

**Intelligence Artificielle**

**Année universitaire : 2025 / 2026**

**Professeur Khalid BENABBES**

# Plan

## 1. Introduction à l'Intelligence Artificielle

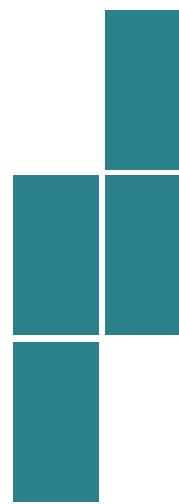
- Définition et concepts fondamentaux de l'Intelligence Artificielle
- Histoire de l'IA et ses développements récents
- Enjeux, éthique et perspectives de l'IA.

## 2. Machine Learning: Apprentissage Automatique

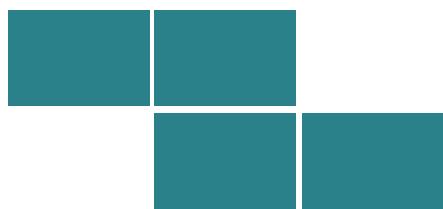
- Distinction entre IA programmée explicitement et IA
- Cycle de vie d'un modèle ML : collecte, nettoyage, entraînement, évaluation, déploiement.
- Types de machine Learning (Apprentissage)
- Algorithmes d'apprentissage supervisé
- Apprentissage non supervisé
- Évaluation des modèles

## 4. Deep learning: Apprentissage profond

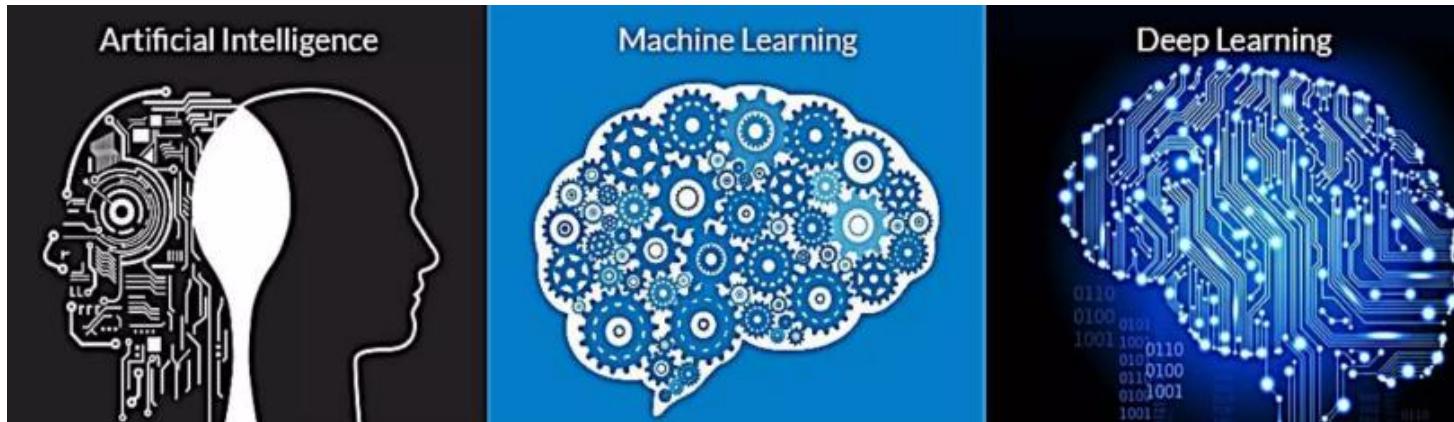
- Fondamentaux du Deep Learning
- Architecture de Réseaux de neurones
- Les hyperparamètres de réseaux de neurone
- Présentation de TensorFlow/Keras
- Comprendre et préparer les données
- Entraînement et déploiement du modèle MLP
- Application de l'IA en Santé



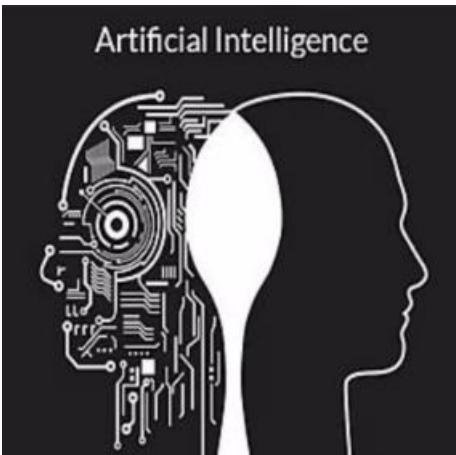
# Chapitre 1 : Langage Python pour la science des données



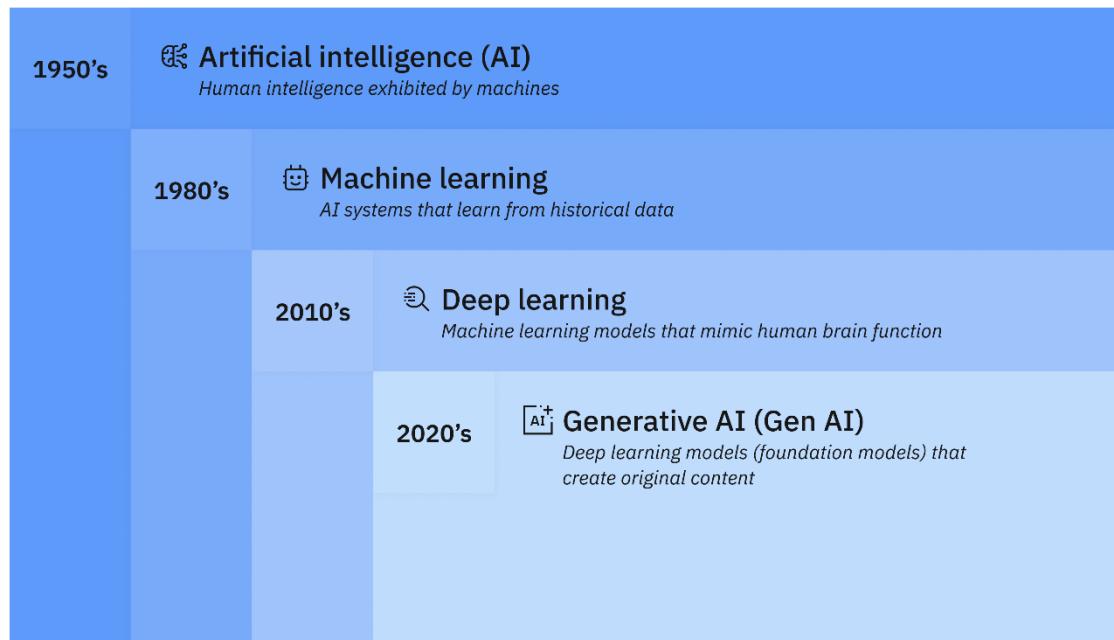
# IA



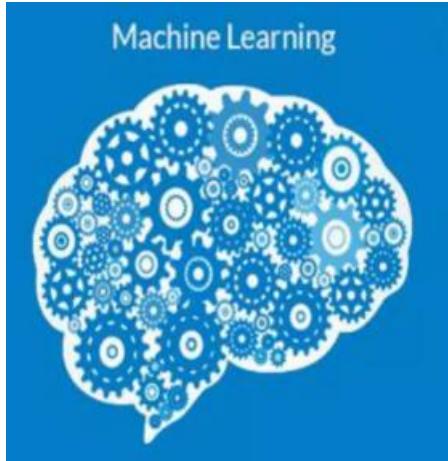
## ● Qu'est ce que l'intelligence Artificielle?



- **L'Intelligence Artificielle (IA)** est un ensemble **d'algorithmes** et de **règles utilisés** pour résoudre un **problème de façon** autonome.
- Domaine de l'informatique visant à créer des systèmes capables de simuler l'intelligence humaine.
- Elle se compose, entre autres, de **Machine Learning (ML)** et de **Deep Learning (DL)**.



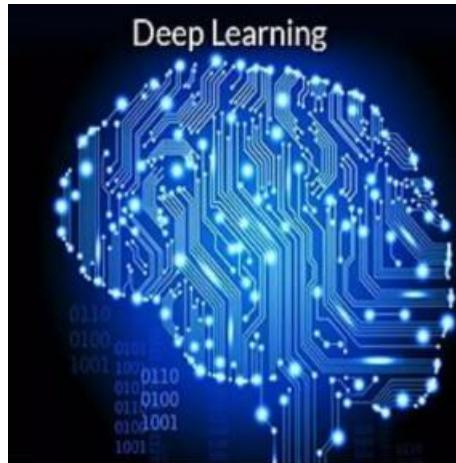
- **Qu'est ce que le Machine Learning (ML)?**



Le machine Learning (apprentissage automatique) regroupe des algorithmes capables de résoudre des problèmes ou de prédire des événements en apprenant à partir de données, sans règles explicitement programmées.



- Qu'est ce que le Deep Learning (DL)?



**Le Deep Learning (apprentissage profond) :**

- Sous-domaine du machine Learning
- S'inspire du fonctionnement des neurones biologiques

**Différences avec les approches classiques :**

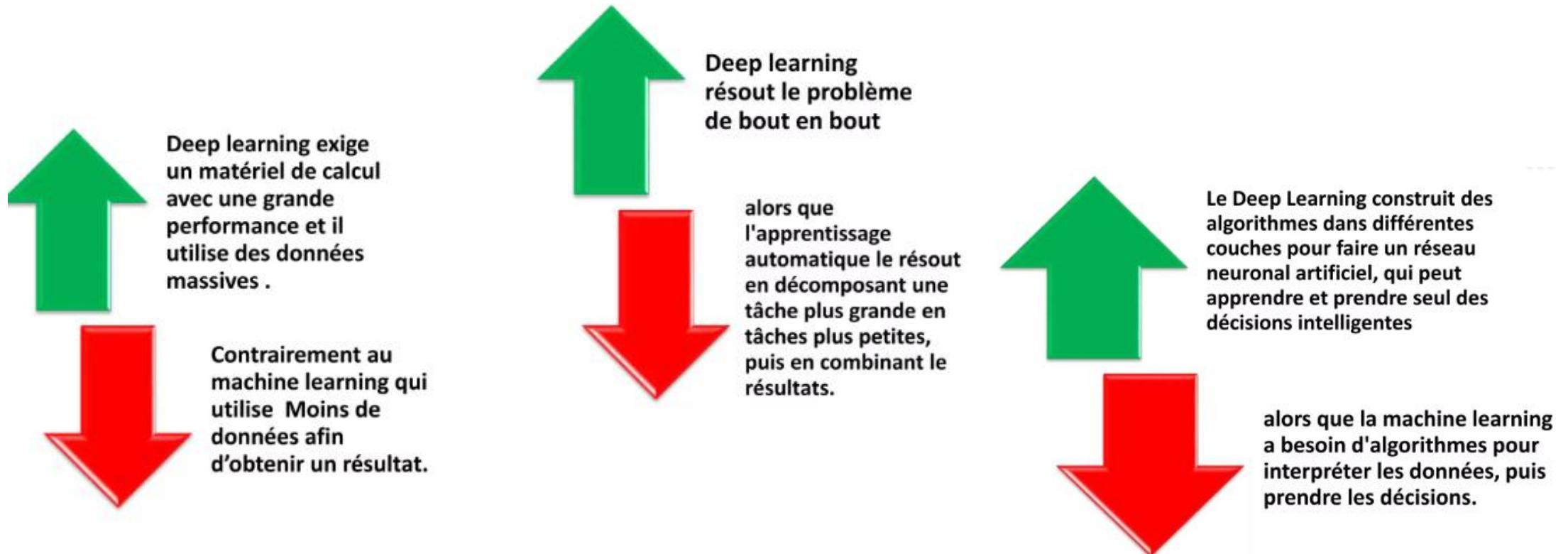
Approches classiques :

- Requièrent une extraction manuelle des caractéristiques (features)

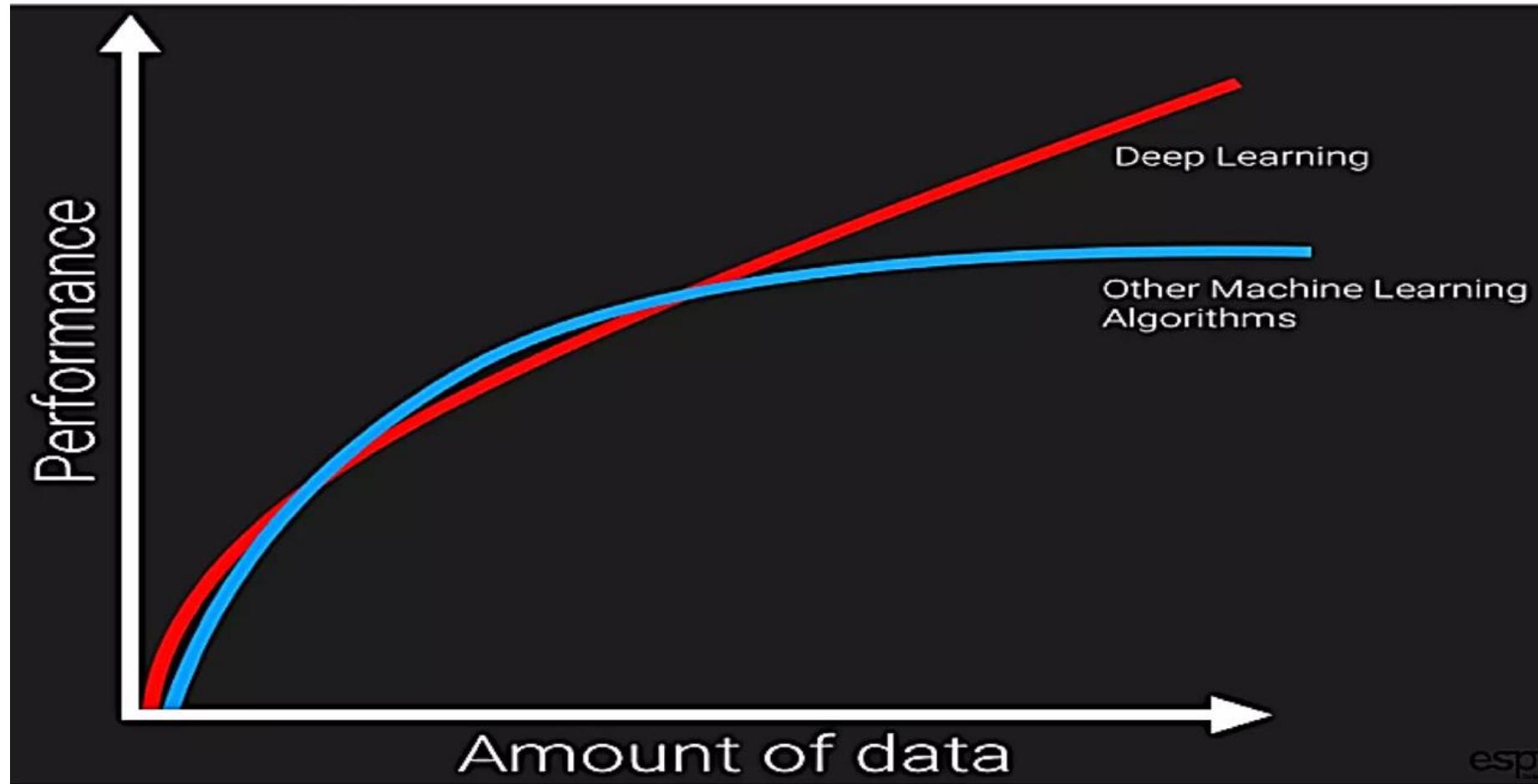
Deep Learning :

- Utilise des réseaux de neurones artificiels multicouches
- Apprend automatiquement des représentations complexes des données
- Permet une analyse plus approfondie
- Ne nécessite pas d'intervention humaine explicite

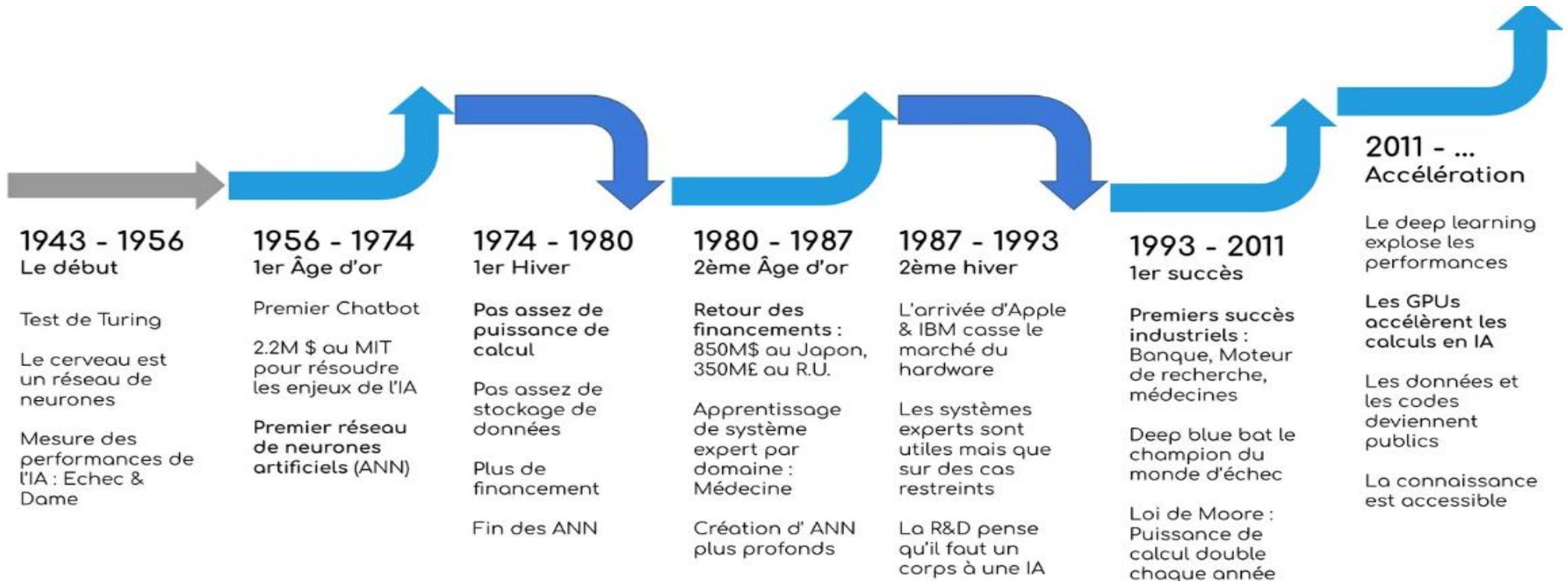
- Machine Learning VS Deep Learning



# Machine Learning et Deep Learning en utilisant python



## ● Histoire de l'IA et ses développements récents



- Histoire de l'IA et ses développements récents

Période	Étape clé	Innovations techniques	Contexte machine
<b>1940 – 1956</b>	Fondations théoriques	Test de Turing, machine abstraite	Mainframes primitifs
<b>1950 – 1974</b>	Premiers programmes & ML	Jeux, apprentissage rudimentaire	Ordinateurs programmables
<b>1980 – 1990</b>	Systèmes experts & hiver	Raisonnement symbolique	Machines limitées
<b>2000 – 2010</b>	Deep Learning	Réseaux neuronaux profonds	GPU, big data
<b>2010 – Aujourd'hui</b>	IA générative & usages massifs	GAN, transformateurs, IA grand public	Cloud, supercalculateurs distribués

Synthèse visuelle

## ● Applications majeures de l'IA

**Santé & médecine** : diagnostic assisté par IA, imagerie médicale, découverte de médicaments

**Vision & reconnaissance d'images** : voitures autonomes, surveillance, contrôle qualité industrielle

**Langage & traitement du texte** : chatbots, traduction automatique, rédaction assistée (ex. GPT)

**Robotique & systèmes autonomes** : drones, robots de service, robots industriels

**Finance & assurance** : détection de fraude, scoring de crédit, trading algorithmique

**Secteur des services & marketing** : recommandation personnalisée, ciblage publicitaire, assistants virtuels

## ● Le lexique de l'intelligence artificielle

### Modèle de langage

- Modèle mathématique qui prédit la **probabilité d'une séquence de mots**.
- Applications : traduction automatique, reconnaissance vocale, synthèse vocale, chatbots.

### Prompt

- Instruction donnée à une IA générative.
  - Détermine la qualité et la pertinence des réponses générées.
- 👉 Un bon prompt = un bon résultat ✅

### Réseaux de neurones artificiels

- Modèle inspiré du cerveau humain 🧠.
- Composé de « neurones » interconnectés capables d'**apprendre des représentations complexes** à partir de données.

## ● Le lexique de l'intelligence artificielle

### Machine Learning (Apprentissage automatique)

- Ensemble de techniques permettant à une IA d'apprendre automatiquement à partir de données et d'améliorer ses performances.

### Deep Learning (Apprentissage profond)

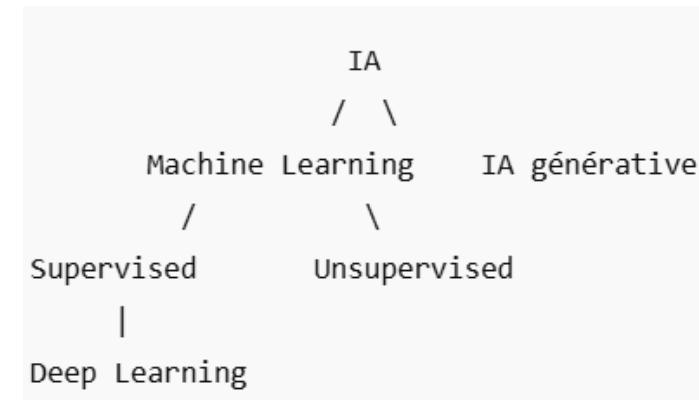
- Sous-discipline du Machine Learning.
- Utilise des **réseaux de neurones profonds** pour apprendre des données massives.

### Supervised Learning (Apprentissage supervisé)

- Le modèle est entraîné à partir de **données étiquetées** (entrées + sorties connues).
- Ex. : classification d'images, traduction, prédiction.

### Unsupervised Learning (Apprentissage non supervisé)

- Le modèle apprend **sans sorties prédefinies** à repérer des structures dans les données brutes.
- Ex. : clustering, réduction de dimension.



- **Enjeux, éthique et perspectives de l'IA.**

## Enjeux

■ **Enjeux** = Ce qu'on veut atteindre ou les opportunités à saisir.

### 1. Enjeux économiques et industriels

- **Accélération de la productivité** dans tous les secteurs : santé, agriculture, industrie, logistique, finance...
- **Création de nouveaux métiers** autour de la donnée, de la cybersécurité et du développement IA.
- **Compétitivité mondiale** : course stratégique entre puissances économiques (USA, UE, Chine...).

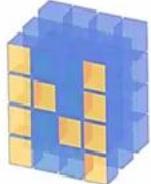
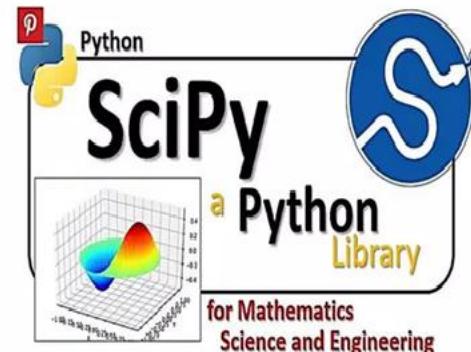
### 2. Enjeux sociaux et sociétaux

- **Transformation de l'éducation** : personnalisation des apprentissages, outils intelligents, nouveaux modèles pédagogiques.
- **Amélioration des services publics** (santé, mobilité, urbanisme).
- Risques de **fracture numérique** → inégalités d'accès aux technologies.
- **Impact sur l'emploi** : automatisation de tâches, requalification des compétences, disparition/émergence de métiers.

### 3. Enjeux éthiques et environnementaux

- **Transparence** et confiance dans les décisions des systèmes IA.
- **Biais algorithmiques** et discriminations potentielles.
- **Empreinte écologique** importante (consommation énergétique des modèles massifs).
- **Cadres légaux et réglementaires** encore en construction à l'échelle mondiale.

