Disciplina: Inteligență Artificială
Limbaj / platformă: C# (.NET), Windows Forms

Inferență prin enumerare în rețele bayesiene

Limbaj / platformă: C# (.NET), Windows Forms
Autor:
Chiriac Gabriel
Ciaușu Ioan-Călin
Data: Ianuarie 2026

1. Introducere. Descrierea problemei

O rețea bayesiană este un model probabilistic care reprezintă variabile aleatoare și relațiile dintre ele. În multe situații reale (diagnostic medical, detectarea fraudei, estimare risc, etc.) nu avem certitudini, ci doar probabilități. De aceea, este util un mecanism de inferență: putem calcula probabilitatea unui eveniment (variabilă interogată / *query*) având deja unele observații (variabile de evidență / *evidence*).

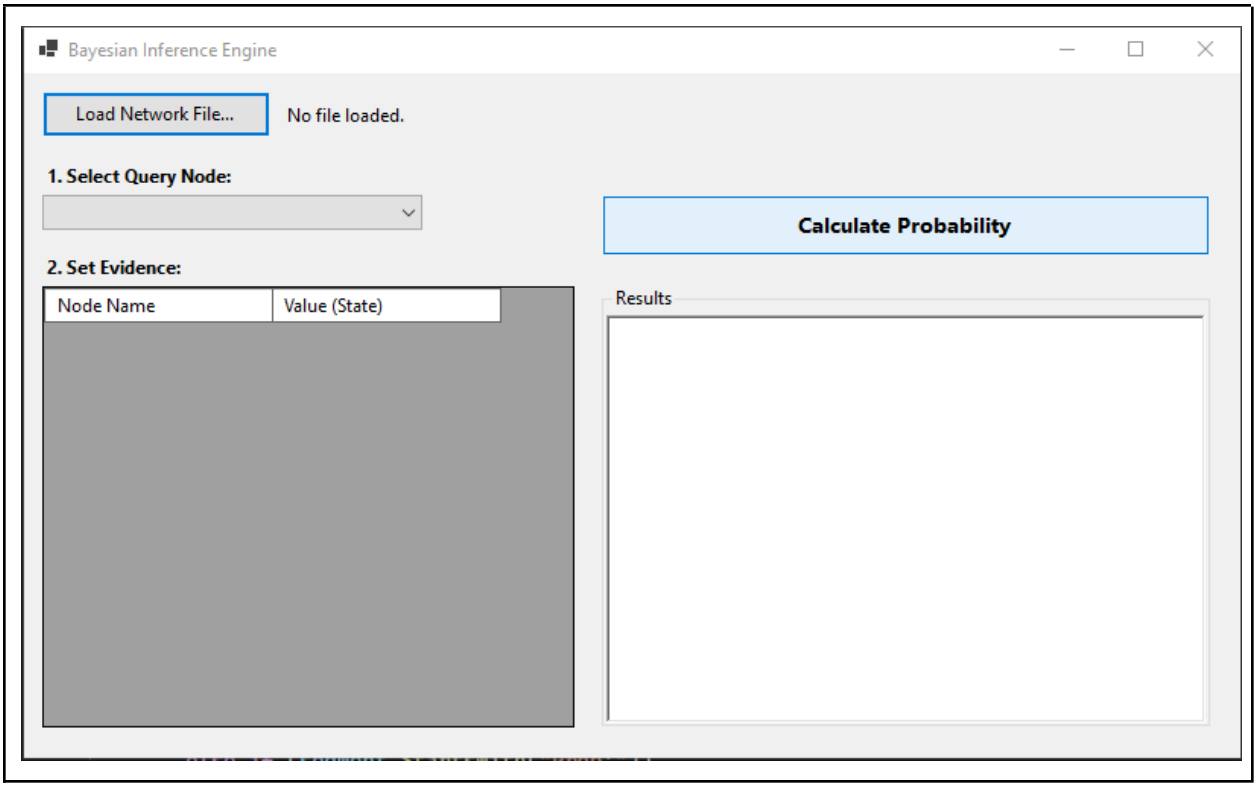
Cerința Proiectului 8 este implementarea **inferenței prin enumerare** pentru rețele bayesiene, cu funcționalitate similară laboratorului 11: programul trebuie să citească rețeaua din fișier, să permită setarea evidențelor și interogarea celorlalte noduri și să fie testat pe cel puțin două rețele.

Obiectivul aplicației realizate:

Aplicația încarcă o rețea bayesiană din fișier text (structură + parametri), apoi permite utilizatorului să:

- aleagă un nod „Query”;
- seteze evidențe pentru celelalte noduri (True/False/Unknown);
- calculeze și afișeze probabilitățile rezultate:
    - P(Query=True | Evidence)
    - P(Query=False | Evidence)

Interfața aplicației la pornire



2. Fundamente teoretice

2.1 Rețele bayesiene - definiție și interpretare

O rețea bayesiană este un graf orientat aciclic (DAG) în care:

- nodurile reprezintă variabile (evenimente),
- arcele reprezintă relații de dependență/cauzalitate/corelație,
- fiecare nod are asociat un tabel de probabilități: marginal (dacă nu are părinți) sau condiționat (dacă are părinți).
  LaboratorIA11

În cazul variabilelor binare (True/False), un nod fără părinți are un singur parametru independent (probabilitatea de True), iar un nod cu n părinți are 2*2^n intrări în CPT (pentru toate combinațiile de valori ale părinților).

2.2 Factorizarea distribuției comune (regula produsului)

Modelul bayesian presupune că o variabilă depinde doar de părinții săi. Distribuția comună se poate scrie ca produs al probabilităților condiționate:

$$P(x1,\ldots,xn)=\prod iP(xi \mid Parents(xi))P(x1,\ldots,xn)=i\prod P(xi \mid Parents(xi))$$

într-o ordine topologică în care părinții apar înaintea copiilor.

