**KAUNO TECHNOLOGIJOS UNIVERSITETAS  
INFORMATIKOS FAKULTETAS**

**INTELEKTIKOS PAGRINDAI (P176B101)  
*Laboratorinio darbo nr. 3 ataskaita***

Atliko:

IFF-4/1 grupės studentas  
Aidas Balčaitis

Priėmė:

doc. Germanas Budnikas

Turinys

[1. Užduotis 3](#_Toc479288616)

[1.1. Užduoties aprašymas 3](#_Toc479288617)

[1.2. Programos Naudojimas 3](#_Toc479288618)

[2. Tyrimas 4](#_Toc479288619)

[3. Išvados 6](#_Toc479288620)

# Užduotis

## Užduoties aprašymas

Sukurti programą SPAMui klasifikuoti panaudojant Bajeso teoremą. Ištirti priklausomybę tarp programoje naudojamų nustatymų ir klasifikatoriaus darbo efektyvumo (*false positive*, *true positive*).

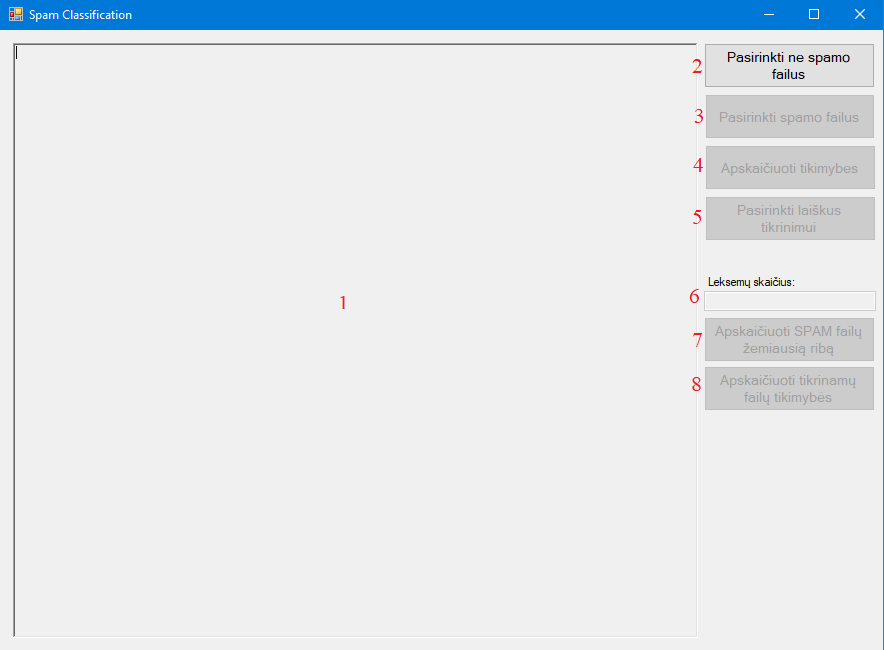
Eiga:

* Sudaromi 2 failų katalogai su SPAM‘u ir NESPAM‘u (apsimokymo duomenys).
* Sudaroma duomenų struktūra (*dictionary*), į kurią įrašomi kiekvienos leksemos pasirodymo kiekiai SPAM ir NESPAM failuose.
* Apskaičiuojama kiekvienos leksemos spamiškumo tikimybė pagal formulę: 
* Jei leksema nebuvo rast tarp apsimokymo duomenų, jai priskiriama numatytoji reikšmė – 0.4.
* Iš analizuojamo failo arba failų pasirenkamas tam tikras leksemų skaičius (pvz. 15-20), kurių spamiškumo tikimybės yra maksimaliai nutolusios nuo neutralios.
* Apskaičiuojama testuojamo failo tikimybė pagal formulę: 

## Programos Naudojimas

Darbui atlikti buvo sukurta grafinė vartotojo sąsaja C# kalbos pagrindu.

