

UNIVERSITATEA TEHNICĂ „GHEORGHE ASACHI” IAȘI  
FACULTATEA AUTOMATICĂ ȘI CALCULATOARE  
DOMENIUL: CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI  
PROFIL DE STUDIU: TEHNOLOGIA INFORMAȚIEI

# PROIECT INTELIGENȚĂ ARTIFICIALĂ

**Rezolvarea problemei de optimizare a unei mașini cu vectori suport cu  
ajutorul unui algoritm evolutiv**

**Studenti:**

Lungu Ștefania-Paraschiva  
Poleac Alexandra-Cătălina

**An universitar 2022-2023**

## CUPRINS

<b>Capitolul 1. Descrierea problemei considerate</b>	<b>3</b>
<b>Capitolul 2. Aspecte teoretice privind algoritmul</b>	<b>4</b>
<b>Capitolul 3. Modalitatea de rezolvare</b>	<b>5</b>
1. Cuantificarea unui set de date	5
2. Funcția de fitness	7
3. Determinarea soluției problemei – determinarea parametrilor alpha	7
4. Funcția de calcul pentru nucleul polinomial de grad II	8
5. Determinarea claselor	8
<b>Capitolul 5. Concluzii</b>	<b>10</b>
<b>Capitolul 6. Bibliografie</b>	<b>10</b>
<b>Capitolul 7. Rolul fiecărui membru al echipei</b>	<b>11</b>

## Capitolul 1. Descrierea problemei considerate

Problema de bază a proiectului este clasificarea familiilor dintr-o grădiniță care se împarte în 4 categorii: *not\_recommend*, *recommend*, *very\_recommend*, *priority*. Fiecare instanță conține un număr de atribute prin intermediul cărora se poate face clasificarea:

- parents: ocupația părinților
- has\_nurs: grădinița copilului
- form: forma familiei
- children: numărul de copii
- housing: starea locuinței
- finance: starea financiară
- social: statutul social
- health: starea de sănătate

Setul de date utilizat este cel din UCI Repository:

<https://archive.ics.uci.edu/ml/datasets/Nursery?fbclid=IwAR0kX5DJ-OHfStIAkplc4kBjINTo4ZnBfAKDLKjX-5DKwrF9CiWdR0oEb-s>

Soluția își propun utilizarea unui algoritm evolutiv cu ajutorul căruia să se determine parametrii SVM-ului ( $\alpha$ ,  $bias$  și  $w$ ) necesari în procesul de clasificare.

## Capitolul 2. Aspecte teoretice privind algoritmul

Problema duală a SVM presupune maximizarea a  $W(\alpha)$  cu ajutorul algoritmului evolutiv. Trebuie să determinăm dreapta care maximizează marginea ce desparte cele două clase de obiecte, adică vectorii suport. Vectorii suport sunt o mulțime de puncte într-un spațiu  $n$  dimensional care sprijină “marginea”.

Scopul acestui algoritm este de a determina valorile suport care maximizează marginea.

Vom folosi algoritmul evolutiv varianta standard prezentată la curs și la laborator.

Problema duală a SVM:

$$\max_{\alpha_i} \left( \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \right)$$

cu următoarele constrângeri:

$$\alpha_i \geq 0, i = 1..N$$

$$\sum_{i=1}^N \alpha_i y_i = 0.$$

Am folosit nucleul polinomial de gradul 2:

$$\blacksquare K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z} + 1)^2$$

Genele din cromozomi sunt valorile  $\alpha$ -urilor, iar funcția de fitness este următoarea:

$$W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle$$

Apoi din acești  $\alpha$  vom calcula parametrii  $\mathbf{w}$  și  $\mathbf{b}$ :

