



Concours Biologie
Epreuve d'Informatique

Date : Mardi 11 Juin 2019 Heure : 12 H Durée : 2 H Nbr pages : 7

Barème : PROBLEME 1 : 10 points
PROBLEME 2 : 10 points

DOCUMENTS NON AUTORISES
L'USAGE DES CALCULATRICES EST INTERDIT
II FAUT RESPECTER IMPERATIVEMENT LES NOTATIONS DE L'ENONCE
VOUS POUVEZ EVENTUELLEMENT UTILISER LES FONCTIONS PYTHON
DECRIRES A L'ANNEXE (PAGE 7)

Présentation Générale

L'apprentissage automatique supervisé permet d'élaborer des programmes capables d'apprendre automatiquement à partir d'un ensemble de données (**dataset**) comportant des valeurs d'observations et les décisions qui leur sont associées. Il a ainsi pour objectif de produire un modèle capable de prédire la décision à prendre pour des nouvelles valeurs d'observations.

Par exemple, on peut donner au programme d'apprentissage un ensemble de données contenant les observations relatives à des patients (**tension** artérielle, **âge** du patient et présence d'une **tachycardie** sinusoïdale) et expliquer lesquels ont un risque élevé de crise cardiaque (**décision** = 1) et lesquels ont un risque très faible (**décision** = 0). Comme illustré dans le tableau suivant, les lignes représentent les valeurs des observations relatives à un patient et la dernière colonne représente la décision finale. Une fois l'apprentissage terminé, à partir des observations d'un nouveau patient, le programme, appelé aussi **classifieur**, devra déterminer automatiquement la décision à prendre (0 ou 1).

tension	âge	tachycardie	décision
110	50	1	0
119	36	0	0
82	72	0	1
81	70	0	1
56	50	1	1

Table 1 - Exemple de données d'apprentissage.

PROBLEME 1

L'objectif de ce problème est d'implémenter des fonctions utilitaires pour la construction d'un classifieur automatique à partir des données d'apprentissage.

Description

On dispose d'un fichier texte nommé '**donneesObservations.txt**' contenant les observations relatives à un ensemble de patients. La première ligne est réservée aux noms des observations, séparés par '#', les autres lignes contiennent les valeurs de ces observations, séparées par '#', où chaque ligne est relative à un patient.

Exemple

Les données de la table 1 sont représentées dans le fichier comme suit :

```
tension#age#tachycardie#decision
110#50#1#0
119#36#0#0
82#72#0#1
81#70#0#1
56#50#1#1
```

DSET est un dictionnaire représentant les données d'apprentissage, construit à partir du fichier '**donneesObservations.txt**', dont :

- chaque clé est le nom d'une observation, de type chaîne de caractères ;
- chaque valeur est une liste contenant les valeurs, de type entier, prises par l'observation.

Exemple

Les données d'apprentissage illustrées dans la table 1 sont représentées par le dictionnaire suivant :

```
DSET={'tension': [110,119,82,81,56], 'age': [50,36,72,70,50],
'tachycardie': [1,0,0,0,1], 'decision': [0,0,1,1,1]}
```

NB :

DSET['tension'][0], **DSET['age'][0]**, **DSET['tachycardie'][0]** et **DSET['decision'][0]** décrivent respectivement les valeurs de la tension, l'âge, la tachycardie et la décision associées au premier patient. La taille de chaque liste associée à chaque clé représente le nombre de patients de **DSET**.

Travail demandé

N.B : Dans la suite, les fonctions demandées seront écrites en langage Python en respectant la nomenclature suivante.

Nom	Type	Description
DSET DSET1 DSET2	<i>dict</i>	Dictionnaires représentants des données d'apprentissage
fname	<i>str</i>	Nom physique du fichier des données d'apprentissage ayant la même structure que ' donneesObservations.txt '
obs	<i>str</i>	Chaîne de caractères représentant le nom d'une observation
DISTR	<i>dict</i>	Dictionnaire représentant la distribution des probabilités des décisions
QUAL	<i>Dict</i>	Dictionnaire indiquant les observations qualitatives et les observations quantitatives
S	<i>float</i>	Seuil de découpage
Lc	<i>list</i>	Liste de tuples où chaque tuple est formé par la valeur d'une observation et la valeur de la décision associée
vBest	<i>float</i>	Impureté donnant le meilleur seuil
sBest	<i>float</i>	Meilleur seuil
LSeuil	<i>list</i>	Liste des seuils possibles de découpage

1. Ecrire une fonction **LoadFile** qui prend en paramètre une chaîne caractères **fname** contenant le nom d'un fichier de même format que '**donneesObservations.txt**', récupère et retourne son contenu dans le dictionnaire **DSET**.
2. Ecrire une fonction nommée **CountValues** qui prend en paramètres **DSET** et une chaîne de caractères **obs** contenant le nom d'une observation et retourne le nombre de valeurs distinctes pour **obs**.

