UNIVERSITATEA TEHNICĂ "GHEORGHE ASACHI" DIN IAȘI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

SPECIALIZAREA CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

DISCIPLINA INTELIGENȚĂ ARTIFICIALĂ

APLICAȚIE REȚELE BAYESIENE

Coordonator,

Florin Leon

Studenți,

Vlad Batalan

Nicoleta-Ecaterina Mihalache

Iosif-Marcelin Petrișor

Iași, 2022

Cuprins

Capitolul 1. Descrierea problemei considerate

Proiectul de față abordează tema inferenței prin enumerare în rețele bayesiene.

Scopul programului realizat este acela de a putea citi dintr-un fișier structura și parametrii unei rețele bayesiene și de a afișa pe interfață rețeaua construită. Totodată, acesta permite setarea valorilor nodurilor evidență precum și interogarea celorlalte noduri.

Astfel, algoritmul este unul general, fiind capabil să citească și să construiască orice rețea.

Capitolul 2. Aspecte teoretice privind algoritmul

2.1 Rețea bayesiană

O rețea bayesiană este un model grafic probabilistic, adică un graf orientat aciclic cu o mulțime de noduri, care reprezintă evenimente aleatorii, conectate de arce, care reprezintă dependențe condiționate între evenimente. Ideea de bază este că un nod depinde doar de „părinții" săi, adică de o submulțime a nodurilor din rețea, nu de toate celelalte noduri, ceea ce ar implica folosirea distribuției comune de probabilitate. Altfel spus, fiecare nod este independent condiționat de predecesorii din șirul ordonat al nodurilor, dați fiind părinții nodului.

2.2 Algoritmul Bayes-Ball

Algoritmul Bayes-Ball reprezintă o modalitate simplă de a determina relațiile de independență și dependență condiționată într-o rețea bayesiană.

Pentru exemplificarea algoritmului vom lua următorul exemplu: se presupune că o minge este trimisă dintr-un nod în rețea. Aceasta poate trece în moduri diferite, în funcție de cine o trimite (nod copil sau nod părinte) și starea nodului care o privește (observat sau neobservat). Nodurile la care mingea nu ajunge se numesc noduri independente sau condiționate de nodul de start.

2.3 Sortare topologică

Sortarea topologică a unui graf este o ordonare liniară a nodurilor sale astfel încât, pentru fiecare arc A -> B, A apare înaintea lui B. Pentru un graf aceasta asigură faptul că nodurile părinte vor apărea înaintea nodurilor copil. Există mai mulți algoritmi de sortare topologică, dar cel folosit în aplicație este algoritmul lui Kahn.

2.4 Inferență prin enumerare

Scopul inferenței prin enumerare este de a calcula probabilitatea unei variabile interogate, date fiind variabilele observate. Ideea de bază a procedeului este calcularea unui produs de probabilități condiționate, însă în cazul variabilelor despre care nu se cunoaște nimic, se sumează variabilele corespunzătoare tuturor valorilor acestora. Astfel, putem răspunde la orice întrebare referitoare la evenimentele codate în rețea. Având în vedere niște evidențe, adică observații sau

evenimente despre care știm că s-au întâmplat, putem calcula probabilitățile tuturor celorlalte noduri din rețea.

Capitolul 3. Modalitatea de rezolvare

3.1 Descrierea algoritmului

Având ca date de intrare configurația arborescentă a unei rețele Bayesiene și observațiile făcute asupra rețelei, algoritmul urmărește să evalueze care sunt probabilitățile de realizare a fiecărei stări din domeniul nodului țintă. Formula de calcul utilizată pentru evaluarea unui query efectuat asupra nodului *Np* al unui arbore care are k noduri denumite: *N1, N2, N3, N4, … , Np,…, Nk,* fiecare având câte *Si* stări în domeniu este:

$$P(Np_{St} | Lista_{observate}) = \alpha \cdot (\sum_{n1 \in D_{N1\ neobservat}} \dots \sum_{nh \in D_{Nh\ neobservat}} P(Lista_{observate}, n1, n2, \dots, nh))$$

Semnificație variabile: *St* este a t-a stare a nodului *Np*, Lista observate este lista nodurilor observate, iar *n1, n2, …, nh* sunt variabilele corespunzătoare nodurilor neobservate.

Pseudocod sursa: Inteligenta artificiala, http://florinleon.byethost24.com/curs_ia.html

function ENUMERATION-ASK(X, **e**, *bn*) **returns** a distribution over X
  inputs: X, the query variable
    e, observed values for some set of variables **E**
    bn, a Bayes net
  Q ← a distribution over X, where **Q**(x_i) is P(X=x_i)
  for each value x_i that X can have **do**
    Q(x_i) ← ENUMERATE-ALL(*bn*.VARS, $\mathbf{e}_{x_i}$), where $\mathbf{e}_{x_i}$ is the evidence **e** plus the assignment X=x_i
  return NORMALIZE(**Q**)

function ENUMERATE-ALL(*vars*, **e**) **returns** a probability (a real number in [0,1])
  inputs: *vars*, a list of all the variables
    e, observed values for some set of variables **E**
  if EMPTY(*vars*) **then return** 1.0
  Y ← FIRST(*vars*)
  if Y is assigned a value (call it y) in **e then**
    return P(Y=y | values assigned to Y's parents in **e**) × ENUMERATE-ALL(REST(*vars*), **e**)
  else
    return $\sum_{y_i}$ [P(Y=y_i | values assigned to Y's parents in **e**) × ENUMERATE-ALL(REST(*vars*), $\mathbf{e}_{y_i}$)],
      where $\mathbf{e}_{y_i}$ is the evidence **e** plus the assignment Y=y_i

Algoritmul implementat pentru soluționarea unei rețele Bayesiene este compus din două funcții: *SolutioneazaRetea* și *EnumerateAll*.