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y^{(i)} \mathbf{x}^{(i)}$$

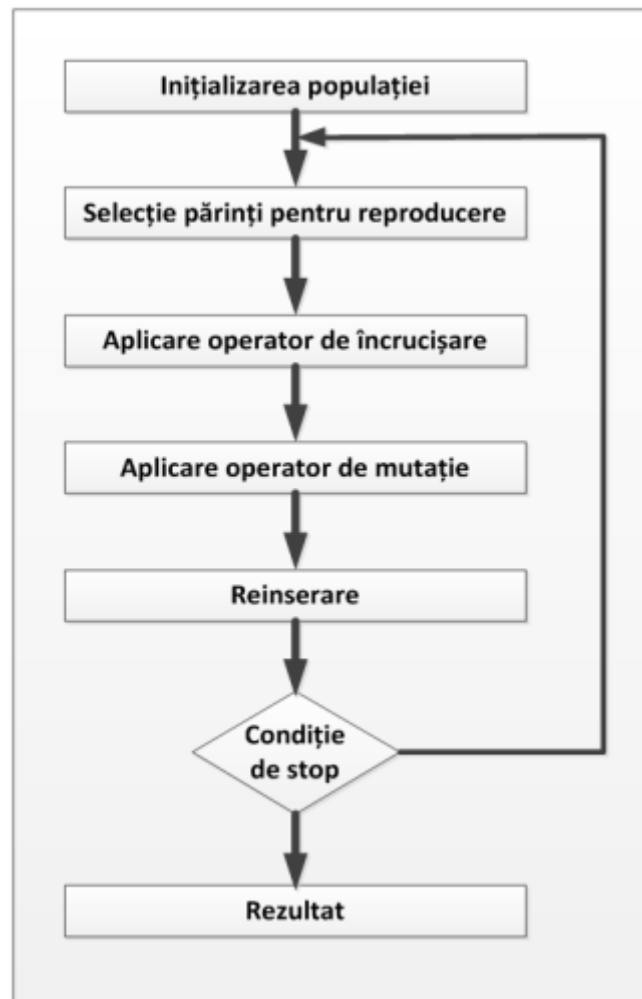
$$b = \frac{1}{|S|} \sum_{s \in S} \left( y_s - \sum_{t \in S} \alpha_t y_t (\mathbf{x}_t \cdot \mathbf{x}_s) \right)$$

unde  $S$  este mulțimea vectorilor suport iar  $|S|$  este numărul lor

De asemenea, am folosit ca metoda de optimizare algoritmul evolutiv asupra populației de date extrase din fișier, asupra cărora se aplică transformările specifice evoluției naturale:

- Selecția prin care determinăm părinții ce se vor reproduce pentru a forma următoarea generație. Se vor alege apoi indivizii cei mai adaptați (care se aproprie cel mai mult de soluția problemei).
- Încrucișarea prin care se generează un copil pornind de la doi părinți.
- Mutația ce se folosește pentru a asigura diversitatea populației. Asupra noilor copii generați se aplica transformări aleatorii, astfel se permite apariția unor noi trăsături care nu ar fi apărut în cadrul populației prin selecție și încrucișare.

Structura algoritmului evolutiv este:



# Capitolul 3. Modalitatea de rezolvare

## 1. Cuantificarea unui set de date

```
internal class DataManipulation
{
    public string[] lines = File.ReadAllLines(@"C:\Users\STEFANIA\Desktop\Facultate\IA\PROIECT 2023\nursery.data");
    public int[,] valuesClasified;
    public int[,] valuesNonRecommend = new int[37, 8];
    public int[,] valuesRecommend = new int[37, 8];
    public int[,] valuesVeryRecommend = new int[22, 8];
    public int[,] valuesPriority = new int[46, 8];
    int NonRecomLines = 0;
    int RecomLines = 0;
    int VeryRecomLines = 0;
    int PriorityLines = 0;
    int noInstances;
    1 reference
    public void fileRead()
    {
        //string[] lines = File.ReadAllLines(@"C:\Users\STEFANIA\Desktop\Facultate\IA\PROIECT 2023\nursery.data");

        valuesClasified = new int[lines.Length, 9];
        //int[,] valuesNonRecommend = new int[37, 8];
        //int[,] valuesRecommend = new int[37, 8];
        //int[,] valuesVeryRecommend = new int[22, 8];
        //int[,] valuesPriority = new int[46, 8];

        noInstances = lines.Length;
        for (int i = 0; i < lines.Length; i++)
        {
            string[] values = lines[i].Split(',');

            switch (values[0]) // parents = ocupatia parintilor
            {
                case "usual":
                    valuesClasified[i, 0] = 1;
                    break;
                case "pretentious":
                    valuesClasified[i, 0] = 2;
                    break;
                case "great_pret":
                    valuesClasified[i, 0] = 3;
                    break;
                default:
                    break;
            }
        }
    }
}
```

```

// Clasificare date in mai multi vectori in functie de ultimul parametru (very-recom, recom, non-recom, priority)
for (int k = 0; k < lines.Length - 1; k++)
{
    switch (valuesClassified[k, 8])
    {
        case -1:
            for (int j = 0; j < 7; j++)
            {
                valuesNonRecommend[NonRecomLines, j] = valuesClassified[k, j];
                //Console.WriteLine("NR " + valuesNonRecommend[k, j] + " ");
            }
            NonRecomLines++;
            break;
        case 0:
            for (int j = 0; j < 7; j++)
            {
                valuesRecommend[RecomLines, j] = valuesClassified[k, j];
                //Console.WriteLine("R " + valuesRecommend[k, j] + " ");
            }
            RecomLines++;
            break;
        case 1:
            for (int j = 0; j < 7; j++)
            {
                valuesVeryRecommend[VeryRecomLines, j] = valuesClassified[k, j];
                //Console.WriteLine("VR " + valuesVeryRecommend[k, j] + " ");
            }
            VeryRecomLines++;
            break;
        case 2:
            for (int j = 0; j < 7; j++)
            {
                valuesPriority[PriorityLines, j] = valuesClassified[k, j];
                //Console.WriteLine("P " + valuesPriority[k, j] + " ");
            }
            PriorityLines++;
            break;
        default:
            break;
    } //end switch
} //end for

