

# E09 Bayesian Network

---

孙新梦 1834149

October 26, 2020

## 目录

|          |                                  |          |
|----------|----------------------------------|----------|
| <b>1</b> | <b>Pomegranate Installation</b>  | <b>2</b> |
| <b>2</b> | <b>Building Bayesian Network</b> | <b>2</b> |
| <b>3</b> | <b>Tasks</b>                     | <b>3</b> |
| 3.1      | Burglary . . . . .               | 3        |
| 3.2      | Diagnosing . . . . .             | 4        |
| <b>4</b> | <b>Codes and Results</b>         | <b>7</b> |
| 4.1      | 1.Burglary . . . . .             | 7        |
| 4.2      | 2.Diagnosis . . . . .            | 11       |

# 1 Pomegranate Installation

## Under Linux:

1. Install python first (**python 2**, not python 3).
2. Run `sudo apt-get install python-pip` to install pip.
3. Run `sudo pip install pomegranate` to install pomegranate.

```
al2017@osboxes:~$ pip
The program 'pip' is currently not installed. You can install it by typing:
sudo apt install python-pip
al2017@osboxes:~$ sudo apt install python-pip
[sudo] password for al2017:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-4.10.0-28 linux-headers-4.10.0-28-generic
  linux-headers-4.10.0-33 linux-headers-4.10.0-33-generic
  linux-headers-4.10.0-35 linux-image-4.10.0-28-generic
  linux-image-4.10.0-33-generic linux-image-4.10.0-35-generic
  linux-image-extra-4.10.0-28-generic linux-image-extra-4.10.0-33-generic
  linux-image-extra-4.10.0-35-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libxpat1-dev libpython-all-dev libpython-dev libpython2.7-dev python-all
  python-all-dev python-dev python-pip python-pip-whl python-pkg-resources
  python-setuptools python-wheel python2.7-dev
Suggested packages:
  python-setuptools-doc
The following NEW packages will be installed:
  libxpat1-dev libpython-all-dev libpython-dev libpython2.7-dev python-all
  python-all-dev python-dev python-pip python-pip-whl python-pkg-resources
  python-setuptools python-wheel python2.7-dev
0 upgraded, 13 newly installed, 0 to remove and 113 not upgraded.
Need to get 29.8 MB of archives.
After this operation, 45.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

```
al2017@osboxes:~$ sudo pip install pomegranate
The directory /home/al2017/.cache/pip/http or its parent directory is not owned by the current user and the
cache has been disabled. Please check the permissions and owner of that directory. If executing pip with sudo,
you may want sudo's -H flag.
The directory /home/al2017/.cache/pip/ or its parent directory is not owned by the current user and caching w
heels has been disabled. check the permissions and owner of that directory. If executing pip with sudo, you na
y want sudo's -H flag.
Collecting pomegranate
  Downloading pomegranate-0.8.1-cp27-cp27mu-manylinux1_x86_64.whl (9.1MB)
  100% |#####| 9.2MB 115kB/s
Collecting networkx<2.0, >=1.8.1 (from pomegranate)
  Downloading networkx-1.11-py2.py3-none-any.whl (1.3MB)
  100% |#####| 1.3MB 478kB/s
Collecting numpy>=1.8.0 (from pomegranate)
  Downloading numpy-1.13.3-cp27-cp27mu-manylinux1_x86_64.whl (16.0MB)
  100% |#####| 16.7MB 76kB/s
Collecting scipy>=0.17.0 (from pomegranate)
  Downloading scipy-1.0.0-cp27-cp27mu-manylinux1_x86_64.whl (46.7MB)
  100% |#####| 46.7MB 38kB/s
Collecting joblib>=0.9.0b4 (from pomegranate)
  Downloading joblib-0.11-py2.py3-none-any.whl (176kB)
  100% |#####| 184kB 793kB/s
Collecting decorator>=3.4.0 (from networkx<2.0, >=1.8.1->pomegranate)
  Downloading decorator-4.1.2-py2.py3-none-any.whl
Installing collected packages: decorator, networkx, numpy, scipy, joblib, pomegranate
Successfully installed decorator-4.1.2 joblib-0.11 numpy-1.13.3 pomegranate-0.8.1 scipy-1.0.0
You are using pip version 8.1.1, however version 9.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

## Under Windows

You can also run `pip install pomegranate` if you have installed pip. If you don't know how to install pip, please click <https://jingyan.baidu.com/article/e73e26c0d94e0524adb6a7ff.html>.

