lRegressor.cpp Source File

**#include <iostream>**

**#include <string>**

**#include <fstream>**

**#include <math.h>**

**using** **namespace** std;

**class** **LRegressor** {

**public**:

LRegressor() {r = 0;}

**int** calculateR(string input\_file);

**double** getR() {**return** r;}

**double** getT() {**return** t;}

**double** getP() {**return** p;}

**private**:

**int** findP(**int** n);

**double** r;

**double** t;

**double** p;

};

**int** LRegressor::calculateR(string input\_file) {

ifstream input;

input.open(input\_file.c\_str());

**double** x = 0, y = 0;

**double** Xp2, Yp2, XY;

**double** sumX = 0, sumY = 0, n = 0;

**double** sumXY = 0, sumXp2 = 0, sumYp2 = 0;

**double** num, denom;

**int** c = input.peek();

**if** (c == EOF || input.fail())

{

cout << input\_file << " is empty or doesn't exist.\n";

**return** 0;

}

input >> x >> y;

**while**(input.good() && x != 0)

{

sumX += x;

sumXp2 += (x\*x);

sumY += y;

sumYp2 += (y\*y);

sumXY += (x\*y);

n++;

x = 0, y = 0;

input >> x >> y;

}

num = n \* sumXY - sumX \* sumY;

denom = sqrt((n \* sumXp2 - pow(sumX, 2)) \* (n \* sumYp2 - pow(sumY, 2)));

r = num / denom;

t = fabs(r) \* sqrt(n - 2)/sqrt(1 - pow(r, 2));

**int** code = findP(n);

**if**(code == 0)

**return** 0;

**return** 1;

}

**int** LRegressor::findP(**int** n) {

**int** DoF = n - 2;

**if**( DoF > 5 || DoF <=0)

{

cout << "number of items is out of range!\n";

**return** 0;

}

**int** col = 0;

ifstream tstream;

tstream.open("tTable.txt");

**double** lowerT, upperT, lowerP, upperP;

tstream.seekg((DoF-1)\*7\*8);

tstream >> lowerT >> upperT;

**while** (t > upperT)

{

lowerT = upperT;

tstream >> upperT;

col++;

}

tstream.seekg(279 + col\*6);

tstream >> lowerP >> upperP;

**double** tRange, pRange, tDiff, pDiff;

tRange = upperT - lowerT;

pRange = upperP - lowerP;

tDiff = upperT - t;

p = upperP - (tDiff \* pRange / tRange);

**return** 1;

}

**int** main (**int** argc, **char**\* argv[]) {

string filename;

**if**(argc == 1)

filename = "input.txt";

**else**

filename = string(argv[2]);

LRegressor LR;

**int** code = LR.calculateR(filename);

**if**(code == 0)

{

cout << "Program Failed!\n";

**return** 0;

}

**double** R = LR.getR();

**double** P = LR.getP();

cout << "Calculcation done!\n";

cout << "R: " << R << "\n";

cout << "R^2(Correlation): " << pow(R, 2) << "\n";

cout << "T: " << LR.getT() << "\n";

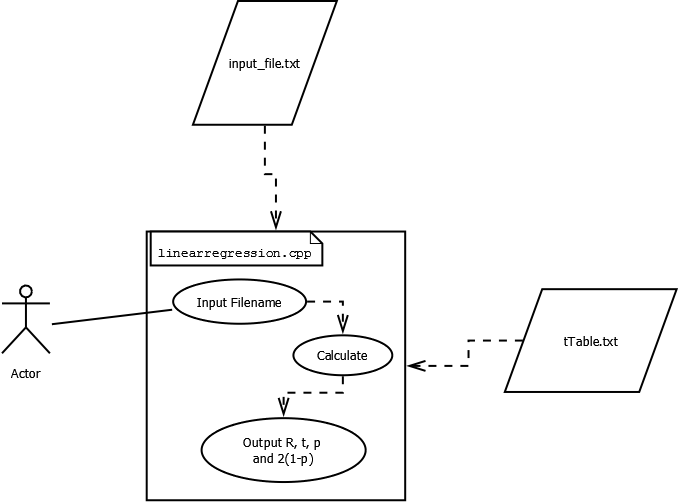
cout << "P: " << P << "\n";

cout << "Significance(2\*(1-p)): " << 2\*(1-P) << "\n";

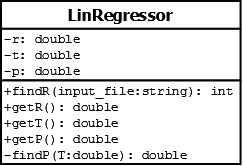
**return** 1;

}

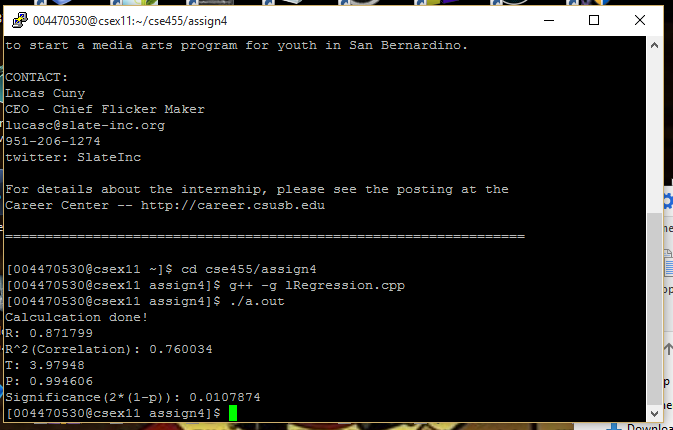
Assignment #4: Use-Case Diagram



Assignment #4 UML Diagram



Screenshot showing Compilation and functionality of Assignment 4



Output Table of Assignment #4

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Expected Value | | | | | Actual Value | | | | |
| r | r2 | t | p | 2\*(1-p) | r | r2 | t | p | 2\*(1-p) |
| 0.871 | 0.75 | 3.96 | 0.99447 | .0110 | 0.872 | 0.76 | 3.98 | 0.994606 | 0.0108 |