1 Objectif

Implémenter l'algorithme des K plus proches voisins.

2 Description

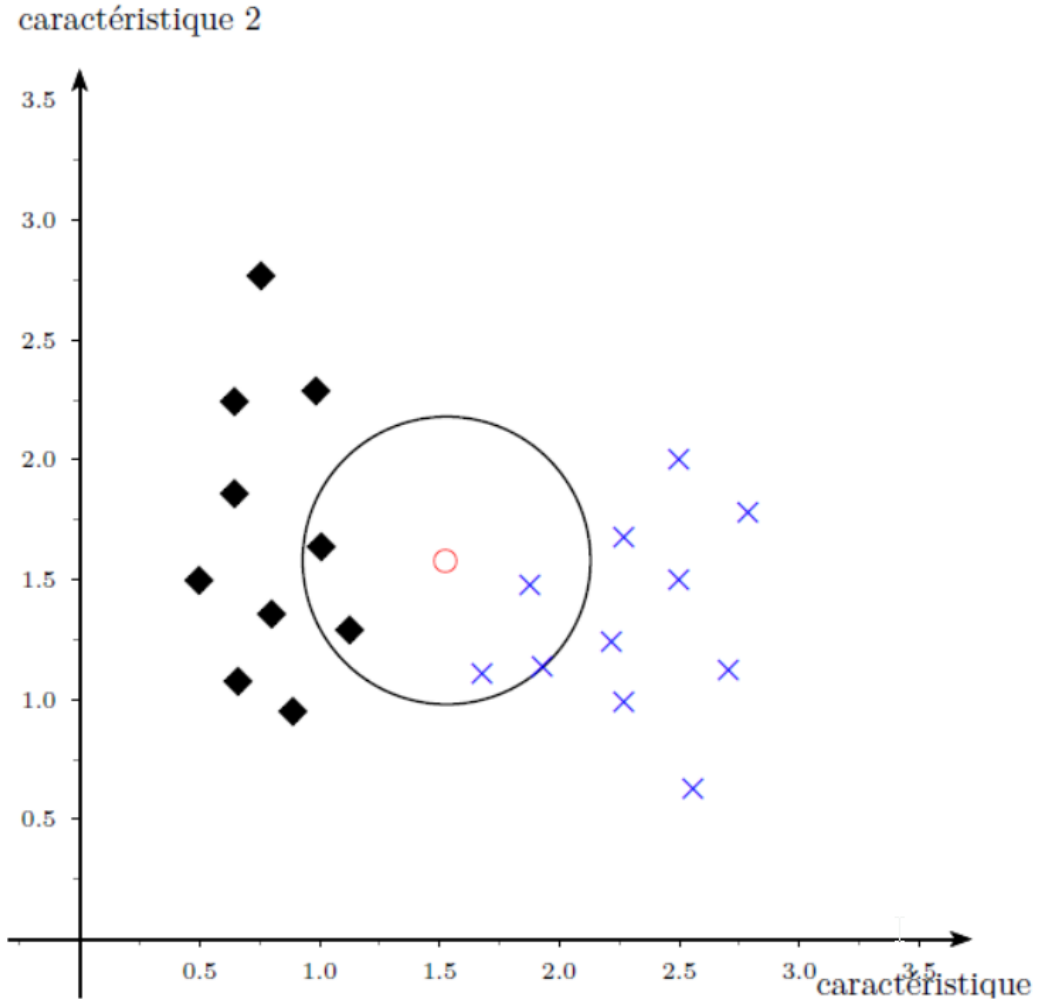
L'algorithme KNN (K Nearest Neighbors en anglais) est un algorithme essentiel en apprentissage automatique ou «Machine Learning ».

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 d'un animal on souhaite déterminer algorithmiquement si c'est un chat ou 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 aux données de chats et de chiens. On cherche parmi les échantillons les k plus proches voisins de l'image d'animal. Si dans les k plus proches voisins, il y a plus de chats alors on affirme que c'est un chat 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 des données chiffrées de deux caractéristiques.



On cherche à déterminer à quelle classe d'objet appartient le nouvel objet dont on connaît les deux caractéristiques. 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 croix.

3 Historique

Cet algorithme a été introduit en 1951 par Fix et Hodges dans un rapport de la faculté de médecine aéronautique de la US Air Force.

4 Exemple introductif

4.1 Présentation

On considère un jeu de données constitué de la façon suivante :

- les données sont réparties suivant deux types : le type 1 et le type 2,
- les données n'ont que deux caractéristiques : caractéristique 1 et caractéristique 2,

On imagine la situation suivante dans un jeu :

- Vous avez deux types de personnages : les fantassins (type 1 : "fantassin") et les chevaliers (type 2 : "chevalier").
- Vous avez deux types de caractéristiques : la force (caractéristique 1 : nombre entre 0 et 20) et le courage (Caractéristique 2 : nombre entre 0 et 20).
- Vous avez une collection de personnages dont vous connaissez les caractéristiques et le type.

On introduit un nouveau personnage dont on ne connait pas le type.

Vous possédez les caractéristiques de ce nouveau personnage.

