



TORNADE.IO
WEB - CYBERSÉCURITÉ



EMILIE

BOUT

Co-Fondateur de [tornade.io](#) et de [learning.tornade.io](#)

Docteur en cybersécurité à Tornade.io

emilie@tornade.io

Traitement de données avec Python

13/10/2025 - Next-U

Visualisation des données avec Matplotlib

Partie 1- 45 -60 minutes



- Identifier les tendances, relations et pattern
- Déetecter les anomalies
- Transformer les chiffres en information visuelle claire

Une Anomalie ?

Une **anomalie** (ou **outlier**) est une observation qui **s'écarte significativement** du comportement attendu d'une série de données.

Tableau 1

Type	Description	Exemple
Point isolé	Une valeur unique très différente des autres	Température 40°C alors que la moyenne est 15°C
Anomalie collective	Plusieurs points qui forment un motif inhabituel	Plusieurs jours consécutifs de pluie intense inhabituelle
Changement de tendance	Le comportement global change brusquement	Température qui chute soudainement de 20°C à 5°C
Saisonnalité inhabituelle	Variation qui ne correspond pas au cycle attendu	Pic de pluie en plein été dans une zone sèche

Une Anomalie ?

Pourquoi détecter des anomalies ?

- Identifier des **erreurs de mesure** ou de saisie
- Déetecter des **événements exceptionnels** (climatiques, économiques, industriels...)
- Améliorer la **qualité des données** pour le Machine Learning
- Déetecter des **fraudes** ou comportements inhabituels

Méthodes statistiques simples:

- Ecart Type une valeur est considérée comme anormale si elle est trop loin de la moyenne
- **Interquartile Range (IQR)** : basée sur le 1er et 3e quartile

```
moyenne = df["temperature"].mean()  
ecart_type = df["temperature"].std()  
anomalies = df[(df["temperature"] > moyenne + 3*ecart_type) |  
               (df["temperature"] < moyenne - 3*ecart_type)]
```

```
Q1 = df["temperature"].quantile(0.25)  
Q3 = df["temperature"].quantile(0.75)  
IQR = Q3 - Q1  
anomalies = df[(df["temperature"] < Q1 - 1.5*IQR) | (df["temperature"] > Q3 + 1.5*IQR)]
```

Visualiser des données

Matplotlib:

Matplotlib est une bibliothèque Python pour créer des graphes et visualisations 2D.

- Elle est très utilisée pour explorer les données, vérifier des tendances et communiquer les résultats.
- Le module principal pour tracer s'appelle pyplot.

```
import matplotlib.pyplot as plt
```

Visualiser des données temporelles

Objectif :

Apprendre à représenter des données qui évoluent dans le temps (séries temporelles).

Exemple :

Visualiser un DataFrame avec des dates et une valeur qui varie (ex. température).

```
# Création de données temporelles simulées

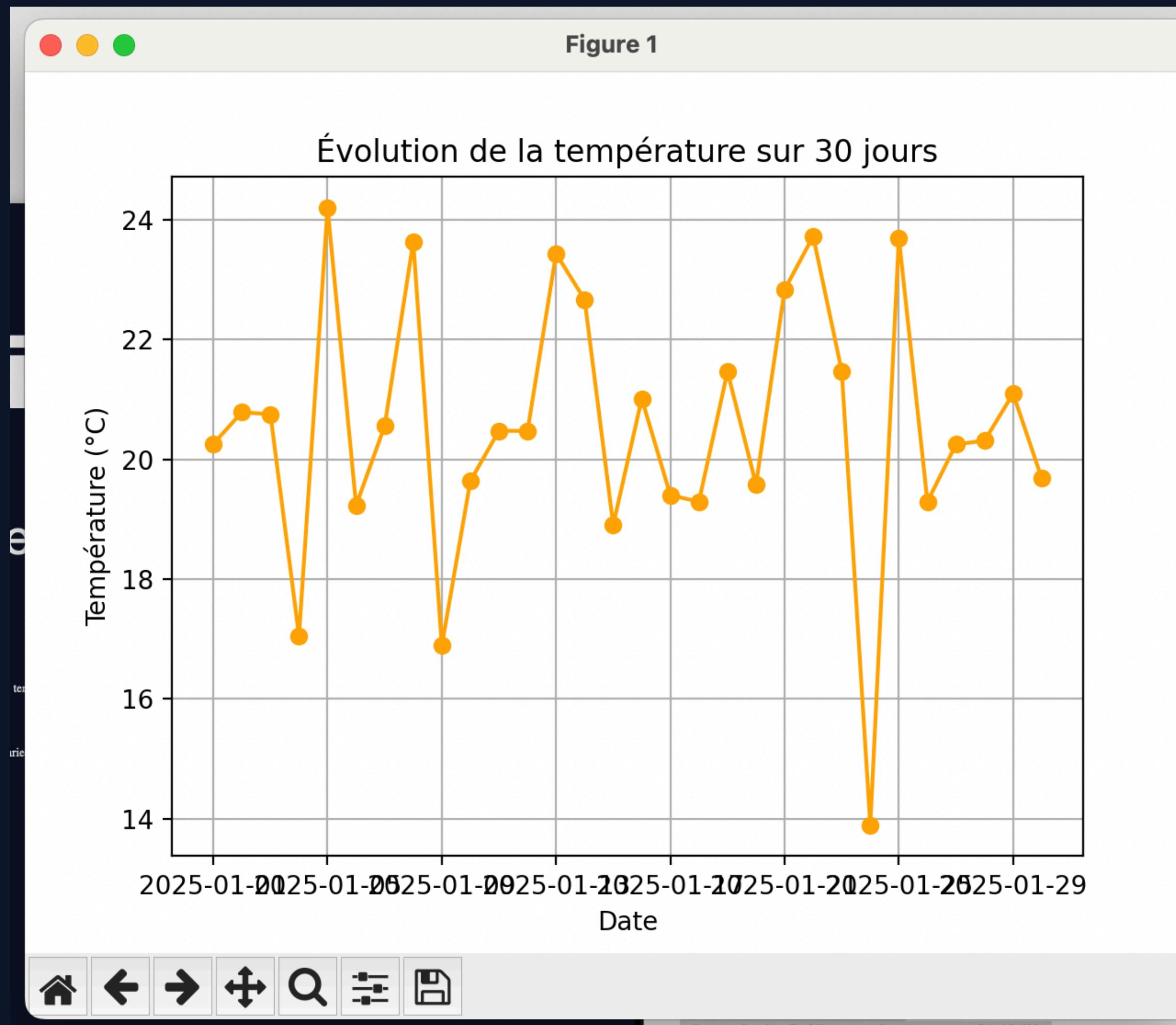
dates = pd.date_range(start="2025-01-01", periods=30, freq="D")
temperature = np.random.normal(20, 2, size=30) # température
autour de 20°C

df = pd.DataFrame({"date": dates, "temperature": temperature})

# Tracé

plt.plot(df["date"], df["temperature"], color="orange",
marker="o")
plt.title("Évolution de la température sur 30 jours")
plt.xlabel("Date")
plt.ylabel("Température (°C)")
plt.grid(True)
plt.show()
```

Visualiser des données



Problèmes :

Impossible de lire les dates correctement et ne se suivent pas...

Structures de données avancées

Comment faciliter les tracés temporels ?

Dans un DataFrame **Pandas**, chaque ligne a un **index** – c'est comme une *étiquette unique* qui identifie la ligne

```
date      temperature
0 2025-01-01      18.2
1 2025-01-02      19.1
2 2025-01-03      17.8    Ici 0,1,2 sont des index ...
```

On veut représenter une série temporelle, la date doit donc être l'axe principale.

```
date      Température
2025-01-01      18.2
2025-01-02      19.1
2025-01-03      17.8
```

```
df = df.set_index("date")
```

Structures de données avancées

Quand la date est l'index, Pandas/Matplotlib **reconnaissent automatiquement** que c'est une série temporelle :

```
df[« temperature"].plot()
```

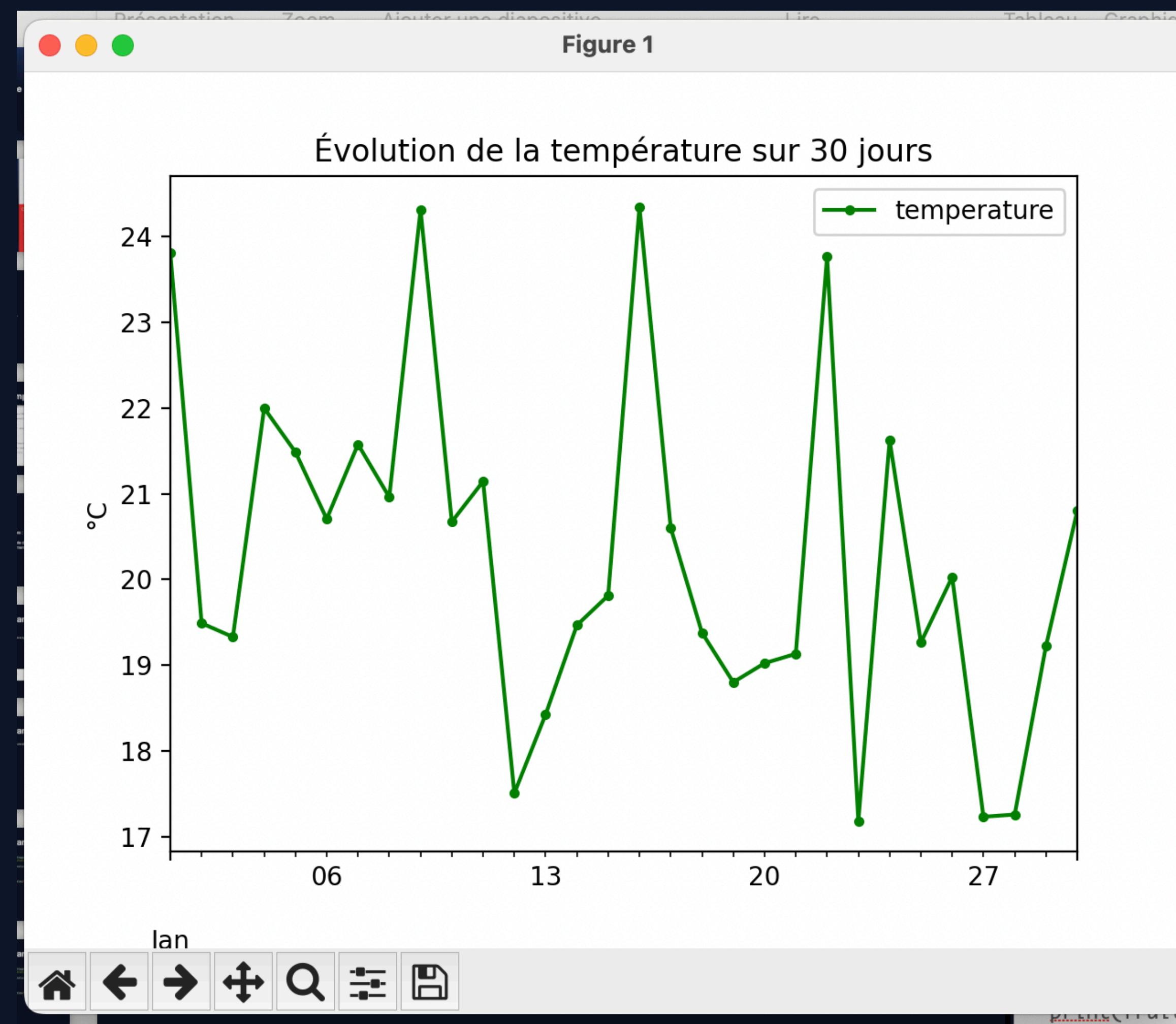
trace directement une courbe avec des dates sur l'axe X ⓘ

