FORTIN Côme

MPSI2

18/06/20

DS d'informatique 09

1) Select idpatient from medical
where etat = "hernie discale";

✓

2) Select nom, prénom from patient
where id = (select idpatient from medical
where etat = "spondylolisthésis");

3) Select etat, count(*) as nbr_patient
from medical
group by etat

Il manque le distinct

4) Numpy offre des outils de calcul entre tableau, évitant de devoir écrire des programme.

Ce n'est pas le principal argument.

5)

6)
```
def separationParGroupe(data, etat):
    normal = []
    hernie = []
    spon = []
    for i in range(len(etat)):
        if etat[i] = 0:
            normal.append(data[i])
        elif etat[i] = 1:
            hernie.append(data[i])
        else:
            spon.append(data[i])
    return array(normal, hernie, spon)
```

7)
(ARG S1) = (N; n; i)

(ARG S2) = (groupe[j]; groupe[i]; marker = mark[k])

TEST = : i != j

8) Les diagrammes de la diagonale permettent de voir le nombre de patients entre chaque tranche d'unité pour chaque attribut.
Les diagrammes hors diagonale permettent de mettre une éventuelle corrélation entre chaque attribut.

9) Posons $x_{normj} = \frac{x_j - \min(X)}{\max(X) - \min(X)}$ ✓

10)
```
def min_max(X):
    min = X[0]
    max = X[0]
    for i in X:
        if i < min:
            min = i
        elif i > max:
            max = i
    return min, max
```
✓

Un petit mot sur la complexité ?

11)
```
def distance(z, data):
    liste_distance = []
    for i in range(N):
        D = []
        for j in range(n):
            D.append((z[j] - data[i][j])**2)
        distance = numpy.sqrt(sum D)
        liste_distance.append(distance)
    return liste_distance
```
✓

12) Dans la partie 1, l'algorithme crée une liste T avec des listes contenant la distance entre le n-uplet z et l'un des n-uplet i de data, ainsi que le numéro i de ce même n-uplet de data. Ensuite la liste est triée. ✓

Dans l'étape 2, l'algorithme va créer une liste select comportant nb 0. Ensuite, $\forall i \in [\![0; K-1]\!]$, à la position etat[T[i][1]], il rajoute 1 à select.

15)
```
def moyenne(x):
    mu=0
    for i in range(len(x)):
        mu = mu + x[i]
    return mu/len(x)
```

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

liste=[]

16)
```
def synthese(data, etat):
    ndata=reparationParGroupe(data, etat)
    for i in range(len(ndata[0])):
        X=[]
        for j in range(len(ndata)):
            X.append(ndata[j][i])
        liste.append([moyenne(X), variance(X)])
    return liste
```