suit :

$$\nabla_{\theta} MSE = \begin{bmatrix} \frac{\partial MSE(\theta)}{\partial \theta_0} \\ \frac{\partial MSE(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial MSE(\theta)}{\partial \theta_j} \\ \vdots \\ \frac{\partial MSE(\theta)}{\partial \theta_n} \end{bmatrix}$$

La mise à jour des paramètres  $\theta$  s'effectue comme suit :

$$\theta \text{ (étape } k) = \theta \text{ (étape } k-1) - \eta \nabla_{\theta} MSE$$

# Descente de gradient

Approche stochastique

La descente de gradient ordinaire utilise à chaque itération la totalité du jeu de données d'apprentissage → calcul lent et descente régulière

La descente stochastique consiste à choisir au hasard une seule instance  $i$  à chaque itération. La dérivée partielle de la Loss par rapport à un paramètre  $\theta_j$  devient:  $\frac{\partial MSE(\theta)}{\partial \theta_j} = 2 \times x_{ij}(X_i \cdot \theta - y_i)$

Lorsque la fonction Loss à minimiser n'est pas convexe (présence de minima locaux), l'approche stochastique introduit « des sauts irréguliers » d'une itération à l'autre permettant « d'échapper » au piège du minimum local.



# Descente de gradient

Approche stochastique

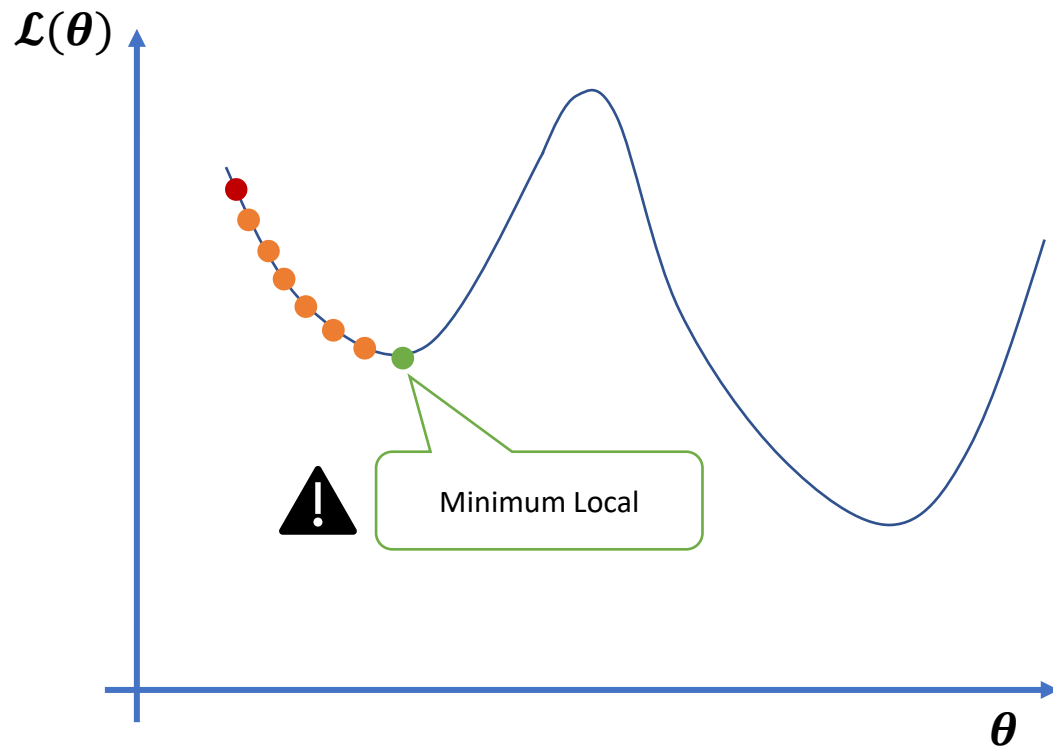


Illustration de l'approche classique en présence de minima locaux

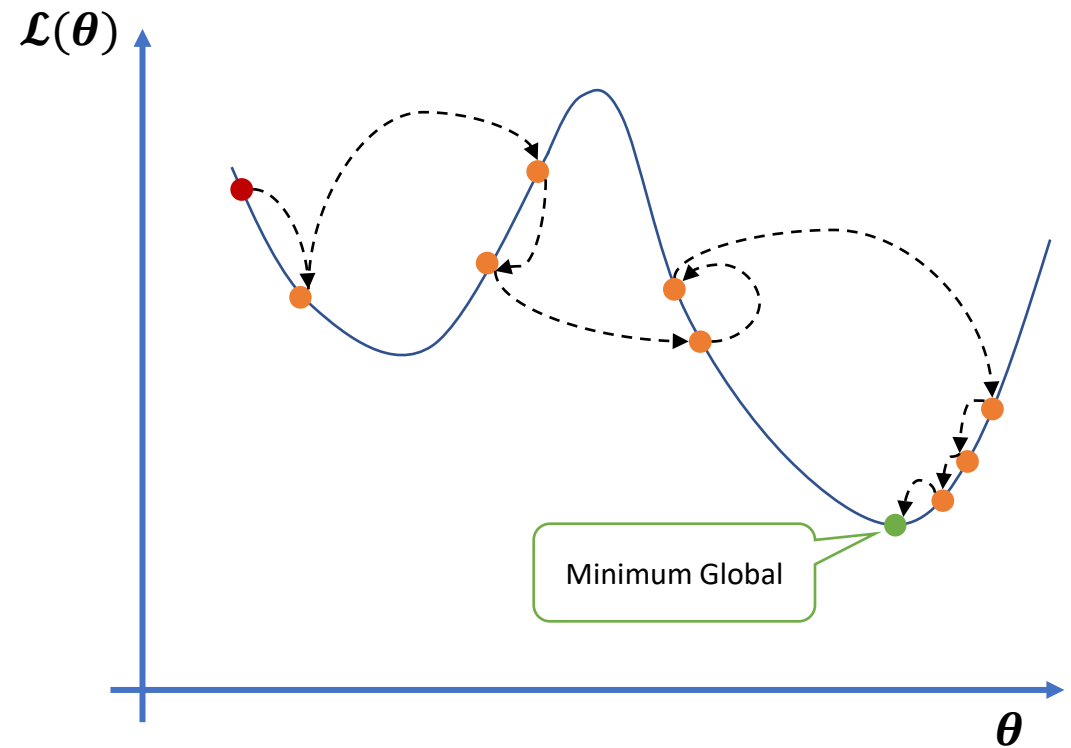


Illustration de l'approche stochastique en présence de minima locaux

# Descente de gradient

## Approche stochastique

- Sans critère d'arrêt, l'approche stochastique continue l'exploration indéfiniment même si le minimum global est visité.
- Une astuce consiste à fixer un nombre d'itérations (epochs) et à réduire progressivement le pas d'apprentissage pour minimiser dans un premier temps le risque de converger vers un minimum local puis d'accélérer la convergence vers un minimum « espéré » global dans un deuxième temps.

# Descente de gradient

Approche par mini-batch

