*Національний технічний університет України*

*«Київський політехнічний інститут»*

***Факультет інформатики та обчислювальної техніки***

Лабораторна робота №4

*з курсу "****Теорія ймовірностей****"*

**Тема: “Статистична перевірка гіпотез”**

***Виконав:***

*Долинний О.В.*

***Група*** *ІО-31*

***Перевірив:***

*Марковський О.П.*

***Київ - 2014р.***

**Завдання Б.**

Аналітично визначити критерій прийняття рішення щодо справедливості двох, гіпотез.

Експериментально виконати генерацію вибірок величин за обома гіпотезами, їх розпізнавання за визначеним критерієм, та вивести ймовірності помилок першого та другого роду.

Критерій: Неймана-Пірсона

*f0(y): H(σ1)*

*f1(y): H(σ2)*

**Клас NormalFunction**

**package** lab\_4;

**import** java.util.Random;

**import** **static** java.lang.Math.\*;

**public** **class** NormalFunction {

**private** **double** sigma;

**private** **double** m;

**public** NormalFunction(**double** sigma, **double** m) {

**this**.sigma = sigma;

**this**.m = m;

}

**public** **double** generateNumber() {

Random rand = **new** Random();

**double** summ = 0;

**for** (**int** i = 0; i < 12; i++) {

summ += rand.nextDouble();

}

**return** (summ - 6) \* sigma + m;

}

**public** **double** get(**double** x) {

**double** arg = -(x - m) \* (x - m) / (2 \* sigma \* sigma);

**return** 1 / (*sqrt*(2 \* ***PI***) \* sigma) \* *exp*(arg);

}

**public** **double** getSigma() {

**return** sigma;

}

**public** **double** getM() {

**return** m;

}

}

**Клас Test**

**package** lab\_4;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.Collections;

**public** **class** Test {

**private** **double** P;

**private** **double** c1;

**private** **double** c2;

**private** NormalFunction f0;

**private** NormalFunction f1;

**public** **static** **void** main(String[] args) {

Test t = **new** Test();

**int** SecontErrorCounter = 0;

**int** FirstErrorCounter = 0;

**for** (**int** i = 0; i < 10000; i++) {

**double** x = t.f0.generateNumber();

**if** (!t.expression(x)) {

SecontErrorCounter++;

}

}

**for** (**int** i = 0; i < 1000000; i++) {

**double** x = t.f1.generateNumber();

**if** (t.expression(x))

FirstErrorCounter++;

}

t.getBounds();

System.***out***.println("Probability of first error:" + FirstErrorCounter/ 10000.0);

System.***out***.println("Probability of second error:" + SecontErrorCounter/ 10000.0);

}

**public** Test() {

**double** m0 = 1;

**double** s0 = 1;

**double** m1 = -1;

**double** s1 = 0.5;

f0 = **new** NormalFunction(s0, m0);

f1 = **new** NormalFunction(s1, m1);

P = 0.8;

c1 = 1;

c2 = 1;

}

**private** **void** getBounds() {

**double** s0 = f0.getSigma();

**double** m0 = f0.getM();

**double** s1 = f1.getSigma();

**double** m1 = f1.getM();

**double** l = P \* c1 / ((1 - P) \* c2);

**double** a = s1 \* s1 - s0 \* s0;

**double** b = 2 \* m1 \* s0 \* s0 - 2 \* m0 \* s1 \* s1;

**double** c = m0 \* m0 \* s1 \* s1 - m1 \* m1 \* s0 \* s0 - 2 \* s0 \* s0 \* s1

\* s1 \* Math.*log*(s1 \* l / s0);

**double** D = b \* b - 4 \* a \* c;

**double** y1 = (-b - Math.*sqrt*(D)) / (2 \* a);

**double** y2 = (-b + Math.*sqrt*(D)) / (2 \* a);

System.***out***.println("Bounds: { " + y1 + " ; " + y2 + " }");

}

**public** **boolean** expression(**double** x) {

**return** f1.get(x) / f0.get(x) < P \* c1 / ((1 - P) \* c2);

}

}