Probability & Bayesian Networks (Part A)

Gloria Beraldo (gloria.beraldo@unipd.it)
Department of Information Engineering, University of Padova

Topics:

- Working with probabilities in Python
- Extract Probability from Data
- Bayesian Network in Python

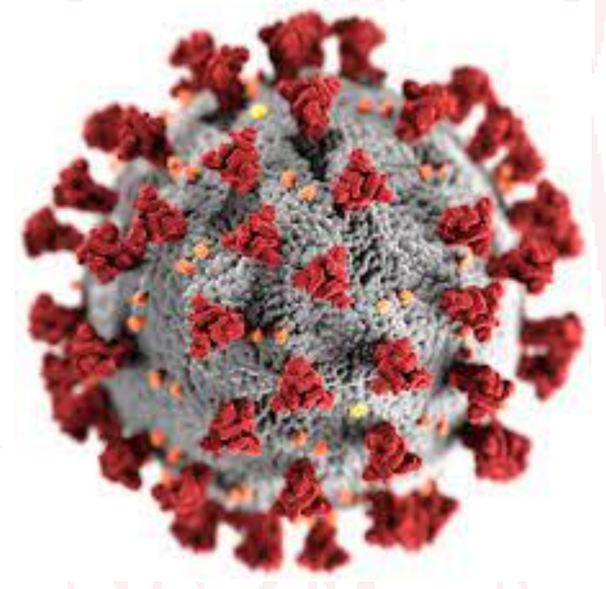
DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Working with probabilities in Python

Let's assume we want to calculate the probability of having COVID-19 based on three possible symptoms:
* high fever,
* continuous cough,
* loss of taste/smell.

We know there are 1% chances of being infected, and in such case probabilities of developing the symptoms are as follows:
- high fever: 10% with COVID, only 2% without
- continuous cough: 20% with COVID, only 5% without
- loss of taste/smell: 15% with COVID, only 0.1% without

QUESTION: What is the probability of having COVID given the presence of all the three symptoms?



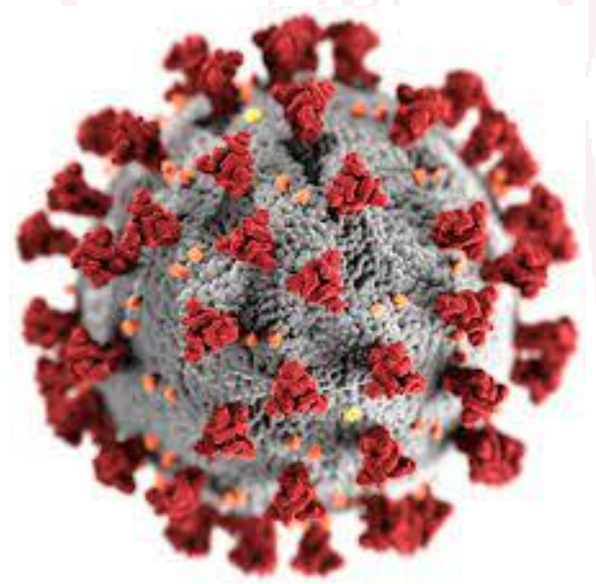
Working with probabilities in Python

We have **four random variables**, which can be either **true** or **false:**
- COVID disease D
- fever F
- cough C
- loss of taste/smell L

For simplicity, we will use lower-case letters to indicate single events:
e.g., d means $D = true$ and $\neg d$ means $D = false$

QUESTION: What is the probability of having COVID given the presence of all the three symptoms?

The task then consists in computing the probability $P(d \mid f, c, l)$.



Working with probabilities in Python

We can write the following probability distributions, given by the problem:

$P(d) = 0.01 \quad P(\neg d) = 0.99$

or simply $P(D) = \langle 0.01, 0.99 \rangle$

Similarly, for the conditional probabilities of the symptoms:

$P(F|d) = \langle 0.1, 0.9 \rangle$

$P(F|\neg d) = \langle 0.02, 0.98 \rangle$

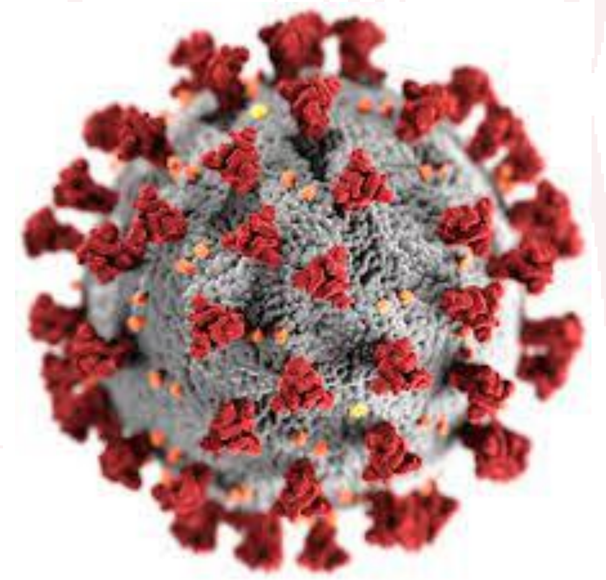
$P(C|d) = \langle 0.2, 0.8 \rangle$

$P(C|\neg d) = \langle 0.05, 0.95 \rangle$

$P(L|d) = \langle 0.15, 0.85 \rangle$

$P(L|\neg d) = \langle 0.001, 0.999 \rangle$

We know there are **1% chances** of being **infected**, and in such case the probabilities of developing the symptoms are as follows:
- high fever: 10% with COVID, only 2% without
- continuous cough: 20% with COVID, only 5% without
- loss of taste/smell: 15% with COVID, only 0.1% without



Working with probabilities in Python

Let's put these values Python arrays:

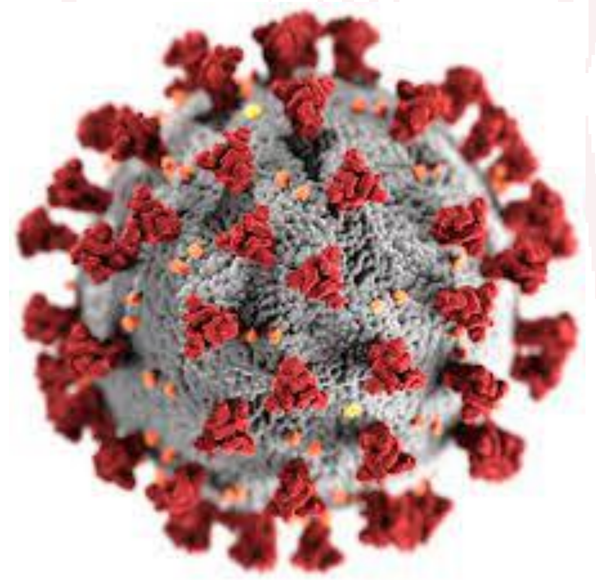
```
import numpy as np

p_disease = np.array([0.01, 0.99])
print('P(D) = ', p_disease)

p_fever_disease = np.array([[0.1, 0.9], [0.02, 0.98]])
print('\nP(F|D) =\n', p_fever_disease)

p_cough_disease = np.array([[0.2, 0.8], [0.05, 0.95]])
print('\nP(C|D) =\n', p_cough_disease)

p_loss_disease = np.array([[0.15, 0.85], [0.001, 0.999]])
print('\nP(L|D) =\n', p_loss_disease)
```

```
P(D) =  [0.01 0.99]

P(F|D) =
 [[0.1  0.9 ]
 [0.02 0.98]]

P(C|D) =
 [[0.2  0.8 ]
 [0.05 0.95]]

P(L|D) =
 [[0.15  0.85 ]
 [0.001 0.999]]
```

