Representation des connaissances et raisonnement 2

« Rapport du TP °1 : Théorie des fonctions de croyance»

CHIKH Khadidja

Master 2 SII
Groupe 1

<u>02/12/2019</u>

Introduction :

Dans ce TP , il a été demandé de modéliser un ou plusieurs exercices de la série de TD « théorie des fonctions de croyance » en exploitant une des toolbox existantes.
Pour cela, nous avons utilisé la bibliothèque **py_dempster_shafer (0.7)**

Théorie des fonctions de croyances/ Dempster Shafer ;

La théorie de l'évidence, de croyance ou la théorie du raisonnement plausible est une théorie mathématiques basée sur la notion de « preuve », introduite par Arthur Dempster en 1968 et développée par Glenn Shafer en 1976. C'est une généralisation de l'inférence Bayésienne au traitement de l'incertain, elle permet de manipuler des événements non nécessairement exclusifs. Cette capacité lui confère l'avantage de pouvoir représenter explicitement l'incertitude sur un événement.

La bibliothèque «py_dempster_shafer » :

Fournie par Python, c'est une librairie conçue pour l'interpretation des calculs dans la théorie d'evidence de Dempster Shafer.

Implémentation :

Les figures au dessus illustrent la modélisation de **l'exercice 1** de la série TD, suivie par les résultats (des parties) des calculs des : degré de plausibilité, degré de croyance et combinaisons :

Modélisation :

La modélisation des définitions des stratégies financières données par les trois experts en utilisant la fonction *MassFunction* , ainsi que les calculs concernant chacune en utilisant les fonctions *bel()* et *pl()*:

```
from __future__ import print_function
from pyds import MassFunction
from itertools import product
def exo1():
    print("***************EXO1***************",'\n')
    e1=MassFunction([({'S1'}, 0.2), ({'S3', 'S4'}, 0.3),({'S1','S2', 'S5','S3', 'S4'},0.5)])
    e2=MassFunction([({'S1','S2', 'S5'}, 0.45),({'S3'}, 0.22),({'S1','S2', 'S5','S3', 'S4'},0.33)])
    e3=MassFunction([({'S1'},0.2),({'S2'},0.2), ({'S3'},0.2),({'S4'},0.2),({'S5'},0.2)])
    print("***************CALCUL DES BEL***************",'\n')
    print('bel_e1 =', e1.bel(),'\n')
    print('bel_e2 =', e2.bel(),'\n')
    print('bel_e3 =', e3.bel(),'\n')
    print("***************CALCUL DES PL***************",'\n')
    print('pl_e1 =', e1.pl(),'\n')
    print('pl_e2 =', e2.pl(),'\n')
    print('pl_e3 =', e3.pl(),'\n')
    print("***************COMBINAISON***************",'\n')
    print('Dempster\'s combination rule for e1 & e2 & e3 =',e1 & e3 & e2,'\n')
```

Résultats :