Structures de données avancées

```
# Création de données temporelles simulées
dates = pd.date_range(start="2025-01-01", periods=30, freq="D")
temperature = np.random.normal(20, 2, size=30) # température autour de
20°C
df = pd.DataFrame({"date": dates, "temperature": temperature})
df = df.set_index("date")
print(df.head())

# Tracé
df.plot(y="temperature", color="green", marker=".",
         title="Température
quotidienne")
plt.title("Évolution de la température sur 30 jours")
plt.xlabel("Date")
plt.ylabel("°C")
plt.show()
```

Structures de données avancées



Structures de données avancées

Utilisation d'un index temporel pour zoomer

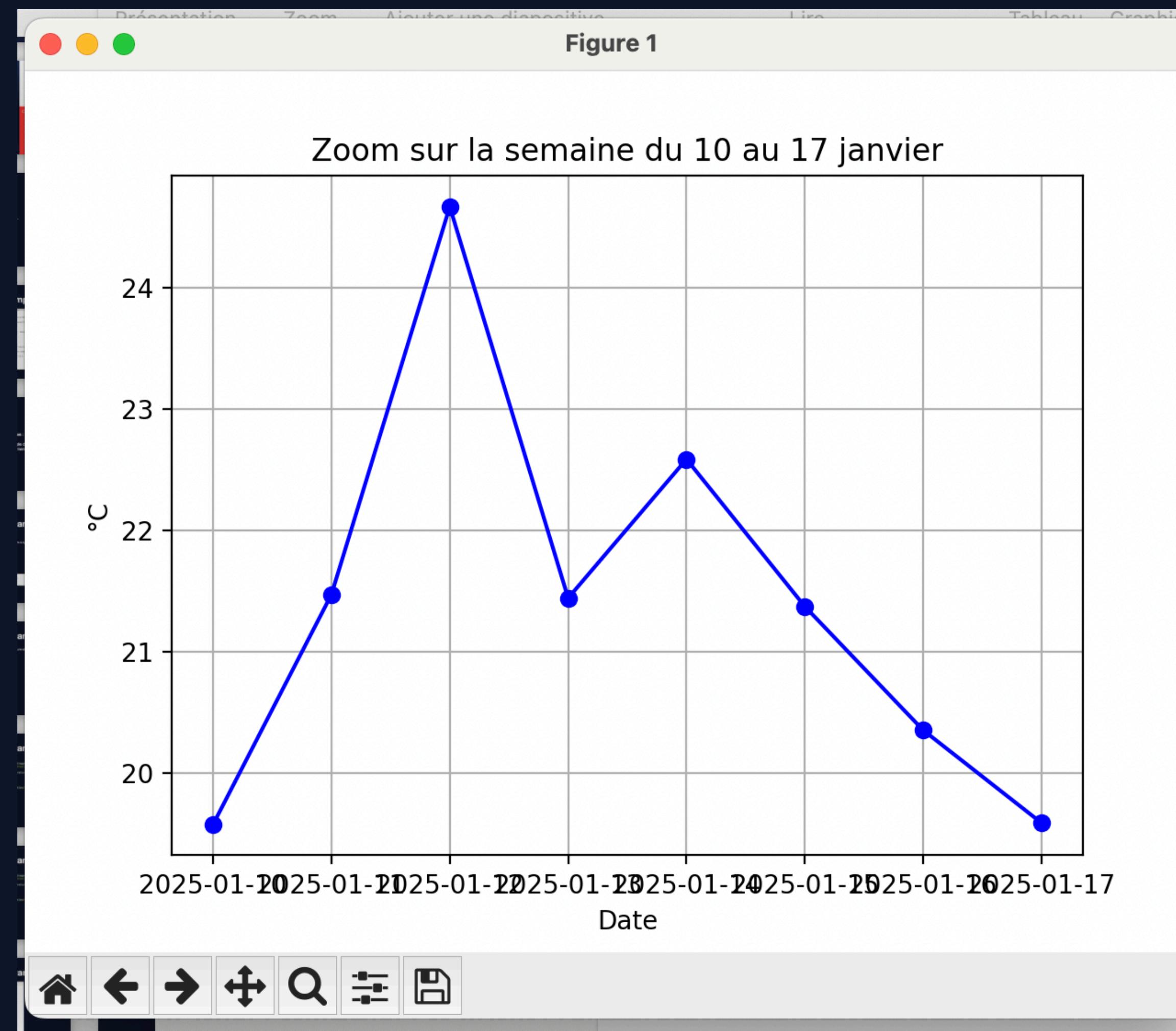
Sélectionner une **période précise** dans la série (ex : une semaine).

Grâce à l'index temporel, on peut filtrer avec des chaînes de dates : "2025-01-10":"2025-01-17".

```
# Création de données temporelles simulées
dates = pd.date_range(start="2025-01-01", periods=30, freq="D")
temperature = np.random.normal(20, 2, size=30) # température autour de 20°C
df = pd.DataFrame({"date": dates, "temperature": temperature})
df = df.set_index("date")
print(df.head())

# Tracé
zoom = df.loc["2025-01-10":"2025-01-17"]
plt.plot(zoom.index, zoom["temperature"], color="blue", marker="o")
plt.title("Zoom sur la semaine du 10 au 17 janvier")
plt.xlabel("Date")
plt.ylabel("°C")
plt.grid(True)
plt.show()
```

Structures de données avancées



Structures de données avancées

Tracer des séries temporelles avec différentes variables

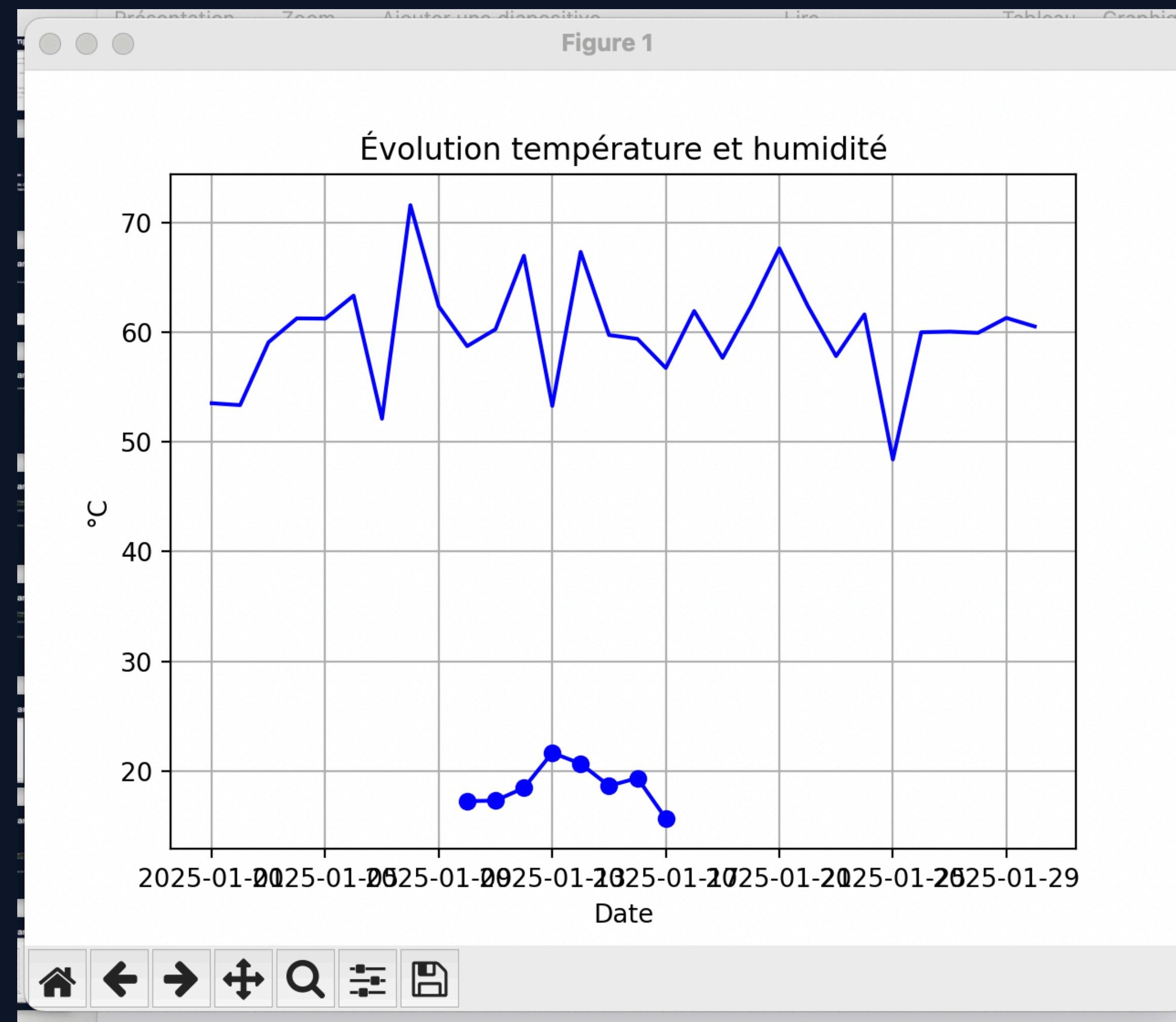
Comparer plusieurs mesures sur la même période (ex : température & humidité).

On ajoute une nouvelle variable au DataFrame

```
# Création de données temporelles simulées
dates = pd.date_range(start="2025-01-01", periods=30, freq="D")
temperature = np.random.normal(20, 2, size=30) # température autour de 20°C
humidite = np.random.normal(60, 5, size=30)
df = pd.DataFrame({"date": dates, "temperature": temperature, "humidite": humidite})
df = df.set_index("date")
print(df.head())

# Tracé
zoom = df.loc["2025-01-10":"2025-01-17"]
plt.plot(zoom.index, zoom["temperature"], color="blue", marker="o")
plt.plot(df.index, df["humidite"], label="Humidité (%)", color="blue")
plt.title("Évolution température et humidité")
plt.xlabel("Date")
plt.ylabel("°C")
plt.grid(True)
plt.show()
```

Structures de données avancées



Structures de données avancées

Tracé de deux variables avec axes différents

Afficher deux échelles différentes sur le même graphique (ex : température °C et humidité %).

