Concours Commun INP (CCINP), France

Session 2019, filière PSI

Informatique

Durée : 3 heures

Ce que traite le sujet

- Parties I et II : présentation et analyse des données
- Partie III.1 : méthode KNN
- Partie III.2 : classification naïve bayésienne

Sujet reproduit sans retouche à partir de la page suivante.

Résumé des parties, chapitres de cours liés et autres années :
prepahelp.com/annales/ccinp/psi/2019/info

Prepa Help ne revendique aucun droit sur le sujet, propriété de l'organisateur du concours.

CONCOURS COMMUN INP

ÉPREUVE SPÉCIFIQUE - FILIÈRE PSI

INFORMATIQUE

Vendredi 3 mai : 8 h - 11 h

N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

Les calculatrices sont interdites

Le sujet est composé de trois parties indépendantes.

L'épreuve est à traiter en langage **Python**, sauf les questions sur les bases de données qui seront traitées en langage **SQL**. La syntaxe de Python est rappelée en **Annexe**, page 12.
Les différents algorithmes doivent être rendus dans leur forme définitive sur la copie en respectant les éléments de syntaxe du langage (les brouillons ne sont pas acceptés).
Il est demandé au candidat de bien vouloir rédiger ses réponses **en précisant bien le numéro de la question traitée et, si possible, dans l'ordre des questions**. Bien que largement indépendante, la **partie III** fait appel aux données et à la définition de fonctions définies dans la **partie II**.
La réponse ne doit pas se cantonner à la rédaction de l'algorithme sans explication, les programmes doivent être expliqués et commentés.

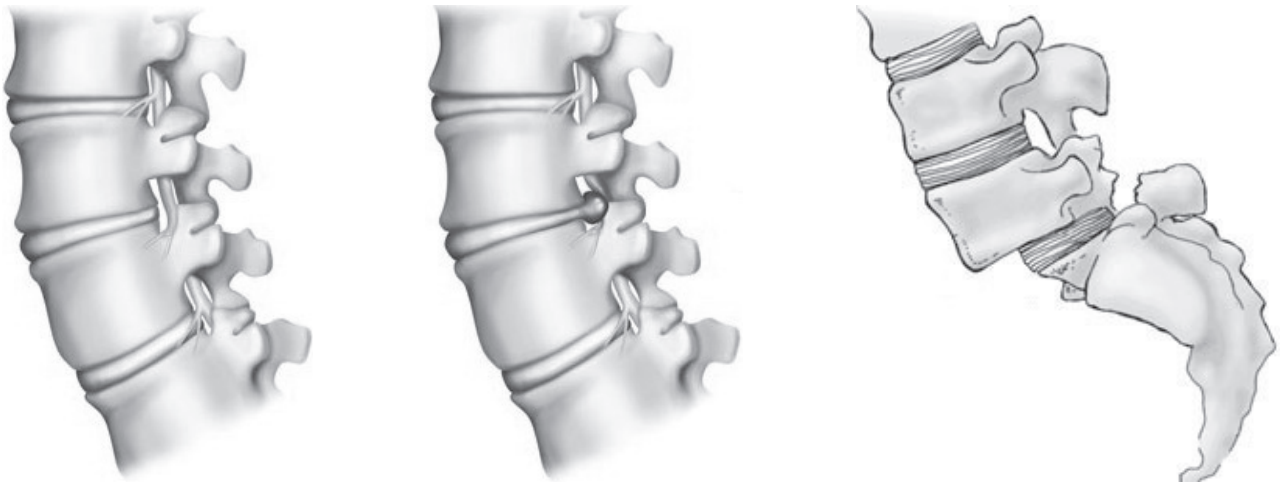
Sujet : page 1 à page 11
Annexe : page 12

Intelligence Artificielle - Application en médecine

Partie I - Présentation

L'Intelligence Artificielle est de plus en plus utilisée dans de nombreux domaines divers et variés. Les techniques d'apprentissage machine, qui permettent à un logiciel d'apprendre automatiquement à partir de données au lieu d'être explicitement programmé, fournissent des résultats impressionnants. Les applications sont nombreuses dans la reconnaissance d'image, la compréhension de la parole ou de textes, dans l'assistance à la décision, dans la classification de données, dans la robotique...
L'Intelligence Artificielle progresse également dans le domaine de la santé. Le logiciel Watson développé par IBM peut analyser les données d'un patient : ses symptômes, ses consultations médicales, ses antécédents familiaux, ses données comportementales, ses résultats d'examen, etc. Il établit alors une prévision de diagnostic le plus vraisemblable et propose des options de traitement en s'appuyant sur une base de données établie sur un grand nombre de patients.

Objectif
L'objectif du travail proposé est de découvrir quelques techniques d'Intelligence Artificielle en s'appuyant sur un problème médical. À partir d'une base de données comportant des propriétés sur le bassin et le rachis lombaire (figures 1), on cherche à déterminer si un patient peut être considéré comme « normal » ou bien si ces données impliquent un développement anormal de type hernie discale (saillie d'un disque intervétébral) ou spondylolisthésis (glissement du corps vertébral par rapport à la vertèbre sous-jacente).