BEL :

```
bel_e1 = {frozenset(): 0.0, frozenset({'S3'}): 0.0, frozenset({'S1'}): 0.2, frozenset({'S5'}): 0.0, frozenset({'S4'}):
0.0, frozenset({'S2'}): 0.0, frozenset({'S3', 'S1'}): 0.2, frozenset({'S3', 'S5'}): 0.0, frozenset({'S4', 'S3'}): 0.3,
frozenset({'S3', 'S2'}): 0.0, frozenset({'S1', 'S5'}): 0.2, frozenset({'S4', 'S1'}): 0.2, frozenset({'S1', 'S2'}): 0.2,
frozenset({'S4', 'S5'}): 0.0, frozenset({'S2', 'S5'}): 0.0, frozenset({'S4', 'S2'}): 0.0, frozenset({'S3', 'S1',
'S5'}): 0.2, frozenset({'S4', 'S3', 'S1'}): 0.5, frozenset({'S3', 'S1', 'S2'}): 0.2, frozenset({'S4', 'S3', 'S5'}):
0.3, frozenset({'S3', 'S2', 'S5'}): 0.0, frozenset({'S4', 'S3', 'S2'}): 0.3, frozenset({'S4', 'S1', 'S5'}): 0.2,
frozenset({'S1', 'S2', 'S5'}): 0.2, frozenset({'S4', 'S1', 'S2'}): 0.2, frozenset({'S4', 'S2', 'S5'}): 0.0,
frozenset({'S4', 'S3', 'S1', 'S5'}): 0.5, frozenset({'S3', 'S1', 'S2', 'S5'}): 0.2, frozenset({'S4', 'S3', 'S1',
'S2'}): 0.5, frozenset({'S4', 'S3', 'S2', 'S5'}): 0.3, frozenset({'S4', 'S1', 'S2', 'S5'}): 0.2, frozenset({'S3', 'S1',
'S5', 'S4', 'S2'}): 1.0}
bel_e2 = {frozenset(): 0.0, frozenset({'S4'}): 0.0, frozenset({'S3'}): 0.22, frozenset({'S1'}): 0.0, frozenset({'S2'}):
0.0, frozenset({'S5'}): 0.0, frozenset({'S4', 'S3'}): 0.22, frozenset({'S4', 'S1'}): 0.0, frozenset({'S4', 'S2'}): 0.0,
frozenset({'S4', 'S5'}): 0.0, frozenset({'S3', 'S1'}): 0.22, frozenset({'S3', 'S2'}): 0.22, frozenset({'S3', 'S5'}):
0.22, frozenset({'S1', 'S2'}): 0.0, frozenset({'S1', 'S5'}): 0.0, frozenset({'S2', 'S5'}): 0.0, frozenset({'S4', 'S3',
'S1'}): 0.22, frozenset({'S4', 'S3', 'S2'}): 0.22, frozenset({'S4', 'S3', 'S5'}): 0.22, frozenset({'S4', 'S1', 'S2'}):
0.0, frozenset({'S4', 'S1', 'S5'}): 0.0, frozenset({'S4', 'S2', 'S5'}): 0.0, frozenset({'S3', 'S1', 'S2'}): 0.22,
frozenset({'S3', 'S1', 'S5'}): 0.22, frozenset({'S3', 'S2', 'S5'}): 0.22, frozenset({'S1', 'S2', 'S5'}): 0.45,
frozenset({'S4', 'S3', 'S1', 'S2'}): 0.22, frozenset({'S4', 'S3', 'S1', 'S5'}): 0.22, frozenset({'S4', 'S3', 'S2',
'S5'}): 0.22, frozenset({'S4', 'S1', 'S2', 'S5'}): 0.45, frozenset({'S3', 'S1', 'S2', 'S5'}): 0.67, frozenset({'S3',
'S1', 'S5', 'S4', 'S2'}): 1.0}
bel_e3 = {frozenset(): 0.0, frozenset({'S4'}): 0.2, frozenset({'S3'}): 0.2, frozenset({'S1'}): 0.2, frozenset({'S2'}):
0.2, frozenset({'S5'}): 0.2, frozenset({'S4', 'S3'}): 0.4, frozenset({'S4', 'S1'}): 0.4, frozenset({'S4', 'S2'}): 0.4,
frozenset({'S4', 'S5'}): 0.4, frozenset({'S3', 'S1'}): 0.4, frozenset({'S3', 'S2'}): 0.4, frozenset({'S3', 'S5'}): 0.4,
frozenset({'S1', 'S2'}): 0.4, frozenset({'S1', 'S5'}): 0.4, frozenset({'S2', 'S5'}): 0.4, frozenset({'S4', 'S3',
'S1'}): 0.6000000000000001, frozenset({'S4', 'S3', 'S2'}): 0.6000000000000001, frozenset({'S4', 'S3', 'S5'}):
0.6000000000000001, frozenset({'S4', 'S1', 'S2'}): 0.6000000000000001, frozenset({'S4', 'S1', 'S5'}):
0.6000000000000001, frozenset({'S4', 'S2', 'S5'}): 0.6000000000000001, frozenset({'S3', 'S1', 'S2'}):
```

