BERTON Margot

Devoir d'Informatique

MPSI 2

Q1/
```
SELECT idpatient
FROM MEDICAL
WHERE etat = "hernie discale";
```

Q2/
```
SELECT nom, prenom
FROM PATIENT
JOIN MEDICAL ON PATIENT.id = MEDICAL.idpatient
WHERE etat = "spondylolisthesis";
```

Q3/
```
SELECT etat, COUNT(*) AS "Nombre de patients"
FROM MEDICAL
GROUP BY etat;
```

Q4/ Il est intéressant d'utiliser des tableaux de la bibliothèque numpy quand ils sont de grandes tailles car on peut

accéder à n'importe quelle case du tableau plus rapidement qu'avec des listes Python. L'ordinateur y accède plus directement.

Q5/ Les réels du tableau sont codés sur 32 bits = 4 octets et pour N = 100 000, le tableau comporte 6 × 100 000 = 600 000 cases. Il prend donc une place de 4 × 600 000 = 2 400 000 octets, soit 2,4 Mo. Les valeurs du vecteur sont codées sur 1 octet et il comporte N = 100 000 valeurs. Il prend une place de 1 × 100 000 octets soit 0,1 Mo.

Au final, le tableau et le vecteur occupent

2,5 Mo.

Q6/
```
def separationParGroupe(data, etat):
    normal = []
    hernie = []
    spondy = []
    n = len(data)
    for i in range n:
        if etat[i] == 0:
            normal.append(data[i,:])
        elif etat[i] == 1:
            hernie.append(data[i,:])
        else:
            spondy.append(data[i,:])
    return [normal, hernie, spondy]
```

Q7/ ARGS 1 =

ARGS 2 =

ARGS3 =

TEST =

Q8/ Ces figures ont pour but de mettre en évidence les caractéristiques propres à chaque état. Les histogrammes sur la diagonale font apparaître une moyenne, de normalité. Si le patient s'éloigne trop de cette moyenne, on pourrait lui diagnostiquer une pathologie. Les figures hors-diagonale permettent de situer le patient dans un groupe état.

Q9/ $$x_{normj} = \frac{x_i - \min(X)}{\max(X) - \min(X)}$$

Q10/
```
def min_max (X):
    n = len(X)
    min = X[0]
    max = X[0]
    for i in range n:
        if X[i] < min:
            min = X[i]
        if X[i] > max:
            max = X[i]
    return min, max
```

Q11/
```
def distance (z, data):
    dist = []
    n = len(data)
    l = len( data[0])
```

```
    for i in range n:
        aux = 0
        for j in range l:
            aux = aux + (z[j] - data[i][j]) **2
        dist.append(numpy.sqrt(aux))
    return dist
```

Q 12/ La partie 1 calcule la distance euclidienne entre les données du n-uplet z et les données de chaque patient de data, puis elle trie ces distances par ordre croissant.

La partie 2 crée une liste de taille nb dont toutes les valeurs sont 0. Puis, elle incrémente de 1 ce tableau select aux positions 0, 1, 2 à chaque fois qu'un proche voisin (jusqu'au k-ième) appartient à l'état 0, 1 ou 2.

La partie 3 regarde quel état concentre le plus d'individus.

T représente les distances euclidiennes entre z et le patient i, triée dans l'ordre croissant.
dist représente les distances euclidiennes entre z et chaque ligne de data.
select représente le nombre de proches voisins par état.
ind est l'état présentant le plus grand nombre de proches voisins.

Q13/

Q14/ On obtient un meilleur taux de réussite en considérant entre 7 et 11 proches voisins.