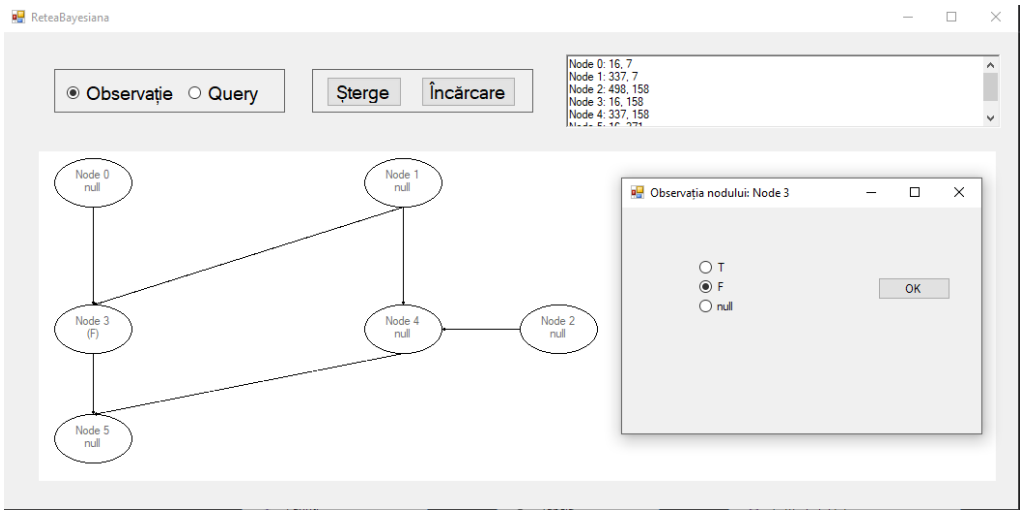
SolutioneazaRetea este metoda responsabilă cu rezolvarea unei rețele Bayesiene. Aceasta primește ca parametrii de intrare rețeaua pe care se face discuția calculului probabilităților și nodul țintă căruia îi dorim să îi aflăm posibilitățile fiecărei stări. Modul prin care lucrează această funcție este acela de a evalua probabilitatea în fiecare stare a nodului țintă și de a afla coeficientul de scalare a probabilităților. Odată aflat, se aplică o normalizare pe lista de probabilități soluție în așa fel încât suma acestora să dea 1.

EnumerateAll este o metodă recurentă care are ca scop evaluarea sumei de probabilități având primit ca parametru o listă de variabile care semnifică acele noduri pentru care trebuie făcută evaluarea. Această listă de noduri trebuie să conțină elementele sortate topologic pentru a realiza calculele în mod eficient. Pentru variabilele care au deja o stare observabilă setată, va returna valoarea probabilității ori valoarea funcției evaluate pentru restul variabilelor netratate. Pentru acele variabile care nu au o stare, pentru fiecare stare din domeniu, se va seta temporar observația nodului și va evalua restul variabilelor rămase. În final, va aduna toate probabilitățile venite de pe fiecare stare.

3.2 Arhitectura aplicației

3.2.1 Interfața aplicației

Interfața aplicație permite acțiuni precum încărcarea rețelei Bayesiene dintr-un fișier de tip .xml și reprezentarea grafică a acesteia în cadrul spațiului de desenare, ștergerea configurației rețelei curente, posibilitatea de a face observații asupra nodurilor din rețea și interogarea unui nod de rețea, în urma căreia va apărea o fereastră de tip pop-up cu probabilitățile asociate fiecărei stări din domeniul de existență al nodului ales.

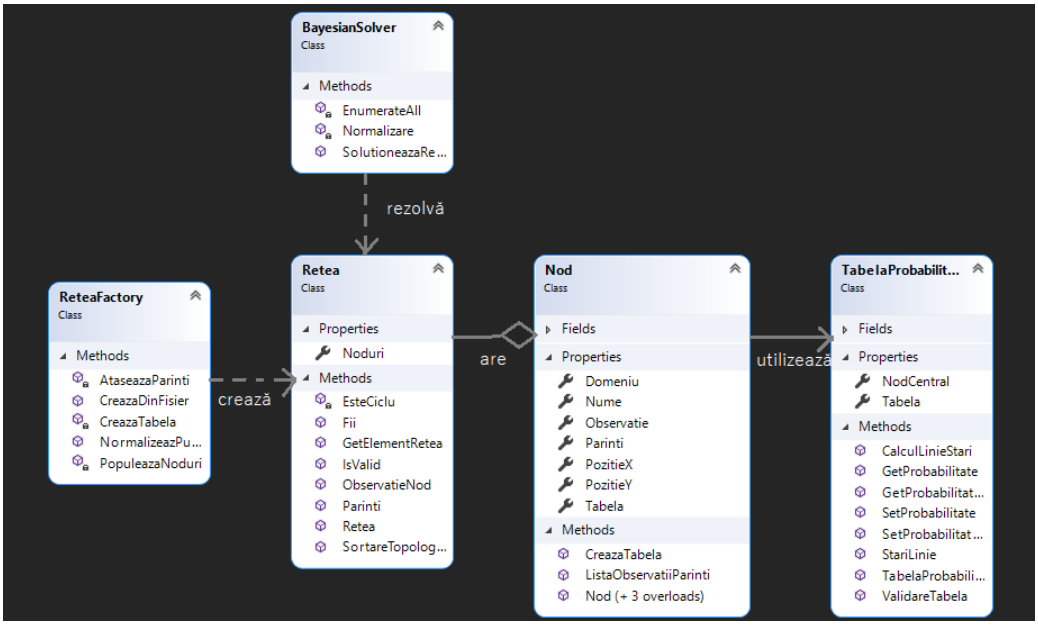


3.2.2 Structura backend-ului

Pentru reprezentarea unei rețele Bayesiene ce poate fi utilizată cu ușurință atât în partea de interfață a aplicației, cât și în modulele de deserializare și de rezolvare a unei astfel de rețele,

s-a utilizat paradigma OOP. Astfel, o componentă de tip Rețea conține o listă de noduri. Fiecare nod are la rândul său asociată o tabelă de probabilități, dar și lista părinților acestui nod.

Celelalte două componente, ReteaFactory și BayesianSolver, utilizează rețele Bayesiene. ReteaFactory este responsabil cu deserializarea unei rețele dintr-un fișier XML, pe când ReteaSolver este responsabil cu soluționarea unei rețele de acest tip.