PL :

```
pl_e1 = {frozenset(): 0.0, frozenset({'S3'}): 0.8, frozenset({'S1'}): 0.7, frozenset({'S5'}): 0.5, frozenset({'S4'}):
0.8, frozenset({'S2'}): 0.5, frozenset({'S3', 'S1'}): 1.0, frozenset({'S3', 'S5'}): 0.8, frozenset({'S4', 'S3'}): 0.8,
frozenset({'S3', 'S2'}): 0.8, frozenset({'S1', 'S5'}): 0.7, frozenset({'S4', 'S1'}): 1.0, frozenset({'S1', 'S2'}): 0.7,
frozenset({'S4', 'S5'}): 0.8, frozenset({'S2', 'S5'}): 0.5, frozenset({'S4', 'S2'}): 0.8, frozenset({'S3', 'S1',
'S5'}): 1.0, frozenset({'S4', 'S3', 'S1'}): 1.0, frozenset({'S3', 'S1', 'S2'}): 1.0, frozenset({'S4', 'S3', 'S5'}):
0.8, frozenset({'S3', 'S2', 'S5'}): 0.8, frozenset({'S4', 'S3', 'S2'}): 0.8, frozenset({'S4', 'S1', 'S5'}): 1.0,
frozenset({'S1', 'S2', 'S5'}): 0.7, frozenset({'S4', 'S1', 'S2'}): 1.0, frozenset({'S4', 'S2', 'S5'}): 0.8,
frozenset({'S4', 'S3', 'S1', 'S5'}): 1.0, frozenset({'S3', 'S1', 'S2', 'S5'}): 1.0, frozenset({'S4', 'S3', 'S1',
'S2'}): 1.0, frozenset({'S4', 'S3', 'S2', 'S5'}): 0.8, frozenset({'S4', 'S1', 'S2', 'S5'}): 1.0, frozenset({'S3', 'S1',
'S5', 'S4', 'S2'}): 1.0}
pl_e2 = {frozenset(): 0.0, frozenset({'S4'}): 0.33, frozenset({'S3'}): 0.55, frozenset({'S1'}): 0.78,
frozenset({'S2'}): 0.78, frozenset({'S5'}): 0.78, frozenset({'S4', 'S3'}): 0.55, frozenset({'S4', 'S1'}): 0.78,
frozenset({'S4', 'S2'}): 0.78, frozenset({'S4', 'S5'}): 0.78, frozenset({'S3', 'S1'}): 1.0, frozenset({'S3', 'S2'}):
1.0, frozenset({'S3', 'S5'}): 1.0, frozenset({'S1', 'S2'}): 0.78, frozenset({'S1', 'S5'}): 0.78, frozenset({'S2',
'S5'}): 0.78, frozenset({'S4', 'S3', 'S1'}): 1.0, frozenset({'S4', 'S3', 'S2'}): 1.0, frozenset({'S4', 'S3', 'S5'}):
1.0, frozenset({'S4', 'S1', 'S2'}): 0.78, frozenset({'S4', 'S1', 'S5'}): 0.78, frozenset({'S4', 'S2', 'S5'}): 0.78,
frozenset({'S3', 'S1', 'S2'}): 1.0, frozenset({'S3', 'S1', 'S5'}): 1.0, frozenset({'S3', 'S2', 'S5'}): 1.0,
frozenset({'S1', 'S2', 'S5'}): 0.78, frozenset({'S4', 'S3', 'S1', 'S2'}): 1.0, frozenset({'S4', 'S3', 'S1', 'S5'}):
1.0, frozenset({'S4', 'S3', 'S2', 'S5'}): 1.0, frozenset({'S4', 'S1', 'S2', 'S5'}): 0.78, frozenset({'S3', 'S1', 'S2',
'S5'}): 1.0, frozenset({'S3', 'S1', 'S5', 'S4', 'S2'}): 1.0}
pl_e3 = {frozenset(): 0.0, frozenset({'S4'}): 0.2, frozenset({'S3'}): 0.2, frozenset({'S1'}): 0.2, frozenset({'S2'}):
0.2, frozenset({'S5'}): 0.2, frozenset({'S4', 'S3'}): 0.4, frozenset({'S4', 'S1'}): 0.4, frozenset({'S4', 'S2'}): 0.4,
frozenset({'S4', 'S5'}): 0.4, frozenset({'S3', 'S1'}): 0.4, frozenset({'S3', 'S2'}): 0.4, frozenset({'S3', 'S5'}): 0.4,
frozenset({'S1', 'S2'}): 0.4, frozenset({'S1', 'S5'}): 0.4, frozenset({'S2', 'S5'}): 0.4, frozenset({'S4', 'S3',
'S1'}): 0.6000000000000001, frozenset({'S4', 'S3', 'S2'}): 0.6000000000000001, frozenset({'S4', 'S3', 'S5'}):
0.6000000000000001, frozenset({'S4', 'S1', 'S2'}): 0.6000000000000001, frozenset({'S4', 'S1', 'S5'}):
0.6000000000000001, frozenset({'S4', 'S2', 'S5'}): 0.6000000000000001, frozenset({'S3', 'S1', 'S2'}):
```

