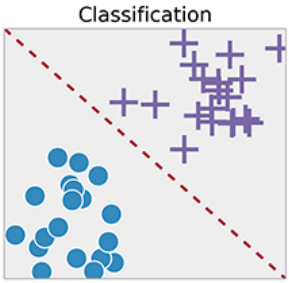
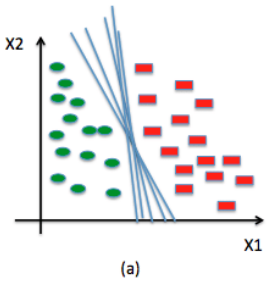
Cycle 8: Mettre en œuvre une démarche de résolution numérique

Chapitre 3 : Apprentissage SUPERVISE – la CLASSIFICATION





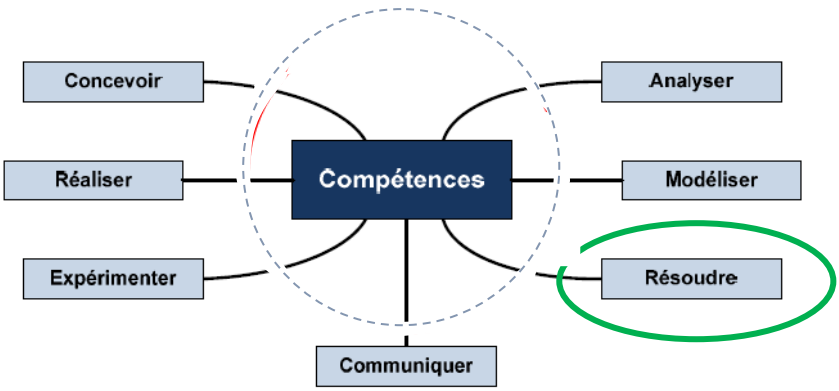
Problématique

Qu'est-ce que l'apprentissage supervisé ?
Quels sont les principes et techniques de classification

Savoir

C. Résoudre:

- C3 : mettre en œuvre une démarche de résolution numérique

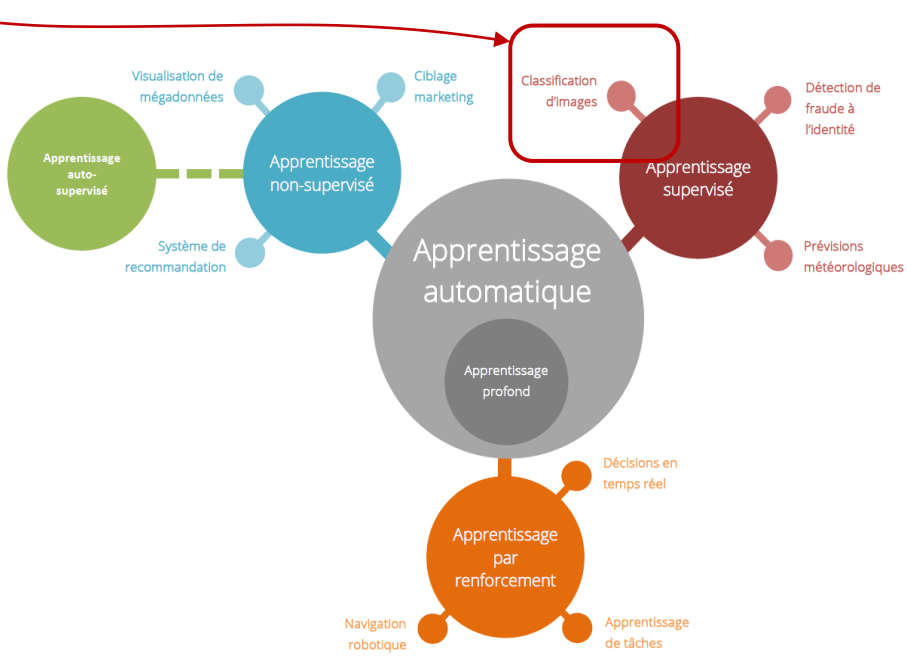


Apprentissage supervisé – la classification

1. Situation de la classification en IA

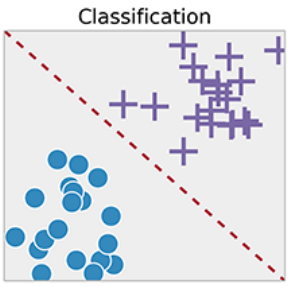
L'IA repose sur une phase très importante pour que la machine devienne intelligente, celle de l'**APPRENTISSAGE** basée sur **un très grand nombre d'exemples**. C'est ce que l'on appelle le **machine learning.**

