20210602_Wirtschaftsinformatik_MV2

June 2, 2021

1 Bayes'sche Theorem

[1]:
```
"""Das Bayes'sche Theorem ist ein fundamentales Theorem in der Bayes'schen␣
↪Statistik, da es von den Bayes'schen Methoden verwendet wird, um␣
↪Wahrscheinlichkeiten, die Glaubensgrade sind, nach Erhalt neuer Daten zu␣
↪aktualisieren. Bei zwei Ereignissen A und B wird die bedingte␣
↪Wahrscheinlichkeit von A unter der Voraussetzung, dass B wahr ist, wie folgt␣
↪ausgedrückt:"""
```

[1]:
```
"Das Bayes'sche Theorem ist ein fundamentales Theorem in der Bayes'schen
Statistik, da es von den Bayes'schen Methoden verwendet wird, um
Wahrscheinlichkeiten, die Glaubensgrade sind, nach Erhalt neuer Daten zu
aktualisieren. Bei zwei Ereignissen A und B wird die bedingte Wahrscheinlichkeit
von A unter der Voraussetzung, dass B wahr ist, wie folgt ausgedrückt:"
```

[2]:
```
from IPython.display import Image
# Load image from local storage
Image(filename = '/Users/h4/Desktop/Screenshot 2021-06-02 at 09.13.22.png')
```

[2]:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

[3]:
```
"""Die Wahrscheinlichkeit des Beweises P(B) kann mit dem Gesetz der␣
↪Gesamtwahrscheinlichkeit berechnet werden. Wenn"""
```

[3]:
```
'Die Wahrscheinlichkeit des Beweises P(B) kann mit dem Gesetz der
Gesamtwahrscheinlichkeit berechnet werden. Wenn'
```

[6]:
```
from IPython.display import Image
# Load image from local storage
Image(filename = '/Users/h4/Desktop/Screenshot 2021-06-02 at 09.13.32.png')
```

[6]:

$$\{A_1, A_2, \ldots, A_n\}$$

[7]:
```
"""eine Partition des Stichprobenraums ist, der die Menge aller Ergebnisse␣
↪eines Experiments ist, dann:"""
```

[7]:
```
'eine Partition des Stichprobenraums ist, der die Menge aller Ergebnisse eines
Experiments ist, dann:'
```

[8]:
```
from IPython.display import Image
# Load image from local storage
Image(filename = '/Users/h4/Desktop/Screenshot 2021-06-02 at 09.13.38.png')
```

[8]:

$$P(B) = P(B \mid A_1)P(A_1) + P(B \mid A_2)P(A_2) + \cdots + P(B \mid A_n)P(A_n) = \sum_i P(B \mid A_i)P(A_i)$$

2 Bayesian Netzwerk

[9]:
```
"""Ein Bayes'sches Netzwerk ist ein probabilistisches grafisches Modell (eine␣
↪Art statistisches Modell), das einen Satz von Variablen und deren bedingte␣
↪Abhängigkeiten über einen gerichteten azyklischen Graphen (DAG) darstellt.␣
↪Bayes'sche Netze sind ideal, um ein aufgetretenes Ereignis zu nehmen und die␣
↪Wahrscheinlichkeit vorherzusagen, dass eine von mehreren möglichen bekannten␣
↪Ursachen der beitragende Faktor war."""
```

[9]:
```
"Ein Bayes'sches Netzwerk ist ein probabilistisches grafisches Modell (eine Art
statistisches Modell), das einen Satz von Variablen und deren bedingte
Abhängigkeiten über einen gerichteten azyklischen Graphen (DAG) darstellt.
Bayes'sche Netze sind ideal, um ein aufgetretenes Ereignis zu nehmen und die
Wahrscheinlichkeit vorherzusagen, dass eine von mehreren möglichen bekannten
Ursachen der beitragende Faktor war."
```

