THOMAS Germain

DS09

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

2) SELECT PATIENT.nom, PATIENT.prenom
FROM PATIENT, MEDICAL
WHERE PATIENT.id = MEDICAL.idpatient
AND etat = 'spondylolisthésis';

Jointure

3) SELECT ~~etat~~ MEDICAL.etat, COUNT(PATIENT.id)
FROM MEDICAL, PATIENT
WHERE MEDICAL.idpatient = PATIENT.id

4) La bibliothèque est préférable quant à l'affichage des tableaux et plus efficace pour les calculs matriciels

5) data : N x n cases x 32 bits = ~~000 000~~ bits
= 100 000 x 6 x 32 = 19 200 000

+ N x 8 = 800 000 bits

= 20 000 000 bits

/8 = 2 500 000 octets = 2,5 Mo

6) def Separation Par Groupe (data, etat)

```
N = len(data[:,0])
normal, hemie, spondy = [], [], []
for i in range(N):
    if etat[i] == 0:
        normal.append(data[i,:])
    elif etat[i] == 1:
        hemie.append(data[i,:])
    elif etat[i] == 2:
        spondy.append(data[i,:])
return [normal, hemie, spondy]
```

Oui

7) ARGS 1 : (6, 6, 6*c +(j+1))

ARGS 2 : (groupe[k][:, i], groupe[k][:, j], mark[k])

TEST : (i != j)

ARGS 3 : (data[:, i])

8) Les diagrammes hors diagonale permettent d'établir des corrélations entre les états.

Les diagrammes diagonaux permettent de connaître les effectifs ayant emprunté sur un certain état.

9) $x_{norm_j} = \frac{x_j - min(X)}{masc(X) - min(X)}$

10) def min-masc (X):

min, masc = X[0], X[0]

for i in range (len (X))

if X[i] < min :

min = X[i]

if X[i] > masc :

masc = X[i]

return min, masc

11)
```
def distance(z, data):
    L_dist = []
    N = len(data[:,0])
    n = len(data[0])
    for i in range(N):
        dist = 0
        for j in range(n):
            dist += (z[j] - data[i,j])**2
        L_dist.append(sqrt(dist))
    return L_dist
```

12) Partie 1 : Création de la liste triée contenant les distances (T)

Partie 2 : On sélectionne les K plus proches voisins

Partie 3 :

13) La diagonale indique le nombre de bons résultats par état

14) Le taux de réussite est assez stable entre 70% et 75% avec une valeur de K optimale aux alentours de 11

15) def moyenne (x) :

```
    n = len(x)
    moy = 0
    for i in range (n) :
        moy += x[i]
    return moy / n
```

```
def variance (x) :
    n = len(x)
    var = 0
    mu = moyenne (x)
    for i in range (n) :
        var += (x[i] - mu) ** 2
    return var / n
```

17) def gaussienne (a, moy, v)

```
    return (1 / sqrt(2 * pi * v)) * exp((-(a - moy)**2
                                         /(2 * var))
```