For more, please click the homepage of Pomegranate - <https://github.com/jmschrei/pomegranate> for help.

```
PS C:\Users\Lau ChiuSui\Desktop\pomegranate-0.8.1> pip install pomegranate
Collecting pomegranate
  Downloading pomegranate-0.8.1-cp27-cp27m-win32.whl (3.4MB)
  100% |#####| 3.4MB 227kB/s
Collecting scipy>=0.17.0 (from pomegranate)
  Downloading scipy-1.0.0-cp27-none-win32.whl (26.4MB)
  100% |#####| 26.4MB 40kB/s
Collecting joblib>=0.9.0b4 (from pomegranate)
  Downloading joblib-0.11-py2.py3-none-any.whl (176kB)
  100% |#####| 184kB 957kB/s
Collecting networkx<2.0, >=1.8.1 (from pomegranate)
  Downloading networkx-1.11-py2.py3-none-any.whl (1.3MB)
  100% |#####| 1.3MB 463kB/s
Requirement already satisfied: numpy>=1.8.0 in d:\softwares\python27\lib\site-packages (from pomegranate)
Collecting decorator>=3.4.0 (from networkx<2.0, >=1.8.1->pomegranate)
  Downloading decorator-4.1.2-py2.py3-none-any.whl
Installing collected packages: scipy, joblib, decorator, networkx, pomegranate
Found existing installation: networkx 2.0
Uninstalling networkx-2.0:
  Successfully uninstalled networkx-2.0
Successfully installed decorator-4.1.2 joblib-0.11 networkx-1.11 pomegranate-0.8.1 scipy-1.0.0
```

# 2 Building Bayesian Network

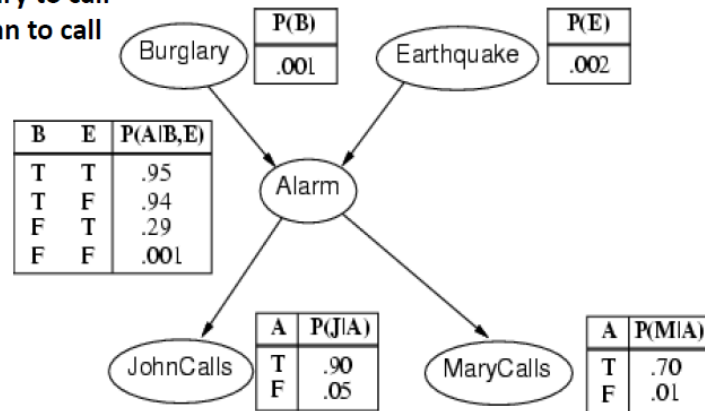
Please refer to Tutorial\_4\_Bayesian\_Networks.pdf. I will explain it in class.

### 3 Tasks

#### 3.1 Burglary

- A burglary can set the alarm off
- An earthquake can set the alarm off
- The alarm can cause Mary to call
- The alarm can cause John to call

Note that these tables only provide the probability that  $X_i$  is true.  
(E.g.,  $\Pr(A \text{ is true} | B, E)$ )  
The probability that  $X_i$  is false is 1- these values



Please code to calculate:

1.  $P(A)$
2.  $P(J\bar{M})$
3.  $P(A|J\bar{M})$
4.  $P(B|A)$
5.  $P(B|J\bar{M})$
6.  $P(J\bar{M}|\bar{B})$

```
P(Alarm) =
0.002516442

P(J&&~M) =
0.050054875461

P(A | J&&~M) =
0.0135738893313

P(B | A) =
0.373551228282

P(B | J&&~M) =
0.0051298581334

P(J&&~M | ~B) =
0.049847949
```

