ARTIFICIAL INTELLIGENCE.

LAB 9.

Salil Satyam
RA1911003010675.

AIM: Implementation of uncertain methods (Dampster Shafer Theory).

PROBLEM FORMULATION:

To solve inference problem representing uncertain method to obtain a belief function.
using the mass function which has built-in combination rules obtain the Dampster rule of combination.

initial state:

m1 = {'a': 0.4, 'b': 0.2, 'ab': 0.1, 'abc': 0.3}
m2 = {'b': 0.5, 'c': 0.2, 'ac': 0.3, 'a': 0.0}.

final state:

{'ac': 0.157894, 'c': 0.105263, 'b': 0.5263157,
'ab': 0.0, 'abc': 0.0, 'a': 0.21052631}

PROBLEM SOLVING:

The combination is calculated from the two sets of masses m_1 and m_2 in the following manner:

- $m_{1,2}(\phi) = 0.$
- $m_{1,2}(A) = (m_1 \oplus m_2)(A) = \frac{1}{1-K} \sum_{B \cap C = A \neq \phi} m_1(B) m_2(C)$

where,

$$K = \sum_{B \cap C = \phi} m_1(B) m_2(C).$$

combination of m_1 & m_2.
{'b'}: 0.5, {'a'}: 0.2499, {'c', 'a'}: 0.1499, {'c'}: 0.0999}

ARTIFICIAL INTELLIGENCE

LAB 9

IMPLEMENTATION OF UNCERTAIN METHODS – DEMPSTER SHAFER THEORY

ALGORITHM:

Step 1: Start
Step 2: Each piece of evidence is represented by a separate belief function
Step 3: Combination rules are then used to successively fuse all these belief
functions in order to obtain a belief function representing all available evidence.
Step 4: Specifically, the combination (called the joint mass) is calculated from
the two sets of masses m1 and m2 in the following manner:

- $m_{1,2}(\varnothing) = 0$
- $m_{1,2}(A) = (m_1 \oplus m_2)(A) = (1/1-K) \sum_{B \cap C = A \neq \varnothing} m_1(B)\, m_2(C)$

where,

- $K = \sum_{B \cap C = \varnothing} m_1(B)\, m_2(C)$ K

K is a measure of the amount of conflict between the two mass sets.
Step 5: In python Mass-Function has the built-in combination rules.
Step 6: Stop

SOURCE CODE:

```
from numpy import *
def DempsterRule(m1, m2):
## extract the frame of discernment
sets=set(m1.keys()).union(set(m2.keys()))
result=dict.fromkeys(sets,0)
## Combination process
for i in m1.keys():
for j in m2.keys():
if set(str(i)).intersection(set(str(j))) == set(str(i)):
result[i]+=m1[i]*m2[j]
elif set(str(i)).intersection(set(str(j))) == set(str(j)):
result[j]+=m1[i]*m2[j]
## normalize the results
f= sum(list(result.values()))
for i in result.keys():
result[i] /=f
return result
m1 = {'a':0.4, 'b':0.2, 'ab':0.1, 'abc':0.3}
m2 = {'b':0.5, 'c':0.2, 'ac':0.3, 'a':0.0}
print(DempsterRule(m1, m2))
```

```
{'b': 0.5263157894736842, 'ab': 0.0, 'ac': 0.15789473684210523, 'abc': 0.0, 'c': 0.10526315789473682, 'a': 0.21052631578947364}

Process exited with code: 0
```

RESULT: Hence, the Implementation of Dempster Shafer's Theory was done successfully.