2.3 Inferența prin enumerare (exactă)

Scopul inferenței prin enumerare este calcularea probabilității unei variabile interogate (Query), date fiind variabilele observate (Evidență).

Ideea este:

- calculăm un produs de probabilități condiționate;
- pentru variabilele „ascunse” (care nu sunt nici evidență, nici query), sumăm peste toate valorile posibile (True și False).

Pentru a obține o distribuție validă asupra valorilor query-ului, se folosește normalizarea (coeficientul $\alpha\alpha$) astfel încât suma probabilităților să fie 1.

3. Cerințe și modul de rezolvare

3.1 Cerințele proiectului (extras)

Proiectul 8 impune:

- inferență prin enumerare în rețele bayesiene;

- citirea structurii și parametrilor rețelei din fișier (generic);
- interfață simplă, dar care permite interogări;
- setarea evidențelor și interogarea celorlalte noduri;
- testare pe cel puțin două rețele.
  CerinteProiectIA2024

3.2 Abordarea implementată

Implementarea este împărțită în două componente:

1. **Motor de inferență (logică):**
   - clasele `Node` și `BayesianNet` (încărcare rețea + enumerare).
2. **Interfață utilizator (Windows Forms):**
   - încărcare fișier;
   - selectare query;
   - setare evidențe în tabel;
   - calcul și afișare rezultat.

4. Structura aplicației și proiectare

4.1 Reprezentarea nodului (clasa Node)

Fiecare nod conține:

- `Name` - nume nod;
- `ParentNames` - lista părinților;
- `Probabilities` - probabilitatea P(True) (rădăcină) sau CPT (cu 2n2n intrări).

Fragment de cod - definirea structurii nodului:

```
public class Node
{
   public string Name { get; set; }
   public List<string> ParentNames { get; set; } = new List<string>();
   public List<double> Probabilities { get; set; } = new List<double>();
}
```

4.2 Accesarea probabilităților din CPT

Metoda `GetProbability(value, evidence)` întoarce:

- P(Node=True∣Parents)P(Node=True∣Parents) dacă `value==true`
- 1−P(Node=True∣Parents)1−P(Node=True∣Parents) dacă `value==false`

Când există părinți, indexul în CPT este calculat în stil „biți”, pe baza valorilor părinților din `evidence`.

Fragment de cod - GetProbability():

```
public double GetProbability(bool value, Dictionary<string, bool> evidence)
{
   double pTrue;

   if (ParentNames.Count == 0)
   {
       pTrue = Probabilities[0];
   }
   else
   {
       int index = 0;
       foreach (var parent in ParentNames)
       {
           bool parentVal = evidence[parent];
           index = (index << 1) | (parentVal ? 1 : 0);
       }
       pTrue = Probabilities[index];
   }

   return value ? pTrue : (1.0 - pTrue);
}
```

Observație: Convenția implementată este: pentru părinți `[A, B]` indexul este

- 0: (F,F)
- 1: (F,T)
- 2: (T,F)
- 3: (T,T).

5. Formatul fișierului de intrare

Aplicația citește rețeaua dintr-un fișier text în care fiecare linie definește un nod. Liniile goale sau comentariile (care încep cu #) sunt ignorate.

Structura unei linii:

- `Nod:<nume>`
- `Parinti:<lista>` sau `Parinti:None`
- `Prob:<valoare>` pentru nod fără părinți
- `CPT:[v1,v2,...]` pentru nod cu părinți (în ordinea combinațiilor)

Exemplu generic (nod rădăcină):

Nod: Angajat; Parinti: None; Prob: 0.85

Exemplu generic (nod cu părinți):

Nod: RataMica; Parinti: VenitMare; CPT: [0.30, 0.90]

Fragment de cod - LoadFromFile():

```
public void LoadFromFile(string filePath)
{
   Nodes.Clear();
   var lines = File.ReadAllLines(filePath);

   foreach (var line in lines)
   {
       if (string.IsNullOrWhiteSpace(line) || line.StartsWith("#")) continue;

       var parts = line.Split(';');
       var node = new Node();

       foreach (var part in parts)
       {
           var segment = part.Trim();
           if (segment.StartsWith("Nod:"))
           {
                       node.Name = segment.Substring(4).Trim();
           }
           else if (segment.StartsWith("Parinti:"))
           {
               var parentsStr = segment.Substring(8).Trim();
               if (!parentsStr.Equals("None", StringComparison.OrdinalIgnoreCase))
               {
                   node.ParentNames = parentsStr.Split(',').Select(p => p.Trim()).ToList();
               }
           }
           else if (segment.StartsWith("Prob:"))
           {
               var valStr = segment.Substring(5).Trim();
               node.Probabilities.Add(double.Parse(valStr, CultureInfo.InvariantCulture));
           }
           else if (segment.StartsWith("CPT:"))
           {
               var arrayStr = segment.Substring(4).Trim().Trim('[', ']');
               var vals = arrayStr.Split(',')
                                  .Select(v => double.Parse(v,
CultureInfo.InvariantCulture));
               node.Probabilities.AddRange(vals);
           }
       }
       Nodes.Add(node);
   }
}
```

Exemplu de fișier .txt încărcat

```
Nod: Angajat; Parinti: None; Prob: 0.85
Nod: IstoricBun; Parinti: None; Prob: 0.70
Nod: VenitMare; Parinti: Angajat; CPT: [0.10, 0.80]
Nod: RataMica; Parinti: VenitMare; CPT: [0.30, 0.90]
Nod: Aprobare; Parinti: VenitMare, IstoricBun, RataMica; CPT: [0.01, 0.20, 0.30, 0.60, 0.40, 0.70, 0.85, 0.99]
```

6. Implementarea algoritmului de inferență prin enumerare

Algoritmul este inspirat din forma clasică (Russell & Norvig / AIMA) și din explicația laboratorului 11, unde variabilele necunoscute sunt enumerate prin sumare, iar rezultatul se normalizează prin αα.

