



# Introduction à la Data Science et au Machine Learning



Amina MARIE

# Objectifs

---


- Comprendre les concepts fondamentaux de la data science et du machine learning.
- Explorer les applications réelles de ces disciplines.
- Découvrir et se familiariser avec les outils et bibliothèques utilisés en data science.



# Qu'est-ce que la Data Science

## **Data Science :**

La science qui utilise des méthodes, algorithmes et systèmes pour extraire des connaissances à partir de données.






# Qu'est-ce que la Data Science

La **data science**, c'est comme être un détective dans une immense bibliothèque magique où chaque livre contient des informations mystérieuses.


- Les **données**, ce sont les livres
- Le **data scientist**, c'est le détective
- Les **algorithmes** sont comme les méthodes de déduction du détective
- Le **résultat**, c'est le trésor du détective




# Qu'est-ce que le Machine Learning

## **Machine Learning (ML) :**

Une branche de l'intelligence artificielle où les machines apprennent à partir de données sans être explicitement programmées.





# Qu'est-ce que le Machine Learning

Le **machine learning**, c'est comme apprendre à un enfant à reconnaître des animaux dans des photos.

- Les **données**, ce sont les photos
- L'**algorithme**, c'est le cerveau de l'enfant
- Le **data scientist**, c'est toi
- Une fois qu'il a vu assez d'exemples, l'enfant peut reconnaître tout seul un animal qu'il n'a jamais vu, juste en s'appuyant sur ce qu'il a appris.



## Application concrètes de la DataScience

- **Santé** : Détection précoce des maladies via des modèles prédictifs.
- **Finance** : Analyse de risques et prévention des fraudes.
- **Marketing** : Personnalisation des offres et ciblage des clients.
- **Transport** : Optimisation des itinéraires et maintenance prédictive.



# Outils et bibliothèques en DataScience

- **Python** : Langage polyvalent et accessible.
- **Jupyter Notebook** : Environnement interactif pour coder et visualiser des données.
- **Pandas** : Manipulation de données.
- **Scikit-learn** : Algorithmes de machine learning.
- **Matplotlib & Seaborn** : Visualisation.





# Exercice

Vous êtes récemment embauché  
comme data scientist dans une  
startup technologique spécialisée  
dans l'analyse des données  
environnementales.



# Exercice

Votre 1ère mission consiste à explorer et analyser un ensemble de données météorologiques fourni par l'entreprise.

L'objectif : dégager des insights initiaux sur les tendances des températures, précipitations et autres variables climatiques afin d'éclairer des décisions stratégiques



# Consignes

- **Dataset** : weather\_data.csv (doit inclure des colonnes comme "Date", "Temperature", "Precipitation").
- **Environnements** : Python, Jupyter Notebook ou un IDE comme VSCode.
- **Bibliothèques à installer** : pandas, matplotlib, seaborn



## Livrables attendus

- Un fichier initial et le fichier transformé (weather\_data\_transformed.csv).
- Une visualisation claire des tendances et relations (graphiques générés).
- Partager jupyter notebook

Mon mail: [amina.marie@mail-formateur.net](mailto:amina.marie@mail-formateur.net)



