CRANCÉE Eliott MPSI 1

Informatique : DS n°9

Q1 :

```
SELECT idpatient FROM MEDICAL WHERE
etat = "hernie discale"
```

Q2 :

```
SELECT nom, prenom FROM PATIENT JOIN MEDICAL
ON PATIENT.id = MEDICAL.idpatient WHERE
etat = "spondylolis-thesis"
```

Q3 :

```
SELECT etat, count(etat) FROM MEDICAL GROUP BY etat
```

Q4 :

Le tableaux de Numpy sont plus performants que les tableaux usuels de Python.

De quel point de vue ?

Q5:

Le tableau data contient N x n coefficients codés sur 32 bits donc sur 4 octet.
Il faut donc 4 x N x n = 4 x 100 000 x 6 = 2 400 000 o = 2,4 Mo de mémoire pour le stocker
Le vecteur etat contient N coefficients codés sur 8 bits donc sur un octet
Il faut donc N = 100 000 o = 0,1 Mo de mémoire pour le stocker.
Il faut donc au total 2,5 Mo de stockage.

Q6:

```
def separationParGroupe(data, etat):
    N = len(etat)
    T1, T2, T0 = [], [], []
    for k in range(1, N):
        if etat[k] = 0:
            T0 += [data[k]]
        elif etat[k] = 1:
            T1 += [data[k]]
        else:
            T2 += [data[k]]
    return [T0, T1, T2]
```

Q7:

ARGS1 vaut n, n, (i-1)*n+j

ARGS2 vaut [x[i] for x in groupes[k]],
[x[j] for x in groupes[k]],
marker = mark[k]

ARGS3 vaut [x[i] for x in groupes[k]]

TEST vaut i != j

Q8:

Ceux de la diagonales permettent de connaitre la separation des personnes suivant leur valeur pour un attribut donné.
Les autres permettent de représenter la corelation entre deux attributs

Q9:

$$x_{normij} = \frac{x_j - min(X)}{max(X) - min(X)}$$

Q10:

```
def min_max(X):
    min, max = X[0], X[0]
    for x in X:
        if x > max:
            max = x
        elif x < min:
            min = x
    return min, max
```

Q11:

```
def distance(z, data):
    N = len(data)
    n = len(z)
    liste = []
    for k in range(1, N):
        distance_euclid = 0
        for i in range(n):
            distance_euclid = (data[k][i] - z[i]) ** 2
        liste += [sqrt(distance_euclid)]
    return liste
```

Q12:

La partie 1 crée la liste T à partir de la liste renvoyé par la fonction distance en ajoutant l'indice de vecteur correspondant.

La partie 2 compte dans la liste select le nombre de patient par état parmis les K voisins retenus.

La partie 3 permet de trouver pour quel état le nombre de patient est le plus élevé parmis la liste select.

T est la liste T de l'énoncé
dist est la liste que renvoie la fonction distance
select est la liste contenant le nombre de patient par état
ind est l'indice de l'état ayant le plus de patient.

Q14:

Il semble que K ne doit pas être trop petit ni trop grand. Ici une valeur de $K = \frac{N}{10}$ semble optimale. Cependant le taux de réussite est inférieur à 75 % ce qui n'est pas très précis.

Q15:

```
def moyenne(x):
    s = 0
    for k in x :
        s += k
    return s/len(x)
```

```
def variance(x):
    moyenne = moyenne(x)
    s = 0
    for k in x:
        s += (k - moyenne)**2
    return s/len(x)
```

Q16:

```
def synthese(data, etat):
    N = len(data)
    n = len(etat)
    nb = min_max(etat)[1]
    liste = [[0]*n]*(nb+1)
    groupes = separationParGroupe(data, etat)
    for k in range(nb+1)
        for i in range(n):
            liste[k][i] = [moyenne(groupes[k][:,i]),
                           variance(groupes[k][:,i]]
    return liste
```