Type de document	TP I.A.	Classe	1ère +Tle	Durée	2h	Date	23/02/2024
Thème et contenu(s)	Algorithmique, traitement des données, boucles						
Capacités attendues	Savoir lire un fichier .csv contenant un dataset, comprendre l'algorithme						
Prérequis	Algorithmique simple Python, mesure mathématique d'une distance Euclidienne						
Description	TP de Machine Learning à faire avec le populaire Iris Dataset, partie 1/2						

Le but de ce TP est de vous initier à l'**Intelligence Artificielle** (IA) en vous faisant mettre en œuvre pas à pas un algorithme simple **d'apprentissage supervisé** permettant de résoudre des problèmes de classification : **l'algorithme des K plus proches voisins**, ou **KNN** (K-Nearest Neighbours).

I) Lecture d'un fichier .csv

L'**apprentissage supervisé** est une des techniques utilisées en intelligence artificielle, l'idée est que l'on dispose d'un jeu de **données étiquetées**, c'est-à-dire que l'on sait à l'avance ce qu'elles représentent. On utilise ce jeu de données pour prédire le type de nouvelles données non étiquetées.

Commençons avec un exemple simple : on dispose de deux sortes de fruits sphériques, les fruits A et les fruits B. On dispose aussi d'une machine qui mesure leur diamètre. Pour chaque fruit, on consigne dans un tableau Excel son **type** et son **diamètre**. On enregistre ce tableau au format **`.csv`**.

Ci-dessous figure le fichier CSV **`fruits.csv`** ouvert dans Excel ou bien dans un éditeur de texte classique. Pour mémoire, un **`.csv`** n'est rien d'autre qu'un fichier texte dans lequel la 1ère ligne contient le **nom des colonnes** séparés par un **`;`** et les lignes suivantes les **données** séparées aussi par **`;`**

fruits.csv (ouvert dans Excel)

Diametre (mm)	Type
50	A
120	B
114	B
72	A
123	B
48	A
57	A
75	B
98	B
43	A

fruits.csv (ouvert dans un éditeur de texte)

```
Diametre (mm);Type
50;A
120;B
114;B
72;A
123;B
48;A
57;A
75;B
98;B
43;A
```

On peut se poser la question suivante : si on nous donne un nouveau fruit dont seul le diamètre est connu, peut-on deviner son type ?

Pour y répondre, un début de solution serait de représenter graphiquement nos fruits. C'est facile, on a une seule dimension ! Voici le croquis de ce que je voudrais obtenir :

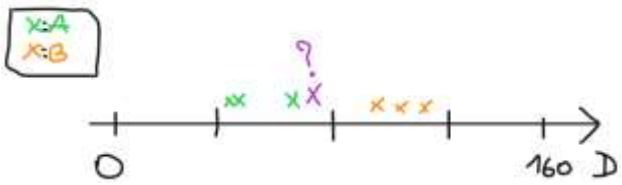


Figure 1

Mais avant d'arriver à cela, il faut d'abord extraire les données pour pouvoir les lire. C'est le but du code suivant qui se trouve dans le fichier **`lecture_csv_fruits.py`**. Pour le faire fonctionner,

assurez-vous bien entendu qu'il se trouve dans le même répertoire que **fruits.csv**, sinon changez le chemin à la ligne 5 ! En encadré, j'ai fait figurer le résultat de son exécution.

```
1   #importe la bibliothèque CSV
2   import csv
3
4   #création d'un handler (manipulateur) de fichier
5   fichier_csv = open('fruits.csv')
6
7   #création d'un objet de type reader
8   #on passe le handler de fichier en paramètre
9   #on n'oublie pas de lui indiquer le séparateur
10  lecteur_csv = csv.reader(fichier_csv, delimiter=";")
11
12  #déclaration d'un compteur p
13  cpt = 0
14
15  #parcours du lecteur csv
16  for ligne in lecteur_csv:
17      print("ligne",cpt,":",ligne)
18      cpt = cpt + 1
```

```
ligne 0 : ['Diametre (mm)', 'Type']
ligne 1 : ['50', 'A']
ligne 2 : ['120', 'B']
ligne 3 : ['114', 'B']
ligne 4 : ['72', 'A']
ligne 5 : ['123', 'B']
ligne 6 : ['48', 'A']
ligne 7 : ['57', 'A']
ligne 8 : ['75', 'B']
ligne 9 : ['98', 'B']
ligne 10 : ['43', 'A']
```

