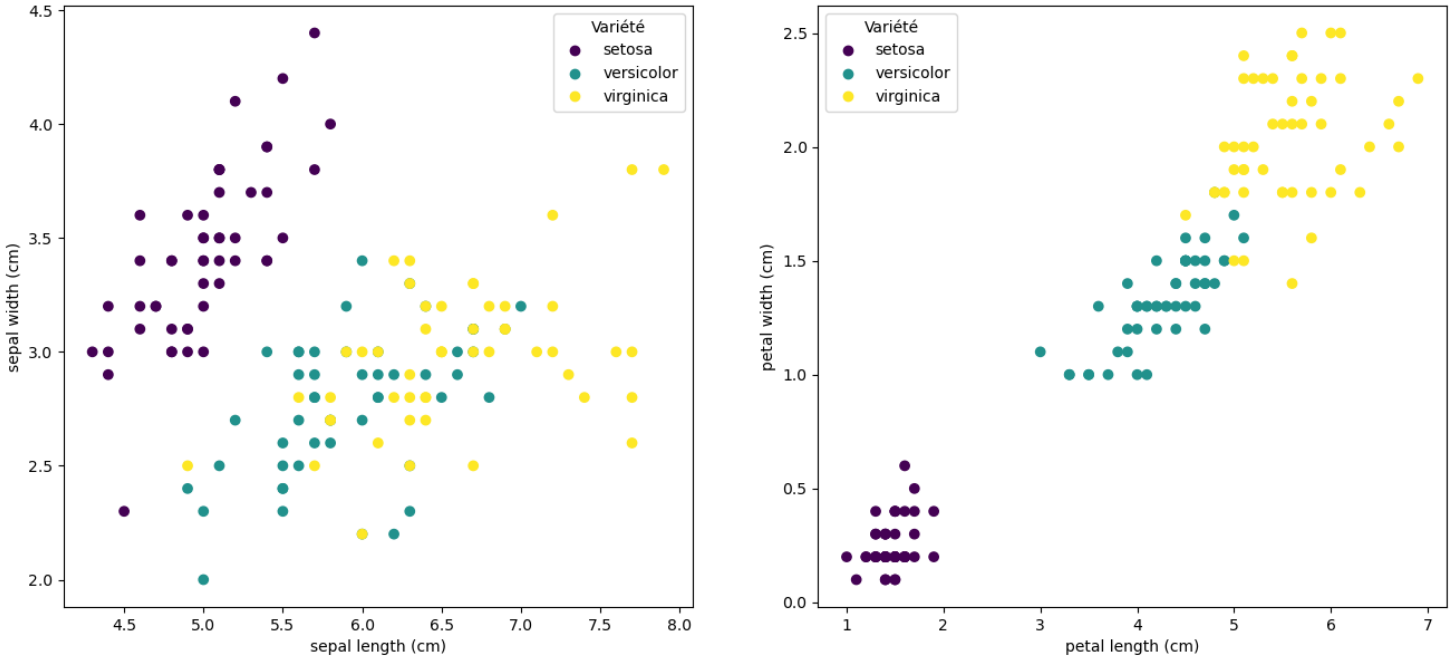
Nous nous intéressons dans ce TP à un problème classique de *machine learning* (apprentissage automatique) : le problème de classification. Nous cherchons à reconnaitre et identifier trois variétés d'iris en implémentant l'algorithme algorithme des k plus proches voisins ou *k-nearest neighbors* (kNN). Pour charger les données, on exécutera le code suivant :

```
import numpy as np
import matplotlib.pyplot as plt
import random as rd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
iris = load_iris()
```

1 Prise en main des données

Chaque iris est caractérisé par les cinq informations suivantes : la variété - codée par un entier, *Iris setosa* (0), *Iris versicolor* (1) et *Iris virginica* (2) - ainsi que les longueur et largeur moyennes des sépales et des pétales (en cm). L'instruction `iris.target[i]` renvoie l'entier codant pour la variété de la $(i+1)$-ème fleur de la banque de données `iris`, tandis que l'instruction `iris.data[i]` renvoie un tableau (`array`) de flottants contenant, dans l'ordre, la longueur moyenne des sépales, la largeur moyenne des sépales, la longueur moyenne des pétales, la largeur moyenne des pétales de la $(i+1)$-ème fleur.

