Sieć Bayesowska wspomagająca podejmowanie decyzji o inwestycji

Spis treści

[Treść zadania 2](#_Toc188599770)

[Tworzenie sieci 2](#_Toc188599771)

[Opis Zmiennych 3](#_Toc188599772)

[Global Events - G 3](#_Toc188599773)

[Opis 3](#_Toc188599774)

[Prawdopodobieństwa 3](#_Toc188599775)

[Market Condition - M 4](#_Toc188599776)

[Opis 4](#_Toc188599777)

[Prawdopodobieństwa 4](#_Toc188599778)

[Company Income - I 4](#_Toc188599779)

[Opis 4](#_Toc188599780)

[Prawdopodobieństwa 4](#_Toc188599781)

[Company Reputation - R 4](#_Toc188599782)

[Opis 4](#_Toc188599783)

[Prawdopodobieństwa 4](#_Toc188599784)

[Action Cost Change - C 4](#_Toc188599785)

[Opis 4](#_Toc188599786)

[Prawdopodobieństwa 5](#_Toc188599787)

[Investment Decision - D 5](#_Toc188599788)

[Opis 5](#_Toc188599789)

[Prawdopodobieństwa 5](#_Toc188599790)

[Treść zadania 2 6](#_Toc188599791)

[Treść zadania 3 13](#_Toc188599792)

# Treść zadania

A screenshot of a computer

Description automatically generated

# Tworzenie sieci

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Opis Zmiennych

A screenshot of a computer

Description automatically generated

## Global Events - G

### Opis

Prawdopodobieństwo wystąpienia wydarzenia globalnego z danym możliwym wpływem na firmę

### Prawdopodobieństwa

|  |  |  |  |
| --- | --- | --- | --- |
| L.p. | Rodzaj | Wartość | Opis |
| 1. | good | 0.1 | Pozytywnie wpływające wydarzenie |
| 2. | neutral | 0.6 | Wydarzenie neutralne |
| 3. | bad | 0.3 | Negatywnie wpływające wydarzenie |

## Market Condition - M

### Opis

Stan rynku, który może być sprzyjający lub nie

### Prawdopodobieństwa

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| L.p. | Rodzaj | G | Wartość | Opis |
| 1. | positive | good | 0.8 |  |
| 2. | negative | good | 0.2 |  |
| 3. | positive | neutral | 0.5 |  |
| 4. | negative | neutral | 0.5 |  |
| 5. | positive | bad | 0.8 |  |
| 6. | negative | bad | 0.2 |  |

## Company Income - I

### Opis

Ocena satysfakcji przychodu firmy

### Prawdopodobieństwa

|  |  |  |  |
| --- | --- | --- | --- |
| L.p. | Rodzaj | Wartość | Opis |
| 1. | grow | 0.25 | Wzrost przychodu |
| 2. | not\_grow | 0.75 | Brak wzrostu przychodu |

## Company Reputation - R

### Opis

### Prawdopodobieństwa

|  |  |  |  |
| --- | --- | --- | --- |
| L.p. | Rodzaj | Wartość | Opis |
| 1. | positive | 0.6 | Pozytywna reputacja |
| 2. | negative | 0.4 | Negatywna reputacja |

## Action Cost Change - C

### Opis

Prawdopodobieństwo zmiany kosztów akcji

Zależy od stanu rynku, przychodów firmy oraz jej reputacji.

Po uszeregowaniu tych 3 czynników można przypisać większe prawdopodobieństwo tego, że pozytywny element będzie miał znaczenie (dostanie +20, +30 lub +40 procent w zależności od znaczenia)

Oto kolejność:

1. Przychody firmy, +40%
2. Reputacja firmy, +30%
3. Stan rynku, +20%

### Prawdopodobieństwa

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| L.p. | Rodzaj | M (+20%) | I (+40%) | R (+30%) | Wartość | Opis |
| 1. | raise | positive | grow | positive | 0.9 |  |
| 2. | not\_raise | positive | grow | positive | 0.1 |  |
| 3. | raise | positive | grow | negative | 0.6 |  |
| 4. | not\_raise | positive | grow | negative | 0.4 |  |
| 5. | raise | positive | not\_grow | positive | 0.5 |  |
| 6. | not\_raise | positive | not\_grow | positive | 0.5 |  |
| 7. | raise | positive | not\_grow | negative | 0.2 |  |
| 8. | not\_raise | positive | not\_grow | negative | 0.8 |  |
| 9. | raise | negative | grow | positive | 0.7 |  |
| 10. | not\_raise | negative | grow | positive | 0.3 |  |
| 11. | raise | negative | grow | negative | 0.4 |  |
| 12. | not\_raise | negative | grow | negative | 0.6 |  |
| 13. | raise | negative | not\_grow | positive | 0.3 |  |
| 14. | not\_raise | negative | not\_grow | positive | 0.7 |  |
| 15. | raise | negative | not\_grow | negative | 0 |  |
| 16. | not\_raise | negative | not\_grow | negative | 1 |  |

## Investment Decision - D

### Opis

Prawdopodobieństwo, żeby podjąć decyzję o inwestycji.