La descente de gradient par mini-batch utilise un sous-ensemble de données d'apprentissage tiré aléatoirement à chaque itération. Cette approche combine les avantages des deux approches stochastique et ordinaire en :

- Minimisant le risque de converger vers un minimum local
- Assurant une convergence plus régulière que l'approche stochastique
- Tirant profit de la puissance de calcul matriciel de l'approche ordinaire

# Descente de gradient

Et sur Python ?

## Descente de gradient ordinaire MSE

```
from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit([[0, 0], [1, 1], [2, 2]], [1, 2, 3])
```

LinearRegression()

## Descente de gradient stochastique MSE

```
from sklearn.linear_model import SGDRegressor
reg = SGDRegressor(fit_intercept=True, learning_rate="constant", eta0=0.001, max_iter=100000)
reg.fit([[0, 0], [1, 1], [2, 2]], [1, 2, 3])
```

SGDRegressor(eta0=0.001, learning\_rate='constant', max\_iter=100000)

# Descente de gradient

Et sur Python ?

## Descente de gradient ordinaire MSE

```
from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit([[0, 0], [1, 1], [2, 2]], [1, 2, 3])
```

```
LinearRegression()
```

```
reg.intercept_
```

```
1.0000000000000002
```

```
reg.coef_
```

```
array([0.5, 0.5])
```

```
reg.predict([[3,4]])
```

```
array([4.5])
```

## Descente de gradient stochastique MSE

```
from sklearn.linear_model import SGDRegressor
reg = SGDRegressor(fit_intercept=True, learning_rate="constant", eta0=0.001, max_iter=100000)
reg.fit([[0, 0], [1, 1], [2, 2]], [1, 2, 3])
```

```
SGDRegressor(eta0=0.001, learning_rate='constant', max_iter=100000)
```

```
reg.intercept_
```

```
array([0.48806976])
```

```
reg.coef_
```

```
array([0.5703475, 0.5703475])
```

```
reg.predict([[3,4]])
```

```
array([4.48050224])
```

# Régularisation

## Principe

La régularisation est une méthode utilisée pour limiter les risques de surapprentissage en réduisant la complexité du modèle en phase d'apprentissage.

