

Прізвище: Дорош
Ім'я: Володимир
По-батькові: Юрійович
Група: КН-405
Варіант: 5
Github: https://github.com/sQverful/nulp_tpr_labs



Кафедра: САПР
Дисципліна: Теорія прийняття рішень
Перевірив: Кривий Р.З.

ЗВІТ
до лабораторної роботи №2
на тему "Моделі прийняття рішень. Дерево рішень"

Мета роботи: Одержання практичних навичок використання дерева рішень для вирішення проблем.
Задача. Опис

Компанія розглядає питання про будівництво заводу. Можливі три варіанти:

А) Побудувати великий завод вартістю $M1$ тис. доларів. При цьому варіанті можливі великий попит (річний дохід в розмірі $D1$ тис. доларів протягом наступних 5 років) з ймовірністю $P1$ і низький попит (щорічні збитки $D2$ тис. доларів) з ймовірністю $P2$.

Б) Побудувати маленький завод вартістю $M2$ тис. Доларів. При цьому варіанті можливі великий попит (річний дохід в розмірі $D1$ тис. Доларів протягом наступних 5 років) з ймовірністю $P1$ і низький попит (щорічні збитки $D2$ тис. доларів) з ймовірністю $P2$

В) Відкласти будівництво заводу на 1 рік для збору додаткової інформації, яка може бути позитивною або негативною з ймовірністю $P3$ і $P4$ відповідно. У разі позитивної інформації можна побудувати заводи з зазначеним вище розцінками, а ймовірності великого і низького попиту змінюються на $P1$ і $P2$ відповідно. Доходи на наступні 4 роки залишаються колишніми. У разі негативної інформації компанія заводи будувати не буде.

Порядок вирішення завдання:

- 1) Зобразити дерево рішень, що відповідає умовам завдання.
- 2) Провести розрахунок очікуваних доходів для всіх вузлів.
- 3) Вибрати найбільш ефективний варіант рішення.
- 4) Описати порядок виконання роботи.
- 5) Реалізувати програмне забезпечення, яке б розв'язувало дану задачу. Мова програмування неважлива. Обов'язково: дані мають зчитуватись з файлу і виводитись у табличній формі. **Короткі теоретичні відомості:**

Дерево ухвалення рішень (також можуть називатися деревами класифікацій або регресійними деревами) — використовується в галузі статистики та аналізу даних для прогнозних моделей. Структура дерева містить такі елементи: «листя» і «гілки». На ребрах («гілках») дерева ухвалення рішення записані атрибути, від яких залежить цільова функція, в «листі» записані значення цільової функції, а в інших вузлах — атрибути, за якими розрізняються випадки. Щоб класифікувати новий випадок, треба спуститися по дереву до листа і видати відповідне значення. Подібні дерева рішень широко використовуються в інтелектуальному аналізі даних. Мета полягає в тому, щоб створити модель, яка прогнозує значення цільової змінної на основі декількох змінних на вході.

Індивідуальне завдання:

Варіант	А					Б					В			
	M1	D1	P1	D2	P2	M2	D1	P1	D2	P2	P3	P4	P1	P2
5.	720	250	0.8	-65	0.2	250	200	0.8	-65	0.2	0.7	0.3	0.9	0.1

Виконання індивідуального завдання:

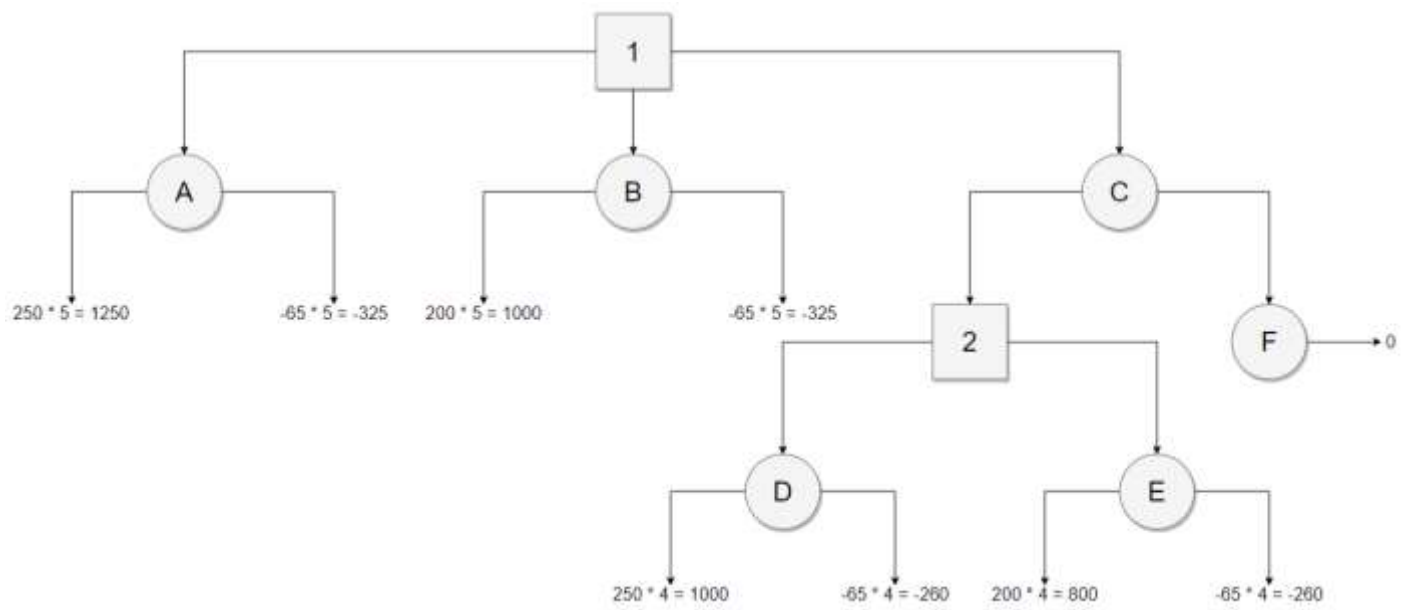


Рис.1. Дерево рішень.

Розрахунки очікуваних доходів.

- $EMV(A) = 0,8 * 1250 + 0,2 * (-325) - 720 = 215$
 $EMV(B) = 0,8 * 1000 + 0,2 * (-325) - 250 = 485$
- $EMV(D) = 0,9 * 1000 + 0,1 * (-260) - 720 = 154$
 $EMV(E) = 0,9 * 800 + 0,1 * (-260) - 250 = 444$
 $EMV(2) = \max \{EMV(D), EMV(E)\} = \max \{154, 444\} = 444 = EMV(E)$
 $EMV(C) = 0,7 * 444 + 0,3 * 0 = 310.8$
- $EMV(1) = \max \{EMV(A), EMV(B), EMV(C)\} = \max \{215; 485; 310.8\} = 485 = EMV(B)$

Код програми:

```
public class TreeDecision {
    public static void main(String[] args) {
        String input = Util.readFile("part2.txt");
    }
}
```

```

        System.out.println(findDecision(input));
    }

    public static String[] getEMV(String input) {
        String[] nodes = new String[6];
        double[] values = ArrayProcessing.getAllNumbers(input);

        TreeNode A = new TreeNode("A", values[0], values[1] * values[14], values[2], values[4],
values[3] * values[14]);
        TreeNode B = new TreeNode("B", values[5], values[6] * values[14], values[7], values[9],
values[8] * values[14]);
        TreeNode D = new TreeNode("D", values[0], values[1] * (values[14] - 1), values[12],
values[13], values[3] * (values[14] - 1));
        TreeNode E = new TreeNode("E", values[5], values[6] * (values[14] - 1), values[12],
values[13], values[8] * (values[14] - 1));

        double EMVofA = calculateNode(A);
        double EMVofB = calculateNode(B);
        double EMVofD = calculateNode(D);
        double EMVofE = calculateNode(E);
        double EMVofF = 0;

        TreeNode C;
        if (EMVofD > EMVofE) {
            C = new TreeNode("C", 0, EMVofD, 0.7, 0.3, 0);
        } else {
            C = new TreeNode("C", 0, EMVofE, 0.7, 0.3, 0);
        }
        double EMVofC = calculateNode(C);

        double[] results = {EMVofA, EMVofB, EMVofC, EMVofD, EMVofE, EMVofF};
        String[] names = {"EMV(A) = ", "EMV(B) = ", "EMV(C) = ", "EMV(D) = ", "EMV(E) = ",
"EMV(F) = ",};

        for (int i = 0; i < nodes.length; i++) {
            nodes[i] = names[i] + results[i];
        }

        return nodes;
    }

    public static String findDecision(String input) {
        StringBuilder sbResult = new StringBuilder();
        double[] values = ArrayProcessing.getAllNumbers(input);

        TreeNode A = new TreeNode("A", values[0], values[1] * values[14], values[2], values[4],
values[3] * values[14]);
        TreeNode B = new TreeNode("B", values[5], values[6] * values[14], values[7], values[9],
values[8] * values[14]);
        TreeNode D = new TreeNode("D", values[0], values[1] * (values[14] - 1), values[12],
values[13], values[3] * (values[14] - 1));
        TreeNode E = new TreeNode("E", values[5], values[6] * (values[14] - 1), values[12],
values[13], values[8] * (values[14] - 1));

        double EMVofA = calculateNode(A);
        double EMVofB = calculateNode(B);
        double EMVofD = calculateNode(D);
        double EMVofE = calculateNode(E);

        TreeNode C;