# Machine Learning en utilisant Python



NumPy

FOR MACHINE LEARNING



Machine Learning with Scikit-Learn



# Deep Learning en utilisant python



ANACONDA®



TensorFlow



# Machine Learning en utilisant python

- Installation de l'environnement de travail



**ANACONDA®**

<https://www.anaconda.com/download/success>

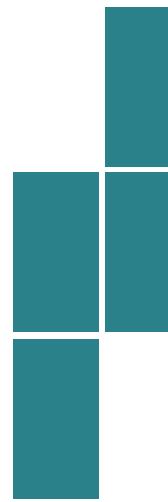


Visual Studio Code

<https://code.visualstudio.com/download>

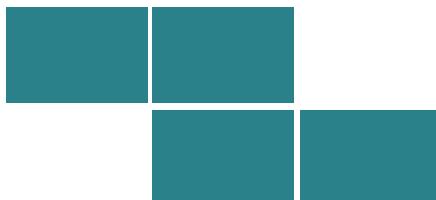


- Installation de l'environnement (**Anaconda, Jupyter, Notebook, Vscode**).
- Langage principal : Python
- Environnement virtuel : venv, conda



# Chapitre 1: Machine Learning

- ✓ Découvrir le machine Learning
- ✓ Domaines d'application de machine Learning
- ✓ Types de machine Learning (Apprentissage)
- ✓ Présentation de scikit-learn



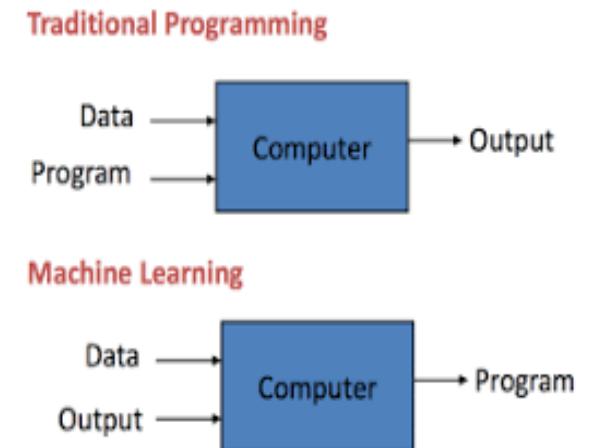
# Machine Learning

## ● Découvrir le machine Learning

- Un domaine d'étude qui donne aux ordinateurs la capacité d'apprendre sans être explicitement programmés.

En quoi est-ce différent de la programmation traditionnelle ?

- Programmation traditionnelle : On fournit les entrées, la logique du programme, et on exécute le programme pour obtenir une sortie.
- Machine Learning : On fournit les entrées et les sorties, et on entraîne la machine pour qu'elle crée sa propre logique, qui est ensuite évaluée lors des tests.

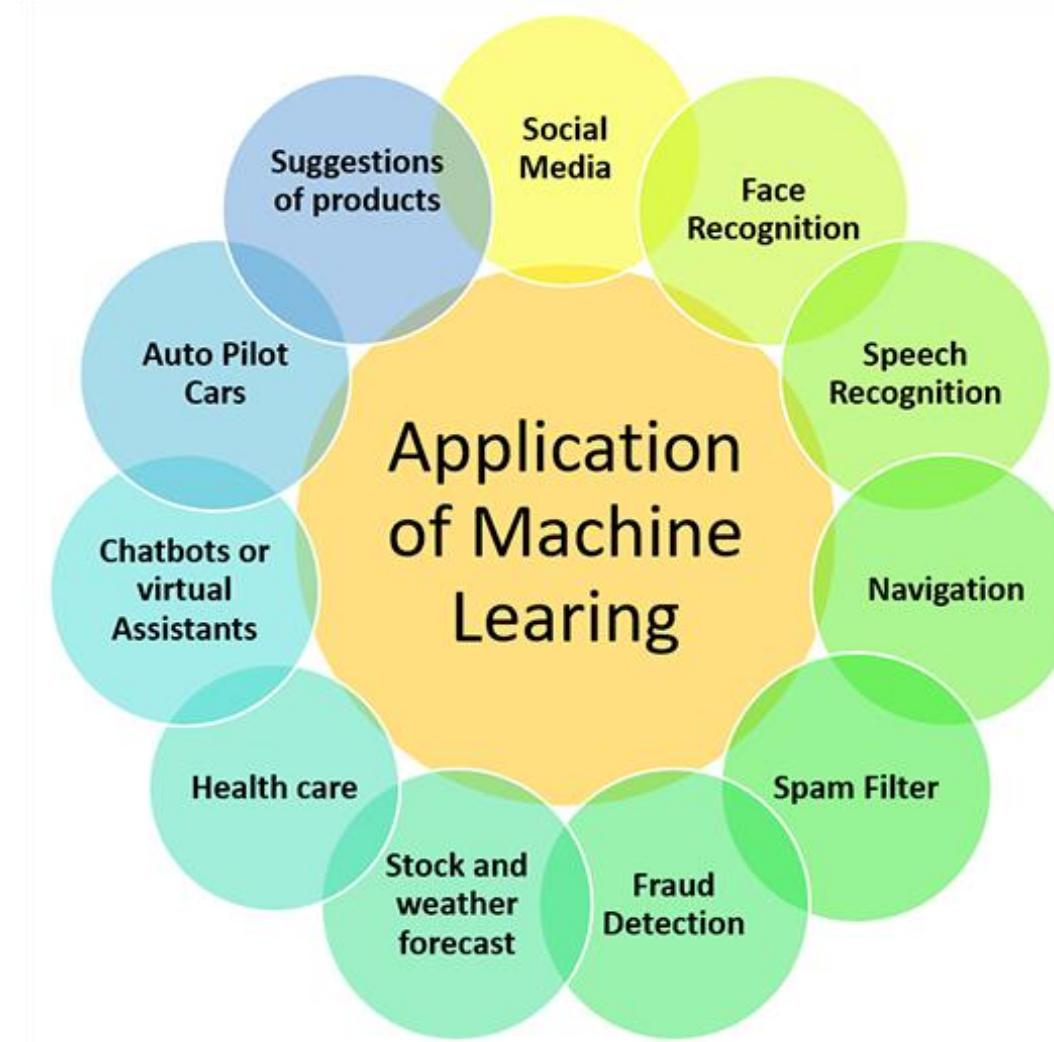


## ● Terminologies à connaître avant de commencer le Machine Learning

- **Modèle** : Une représentation spécifique apprise à partir des données en appliquant un algorithme de machine learning. Un modèle est aussi appelé hypothèse.
- **Feature (Caractéristique)** : Une propriété mesurable individuelle de nos données. Un ensemble de caractéristiques quantitatives ou qualitatives peut être décrit par un vecteur de caractéristiques. Par exemple, pour prédire un fruit, les caractéristiques peuvent être la couleur, l'odeur, le goût, etc.
- **Target (Label)** : La variable cible ou étiquette est la valeur à prédire par notre modèle. Par exemple, pour le cas des fruits, l'étiquette serait le nom du fruit comme pomme, orange, banane, etc.
- **Entraînement** : L'idée est de fournir un ensemble d'entrées (caractéristiques) et leurs sorties attendues (étiquettes). Après l'entraînement, nous obtenons un modèle (hypothèse) qui pourra mapper de nouvelles données à l'une des catégories apprises
- **Prédiction** : Une fois le modèle prêt, il peut être alimenté avec de nouvelles entrées pour fournir une sortie prédite (étiquette).

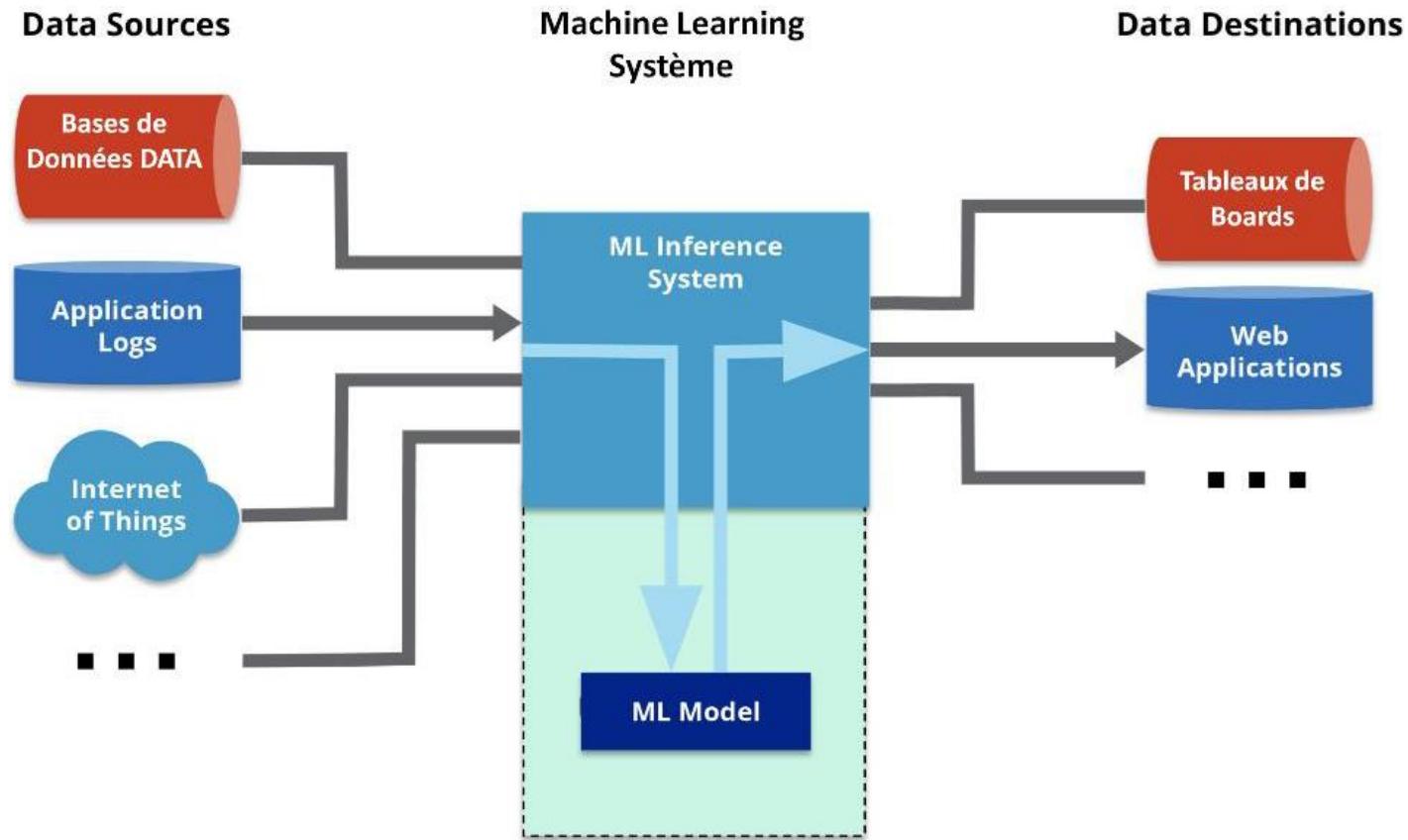
# Machine Learning

## ● Domaines d'application de machine Learning



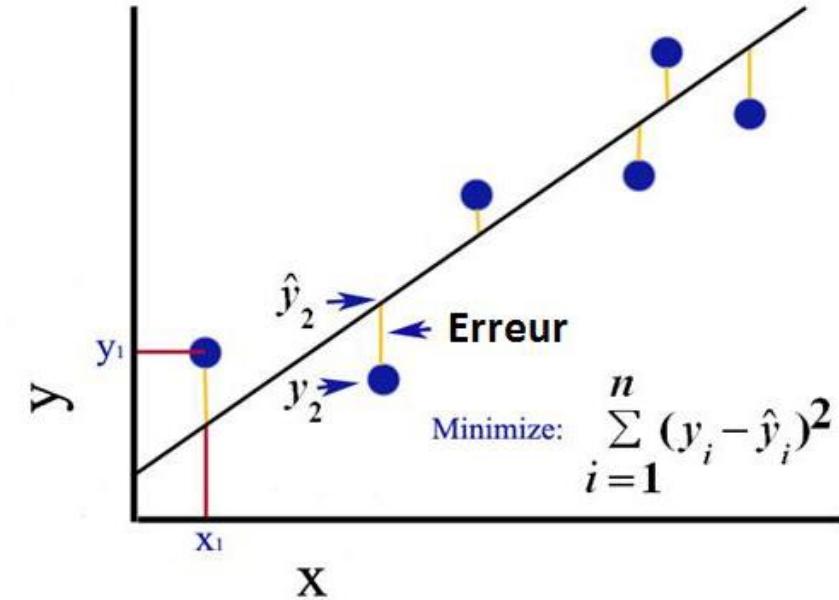
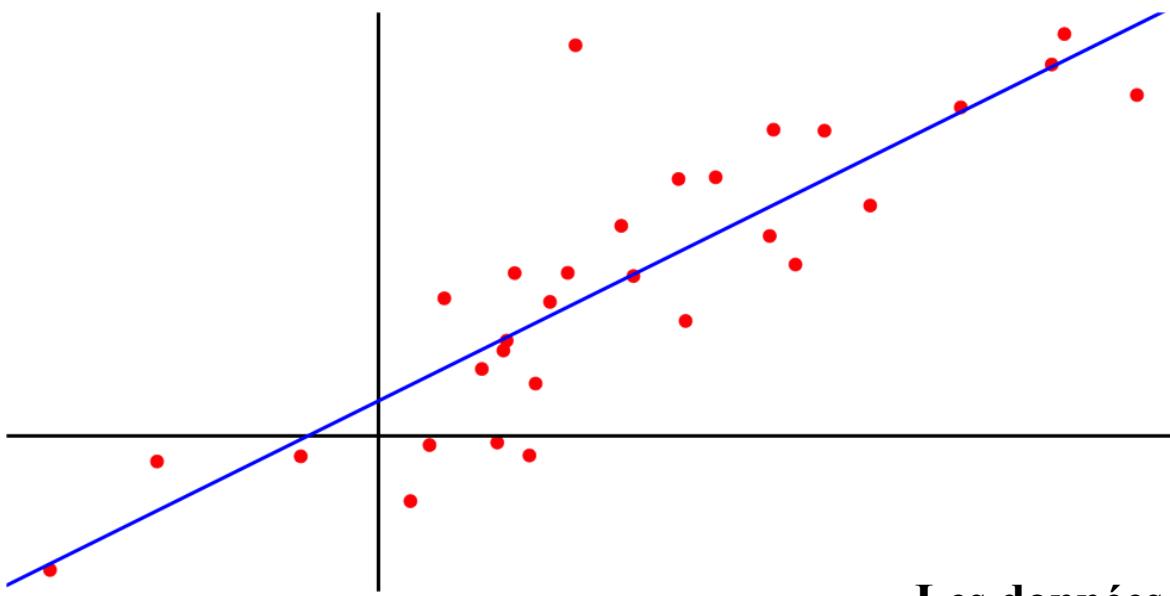
# Machine Learning

## Architecture Usuelle de Machine Learning



# Appliquer la régression et la classification

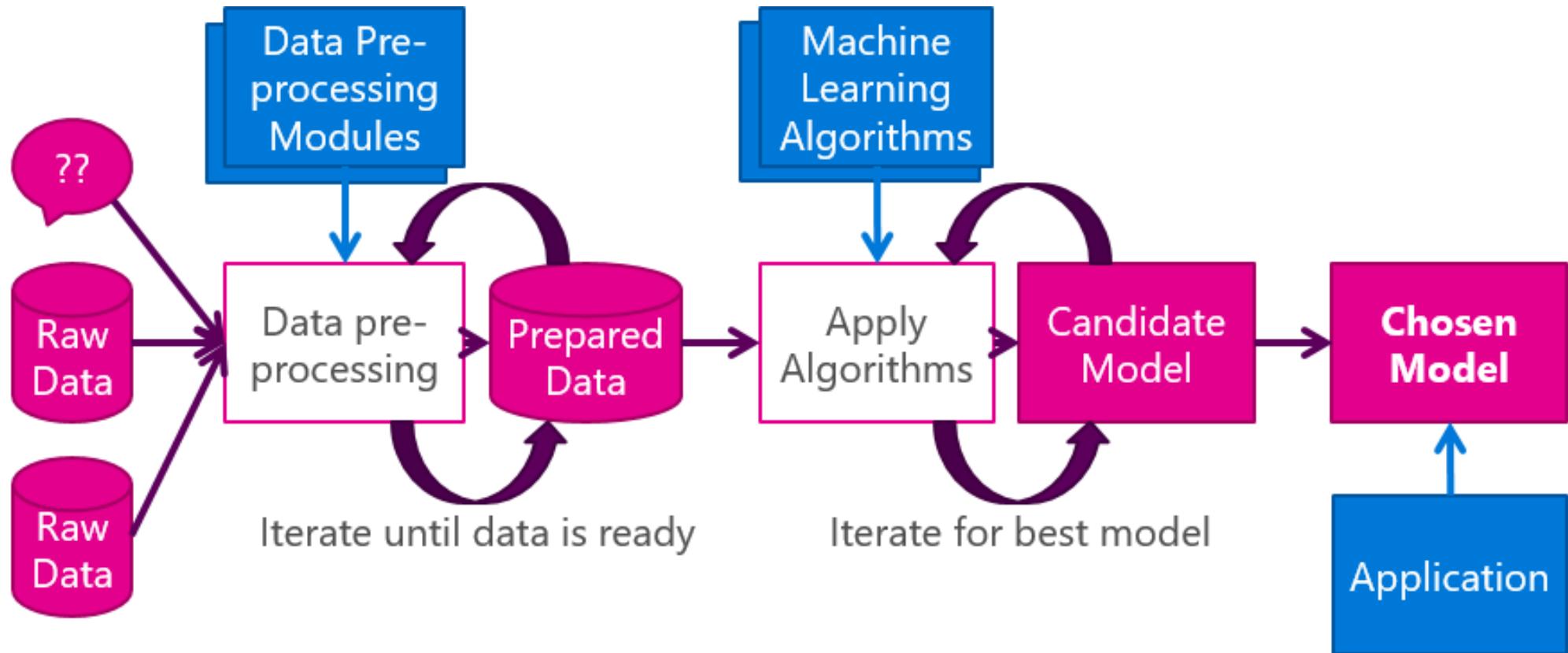
## Apprentissage



Les données sont modélisées :  $y = ax + b$

# Appliquer la régression et la classification

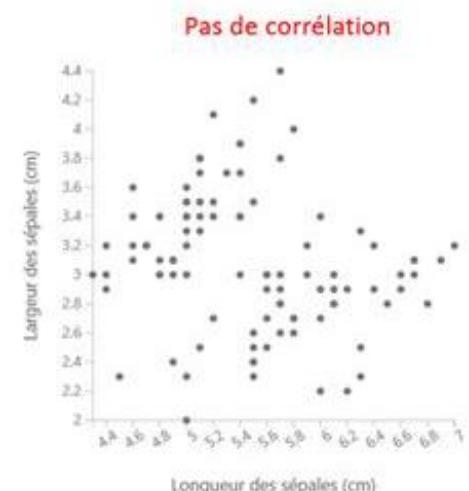
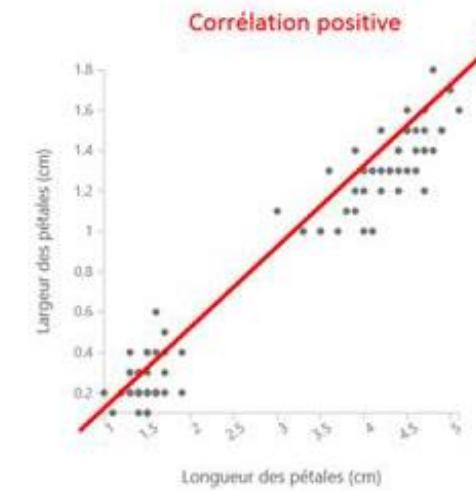
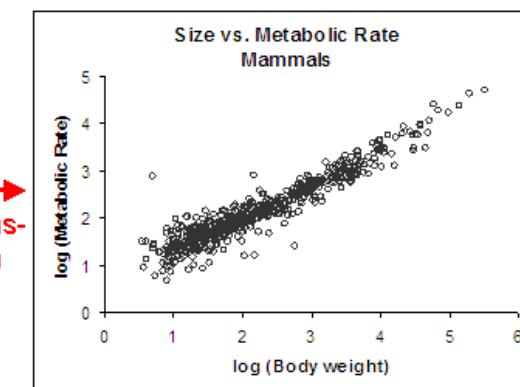
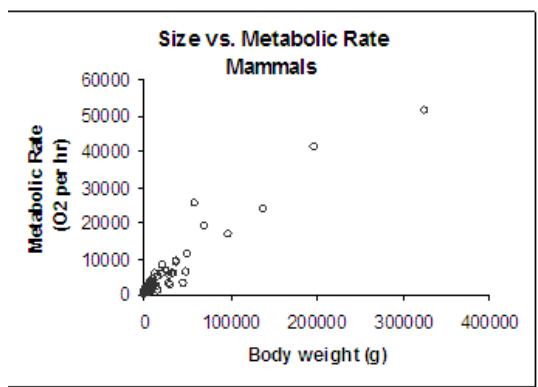
## Machine Learning Pipeline



# Appliquer la régression et la classification

## Transformation des données -Exemple

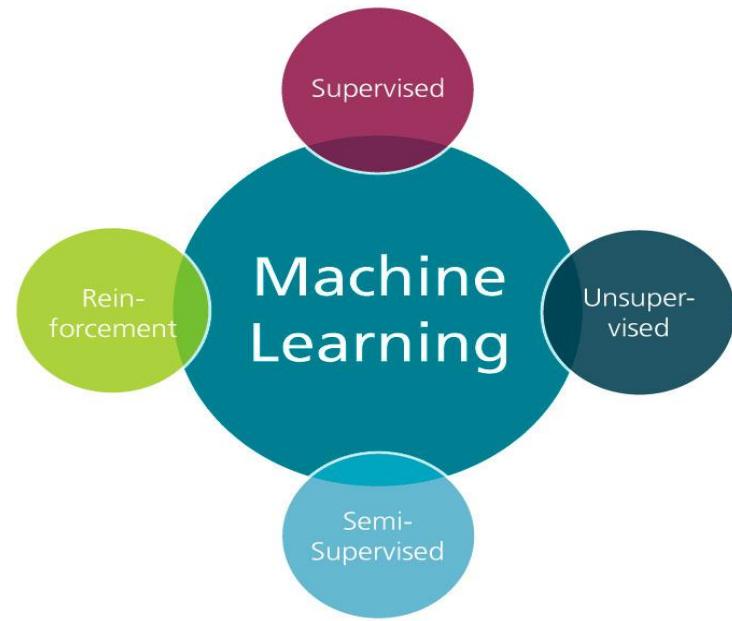
Longueur des sépales (cm)	Largeur des sépales (cm)	Longueur des pétales (cm)	Largeur des pétales (cm)	Prédiction
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa



# Machine Learning

## ● Types de machine Learning (Apprentissage)

- Apprentissage supervisé
- Apprentissage non supervisé
- Apprentissage semi-supervisé
- Apprentissage par renforcement



# Machine Learning

## ● Types de machine Learning (Apprentissage)

### □ Apprentissage supervisé

- L'apprentissage supervisé se produit lorsque le modèle est entraîné sur un ensemble de données étiquetées.
- Un ensemble de données étiqueté contient à la fois les paramètres d'entrée et de sortie.
- Dans ce type d'apprentissage, les ensembles d'entraînement et de validation sont étiquetés comme le montre la figure ci-dessous.

User ID	Gender	Age	Salary	Purchased	Temperature	Pressure	Relative Humidity	Wind Direction	Wind Speed
15624510	Male	19	19000	0	10.69261758	986.882019	54.19337313	195.7150879	3.278597116
15810544	Male	35	20000	1	13.59184184	987.8729248	48.0648859	189.2951202	2.909167767
15668575	Female	26	43000	0	17.70494885	988.1119385	39.11965597	192.9273834	2.973036289
15603246	Female	27	57000	0	20.95430404	987.8500366	30.66273218	202.0752869	2.965289593
15804002	Male	19	76000	1	22.9278274	987.2833862	26.06723423	210.6589203	2.798230886
15728773	Male	27	58000	1	24.04233986	986.2907104	23.46918024	221.1188507	2.627005816
15598044	Female	27	84000	0	24.41475295	985.2338867	22.25082295	233.7911987	2.448749781
15694829	Female	32	150000	1	23.93361956	984.8914795	22.35178837	244.3504333	2.454271793
15600575	Male	25	33000	1	22.68800023	984.8461304	23.7538641	253.0864716	2.418341875
15727311	Female	35	65000	0	20.56425726	984.8380737	27.07867944	264.5071106	2.318677425
15570769	Female	26	80000	1	17.76400389	985.4262085	33.54900114	280.7827454	2.343950987
15606274	Female	26	52000	0	11.35600745	986.0306607	63.78138843	20.15406236	1.550191426

# Machine Learning

## ● Types de machine Learning (Apprentissage)

### □ Types d'apprentissage supervisé

1. **Classification** : - Une tâche d'apprentissage supervisée où la sortie à des étiquettes.

