

# REGRESSIONE LINEARE MULTIPLA

*Relazione Progetto*

*Metodi Matematici e Statistici (6 CFU)*

*A.A. 2017/2018*

**Mario Raciti**

# ***Sommario***

## **❖ Introduzione**

## **❖ Regressione Lineare Semplice**

- Metodo dei Minimi Quadrati
- Coefficiente di Correlazione (Pearson)

## **❖ Regressione Lineare Multipla**

- Generalizzazione della Regressione Lineare Semplice
- Coefficiente di Determinazione
- Intervallo di Confidenza

## **❖ Multi Linear Regression Java**

- Funzionamento
- Package Jama

## **❖ Applicazione Pratica**

## **❖ Conclusioni**

## **❖ Referenze**

# Introduzione

*Nel 1800 **Giuseppe Piazzi**, presbitero e astronomo italiano, scoprì quella che sembrava essere una nuova stella e ne seguì il movimento per 41 giorni prima di perderne le tracce a causa del maltempo. Il rinomato matematico **Carl Frederick Gauss** utilizzò il suo appena inventato **metodo dei minimi quadrati** per prevedere la futura posizione di quella stella. Effettivamente tale posizione corrispose alla previsione di Gauss e, a quanto pare, il corpo celeste era un piccolo pianeta, **Cerere**. Il famoso calcolo di Gauss predisse la posizione di quel corpo utilizzando 6 variabili. Ad oggi il metodo dei minimi quadrati viene utilizzato in varie discipline quali fisica, chimica, ingegneria, medicina, economia, psicologia.*

Il **problema statistico**, che Gauss risolvette con il metodo dei minimi quadrati e che consiste nel trovare una funzione in grado di descrivere la **relazione** fra due serie di dati, prende il nome di **analisi di regressione**.

L'**algoritmo Multi Linear Regression**, scritto in linguaggio Java e proposto in questo progetto, mira alla risoluzione di un'analisi di **regressione lineare multipla**, la quale costituisce una generalizzazione alla **regressione lineare semplice** trattata dal corso.

# *Regressione Lineare Semplice*

In statistica la **regressione lineare** rappresenta un metodo di stima del valore atteso condizionato di una variabile **dipendente**  $Y$ , dati i valori di altre variabili **indipendenti** (o predittori),  $X_1, X_2, \dots, X_k$ .

La retta di regressione viene ottenuta mediante il metodo dei minimi quadrati.

Il modello di **regressione lineare semplice**, così definita in quanto troviamo un solo predittore, è:

$$y_i = \beta_0 + \beta_1 x_i + w_i$$

con  $i = 1, \dots, n$  e  $w_1, \dots, w_n$  variabili aleatorie indipendenti distribuite secondo una normale  $N(0, \sigma^2)$  in cui  $\sigma^2$  è una quantità incognita indipendente da  $i$ .

## Metodo dei Minimi Quadrati

Il metodo dei **minimi quadrati** è una tecnica utilizzata per trovare la **funzione** che **minimizza** la somma dei quadrati delle distanze tra i dati osservati e quelli della curva rappresentante la funzione stessa, cercando cioè, quella funzione  **$f(x)$**  tale che sia **minima** la funzione **residuo**:

$$g(f) = \sum [f(x_i) - y_i]^2$$

che rappresenta proprio la somma degli “errori”.

Introdotta tale tecnica, per determinare una stima di  $\beta_0$  e  $\beta_1$  basterà cercare i valori di questi ultimi per i quali la seguente quantità sia **minima**:

$$g(\beta_0, \beta_1) = \sum [y_i - \beta_0 - \beta_1 x_i]^2$$

Dunque, per risolvere il problema è sufficiente cercare i valori che **annullano** le **derivate parziali** di tale funzione a due variabili.

Svolgendo i calcoli sopra indicati si ottengono le seguenti soluzioni  $b_0$  e  $b_1$ :

$$b_0 = \bar{y} - b_1 \bar{x}$$

$$b_1 = \frac{\sum (y_i - \bar{y}) x_i}{\sum (x_i - \bar{x})^2}$$

## Coefficiente di Pearson

Poiché il metodo sopra descritto fornisce la retta che meglio approssima i dati ma non il grado di tale approssimazione, è utile ricorrere al **coefficiente di correlazione lineare** (o di **Pearson**) definito nel seguente modo:

$$r_{xy} = C_{xy} / (S_x S_y)$$

dove  $C_{xy}$  è la covarianza,  $S_x S_y$  gli scarti quadratici medi (dev. standard).

