KHATIB
Rabab

Devoir d'informatique

1/ SELECT idpatient FROM MEDICAL WHERE
etat = 'hernie discale' ; ✓

2/ SELECT DISTINCT nom, prenom FROM PATIENT
JOIN MEDICAL
ON PATIENT.id = MEDICAL.idpatient
WHERE etat = 'spondylolisthesis' ; ✓

3/ SELECT * FROM (SELECT etat, COUNT (etat) AS et
FROM MEDICAL
GROUP BY etat)
WHERE et > 1 ; ?

Pas utile

Distinct

4/ Par des tableaux de grande taille, la bibliothèque Numpy permet un accès rapide aux données et de plus ce module permet de considérer séparement lignes et colonnes. ✓

5/ La table data est de taille N x n donc $100000 \times 6 = 600000$ et elle est codée sur 32 bits donc 4 octets.
Donc, pour cette table sont nécessaires $600000 \times 4 = 2400000$ octets, donc 2,4 Mo.
La table état est un vecteur de taille N codé sur 8 bits, donc sur un octet. Elle nécessite donc de 10^5 octets, donc 0,1 Mo.
Donc pour stocker les deux tables sont nécessaires 2,5 Mo ✓

6/ def separationParGroupe (data, etat):

```
    L = [[], [], []]
    for i in range (len(data)):
        L[etat[i]].append(data[i])
    return L
```

✓

7/ D'après la documentation du module matplotlib :

* ARGS 1

Le tableau de figures a n lignes et n colonnes. (car n = len(data[0]))
On numérote à partir de 1 la figure qu'on veut donc $k = i \times n + j + 1$ car i est l'abscisse et j l'ordonnée du tableau.

Donc ARGS 1 = (n, n, i*n+j+1) ✓

* TEST

Quand le test n'est pas vérifié, on trace un histogramme. Donc le test vérifie si on se trouve sur la diagonale donc :

TEST = i != j ✓

* ARGS 2

i désigne les abscisses donc datax et j les ordonnées donc datay. Donc,

ARGS 2 = (groupe[i], groupe[j], marker = mark[k]) ✓

* ARGS 3

L'histogramme des fréquences de l'attribut i s'obtient avec la colonne de la table data

correspondante à cet attribut. Donc,

ARGS3 = (data[:,j])

8/ Les diagrammes de la diagonale permettent de conjecturer une valeur moyenne (grâce aux fréquences) et ceux hors diagonale servent à étudier une éventuelle corrélation entre deux attributs. ✓

9/ Pour ramener toutes les valeurs de Xnorm entre 0 et 1, on soustrait (min(X)) et pour les proportionner correctement entre elles, on divise par la différence entre (max(X)) et (min(X)). On a donc,

$$x_{norm\,j} = \frac{x_j - (\min(X))}{(\max(X)) - (\min(X))}$$

✓

10/
```
def min_max(X):
    min, max = X[0], X[0]
    for i in range(len(X)):
        if X[i] < min:
            min = X[i]
        if X[i] > max:
            max = X[i]
```

✓

Que dire de la complexité ?

11/
```
def distance(z, data):
    L = []
```

✓

```
    for i in range (len(data)):
        for k in range (len(data[0])):
        L.append ( z[i] - data[i][k])
    return L
```

12/ PARTIE 1: création et tri d'un tableau de tableaux contenants la distance relative à une ligne du tableau data et le numéro de ligne, pour chaque ligne.

PARTIE 2: création d'un tableau qui associe une valeur à chaque patient selon sa proximité par rapport à un patient

PARTIE 3: renvoie l'indice du patient plus proche.

T : tableau de tableaux avec distances euclidiennes de chaque patient et leur id.

dist: liste avec la distance entre chaque n-uplet x et le n-uplet z.

select: liste de 'poids de proximité' pour chaque patient

ind : entier désignant l'id du patient plus proche