МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

# Кафедра Інформатики

### ЗВІТ

### з лабораторної роботи № 3

з дисципліни: «[Машинне навчання](https://dl.nure.ua/course/view.php?id=4882)»

Виконав: Перевірила:

ст. гр. ІТІНФ-20-1 Шафроненко А.Ю.

Самченко С. О.

Харків 2022

**Мета заняття:** ознайомитися на практиці з деревами рішень та алгоритмом ID3

**Завдання:**

**Варіант №2:** Для наступної вибірки «Хвороби» за допомогою алгоритму ID3 побудувати дерево рішень:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Возраст** | **Пол** | **Стадия болезни** | **Что болит** | **t °С** | **Прочие ощущения** | **Класс** |
| 1 | Пожилой | Жен. | Средняя | Голова | 36,6 | Озноб | + |
| 2 | Пожилой | Муж. | Средняя | Голова | 36,4 | Озноб | + |
| 3 | Пожилой | Муж. | Поздняя | Голова | 36,7 | Озноб | + |
| 4 | Средний | Жен. | Начальная | Голова | 37,8 | Никаких | + |
| 5 | Пожилой | Жен. | Начальная | Ничего | 38,0 | Нет аппетита | + |
| 6 | Пожилой | Муж. | Начальная | Спина | 36,3 | Нет аппетита | + |
| 7 | Пожилой | Муж. | Начальная | Ничего | 36,6 | Нет аппетита | + |
| 8 | Пожилой | Муж. | Начальная | Голова | 38,2 | Озноб | + |
| 9 | Молодой | Муж. | Начальная | Ничего | 36,6 | Никаких | – |
| 10 | Средний | Жен. | Средняя | Живот | 37,7 | Тошнота | – |
| 11 | Пожилой | Жен. | Начальная | Руки | 36,7 | Никаких | – |
| 12 | Пожилой | Муж. | Поздняя | Ноги | 36,4 | Нет аппетита | – |

**Хід Роботи:**

З результату роботи програми:

Entropy = - p+ log2 p+ - p- log2 p- = - 8/12 \* log2 (8/12) - 4/12 \* log2 (4/12) = 0.918296

Для кожного атрибута розрахуємо виграш інформації як різницю отриманого значення ентропії вибірки та ентропії кожного атрибута:

Gain (S, Возраст) = Entropy – 8/12 Entropy (Пожилой) – 2/12 \* Entropy (Средний) – 1/12\* Entropy (Молодой).

Gain (S, Outlook) = 0.94 – 8/12 \*0,971 – 2/12\*0,971 – 1/12\*0 = 0.046

Для атрибута Пол:

Gain (S, Пол) =0.029

Для атрибута Стадия болезни:

Gain (D, Стадия болезни) = 0.151

Для атрибута Что болит:

Gain (D, Что болит) = 0.548

Для атрибута t, С:

Gain (D, t, С) = 0.028

Для атрибута Прочие ощущения:

Gain (D, Прочие ощущения) = 0.18

Максимальний виграш інформації у атрибута Что болит = 0.548, тому він попадає в корінь дерева рішень:

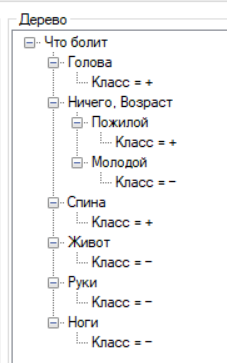
****

Рис. 1 Результати роботи алгоритма ID3

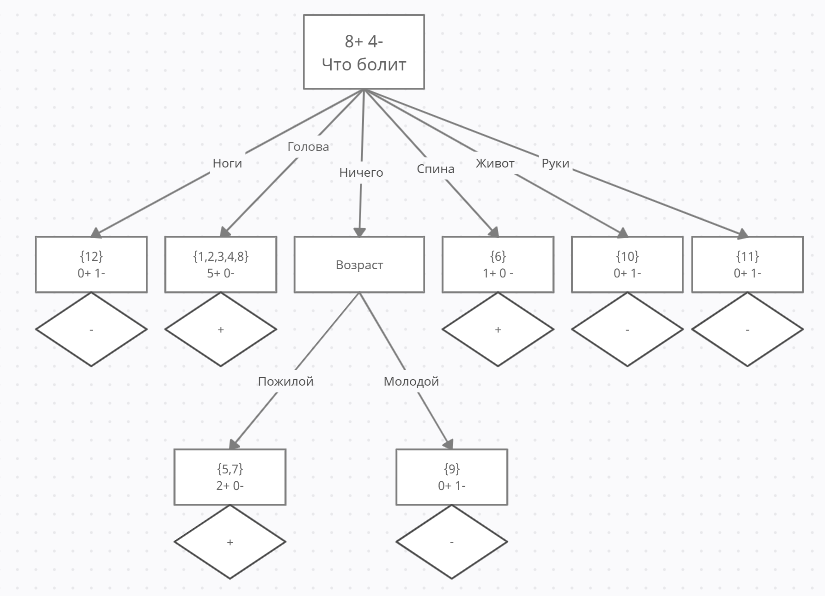


Рис. 2 Зручне представлення дерева рішень

Код програми:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.OleDb;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

namespace decision\_tree

{

    public partial class Form1 : Form

    {

        public Form1()

