Intelligence artificielle

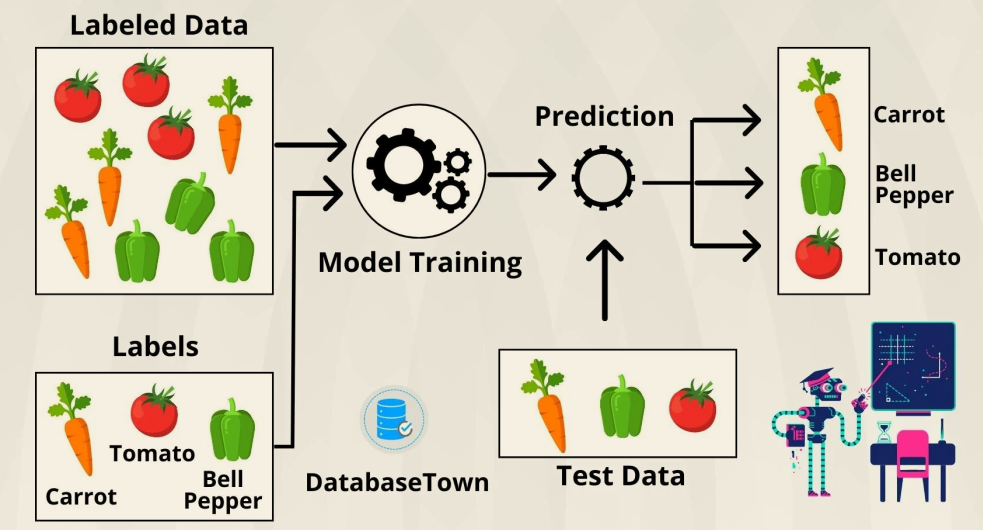
Cours

[ ] qu'on appelle aussi [ ]

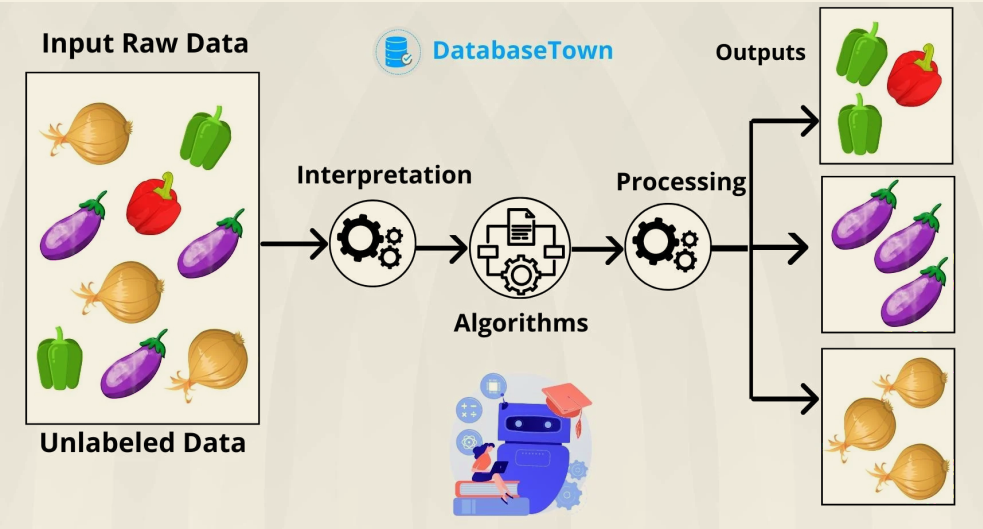
est un domaine d'étude de l'intelligence artificielle qui vise à donner aux machines la capacité d'apprendre. Cette technologie très puissante a permis le développement des voitures autonomes, de la reconnaissance vocale, et de tous les systèmes dits intelligents depuis le début du siècle.

Il existe deux principales méthodes d'apprentissage :

❑ **L'apprentissage supervisé** exploite des bases de données d'entraînement qui contiennent des labels ou des données contenant les réponses aux questions que l'on se pose.