2.1 Beispiel 1

[10]:
```
import pandas as pd
data = pd.DataFrame(data={'fruit': ["banana", "apple", "banana", "apple",␣
 ↪"banana","apple", "banana",
                                    "apple", "apple", "apple", "banana",␣
 ↪"banana", "apple", "banana",],
                           'tasty': ["yes", "no", "yes", "yes", "yes", "yes",␣
 ↪"yes",
                                    "yes", "yes", "yes", "yes", "no", "no",␣
 ↪"no"],
                           'size': ["large", "large", "large", "small", "large",␣
 ↪"large", "large",
                                    "small", "large", "large", "large",␣
 ↪"large", "small", "small"]})
print(data)
```

```
     fruit tasty   size
0   banana   yes  large
1    apple    no  large
2   banana   yes  large
3    apple   yes  small
4   banana   yes  large
5    apple   yes  large
6   banana   yes  large
7    apple   yes  small
8    apple   yes  large
9    apple   yes  large
10  banana   yes  large
11  banana    no  large
12   apple    no  small
13  banana    no  small
```

[12]:
```
# A1 -- fruit
# A2 -- size
# B  -- tasty
# A1-->B<--A2

# P(B|A1)
```

[13]:
```
!pip install pgmpy
```

```
Requirement already satisfied: pgmpy in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (0.1.14)
Requirement already satisfied: joblib in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from pgmpy) (0.17.0)
Requirement already satisfied: statsmodels in
```

```
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from pgmpy) (0.12.0)
Requirement already satisfied: pandas in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from pgmpy) (1.1.3)
Requirement already satisfied: scipy in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from pgmpy) (1.5.2)
Requirement already satisfied: pyparsing in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from pgmpy) (2.4.7)
Requirement already satisfied: torch in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from pgmpy) (1.8.1)
Requirement already satisfied: scikit-learn in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from pgmpy) (0.23.2)
Requirement already satisfied: networkx in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from pgmpy) (2.5)
Requirement already satisfied: tqdm in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from pgmpy) (4.50.2)
Requirement already satisfied: numpy in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from pgmpy) (1.20.3)
Requirement already satisfied: patsy>=0.5 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from statsmodels->pgmpy)
(0.5.1)
Requirement already satisfied: pytz>=2017.2 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from pandas->pgmpy)
(2019.3)
Requirement already satisfied: python-dateutil>=2.7.3 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from pandas->pgmpy) (2.8.1)
Requirement already satisfied: typing-extensions in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from torch->pgmpy)
(3.7.4.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from scikit-learn->pgmpy)
(2.1.0)
Requirement already satisfied: decorator>=4.3.0 in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from networkx->pgmpy)
(4.4.2)
Requirement already satisfied: six in
/Users/h4/opt/anaconda3/lib/python3.8/site-packages (from
patsy>=0.5->statsmodels->pgmpy) (1.15.0)
```

[14]:
```
# Bayes Netzwerk Model aufstellen

from pgmpy.models import BayesianModel

model = BayesianModel([('fruit', 'tasty'), ('size', 'tasty')]) # fruit -> tasty␣
 ↪<- size
```

[16]:
```
# Parameter müssen ermittelt werden. Counts = Häufigkeit
```

```
from pgmpy.estimators import ParameterEstimator

pe = ParameterEstimator(model, data)

print('\n', pe.state_counts(('fruit'))) #unconditional
print('\n', pe.state_counts(('tasty'))) #conditional
```

```
         fruit
apple        7
banana       7

 fruit apple       banana
size  large small  large small
tasty
no      1.0   1.0    1.0   1.0
yes     3.0   2.0    5.0   0.0
```

[20]:
```
# Conditional Probability Distribution. CPD = Bedingte Wahrscheinlichkeiten

from pgmpy.estimators import MaximumLikelihoodEstimator

mle = MaximumLikelihoodEstimator(model, data)

print(mle.estimate_cpd('fruit'))  # unconditional
print(mle.estimate_cpd('tasty'))  # conditional
```

```
+---------------+-----+
| fruit(apple)  | 0.5 |
+---------------+-----+
| fruit(banana) | 0.5 |
+---------------+-----+
+------------+--------------+--------------------+---------------------+--------
-------+
| fruit      | fruit(apple) | fruit(apple)       | fruit(banana)       |
fruit(banana) |
+------------+--------------+--------------------+---------------------+--------
-------+
| size       | size(large)  | size(small)        | size(large)         |
size(small)   |
+------------+--------------+--------------------+---------------------+--------
-------+
| tasty(no)  | 0.25         | 0.3333333333333333 | 0.16666666666666666 | 1.0
|
+------------+--------------+--------------------+---------------------+--------
-------+
| tasty(yes) | 0.75         | 0.6666666666666666 | 0.8333333333333334  | 0.0
|
```

```
+-----------+-------------+------------------+--------------------+--------
-------+
```