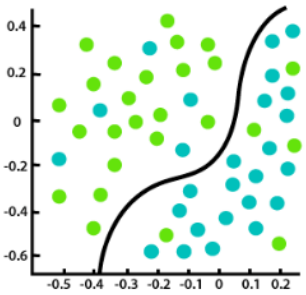
Au sein du machine learning, en apprentissage SUPERVISE, on retrouve les algorithme de régressions vus précédemment et la **CLASSIFICATION.**



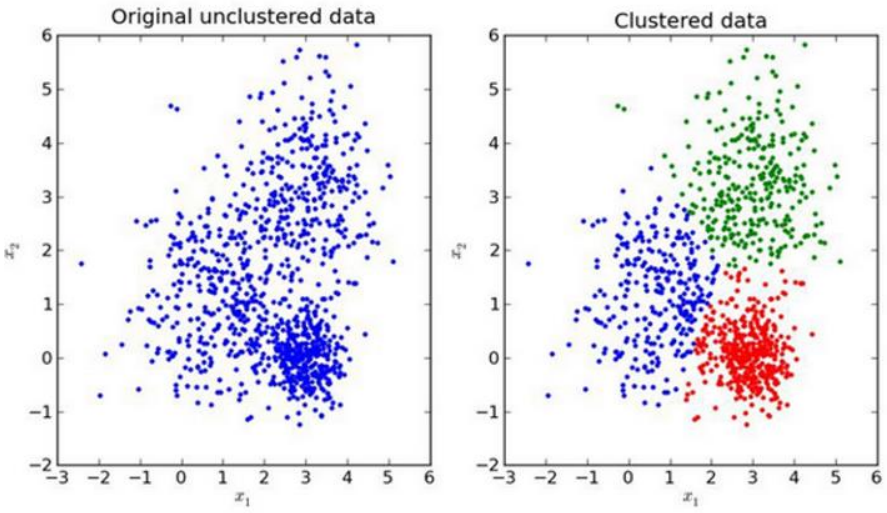
2. Principes de la classification en apprentissage supervisé

Le classement automatique ou classification supervisée est la **catégorisation algorithmique d'objets.** Elle consiste à attribuer une classe ou catégorie à chaque objet (ou individu) à classer, en se fondant sur des données statistiques.
La classification est très largement utilisée en **reconnaissance de formes.**





Apprentissage supervisé – la classification



Clustering

Il ne faut pas confondre la classification avec le **clustering (méthode d'apprentissage non supervisé)**. On donne à l'ordinateur un ensemble de données et à l'issue de la phase d'apprentissage il doit être capable de les classer dans des groupes.

Ex : un exemple d'apprentissage non supervisé est ce qui est fait dans google news. En effet plusieurs articles de presse arrivent tous les matins chez google et ils sont classés automatiquement dans des catégories : sport, économie, culture, politique en fonction du sujet traité par l'article.

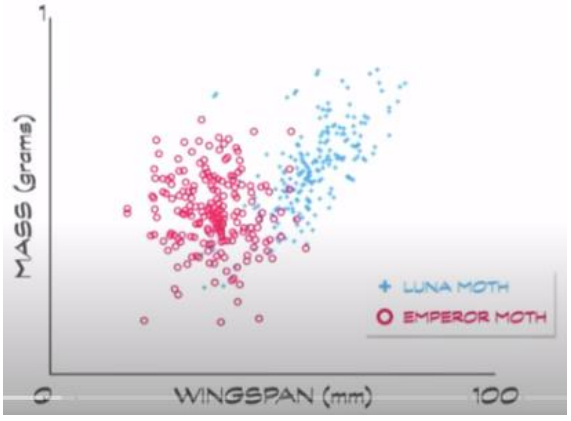
3. Première approche de la classification

Objectif : si on rencontre un nouveau papillon, est-il dit lunaire ou empereur ?

On va partir d'une **première base d'apprentissage** ou on classe des papillons selon leur poids et tailles des ailes pour savoir s'ils sont empereurs ou lunaire. On a un ensemble de plus de 1000 données.