1. Pagrindinis programos langas. Jame rašomi įvykdyti veiksmai.
2. Mygtukas skirtas pasirinkti SPAM failus.
3. Mygtukas skirtas pasirinkti NE SPAM failus.
4. Mygtukas apskaičiuojantis visų apsimokymo duomenų leksemų tikimybes ir įrašantis jas į duomenų struktūrą (dictionary).
5. Mygtukas skirtas pasirinkti laiškus, kuriems norime nustatyti spamiškumo tikimybę.
6. Įvedimo laukas, skirtas leksemų skaičiaus pasirinkimui. Šis skaičius naudojamas nustatant pasirinktų laišku spamiškumo tikimybes.
7. Mygtukas, skirtas rasti spam failų mažiausią spamiškumo tikimybę.
8. Mygtukas, apskaičiuojantis 5 mygtuko paspaudimu pasirinktų laišku spamiškumo tikimybes.



pav. 1 Vartotojo sąsaja

# Tyrimas

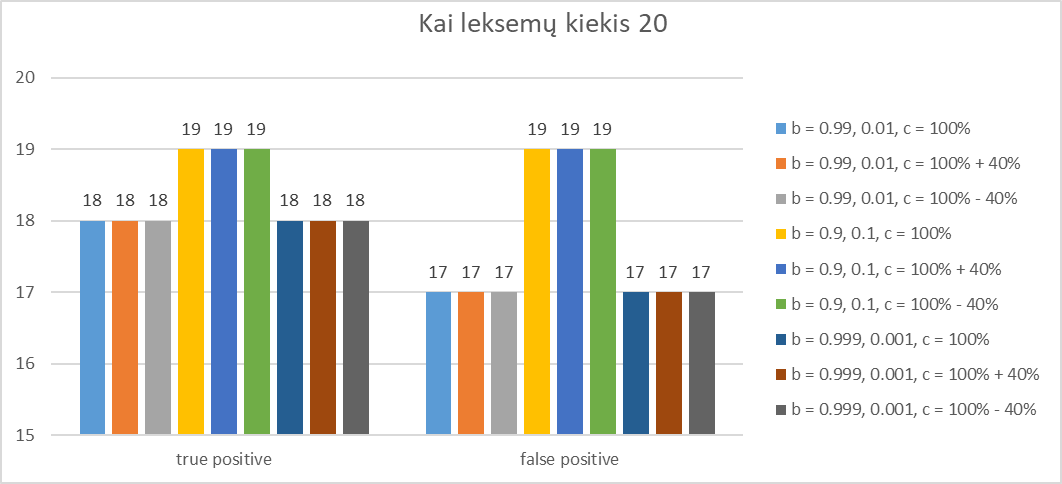
Iš viso tyrimui panaudojau 60 failų: 20 failų buvo sugeneruoti atsitiktinaim 20 failų buvo panašūs į tyrime naudojamus ne spam failus ir 20 failų buvo panašūs į tyrime naudojamus spam failus. Tyrimui atlikti keisime tris kintamuosius:

* a – Leksemų kiekis, pagal kurį nustatysime failo spamiškumo tikimybę. Naudosime šias reikšmes:
  + 20
  + 100
  + 500
* b – Jei leksema pasirodo tik spame, jos spamiškumo tikimybę priskiriame 0.9, 0.99, 0.999, jei leksema pasirodo tik ne spam laiškuose, jos spamiškumo tikimybę priskiriame 0.1, 0.01, 0.001.
  + 0.99, 0.01
  + 0.9, 0.1
  + 0.999, 0.001
* c – kintamasis, gaunamasis randant mažiausią spamiško failo tikimybę, kad jis spam failas. Tyrime keisime reikšmes pridėję, arba atėmę procentinę šio kintamojo reikšmę prie jo paties. Šio kintamojo reikšmės:
  + 100%
  + 100% + 40%
  + 100% - 40%