$\mathbf{P}(F|d) = \langle 0.1, 0.9\rangle$
$\mathbf{P}(F|\neg d) = \langle 0.02, 0.98\rangle$

$\mathbf{P}(C|d) = \langle 0.2, 0.8\rangle$
$\mathbf{P}(C|\neg d) = \langle 0.05, 0.95\rangle$

$\mathbf{P}(L|d) = \langle 0.15,0.85\rangle$
$\mathbf{P}(L|\neg d) = \langle 0.001,0.999\rangle$



Working with probabilities in Python

Applying Bayes rule, we can write:

$$P(d|f,c,l) = \frac{P(f,c,l|d)P(d)}{P(f,c,l)}$$

At the numerator, we can exploit the fact that **the three symptoms are conditionally independent**, given the disease (i.e. if I know I have COVID, my chances of having a fever do not change by the fact of having also a cough or loss of taste/smell).

Therefore $P(f,c,l|d)P(d) = P(f|d)P(c|d)P(l|d)P(d)$

We can also apply **the law of total probability to the denominator** and write:

$$P(f,c,l) = \sum_{x \in D} P(f,c,l|x)P(x)$$

$$= P(f,c,l|d)P(d) + P(f,c,l|\neg d)P(\neg d)$$

$$= P(f|d)P(c|d)P(l|d)P(d) + P(f|\neg d)P(c|\neg d)P(l|\neg d)P(\neg d).$$

Working with probabilities in Python

By substituting the above expressions for the numerator and denominator, we can write the answer to the original question:

$$P(d|f,c,l) = \frac{P(f,c,l|d)P(d)}{P(f,c,l)}$$

$$P(d|f,c,l) = \frac{P(f|d)P(c|d)P(l|d)P(d)}{P(f|d)P(c|d)P(l|d)P(d)+P(f|\neg d)P(c|\neg d)P(l|\neg d)P(\neg d).}$$

Working with probabilities in Python

Let's implement this in Python to compute the actual probability value

Starting from the numerator:

```
numerator = p_fever_disease[0,0] * p_cough_disease[0,0] * p_loss_disease[0,0] * p_disease[0]
print('P(f,c,l|d) P(d) =\n', numerator)
```

```
P(f,c,l|d) P(d) =
 3.0000000000000004e-05
```

At the denominator we have the sum of the symptoms probability with and without disease:

```
denominator = p_fever_disease[0,0] * p_cough_disease[0,0] * p_loss_disease[0,0] * p_disease[0] +\
    p_fever_disease[1,0] * p_cough_disease[1,0] * p_loss_disease[1,0] * p_disease[1]
print('P(f,c,l|d)P(d) + P(f,c,l|¬d)P(¬d) =\n', denominator)
```

```
P(f,c,l|d)P(d) + P(f,c,l|¬d)P(¬d) =
 3.0990000000000007e-05
```

Finally, the probability of the disease given all the symptoms are present is the following

```
result = numerator / denominator
print('P(d|f,c,l) =\n', result)
```

```
P(d|f,c,l) =
 0.9680542110358179
```

So the actual probability of having COVID, given the presence of all the three symptoms, is almost 97%!

Extract Probability from Data

Let's see in practice how you could extract some probabilities from data.

Consider the following table, listing 10 students and their final grades (either A, B, or C) obtained in Year 1, Year 2 and Year 3:

Student	Grade Y1	Grade Y2	Grade Y3
John	A	A	B
Sarah	C	C	B
Eric	A	B	B
Paul	B	C	A
Susanne	A	A	A
Beth	B	A	B
Jack	B	C	B
Rachel	B	A	A
Tom	B	C	C
Jenny	B	A	B

Assuming three random variables, G_1, G_2, and G_3, to represent the grades at each year,

how can we compute some useful probabilities from this table like, for example, $P(G_1=A)$ or $P(G_1=B,G_2=C)$?

Extract Probability from data

Let's count the instances in the table:

- in Year 1 there are three A grades out of ten,

therefore P(G1=A)=3/10=0.3

- we can also see there are only three cases in which G1=B and G2=C,

therefore P(G1=B,G2=C)=0.3.

A little more complicated is to extract a conditional probability.
For example, what about P(G3=A|G2=C)?

- there are four cases in which G2=C (Sarah, Paul, Jack, and Tom)
- among these, **only Paul got an A** in Year 3,

therefore P(G3=A|G2=C)=1/4=0.25.

Let's see if this could also be calculated differently:

- there is **only one case (Paul) in which G2=C and G3=A**,

therefore P(G2=C,G3=A)=1/10

- also, the probability P(G2=C)=4/10

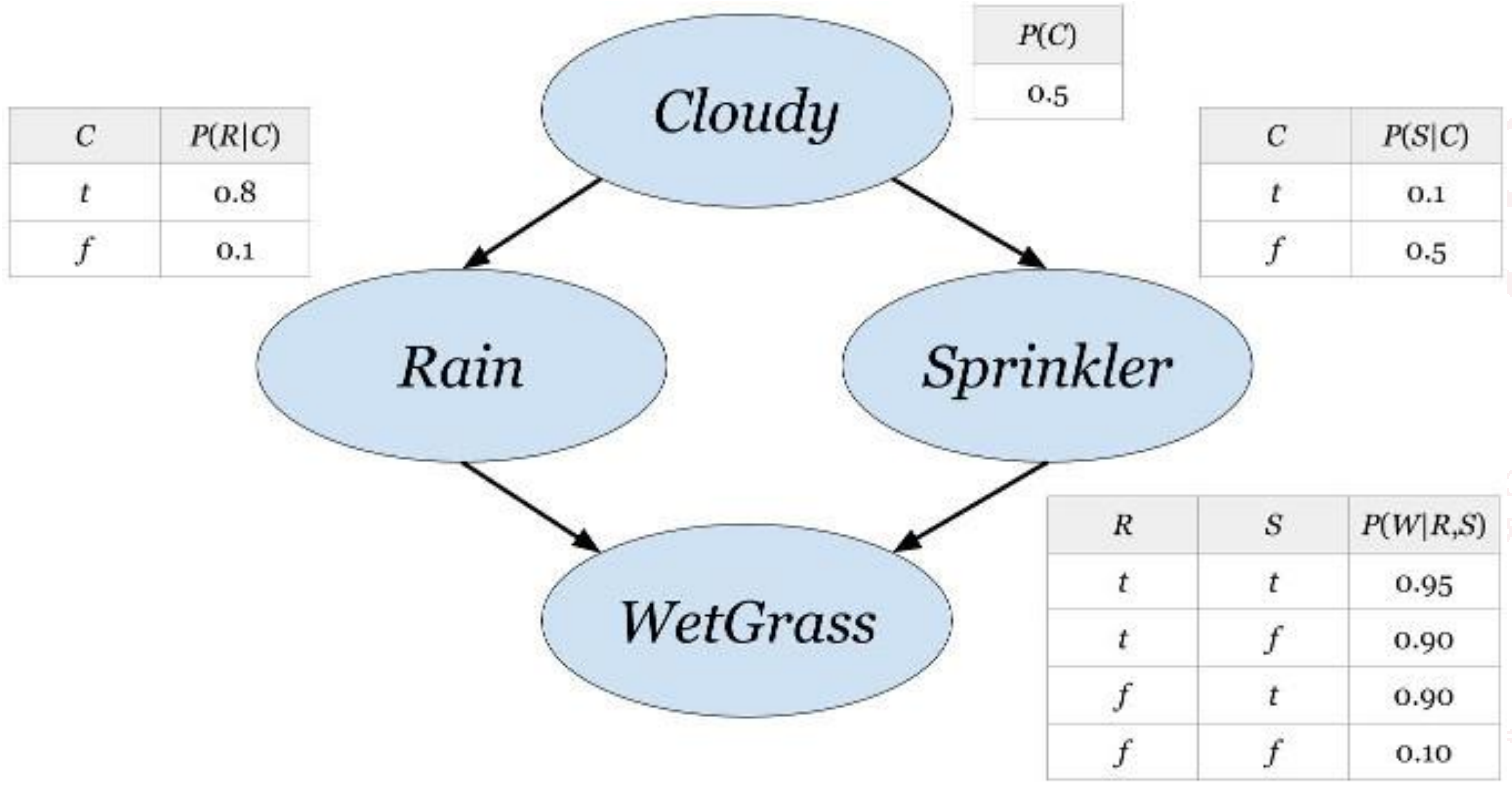
- therefore $P(G3=A|G2=C) = \frac{P(G2=C,G3=A)}{P(G2=C)} = \frac{1}{10} * \frac{10}{4} = \frac{1}{4} = 0.25$

Student	Grade Y1	Grade Y2	Grade Y3
John	A	A	B
Sarah	C	C	B
Eric	A	B	B
Paul	B	C	A
Susanne	A	A	A
Beth	B	A	B
Jack	B	C	B
Rachel	B	A	A
Tom	B	C	C
Jenny	B	A	B

The above example is based on the assumption that the given data captures the real probability distribution of student grades *P(G1, G2, G3)*. In general this is not true. Typically, the more data you have, the better it is, as long as the samples cover sufficiently well the underlying probability distribution.

Bayesian Network in Python

Let's consider the scenario in the figure, where the WetGrass can be caused a Sprinkler or the Rain, both of which depend on the Cloudy weather:



P(C)
0.5

C	P(R|C)
t	0.8
f	0.1

C	P(S|C)
t	0.1
f	0.5

R	S	P(W|R,S)
t	t	0.95
t	f	0.90
f	t	0.90
f	f	0.10

We want to compute the probability distribution of the wet grass given that is cloudy.

Bayesian Network in Python

We want to compute the probability distribution of the wet grass given that is cloudy.

Let's start by exploiting the structure of the Bayesian Network
to decompose the query:

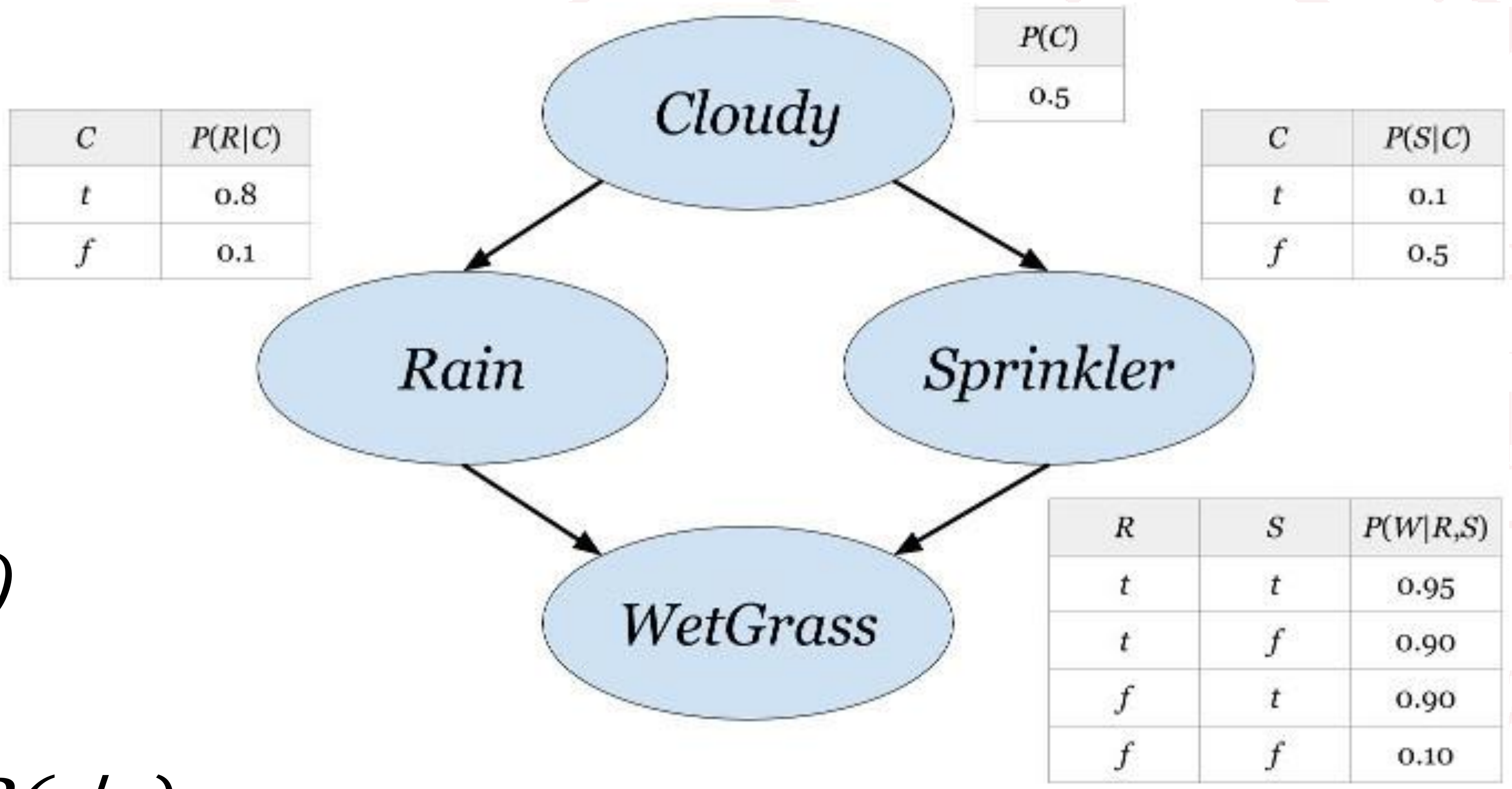
$$\mathbf{P}(W|c) = \alpha\, \mathbf{P}(W,c)$$

