

Robrien

DS Info

Hörigens

Q1

```
select idpatient from MEDICAL  
where etat = "hernie discale";
```

Q2

```
select nom, prenom from PATIENT join MEDICAL  
on PATIENT.id = idpatient  
where etat = "spondylolisthésis";
```

Q3

```
select etat, count(etat) from MEDICAL;
```

Q2

```
select nom, prenom from PATIENT join  
(select idpatient from MEDICAL  
where etat = "spondylolisthésis"  
on id = idpatient;
```

Q3

```
select etat, count(etat) from MEDICAL;
```

Distinct

Q4

La bibliothèque Numpy permet de faire des opérations rapidement sur des tableaux de grande taille. On peut réaliser des opérations pratiques sur les tableaux avec Numpy

Pas uniquement



Q5 Tableaux: tableaux à  $n=6$  colonnes et  $N$  lignes  
 $\Rightarrow 6N$  cases, et chaque case contient 32 bits  $= 32 \times 8$   
 Donc tableaux contient  $32 \times 8 \times 6 \times 100\ 000\ 0$   
 $= 153,6\ Mo$

vecteurs:  $N$  cases de 8 bits

Donc  $8 \times 8 \times N$   
 $= 6,4\ Mo$

Autotal mémoire totale  $= 160,0\ Mo$

Q6)

def normalisationParGroupe(data, etab):

Q9)  $x_{normj} = \frac{x_j - \min(x)}{\max(x) - \min(x)}$

$x_j \in [\min(x); \max(x)]$

$x_j - \min(x) \in [0; \max(x) - \min(x)]$

$\frac{x_j - \min(x)}{\max(x) - \min(x)} \in [0; 1]$

$\max(x) - \min(x)$

On pose donc  $x_{normj} = \frac{x_j - \min(x)}{\max(x) - \min(x)}$

Q10)

def min\_max(x):

$m = x[0]$

$M = x[0]$

for i in range(len(x)):

if  $x[i] > M$ :

$M = x[i]$

if  $x[i] < m$ :

$m = x[i]$

return m, M.

Que dire de la complexité?



Q12 Partie 1 (3 à 7): T est une liste contenant des doubles qui contiennent la distance entre le  $m$ -uplet  $i$  et le  $m$ -uplets à donner ainsi que le  $i$ .  
Cette partie permet donc de calculer la distance du  $m$ -uplet ~~avec~~ interconnecté avec tous les autres.  
Et elle permet de les trier selon leurs distances

Partie 2: Cette partie trouve les voisins les plus proches du  $m$ -uplets interconnecté.

Partie 3: elle renvoie le ~~uplet~~ numéro du  $m$ -uplet le plus proche des  $m$ -uplets interconnecté.

dist = distance entre  $z$  et  $data$ .

~~select - liste~~ ~~not~~ permettant de localiser par la suite le

Q13 sur la diagonale (où  $j=i$ )

Cette diagonale permet de savoir le nombre de fois que l'état test correspondait à l'état prédit.

Q14 la courbe n'est pas très régulière

On remarque un maximum de réussite en  $K=10$ . Normalement, plus  $K$  augmente, plus le taux de réussite augmente. Ici, ce n'est pas le cas, donc l'algorithme n'est pas très efficace.



Q15

Fonction sum  
interdite

```
def moyenne(x):
    return (sum(x)) / (len(x))
```

def variance(x): rappel:  $\sum_{i=0}^{n-1} \mu = n\mu$

$$\text{Donc } \sigma^2 = \frac{\sum_{i=0}^{n-1} x_i - n\mu^2}{n}$$

$$\sigma^2 = \mu - \mu^2$$

```
def variance(x):
    V = moyenne(x) * (1 - moyenne(x))
    return V
```

Q16

```
def synthese(data, data2):
    A = []
    for i in [0, 1, 2]:
        for L in range(len(data)):
            T = [moyenne(data[L]), sqrt(variance(data[L]))]
            A.append(T)
    return A
```

Q17)

```
def gaussienne(a, moy, v):
    G = 1
    for i in range(len(R)):
        G = G * exp(-(a - moy)**2 / (2*v))
    sqrt(2*pi*v)
    return G
```