(a) Normal (b) Hernie discale (c) Spondylolisthésis

Figures 1 – Différentes configurations des vertèbres

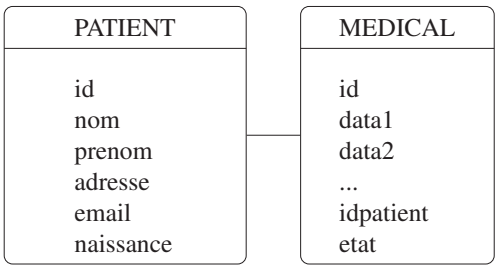
Le sujet abordera les points suivants :
- analyse et représentation des données,
- prédiction à l'aide de la méthode KNN,
- apprentissage et prédiction à l'aide de la méthode dite « Naïve Bayes ».

Dans tout le sujet, il sera supposé que les modules python `numpy`, `matplotlib.pyplot` sont déjà importés dans le programme.

Partie II - Analyse des données

La base de données médicale contient des informations administratives sur les patients et des informations médicales. Pour simplifier le problème, on considère deux tables : PATIENT et MEDICAL.

La table PATIENT contient les attributs suivants :
- id : identifiant d'un individu (entier), clé primaire ;
- nom : nom du patient (chaîne de caractères) ;
- prenom : prénom du patient (chaîne de caractères) ;
- adresse : adresse du patient (chaîne de caractères) ;
- email : (chaîne de caractères) ;
- naissance : année de naissance (entier).



La table MEDICAL contient les attributs suivants :
- id : identifiant d'un ensemble de propriétés médicales (entier), clé primaire ;
- data1 : donnée (flottant) ;
- data2 : donnée (flottant) ;
- ... ;
- idpatient : identifiant du patient représenté par l'attribut id de la table PATIENT (entier) ;
- etat : description de l'état du patient (chaîne de caractères).

Les attributs data1, data2... sont des données relatives à l'analyse médicale souhaitée (dans notre cas des données biomécaniques). L'attribut « etat » permet d'affecter un label à un ensemble de données médicales : « normal », « hernie discale », « spondylolisthésis ».

Q1. Écrire une requête SQL permettant d'extraire les identifiants des patients ayant une « hernie discale ».

Q2. Écrire une requête SQL permettant d'extraire les noms et prénoms des patients atteints de « spondylolisthésis ».

Q3. Écrire une requête SQL permettant d'extraire chaque état et le nombre de patients pour chaque état.

Une telle base de données permet donc de faire de nombreuses recherches intéressantes pour essayer de trouver des liens entre les données des patients et une maladie. Cependant, compte-tenu du nombre d'informations disponibles, il est nécessaire de mettre en place des outils pour aider à la classification de nouveaux patients. C'est l'objet des algorithmes d'Intelligence Artificielle développés dans la suite.
Pour l'étude qui va suivre, on extrait de la base de données les attributs biomécaniques de chaque patient que l'on stocke dans un tableau de réels (codés sur 32 bits) à N lignes (nombre de patients) et n colonnes (nombre d'attributs égal à 6 dans notre exemple). On nomme `data` ce tableau. L'état de santé est stocké dans un vecteur de taille N contenant des valeurs entières (codées sur 8 bits) correspondant aux différents états pris en compte (0 : normal, 1 : hernie discale, 2 : spondylolisthésis). On le nomme : `etat`.

Les variables `data` et `etat` sont stockées dans des objets de type `array` de la bibliothèque Numpy. Des rappels quant à l'utilisation de ce module Python sont donnés dans l'**Annexe**, page 12.

Q4. Citer un intérêt d'utiliser la bibliothèque de calcul numérique Numpy quand les tableaux sont de grande taille.

Q5. Déterminer la quantité de mémoire totale en Mo (1 Mo = 1 000 000 octets) nécessaire pour stocker le tableau et le vecteur des données si $N = 100\,000$. On supposera que les données sont représentées en suivant la norme usuelle IEEE 754.

Les 6 attributs considérés dans notre exemple (les n colonnes du tableau) sont définis ci-après :

- angle d'incidence du bassin en $^\circ$;
- angle d'orientation du bassin en $^\circ$;
- angle de lordose lombaire en $^\circ$;
- pente du sacrum en $^\circ$;
- rayon du bassin en mm ;
- distance algébrique de glissement de spondylolisthésis en mm.

Les labels de ces attributs sont stockés dans une liste nommée :
`label_attributs = ['incidence_bassin', 'orientation_bassin', 'angle_lordose', 'pente_sacrum', 'rayon_bassin', 'glissement_spon']`.