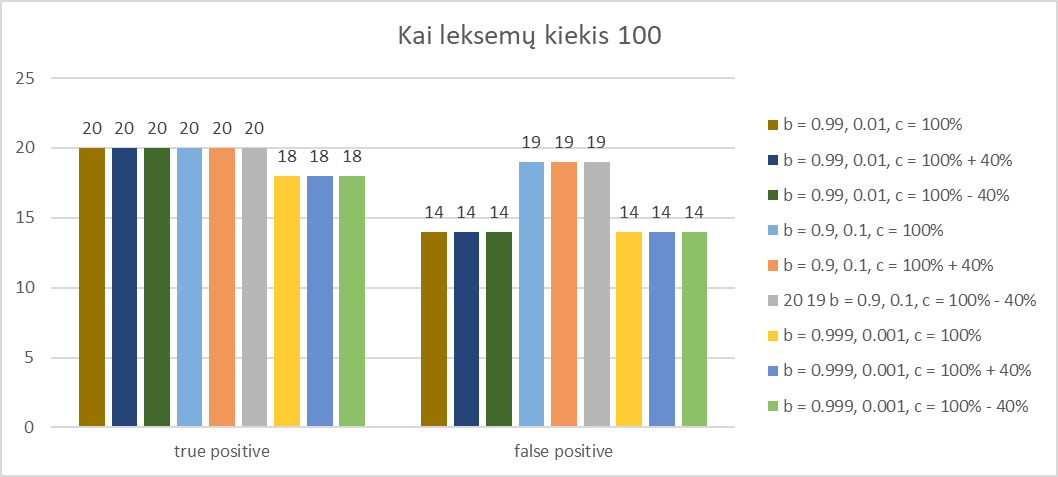
Tyrime vertinsime 2 kriterijus:

* True positive – spam laiškas, kuris klasifikatoriaus buvo priskirtas prie spam‘o.
* False positive – ne spam laiškas, kuris klasifikatoriaus buvo priskirtas prie spam‘o.

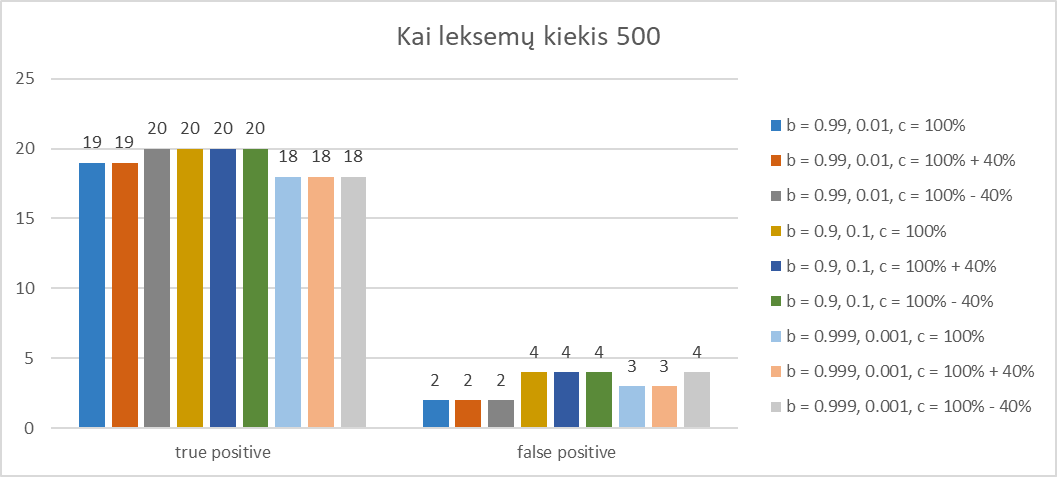
Pirmasis grafikas, pav. 2, kai leksemų kiekis 20.



pav. 2 Klasifikatoriaus true positive ir false positive reiškmės, kai leksemų kiekis 20.



pav. 3 Klasifikatoriaus true positive ir false positive reiškmės, kai leksemų kiekis 100.



pav. 4 Klasifikatoriaus true positive ir false positive reiškmės, kai leksemų kiekis 500.

# Programos kodas

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace SpamClassification

{

    public partial class Form1 : Form

    {

        private Data table;

        private List<fileData> filesData;

        private List<fileData> spamFilesData;

        public List<string> spamFiles;

        public static double defaultProbability = 0.4;

        public static double onlyNotInSpam = 0.01;

        public static double onlyInSpam = 0.99;

        public static string defaultLexemCount = "20";

        public static double lowestSpamProbability = 1;

        private class fileData

        {

            public string fileName { set; get; }

            private Dictionary<string, double> wordsProbabilities;

            public double probability { set; get; }

            public fileData()

            {

                wordsProbabilities = new Dictionary<string, double> { };

                fileName = "";

                probability = 0.0;

            }

            public void addWord(string word, Data data)

            {

                if (!this.wordsProbabilities.ContainsKey(word))

                {

                    if (data.ContainsKey(word))

