HUYNH
Joey

DS9

Question 1:

```
SELECT idpatient FROM MEDICAL WHERE
etat = 'hernie discale' ;
```

Question 2:

```
SELECT nom, prénom FROM PATIENT, MEDICAL
WHERE PATIENT.id = idpatient AND
etat = 'spondylolisthésis'
```

Question 3:

```
SELECT etat, COUNT(*) AS 'nb de patient'
FROM MEDICAL GROUP BY etat
```

Question 4:
Le type ansy de Numpy prend moins de place en mémoire.

Question 5.

Il faut

$6 \times 32 \times N + N \times 8$ bits

soit en octets

$6 \times 4 \times N + N$

$= N (24 + 1)$

$= 25 N$

$= 25 . 10^5$

$= 2,5 . 10^6$ octets

Donc 2,5 Mo sont nécessaires

Question 6 :

```
def separationParGroupe(data, etat):
    A,B,C = [],[],[]
    N = etat.shape()[0]
    for i in range(N):
        if etat[i][0] == 0:
            T = []
            for j in range(6):
                T.append(data[i][j])
            A.append(T)
        elif etat[i][0] == 1:
            T = []
            for j in range(6):
                T.append(data[i][j])
            B.append(T)
        else:
            T = []
            for j in range(6)
                T.append(data[i][j])
            C.append(T)
    return A,B,C
```

Question 9 : $$x_{norm\,j} = \frac{x_j - \min(X)}{\max(X) - \min(X)}$$

Question 10 :

```
def min_max(X):
    n = len(X)
    mini, maxi = X[0], X[0]
    for k in range(1, n):
        if X[k] > maxi:
            maxi = X[k]
        elif X[k] < mini:
            mini = X[k]
    return mini, maxi
```

Question 11 :

```
def distance(z, data):
    N = data.shape()[0]
    L = []
    for i in range(N):
        S = 0
        for j in range(6):
            S = S + (data[i][j] - z[j]) **2
        L.append(numpy.sqrt(S))
    return L
```

Question 12 :

- La partie 1 crée une liste triée dont les éléments sont des listes contenant la distance entre la ligne i de data et z, ainsi que le numéro de la ligne i.

- La partie 2 crée une liste de longueur le nombre d'état. Puis elle compte le nombre de patient ayant chaque état, parmi le K plus proches voisins de z.

- La partie 3 retourne l'état dans lequel sont la majorité des patients parmi les K plus proches voisins de z.

- dist est la liste des distances entre z et chaque ligne de data
- T est la liste dont les éléments sont des listes contenant la distance entre z et la ligne i de data, ainsi que le nombre i.
- select est le tableau contenant pour chaque état, le nombre de patients ayant cet état parmi les K plus proches voisins de z.
- ind est l'indice du maximum du tableau précédent.

Question 13 :

Les nombres sur la diagonale indiquent, pour chaque état, le nombre de patients qui sont dans le même état que la majorité de leur K plus proches voisins.

La somme des éléments diagonaux est égale au nombre de patients qui ont reçu un diagnostic juste.

Ligne 1: - 23 % des patients dans l'état 0 ont la majorité de leurs K plus proches voisins dans l'état 0.

- 4 % des patients dans l'état 0 ont la majorité de leurs K plus proches voisins dans l'état 1

- 7 % des patients dans l'état 0 ont la majorité de leurs K plus proches voisins dans l'état 2.

Colonne 1: - 7 % des patients dans l'état 1 ont la majorité de leurs K plus proches voisins dans l'état 0

- 5 % des patients dans l'état 2 ont la majorité de leurs K plus proches voisins dans l'état 0.

Cette matrice sert à déterminer la pertinence des attributs choisis pour réaliser le diagnostic, elle détaille les faux-diagnostics.

Question 14:
La valeur de K ne semble pas avoir d'effet sur le taux de réussite.
L'algorithme n'est pas totalement efficace mais il y a plus de diagnostics justes que de diagnostics erronés.

Question 15:

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

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

Question 21

Pour K = 8, le taux de réussite de la méthode KNN est de 74 %