Il coefficiente di correlazione è un numero compreso nell'intervallo **[-1, 1]**. Pertanto, se esso **coincide** con gli estremi **-1** o **1** allora i **dati** sono perfettamente **allineati** con la retta di regressione. Inoltre se  $r_{xy} > 0$  allora quest'ultima è **ascendente**.

# Regressione Lineare Multipla

L'analisi di regressione lineare semplice può essere estesa al caso in cui vi siano **più di un predittore** per la variabile **dipendente Y**. In questo caso si parla di **regressione lineare multipla**, il cui modello è:

$$y_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_k x_{ik} + w$$

Lo scopo è **stimare** il vettore di parametri  $(\beta_0, \dots, \beta_k)$  in modo da **minimizzare** la somma dei quadrati degli **errori**.

Si suppone che il primo predittore  $x_1$  assuma sempre il valore 1. Per questo motivo è anche chiamato **fattore costante**. Inoltre si suppone che i vettori  $x_1, \dots, x_k$  siano tra loro **linearmente indipendenti**.

Per questo tipo di regressione lineare è più opportuno svolgere l'analisi passando alla **forma matriciale**. Così facendo otteniamo un sistema di  $n$  equazioni in  $n$  incognite, quindi determinato a priori,  $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$  :

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$$

Supponendo che la matrice  $\mathbf{X}$  sia di colonne di rango completo, è possibile **stimare** il **vettore  $\boldsymbol{\beta}$**  risolvendo le equazioni normali  $\mathbf{X}^T \mathbf{X} \boldsymbol{\beta} = \mathbf{X}^T \mathbf{y}$ .

Il modo più semplice per fare ciò è calcolare esplicitamente  $\mathbf{A} = \mathbf{X}^T \mathbf{X}$  e  $\mathbf{b} = \mathbf{X}^T \mathbf{y}$  e risolvere il sistema di equazioni  $\mathbf{A}\mathbf{x} = \mathbf{b}$  utilizzando l'eliminazione gaussiana.

## Coefficiente di Determinazione

Nelle regressioni lineari semplici il **coefficiente di determinazione** (solitamente indicato con  $R^2$ ) corrisponde con il quadrato del coefficiente di correlazione:

$$R^2 = 1 - (SSR / SST)$$

dove SSR è la devianza residua (o di dispersione) e SST è la devianza totale.

Il coefficiente  $R^2$  è un numero che varia tra 0 e 1. Un valore molto prossimo ad 1 indica che il modello spiega molto bene i dati.

## Intervallo di Confidenza

Si può essere interessati a **stimare** uno dei coefficienti di regressione, anziché a valutarne la significatività. Nel caso del modello di regressione multipla, **l'intervallo di confidenza** per il generico coefficiente di regressione  $k$  è il seguente:

$$\beta_k \pm t_{n-p-1} S_{\beta k}$$

dove  $S_{\beta k}$  è la deviazione standard del coefficiente  $\beta_k$  e  $t_{n-p-1}$  è il test con distribuzione  $t$  con  $n - p - 1$  gradi di libertà.

# Multiple Linear Regression Java

Un **algoritmo** numericamente stabile per il calcolo di  $\beta$  consiste nell'ottenere la **fattorizzazione** QR  $X = QR$  per poi risolvere il sistema triangolare  $R\beta = Q^T y$  tramite sostituzione. Questo è ciò che fa esattamente l'approccio risolutivo proposto dal package **Jama**, per il linguaggio Java, quando si è in presenza di un sistema determinato (assumendo che la matrice abbia il rango completo di colonne).

Il programma **MultipleLinearRegression.java** implementa in modo **semplice** tale algoritmo, calcolando con precisione la **retta di regressione** e il **coefficiente di determinazione** per i dati passati in input.