## 3.2 Diagnosing

### Variables and their domains

- 1 (1) PatientAge: ['0–30', '31–65', '65+']
- 2 (2) CTScanResult: ['Ischemic Stroke', 'Hemorrhagic Stroke']
- 3 (3) MRIScanResult: ['Ischemic Stroke', 'Hemorrhagic Stroke']
- 4 (4) StrokeType: ['Ischemic Stroke', 'Hemorrhagic Stroke', 'Stroke Mimic']
- 5 (5) Anticoagulants: ['Used', 'Not used']
- 6 (6) Mortality: ['True', 'False']
- 7 (7) Disability: ['Negligible', 'Moderate', 'Severe']

### CPTs

**Note:** [CTScanResult, MRIScanResult, StrokeType] means:

$P(\text{StrokeType}=\dots \mid \text{CTScanResult}=\dots \wedge \text{MRIScanResult}=\dots)$

- 8 (1)
- 9 [PatientAge]
- 10
- 11 ['0–30', 0.10],
- 12 ['31–65', 0.30],
- 13 ['65+', 0.60]
- 14
- 15 (2)
- 16 [CTScanResult]
- 17
- 18 ['Ischemic Stroke', 0.7],
- 19 ['Hemorrhagic Stroke', 0.3]
- 20
- 21 (3)
- 22 [MRIScanResult]
- 23
- 24 ['Ischemic Stroke', 0.7],
- 25 ['Hemorrhagic Stroke', 0.3]

```

26
27 (4)
28 [ Anticoagulants ]
29
30 [ Used ',0.5] ,
31 [ 'Not used ',0.5]
32
33 (5)
34 [ CTScanResult , MRIScanResult ,StrokeType])
35
36 [ 'Ischemic Stroke ', 'Ischemic Stroke ', 'Ischemic Stroke ',0.8] ,
37 [ 'Ischemic Stroke ', 'Hemorrhagic Stroke ', 'Ischemic Stroke ',0.5] ,
38 [ 'Hemorrhagic Stroke ', 'Ischemic Stroke ', 'Ischemic Stroke ',0.5] ,
39 [ 'Hemorrhagic Stroke ', 'Hemorrhagic Stroke ', 'Ischemic Stroke
    ',0] ,
40
41 [ 'Ischemic Stroke ', 'Ischemic Stroke ', 'Hemorrhagic Stroke ',0] ,
42 [ 'Ischemic Stroke ', 'Hemorrhagic Stroke ', 'Hemorrhagic Stroke
    ',0.4] ,
43 [ 'Hemorrhagic Stroke ', 'Ischemic Stroke ', 'Hemorrhagic Stroke
    ',0.4] ,
44 [ 'Hemorrhagic Stroke ', 'Hemorrhagic Stroke ', 'Hemorrhagic Stroke
    ',0.9] ,
45
46 [ 'Ischemic Stroke ', 'Ischemic Stroke ', 'Stroke Mimic ',0.2] ,
47 [ 'Ischemic Stroke ', 'Hemorrhagic Stroke ', 'Stroke Mimic ',0.1] ,
48 [ 'Hemorrhagic Stroke ', 'Ischemic Stroke ', 'Stroke Mimic ',0.1] ,
49 [ 'Hemorrhagic Stroke ', 'Hemorrhagic Stroke ', 'Stroke Mimic ',0.1] ,
50
51 (6)
52 [StrokeType , Anticoagulants , Mortality]
53
54 [ 'Ischemic Stroke ', 'Used ', 'False ',0.28] ,

