**Aishwary Sharma**

**2022CH11437**

**Group-1**

**Tut-7**

Q1.

**A-**

**(1)** Summation Method :

// Header files and namespaces

#include<bits/stdc++.h>

using namespace std;

void lstf(vector<int> v){

for(int i= 0;i< v.size();i++){

cout << v[i]<< " ";

}

}

//Summation method for 1 independent variable (x,y)

vector<double> summation1reg(vector<double>x , vector<double>y){

//y\_pred = a1\*x + a0

//a1 = (N\*SIGMA(xi\*yi)-SIGMA(xi)\*SIGMA(yi))/N\*SIGMA(xi^2)-(SIGMA(xi))^2

//a0 = SIGMA(yi)/N - (SIGMA(xi)/N)\*a1

int N = x.size();

double sum\_xi = 0;

for(int i= 0; i<N;i++){

sum\_xi += x[i];

}

// cout <<"sum\_xi = " <<sum\_xi ;

double sum\_yi = 0;

for(int i= 0; i<N;i++){

sum\_yi += y[i];

}

//cout <<"sum\_yi = " <<sum\_yi ;

double sum\_xi\_sq = 0;

for(int i= 0; i<N;i++){

sum\_xi\_sq += (x[i])\*(x[i]);

}

//cout <<"sum\_xi\_sq = " <<sum\_xi\_sq;

double sum\_xi\_yi = 0;

for(int i= 0; i<N;i++){

sum\_xi\_yi += (x[i])\*(y[i]);

}

//cout <<"sum\_xi\_yi = " <<sum\_xi\_yi ;

double a1 = ((N\*sum\_xi\_yi)-(sum\_xi\*sum\_yi))/((N\*sum\_xi\_sq)-(sum\_xi\*sum\_xi));

double a0 = ((sum\_yi/N)-(sum\_xi/N)\*a1);

cout << "y\_pred = "<< a1 << "x + "<< a0 ;

cout << "y\_pred values : " << endl ;

vector<double> y\_pred ;

for(auto val : x ){

double value = a1\*val + a0 ;

y\_pred.push\_back(value);

cout<< value << endl;

}

return y\_pred;

}

double y\_avg(vector<double>y){

double sum = 0;

for(int i = 0; i<y.size();i++){

sum += y[i];

}

return sum/y.size() ;

}

double S\_r(vector<double>y, vector<double>y\_pred){

double Sr = 0 ;

for(int i = 0; i<y.size();i++){

Sr += pow((y[i]-y\_pred[i]),2) ;

}

return Sr ;

}

double S\_t(vector<double>y, double y\_avg){

double St = 0 ;

for(int i = 0; i<y.size();i++){

St += pow((y[i]-y\_avg),2) ;

}

return St ;

}

int main(){

vector<double> x = {1,2,3,4,5,6,7};

vector<double> y = {0.5,2.5,2.0,4.0,3.5,6.0,5.5};

vector<double>y\_pred = summation1reg(x,y);

double yavg = y\_avg(y);

double St = S\_t(y,yavg);

double Sr = S\_r(y,y\_pred);

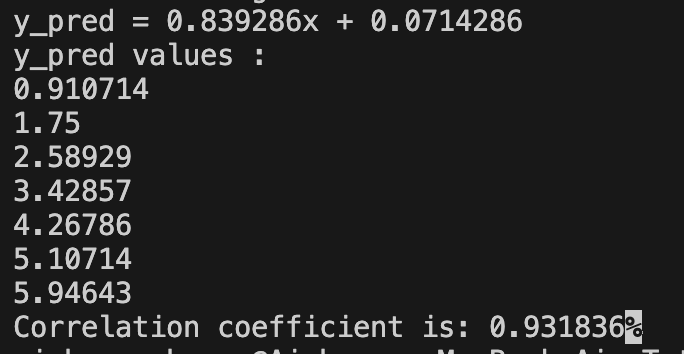
double r2 = (St-Sr)/St ;