LABELED DATA

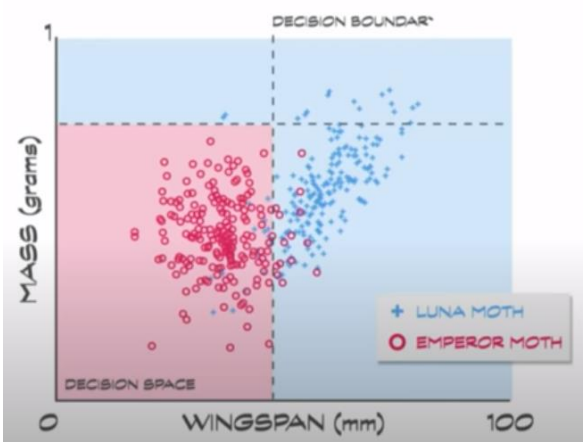
MASS	WINGSPAN	MOTH SPECIES (LABEL)
2.5	41	EMPEROR
3.7	36	EMPEROR
4.2	72	LUNA
3.2	27	EMPEROR
5.5	60	LUNA
5.1	51	EMPEROR
4.0	43	EMPEROR
3.6	40	EMPEROR
8.2	75	LUNA
7.6	82	LUNA
...	...	...



Et la première chose que l'on peut faire sous python, c'est tracer ces données pour les visualiser.

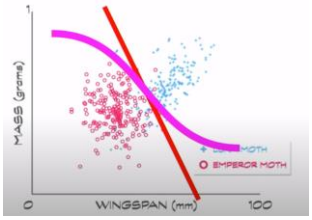
Première remarque, si on découpe l'espace, on remarque qu'à partir d'une certains taille d'aile on a plutôt tendance à avoir des papillons lunaires….Idem pour un certains poids…

donc les papillons empereurs sont plutôt en bas gauche du tracé. Sur cette observation, je peux faire un **premier code python avec une instruction IF et je teste cela sur l'ensemble des données de départ**. J'observe 80% de succés mais c'est **pas très précis…**




```
if x.wingspan<40 and x.mass<0.8:
    return 'imperial'
else:
    return 'luna'
```

On peut optimiser la frontière par une droite affine ou même un polynôme…mais on Retombe dans les méthodes de régressions que l'on a déjà vue.

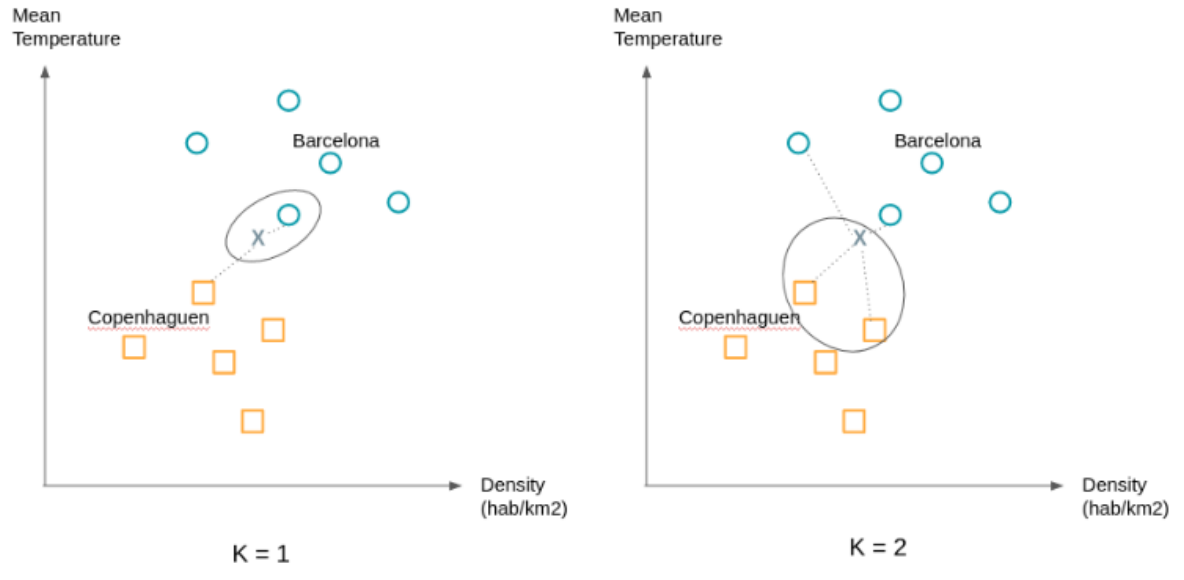


Alors comment faire pour classifier avec une autre approche ?