```

```

55 ['Hemorrhagic Stroke ', 'Used ', 'False ',0.99],
56 ['Stroke Mimic ', 'Used ', 'False ',0.1],
57 ['Ischemic Stroke ', 'Not used ', 'False ',0.56],
58 ['Hemorrhagic Stroke ', 'Not used ', 'False ',0.58],
59 ['Stroke Mimic ', 'Not used ', 'False ',0.05],
60
61 ['Ischemic Stroke ', 'Used ', 'True ',0.72],
62 ['Hemorrhagic Stroke ', 'Used ', 'True ',0.01],
63 ['Stroke Mimic ', 'Used ', 'True ',0.9],
64 ['Ischemic Stroke ', 'Not used ', 'True ',0.44],
65 ['Hemorrhagic Stroke ', 'Not used ', 'True ',0.42 ],
66 ['Stroke Mimic ', 'Not used ', 'True ',0.95]
67
68 (7)
69 [StrokeType, PatientAge, Disability]
70
71 ['Ischemic Stroke ', '0-30 ', 'Negligible ', 0.80],
72 ['Hemorrhagic Stroke ', '0-30 ', 'Negligible ', 0.70],
73 ['Stroke Mimic ', '0-30 ', 'Negligible ',0.9],
74 ['Ischemic Stroke ', '31-65 ', 'Negligible ', 0.60],
75 ['Hemorrhagic Stroke ', '31-65 ', 'Negligible ', 0.50],
76 ['Stroke Mimic ', '31-65 ', 'Negligible ',0.4],
77 ['Ischemic Stroke ', '65+ ', 'Negligible ',0.30],
78 ['Hemorrhagic Stroke ', '65+ ', 'Negligible ',0.20],
79 ['Stroke Mimic ', '65+ ', 'Negligible ',0.1],
80
81 ['Ischemic Stroke ', '0-30 ', 'Moderate ',0.1],
82 ['Hemorrhagic Stroke ', '0-30 ', 'Moderate ',0.2],
83 ['Stroke Mimic ', '0-30 ', 'Moderate ',0.05],
84 ['Ischemic Stroke ', '31-65 ', 'Moderate ',0.3],
85 ['Hemorrhagic Stroke ', '31-65 ', 'Moderate ',0.4],
86 ['Stroke Mimic ', '31-65 ', 'Moderate ',0.3],
87 ['Ischemic Stroke ', '65+ ', 'Moderate ',0.4],

```

```

88 [ 'Hemorrhagic Stroke ', '65+' , 'Moderate ', 0.2 ] ,
89 [ 'Stroke Mimic ', '65+' , 'Moderate ', 0.1 ] ,
90
91 [ 'Ischemic Stroke ', '0-30' , 'Severe ', 0.1 ] ,
92 [ 'Hemorrhagic Stroke ', '0-30' , 'Severe ', 0.1 ] ,
93 [ 'Stroke Mimic ', '0-30' , 'Severe ', 0.05 ] ,
94 [ 'Ischemic Stroke ', '31-65' , 'Severe ', 0.1 ] ,
95 [ 'Hemorrhagic Stroke ', '31-65' , 'Severe ', 0.1 ] ,
96 [ 'Stroke Mimic ', '31-65' , 'Severe ', 0.3 ] ,
97 [ 'Ischemic Stroke ', '65+' , 'Severe ', 0.3 ] ,
98 [ 'Hemorrhagic Stroke ', '65+' , 'Severe ', 0.6 ] ,
99 [ 'Stroke Mimic ', '65+' , 'Severe ', 0.8 ]

```

## Calculation

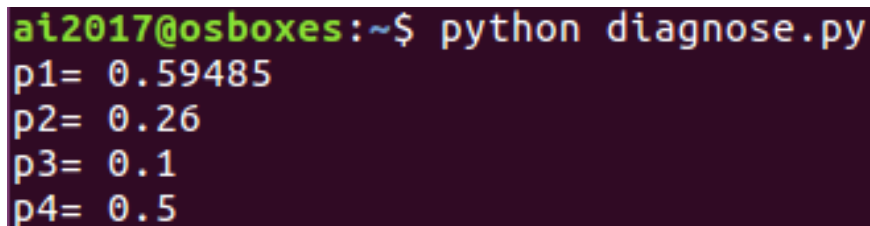
Please code to calculate the following probability value:

$p1 = P(\text{Mortality}=\text{'True'} \mid \text{PatientAge}=\text{'31-65'} \wedge \text{CTScanResult}=\text{'Ischemic Stroke'})$