6.1 Funcția EnumerationAsk

`EnumerationAsk(X, e, bn)` calculează:

- P(X=True∣e)P(X=True∣e) și P(X=False∣e)P(X=False∣e)
  prin două apeluri ale funcției recursive `EnumerateAll`, apoi normalizează.

```
public static double[] EnumerationAsk(string X, Dictionary<string, bool> e,
BayesianNet bn)
{
   var Q = new double[2];

   var eTrue = new Dictionary<string, bool>(e);
   eTrue[X] = true;
   double pTrue = EnumerateAll(bn.Nodes, eTrue);

   var eFalse = new Dictionary<string, bool>(e);
   eFalse[X] = false;
   double pFalse = EnumerateAll(bn.Nodes, eFalse);

   double sum = pTrue + pFalse;
   return new double[] { pTrue / sum, pFalse / sum };
}
```

6.2 Funcția recursivă EnumerateAll

`EnumerateAll(vars, e)`:

- dacă lista este goală → 1;

- dacă variabila curentă YY este în evidență → înmulțește cu $P(Y \mid Parents(Y))P(Y \mid Parents(Y))$;
- altfel → face sumare peste `True` și `False`.

```
public static double EnumerateAll(List<Node> vars,Dictionary<string,bool> e)
{
   if (vars.Count == 0) return 1.0;

   var Y = vars[0];
   var rest = vars.Skip(1).ToList();

   if (e.ContainsKey(Y.Name))
   {
       double probY = Y.GetProbability(e[Y.Name], e);
       return probY * EnumerateAll(rest, e);
   }
   else
   {

       var eTrue = new Dictionary<string, bool>(e);
       eTrue[Y.Name] = true;
       double term1;
       term1 = Y.GetProbability(true,eTrue)* EnumerateAll(rest,eTrue);

       var eFalse = new Dictionary<string, bool>(e);
       eFalse[Y.Name] = false;
       double term2;
       term2 = Y.GetProbability(false,eFalse) * EnumerateAll(rest,eFalse);

       return term1 + term2;
   }
}
```

Observație importantă (ordine topologică):

În laborator se recomandă sortarea topologică a variabilelor pentru eficiență și corectitudinea accesării părinților înaintea copiilor.

În implementarea curentă, se presupune că nodurile din fișier sunt furnizate în această ordine (părinții apar înaintea copiilor). Dacă ai nevoie, în viitor se poate adăuga un algoritm de sortare topologică după citirea rețelei.

7. Interfața cu utilizatorul (GUI)

Aplicația are o interfață grafică simplă (conform cerinței proiectului) care permite:

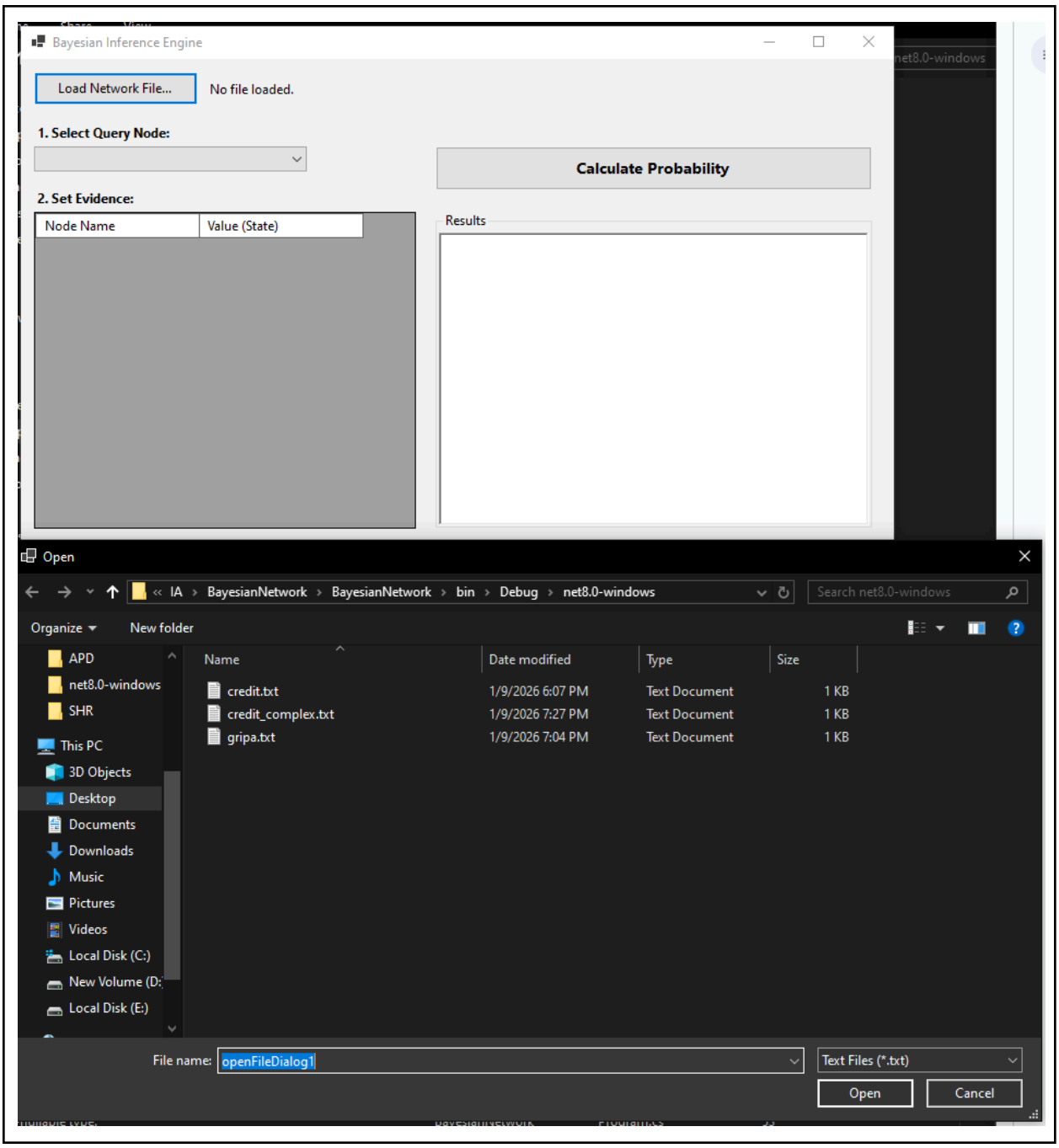
- încărcarea fișierului rețelei;
- selectarea nodului Query dintr-un combobox;
- setarea evidențelor într-un tabel cu valori `Unknown / True / False`;

