

DS info

2. Analyse des données

1. SELECT idpatient FROM
(SELECT * FROM PATIENT JOIN
MEDICAL ON PATIENT.id = MEDICAL.
idpatient) WHERE etat = "hernie discale";

2. SELECT nom, prenom FROM
(SELECT * FROM PATIENT JOIN
MEDICAL ON PATIENT.id = MEDICAL.idpatient)
WHERE etat = "spondylolisthésis";

3.

4. C'est plus lisible au sans
game away, on aura les tableaux
sans game matricielle!

5. Le tableau data contient 100 000
lignes et 6 colonnes chaque case codée
sur 32 bits cod 4 octets aucun *

6. def separationParGroupe (data, etat) :

 A = []

 B = []

 C = []

 for i in range (len(data)) :

 if etat[i][0] == 0 :

 A.append(data[i])

 elif etat[i][0] == 1 :

 B.append(data[i])

 else :

 C.append(data[i])

 return [A, B, C].

7. La double boucle permet de dessiner indépendamment chaque graphique de coord. (i, j) pour $i, j \in [0, m-1]$ ainsi on commence par sélectionner le graphique correspondant dans la matrice à l'aide de la ligne $ax1 = plt.subplot(m, m, c \times m + j + 1)$
Ainsi $ARGS1 = m, m, c \times m + j + 1$.

3. Apprentissage et prédiction

3. on aurait $x_{normj} = \frac{x_j - \min(x)}{\max(x) - \min(x)}$

10. def min-max(x):

a = 0

b = 0

for a in range(len(x)):

i) len(x[a]) > b:

b = len(x[a])

elif len(x[a]) < a:

a = len(x[a])

return a, b

on a donc une complexité linéaire.

11. def distance(z, data):

A = []

for i in range(len(data[0])):

A.append(data[i])

for i in range(len(data[0])):

A.append(data[i])

* il requiert au total $N \times m \times 4$ octets
soit 2,4 Mo. De même, état
requiert 0,1 Mo ainsi
les deux requièrent donc 2,5 Mo.

12. • la partie 1 crée une liste

T et la partie 2 crée une liste de 0

• partie 3:

14. On remarque que l'algorithme
est plutôt efficace mais
le pourcentage ne dépasse
pas 75%. ainsi il n'est
quand même pas totalement
fiable.