II. Algorithme des k plus proches voisins

1) Introduction : un problème de classification

Nous allons à présent travailler sur un autre algorithme complexe. Cet algorithme d'apprentissage automatique est souvent appelé (même en bon français), **algorithme de machine learning**. L'idée est d'utiliser un grand nombre de données afin "d'apprendre à la machine" à résoudre un certain type de problème. Nous travaillerons sur un exemple dans ce chapitre pour fixer les idées.

Cette idée d'apprentissage automatique n'a pas attendu l'informatique et a été mise au point par Edgar Anderson en 1936. Celui-ci avait collecté 3 espèces d'iris : "iris setosa", "iris virginica" et "iris versicolor" et avait mesuré plusieurs de leurs caractéristiques (largeur et longueur des pétales principalement). Il se demandait alors si connaissant uniquement la longueur et la largeur des pétales, il était possible de connaitre l'espèce d'iris.



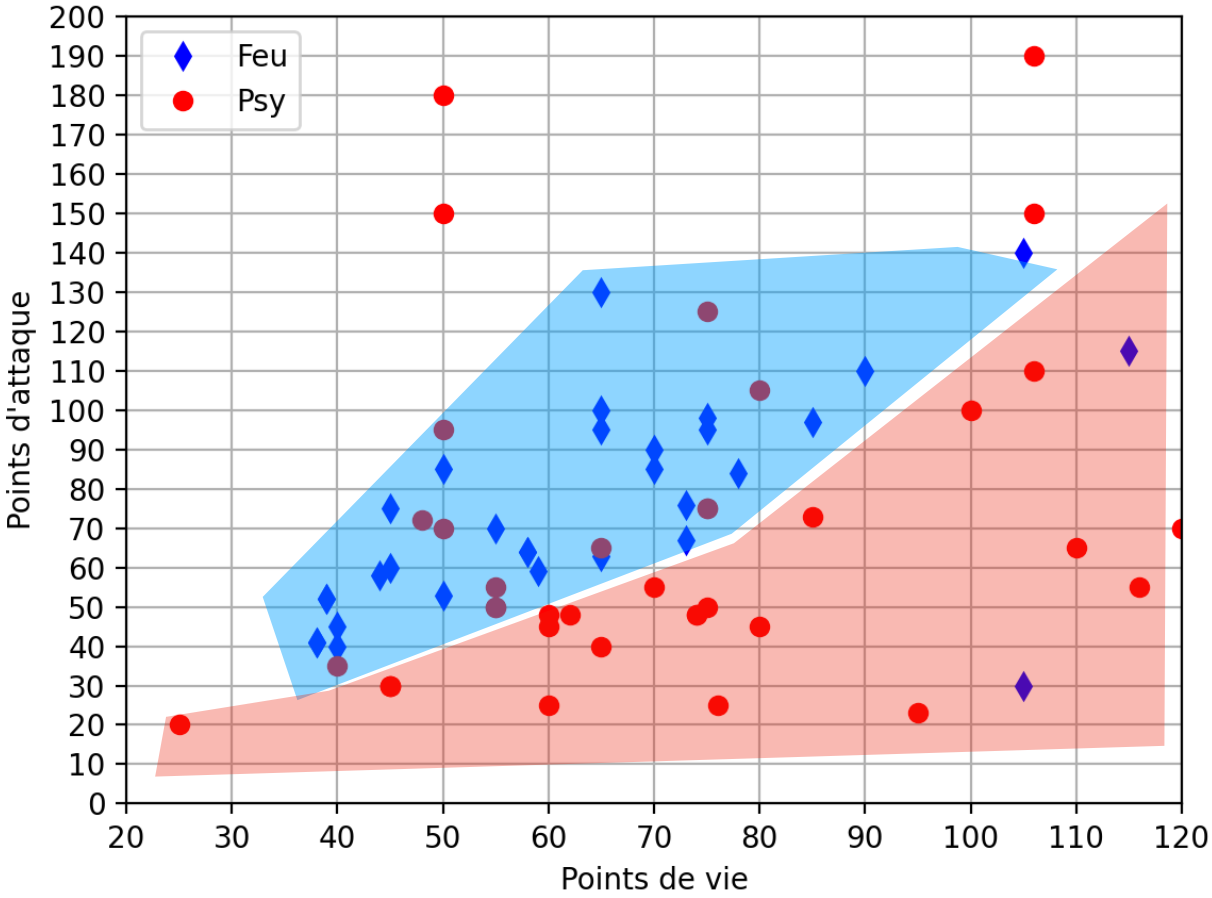
De nos jours, le machine learning est très à la mode car, il est devenu tout algorithme de machine learning avancé est basé sur la qualité et la quantité des données recueillis. (les données qui permettront à la machine d'apprendre à résoudre un problème). Il faut alors être capable de faire un premier tri afin d'organiser les données en catégories…