- afișarea rezultatului în zona „Results”.

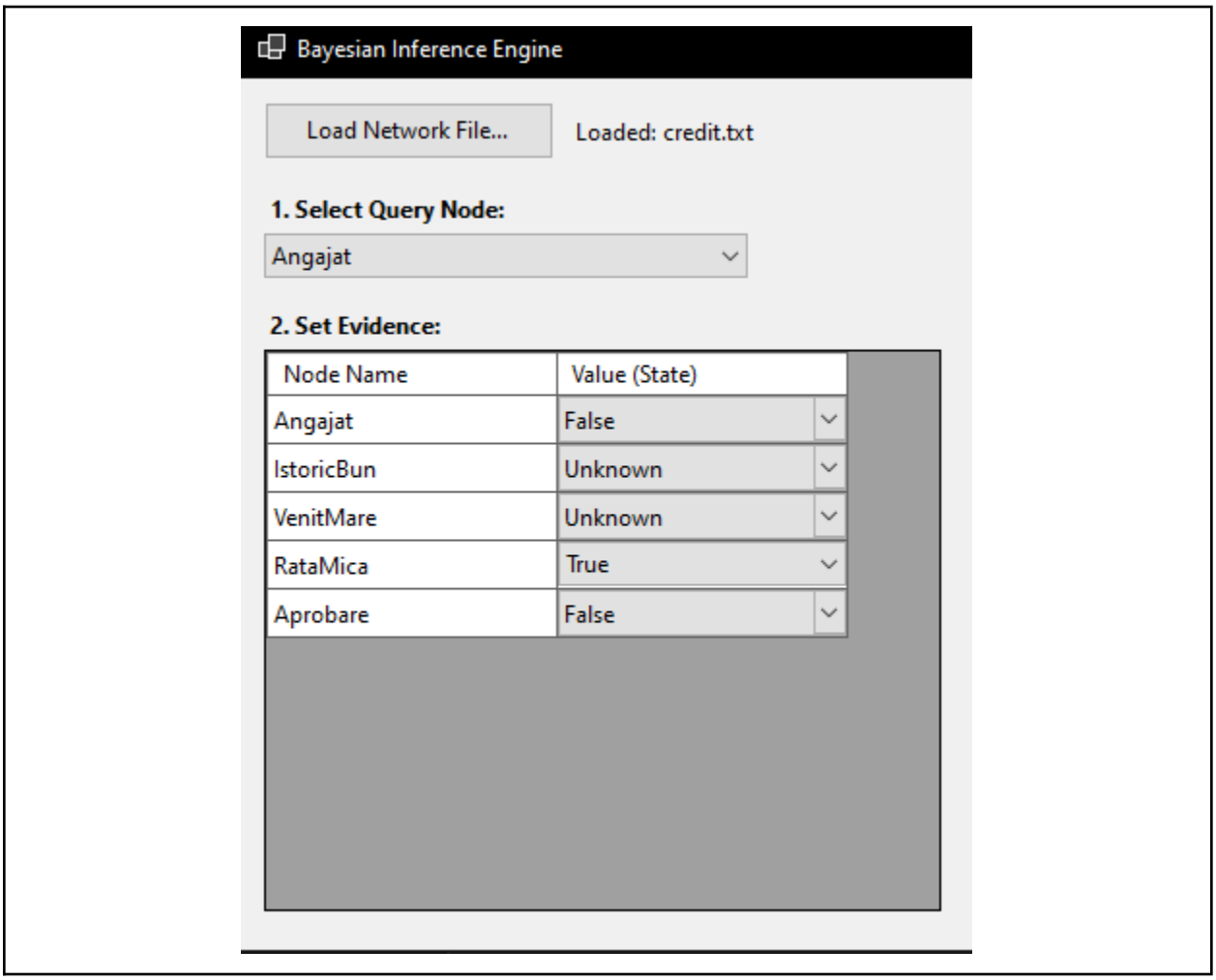
7.1 Încărcarea fișierului

La apăsarea butonului **Load Network File...**, utilizatorul alege fișierul; rețeaua este încărcată și se populază controalele UI.