Le tableau suivant montre les premières valeurs du tableau `data`.

incidence_bassin	orientation_bassin	angle_lordose	pente_sacrum	rayon_bassin	glissement_spon
63,03	22,55	39,61	40,48	98,67	– 0,25
39,06	10,06	25,02	29,0	114,41	4,56
68,83	22,22	50,09	46,61	105,99	– 3,53
69,3	24,65	44,31	44,64	101,87	11,21
49,71	9,65	28,32	40,06	108,17	7,92
40,25	13,92	25,12	26,33	130,33	2,23
48,26	16,42	36,33	31,84	94,88	28,34

Tableau 1 – Données (partielles) des patients à diagnostiquer

Avant de traiter les données pour réaliser une prédiction (ou diagnostic), il peut être intéressant de les visualiser en traçant un attribut en fonction d'un autre attribut et en utilisant des motifs différents selon les valeurs du vecteur `etat`.
On obtient, à partir des données exploitées, les courbes de la **figure 2** (zoom sur la **figure 3**).

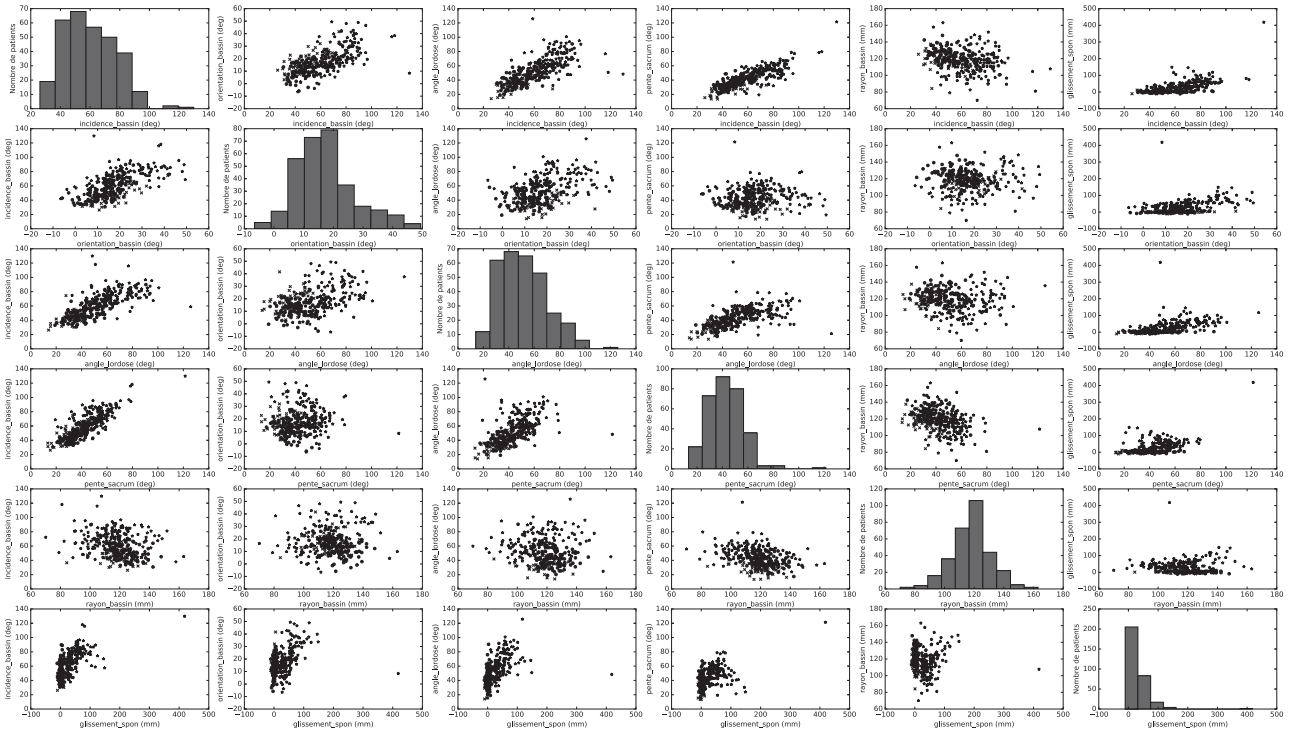


Figure 2 – Répartition des données

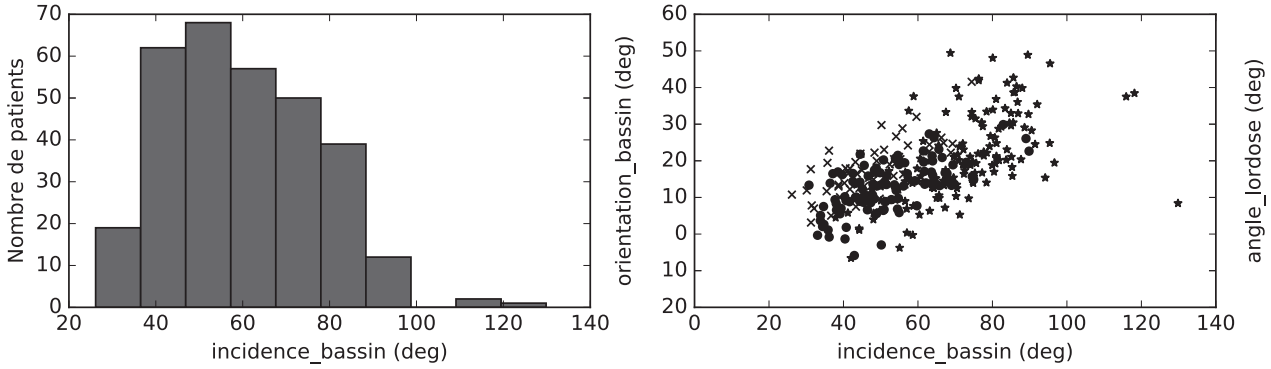


Figure 3 – Zoom sur les figures de coordonnées (1,1) et (1,2) de la matrice de répartition des données (**figure 2**)

Cette figure est une matrice dont le terme (i, i) de la diagonale est l'histogramme des fréquences de l'attribut i et les termes extra-diagonaux (i, j) représentent l'attribut i en fonction de l'attribut j pour chaque ligne du tableau `data`.

Sur les figures hors diagonale, on représente chaque donnée d'attribut i en fonction de l'attribut j à l'aide d'un symbole dépendant de l'état du patient (un rond 'o' si l'état du patient est normal, une croix 'x' si l'état est "hernie discale" et une étoile '*' si l'état est "spondylolisthésis").
Pour réaliser cette figure, il faut séparer les données en fonction de l'état du patient afin d'affecter un symbole par état. Ceci revient à classer les patients en plusieurs groupes.

Q6. Écrire une fonction `separationParGroupe(data,etat)` qui sépare le tableau `data` en 3 sous-tableaux en fonction des valeurs du vecteur `etat` correspondantes (on rappelle que celui-ci contient les valeurs 0, 1 et 2 uniquement). La fonction doit renvoyer une liste de taille 3 de sous-tableaux. Les sous-tableaux ne seront pas nécessairement représentés par un type `'array'`; `'liste de listes'`, `'liste d'array'` conviennent également.