- La classification consiste à prédire une catégorie ou une étiquette discrète..

- **Type de sortie** : Une variable catégorielle (ex. : "oui" ou "non", "chat" ou "chien").

- **Exemples** :

- **Filtrage des e-mails** : Spam ou Non-Spam.
- **Reconnaissance d'images** : Identifier un véhicule comme "voiture" ou "moto".
- **Diagnostic médical** : Malade ou Non-Malade.

2. **Régression** : Une tâche d'apprentissage supervisé où la sortie a une valeur continue (valeur réelle).

- **Exemples** :

- **Prix de l'immobilier** : Prédire le prix d'une maison.
- **Température future** : Prédire la météo du lendemain.
- **Prévisions économiques** : Prédire les ventes d'une entreprise.

## ● Exemples d'algorithmes d'apprentissage supervisé

- Régression linéaire
- Plus proche voisin (K-Nearest Neighbors)
- Naive Bayes gaussien
- Arbres de décision
- Machines à vecteurs de support (SVM)
- Forêts aléatoires

## ● Types de machine Learning (Apprentissage)

### □ Apprentissage non supervisé

L'apprentissage non supervisé consiste à entraîner une machine en utilisant des informations qui ne sont ni classées ni étiquetées. La tâche de la machine est de regrouper les informations non triées selon des similarités, des motifs et des différences sans aucune formation préalable des données.

### □ Types d'apprentissage non supervisé

- **Clustering** : Découvrir les regroupements inhérents aux données, comme regrouper les clients par comportement d'achat.
- **Association** : Découvrir des règles qui décrivent de grandes portions de vos données, comme les personnes qui achètent X ont tendance à acheter Y.

### □ Exemples d'algorithmes d'apprentissage non supervisé

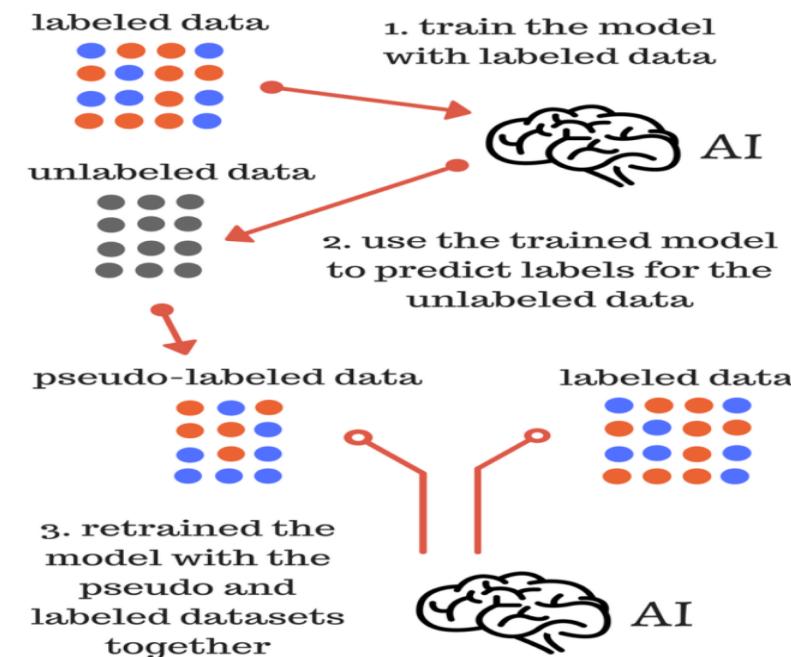
- K-means
- DBSCAN
- Agglomerative

# Machine Learning

## ● Types de machine Learning (Apprentissage)

### □ Apprentissage semi-supervisé

- Pour contrer les inconvénients de l'apprentissage supervisé et non supervisé, le concept d'apprentissage semi-supervisé a été introduit. Dans ce type d'apprentissage, l'algorithme est entraîné sur une combinaison de données étiquetées et non étiquetées.
- Généralement, cette combinaison contient une très petite quantité de données étiquetées et une très grande quantité de données non étiquetées
- Dans l'apprentissage semi-supervisé, les données étiquetées sont utilisées pour apprendre un modèle et, à l'aide de ce modèle, les données non étiquetées sont étiquetées, ce qu'on appelle le pseudo-étiquetage, puis, à l'aide de l'ensemble des données, le modèle est formé en vue d'une utilisation ultérieure.



# Machine Learning

## ● Types de machine Learning (Apprentissage)

### □ Apprentissage par renforcement

- L'apprentissage par renforcement (RL) est une méthode dans laquelle un agent apprend à prendre des décisions optimales en interagissant avec un environnement, à l'aide d'algorithmes, afin de maximiser une récompense cumulative.

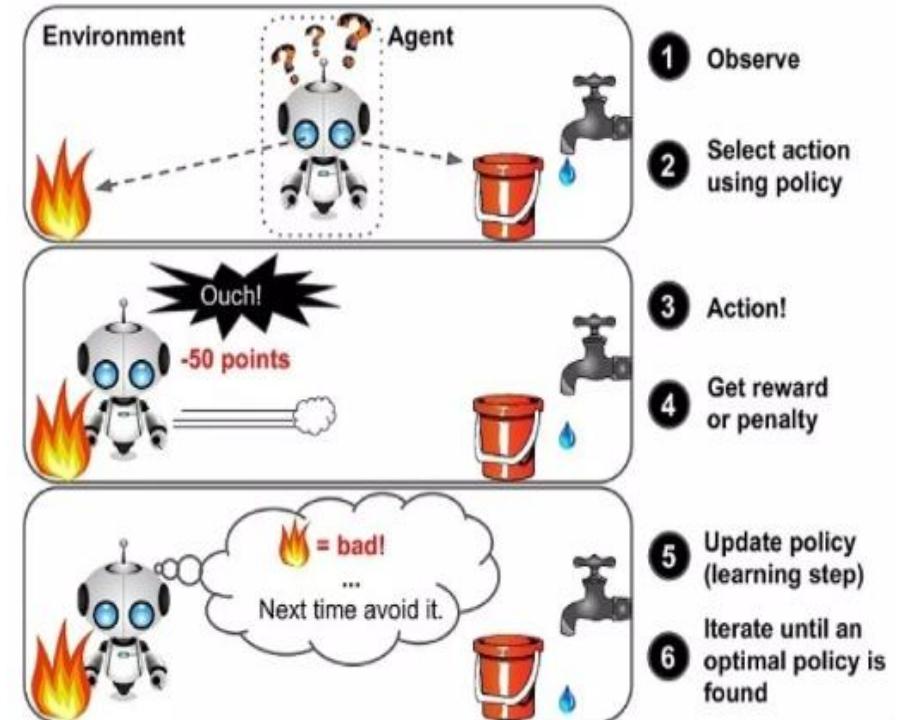
- **Analogies :**

*Comme un chien dressé : Un agent RL explore des actions aléatoires (comme un chien qui saute ou aboie), mais n'est "récompensé" que lorsqu'il trouve le bon comportement (ex : s'asseoir). Plus il répète l'action gratifiante, plus il l'optimise.*

- **Applications phares :**

- ✓ Robotique autonome
- ✓ Trading algorithmique
- ✓ Jeux intelligents

- **Outils techniques:** Q-Learning, Policy Gradients,..



## ● Présentation de scikit-learn

- Bibliothèque libre d'utilisation et facile à utiliser
- Bibliothèque dédiée au machine Learning pour python
- Une large adoption dans le monde du travail
- Implémentation de « quasiment » tous les grands algorithmes du machine Learning
- Documentation détaillée de l'API
- Support d'une grande communauté

```
!pip install scikit-learn
pip install scikit-learn
pip install --upgrade scikit-learn
python -m pip show scikit-learn
```



## ● Fonctions clés de Scikit-learn

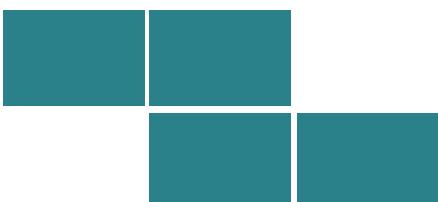
Fonctionnalité	Modules principaux
Prétraitement des données	sklearn.preprocessing
Séparation des jeux de données	sklearn.model_selection.train_test_split
Modèles de classification	sklearn.linear_model, sklearn.tree, sklearn.svm, sklearn.neighbors
Modèles de régression	sklearn.linear_model, sklearn.ensemble
Clustering	sklearn.cluster
Évaluation	sklearn.metrics
Sélection de modèles	sklearn.model_selection.GridSearchCV

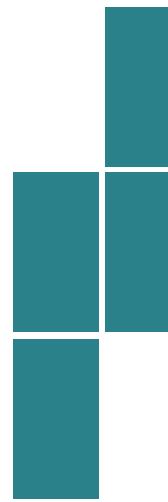
```
# 1. Import des bibliothèques nécessaires
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```



## Chapitre 3 : Appliquer la régression et la classification

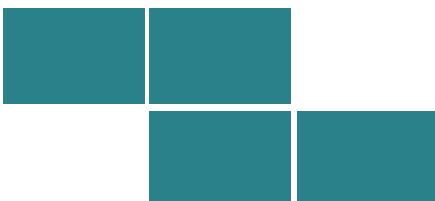
- ✓ Préparer les données d'entraînement (Prétraitement)
- ✓ Développer un modèle de régression linéaire
- ✓ Tester un modèle de régression et effectuer une validation croisée
- ✓ Réduire la dimensionnalité (ACP)
- ✓ Appliquer le clustering (Algorithme K-means)
- ✓ Développer un modèle de classification (Arbre de décision)
- ✓ Gérer les hyperparamètres (GridSearchCV)





## Chapitre 3-1 : Appliquer la régression et la classification

- ✓ Collecter et Préparer les données d'entraînement (Prétraitement)
- ✓ Développer un modèle de régression linéaire
- ✓ Tester un modèle de régression et effectuer une validation croisée



# Appliquer la régression et la classification

## ● Préparer les données

C'est l'étape **la plus importante**, qui consomme beaucoup de temps

- Nettoyage des données
- Transformation des données
  - Données textuelles
  - Données numériques



# Appliquer la régression et la classification

## ● Collecter et Préparer les données d'entraînement (Prétraitement)

### 1. Collecte des données

#### a. Sources possibles :

- Fichiers CSV/Excel
- Bases de données SQL / NoSQL
- APIs (ex. Twitter, météo, etc.)
- Web scraping (avec BeautifulSoup, Scrapy)
- Open Data (Kaggle, UCI, data.gouv.fr)

### 2. Chargement du Dataset

```
import pandas as pd
```

```
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv')
df.head()
```

### 3. Prétraitement des données

#### a. Exploration initiale

```
print(df.info())
print(df.describe())
print(df.isnull().sum())
print(f"Shape: {df.shape}")
```

# Appliquer la régression et la classification

## ● Collecter et Préparer les données d'entraînement (Prétraitement)

### 1. Collecte des données

#### a. Sources possibles :

- Fichiers CSV/Excel
- Bases de données SQL / NoSQL
- APIs (ex. Twitter, météo, etc.)
- Web scraping (avec BeautifulSoup, Scrapy)
- Open Data (Kaggle, UCI, data.gouv.fr)

**Structure de données(Dataset):** Tableau de valeurs disposées en lignes et en colonnes. Les lignes sont des cas (observations) et les colonnes sont des variables.

	Features ( $x$ )				Label ( $y$ )
	$x_1$	$x_2$	$x_3$	$x_4$	$y$
Obs-1					
Obs-2					
Obs-3					
Obs-4					

- Features
- Input variables
- Independent variables

- Target/Label
- Output variable
- Dependent variable

- Real estate : area, distance from city centre, floor number, ... + price (continuous)

- Image : pixel=3 features (RGB) + cat or dog (categorical)

An image of 100x100 pixels in color contains 30000 features (too much!)

# Appliquer la régression et la classification

## ● Collecter et Préparer les données d'entraînement (Prétraitement)

### 2. Prétraitement des données

#### b. Nettoyage des données

- Valeurs manquantes

```
df = df.dropna() # ou bien on peut imputer avec la moyenne/mode
```

- Doublons

```
df = df.drop_duplicates()
```

#### b. Analyse et encodage des variables catégorielles

```
# 3. Séparer X et y  
X = df.drop(columns=['medv'])  
y = df['medv']
```

```
# 4. Identifier colonnes numériques et catégorielles  
numeric_features = X.select_dtypes(include=[np.number]).columns  
categorical_features = X.select_dtypes(include=['object']).columns
```

# Appliquer la régression et la classification

## ● Collecter et Préparer les données d'entraînement (Prétraitement)

### 2. Prétraitement des données

```
# 6. Encodage des catégorielles avec LabelEncoder (colonne par colonne)
X_categorical_encoded = X[categorical_features].copy()
for col in categorical_features:
    le = LabelEncoder()
    X_categorical_encoded[col] = le.fit_transform(X_categorical_encoded[col])
```

#### d. Normaliser les colonnes numériques

```
scaler = StandardScaler()
X_numeric_scaled = scaler.fit_transform(X[numeric_features])
```

#### e. Fusionner les colonnes traitées

---

```
X_preprocessed = np.concatenate([X_numeric_scaled, X_categorical_encoded.values], axis=1)
```

NB Numpy,pandas:

axis=0 → opération par colonne

axis=1 → opération par ligne

# Appliquer la régression et la classification

## ● Collecter et Préparer les données d'entraînement (Prétraitement)

### B. Prétraitement des données

#### e. Déetecter les valeurs aberrantes:

```
for col in num_cols:  
    plt.figure(figsize=(6, 1))  
    sns.boxplot(data=df, x=col)  
    plt.show()
```

On peut aussi supprimer les outliers avec des Z-scores  $|Z| > 3$  :

$|Z| \leq 3$  sont considérés comme non outliers

```
from scipy import stats  
df = df[(np.abs(stats.zscore(df[num_cols])) < 3).all(axis=1)]
```

Le Z-score mesure à quelle distance (en nombre d'écart-types) se trouve une valeur par rapport à la moyenne de sa colonne.

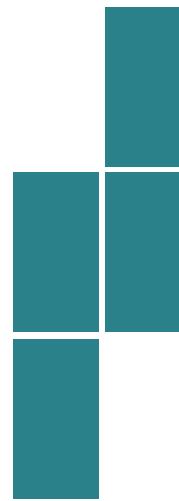
$$Z = \frac{X_i - \mu}{\sigma}$$

- $X_i$  : la valeur de l'observation
- $\mu$  : la moyenne de la colonne
- $\sigma$  : l'écart-type de la colonne

👉 Si  $Z = 0$ , la valeur est exactement égale à la moyenne.

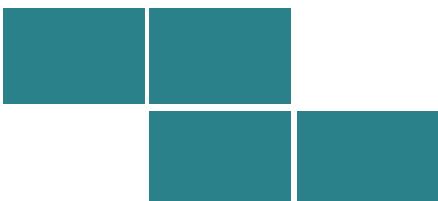
👉 Si  $|Z| = 1$ , la valeur est à une distance d'un écart-type de la moyenne.

👉 Si  $|Z| > 3$ , elle est considérée comme **extrême** (outlier potentiel).



## Chapitre 3-2: Algorithmes ML: régression et classification

- ✓ 1. Régression linéaire
- ✓ 2. Machines à vecteurs support (SVM)
- ✓ 3. k-plus proches voisins (KNN)
- ✓ 4. Arbres de décision et Random Forest
- ✓ 5. XGBoost



# Appliquer la régression et la classification

## ● 1. Développer un modèle de régression linéaire

### De quoi s'agit-il?

La régression linéaire est une méthode de machine learning utilisée pour prédire la valeur d'une variable cible en fonction d'une ou plusieurs variables.

Une relation linéaire est considérée entre la variable cible et la/les variables explicatives.

# Appliquer la régression et la classification

## ● Développer un modèle de régression linéaire

De quoi s'agit-il?

Régression linéaire simple :

- Une seule variable cible et une seule variable explicative
- $Y = f(X)$

Régression linéaire multiple:

- Une seule variable cible et plusieurs variables explicatives
- $Y=f(X_1, X_2, \dots, X_n)$

## ● Développer un modèle de régression linéaire

### Hypothèses

- Toutes les variables sont quantitatives
- Pas de données manquantes
- Une relation linéaire lie la variable cible aux variables explicatives
- Les variables explicatives sont indépendantes
- Les erreurs de prédiction suivent une distribution normale

# Appliquer la régression et la classification

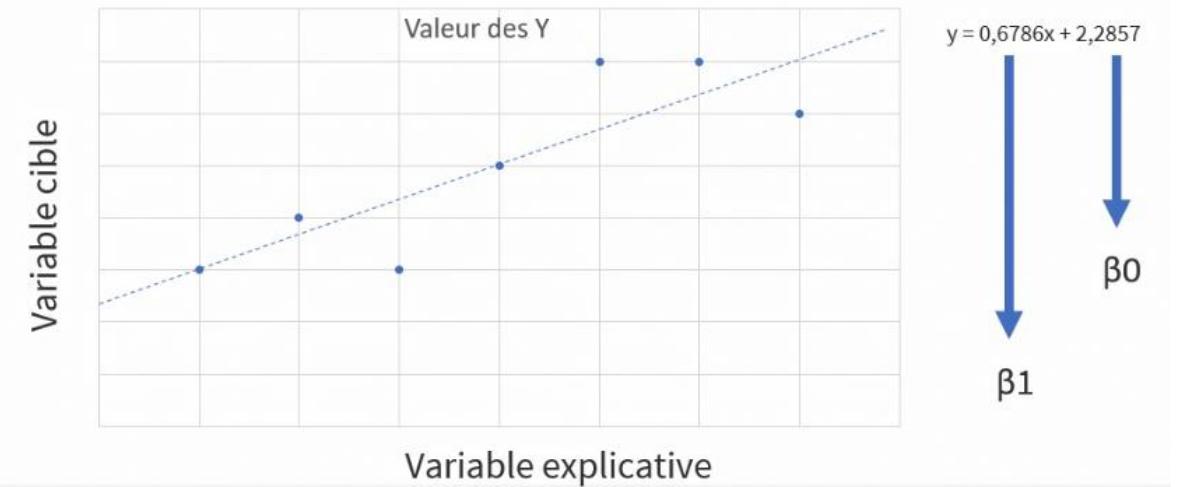
## ● Développer un modèle de régression linéaire

### Comprendre la régression linéaire

Variable explicative

Variable cible

Valeur des X	Valeur des Y
1	3
2	4
3	3
4	5
5	7
6	7
7	6



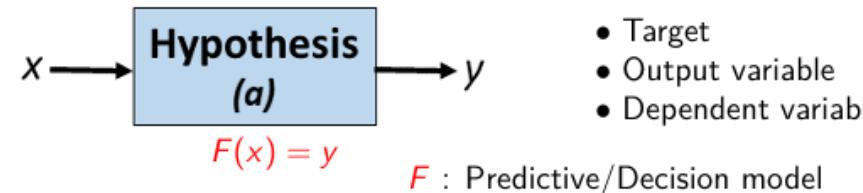
# Appliquer la régression et la classification

## ● Développer un modèle de régression linéaire

### e. Séparer en train/test \*\*après\*\* le prétraitement

```
X_train, X_test, y_train, y_test = train_test_split(X_preprocessed, y, test_size=0.2, random_
```

- Features
- Input variables
- Independent variables



- Target
- Output variable
- Dependent variable

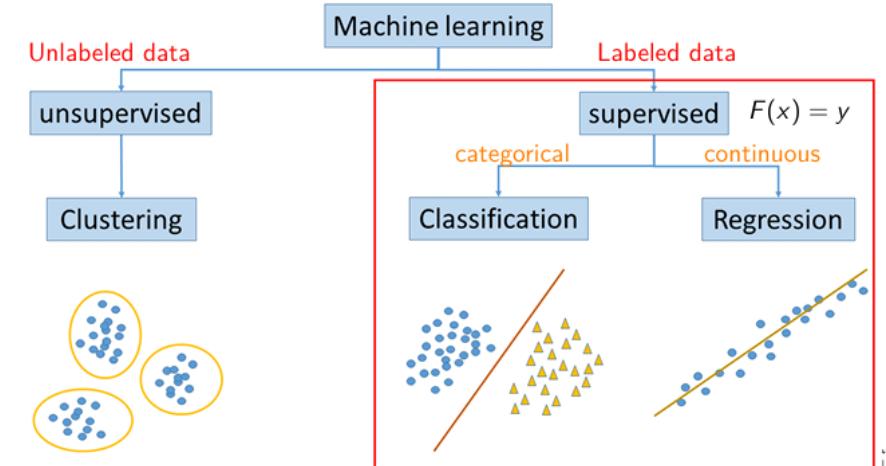
### f. Entraînement du modèle

```
model = LinearRegression()  
model.fit(X_train, y_train)
```

#### Choosing a model :

- $F(x) = a_0 + a_1x$  (Linear regression)
- $F(x) = a_0 + a_1x + a_2x^2$  (polynomial)
- $F(x) = a_0 + a_1x_1 + a_2x_2 + a_3x_3$  (multivariable)
- $F(x) = \frac{1}{1+e^{-(a_0+a_1x)}}$  (classification)
- ANN, SVM, Random Forest, CNN, ...

$a = (a_0, a_1, \dots, a_p)^t$  the model weights



# Appliquer la régression et la classification

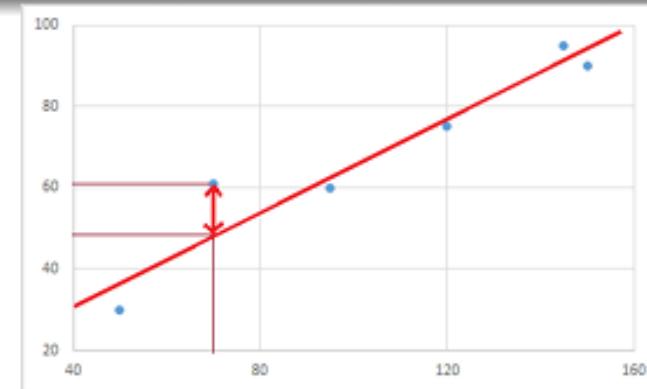
## ● Développer un modèle de régression linéaire

### e. Entrainement du modèle

#### Training set

L'ensemble d'apprentissage est un ensemble de couple entrée-sortie  $\mathcal{A} = \{(x(i), y(i)), i = 1 \cdots, m\}$  que nous observons et que nous utilisons pour déterminer les poids du modèle.

	A	B
1	superficie (m²)	prix k\$ (y)
2	50	30
3	150	90
4	70	61
5	95	60
6	120	75
7	145	95



#### Training

L'apprentissage (Training) est le processus permettant de déterminer les poids  $a$  tels que les écarts entre la sortie estimée  $F(x(i))$  et la sortie observée  $y(i)$  soient les plus petits possibles.

# Appliquer la régression et la classification

## ● Développer un modèle de régression linéaire

### e. Prédiction et évaluation