[21]:
```
# Bayes'sche Netzwerk
# Calibrate all CPDs of `model` using MLE:

model.fit(data, estimator=MaximumLikelihoodEstimator)
```

[22]:
```
from pgmpy.estimators import BayesianEstimator

est = BayesianEstimator(model, data)

print(est.estimate_cpd('tasty', equivalent_sample_size=10, prior_type='BDeu'))
```

```
+-----------+--------------------+--------------------+--------------------+--
------------------+
| fruit     | fruit(apple)       | fruit(apple)       | fruit(banana)      |
fruit(banana)       |
+-----------+--------------------+--------------------+--------------------+--
------------------+
| size      | size(large)        | size(small)        | size(large)        |
size(small)         |
+-----------+--------------------+--------------------+--------------------+--
------------------+
| tasty(no) | 0.34615384615384615 | 0.4090909090909091 | 0.2647058823529412 |
0.6428571428571429  |
+-----------+--------------------+--------------------+--------------------+--
------------------+
| tasty(yes) | 0.6538461538461539  | 0.5909090909090909 | 0.7352941176470589 |
0.35714285714285715 |
+-----------+--------------------+--------------------+--------------------+--
------------------+
```

2.2 Beispiel 2

[23]:
```
# Import notwendige Packete
import numpy as np
import pandas as pd
from pgmpy.models import BayesianModel
from pgmpy.estimators import BayesianEstimator

# generate data
data = pd.DataFrame(np.random.randint(low=0, high=2, size=(5000, 4)),␣
 ↪columns=['A', 'B', 'C', 'D'])
model = BayesianModel([('A', 'B'), ('A', 'C'), ('D', 'C'), ('B', 'D')]) #␣
 ↪D-->C<--A-->B-->D
```

```
model.fit(data, estimator=BayesianEstimator, prior_type="BDeu") # default␣
 ↪equivalent_sample_size=5
for cpd in model.get_cpds():
    print(cpd)
```

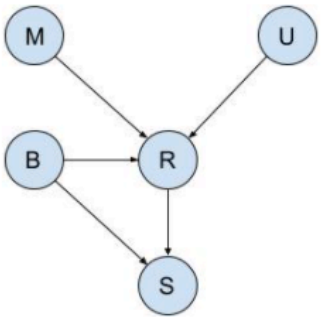
```
+------+----------+
| A(0) | 0.484615 |
+------+----------+
| A(1) | 0.515385 |
+------+----------+
+------+--------------------+--------------------+
| A    | A(0)               | A(1)               |
+------+--------------------+--------------------+
| B(0) | 0.49732014017728304 | 0.4978678038379531 |
+------+--------------------+--------------------+
| B(1) | 0.502679859822717  | 0.502132196162047  |
+------+--------------------+--------------------+
+------+--------------------+--------------------+--------------------+-------
-------------+
| A    | A(0)               | A(0)               | A(1)               | A(1)
|
+------+--------------------+--------------------+--------------------+-------
-------------+
| D    | D(0)               | D(1)               | D(0)               | D(1)
|
+------+--------------------+--------------------+--------------------+-------
-------------+
| C(0) | 0.5116351547891128 | 0.528226631212927  | 0.48825979928043933 |
0.4888822711931705 |
+------+--------------------+--------------------+--------------------+-------
-------------+
| C(1) | 0.4883648452108872 | 0.47177336878707304 | 0.5117402007195607  |
0.5111177288068295 |
+------+--------------------+--------------------+--------------------+-------
-------------+
+------+--------------------+--------------------+
| B    | B(0)               | B(1)               |
+------+--------------------+--------------------+
| D(0) | 0.5028106805862277 | 0.5055677072976735 |
+------+--------------------+--------------------+
| D(1) | 0.4971893194137723 | 0.4944322927023265 |
+------+--------------------+--------------------+
```