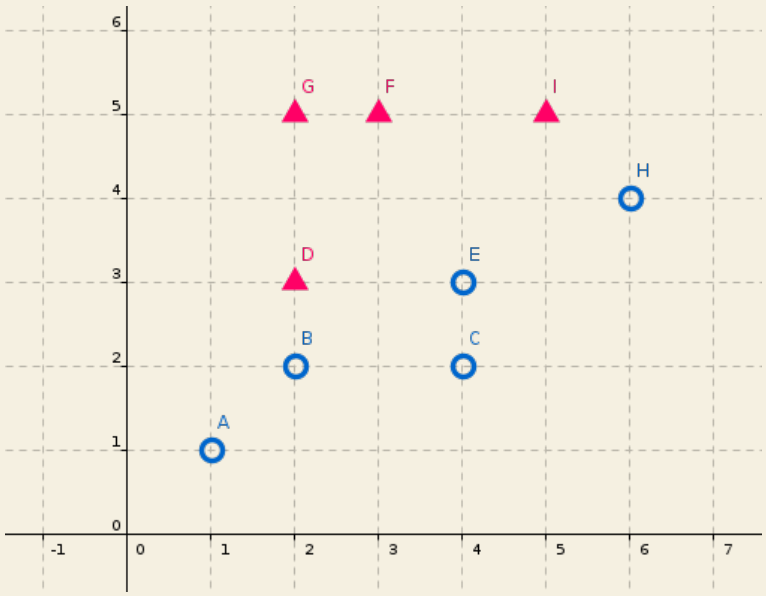
❑ **L'apprentissage non supervisé** est une branche du machine learning, caractérisée par l'analyse et le regroupement de données non-étiquetées. Pour cela, ces algorithmes apprennent à trouver des schémas ou des groupes dans les données, avec très peu d'intervention humaine.



D.W.

Exercice 1 :

On donne des points dans un repère orthonormé, chacun étant associé à une classe : **Triangle** ou **Cercle**.



1. Proposer une liste python `donnees` où chaque élément est un tuple de la forme

`(coordonnées, classe)`

où coordonnées est un tuple faisant référence aux coordonnées du point.

La liste commence par exemple par `donnees = [((1, 1), "Cercle"), ...]`.

2. a. Écrire une fonction `distance(p1:tuple, p2:tuple) -> float` qui renvoie la distance euclidienne entre les points `p1` et `p2`.

b. En déduire une implantation d'une fonction `distance_donnees(p:tuple, L:list) -> list` qui détermine la distance entre un point `p` et chacun des points contenus dans le jeu de données `L`.

La fonction renverra une liste où les éléments sont des tuples de la forme `(distance, classe)`.

3. Écrire une fonction `selection(L:list) -> None` qui réalise un tri par sélection d'une liste `L` formée d'éléments de la forme `(distance, classe)` afin d'obtenir une liste triée par ordre croissant de distance.

4. Écrire une fonction `knn(L:list, n:int) -> dict` qui à une liste d'éléments de la forme `(distance, classe)` renvoie le nombre d'apparition de chacune des classes dans les `n` éléments les plus proches.

Un résultat possible à l'instruction `knn(donnees, 5)` serait `{"Triangle" : 3, "Cercle" : 2}`

5. a. Écrire une fonction `max_classe(d:dict) -> str` qui renvoie la clé d'un dictionnaire associée à la plus grande valeur.

Par exemple, avec le dictionnaire `{"Triangle" : 3, "Cercle" : 2}`, on aurait `"Triangle"`.

b. Écrire une fonction `prediction(p:tuple, L:list, n:int) -> str` qui renvoie la classe probable à laquelle appartient le point `p` à l'aide du jeu de données contenu dans la liste `L`.

On choisira pour cela, la classe la plus présente dans le dictionnaire obtenu avec la fonction `knn`.

Cours

L'algorithme des k plus proche voisins est un algorithme de classification à partir d'un jeu de données étiquetées. Il procède en trois étapes :

- ❏ Calcul des distances entre chaque point du jeu de données et la donnée test (donnée dont on veut déterminer l'appartenance à une classe),
- ❏ Tri des données en fonction de la distance à la donnée test,
- ❏ Détermination de la classe la plus représentée parmi les k données les plus proches.

La **distance** est à choisir en fonction des données que l'on souhaite classifier.

Exercice 2 :

On a répertorié la taille et le poids pour différents individus. On a obtenu les tableaux suivants :

Pour les hommes

Taille	170	189	175	164	175	184	178	179	182	174	172	185	178	180	189
Poids	60	72	64	72	61	78	68	79	74	62	74	80	73	70	72

Pour les femmes

Taille	169	172	174	168	161	162	160	165	158	164	158	163	170	172	174
Poids	57	51	55	50	50	48	52	53	51	53	49	50	53	70	62

1. Quelle variable `donnees` peut-on choisir pour constituer un jeu de données étiquetées ?

2. a. Écrire une fonction `donnees_par_classe(jeu:list, c:str) -> list` qui renvoie la liste de données issues de `jeu` en fonction de la classe d'appartenance de ces données via l'argument `c`.

Par exemple, `donnees_par_classe(donnees, 'H')` renverrait la liste des hommes.

b. On propose la fonction suivante :

```
def tracer_donnees(L : list) -> None:
    """   Trace dans un repère les éléments du jeu de données étiquetées L. """
    x = [el[0][0] for el in L]
    y = [el[0][1] for el in L]
    plt.scatter(x, y, label=L[0][1])
```

Tester alors le script suivant :

```
... #code précédent

import matplotlib.pyplot as plt

tracer_donnees(donnees_par_classe(donnees, 'H'))
tracer_donnees(donnees_par_classe(donnees, 'F'))
plt.legend()
plt.show()
```

c. Placez-vous sur le graphique en insérant l'instruction :

```
plt.scatter(..., ..., color='black', marker='P', label='Moi')
```

et vérifier à l'aide de knn s'il y a cohérence entre votre morphologie et votre genre.