Nous allons étudier un algorithme d'apprentissage assez simple à appréhender : l'algorithme des "**k plus proches voisins**" (en anglais "**k-nearest neighbors**" : knn).

Dans cette partie, nous allons travailler à classifier des Pokémon et voir s'il est **possible de déterminer le type d'un pokémon connaissant ses caractéristiques** (points de vie et points d'attaque). Nous nous baserons sur le fichier pokemons.csv dont un aperçu est disponible ci-dessous :

Soporifik	60	48	45	42	Psy
Hypnomade	85	73	70	67	Psy
Mewtwo	106	110	90	130	Psy
Magmar	65	95	57	93	Feu
Pyroli	65	130	60	65	Feu

Si on trace tous nos Pokemon Psy et Feu dans un graphique, on obtient le graphique ci-dessous :



À partir de cet échantillon, on veut prédire la classification d'un Pokemon mystère à partir de la donnée de ses points de vie et de sa valeur d'attaque.

❐ — À faire vous-même **7** —

- ❖ Trouvez au moins un argument permettant d'affirmer qu'une telle classification est possible.
- ❖ À partir de ces données, comment feriez-vous pour affirmer qu'un Pokemon est d'un type ou d'un autre ?

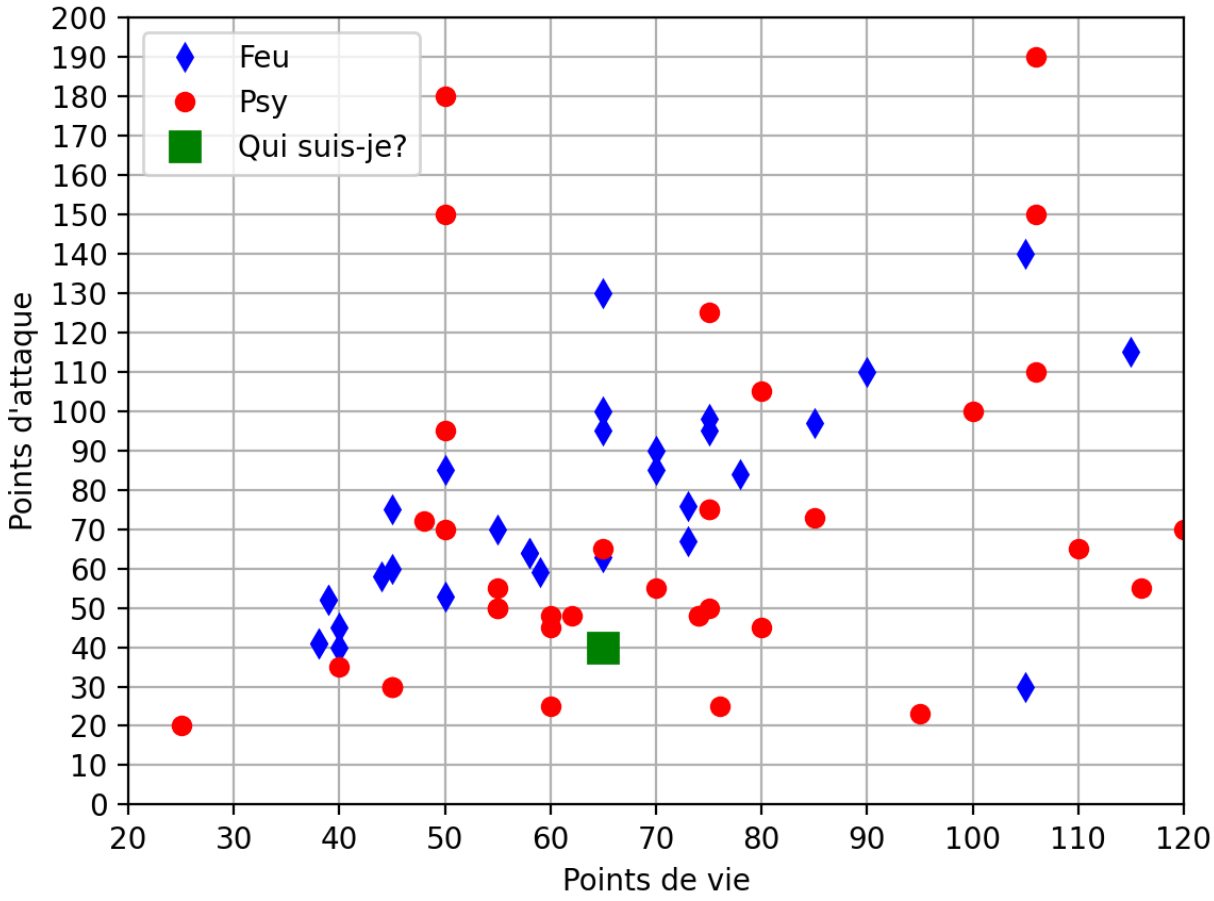
2) Algorithme de prédiction : un problème de classification