2.3 Beispiel 3

"""M : Eine Vorhersage von einem ML-Modell, das den Inhalt lesen kann und eine Punktzahl (Wahrscheinlichkeit) angibt, dass dieser Inhalt gekennzeichnet werden sollte. U : Ein anderer Benutzer kennzeichnet den Inhalt. B : Das Konto wurde zuvor wegen eines schlechten Inhalts gesperrt. R : Punktzahl (Wahrscheinlichkeit), dass der Inhalt von der Plattform entfernt werden sollte. S : Score (Wahrscheinlichkeit), dass das Konto gesperrt werden sollte. Gehen wir davon aus, dass die Wahrscheinlichkeiten für das Netzwerk wie folgt gegeben sind:"""

[27]:
```
from IPython.display import Image
# Load image from local storage
Image(filename = '/Users/h4/Desktop/Screenshot 2021-06-02 at 09.20.49.png')
```

[27]:



U=	T	F
P(U)	0.15	0.85

M=	T	F
P(M)	0.05	0.95

B=	T	F
P(B)	0.10	0.90

S=	T	F
P(S|RB)	0.40	0.60
P(S|R!B)	0.05	0.95
P(S|!RB)	0.12	0.88
P(S|!R!B)	0.02	0.98

R=	T	F
P(R|MBU)	0.95	0.05
P(R|MB!U)	0.90	0.10
P(R|M!BU)	0.85	0.15
P(R|M!B!U)	0.76	0.24
P(R|!MBU)	0.18	0.82
P(R|!MB!U)	0.06	0.94
P(R|!M!BU)	0.14	0.86
P(R|!M!B!U)	0.04	0.96

[29]:
```
from pgmpy.models import BayesianModel
from pgmpy.factors.discrete import TabularCPD
from pgmpy.inference import VariableElimination
import numpy as np

bayesNet = BayesianModel()

# Gestalten wird das Netzwerk

# Knoten + Verbindungen

bayesNet.add_node('M')
bayesNet.add_node('U')
bayesNet.add_node('R')
bayesNet.add_node('B')
bayesNet.add_node('S')

bayesNet.add_edge('M', 'R') #Verbindung geht von M an R
bayesNet.add_edge('U', 'R')
bayesNet.add_edge('B', 'R')
bayesNet.add_edge('B', 'S')
bayesNet.add_edge('R', 'S')

# Conditional Probabilities

# Die Knoten ohne "Eltern" haben 2^1 Wahrscheinlichkeiten

cpd_M = TabularCPD('M', 2, values = [[0.95], [0.05]]) # Knote = M kann 2␣
 ↪Variablen nehmen, Wahrhscheinlichkeit F, T.

cpd_U = TabularCPD('U', 2, values = [[0.85], [0.15]])

cpd_B = TabularCPD('B', 2, values=[[.90], [.10]])

# Knoten mit 1 Eltern haben 2^2 Wahrscheinlichkeiten

cpd_S = TabularCPD('S', #Knote
                   2,   #Anzahln Variablen von S
                   values = [[0.98, 0.88, 0.95, 0.6], [0.02, 0.12, 0.05, 0.4]],␣
 ↪#Bedingten Wahrscheinlichkeiten F, T
                   evidence=['R', 'B'], # die Knoten die Eltern von R
                   evidence_card=[2,2]) # die jeweiligen States von den␣
 ↪Evindence Knoten

# Knoten mit 2 Eltern haben 2^3 Wahrscheinlichkeiten
```

```
cpd_R = TabularCPD('R', 2,
                   values=[[0.96, .86, .94, .82, .24, .15, .10, .05], [.04, .
 ↪14, .06, .18, .76, .85, .90, .95]],
                   evidence=['M', 'B', 'U'],
                   evidence_card=[2, 2,2])

bayesNet.add_cpds(cpd_M, cpd_U, cpd_B, cpd_S, cpd_R)
```

[30]:
```
bayesNet.check_model()
print("Model is correct.")
```

```
Model is correct.
```