Toutes les données sont résumées sur le graphique ci-dessous (le code fourni pour les plus curieux).




```
couleurs = np.array(['indigo', 'darkcyan', 'gold'])
fig, ax = plt.subplots(1, 2, figsize=(16,7))
for i in range(2):
    ax[i].scatter(iris.data[:,2*i], iris.data[:,2*i+1], color = couleurs[iris.target])
    h = [plt.Line2D([0,0],[0,0],color=couleurs[j],marker='o',linestyle='', label=l)
         for j, l in enumerate(iris.target_names)]
    ax[i].legend(handles=h, title='Variété')
    ax[i].set_xlabel(iris.feature_names[2*i])
    ax[i].set_ylabel(iris.feature_names[2*i + 1])
plt.show()
```

Question 1. Quel est le type des objets `iris.data` et `iris.target` ? Déterminer le nombre de fleurs de la banque et afficher la variété de chacun des iris de la banque.

On pourra utiliser le dictionnaire : `variete = {0 : "setosa", 1: "versicolor", 2 : "virginica"}`.

Question 2. Identifier la variété et les longueurs et largeurs moyennes de sépales et pétales de la première fleur, de la 51ème et de la dernière fleur de la banque. On pourra compléter le code ci-dessous :

```
for k in [...]:
    print(variete[...])
    print("longueur des sépales : ", ..., "cm")
    print("largeur des sépales : ", ..., "cm")
    print("longueur des pétales : ", ..., "cm")
    print("largeur des pétales : ", ..., "cm")
```

2 Classification des variétés

Pour implémenter un algorithme d'apprentissage, on sépare les données en deux (de même taille) : un jeu de données servira à l'entrainement et un autre à tester les performances de l'algorithme.

L'instruction ci-dessous sépare aléatoirement les objets `iris.data` et `iris.target` en deux : un jeu de données d'entrainement (`train_data` et `train_target`) et un de données de test (`test_data` et `test_target`).

```
train_data, test_data, train_target, test_target \
= train_test_split(iris.data, iris.target, test_size=0.5, shuffle=True, random_state=12)
```

Question 3. Combien d'iris de chaque variété sont dédiés aux données d'entrainement ? Identifier la variété de la première fleur du jeu d'entrainement et celle de la première fleur du jeu de test.

On définit la distance (euclidienne) entre deux arrays $\mathtt{x} = [x_0, x_1, x_2, x_3]$ et $\mathtt{y} = [y_0, y_1, y_2, y_3]$ comme le réel :

$$d(\mathtt{x}, \mathtt{y}) = \sqrt{(x_0 - y_0)^2 + (x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}.$$

Question 4. Écrire une fonction `d` prenant en argument deux arrays (de 4 flottants) et renvoyant leur distance. *Vérifier que la distance du $55^{ème}$ et du $68^{ème}$ iris de la banque de données initiale est égale à 1.*

2.1 Détermination du plus proche voisin

On applique ici l'algorithme des k plus proches voisins dans le cas où $k = 1$: pour prédire la variété d'un iris (de la partie test), on lui attribue la variété de son plus proche voisin (pour la distance d) **parmi tous les iris du jeu des données d'entrainement**.

Question 5. Écrire une fonction `variete_plus_proche` qui prend en argument un array `x` de quatre flottants représentant les caractéristiques d'un iris (longueur et largeur moyennes des sépales et pétales) et qui renvoie la variété de l'iris du jeu des données d'entrainement le plus proche de `x`.

Question 6. En appliquant l'algorithme précédent, comparer les variétés prédites par l'algorithme kNN dans le cas $k = 1$ avec les variétés réelles du jeu de données test. Déterminer la précision de votre algorithme, i.e. le pourcentage de variétés correctement prédites.