                        this.wordsProbabilities.Add(word, data.getProb(word)[2]);

                    else

                        this.wordsProbabilities.Add(word, defaultProbability);

                }

            }

            public void addWord(string word)

            {

                if (!this.wordsProbabilities.ContainsKey(word))

                {

                    this.wordsProbabilities.Add(word, 0.0);

                }

            }

            public void updateWordsProbabilities(Data data)

            {

                for(int i = 0; i < wordsProbabilities.Count; i++)

                {

                    if (data.ContainsKey(wordsProbabilities.ElementAt(i).Key))

                        this.wordsProbabilities[wordsProbabilities.ElementAt(i).Key] = data.getProb(wordsProbabilities.ElementAt(i).Key)[2];

                    else

                        this.wordsProbabilities[wordsProbabilities.ElementAt(i).Key] = defaultProbability;

                }

            }

            public void countProbability(IEnumerable<KeyValuePair<string, double>> top)

            {

                double p1 = 1.0,

                    p2 = 1.0;

                foreach (var item in top)

                {

                    p1 \*= item.Value;

                    p2 \*= (1 - item.Value);

                }

                probability = p1 / (p1 + p2);

            }

            public void countProbability(int length)

            {

                double p1 = 1.0,

                    p2 = 1.0;

                if (length >= wordsProbabilities.Count)

                {

                    foreach (var item in wordsProbabilities)

                    {

                        p1 \*= item.Value;

                        p2 \*= (1 - item.Value);

                    }

                    probability = p1 / (p1 + p2);

                }

                else

                {

                    Dictionary<string, double> temp = new Dictionary<string, double> (wordsProbabilities);

                    for (int i = 0; i < temp.Count; i++)

                        temp[temp.ElementAt(i).Key] = Math.Abs(0.5 - temp.ElementAt(i).Value);

                    var top = temp.OrderByDescending(pair => pair.Value).Take(length);

                    foreach (var item in top)

                    {

                        p1 \*= wordsProbabilities[item.Key];

                        p2 \*= (1 - wordsProbabilities[item.Key]);

                    }

                    probability = p1 / (p1 + p2);

                }

            }

            public Dictionary<string, double> getAllValues()

            {

                return wordsProbabilities;

            }

        }

        private class Data

        {

            /\*

            @param int[0] leksemos pasikartojimas spamo laiškuose

            @param int[1] leksemos pasikartojimas ne spamo laiškuose

            \*/

            private Dictionary<string, int[]> table;

            /\*

            @param double[0] leksemos tikimybė spamo laiškuose | P(W|S)

            @param double[1] leksemos tikimybė ne spamo laiškuose | P(W|H)

            @param duoble[2] P(S|W) Apsimokymo duomenų leksemos spamiškumo tikimybė

            \*/

            private Dictionary<string, double[]> probTable;

            public int SpamCount { set; get; } // leksemų kiekis spam laiškuose

            public int HamCount { set; get; } // leksemų kiekis ne spam laiškuose

            public Data()

            {

                table = new Dictionary<string, int[]> { };

                probTable = new Dictionary<string, double[]> { };

                SpamCount = 0;

                HamCount = 0;

            }

            public void CalculateProbabilities()

            {

                foreach (KeyValuePair<string, int[]> pair in this.table)

                {

                    double[] arr = new double[3];

                    arr[0] = (double)pair.Value[0] / SpamCount;

                    arr[1] = (double)pair.Value[1] / HamCount;

                    if (pair.Value[0] == 0)

                        arr[2] = onlyNotInSpam;

                    else if (pair.Value[1] == 0)

                        arr[2] = onlyInSpam;

                    else

                        arr[2] = arr[0] / (arr[0] + arr[1]);

                    probTable.Add(pair.Key, arr);

                }

            }

            public void addSpamPair(string word)

            {

                if (table.ContainsKey(word))

                    table[word][0] += 1;

                else

                {

                    int[] arr = new int[2];

                    arr[0] = 1;

                    arr[1] = 0;

                    table.Add(word, arr);

                    this.SpamCount++;

                }

            }

            public void addNotSpamPair(string word)

            {

                if (table.ContainsKey(word))

                    table[word][1] += 1;

                else

                {

                    int[] arr = new int[2];

                    arr[0] = 0;

                    arr[1] = 1;

                    table.Add(word, arr);

                    this.HamCount++;