[31]:
```
solver = VariableElimination(bayesNet)
```

[32]:
```
result = solver.query(variables=['R'])
print(result)
```

```
Finding Elimination Order: :   0%|          | 0/4 [00:00<?, ?it/s]
  0%|          | 0/4 [00:00<?, ?it/s]
Eliminating: S:   0%|          | 0/4 [00:00<?, ?it/s]
Eliminating: M:   0%|          | 0/4 [00:00<?, ?it/s]
Eliminating: U:   0%|          | 0/4 [00:00<?, ?it/s]
Eliminating: B: 100%|     | 4/4 [00:00<00:00, 205.03it/s]
+------+----------+
| R    |   phi(R) |
+======+==========+
| R(0) |   0.9062 |
+------+----------+
| R(1) |   0.0938 |
+------+----------+
```

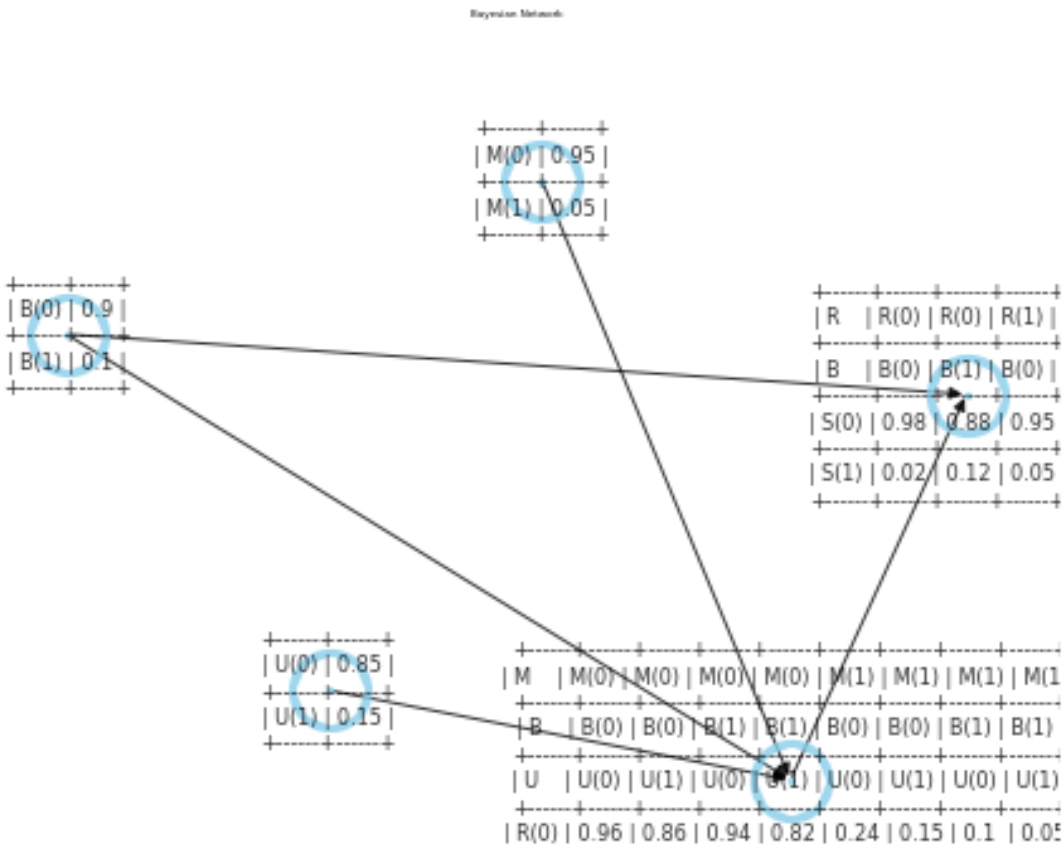
[33]:
```
"""P(R)=
         P(R|MBU)P(M)P(B)P(U)+
         P(R|MBU)P(M)P(B)P(!U)+
         P(R|MBU)P(M)P(!B)P(U)+
         P(R|MBU)P(M)P(!B)P(!U)+
         P(R|MBU)P(!M)P(B)P(U)+
         P(R|MBU)P(!M)P(B)P(!U)+
         P(R|MBU)P(!M)P(!B)P(U)+
         P(R|MBU)P(!M)P(!B)P(!U) ---
         [Using total probability theorem as R depends on M, B, U] =
         0.950.050.10.15+
         0.90.050.10.85+
         0.850.050.90.15+
```

```
        0.760.050.90.85+
        0.180.950.10.15+
        0.060.950.10.85+
        0.140.950.90.15+
        0.040.950.90.85=
        0.09378
"""
```