3. Les K plus proches voisins

C'est très simple, on **affecte à une observation, la classe de ses K plus proches voisins.** Seulement comme l'exemple suivant le montre, le choix de K peut changer beaucoup de choses !!

On cherchera donc à **essayer différentes valeurs de K pour obtenir la séparation la plus satisfaisante**.



K nearest neighbours

Apprentissage supervisé – la classification

Comment déterminer les K plus proches voisins ?

Il faut pour chaque nouvelle observation, calculer la **distance euclidienne** entre celle-ci et celles du data set.

Définition : *distance euclidienne*
Soient 2 points $X=(x_1,\ldots,x_p)$ et $C=(c_1,\ldots,c_p)$ dans un espace de dimension p, la distance euclidienne d est définie par :

$$d = \sqrt{\sum_{i=1}^{p}(x_i - c_i)^2}$$

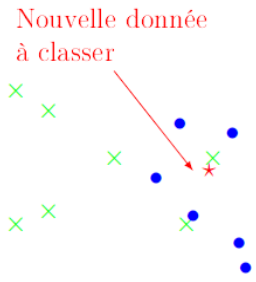
L'algorithme des **K plus proches voisins** est un algorithme de classification en apprentissage supervisé (il peut être facilement généralisable à un problème de régression). On se donne un ensemble de N observations étiquetées. Pour chaque nouvelle observation, il s'agit de déterminer l'étiquette de la nouvelle entrée comme l'étiquette majoritaire des K plus proches voisins de la nouvelle observation.

Algorithme : Algorithme des K plus proches voisins

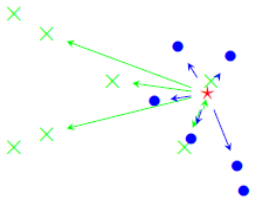
Données : X : Matrice d'observations taille $(N \times (p+1))$; `/* la dernière colonne correspond à l'étiquette */`
K : nombre de voisins à considérer ;
Xs : observations dont on cherche l'étiquette ; taille $(1 \times p)$
Résultat : E : Étiquette de l'observation Xs
Pour $i = 1$ **à** N **Faire**
  Calculer la distance entre Xs et Xi : d_{si}
E ← Étiquette majoritaire parmi les K plus petites distances d_{si}

Visualisation de la méthode des K voisins

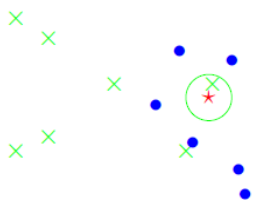
Dans le cas où les observations n'ont que deux caractéristiques (plus l'étiquette), il est classique de représenter l'algorithme comme dans les figures ci-dessous :



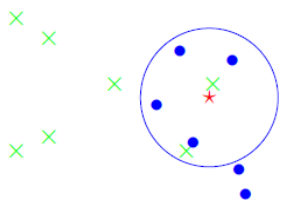
Observations initiales



Calcul des distances d_{SI}



Classement pour $K = 1$



Classement pour $K = 6$

On voit l'importance du choix de K !!

Exemple de classification par les K voisins … Iris de Fisher

Les données :

Nous avons choisi ici de nous baser sur le jeu de données ”iris de Fisher”. Ce jeu de données est composé de **50 entrées**, pour chaque entrée nous avons :

- la longueur des sépales (en cm)
- la largeur des sépales (en cm)
- la longueur des pétales (en cm)
- la largeur des pétales (en cm)
- l’espèce d’iris : Iris setosa, Iris virginica ou Iris versicolor (label)

Vous trouverez sur **mon site le dataset des iris en .csv.**

Visualisation des données :

Nous allons utiliser 3 bibliothèques Python :

- Pandas [3] qui va nous permettre d’importer les données issues du fichier csv
- Matplotlib [4] qui va nous permettre de visualiser les données (tracer des graphiques)
- Scikit-learn [5] qui propose une implémentation de l’algorithme des k plus proches voisins.

Ces bibliothèques sont facilement installables notamment en utilisant la distribution Anaconda.