Combinaison :

```
***************COMBINAISON***************

Dempster's combination rule for e1 & e2 & e3 = {{'S1'}:0.2689655172413793; {'S3'}:0.21674876847290636; {'S2'}:
0.1921182266009852; {'S5'}:0.1921182266009852; {'S4'}:0.13004926108374382}
```

ainsi la stratégie la plus rentable est S1.

Un autre exemple de modélisation avec **l'exercice 3 :**

<u>Modélisation</u>

```python
def exo3():
    print("***************EXO3***************",'\n')
    e1=MassFunction([({'cef'}, 0.4),({'mt'},0.45),({'ct'},0.15)])
    e2=MassFunction([({'mt'}, 0.75),({'ct','cef','mt','on','cb'},0.25)])
    e3=MassFunction([({'mt'},0.35),({'ct'},0.5),({'cb'},0.08),({'on'},0.02),({'ct','cef','cb','mt','on'},0.05)])
    print("***************CALCUL DES BEL***************",'\n')
    print('bel_e1 =', e1.bel(),'\n')
    print('bel_e2 =', e2.bel(),'\n')
    print('bel_e3 =', e3.bel(),'\n')
    print("***************CALCUL DES PL***************",'\n')
    print('pl_e1 =', e1.pl(),'\n')
    print('pl_e2 =', e2.pl(),'\n')
    print('pl_e3 =', e3.pl(),'\n')
    print("***************COMBINAISON***************",'\n')
    print('loi de combinaison de Dempster pour e1 & e2 & e3 =',e1 & e2 & e3,'\n')
    print('combinaison conjonctive non normalisée de e1, e2, and e3 =',e1.combine_conjunctive([e2, e3], normalization=False),'\n')
    print('combinaison disjonctive  de e1, e2, et e3 =',e1.combine_disjunctive([e2, e3]),'\n')
    print('\n***** Cadre de discernement,éléments focaux et noyau*****\n')
    print('Cadre de discernement de e1 =', e1.frame(),'\n')
    print('éléments focaux de e1 =', e1.focal(),'\n')
    print('noyau de e1 =', e1.core(),'\n')
    print('noyaux combinés de e1 et e3 =', e1.core(e3),'\n')
    print('poids de conflict entre e1, e2, and e3 =',e1.conflict([e2, e3]),'\n')
    print('*****Transformation pignistique*****','\n')
    print('Transformation pignistique de e1 =', e1.pignistic(),'\n')
    print('Transformation pignistique de e2 =', e2.pignistic(),'\n')
    print('Transformation pignistique de e3 =', e3.pignistic(),'\n')
    print("*****Mesure d'incertitude de conflit local*****",'\n')
    print('conflit local  de e1 =', e1.local_conflict(),'\n')
    print('entropie de la transformation pignistique de e3 =',e3.pignistic().local_conflict(),'\n')
    print('*****Echantillonnage*****')
    print('échantillon aléatoire depuis e1 = ',e1.sample(5, quantization=False),'\n')
    hist = {'a':2, 'b':0, 'c':1}
    print('histogramme:', hist,'\n')
    print('vraisemblance maximale:',MassFunction.from_samples(hist, 'bayesian', s=0),'\n')
    print('lissage de Laplace:',MassFunction.from_samples(hist, 'bayesian', s=1),'\n')
```