Pour définir la figure, les instructions suivantes sont exécutées :

```
1  fig = plt.figure()
2  mark = ['o','x','*']
3  label_attributs = ['incidence_bassin (deg)', 'orientation_bassin (deg)',
4                     'angle_lordose (deg)', 'pente_sacrum (deg)',
5                     'rayon_bassin (mm)',    'glissement_spon (mm)']
6  groupes = separationParGroupe(data,etat)
7  for i in range (len(groupes)):
8      groupes[i] = array(groupes[i])
9
10 n=len(data[0])
11 for i in range(n):
12     for j in range(n):
13         ax1 = plt.subplot(ARGS1)
14         plt.ylabel(label_attributs[j]) #mettre un label à l'axe y
15         if TEST :
16             for k in range(len(groupes)):
17                 plt.xlabel(label_attributs[i]) #mettre un label à l'axe x
18                 ax1.scatter(ARGS2)
19         else:
20             plt.xlabel("Nombre de patients") #mettre un label à l'axe x
21             ax1.hist(ARGS3)
22 plt.show()
```

Les instructions précédentes utilisent la fonction `separationParGroupe` définie à la question **Q6**, puis les éléments de la liste `groupes` sont convertis en `array` afin de pouvoir extraire les colonnes plus facilement.
La documentation du module `matplotlib` (`plt`) renseigne sur les arguments des fonctions `subplot`, `scatter` et `hist`.

```
ax1 = plt.subplot(a,b,k)
```

Cette instruction permet de sélectionner parmi un tableau de figures de taille a (nombre de lignes), b (nombre de colonnes), la k^e en numérotant les sous-figures de 1 à $m \times n$ en partant du haut gauche vers le bas droite (en allant de gauche à droite et de haut en bas).

```
ax1.scatter(datax, datay, marker=mark[k])
```

Cette commande permet de tracer sur la sous-figure ax1 un nuage de points d'abscisses un vecteur `datax` et d'ordonnées un vecteur `datay` avec un symbole à choisir parmi ceux de la liste `mark`.

```
ax1.hist(datax)
```

Cette commande permet de tracer un histogramme des données `datax` sur la sous-figure ax1.

Q7. Définir les arguments ARGS1, ARGS2, ARGS3 ainsi que la condition TEST définis dans le script précédent permettant d'obtenir la **figure 2**.

Q8. Préciser l'utilité des diagrammes de la diagonale ainsi que celle des diagrammes hors diagonale.