Le principe général de la régularisation consiste à orienter le modèle vers les formes les plus simples. Dans le cas de la régression linéaire, cela se traduit par :

- Le maintien de valeur aussi petites que possibles pour les paramètres  $\theta_j$  avec  $j \geq 1$  → régularisation  $\ell_2$  ou **Ridge**
- L'élimination des variables les moins importantes en réduisant le nombre de paramètres  $\theta_j$  avec  $j \geq 1$  non-nuls → régularisation  $\ell_1$  ou **Lasso**
- La combinaison des deux stratégies (ridge et lasso) → régularisation **Elastic Net**

# Régularisation

## Ridge

L'idée est d'ajouter une pénalité à la Loss pour maintenir les valeurs des coefficients  $\theta_j$  avec  $j \geq 1$  aussi petites que possible.

$$\mathcal{L}(\theta) = \text{MSE}(\theta) + \alpha \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

$\sum_{j=1}^n \theta_j^2$  correspond au carré de la norme  $\ell_2$  du vecteur  $w = [\theta_1, \dots, \theta_j, \dots, \theta_n]$  notée  $\|w\|_2$

La régularisation Ridge est sensible à l'échelle des variables ➔ normalisation des données

# Régularisation

## Lasso

L'idée est d'ajouter une pénalité à la Loss pour limiter le nombre de coefficients  $\theta_j$  avec  $j \geq 1$  non-nuls.

$$\mathcal{L}(\theta) = MSE(\theta) + \alpha \sum_{j=1}^n |\theta_j|$$

$\sum_{j=1}^n |\theta_j|$  correspond à la norme  $\ell_1$  du vecteur  $w = [\theta_1, \dots, \theta_j, \dots, \theta_n]$  notée  $\|w\|_1$



# Régularisation

## Elastic net

L'idée est de combiner les pénalités  $\ell_1$  et  $\ell_2$  et de doser l'intensité relative de chacune des deux régularisation à travers un paramètre  $r$

$$\mathcal{L}(\theta) = MSE(\theta) + \alpha \left( \frac{1-r}{2} \sum_{j=1}^n \theta_j^2 + r \sum_{j=1}^n |\theta_j| \right)$$

# Modèle de régression polynomiale

Extension du modèle linéaire – Exemple à deux variables et degré 2

Un modèle linéaire à deux variables prend la forme suivante :

$$\begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ \dots & \dots & \dots \\ 1 & x_{i1} & x_{i2} \\ \dots & \dots & \dots \\ 1 & x_{m1} & x_{m2} \end{bmatrix} \times \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \dots \\ \hat{y}_i \\ \dots \\ \hat{y}_m \end{bmatrix}$$

Un modèle polynomial de degré 2 aura la forme suivante :

$$\begin{bmatrix} 1 & x_{11} & x_{12} & x_{11}^2 & x_{12}^2 & x_{11} \cdot x_{12} \\ 1 & x_{21} & x_{22} & x_{21}^2 & x_{22}^2 & x_{21} \cdot x_{22} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_{i1} & x_{i2} & x_{i1}^2 & x_{i2}^2 & x_{i1} \cdot x_{i2} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_{m1} & x_{m2} & x_{m1}^2 & x_{m2}^2 & x_{m1} \cdot x_{m2} \end{bmatrix} \times \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \dots \\ \hat{y}_i \\ \dots \\ \hat{y}_m \end{bmatrix}$$

# Modèle de régression polynomiale

Extension du modèle linéaire – Exemple à deux variables et degré 2

Equivalent à l'introduction de nouvelles variables dans un modèle linéaire:

$$\begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ 1 & x_{21} & x_{22} & x_{23} & x_{24} & x_{25} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_{i1} & x_{i2} & x_{i3} & x_{i4} & x_{i5} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_{m1} & x_{m2} & x_{m3} & x_{m4} & x_{m5} \end{bmatrix} \times \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \dots \\ \hat{y}_i \\ \dots \\ \hat{y}_m \end{bmatrix}$$

TP

Notebook Linear\_regression

# Introduction au Machine Learning en Python

Master SIREN – Parcours IRN

(3/3)

UE: Econométrie et science des données

---

**Mustapha Sali**

Maître de Conférences HDR - Université Paris Dauphine

[mustapha.Sali@dauphine.psl.eu](mailto:mustapha.Sali@dauphine.psl.eu)

# Plan

## Arbres de décision pour la régression

- Principe
- Algorithme CART
- Hyperparamètres et régularisation
- Exemple sur ScikitLearn

## Forêts aléatoires et apprentissage d'ensemble

- Principe
- Bagging, pasting et bootstrapping
- TP sur données California-Housing

# Arbres de décision pour la régression

# Principe

- Les **arbres de décision** sont utilisés à la fois pour des tâches de **régression** et des tâches de **classification**
- Le principe de base est de **séparer progressivement** le jeu de donnée d'entraînement **par rapport à un seuil** appliqué aux variables qui réduisent le plus « l'impureté ». La séparation génère des branches qui pointent sur des nœuds de décision.
- Dans le cas de la régression, **la réduction de l'impureté correspond à la réduction de l'écart type (ou la MSE) du label** obtenue par séparation. Il s'agit de séparer progressivement les données pour **constituer des ensembles homogènes** du point de vue du label.
- Les dernières branches de l'arbre pointent sur des **nœuds terminaux** appelés **feuilles qui donnent la prédiction**



