

II - Analyse des données

- 1) `SELECT id from medical where etat = "hernie discale";` ✓
- 2) `Select nom, prenom from patient join medical on patient.id = idpatient where etat = "spondylolisthésis";` ✓
- 3) `Select etat, count(*) from medical group by etat;` ✓
- 4) Numpy possède des outils permettant de facilement manipuler des tableaux de grande taille. ✓
- 5) On a 8 bits = 1 octet  
donc chaque valeur du vecteur est de 1 octet ✓  
et chaque valeur du tableau est de 4 octets  
On a donc une taille totale de  $4 \times 6N + N = 25N = 2,5 Mo$  ✓

6) def separationByGroupe(data, etat):

zero = []

un = []

deux = []

for i in range(len(etat)):

temp = data[i, :]

if etat[i] == 0:

zero.append(temp)

else:

if etat[i] == 1:

un.append(temp)

else:

deux.append(temp)

return([zero, un, deux])

7) ARG1 :  $n, n, i+1+n*j$

ARG2 : groupes[k][:, i], groupes[k][:, j], marker = mark[k]

ARG3 : data[i, i]

TEST :  $i \neq j$

8) Les diagrammes de la diagonale permettent de voir la fréquence des valeurs des attributs

Les diagrammes hors-diagonale permettent de voir le lien entre les différents attributs en général et en fonction des états.

## IV. Apprentissage et prédiction

### 1. Méthode KNN

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

10) def min\_max(X):

min = X[0]

max = X[0]

for k in X:

if min > k:

min = k

if k > max:

max = k

return (min, max)

11) def distance(z, data):

N = []

n = len(z)

tmin = []

tmax = []

for k in range(n):

min, max = min\_max(data[:, k])

tmin.append(min)

tmax.append(max)

for x in data:

S = 0

for k in range(n):

S += ((z[k] - x[k]) / (tmax[k] - tmin[k])) \*\* 2

$S = \text{opt}(s)$

$N = \text{app}(s)$

$\text{return}(N)$

12) - La partie 1 crée la liste  $T$  et la trie **logique**

- La partie 2 compte parmi les  $K$  plus proches voisins les différents états
- La partie 3 donne l'état le plus présent parmi les  $K$  plus proches voisins.
- $T$  est la liste d'entrée au début de la question
- $\text{dist}$  est la liste contenant la distance entre chaque  $n$ -tuple  $x$  et le  $n$ -tuple  $z$
- $\text{select}$  est le compte des états parmi les  $K$  plus proches voisins
- $\text{ind}$  est l'état le plus présent parmi les  $K$  plus proches voisins.

13) La diagonale correspond au nombre de fois où l'algorithme a correctement prédit l'état (le cas  $i, i$  correspondant à l'état  $i$ )

La première ligne indique que lorsque le patient était sain, l'algorithme a prédit 23 fois qu'il était sain, 4 fois qu'il avait une hernie discale et 7 fois qu'il avait une spondylolisthésis.

La première colonne indique que lorsque l'algorithme avait prédit que le patient était sain, 23 fois le patient était sain, 7 fois il avait une hernie discale et 9 fois il avait une spondylolisthésis.

Cette matrice nous permet de comparer l'état donné par l'algorithme et l'état réel pour un  $K$  donné.

14) On observe sur la courbe que le  $K$  optimal est autour de 10. Ce qui nous donnerais une efficacité de environ 74%.



## 2. Méthode de classification naïve bayésienne

15) def moyenne(x):

$s = 0$

$n = \text{len}(x)$

for k in x:

$s += k$

return (s/n)

def variance(x):

$s = 0$

$n = \text{len}(x)$

$m = \text{moyenne}(x)$

for k in x:

$s += (x - m) * x$

return (s/n)

16) def synthese(data, etat):

data\_0 = separation\_par\_bouye(data, etat)

N = []

for X in data\_0:

N1 = []

for Y in X:

$m = \text{moyenne}(Y)$

$v = \text{variance}(Y)$

N1.append([m, v])

N.append(N1)

return (N)

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

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

18) def probabilite\_Groupe(z, data, etat):

synt = synthese(data, etat)

tbl = []

for X in synt:

P = 1

for k in range(len(X)):

m, v = X[k]

P \*= gaussienne(z[k], m, v)

tbl.append(P)

return(tbl)

19) def prediction(z, data, etat):

tbl = probabilite\_Groupe(z, data, etat)

i = 0

max = tbl[0]

for k in range(1, len(tbl)):

if tbl[k] > max:

max = tbl[k]

i = k

return(i)

20) On voit dans les probabilités données qu'elles sont distantes de guinssomes de 20, le logarithme permet donc de réduire la taille des objets étudiés sans perdre en précision.

27) Pourcentage de réussite :  $\frac{\text{cas identifiés}}{\text{cas totaux}} \times 100$

$$\% \text{ réussite KKV} : \frac{23 + 17 + 40}{23 + 4 + 7 + 7 + 17 + 7 + 5 + 2 + 40} \times 100 = 74\%$$

$$\% \text{ réussite questionnaire} : \frac{23 + 10 + 49}{23 + 9 + 8 + 9 + 10 + 7 + 10 + 7 + 49} \times 100 = 68,3\%$$

Il semble que la méthode KKV soit plus performante ici.