

# Inferența prin Enumerare în Rețele Bayesiene

Studenti:  
Vartolomei Mihai-Sebastian  
Frunza Claudia-Elena

Grupa:  
1405B

## 1. Descrierea Problemei

Rețelele bayesiene sunt structuri grafice utilizate pentru reprezentarea relațiilor probabilistice dintre variabile. Acestea sunt deosebit de utile în modelarea sistemelor complexe, unde relațiile cauzale pot fi exprimate matematic și utilizate pentru a realiza predicții sau inferențe. Proiectul propune implementarea unui algoritm exact de inferență – inferența prin enumerare – care permite calcularea probabilităților marginale pentru un set de variabile interogate, dată fiind o anumită evidență.

Obiectivul este de a dezvolta un program capabil să citească structura și parametrii rețelei dintr-un fișier, să accepte interogări din partea utilizatorului și să ofere probabilități condiționate pentru variabilele dorite.

## 2. Aspecte Teoretice

### 2.1 Rețele Bayesiene

O rețea bayesiană este un graf orientat aciclic (DAG), unde fiecare nod reprezintă o variabilă aleatoare, iar arcele indică dependențe condiționate. Fiecare nod este asociat cu un tabel de probabilități condiționate (CPT).

$$P(A | B) = \frac{P(B|A) \cdot P(A)}{P(B)} \quad (1)$$

(1) Aceasta exprimă relația dintre probabilitatea condiționată  $P(A | B)$ , probabilitatea marginală  $P(A)$ , probabilitatea condiționată inversă  $P(B | A)$ , și probabilitatea marginală  $P(B)$ .

Teorema lui Bayes constituie baza matematică a inferenței.

## 2.2 Inferența prin Enumerare

Aceasta constă în calcularea distribuției probabilistice pentru o variabilă interogată, dată fiind evidența, prin următorii pași:

1. Se sortează nodurile topologic astfel încât părinții să fie prelucrați înaintea fiilor.
2. Se aplică regula de înmulțire a probabilităților pentru variabilele observate.
3. Se sumează probabilitățile pentru variabilele neobservate.
4. Se normalizează rezultatul astfel încât suma distribuției probabilistice să fie 1.

## 3. Modalitatea de Rezolvare

Programul este implementat în Python și este structurat în trei module principale:

### 3.1 `bayesian_network.py`

Acest modul definește clasa pentru reprezentarea rețelei bayesiene:

- **Încărcarea Structurii:** Rețeaua este construită dintr-un fișier JSON care specifică nodurile, părinții acestora și probabilitățile condiționate asociate fiecărui nod.
- **Metode Cheie:**
  - `get_parents(node)` – returnează lista de părinți ai unui nod.
  - `get_probability(node, value, evidence)` – calculează probabilitatea condiționată pentru un nod dat o evidență.

### 3.2 `inference_engine.py`

Motorul de inferență implementează algoritmul de enumerare:

- **Metoda `enumerate_all(variables, evidence)`:** Calculează produsul și suma probabilităților pentru toate variabilele din rețea, incluzând cele observate și neobservate.
- **Metoda `query(query_var, evidence)`:** Returnează distribuția de probabilitate pentru o variabilă interogată, dată fiind evidența specificată.

### 3.3 `main.py`

Asigură interfața cu utilizatorul:

- **Încărcarea Rețelei:** Programul citește structura și parametrii dintr-un fișier JSON specificat de utilizator.
- **Interogări:** Permite utilizatorului să seteze evidența și să solicite probabilități pentru variabilele dorite.

## 4. Codul sursă (Fragmente Relevante)

### 4.1 Rețea Bayesiană

```
3 class BayesianNetwork: 2 usages
4     def __init__(self, structure_file):
5         """Initialize the Bayesian Network from a file."""
6         self.nodes = {}
7         self.probabilities = {}
8         self.load_structure(structure_file)
9
10    def load_structure(self, structure_file): 1 usage
11        """Load network structure and probabilities from a JSON file."""
12        with open(structure_file, 'r') as f:
13            data = json.load(f)
14
15        for node, details in data.items():
16            self.nodes[node] = details['parents']
17            self.probabilities[node] = details['probabilities']
```

- Constructorul clasei (`__init__`) inițializează un dicționar pentru a stoca nodurile și tabelele lor de probabilități condiționate. Apoi, funcția `load_structure` încarcă structura rețelei dintr-un fișier JSON.
- Fiecare nod are o listă de părinți (variabile care îl influențează direct) și un tabel de probabilități condiționate asociate fiecărui nod, pe baza valorilor părinților săi.