                }

            }

            public int[] getValue(string word)

            {

                return table[word];

            }

            public double[] getProb(string word)

            {

                return probTable[word];

            }

            public bool ContainsKey(string word)

            {

                return probTable.ContainsKey(word);

            }

            public Dictionary<string, int[]> getAllValues()

            {

                return table;

            }

        }

        public Form1()

        {

            InitializeComponent();

            button1.Focus();

            button2.Enabled = false;

            button3.Enabled = false;

            button4.Enabled = false;

            button5.Enabled = false;

            button6.Enabled = false;

            textBox1.Enabled = false;

            table = new Data();

            filesData = new List<fileData>();

            spamFilesData = new List<fileData>();

            spamFiles = new List<string>();

        }

        private void button1\_Click(object sender, EventArgs e)

        {

            try

            {

                OpenFileDialog fop = new OpenFileDialog();

                fop.Multiselect = true;

                fop.InitialDirectory = "D:\\STUDIJOS\\6 semestras\\Intelektikos pagrindai\\L3";

                fop.Filter = "txt|\*.txt";

                fop.Title = "Pasirinkti ne spamo failus";

                String pattern = @"[A-Za-z0-9$'""]+";

                int HamLettersCount = 0;

                if (fop.ShowDialog() == DialogResult.OK)

                {

                    foreach (String filename in fop.FileNames)

                    {

                        FileStream FS = new FileStream(filename, FileMode.Open, FileAccess.Read);

                        var lines = File.ReadLines(filename);

                        foreach (var line in lines)

                        {

                            foreach (Match m in Regex.Matches(line, pattern))

                            {

                                table.addNotSpamPair(m.Value);

                            }

                        }

                        HamLettersCount++;

                        FS.Close();

                    }

                    richTextBox1.Text += "Iš viso nuskaityta HAM failų: " + HamLettersCount + "\n";

                    MessageBox.Show("Ne spamo failai sėkmingai nuskaityti!", "Information", MessageBoxButtons.OK, MessageBoxIcon.Information);

                    button2.Enabled = true;

                    button2.Focus();

                }

            }

            catch (Exception ex)

            {

                MessageBox.Show("Error: Could not read file from disk. Original error: " + ex.Message);

            }

        }

        private void button2\_Click(object sender, EventArgs e)

        {

            try

            {

                OpenFileDialog fop = new OpenFileDialog();

                fop.Multiselect = true;

                fop.InitialDirectory = "D:\\STUDIJOS\\6 semestras\\Intelektikos pagrindai\\L3";

                fop.Filter = "txt|\*.txt";

                fop.Title = "Pasirinkti spamo failus";

                String pattern = @"[A-Za-z0-9$'""]+";

                int spamLettersCount = 0;

                if (fop.ShowDialog() == DialogResult.OK)

                {

                    foreach (String filename in fop.FileNames)

                    {

                        fileData temp = new fileData();

                        temp.fileName = filename;

                        FileStream FS = new FileStream(filename, FileMode.Open, FileAccess.Read);

                        var lines = File.ReadLines(filename);

                        foreach (var line in lines)

                        {

                            foreach (Match m in Regex.Matches(line, pattern))

                            {

                                table.addSpamPair(m.Value);

                                temp.addWord(m.Value);

                            }

                        }

                        spamLettersCount++;

                        spamFiles.Add(filename);

                        spamFilesData.Add(temp);

                        FS.Close();

                    }

                    MessageBox.Show("Spamo failai sėkmingai nuskaityti!", "Information", MessageBoxButtons.OK, MessageBoxIcon.Information);

                    richTextBox1.Text += "Iš viso nuskaityta SPAM failų: " + spamLettersCount + "\n";

                    button3.Enabled = true;

                    button3.Focus();

                }

            }

            catch (Exception ex)

            {

                MessageBox.Show("Error: Could not read file from disk. Original error: " + ex.Message);

            }

        }

        private void openFileDialog1\_FileOk(object sender, CancelEventArgs e)

        {

        }

        private void button3\_Click(object sender, EventArgs e)

        {

            table.CalculateProbabilities();

            double[] arr = new double[3];

            arr = table.getProb("is");