// cout << St << " " << Sr << " " << r2;

cout << "Correlation coefficient is: "<< sqrt(r2);

}

Output-



**(2)** Matrix Method : (Implementation for in general multilinear Regression model with any number of parameters)

#include<bits/stdc++.h>

using namespace std;

// printing matrix

void lstf(vector<vector<double>>arr){

for(int i = 0 ; i < arr.size() ; i++){

for(int j = 0; j< arr[0].size() ; j++){

cout << arr[i][j] << " ";

}

cout << endl ;

}

}

void pivot(vector < vector<double> >& v , vector < vector<double> >& I, int k ){

// k --> arr[k][k]

int n = v.size() ;

double big = v[k][k] ;

int p = k ;

for(int i = k+1 ; i< n; i++){

if(v[i][k] > big){

big = v[i][k];

p = i ;

}

}

if(p != k ){

swap(v[p],v[k]);

swap(I[p],I[k]);

}

}

vector<vector<double>> inverseMatrix( vector<vector<double>>& mat) {

int n = mat.size();

int m = mat[0].size();

if (n != m){

cout << "Dimension error" << endl ;

exit(1);

}

vector<vector<double>>Iden(n,vector<double>(n,0));

for(int i = 0 ; i< n; i++){

Iden[i][i] = 1 ;

}

for(int k = 0; k<n ; k++){

pivot(mat,Iden,k);

double div = mat[k][k];

if(abs(div) < 1e-05 ){

cout << "Not invertible";

exit(1);

}

for(int j = k ; j< n ; j++){

mat[k][j]= mat[k][j]/div;

}

for(int j = 0 ; j< n ; j++){

Iden[k][j]= Iden[k][j]/div;

}

for(int l = 0; l< n; l++){

if(l == k){

continue;

}

else{

double factor = mat[l][k];

for(int j = k ; j<n; j++ ){

mat[l][j] = mat[l][j] -factor\*mat[k][j];

if(abs(mat[l][j])<1e-05){

mat[l][j] = 0;

}

}

for(int j = 0 ; j<n; j++ ){

Iden[l][j] = Iden[l][j] -factor\*Iden[k][j];

if(abs(Iden[l][j])<1e-05){

Iden[l][j] = 0;

}

}

}

}

}

cout << "the inverse matrix: " << endl ;

lstf(Iden);

return Iden ;

}

// Matrix Multipication

vector<vector<double>> matrixMultiply( vector<vector<double>>& A, vector<vector<double>>& B) {

int m = A.size(); // Number of rows in A

int n = A[0].size(); // Number of columns in A

int p = B[0].size(); // Number of columns in B

// Initialize the result matrix C with dimensions mxp

vector<vector<double>> C(m, vector<double>(p, 0));

// Perform matrix multiplication

for (int i = 0; i < m; ++i) {

for (int j = 0; j < p; ++j) {

for (int k = 0; k < n; ++k) {

C[i][j] += A[i][k] \* B[k][j];

}

}

}

return C;

}

// Transpose of matrix

vector<vector<double>> transpose( vector<vector<double>>& matrix) {

int rows = matrix.size();

int cols = matrix[0].size();

// Initialize the transpose matrix with dimensions cols x rows

vector<vector<double>> transpose(cols, vector<double>(rows, 0));

// Fill in the transpose matrix

for (int i = 0; i < rows; ++i) {

for (int j = 0; j < cols; ++j) {

transpose[j][i] = matrix[i][j];

}

}

return transpose;

}

int main(){

ifstream in("Input1.txt");

//vector < vector<double> > test = {{2,-1,0,1},{0,1,2,3},{0,1,1,2},{2,5,1,4}};

//inverseMatrix(test);

int m , n;

//cout << "No. of rows in data- ";

in >> m ;

//cout << "No. of independent variables- ";

in >> n ;

//cout << " y1 - ym: ";

// cout << m << " " << n ;

vector<vector<double>> Y(m,vector<double>(1,0)) ;

for(int i =0; i<m; i++){

in >> Y[i][0];