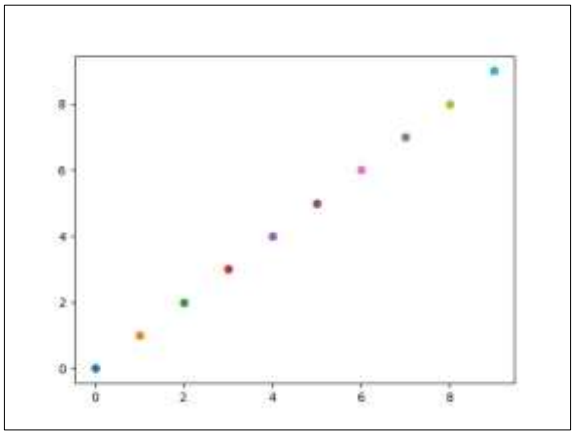
On remarque donc que chaque ligne est une liste contenant autant d'éléments que de colonnes dans le tableau Excel. Chaque élément est de type **<str>**. Pour chaque ligne (sauf la ligne 0) on a donc une liste de 2 chaînes de caractères représentant : le diamètre du fruit, et son type.

II) Utilisation basique de Matplotlib

Maintenant qu'on sait extraire les données d'un fichier .csv, nous allons chercher à les représenter graphiquement. Pour cela, nous allons utiliser la bibliothèque incontournable **Matplotlib**. Cette bibliothèque Python est très populaire car elle permet de faire énormément de choses, mais, heureusement, on peut l'utiliser simplement et arriver à des résultats très satisfaisants en quelques minutes.

Nous allons l'utiliser pour afficher des points, que nous n'allons pas lier entre eux (cela n'aurait pas de sens car nous représentons des éléments distincts !). Je vous montre un exemple enfantin dans **tuto_matplotlib.py** en 6 lignes de code que voici, ainsi que le résultat de leur exécution en encadré :

```
1   import matplotlib.pyplot as plt
2
3   for i in range(10):
4       plt.plot(i,i,"o")
5
6   plt.show()
```



Ici, le caractère 'o' est un alias pour dire que nous voulons des points ronds. On a affiché 10 points de coordonnées **(x,y)** avec **x=y** et **0≤x<10** c'est-à-dire que ces points se situent sur une droite. Or

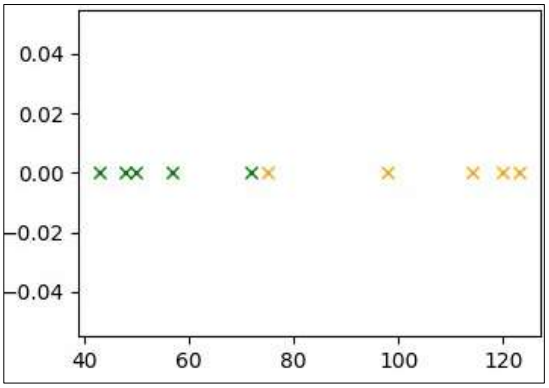
nous voulons, comme le montre le croquis de la figure 1, que les diamètres soient sur l'axe des abscisses. On mettra donc les coordonnées **y** à 0 lorsqu'on fera notre implémentation. Ces points sont, par défaut, représentés avec des couleurs différentes.

Modifions à présent **`lecture_csv_fruits.py`** pour que :

- La ligne 0 ne soit pas utilisée (elle ne représente pas un fruit)
- Les données numériques extraites sous forme de **`<str>`** soient converties en **`<int>`**
- Le type de fruit soit isolé afin de pouvoir colorier nos points
- Les diamètres des fruits soient représentés avec des croix, pas des ronds