### 4.2 Motorul de Inferență

```
5 def enumerate_all(self, variables, evidence): 3 usages
6     """Enumerate all variables recursively for inference."""
7     if not variables:
8         return 1.0
9
10    first, rest = variables[0], variables[1:]
11
12    if first in evidence:
13        prob = self.network.get_probability(first, evidence[first], evidence)
14        return prob * self.enumerate_all(rest, evidence)
15    else:
16        total = 0
17        for value in [True, False]:
18            extended_evidence = evidence.copy()
19            extended_evidence[first] = value
20            prob = self.network.get_probability(first, value, extended_evidence)
21            total += prob * self.enumerate_all(rest, extended_evidence)
22        return total
23
```

- `enumerate_all` calculează probabilitatea combinată pentru toate variabilele rețelei. Dacă variabila curentă este prezentă în evidență, funcția folosește această valoare și continuă recursiv. Pentru variabilele neobservate, se calculează o sumă ponderată pe baza valorilor posibile.

```

24     def query(self, query_var, evidence):
25         """Calculate the probability distribution of the query variable."""
26         variables = list(self.network.nodes.keys())
27         probabilities = {}
28
29         for value in [True, False]:
30             extended_evidence = evidence.copy()
31             extended_evidence[query_var] = value
32             probabilities[value] = self.enumerate_all(variables, extended_evidence)
33
34         # Normalize probabilities
35         total = sum(probabilities.values())
36         for value in probabilities:
37             probabilities[value] /= total
38
39         return probabilities
40

```

- `query` este funcția principală pentru a calcula distribuția probabilistică a unei variabile interogate. Ea extinde evidența cu fiecare valoare posibilă a variabilei interogate și folosește `enumerate_all` pentru a calcula probabilitățile brute, care apoi sunt normalizate pentru a obține o distribuție validă.

### 4.3 JSON

```

1
2  "Gripa": {
3      "parents": [],
4      "probabilities": {
5          "True": 0.1,
6          "False": 0.9
7      }
8  },
9  "Abces": {
10     "parents": [],
11     "probabilities": {
12         "True": 0.05,
13         "False": 0.95
14     }
15 },

```

Fișierul JSON este utilizat pentru a defini structura rețelei bayesiene, incluzând:

- Nodurile rețelei și părinții lor.
- Tabelele de probabilități condiționate asociate fiecărui nod.

#### 4.4 Interfața cu Utilizatorul

```
14     while True:
15         print("\nOptions:")
16         print("1. Query a variable")
17         print("2. Exit")
18         choice = input("Choose an option: ")
19
20         if choice == "1":
21             query_var = input("Enter the query variable: ")
22             evidence = {}
23             print("Enter evidence variables (leave empty to finish):")
24
25             while True:
26                 evidence_var = input("Evidence variable: ")
27                 if not evidence_var:
28                     break
29                 value = input(f"Value for {evidence_var} (True/False): ").lower() == "true"
30                 evidence[evidence_var] = value
31
32             probabilities = engine.query(query_var, evidence)
33             print(f"\nProbability distribution for {query_var}:")
34             for value, prob in probabilities.items():
35                 print(f"    {value}: {prob:.4f}")
36
37         elif choice == "2":
38             print("Goodbye!")
39             break
40         else:
41             print("Invalid option. Please try again.")
42
```

- Interfața permite utilizatorului să efectueze interogări asupra rețelei bayesiene prin selectarea unei variabile și furnizarea de evidențe relevante.
- Utilizatorul introduce o variabilă de interogare (`query_var`) și apoi poate adăuga mai multe variabile de evidență, cu valorile lor (True sau False).
- După ce sunt furnizate toate evidențele, funcția `query` a motorului de inferență este apelată pentru a calcula distribuția probabilistică a variabilei interogate.
- Rezultatele sunt afișate sub forma probabilităților normalizate pentru fiecare valoare a variabilei interogate.

## 5. Rularea Programului

Programul este proiectat pentru a fi intuitiv, iar utilizatorul poate interacționa cu interfața prin intermediul unei console. Mai jos sunt pașii detaliați pentru rularea programului:

- **Pornirea Programului**

- Se rulează scriptul `main.py` din mediul Python.
- Utilizatorul este întâmpinat cu un mesaj de introducere și cu opțiunile disponibile:

Unset

Welcome to the Bayesian Network Inference Tool!