```
y_pred = model.predict(X_test)  
print("MSE :", mean_squared_error(y_test, y_pred))  
print("R² :", r2_score(y_test, y_pred))
```

#### R<sup>2</sup> (Coefficient de Détermination)

**Définition :** Mesure la proportion de la variance de la variable cible expliquée par le modèle.

**Formule :**

- $y_i$  = valeur réelle
- $\hat{y}_i$  = valeur prédictée
- $\bar{y}$  = moyenne des valeurs réelles

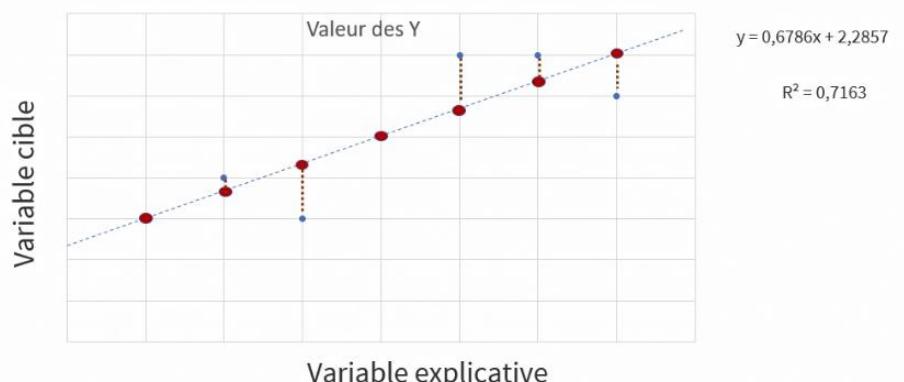
où  $\bar{y}$  est la moyenne des valeurs réelles.

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

**Caractéristiques :**

- Variante entre 0 et 1 (ou  $-\infty$  à 1 pour des modèles très mauvais).
- **1** : modèle parfait.
- **0** : modèle aussi bon qu'une prédiction par la moyenne.
- **< 0** : modèle pire qu'une simple moyenne.

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



Cas d'usage	R <sup>2</sup>	MAE/MSE/RMSE
Comparer des modèles sur différentes échelles	✓	✗ (dépend des unités)
Évaluer la proportion de variance expliquée	✓	✗
Mesurer l'erreur absolue en unité réelle (ex. euros, mètres)	✗	✓
Pénaliser fortement les grandes erreurs (ex. risques financiers)	✗	✓ (MSE/RMSE)

# Appliquer la régression et la classification

## ● Développer un modèle de régression linéaire

### e. Prédiction et évaluation

**Matrice de confusion:** A partir de la matrice, on calcule des indicateurs de performance :

n=165	Predicted:		
	NO	YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Métrique	Formule	Interprétation
Précision (Precision)	$VP / (VP + FP)$	% de prédictions positives correctes.
Rappel (Recall/Sensibilité)	$VP / (VP + FN)$	% de vrais positifs détectés.
F1-Score	$2 \times (\text{Précision} \times \text{Rappel}) / (\text{Précision} + \text{Rappel})$	Moyenne harmonique de précision et rappel.
Exactitude (Accuracy)	$(VP + VN) / \text{Total}$	% de prédictions correctes (global).
Spécificité	$VN / (VN + FP)$	% de vrais négatifs correctement identifiés.

# Appliquer la régression et la classification

## ● Développer un modèle de régression linéaire

### e. Prédiction et évaluation

**Matrice de confusion:** A partir de la matrice, on calcule des indicateurs de performance :

		P		
		Prédit Chat	Prédit Chien	Prédit Oiseau
N	Réel Chat	45	VP	3 FP
	Réel Chien	1	FP	50 VP
	Réel Oiseau	0	FP	2 FP 40 VP

- **Diagonale** = Bonnes prédictions.
- **Hors diagonale** = Confusions (ex. 3 chats classés comme chiens).

FN

Cas	Formule correcte
Binaire (2 classes)	$\frac{VP+VN}{Total}$

Multiclasse ( $\geq 3$  classes)

$$\frac{\sum_i VP_i}{Total} = \frac{\text{somme(diagonale)}}{\text{somme(toutes les cases)}}$$

Classe	VP	FP	FN	VN
Chat	45	1	5	96
Chien	50	5	5	87
Oiseau	40	6	2	99

Total des valeurs – VP+FP+FN

Réalité \ Prédit	Chat	Chien	Oiseau	Total (réel)
Chat	45	3	2	50
Chien	1	50	4	55
Oiseau	0	3	40	43
Total (prédit)	46	56	46	148

#### 2. Formule de l'Accuracy (exactitude)

$$\text{Accuracy} = \frac{\text{Nombre de bonnes prédictions}}{\text{Nombre total d'observations}} = \frac{TP_{total}}{N_{total}}$$

# Appliquer la régression et la classification

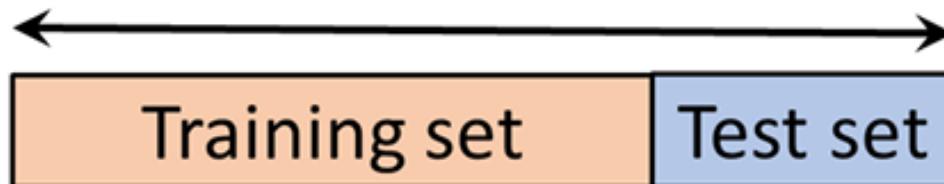
## ● Développer un modèle de régression linéaire

### F. Généralisation

#### Généralisation

Capacité du modèle à faire de bonnes prédictions pour de nouvelles observations.

- **Data splitting**



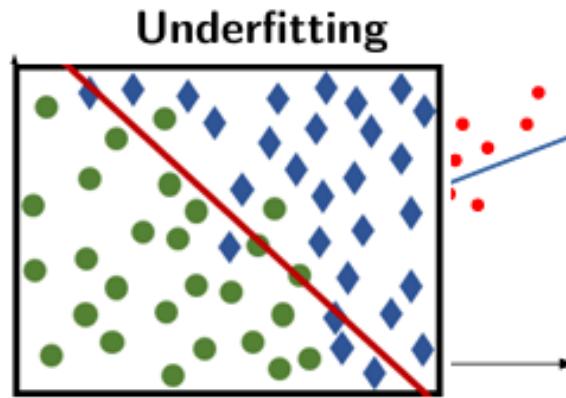
- **Training set:** pour trouver les poids  $a$
- **Test set:** pour s'assurer que le modèle généralise

# Appliquer la régression et la classification

## ● Développer un modèle de régression linéaire

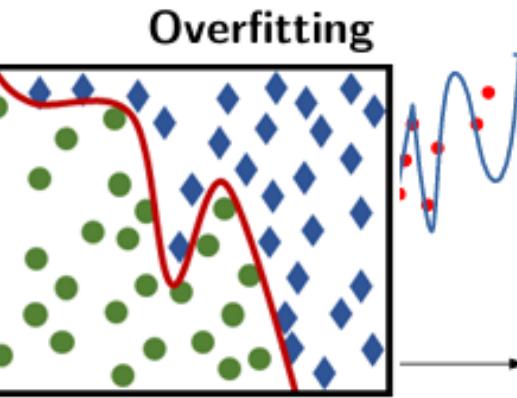
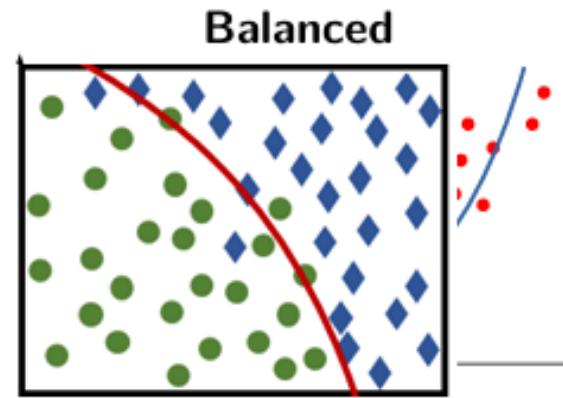
### ▪ F. Généralisation

#### Biais vs variance



Biais

Hypothèse erronée dans l'algorithme d'apprentissage. Un biais élevé correspond à une mauvaise représentation de la relation entre la sortie  $y$  et l'entrée  $x$ . Il conduit à un problème d'underfitting.



Variance

Erreur due à la sensibilité du modèle au bruit. Le modèle colle trop aux données d'apprentissage. Une variance forte est synonyme d'un problème d'overfitting.

# Appliquer la régression et la classification

## ● Développer un modèle de régression linéaire

### ▪ F. Généralisation

Lors de l'entraînement d'un modèle de **Machine Learning**, il est crucial de diviser les données en **trois ensembles distincts** pour garantir une bonne **généralisation** du modèle et éviter le surapprentissage (**overfitting**).

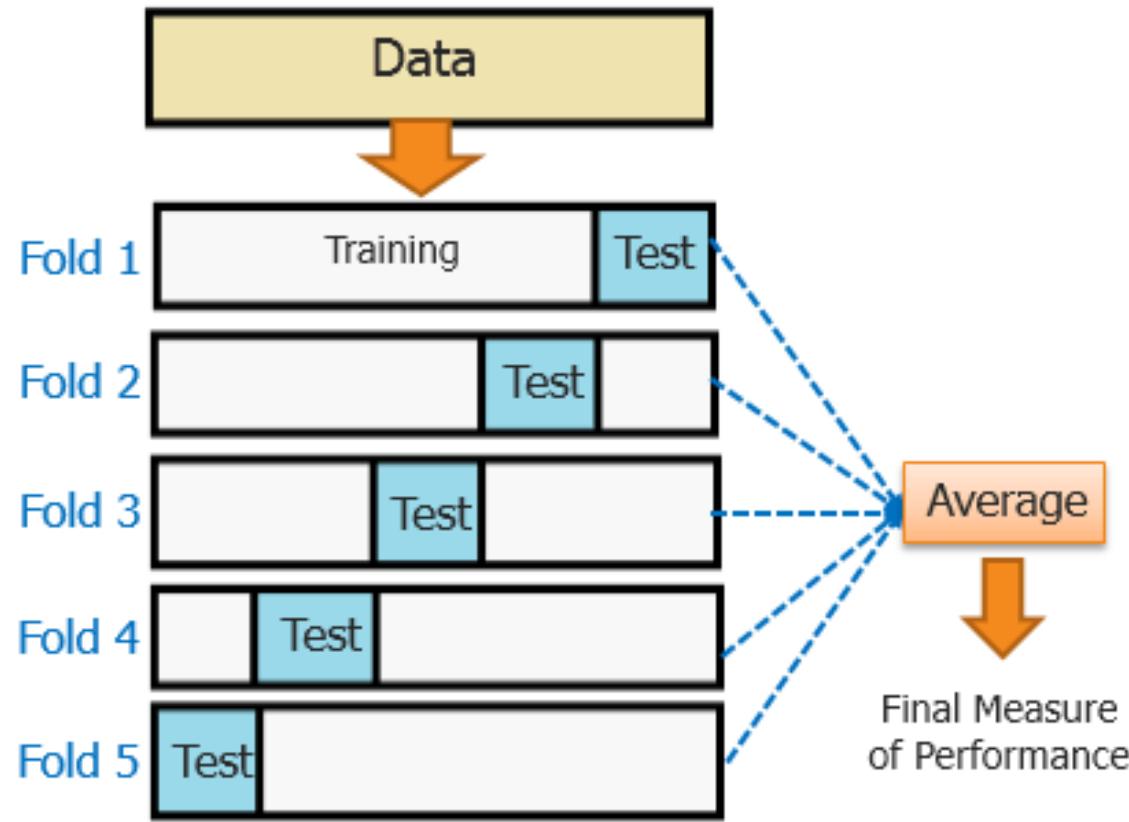
Entraînement	Ajuste les <b>paramètres</b> du modèle	Utilisé pour l'apprentissage	<b>60-80%</b>
Validation	Ajuste les <b>hyperparamètres</b> et évite l'overfitting	Utilisé pendant l'entraînement	<b>10-20%</b>
Test	Évalue les <b>performances finales</b> du modèle	Utilisé après l'entraînement	<b>10-20%</b>

On utilise **Scikit-learn** pour diviser les données.



# Appliquer la régression et la classification

- Développer un modèle de régression linéaire
  - G. Cross validation



## ● Développer un modèle de régression linéaire

### □ Workflow du TP :

1. Chargement et exploration des données
2. Exploration des données (EDA):  
→ Nettoyage (doublons, outliers)
1. Prétraitement  
→ Normalisation, encodage, sélection de variables
1. Modélisation et évaluation
2. Visualisation des résultats

## ● 2. Développer un modèle de SVM

### ▪ Introduction au SVM

Le **Support Vector Machine** (SVM) est un algorithme d'apprentissage **supervisé** introduit dans les années 1990 par **Vladimir Vapnik**.

Il est utilisé pour :

- La **classification** (binaire ou multiclasse) (SVC)
- La **régression** (SVR)

**Idée clé :**

Trouver l'**hyperplan optimal** qui sépare les classes avec **la plus grande marge possible**.

# Appliquer la régression et la classification

## ● 2. Développer un modèle de SVM

### ▪ Paramètres importants d'un modèle SVM (scikit-learn)

#### Commun (classification SVC, régression SVR)

- kernel : fonction noyau ('linear', 'rbf' (par défaut), 'poly', 'sigmoid').
- C : régularisation (grand = marge plus étroite, peu d'erreurs ; petit = marge plus large, plus tolérant).
- gamma : (pour rbf, poly, sigmoid) influence locale des points (grand = frontière complexe ; petit = frontière lisse).
- degree : degré du polynôme (si kernel='poly').
- coef0 : constante d'offset (pour poly et sigmoid).
- shrinking : heuristique d'optimisation (True par défaut).
- tol : tolérance d'arrêt (précision de l'optimisation).
- max\_iter : itérations max (-1 = illimité).
- cache\_size : taille du cache kernel (MB).

#### Spécifique classification (SVC)

- class\_weight : poids des classes ('balanced' ou dict) pour les jeux déséquilibrés.
- probability : calcule des probabilités (logistic calibration) — plus lent.
- decision\_function\_shape : 'ovr' (par défaut) ou 'ovo' pour le multi-classe.
