Léo BONNEFOY MPSI2

DS info 9

Question 1

```
Select idpatient From Medical Where etat = `hernie discale'
```

Question 2

```
Select nom, prenom From Patient Where
        id = (select idpatient From Medical Where
        etat = `spondylosisthesis)
```

Question 3

```
Select Count(idpatient) from Medical Groups By etat
```

Question 4

On peut utiliser le moduler Array.make
et numpy a de nombreux fonctions pour manipuler ses tableaux.

Question 5

Pour le vecteur : N valeurs toutes de taille 8 bits

donc taille vecteur = N x 8 = 800 000 bits

or 1 octet = 8 bits

donc taille vecteur = 100 000 octets

= 0,1 Mo

Pour le tableau : 1 réel codé sur 32 bits

Pour N lignes et 6 colonnes

donc N×6 réels

donc N×6×32 bits

soit 19 200 000 bits

soit 2 400 000 octets

soit taille tableau = 2,4 Mo.

Taille totale : 2,5 Mo.

Question 6

```
def SeparationParGroupe(data, etat):
    tab = [[],[],[],[]]
    let N = len(data)
    for i in range(0, N):
        if etat.[i] = 0 :
            then tab.[i] = data.[i]
        if etat.[i] = 1 :
            then tab.[i] = data.[i]
        if etat.[i] = 2 :
            then tab.[i] = data.[i]
    return tab.
```

Question 7

(ARGS 1) = (n, n, jet ixm)

ARGS 2 = scatter(groupes[[i],:], groupes[:,[j]])

ARGS 3 = hist(groupes([i], :)

TEST : if i ≠ j:

Question 8

Les diagrammes de la diagonale servent à avoir un nombre physique de patients en fonction des 6

attributs ce qui permet de voir où se situe la moyenne.
Ceux hors diagonales servent à voir la cause à effet
d'un attribut sur un autre.

Question 9

$$x_{norm_j} = \frac{x_j}{max(X) + min(X)}$$

Question 10

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

Question 11

```
def distance( z, data) =
        liste_distance = [ ]
        n = len(data)
        N = len(data.[0])
        for i in range(n):
            distance = sqrt((z.[0] - data[i][0])^2
+(z.[1] - data.[i][1])^2 + (z.[2] - data.[i][2])^2
+(z.[3] - data.[i][3])^2 + (z.[4] - data.[i][4])^2
```

```
+ (g[5] - data[i][5])^2)
            liste_distance.append(distance)
    return liste_distance
```

Question 12

La partie 1 : création de la liste T voulu

dist[i] = distance entre le n-uplet n°i

et 3.

i = valeur de l'état du n-uplet n°i

On a bien une liste T de longueur len(dist)

a 2 éléments. et trie la liste T.

La partie 2 : Création d'une liste de longueur nb

rempli de valeur 0

Pour les K éléments voulu, on compte combien

chaque état est représenté car on ajoute 1

à select à l'endroit de l'état dès que l'on

en a un parmis les K.

Partie 3 : On renvoit le maximum de select

et on renvoit le n° du cas associé

T = liste à 2 éléments

dist = liste des distances entre 3 et chaque ligne du tableau data

select déjà dis dans la partie 2

ind aussi.

Question 13

Sur les diagonales : le nombre de fois que l'état prédit du patient est réellement le cas

Sur les autres endroits : le nombre de fois où l'état du patient a été ~~confondu~~ avec d'état prédit différent et avec donc quel état d'algo KNN s'est trompé.

Question 14

On ne dépasse jamais les 75% de taux de réussite et le fait d'augmenter trop K semble ne pas être forcément plus précis car le pic de réussite se situe autour de K = 10

Question 15

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