Selectarea fișierului din OpenFileDialog



După încărcare: combobox populat + evidențe populate



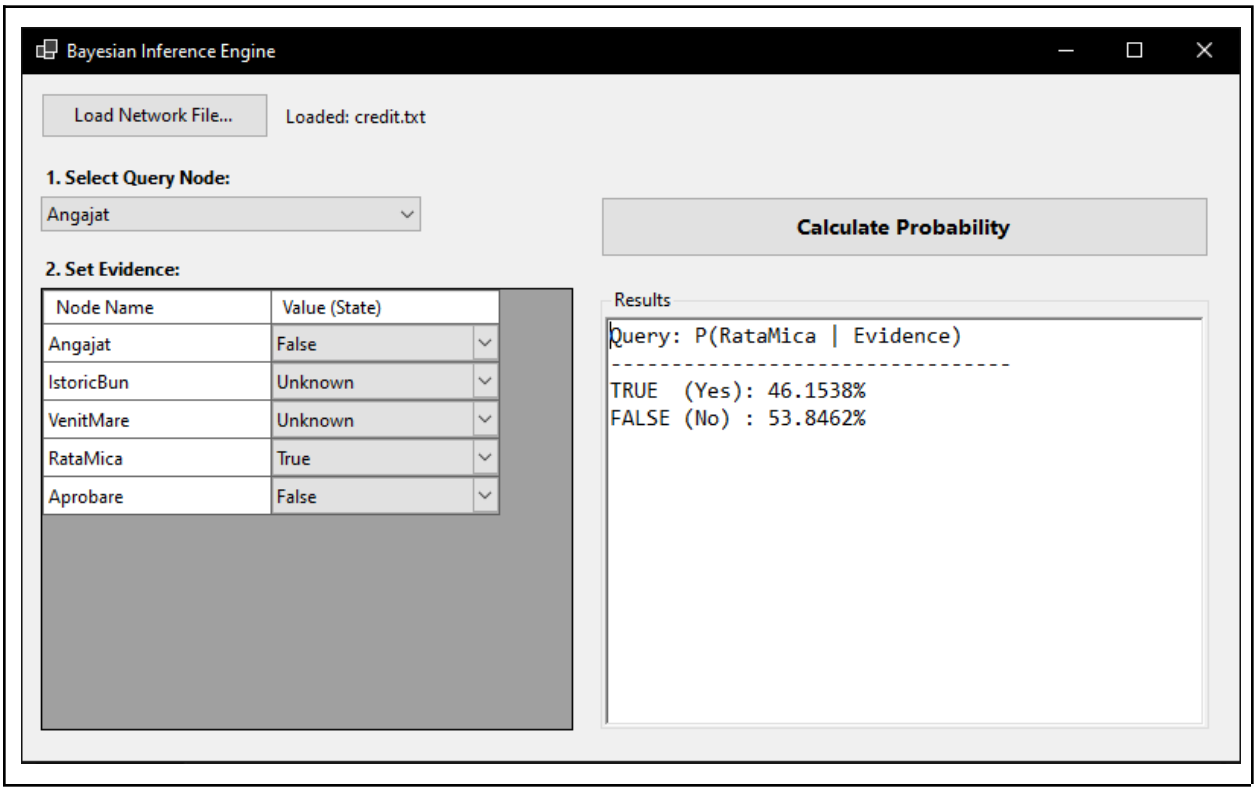
7.2 Blocarea setării evidenței pe Query

Când utilizatorul schimbă query-ul, rândul respectiv din grid este blocat pe optiunea de *Unknown*.

7.3 Calculul probabilităților

Butonul **Calculate Probability** citește evidențele din tabel, construiește dicționarul de evidențe și apelează `EnumerationAsk`.

Rezultatul afișat în zona Results



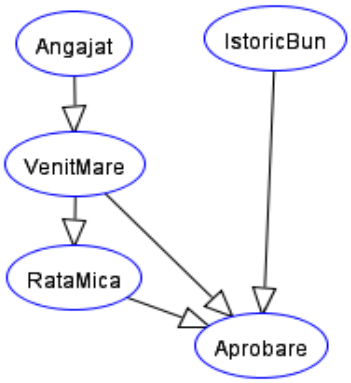
8. Testare și rezultate

Mai jos sunt două seturi de teste.

8.1 Rețeaua 1 - Credit simplu

Fișier: *credit.txt*
Descriere: rețea de evaluare risc de neplata al un credit bancar cu 5 noduri