### Prawdopodobieństwa

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| L.p. | Rodzaj | C | Wartość | Opis |
| 1. | yes | raise | 0.6 | Wzost cen akcji motywuje do stabilizacji |
| 2. | no | raise | 0.4 |  |
| 3. | yes | not\_raise | 0.7 | Spacek cen kacji motywuje do ryzyka |
| 4. | no | not\_raise | 0.3 |  |

# Źródła

<https://www.wiadomoscihandlowe.pl/konsument-i-trendy-zakupowe/reakcja-lancuchowa-jak-globalne-zmiany-wplywaja-na-dzialania-konsumentow-w-branzy-fmcg-2520728>

<https://pap-mediaroom.pl/biznes-i-finanse/ekspert-procesy-globalizacji-maja-zarowno-pozytywne-jak-i-negatywne-konotacje>

<https://www.paih.gov.pl/eksport/abc_eksportu/strategie_wejscia_na_rynki_zagraniczne/>

<https://min-pan.krakow.pl/wydawnictwo/wp-content/uploads/sites/4/2021/12/IWASZCZUK-Ryzyko.pdf>

„Według reguły Pareto, w każdej organizacji około 20% kluczowych czynników ryzyka odpowiada za 80% potencjalnych strat lub niewykorzystanie 80% możliwych szans.”

<https://www.tiger.edu.pl/ksiazki/PolitykaFinansowaTransformacjaWzrost.pdf>

“W związku ze wzrostem wydajności i poprawą zarządzania w sektorze państwowym można będzie dopuścić do wzrostu płac realnych przez znaczącą liberalizację "popiwku"(po wprowadzeniu planowanych w Ministerstwie Finansów zmian dotyczyć on będzie już tylko ok. 25% zakładów) i jego zniesienie z końcem roku”

<https://sentione.com/pl/blog/7-najlepszych-narzedzi-do-zarzadzania-reputacja-w-sieci>

<https://zaufane.pl/analiza-badania-wplyw-opinii-na-lokalne-firmy>

„60% napisało recenzje pozytywnych doświadczeń”

<https://blogi.bossa.pl/2024/07/05/ceny-akcji-a-dane-makro/>

A screenshot of a computer

Description automatically generated

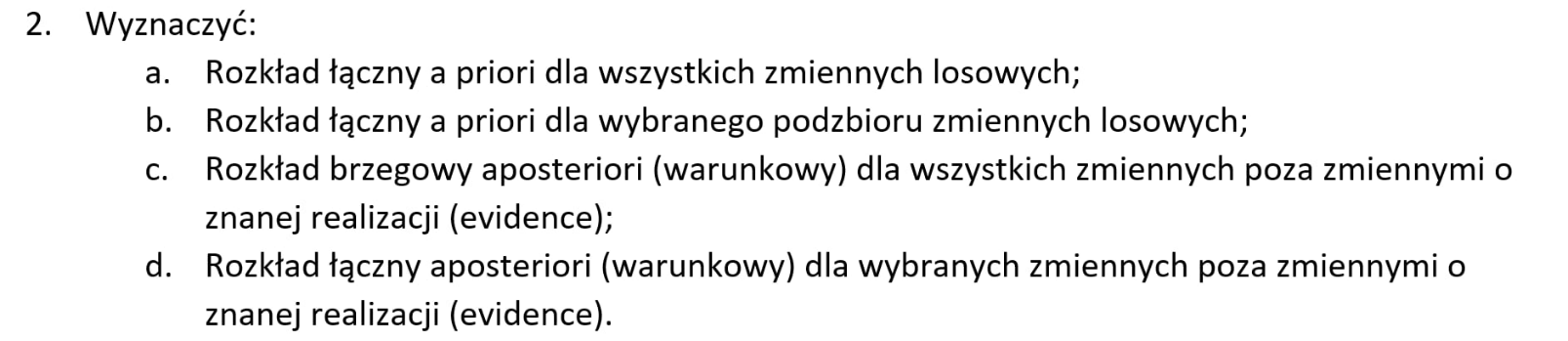
<https://brand24.pl/blog/jak-zarzadzac-reputacja-marki-w-internecie-garsc-praktycznych-wskazowek/>

„Era internetu spowodowała, że zachowanie dobrego imienia firmy stanowi dziś trudne wyzwanie. Komunikacja na linii marka-klient przeniosła się do innego wymiaru, gdzie występuje niemal 24/7, a informacje wśród internautów rozprzestrzeniają się z prędkością światła. To z kolei może być dużym problemem w przypadku negatywnych treści, rodząc w ten sposób kryzysy wizerunkowe.”