Exemple : L'appel de **CountValues(DSET, 'age')** retourne 4.

3. Ecrire une fonction nommée **EvalDistr** qui prend en paramètres **DSET** et retourne un dictionnaire **DISTR** où :
 - chaque clé, notée **d**, est une valeur de la décision (0 ou 1);
 - chaque valeur est la probabilité d'apparition de **d** dans **DSET** exprimée par :

$$p(\text{decision} = \mathbf{d} | \mathbf{DSET}) = \frac{\text{nombre de décisions de DSET égales à } \mathbf{d}}{\text{nombre de décisions de DSET}} \quad (\text{Eq.1})$$

Si **DSET** est vide, **DISTR** = {0:0.5, 1:0.5}.

Exemple : L'appel de **EvalDistr(DSET)** retourne {0: 0.4, 1: 0.6}.

4. Ecrire une fonction, nommée **IsPure**, qui prend en paramètre **DSET** et retourne un booléen égal à **True** si **DSET** est pur et **False** sinon, sachant que **DSET** est pur, si et seulement si, toutes les valeurs des décisions sont égales.
5. Ecrire une fonction **IsQualitative** qui prend en paramètre **DSET** et retourne un dictionnaire **QUAL** dont :
 - les clés sont les mêmes que **DSET** ;
 - chaque valeur est un booléen égal à **True** si la clé est une mesure qualitative (0 ou 1) et **False** si elle est quantitative (valeur numérique pas nécessairement binaire).

Exemple : L'appel **IsQualitative(DSET)** retourne :

```
{'tension':False, 'age':False, 'tachycardie':True, 'decision':True}
```

6. Ecrire une fonction **Cut** qui permet de découper **DSET** en deux dictionnaires. Elle prend en paramètres :
 - un dictionnaire **DSET**,
 - une chaîne de caractères **obs** correspondant au nom d'une observation,
 - un réel **S** représentant le seuil de découpage et prenant la valeur 0.5 par défaut.

Cette fonction retourne un tuple formé par deux listes **L1** et **L2** :

- **L1** contient les dictionnaires **DSET1** et **DSET2** tels que :
 - **DSET1** et **DSET2** doivent contenir les mêmes clés que **DSET**, exceptée **obs** ;
 - **DSET1** est formée par les patients de **DSET** où la valeur de l'observation **obs**, notée **Vobs**, est supérieure ou égale à **S** ;
 - **DSET2** est formée par les autres patients de **DSET** ;
- **L2** contient les probabilités p_1 et p_2 décrites par les équations (Eq.2) et (Eq.3) :

$$p_1 = p(\mathbf{Vobs} \geq \mathbf{S} | \mathbf{DSET}) = \frac{\text{nombre de patients de DSET1}}{\text{nombre de patients de DSET}} \quad (\text{Eq.2})$$

$$p_2 = p(\mathbf{Vobs} < \mathbf{S} | \mathbf{DSET}) = \frac{\text{nombre de patients de DSET2}}{\text{nombre de patients de DSET}} = 1 - p_1 \quad (\text{Eq.3})$$

Exemple : L'appel **Cut(DSET, 'age', 70)** retourne le tuple suivant :

```
([{'tension':[82,81], 'tachycardie':[0,0], 'decision':[1,1]},
  {'tension':[110,119,56], 'tachycardie':[1,0,1], 'decision':[0,0,1]}],
 [0.4, 0.6])
```

7. Ecrire une fonction **Impurity** qui prend en paramètres **DSET**, **obs** et un seuil **S** (ayant par défaut la valeur 0.5). Cette fonction retourne un réel mesurant la qualité du découpage de **DSET** en deux dictionnaires **DSET1** et **DSET2**, calculé selon l'équation (Eq.4).

$$p_1 \times \min_{d \in \{0,1\}} (p(\text{decision} = d | \text{DSET1})) + p_2 \times \min_{d \in \{0,1\}} (p(\text{decision} = d | \text{DSET2})) \quad (\text{Eq.4})$$

où :

- **DSET1** et **DSET2** résultent du découpage du dictionnaire **DSET** selon **obs** et **S** ;
- p_1 et p_2 sont décrites par les équations (Eq.2) et (Eq.3) ;
- $p(\text{decision} = d | \text{DSET})$ est donnée par l'équation (Eq.1).

