

Algorithme des k-plus proches voisins (k-Nearest Neighbours souvent abrégé KNN en anglais)

Principe de l'algorithme

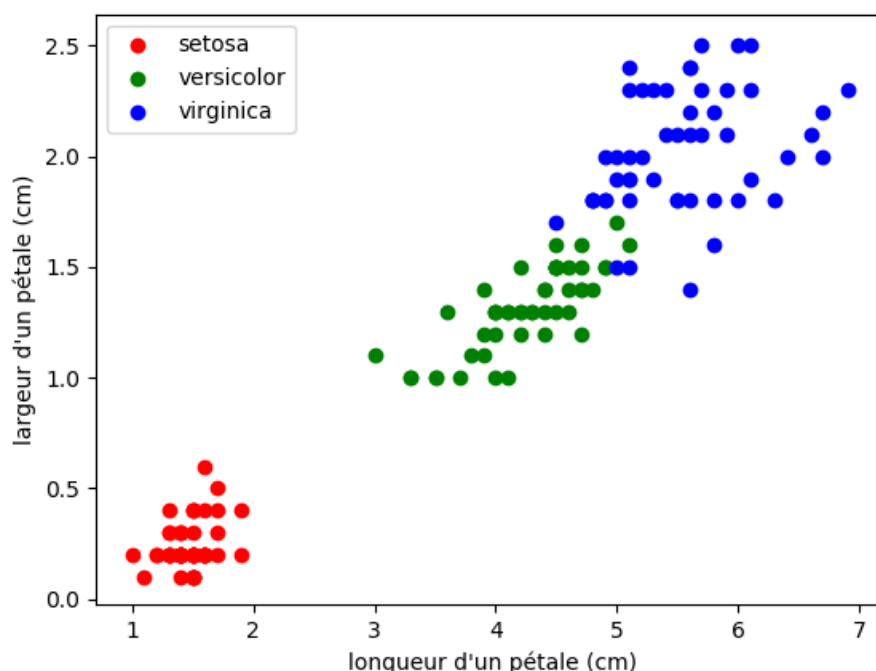
L'algorithme des k plus proches voisins appartient à la famille des **algorithmes d'apprentissage automatique** (machine learning). L'idée d'apprentissage automatique ne date pas d'hier, puisque le terme de machine learning a été utilisé pour la première fois par l'informaticien américain Arthur Samuel en 1959. Les algorithmes d'apprentissage automatique ont connu un fort regain d'intérêt au début des années 2000 notamment grâce à la quantité de données disponibles sur internet. L'algorithme des k plus proches voisins est un algorithme d'apprentissage supervisé, il est nécessaire d'avoir des données labellisées. À partir d'un ensemble E de données labellisées, il sera possible de classer (déterminer le label) d'une nouvelle donnée (donnée n'appartenant pas à E). À noter qu'il est aussi possible d'utiliser l'algorithme des k plus proches voisins à des fins de régression (on cherche à déterminer une valeur à la place d'une classe), mais cet aspect des choses ne sera pas abordé ici.

Remarque : de nombreuses sociétés (exemple les GAFAM) utilisent les données concernant leurs utilisateurs afin de "nourrir" des algorithmes de machine learning qui permettront à ces sociétés d'en savoir toujours plus sur nous et ainsi de mieux cerner nos "besoins" en termes de consommation.

Pour faire fonctionner cet algorithme, nous avons besoin d'un **jeu de données**

d'apprentissage. Pour que la notion soit plus facile à appréhender nous allons utiliser ici un jeu de données à deux dimensions afin de pouvoir représenter la situation par un graphique. Nous allons utiliser le jeu de données du fichier 'iris.csv' Nous n'utiliserons comme descripteurs d'une variété d'iris que la taille des pétales (longueur et largeur).

Si on place les données dans un repère en représentant en abscisses la longueur des pétales, en ordonnées la largeur des pétales et en attribuant une couleur différente à chaque variété d'iris on obtient la figure suivante :



Fonctionnement de l'algorithme :