Résultats :

BEL :

```
bel_e1 = {frozenset(): 0.0, frozenset({'ct'}): 0.15, frozenset({'cef'}): 0.4, frozenset({'mt'}): 0.45, frozenset({'ct',
'cef'}): 0.55, frozenset({'ct', 'mt'}): 0.6, frozenset({'cef', 'mt'}): 0.8500000000000001, frozenset({'ct', 'cef',
'mt'}): 1.0}

bel_e2 = {frozenset(): 0.0, frozenset({'cef'}): 0.0, frozenset({'mt'}): 0.75, frozenset({'on'}): 0.0, frozenset({'ct'}):
0.0, frozenset({'cb'}): 0.0, frozenset({'cef', 'mt'}): 0.75, frozenset({'on', 'cef'}): 0.0, frozenset({'ct', 'cef'}):
0.0, frozenset({'cef', 'cb'}): 0.0, frozenset({'on', 'mt'}): 0.75, frozenset({'ct', 'mt'}): 0.75, frozenset({'cb',
'mt'}): 0.75, frozenset({'on', 'ct'}): 0.0, frozenset({'on', 'cb'}): 0.0, frozenset({'ct', 'cb'}): 0.0, frozenset({'on',
'cef', 'mt'}): 0.75, frozenset({'ct', 'cef', 'mt'}): 0.75, frozenset({'cb', 'cef', 'mt'}): 0.75, frozenset({'on', 'ct',
'cef'}): 0.0, frozenset({'on', 'cef', 'cb'}): 0.0, frozenset({'ct', 'cef', 'cb'}): 0.0, frozenset({'on', 'ct', 'mt'}):
0.75, frozenset({'on', 'cb', 'mt'}): 0.75, frozenset({'cb', 'ct', 'mt'}): 0.75, frozenset({'on', 'ct', 'cb'}): 0.0,
frozenset({'on', 'ct', 'cef', 'mt'}): 0.75, frozenset({'on', 'cb', 'cef', 'mt'}): 0.75, frozenset({'cb', 'ct', 'cef',
'mt'}): 0.75, frozenset({'on', 'ct', 'cef', 'cb'}): 0.0, frozenset({'on', 'ct', 'cb', 'mt'}): 0.75, frozenset({'mt',
'on', 'ct', 'cef', 'cb'}): 1.0}

bel_e3 = {frozenset(): 0.0, frozenset({'mt'}): 0.35, frozenset({'on'}): 0.02, frozenset({'ct'}): 0.5,
frozenset({'cef'}): 0.0, frozenset({'cb'}): 0.08, frozenset({'on', 'mt'}): 0.37, frozenset({'ct', 'mt'}): 0.85,
frozenset({'cef', 'mt'}): 0.35, frozenset({'cb', 'mt'}): 0.43, frozenset({'on', 'ct'}): 0.52, frozenset({'on', 'cef'}):
0.02, frozenset({'on', 'cb'}): 0.1, frozenset({'ct', 'cef'}): 0.5, frozenset({'ct', 'cb'}): 0.58, frozenset({'cef',
'cb'}): 0.08, frozenset({'on', 'ct', 'mt'}): 0.87, frozenset({'on', 'cef', 'mt'}): 0.37, frozenset({'on', 'cb', 'mt'}):
0.44999999999999996, frozenset({'ct', 'cef', 'mt'}): 0.85, frozenset({'cb', 'ct', 'mt'}): 0.9299999999999999,
frozenset({'cb', 'cef', 'mt'}): 0.43, frozenset({'on', 'ct', 'cef'}): 0.52, frozenset({'on', 'ct', 'cb'}): 0.6,
frozenset({'on', 'cef', 'cb'}): 0.1, frozenset({'ct', 'cef', 'cb'}): 0.58, frozenset({'on', 'ct', 'cef', 'mt'}): 0.87,
frozenset({'on', 'ct', 'cb', 'mt'}): 0.95, frozenset({'on', 'cb', 'cef', 'mt'}): 0.44999999999999996, frozenset({'cb',
'ct', 'cef', 'mt'}): 0.9299999999999999, frozenset({'on', 'ct', 'cef', 'cb'}): 0.6, frozenset({'mt', 'on', 'ct', 'cef',
'cb'}): 1.0}
```


