*1.x、y坐标点封装*

**public** **class** DataPoint {

/\*\* the x value \*/

**public** **float** x;

/\*\* the y value \*/

**public** **float** y;

/\*\*

\* Constructor.

\*

\* **@param** x

\* the x value

\* **@param** y

\* the y value

\*/

**public** DataPoint(**float** x, **float** y) {

**this**.x = x;

**this**.y = y;

}

}

*2.实现回归*

import java.math.BigDecimal;

import java.util.ArrayList;

public class RegressionLine // implements Evaluatable

{

/\*\* sum of x \*/

private double sumX;

/\*\* sum of y \*/

private double sumY;

/\*\* sum of x\*x \*/

private double sumXX;

/\*\* sum of x\*y \*/

private double sumXY;

/\*\* sum of y\*y \*/

private double sumYY;

/\*\* sum of yi-y \*/

private double sumDeltaY;

/\*\* sum of sumDeltaY^2 \*/

private double sumDeltaY2;

/\*\* 误差 \*/

private double sse;

private double sst;

private double E;

private String[] xy;

private ArrayList listX;

private ArrayList listY;

private int XMin, XMax, YMin, YMax;

/\*\* line coefficient a0 \*/

private float a0;

/\*\* line coefficient a1 \*/

private float a1;

/\*\* number of data points \*/

private int pn;

/\*\* true if coefficients valid \*/

private boolean coefsValid;

/\*\*

\* Constructor.

\*/

public RegressionLine() {

XMax = 0;

YMax = 0;

pn = 0;

xy = new String[2];

listX = new ArrayList();

listY = new ArrayList();

}

/\*\*

\* Constructor.

\*

\* @param data

\* the array of data points

\*/

public RegressionLine(DataPoint data[]) {

pn = 0;

xy = new String[2];

listX = new ArrayList();

listY = new ArrayList();

for (int i = 0; i < data.length; ++i) {

addDataPoint(data[i]);

}

}

/\*\*

\* Return the current number of data points.

\*

\* @return the count

\*/

public int getDataPointCount() {

return pn;

}

/\*\*

\* Return the coefficient a0.

\*

\* @return the value of a0

\*/

public float getA0() {

validateCoefficients();

return a0;

}

/\*\*

\* Return the coefficient a1.

\*

\* @return the value of a1

\*/

public float getA1() {

validateCoefficients();

return a1;

}

/\*\*

\* Return the sum of the x values.

\*

\* @return the sum

\*/

public double getSumX() {

return sumX;

}

/\*\*

\* Return the sum of the y values.

\*

\* @return the sum

\*/

public double getSumY() {

return sumY;

}

/\*\*

\* Return the sum of the x\*x values.

\*

\* @return the sum

\*/

public double getSumXX() {

return sumXX;

}

/\*\*

\* Return the sum of the x\*y values.

\*

\* @return the sum

\*/

public double getSumXY() {

return sumXY;

}

public double getSumYY() {

return sumYY;

}

public int getXMin() {

return XMin;

}

public int getXMax() {

return XMax;

}

public int getYMin() {

return YMin;

}

public int getYMax() {

return YMax;

}

/\*\*

\* Add a new data point: Update the sums.

\*

\* @param dataPoint

\* the new data point

\*/

public void addDataPoint(DataPoint dataPoint) {

sumX += dataPoint.x;

sumY += dataPoint.y;

sumXX += dataPoint.x \* dataPoint.x;

sumXY += dataPoint.x \* dataPoint.y;

sumYY += dataPoint.y \* dataPoint.y;

if (dataPoint.x > XMax) {

XMax = (int) dataPoint.x;

}

if (dataPoint.y > YMax) {

YMax = (int) dataPoint.y;

}

// 把每个点的具体坐标存入ArrayList中，备用

xy[0] = (int) dataPoint.x + "";

xy[1] = (int) dataPoint.y + "";

if (dataPoint.x != 0 && dataPoint.y != 0) {

System.out.print(xy[0] + ",");

System.out.println(xy[1]);

try {

// System.out.println("n:"+n);

listX.add(pn, xy[0]);

listY.add(pn, xy[1]);

} catch (Exception e) {

e.printStackTrace();

}

/\*

\* System.out.println("N:" + n); System.out.println("ArrayList

\* listX:"+ listX.get(n)); System.out.println("ArrayList listY:"+

\* listY.get(n));

\*/

}

++pn;

coefsValid = false;

}

/\*\*

\* Return the value of the regression line function at x. (Implementation of

\* Evaluatable.)