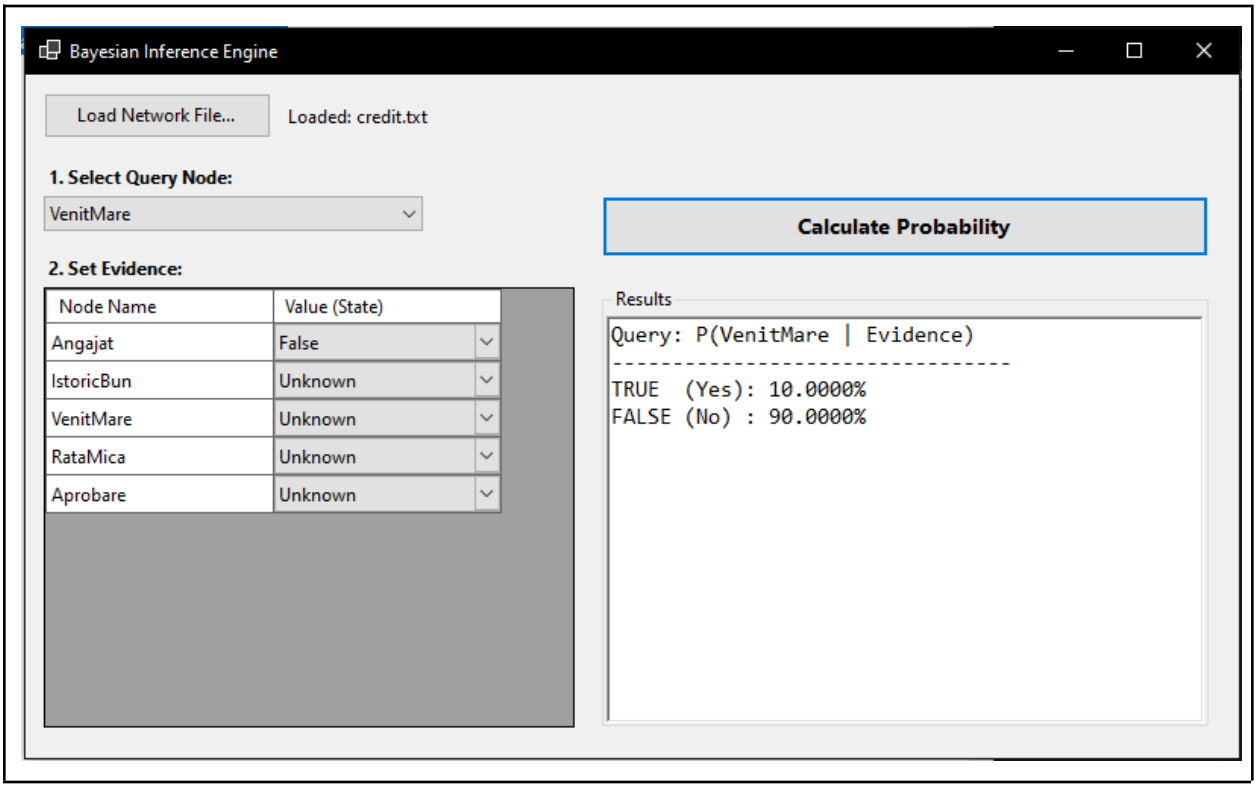
Nod: Angajat; Parinti: None; Prob: 0.85
Nod: IstoricBun; Parinti: None; Prob: 0.70
Nod: VenitMare; Parinti: Angajat; CPT: [0.10, 0.80]
Nod: RataMica; Parinti: VenitMare; CPT: [0.30, 0.90]
Nod: Aprobare; Parinti: VenitMare, IstoricBun, RataMica; CPT: [0.01, 0.20, 0.30, 0.60, 0.40, 0.70, 0.85, 0.99]



Test 1:

- **Query:** *Angajat*
- **Evidence:** Angajat = False
- **Rezultat:**
  - **P(Q1=True | evidence)**=10%
  - **P(Q1=False | evidence)**=90%
- **Observatie:** Test de baza, presupune doar analiza nodului Angajat. (*Nod: VenitMare; Parinti: Angajat; CPT: [0.10, 0.80]*)

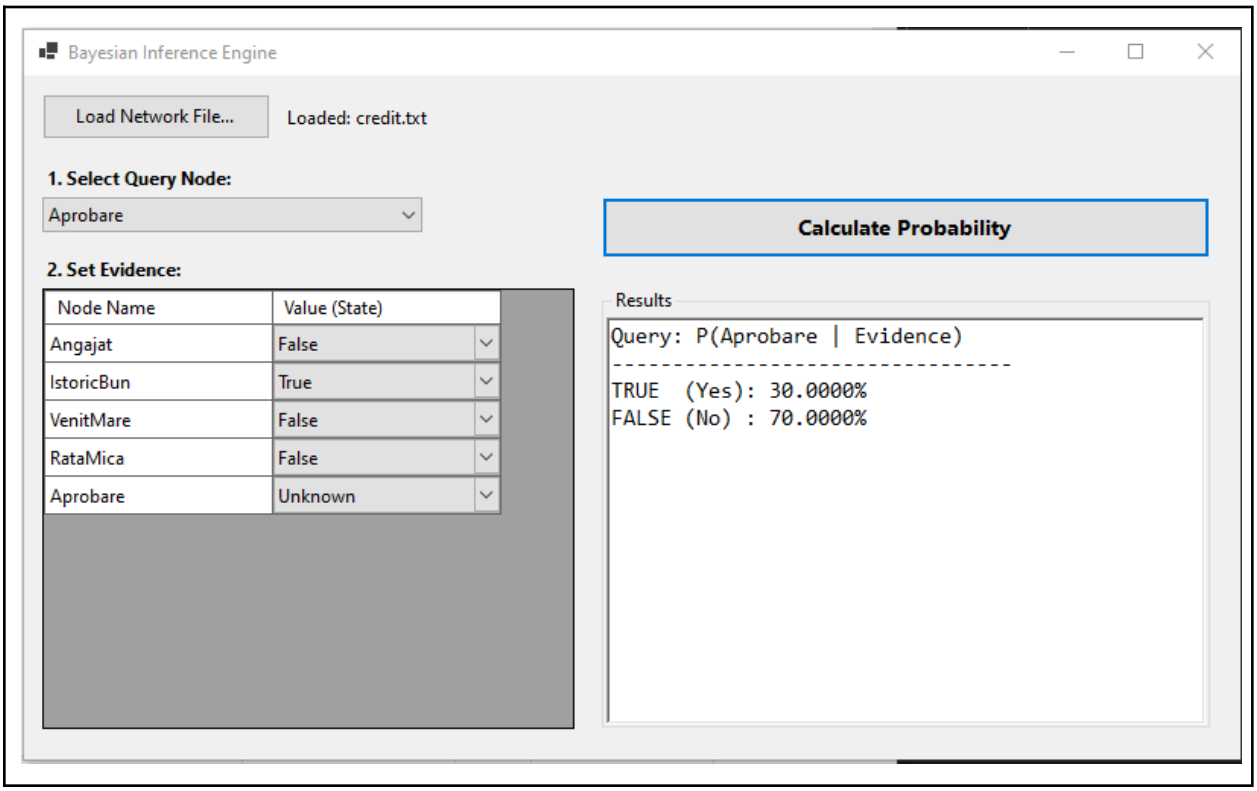
Rețeaua 1 încărcată + evidențe + rezultat



