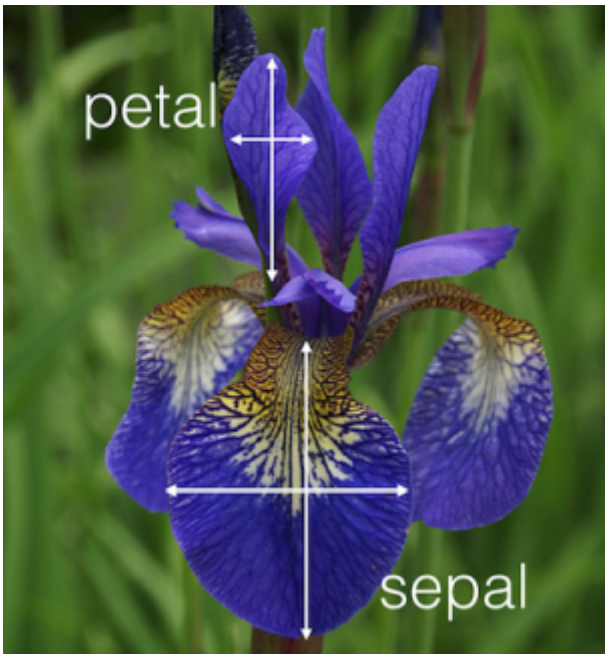
TP Manipulation et visualisation des données



Objectifs

- Maîtriser les opérations fondamentales de pandas (chargement, exploration, manipulation)
- Savoir lire/écrire des fichiers CSV
- Effectuer des agrégations et des statistiques par groupe
- Gérer les valeurs manquantes
- Produire des visualisations variées avec matplotlib

Prérequis

- Python , pandas, matplotlib, numpy, Seaborn

Partie 1 — Chargement et exploration

1. Importer le dataset Iris depuis scikit-learn (`from sklearn.datasets import load_iris`)
2. Construire un DataFrame avec les colonnes : `SepalLengthCm`, `SepalWidthCm`, `PetalLengthCm`, `PetalWidthCm`, `Species`
3. Afficher les 5 premières et 5 dernières lignes (`head()`, `tail()`)
4. Afficher la forme du DataFrame (`shape`)
5. Afficher `df.info()` et `df.describe()`
6. Afficher les types de données (`dtypes`)

Partie 2 — Sélections et indexation

1. Sélectionner uniquement les colonnes `SepalLengthCm` et `PetalLengthCm`
2. Utiliser `loc[]` pour sélectionner les lignes 10 à 20 (inclus), colonnes `Species` et `PetalWidthCm`
3. Utiliser `iloc[]` pour sélectionner les 5 premières lignes, 2 premières colonnes
4. Filtrer les fleurs où `SepalLengthCm > 6.0`
5. Filtrer avec plusieurs conditions : `SepalLengthCm > 5.5` ET `PetalLengthCm < 4.0`
6. Utiliser la méthode `query()` pour filtrer l'espèce *versicolor*

Partie 3 — Manipulation des colonnes et lignes

1. Ajouter une colonne `PetalRatio = PetalLengthCm / PetalWidthCm`
2. Ajouter une colonne `SepalArea = SepalLengthCm * SepalWidthCm`
3. Créer une colonne catégorielle `Taille` :
    - "petite" si `SepalLengthCm < 5.0`
    - "moyenne" si `5.0 <= SepalLengthCm < 6.5`
    - "grande" sinon
4. Supprimer la colonne `SepalArea`
5. Supprimer les lignes où `SepalLengthCm < 5.0`
6. Trier le DataFrame par `PetalLengthCm` décroissant
7. Réinitialiser l'index après les suppressions

Partie 4 — Gestion des valeurs manquantes

1. Introduire artificiellement des `NaN` dans 5 cellules aléatoires de la colonne `PetalWidthCm` :

```
import numpy as np
df.loc[df.sample(5).index, "PetalWidthCm"] = np.nan
```

2. Compter le nombre de valeurs manquantes par colonne (`isna().sum()`)
3. Afficher les lignes contenant des `NaN` (`df[df.isna().any(axis=1)]`)
4. Remplacer les `NaN` par la moyenne de la colonne (`fillna()`)
5. Supprimer les lignes contenant des `NaN` (sur une copie, avec `dropna()`)

Partie 5 — Agrégations et groupby

1. Compter le nombre d'occurrences par espèce (`value_counts()`)
2. Calculer la moyenne de chaque variable numérique par espèce (`groupby().mean()`)
3. Calculer plusieurs statistiques par espèce : moyenne, écart-type, min, max

```
df.groupby("Species").agg(["mean", "std", "min", "max"])
```

4. Calculer les percentiles 25%, 50%, 75% de `PetalLengthCm` par espèce
5. Créer un tableau croisé entre `Species` et `Taille` (`pd.crosstab()`)

Partie 6 — Lecture/écriture de fichiers

1. Exporter le DataFrame en CSV sans l'index :

```
df.to_csv("iris_enrichi.csv", index=False)
```

2. Recharger le fichier CSV dans un nouveau DataFrame
3. Vérifier que les données sont identiques
4. Exporter en CSV avec séparateur `;` et recharger

Partie 7 — Visualisations

7.1 Histogramme

- Tracer l'histogramme de `SepalLengthCm` (15 bins)

7.2 Nuage de points simple

- Tracer `SepalLengthCm` vs `PetalLengthCm`

7.3 Nuage de points coloré par espèce

- Même graphique mais avec une couleur différente par espèce et une légende

7.4 Boxplots par espèce

- Tracer le boxplot de `PetalLengthCm` pour chaque espèce

7.5 Diagramme en barres

- Afficher le nombre de fleurs par espèce

7.6 Matrice de corrélation

- Calculer `df.corr()` sur les colonnes numériques
- Afficher sous forme de heatmap avec `plt.imshow()` ou `plt.matshow()`

7.7 (Bonus) Subplots multiples

- Créer une figure avec 4 sous-graphiques (2×2) montrant les histogrammes des 4 variables numériques

---

Partie 8 — Exercices bonus (optionnel)

1. Outlier : Identifier la fleur avec le plus grand `PetalRatio` — est-ce cohérent ?
2. Statistiques avancées : Calculer le coefficient de variation (std/mean) par espèce
3. Apply : Utiliser `apply()` pour créer une colonne indiquant si la fleur est "typique" (toutes les mesures proches de la moyenne de son espèce ± 1 écart-type)

Conseils

- Utilisez `load_iris(as_frame=True)` pour obtenir directement un DataFrame
- Pour le scatter coloré : créez un dictionnaire `colors = {"setosa": "red", ...}` puis mappez
- Pour le boxplot : `plt.boxplot()` avec une liste de séries
- Pensez à `plt.figure()` avant chaque graphique
- `plt.tight_layout()` évite les chevauchements