# Créer le dataset

```
import pandas as pd
import numpy as np

# Fixer le nombre total de lignes
num_rows = 1000

# Générer des dates aléatoires sur une année (2023)
dates = pd.date_range(start="2023-01-01", end="2023-12-31", freq="H") # Observations horaires
random_dates = np.random.choice(dates, num_rows, replace=True) # Sélection aléatoire parmi ces dates

# Générer des données météorologiques aléatoires
np.random.seed(42) # Pour des résultats reproductibles

# Températures en °C (variation avec bruit)
temperature = np.round(10 + 15 * np.sin(2 * np.pi * pd.to_datetime(random_dates).dayofyear / 365) +
                        np.random.normal(0, 3, num_rows), 1)

# Précipitations en mm (valeurs discrètes, majorité des jours secs)
precipitation = np.random.choice([0, 0, 5, 10, 15, 20], size=num_rows, p=[0.6, 0.2, 0.1, 0.05, 0.03, 0.02])

# Humidité en % (variation saisonnière avec bruit)
humidity = np.round(np.clip(50 + 20 * np.sin(2 * np.pi * pd.to_datetime(random_dates).dayofyear / 365) +
                            np.random.normal(0, 10, num_rows), 30, 100), 1)

# Coordonnées géographiques fictives
latitude = np.random.uniform(-90, 90, num_rows)
longitude = np.random.uniform(-180, 180, num_rows)

# Création du DataFrame
weather_data = pd.DataFrame({
    "Date": random_dates,
    "Temperature": temperature,
    "Precipitation": precipitation,
    "Humidity": humidity,
    "Latitude": latitude,
    "Longitude": longitude
})

# Ajouter des colonnes dérivées
weather_data["Temperature_F"] = weather_data["Temperature"] * 9/5 + 32 # Conversion en Fahrenheit
weather_data["Is_Hot"] = (weather_data["Temperature"] > 25).astype(int) # Indicateur binaire pour journées chaudes

# Trier les données par date
weather_data = weather_data.sort_values(by="Date").reset_index(drop=True)

# Enregistrer dans un fichier CSV
weather_data.to_csv("weather_data_1000.csv", index=False)

print("Dataset météo avec 1000 lignes généré et sauvegardé dans 'weather_data_1000.csv'.")
```

# Etape 1

## Chargement des données dans Python avec pandas

**Objectif :** Se familiariser avec la bibliothèque pandas pour importer un fichier CSV.

**Consigne :** Téléchargez le fichier `weather_data.csv` fourni. Utilisez pandas pour charger les données dans un DataFrame.

### Vérifiez :

- Le nombre de lignes et colonnes du DataFrame (`data.shape`).
- Les types de données de chaque colonne (`data.dtypes`)

## Etape 2

### Résumé statistique des données

**Objectif :** Comprendre les données grâce à des résumés statistiques.

**Consigne :** Utilisez la méthode `.describe()` pour générer des statistiques descriptives.

**Question :**

Quels sont les minimums et maximums des températures dans cet ensemble de données ?

## Etape 3

### Nettoyage des données (gestion des valeurs manquantes)

**Objectif :** Traiter les données incomplètes.

**Consigne :**

- Identifiez les colonnes contenant des valeurs manquantes avec `.isnull().sum()`.
- Remplissez les valeurs manquantes des températures par la moyenne de la colonne.



## Etape 4

### Création de colonnes dérivées

**Objectif :** Ajouter une nouvelle colonne basée sur des calculs.

**Consigne :**

- Créez une colonne appelée Temp\_Fahrenheit qui convertit les températures de Celsius en Fahrenheit.
- Formule :  $\text{Temp\_Fahrenheit} = \text{Temp\_Celsius} * 9/5 + 32.$

## Etape 5

### Filtrage des données

**Objectif :** Sélectionner des données selon des conditions spécifiques.

**Consigne :**

- Filtrez et affichez toutes les lignes où la température dépasse 30 °C.

## Etape 6

### Visualisation des données avec Matplotlib

**Objectif :** Apprendre à créer un graphique simple pour visualiser une colonne.

**Consigne :**

- Créez un histogramme de la distribution des températures.
- Ajoutez des titres et des étiquettes aux axes.

## Etape 7

### Exploration des relations avec Seaborn

**Objectif :** Identifier des relations entre deux variables.

**Consigne :**

- Utilisez Seaborn pour créer un nuage de points entre les températures et les précipitations.

## Etape 8

### Détection des tendances temporelles

**Objectif :** Analyser les données en fonction du temps.

**Consigne :**

- Convertissez la colonne "Date" en un format datetime avec pandas.
- Créez un graphique montrant les variations des températures au fil du temps.

## Etape 9

### Résumé des corrélations

**Objectif :** Analyser les relations statistiques entre toutes les variables.

**Consigne :**

- Calculez la matrice de corrélation des colonnes numériques.
- Affichez cette matrice sous forme de heatmap avec Seaborn.