Capitolul 4. Părți semnificative din codul sursă

4.1 Codul aferent rezolvării unei rețele Bayesiene

```
/// <summary>
/// Metoda statica responsabila cu rezolvarea unui query
/// </summary>
/// <param name="retea"> Reteaua asupra careia executa query-ul </param>
/// <param name="targetNod"> Nodul asupra caruia se realizeaza calcule </param>
/// <returns> Lista cu probabilitatile asignate fiecarei stari ale nodului </returns>

public static double[] SolutioneazaRetea(Retea retea, Nod targetNod)
{
	// Rezultatul care trebuie intors la final
	double[] rezultat = new double[targetNod.Domeniu.Count];

	// Cazul in care nodul este deja observat, se cunoaste deja raspunsul
	if (targetNod.Observatie != null)
	{
		for (int indexStare = 0; indexStare < rezultat.Length; indexStare++)
		{
```

```
                    if (targetNod.Domeniu[indexStare].Equals(targetNod.Observatie))
                    {
                        rezultat[indexStare] = 1.0;
                    }
                    else
                    {
                        rezultat[indexStare] = 0.0;
                    }
                }
                return rezultat;
        }

        // Lista de noduri care mai trebuie vizitate
        List<Nod> vars = new List<Nod>();

        // Pentru fiecare stare, calculez probabilitatea stării respective
        for(int indexStare = 0; indexStare < rezultat.Length; indexStare++)
        {
                // Setez observație pentru nodul respectiv
                targetNod.Observatie = targetNod.Domeniu[indexStare];

                // Reimprospatez vars
                vars.Clear();
                vars.AddRange(retea.Noduri);

                // Calculez lista
                rezultat[indexStare] = EnumerateAll(vars);
        }

        targetNod.Observatie = null;

        // Normalizarea raspunsului
        Normalizare(rezultat);

        // Intoarcerea raspunsului
        return rezultat;
}


/// <summary>
/// Metoda recursiva pentru calcului unei probabilitati din reteaua Bayesiana
/// </summary>
/// <param name="vars"> Lista cu nodurile rămase de procesat (sortata Topologic)
</param>
/// <returns> Probabilitatea variabilei </returns>

private static double EnumerateAll(List<Nod> vars)
{
        // Daca lista este goala, returneaza 1
        if (vars.Count == 0) return 1.0;

        // Extrage nodul care trebuie procesat
        Nod firstNode = vars[0];
        vars.RemoveAt(0);

        // Construieste lista de Observatii ale parintilor
        string[] observatiiParinti = firstNode.ListaObservatiiParinti();

        // Dacă starea lui firstNode este observată
        if (firstNode.Observatie != null)
        {
                // Generez o lista noua de elemente
                List<Nod> rest = new List<Nod>(vars);
```

```
            // Returnează răspuns
            double result = firstNode.Tabela.GetProbabilitateStari(observatiiParinti,
        firstNode.Observatie) * EnumerateAll(rest);

            return result;
        }

        // Nodul curent nu contine o stare cunoscută
        // Evaluăm sumele cand setam nodul în toate starile
        double sumProbs = 0;

        // Pentru fiecare stare pe care o poate avea nodul curent
        for (int indexStare = 0; indexStare < firstNode.Domeniu.Count; indexStare++)
        {
            // Generez o lista noua de elemente
            List<Nod> rest = new List<Nod>(vars);

            // Setez nodul ca avand starea ca observatie
            firstNode.Observatie = firstNode.Domeniu[indexStare];

            // Update suma
            sumProbs += firstNode.Tabela.GetProbabilitateStari(observatiiParinti,
        firstNode.Observatie) * EnumerateAll(rest);
        }

        // Reseteaza observatia nodului curent la null
        firstNode.Observatie = null;

        // Returnează rezultatul calculului
        return sumProbs;
}
```

4.2 Codul aferent populării unei rețele Bayesiene cu noduri extrase dintr-un fișier xml.

```
/// <summary>
/// Metodă ce populează rețeaua cu noduri. Totodată nodurile respective conțin și o
poziție relativă pe interfață
/// </summary>
/// <param name="xmlNodeList"> lista cu elemente extrase din xml</param>
/// <returns> lista noduri</returns>

private static List<Nod> PopuleazaNoduri(XmlNodeList xmlNodeList)
{
        List<Nod> noduri = new List<Nod>();

        // parcurgem lista de noduri din xml
        foreach (XmlNode xmlNode in xmlNodeList)
        {
            List<string> domeniu = new List<string>();

            // lista noduri cu starile
            XmlNodeList xmlStari = xmlNode.SelectNodes("OUTCOME");

            //pentru fiecare nod se adauga starea in domeniu
            foreach (XmlNode xmlNode1 in xmlStari)
            {
                string stare = xmlNode1.InnerText;
                domeniu.Add(stare);
```

```
            }

            // un singur nod in xml ce contine pozitiile
            XmlNode xmlPozitii = xmlNode.SelectSingleNode("PROPERTY");

            //regex ce ajuta la extragerea pozitiilor
            string pattern = @"[0-9]+\.[0-9]+";
            Regex rg = new Regex(pattern);
            MatchCollection matches = rg.Matches(xmlPozitii.InnerText);

            // convertirea pozitiilor
            double pozx = Convert.ToDouble(matches[0].Value);
            double pozy = Convert.ToDouble(matches[1].Value);
            Nod nod = new Nod(xmlNode["NAME"].InnerText,domeniu,(int)pozx,(int)pozy);

            //adaugarea nod in lista
            noduri.Add(nod);
        }
        return noduri;
}
```

4.3 Codul aferent desenării unui nod și a legăturii dintre două noduri din rețea

```
/// <summary>
/// Metoda responsabilă cu desenarea unui nod
/// </summary>
/// <param name="nod"></param>
private void DeseneazaNod(Nod nod)
{
        using (Graphics G = Graphics.FromImage(reteaPictureBox.Image))
        {
            Pen pen = new Pen(Color.Black, 1);
            //Creare textBox și setarea tuturor proprietăților
            TextBox textBox = new TextBox();
            textBox.Location = new Point(nod.PozitieX + _nodPadding, nod.PozitieY +
_nodPadding);
            textBox.Text = nod.Nume + "\r\n null";
            textBox.Multiline = true;
            textBox.Width = _nodWitdh - 2 * _nodPadding;
            textBox.Height = _nodHeight - 2 * _nodPadding;
            textBox.BackColor = Color.White;
            textBox.ForeColor = Color.Black;
            textBox.Visible = true;
            textBox.BorderStyle = BorderStyle.None;
            textBox.TextAlign = HorizontalAlignment.Center;
            textBox.Enabled = false;
            //adaugarea textBox-ului pe pictureBox
            reteaPictureBox.Controls.Add(textBox);
            //Adaugarea nodului și a textBox-ului în NoduriTextboxes Dictionary
            NoduriTextboxes.Add(nod, textBox);
            textBox.BringToFront();

            //Desenarea elipsei ce inconjoara nodul dupa pozitiile date
            G.DrawEllipse(pen, new Rectangle(nod.PozitieX, nod.PozitieY, _nodWitdh,
_nodHeight));
         }
            // Se dă refresh la pictureBox pentru adăugarea noilor elemente
            reteaPictureBox.Refresh();
}

/// <summary>
```