Nous voulons ici deux sous graphiques -> subplot()

```
# Création de données temporelles simulées
dates = pd.date_range(start="2025-01-01", periods=30, freq="D")
temperature = np.random.normal(20, 2, size=30) # température autour de 20°C
humidite = np.random.normal(60, 5, size=30)
df = pd.DataFrame({"date": dates, "temperature": temperature, "humidite": humidite})
df = df.set_index("date")
print(df.head())

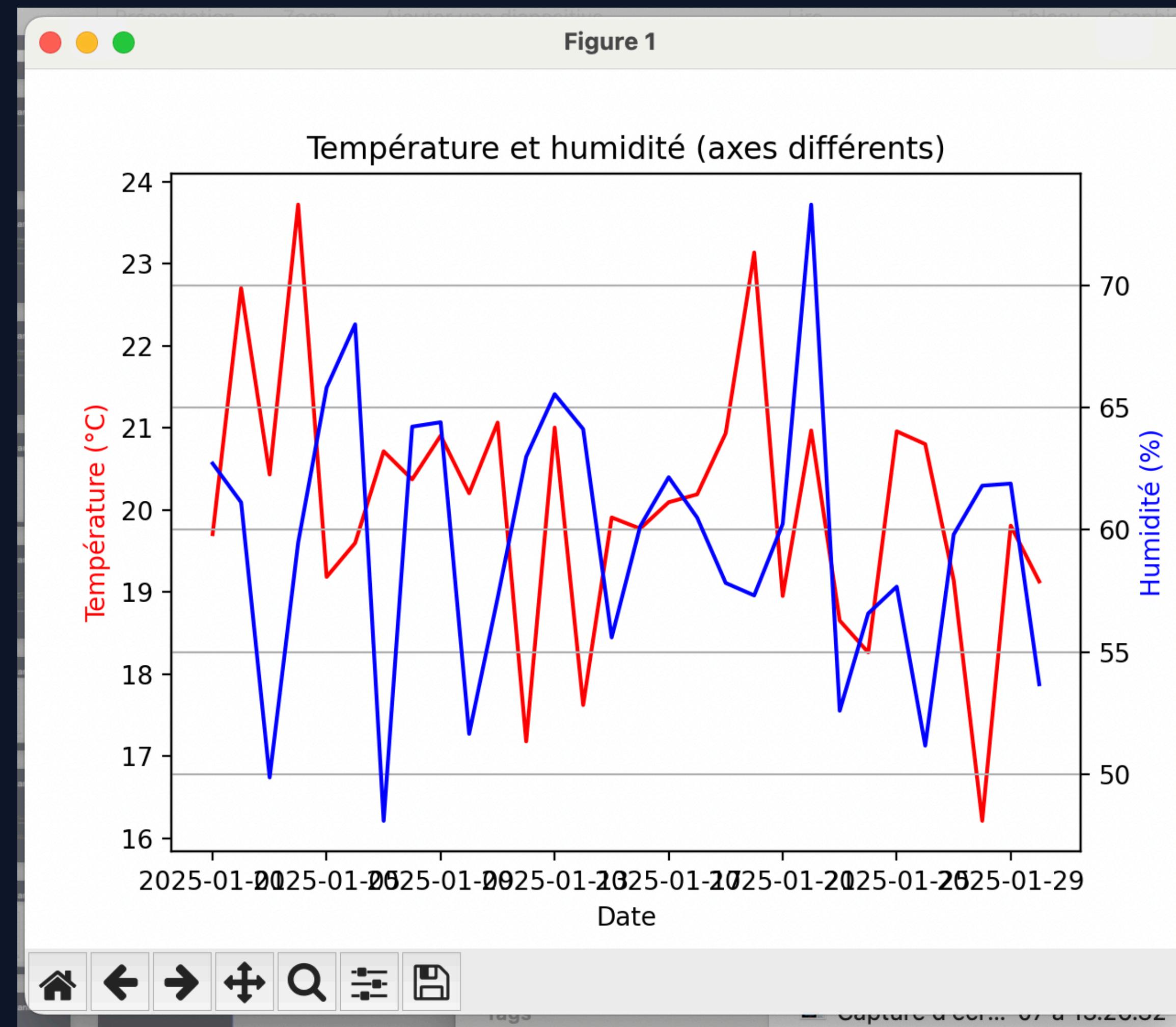
# Tracé
fig, ax1 = plt.subplots()

ax2 = ax1.twinx() # second axe Y

ax1.plot(df.index, df["temperature"], color="red", label="Température (°C)")
ax2.plot(df.index, df["humidite"], color="blue", label="Humidité (%)")

ax1.set_xlabel("Date")
ax1.set_ylabel("Température (°C)", color="red")
ax2.set_ylabel("Humidité (%)", color="blue")
plt.title("Température et humidité (axes différents)")
plt.grid(True)
plt.show()
```

Structures de données avancées



Structures de données avancées

Créer une fonction de tracé réutilisable

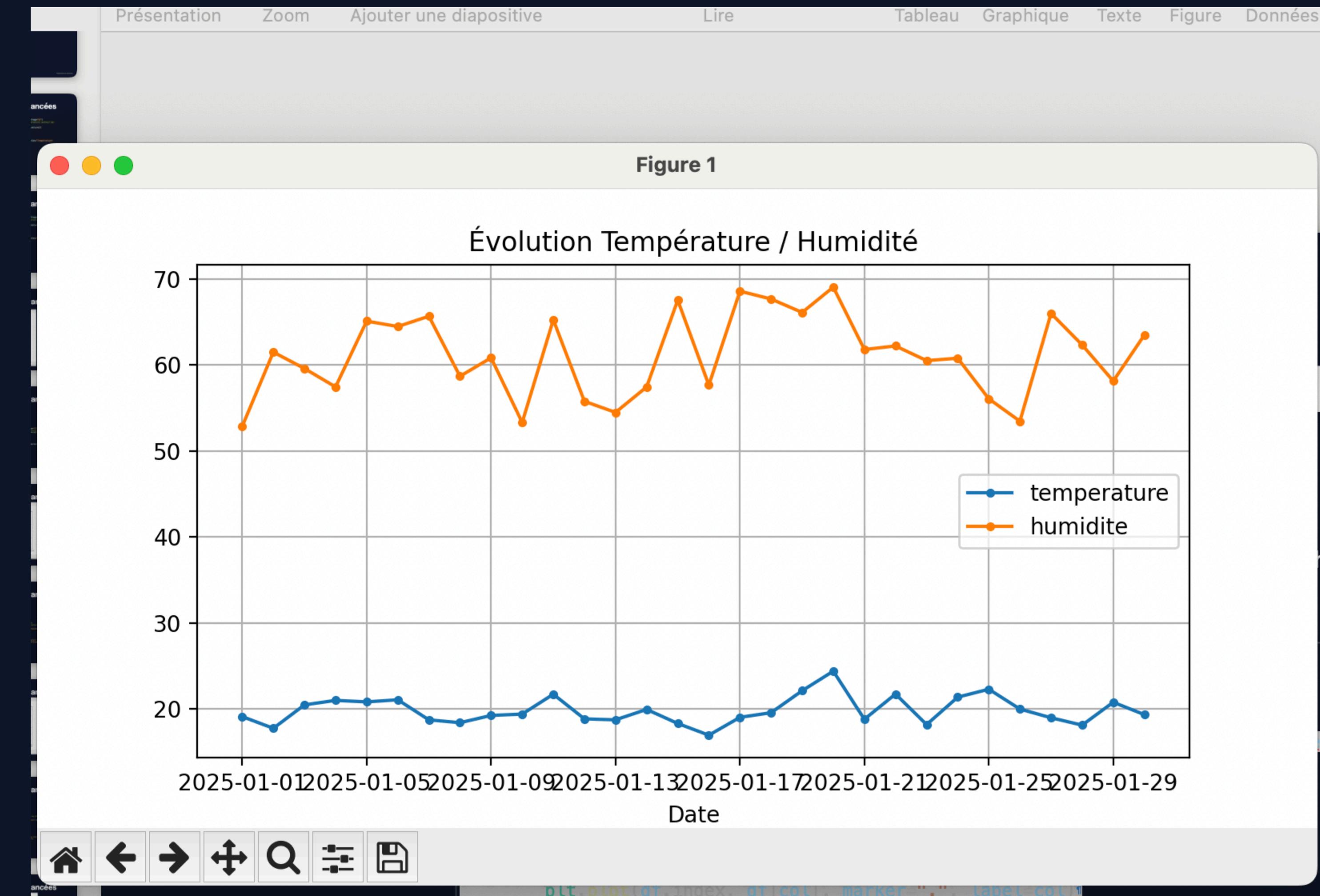
Il est possible de créer une fonction permettant de configurer et d'afficher un plot. Utile quand on doit afficher plusieurs fois la série.

```
# Création de données temporelles simulées
dates = pd.date_range(start="2025-01-01", periods=30, freq="D")
temperature = np.random.normal(20, 2, size=30) # température autour de 20°C
humidite = np.random.normal(60, 5, size=30)
df = pd.DataFrame({"date": dates, "temperature": temperature, "humidite": humidite})
df = df.set_index("date")
print(df.head())

def tracer_serie(df, colonnes, titre="Série temporelle"):
    plt.figure(figsize=(8,4))
    for col in colonnes:
        plt.plot(df.index, df[col], marker=".", label=col)
    plt.title(titre)
    plt.xlabel("Date")
    plt.legend()
    plt.grid(True)
    plt.show()

# Utilisation
tracer_serie(df, ["temperature", "humidite"], "Évolution Température / Humidité")
```

Structures de données avancées



Structures de données avancées

Utilisation de la fonction de tracage

Il est possible de réutiliser la fonction sur n'importe quel DataFrame ayant des données temporelles :

```
# Création de données temporelles simulées
dates = pd.date_range(start="2025-01-01", periods=30, freq="D")
temperature = np.random.normal(20, 2, size=30) # température autour de 20°C
humidite = np.random.normal(60, 5, size=30)
df = pd.DataFrame({"date": dates, "temperature": temperature, "humidite": humidite})
df = df.set_index("date")
print(df.head())

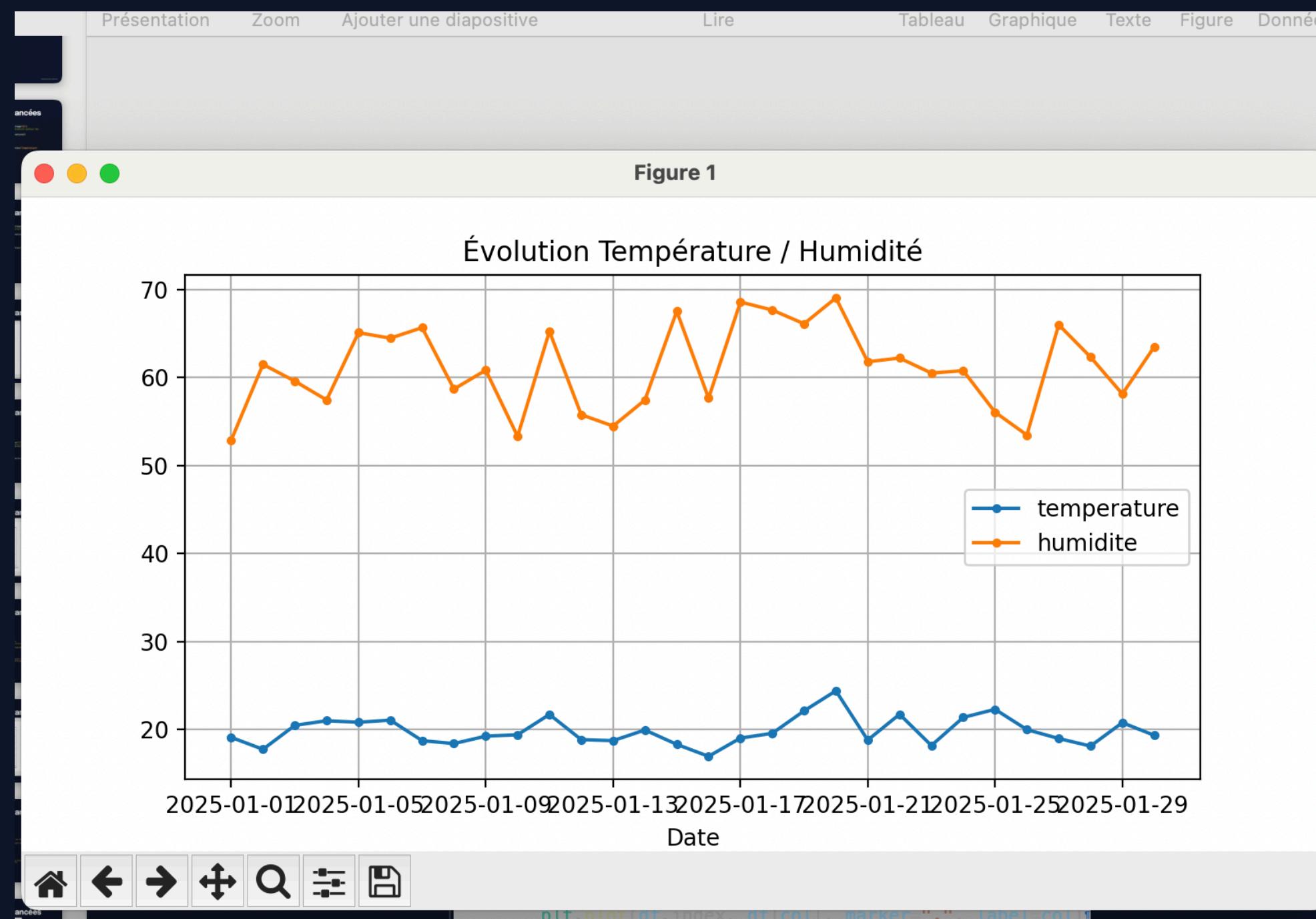
def tracer_serie(df, colonnes, titre="Série temporelle"):
    plt.figure(figsize=(8,4))
    for col in colonnes:
        plt.plot(df.index, df[col], marker=".", label=col)
    plt.title(titre)
    plt.xlabel("Date")
    plt.legend()
    plt.grid(True)
    plt.show()

# Utilisation
tracer_serie(df, ["temperature", "humidite"], "Évolution Température / Humidité")

# Exemple avec d'autres données simulées
df2 = pd.DataFrame({
    "date": pd.date_range("2025-02-01", periods=15, freq="D"),
    "vent": np.random.normal(15, 3, size=15),
    "pluie": np.random.normal(5, 1, size=15)
}).set_index("date")

tracer_serie(df2, ["vent", "pluie"], "Évolution du vent et de la pluie")
```