<https://www.bankier.pl/wiadomosc/Strategie-inwestora-gieldowego-1710352.html>

<https://www.parkiet.com/inwestycje/art25356801-metody-podejmowania-decyzji>

# Rozwiązanie zadania 2



#### Podpunkt A

Dodanie elementu do sprawdzenia rozkładu:

A diagram of a company

Description automatically generated

Ustawienie wartości 1 0 dla każdej wartości:

A screenshot of a computer

Description automatically generated

Włączenie opcji w Genie:

A screenshot of a computer

Description automatically generated

Zaznaczenie JPD Nodes:

A screenshot of a computer

Description automatically generated

Rozkład apriori:

A screenshot of a computer

Description automatically generated

A screenshot of a video game

Description automatically generated

#### Podpunkt B

Dla podzdioru wykluczając Global Events oraz Market Condition

A screenshot of a computer

Description automatically generated

Wynik:

A screenshot of a computer

Description automatically generated

#### Podpunkt C

Dla Global Events = „good” oraz Company Reputation = “positive”

A screenshot of a computer

Description automatically generatedA screenshot of a graph

Description automatically generatedA screenshot of a graph

Description automatically generatedA screenshot of a graph

Description automatically generated

#### Podpunkt D

Podzbiór z wykluczeniem Company Income

A screenshot of a computer

Description automatically generated

Wynik:

A screenshot of a computer

Description automatically generated

# Rozwiązanie zadania 3

A screenshot of a computer

Description automatically generated

Tworzenie pliku:

A screenshot of a computer

Description automatically generated

Utworzony plik:

A screenshot of a computer

Description automatically generated

# Rozwiązanie zadania 4

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Sieć po realizacji zadań

A diagram of a company

Description automatically generated

# Rozwiązania w języku Python

## Tworzenie sieci

### Kod

from pgmpy.factors.discrete import TabularCPD  
from pgmpy.models import BayesianNetwork  
  
  
def create\_bayesian\_network():  
 # Definicja modelu  
 model = BayesianNetwork([  
 ('G', 'M'),  
 ('M', 'C'),  
 ('I', 'C'),  
 ('R', 'C'),  
 ('C', 'D')  
 ])  
  
 # Tworzenie CPDs  
  
 # Global Events (G)  
 cpd\_G = TabularCPD(  
 variable='G',  
 variable\_card=3,  
 values=[[0.1], [0.6], [0.3]],  
 state\_names={'G': ['good', 'neutral', 'bad']}  
 )  
  
 # Market Condition (M) zależne od G  
 cpd\_M\_given\_G = TabularCPD(  
 variable='M',  
 variable\_card=2,  
 values=[  
 [0.8, 0.5, 0.8], # P(M=positive|G)  
 [0.2, 0.5, 0.2] # P(M=negative|G)  
 ],  
 evidence=['G'],  
 evidence\_card=[3],  
 state\_names={  
 'M': ['positive', 'negative'],  
 'G': ['good', 'neutral', 'bad']  
 }  
 )  
  
 # Company Income (I)  
 cpd\_I = TabularCPD(  
 variable='I',  
 variable\_card=2,  
 values=[[0.25], [0.75]],  
 state\_names={'I': ['grow', 'not\_grow']}  
 )  
  
 # Company Reputation (R)  
 cpd\_R = TabularCPD(  
 variable='R',  
 variable\_card=2,  
 values=[[0.6], [0.4]],  
 state\_names={'R': ['positive', 'negative']}  
 )  
  
 # Action Cost Change (C) zależne od M, I i R  
 cpd\_C = TabularCPD(  
 variable='C',  
 variable\_card=2,  
 values=[  
 [0.9, 0.6, 0.5, 0.2, 0.7, 0.4, 0.3, 0.0], # P(C=raise|...)  
 [0.1, 0.4, 0.5, 0.8, 0.3, 0.6, 0.7, 1.0] # P(C=not\_raise|...)  
 ],  
 evidence=['M', 'I', 'R'],  
 evidence\_card=[2, 2, 2],  
 state\_names={  
 'C': ['raise', 'not\_raise'],  
 'M': ['positive', 'negative'],  
 'I': ['grow', 'not\_grow'],  
 'R': ['positive', 'negative']  
 }  
 )  
  