            Console.WriteLine("Is tikimybės: " + arr[0] + ", " + arr[1] + ", " + arr[2]);

            button4.Enabled = true;

            button4.Focus();

        }

        private void button4\_Click(object sender, EventArgs e)

        {

            try

            {

                OpenFileDialog fop = new OpenFileDialog();

                fop.Multiselect = true;

                fop.InitialDirectory = "D:\\STUDIJOS\\6 semestras\\Intelektikos pagrindai\\L3";

                fop.Filter = "txt|\*.txt";

                fop.Title = "Pasirinkti laiškus tikrinimui";

                String pattern = @"[A-Za-z0-9$'""]+";

                int lettersCount = 0;

                if (fop.ShowDialog() == DialogResult.OK)

                {

                    foreach (String filename in fop.FileNames)

                    {

                        fileData temp = new fileData();

                        temp.fileName = filename;

                        FileStream FS = new FileStream(filename, FileMode.Open, FileAccess.Read);

                        var lines = File.ReadLines(filename);

                        foreach (var line in lines)

                        {

                            foreach (Match m in Regex.Matches(line, pattern))

                            {

                                temp.addWord(m.Value, table);

                            }

                        }

                        lettersCount++;

                        FS.Close();

                        filesData.Add(temp);

                    }

                    richTextBox1.Text += "Pasirinktų testuoti laiškų kiekis: " + lettersCount + "\n";

                    MessageBox.Show("Failai tikrinimui nuskaityti!", "Information", MessageBoxButtons.OK, MessageBoxIcon.Information);

                    textBox1.Enabled = true;

                    textBox1.Text = defaultLexemCount;

                }

            }

            catch (Exception ex)

            {

                MessageBox.Show("Error: Could not read file from disk. Original error: " + ex.Message);

            }

        }

        private void textBox1\_TextChanged(object sender, EventArgs e)

        {

            button5.Enabled = false;

            if (!Regex.IsMatch(textBox1.Text, @"^\d+$"))

            {

                textBox1.Text = "Įveskite natūralų skaičių!";

                textBox1.SelectAll();

                textBox1.Focus(); //you need to call this to show selection if it doesn't has focus

                button6.Enabled = false;

            }

            else if(textBox1.Enabled) button6.Enabled = true;

        }

        private void button5\_Click(object sender, EventArgs e)

        {

            richTextBox1.Text += "\n-----------------------------------------------------------------\n";

            richTextBox1.Text +=  "Kai leksemų skaičius: " + textBox1.Text + "\n\n";

            int false\_positive = 0;

            int true\_positive = 0;

            foreach (var file in filesData)

            {

                string name = Path.GetFileName(file.fileName);

                file.countProbability(int.Parse(textBox1.Text));

                richTextBox1.Text += "Tikimybė, kad failas " + file.fileName + " yra SPAM: " + file.probability + "\n";

                if (file.probability >= lowestSpamProbability)

                {

                    richTextBox1.Text += "Failas yra SPAM\n";

                    if (Regex.IsMatch(name, @"ne"))

                        false\_positive++;

                    else true\_positive++;

                }

                else

                {

                    richTextBox1.Text += "Failas yra NE SPAM\n";

                }

            }

            richTextBox1.Text += "true\_positive = "+ true\_positive + "\n";

            richTextBox1.Text += "false\_positive = " + false\_positive + "\n";

        }

        private void button6\_Click(object sender, EventArgs e)

        {

            foreach (fileData file in spamFilesData)

            {

                file.updateWordsProbabilities(table);

            }

            foreach (fileData file in spamFilesData)

            {

                file.countProbability(int.Parse(textBox1.Text));

            }

            foreach (fileData file in spamFilesData)

            {

                if (file.probability < lowestSpamProbability)

                    lowestSpamProbability = file.probability;

            }

            //lowestSpamProbability += lowestSpamProbability \* 0.4;

            richTextBox1.Text += "lowestSpamProbability: " + lowestSpamProbability + "\n";

            button5.Enabled = true;

        }

    }

}

# Išvados

Kurdamas programą susipažinau su bajeso teorimą, išmokau ją taikyti laiškų filtravimui. Tyrimo metu klasifikatorius geriausiai pasirodė su tokiais kintamaisiais:

* a = 500
* b = 0.99, 0.01
* c = 100% - 40%