        {

            InitializeComponent();

        }

        int ent = 0;

        private DataTable elements = new DataTable();

        private readonly DataTable entropy = new DataTable();

        private void n1ToolStripMenuItem\_Click(object sender, EventArgs e)

        {

            tabControl1.Visible = true;

            файлToolStripMenuItem.Visible = true;

            деревьяToolStripMenuItem.Visible = true;

            лабораторныеРаботыToolStripMenuItem.Visible = false;

        }

        private void UpdateCombobox()

        {

            comboBox1.Items.Clear();

            foreach (var column in elements.Columns)

                comboBox1.Items.Add(column.ToString());

            if (comboBox1.Items.Count > 0)

                comboBox1.SelectedIndex = 0;

        }

        private void Form1\_Load(object sender, EventArgs e)

        {

        }

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

        {

            ent = 0;

            var lst = new List<int>();

            var lst2 = elements.Rows.Cast<DataRow>().ToList();

            var gainInd = comboBox1.SelectedIndex;

            if (checkBox1.Checked)

            {

                var gain = double.MaxValue;

                for (var i = 0; i < elements.Columns.Count; ++i)

                {

                    var tmp = BuildInfo(i, lst2);

                    if (tmp < gain)

                    {

                        gain = tmp;

                        gainInd = i;

                    }

                }

            }

            dataGridView2.Columns.Clear();

            ent = 0;

            var rt = BuildID3(gainInd, lst, lst2, "");

            treeView1.Nodes.Clear();

            treeView1.Nodes.Add(rt.Title);

            AddNodes(rt.Childs, treeView1.Nodes[0]);

            treeView1.ExpandAll();

        }

        private void AddNodes(List<TreeNode> list, System.Windows.Forms.TreeNode treeNode)

        {

            foreach (var node in list)

            {

                var nd = treeNode.Nodes.Add(node.Title);

                AddNodes(node.Childs, nd);

            }

        }

        private TreeNode BuildID3(int p, List<int> used, List<DataRow> rows, string val)

        {

            var root = new TreeNode(elements.Columns[p].ToString());

            if (!String.IsNullOrEmpty(val))

                root = new TreeNode(val + ", " + elements.Columns[p]);

            var info = BuildInfo(p, rows);

            int t = ent + 1;

            dataGridView2.Columns.Add("ff", ""+t);

            dataGridView2.Rows[0].Cells[ent].Value = info;

            ent++;

            if (info == 0.0)

            {

                var nd = new TreeNode(val);

                nd.Childs.Add(new TreeNode(elements.Columns[p] + " = " + rows[0][p]));

                return nd;

            }

            var maxGain = 0.0;

            var indMaxGain = -1;

            used.Add(p);

            for (var i = 0; i < elements.Columns.Count; ++i)

            {

                if(used.Contains(i))

                    continue;

                var gain = BuildGain(info, i, rows, p);

                //if (isC45)

                    gain /= BuildInfo(i, rows);

                if (gain > maxGain)

                {

                    maxGain = gain;

                    indMaxGain = i;

                }

            }

            used.Add(indMaxGain);

            if (indMaxGain == -1)

            {

                foreach (DataRow r in rows)

                    root.Childs.Add(new TreeNode(elements.Columns[p] + " = " + r[p]));

                return root;

            }

            root = new TreeNode((string.IsNullOrEmpty(val) ? "" : val + ", ") + elements.Columns[indMaxGain]);

            var listRootValues = new Dictionary<string, List<DataRow>>();

            foreach (DataRow row in rows)

            {

                var vl = row[indMaxGain].ToString();

                if (listRootValues.ContainsKey(vl))

                    listRootValues[vl].Add(row);

                else

                    listRootValues.Add(vl, new List<DataRow> { row });

            }

            foreach (var value in listRootValues)

            {

                var node = BuildID3(p, new List<int>(used), value.Value, value.Key);

                root.Childs.Add(node);

            }

            return root;

        }

        private double BuildGain(double info, int i, List<DataRow> rows, int p)

        {

            var result = info;

            return result - BuildInfoS(i, rows, p);

        }

        private double BuildInfoS(int i, List<DataRow> rows, int p)

        {

            var result = 0.0;

            var listRootFullValues = new Dictionary<string, List<DataRow>>();

            foreach (DataRow row in rows)

            {

                var val = row[i].ToString();

                if (listRootFullValues.ContainsKey(val))

                    listRootFullValues[val].Add(row);

                else

                    listRootFullValues.Add(val, new List<DataRow> { row });

