ALGORITHMIQUE

Algorithme des k plus proches voisins

Le programme :

Algorithme des k plus proches voisins	*Écrire un algorithme qui prédit la classe d'un élément en fonction de la classe majoritaire de ses k plus proches voisins.*	*Il s'agit d'un exemple d'algorithme d'apprentissage.*

Pré-requis :

- Parcours séquentiel d'un tableau
- Indexation de tables
- Recherche dans une table

Partie 1 : Trouver un jeu de données

Sur le site http://www.allrugby.com, on a relevé la taille et le poids des différents joueurs du Top14 ainsi que leur poste sur le terrain au cours de la saison 2019-2020.

1. Travail à faire à la maison

Pour réviser ce qui a été fait dans un précédent chapitre, il faudra concevoir un programme en **python** qui :

- chargera le fichier JoueursTop14.csv (ce fichier est encodé en UTF-8 avec pour séparateur le point-virgule) ;
- extraira les données des joueurs de l'équipe de Toulouse (vous pouvez faire un autre choix...) ;
- construira un repère où chaque point représentera un joueur et sera positionné selon les critères de taille (en abscisse) et de poids (en ordonnée).

2. Synthèse du travail

a. Quelques compléments

On propose de repartir d'un fichier correction contenant trois fonctions :

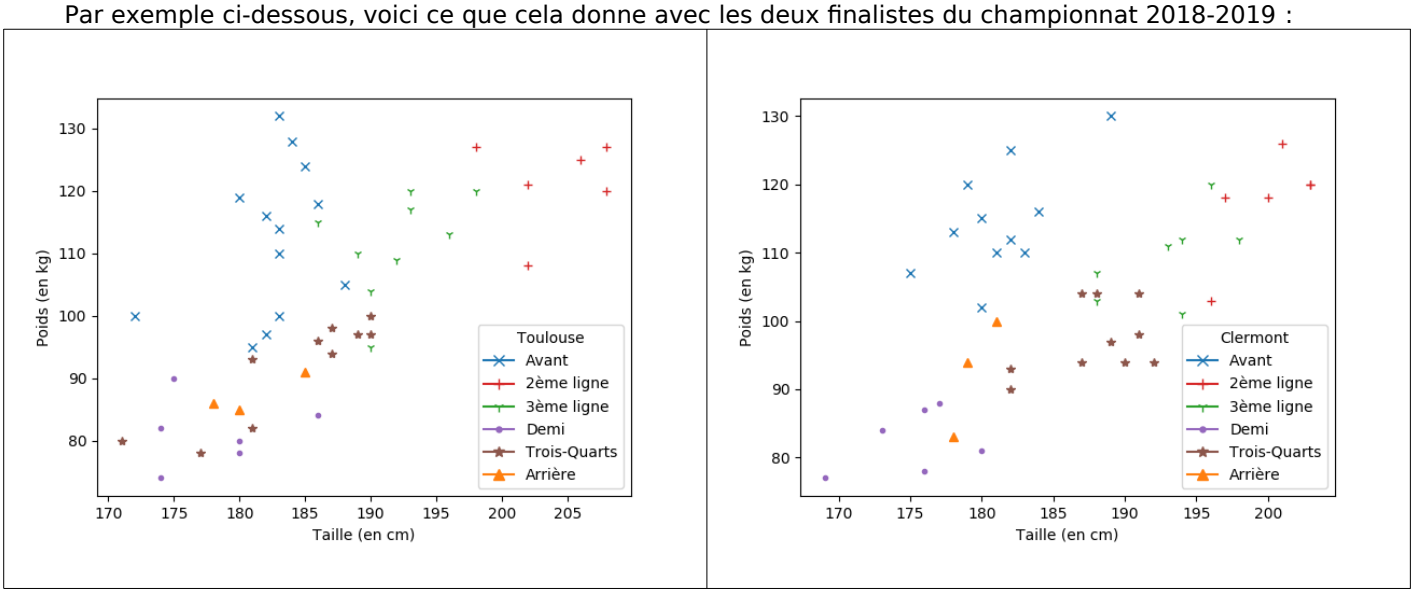
- extractionDonnees : charge le fichier csv et récupère les descripteurs (sous forme d'une liste) et l'ensemble des données de chaque joueur du top14 (sous forme d'une liste de listes) ;
- extraireEquipe qui ne sert qu'à extraire les données d'une équipe ;
- representation qui construit le repère. Chaque point représente un joueur selon ses caractéristiques de taille (en abscisse) et de poids (en ordonnées).

Les points sont d'une forme et d'une couleur différentes suivant le poste du joueur. On a choisi 6 catégories : les avants, les deuxièmes ligne, les troisièmes ligne, les demis (mêlée et ouverture), les trois-quarts et les arrières.

b. Objectif

Quelle que soit l'équipe choisie (y compris si on prend tous les joueurs), on se rend compte que la morphologie du joueur peut conditionner son poste sur le terrain.

Par exemple ci-dessous, voici ce que cela donne avec les deux finalistes du championnat 2018-2019 :



c. Quelques tests

Pour chacune de ces deux équipes, quel type de poste peut occuper un joueur :

- mesurant 1,95 m et pesant 112 kg ?
- mesurant 1,75 m et pesant 91 kg ?
- mesurant 1,80 m et pesant 90 kg ?

Partie 2 : Algorithme des k plus proches voisins (algorithme knn)

Définition

L'algorithme des k plus proches voisins (ou k-nearest neighbor) est une méthode d'apprentissage supervisé dédié à la classification de données.
On dispose d'une base de données d'apprentissage. À l'aide des similarités avec les exemples de cette base, l'algorithme doit classer les exemples non étiquetés.

Dans notre contexte, on souhaite déterminer quel type de poste peut prendre un joueur connaissant sa taille et son poids.

1. Le principe de l'algorithme en langage courant

On dispose d'un **ensemble de données**, d'une **fonction distance** et d'un **entier k**.
Chaque donnée contient deux types d'informations :

- deux données numériques destinées à la comparaison de deux éléments de l'ensemble ;
- un troisième critère destiné à la classification d'un élément.

a. Dans notre exemple des joueurs de rugby, que sont ces types d'informations ?

b. Fonction distance

Plusieurs fonctions distances existent dans un repère orthonormé du plan mais nous allons utiliser la distance euclidienne (vue en cours de mathématiques en 2nde).

Formule de la distance

Dans un repère orthonormé du plan, on définit les points $A(x_A ; y_A)$ et $B(x_B ; y_B)$, alors la longueur AB vaut $\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$

Construire cette fonction dans le langage python.

c. Le principe de l'algorithme

Un ensemble de n données est fourni, chacune de ces données contient deux données numériques $(x_i ; y_i)$ et une donnée supplémentaire correspondant à une classe c_i.

Un nouvel élément N étant donné avec ses valeurs numériques $(x_N ; y_N)$, on s'intéresse à la classe c_N inconnue à laquelle il pourrait appartenir. Voici les différentes étapes permettant d'apporter une réponse :

- Parcourir l'ensemble des n données et calculer les distances entre $(x_i ; y_i)$ et $(x_N ; y_N)$ pour chaque *i* ;
- Extraire de la liste des données les k plus proches éléments (correspondant aux plus petites distances précédentes). On dispose alors seulement de k éléments de la liste des données.
- Dans ces k éléments, compter le nombre d'occurrences de chaque classe présente.
- Attribuer au nouvel élément N la classe la plus fréquente.

Construire dans un langage courant chacune des étapes suivantes.

d. Ouvrir le fichier csv avec LibreOffice Calc et l'enregistrer au format ods.

- Ajouter une colonne pour calculer la distance avec la taille et le poids d'un nouveau joueur, les colonnes supplémentaires auront cet aspect :

H	I	J
distance	taille joueur	poids joueur
24,08318916	185	98

  La formule de la distance saisie au tableur en H5 aura cet aspect :
  =RACINE((F2–I2)^2+(G2–J2)^2)
  Cependant il reste un problème quand on la recopie vers le bas. Corriger cette erreur.

- Filtrer les données pour ne plus voir apparaître que celle dont l'équipe nous intéresse.
- Trier enfin les données suivant la distance.

2. Traduction en python

a. Construire une fonction distance prenant 4 arguments.

b. Construire une fonction classification prenant comme arguments k, data (qui correspondra à la liste des données extraites du fichier csv), taille et poids (ces deux derniers arguments étant les valeurs d'un nouveau joueur).

- Dans cette fonction, il faudra extraire les k joueurs dont la taille et le poids sont les plus proches de celles du nouveau joueur. Décrire dans un premier temps le processus permettant d'extraire ces données.
- Déterminer enfin le type de poste le plus courant dans cette liste de k joueurs. Le nouveau joueur sera alors considéré comme de ce type de poste.

Partie 3 : Optimisation de l'algorithme

La démarche consistant à utiliser le tableur pour extraire les k données les plus proches incite à utiliser le tri de données en prenant la distance comme critère de tri. Cependant ce n'est pas optimal (la complexité est dans ce cas de l'ordre de $n\log_2(n)$).

Il est cependant possible de concevoir un algorithme de complexité $k \times n$.
Pour cela, il convient d'effectuer la sélection des k plus proches voisins pendant le parcours de la liste des données.
Dans la fonction classificationEfficace, on parcourt la liste data contenant *n* éléments. L'extraction des données se fait alors au fur et à mesure. Dans le cas où on positionne un nouvel élément dans l'extraction, on parcourt au maximum toute la liste extraction qui contient k éléments pour le positionner au bon endroit (du plus proche au plus éloigné).

La fonction rechercheTypePoste revient juste à parcourir la liste des k valeurs et à identifier le type de poste le plus courant.