 # Investment Decision (D) zależne od C  
 cpd\_D = TabularCPD(  
 variable='D',  
 variable\_card=2,  
 values=[  
 [0.6, 0.7], # P(D=yes|C)  
 [0.4, 0.3] # P(D=no|C)  
 ],  
 evidence=['C'],  
 evidence\_card=[2],  
 state\_names={  
 'D': ['yes', 'no'],  
 'C': ['raise', 'not\_raise']  
 }  
 )  
  
 # Dodanie CPDs do modelu  
 model.add\_cpds(cpd\_G, cpd\_M\_given\_G, cpd\_I, cpd\_R, cpd\_C, cpd\_D)  
  
 # Sprawdzenie poprawności modelu  
 if model.check\_model():  
 print("The Bayesian Network is correctly defined.")  
 else:  
 print("There is an error in the Bayesian Network definition.")  
  
 return model  
  
  
def display\_model(model):  
 print("\nCPDs in the model:")  
 for cpd in model.cpds:  
 print(cpd)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 bayesian\_network = create\_bayesian\_network()  
 display\_model(bayesian\_network)

### Wykonanie

The Bayesian Network is correctly defined.

CPDs in the model:

+------------+-----+

| G(good) | 0.1 |

+------------+-----+

| G(neutral) | 0.6 |

+------------+-----+

| G(bad) | 0.3 |

+------------+-----+

+-------------+---------+------------+--------+

| G | G(good) | G(neutral) | G(bad) |

+-------------+---------+------------+--------+

| M(positive) | 0.8 | 0.5 | 0.8 |

+-------------+---------+------------+--------+

| M(negative) | 0.2 | 0.5 | 0.2 |

+-------------+---------+------------+--------+

+-------------+------+

| I(grow) | 0.25 |

+-------------+------+

| I(not\_grow) | 0.75 |

+-------------+------+

+-------------+-----+

| R(positive) | 0.6 |

+-------------+-----+

| R(negative) | 0.4 |

+-------------+-----+

+--------------+-------------+-----+-------------+-------------+

| M | M(positive) | ... | M(negative) | M(negative) |

+--------------+-------------+-----+-------------+-------------+

| I | I(grow) | ... | I(not\_grow) | I(not\_grow) |

+--------------+-------------+-----+-------------+-------------+

| R | R(positive) | ... | R(positive) | R(negative) |

+--------------+-------------+-----+-------------+-------------+

| C(raise) | 0.9 | ... | 0.3 | 0.0 |

+--------------+-------------+-----+-------------+-------------+

| C(not\_raise) | 0.1 | ... | 0.7 | 1.0 |

+--------------+-------------+-----+-------------+-------------+

+--------+----------+--------------+

| C | C(raise) | C(not\_raise) |

+--------+----------+--------------+

| D(yes) | 0.6 | 0.7 |

+--------+----------+--------------+

| D(no) | 0.4 | 0.3 |

+--------+----------+--------------+

Process finished with exit code 0

## Zadanie 2A

### Kod

from itertools import product  
  
from investment\_decision import create\_bayesian\_network  
  
  
def compute\_joint\_distribution(model):  
 *""" Computes the joint probability distribution for the Bayesian Network. """* cpds = {cpd.variable: cpd for cpd in model.cpds}  
  
 variables = [cpd.variable for cpd in model.cpds]  
 state\_cardinalities = [cpd.variable\_card for cpd in model.cpds]  
 all\_states = list(product(\*[range(card) for card in state\_cardinalities]))  
  
 joint\_distribution = []  
 for state in all\_states:  
 prob = 1.0  
 state\_dict = dict(zip(variables, state))  
  
 for var in variables:  
 cpd = cpds[var]  
 evidence\_vars = cpd.get\_evidence()  
  
 if evidence\_vars:  
 evidence\_values = [state\_dict[e] for e in evidence\_vars]  
 prob \*= cpd.values[tuple([state\_dict[var]] + evidence\_values)]  
 else:  
 prob \*= cpd.values[state\_dict[var]]  
  
 joint\_distribution.append((state, prob))  
  
 return joint\_distribution  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 bayesian\_network = create\_bayesian\_network()  
  
 joint\_dist = compute\_joint\_distribution(bayesian\_network)  
  
 print("Joint Probability Distribution:")  
 for state, prob in joint\_dist:  
 print(f"State {state}: P = {prob:.6f}")

### Wykonanie

The Bayesian Network is correctly defined.