Structures de données avancées



Structures de données avancées

Annoter les données de séries temporelles

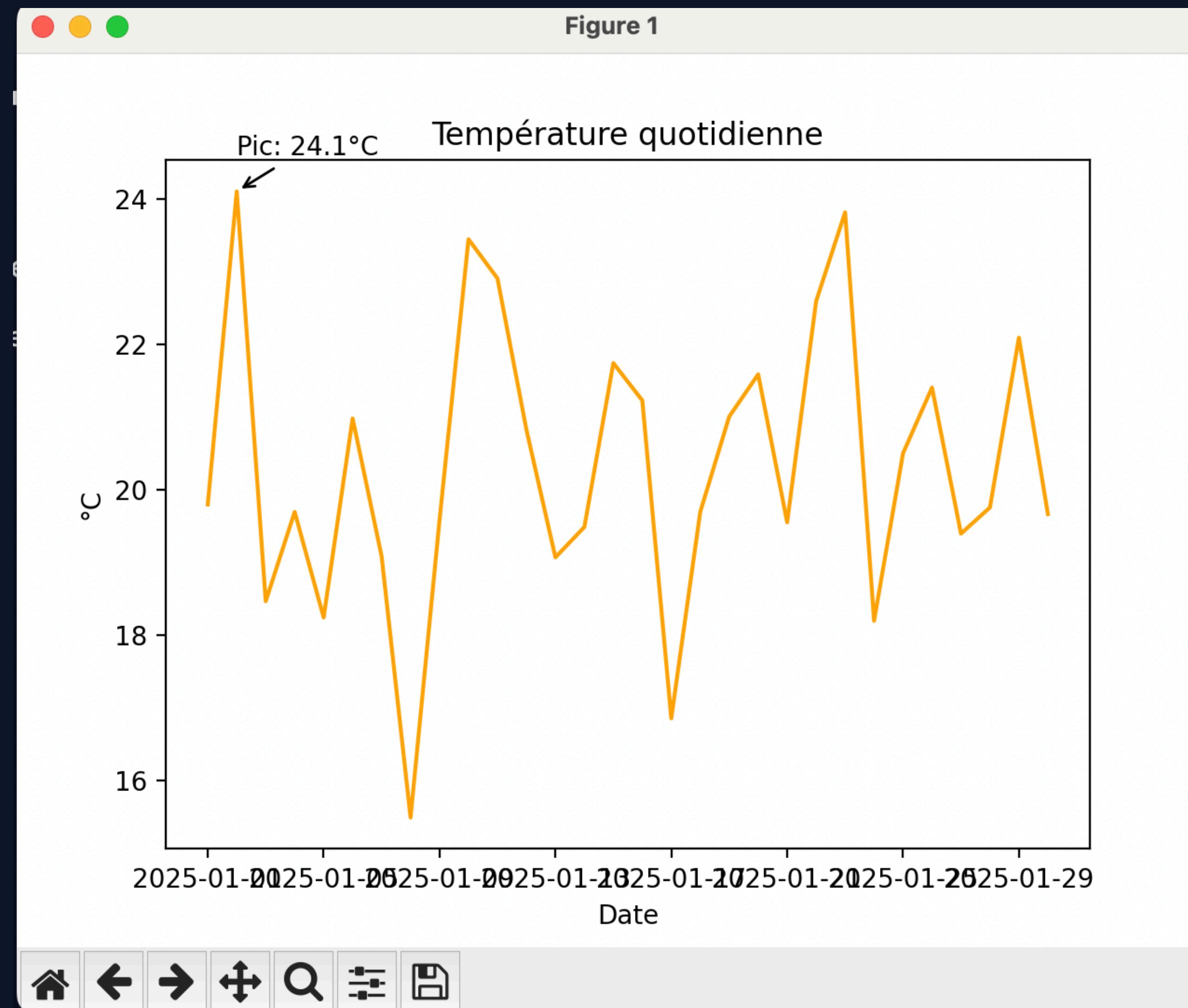
Mettre en évidence un événement ou une valeur clé (ex : pic, minimum...).

```
# Création de données temporelles simulées
dates = pd.date_range(start="2025-01-01", periods=30, freq="D")
temperature = np.random.normal(20, 2, size=30) # température autour de 20°C
humidite = np.random.normal(60, 5, size=30)
df = pd.DataFrame({"date": dates, "temperature": temperature, "humidite": humidite})
df = df.set_index("date")
print(df.head())
max_temp = df["temperature"].max()
max_date = df["temperature"].idxmax()

plt.plot(df.index, df["temperature"], color="orange")
plt.title("Température quotidienne")
plt.xlabel("Date")
plt.ylabel("°C")

# Annotation du pic
plt.annotate(
    f"Pic: {max_temp:.1f}°C",
    xy=(max_date, max_temp),
    xytext=(max_date, max_temp+0.5),
    arrowprops=dict(facecolor='black', arrowstyle='->')
)
plt.show()
```