}

vector<vector<double>> X(m,vector<double>(n+1,1));

for(int i = 1; i<n+1 ;i++){

//cout << "Feature "<< i<<" from 1-m";

for(int j = 0; j<m ; j++){

in >> X[j][i];

}

}

//vector<vector<double>> X = {{1,1,1,1,1,1,1},{1,2,3,4,5,6,7}};

//vector<vector<double>> Y = {{0.5},{2.5},{2.0},{4.0},{3.5},{6.0},{5.5}};

cout << " X: " << endl ;

lstf(X);

cout << "Y: " << endl ;

lstf(Y);

vector<vector<double>>transpose\_X = transpose(X);

cout << "transpose(X) : "<< endl ;

lstf(transpose\_X);

vector<vector<double>>XTcrsX = matrixMultiply(transpose\_X,X);

cout << "XTcross X : "<< endl ;

lstf(XTcrsX);

vector<vector<double>> XTcrsX\_inv = inverseMatrix(XTcrsX);

cout << " Xt x X inverse" << endl ;

lstf(XTcrsX\_inv);

vector<vector<double>>XTcrsY = matrixMultiply(transpose\_X,Y) ;

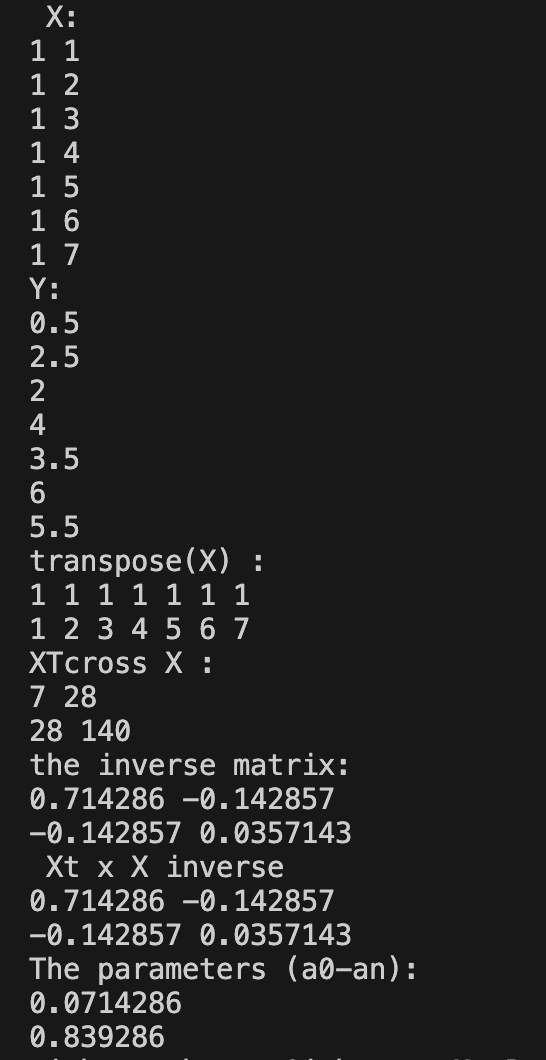
vector<vector<double>> phiVector = matrixMultiply(XTcrsX\_inv,XTcrsY);

cout << "The parameters (a0-an): " << endl ;

lstf(phiVector);

}

Output-



B- (Interchanging x and y)

A black screen with white text

Description automatically generated

The values of parameters (a0 and a1) have changed. The regression line remains the same just the orientation is changed.

Q2.