Joint Probability Distribution:

State (0, 0, 0, 0, 0, 0): P = 0.006480

State (0, 0, 0, 0, 0, 1): P = 0.004320

State (0, 0, 0, 0, 1, 0): P = 0.000840

State (0, 0, 0, 0, 1, 1): P = 0.000360

State (0, 0, 0, 1, 0, 0): P = 0.003360

State (0, 0, 0, 1, 0, 1): P = 0.002240

State (0, 0, 0, 1, 1, 0): P = 0.001680

State (0, 0, 0, 1, 1, 1): P = 0.000720

State (0, 0, 1, 0, 0, 0): P = 0.010800

State (0, 0, 1, 0, 0, 1): P = 0.007200

State (0, 0, 1, 0, 1, 0): P = 0.012600

State (0, 0, 1, 0, 1, 1): P = 0.005400

State (0, 0, 1, 1, 0, 0): P = 0.004320

State (0, 0, 1, 1, 0, 1): P = 0.002880

State (0, 0, 1, 1, 1, 0): P = 0.011760

State (0, 0, 1, 1, 1, 1): P = 0.005040

State (0, 1, 0, 0, 0, 0): P = 0.001080

State (0, 1, 0, 0, 0, 1): P = 0.000720

State (0, 1, 0, 0, 1, 0): P = 0.000840

State (0, 1, 0, 0, 1, 1): P = 0.000360

State (0, 1, 0, 1, 0, 0): P = 0.000480

State (0, 1, 0, 1, 0, 1): P = 0.000320

State (0, 1, 0, 1, 1, 0): P = 0.000840

State (0, 1, 0, 1, 1, 1): P = 0.000360

State (0, 1, 1, 0, 0, 0): P = 0.001080

State (0, 1, 1, 0, 0, 1): P = 0.000720

State (0, 1, 1, 0, 1, 0): P = 0.005040

State (0, 1, 1, 0, 1, 1): P = 0.002160

State (0, 1, 1, 1, 0, 0): P = 0.000000

State (0, 1, 1, 1, 0, 1): P = 0.000000

State (0, 1, 1, 1, 1, 0): P = 0.004200

State (0, 1, 1, 1, 1, 1): P = 0.001800

State (1, 0, 0, 0, 0, 0): P = 0.024300

State (1, 0, 0, 0, 0, 1): P = 0.016200

State (1, 0, 0, 0, 1, 0): P = 0.003150

State (1, 0, 0, 0, 1, 1): P = 0.001350

State (1, 0, 0, 1, 0, 0): P = 0.012600

State (1, 0, 0, 1, 0, 1): P = 0.008400

State (1, 0, 0, 1, 1, 0): P = 0.006300

State (1, 0, 0, 1, 1, 1): P = 0.002700

State (1, 0, 1, 0, 0, 0): P = 0.040500

State (1, 0, 1, 0, 0, 1): P = 0.027000

State (1, 0, 1, 0, 1, 0): P = 0.047250

State (1, 0, 1, 0, 1, 1): P = 0.020250

State (1, 0, 1, 1, 0, 0): P = 0.016200

State (1, 0, 1, 1, 0, 1): P = 0.010800

State (1, 0, 1, 1, 1, 0): P = 0.044100

State (1, 0, 1, 1, 1, 1): P = 0.018900

State (1, 1, 0, 0, 0, 0): P = 0.016200

State (1, 1, 0, 0, 0, 1): P = 0.010800

State (1, 1, 0, 0, 1, 0): P = 0.012600

State (1, 1, 0, 0, 1, 1): P = 0.005400

State (1, 1, 0, 1, 0, 0): P = 0.007200

State (1, 1, 0, 1, 0, 1): P = 0.004800

State (1, 1, 0, 1, 1, 0): P = 0.012600

State (1, 1, 0, 1, 1, 1): P = 0.005400

State (1, 1, 1, 0, 0, 0): P = 0.016200

State (1, 1, 1, 0, 0, 1): P = 0.010800

State (1, 1, 1, 0, 1, 0): P = 0.075600

State (1, 1, 1, 0, 1, 1): P = 0.032400

State (1, 1, 1, 1, 0, 0): P = 0.000000

State (1, 1, 1, 1, 0, 1): P = 0.000000

State (1, 1, 1, 1, 1, 0): P = 0.063000

State (1, 1, 1, 1, 1, 1): P = 0.027000

State (2, 0, 0, 0, 0, 0): P = 0.019440

State (2, 0, 0, 0, 0, 1): P = 0.012960

State (2, 0, 0, 0, 1, 0): P = 0.002520

State (2, 0, 0, 0, 1, 1): P = 0.001080

State (2, 0, 0, 1, 0, 0): P = 0.010080

State (2, 0, 0, 1, 0, 1): P = 0.006720

State (2, 0, 0, 1, 1, 0): P = 0.005040

State (2, 0, 0, 1, 1, 1): P = 0.002160

State (2, 0, 1, 0, 0, 0): P = 0.032400

State (2, 0, 1, 0, 0, 1): P = 0.021600

State (2, 0, 1, 0, 1, 0): P = 0.037800

State (2, 0, 1, 0, 1, 1): P = 0.016200

State (2, 0, 1, 1, 0, 0): P = 0.012960

State (2, 0, 1, 1, 0, 1): P = 0.008640

State (2, 0, 1, 1, 1, 0): P = 0.035280

State (2, 0, 1, 1, 1, 1): P = 0.015120

State (2, 1, 0, 0, 0, 0): P = 0.003240

State (2, 1, 0, 0, 0, 1): P = 0.002160

State (2, 1, 0, 0, 1, 0): P = 0.002520

State (2, 1, 0, 0, 1, 1): P = 0.001080

State (2, 1, 0, 1, 0, 0): P = 0.001440

State (2, 1, 0, 1, 0, 1): P = 0.000960

State (2, 1, 0, 1, 1, 0): P = 0.002520

State (2, 1, 0, 1, 1, 1): P = 0.001080

State (2, 1, 1, 0, 0, 0): P = 0.003240

State (2, 1, 1, 0, 0, 1): P = 0.002160

State (2, 1, 1, 0, 1, 0): P = 0.015120

State (2, 1, 1, 0, 1, 1): P = 0.006480

State (2, 1, 1, 1, 0, 0): P = 0.000000