$p2 = P(\text{Disability}=\text{'Moderate'} \mid \text{PatientAge}=\text{'65+'} \wedge \text{MRIScanResult}=\text{'Hemorrhagic Stroke'})$

$p3 = P(\text{StrokeType}=\text{'Stroke Mimic'} \mid \text{PatientAge}=\text{'65+'} \wedge \text{CTScanResult}=\text{'Hemorrhagic Stroke'} \wedge \text{MRIScanResult}=\text{'Ischemic Stroke'})$

$p4 = P(\text{Anticoagulants}=\text{'Not used'} \mid \text{PatientAge}=\text{'0-30'})$



```

ai2017@osboxes:~$ python diagnose.py
p1= 0.59485
p2= 0.26
p3= 0.1
p4= 0.5

```

Please solve the 2 tasks and hand in a file named E08\_YourNumber.pdf, and send it to ai\_2020@foxmail.com

## 4 Codes and Results

### 4.1 1.Burglary

#### Code

```

100 from pomegranate import *
101 import pandas as pd

```

```

102
103 Burglary = DiscreteDistribution(
104     {
105         'T': 0.001,
106         'F': 0.999
107     }
108 )
109
110 EarthQuake = DiscreteDistribution(
111     {
112         'T': 0.002,
113         'F': 0.998
114     }
115 )
116
117 Alarm = ConditionalProbabilityTable(
118     [
119         ['T', 'T', 'T', 0.95],
120         ['T', 'F', 'T', 0.94],
121         ['F', 'T', 'T', 0.29],
122         ['F', 'F', 'T', 0.001],
123
124         ['T', 'T', 'F', 0.05],
125         ['T', 'F', 'F', 0.06],
126         ['F', 'T', 'F', 0.71],
127         ['F', 'F', 'F', 0.999],
128     ],
129     [Burglary, EarthQuake]
130 )
131
132 JohnC = ConditionalProbabilityTable(
133     [
134         ['T', 'T', 0.9],

```



```

135         [ 'T' , 'F' , 0.1] ,
136         [ 'F' , 'T' , 0.05] ,
137         [ 'F' , 'F' , 0.95] ,
138     ] ,
139     [ Alarm]
140 )
141
142 MaryC = ConditionalProbabilityTable(
143     [
144         [ 'T' , 'T' , 0.7] ,
145         [ 'T' , 'F' , 0.3] ,
146         [ 'F' , 'T' , 0.01] ,
147         [ 'F' , 'F' , 0.99] ,
148     ] ,
149     [ Alarm]
150 )
151
152
153 s1 = Node(Burglary , name="Burglary")
154 s2 = Node(EarthQuake , name="EarthQuake")
155 s3 = Node(Alarm , name="Alarm")
156 s4 = Node(JohnC , name="JohnC")
157 s5 = Node(MaryC , name="MaryC")
158
159 model = BayesianNetwork("Buglary Problem")
160
161 model.add_states(s1 , s2 , s3 , s4 , s5)
162
163 model.add_edge(s1 , s3)
164 model.add_edge(s2 , s3)
165 model.add_edge(s3 , s4)
166 model.add_edge(s3 , s5)
167

```

```

168 model.bake()
169
170 idx = ['P(Alarm)', 'P(J && ~M)', 'P(B | A)', 'P(A | J && ~M)',
171        'P(B | J && ~M)', 'P(J && ~M | ~B)']
172
173 df = pd.DataFrame(index=idx, columns=['Probability'])
174
175 # P(A)
176 df['Probability']['P(Alarm)'] = str(
177     model.predict_proba({})[2].parameters[0]['T'])
178
179 # P(J && ~M)
180 PJ = model.predict_proba({'MaryC': 'F'})[3].parameters[0]['T']
181 PM = model.predict_proba({'JohnC': 'T'})[4].parameters[0]['F']
182 df['Probability']['P(J && ~M)'] = str(PJ * PM)
183
184 # P(A | J && ~M)
185 df['Probability']['P(A | J && ~M)'] = str(model.predict_proba(
186     {'JohnC': 'T', 'MaryC': 'F'})[2].parameters[0]['T'])
187
188 # P(B | A)
189 df['Probability']['P(B | A)'] = str(
190     model.predict_proba({'Alarm': 'T'})[0].parameters[0]['T'])
191
192 # P(B | J && ~M)
193 df['Probability']['P(B | J && ~M)'] = str(model.predict_proba(
194     {'JohnC': 'T', 'MaryC': 'F'})[0].parameters[0]['T'])
195
196 # P(J && ~M | ~B)
197 PJ = model.predict_proba({'MaryC': 'F', 'Burglary': 'F'})[3].
198     parameters[0]['T']
199 PM = model.predict_proba({'JohnC': 'T', 'Burglary': 'F'})[4].
200     parameters[0]['F']