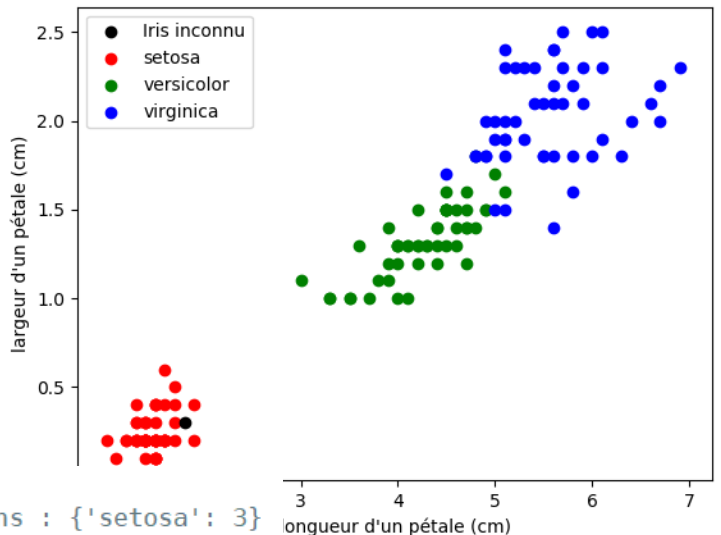
1. L'algorithme calcule les distances entre la nouvelle donnée et toutes les données du jeu d'apprentissage. Nous utiliserons ici la *distance Euclidienne* classique entre deux points, c'est à dire la longueur du segment reliant ces deux points.
2. L'algorithme ne conserve que les k plus proches voisins (d'où le nom de l'algorithme).
3. L'algorithme attribue à la nouvelle donnée **la classe** la plus fréquente parmi ces k voisins.

Remarque : il faut choisir une valeur de k avant d'utiliser l'algorithme. Les méthodes pour choisir la valeur de k la plus optimale ne seront pas développées ici.

Exemple n°1 :

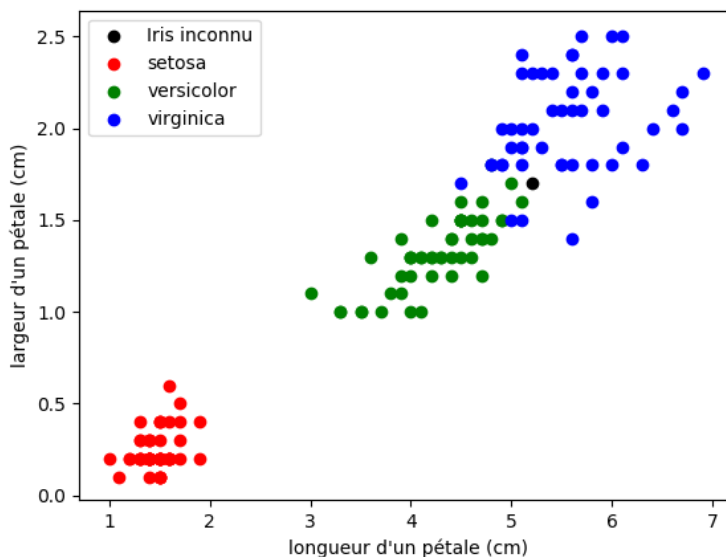
Si on applique l'algorithme avec $k = 3$ à un nouveau pétale de longueur 1,8 cm et de largeur 0,3 cm (point noir ci contre), on obtient :

```
>>> classeFleur(1.8,0.3,3)
Voici les 3 plus proches voisins : {'setosa': 3}
'setosa'
```

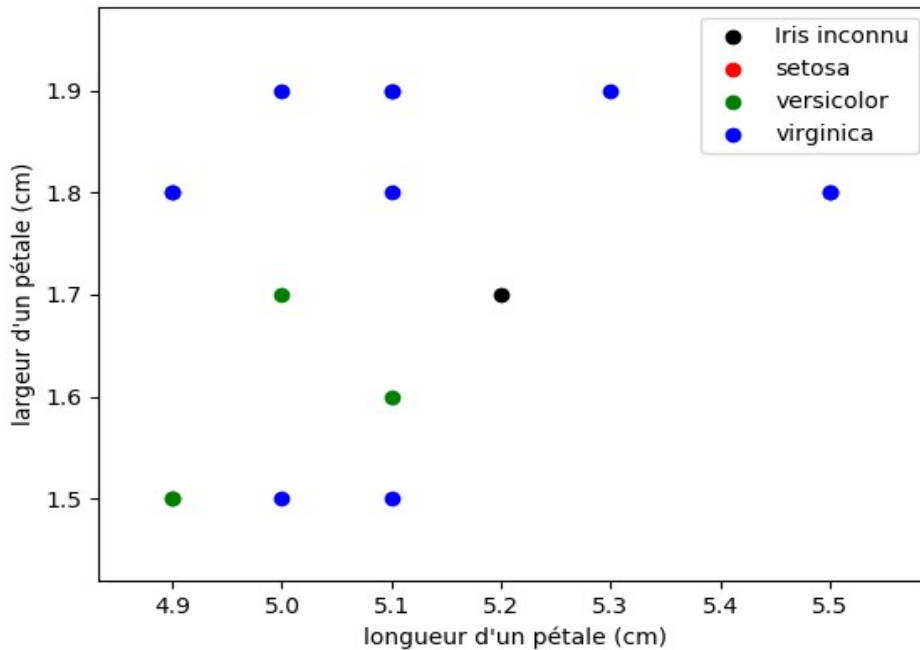


L'algorithme classe la nouvelle donnée dans la classe 'setosa'.

