ABIJNEV
Gaël
MPSI 1)

18/06/2020

DS Informatique

Question 1:
```
Select id From Medicap
where etat = "hernie discale" ;
```

Question 2:
```
Select nom, prenom
From ~~medicap~~ Patient
Join Medicap on ~~idp~~ patient.id = medicap.idpatient
where état = "spondylolisthésis" ;
```

Question 3:
```
Select Count(idpatient), etat
From medicap
Group by etat ;
```

Question 4: L'utilisation de la bibliothèque numérique Numpy est plus pratique car on a accès à plusieurs fonctions de Numpy qui nous permettent de manipuler les tableaux de grande taille avec plus de facilités. ~~que~~

Question 5 • Tout d'abord le tableau est formé de 100 000 lignes et de 6 colonnes. Chaque réel est codé sur 32 bits soit 4 octets. On obtient donc en tout 100 000 x 6 x 4 octets = 2 400 000 octets donc 2,4 Mo

• Pour ce qui est du vecteur de données, il est de taille N et chaque entier est codé sur 8 bits soit 1 octet. Donc il y a au total 100 000 octets soit 0,1 Mo

Finalement, pour stocker les tableaux data et etat lorsqu'il y a 100 000 patients, il faut 2,5 Mo

Question 6:

```
def separation_par_groupe(data, etat)
    sous_tableaux = [[]*3]
    For k in range len(data):
        sous_tableaux[etat[k]].append(data[k])
    return sous_tableaux
```

Question 7:

```
Arg S1 = (n, n, (n)rjr1)
Test = if i != j :
Arg S3 = label_attribute[i]
Arg S2 = (
```

Question 8: Les diagrammes sur la diagonale permettent d'avoir pour chaque attribut des statistiques sur les patients et ainsi de leur attribuer un symbole. Les tableaux hors de la diagonale permettent de voir si les attributs sont reliés entre eux (si un attribut a une incidence sur un autre)

Question 9: $x_{normj} = \frac{x_j + \min(x) + \max(x)}{3}$

Question 10:
```
def min_max(X)
    n = len(X)
    min = X[0]
    max = X[n-1]
    for k in range(n):
        if X[k] > max:
            max = X[k]
        IF X[k] < min:
            min = X[k]
    return min, max
```

Question 11:
```
def distance(z, data)
    N = len(data[0])
    L = list()
    from math import sqrt
    S = 0
    for i in range(N+1):
        for j in range(N):
            a = z[j]**2 - data[i][j]**2
            S += a
        L.append(sqrt(S))
```

12) Dans la partie 2: on crée une liste T dans laquelle on place toutes les valeurs de distances euclidienne calculées dans la question précédente. On trie ensuite cette liste par ordre croissant
La variable locale T est une liste de listes à 2 éléments contenant: • la distance entre le n-uplet à classer et le n-uplet connu
• La valeur de l'état associé au n-uplet connu

La variable locale dist correspondant à la liste de taille N contenant les distances entre chaque n-uplet x et le n-uplet z

Question 14 : On peut dire que plus le nombre de voisins est grand, plus le taux de réussite de l'algorithme est élevé (même si cela se stabilise vers la fin). Si n vaut, lorsque le nombre de proches voisins est petit (moins de 5 personnes) le taux de réussite de l'algorithme est inférieur à 70%

Question 15: def moyenne(x):
n = len(x)
somme = 0
for k in range(n):
somme = somme + X[k]
return(somme/n)

def Variance(X)
n = len(X)
moy = 0
var = 0
for k in range(n):
moy = moy + X[k]
var = var + X[k]**2
retur (var/n - (moy/n)**2)