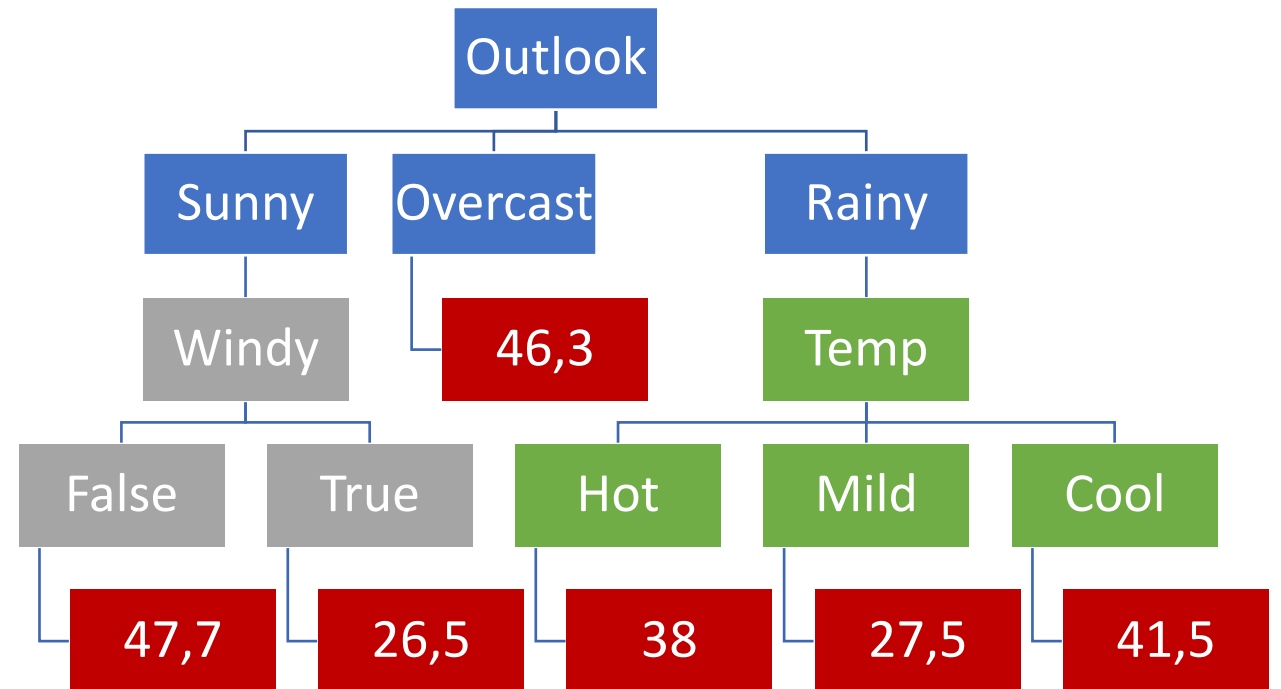
# Principe

Outlook	Temp	Humidity	Windy	Hours played
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	62
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30

# Principe

Outlook	Temp	Humidity	Windy	Hours played
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	62
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30

**Exemple d'arbre de décision**  
(Les feuilles sont représentées en rouge)



# Algorithme CART

## Classification And Regression Tree

Outlook	Écartype de Hours played	Effectif
Overcast	7,60	4
Rainy	7,53	5
Sunny	13,83	5
Std(Outlook,Hours played)	9,80	
Std(Hours played)	11,60	
Std réduction	1,80	

$$= \frac{4}{14} \times 7,60 + \frac{5}{14} \times 7,53 + \frac{5}{14} \times 13,83$$

# Algorithme CART

## Classification And Regression Tree

Outlook	Écartype de Hours played	Effectif
Overcast	7,60	4
Rainy	7,53	5
Sunny	13,83	5
Std(Outlook,Hours played)		
	9,80	
Std(Hours played)		
	11,60	
Std réduction		
	1,80	

Humidity	Écartype de Hours played	Effectif
High	11,92	7
Normal	10,91	7
Std(Outlook,Humidity)		
	11,42	
Std(Hours played)		
	11,60	
Std réduction		
	0,18	

Temps	Écartype de Hours played	Effectif
Cool	13,94	4
Hot	9,22	4
Mild	10,12	6
Std(Outlook,Temp)		
	10,95	
Std(Hours played)		
	11,60	
Std réduction		
	0,65	

Windy	Écartype de Hours played	Effectif
High	9,89	8
Normal	13,19	6
Std(Outlook,Windy)		
	11,30	
Std(Hours played)		
	11,60	
Std réduction		
	0,30	

