5.1 Algorithme des K plus proches voisins

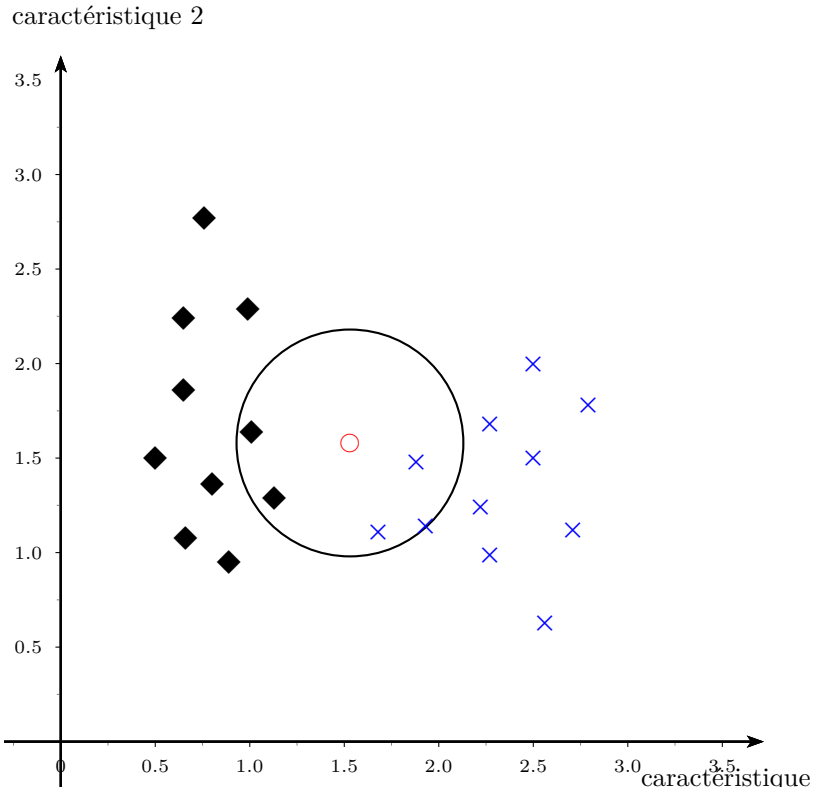
L'algorithme KNN (K Nearest Neighbors en anglais) est un algorithme essentiel en apprenstissage automatique ou «Machine Learning ».

a. Application 1 : classification

Comme son nom l'indique, cet algorithme permet de déterminer les K plus **proches** voisins pour ensuite, par exemple, classifier une donnée dans une catégorie ou une autre.

Considérons une image, celle d'un animal, on souhaite déterminer algorithmiquement s'il s'agit d'un chat ou d'un chien. Pour cela on dispose d'un grand nombre de données sur les chats et les chiens. On compare l'image de l'animal à ces données. On cherche parmi les échantillons les k plus proches voisins de l'image de l'animal. Si dans les k plus proches voisins, il y a plus de chats alors on affirme que c'en est un sinon on affirme que c'est un chien.

Graphiquement, on peut illustrer le principe comme suit. On a deux classes d'objets, les losanges et les croix, pour lesquels on possède les données chiffrées de deux caractéristiques.



On cherche à déterminer à quelle classe d'objet appartient le nouvel objet dont on connaît les deux caractéristiques. Ici on choisit arbitrairement $K = 5$ on cherche donc ses cinq plus proches voisins. Parmi les cinq plus proches voisins, trois sont de classe croix. On classe donc le nouvel objet comme étant une croix.

b. Application 2 : algorithme de suggestion

L'algorithme de suggestion consiste à suggérer à un utilisateur qui aurait sélectionné ou acheté un produit d'autres articles similaires.
La technique est simple : il suffit d'avoir des statistiques sur un grand nombre d'utilisateurs et de chercher parmi les statistiques les k plus proches voisins de l'utilisateur puis de lui proposer le comportement majoritaire chez ses k plus proches voisins.

c. Implémentation :

• On aura besoin d'écrire une fonction qui retourne une distance entre deux points. Il existe de nombreux types de distance, on parle parfois de métrique. Mais alors quelle distance choisir ? Cela dépend du problème et de ce que l'on désire obtenir comme solution.

La distance euclidienne dans un repère orthonormé peut faire l'affaire, on peut examiner ce que cela donne avec cette métrique. Mais attention elle ne donnera pas de bons résultats pour d'autres problèmes.