## Etape 10

### Exportation des données transformées

**Objectif :** Enregistrer le travail réalisé pour une utilisation ultérieure.

**Consigne :**

- Enregistrez le DataFrame transformé dans un nouveau fichier CSV

# Listes Fonctions et méthodes fréquemment utilisées

## 1. Chargement et Prévisualisation des Données

### Pandas (Bibliothèque pour la manipulation des données)

- **pd.read\_csv(filepath)** : Charge un fichier CSV dans un DataFrame pandas.

Exemple : `data = pd.read_csv("weather_data.csv")`.

- **DataFrame.head(n)** : Affiche les n premières lignes d'un DataFrame (5 par défaut).

Exemple : `print(data.head())`.



# Listes Fonctions et méthodes fréquemment utilisées

- **DataFrame.shape** : Renvoie le nombre de lignes et de colonnes sous la forme (rows, columns).

Exemple : `data.shape`.

- **DataFrame.dtypes** : Affiche les types de données de chaque colonne.

Exemple : `print(data.dtypes)`.



# Listes Fonctions et méthodes fréquemment utilisées

## 2. Résumé Statistique et Exploration

- **DataFrame.describe()** : Donne des statistiques descriptives des colonnes numériques (moyenne, écart-type, minimum, etc.).

Exemple : `print(data.describe())`.

- **DataFrame.isnull().sum()** : Compte le nombre de valeurs manquantes (NaN) par colonne.

Exemple : `print(data.isnull().sum())`.

- **DataFrame['Column'].value\_counts()** : Renvoie la fréquence de chaque valeur unique dans une colonne.

Exemple : `print(data['City'].value_counts())`.

# Listes Fonctions et méthodes fréquemment utilisées


## 3. Nettoyage des Données

- **DataFrame.fillna(value)** : Remplit les valeurs manquantes avec un value spécifié.

Exemple : `data['Temperature'] = data['Temperature'].fillna(20).`

- **DataFrame.dropna()** : Supprime les lignes contenant des valeurs manquantes.

Exemple : `data = data.dropna()`



# Listes Fonctions et méthodes fréquemment utilisées

## 4. Création et Modification de Colonnes

- **Créer une nouvelle colonne :**

- Syntaxe : `DataFrame['Nouvelle_Colonne'] = Calcul ou Valeur.`
- Exemple : `data['Temp_Fahrenheit'] = data['Temperature'] * 9/5 + 32.`

- **Appliquer une fonction à une colonne :**

- Syntaxe : `DataFrame['Colonne'].apply(fonction).`
- Exemple : `data['Rounded_Temp'] = data['Temperature'].apply(round).`

# Listes Fonctions et méthodes fréquemment utilisées

## 5. Filtrage des Données

- **Filtrer avec une condition :**
  - Syntaxe : `DataFrame[DataFrame['Colonne'] condition]`.
  - Exemple : `hot_days = data[data['Temperature'] > 30]`.
- **Sélection de colonnes spécifiques :**
  - Syntaxe : `DataFrame[['Colonne1', 'Colonne2']]`.
  - Exemple : `selected_columns = data[['Date', 'Temperature']]`

# Listes Fonctions et méthodes fréquemment utilisées

## 6. Visualisation des Données

### Matplotlib (Bibliothèque de visualisation basique)

- **plt.hist(data['Colonne'], bins=n)** : Crée un histogramme avec n catégories.
- **plt.plot(x, y)** : Trace une courbe reliant les points (x, y).
- **plt.title()** : Ajoute un titre au graphique.
- **plt.xlabel()** et **plt.ylabel()** : Ajoutent des étiquettes aux axes X et Y.

# Listes Fonctions et méthodes fréquemment utilisées

## 6. Visualisation des Données

### Seaborn (Bibliothèque de visualisation avancée)

- **`sns.scatterplot(x='Colonne1', y='Colonne2', data=DataFrame)`** : Crée un nuage de points.
- **`sns.heatmap(data, annot=True)`** : Crée une carte thermique annotée.

A large orange circle on the left side of the slide, partially cut off by the edge.

# Listes Fonctions et méthodes fréquemment utilisées

## 7. Exportation des Données

- **DataFrame.to\_csv(filepath, index=False)** : Exporte un DataFrame dans un fichier CSV.

Exemple : `data.to_csv("weather_data_transformed.csv", index=False)`.