```

```

199 df['Probability'] ['P(J && ~M | ~B)'] = str(PJ * PM)
200
201 print(df)

```

### Result

|                 | Probability           |
|-----------------|-----------------------|
| P(Alarm)        | 0.0025164420000000935 |
| P(J && ~M)      | 0.048624757853553476  |
| P(B   A)        | 0.3735512282818995    |
| P(A   J && ~M)  | 0.01357388933131146   |
| P(B   J && ~M)  | 0.005129858133403527  |
| P(J && ~M   ~B) | 0.04894072965276358   |

## 4.2 2.Diagnosis

### Code

```

203
204 from pomegranate import *
205
206 PatientAge = DiscreteDistribution(
207     {
208         'A': 0.10,
209         'B': 0.30,
210         'C': 0.60
211     }
212 )
213
214 CTScanResult = DiscreteDistribution(
215     {
216         'IS': 0.7,
217         'HS': 0.3
218     }

```

```

219 )
220
221 MRIScanResult = DiscreteDistribution(
222     {
223         'IS ': 0.7,
224         'HS': 0.3
225     }
226 )
227
228 Anticoagulants = DiscreteDistribution(
229     {
230         'T': 0.5,
231         'F': 0.5
232     }
233 )
234
235 StrokeType = ConditionalProbabilityTable(
236     [
237         [ 'IS ', 'IS ', 'IS ', 0.8 ],
238         [ 'IS ', 'HS', 'IS ', 0.5 ],
239         [ 'HS', 'IS ', 'IS ', 0.5 ],
240         [ 'HS', 'HS', 'IS ', 0.0 ],
241
242         [ 'IS ', 'IS ', 'HS', 0.0 ],
243         [ 'IS ', 'HS', 'HS', 0.4 ],
244         [ 'HS', 'IS ', 'HS', 0.4 ],
245         [ 'HS', 'HS', 'HS', 0.9 ],
246
247         [ 'IS ', 'IS ', 'SM', 0.2 ],
248         [ 'IS ', 'HS', 'SM', 0.1 ],
249         [ 'HS', 'IS ', 'SM', 0.1 ],
250         [ 'HS', 'HS', 'SM', 0.1 ],
251     ],

```

```

252     [CTScanResult, MRIScanResult]
253 )
254
255 Mortality = ConditionalProbabilityTable(
256     [
257         ['IS', 'T', 'F', 0.28],
258         ['HS', 'T', 'F', 0.99],
259         ['SM', 'T', 'F', 0.10],
260         ['IS', 'F', 'F', 0.56],
261         ['HS', 'F', 'F', 0.58],
262         ['SM', 'F', 'F', 0.05],
263         ['IS', 'T', 'T', 0.72],
264         ['HS', 'T', 'T', 0.01],
265         ['SM', 'T', 'T', 0.90],
266         ['IS', 'F', 'T', 0.44],
267         ['HS', 'F', 'T', 0.42],
268         ['SM', 'F', 'T', 0.95],
269     ],
270     [StrokeType, Anticoagulants]
271 )
272
273 Disability = ConditionalProbabilityTable(
274     [
275         ['IS', 'A', 'N', 0.80],
276         ['HS', 'A', 'N', 0.70],
277         ['SM', 'A', 'N', 0.90],
278         ['IS', 'B', 'N', 0.60],
279         ['HS', 'B', 'N', 0.50],
280         ['SM', 'B', 'N', 0.40],
281         ['IS', 'C', 'N', 0.30],
282         ['HS', 'C', 'N', 0.20],
283         ['SM', 'C', 'N', 0.10],
284     ]