`distance_euclidienne(point1, point2)` avec `point1` et `point2` deux listes de nombres de même dimension n
Rappel la formule mathématique pour calculer la distance entre deux points dans un repère orthornormé est
$d = \sqrt{(x'_1 - x_1)^2 + (x'_2 - x_2)^2 + (x'_3 - x_3)^2 + \ldots + (x'_n - x_n)^2}$

```
def distance_euclidienne(point1, point2):
    distance_carre = 0
    for a, b in zip(point1, point2):
        distance_carre .....................
    return math.sqrt(distance_carre)
```

• On aura besoin d'une fonction qui parmi des `donnees` retourne les `k` plus proches voisins de `point`.
`knn(donnees, point, k)`. Proposer un algorithme en langage naturel pour cette fonction `knn`.
• Test et applications :
Ci-dessous on donne un tableau de statisiques : première colonne taille en inches et seconde colonne poids en pounds.
Déterminer les 3 plus proches voisins de `point` :

```
data = [
        [65.75, 112.99],
        [71.52, 136.49],
        [69.40, 153.03],
        [68.22, 142.34],
        [67.79, 144.30],
        [68.70, 123.30],
        [69.80, 141.49],
        [70.01, 136.46],
        [67.90, 112.37],
        [66.49, 127.45],
       ]

point = [70, 140]
```

▶ Exercice 1 Classer des couleurs

Une couleur est un triplet d'entiers compris entre 0 et 255. On souhaite avec la méthode des knn écrire un algorithme qui, lorsqu'on lui donne un triplet d'entiers, renvoie la couleur correspondante en français.

1. Pour cela il faut créer un jeu de données étiqueteés pour « nourrir » le modèle. Créer une liste de listes `data` avec un nombre suffisant de données (au moins 50) pour que l'algorithme « apprenne » à reconnaître les couleurs rouge, vert et bleu.

```
data_couleur = [
               [220, 54, 54, 'rouge'],
               [255,0,0, 'rouge'],
               [27,41,180, 'bleu'],
               [55,180,27, 'vert'],
               [180,61,27, 'rouge'],
               ...


               ]
```

2. Tester l'algorithme avec un jeu de tests suffisants.

d. Applications

▶ Exercice 2 algorithme de recommandations

Votre professeur, M. Kbida, s'inscrit pour la première sur un réseau social, pour cela il note de 0 à 10 son intéret pour les thèmes suivants : Politique, Sport, Lecture, Art, Cuisine, Voyage, Jeux Vidéo, Musique et Animaux.
Le réseau social possède les réponses à ce même questionnaire de plusieurs autres de ses membres dans le fichier `reseaux_sociaux.csv`.

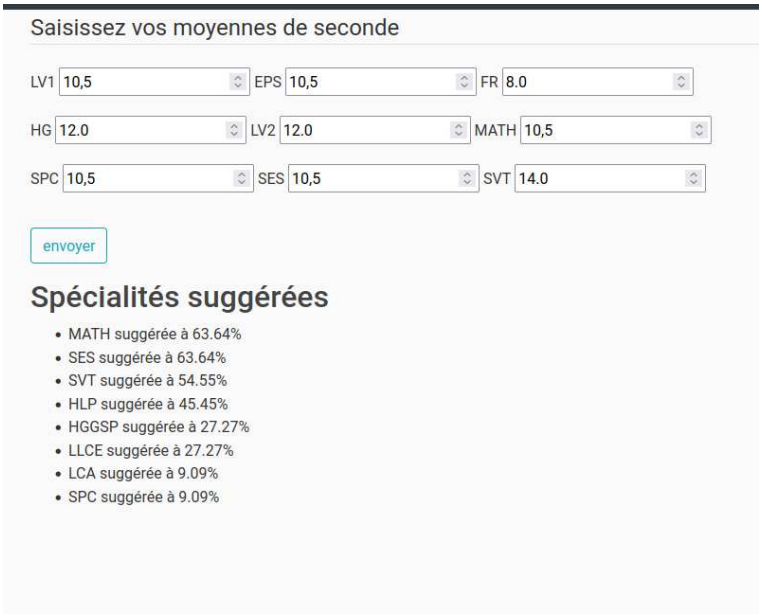
1. Écrire une fonction suggestion de contacts qui propose $k = 5$ profils proches à celui de M. Kbida. Sachant que son profil est `M_Kbida = [8, 5, 3,1,7,9,2,5,3]`

2. Que peut poser comme problèmes cette technique si elle est déployée à grande échelle ?

▶ Exercice 3 algorithme de recommandations

Les élèves de seconde se demandent souvent quels enseignements de spécialités ils devraient choisir. On dispose des moyennes des élèves de premières lorsqu'ils étaient en classe de seconde ainsi que les spécialités qu'ils ont choisies `spe_premiere.csv`.

1. Écrire une fonction python qui permet à un élève de seconde de saisir ses moyennes en seconde et qui va lui proposer les choix de spécialités les plus fréquemment pris par ses $k = 7$ plus proches voisins.



2. Quels seraient, à votre avis, les limites et inconvénients d'une telle méthode d'aide à la décision ?