(1)- Summation method for 2 independent variables

// summation menthod for 2 independent variables

#include<bits/stdc++.h>

using namespace std;

// printing matrix

void lstf(vector<vector<double>>arr){

for(int i = 0 ; i < arr.size() ; i++){

for(int j = 0; j< arr[0].size() ; j++){

cout << arr[i][j] << " ";

}

cout << endl ;

}

}

void pivot(vector <vector<double> >& v , int k ){

// k --> arr[k][k]

int n = v.size() ;

double big = v[k][k] ;

int p = k ;

for(int i = k+1 ; i< n; i++){

if(v[i][k] > big){

big = v[i][k];

p = i ;

}

}

if(p != k ){

swap(v[p],v[k]);

}

}

vector<double> GaussElimination(vector<vector<double>>AugMatrix){

int n = AugMatrix.size();

for (int k = 0 ; k < n ; k++){

pivot(AugMatrix,k);

for (int i = k+1 ; i < n; i++){

double factor = AugMatrix[i][k]/AugMatrix[k][k];

for(int j = 0; j<n+1; j++){

AugMatrix[i][j] = AugMatrix[i][j]- factor\* AugMatrix[k][j];

if (abs(AugMatrix[i][j]) < 1e-05){

AugMatrix[i][j] = 0 ;

}

}

}

}

vector<double> sol(n);

double xn = AugMatrix[n-1][n]/ AugMatrix[n-1][n-1];

sol[n-1] = xn ;

for(int i = n-2; i >= 0; i--){

double bi = AugMatrix[i][n];

double Xi = 0;

for(int j = i+1 ; j< n ; j++){

Xi = Xi + AugMatrix[i][j]\*sol[j];

}

double xi = (bi-Xi)/AugMatrix[i][i];

sol[i]= xi;

}

return sol ;

}

void fns(vector<double>x, vector<double>u, vector<double>y){

int n = x.size();

cout << "n = " << n << endl;

double sum\_xi = 0;

for(int i= 0; i<n ; i++){

sum\_xi += x[i];

cout << sum\_xi << " ";

}

cout << endl ;

double sum\_ui = 0;

for(int i= 0; i<n ; i++){

sum\_ui += u[i];

}

double sum\_xi\_xi = 0;

for(int i= 0; i<n ; i++){

sum\_xi\_xi += x[i]\*x[i];

}

double sum\_ui\_ui = 0;

for(int i= 0; i<n ; i++){

sum\_ui\_ui += u[i]\*u[i];

}

double sum\_ui\_xi = 0;

for(int i= 0; i<n ; i++){

sum\_ui\_xi += u[i]\*x[i];

}

double sum\_yi\_xi = 0;

for(int i= 0; i<n ; i++){

sum\_yi\_xi += y[i]\*x[i];

}

double sum\_yi\_ui = 0;

for(int i= 0; i<n ; i++){

sum\_yi\_ui += y[i]\*u[i];

}

double sum\_yi = 0;

for(int i= 0; i<n ; i++){

sum\_yi += y[i];

}

//Augumented Matrix

vector<vector<double>> A = {{ double(n), sum\_xi, sum\_ui, sum\_yi},

{sum\_xi, sum\_xi\_xi,sum\_ui\_xi,sum\_yi\_xi},

{sum\_ui,sum\_ui\_xi, sum\_ui\_ui,sum\_yi\_ui}};

lstf(A);

vector<double> coefficients = GaussElimination(A) ;

cout << "The coefficients are (ao- a3): " << endl ;

for (auto x: coefficients){

cout << x << endl ;

}

}

int main(){

ifstream in("Input-2.txt");

int m ;

//cout << "No. of rows in data- ";

in >> m ;

cout << m << endl ;

vector<double> y(m) ;

for(int i =0; i<m; i++){

in >> y[i];

// cout << y[i] << " ";

}

// cout << endl ;

vector<double> x(m);

for(int i = 0; i<m ;i++){

in >> x[i];

// cout << x[i] << " ";

}

// cout << endl;

vector<double> u(m);

for(int i = 0; i<m ;i++){

in >> u[i];

// cout << u[i] << " ";

}

fns(x,u,y) ;

}

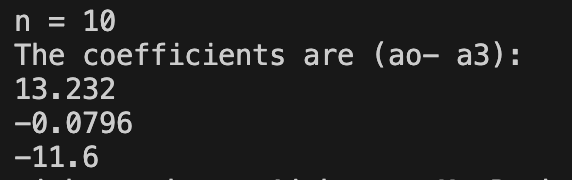
**Output-**

For(y1)