Exercice 3 :

On a récupéré un jeu de données lié aux joueurs du Top 14.
Vous pouvez télécharger ces données sur **https://dwmaths.github.io**.
Ce fichier est organisé suivant le modèle ci-dessous :

Équipe	Nom	Poste	Type	Naissance	Taille	Poids

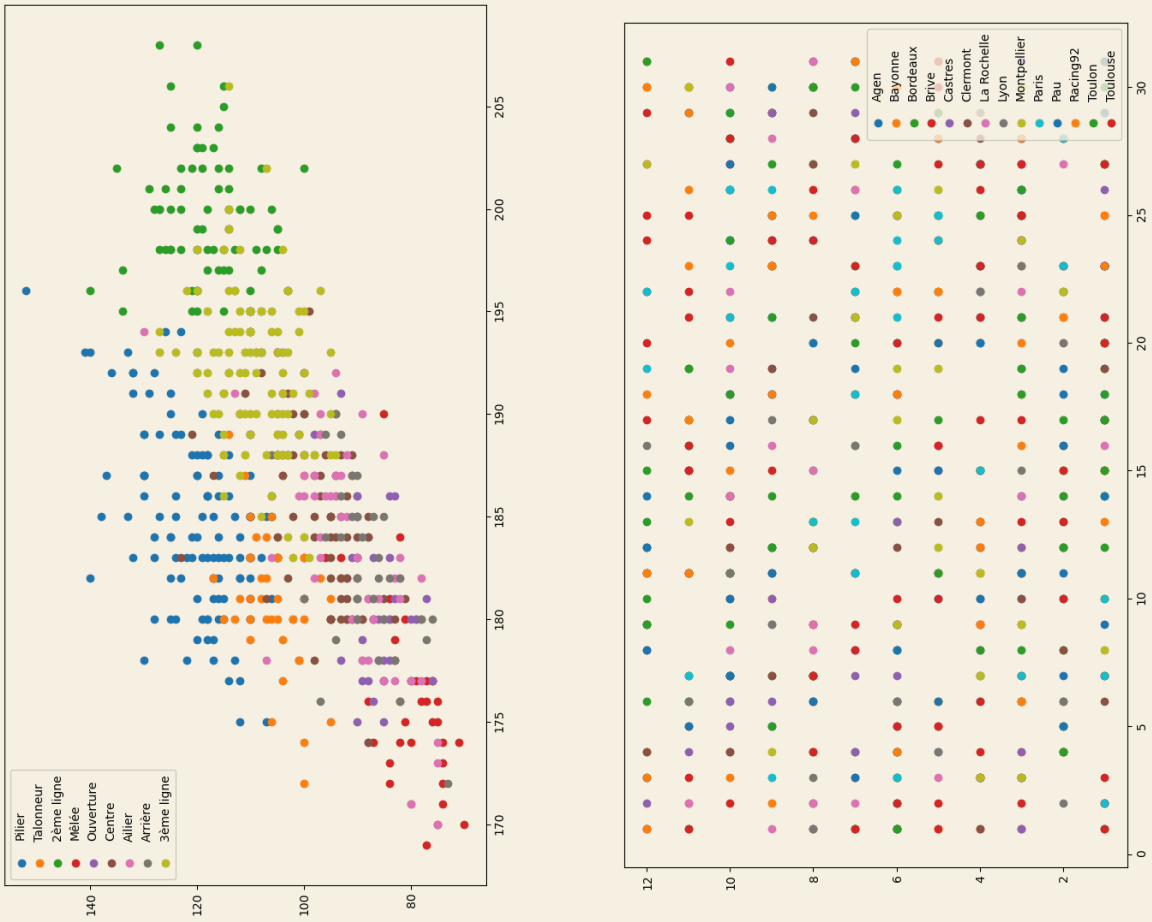
Pour récupérer l'ensemble des joueurs dans une liste, on écrit le code :

```
f = open("JoueursTop14.csv", "r")
buff = f.readlines()
f.close()
joueurs = [buff[k][:-1].split(';') for k in range(1, len(buff))]
```

1. En calquant sur l'exercice précédent, représenter les graphiques associés aux jeux de données suivants :

- ❏ `((taille, poids), poste)`
- ❏ `((jour de naissance, mois de naissance), équipe)`

Vous devriez obtenir des graphiques semblables à :



1. Sur quel poste devriez-vous jouer suivant votre morphologie ?

2. Dans quelle équipe devriez-vous jouer suivant votre date de naissance ?