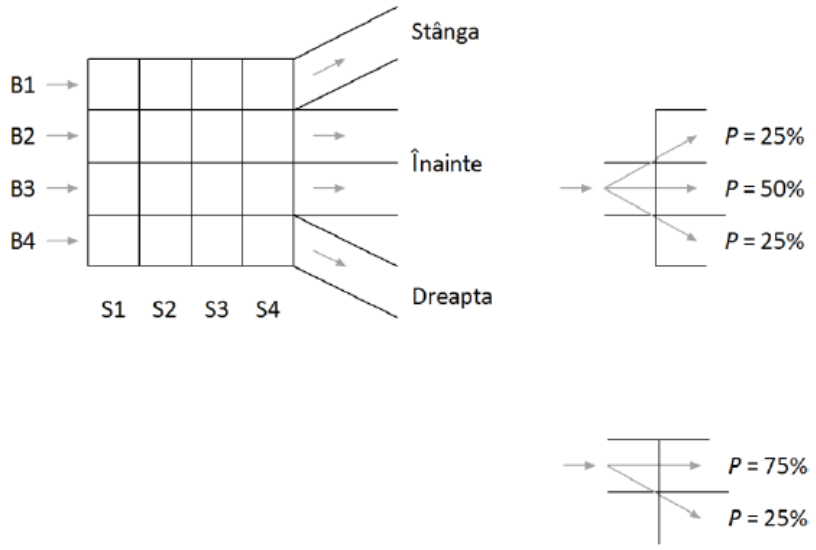
```
/// Metoda responsabilă cu desenarea legăturii dintre doua noduri
/// </summary>
/// <param name="nod"></param>
private void DeseneazaLegatura(Nod nod)
{
      using (Graphics G = Graphics.FromImage(reteaPictureBox.Image))
      {
            Pen pen = new Pen(Color.Black, 1);
            //Seteaza stilul pentru inceputul și finalul liniei ce va fi desenata cu
pen
            pen.StartCap = LineCap.Round;
            pen.EndCap = LineCap.ArrowAnchor;
            int startX=0, startY=0, finalX=0, finalY=0;

            //Pentru nodul primit ca parametru trebuie sa ii vad prima data lista de
parinti
            List<Nod> parinti = _retea.Parinti(nod);

            //Dupa sa ii parcurg lista de parinti si pentru fiecare nod parinte
calculez distanta intre nod si nodul primit ca parametru (adica nodul copil)
            foreach (Nod parinte in parinti)
            {
                  if (GenerareDirectie(parinte, nod) == "DREAPTA")
                  {
                        startX = parinte.PozitieX;
                        startY = parinte.PozitieY + _nodHeight / 2;
                        finalX = nod.PozitieX + _nodWitdh;
                        finalY = nod.PozitieY + _nodHeight / 2;
                  }
                  else if (GenerareDirectie(parinte, nod) == "STANGA")
                  {
                        startX = parinte.PozitieX + _nodWitdh;
                        startY = parinte.PozitieY + _nodHeight / 2;
                        finalX = nod.PozitieX;
                        finalY = nod.PozitieY + _nodHeight / 2;
                  }
                  else if(GenerareDirectie(parinte, nod) == "SUS")
                  {
                        startX = parinte.PozitieX + _nodWitdh / 2;
                        startY = parinte.PozitieY + _nodHeight;
                        finalX = nod.PozitieX + _nodWitdh / 2;
                        finalY = nod.PozitieY;
                  }
                  else if(GenerareDirectie(parinte, nod) == "JOS")
                  {
                        startX = parinte.PozitieX + _nodWitdh / 2;
                        startY = parinte.PozitieY;
                        finalX = nod.PozitieX + _nodWitdh / 2;
                        finalY = nod.PozitieY + _nodHeight;
                  }
                  else
                  {
                        MessageBox.Show("Eroare!", " Nodurile sunt foarte
apropiate!", MessageBoxButtons.OK);
                  }
                  // În funcție de pozițiile calculate mai sus se desenează linia
între cele două noduri
                  G.DrawLine(pen, startX, startY, finalX, finalY);
            }
        }
        // Se face refresh la pictureBox pentru adăugarea noilor elemente
        reteaPictureBox.Refresh();
}
```

Capitolul 5. Rezultate obținute în diferite situații

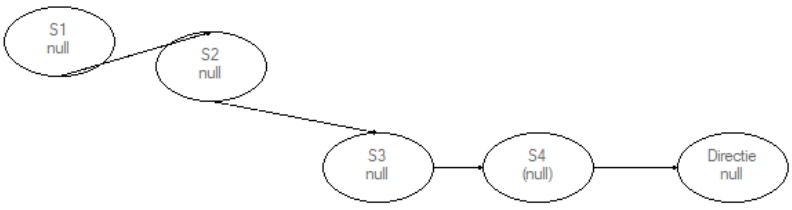
5.1 Aplicația de laborator

Aplicația de laborator studiată pune în evidență următoarea problemă:

"Strada incidentă în intersecție, cu sens unic, are 4 benzi de circulație (B1, B2, B3, B4). Până la intersecție sunt 4 sectoare de drum (S1, S2, S3, S4). Din fiecare sector, o mașină poate merge înainte, cu probabilitatea de 50% sau la dreapta, respectiv stânga, cu probabilitățile de 25%. De pe benzile laterale nu se poate ieși în afara drumului, prin urmare probabilitatea de a merge înainte este 75%. La capătul străzii, mașinile pot merge pe unul din cele 3 drumuri (Stânga, Înainte, Dreapta)." (Florin Leon - Inteligență Artificială, laboratorul 11)