- break\_ties : améliore les prédictions probables quand decision\_function\_shape='ovr'.

#### Spécifique régression (SVR)

- epsilon : largeur du tube  $\epsilon$ -insensible (erreur "gratuite").
- (Le reste: mêmes kernel, C, gamma, degree, coef0, etc.)

# Appliquer la régression et la classification

## ● 2. Développer un modèle de SVM

### ▪ Quand utiliser SVM

- Problème de classification avec deux classes cibles
- Quand le rapport caractéristiques étudiées / taille des instances est élevé
- Données avec plusieurs valeurs aberrantes

## ● 2. Développer un modèle de SVM

### ▪ Quand éviter SVM

- Beaucoup de lignes et peu de caractéristiques étudiées
- Compréhension précise des prédictions
- Durée d'exécution

## ● 2. Développer un modèle de SVM

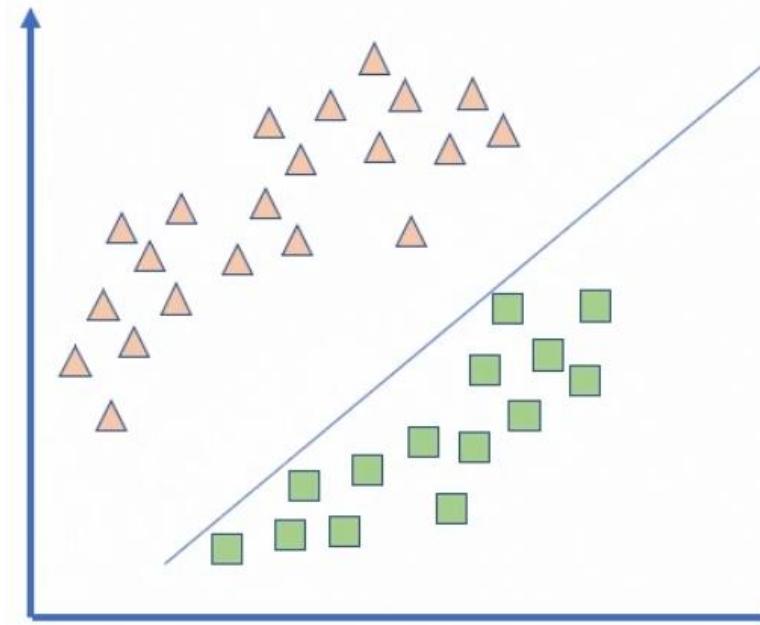
- L'astuce de kernel

Quand les données ne sont pas linéairement séparables, une fonction noyau (Kernel) est introduite pour transformer ces données dans un espace de plus grande dimension, dans lequel le problème résolu est linéaire.

# Appliquer la régression et la classification

## ● 2. Développer un modèle de SVM

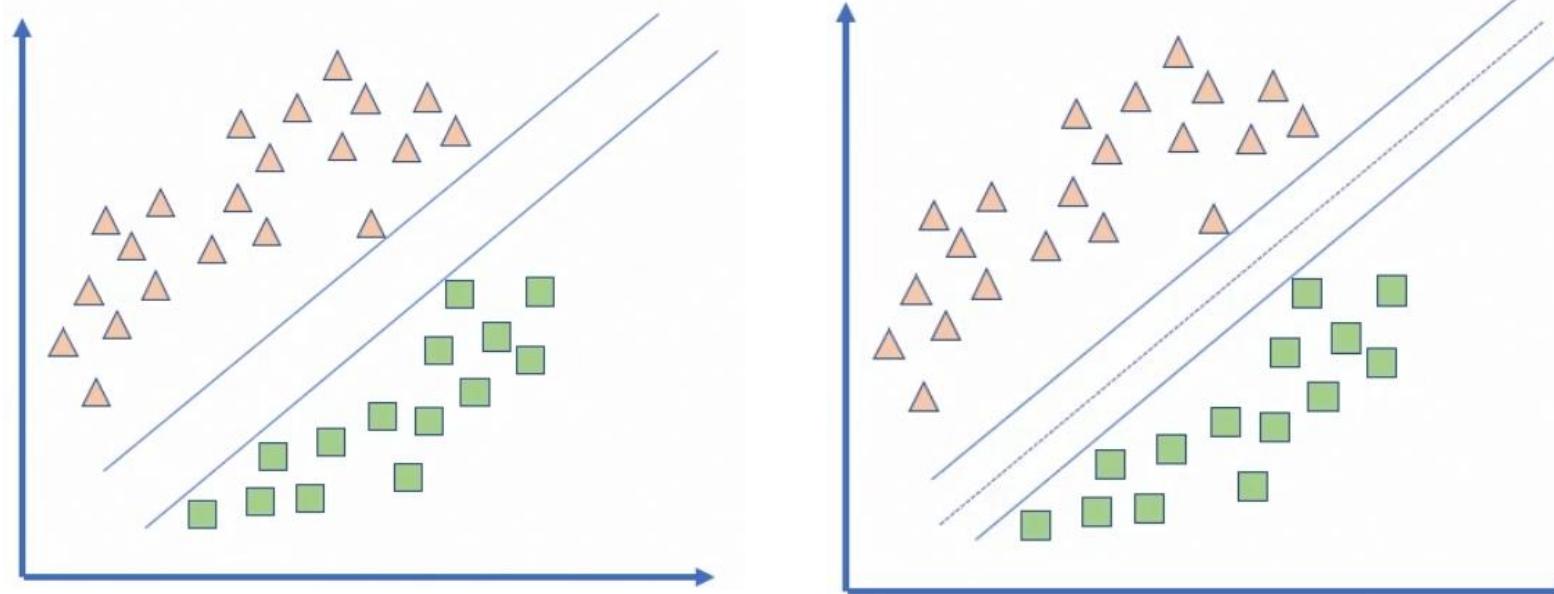
- Comprendre SVM



# Appliquer la régression et la classification

## ● 2. Développer un modèle de SVM

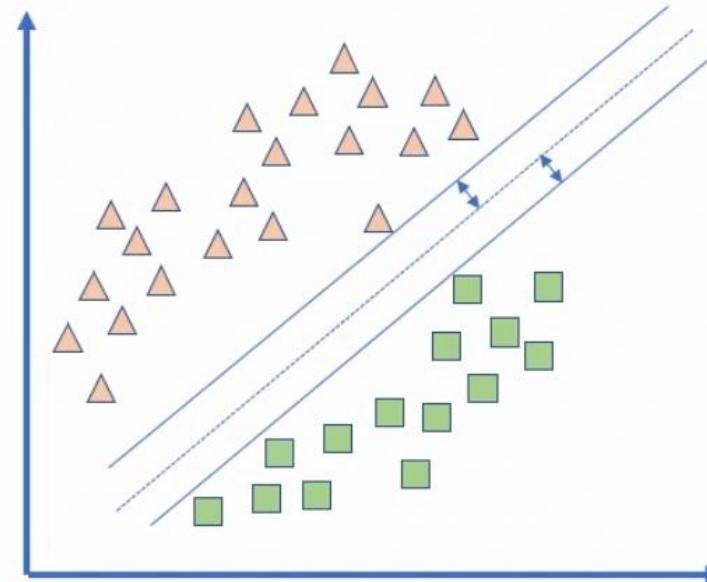
- Comprendre SVM



# Appliquer la régression et la classification

## ● 2. Développer un modèle de SVM

### ▪ Comprendre SVM



Calcul de la marge



Maximisation de la marge

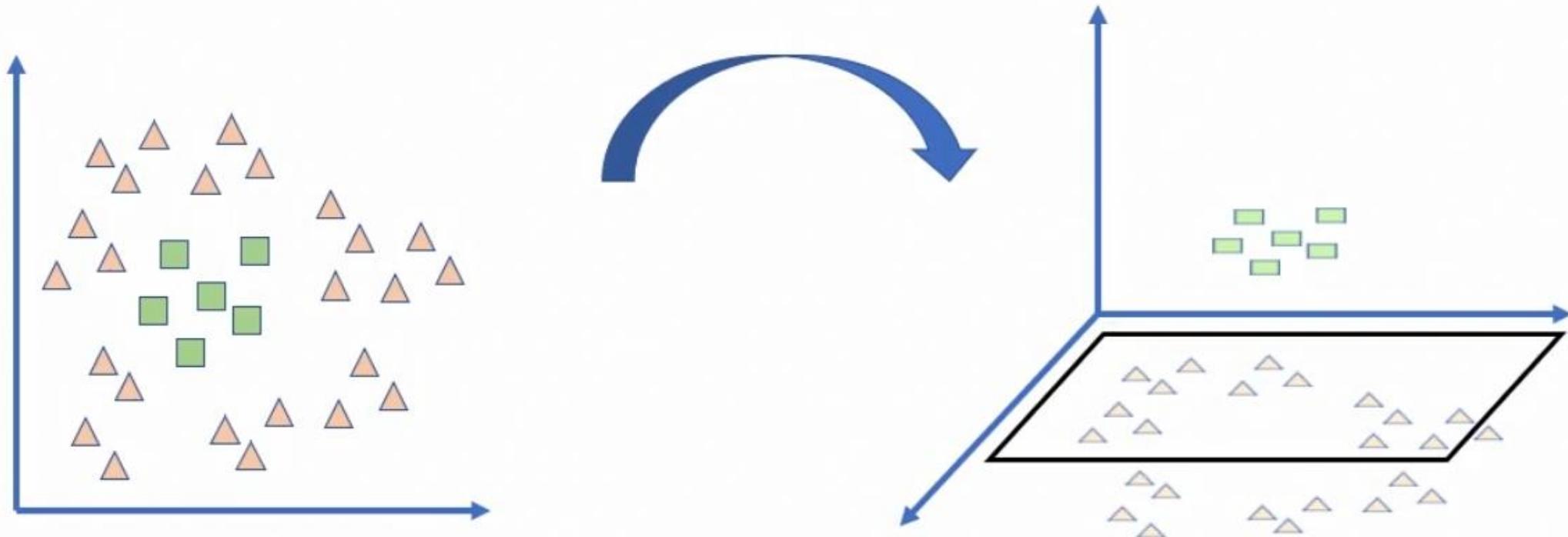
**Marge:** La marge correspond à la distance entre l'hyperplan et les points les plus proches de chaque classe. Ces points particuliers sont appelés vecteurs de support.

# Appliquer la régression et la classification

## ● 2. Développer un modèle de SVM

- **SVM non linéaire** : les données nécessitent une transformation via une **fonction noyau (kernel)** pour devenir séparables.
- **SVM linéaire** : les données sont séparables par une droite / un plan.

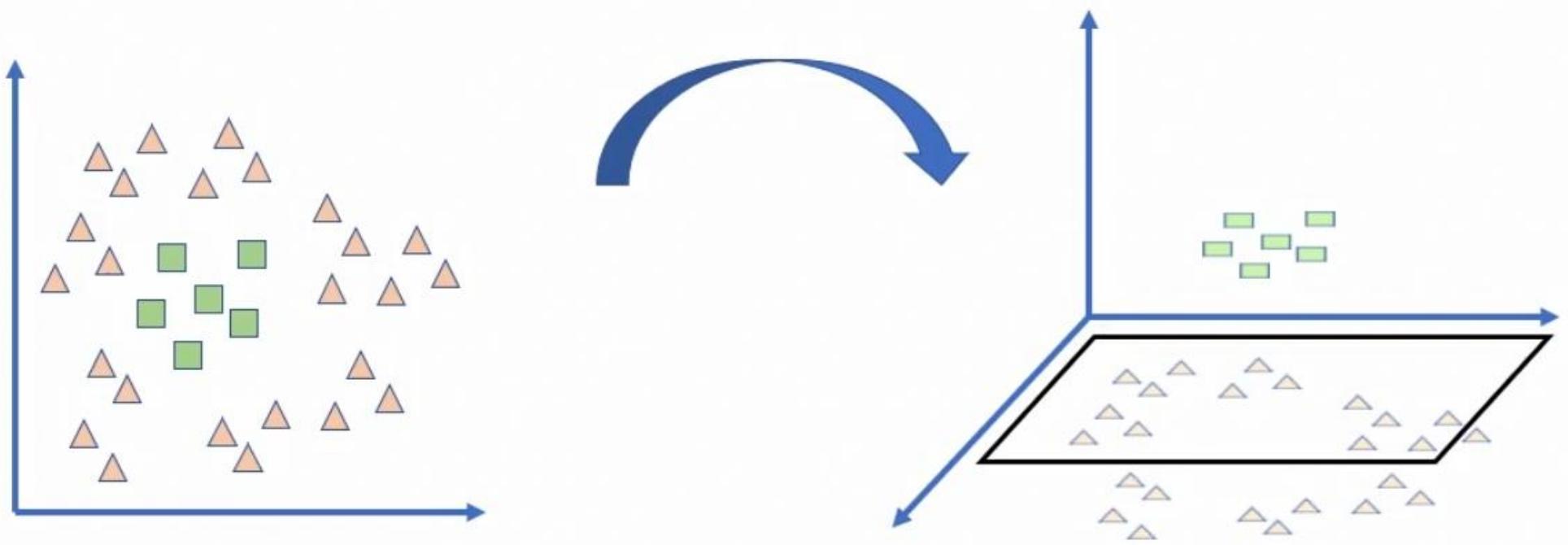
- **L'astuce de kernel**



# Appliquer la régression et la classification

## ● 2. Développer un modèle de SVM

- L'astuce de kernel



# Appliquer la régression et la classification

## ● 2. Développer un modèle de SVM

### ▪ Paramètres importants d'un modèle SVM (scikit-learn)

✓ Machine à vecteurs de support (SVM - Support Vector Machine):

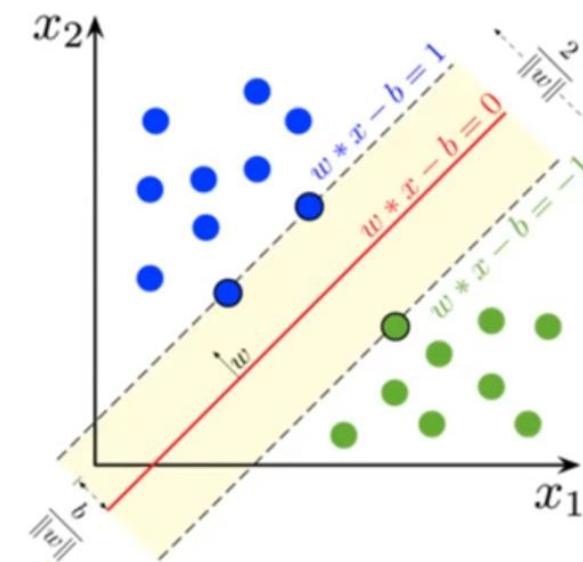
- Crée un hyperplan qui sépare les classes dans l'espace des caractéristiques
- Classification binaire et multiclases
- Paramètres:

➤ Hyperparamètres:

- Noyau (linéaire, polynomial, RBF),
- paramètre de régularisation (C),
- Paramètre du noyau

➤ Paramètres entraînables;

- Vecteurs de support
- Poids  $W$
- Biais  $b$

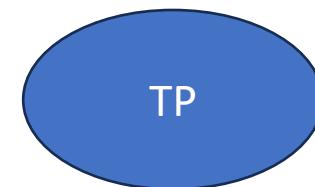


# Appliquer la régression et la classification

## ● 2. Développer un modèle de SVM

### ▪ Bonnes pratiques avant entraînement

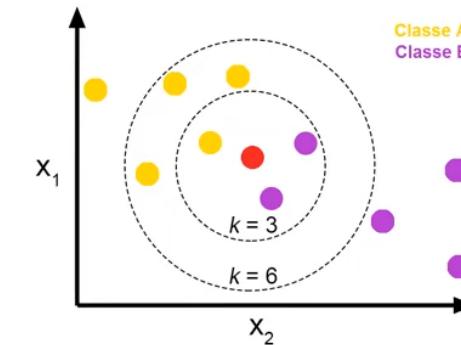
- Standardiser les features (SVM y est sensible) → StandardScaler.
- Séparer train/test (ou CV stratifiée).
- Recherche d'hyperparamètres (Grid/Random/Bayes).
- Évaluer avec des métriques adaptées (accuracy/F1 pour clas., RMSE/MAE pour régr.).
- Sauvegarder le pipeline complet (scaler + SVM) avec joblib.



# Appliquer la régression et la classification

## ● 2. k-plus proches voisins (KNN): Algorithme de classification

- Classifieur cherche le groupe des  $K$  objets similaires ou proches à l'objet ou à l'individu et lui attribue la classe des voisins prédominants
  - Utilise la notion de voisinage
  - Paramètres:
    - Hyperparamètres:
      - Le nombre de voisins ( $K$ ),
      - la métrique de distance (Euclidienne, Manhattan, ...)



# Appliquer la régression et la classification

## ● 2. k-plus proches voisins (KNN): Algorithme de classification

### ▪ Déterminer K

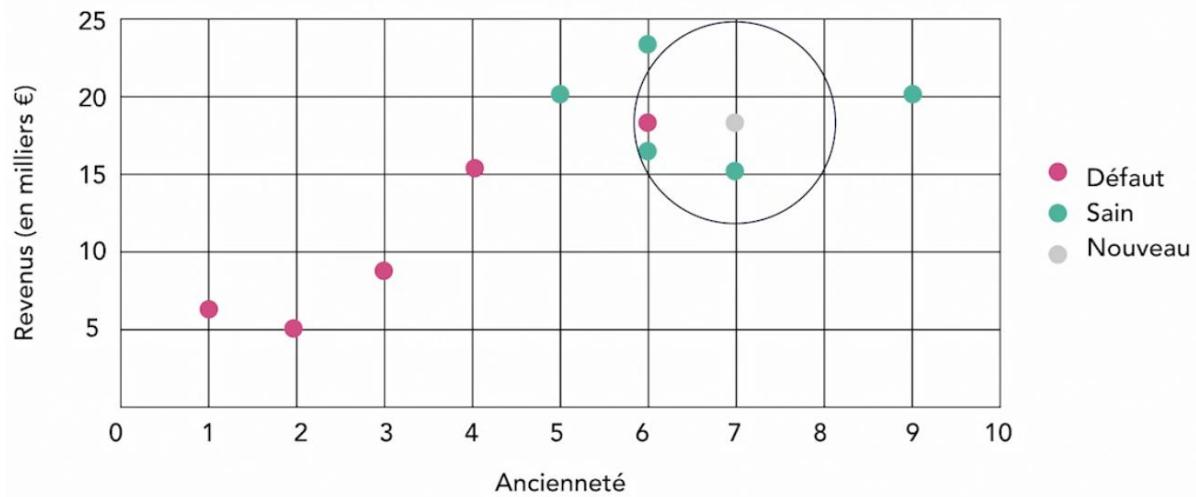
- Déterminer la valeur  $K$  pour laquelle l'erreur de classification en validation est la plus faible

Plus  $K$  est grand, moins la classification est sensible au bruit.

# Appliquer la régression et la classification

## ● 2. k-plus proches voisins (KNN): Algorithme de classification

### Application avec $K = 3$



Distance Euclidienne :  $\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$

Distance de Manhattan :  $\sum_{i=1}^k |x_i - y_i|$

Distance de Minkowski :  $\left( \sum_{i=1}^k (|x_i - y_i|)^p \right)^{\frac{1}{p}}$

# Appliquer la régression et la classification

## ● 2. k-plus proches voisins (KNN)

- **Algorithme de régression et classification**
- **Choisir la mesure de distance**
- En fonction des caractéristiques des données
- Tester différentes métriques de distance et différentes valeurs de  $K$
- Standardiser des données

Le KNN est **très efficace pour les données de petite à moyenne taille**, car :

- il **ne nécessite pas d'entraînement complexe** (aucun modèle n'est construit) ;
- il suffit de **stocker les données** et de **calculer les distances** lors de la prédiction ;
- les calculs restent rapides si le nombre d'échantillons est limité (ex. < 10 000).

**Exemples d'utilisation :**

- Reconnaissance de motifs simples
- Jeux de données de démonstration (ex. Iris, Wine, Breast Cancer)
- Classification d'images ou de textes à petite échelle

# Appliquer la régression et la classification

## ● 2. k-plus proches voisins (KNN): Algorithme de classification

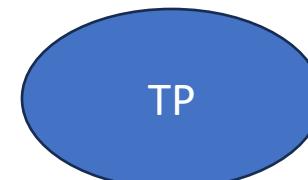
### Avantages :

- L'algorithme **K-NN** est simple et facile à mettre en oeuvre.
- Il n'est pas nécessaire de créer un modèle, de régler plusieurs paramètres ou de formuler des hypothèses supplémentaires.
- L'algorithme est polyvalent. Il peut être utilisé pour la classification ou la **régression**.

### Inconvénients :

- L'algorithme devient beaucoup plus lent à mesure que le nombre d'observation et de variables indépendantes augmente.

Étant l'un des algorithmes les plus simples de Machine Learning, il est hautement implémenté pour développer des systèmes basés sur l'apprentissage, intuitifs et intelligents qui pourraient effectuer et prendre de petites décisions tout seuls

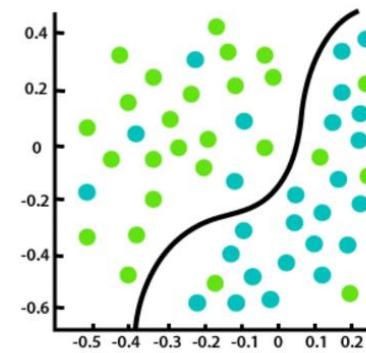


# Appliquer la régression et la classification

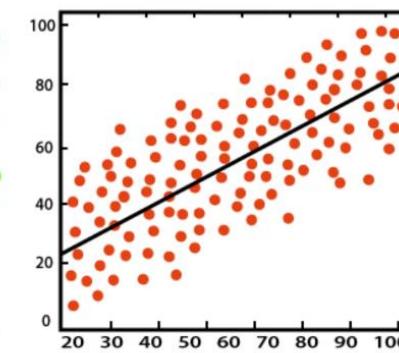
## 4. Arbres de décision

### ■ Introduction

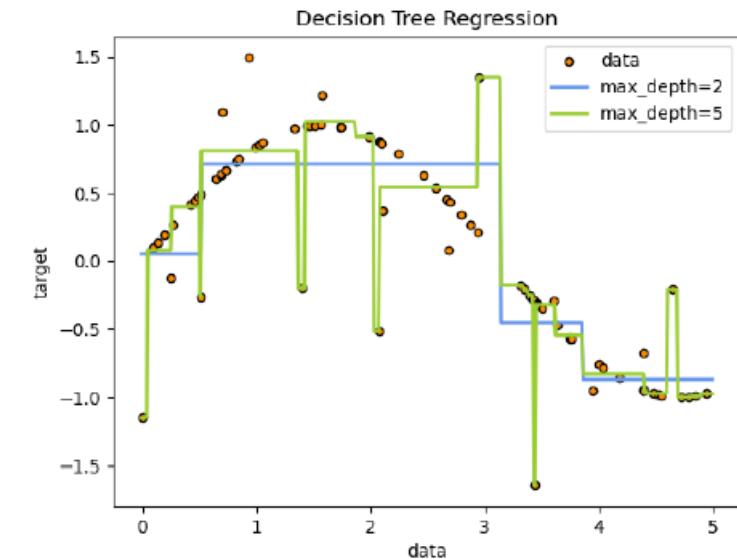
**Les arbres de décision (AD)** sont une méthode d'apprentissage supervisé utilisée pour la classification et la régression. Ils construisent un modèle basé sur des règles simples "si-alors-sinon", apprises à partir des caractéristiques des données, pour prédire une variable cible.



Classification



Regression

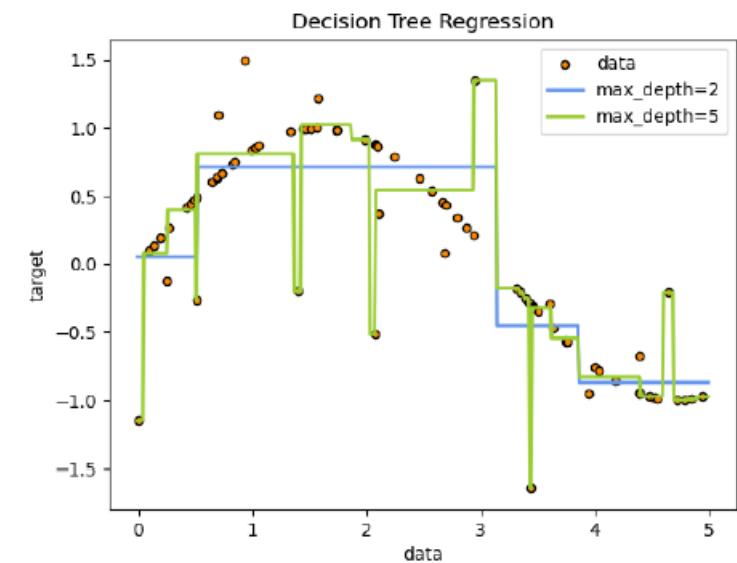