α normalization constant

$$= \alpha \sum_{r,s} \mathbf{P}(W,c,r,s)$$

$$= \alpha \sum_{r,s} \mathbf{P}(W|r,s)P(r|c)P(s|c)P(c)$$

$$= \alpha\, P(c) \sum_{r} P(r|c) \sum_{s} \mathbf{P}(W|r,s)P(s|c)$$



P(C)
0.5

C	P(R|C)
t	0.8
f	0.1

C	P(S|C)
t	0.1
f	0.5

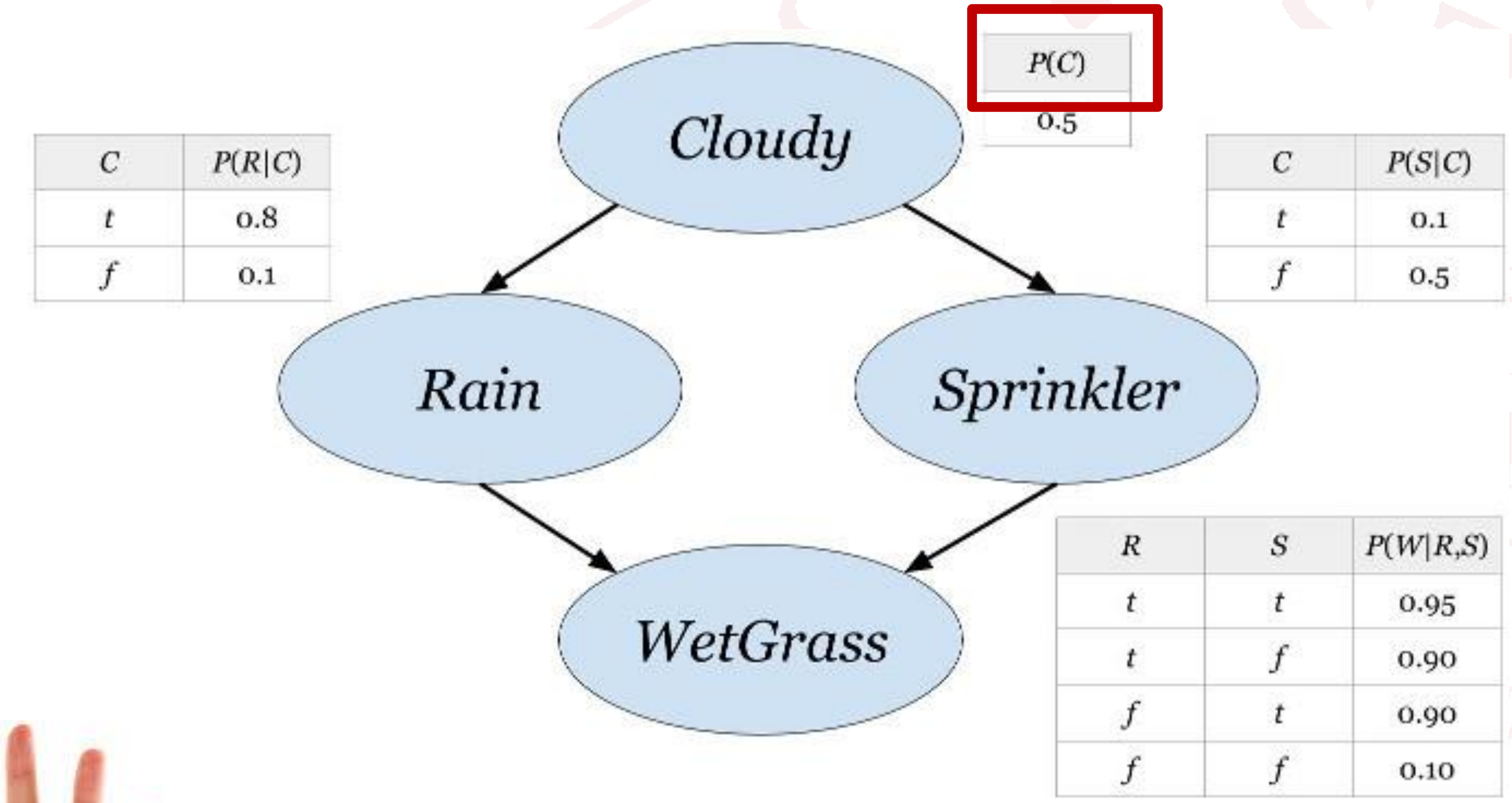
R	S	P(W|R,S)
t	t	0.95
t	f	0.90
f	t	0.90
f	f	0.10