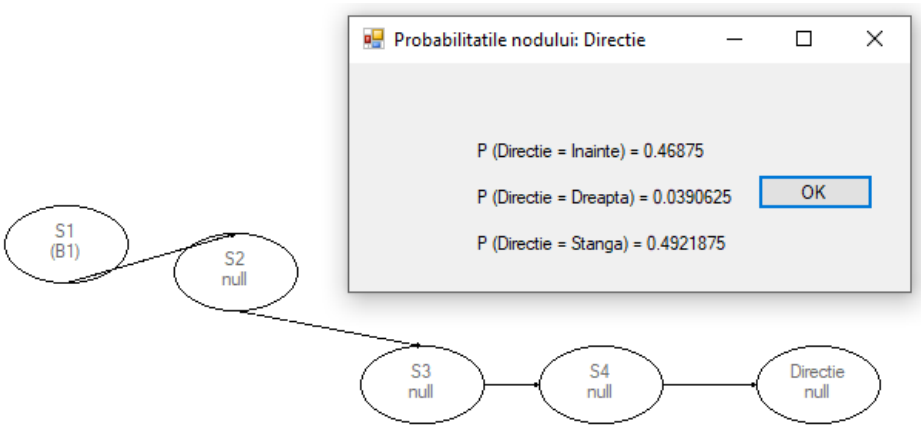


Modelarea problemei presupune următoarea rețea Bayesiană:

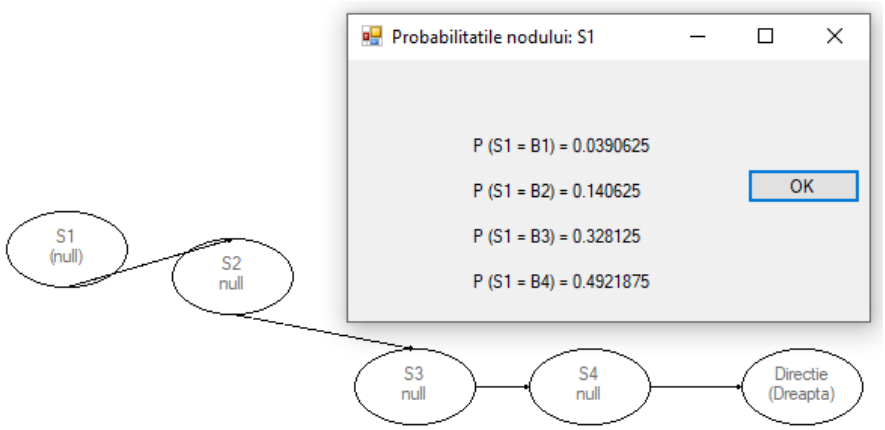


Cele 4 noduri de tip sector de drum pot avea stări din domeniul (B1, B2, B3, B4), iar nodul Direcție are stări în domeniul (Înainte, Stânga, Dreapta). Având observații clare, putem calcula în funcție de acestea, putem anticipa în ce direcție se va deplasa mașina în final.

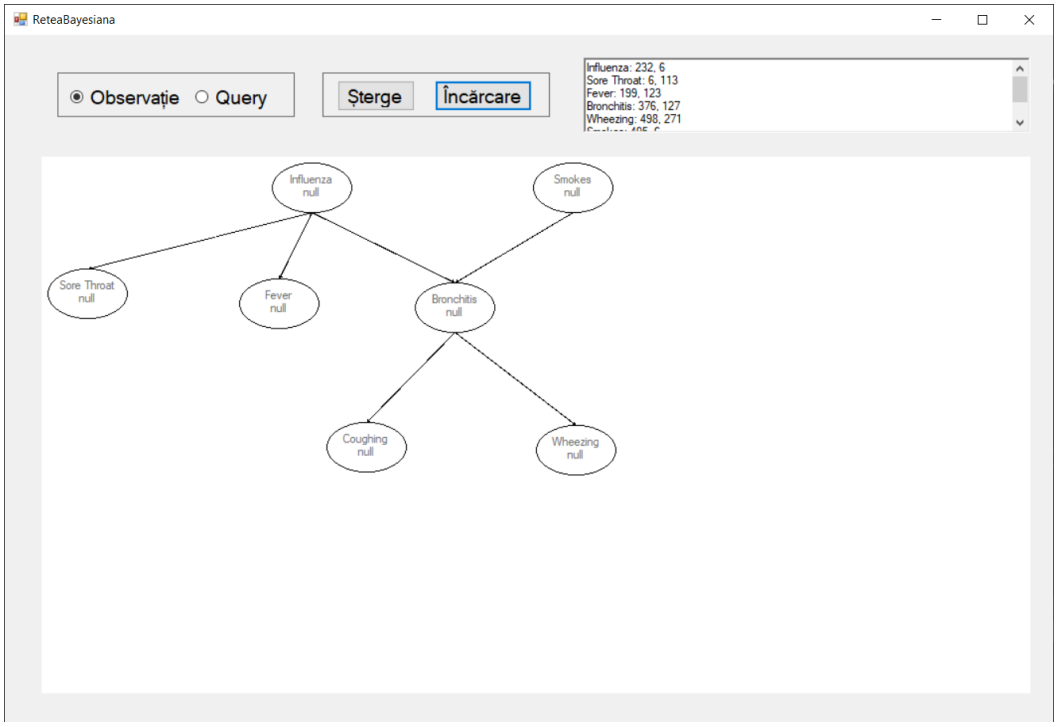
De exemplu, dacă știm la S1 se afla pe banda B1, putem calcula probabilitățile ca mașina să iasa pe șosea prin orice direcție.



O altă aplicație utilă ar putea fi cazul în care observăm efectul (direcția pe care a părăsit mașina banda) și să încercăm să anticipăm pe ce bandâ s-ar fi putut afla în starea S1.



5.2 Aplicația Smoker



În rețeaua din figura de mai sus este modelată următoarea problemă:

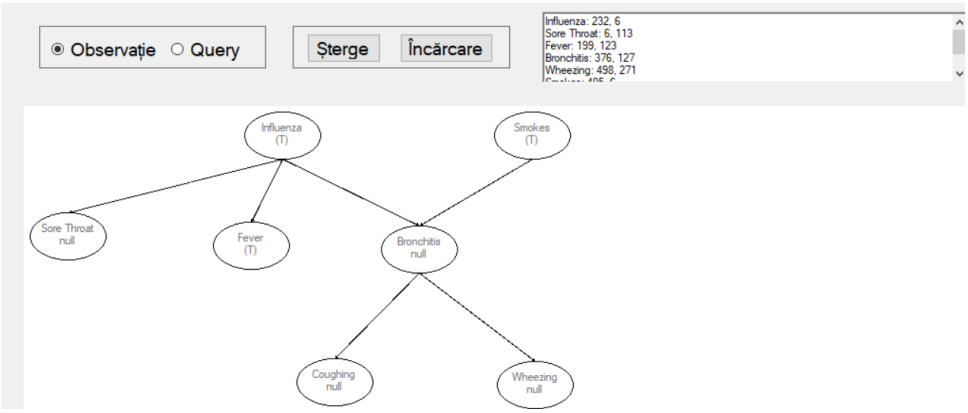
"Oamenii experimentează diferite simptome de boală printre care gâtul uscat, febră, tuse și respirație greoaie. Acestea pot fi simptome ale gripei sau bronșitei dar și fumatul are un rol în această ecuație. Se consideră că gripa are ca simptome gâtul uscat și febra, iar împreună cu fumatul pot cauza bronșită. De asemenea simptomele bronșitei sunt tusea și respirația greoaie. Se dorește să se afle probabilitatea apariției unui simptom sau a existenței uneia dintre boli prin modelarea evenimentelor prezentate mai sus printr-o rețea bayesiană."

Exemplu:

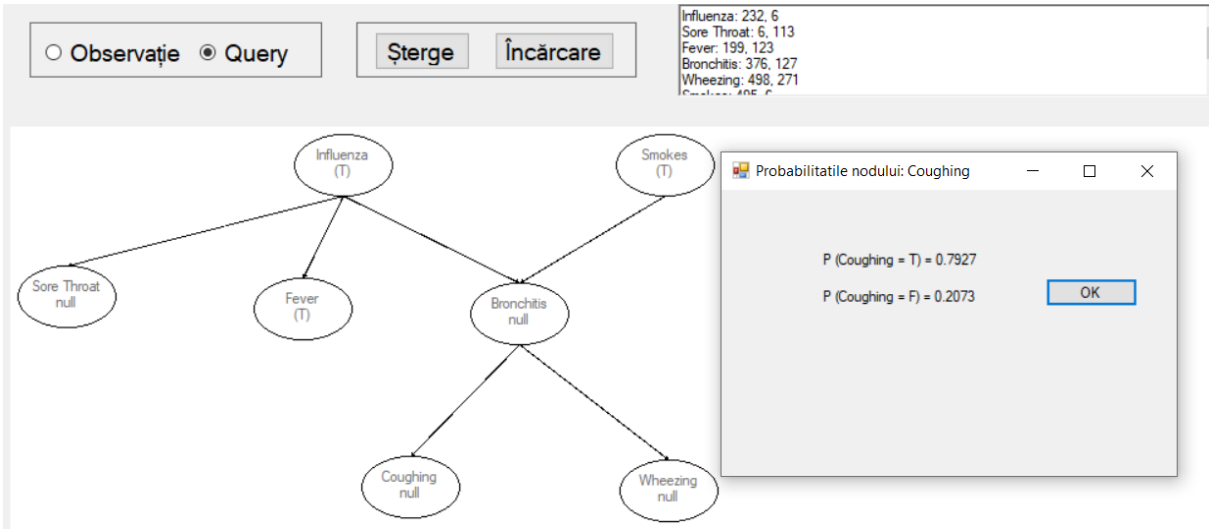
Considerăm că persoana respectivă are gripă,febră și fumează.

1. Dorim să aflăm care e probabilitatea să aibă tusea ca simptom:

   Pentru aceasta trebuie să facem observațiile nodurilor evidență ca în figura de mai jos.

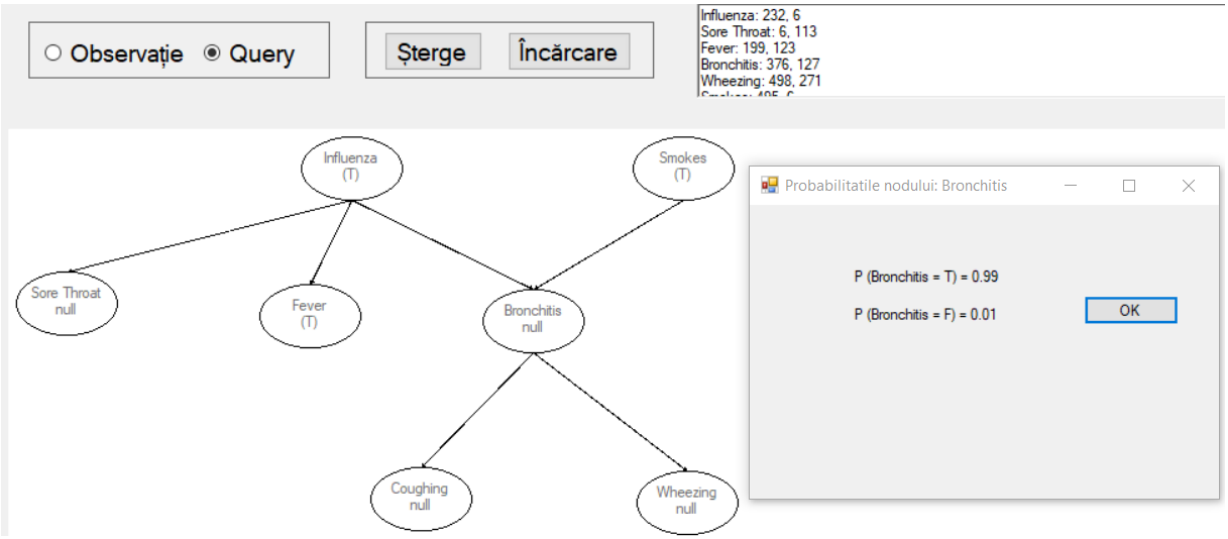


Acum trebuie interogat nodul “Couching” pentru a afla că probabilitatea de a tuși este 0.7927



