Juliette DEBONO NSI 2 18/06/2020

DS Info 3 - Ecrit

Question 1:

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

Question 2:

SELECT nom, prenom FROM PATIENT JOIN MEDICAL ON idpatient = PATIENT.id WHERE etat='spondylolisthésis';

Distinct

Question 3:

select etat, count(*) FROM Medical GROUP BY etat;

Question 4:

Les tableaux du module numpy sont de taille fixe donc ne prennent pas plus de place de prévus, et on peut y effectuer dessus des calculs vectoriels

Question 5:

Le tableau est de taille N×N donc 1 000 000 × 6, codé en 32 bits = 4 octets

ainsi, la qté de mémoire est 1 000 000 × 6 × 4 = 2 400 000 octets

soit 2,4 Mo.

le vecteur est de taille N = 100 000, codé en 8 bits = 1 octets.

Sa quantité de mémoire est donc 100 000 octets = 0,1 Mo.

La quantitée de mémoire totale est donc de 2,5 Mo.

question 6

```
def separationParGroupe(data, etat):
    normal = []
    herniediscale = []
    spondylolisthesis = []
    for i in range(len(data)):
        if etat[i] == 0:
            normal.append(data[i])
        elif etat[i] == 1:
            herniediscale.append(data[i])
        else:
            spondylolisthesis.append(data[i])
    return [normal, herniediscale, spondylolisthesis]
```

question 7

TEST : i != j

ARGS 1 : n, n, i, j

ARGS 2 : group[R][:,0], group[R][:,1], marker=mark[R]

ARGS 3 : data[:,5]

question 8

le diagramme diagonale permet de voir à quelle mesure corporelle il y a le plus de patient, celui des diagrammes

hors diagonales voir s'il y a un lien entre les différentes mesures corporelles des patients.

3. Apprentissage et prédiction

question 9 :

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

question 10 :

```
def min_max(X):
    return min(x), max(x)
```

Je ne sais pas si je peux utiliser min et max, donc sinon autre méthode sans :

ou

```
def min_max(X):
    min, max = (0,0)
    for i in X:
        if i > max:
            max = i
        if i < min:
            min = i
    return min, max
```

question 11

```
def normaliser(data):
    for i in range(data):
        min, max = min_max(data[:])
        data[:] = data[:] / max
    return data
```

```
def distance(z, data):
    data = normaliser(data)
    liste = []
    for i in range(len(data)):
        list.append(mean(abs(data[:]-z)))
    return list
```

je ne sais pas si je peux utiliser mean de module numpy, sinon utiliser fonction de la question 10

Non, vous ne pouvez pas

Un mot sur la complexité ?

question 12.

partie 1: On crée la liste T contenant les distances entre z et les n-uplet connus et indice de ce n-uplet, triée.

partie 2:
On crée la liste select qui contient le nombre de n-uplet dans chaque etats, pour les k-premiers n-uplets (les k plus proche car T triée)

partie 3: regarde pour quel état on a le plus de n-uplets en analysant chaque valeur de select, et renvoie le numéro de l'état en question, donc l'état le plus probable parmi les k plus proches vecteurs.

question 13:
la diagonale de la matrice correspond aux cas où l'état prédit correspond à l'état test. Dans cette matrice, les valeurs de la diagonale sont globalement plus importantes des autres valeurs, donc le test à une certaine fiabilité.
la première ligne correspond au moment où le patient est normal alors qu'il devrait être malade (si on exclut la première valeur qui est sur la diagonale), la 1e colonne correspond au moment où le patient à un état anormal alors qu'il devrait être normal. Toutes les valeurs hors diagonales sont des défauts du modèle

juliette delorme TIPSI 2 18/06/2020

DS info écrit 3 : Suite

question 14

D'algorithme est environ efficace à 73%, il frôle les 75% si on prend k autour de 10, et est vers 70% pour les valeurs de k s'éloignant de 10 : il faut donc prendre k=10 pour un maximum de fiabilité. Cependant, 75% signifie que l'algorithme à 1 chance sur 4 de se tromper dans son diagnostique, ce qui est élevé, et peut être grave surtout en médecine.

même à k=10,

V

question 15 :

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

ou sans sum

```
def moyenne (x)
    sum = 0
    for i in X
        sum += i
    return sum /len(x)
```

OK

```
def variance (x):
    x2= moyenne(x)**2
    var =0
    for i in X:
```

```
        var += i**2
    return var / len(l)
```

A revoir.

question 16

```
def synthese(data, etat):
    etat0, etat1, etat2 = ([], [], [])
    for i in range(len(data)):
        vect = data[i]
        moy = moyenne(vect)
        var = variance(vect)
        if etat[i] == 0:
            etat0.append([moy, var])
        elif etat[i] == 1:
            etat1.append([moy, var])
        else:
            etat2.append([moy, var])
    return [etat0, etat1, etat2]
```

Il manque une boucle

question 17

```
def gaussienne(a, moy, v):
    return (1/((2*pi*v)**0,5) * exp((-(a-moy)**2)/2*v))
```

question 18

```
def probabilite_group(t, data, etat):
    valeur = synthese(data, etat)
```

```
valeur_exp.append(gaussienne(x,
```