Rothrien
Hourigeux

DS Info

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;
```

Q4 La bibliothèque Numpy permet de faire des opérations rapidement sur des tableaux de grande taille. On peut réaliser des opérations matriciales sur les tableaux avec Numpy

Q5 tableaux : tableau à $n = 6$ colonnes et N lignes

$\Rightarrow 6N$ cases, et chaque case contient 32 bits $= 32 \times 8$ o

Donc tableaux contient $32 \times 8 \times 6 \times 100\,000$ o

$= 153,6$ Mo

vecteurs : N cases de 8 bits

Donc $8 \times 8 \times N$ o

$= 6,4$ Mo

Au total mémoire totale $= 160,0$ Mo

Q6)

```
def reparitionParGroupe(data, ecart):
```

Q9) ~~$x_{normj} = max$~~

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

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

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

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
```

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

Partie 2 : Cette partie trouve les k voisins les plus proches du n-uplet interessant.

Partie 3 : elle renvoie le ~~n-uplet~~ numéro du n-uplet le plus proche du n-uplet interessant.

dist = distance entre z et data.
~~select = liste ... permettant de localiser pas la ... le~~

Q13 sur la diagonale (qd j = i)
Cette diagonale permet de savoir le nombre de fois que l'etat test correspondait à l'etat 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

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

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

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

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

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

Q16

```
def synthese(data, ekot):
    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*Π*v)
    return G
```