2. Dorim să vedem care e probabilitatea să facă bronșită

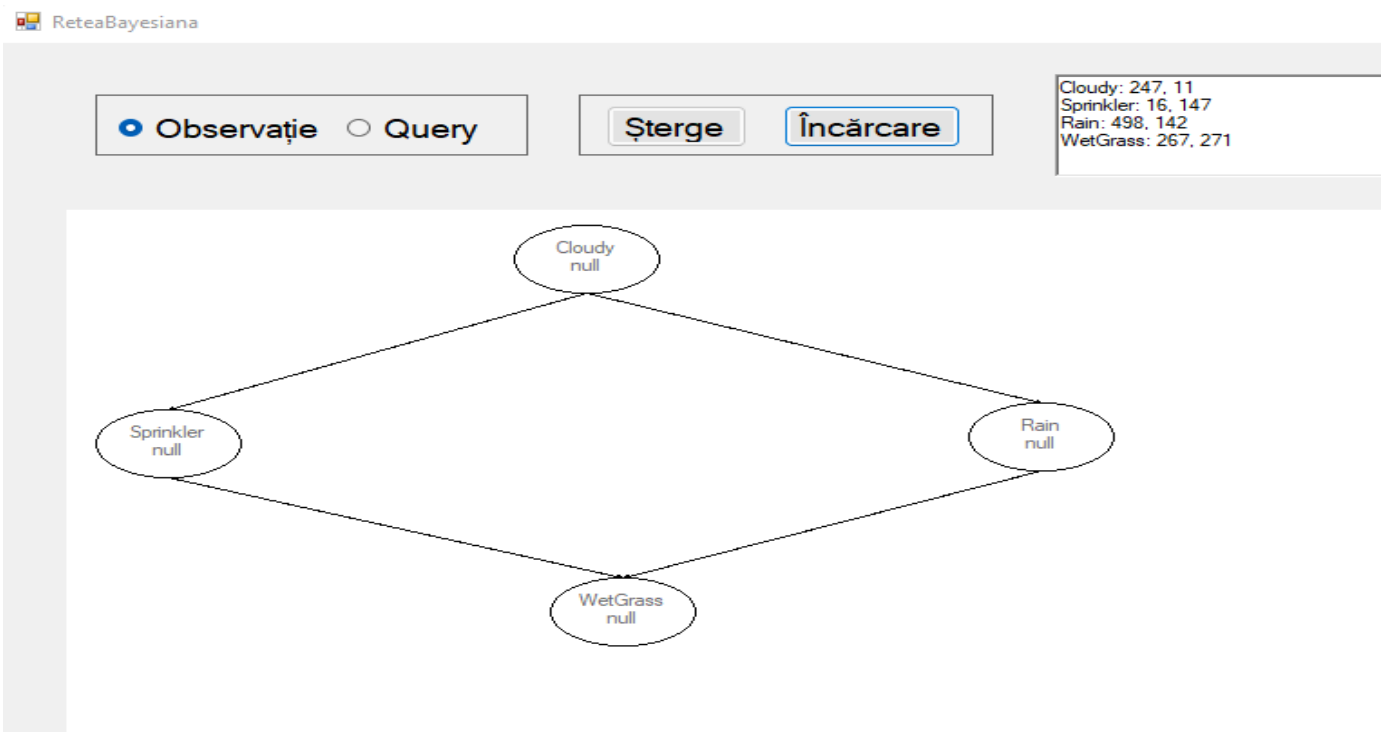
   Cu aceleași observații va trebui să interogăm nodul “Bronchitis” pentru a afla că probabilitatea de a face bronșită este de 99%.



Interpretând rezultatele se pot observa următoarele lucruri:

1. Deoarece în modelarea prin rețele bayesiene un nod este influențat doar de părinții săi, febra nu influențează în vreun fel apariția bronșitei sau a tusei.
2. Sănătatea este importantă și nu tot timpul ușor de întreținut. Astfel, deși nu putem tot timpul să ne ferim de gripă, ar fi bine să evităm fumatul în cazul în care nu vrem să facem bronșită.

5.3 Aplicația Sprinkler

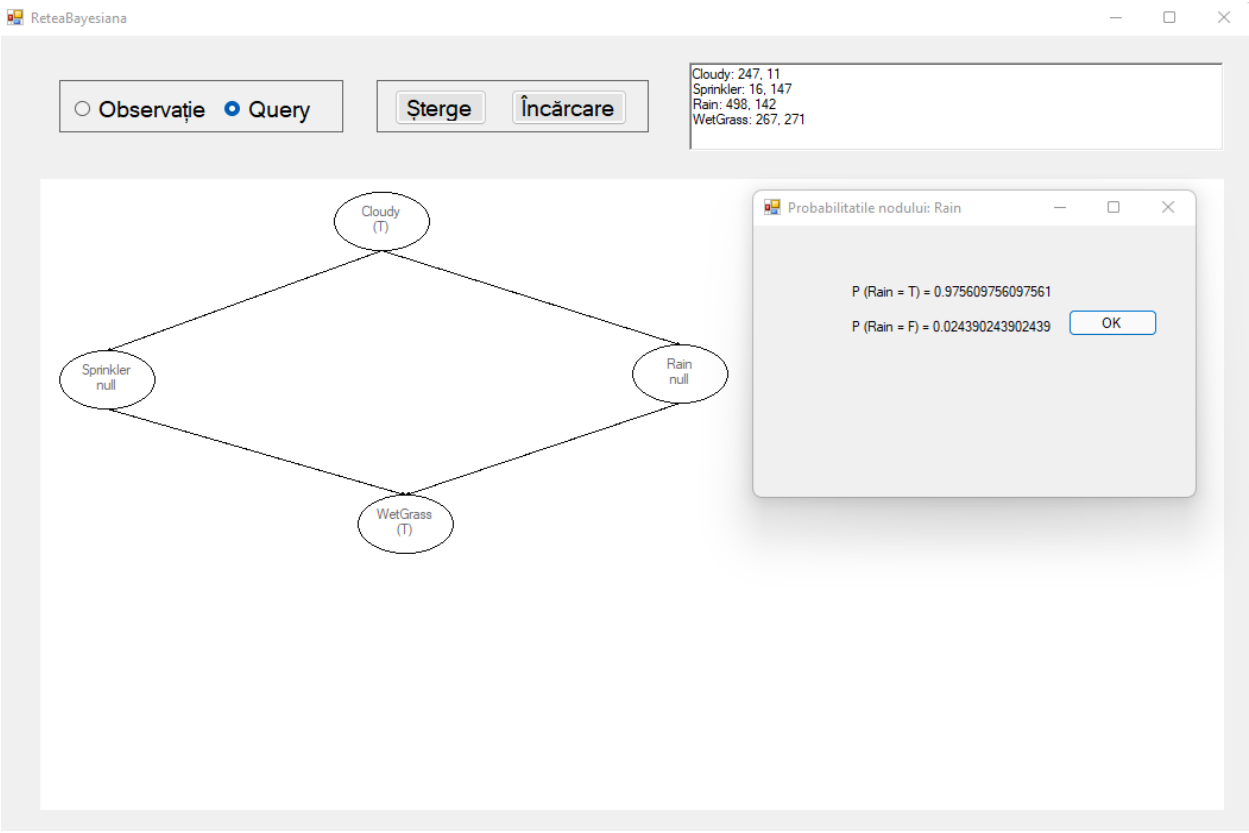


În rețeaua din figura de mai sus este modelată următoarea problemă:

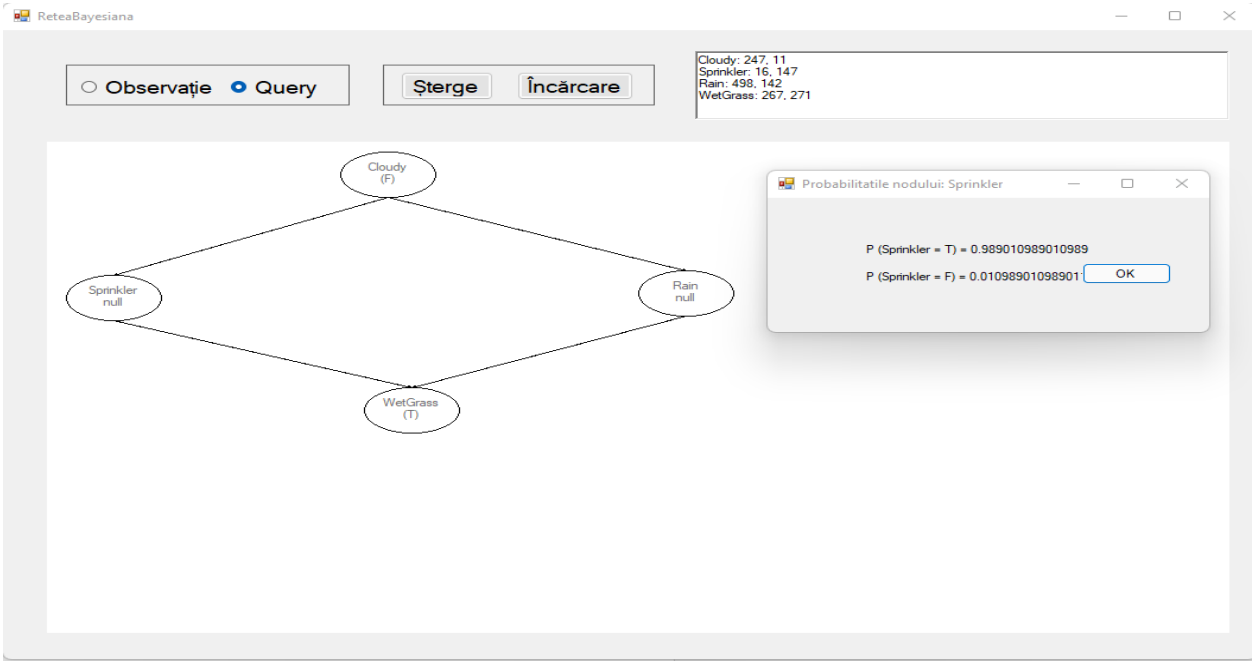
"În viața de zi cu zi ne întâlnim cu diferite situații probabilistice, printre care și problema de mai jos. Se consideră că dacă iarba este umedă atunci ea depinde de faptul că afară a plouat sau că s-a folosit o stropitoare pentru a o uda. Probabilitatea ca iarba să fie umedă este indirect condiționată și de faptul că afară este înnorat, având în vedere și informațiile referitoare la posibilitatea utilizării unei stropitori sau prin simplul fapt că a plouat. Se dorește să se afle care este probabilitatea ca iarba să fie udă în condițiile date prin modelarea evenimentelor prezentate mai sus printr-o rețea bayesiană".

Exemplu:

1. Considerăm că afară este înnorat și că iarba este udă, care este probabilitatea ca starea acesteia să fie din cauza ploii?



2. Considerăm că afară nu este înnorat și că iarba este udă, care este probabilitatea ca starea acesteia să fie din cauza stropitorii?



În urma analizei rezultatelor putem observa că în modelarea cu rețele bayesiene un nod poate fi influențat doar de nodurile părinților săi, așa cum este în exemplul de mai sus: nodul WetGrass este influențat de nodurile Sprinkler și Rain; el nu este influențat de nodul Cloudy.

Capitolul 6. Rolul fiecărui membru

Proiectul a fost modularizat în trei componete diferite care au fost împărțite în mod egal între membrii echipei:

1) Partea de interfață a fost realizată de către Mihalache Ecaterina-Nicoleta.
2) Partea de deserializare dintr-un fișiere xml a unei rețele Bayesiene (ReteaFactory) a fost realizată de către Petrișor Iosif-Marcelin.
3) Partea de structuralizare în obiecte a unei rețele Bayesiene și algoritmul de rezolvare a unui query au fost realizate de către Batalan Vlad.

Procesul de realizare a documentației a fost împărțit în mod egal la toți cei trei membrii ai echipei.

Extra: Aplicația conține un modul de testare atât pentru componenta Rețea, realizată de către Batalan Vlad, cât și unul pentru componenta de citire a unei rețele din xml realizat de către Petrișor Iosif-Marcelin.

Capitolul 7. Concluzii

În concluzie, aplicația ce modelează o rețea bayesiană are ca scop ușurarea procesului de calcul al probabilităților condiționate, prin oferirea unei variante mai simple de calcul și de vizualizare a rezultatelor.

Capitolul 8. Bibliografie

1. XmlNodeList Class
   https://docs.microsoft.com/en-us/dotnet/api/system.xml.xmlnodelist?view=net-6.0
2. XmlDocument Class
   https://docs.microsoft.com/en-us/dotnet/api/system.xml.xmldocument?view=net-6.0
3. XmlNode Class
   https://docs.microsoft.com/en-us/dotnet/api/system.xml.xmlnode?view=net-6.0
4. XmlNode.SelectNodes Method
   https://docs.microsoft.com/en-us/dotnet/api/system.xml.xmlnode.selectnodes?view=net-6.0
5. Regex
   https://regex101.com/
6. Curs Inteligență artificiala - Florin Leon
   http://florinleon.byethost24.com/curs_ia.html
7. Documentație oficială Microsoft pentru Windows Forms
   https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms?view=windowsdesktop-6.0