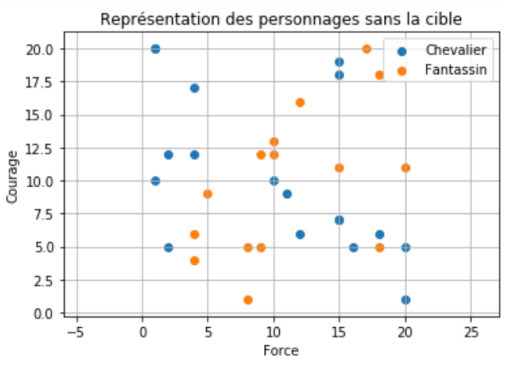
Le but de l'algorithme KNN (k Nearest Neighbors = k plus proches voisins en français) est de déterminer le type (type 1 ou type 2) de ce nouveau personnage.

4.2 Données de base

Les données sont stockées dans un fichier csv « **donnees.csv** ».

Aperçu des données :

Nom	Force	Courage	Type
Ario	20	1	Chevalier
Axal	10	10	Chevalier
Cargo	20	11	Fantassin
Clark	2	12	Chevalier
Fancy	9	5	Fantassin
Fanks	16	5	Chevalier
Faq	15	11	Fantassin
Fool	10	12	Fantassin
Helen	8	1	Fantassin
Karl	11	9	Chevalier
Korg	1	20	Chevalier
Lis	18	18	Fantassin
Lomo	17	20	Fantassin
louli	20	5	Chevalier
Louly	15	18	Chevalier
Loumi	4	12	Chevalier



5 Problème posé

5.1 Avant d'implémenter

On introduit une nouvelle donnée (appelée cible dans notre exemple) avec ses deux caractéristiques :

- une force de 12
- un courage de 12,5

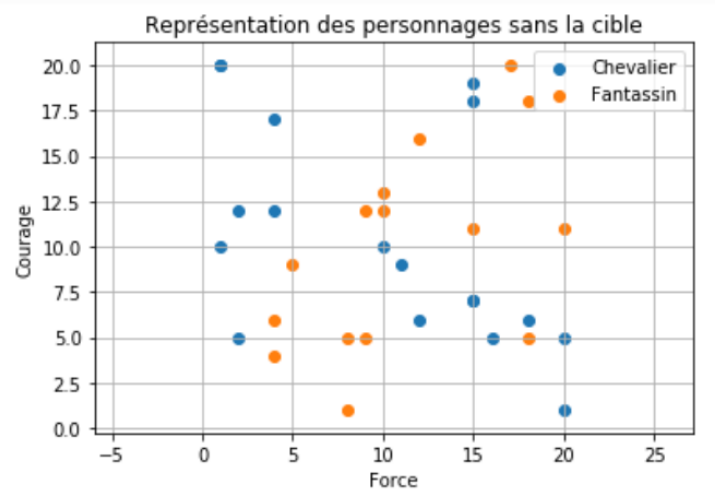
Le but de l'algorithme KNN des k plus proches voisins est de déterminer le type de cette nouvelle donnée.

Question 1. **Placer** sur le graphique ci-dessous la cible.

Question 2. On se fixe un nombre de voisins k=7, **tracer** sur le même graphique un cercle qui englobe les sept plus proches voisins. **Déterminer** le type de la cible.

Question 3. **Recommencer** la même opération avec un nombre de voisins égale à k=13. **Déterminer** le type de la cible.

Question 4. Le résultat est-il modifié ? Quelle est alors l'influence du choix du nombre de voisins ?



5.2 L'algorithme

Pour prédire la classe d'un nouvel élément, il faut des données :

- Un échantillon de données ;
- Un nouvel élément dont on connaît les caractéristiques et dont on veut prédire le type ;
- La valeur de k, le nombre de voisins étudiés.

Une fois ces données modélisées, nous pouvons formaliser l'algorithme de la façon suivante :

- Trouver, dans l'échantillon, les k plus proches voisins de l'élément à déterminer.
- Parmi ces proches voisins, trouver la classification majoritaire.
- Renvoyer la classification majoritaire comme type cherché de l'élément.

Ce qui nous donne l'algorithme naïf suivant :

- Données :
  - une table de données de taille n;
  - une donnée cible ;
  - un entier k plus petit que n;
  - une règle permettant de calculer la distance euclidienne entre deux données.
- Algorithme :
  - Trier les données de la table selon la distance croissante avec la donnée cible.
  - Créer la liste des k premières données de la table triée.
  - Renvoyer cette liste.

5.3 Implémentation

Question 5. **Ecrire** une fonction qui permet de calculer et de **retourner** la distance euclidienne entre deux points à partir d'une liste de données.
Nom de la fonction : distance()

Question 6. **Tester** votre fonction à partir d'un échantillon de données.

Question 7. **Ecrire** une fonction qui permet de déterminer et de **retourner** la liste des k plus proches voisins à partir de la distance euclidienne de données par rapport à la cible.
Nom de la fonction : knn()
Paramètres : donnees, cible, k
Valeur retournée : liste des k plus proches voisins triées par ordre croissant.

Question 8. **Tester** votre fonction à partir d'un échantillon de données.

Question 9. **Déterminer** le type de la cible.

Question 10. En utilisant matplotlib **tracer :**

- Le graphe des données
- La cible
- Les k plus proches voisins

Question 11. **Tester** l'ensemble.