Le fichier **`matplotlib_fruits.py`**, dont le code ainsi que le résultat produit sont donnés ici, prend en compte toutes ces modifications :

```
1   import csv
2
3   #importe la bibliothèque matplotlib et pyplot
4   import matplotlib.pyplot as plt
5
6   fichier_csv = open('fruits.csv')
7   lecteur_csv = csv.reader(fichier_csv, delimiter=";")
8   cpt = 0
9
10  for ligne in lecteur_csv:
11      print("ligne",cpt,":",ligne)
12      if cpt>0:
13
14          #on récupère d et on le convertit en int
15          diametre = int(ligne[0])
16
17          #on récupère le caractère du fruit
18          fruit = ligne[1]
19
20          #en fonction du type de fruit on définit
21          #une couleur (connue par matplotlib !)
22          if fruit == 'A':
23              c = "green"
24          else :
25              c = "orange"
26
27          #on plot le point d à l'ordonnée y=0
28          #en lui appliquant la couleur c
29          plt.plot(diametre,0,'x',color=c)
30
31      cpt = cpt + 1
32
33  #on affiche notre beau graphique
34  plt.show()
```



Pour l'alléger, j'ai uniquement fait figurer les commentaires qui concernent **`matplotlib`**.

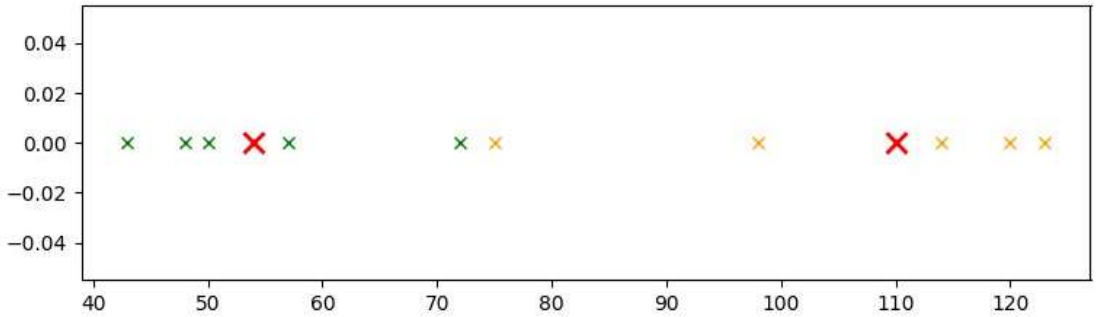
Bravo, vous savez à présent lire un fichier .csv et afficher les données numériques qu'il contient sous forme d'un graphique dont vous contrôlez la mise en forme de base !

III) L'approche K-Nearest-Neighbors (KNN)

Maintenant, grâce au graphique précédent, on peut répondre très facilement aux questions :

- De quel type est un fruit de diamètre d=54 mm ?
- De quel type est un fruit de diamètre d=110 mm ?

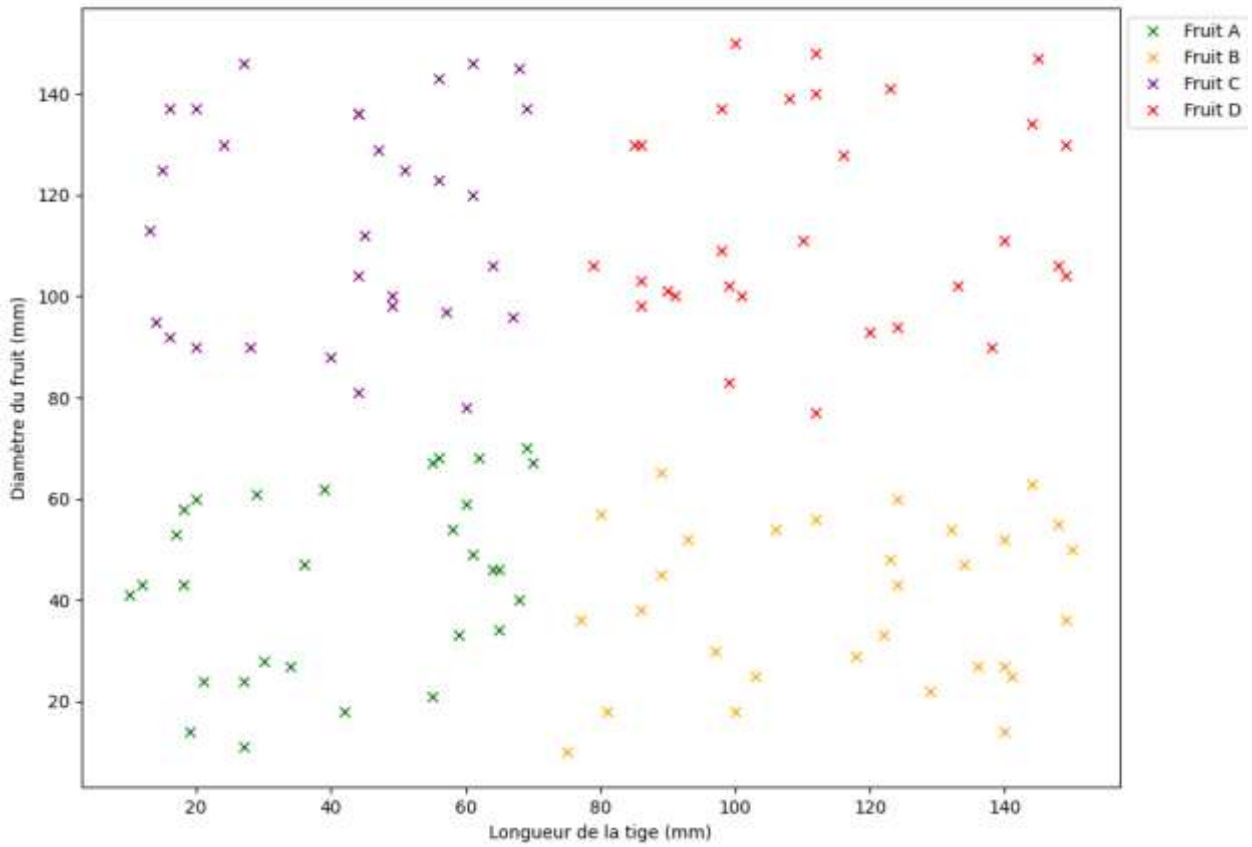
En effet, il suffit de « lire » le graphique et la réponse est évidente, surtout si le diamètre ne se trouve pas dans une zone « litigieuse ».