# Algorithme CART

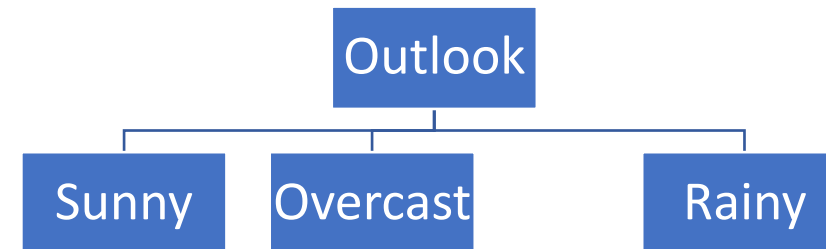
## Classification And Regression Tree

Outlook	Écartype de Hours played	Effectif
Overcast	7,60	4
Rainy	7,53	5
Sunny	13,83	5
Std(Outlook,Hours played)		
	9,80	
Std(Hours played)		
	11,60	
Std réduction		
	1,80	

Humidity	Écartype de Hours played	Effectif
High	11,92	7
Normal	10,91	7
Std(Outlook,Humidity)		
	11,42	
Std(Hours played)		
	11,60	
Std réduction		
	0,18	

Temps	Écartype de Hours played	Effectif
Cool	13,94	4
Hot	9,22	4
Mild	10,12	6
Std(Outlook,Temp)		
	10,95	
Std(Hours played)		
	11,60	
Std réduction		
	0,65	

Windy	Écartype de Hours played	Effectif
High	9,89	8
Normal	13,19	6
Std(Outlook,Windy)		
	11,30	
Std(Hours played)		
	11,60	
Std réduction		
	0,30	



# Algorithme CART

## Classification And Regression Tree

Variables à considérer

Outlook	Rainy	Outlook	Temp	Humidity	Windy	Hours played
		Rainy	Hot	High	False	26
		Rainy	Hot	High	True	30
		Rainy	Mild	High	False	36
		Rainy	Cool	Normal	False	38
		Rainy	Mild	Normal	True	48
	Overcast	Overcast	Hot	High	False	48
		Overcast	Cool	Normal	True	43
		Overcast	Mild	High	True	62
		Overcast	Hot	Normal	False	44
	Sunny	Sunny	Mild	High	False	46
		Sunny	Cool	Normal	False	62
		Sunny	Cool	Normal	True	23
		Sunny	Mild	Normal	False	48
		Sunny	Mild	High	True	30