```

```

using (StreamWriter sw = new StreamWriter(@"C:\Users\STEFANIA\Desktop\Facultate\IA\PROIECT 2023\outDataNonRecomm.data"))
{
    sw.WriteLine("Populatia de date din categoria non-recommended.");
    sw.WriteLine("Numarul de populatii non-recomandate este " + NonRecomLines);
    for (int t = 0; t < valuesNonRecommend.GetLength(0); t++)
    {
        for (int s = 0; s < valuesNonRecommend.GetLength(1); s++)
        {
            sw.Write(valuesNonRecommend[t, s] + " ");
        }
        sw.WriteLine();
    }
} //end using

using (StreamWriter sw = new StreamWriter(@"C:\Users\STEFANIA\Desktop\Facultate\IA\PROIECT 2023\outDataRecomm.data"))
{
    sw.WriteLine("Populatia de date din categoria recommended.");
    sw.WriteLine("Numarul de populatii recomandate este " + NonRecomLines);
    for (int t = 0; t < valuesRecommend.GetLength(0); t++)
    {
        for (int s = 0; s < valuesRecommend.GetLength(1); s++)
        {
            sw.Write(valuesRecommend[t, s] + " ");
        }
        sw.WriteLine();
    }
} //end using

```

## 2. Funcția de fitness

```
3 references
public void ComputeFitness(Chromosome c, int[,] inputValues)
{
    //folosim nucleul polinomial de grad 2 - Curs 12 pagina 50
    int[] xi = new int[8]; // 8 attribute (fara ultimul)
    int[] xj = new int[8]; // vectori in care vom avea valorile din inputValues pentru a le transmite ca paramtru in functia calculata de Kernel

    double sumaAlpha = 0; // prima suma
    double s2 = 0;
    Kernel kernel = new Kernel();

    for (int i = 0; i < c.NoGenes; i++)
    {
        sumaAlpha += c.Genes[i]; //prima suma pentru alpha[i]
    }
    for (int i = 0; i < c.NoGenes; i++)
    {
        for (int j = 0; j < c.NoGenes; j++)
        {
            for (int t = 0; t < 8; t++)
            {
                xi[t] = inputValues[i, t]; // primul parametru din functia PolinomialKernel trebuie sa fie x[i] conform formulei matematice
                // x[i] = x = inputValues[i, 8]. Deoarece parametru functiei este de tip vector, am copiat valorile intr-un vector
                xj[t] = inputValues[j, t]; // al doilea parametru din functia PolinomialKernel trebuie sa fie x[j] conform formulei matematice
                // x[j] = inputValues[j, 8]. Deoarece parametru functiei este de tip vector, am copiat valorile intr-un vector x
            }
            //c.Genes = alpha, inputValues = y conform formulei matematice din Cursul 12 pag 50
            s2 += c.Genes[i] * c.Genes[j] * inputValues[i, 8] * inputValues[j, 8] * kernel.PolinomialKernel(xi, xj);
        }
    }
    c.Fitness = sumaAlpha - (1 / 2) * s2;
}
```