Il est possible d’écrire un programme permettant de visualiser les données sous **forme de graphique** *(abscisse : ”petal_length”, ordonnée : ”petal_width”).*

```
import pandas
import matplotlib.pyplot as plt
iris=pandas.read_csv("iris.csv")
x=iris.loc[:,"petal_length"]
y=iris.loc[:,"petal_width"]
lab=iris.loc[:,"species"]
plt.scatter(x[lab == 0], y[lab == 0], color='g', label='setosa')
plt.scatter(x[lab == 1], y[lab == 1], color='r', label='virginica')
plt.scatter(x[lab == 2], y[lab == 2], color='b', label='versicolor')
plt.legend()
plt.show()
```

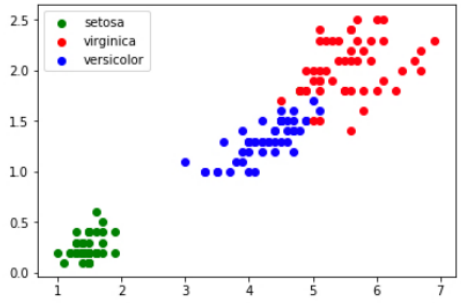


FIGURE 1 – Représentation graphique des données

Utilisation de l'algorithme des K plus proches voisins :

Le graphique ci-dessus (figure 1) montre que les 3 classes (Iris setosa, Iris virginica et Iris versicolor) sont relativement bien séparées. On peut alors ajouter une donnée non labellisée n'appartenant pas à l'ensemble d'origine (voir figure 2). On voit que cette donnée est plutôt une setosa.

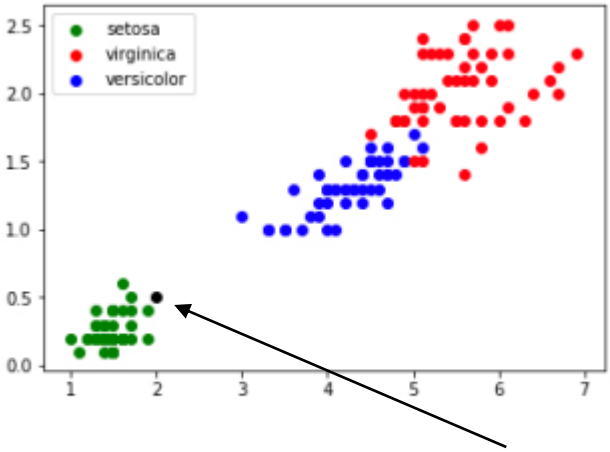


FIGURE 2 – Ajout d'une donnée non labellisée

Dans certains cas (exemple : largeur pétale = 0,75 cm ; longueur pétale = 2,5 cm) il est un peu plus difficile de se prononcer "au premier coup d'oeil" (voir figure 3) :

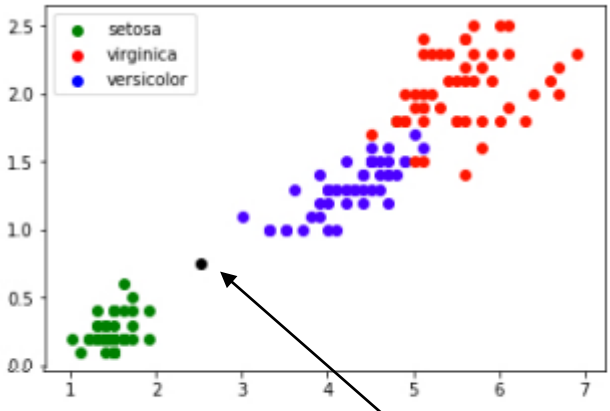


FIGURE 3 – Cas plus difficile...

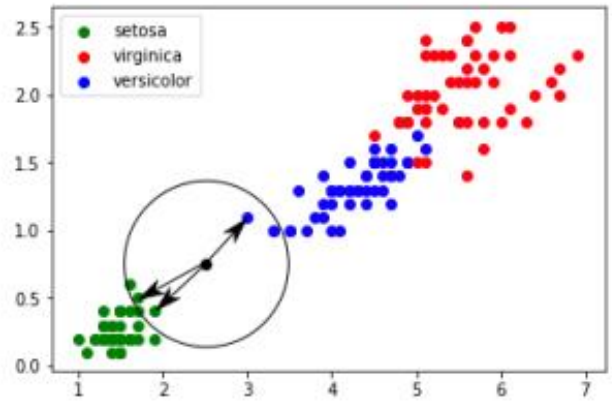
À partir de l'exemple ci-dessus (voir figure 3), la solution proposée est l'algorithme des k plus proches voisins :