Partie III - Apprentissage et prédiction

III.1 - Méthode KNN

La méthode KNN est une méthode d'apprentissage dite supervisée ; les données sont déjà classées par groupes clairement identifiés et on cherche dans quels groupes appartiennent de nouvelles données.
Le principe de la méthode est simple. Après avoir calculé la distance euclidienne entre toutes les données connues des patients et les données d'un nouveau patient à classer, on extrait les K données connues les plus proches. L'appartenance du nouveau patient à un groupe est obtenue en cherchant le groupe majoritaire, c'est-à-dire, le groupe qui apparaît être le plus représentatif parmi les K données connues.
On note $X_i,\ i \in [\![0,\ n-1]\!]$, le vecteur colonne i du tableau de données `data`, c'est-à-dire les valeurs prises par l'attribut i des données des patients déjà classées. On cherche à déterminer à quel groupe appartient un nouveau patient dont les attributs sont représentés par un n-uplet z de valeurs notées $(z_0, z_1, ..., z_{n-1})$.

Préparation des données

Avant de calculer la distance euclidienne, il est préférable de normaliser les attributs pour éviter qu'un attribut ait plus de poids par rapport à un autre. On choisit de ramener toutes les valeurs des attributs entre 0 et 1.
Une technique de normalisation consiste à rechercher pour chaque attribut X le minimum ($\min(X)$) qui est placé à 0 et le maximum ($\max(X)$) qui est placé à 1. On note alors X_{norm} un des vecteurs colonne X après normalisation (toutes les valeurs de X_{norm} sont donc comprises entre 0 et 1).

Q9. Proposer une expression de $x_{norm\,j}$ un élément du vecteur X_{norm} en fonction de l'élément x_j du vecteur X correspondant et de min(X) et max(X).

Q10. Écrire une fonction `min_max(X)` qui retourne les valeurs du minimum et du maximum d'un vecteur X passé en argument. La fonction devra être de complexité linéaire.

Q11. Écrire une fonction `distance(z,data)` qui parcourt les N lignes du tableau *data* et calcule les distances euclidiennes entre le n-uplet z et chaque n-uplet x du tableau de données connues (x représente une ligne du tableau). La fonction doit renvoyer une liste de taille N contenant les distances entre chaque n-uplet x et le n-uplet z.

Détermination des K plus proches voisins

Pour déterminer les K plus proches voisins avec K un entier choisi arbitrairement, il suffit d'utiliser un algorithme de tri efficace. La liste `T` à trier est une liste de listes à 2 éléments contenant :

- la distance entre le n-uplet à classer et un n-uplet connu (on trie par ordre croissant sur ces valeurs) ;
- la valeur de l'état correspondant au n-uplet connu.

On retient l'algorithme suivant :

```
1  def tri(T):
2      if len(T) <= 1:
3          return T
4      else:
5          m = len(T)//2
6          tmp1 = []
7          for x in range(m):
8              tmp1.append(T[x])
9          tmp2 = []
10         for x in range(m,len(T)):
11             tmp2.append(T[x])
12         return fct(tri(tmp1),tri(tmp2))
13
14 def fct(T1,T2):
15     if T1 == []:
16         ..................  #ligne 1 à compléter
17     if T2 == []:
18         ..................  #ligne 2 à compléter
19     if T1[0][0] < T2[0][0]:
20         return [T1[0]]+fct(T1[1:],T2)
21     else:
22         ..................  #ligne 3 à compléter
```

Q12. Donner le nom de ce tri ainsi que son intérêt par rapport à un tri par insertion. Préciser un inconvénient du programme proposé par rapport à ce tri.

Q13. Préciser les lignes 1 à 3 de la fonction `fct` pour que l'algorithme de tri soit fonctionnel.

L'algorithme de la méthode KNN est décrit par la fonction python suivante :

```
1  def KNN(data,etat,z,K,nb):
2      #partie 1
3      T = []
4      dist = distance(z,data)
5      for i in range(len(dist)):
6          T.append([dist[i],i])
7      tri(T)
8
9      #partie 2
10     select = [0]*nb
11     for i in range(K):
12         select[etat[T[i][1]]]+=1
13
14     #partie 3
15     ind = 0
16     res = select[0]
17     for k in range(1,nb):
18         if select[k] > res:
19             res = select[k]
20             ind = k
21     return ind
```

K représente le nombre de voisins proches retenus, *nb* correspond au nombre d'état (3 dans notre exemple) et z correspond aux attributs du patient à classer.

Q14. Expliquer ce que font globalement les parties 1 (lignes 3 à 7), 2 (lignes 10 à 12) et 3 (lignes 15 à 21) de l'algorithme. Préciser ce que représentent les variables locales `T`, `dist`, `select`, `ind`.

Validation de l'algorithme

Pour tester l'algorithme, on utilise un jeu de données supplémentaires normalisées dont l'état des patients est connu (100 patients). On note *datatest* ces données et *etattest* le vecteur d'état connu pour ces patients. On applique ensuite l'algorithme sur chaque élément de ce jeu de données pour une valeur de K fixée.
On définit la fonction suivante qui renvoie une matrice appelée "matrice de confusion".

```
1  def test_KNN(datatest,etattest,data,etat,K,nb):
2      etatpredit = []
3      for i in range(len(datatest)):
4          res = KNN(data,etat,datatest[i],K,nb)
5          etatpredit.append(res)
6
7      mat = np.zeros((nb,nb))
8      for i in range(len(etattest)):
9          mat[etattest[i],etatpredit[i]] += 1
10     return mat
```

On obtient pour $K = 8$ la matrice suivante : $\begin{pmatrix} 23 & 4 & 7 \\ 7 & 11 & 1 \\ 5 & 2 & 40 \end{pmatrix}$.