Bayesian Network in Python

$$P(W|c) = \alpha \, P(c) \textstyle\sum_r P(r|c) \sum_s P(W|r,s)P(s|c)$$

$$P(c) = 0.5$$

```
P_c = 0.5
```



P(C)
0.5

C	P(R|C)
t	0.8
f	0.1

C	P(S|C)
t	0.1
f	0.5

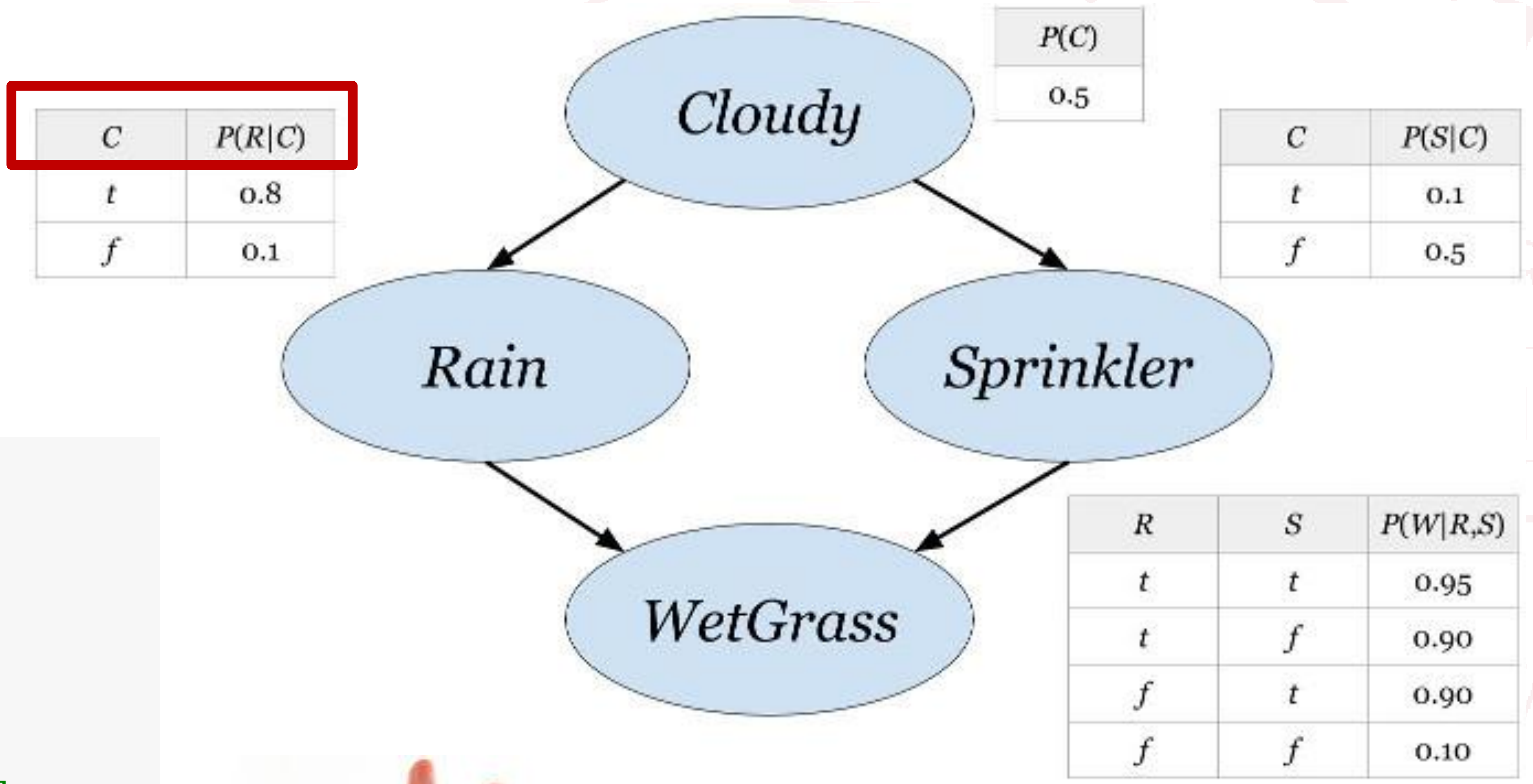
R	S	P(W|R,S)
t	t	0.95
t	f	0.90
f	t	0.90
f	f	0.10

Bayesian Network in Python

$$P(W|c) = \alpha P(c) \sum_r \boxed{P(r|c)} \sum_s P(W|r,s)P(s|c)$$

$$P(R|c) = \langle P(r|c), P(\neg r|c) \rangle$$

```python
import numpy as np

# 'true' and 'false' indexes
t, f = 0, 1

P_R_c = np.array([0.8, 0.2])
# this is a 2D vector, the elements of which can be accessed as follows
print('P(r|c) = ', P_R_c[t])
print('P(¬r|c) = ', P_R_c[f])
```