```

```

285         [ 'IS' , 'A' , 'M' , 0.10] ,
286         [ 'HS' , 'A' , 'M' , 0.20] ,
287         [ 'SM' , 'A' , 'M' , 0.05] ,
288         [ 'IS' , 'B' , 'M' , 0.30] ,
289         [ 'HS' , 'B' , 'M' , 0.40] ,
290         [ 'SM' , 'B' , 'M' , 0.30] ,
291         [ 'IS' , 'C' , 'M' , 0.40] ,
292         [ 'HS' , 'C' , 'M' , 0.20] ,
293         [ 'SM' , 'C' , 'M' , 0.10] ,
294
295         [ 'IS' , 'A' , 'S' , 0.10] ,
296         [ 'HS' , 'A' , 'S' , 0.10] ,
297         [ 'SM' , 'A' , 'S' , 0.05] ,
298         [ 'IS' , 'B' , 'S' , 0.10] ,
299         [ 'HS' , 'B' , 'S' , 0.10] ,
300         [ 'SM' , 'B' , 'S' , 0.30] ,
301         [ 'IS' , 'C' , 'S' , 0.30] ,
302         [ 'HS' , 'C' , 'S' , 0.60] ,
303         [ 'SM' , 'C' , 'S' , 0.80] ,
304     ] ,
305     [StrokeType , PatientAge]
306 )
307
308
309 s1 = Node(PatientAge , name="PatientAge")
310 s2 = Node(CTScanResult , name="CTScanResult")
311 s3 = Node(MRIScanResult , name="MRIScanResult")
312 s4 = Node(Anticoagulants , name="Anticoagulants")
313 s5 = Node(StrokeType , name="StrokeType")
314 s6 = Node(Mortality , name="Mortality")
315 s7 = Node(Disability , name="Disability")
316
317 model = BayesianNetwork(" Diagnosing Problem")

```

```

318
319 model.add_states(s1, s2, s3, s4, s5, s6, s7)
320
321 model.add_edge(s2, s5)
322 model.add_edge(s3, s5)
323
324 model.add_edge(s5, s6)
325 model.add_edge(s4, s6)
326
327 model.add_edge(s5, s7)
328 model.add_edge(s1, s7)
329
330 model.bake()
331
332
333 # P(1)
334 print('P(1) = ', model.predict_proba(
335     {'PatientAge': 'B', 'CTScanResult': 'IS'})[5].parameters[0]['T
    '])
336
337 # P(2)
338 print('P(2) = ', model.predict_proba(
339     {'PatientAge': 'C', 'MRIScanResult': 'HS'})[6].parameters[0]['
    M'])
340
341 # P(3)
342 print('P(3) = ', model.predict_proba(
343     {'PatientAge': 'C', 'CTScanResult': 'HS', 'MRIScanResult': 'IS
    '})[4].parameters[0]['SM'])
344
345 # P(4)
346 print('P(4) = ', model.predict_proba(
347     {'PatientAge': 'A'})[3].parameters[0]['F'])

```

```

348
349 # Helper
350 print('P1 = ', model.predict_proba(
351     {})[5].parameters[0]['T'])
352
353 print('P2 = ', model.predict_proba(
354     {'PatientAge': 'B'})[1].parameters[0]['IS'])
355
356 print('P3 = ', model.predict_proba(
357     {'PatientAge': 'C', 'CTScanResult': 'HS', 'MRIScanResult': 'IS
        '})[4].parameters[0]['HS'])
358
359 print('P4 = ', model.predict_proba(
360     {})[6].parameters[0]['N'])

```

## Result

```

P(1) = 0.5948499999999999
P(2) = 0.26000000000000001
P(3) = 0.100000000000000044
P(4) = 0.5

```