State (2, 1, 1, 1, 0, 1): P = 0.000000

State (2, 1, 1, 1, 1, 0): P = 0.012600

State (2, 1, 1, 1, 1, 1): P = 0.005400

Process finished with exit code 0

## Zadanie 2B

### Kod

from itertools import product  
from collections import defaultdict  
  
from investment\_decision import create\_bayesian\_network  
  
  
def compute\_marginalized\_distribution(model, excluded\_vars):  
 *""" Computes the joint probability distribution excluding specified variables. """* cpds = {cpd.variable: cpd for cpd in model.cpds}  
  
 variables = [cpd.variable for cpd in model.cpds if cpd.variable not in excluded\_vars]  
 state\_cardinalities = [cpd.variable\_card for cpd in model.cpds if cpd.variable not in excluded\_vars]  
  
 marginalized\_distribution = defaultdict(float)  
  
 for full\_state in product(\*[range(cpd.variable\_card) for cpd in model.cpds]):  
 full\_state\_dict = dict(zip([cpd.variable for cpd in model.cpds], full\_state))  
 prob = 1.0  
  
 for var in cpds:  
 cpd = cpds[var]  
 evidence\_vars = cpd.get\_evidence()  
  
 if evidence\_vars:  
 evidence\_values = [full\_state\_dict[e] for e in evidence\_vars]  
 prob \*= cpd.values[tuple([full\_state\_dict[var]] + evidence\_values)]  
 else:  
 prob \*= cpd.values[full\_state\_dict[var]]  
  
 marginalized\_state = tuple(full\_state\_dict[v] for v in variables)  
 marginalized\_distribution[marginalized\_state] += prob  
  
 return marginalized\_distribution  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 bayesian\_network = create\_bayesian\_network()  
  
 excluded\_vars = {"G": "Global Events", "M": "Market Condition"}.keys()  
 marginalized\_dist = compute\_marginalized\_distribution(bayesian\_network, excluded\_vars)  
  
 print("Marginalized Joint Probability Distribution (excluding Global Events and Market Condition):")  
 for state, prob in marginalized\_dist.items():  
 print(f"State {state}: P = {prob:.6f}")

### Wykonanie

The Bayesian Network is correctly defined.

Marginalized Joint Probability Distribution (excluding Global Events and Market Condition):

State (0, 0, 0, 0): P = 0.070740

State (0, 0, 0, 1): P = 0.047160

State (0, 0, 1, 0): P = 0.022470

State (0, 0, 1, 1): P = 0.009630

State (0, 1, 0, 0): P = 0.035160

State (0, 1, 0, 1): P = 0.023440

State (0, 1, 1, 0): P = 0.028980

State (0, 1, 1, 1): P = 0.012420

State (1, 0, 0, 0): P = 0.104220

State (1, 0, 0, 1): P = 0.069480

State (1, 0, 1, 0): P = 0.193410

State (1, 0, 1, 1): P = 0.082890

State (1, 1, 0, 0): P = 0.033480

State (1, 1, 0, 1): P = 0.022320

State (1, 1, 1, 0): P = 0.170940

State (1, 1, 1, 1): P = 0.073260

Process finished with exit code 0

## Zadanie 2C

### Kod

from pgmpy.inference import VariableElimination  
  
from investment\_decision import create\_bayesian\_network  
  
  
def calculate\_marginal\_posterior(model, evidence):  
  
 inference = VariableElimination(model)  
  
 all\_variables = set(model.nodes())  
  
 evidence\_variables = set(evidence.keys())  
  
 query\_variables = all\_variables - evidence\_variables  
  
 marginal\_posteriors = {}  
  
 for variable in query\_variables:  
 marginal\_posteriors[variable] = inference.query(  
 variables=[variable], evidence=evidence, show\_progress=False  
 )  
  
 return marginal\_posteriors  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 bayesian\_network = create\_bayesian\_network()  
  
 evidence = {'G': 'good', 'R': 'positive'}  
  
 marginal\_posteriors = calculate\_marginal\_posterior(bayesian\_network, evidence)  
  
 print("\nMarginal posterior distributions:")  
 for variable, distribution in marginal\_posteriors.items():  
 print(f"{variable}:\n{distribution}")

### Wykonanie

The Bayesian Network is correctly defined.