PL :

```
pl_e1 = {frozenset(): 0.0, frozenset({'ct'}): 0.15, frozenset({'cef'}): 0.4, frozenset({'mt'}): 0.45, frozenset({'ct',
'cef'}): 0.55, frozenset({'ct', 'mt'}): 0.6, frozenset({'cef', 'mt'}): 0.8500000000000001, frozenset({'ct', 'cef',
'mt'}): 1.0}

pl_e2 = {frozenset(): 0.0, frozenset({'cef'}): 0.25, frozenset({'mt'}): 1.0, frozenset({'on'}): 0.25, frozenset({'ct'}):
0.25, frozenset({'cb'}): 0.25, frozenset({'cef', 'mt'}): 1.0, frozenset({'on', 'cef'}): 0.25, frozenset({'ct', 'cef'}):
0.25, frozenset({'cef', 'cb'}): 0.25, frozenset({'on', 'mt'}): 1.0, frozenset({'ct', 'mt'}): 1.0, frozenset({'cb',
'mt'}): 1.0, frozenset({'on', 'ct'}): 0.25, frozenset({'on', 'cb'}): 0.25, frozenset({'ct', 'cb'}): 0.25,
frozenset({'on', 'cef', 'mt'}): 1.0, frozenset({'ct', 'cef', 'mt'}): 1.0, frozenset({'cb', 'cef', 'mt'}): 1.0,
frozenset({'on', 'ct', 'cef'}): 0.25, frozenset({'on', 'cef', 'cb'}): 0.25, frozenset({'ct', 'cef', 'cb'}): 0.25,
frozenset({'on', 'ct', 'mt'}): 1.0, frozenset({'on', 'cb', 'mt'}): 1.0, frozenset({'cb', 'ct', 'mt'}): 1.0,
frozenset({'on', 'ct', 'cb'}): 0.25, frozenset({'on', 'ct', 'cef', 'mt'}): 1.0, frozenset({'on', 'cb', 'cef', 'mt'}):
1.0, frozenset({'cb', 'ct', 'cef', 'mt'}): 1.0, frozenset({'on', 'ct', 'cef', 'cb'}): 0.25, frozenset({'on', 'ct', 'cb',
'mt'}): 1.0, frozenset({'mt', 'on', 'ct', 'cef', 'cb'}): 1.0}

pl_e3 = {frozenset(): 0.0, frozenset({'mt'}): 0.39999999999999997, frozenset({'on'}): 0.07, frozenset({'ct'}): 0.55,
frozenset({'cef'}): 0.05, frozenset({'cb'}): 0.13, frozenset({'on', 'mt'}): 0.42, frozenset({'ct', 'mt'}): 0.9,
frozenset({'cef', 'mt'}): 0.39999999999999997, frozenset({'cb', 'mt'}): 0.48, frozenset({'on', 'ct'}): 0.57,
frozenset({'on', 'cef'}): 0.07, frozenset({'on', 'cb'}): 0.15, frozenset({'ct', 'cef'}): 0.55, frozenset({'ct', 'cb'}):
0.63, frozenset({'cef', 'cb'}): 0.13, frozenset({'on', 'ct', 'mt'}): 0.9199999999999999, frozenset({'on', 'cef', 'mt'}):
0.42, frozenset({'on', 'cb', 'mt'}): 0.5, frozenset({'ct', 'cef', 'mt'}): 0.9, frozenset({'cb', 'ct', 'mt'}): 0.98,
frozenset({'cb', 'cef', 'mt'}): 0.48, frozenset({'on', 'ct', 'cef'}): 0.57, frozenset({'on', 'ct', 'cb'}): 0.65,
frozenset({'on', 'cef', 'cb'}): 0.15, frozenset({'ct', 'cef', 'cb'}): 0.63, frozenset({'on', 'ct', 'cef', 'mt'}):
0.9199999999999999, frozenset({'on', 'ct', 'cb', 'mt'}): 1.0, frozenset({'on', 'cb', 'cef', 'mt'}): 0.5,
frozenset({'cb', 'ct', 'cef', 'mt'}): 0.98, frozenset({'on', 'ct', 'cef', 'cb'}): 0.65, frozenset({'mt', 'on', 'ct',
'cef', 'cb'}): 1.0}
```