Q15. Indiquer l'information apportée par la diagonale de la matrice. Exploiter les valeurs de la première ligne de cette matrice en expliquant les informations que l'on peut en tirer. Faire de même avec la première colonne. En déduire à quoi sert cette matrice.

On peut également tracer l'efficacité de l'algorithme pour différentes valeurs de K sur le jeu de données test (100 éléments sur un échantillon de 310 données). On trace le pourcentage de réussite en fonction de la valeur de K sur la **figure 4**.

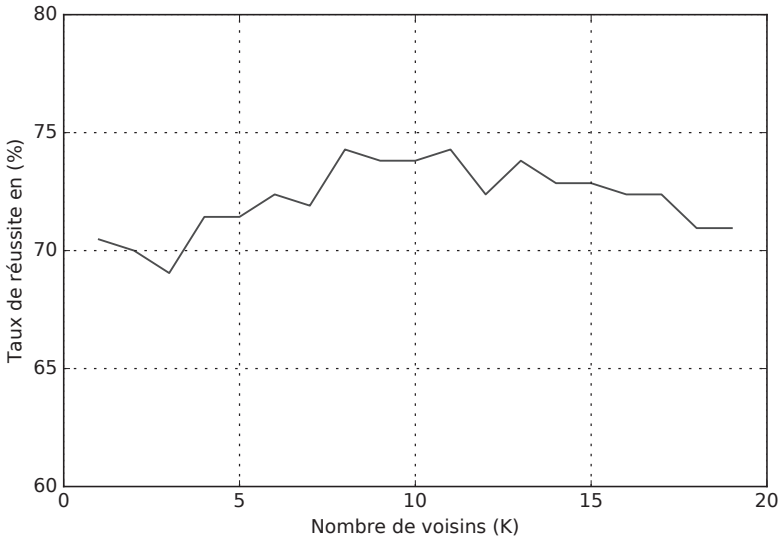


Figure 4 – Pourcentage de réussite de l'algorithme en fonction de K

Q16. Commenter la courbe obtenue et critiquer l'efficacité de l'algorithme.

III.2 - Méthode de classification naïve bayésienne

Une autre méthode simple à mettre en oeuvre et qui fournit de bons résultats, malgré l'hypothèse forte utilisée, est la classification naïve bayésienne. Cette méthode d'apprentissage est supervisée également. La méthode repose sur le théorème de Bayes qui suppose l'indépendance probabilistique des caractéristiques d'un groupe donné.

On suppose que chaque attribut i (colonne de la matrice de données `data`) est modélisable par une variable aléatoire notée X_i pour $i \in [\![0, n-1]\!]$. L'appartenance d'une donnée à un groupe est modélisée par la variable aléatoire Y, dont les valeurs discrètes sont $y_0 = 0$ pour un sujet sain, $y_1 = 1$ pour un sujet atteint d'une hernie discale et $y_2 = 2$ pour un sujet atteint de spondylolisthésis.

L'appartenance de la donnée k (k^{e} ligne de la matrice de données) est connue et indiquée dans le vecteur `etat`.

Le théorème de Bayes permet de déterminer la probabilité qu'une donnée appartienne à un groupe y_j connaissant ses attributs x_i

$$P(Y = y_j | X_0 = x_0, X_1 = x_1, ..., X_{n-1} = x_{n-1}) = \frac{P(Y = y_j) \times P(X_0 = x_0, X_1 = x_1, ..., X_{n-1} = x_{n-1} | Y = y_j)}{P(X_0 = x_0, X_1 = x_1, ..., X_{n-1} = x_{n-1})}$$

où :

- $P(Y = y_j | X_0 = x_0, X_1 = x_1, ..., X_{n-1} = x_{n-1})$ est la probabilité d'appartenir au groupe y_j sachant que les différentes variables aléatoires X_i prennent respectivement les valeurs x_i,
- $P(Y = y_j)$ est la probabilité d'appartenir au groupe y_j,
- $P(X_0 = x_0, X_1 = x_1, ..., X_{n-1} = x_{n-1} | Y = y_j)$ est la probabilité que les différentes variables aléatoires X_i prennent respectivement les valeurs x_i sachant que la donnée appartient au groupe y_j,
- $P(X_0 = x_0, X_1 = x_1, ..., X_{n-1} = x_{n-1})$ est la probabilité que les différentes variables aléatoires X_i prennent respectivement les valeurs x_i.