À partir des données représentées sur le diagramme, on veut prédire la classe d'un Pokémon qui a 65 points de vie et 40 en attaque. Plaçons le dans le graphique

❐ — À faire vous-même **8** —

- ❖ Placez-le dans le graphique et repérez à l'aide d'une règle graduée le type des 6 voisins les plus proches. Que pouvez-vous en déduire ?

❖ Que se passe-t-il si vous choisissez les 10 premiers voisins ? Les 20 premiers voisins ? tous les Pokemons de l'échantillon ?



Parmi les 6 premiers voisins, vous avez du trouver 5 Pokémons de type Psy et 1 Pokémon de type Feu. Il est fort probable que notre Pokémon mystère soit donc de type Psy.

❐ — À faire vous-même **9** —

❖ Faire de même pour un Pokémon avec 80 points de vie et 110 points d'attaque. Qu'en pensez-vous ?

Rem : La valeur k=6 est arbitraire. Cette valeur est appelée le **paramètre de l'algorithme** : elle doit être choisie judicieusement si l'on souhaite obtenir des résultats corrects. En effet, une valeur trop faible provoque un sous-échantillonnage (manque de fiabilité due à un nombre trop faible de données) alors qu'une valeur trop grande provoque un sur-échantillonnage (certaines données n'ont plus rien à voir avec le nouveau point).

Il faut donc trouver un juste milieu, **qui pourrait être adapté en fonction des circonstances !**

❐ — À faire vous-même **10** —

Vous allez maintenant formuler les trois grandes étapes de l'algorithme de prédiction d'un Pokémon parmi un échantillon connu de Pokémon.

- Quelles données avez-vous besoin de donner à votre algorithme pour obtenir une classification ?
- Écrivez en terme algorithmique simple les trois étapes permettant de déterminer la classification de notre Pokémon mystère.

3) Algorithme naïf

Nous allons nous intéresser ici à l'algorithme naïf qui permet déjà d'obtenir de bons résultas de manière assez rapide. Vous allez mettre en place l'algorithme et remplir le code Python à trous ci-dessous.

Vous disposerez des données suivantes :

- une table de données de taille n importée au format csv (utilisez csvReader, Chapitre 9)
- une donné cible (le pokémon mystère)
- un entier k plus petit que n
- une règle permettant de calculer la "distance" entre deux données

❐ — À faire vous-même **11** —

- Reprenez les trois grandes étapes écrites au #10 : réécrivez les en expliquant clairement les opérations effectuées sur la table de données dans ces 3 étapes
- Complétez l'algorithme de k plus proches voisins la fonction en Python page suivante. La fonction **distance calcule la distance euclidienne entre une donnée et la cible.**

```python
def kNN(table, cible, k):

    def distance(donnee, cible):
    # distance entre une donnee et la cible
        return

    def distanceCible(donnee):
    # distance entre une donnee et la cible, à utiliser dans sorted
        return distance(donnee, cible)

    # trie de la table selon le critere
    tableTriee = sorted(                  , key =              )

    # on regarde les k premiers voisins
    prochesVoisins = []
    for i in range(         ):
        prochesVoisins.append(                          )

    # on utilise un dictionnaire pour compter le nombre d'apparitions
    # d'un type ou d'un autre
    dicoType = {}









    # on renvoie le type de la cible
    return typeCible
```