Enter the file path for the Bayesian network structure:

- Se introduce calea către fișierul JSON care definește structura rețelei bayesiene, de exemplu: `network.json`.

- **Selectarea unei Operații**

- După încărcarea rețelei, utilizatorul poate alege una dintre următoarele opțiuni:

Unset

Options:

1. Query a variable

2. Exit

- Se introduce opțiunea dorită (1 pentru interogare sau 2 pentru a ieși din program).

- **Realizarea unei Interogări**

- Dacă utilizatorul selectează opțiunea 1, este întrebat pe care variabilă dorește să o interogheze:

Unset

Enter the query variable:

- După introducerea numelui variabilei (de exemplu, `Gripă`), utilizatorul poate specifica evidențe suplimentare:

Unset

Enter evidence variables (leave empty to finish):

Evidence variable: Oboseală

Value for Oboseală (True/False): True

Evidence variable:

- Evidențele sunt variabile observate care afectează probabilitățile, iar utilizatorul le introduce până când finalizează apăsând **Enter** fără a introduce un nume de variabilă.
- **Obținerea Rezultatelor**
  - Programul calculează distribuția de probabilitate pentru variabila interogată și afișează rezultatele:

Unset

Probability distribution for Gripă:

True: 0.3963

False: 0.6037

- Acestea sunt afișate sub formă de valori numerice normalizate.
- **Repetarea sau Ieșirea**
  - Utilizatorul poate alege să repete procesul selectând din nou opțiunea **1** sau să închidă programul selectând opțiunea **2**.