Test 2:

- **Query: Aprobare**
- **Evidence:** Angajat = False , IstoricBun = True, VenitMare = False, RataMica = False
- **Rezultat:**
  - **P(Q1=True | evidence)**=30%
  - **P(Q1=False | evidence)**=70%
- **Observații:** Vom cumpara cu rezultatul obtinut in aplicatia din laboratorul 11.

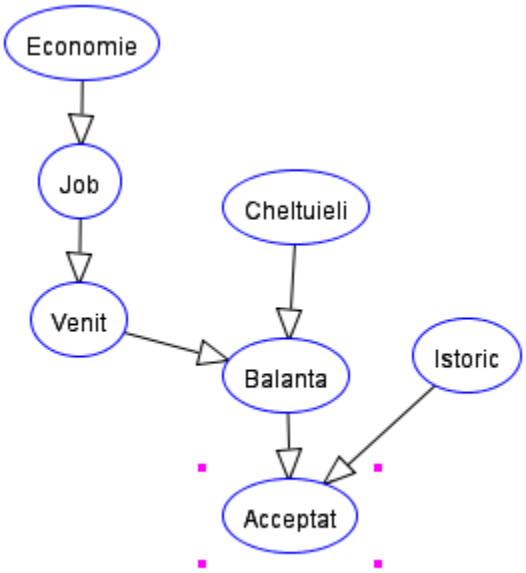
Rețeaua 1 încărcată + evidențe + rezultat



8.2 Rețeaua 2 - Credit Complex

Fișier: *credit_complex.txt*
Descriere: rețea de evaluare risc de neplata al un credit bancar cu 7 noduri

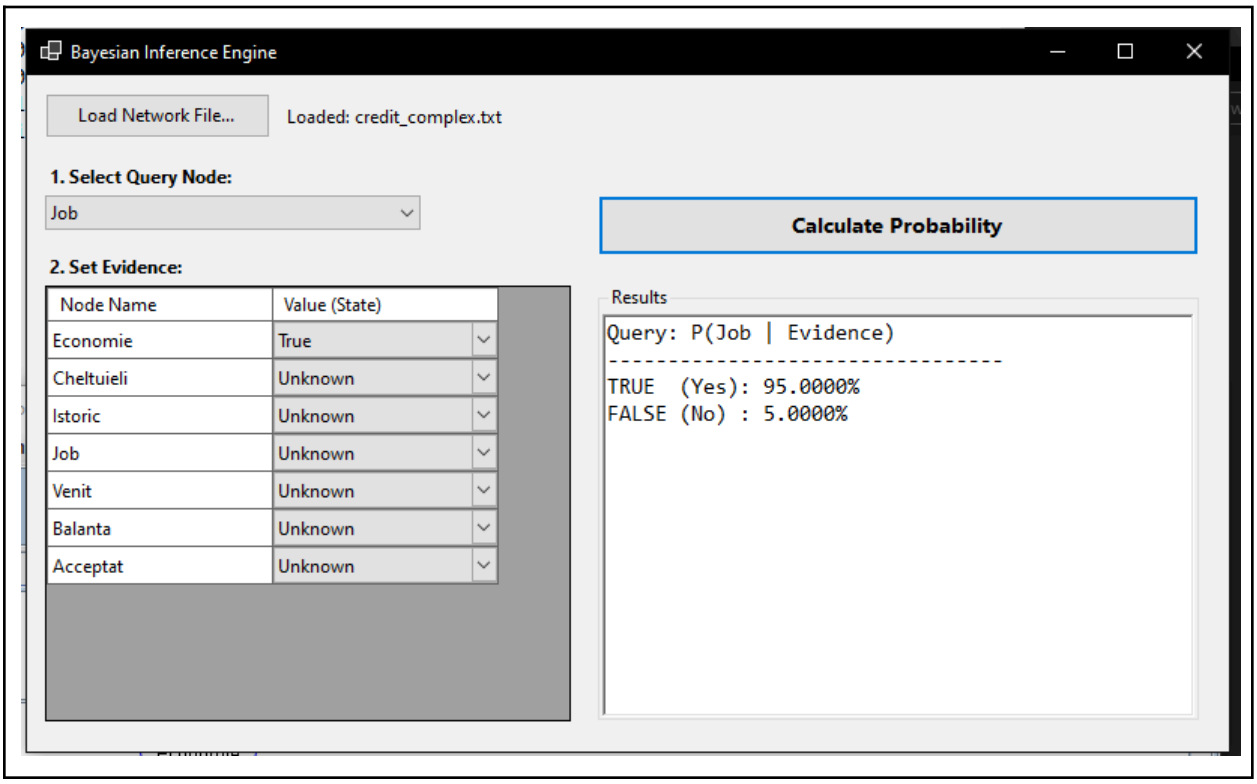
Nod: Economie; Parinti: None; Prob: 0.70
Nod: Cheltuieli; Parinti: None; Prob: 0.60
Nod: Istoric; Parinti: None; Prob: 0.75
Nod: Job; Parinti: Economie; CPT: [0.20, 0.95]
Nod: Venit; Parinti: Job; CPT: [0.05, 0.85]
Nod: Balanta; Parinti: Venit, Cheltuieli; CPT: [0.40, 0.05, 0.95, 0.30]
Nod: Acceptat; Parinti: Balanta, Istoric; CPT: [0.99, 0.40, 0.25, 0.01]