2.2 Détermination des k plus proches voisins

Pour prédire la variété d'un iris du jeu de données test, on détermine ses k plus proches voisins (pour la distance d) parmi les iris du jeu de données d'entrainement. On attribue alors à l'iris test la variété majoritaire parmi les k iris les plus proches. On se donne un array `x` de quatre flottants représentant les caractéristiques d'un iris (longueur et largeur moyennes des sépales et pétales). On cherche à trier tous les iris du jeu de données d'entrainement dans l'ordre croissant de leur distance à l'iris `x`.

Question 7. Compléter l'algorithme ci-dessous afin d'adapter l'algorithme du tri rapide à notre problème. **Au lieu de trier simultanément `train_data` et `train_target`, on triera la liste `L` des indices des iris du jeu des données d'entrainement.**

```
def partition(L,x):
    pivot = ...
    inf, sup = ..., ...
    for e in L[1:]:
        if d(train_data[e],x) < ... :
            ...
        else:
            ...
    return pivot, inf, sup
```

```
def tri_rapide(L, x):
    if ... :
        return L
    pivot, inf, sup = partition(L,x)
    inf = tri_rapide(inf, x)
    sup = tri_rapide(sup, x)
    return ...
```

Question 8. En déduire une fonction `k_plus_proches_voisins` qui prend en argument un entier k et un array `x` de quatre flottant représentant un iris du jeu de données test, et qui détermine la liste des indices des k plus proches voisins de `x`.

Question 9. Écrire une fonction `occurrences` qui prend en argument une liste `L` d'entiers entre 0 et 2 représentant les trois variétés d'iris, et qui renvoie une liste `occ` telle que `occ[i]` soit le nombre d'iris de la variété i pour tout $i \in [\![0, 2]\!]$. *Par exemple, l'instruction `occurrences([0,2,1,1,1,2,0,1,2,0])` devra renvoyer la liste `[3,4,3]`.*

Question 10. Écrire une fonction `variete_majoritaire` qui prend en argument une liste d'occurrences des trois variétés comme renvoyé par la question précédente et qui renvoie la variété majoritaire.

Question 11. En déduire une fonction `prediction_variete` qui prend en argument un entier k et un array `x` représentant les caractéristiques d'un iris du jeu de données test et qui renvoie la variété prédite par l'algorithme des k plus proches voisins.

Vérifier graphiquement à l'aide du code ci-dessous qui affiche les données d'entrainement ainsi que celle de $\boldsymbol{x}$ et sa couleur prédite, en mettant en valeur les données de $\boldsymbol{x}$ et ses k plus proches voisins.

```
def plot_k_plus_proches(x, k):
    indices_voisins = k_plus_proches_voisins(x, k)
    fig, ax = plt.subplots(1, 2, figsize=(16,7))
    for i in range(2):
        ax[i].scatter(train_data[:,2*i], train_data[:,2*i+1], color = couleurs[train_target])
        h = [plt.Line2D([0,0],[0,0],color=couleurs[j],marker='o',linestyle='', label=l) \
             for j, l in enumerate(iris.target_names)]
        ax[i].legend(handles=h, title='Variété :')
        ax[i].set_xlabel(iris.feature_names[2*i])
        ax[i].set_ylabel(iris.feature_names[2*i + 1])
        ax[i].scatter(x[2*i], x[2*i + 1], s=100, marker='X', \
              color=couleurs[prediction_variete(x, k)])
        for j in indices_voisins:
            v = train_data[j]
            ax[i].scatter(v[2*i],v[2*i+1],s=100,marker='o', color=couleurs[train_target[j]])
    plt.show()
```

Question 12. Écrire une fonction `precision` qui détermine pour un entier k donné la proportion de variétés (du jeu de données test) correctement inférées par l'algorithme des k plus proches voisins.

Question 13. Déterminer la précision maximale et la valeur de k qui réalise ce maximum.

Question 14. Représenter graphiquement la précision de l'algorithme sur le jeu de données test en fonction fonction de k pour $k \in [\![1, 50]\!]$. Comment peut-on expliquer la chute de précision observée à partir d'une certaine valeur de k ?