▹ on calcule la distance entre notre point (largeur du pétale = 0,75 cm ; longueur du pétale = 2,5 cm) et chaque point issu du jeu de données "iris" (à chaque fois c'est un calcul de distance entre 2 points = distance euclidienne)

▹ on sélectionne uniquement les k distances les plus petites (les k plus proches voisins)

▹ parmi les k plus proches voisins, on détermine quelle est l'espèce majoritaire. On associe à notre "iris mystère" cette "espèce majoritaire parmi les k plus proches voisins"

Dans l'exemple évoqué ci-dessus (largeur pétale = 0,75 cm ; longueur pétale = 2,5 cm), pour k=3, nous obtenons graphiquement :



Un iris ayant une largeur de pétale égale à 0,75 cm et une longueur de pétale égale à 2,5 cm a une "**forte" probabilité d'appartenir à l'espèce setosa.**

Utilisation de scikit-learn :

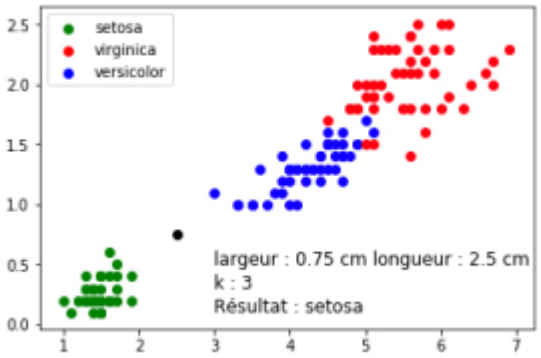
La bibliothèque Python scikit-learn propose un grand nombre d'algorithmes lié à l'apprentissage automatique (c'est sans aucun doute la bibliothèque la plus utilisée en apprentissage automatique). Parmi tous ces algorithmes, scikit-learn propose l'algorithme des k plus proches voisins.

Voici un programme Python permettant de résoudre le problème évoqué ci-dessus (largeur pétale = 0,75 cm ; longueur pétale = 2,5 cm) :

```
import pandas
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier

#traitement CSV
iris=pandas.read_csv("iris.csv")
x=iris.loc[:,"petal_length"]
y=iris.loc[:,"petal_width"]
lab=iris.loc[:,"species"]
#fin traitement CSV

#valeurs
longueur=2.5
largeur=0.75
k=3
#fin valeurs

#graphique
plt.scatter(x[lab == 0], y[lab == 0], color='g', label='setosa')
plt.scatter(x[lab == 1], y[lab == 1], color='r', label='virginica')
plt.scatter(x[lab == 2], y[lab == 2], color='b', label='versicolor')
plt.scatter(longueur, largeur, color='k')
plt.legend()
#fin graphique
```

```
#algo knn
d=list(zip(x,y))
model = KNeighborsClassifier(n_neighbors=k)
model.fit(d,lab)
prediction= model.predict([[longueur,largeur]])
#fin algo knn

#Affichage résultats
txt="Résultat : "
if prediction[0]==0:
  txt=txt+"setosa"
if prediction[0]==1:
  txt=txt+"virginica"
if prediction[0]==2:
  txt=txt+"versicolor"
plt.text(3,0.5, f"largeur : {largeur} cm longueur : {longueur} cm", fontsize=12)
plt.text(3,0.3, f"k : {k}", fontsize=12)
plt.text(3,0.1, txt, fontsize=12)
#fin affichage résultats

plt.show()
```

Nous obtenons le résultat suivant :



Il est ensuite possible de modifier le programme ci-dessus afin **d'étudier les changements induits par la modification du paramètre k (notamment pour k=5)** en gardant toujours les mêmes valeurs de largeur et de longueur (largeur pétale = 0,75 cm ; longueur pétale = 2,5 cm). Il est aussi possible de travailler avec d'autres valeurs de longueur et largeur.

Vous pouvez récupérer le code python et le fichier iris.csv sur mon site ou par le raccourcit sur mon drive ci-dessous.

https://drive.google.com/drive/folders/1RApSM-9NSNfhJ5ZuBRT2p4s5YeSVfhlS?usp=sharing