Structures de données avancées



Structures de données avancées

Tracer des séries temporelles : tout assembler.

```
# Création d'un jeu de données complet
dates = pd.date_range("2025-03-01", periods=60, freq="D")
df = pd.DataFrame({
    "temperature": np.random.normal(18, 3, size=60),
    "humidite": np.random.normal(55, 8, size=60),
    "vent": np.random.normal(15, 2, size=60)
}, index=dates)

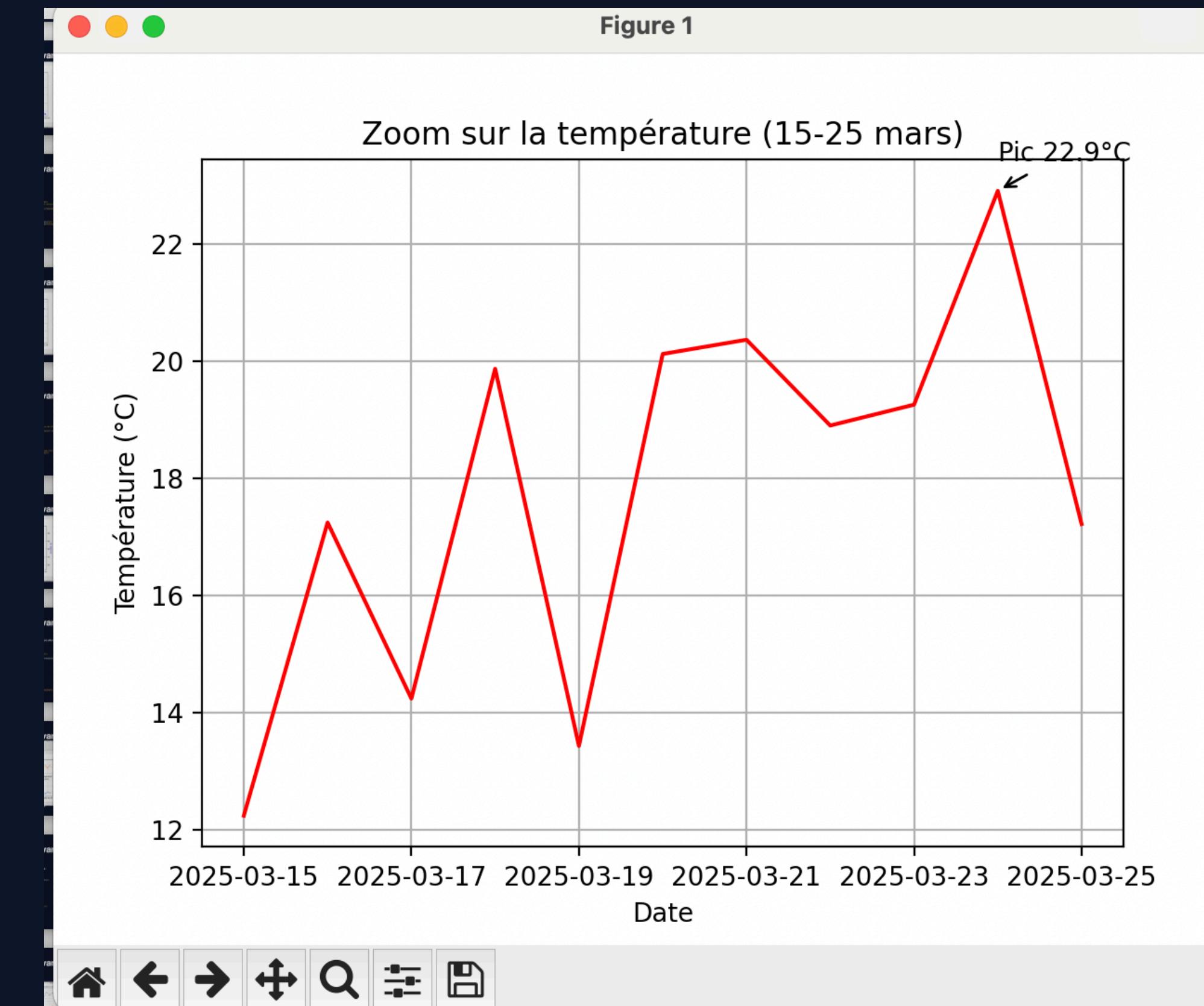
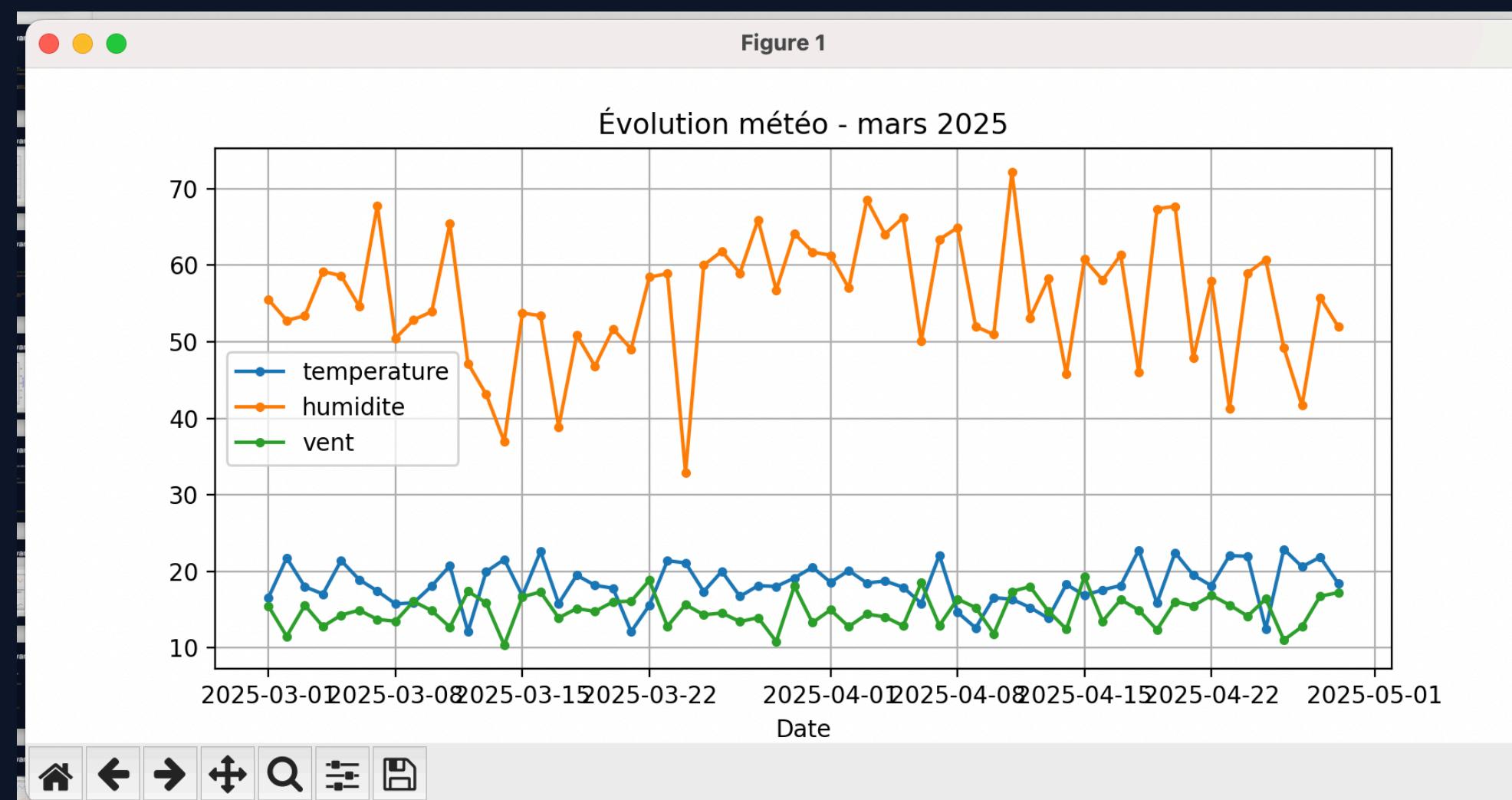
# Fonction de traçage réutilisable
def plot_series(df, cols, titre):
    plt.figure(figsize=(9,4))
    for col in cols:
        plt.plot(df.index, df[col], marker=".", label=col)
    plt.title(titre)
    plt.xlabel("Date")
    plt.legend()
    plt.grid(True)
    plt.show()

# Tracé global
plot_series(df, ["temperature", "humidite", "vent"], "Évolution météo - mars 2025")

# Zoom + annotation
zoom = df.loc["2025-03-15":"2025-03-25"]
max_temp = zoom["temperature"].max()
max_date = zoom["temperature"].idxmax()

plt.plot(zoom.index, zoom["temperature"], color="red")
plt.title("Zoom sur la température (15-25 mars)")
plt.xlabel("Date")
plt.ylabel("Température (°C)")
plt.annotate(f"Pic {max_temp:.1f}°C", xy=(max_date, max_temp),
            xytext=(max_date, max_temp+0.5),
            arrowprops=dict(arrowstyle="->", color="black"))
plt.grid(True)
plt.show()
```

Structures de données avancées



Structures de données avancées

Les différents types de courbes

Tableau 1

 Type de tracé	 Commande	 Description	 Quand l'utiliser	Exemple
Courbe (Line Plot)	`plt.plot()`	Trace une courbe reliant les points dans le temps	Pour suivre une tendance continue	plt.plot(df.index, df["temp"])
Barres (Bar Plot)	`plt.bar()`	Affiche des valeurs discrètes dans le temps	Comparer des quantités sur une période	plot.bar(df.index, df["pluie"])
Nuage de points (Scatter)	`plt.scatter()`	Montre chaque point indépendamment	Detecter des anomalies	plot.scatter(df.index, df["humid"])
Aire (Area Plot)	`plt.fill_between()`	Remplit la zone sous la courbe	Visualiser une quantité cumulée	plot.fill_between(df.index, df["prod"])
Aires empilées Stackplot	`plt.stackplot()`	Empile plusieurs séries dans le temps	Montrer la composition totale	plot.stackplot(df.index, df["vent"], df["soleil"])
Boîtes (Box Plot)	`plt.boxplot()`	Montre la distribution des valeurs (min, max)	Comparer les variations	df.boxplot(column="temp", by='month')
Histogramme (Hist)	`plt.hist()`	Affiche la répartition des valeurs	Etudier la fréquence distribuées	plot.hist(df["temp"], bins=20)
Matrice des chaleur (Heatmap)	`sns.heatmap()`	Montre les patterns temporels sous formes de courbes	Identifier les cycles journaliers	sns.heatmap(pivot, cmap="coolwarm")
Combinaison	`plt.bar()` + `plt.plot()`	Superpose plusieurs types sur le même graphique	Comparer deux indicateurs	plt.bar(df.index, df["pluie"]); plt.plot(df.index,

Structures de données avancées

Exercices:

Vous disposez d'un fichier CSV nommé meteo.csv contenant les données météorologiques quotidiennes pour le mois de janvier 2025.

Après avoir chargés et préparés les données, tracer la température, les précipitations et la vitesse du vent quotidienne sous forme de courbe (3 courbes). Ce traçage doit se faire grâce à une fonction de traçage qui doit permettre de tracer n'importe quelle variable avec un type de graphique choisi (line, bar, scatter).

Calculer la moyenne hebdomadaire de la température et la somme hebdomadaire de la pluie. Tracer ces données en choisissant le type de graphique le plus adapté. Identifier d'éventuelles anomalies ou pics dans les données

Introduction au Machine Learning

Partie 1- 45 -120 minutes



- Identifier les tendances, relations et pattern
- Déetecter les anomalies
- Analyser des résultats

Introduction au Machine Learning

Le Machine Learning (ML) est une branche de l'intelligence artificielle qui permet à un **ordinateur d'apprendre à partir de données** pour effectuer des tâches comme :

- Prédire une valeur (ex : température demain)
- Classer des objets (ex : spam ou non spam)
- Regrouper des objets similaires (ex : segmentation de clients)

Contrairement à un programme classique, où tout est codé manuellement, le ML **apprend à partir d'exemples (données)**.

Introduction au Machine Learning

Types de Machine Learning:

Apprentissage supervisé (Supervised Learning)

- **Principe** : le modèle apprend à partir de **données étiquetées**, c'est-à-dire où la “réponse correcte” est connue.
- **Objectif** : prédire une sortie à partir d'entrées.
- **Types de problèmes** :
 - **Régression** → prédire une valeur continue
 - Exemple : prédire le prix d'une maison à partir de sa surface, nombre de chambres...
 - Algorithmes : Régression linéaire, Random Forest, XGBoost
 - **Classification** → prédire une catégorie
 - Exemple : email = spam ou non spam
 - Algorithmes : Logistic Regression, SVM, KNN, Random Forest

Exemple concret :

Données météo → Prédire s'il va pleuvoir demain (Oui/Non)

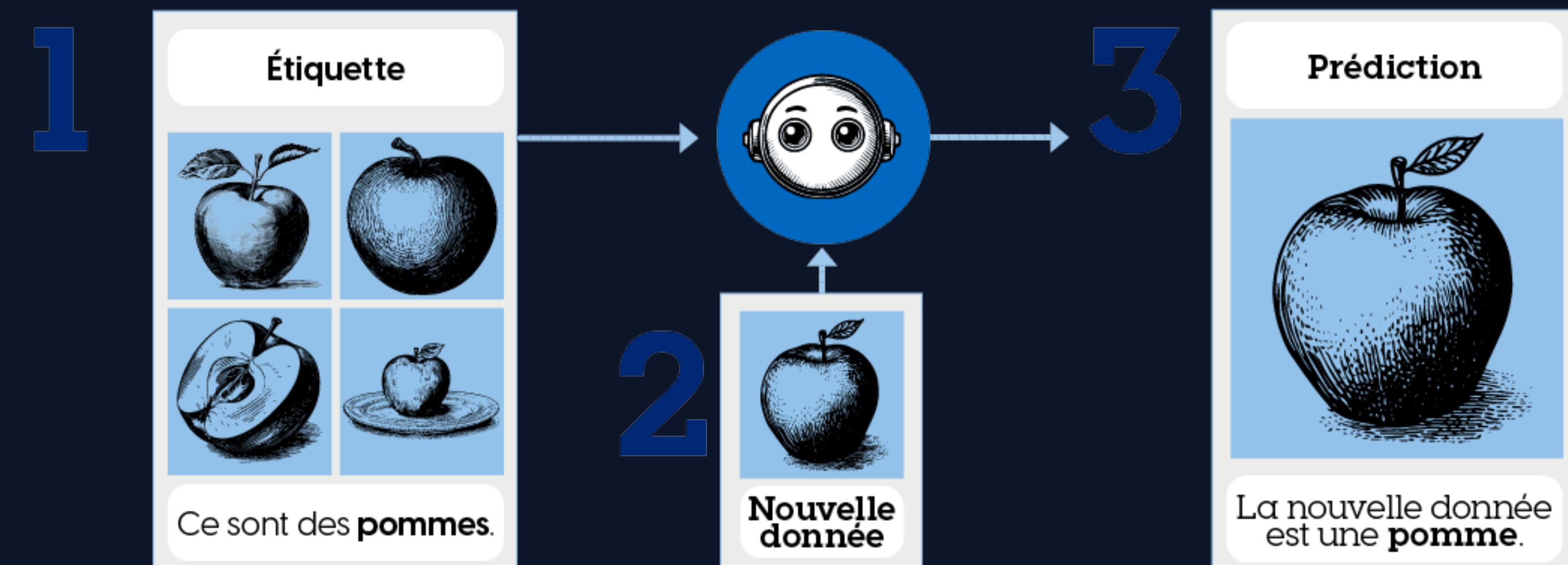
Features : température, humidité, vent

Target : pluie (Oui/Non)