P(C)
0.5

C	P(R|C)
t	0.8
f	0.1

C	P(S|C)
t	0.1
f	0.5

R	S	P(W|R,S)
t	t	0.95
t	f	0.90
f	t	0.90
f	f	0.10

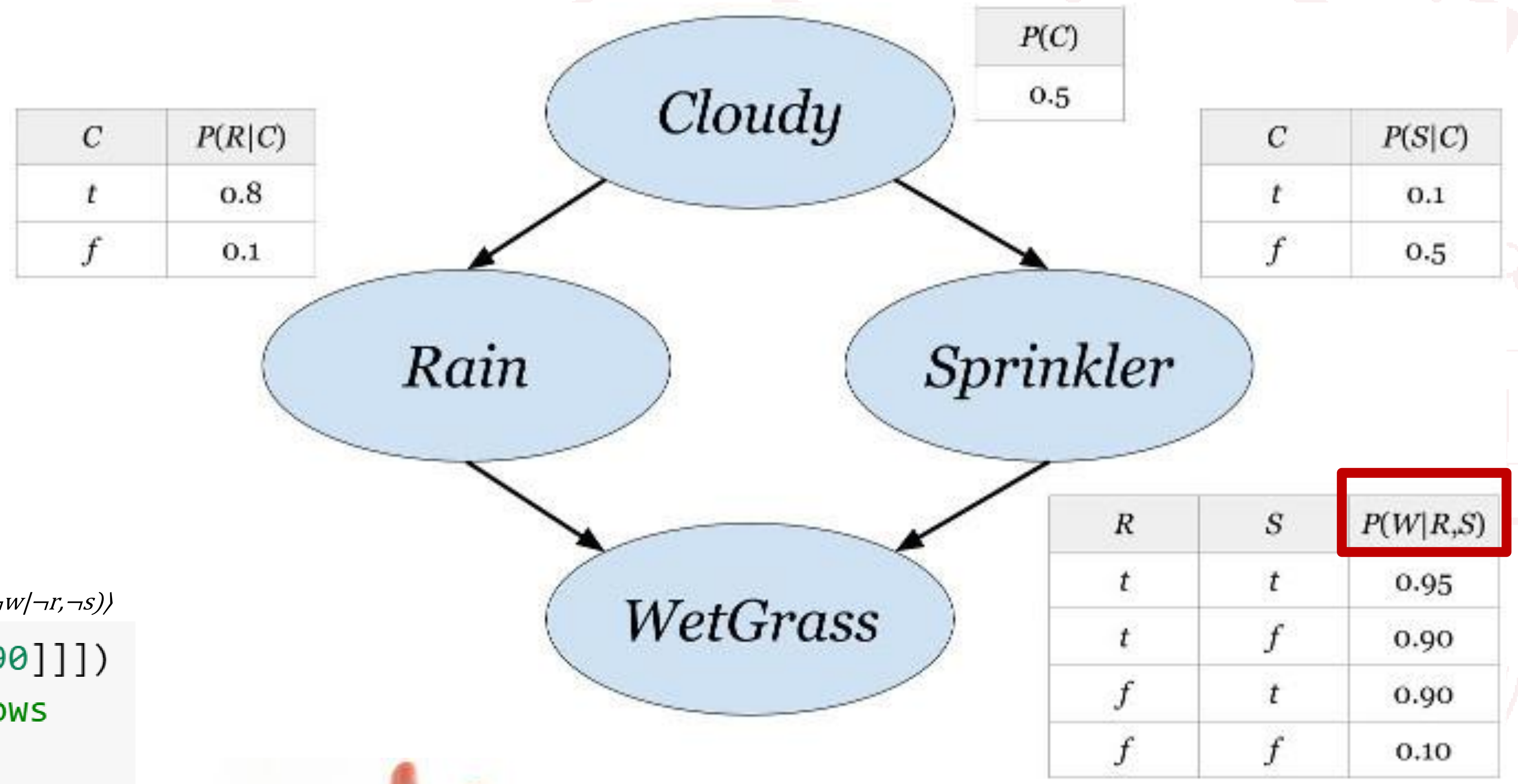
Bayesian Network in Python

$$P(W|c) = \alpha\, P(c) \sum_r P(r|c) \sum_s \boxed{P(W|r,s)}\, P(s|c)$$

$$P(W|R,S)$$

$\langle W(w|r,s), W(w|r,\neg s)\rangle$ $\langle W(w|\neg r,s), W(w|\neg r,\neg s)\rangle$ $\langle W(\neg w|r,s), W(\neg w|r,\neg s)\rangle$ $W(\neg w|\neg r,s), W(\neg w|\neg r,\neg s)\rangle$

```
P_W_RS = np.array([[[0.95, 0.90],[0.90, 0.10]],[[0.05, 0.10], [0.10, 0.90]]])
# this is a 2x2x2 matrix, the elements of which can be accessed as follows
print('P(w|¬r,s) = ', P_W_RS[t,f,t])
```



P(C)
0.5

C	P(R|C)
t	0.8
f	0.1

C	P(S|C)
t	0.1
f	0.5

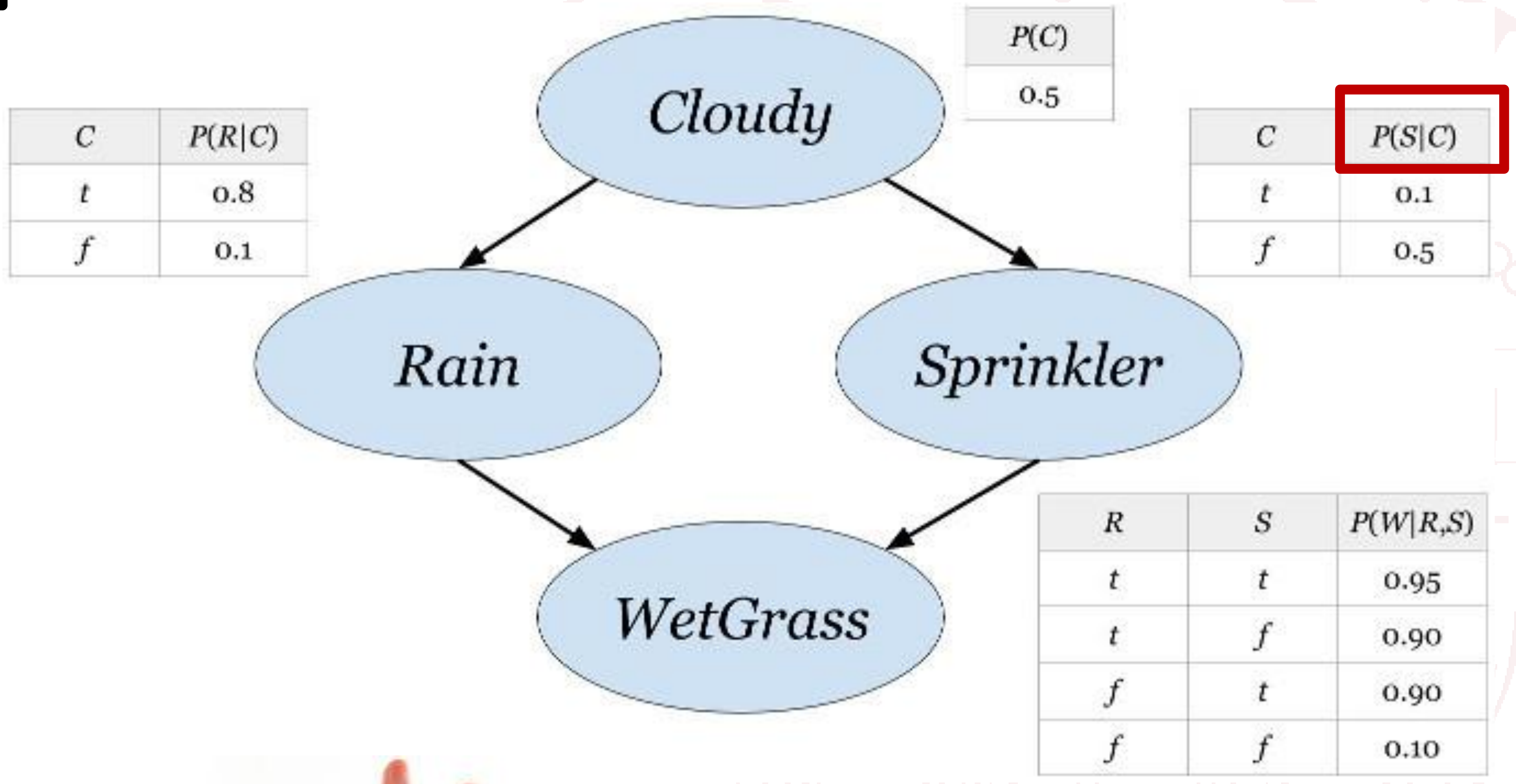
R	S	P(W|R,S)
t	t	0.95
t	f	0.90
f	t	0.90
f	f	0.10

Bayesian Network in Python

$$\mathbf{P}(W|c) = \alpha\, P(c) \sum_r P(r|c) \sum_s \mathbf{P}(W|r,s)\, \boxed{P(s|c)}$$

$$\mathbf{P}(S|c) = \langle P(s|c), P(\neg s|c) \rangle$$

```
P_S_c = np.array([0.1, 0.9])
```