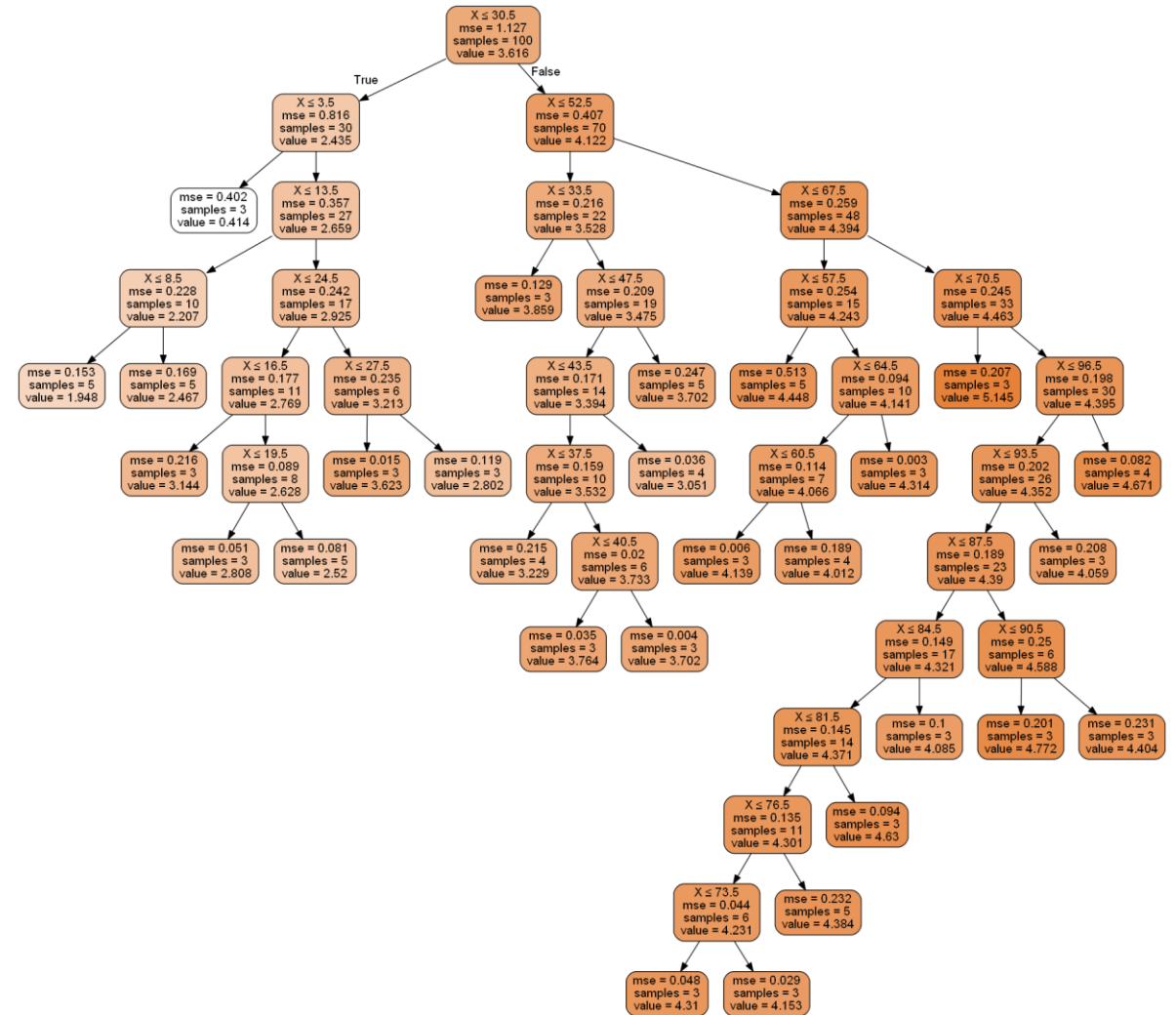
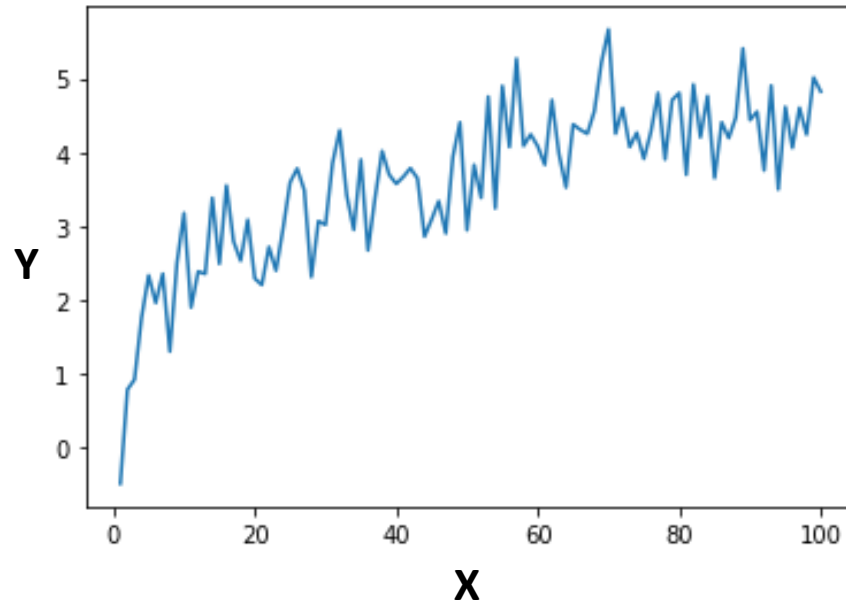
**CART est un algorithme glouton dont les critères d'arrêt sont les suivants:**

- Profondeur maximale de l'arbre
- Effectif minimum pour envisager une séparation
- Effectif minimum dans les feuilles
- Nombre maximum de feuilles