Marginal posterior distributions:

D:

+--------+----------+

| D | phi(D) |

+========+==========+

| D(yes) | 0.6440 |

+--------+----------+

| D(no) | 0.3560 |

+--------+----------+

I:

+-------------+----------+

| I | phi(I) |

+=============+==========+

| I(grow) | 0.2500 |

+-------------+----------+

| I(not\_grow) | 0.7500 |

+-------------+----------+

C:

+--------------+----------+

| C | phi(C) |

+==============+==========+

| C(raise) | 0.5600 |

+--------------+----------+

| C(not\_raise) | 0.4400 |

+--------------+----------+

M:

+-------------+----------+

| M | phi(M) |

+=============+==========+

| M(positive) | 0.8000 |

+-------------+----------+

| M(negative) | 0.2000 |

+-------------+----------+

## Zadanie 2D

### Kod

from pgmpy.inference import VariableElimination  
from investment\_decision import create\_bayesian\_network  
  
  
def get\_joint\_posterior\_with\_target\_vars(bayesian\_network, observed\_evidence, target\_variables=None):  
  
 inference = VariableElimination(bayesian\_network)  
  
 all\_vars = set(bayesian\_network.nodes())  
 observed\_vars = set(observed\_evidence.keys())  
  
 target\_vars = target\_variables if target\_variables else (all\_vars - observed\_vars)  
  
 joint\_distribution = inference.query(variables=list(target\_vars), evidence=observed\_evidence, show\_progress=False)  
  
 return joint\_distribution  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 bayesian\_network = create\_bayesian\_network()  
  
 evidence = {'G': 'good', 'R': 'positive'}  
  
 target\_variables = ['M', 'C']  
  
 marginal\_posteriors = get\_joint\_posterior\_with\_target\_vars(bayesian\_network, evidence, target\_variables)  
  
 print("\nJoint posterior distributions for selected variables:")  
 print(f"{marginal\_posteriors}")

### Wykonanie

The Bayesian Network is correctly defined.

Joint posterior distributions for selected variables:

+-------------+--------------+------------+

| M | C | phi(M,C) |

+=============+==============+============+

| M(positive) | C(raise) | 0.4800 |

+-------------+--------------+------------+

| M(positive) | C(not\_raise) | 0.3200 |

+-------------+--------------+------------+

| M(negative) | C(raise) | 0.0800 |

+-------------+--------------+------------+

| M(negative) | C(not\_raise) | 0.1200 |

+-------------+--------------+------------+

Process finished with exit code 0

### Rozwiązanie zadania 3 w Pythonie

#### Kod

import numpy as np  
from pgmpy.sampling import BayesianModelSampling  
  
from investment\_decision import create\_bayesian\_network  
  
  
def generate\_synthetic\_data(model, num\_samples, missing\_rate, random\_seed):  
 # Ustawienie ziarna losowości  
 np.random.seed(random\_seed)  
  
 # Generowanie próbek  
 sampler = BayesianModelSampling(model)  
 data = sampler.forward\_sample(size=num\_samples, seed=random\_seed)  
  
 # Wprowadzanie brakujących wartości  
 data = data.applymap(lambda x: x.decode('utf-8') if isinstance(x, bytes) else x) # Konwersja bajtów do stringów  
 for col in data.columns:  
 # Losujemy, które wartości w tej kolumnie mają być brakujące  
 missing\_indices = data.sample(frac=missing\_rate).index  
 data.loc[missing\_indices, col] = np.nan  
  
 return data  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # Tworzenie sieci Bayesowskiej  
 bayesian\_network = create\_bayesian\_network()  
  
 # Generowanie danych syntetycznych  
 num\_samples = 1000  
 missing\_rate = 0.02 # 2%  
 random\_seed = 1234  
  
 data = generate\_synthetic\_data(bayesian\_network, num\_samples, missing\_rate, random\_seed)  
  
 # Zapisywanie danych do pliku CSV  
 data.to\_csv('synthetic\_data.csv', index=False)  
 print("Synthetic data generated and saved to 'synthetic\_data.csv'.")

#### Wykonanie

The Bayesian Network is correctly defined.