Exemple

L'impureté du découpage illustré dans l'exemple de la question 6 est donnée par :

$$\text{Impurity}(\text{DSET}, 'age', 70) = 0.4 \times \min(0, 1) + 0.6 \times \min\left(\frac{2}{3}, \frac{1}{3}\right) = 0.2$$

8. Ecrire une fonction **SortObs** qui prend en paramètres **DSET** et **obs**, puis retourne une liste **Lc** obtenue en effectuant les étapes suivantes :

- à partir des deux clés **obs** et 'decision', créer la liste **Lc** qui est une liste de tuples (**v**,**d**) où **v** est une valeur de l'observation **obs** et **d** la valeur de la décision associée,
- puis, trier **Lc** par ordre croissant suivant les valeurs de **obs**.

Exemple : L'appel de **SortObs**(**DSET**, 'age') retourne :

[(36, 0), (50, 0), (50, 1), (70, 1), (72, 1)]

9. Ecrire une fonction **BestCut** qui prend en paramètres **DSET**, **obs** et **QUAL** et retourne un tuple (**vBest**, **sBest**) où **vBest** est la meilleure impureté et **sBest** est le meilleur seuil de découpage associé. **vBest** et **sBest** sont déterminés comme suit :

- Cas des observations qualitatives :
 - **sBest**= 0.5
 - **vBest**= **Impurity**(**DSET**,**obs**).
- Cas des observations quantitatives :
 - Créer **Lc**, la liste formée par les tuples (v_i, d_i) triée par ordre croissant en utilisant la fonction **SortObs**.
 - Créer à partir de **Lc**, une liste **LSeuil**, contenant les seuils possibles de découpage de **DSET**, selon les conditions suivantes :
 - Si **DSET** est pur alors **LSeuil** contient la valeur maximale des v_i de la liste **Lc**.
 - Si **DSET** est impur alors **LSeuil** est formée par les valeurs $\frac{v_i + v_{i+1}}{2}$ pour les tuples adjacents (v_i, d_i) et (v_{i+1}, d_{i+1}) de **Lc** avec $d_i \neq d_{i+1}$.
 - Déterminer le meilleur seuil **sBest** de la liste **LSeuil** associée à la valeur minimale de l'impureté, **vBest**.

PROBLEME 2

Soit la base de données relationnelle intitulée 'classifieurs.db' contenant la description des données d'apprentissage avec les classifieurs automatiques créés autour de ces données, représentée par le schéma relationnel suivant :

▪ **DataSet** (ds_id, ds_name, nb_instances, format, ds_description)

La table DataSet décrit les données d'apprentissage, avec les colonnes :

- ds_id : identifiant du dataset (entier), *clé primaire*.
- ds_name : nom du dataset (chaîne de caractères).
- nb_instances : nombre de lignes du dataset (entier).
- format : le format des données du dataset (chaîne de caractères).
- ds_description : le sommaire du dataset (chaîne de caractères).

▪ **Classifieur** (cls_id, cls_description, error_rate, language, ds_id)

La table Classifieur décrit un classifieur automatique construit à partir d'un dataset, avec les colonnes :

- cls_id : identifiant du classifieur (chaîne de caractère), *clé primaire*.
- cls_description : description du classifieur (chaîne de caractères).
- error_rate : un nombre réel entre 0 et 1 qui décrit le pourcentage des données où les décisions prédites par le classifieur sont différentes de la réalité.
- language : nom du langage de programmation utilisé pour implémenter le classifieur (chaîne de caractère).
- ds_id : identifiant du dataset utilisé pour l'apprentissage du classifieur, *clé étrangère*.

▪ **Method** (m_name, category, m_description)

La table Method décrit les méthodes utilisées dans le domaine de l'apprentissage automatique pour la construction des classifieurs, avec les colonnes :

- m_name : le nom de la méthode (chaîne de caractères), *clé primaire*.
- category : la catégorie de la méthode (chaîne de caractères).
- m_description : la description de la méthode (chaîne de caractères).

▪ **Combine** (cls_id, m_name, description)

La table Combine décrit les méthodes de classification utilisées dans chaque classifieur, de clé primaire (cls_id, m_name), avec les colonnes :

- cls_id : identifiant du classifieur (chaîne de caractère), *clé étrangère*.
- m_name : le nom de la méthode (chaîne de caractère), *clé étrangère*.
- description : stratégie d'intégration de la méthode dans le classifieur (chaîne de caractères).