L'hypothèse naïve bayésienne suppose que tous les attributs sont indépendants et donc que :

$$P(X_0 = x_0, X_1 = x_1, ..., X_{n-1} = x_{n-1} | Y = y_j) = \prod_i P(X_i = x_i | Y = y_j).$$

Comme le dénominateur $P(X_0 = x_0, X_1 = x_1, ..., X_{n-1} = x_{n-1})$ est indépendant du groupe considéré et donc constant, on ne considère que le numérateur :

$$P(Y = y_j) \times P(X_0 = x_0, X_1 = x_1, ..., X_{n-1} = x_{n-1} | Y = y_j).$$

Pour déterminer le groupe le plus probable auquel appartient une donnée z à classer, représentée par le n-uplet $(z_0, z_1, ..., z_{n-1})$, on choisit la probabilité maximale $P(Y = y_j) \times \prod_i P(X_i = z_i | Y = y_j)$ parmi les j groupes.

Des lois de probabilités diverses sont utilisées pour estimer $P(X_i = z_i | Y = y_j)$; nous utiliserons une loi de distribution gaussienne.

Apprentissage

La première étape consiste à séparer les données selon les groupes auxquels elles appartiennent. On réutilise pour cela la fonction `separationParGroupe(data,etat)` définie à la question **Q6**.

Pour calculer la probabilité conditionnelle $P(X_i = z_i | Y = y_j)$ d'une donnée z représentée par le n-uplet $(z_0, z_1, ..., z_{n-1})$, on utilise une distribution gaussienne de la forme

$$P(X_i = z_i | Y = y_j) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(z_i - \mu_{x_i,y_j})^2}{2\sigma^2_{x_i,y_j}}\right)$$

où μ_{x_i,y_j} et $\sigma^2_{x_i,y_j}$ sont la moyenne et la variance de la variable aléatoire X_i, estimées à partir des valeurs d'attribut des données de la i-ième colonne du tableau `data` pour le groupe correspondant à y_j.

On rappelle que, pour un vecteur x de dimension n, $\mu = \frac{\sum_{i=0}^{i=n-1} x_i}{n}$ et $\sigma^2 = \frac{\sum_{i=0}^{i=n-1} (x_i - \mu)^2}{n}$.

Q17. Écrire deux fonctions de complexité linéaire `moyenne(x)` et `variance(x)` permettant de renvoyer la moyenne et la variance d'un vecteur x de dimension quelconque.

Q18. Proposer une fonction `synthese(data,etat)` qui renvoie une liste composée de doublets [moyenne,variance] pour chaque attribut de la matrice `data` en les regroupant selon les valeurs du vecteur `etat` :
$[[[\mu_{x_0,y_0}, \sigma_{x_0,y_0}], [\mu_{x_1,y_0}, \sigma_{x_1,y_0}], \ldots, [\mu_{x_5,y_0}, \sigma_{x_5,y_0}]],$
$[[\mu_{x_0,y_1}, \sigma_{x_0,y_1}], [\mu_{x_1,y_1}, \sigma_{x_1,y_1}], \ldots, [\mu_{x_5,y_1}, \sigma_{x_5,y_1}]],$
$[[\mu_{x_0,y_2}, \sigma_{x_0,y_2}], [\mu_{x_1,y_2}, \sigma_{x_1,y_2}], \ldots, [\mu_{x_5,y_2}, \sigma_{x_5,y_2}]]].$

Cette fonction appliquée au tableau `data` complet (non normalisé) renvoie une liste qui contient les valeurs suivantes :

	x_1	x_2	x_3	x_4	x_5	x_6
etat=0	[51.68, 12.3]	[12.82, 6.74]	[43.54, 12.29]	[38.86, 9.57]	[123.89, 8.96]	[2.18, 6.27]
etat=1	[47.63, 10.6]	[17.39, 6.95]	[35.46, 9.68]	[30.23, 7.49]	[116.47, 9.27]	[2.48, 5.48]
etat=2	[71.51, 15.05]	[20.74, 11.46]	[64.11, 16.34]	[50.76, 12.27]	[114.51, 15.52]	[51.89, 39.97]

Prédiction

On considère pour la deuxième étape une donnée z à classer représentée par le n-uplet $(z_0, z_1, ..., z_{n-1})$. Pour réaliser la prédiction, il faut tout d'abord calculer la probabilité d'appartenance à un groupe selon la loi gaussienne choisie $P(X_i = z_i|Y = y_j)$ en fonction de l'attribut considéré, puis multiplier ces probabilités pour un même groupe y_j pour obtenir $\prod_i P(X_i = z_i|Y = y_j)$.

Q19. Écrire une fonction `gaussienne(a,moy,v)` qui calcule la probabilité selon une loi gaussienne de moyenne *moy* et de variance *v* pour un élément *a* de **R**.

Q20. Déduire de la description de l'algorithme de classification naïve bayésienne une fonction `probabiliteGroupe(z,data,etat)` prenant en argument le vecteur `z` qui contient le n-uplet de la donnée z, le tableau des données connues `data` et le vecteur d'appartenance à un groupe de chacune de ces données `etat`. Cette fonction renvoie la probabilité d'appartenance à chacun des $nb = 3$ groupes sous la forme d'une liste de trois valeurs. On utilisera la fonction `synthese(data,etat)` et la fonction `gaussienne(a,moy,v)`.