## 3. Determinarea soluției problemei

```
public Chromosome Solve(IOptimizationProblem p, int populationSize, int maxGenerations, double crossoverRate, double mutationRate, int[,] inputValues)
{
    // throw new Exception("Aceasta metoda trebuie completata");

    Chromosome[] population = new Chromosome[populationSize];
    for (int i = 0; i < population.Length; i++)
    {
        population[i] = p.MakeChromosome();
        p.ComputeFitness(population[i], inputValues);
    }

    for (int gen = 0; gen < maxGenerations; gen++)
    {
        Chromosome[] newPopulation = new Chromosome[populationSize];
        newPopulation[0] = Selection.GetBest(population); // elitism

        for (int i = 1; i < populationSize; i++)
        {
            // selectare 2 parinti: Selection.Tournament
            Chromosome mama = Selection.Tournament(population);
            Chromosome tata = Selection.Tournament(population);
            while (mama == tata)
                mama = Selection.Tournament(population);

            // generarea unui copil prin aplicare crossover: Crossover.Arithmetic
            Chromosome copil = Crossover.Arithmetic(mama, tata, crossoverRate);

            // aplicare mutatie asupra copilului: Mutation.Reset
            Mutation.Reset(copil, crossoverRate);

            // calculare fitness pentru copil: ComputeFitness din problema p
            p.ComputeFitness(copil, inputValues);

            // introducere copil in newPopulation
            newPopulation[i] = copil;
        }

        for (int i = 0; i < populationSize; i++)
            population[i] = newPopulation[i];
    }

    return Selection.GetBest(population);
}
```



## 4. Funcția de calcul pentru nucleul polinomial de grad II

```
2 references
public class Kernel
{
    1 reference
    public int PolinomialKernel(int[] xi, int[] xj) //Folosim nucleul polinomial de grad 2
    {
        int k = 0;
        for(int i = 0; i < xi.Length; i++)
        {
            k += xi[i] * xj[i];
        }
        return (k + 1) * (k + 1);
    }
}
```

## 5. Determinarea parametrilor SVM-ului - alfa, w (weights) și bias

```
EvolutionaryAlgorithm ea = new EvolutionaryAlgorithm();
Chromosome solution = ea.Solve(new Equation(), 120, 100, 0.95, 0.2, valuesClassified);

int nrInstante = 120;
double[] alfa = new double[nrInstante]; //nr de instante = 120
Console.WriteLine("Coeficientii alfa sunt ");
for(int i = 0; i < nrInstante; i++)
{
    alfa[i] = solution.Genes[i];
    Console.WriteLine("Pe linia " + i + " avem coeficientul " + alfa[i] + " ");
}

Console.WriteLine("-----");
double w = 0;
for (int i = 0; i < nrInstante; i++)
{
    w += alfa[i] * valuesClassified[i, 8];
}
Console.WriteLine("Coeficientul w este " + w);
Console.WriteLine("-----");
int[] xi = new int[8]; // 8 atribute (fara ultimul)
int[] xj = new int[8]; // vectori in care vom avea valorile din inputValues pentru a le transmite ca paramtru in functia calculata de Kernel

double bias = 0;
double s2 = 0;
Kernel kernel = new Kernel();

for (int i = 0; i < nrInstante; i++)
{
    for (int j = 0; j < nrInstante; j++)
    {
        for (int t = 0; t < 8; t++)
        {
            xi[t] = valuesClassified[i, t]; // primul paramtru din functia PolinomialKernel trebuie sa fie x[i] conform formulei matematice
            // x[i] = x = inputValues[i, 8]. Deoarece paramtru functiei este de tip vector, am copiat valorile intr-un vector x
            xj[t] = valuesClassified[j, t]; // al doilea paramtru din functia PolinomialKernel trebuie sa fie x[j] conform formulei matematice
            // x[j] = inputValues[j, 8]. Deoarece paramtru functiei este de tip vector, am copiat valorile intr-un vector x
        }
        //c.Genes = alpha, inputValues = y conform formulei matematice din Cursul 12 pag 50
        s2 += valuesClassified[j, 8] * alfa[j] * kernel.PolinomialKernel(xi, xj);
    }
    bias += valuesClassified[i, 8] - s2;
}
bias /= nrInstante;
Console.WriteLine("Bias este " + bias);
```