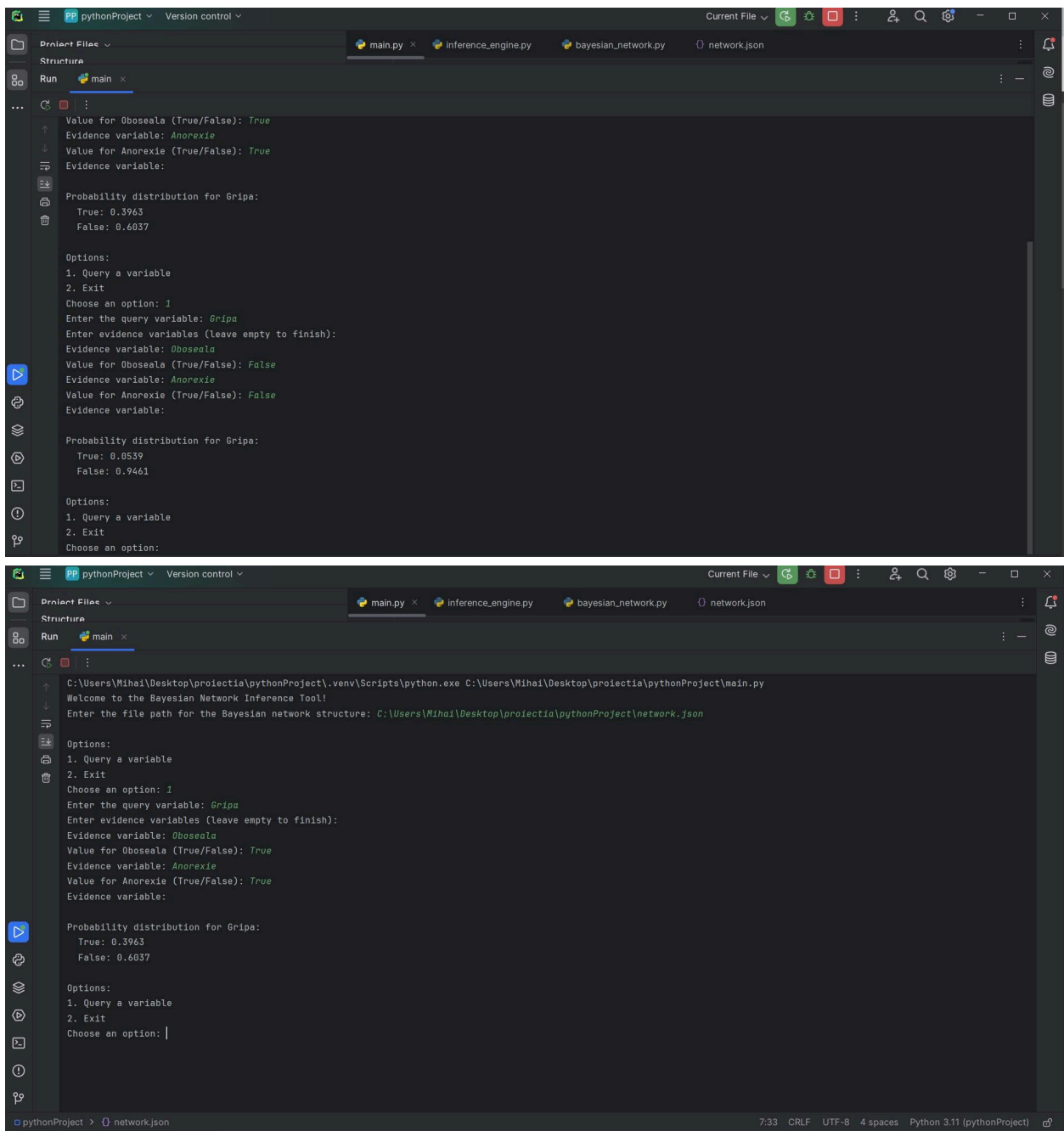
## **6. Rezultatele Obținute**

### **6.1 Rețele Utilizate pentru Testare**

1. Rețea pentru Diagnostic Medical: Gripă, Abces, Febră, Oboseală, Anorexie.
2. Rețea de Trafic Rutier: Ploaie, Trafic, Accident.

## 6.2 Capturi de Ecran cu rezultate

### 1. Diagnostic Medical:



```
pythonProject Version control Current File main.py inference_engine.py bayesian_network.py network.json
Structure
Run main
...
Value for Oboseala (True/False): True
Evidence variable: Anorexie
Value for Anorexie (True/False): True
Evidence variable:
Probability distribution for Gripa:
True: 0.3963
False: 0.6037
Options:
1. Query a variable
2. Exit
Choose an option: 1
Enter the query variable: Gripa
Enter evidence variables (leave empty to finish):
Evidence variable: Oboseala
Value for Oboseala (True/False): False
Evidence variable: Anorexie
Value for Anorexie (True/False): False
Evidence variable:
Probability distribution for Gripa:
True: 0.8539
False: 0.9461
Options:
1. Query a variable
2. Exit
Choose an option:

C:\Users\Mihai\Desktop\proiectia\pythonProject\.venv\Scripts\python.exe C:\Users\Mihai\Desktop\proiectia\pythonProject\main.py
Welcome to the Bayesian Network Inference Tool!
Enter the file path for the Bayesian network structure: C:\Users\Mihai\Desktop\proiectia\pythonProject\network.json
Options:
1. Query a variable
2. Exit
Choose an option: 1
Enter the query variable: Gripa
Enter evidence variables (leave empty to finish):
Evidence variable: Oboseala
Value for Oboseala (True/False): True
Evidence variable: Anorexie
Value for Anorexie (True/False): True
Evidence variable:
Probability distribution for Gripa:
True: 0.3963
False: 0.6037
Options:
1. Query a variable
2. Exit
Choose an option: |
```