# Appliquer la régression et la classification

## 4. Arbres de décision

### ▪ Introduction

Ils fonctionnent comme une approximation par morceaux, où chaque niveau de profondeur apporte des règles plus complexes et affine la précision du modèle.



## 4. Arbres de décision

### ▪ Comment créer un arbre de décision

Créer un arbre de décision est un processus assez simple. Vous pouvez utiliser des programmes technologiques ou simplement dessiner avec un stylo sur du papier. Si nous supposons que vous avez un objectif de recherche spécifique ou un problème et que vos données ont déjà été collectées, vous pouvez créer un arbre de décision en trois étapes.

1. Dessiner le nœud initial
2. Nœuds d'expansion:
3. Atteindre les nœuds finaux:



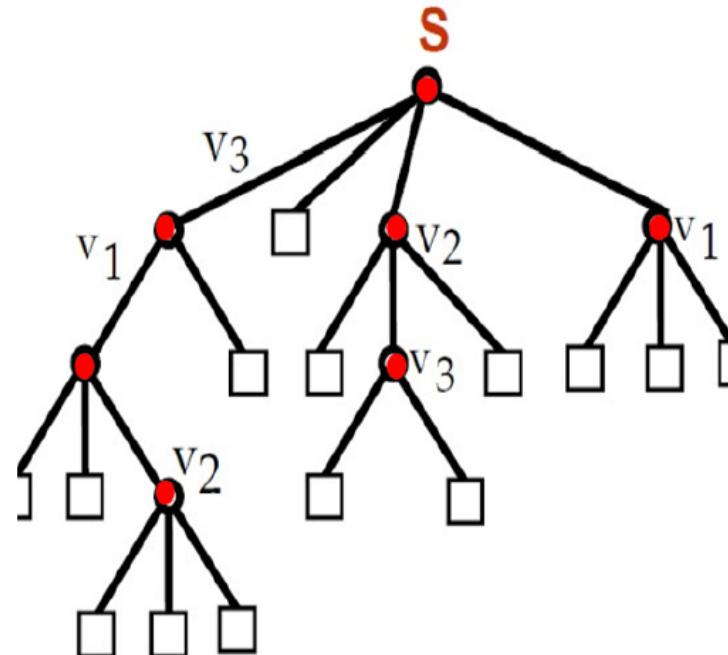
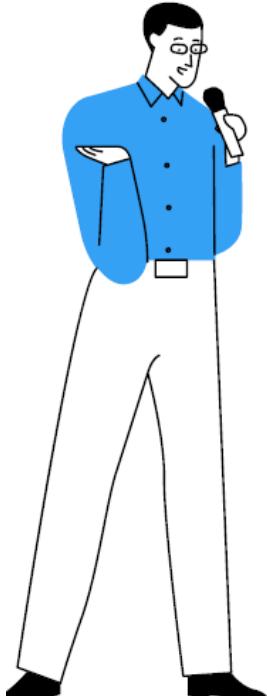
# Appliquer la régression et la classification

## 4. Arbres de décision

- **Noeud:**

Nœud racine: Premier nœud, point de départ de l'arbre.

Nœud interne : Un nœud qui représente un test sur un attribut.

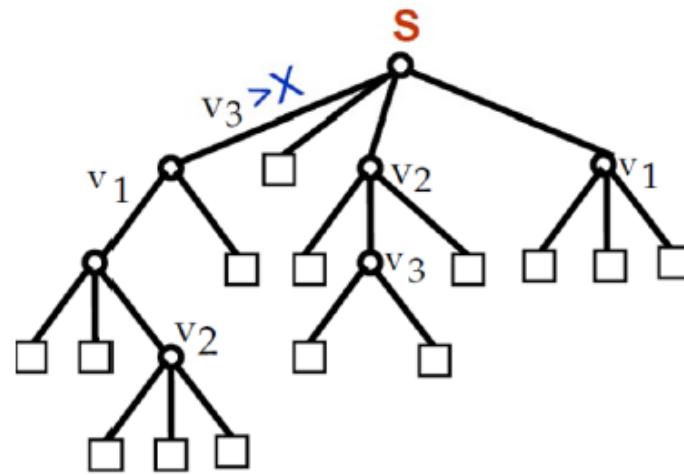


# Appliquer la régression et la classification

## 4. Arbres de décision

- **Une branche:**

Chaque branche représente un choix qui résulte du test sur un attribut.

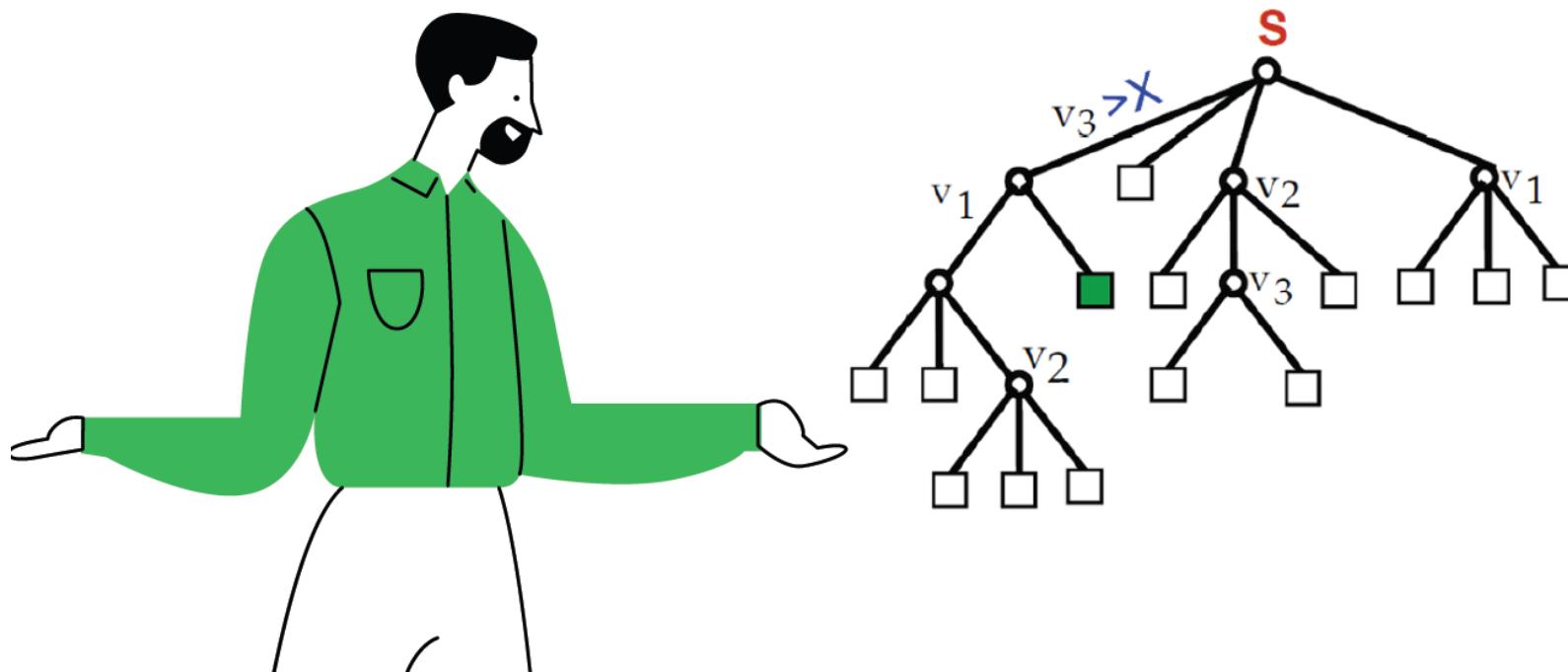


# Appliquer la régression et la classification

## 4. Arbres de décision

- **Une feuille:**

Noeuds terminaux qui donnent la décision finale (classe prédictive).

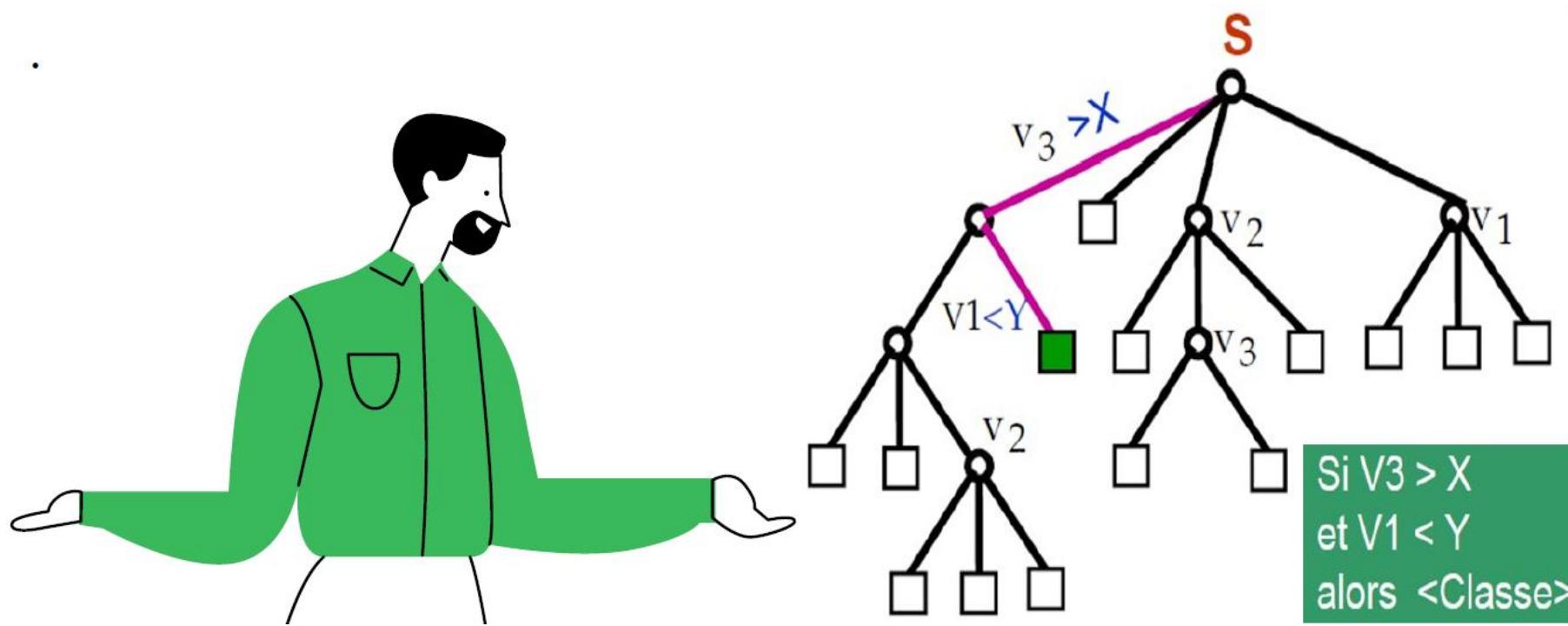


# Appliquer la régression et la classification

## 4. Arbres de décision

- Un parcours:

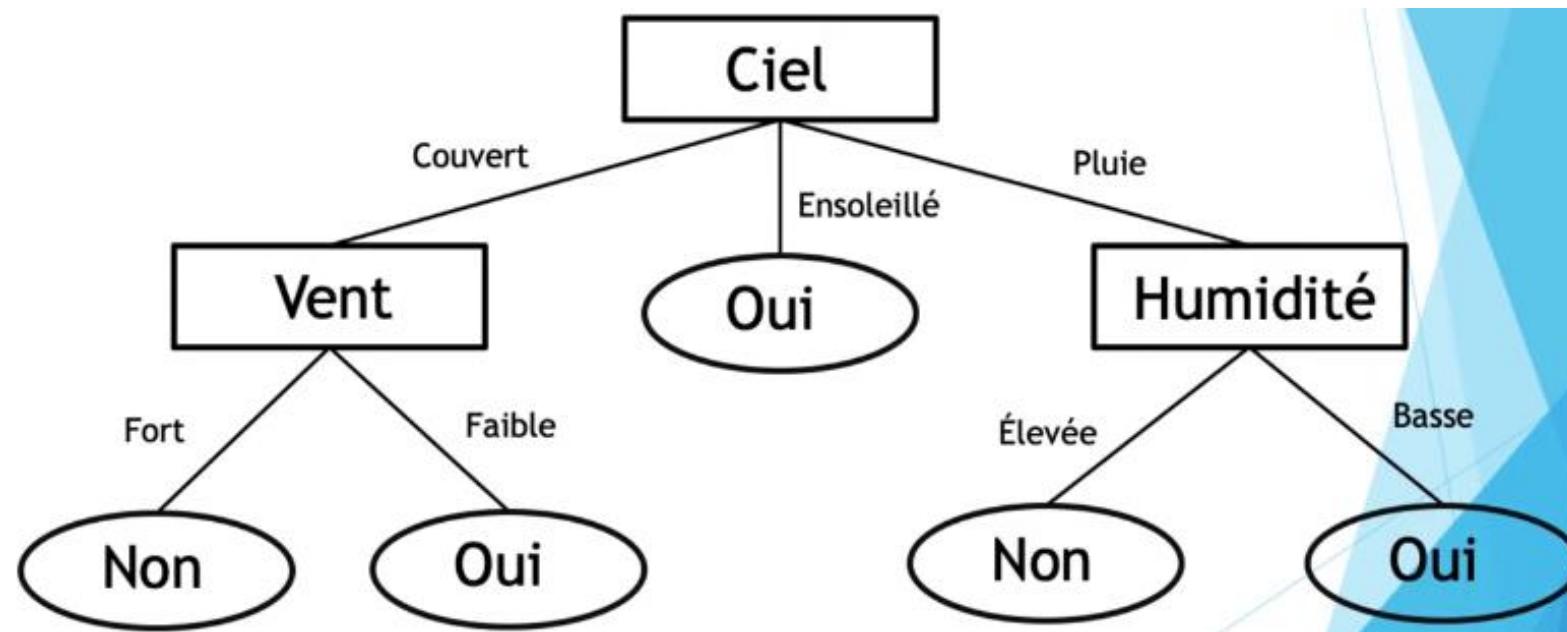
- C'est le chemin complet du nœud racine jusqu'à une feuille.
- Il correspond à une séquence de tests et de réponses.



# Appliquer la régression et la classification

## 4. Arbres de décision

- Exemple: vais-je jouer au tennis?



# Appliquer la régression et la classification

## 4. Arbres de décision

### Le jeu de données

#### Base de données iris

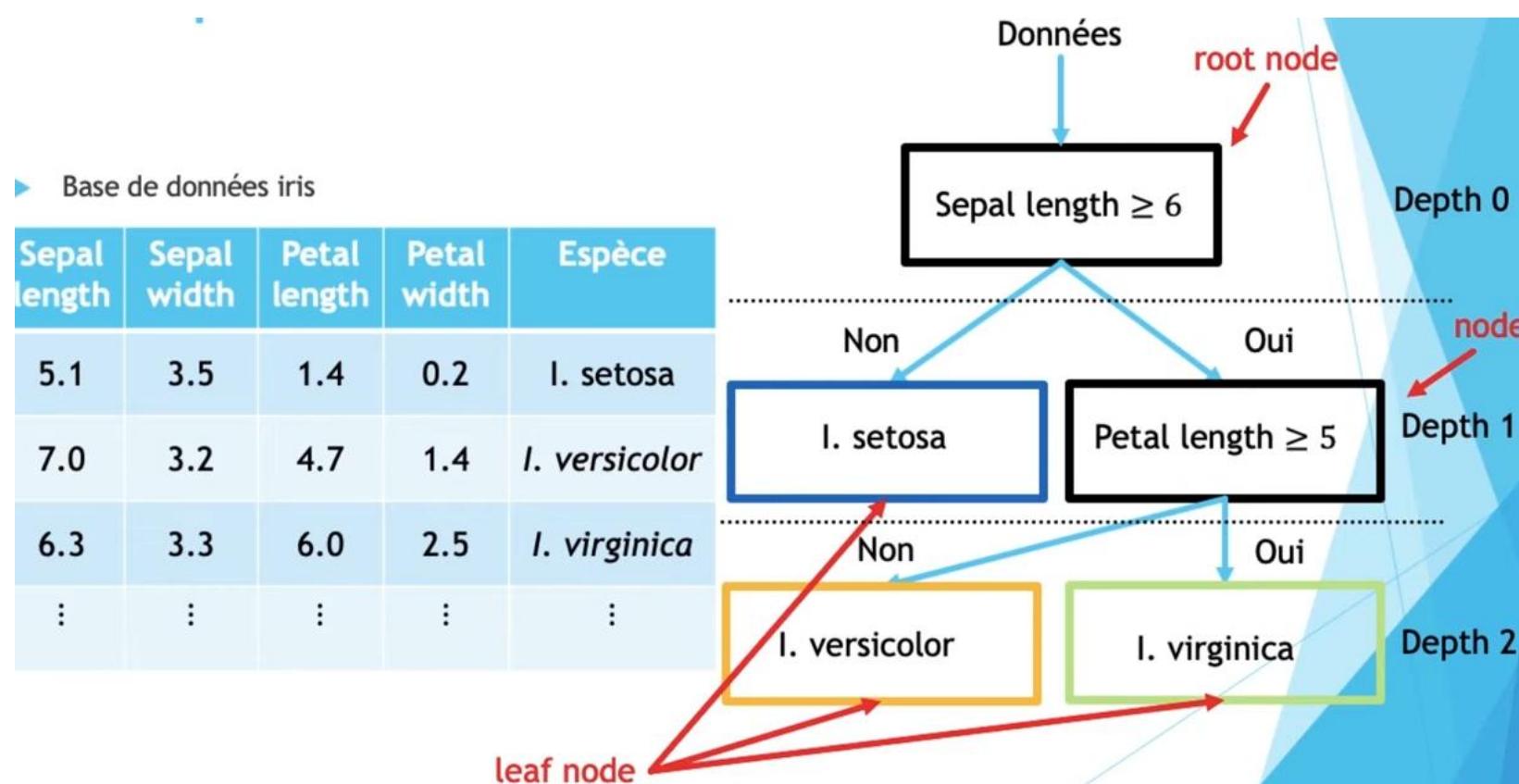
Sepal length	Sepal width	Petal length	Petal width	Espèce
5.1	3.5	1.4	0.2	I. setosa
7.0	3.2	4.7	1.4	I. versicolor
6.3	3.3	6.0	2.5	I. virginica
:	:	:	:	:



# Appliquer la régression et la classification

## 4. Arbres de décision

- Votre premier modèle d'arbre de décision



# Appliquer la régression et la classification

## 4. Arbres de décision

### ▪ Estimation de la probabilité d'appartenance

Un arbre de décision peut également estimer la probabilité qu'une observation appartienne à une classe k particulière.

Par exemple :

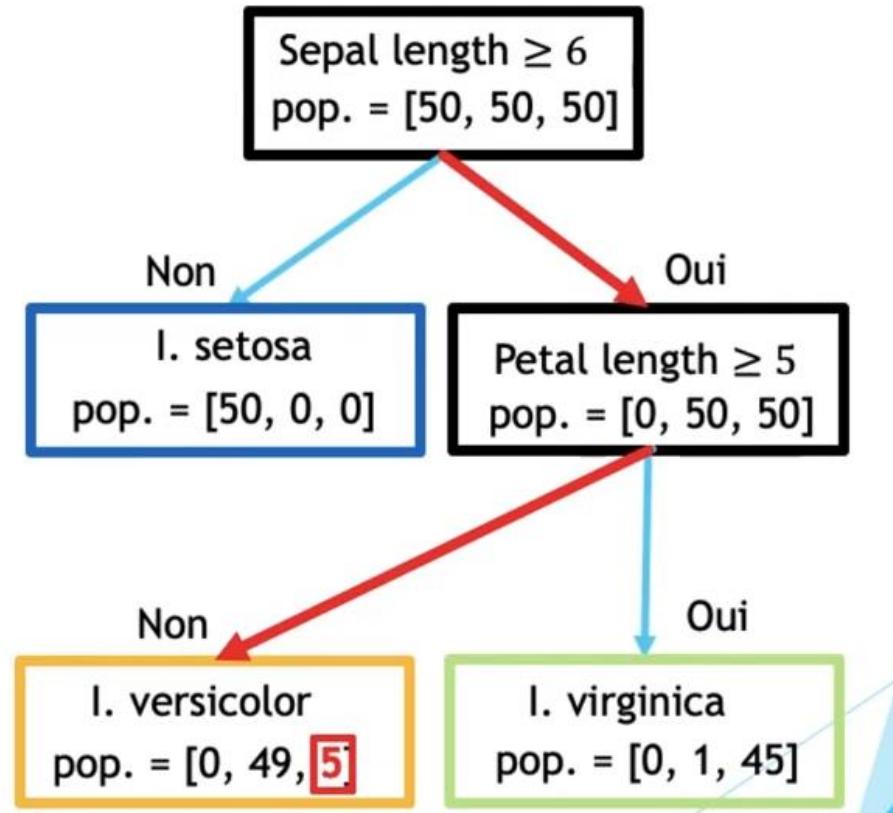
Nous voulons identifier une fleur avec un sépale de 6 cm et un pétales de 4 cm.

$$0/54 = 0\% \text{ Setosa}$$

$$49/54 = 90.7\% \text{ Versicolor}$$

$$5/54 = 9.3\% \text{ Virginica}$$

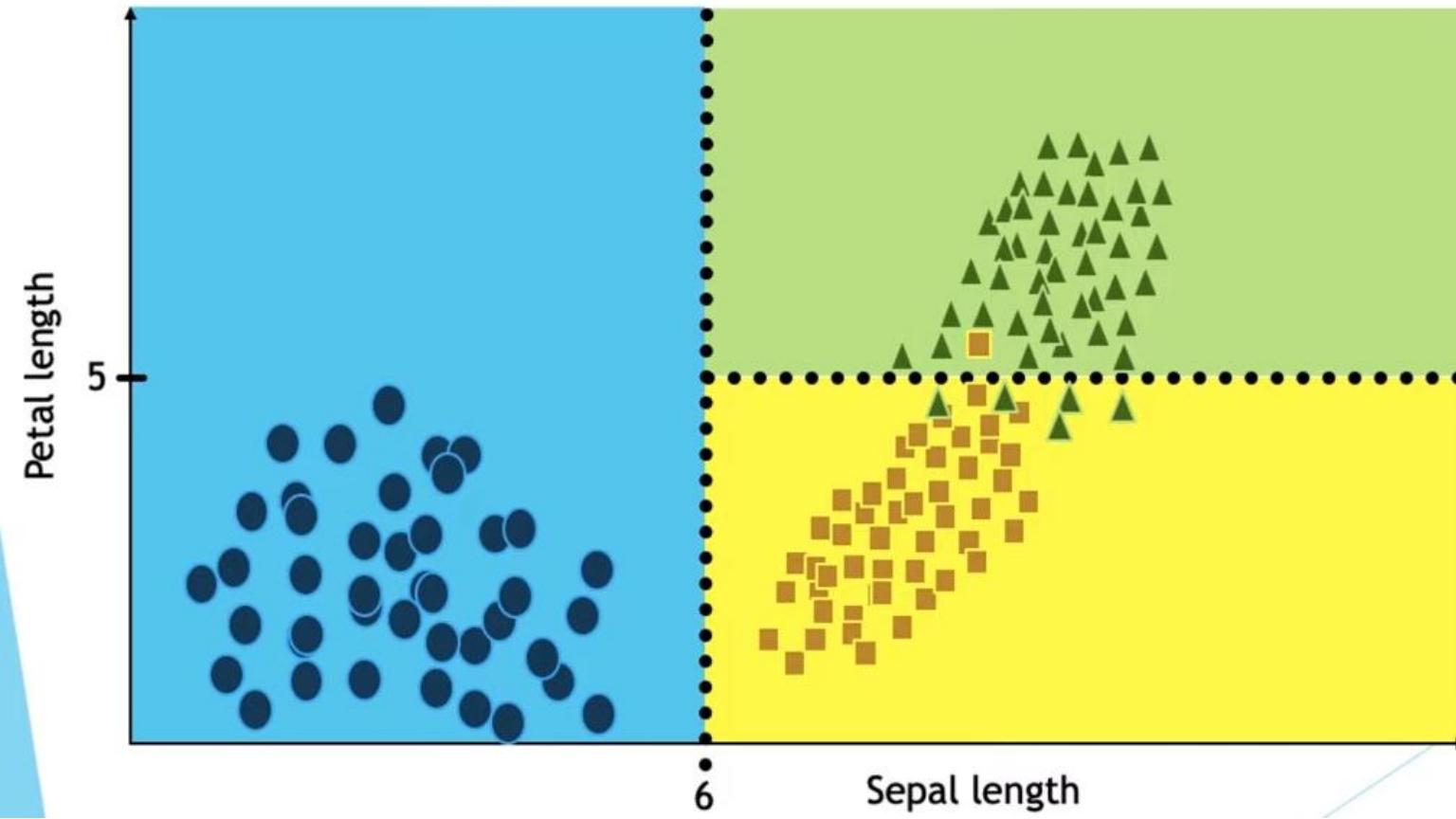
population = [nb setosa, nb versicolor, nb virginica]



# Appliquer la régression et la classification

## 4. Arbres de décision

- La frontière de décision

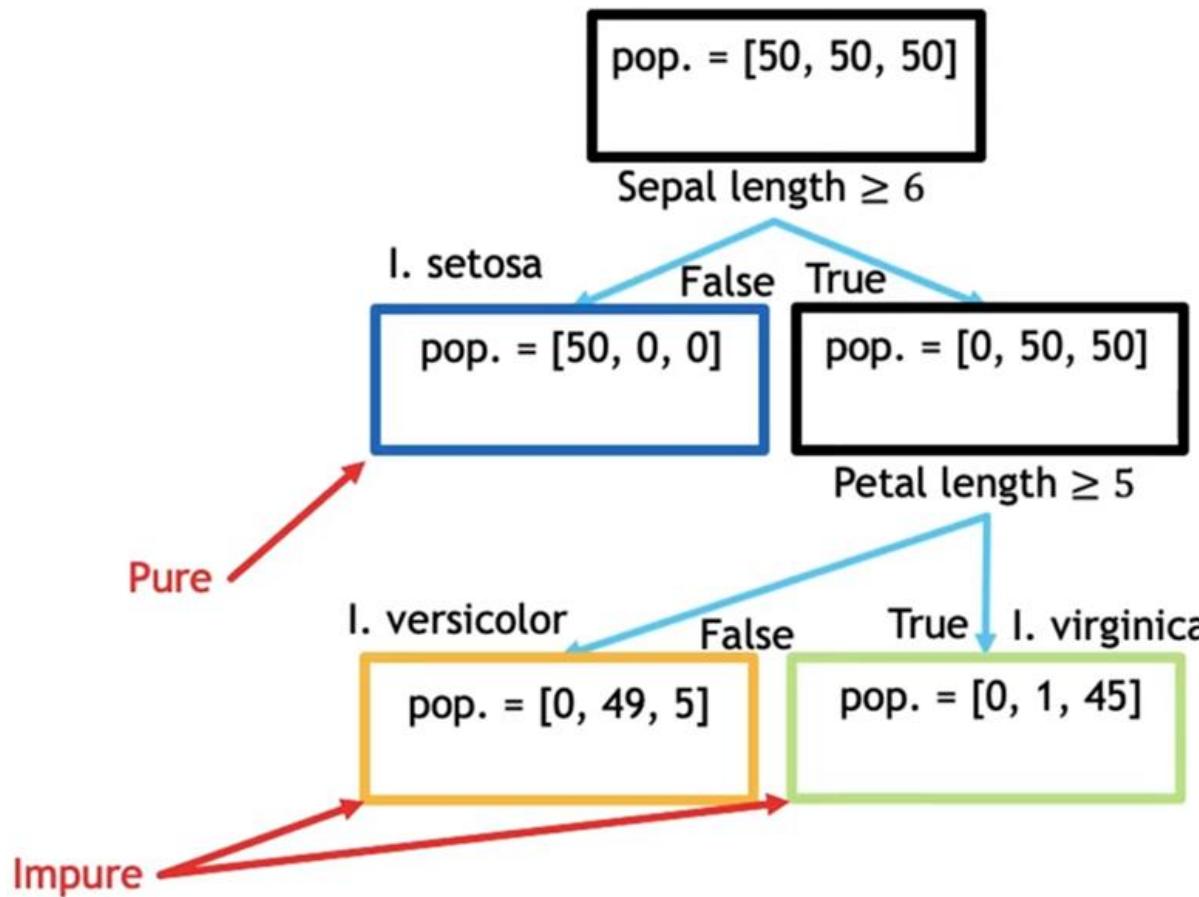


# Appliquer la régression et la classification

## 4. Arbres de décision

- Pureté

population = [nb setosa, nb versicolor, nb virginica]



# Appliquer la régression et la classification

## 4. Arbres de décision

### Indice de Gini

Comment créer la bonne inéquation qui conduira à une meilleure classification ?

$$G_i = 1 - \sum_{k=1}^n p_{i,k}^2$$

$p_{i,k}$  est le ratio du nombre d'individus de la classe k parmi la population du  $i^{\text{ème}}$  noeud.

$$\text{gini} = 1 - \left(\frac{50}{150}\right)^2 - \left(\frac{50}{150}\right)^2 - \left(\frac{50}{150}\right)^2 = 0.667$$

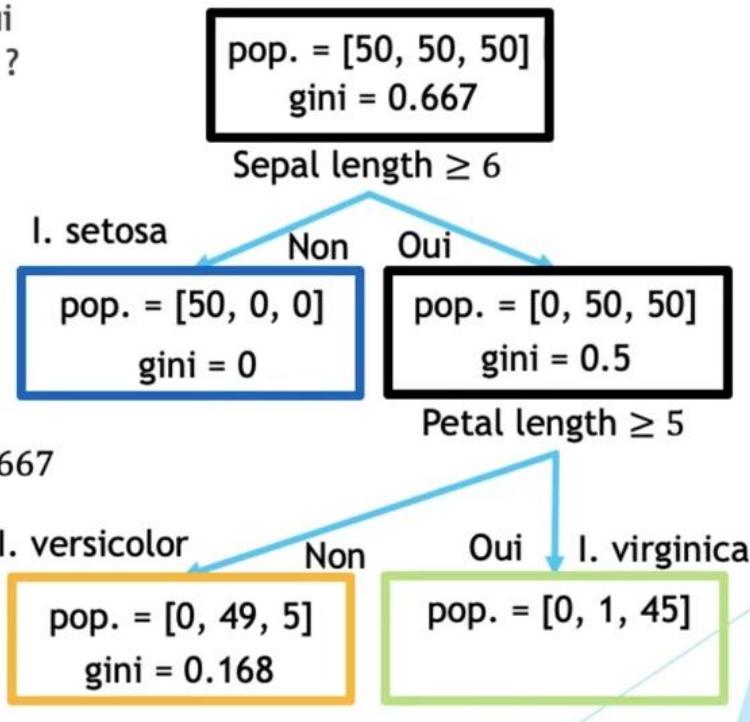
$$\text{gini} = 1 - \left(\frac{50}{50}\right)^2 - \frac{0}{50} - \frac{0}{50} = 0$$

$$\text{gini} = 1 - \frac{0}{50} - \left(\frac{50}{100}\right)^2 - \left(\frac{50}{100}\right)^2 = 0.5$$

$$\text{gini} = 1 - \frac{0}{54} - \left(\frac{49}{54}\right)^2 - \left(\frac{5}{54}\right)^2 = 0.168$$

$$\text{gini} = 1 - \frac{0}{46} - \left(\frac{1}{46}\right)^2 - \left(\frac{45}{46}\right)^2 = 0.043$$

population = [nb setosa, nb versicolor, nb virginica]



# Appliquer la régression et la classification

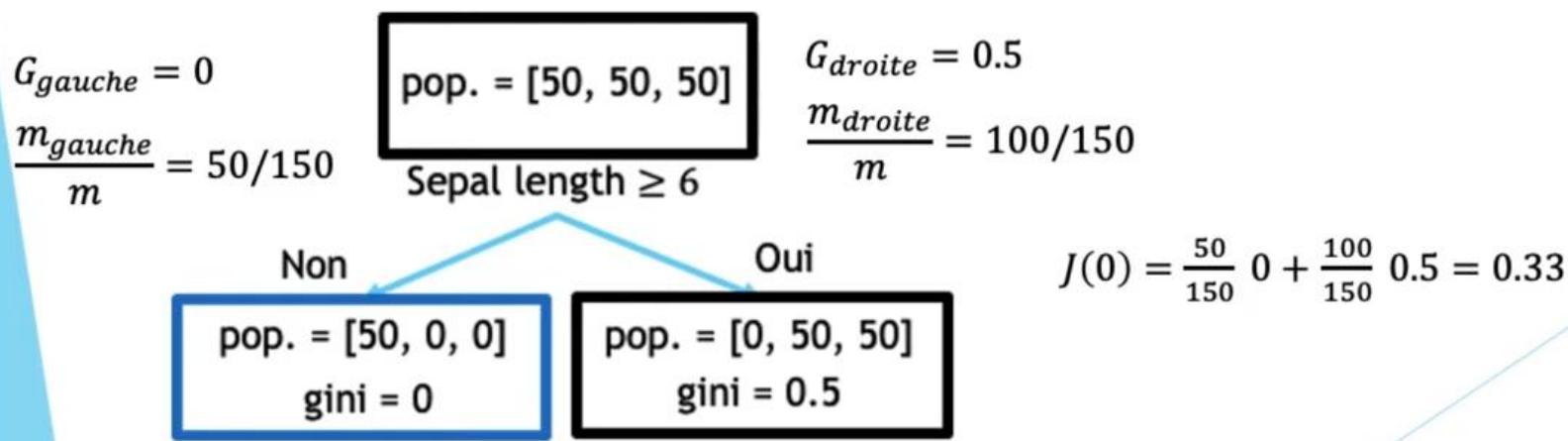
## 4. Arbres de décision

### Coût du nœud

On veut calculer pour la pureté de notre nœud  $k$

$$J(k) = \frac{m_{gauche}}{m} G_{gauche} + \frac{m_{droite}}{m} G_{droite}$$

Ou  $\begin{cases} G_{gauche/droite} \text{ mesure l'impureté du sous ensemble droite/gauche} \\ m_{gauche/droite} \text{ est la proportion de notre population du sous ensemble droite/gauche} \end{cases}$

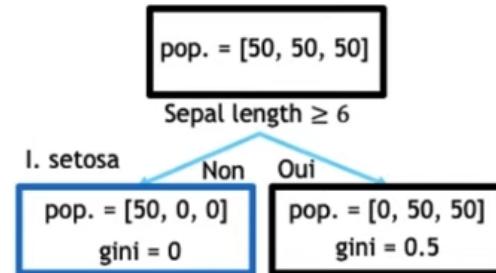


# Appliquer la régression et la classification

## 4. Arbres de décision

### Choisir des noeuds

1)



$$G_{gauche} = 0$$

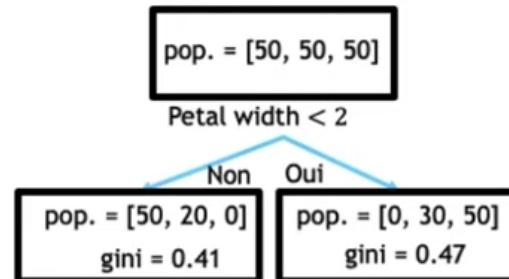
$$m_{gauche} = 50/150$$

$$J(0) = \frac{50}{150} \cdot 0 + \frac{100}{150} \cdot 0.50 = 0.33$$

$$G_{droite} = 0.5$$

$$m_{droite} = 100/150$$

2)



$$G_{gauche} = 0.41$$

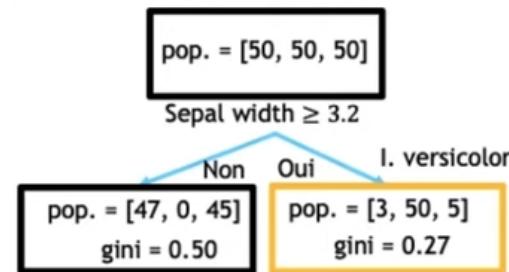
$$m_{gauche} = 70/150$$

$$J(0) = \frac{70}{150} \cdot 0.41 + \frac{80}{150} \cdot 0.47 = 0.44$$

$$G_{droite} = 0.47$$

$$m_{droite} = 80/150$$

3)



$$G_{gauche} = 0.60$$

$$m_{gauche} = 92/150$$

$$J(0) = \frac{92}{150} \cdot 0.0 + \frac{58}{150} \cdot 0.27 = 0.41$$

$$G_{droite} = 0.27$$

$$m_{droite} = 58/150$$

# Appliquer la régression et la classification

## 4. Arbres de décision

- Exemple: jeux de données de prix de maison

► Base de données prix de maison

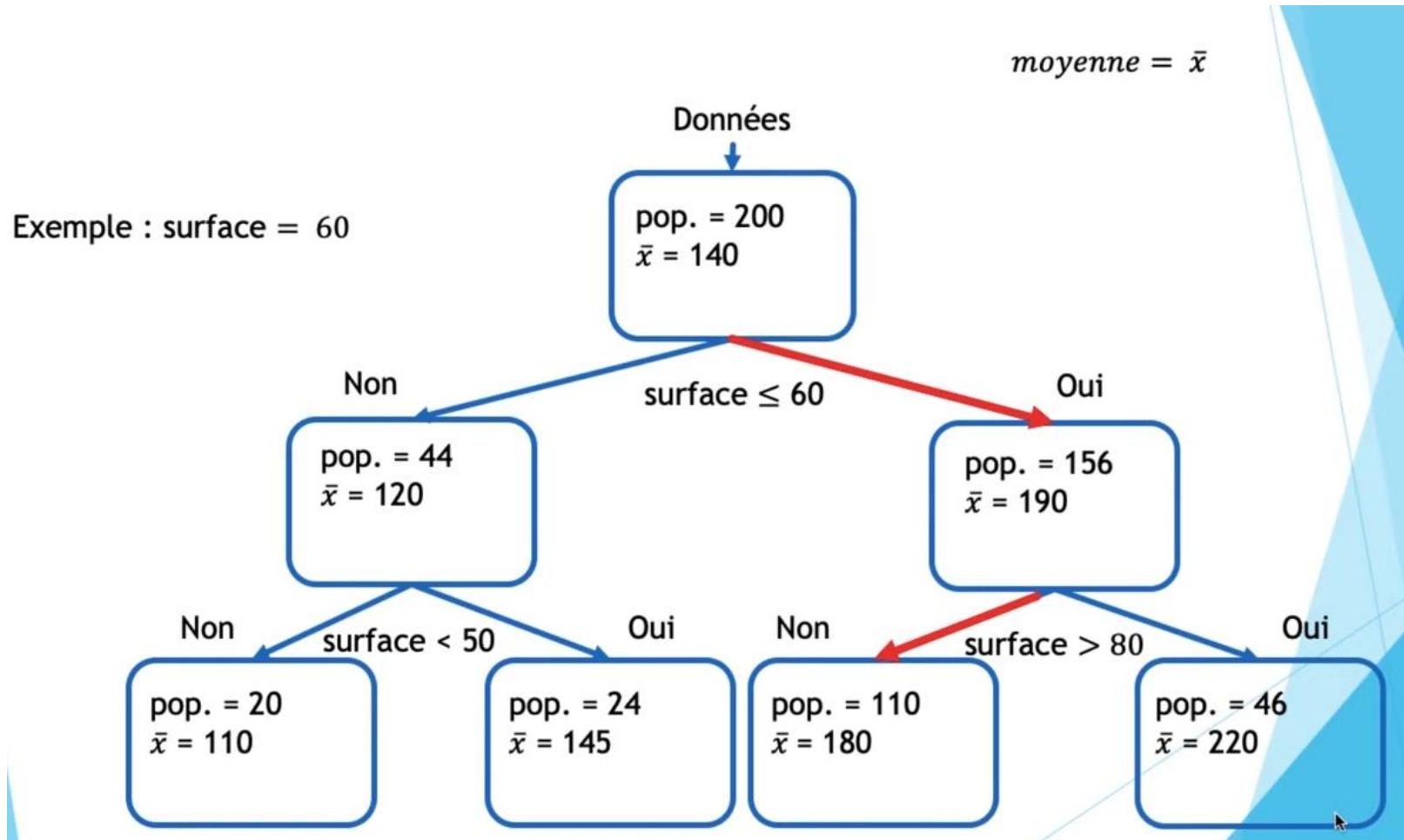
Nb pièces	surface	Garage	Année	Prix (k€)
3	72	1	2017	180
2	58	1	2010	140
3	76	0	1998	160
:	:	:	:	:



# Appliquer la régression et la classification

## 4. Arbres de décision

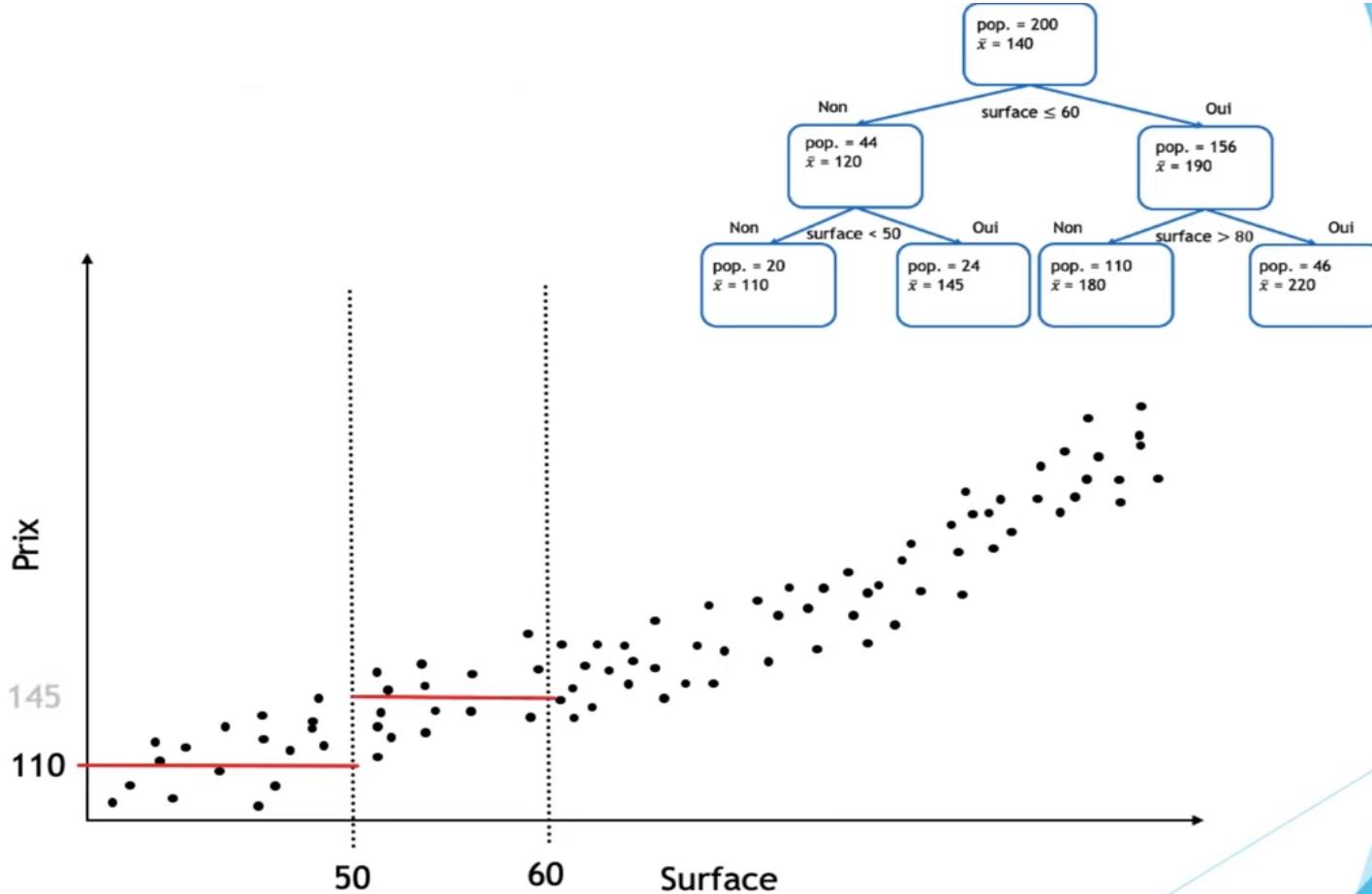
- Exemple: jeux de données de prix de maison



# Appliquer la régression et la classification

## 4. Arbres de décision

- Exemple: frontière de décision



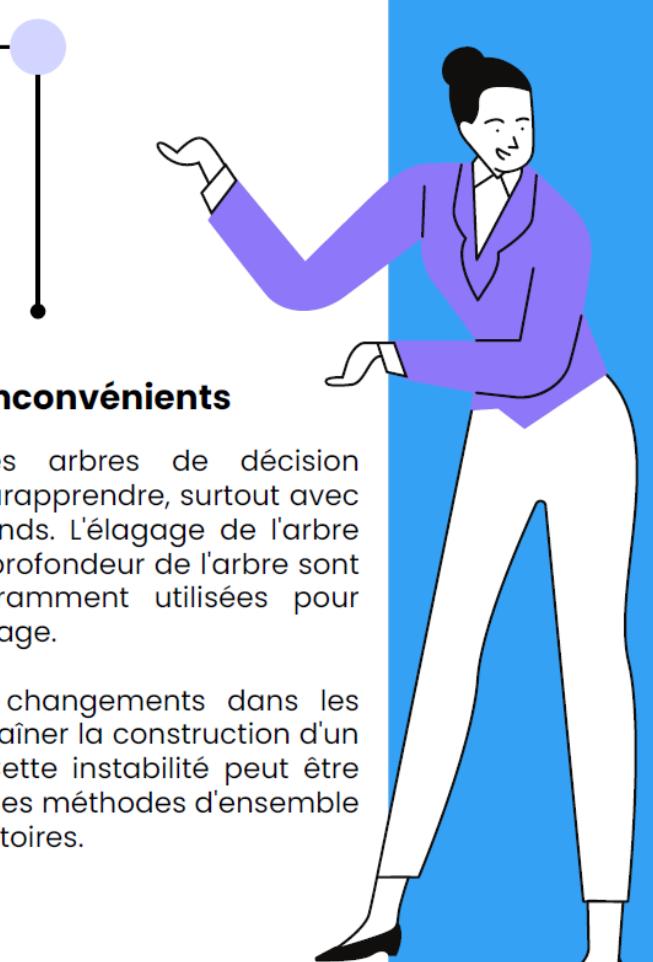
# Appliquer la régression et la classification

## 4. Arbres de décision



### Avantages

- Interprétabilité: Les arbres de décision sont faciles à comprendre et à interpréter, même pour des personnes sans expertise en machine learning.
- Polyvalence: Ils peuvent être utilisés pour des problèmes de classification et de régression.
- Prétraitement minimal: Ils nécessitent peu de prétraitement des données, ne sont pas affectés par la normalisation des caractéristiques et peuvent gérer à la fois des caractéristiques numériques et catégorielles.



### Inconvénients

- Surapprentissage: Les arbres de décision peuvent facilement surapprendre, surtout avec des arbres très profonds. L'élagage de l'arbre ou la limitation de la profondeur de l'arbre sont des techniques couramment utilisées pour éviter le surapprentissage.