Introduction au Machine Learning

Types de Machine Learning:

Apprentissage supervisé (Supervised Learning)



Introduction au Machine Learning

Types de Machine Learning:

Apprentissage non supervisé (UnSupervised Learning)

- **Principe** : le modèle apprend à partir de **données non étiquetées**, sans réponse connue.
- **Objectif** : trouver des **motifs, regroupements ou anomalies** dans les données.
- **Types de problèmes** :
 - **Clustering (regroupement)** → regrouper les objets similaires
 - Exemple : segmentation de clients selon leur comportement d'achat
 - Algorithmes : K-Means, DBSCAN, Hierarchical Clustering
 - **Réduction de dimensionnalité** → simplifier les données tout en conservant l'information
 - Exemple : PCA (Principal Component Analysis) pour visualiser des données 3D en 2D
 - **Détection d'anomalies** → trouver les valeurs inhabituelles
 - Exemple : transactions bancaires frauduleuses

Exemple concret :

Données météo → Regrouper les jours similaires selon température et précipitations

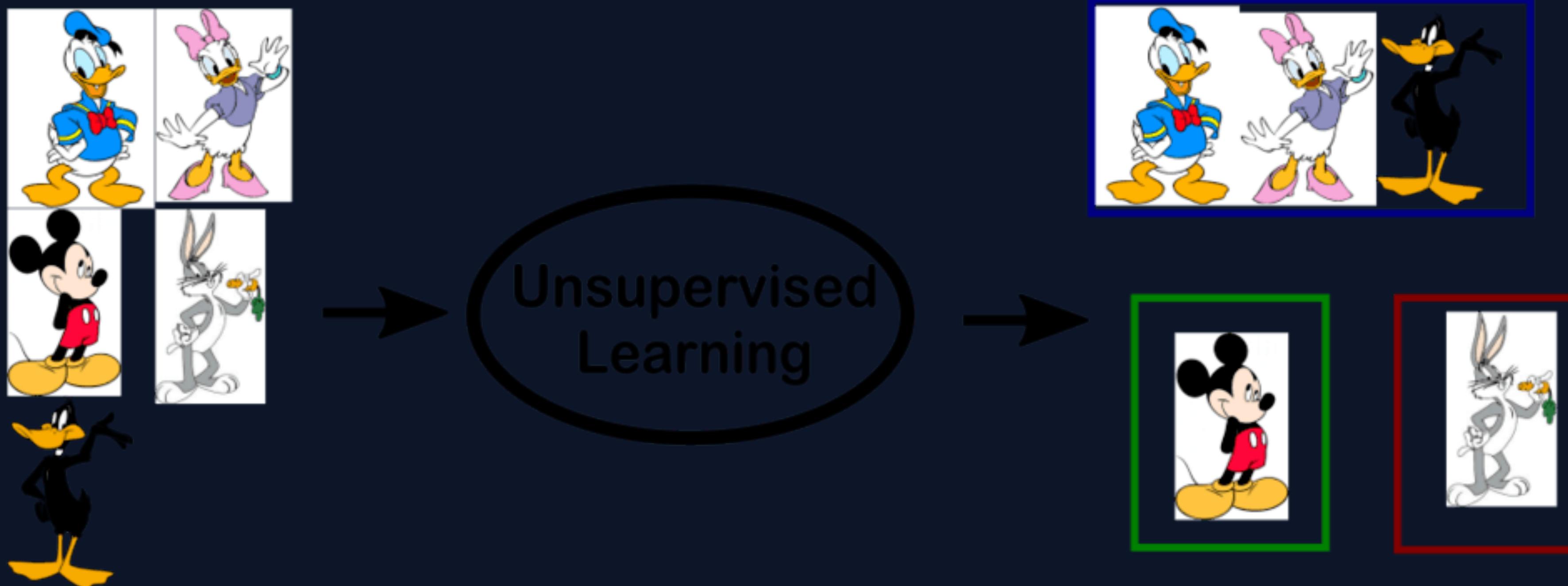
Features : température, pluie, vent

Target : aucune

Introduction au Machine Learning

Types de Machine Learning:

Apprentissage non supervisé (UnSupervised Learning)



Introduction au Machine Learning

Types de Machine Learning:

Apprentissage par renforcement (Reinforcement Learning)

- **Principe** : le modèle apprend par **essais/erreurs**, en recevant des **récompenses ou pénalités** selon ses actions.
- **Objectif** : apprendre une **stratégie optimale** pour maximiser la récompense sur le long terme.
- **Éléments clés** :
 - **Agent** → prend des décisions
 - **Environnement** → monde dans lequel agit l'agent
 - **Récompense** → feedback sur la qualité de l'action

Exemples :

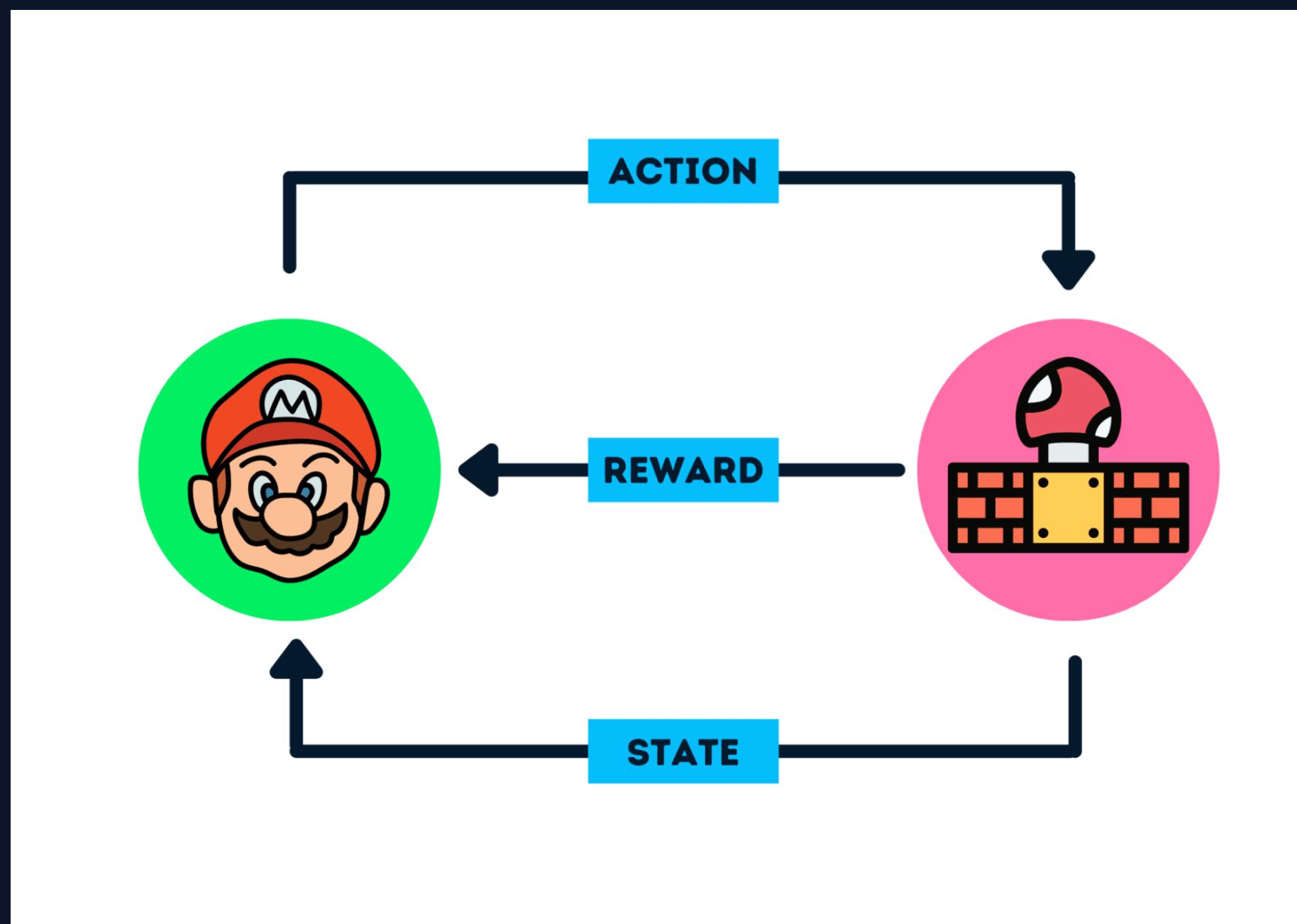
- Jeu vidéo → l'agent apprend à gagner
- Robot → apprendre à marcher ou attraper un objet
- Gestion de stock → maximiser les profits en choisissant quand commander

Algorithmes courants : Q-Learning, Deep Q-Network (DQN), Policy Gradient

Introduction au Machine Learning

Types de Machine Learning:

Apprentissage non supervisé (UnSupervised Learning)



Introduction au Machine Learning

Exercices:

Une entreprise de e-commerce souhaite **prévoir le chiffre d'affaires de demain** à partir de ses données historiques : nombre de visites, produits vendus, promotions en cours, météo, etc.

Question : Quel type de Machine Learning utiliser ?

- A) Supervisé
- B) Non supervisé
- C) Renforcement

Indice : On connaît la valeur qu'on veut prédire (chiffre d'affaires).

Introduction au Machine Learning

Exercices:

Une boutique en ligne veut **regrouper ses clients selon leur comportement d'achat** afin de leur proposer des promotions personnalisées.

- On a seulement des données sur les achats passés, sans “étiquette” indiquant le type de client.

Question : Quel type de ML utiliser ?

- A) Supervisé
- B) Non supervisé
- C) Renforcement

Indice : Il n'y a pas de label à prédire.

Introduction au Machine Learning

Exercices:

Un robot virtuel doit **apprendre à se déplacer dans un labyrinthe** pour atteindre la sortie.

- À chaque mouvement, il reçoit une récompense positive s'il se rapproche de la sortie, et une pénalité s'il se cogne contre un mur.

Question : Quel type de ML utiliser ?

- A) Supervisé
- B) Non supervisé
- C) Renforcement

Introduction au Machine Learning

Exercices:

Une banque souhaite **identifier les transactions frauduleuses** à partir de l'historique de toutes les transactions : montant, localisation, type de paiement, heure de la transaction, etc.

- Certaines transactions sont annotées comme frauduleuses, mais beaucoup ne le sont pas.

Question : Quel type de ML utiliser ?

- A) Supervisé
- B) Non supervisé
- C) Renforcement

Introduction au Machine Learning

Exercices:

Un service météo veut **prévoir s'il va pleuvoir demain** en se basant sur les données historiques : température, humidité, vent, pression, pluviométrie, etc.

Question : Quel type de ML utiliser ?