\*

\* @param x

\* the value of x

\* @return the value of the function at x

\*/

public float at(int x) {

if (pn < 2)

return Float.NaN;

validateCoefficients();

return a0 + a1 \* x;

}

/\*\*

\* Reset.

\*/

public void reset() {

pn = 0;

sumX = sumY = sumXX = sumXY = 0;

coefsValid = false;

}

/\*\*

\* Validate the coefficients. 计算方程系数 y=ax+b 中的a

\*/

private void validateCoefficients() {

if (coefsValid)

return;

if (pn >= 2) {

float xBar = (float) sumX / pn;

float yBar = (float) sumY / pn;

a1 = (float) ((pn \* sumXY - sumX \* sumY) / (pn \* sumXX - sumX

\* sumX));

a0 = (float) (yBar - a1 \* xBar);

} else {

a0 = a1 = Float.NaN;

}

coefsValid = true;

}

/\*\*

\* 返回误差

\*/

public double getR() {

// 遍历这个list并计算分母

for (int i = 0; i < pn - 1; i++) {

float Yi = (float) Integer.parseInt(listY.get(i).toString());

float Y = at(Integer.parseInt(listX.get(i).toString()));

float deltaY = Yi - Y;

float deltaY2 = deltaY \* deltaY;

/\*

\* System.out.println("Yi:" + Yi); System.out.println("Y:" + Y);

\* System.out.println("deltaY:" + deltaY);

\* System.out.println("deltaY2:" + deltaY2);

\*/

sumDeltaY2 += deltaY2;

// System.out.println("sumDeltaY2:" + sumDeltaY2);

}

sst = sumYY - (sumY \* sumY) / pn;

// System.out.println("sst:" + sst);

E = 1 - sumDeltaY2 / sst;

return round(E, 4);

}

// 用于实现精确的四舍五入

public double round(double v, int scale) {

if (scale < 0) {

throw new IllegalArgumentException(

"The scale must be a positive integer or zero");

}

BigDecimal b = new BigDecimal(Double.toString(v));

BigDecimal one = new BigDecimal("1");

return b.divide(one, scale, BigDecimal.ROUND\_HALF\_UP).doubleValue();

}

public float round(float v, int scale) {

if (scale < 0) {

throw new IllegalArgumentException(

"The scale must be a positive integer or zero");

}

BigDecimal b = new BigDecimal(Double.toString(v));

BigDecimal one = new BigDecimal("1");

return b.divide(one, scale, BigDecimal.ROUND\_HALF\_UP).floatValue();

}

}

*3.线性回归测试*

**public** **class** LinearRegression {

**private** **static** **final** **int** ***MAX\_POINTS*** = 10;

**private** **double** E;

/\*\*

\* Main program.

\*

\* **@param** args

\* the array of runtime arguments

\*/

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

RegressionLine line = **new** RegressionLine();

line.addDataPoint(**new** DataPoint(1, 136));

line.addDataPoint(**new** DataPoint(2, 143));

line.addDataPoint(**new** DataPoint(3, 132));

line.addDataPoint(**new** DataPoint(4, 142));

line.addDataPoint(**new** DataPoint(5, 147));

*printSums*(line);

*printLine*(line);

}

/\*\*

\* Print the computed sums.

\*

\* **@param** line

\* the regression line

\*/

**private** **static** **void** printSums(RegressionLine line) {

System.***out***.println("\n数据点个数 n = " + line.getDataPointCount());

System.***out***.println("\nSum x = " + line.getSumX());

System.***out***.println("Sum y = " + line.getSumY());

System.***out***.println("Sum xx = " + line.getSumXX());

System.***out***.println("Sum xy = " + line.getSumXY());

System.***out***.println("Sum yy = " + line.getSumYY());

}

/\*\*

\* Print the regression line function.

\*

\* **@param** line

\* the regression line

\*/

**private** **static** **void** printLine(RegressionLine line) {

System.***out***.println("\n回归线公式: y = " + line.getA1() + "x + "

+ line.getA0());

System.***out***.println("误差： R^2 = " + line.getR());

}

//y = 2.1x + 133.7 2.1 \* 6 + 133.7 = 12.6 + 133.7 = 146.3

//y = 2.1x + 133.7 2.1 \* 7 + 133.7 = 14.7 + 133.7 = 148.4

}

运行结果：

