HAMADI
Kalima

DS Informatique

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

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

3- SELECT etat, COUNT(idpatient) FROM MEDICAL GROUP BY etat

4- On peut facilement extraire des lignes et obtenir un tableau plus petit. La bibliotheque numpy est plus efficace pour gerer les tableaux que Python (*)

6- def separationParGroup(data, etat):
    etat 0, etat 1, etat 2 = [], [], []
    for k in range(len(data)):
        if etat k == 0:
            etat 0.append(data[k])
        elif etat[k] == 1:
            etat 1.append(data[k])
        else:
            etat 2.append(data[k])
    return etat 0, etat 1, etat 2.

7- ARGS1 = n^2

. ARGS2 sert à représenter les trois sous groupe en fonction de k. , donc ARGS2 = [groupes[i], groupe[j], mask=k)

. L'histogramme de ARGS3 compte le nombre de données.

donc ARGS3 = data[k]

TEST = 'j != i'

8- Les diagrammes de la diagonale permettent de voir la valeur moyenne des données

Les diagrammes hors diagonale permettent de faire des comparaisons selon la répartition des points

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

10- def min_max(X):

```
    min = X[]
    max = X[]
    for k in X:
        if k < min:
            min = k
        elif k > max:
            max = k
    return min, max
```

11- def distance(z, data):

```
    d = 0
    for k in range(len(z))
        d = sqrt((z[k] - data[k]) ** 2)
    return d
```

12 - Partie 1 : on ajoute un tableau contenant la distance de 1 et i à une liste T, puis on la trie.

Partie 2 : select représente.

13 - La diagonale de la matrice nous indique le taux de réussite de l'algorithme.

14 - L'algorithme n'est pas efficace pour les petites valeurs (entre 0 et 5) et pour les grandes valeurs (entre 15 et 20).

15 -
```
def moyenne(x):
    s = 0
    for k in x:
        s = k
    return (s / len(x))

def variance(x):
    s = 0
    m = moyenne(y)
    for k in y:
        s = k**2
```

return ((s / len(y)) - m**2)

16- def synthese (data, etat):
    m = moyenne (x)
    v = variance (x)
    for k in range (len (data)):
        if etat == 0:

        if etat == 1:

        if etat == 2:

    return([m, v])

17- def gaussienne (a, moy, v):
    moy = moyenne (X)
    v = variance (X)
    p = []
    for a in range (len (X)):

    return p

18. def probabilite Groupe (g, data, etat):

    return p

19. def prediction (

(*) S N = 100 000 ligne, n = 6 colonne
codé sur 32 bits = 4 octet donc la quantité de mémoire est :
2,4 Mo