



Concours Biologie et Géologie

Corrigé épreuve d'informatique

PROBLEME 1

1. La fonction LoadFile

```
def LoadFile(fname):  
    f = open(fname, 'r')  
    ch = f.readline()  
    Cle = ch[:len(ch)-1].split('#')  
    DSET = {i:[] for i in Cle}  
    while True:  
        ch = f.readline()  
        if ch=='':  
            break  
        else:  
            Val = ch[:len(ch)-1].split('#')  
            for i in range(len(Val)):  
                DSET[Cle[i]].append(Val[i])  
    f.close()  
    return DSET
```

2. La fonction CountValues

```
def CountValues(DSET, obs):  
    return len(set(DSET[obs]))
```

3. La fonction EvalDistr

```
def EvalDistr(DSET):  
    if DSET=={}:  
        DISTR={0:0.5,1:0.5}  
    else:  
        l=DSET['decision']  
        p=l.count(0)/len(l)  
        DISTR={0:p,1:1-p}  
    return DISTR
```

4. La fonction IsPure 5 pts

```
def IsPure(DSET):  
    DISTR=EvalDistr(DSET)  
    return 1 in DISTR.values() #(DISTR[0]==1 or DISTR[1]==1)
```

5. La fonction IsQualitative

```
def IsQualitative(DSET):
    QUAL={i:True for i in DSET}
    for i in DSET:
        for j in DSET[i]:
            if j not in [0,1]:
                QUAL[i]=False
                break
    return QUAL
```

6. La fonction Cut

```
def Cut(DSET,obs,S=0.5):
    L=list(DSET.keys())
    L.remove(obs)
    DSET1={i:[] for i in L}
    DSET2={i:[] for i in L}
    for i in range(len(DSET[obs])):
        if DSET[obs][i] <= S:
            for j in DSET1:
                DSET1[j].append(DSET[j][i])
        else:
            for j in DSET2:
                DSET2[j].append(DSET[j][i])
    p1=len(DSET1[L[0]])/len(DSET[L[0]])
    p2=1-p1
    return ([DSET1,DSET2],[p1,p2])
```

7. La fonction Impurity

```
def Impurity(DSET,obs,S=0.5):
    t=Cut(DSET,obs,S)
    DSET1,DSET2=t[0][0],t[0][1]
    p1,p2=t[1][0],t[1][1]
    d1,d2=EvalDistr(DSET1),EvalDistr(DSET2)
    return p1*min(d1.values())+p2*min(d2.values())
```

8. La fonction SortObs

```
def SortObs(DSET,obs):
    Lc = [(DSET[obs][i],DSET['decision'][i]) for i in
range(len(DSET[obs]))]
    Lc.sort()
    return Lc
```

9. La fonction BestCut

```
def BestCut(DSET,obs,QUAL):
    if QUAL[obs]:
        return (0.5,Impurity(DSET,obs))
    else:
        Lc=SortObs(DSET,obs)
        if Ispure(DSET):
            LSeuil=[max(Lc)[0]]
        else:
            LSeuil=[]
            for i in range(len(Lc)-1):
                if Lc[i][1]!=Lc[i+1][1]:
                    LSeuil.append((Lc[i]+Lc[i+1])/2)
```

```

vBest=Impurity(DSET,obs,LSeuil[0])
sBest=Lseuil[0]
for i in range(1,len(LSeuil)):
    v=Impurity(DSET,obs,LSeuil[i])
    if v < vBest:
        vBest=v
        sBest= Lseuil[i]
return(vBest, sBest)

```

PROBLEME 2

Partie 1

1. $\pi_{ds_id, ds_name, ds_description} \left(\sigma_{format='csv'}(DataSet) \right)$
2. $R = Method \bowtie_{m_name} Combine \bowtie_{cls_id} Classifieur$
 $\pi_{cls_description} \left(\sigma_{language='python' \text{ et } categorie='KNN'}(R) \right)$

Ou

$\pi_{cls_description}(\sigma_{language='Python' \text{ et } category='KNN'}(Classifieur \bowtie_{cls_id} Combine \bowtie_{m_name} Method))$

Partie 2

3. UPDATE DataSet SET format = 'docx' WHERE format= 'txt'
4. DELETE FROM Classifieur WHERE langage= 'Pascal'
5. SELECT ds_id FROM Classifieur WHERE error_rate < 0.3
6.
 SELECT ds_name FROM DataSet WHERE
 nb_instances= (SELECT max(nb_instances) FROM DataSet)
7.
 SELECT ds_id, ds_name FROM DataSet
 EXCEPT
 SELECT ds_id, ds_name FROM DataSet D, Classifieur C
 WHERE D.ds_id = C.ds_id AND C.langage <> 'python'
8.
 SELECT ds_id FROM DataSet
 EXCEPT
 SELECT ds_id FROM Classifieur
9.
 SELECT COUNT(cls_id) FROM Classifieur
 WHERE error_rate > (SELECT AVG(error_rate) FROM Classifieur);
 SELECT ds_id FROM Classifieur
10.
 SELECT ds_id, COUNT(M.m_name)
 FROM DataSet D, Classifieur C, Method M, Combine CM
 WHERE D.ds_name=C.ds_name
 AND C.cls_id = CM.cls_id
 AND M.m_name = CM.m_name
 GROUP BY ds_id;

- 11.
- ```
SELECT * FROM DataSet WHERE ds_id
 IN
 (SELECT ds_id FROM Classifieur WHERE
 Error_rate = (SELECT MIN(error_rate) FROM classifieur))
```
- 12.
- ```
SELECT ds_id, ds_name FROM DataSet D, Classifieur C
                        WHERE D.ds_name=C.ds_name
                        AND C.langage = 'java'
                        GROUP BY ds_id HAVING COUNT(*) >=3
```

Partie 3

- 13.
- ```
import sqlite3 as sq
bd = sq.connect('classifieurs.db')
cur = bd.cursor()

cur.execute('CREATE TABLE Method (m_name text primary key, categorie text,
m_description text)')

for c in dict_M:
 cur.execute('INSERT INTO Method VALUES (?, ?, ?)', (cle, dict_M[c][0],
dict_M[c][1]))

cur.execute('SELECT ds_id, avg(error_rate) FROM DataSet D JOIN Classifieur C ON D.ds_id
= C.ds_id GROUP BY ds_id')

L = cur.fetchall()
X = [i[0] for i in L]
Y = [i[1] for i in L]

import matplotlib.pyplot as plt
plt.plot(X,Y)
plt.show()
```