P(C)
0.5

C	P(R|C)
t	0.8
f	0.1

C	P(S|C)
t	0.1
f	0.5

R	S	P(W|R,S)
t	t	0.95
t	f	0.90
f	t	0.90
f	f	0.10

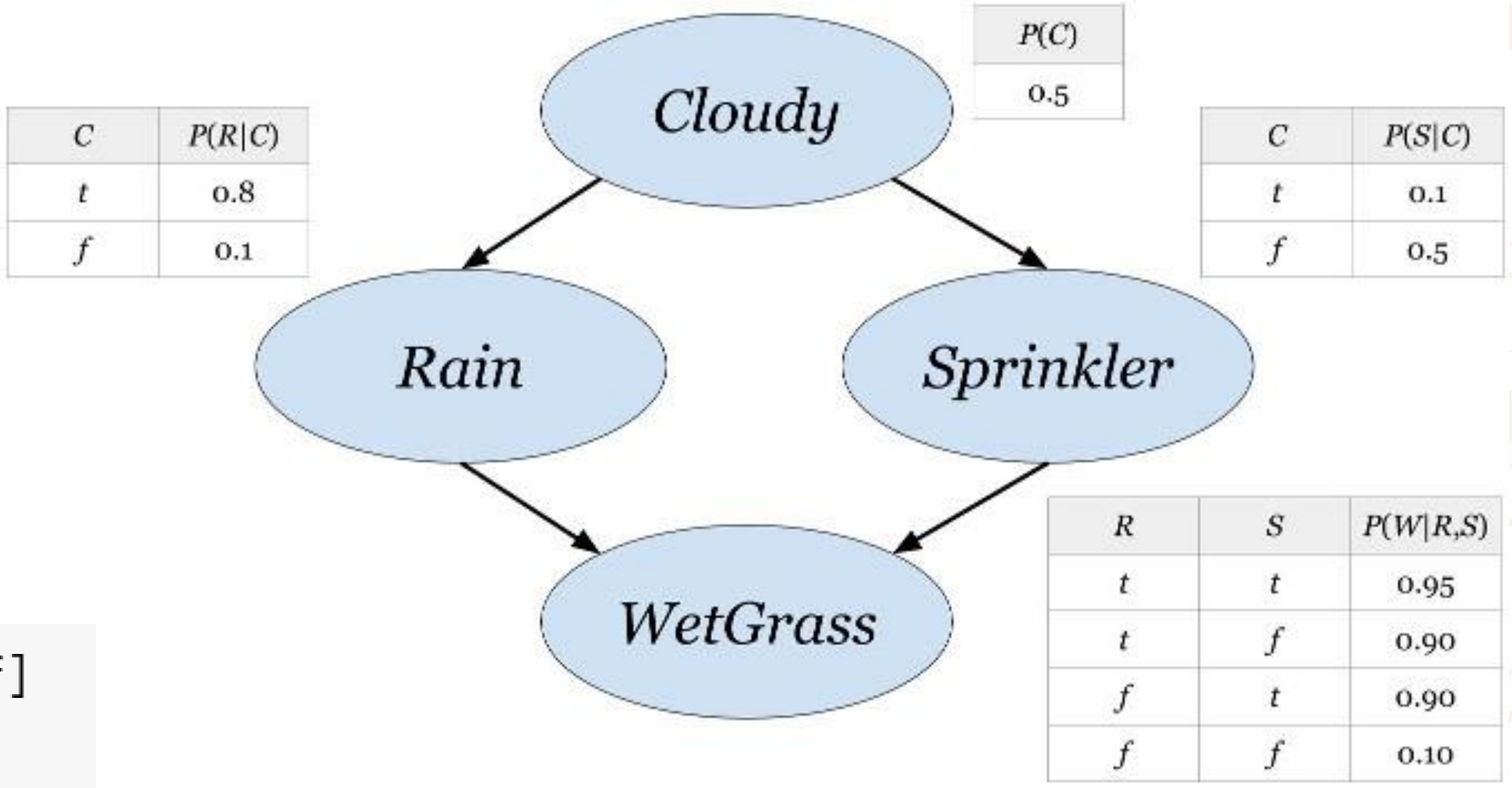
Bayesian Network in Python

$$P(W|c) = \alpha P(c) \sum_r P(r|c) \sum_s P(W|r,s)P(s|c)$$

Starting from the right side of the query equation, we can sum out S by computing

$$P(W|r,s)P(s|c) + P(W|r,\neg s)P(\neg s|c)$$

```
Phi_S = P_W_RS[:,:,t] * P_S_c[t] + P_W_RS[:,:,f] * P_S_c[f]
print(Phi_S)
```



P(C)
0.5

C	P(R|C)
t	0.8
f	0.1

C	P(S|C)
t	0.1
f	0.5

R	S	P(W|R,S)
t	t	0.95
t	f	0.90
f	t	0.90
f	f	0.10

Bayesian Network in Python

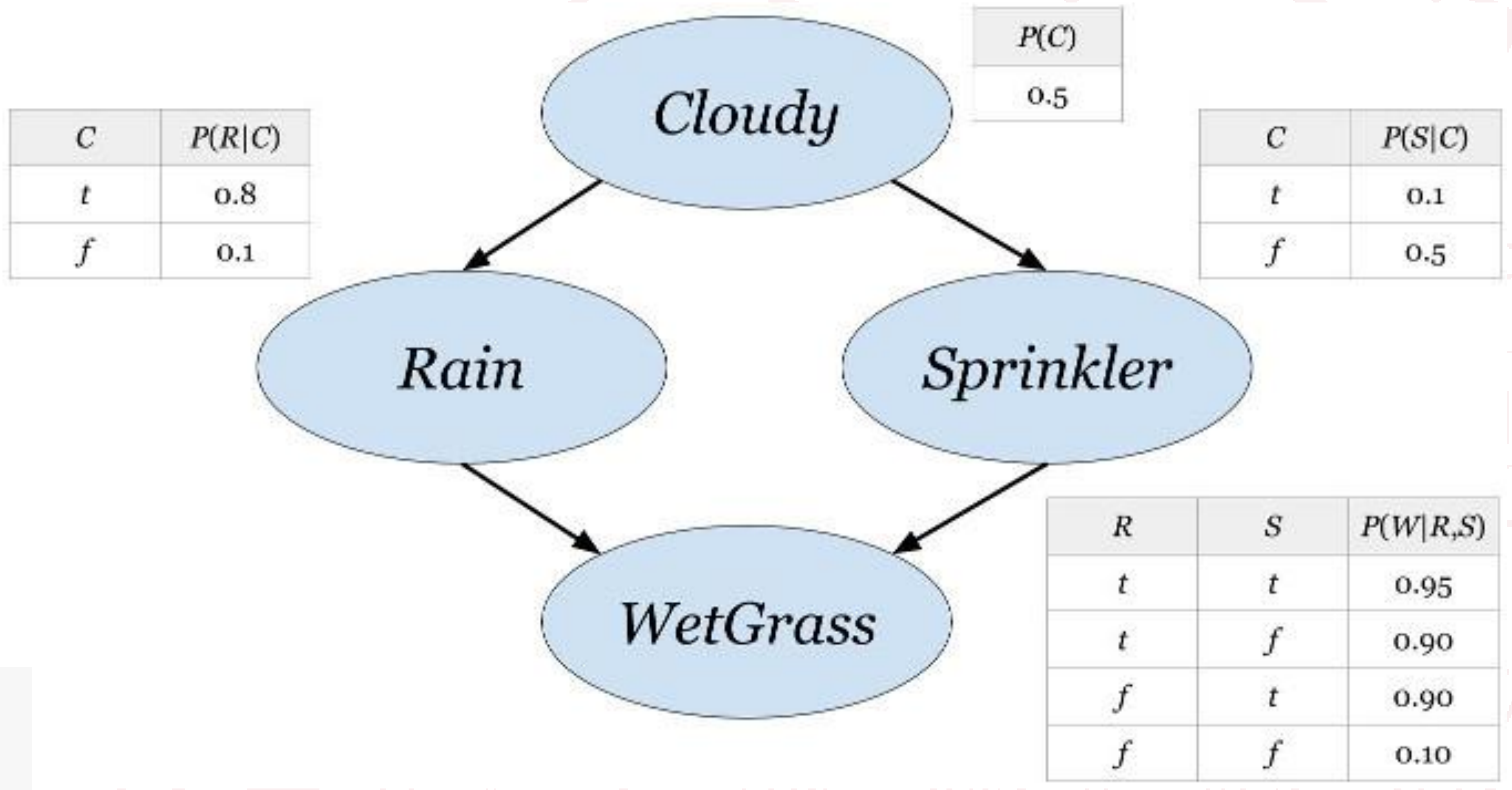
$$\mathbf{P}(W|c) = \alpha\, P(c) \sum_r P(r|c) \sum_s \mathbf{P}(W|r,s) P(s|c)$$