Sur le patient dont les caractéristiques sont les suivantes : 48,26, 16,42, 36,33, 31,84, 94,88, 28,34, on obtient les probabilités suivantes : appartenance au groupe 0 : $1{,}1 \cdot 10^{-15}$, appartenance au groupe 1 : $7{,}4 \cdot 10^{-16}$, appartenance au groupe 2 : $5{,}4 \cdot 10^{-13}$.
La décision de l'appartenance à un groupe particulier est prise en déterminant le maximum parmi les probabilités de groupes déterminées.

Q21. Écrire une fonction `prediction`, dont vous préciserez les arguments, qui renvoit le numéro du groupe auquel appartient un élément z.

Usuellement la méthode naïve bayésienne est utilisée dans un espace logarithmique ; c'est-à-dire qu'au lieu de calculer la probabilité d'appartenance à un groupe comme vu précédemment, on calcule le logarithme de cette quantité.

Q22. Proposer une explication qui justifie cette utilisation du logarithme.

L'algorithme mis en place peut être testé sur le jeu de 100 données test utilisé pour l'algorithme KNN dont on connaît déjà l'appartenance à chacun des groupes. On obtient alors la matrice de confusion suivante : $\begin{pmatrix} 23 & 9 & 8 \\ 9 & 10 & 1 \\ 10 & 1 & 49 \end{pmatrix}$.

Q23. Calculer le pourcentage de réussite de la méthode KNN, à partir de la matrice de confusion pour $K = 8$ (page 8), ainsi que celui de la méthode naïve bayésienne à partir de la matrice ci-dessus. Discuter de la pertinence de chacune des deux méthodes sur l'exemple traité.

FIN

ANNEXE

Rappels des syntaxes en Python

Remarque : sous Python, l'import du module numpy permet de réaliser des opérations pratiques sur les tableaux : from numpy import *. Les indices de ces tableaux commencent à 0.

	Python
tableau à une dimension	`L=[1,2,3]` (liste) `v=array([1,2,3])` (vecteur)
accéder à un élément	`v[0]` renvoie 1 (`L[0]` également)
ajouter un élément	`L.append(5)` uniquement sur les listes
tableau à deux dimensions (matrice)	`M=array([[1,2,3],[3,4,5]])`
accéder à un élément	`M[1,2]` donne 5
extraire une portion de tableau (2 premières colonnes)	`M[:,0:2]`
extraire la colonne i	`M[:,i]`
extraire la ligne i	`M[i,:]`
tableau de 0 (2 lignes, 3 colonnes)	`zeros((2,3))`
dimension d'un tableau `T` de taille (i, j)	`T.shape` donne `[i,j]`

produit matrice-vecteur	`a = array([[1,2,3],[4,5,6],[7,8,9]])`
	`b = array([1,2,3])`
	`print(a.dot(b))`
	`>>> array([14,32,50])`

séquence équirépartie quelconque de 0 à 10.1 (exclus) par pas de 0.1	`arange(0,10.1,0.1)`
définir une chaîne de caractères	`mot='Python'`
taille d'une chaîne	`len(mot)`
extraire des caractères	`mot[2:7]`
boucle For	`for i in range(10):` `  print(i)`

condition If	`if (i>3):`
	`  print(i)`
	`else:`
	`  print('hello')`

définir une fonction qui possède un argument et renvoie 2 résultats	`def f(param):`
	`  a = param`
	`  b = param*param`
	`  return a,b`

tracé d'une courbe de deux listes de points x et y	`plot(x,y)`
tracé d'une courbe de trois listes de points x, y et z	`gca(projection='3d').plot(x,y,z)`
ajout d'un titre sur les axes d'une figure	`xlabel(texte)` `ylabel(texte)`
ajout d'un titre principe sur une figure	`title(texte)`

IMPRIMERIE NATIONALE – 19 1157 – D'après documents fournis

Travailler les chapitres de ce sujet

Prepa Help suit le programme de PSI chapitre par chapitre : cours détaillé, méthodes, pièges, exercices et problèmes corrigés pas à pas, tirés des concours. L'essai gratuit donne accès aux chapitres offerts de ta filière, sans carte bancaire.

Chapitres liés à ce sujet

- Bases de Python : expressions, types, instructions, fonctions
- Architecture de l'ordinateur et représentation des nombres
- Fichiers texte et lecture de données
- Apprentissage automatique : k plus proches voisins et k-moyennes
- Récursivité
- Tris par sélection, à bulles, par insertion et par comptage
- Variables aléatoires discrètes



Ouvrir l'essai gratuit

Scanne le code ou clique sur le bouton.

prepahelp.com/#essai

Une question sur le concours ou sur ta filière : **écrire sur WhatsApp**

Les autres années et épreuves : prepahelp.com/annales/ccinp/psi