A black background with white text

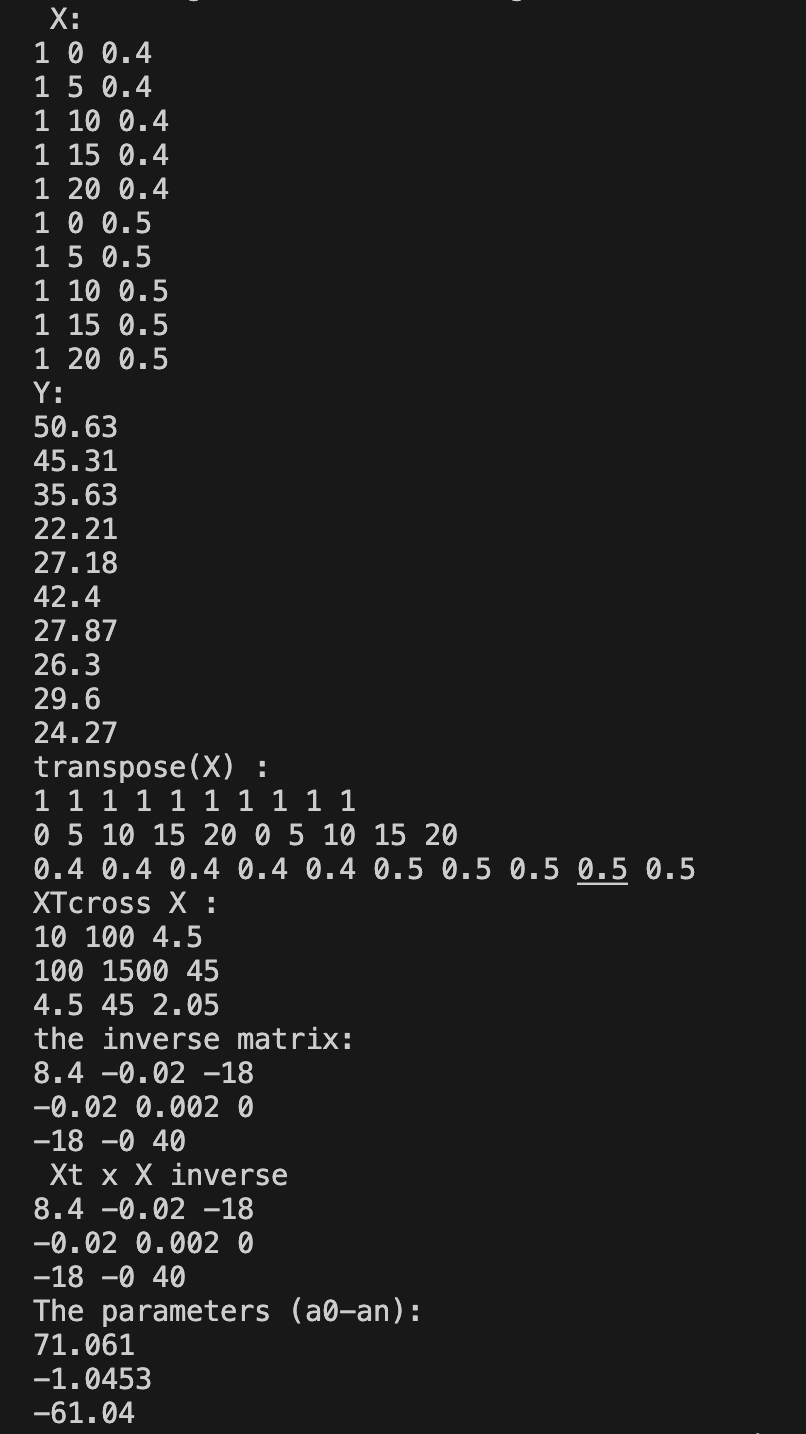
Description automatically generated

For(y2)



Q2- Using Matrix Method (code same as A-2)

For (y1):



For y2:

