Pilier ou pas ?

DILLON LEWIS

- Nationalité : Pays-de-Galles
- Age : 23
- Date de naissance : 04/01/1996
- Poids : 118 kg
- Taille : 183 cm

DUPONT Antoine

- Nationalité : Française
- Age : 23
- Date de naissance : 15/11/1996
- Taille : 174 cm
- Poids : 82 kg

? Parmi les informations dont vous disposez lesquels vous semblent pertinente pour déterminer le poste qu'occupe un joueur ?

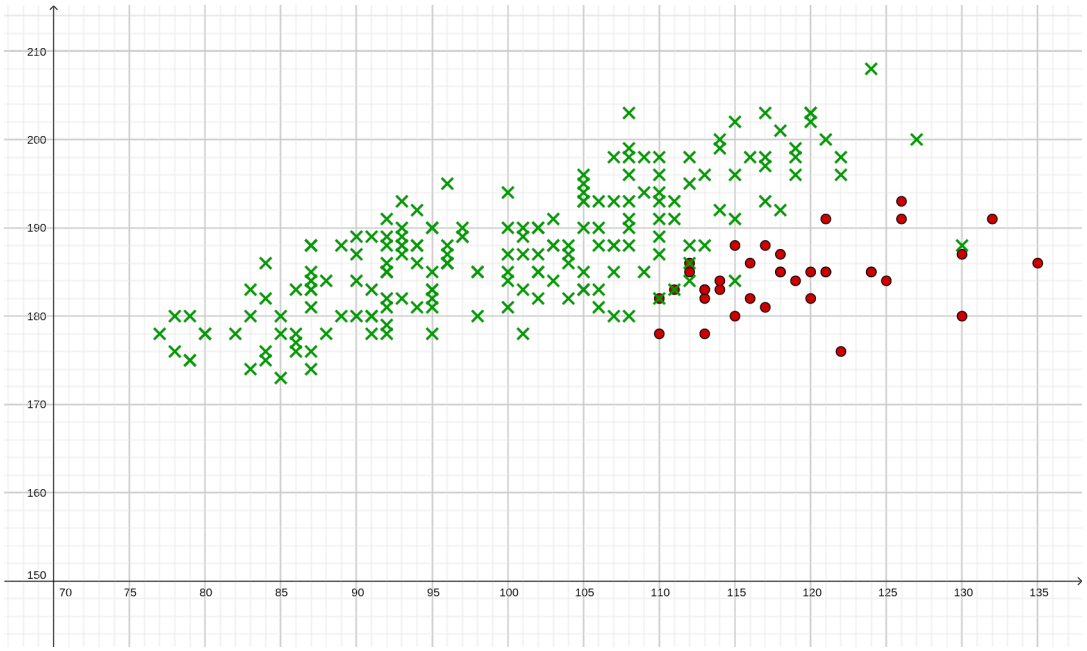
. . . . . .

. . . . . .

On a placé sur le graphique ci-dessous un ensemble de joueurs avec pour abscisse le poids des joueurs et pour ordonnée leur taille.

— Les piliers sont représentés par un •

— Les autres joueurs sont représentés par un ×



? Selon vous DILLON Lewis occupe-t-il le poste de pilier ? Justifier.

. . . . . .

. . . . . .

? Selon vous Antoine DUPONT occupe-t-il le poste de pilier ? Justifier.

. . . . . .

. . . . . .

Les données

Pour créer notre algorithme de classification, nous allons nous servir d'un fichier de données labélisées `rugby.csv` dont voici un extrait.

	Pays	Nom	Poste	Age	Date de naissance	Taille	Poids
0	France	Cyril BAILLE	Pilier	26	15/09/1993	182	116
1	Cameroun	Dany PRISO	Pilier	25	02/01/1994	182	110
2	Angleterre	Maro ITOJE	2ème ligne	25	28/10/1994	196	115
3	France	Yacouba CAMARA	3ème ligne	25	02/06/1994	195	112
4	Pays-de-Galles	George NORTH	Ailier	27	13/04/1992	194	109
...	...	...	...	...	...	...	...

Algorithme des K-plus proches voisins

i Le k-NN (k-nearest neighbors) est un algorithme d'apprentissage supervisé simple et intuitif, utilisé pour résoudre des problèmes de classification ou de régression.
Il repose sur une idée fondamentale :
« Pour prédire la classe (ou la valeur) d'un nouvel élément, regarde les k éléments les plus proches dans les données connues, et choisis la majorité (ou moyenne). »

• Préparation des outils

Pour implémenter l'algorithme des K-NN, nous avons besoin de plusieurs outils

- Une fonction **`traitement_donnees`** qui à partir des données brutes formate en données structurées les informations nécessaires
- Une fonction **`distance`** qui permettra de mesurer la *distance* entre 2 joueurs
- Une fonction **`mode`** qui donnera l'élément le plus fréquent d'une liste.

•• `traitement_donnees`

Le code ci-dessous permet d'obtenir une liste de dictionnaire, ou chaque dictionnaire aura pour clés, les en-tête du fichier CSV **`rugby.csv`** (”Pays”,”Nom”,”Poste”,”Age”,”Date de naissance”,”Taille”,”Poids”)

```
import csv

with open("rugby.csv","r") as f:
    datas = [ dict(r) for r in csv.DictReader(f)]
```

À faire

:

Créer une fonction **`traitement_donnees`** qui à partir de la variable **`datas`**, renvoie une liste de dictionnaire de la forme :

```
[{"label":"Pilier","caracteristiques":(182,116)},
 {"label":"2ème ligne","caracteristiques":(196,115)},
 ...
]
```

•• `distance`

Pour mesurer la distance entre deux points du plan A et B, on choisira la distance euclidienne :

$$d(x_A, y_A, x_B, y_B) = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

À faire

:

Ecrire une fonction distance qui prend en paramètre deux tuples (couple de nombres) **`A`** (**`A = (xA,yA)`**), et **`B`** (**`B = (xB,yB)`**) et qui renvoie la distance euclidienne qui sépare les deux points A et B. Par la suite on appliquera la fonction **`distance`** aux valeurs de la clé **`caracteristiques`** de deux joueurs.

•• mode

Le mode d'une liste est l'élément qui revient le plus fréquemment dans une liste.

> **À faire**
>
> :
>
> Ecrire une fonction mode qui prend en paramètre un tableau `tab` et qui renvoie l'élément le plus fréquent de cette liste.

L'algortihme des k-NN

•• Les entrées

Pour implémenter la fonction `k_nn`, nous aurons besoin des entrées suivantes.

- `k` : nombre de voisins à considérer
  - exemple : 3
- `dataset` : liste d'exemples connus, chacun sous la forme d'un dictionnaire `{ "label": ... , "caracteristiques" : ...}`
  - exemple : ("label" :"Pilier", "caracteristiques" : (188, 98))
- `nouveau` : le point à classer,
  - exemple : : (188, 98)

•• Etapes de l'algorithme

1. Pour chaque élément `{ label: ..., caracteristiques: ... }` du `dataset`, on calcule la distance entre `caracteristiques` et `nouveau` à l'aide de la fonction distance. On rajoute la clé `distance` à chaque entrée du dataset
2. On trie par ordre décroissant le dataset suivant la clé distance, on utilisera l'instruction ci-dessous :

```
dataset.sort(key=lambda x: x["distance"], reverse=True)
```

3. Sélection des k plus proches voisins.
   On garde uniquement les k premiers éléments du tri (`dataset`).
4. On retourne le `mode` de la séléction précédente.

> **À faire**
>
> :
>
> Compléter le code ci-dessous pour qu'il renvoie le label qu'il a trouver pour `nouveau`

```
def k_nn(k,dataset, nouveau):
    """
    Fonction retourne le mode d'une liste.

    @param k (int): nombre de voisins utilisés.
    @param donnees (list): liste de dictionnaire ayant une clé 'label' et une clé
    ↪  'caracteristiques'.
    @param nouveau (n-upplet): caractéristiques de la nouvelle donnée.

    @result (str): le label affecté à la nouvelle entrée.
    """

```