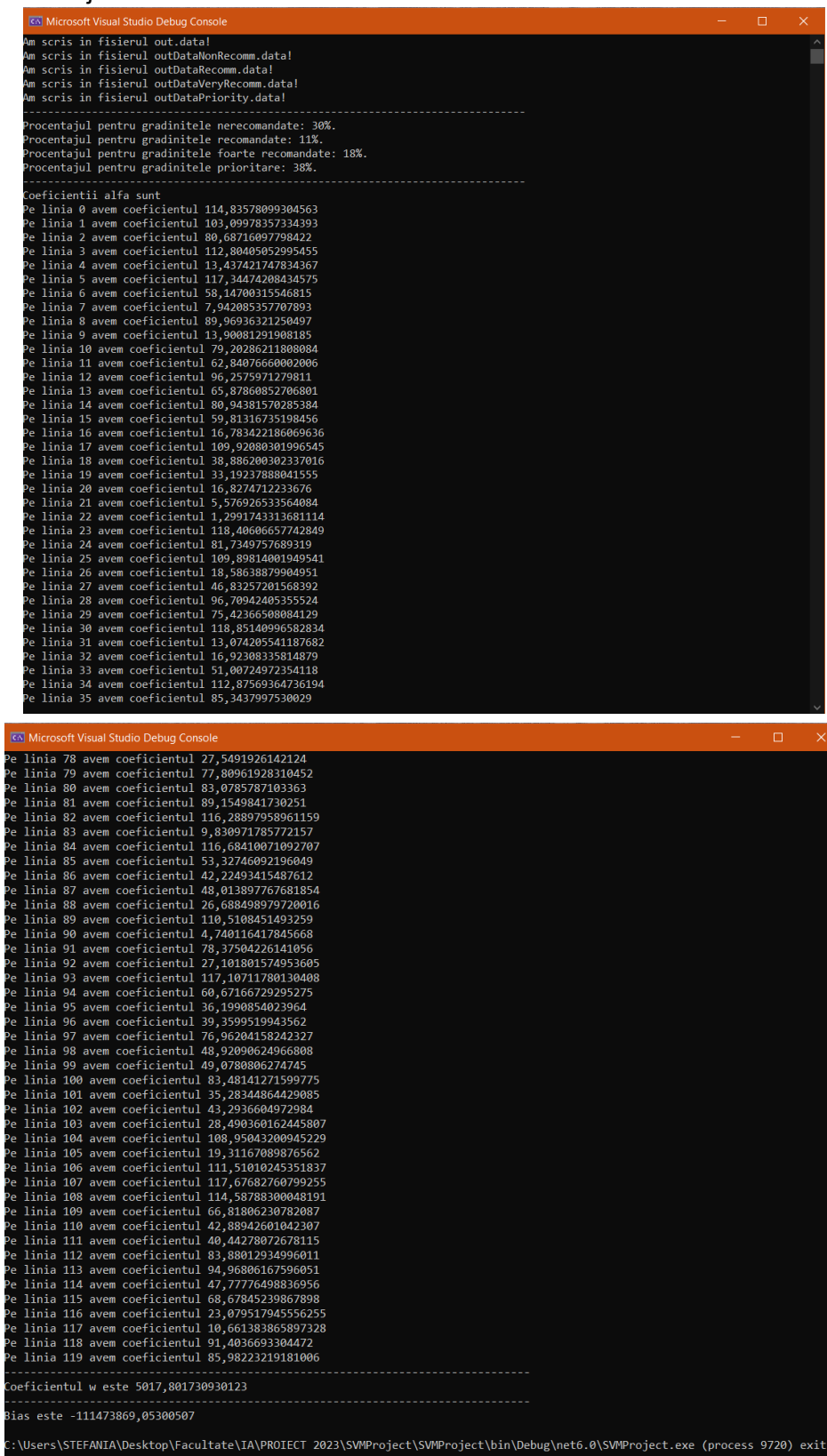
## 6. Determinarea sub-claselor / împărțirea vectorului pe categorii

```
int NonRecomLines = 0;
int RecomLines = 0;
int VeryRecomLines = 0;
int PriorityLines = 0;

for (int k = 0; k < lines.Length - 1; k++)
{
    switch (valuesClasified[k, 8])
    {
        case -1:
            for (int j = 0; j < 7; j++)
            {
                valuesNonRecommend[NonRecomLines, j] = valuesClasified[k, j];
                //Console.WriteLine("NR " + valuesNonRecommend[k, j] + " ");
            }
            NonRecomLines++;
            break;
        case 0:
            for (int j = 0; j < 7; j++)
            {
                valuesRecommend[RecomLines, j] = valuesClasified[k, j];
                //Console.WriteLine("R " + valuesRecommend[k, j] + " ");
            }
            RecomLines++;
            break;
        case 1:
            for (int j = 0; j < 7; j++)
            {
                valuesVeryRecommend[VeryRecomLines, j] = valuesClasified[k, j];
                //Console.WriteLine("VR " + valuesVeryRecommend[k, j] + " ");
            }
            VeryRecomLines++;
            break;
    }
}
```