```

```

        if (EMVofD > EMVofE) {
            C = new TreeNode("C", 0, EMVofD, 0.7, 0.3, 0);
        } else {
            C = new TreeNode("C", 0, EMVofE, 0.7, 0.3, 0);
        }
        double EMVofC = calculateNode(C);

        double[] arrResult = {EMVofA, EMVofB, EMVofC};
        Arrays.sort(arrResult);

        double resultValue = arrResult[2];
        String name = "Err";

        if (resultValue == EMVofA) {
            name = "A";
        } else if (resultValue == EMVofB) {
            name = "B";
        } else if (resultValue == EMVofC) {
            name = "C";
        }
        sbResult.append(name + ": " + arrResult[2]);

        return sbResult.toString();
    }

    public static double calculateNode(TreeNode node) {
        double result = 0;
        double M1 = node.getM1();
        double D1 = node.getD1();
        double P1 = node.getP1();
        double P2 = node.getP2();
        double P3;
        double P4;
        double D2 = node.getD2();

        result = P1 * D1 + P2 * D2 - M1;

        return result;
    }

    public static class TreeNode {
        private String name;
        private double M1;
        private double D1;
        private double P1;
        private double P2;
        private double P3;
        private double P4;
        private double D2;

        public TreeNode() {
        }

        public TreeNode(String name, double m1, double d1, double p1, double p2, double d2) {
            this.name = name;
            M1 = m1;
            D1 = d1;
            P1 = p1;
            P2 = p2;
            D2 = d2;
        }
    }

```

```

public TreeNode(double p1, double p2, double p3, double p4) {
    P1 = p1;
    P2 = p2;
    P3 = p3;
    P4 = p4;
}

public double getM1() {
    return M1;
}

public double getD1() {
    return D1;
}

public double getP1() {
    return P1;
}

public double getP2() {
    return P2;
}

public double getP3() {
    return P3;
}

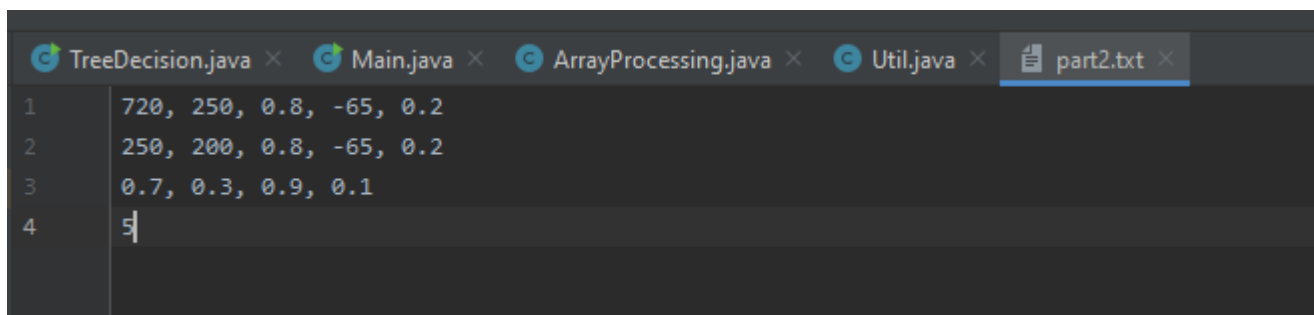
public double getP4() {
    return P4;
}

public double getD2() {
    return D2;
}

public String getName() {
    return this.name;
}
}

```

Результат виконання програми:



```

TreeDecision.java × Main.java × ArrayProcessing.java × Util.java × part2.txt ×
1 720, 250, 0.8, -65, 0.2
2 250, 200, 0.8, -65, 0.2
3 0.7, 0.3, 0.9, 0.1
4 5

```

Рис.2. Файл із вхідними даними part2.txt.

```
Main x
columnLengths = {0=14, 1=14, 2=27, 3=14, 4=14, 5=12, 6=8}
formatString = | %14s | %14s | %27s | %14s | %14s | %12s | %8s |

Line = +-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
|          A |          B |          C |          D |          E |          F | Solution |
+-----+-----+-----+-----+-----+-----+
| EMV(A) = 215.0 | EMV(B) = 485.0 | EMV(C) = 310.79999999999995 | EMV(D) = 154.0 | EMV(E) = 444.0 | EMV(F) = 0.0 | B: 485.0 |
+-----+-----+-----+-----+-----+-----+

Process finished with exit code 0
```

Рис.3. Результат виконання програми.

Висновок: Під час виконання даної лабораторної роботи, я одержав навички використання дерева для вирішення проблем, а саме вирішив задачу побудови заводів із певними вхідними даними.