Exemple n°2:, où l'on essaie de classer une fleur ayant une longueur de pétale d'environ **5,2** cm et une largeur de pétale d'environ **1,7** cm :



Effectuons un zoom autour du point concerné :



Remarques :

- les points sont sur une grille car les données sont précises au mm près,
- le repère étant orthonormé sur la figure ci-dessus, on peut mesurer les longueurs sur le graphique.

Questions :

1. Donner les 3 plus proches voisins du point noir. Quelle variété va être attribuée à cette fleur par l'algorithme si $k = 3$?
2. Même question que précédemment mais avec $k = 7$.
3. Que se passe-t-il si $k = 1$?

Implémentation en Python

Le but de cette partie est de **compléter** l'algorithme des k plus proches voisins pour classer de nouvelles fleurs dans la bonne espèce d'iris.

Rappel : il faudra sauvegarder le fichier 'iris.csv' dans le même répertoire que votre script Python pour plus de facilité.

1. Compléter la fonction **importFichier(fichier)** qui permet d'importer un fichier CSV dont le nom est passé en paramètre et qui renvoie une liste contenant les données du fichier sans la première ligne d'en-tête.

```
>>> importFichier('iris.csv')
[['5.1', '3.5', '1.4', '0.2', 'setosa'], ['4.9', '3.0', '1.4', '0.2', 'setos
a'],
 ['4.7', '3.2', '1.3', '0.2', 'setosa'], ['4.6', '3.1', '1.5', '0.2', 'setos
a'],
 ['5.0', '3.6', '1.4', '0.2', 'setosa'], ['5.4', '3.9', '1.7', '0.4', 'setos
a'],
 ['4.6', '3.4', '1.4', '0.3', 'setosa'], ['5.0', '3.4', '1.5', '0.2', 'setos
a']]
```

2. Écrire une fonction **distance(point1,point2)** qui prend en paramètres deux points (un point est un tuple ou couple de la forme (x,y)) et qui renvoie la distance Euclidienne entre ces deux points.
N'oubliez pas d'importer la fonction racine carrée de la bibliothèque math : **from math import sqrt**

```
>>> distance((1,3.5),(1.5,3))
0.7071067811865476

>>> point1 = (1,5)
>>> point2 = (3,8)
>>> distance(point1,point2)
3.605551275463989
```

Rappel : $d(A, B) = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$.

3. Écrire une fonction **insertion(liste,elt)** où *liste* est une liste de couples (distance : float, variété : str), triée par ordre croissant des distances et *elt* un couple (distance,variété).

La fonction *insertion* renvoie la liste dans laquelle l'élément a été inséré à la bonne position pour conserver l'ordre croissant des distances.

Vous pouvez vous aider du travail fait sur le tri par insertion.

```
>>> liste = [(1.2,'setosa'),(2,'setosa'),(2.3,'virginica')]
>>> elt = (1.5,'versicolor')
>>> Insertion(liste,elt)
[(1.2, 'setosa'), (1.5, 'versicolor'), (2, 'setosa'), (2.3, 'virginica')]
```

4. Écrire une fonction **compteVarietes(listeVoisins)** qui prend en paramètre une liste de couples (distance,variete) comme précédemment et qui compte le nombre de fleurs de chaque variété dans la liste. Le résultat est un dictionnaire de clefs les noms des variétés et de valeurs le nombre de fleurs.

5. Écrire une fonction **maxFleurs(dico)** qui prend en paramètre un dictionnaire

```
>>> compteVarietes([(1.2, 'setosa'), (1.5, 'versicolor'), (2, 'setosa'), (2.3, 'virginica')])
{'setosa': 2, 'versicolor': 1, 'virginica': 1}
```

contenant le nombre de fleurs de chaque variété (clefs : variétés, valeurs : nombre) et qui renvoie la variété la plus présente dans le dictionnaire.

```
>>> dico = {'setosa': 2, 'versicolor': 1, 'virginica': 1}
>>> maxFleurs(dico)
'setosa'
```

TESTER l'algorithme :

Test de la fonction afficheFleurs Exemple :

afficheFleurs(importFichier('iris.csv'),1.8,0.3)

Test de la fonction classeFleur Corriger vos réponses de l'exemple 2.