- Instabilité: De petits changements dans les données peuvent entraîner la construction d'un arbre très différent. Cette instabilité peut être atténuée en utilisant des méthodes d'ensemble comme les forêts aléatoires.

## ● 4. Arbres de décision

### ■ Rappel: Problème de biais et la variance

#### Biais (Bias)

C'est l'erreur due à un **modèle trop simple**, qui **n'apprend pas assez** des données.

- Il **généralise mal**, car il ne capture pas la complexité du phénomène.
- Exemple : une droite (régression linéaire) pour modéliser une relation non linéaire.

→ **Sous-apprentissage (Underfitting) = Biais élevé, Variance faible**

#### Variance (Variance)

C'est l'erreur due à un **modèle trop complexe**, qui **s'adapte trop** aux données d'entraînement.

- Il capture même le bruit.
- Les performances chutent sur les données de test.

→ **Sur-apprentissage (Overfitting) = Variance élevée, Biais faible**

# Appliquer la régression et la classification

## 4. Arbres de décision

### Les principales méthodes d'arbre de décision

Méthode	Auteur / Année	Type	Critère de découpe	Particularités
ID3 (Iterative Dichotomiser 3)	Quinlan, 1986	Classification	Entropie (Information Gain)	Premier algorithme historique ; ne gère pas les valeurs continues ni le surapprentissage.
C4.5	Quinlan, 1993	Classification	Gain Ratio (rapport du gain d'information)	Améliore ID3 : gère valeurs continues, attributs manquants, élagage, choix du meilleur attribut.
C5.0	Quinlan, 1997	Classification	Gain Ratio	Version commerciale améliorée (plus rapide, moins de mémoire).
CART (Classification And Regression Trees)	Breiman et al., 1984	Classification / Régression	Indice de Gini (ou réduction de variance pour régression)	Méthode la plus utilisée — c'est celle implémentée dans scikit-learn.
CHAID (Chi-squared Automatic Interaction Detector)	Kass, 1980	Classification	Test du $\chi^2$ (Chi <sup>2</sup> )	Produit des arbres multiniveaux (plus de 2 branches).
QUEST (Quick, Unbiased, Efficient Statistical Tree)	Loh & Shih, 1997	Classification	Test de séparation statistique (p-value)	Évite les biais de sélection d'attributs.

## 4. Arbres de décision

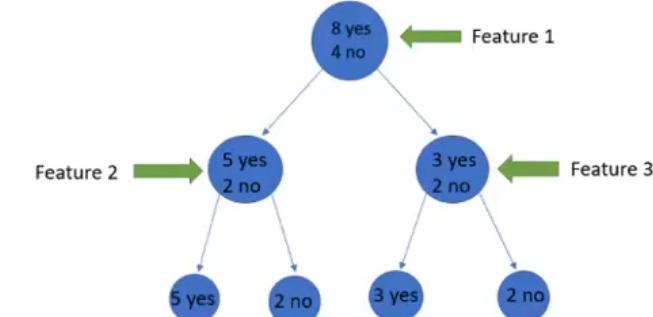
### ▪ Méthodes utilisées par un arbre de décision

- Critères de découpe (impureté)
  - `gini` (Indice de Gini) – par défaut en classification.
  - `entropy` (Information gain / entropie de Shannon).
  - `log_loss` (perte logistique, proche de l'entropie).
  - (*Régression*) : réduction de variance ( `squared_error` ), `friedman_mse` , `poisson` .
- Stratégie de split
  - `splitter="best"` : teste la meilleure coupure à chaque nœud.
  - `splitter="random"` : choisit aléatoirement parmi des coupures candidates (utile pour réduire la variance).
- Pruning (élagage)
  - Pré-élagage : limite la croissance de l'arbre pour éviter l'overfitting.
  - Post-élagage via la complexity pruning : contrôle par `ccp_alpha` (coût-complexité minimale).

# Appliquer la régression et la classification

## 4. Arbres de décision

- Former un arbre par division récursive des caractéristiques
- Paramètres:
  - Hyperparamètres:
    - Profondeur maximale de l'arbre
  - Paramètres entraînables:
    - Critère de division



## 4. Arbres de décision

### ▪ Méthodes utilisées par un arbre de décision

- Critères de découpe (impureté)
  - `gini` (Indice de Gini) – par défaut en classification.
  - `entropy` (Information gain / entropie de Shannon).
  - `log_loss` (perte logistique, proche de l'entropie).
  - (*Régression*) : réduction de variance ( `squared_error` ), `friedman_mse` , `poisson` .
- Stratégie de split
  - `splitter="best"` : teste la meilleure coupure à chaque nœud.
  - `splitter="random"` : choisit aléatoirement parmi des coupures candidates (utile pour réduire la variance).
- Pruning (élagage)
  - Pré-élagage : limite la croissance de l'arbre pour éviter l'overfitting.
  - Post-élagage via la complexity pruning : contrôle par `ccp_alpha` (coût-complexité minimale).

## 4. Arbres de décision

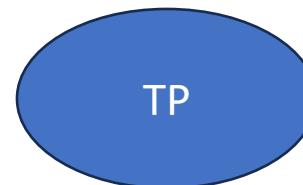
### ▪ Hyperparamètres d'entraînement (principaux)

- `max_depth` : profondeur max de l'arbre (régularisation clé).
- `min_samples_split` : nb mini d'échantillons pour scinder un nœud.
- `min_samples_leaf` : nb mini dans une feuille (fort pour lisser les frontières).
- `max_leaf_nodes` : nb max de feuilles.
- `min_impurity_decrease` : seuil minimal de gain d'impureté pour accepter un split.
- `criterion` : `gini` / `entropy` / `log_loss`.
- `splitter` : `best` / `random`.
- `class_weight` : équilibrer les classes ( "balanced" ).
- `random_state` : reproductibilité.
- `ccp_alpha` : élagage par coût-complexité (plus grand  $\Rightarrow$  arbre plus simple).

## ● 4. Arbres de décision

### ■ Bonnes pratiques

- Toujours séparer train/test (et idéalement validation croisée).
- Standardisation non requise pour les arbres.
- Gérer les catégories : encoder (One-Hot) si nécessaire.
- Gérer les valeurs manquantes (imputation) — `DecisionTreeClassifier` n'accepte pas `NaN`.



## ● 4. Forêt aléatoire

- **Random forest (foret aleatoire)** est un classifieur ensembliste qui utilise plusieurs arbres de décision de type CART.
- Une forêt aléatoire (Random Forest) est un ensemble d'arbres de décision construits à partir de sous-échantillons aléatoires du jeu de données.
- Le terme « random decision forests » est initialement proposé Tin Kam Ho of Bell Labs in 1995.
- [Breimans 2001] a proposé la version mature de “Random forest” en utilisant deux idées:
  - Bagging** (Bootstrap Aggregating).
  - La sélection aléatoire des attributs (random space selection)

**Objectif : Réduire la variance des arbres de décision tout en maintenant un faible biais.**

## ● 4. Forêt aléatoire

- Prérequis
  - Les arbres de décision
  - L'ensemble Learning

## ● 4. Forêt aléatoire

### ▪ Les problèmes d'arbre de décision

- ▶ Les arbres de décision sont facile à entraîner, facile à utilisé et très interprétable, on peut savoir assez rapidement les règles qui permettent la prédiction ou la classification de notre exemple.
  
- ▶ Mais derrière ces qualités apparentes on peut leur reprocher de ne pas être assez précis ni assez généralisable. En effet, l'arbre est performant avec le jeu d'entraînement mais il n'est pas assez flexible pour donner de bonnes performances sur des données de test. Or, but de l'un algorithme de machine learning est de pouvoir classifier des données dont on ne connaît pas encore la prédiction donc ce sont des données autres que celle d'entraînement.

## ● 4. Forêt aléatoire

### ■ Arbre CART

-CART = *Classification And Regression Tree*

- Fonctionne par **partition récursive** de l'espace des attributs.

-Critère d'impureté :

- Gini pour la classification