Intuitivement, vous avez comparé cet élément inconnu avec des éléments connus, et pas n'importe lesquels : les plus proches. Vous en avez ensuite déduit le type du fruit inconnu.
En fait vous avez considéré le **nombre de voisins** de ce nouveau fruit dont on cherchait le type, comme le montre la figure suivante.

Cependant, nous sommes dans un cas très (trop) simple : le type du fruit ne dépend que d'une seule valeur, son diamètre. Et nous n'avons que 2 types de fruits.

Prenons un cas de données étiquetées comportant un autre **paramètre**, la longueur de la tige, et considérons des fruits pouvant être désormais de 4 sortes.
On va donc avoir une représentation en 2 dimensions, avec par exemple la longueur de la tige en abscisse, et le diamètre du fruit en ordonnée.
Vous allez utiliser le dataset **`fruits_abcd.csv`**. Je vous laisse le soin de compléter le fichier **`matplotlib_fruits_ABC.py`** pour obtenir le résultat suivant :



```
import csv

#importe la bibliothèque matplotlib et pyblot
import matplotlib.pyplot as plt

[redacted]
lecteur_csv = csv.reader(fichier_csv, delimiter=";")
cpt = 0

for ligne in lecteur_csv:
    print("ligne",cpt,":",ligne)
    if cpt>0:

        #on récupère tige et on le convertit en int
        [redacted]

        #on récupère le diamètre et on le convertit en int
        [redacted]

        #on récupère le type du fruit
        [redacted]

        #en fonction du type de fruit on définit
        #une couleur (connue par matplotlib !)
        if fruit == 'A':
            c = "green"
        elif fruit == 'B' :
            c = "orange"
        [redacted]
            c = "purple"
        else:
            [redacted]
        #on plot le point d à l'ordonnée y=0
        #avec la couleur c
        [redacted]

    cpt = cpt + 1
plt.xlabel("Longueur de la tige (mm)")
plt.ylabel("Diamètre du fruit (mm)")
plt.legend(['Fruit A', 'Fruit B', 'Fruit C', 'Fruit D'],bbox_to_anchor=(1,1))

#on affiche notre beau graphique
plt.show()
```

La première partie de ce TP touche à sa fin. Vous savez désormais utiliser la bibliothèque **CSV** de Python pour ouvrir un fichier au format .csv, lire les données qu'il contient, et les représenter graphiquement avec **Matplotlib** et **Pyplot**.

C'est indispensable pour aller plus loin et appliquer notre algorithme KNN à plusieurs **datasets**, réels ou fictifs, afin de faire des prédictions de classification !