            }

            foreach (var value in listRootFullValues)

            {

                var rows2 = rows.Where(dataRow => (string)dataRow[i] == value.Key).ToList();

                result = result + ((float)(value.Value.Count) / rows.Count) \* BuildInfo(p, rows2);

            }

            return result;

        }

        private double BuildInfo(int i, List<DataRow> rows)

        {

            var result = 0.0;

            var listRootFullValues = new Dictionary<string, List<DataRow>>();

            foreach (DataRow row in rows)

            {

                var val = row[i].ToString();

                if (listRootFullValues.ContainsKey(val))

                    listRootFullValues[val].Add(row);

                else

                    listRootFullValues.Add(val, new List<DataRow> { row });

            }

            foreach (var value in listRootFullValues)

                result = result + ((float)(value.Value.Count) / rows.Count) \* Math.Log((float)(value.Value.Count) / rows.Count, 2);

            return -result;

        }

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

        {

            tabControl1.Visible = false;

            файлToolStripMenuItem.Visible = false;

            лабораторныеРаботыToolStripMenuItem.Visible = true;

            деревьяToolStripMenuItem.Visible = false;

        }

        private void checkBox1\_CheckedChanged(object sender, EventArgs e)

        {

            if (checkBox1.Checked == true)

            {

                comboBox1.Enabled = false;

            }

            else

            {

                comboBox1.Enabled = true;

            }

        }

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

        {

            elements.Clear();

            dataGridView1.Columns.Clear();

            treeView1.Nodes.Clear();

            dataGridView2.Columns.Clear();

            OpenFileDialog ofd = new OpenFileDialog();

            ofd.InitialDirectory = Application.StartupPath;

            ofd.DefaultExt = "\*.mdb;\*.accdb";

            ofd.Title = "Выберите базу данных";

            string commandText = "SELECT \* FROM decision\_tree";

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

            {

                string conStr = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" +ofd.FileName;

                OleDbConnection conn = new OleDbConnection(conStr);

                OleDbCommand cmd = new OleDbCommand(commandText, conn);

                conn.Open();

                OleDbDataAdapter da = new OleDbDataAdapter(cmd);

                DataTable employees = new DataTable();

                da.Fill(employees);

                dataGridView1.DataSource = employees;

                elements = employees;

                UpdateCombobox();

            }

            button1.Enabled = true;

        }

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

        {

        }

        private void загрузитьToolStripMenuItem1\_Click(object sender, EventArgs e)

        {

            treeView1.Nodes.Clear();

            elements.Clear();

            dataGridView1.Columns.Clear();

            dataGridView2.Columns.Clear();

            OpenFileDialog ofd = new OpenFileDialog();

            ofd.InitialDirectory = Application.StartupPath;

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

            {

                LoadTree(treeView1, ofd.FileName);

                treeView1.ExpandAll();

            }

        }

        private void сохранитьToolStripMenuItem1\_Click(object sender, EventArgs e)

        {

        }

        public static void SaveTree(TreeView tree, string filename)

        {

            using (Stream file = File.Open(filename, FileMode.Create))

            {

                BinaryFormatter bf = new BinaryFormatter();

                bf.Serialize(file, tree.Nodes.Cast<System.Windows.Forms.TreeNode>().ToList());

            }

        }

        public static void LoadTree(TreeView tree, string filename)

        {

            using (Stream file = File.Open(filename, FileMode.Open))

            {

                BinaryFormatter bf = new BinaryFormatter();

               object obj = bf.Deserialize(file);

                System.Windows.Forms.TreeNode[] nodeList = (obj as IEnumerable<System.Windows.Forms.TreeNode>).ToArray();

                tree.Nodes.AddRange(nodeList);

            }

        }

        private void SaveToolStripMenuItem1\_Click\_1(object sender, EventArgs e)

        {

            SaveFileDialog save = new SaveFileDialog();

            save.InitialDirectory = Application.StartupPath;

            save.Filter = "Текстовый файл (\*.txt)|\*.txt";

            save.DefaultExt = "\*.txt";

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

            {

                SaveTree(treeView1,save.FileName+".txt");

            }

        }

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

        {

        }

        private void radioButton1\_CheckedChanged(object sender, EventArgs e)

        {

        }

        private void dataGridView2\_CellContentClick(object sender, DataGridViewCellEventArgs e)

        {

        }

    }

    public class TreeNode

    {

        public string Title = "";

        public List<TreeNode> Childs = new List<TreeNode>();

        public TreeNode(string p)

        {

            Title = p;

        }

        public override string ToString()

        {

            return Title;

        }

    }

}

**Висновок:** У ході лабораторної роботи я ознайомився з роботою алгоритму ID3 а також реалізував створення дерева рішення за допомогою цього алгоритму. А також навчився знаходити значення ентропії, виграш інформації та корінь рішення дерева.