- MSE pour la régression

**Problème :** Un seul arbre → instable, forte variance.

**Solution :** Agréger plusieurs arbres → Random Forest.

## ● 4. Forêt aléatoire

### ▪ Palier les problèmes d'arbre de décision

- ▶ Pour palier à ce problème de généralisation, les random forest ont vu le jour. Cet algorithme permet d'utiliser plusieurs arbres de décision afin de créer une forêt et d'améliorer la généralisation de l'ensemble du modèle. On va emprunter la théorie de l'ensemble learning afin de faire fonctionner ses arbres ensemble.
  
- ▶ Le random forest combine donc la simplicité des arbres de décision grâce à l'ensemble learning afin de gagner de la flexibilité et de la généralisation. Ce qui donne des modèles de bien meilleur qualité.

## ● 4. Forêt aléatoire

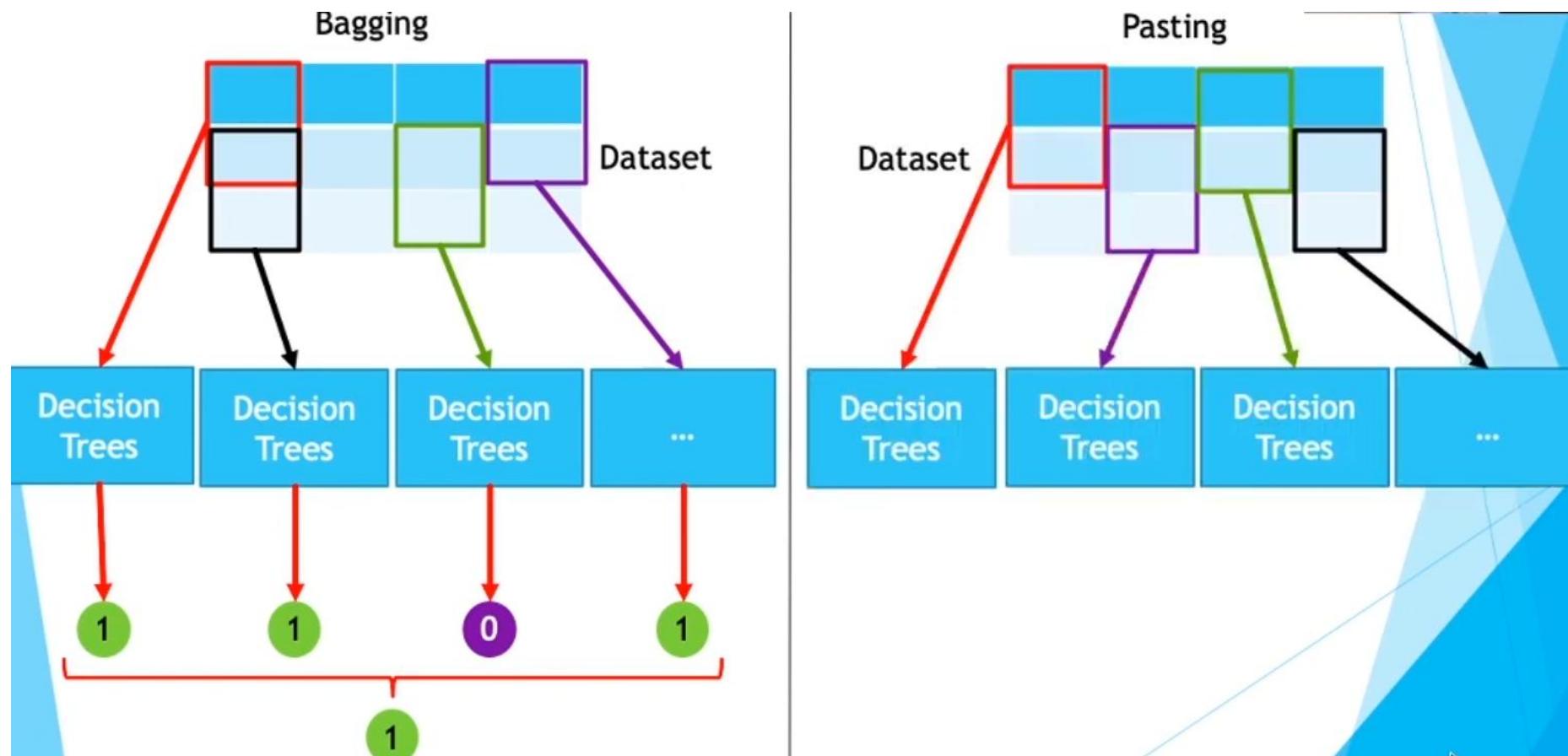
### ▪ Méthodes d'ensemble

- **Principe:** entraîner plusieurs classificateurs et utiliser l'union de ces méthodes lors de la décision finale (moyenne des résultats individuels ou vote majoritaire)
- Deux classes de méthodes ensemblistes
  - Les méthodes à base de boosting
  - Les méthodes à base de bagging
- Pour le boosting l'apprentissage se fait **séquentiellement**, et les classificateurs doivent être faibles « weak learner» (de même type ou non)
- Pour le bagging, l'apprentissage se fait **simultanément**, et les classificateurs individuels doivent être de même type

# Appliquer la régression et la classification

## 4. Forêt aléatoire

### ▪ Bagging & Pasting



# Appliquer la régression et la classification

## 4. Forêt aléatoire

### ▪ Bagging

Le **Bagging** (pour *Bootstrap Agrégation*) est une **méthode d'ensemble** visant à **réduire la variance** d'un modèle instable (comme un arbre de décision)

En combinant plusieurs versions du même modèle entraînées sur des **échantillons aléatoires** du jeu de données.

#### ⚙️ Principe de fonctionnement

##### 1 Échantillonnage aléatoire avec remise (Bootstrap)

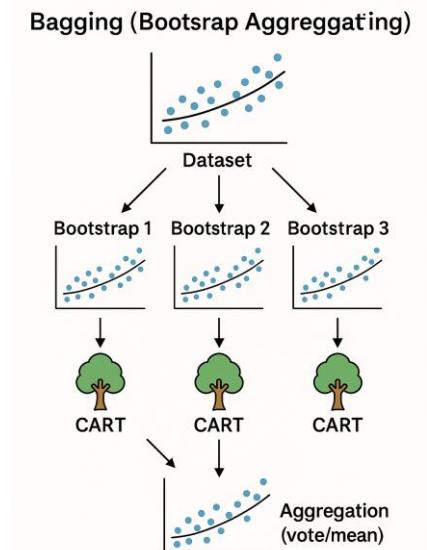
- On crée plusieurs sous-ensembles du jeu d'entraînement en tirant **avec remise**.
- Chaque sous-ensemble est de même taille que l'ensemble original.

##### 2 Entraînement multiple

- On entraîne un **modèle de base** (souvent un arbre CART) sur chaque échantillon bootstrapé.
- Chaque modèle apprend une version légèrement différente du problème.

##### 3 Agrégation des prédictions

- En **classification** → Vote majoritaire 📁
- En **régression** → Moyenne arithmétique 📈



# Appliquer la régression et la classification

## 4. Forêt aléatoire

### Creation du dataset

Data set du Random Forest

Nb de pièces	Surface	Garage	Prix
4	80	1	220 000
3	70	0	190 000
2	40	1	140 000
4	60	0	170 000
3	70	1	200 000

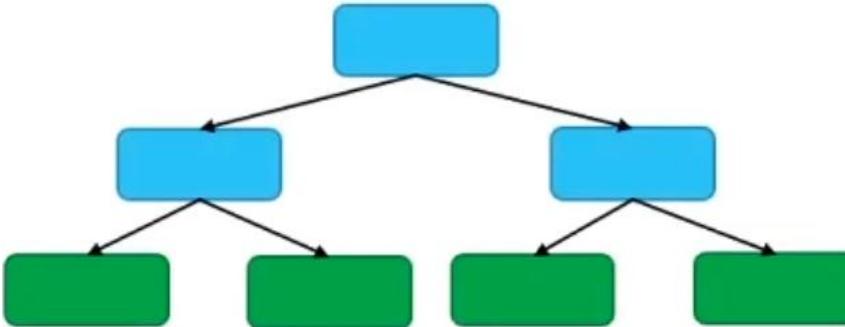
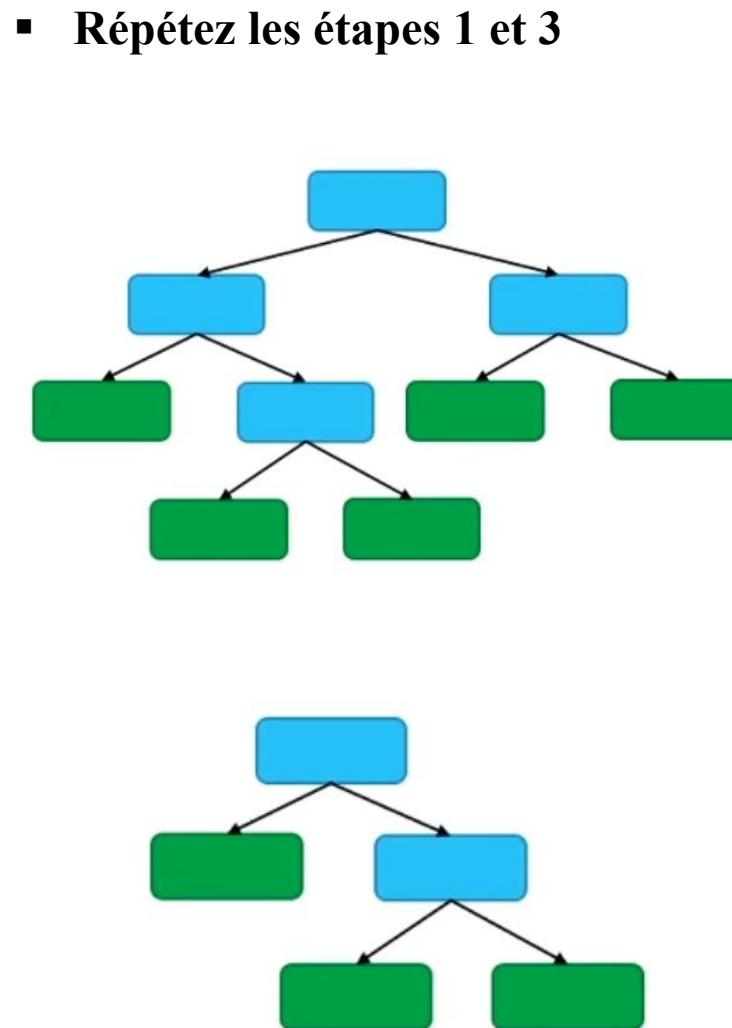
Data set de l'arbre de décision

Nb de pièces	Surface	Garage	Prix
3	70	0	190 000
4	60	0	170 000
3	70	1	200 000
3	70	0	190 000

Pour maximiser la variété des arbres !

# Appliquer la régression et la classification

## 4. Forêt aléatoire

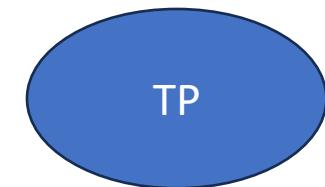
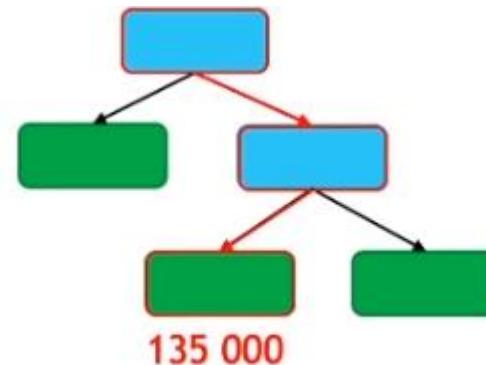
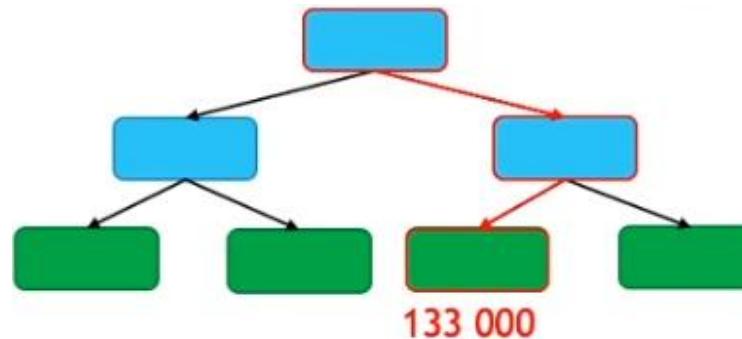
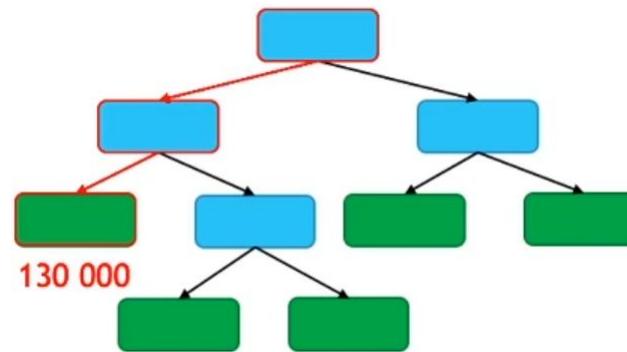


# Appliquer la régression et la classification

## 4. Forêt aléatoire

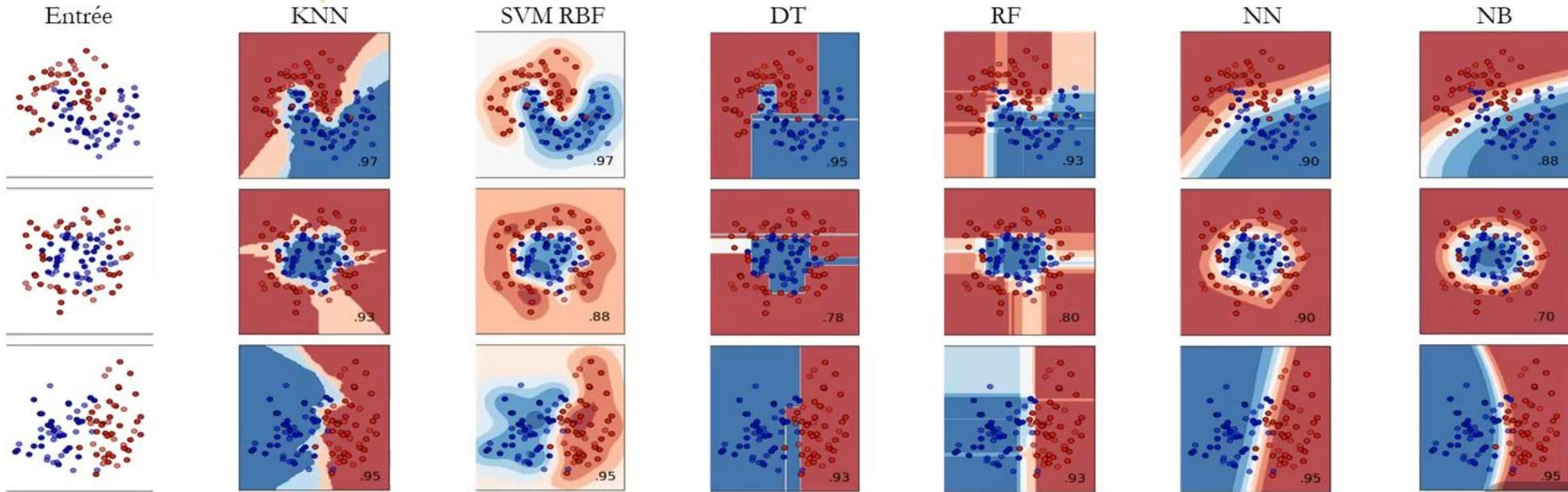
- Répétez les étapes 1 et 3

Prédiction moyenne: 132 666



# Appliquer la régression et la classification

## Quel algorithme choisir?



# Appliquer la régression et la classification

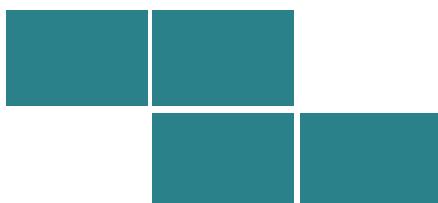
## ● Quiz

- Quel algorithme est le plus adapté à un problème de classification binaire ?
  - a) Régression linéaire
  - b) K-Means
  - c) SVM
  - d) PCA
- Quel algorithme n'a pas de paramètres entraînables
  - a) KNN
  - b) SVM
  - c) Régression Logistique
  - d) DT



## Chapitre 3-4: Algorithmes ML: Reduction de dimension

- ✓ ACP (Analyse en Composantes Principales)
- ✓ UMAP (Uniform Manifold Approximation and Projection)
- ✓ t-SNE



## ● ACP (Analyse en Composantes Principales)

- Introduction

- Les jeux de données contiennent souvent **beaucoup de variables** (parfois corrélées).
- Les algorithmes de ML (K-means, SVM, etc.) sont sensibles à :
- La **redondance** des variables,
- Le **bruit** et la **corrélation**.

**Solution :** Projeter les données dans un espace de plus faible dimension tout en gardant l'essentiel de l'information → ACP.

## ● ACP (Analyse en Composantes Principales)

- Introduction
- L'ACP est un **outil puissant de prétraitement** pour ML.
- Elle permet de **résumer** l'information tout en **réduisant le surapprentissage**.
- Idéale avant :
  - **K-means**
  - **SVM**
  - **Réseaux de neurones**
  - Étape clé dans tout **pipeline de Data Science**.

## ● ACP (Analyse en Composantes Principales)

- Pourquoi ACP peut aider les algorithmes de ML

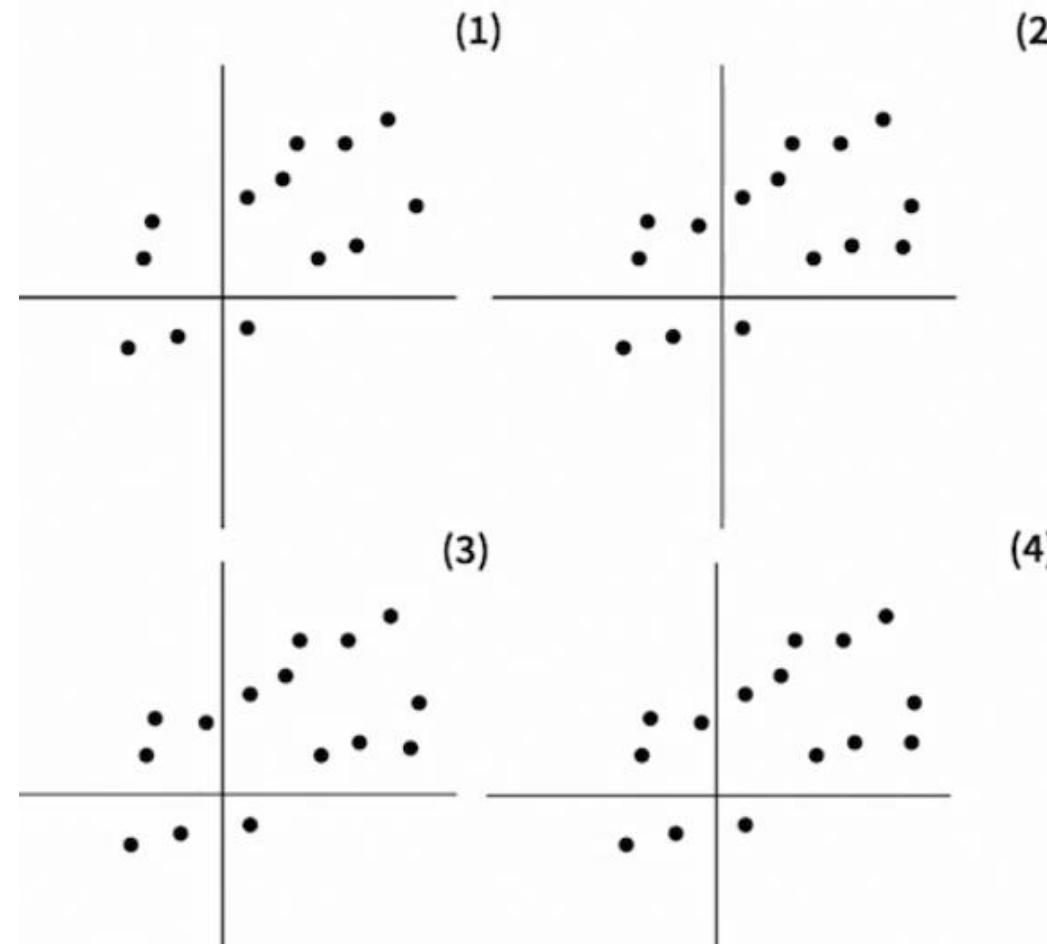
L'Analyse en Composantes Principales (ACP) n'est pas obligatoire, mais elle est **fortement recommandée** quand :

- Les données sont de **haute dimension ou corrélées**,
- Tu veux **améliorer la stabilité et la visualisation**,
- Tu veux **accélérer** le clustering.

Situation	Problème	Pourquoi l'ACP aide
Variables corrélées	Redondance, distances biaisées	ACP produit des variables indépendantes
Beaucoup de variables ( $p > 10$ )	Bruit, surapprentissage	ACP résume l'information en quelques axes
Données bruitées	Clusters mal formés	ACP capture la structure principale
Visualisation souhaitée	K-means agit en espace n-dim	ACP permet une représentation 2D claire

# Algorithmes ML: Reduction de dimension

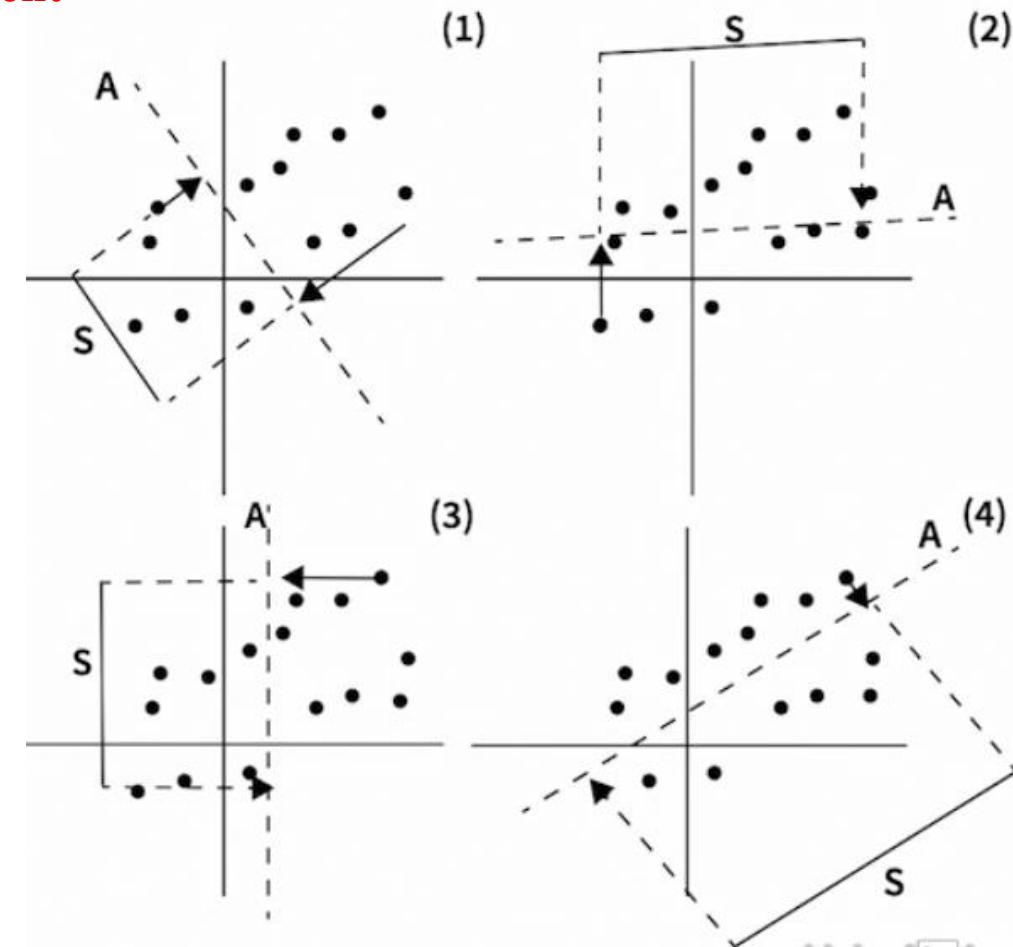
- ACP (Analyse en Composantes Principales)



# Algorithmes ML: Reduction de dimension

## ● ACP (Analyse en Composantes Principales)

Objectif: Rechercher de plus grand segment



## ● ACP (Analyse en Composantes Principales)

### ▪ Les Étapes de l'ACP

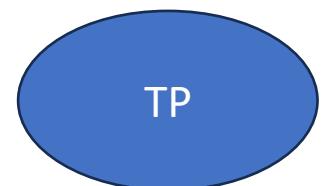
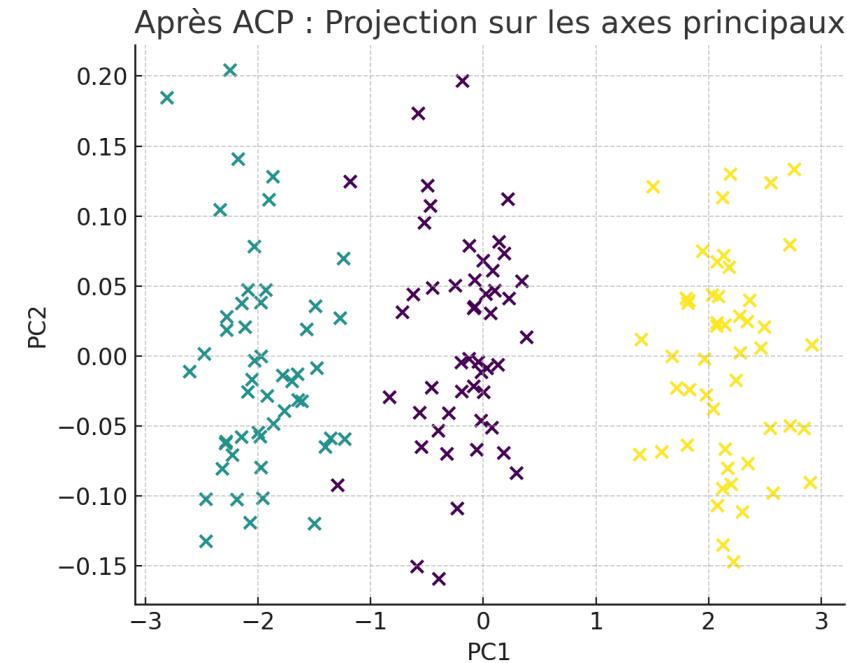
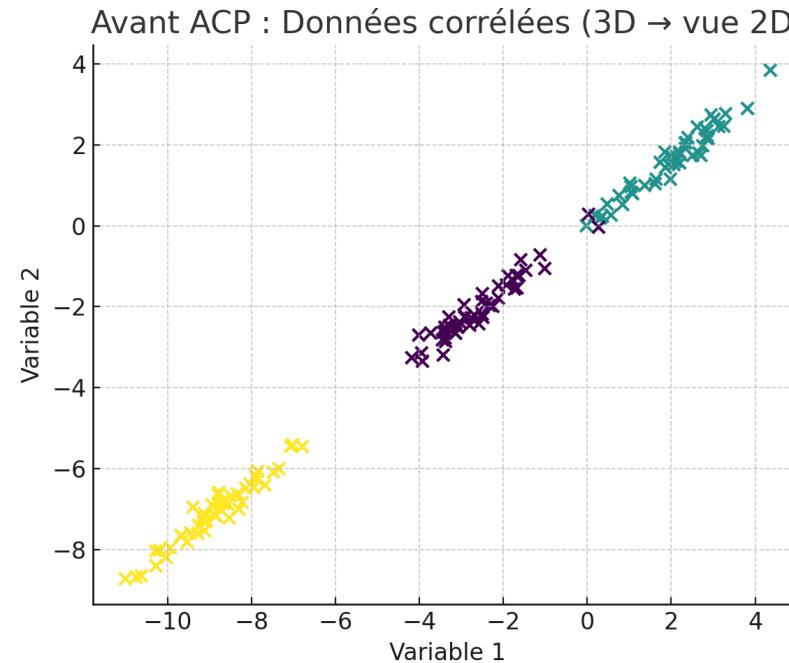
- 1 Standardiser les données**
- 2 Calculer la matrice de covariance / corrélation**
- 3 Extraire les valeurs propres et vecteurs propres**
- 4 Construire les composantes principales**
- 5 Projeter les données**
- 6 Interpréter la variance expliquée**

Données brutes → Nettoyage → Standardisation → ACP → Nouvelles dimensions → Modèle ML (K-means)

# Algorithmes ML: Reduction de dimension

## ● ACP (Analyse en Composantes Principales)

### ▪ Après ACP : Projection sur les axes principaux



À gauche : données initiales sur trois variables corrélées (vue 2D simplifiée).

À droite : données projetées sur les deux premières composantes principales (PC1, PC2) après ACP, montrant une meilleure séparation des clusters potentiels.



**Merci**

