Tutorial 5

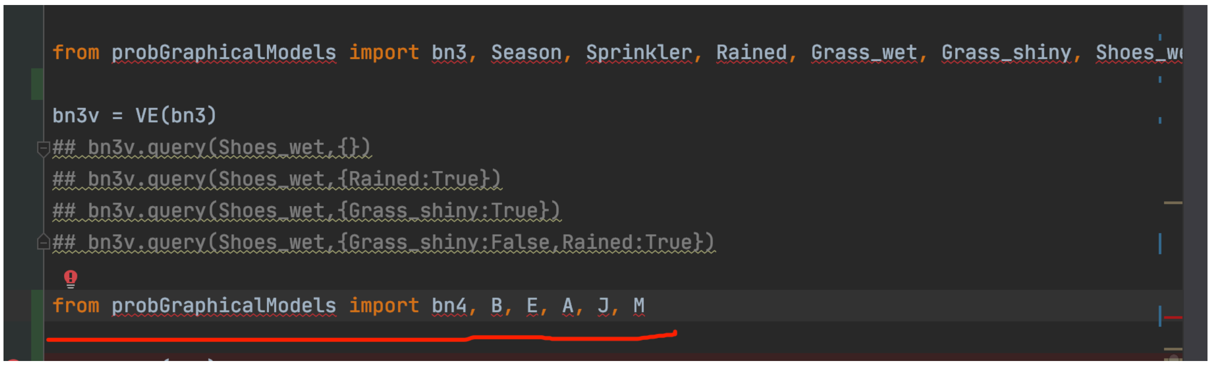
Name: Zhaokun Su

Breakout group: 4

Zid: z5235878

Compute P(Burglary|Alarm) using the AIPython program probVE.py. Answers to method calls such as bn4v.query(B,{A:True}) are in a list form [F,T] giving the probability that B is false and the probability that B is true, in conjunction with the list of conditions (here that A is true). The desired answer is then calculated by normalization. The Bayesian network is encoded as bn4 in probGraphicalModels.py.

We firstly import our variables we defined in probGraphicalModels.py.

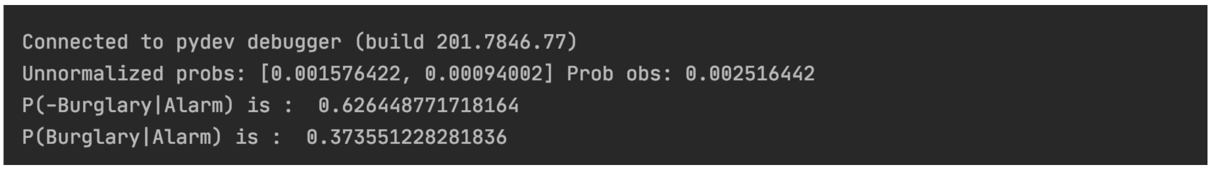


```
from probGraphicalModels import bn3, Season, Sprinkler, Rained, Grass_wet, Grass_shiny, Shoes_we

bn3v = VE(bn3)
## bn3v.query(Shoes_wet,{})
## bn3v.query(Shoes_wet,{Rained:True})
## bn3v.query(Shoes_wet,{Grass_shiny:True})
## bn3v.query(Shoes_wet,{Grass_shiny:False,Rained:True})

from probGraphicalModels import bn4, B, E, A, J, M
```

And then, we make our query to calculate P(B|A) and P(-B|A) given A is true:

```
bn4v = VE(bn4)
result = bn4v.query(B, {A: True})
for key, value in result.items():
    if key is False:
        print("P(-Burglary|Alarm) is : ", value)
    if key is True:
        print("P(Burglary|Alarm) is : ", value)
```

finally, we get the results (normalization result):



```
Connected to pydev debugger (build 201.7846.77)
Unnormalized probs: [0.001576422, 0.00094002] Prob obs: 0.002516442
P(-Burglary|Alarm) is :  0.626448771718164
P(Burglary|Alarm) is :  0.373551228281836
```