[33]:
```
'P(R) =P(R|MBU)P(M)P(B)P(U)+P(R|MBU)P(M)P(B)P(!U)+P(R|MBU)P(M)P(!B)P(U)
+P(R|MBU)P(M)P(!B)P(!U)+P(R|MBU)P(!M)P(B)P(U)+P(R|MBU)P(!M)P(B)P(!U)
+P(R|MBU)P(!M)P(!B)P(U)+P(R|MBU)P(!M)P(!B)P(!U) --- [Using total probability
theorem as R depends on M, B, U] =0.950.050.10.15+0.90.050.10.85+0.850.050.90.15
+0.760.050.90.85+0.180.950.10.15+0.060.950.10.85
+0.140.950.90.15+0.040.950.90.85 =0.09378\n'
```

[34]:
```
import networkx as nx
import pylab as plt

labeldict = {}
labeldict["M"] = cpd_M
labeldict["U"] = cpd_U
labeldict["B"] = cpd_B
labeldict["R"] = cpd_R
labeldict["S"] = cpd_S

nx.draw(bayesNet,
        labels=labeldict,
        with_labels=True,
        font_size=8,
        node_size=10,
        node_color="skyblue",
        node_shape="o",
        alpha=0.8,
        linewidths=30,
        pos=nx.spring_layout(bayesNet))

plt.title("Bayesian Network", fontsize=4, y=1.12)

plt.show()
```

```
Finding Elimination Order: : 100%|      | 4/4 [02:23<00:00, 35.82s/it]
```



[35]:
```
"""Lassen Sie uns die Wahrscheinlichkeit für "Inhalt sollte von der Plattform␣
↪entfernt werden, wenn unser ML-Modell ihn markiert" ermitteln"""
```

[35]:
```
'Lassen Sie uns die Wahrscheinlichkeit für "Inhalt sollte von der Plattform
entfernt werden, wenn unser ML-Modell ihn markiert" ermitteln'
```

[37]:
```
"""P(R|M)=0.95*0.1*0.15 + 0.9*0.1*0.85 +0.85*0.9*0.15 + 0.76*0.9*0.85 = 0.
↪7869"""
```

[37]:
```
'P(R|M)=0.95*0.1*0.15 + 0.9*0.1*0.85 +0.85*0.9*0.15 + 0.76*0.9*0.85 = 0.7869'
```

[38]:
```
result = solver.query(variables=['R'], evidence={'M': 1})
print("R| M", result)
```

```
Finding Elimination Order: :   0%|          | 0/3 [00:00<?, ?it/s]
  0%|          | 0/3 [00:00<?, ?it/s]
Eliminating: S:   0%|          | 0/3 [00:00<?, ?it/s]
Eliminating: U:   0%|          | 0/3 [00:00<?, ?it/s]
Finding Elimination Order: : 100%|      | 3/3 [00:00<00:00, 159.15it/s]
Eliminating: B: 100%|      | 3/3 [00:00<00:00, 205.52it/s]
```

```
R| M +------+----------+
| R    |   phi(R) |
+======+==========+
| R(0) |   0.2131 |
+------+----------+
| R(1) |   0.7869 |
+------+----------+
```

[39]:
```
"""Zum Beispiel finden wir "Konto sollte gesperrt werden, da es zuvor gesperrt␣
↪wurde"""
```

[39]:
```
'Zum Beispiel finden wir "Konto sollte gesperrt werden, da es zuvor gesperrt
wurde'
```

[40]:
```
"""P(S|B)"""

result = solver.query(variables=['S'], evidence={'M':1})
print("S|B", result)
```

```
Finding Elimination Order: :   0%|          | 0/3 [00:00<?, ?it/s]
  0%|          | 0/3 [00:00<?, ?it/s]
Eliminating: U:   0%|          | 0/3 [00:00<?, ?it/s]
Eliminating: R:   0%|          | 0/3 [00:00<?, ?it/s]
Eliminating: B: 100%|     | 3/3 [00:00<00:00, 268.33it/s]
```

```
S|B +------+----------+
| S    |   phi(S) |
+======+==========+
| S(0) |   0.9237 |
+------+----------+
| S(1) |   0.0763 |
+------+----------+
```

[ ]:
```

```