## Capitolul 4. Rezultatele obținute prin rularea programului

Rezultatele obținute în urma rulării script-ului de clasificare se pot observa în imaginea de mai jos:



```
Microsoft Visual Studio Debug Console
Am scris in fisierul out.data!
Am scris in fisierul outDataNonRecomm.data!
Am scris in fisierul outDataRecomm.data!
Am scris in fisierul outDataVeryRecomm.data!
Am scris in fisierul outDataPriority.data!

-----
Procentajul pentru gradinitile nerecomandate: 30%.
Procentajul pentru gradinitile recomandate: 11%.
Procentajul pentru gradinitile foarte recomandate: 18%.
Procentajul pentru gradinitile prioritare: 38%.
-----

Coeficientii alfa sunt
Pe linia 0 avem coeficientul 114,83578099304563
Pe linia 1 avem coeficientul 103,09978357334393
Pe linia 2 avem coeficientul 80,68716097798422
Pe linia 3 avem coeficientul 112,80405052995455
Pe linia 4 avem coeficientul 13,437421747834367
Pe linia 5 avem coeficientul 117,34474208434575
Pe linia 6 avem coeficientul 58,14700315546815
Pe linia 7 avem coeficientul 7,942085357707893
Pe linia 8 avem coeficientul 89,96936321250497
Pe linia 9 avem coeficientul 13,90081291908185
Pe linia 10 avem coeficientul 79,20286211808084
Pe linia 11 avem coeficientul 62,84076660002006
Pe linia 12 avem coeficientul 96,2575971279811
Pe linia 13 avem coeficientul 65,87860852706801
Pe linia 14 avem coeficientul 80,94381570285384
Pe linia 15 avem coeficientul 59,81316735198456
Pe linia 16 avem coeficientul 16,783422186069636
Pe linia 17 avem coeficientul 109,92080301996545
Pe linia 18 avem coeficientul 38,886200302337016
Pe linia 19 avem coeficientul 33,19237888041555
Pe linia 20 avem coeficientul 16,8274712233676
Pe linia 21 avem coeficientul 5,576926535564004
Pe linia 22 avem coeficientul 1,9991743313681114
Pe linia 23 avem coeficientul 118,40606657742849
Pe linia 24 avem coeficientul 81,7249757680319
Pe linia 25 avem coeficientul 109,89814001049541
Pe linia 26 avem coeficientul 18,586388799004951
Pe linia 27 avem coeficientul 46,83257201568392
Pe linia 28 avem coeficientul 96,70942405355524
Pe linia 29 avem coeficientul 75,42366580804129
Pe linia 30 avem coeficientul 118,85140996582834
Pe linia 31 avem coeficientul 13,074205541187682
Pe linia 32 avem coeficientul 16,92308335814879
Pe linia 33 avem coeficientul 51,00724972354118
Pe linia 34 avem coeficientul 112,87569364736194
Pe linia 35 avem coeficientul 85,3437997530029

Pe linia 78 avem coeficientul 27,5491926142124
Pe linia 79 avem coeficientul 77,80961928310452
Pe linia 80 avem coeficientul 83,0785787103363
Pe linia 81 avem coeficientul 89,1549841730251
Pe linia 82 avem coeficientul 116,28897958961159
Pe linia 83 avem coeficientul 9,830971785772157
Pe linia 84 avem coeficientul 116,68410071092707
Pe linia 85 avem coeficientul 53,32746092196049
Pe linia 86 avem coeficientul 42,22493415487612
Pe linia 87 avem coeficientul 48,013897767681854
Pe linia 88 avem coeficientul 26,688498979720016
Pe linia 89 avem coeficientul 110,5108451493259
Pe linia 90 avem coeficientul 4,740116417845668
Pe linia 91 avem coeficientul 78,37504226141056
Pe linia 92 avem coeficientul 27,101801574953605
Pe linia 93 avem coeficientul 117,10711780130408
Pe linia 94 avem coeficientul 60,67166729295275
Pe linia 95 avem coeficientul 36,1990854023964
Pe linia 96 avem coeficientul 39,3599519943562
Pe linia 97 avem coeficientul 76,96204158242327
Pe linia 98 avem coeficientul 48,92090624966808
Pe linia 99 avem coeficientul 49,0780806274745
Pe linia 100 avem coeficientul 83,48141271599775
Pe linia 101 avem coeficientul 35,28344864429085
Pe linia 102 avem coeficientul 43,2936604972984
Pe linia 103 avem coeficientul 28,490360162445807
Pe linia 104 avem coeficientul 108,95043200945229
Pe linia 105 avem coeficientul 19,31167089876562
Pe linia 106 avem coeficientul 111,51010245351837
Pe linia 107 avem coeficientul 117,67682760799255
Pe linia 108 avem coeficientul 114,58788300048191
Pe linia 109 avem coeficientul 66,81806230782087
Pe linia 110 avem coeficientul 42,88942601042307
Pe linia 111 avem coeficientul 40,44278072678115
Pe linia 112 avem coeficientul 83,88012934996011
Pe linia 113 avem coeficientul 94,96806167596051
Pe linia 114 avem coeficientul 47,7776498836956
Pe linia 115 avem coeficientul 68,67845239867898
Pe linia 116 avem coeficientul 23,079517945556255
Pe linia 117 avem coeficientul 10,661383865897328
Pe linia 118 avem coeficientul 91,4036693304472
Pe linia 119 avem coeficientul 85,98223219181006

-----
Coeficientul w este 5017,801730930123
Bias este -111473869,05300507

C:\Users\STEFANIA\Desktop\Facultate\IA\PROIECT_2023\SVMProject\SVMProject\bin\Debug\net6.0\SVMProject.exe (process 9720) exit
```