Generating for node: D: 100%|██████████| 6/6 [00:00<00:00, 1327.66it/s]

Synthetic data generated and saved to 'synthetic\_data.csv'.

A screenshot of a computer

Description automatically generated

### Rozwiązanie zadania 4 w Pythonie

#### Kod

import pandas as pd  
from pgmpy.estimators import HillClimbSearch, BicScore, MaximumLikelihoodEstimator  
from pgmpy.models import BayesianNetwork  
from pgmpy.utils import get\_example\_model  
  
  
def learn\_bayesian\_network\_from\_data(file\_path):  
 # Wczytanie danych z pliku CSV  
 data = pd.read\_csv(file\_path)  
  
 # Wyświetlenie podstawowych informacji o danych  
 print("Sample of the data:")  
 print(data.head())  
 print("\nSummary of missing values:")  
 print(data.isnull().sum())  
  
 # Usuwanie rekordów z brakującymi wartościami (opcjonalne)  
 data = data.dropna()  
 print(f"\nData after handling missing values (remaining rows: {len(data)}):")  
 print(data.head())  
  
 # Uczenie struktury sieci  
 hc = HillClimbSearch(data)  
 best\_model = hc.estimate(scoring\_method=BicScore(data))  
 print("\nLearned structure:")  
 print(best\_model.edges())  
  
 # Uczenie parametrów sieci  
 model = BayesianNetwork(best\_model.edges())  
 model.fit(data, estimator=MaximumLikelihoodEstimator)  
  
 print("\nLearned CPDs:")  
 for cpd in model.cpds:  
 print(cpd)  
  
 return model  
  
  
def main():  
 # Ścieżka do pliku z danymi syntetycznymi  
 file\_path = 'synthetic\_data.csv'  
  
 # Uczenie modelu  
 learned\_model = learn\_bayesian\_network\_from\_data(file\_path)  
  
 # Zapisywanie wyników do pliku  
 print("\nSaving learned model structure as 'learned\_model.txt'")  
 with open('learned\_model.txt', 'w') as file:  
 for edge in learned\_model.edges():  
 file.write(f"{edge[0]} -> {edge[1]}\n")  
  
 print("\nModel structure saved!")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

#### Wykonanie

Sample of the data:

G M C I R D

0 neutral negative not\_raise not\_grow negative no

1 neutral positive raise not\_grow negative no

2 neutral positive raise not\_grow negative yes

3 bad positive raise not\_grow positive NaN

4 bad negative not\_raise not\_grow NaN yes

Summary of missing values:

G 20

M 20

C 20

I 20

R 20

D 20

dtype: int64

Data after handling missing values (remaining rows: 886):

G M C I R D

0 neutral negative not\_raise not\_grow negative no

1 neutral positive raise not\_grow negative no

2 neutral positive raise not\_grow negative yes

5 neutral positive not\_raise not\_grow negative yes

6 neutral positive not\_raise not\_grow negative yes

0%| | 6/1000000 [00:00<3:47:12, 73.36it/s]

Learned structure:

[('M', 'G'), ('M', 'C'), ('I', 'C'), ('R', 'C')]

Learned CPDs:

+-------------+----------+

| M(negative) | 0.382619 |

+-------------+----------+

| M(positive) | 0.617381 |

+-------------+----------+

+------------+--------------------+---------------------+

| M | M(negative) | M(positive) |

+------------+--------------------+---------------------+

| G(bad) | 0.1504424778761062 | 0.41681901279707495 |

+------------+--------------------+---------------------+

| G(good) | 0.0471976401179941 | 0.14625228519195613 |

+------------+--------------------+---------------------+

| G(neutral) | 0.8023598820058997 | 0.4369287020109689 |

+------------+--------------------+---------------------+

+--------------+--------------------+-----+---------------------+

| I | I(grow) | ... | I(not\_grow) |

+--------------+--------------------+-----+---------------------+

| M | M(negative) | ... | M(positive) |

+--------------+--------------------+-----+---------------------+

| R | R(negative) | ... | R(positive) |

+--------------+--------------------+-----+---------------------+

| C(not\_raise) | 0.4594594594594595 | ... | 0.5155555555555555 |

+--------------+--------------------+-----+---------------------+

| C(raise) | 0.5405405405405406 | ... | 0.48444444444444446 |

+--------------+--------------------+-----+---------------------+

+-------------+----------+

| I(grow) | 0.286682 |

+-------------+----------+

| I(not\_grow) | 0.713318 |

+-------------+----------+

+-------------+----------+

| R(negative) | 0.422122 |

+-------------+----------+

| R(positive) | 0.577878 |

+-------------+----------+

Saving learned model structure as 'learned\_model.txt'

Model structure saved!

Process finished with exit code 0