ARBID
Hamza

DS09 : Informatique

Question 1 :

```
from numpy import *
import matplotlib.pyplot as plt
```

```
SELECT idpatient FROM medical
WHERE etat = 'hernie discale';
```

Question 2 :

```
SELECT p.nom, p.prenom FROM patient AS p
INNER JOIN medical AS m ON m.idpatient = p.id
WHERE etat = 'spondylolisthesis';
```

Question 3 :

```
SELECT etat, COUNT(idpatient) FROM medical
GROUP BY etat;
```

Question 4 :

La gestion des tableaux de grandes tailles par Numpy est plus efficace. De plus, les tableaux Numpy possèdent des fonctions

bien utiles et rapides.

Question 5 :

- Le tableau data est constitué de 100 000 lignes et 6 colonnes et chaque valeur du tableau est codée sur 32 bits = 4 o

Ainsi, la taille de data est $32 \times 600\,000 \times 4$

$= 2\,400\,000$ o $= 2,4$ Mo.

- Le tableau etat est constitué de 100 000 lignes et 1 colonne et chaque valeur du tableau est codée sur 8 bits = 1 o.

Ainsi, la taille de etat est $1 \times 1 \times 100\,000$

$= 100\,000$ o $= 0,1$ Mo.

La quantité de mémoire totale nécessaire est donc de $2,4 + 0,1 = \boxed{2,5 \text{ Mo}}$

Question 6

```
def separationParGroupes(data, etat):
    groupes = [[], [], []]
    for k in range(len(data)):
        groupes[etat[k]].append(data[k])
    return groupes
```

Question 7

ARGS1 : `ax1 = plt.subplot(n, n, i*n+j+1)`

TEST : `if i != j :`

ARGS2 : `ax1.scatter(groupes[k][:,i],`

`groupes[k][:,j],`

`marker = mark[k])`

ARGS3 : `ax1.hist(data[:,i])`

Question 8

- Les diagrammes de la diagonale permettent

de visualiser les moyennes et écart-types de chaque attributs.

. Ceux hors-diagonale permettent de visualiser les corrélations entre attributs et de repérer des zones de l'espace contenant plutôt des points d'un unique état.

question 9

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

question 10

```
def min_max(X):
    mini = maxi = X[0]
    for x in X:
        if x < mini:
            mini = x
        if x > maxi:
            maxi = x
```

return mini, maxi

On parcourt le vecteur une seule fois.

On a donc bien une complexité linéaire.

Question n1

```
def distance(z, data):
    n, dist = len(data), zeros(n)
    for i in range(n):
        x = data[i]
        for j in range(len(x)):
            dist[i] += (x[j]-z[j])**2
        dist[i] = sqrt(dist[i])
    return dist
```

Question n2

- La partie 1 calcule, à l'aide de la fonction distance (ci-dessus), les distances

euclidiennes entre le n-uplet z et chaque n-uplet x du tableau data, en stockant ces distances dans la liste dist. Elle stocke ensuite dans une nouvelle liste T les les listes à 2 éléments [d_i; i] et trie ensuite cette liste.

- La partie 2 compte, dans une liste select à nb éléments, le nombre d'états parmi les K plus proches voisins de z dans les données de data ; les K plus proches voisins sont les K premiers éléments de la T car T est triée.
- La partie 3 calcule le plus petit indice ind du maximum de la liste select c'est-à-dire l'état majoritaire parmi

des K plus proches voisins de z dans les données de data.

* T est la liste des voisins de z.
* dist est la liste des distances euclidiennes entre z et les n-uplets de data.
* Select est la liste des K voisins les plus proches de z.
* incl est l'etat majoritaire parmi les K plus proches voisins de z.

Question 13

La diagonale de la matrice contient le nombre de decisions correctes par chaque état : on a ainsi predit correctement l'état normal de 23 patients, l'hernie discale de 11 patients et la spondylolisthesis de 40 patients.

- La première ligne de la matrice compte les patients dont l'état est normal. Il y a donc 4 patients qui ont été diagnostiqué avec une hernie discale alors qu'il n'avait rien et 7 patients qui ont été faussement diagnostiqué avec la spondylolisthésis alors qu'ils n'avaient rien.

- La première colonne compte les patients dont l'état a été déclaré normal par l'algorithme. Il y a donc 7 patients ayant hernie qui ont été faussement diagnostiqué normal et 5 patients avec la spondylolisthésis qui ont été faussement diagnostiqué normal.

La matrice de confusion sert à mesurer la qualité d'un algorithme de diagnostic.

Question 14

La courbe admet 2 maximums globaux, de 7h 7. atteint en $K = 8$ et $K = 17$. Les meilleurs résultats sont atteints avec ces valeurs.

Question 15

```
def moyenne (u):
    a = 0
    for i in u:
        a += i
    return a / len(u)

def variance (u):
    moy, a = moyenne(u), 0
    for i in u:
        a += (i - moy) ** 2
    return a / len(u)
```

Ces 2 fonctions sont bien de complexité linéaires.

question 16

```
def synthese (data, etat) :

    groupes = separationParGroupe (data, etat)

    syn = []
    for i in range (len(groupes)) :
        groupes[i] = array (groupes[i])

        couples = []

        for j in range(len(groupes[i][0])):

            couples.append ([moyenne (groupes[i][:,j]),
                             variance (groupes[i][:,j])])

        syn.append (couples)

    return syn
```

question 17

```
def gaussienne (a, moy, v):

    return exp(-(a - moy)**2 /(2*v)) /sqrt(2*pi*v)
```

question 18

```
def probabiliteGroupe(z, data, etat):
    s = synthese(data, etat)
    nb, n = len(s), len(etat)
    probas = [0]*nb
    for e in etat:
        probas[e] += 1/n
    for j in range(nb):
        for i in range(len(data[0])):
            probas[j] *= gaussienne(z[i],
                                    s[j][i][0],
                                    s[j][i][1])
    return probas
```

question 19

```
def prediction(z, data, etat):
    probas = probabiliteGroupe(z, data, etat)
    indice_max = 0
```

```
    for j in range(1, len(probas)):
        if probas[j] > probas[indice_max]:
            indice_max = j
    return indice_max
```

Question 20 et 21

Pas de temps de finir