## Fişierele generate:

### - outDataNonRecomm.data

```
Populatia de date din categoria non-recommended.  
Numarul de populatii non-recomandate este 37  
1 1 1 1 1 1 0 0  
1 1 1 1 1 2 0 0  
1 1 1 1 1 2 3 0  
2 1 1 3 1 1 2 0  
1 1 1 1 2 1 0 0  
1 1 1 1 2 1 2 0  
1 1 1 1 2 1 3 0  
1 1 1 1 2 2 0 0  
3 4 1 4 3 1 3 0  
1 1 1 2 1 1 0 0  
1 1 1 2 1 1 2 0  
1 1 1 2 1 1 3 0  
1 1 1 2 1 2 0 0  
1 1 1 2 1 2 2 0  
1 1 1 2 2 1 3 0  
2 1 2 2 2 1 3 0  
1 1 1 2 2 2 0 0  
2 1 4 4 3 2 3 0  
2 2 4 4 3 2 3 0  
1 1 1 2 2 2 2 0  
1 1 1 2 3 2 3 0  
2 3 3 3 2 1 0 0  
1 1 1 3 1 1 0 0  
2 5 4 4 3 2 3 0  
1 1 1 4 1 1 0 0  
1 1 2 1 1 1 0 0  
1 1 2 2 1 1 0 0  
3 4 2 3 1 1 2 0  
1 1 2 2 2 1 2 0  
3 4 2 4 2 2 3 0  
2 2 1 1 3 1 0 0  
2 2 1 4 2 1 2 0  
1 1 3 2 3 2 3 0  
1 1 4 4 3 2 3 0  
2 2 1 4 3 1 2 0  
3 4 4 3 3 1 3 0  
3 5 4 4 3 2 3 0
```