Partie 1 : algèbre relationnelle

Exprimer en algèbre relationnelle les requêtes suivantes :

1. Déterminer les identifiants, les noms et les descriptions des datasets au format 'csv'.
2. Déterminer les descriptions des classifieurs implémentés en 'Python' et qui utilisent la méthode de catégorie 'KNN'.

Partie 2 : SQL

Exprimer en SQL les requêtes suivantes :

3. Modifier dans la table DataSet les formats 'txt' par 'docx'.
4. Supprimer les classifieurs implémentés avec le langage de programmation 'Pascal'.
5. Donner les identifiants des datasets pour lesquels il existe au moins un classifieur avec un taux d'erreur < 0.3 (error_rate).
6. Donner les noms des datasets contenant le plus grand nombre d'instances.
7. Donner les identifiants et les noms des datasets où tous les classifieurs sont implémentés en 'Python'.
8. Donner les identifiants des datasets jamais utilisés par des classifieurs.
9. Donner le nombre de classifieurs dont le taux d'erreur est supérieur à la moyenne des taux d'erreurs de tous les classifieurs.
10. Donner pour chaque dataset le nombre de méthodes de classification utilisées.
11. Donner les informations des datasets associés aux classifieurs produisant le taux d'erreur minimal.
12. Donner les identifiants et les noms des datasets autour desquels il existe au moins trois classifieurs implémentés en 'Java'.

Partie 3 : sqlite3

On dispose d'un dictionnaire **dict_M** contenant des informations relatives aux méthodes utilisées dans le domaine d'apprentissage, où :

- chaque clé est le nom de la méthode
- chaque valeur est une liste contenant respectivement la catégorie et la description.

13. Ecrire les instructions python permettant de :

- Importer le module sqlite3.
- Se connecter à la base de données 'classifieurs.db'.
- Créer le curseur **cur** d'exécution.
- Créer la table Method en exprimant les contraintes mentionnées ci-dessus.
- remplir la table Method à partir du dictionnaire **dict_M**.
- tracer la courbe dont les abscisses sont les identifiants des datasets et les ordonnées sont les taux d'erreur moyen des classifieurs associés.

ANNEXE – Quelques fonctions Python

Opérations utiles sur les itérables (str, tuple, list, dict, etc.)

- **len(it)** retourne le nombre d'éléments de l'itérable it.
- **range(d,f)** retourne la séquence de valeurs entières successives comprises entre d et f exclu.
- **min(it)** (resp. **max(it)**) retourne la valeur minimale (resp. maximale) de l'itérable it.
- **x in it** (resp. **x not in it**) vérifie si x appartient à it (resp. n'appartient pas).
- **sorted(it)** retourne une liste contenant les éléments de it dans l'ordre croissant.
- **lst.sort()** trie la liste lst dans l'ordre croissant.
- **lst.count(val)** retourne le nombre d'occurrences de val dans la liste lst.
- **lst.append(val)** ajoute val à la fin de la liste lst.
- **lst.remove(val)** supprime la première occurrence de val dans la liste lst.
- **lst.index(val)** retourne l'indice de la première occurrence de val dans la liste lst.
- **source.split(motif)** retourne une liste formée par des chaînes de caractères résultant du découpage de la chaîne source autour de la chaîne motif.
- **d.values()** retourne un itérable formé par les valeurs du dictionnaire d.
- **d.items()** retourne un itérable de couples (k,v) où k est une clé du dictionnaire d et v est la valeur associée.

Opérations sur les fichiers

- **f=open (nomF,m)** permet d'ouvrir le fichier nomF en mode m où m='r' ou 'w'.
- **f.close()** permet de fermer un fichier.
- **f.read()** permet de lire et retourner le contenu d'un fichier dans une chaîne de caractères.
- **f.readline()** permet de lire et retourner le contenu de la ligne courante d'un fichier dans une chaîne de caractères.
- **f.readlines()** : permet de lire et retourner le contenu de toutes les lignes d'un fichier dans une liste.

Opérations sur le module matplotlib.pyplot

- **import matplotlib.pyplot as plt** permet le chargement du module
- **plt.plot(x,y)** crée la courbe où les abscisses sont décrites par les valeurs de l'itérable x et les ordonnées par ceux de l'itérable y.
- **plt.show()** affiche une fenêtre contenant le résultat du dessin.