Test 1:

- **Query:** Job
- **Evidence:** Economie = True
- **Rezultat:**
  - **P(Q=True | evidence)**=95%
  - **P(Q=False | evidence)**=5%
- **Observatie:** Query simplu cu rezulatul preluat direct din fisierul de intrare. (Nod: Job; Parinti: Economie; CPT: [0.20, 0.95])

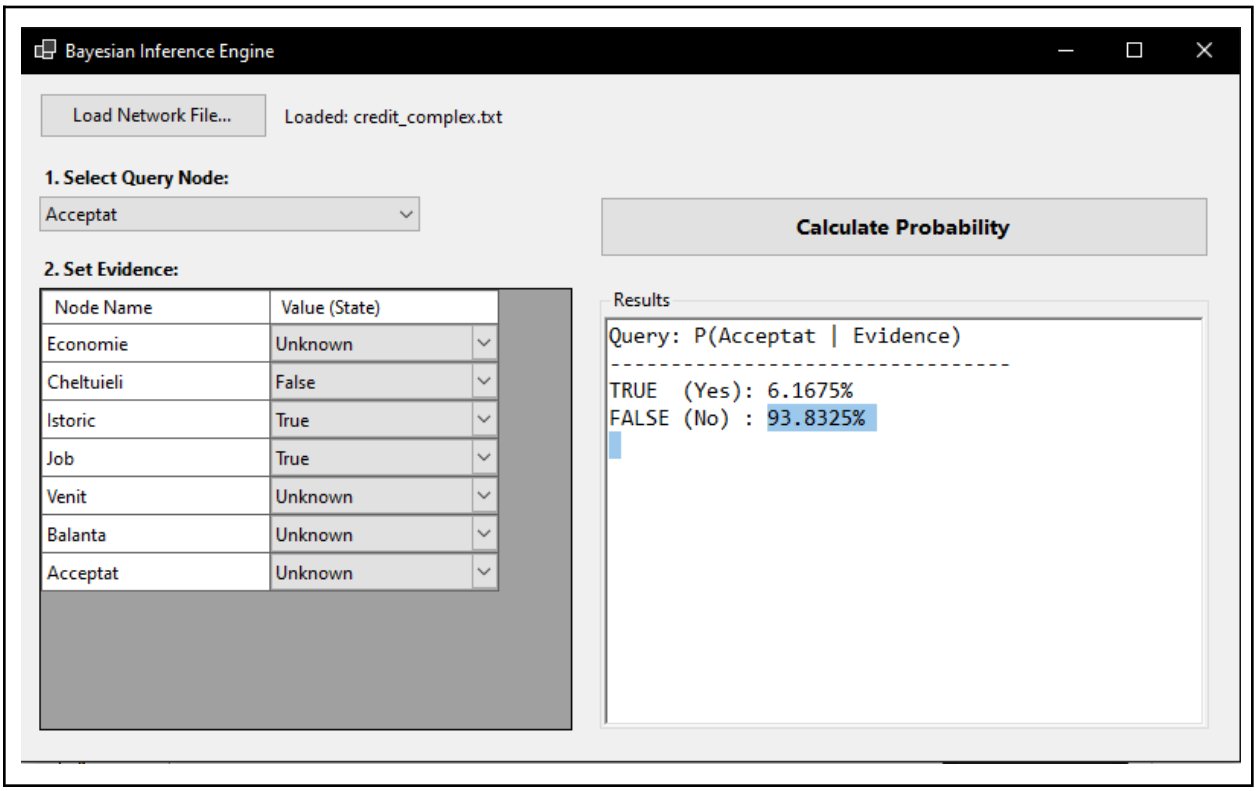
Rețeaua 2, test 1



Test 2:

- **Query:** Acceptat
- **Evidence:** Cheltuieli = False, Job = True, Istoric = True;
- **Rezultat:**
  - **P(Q=True | evidence)**=6.1675%
  - **P(Q=False | evidence)**=93.8325%

Rețeaua 2, test 2



9. Concluzii

Aplicația implementează inferența exactă prin enumerare pentru rețele bayesiene, citind structura și parametrii din fișier și permițând utilizatorului să seteze evidențe și să interogheze un nod ales. Aceasta respectă cerința de interfață simplă și suport generic pentru rețele diferite (prin fișier).

Avantaje:

- implementare generală (rețeaua nu este hardcodată);
- interfață intuitivă;
- rezultate exacte (enumerare completă).

Limitări:

- complexitate mare pentru rețele cu multe noduri (enumerarea crește exponențial);
- se presupune (în forma curentă) că nodurile sunt introduse în fișier în ordine topologică (părinții înaintea copiilor), idee recomandată și în laborator pentru eficiență.

10.Impartirea sarcinilor

1. Interfata grafica: Ciauşu Ioan-Călin
2. Parsare fisier de intrare: Chiriac Gabriel
3. EnumerationAll: Ciauşu Ioan-Călin
4. EnumerateAsk si structura nod: Chiriac Gabriel

11. Bibliografie

1. Florin Leon, *Inteligență artificială - Cursul 10: Rețele bayesiene.*
2. Florin Leon, *Inteligență artificială - Laborator 11: Rețele bayesiene. Inferența prin enumerare*.
3. Cerințe proiect Inteligență Artificială 2024-2025 (Proiectul 8: inferență prin enumerare în rețele bayesiene).