### - outDataPriority.data

```
Populatia de date din categoria priority.  
Numarul de populatii recomandate este 46  
1 1 1 1 1 1 0 0  
2 1 1 1 1 1 0 0  
1 1 1 1 1 1 2 0  
1 1 1 1 1 2 0 0  
1 1 1 1 2 1 0 0  
2 1 1 3 1 1 2 0  
1 1 1 1 2 1 2 0  
3 2 4 1 1 1 3 0  
3 2 4 1 1 1 3 0  
1 1 1 1 2 1 3 0  
1 1 1 1 2 1 3 0  
1 1 1 1 2 2 0 0  
1 1 1 2 1 1 0 0  
1 1 1 2 1 1 2 0  
1 1 1 2 1 1 3 0  
1 1 1 2 1 1 3 0  
1 1 1 2 1 2 0 0  
1 1 1 2 1 2 2 0  
1 1 1 2 1 2 3 0  
1 1 1 2 2 1 3 0  
2 1 2 2 2 1 3 0  
1 1 1 2 2 2 0 0  
2 1 4 4 3 2 3 0  
1 1 1 2 2 2 2 0  
2 3 1 1 1 1 0 0  
1 1 1 2 3 2 3 0  
1 1 1 3 1 1 0 0  
3 1 1 1 1 1 0 0  
1 1 1 3 1 1 2 0  
1 1 1 4 1 1 0 0  
1 1 2 1 1 1 0 0  
1 1 2 2 1 1 0 0  
1 1 2 2 2 1 2 0  
1 1 2 2 2 1 3 0  
1 1 2 2 2 1 3 0  
1 1 3 1 1 1 0 0  
2 2 1 4 2 1 2 0
```

## Capitolul 5. Concluzii

În concluzie, mașinile cu vectori suport (SVM) sunt foarte des utilizate pentru probleme de clasificare, regresie, dar și pentru alte tipuri de probleme. Metoda se bazează pe o fundamentare matematică riguroasă, dar demonstrează o capacitate foarte bună de generalizare.

## Capitolul 6. Bibliografie

1. Prof. Florin Leon, PhD, *Inteligență artificială: mașini cu vector suport*, Editura Tehnopress
2. Prof. Florin Leon, PhD, *Sinteză Curs Inteligență artificială*,  
[http://florinleon.byethost24.com/Curs\\_IA/SintezaIA12.pdf?i=1](http://florinleon.byethost24.com/Curs_IA/SintezaIA12.pdf?i=1)
3. Prof. Florin Leon, PhD, *Curs Inteligență artificială*,  
[http://florinleon.byethost24.com/Curs\\_IA/IA12\\_RN2.pdf](http://florinleon.byethost24.com/Curs_IA/IA12_RN2.pdf)
4. *UCI Machine Learning Repository*,  
<https://archive.ics.uci.edu/ml/datasets/Nursery?fbclid=IwAR0kX5DJ-OHfStIAkpIc4kBjlNT04ZBfAKDLKjX-5DKwrF9CiWdR0oEb-s>
5. *Microsoft.ML.Data, SvmLightLoader Class*,  
<https://learn.microsoft.com/en-us/dotnet/api/microsoft.ml.data.svmlightloader?view=ml-dotnet>
6. Prof. Florin Leon, PhD, *Laborator Inteligență artificială*,  
[http://florinleon.byethost24.com/lab\\_ia.html](http://florinleon.byethost24.com/lab_ia.html)

## Capitolul 7. Rolul fiecărui membru al echipei

- Lungu Ștefania-Paraschiva:
  - class Solve
  - class Equation
  - class Kernel
  - class DataManipulation.readFile()
  - class DataManipulation.writeFile()
  - Documentație
  - Coeficient bias
- Poleac Alexandra-Cătălina:
  - class Selection
  - class Crossover
  - class Mutation
  - class DataManipulation.procent()
  - Documentație
  - Coeficient alpha și w