- A) Supervisé
- B) Non supervisé
- C) Renforcement

Introduction au Machine Learning

Résumé:

Tableau 1

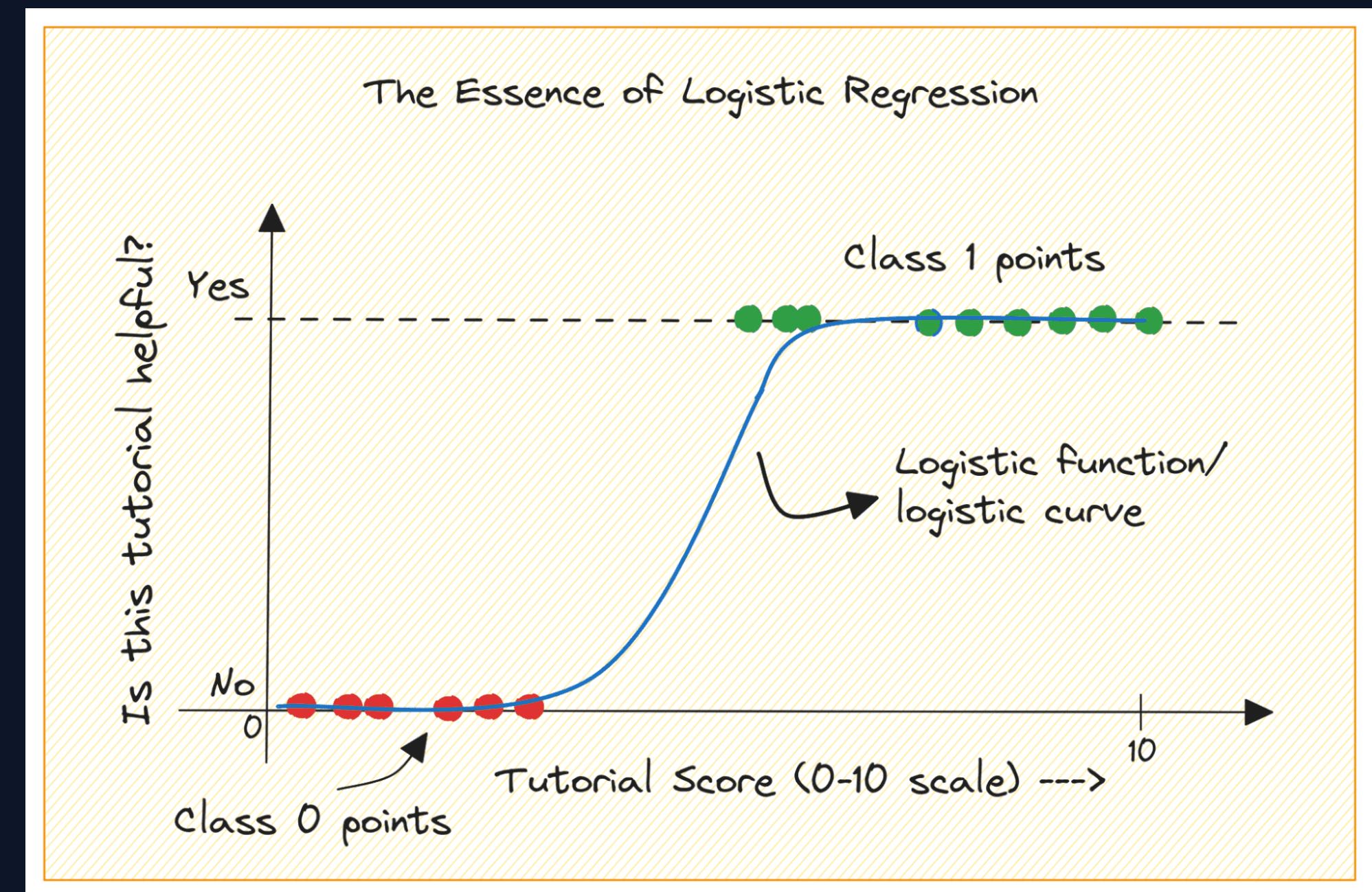
Type de ML	Données	Objectif	Exemple	Algorithmes
Supervisé	Étiquetées	Prédire une sortie	Prédire pluie / prix maison	Régression, Random Forest, SVM
Non supervisé	Non étiquetées	Trouver structure ou anomalies	Segmentation clients, clustering météo	K-Means, PCA, DBSCAN
Renforcement	Feedback / récompense	Apprendre stratégie optimale	Jeu vidéo, robot, trading	Q-Learning, DQN, Policy Gradient

Introduction au Machine Learning

Les modèles de classification simple supervisé:

Logistic Regression (Régression logistique)

- Type : **linéaire**
- Principe : modélise la **probabilité d'appartenir à une classe**
- Sortie : entre 0 et 1, avec un seuil pour décider de la classe
- Avantages : rapide, facile à interpréter
- Limites : ne capture pas bien les relations non linéaires
- Exemple : spam / non spam, pluie / pas pluie

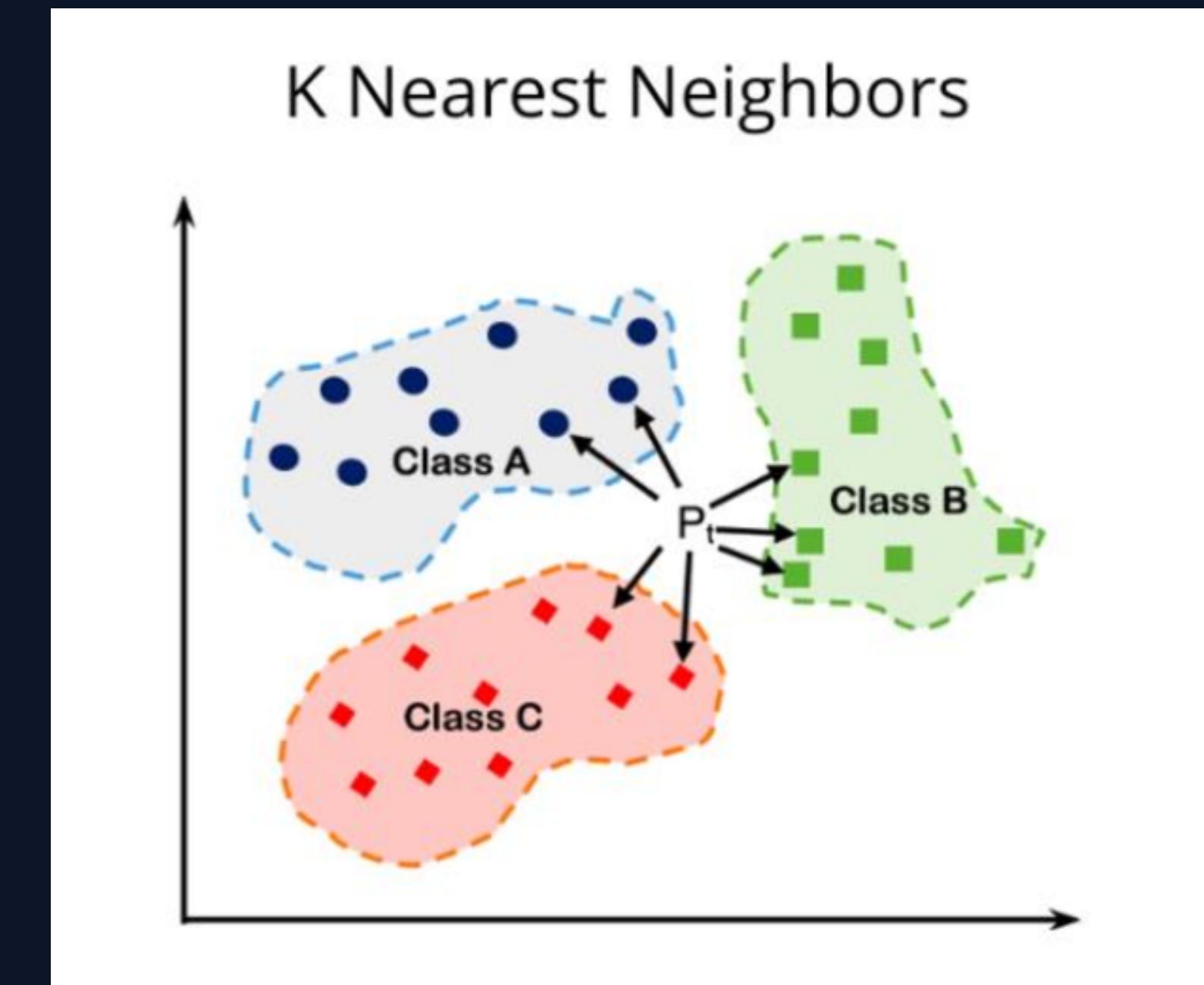


Introduction au Machine Learning

Les modèles de classification simple supervisé:

2. K-Nearest Neighbors (KNN)

- Type : **non paramétrique, instance-based**
- Principe : classe d'un point = classe majoritaire parmi ses **k voisins les plus proches**
- Avantages : simple, intuitif, fonctionne pour petites données
- Limites : lent sur gros datasets, sensible à l'échelle des features
- Exemple : reconnaissance de chiffres, classification de fleurs

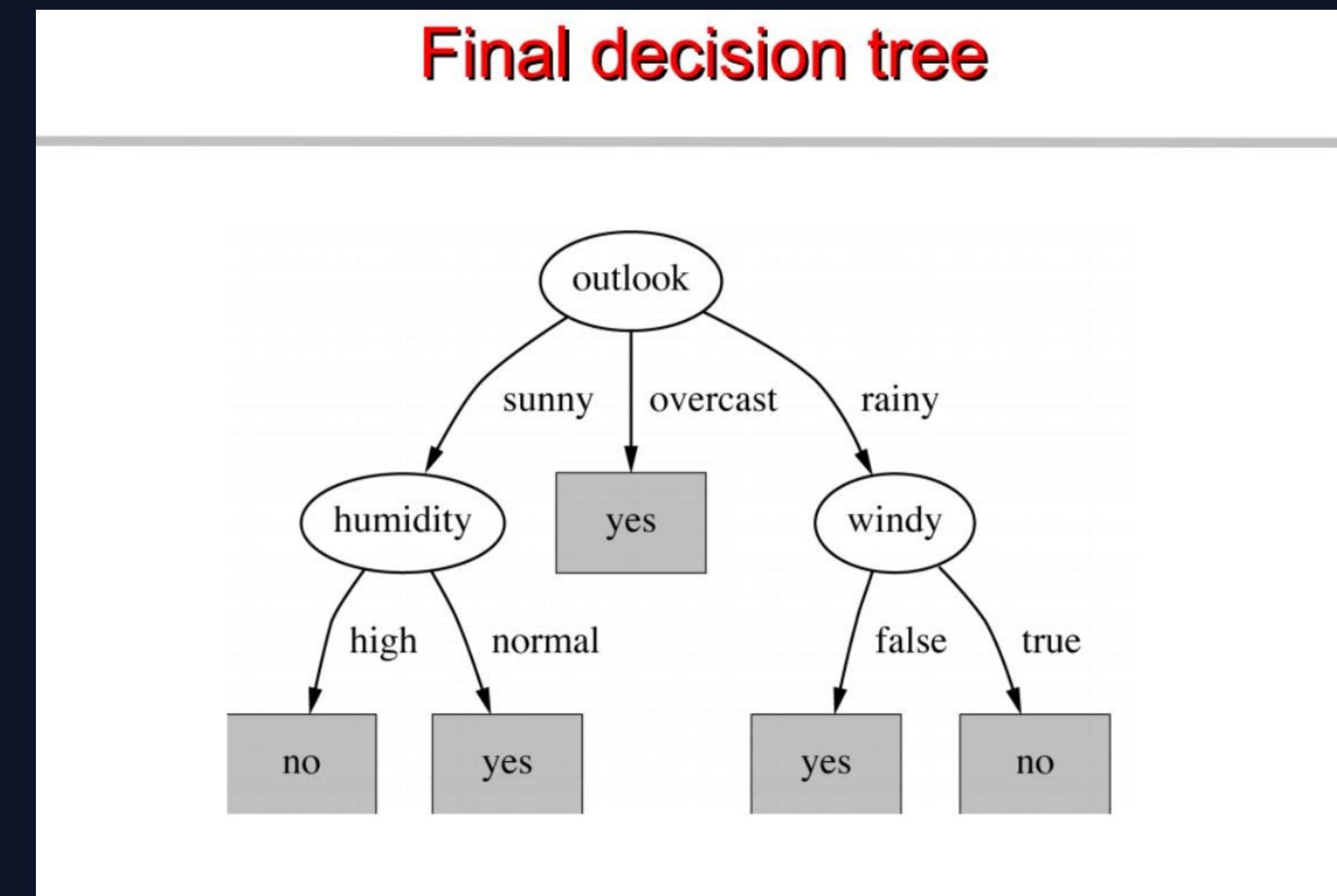


Introduction au Machine Learning

Les modèles de classification simple supervisé:

Decision Tree (Arbre de décision)

- Type : **arbre hiérarchique**
- Principe : divise les données en fonction de **questions simples** sur les features
- Avantages : facile à visualiser, interprétable
- Limites : tendance à surapprendre (overfitting),
sensible aux petits changements dans les données
- Exemple : prédition de maladies, classification météo

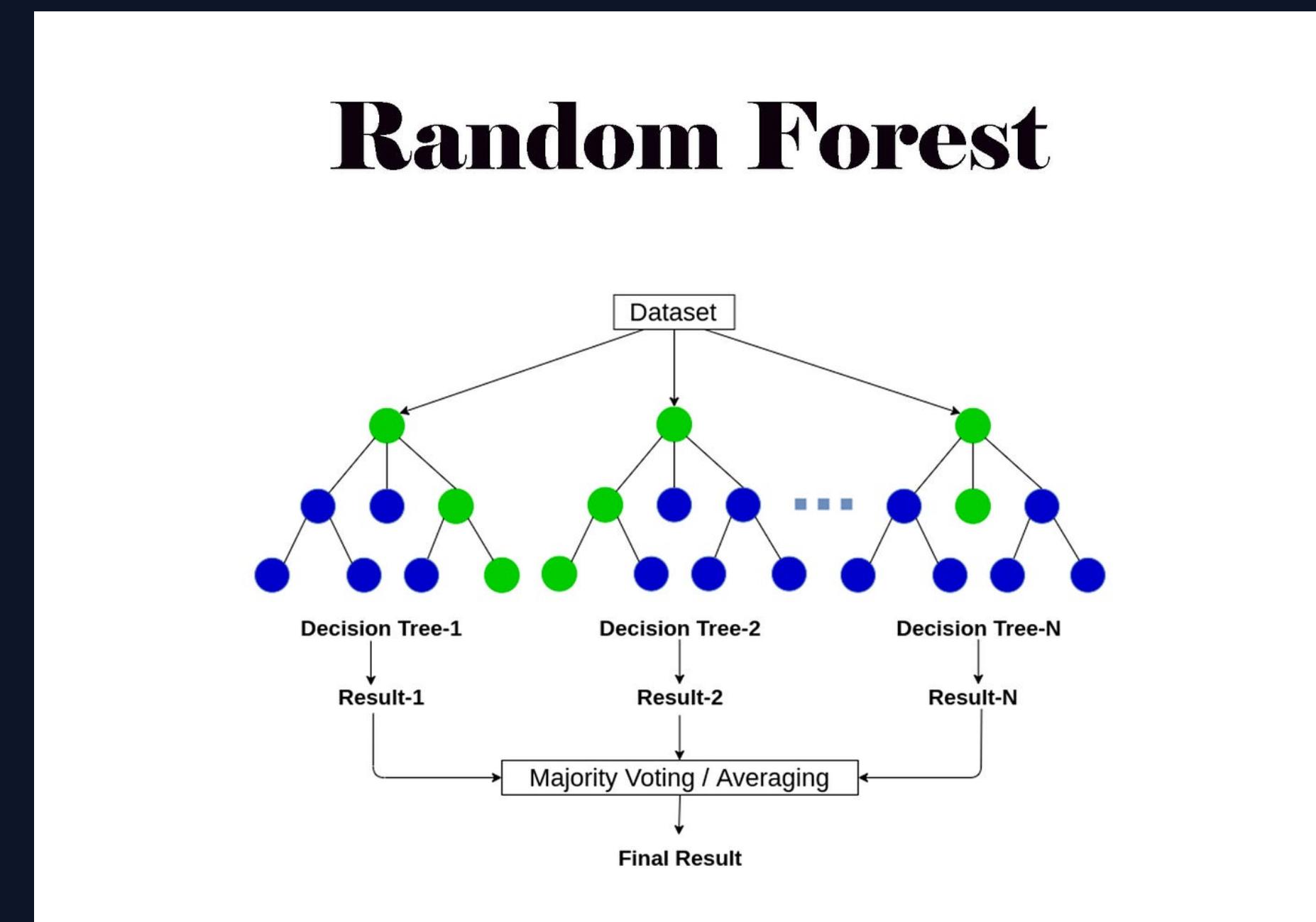


Introduction au Machine Learning

Les modèles de classification simple supervisé:

Random Forest

- Type : **ensemble de Decision Trees**
- Principe : crée plusieurs arbres sur des sous-échantillons et fait un **vote majoritaire**
- Avantages : réduit le surapprentissage, robuste
- Limites : moins interprétable qu'un arbre unique, plus lourd
- Exemple : prédiction de fraude, classification de clients

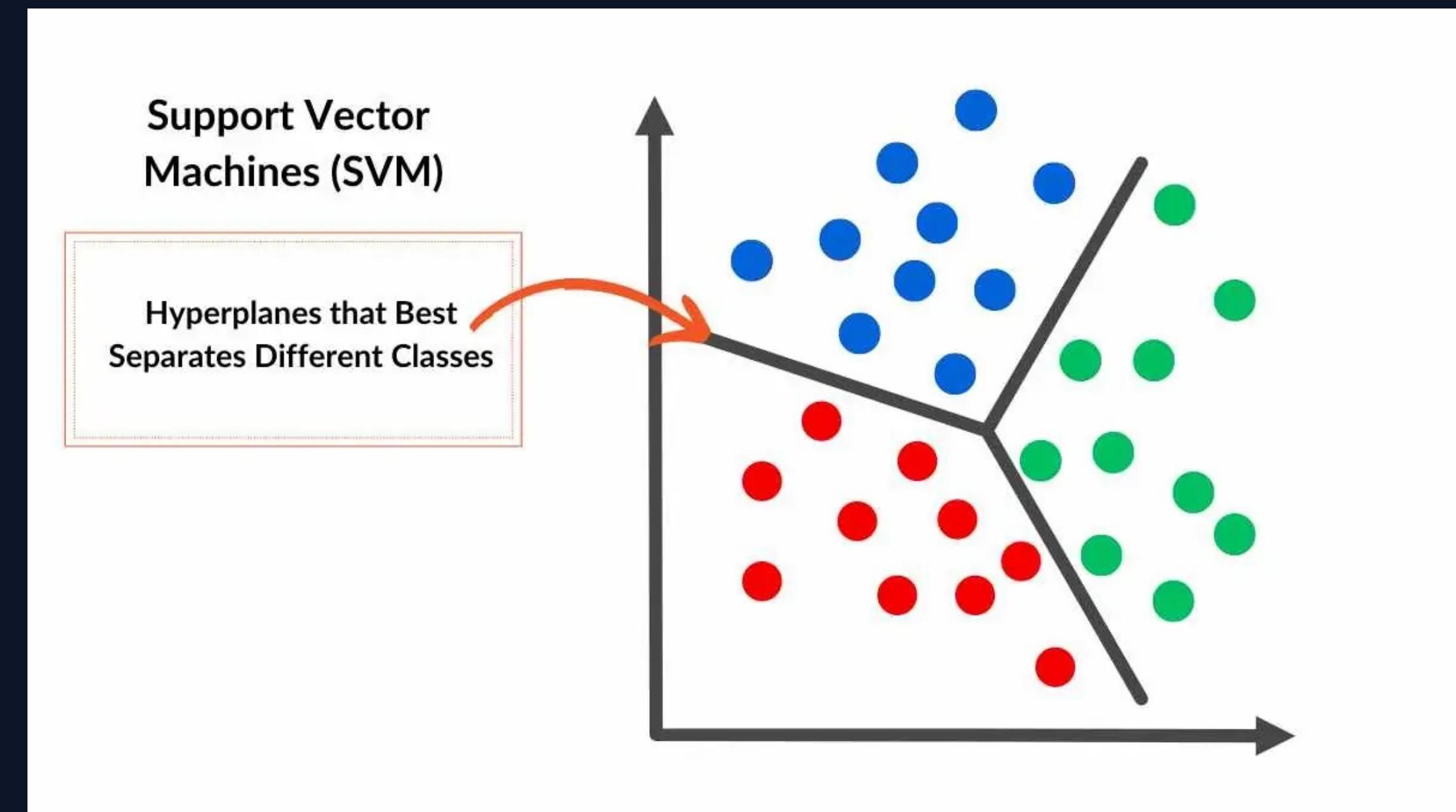


Introduction au Machine Learning

Les modèles de classification simple supervisé:

Support Vector Machine (SVM)

- Type : **linéaire ou non linéaire selon le kernel**
- Principe : cherche l'**hyperplan qui sépare les classes avec le maximum de marge**
- Avantages : efficace pour petits datasets, robuste aux dimensions élevées
- Limites : lent pour grands datasets, choix du kernel délicat
- Exemple : reconnaissance de caractères, classification bioinformatique



Introduction au Machine Learning

Scikit-learn

Qu'est-ce que scikit-learn ?

- Scikit-learn est une **bibliothèque Python de Machine Learning** très populaire.
- Elle permet de faire facilement :
 - **Apprentissage supervisé** : régression, classification
 - **Apprentissage non supervisé** : clustering, réduction de dimension
 - **Prétraitement des données** : normalisation, encodage, gestion des valeurs manquantes
 - **Évaluation de modèles** : métriques, validation croisée
- Elle est **simple, cohérente et compatible avec pandas/numpy**.



Introduction au Machine Learning

Scikit-learn

Installation

- <https://scikit-learn.org/stable/install.html>

```
python -m venv sklearn-env
```

```
sklearn-env\Scripts\activate # activate
```

```
pip install -U scikit-learn
```



Introduction au Machine Learning

Introduction au ML supervisé avec **Scikit-learn**

Le ML supervisé apprend à prédire une variable cible (y) à partir de variables explicatives (X).

X-> Y

- Exemple : prédire la **température demain** (y) à partir de l'humidité, la pression, le vent, etc. (X).
- On a besoin de **données étiquetées** (historique météo avec la valeur de température connue).

Introduction au Machine Learning

Les grandes étapes du ML supervisé avec **Scikit-learn**

Préparer les données

- Séparer X (features) et y (target)
- Optionnel : scaler les données (StandardScaler)

Diviser le dataset

- `train_test_split()` pour créer train/test

Choisir un modèle

- Exemple : `LinearRegression`, `RandomForestClassifier`, `KMeans`

Entraîner le modèle

- `model.fit(X_train, y_train)`

Faire des prédictions

- `y_pred = model.predict(X_test)`

Évaluer le modèle

- `mean_squared_error`, `accuracy_score`, `r2_score`, etc.