**Interogare 1:** Probabilitatea nodului "Gripă" cu evidențele "Oboseală = True" și "Anorexie = True":

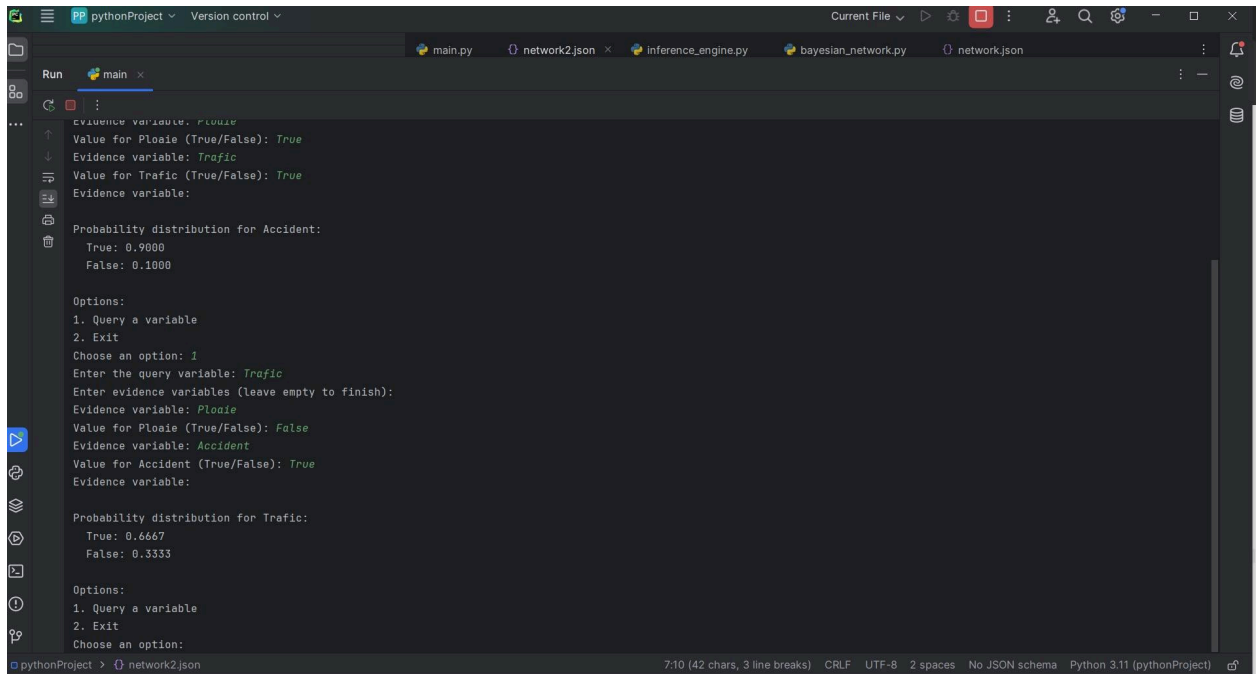
- Rezultat: Distribuția pentru "Gripă" este **True: 0.3963, False: 0.6037**.
- Explicație: Conform tabelor CPT, evidențele indică o probabilitate mai mare ca o persoană să nu aibă gripă în condițiile date.



**Interogare 2:** Probabilitatea nodului "Gripă" cu evidențele "Oboseală = False" și "Anorexie = False":

- c. Rezultat: Distribuția pentru "Gripă" este **True: 0.0539, False: 0.9461**.
- d. Explicație: Absența simptomelor reduce considerabil probabilitatea de gripă.

## 2. Trafic Rutier:



```
pythonProject Version control
Current File
main.py network2.json inference_engine.py bayesian_network.py network.json
Run main
...
Evidence variable: Ploaie
Value for Ploaie (True/False): True
Evidence variable: Trafic
Value for Trafic (True/False): True
Evidence variable:
Probability distribution for Accident:
True: 0.9000
False: 0.1000
Options:
1. Query a variable
2. Exit
Choose an option: 1
Enter the query variable: Trafic
Enter evidence variables (leave empty to finish):
Evidence variable: Ploaie
Value for Ploaie (True/False): False
Evidence variable: Accident
Value for Accident (True/False): True
Evidence variable:
Probability distribution for Trafic:
True: 0.6667
False: 0.3333
Options:
1. Query a variable
2. Exit
Choose an option:
pythonProject > network2.json 7:10 (42 chars, 3 line breaks) CRLF UTF-8 2 spaces No JSON schema Python 3.11 (pythonProject)
```

**Interogare:** Probabilitatea nodului "Trafic" cu evidențele "Ploaie = False" și "Accident = True":

- Rezultat: Distribuția pentru "Trafic" este **True: 0.6667, False: 0.3333**.
- Explicație: Deși nu plouă, prezența unui accident indică un trafic moderat sau intens datorită altor factori posibili.

## 7. Concluzii

Proiectul demonstrează că algoritmul de inferență prin enumerare este eficient pentru rețele bayesiene de dimensiuni mici și medii. Programul oferă o interfață intuitivă și poate fi extins pentru aplicații mai complexe prin optimizări sau utilizarea unor metode de inferență aproximativă.

## 8. Responsabilități

- **Mihai:** Implementarea rețelei bayesiene, dezvoltarea motorului de inferență
- **Elena:** JSON, Documentație.

## **9. Bibliografie**

1. Florin Leon, "Inteligență Artificială – Laborator 11."
2. Florin Leon, "Rețele Bayesiene," Universitatea Tehnică "Gheorghe Asachi" din Iași.
3. [https://en.wikipedia.org/wiki/Bayesian\\_network](https://en.wikipedia.org/wiki/Bayesian_network)
4. <https://seeing-theory.brown.edu/bayesian-inference/index.html>
5. [https://en.wikipedia.org/wiki/Bayesian\\_inference](https://en.wikipedia.org/wiki/Bayesian_inference)