# Hyperparamètres et régularisation

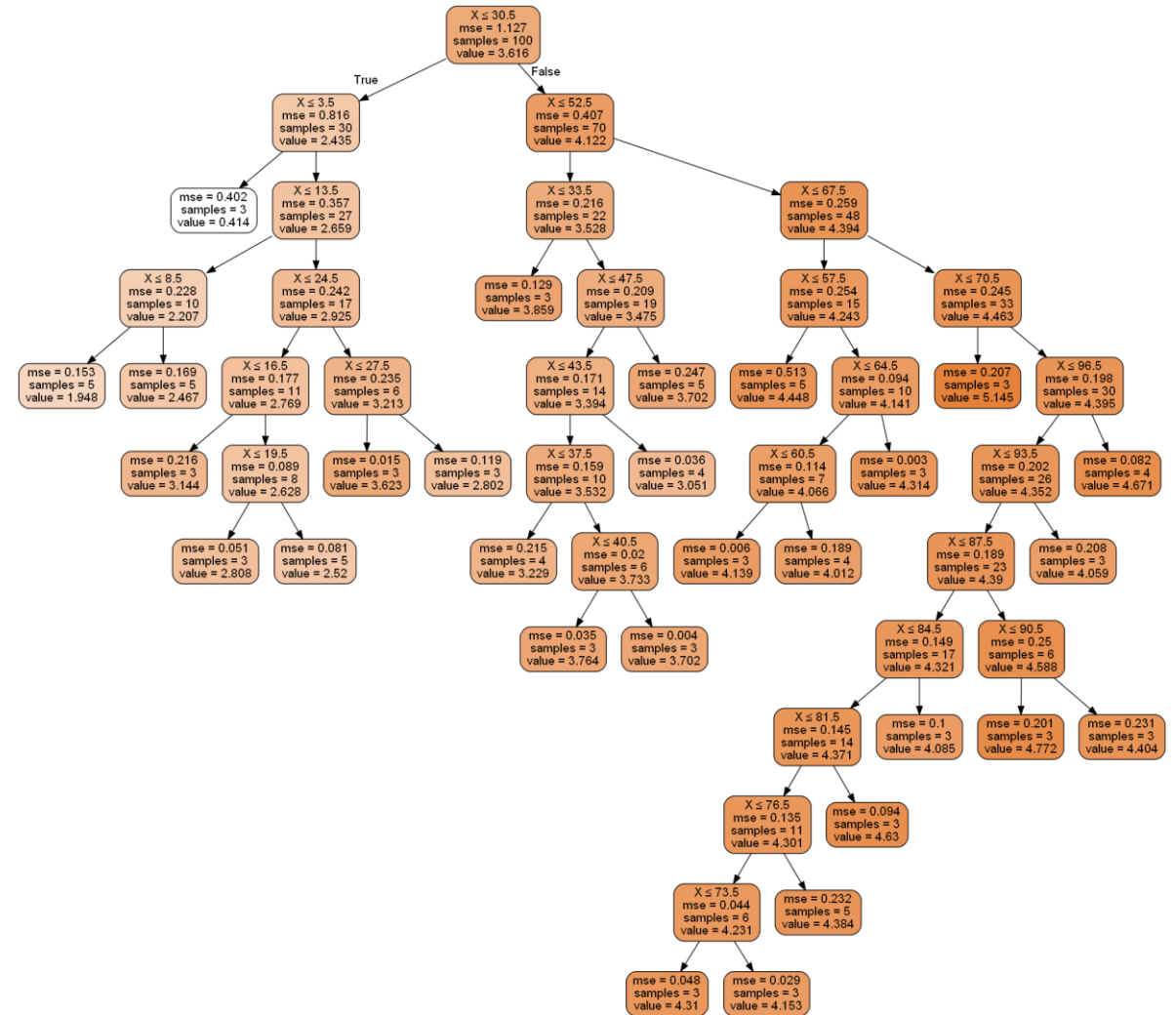
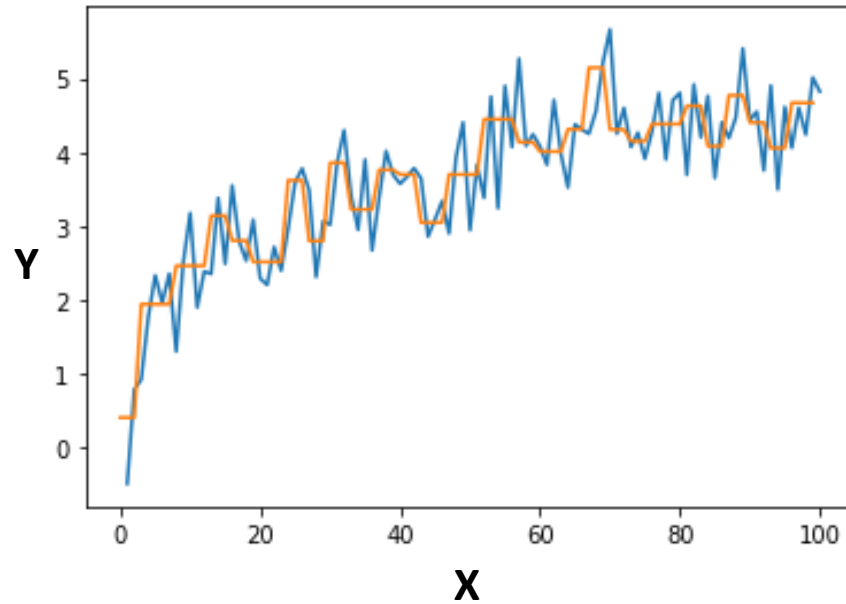
- Les arbres de décision sont des modèles non-paramétrique → **la valeur des hyperparamètres s'adapte aux données**
- Pour éviter le sur-apprentissage, il est important de **fixer à priori le degré d'adaptation du modèle aux donnée**
- La régularisation se fait à travers des limites imposées à certains paramètres. Ces limites correspondent aux **critères d'arrêt de l'algorithme CART**