We remain with a 2x2 matrix indexed by W and R, which we call ΦS(W,R) for short.

We can then proceed by summing out R with

$$P(r|c)\Phi S(W,r)+P(\neg r|c)\Phi S(W,\neg r)$$

```
Phi_R = P_R_c[t] * Phi_S[:,t] + P_R_c[f] * Phi_S[:,f]
print(Phi_R)
```



P(C)
0.5

C	P(R|C)
t	0.8
f	0.1

C	P(S|C)
t	0.1
f	0.5

R	S	P(W|R,S)
t	t	0.95
t	f	0.90
f	t	0.90
f	f	0.10

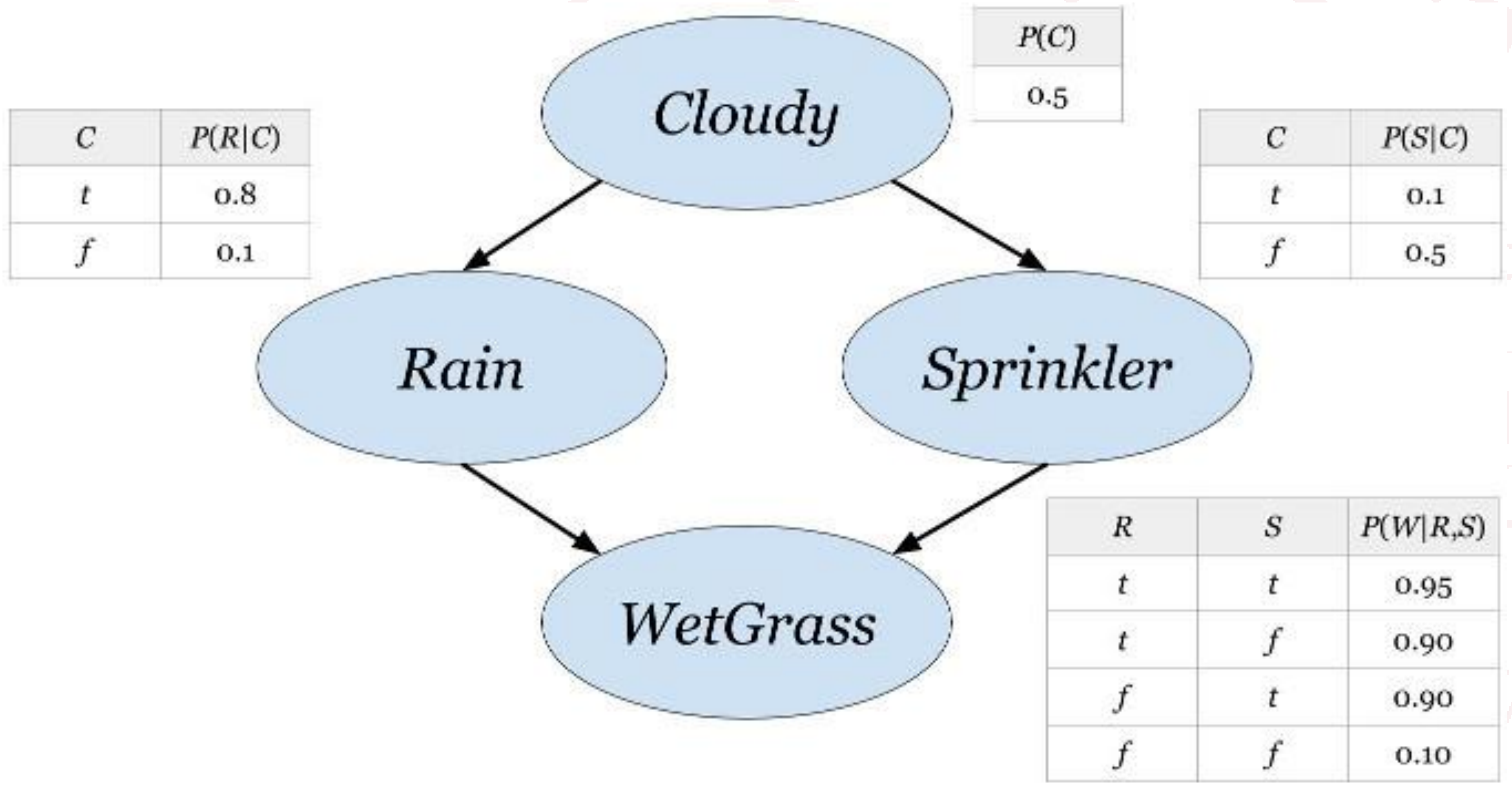
Bayesian Network in Python

$$P(W|c) = \alpha\, P(c) \sum_r P(r|c) \sum_s P(W|r,s) P(s|c)$$

The result is a 2D vector indexed only by W, which we call ΦR(W).

Multiplying the latter for $P(c)$ and normalizing, we obtain the final distribution:

```
P_W_c = P_c * Phi_R
P_W_c = P_W_c / sum(P_W_c)
print('P(W|c) = ', P_W_c)
```



P(C)
0.5

C	P(R|C)
t	0.8
f	0.1

C	P(S|C)
t	0.1
f	0.5

R	S	P(W|R,S)
t	t	0.95
t	f	0.90
f	t	0.90
f	f	0.10

Notice that the last step does not change the result because the previous sum was already normalised and $P(c)$ is just a constant that could be included in α.

Questions