Autres résultats :

```
***************COMBINAISON***************
loi de combinaison de Dempster pour e1 & e2 & e3 = {{'mt'}:0.8753799392097265; {'ct'}:0.10030395136778114; {'cef'}:0.024316109422492405}
combinaison conjonctive non normalisée de e1, e2, and e3 = {set():0.794375; {'mt'}:0.18; {'ct'}:0.020624999999999998; {'cef'}:
0.005000000000000001}
combinaison disjonctive  de e1, e2, et e3 = {{'cb', 'on', 'cef', 'mt', 'ct'}:0.2875; {'mt', 'ct'}:0.264375; {'mt', 'ct', 'cef'}:
0.15000000000000002; {'mt'}:0.118125; {'mt', 'cef'}:0.10500000000000001; {'mt', 'cb'}:0.027000000000000003; {'mt', 'cb', 'cef'}:
0.024000000000000004; {'mt', 'cb', 'ct'}:0.009; {'mt', 'on'}:0.006750000000000001; {'mt', 'on', 'cef'}:0.006000000000000001; {'mt', 'on', 'ct'}:
0.00225}

***** Cadre de discernement,éléments focaux et noyau*****
Cadre de discernement de e1 = frozenset({'mt', 'ct', 'cef'})
éléments focaux de e1 = {frozenset({'mt'}), frozenset({'ct'}), frozenset({'cef'})}
noyau de e1 = frozenset({'mt', 'ct', 'cef'})
noyaux combinés de e1 et e3 = frozenset({'mt', 'ct', 'cef'})

poids de conflict entre e1, e2, and e3 = 1.581701157462501

*****Transformation pignistique*****
Transformation pignistique de e1 = {{'mt'}:0.45; {'cef'}:0.4; {'ct'}:0.15}
Transformation pignistique de e2 = {{'mt'}:0.8; {'cb'}:0.05; {'on'}:0.05; {'cef'}:0.05; {'ct'}:0.05}
Transformation pignistique de e3 = {{'ct'}:0.51; {'mt'}:0.36; {'cb'}:0.09; {'on'}:0.03; {'cef'}:0.01}

*****Mesure d'incertitude de conflit local*****
conflit local  de e1 = 1.4577174691301484
entropie de la transformation pignistique de e3 = 1.5569041396985868
*****Echantillonnage*****
échantillon aléatoire depuis e1 =  [frozenset({'ct'}), frozenset({'mt'}), frozenset({'ct'}), frozenset({'cef'}), frozenset({'mt'})]

histogramme: {'a': 2, 'b': 0, 'c': 1}
vraisemblance maximale: {{'a'}:0.6666666666666666; {'c'}:0.3333333333333333; {'b'}:0.0}
lissage de Laplace: {{'a'}:0.5; {'c'}:0.3333333333333333; {'b'}:0.16666666666666666}
```

Conclusion :

Le langage Python et la librairie *py_dempster_shafer* nous ont permis, à travers ce TP, d'implémenter des fonctions de croyance et d'interpreter plusieurs calculs et voir leurs résultats dans cette théorie de manière flexible et rapide.