# Exemple sur ScikitLearn





# Exemple sur ScikitLearn



Forêts aléatoires

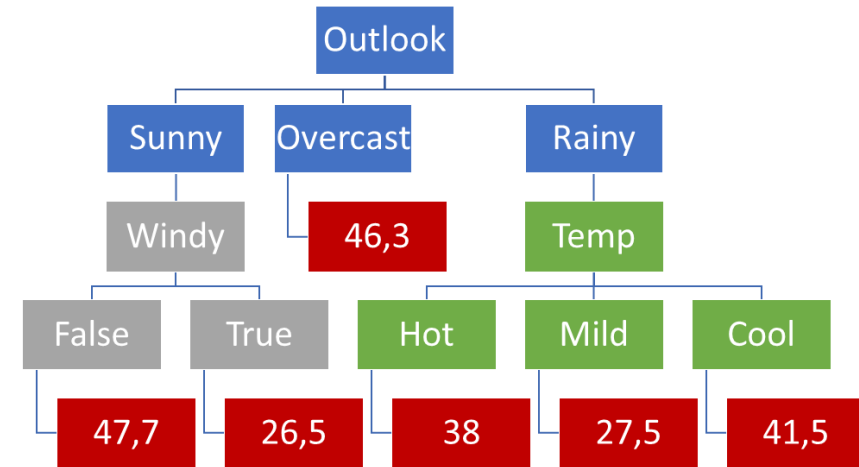
# Principe

- Technique **d'apprentissage d'ensemble** basée sur l'agrégation de plusieurs **arbres de décision** entraînés sur des sous-ensembles du jeu de données
- **Une forêt aléatoire est constituée d'arbres de décision** dont le nombre est fixé par un hyperparamètre
- L'idée centrale des forêts aléatoires est de **compenser les erreurs générées par des modèles de régression élémentaire**
- La compensation est d'autant plus **efficace** que les modèles de régression élémentaires sont relativement **indépendants**

# Principe

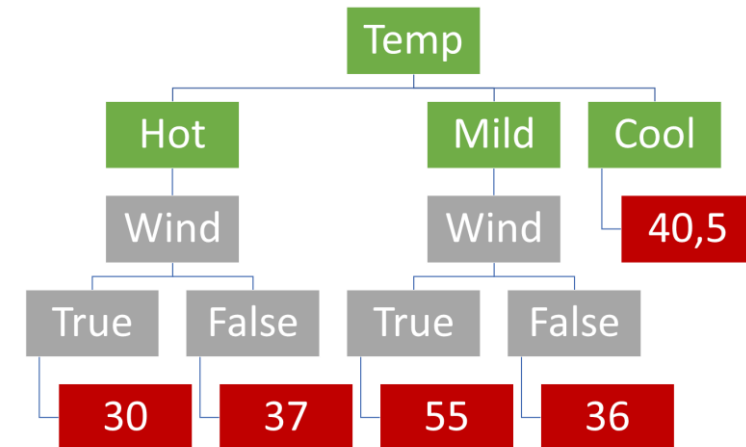
Outlook	Temp	Humidity	Windy	Hours played
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	62
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30

Arbre 1



Temp	Humidity	Windy	Hours played
Hot	High	False	26
Hot	High	True	30
Hot	High	False	48
Mild	High	False	46
Cool	Normal	False	62
Cool	Normal	True	23
Cool	Normal	True	43
Mild	High	False	36
Cool	Normal	False	38

Arbre 2



# Principe

Outlook	Temp	Humidity	Windy	Hours played	Pred Arbre 1	Pred Arbre 2	Pred Moyenne
Rainy	Hot	High	False	26	38	37	37,5
Rainy	Hot	High	True	30	38	30	34
Overcast	Hot	High	False	48	46,3	37	41,65
Sunny	Mild	High	False	46	47,7	36	41,85
Sunny	Cool	Normal	False	62	47,7	40,5	44,1
Sunny	Cool	Normal	True	23	26,5	40,5	33,5
Overcast	Cool	Normal	True	43	46,3	40,5	43,4
Rainy	Mild	High	False	36	27,5	36	31,75
Rainy	Cool	Normal	False	38	41,5	40,5	41
Sunny	Mild	Normal	False	48	47,7	36	41,85
Rainy	Mild	Normal	True	48	27,5	55	41,25
Overcast	Mild	High	True	62	46,3	55	50,65
Overcast	Hot	Normal	False	44	46,3	37	41,65
Sunny	Mild	High	True	30	26,5	55	40,75
RMSE					9,30	12,07	8,44

# Bagging, pasting et bootstrapping

Le **bagging** et le **pasting** sont deux techniques utilisées pour obtenir des **modèles de régression diversifiés**

- Le **bagging** consiste à **choisir aléatoirement un sous-ensemble de données d'entraînement avec remise**
- La **pasting** consiste à **choisir aléatoirement un sous-ensemble de données d'entraînement sans remise**

Le **bootstrapping** consiste à **sélectionner aléatoirement un sous-ensemble de variables** à partir desquelles réaliser l'entraînement.

**L'échantillonnage conjoint des observations et des variables** constitue la **méthode d'ensemble des parcelles aléatoires**

TP

Notebook DT\_RF\_regressor