## Funzionamento

Il programma è strutturato in una **classe** all'interno della quale vengono conservati diversi **stati**: la numerosità **N** di  $y$ , il numero di predittori **p**, i coefficienti di regressione **beta**, la devianza residua **SSR** e la devianza totale **SST**.

```
public class MultipleLinearRegression {  
    private final int N;           // number of  
    private final int p;           // number of dependent variables  
    private final Matrix beta;     // regression coefficients  
    private double SSR;            // sum of squared (residual)  
    private double SST;            // sum of squared (total)  
}
```

La parte più importante che calcola il vettore  $\beta$  è apparentemente resa semplice poiché scritta in appena due righe di codice (questo grazie al package Jama):



```
// find least squares solution
QRDecomposition qr = new QRDecomposition(X);
beta = qr.solve(Y);
```

In questo modo è possibile, tramite l'apposito metodo `getBeta()`, ottenere il j-esimo coefficiente della retta di regressione:

```
public double getBeta(int j) {
    return beta.get(j, 0);
}
```

Infine, il coefficiente di determinazione  $R^2$  viene calcolato secondo la formula che abbiamo richiamato qualche paragrafo fa:

```
public double R2() {
    return 1.0 - SSR/SST;
}
```

dove **SST** è calcolata secondo la definizione matematica e **SSR** con metodi di Jama:

```
// total variation to be accounted for
for (int i = 0; i < N; i++) {
    double dev = y[i] - mean;
    SST += dev*dev;
}
```

```
// variation not accounted for
Matrix residuals = X.times(beta).minus(Y);
SSR = residuals.norm2() * residuals.norm2();
```

Dalla prima di queste due immagini si nota la variabile **mean**, che sarebbe la **media dei valori** di y, così ottenuta:

```
// mean of y[] values
double sum = 0.0;
for (int i = 0; i < N; i++)
    sum += y[i];
double mean = sum / N;
```

Basterà, infine, fornire in **input** le due serie di dati x e y al programma per ottenere l'analisi di regressione multipla. Per semplicità, ci limiteremo ad istanziare staticamente una **matrice** per i predittori e un **array** per la variabile dipendente.

```
public static void main(String[] args) {
    double[][] x = { { 1, 10, 20 },
                     { 1, 20, 40 },
                     { 1, 40, 15 },
                     { 1, 80, 100 },
                     { 1, 160, 23 },
                     { 1, 200, 18 } };
    double[] y = { 243, 483, 508, 1503, 1764, 2129 };
    MultipleLinearRegression regression = new MultipleLinearRegression(x, y);

    StdOut.printf("%.2f + %.2f beta1 + %.2f beta2 (R^2 = %.2f)\n",
                  regression.beta(0), regression.beta(1), regression.beta(2), regression.R2());
}
```

Con opportune modifiche al codice è possibile passare a **runtime** tali valori.

## Package JAMA

Come è stato detto nel paragrafo precedente, il funzionamento dell'algoritmo viene semplificato da un apposito package che fornisce metodi ottimizzati per l'**algebra lineare** in Java. E' molto importante il contributo di questo package per la fase principale dell'algoritmo: la **decomposizione QR**. Inoltre, Jama fornisce una classe **Matrix** che semplifica la gestione delle matrici.

Ovviamente, per compilare ed eseguire il programma **MultipleLinearRegression**, è necessario specificare la dipendenza da questo package.

## Applicazione Pratica

I seguenti dati riportano, per gli anni che vanno dal 1924 al 1955, la qualità del vino di Bordeaux ( $y$ ) e quattro misurazioni meteorologiche del mese di aprile precedente la vendemmia: la somma delle temperature medie giornaliere ( $x_2$ ), il numero di ore di insolazione ( $x_3$ ), il numero di giorni in cui la temperatura ha superato la media stagionale ( $x_4$ ) ed i millimetri di pioggia caduti ( $x_5$ ).

| $y$                          | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|------------------------------|-------|-------|-------|-------|
| { 1, 3064, 1201, 10, 361 },  |       |       |       |       |
| { -1, 3000, 1053, 11, 338 }, |       |       |       |       |
| { 1, 3155, 1133, 19, 393 },  |       |       |       |       |
| { -2, 3080, 970, 4, 467 },   |       |       |       |       |
| { 2, 3245, 1258, 36, 294 },  |       |       |       |       |
| { 3, 3267, 1386, 35, 225 },  |       |       |       |       |
| { -2, 3080, 966, 13, 417 },  |       |       |       |       |
| { -2, 2974, 1185, 12, 488 }, |       |       |       |       |
| { 3, 3038, 1103, 14, 677 },  |       |       |       |       |
| { 1, 3318, 1310, 29, 427 },  |       |       |       |       |
| { 2, 3317, 1362, 25, 326 },  |       |       |       |       |
| { -1, 3182, 1171, 28, 326 }, |       |       |       |       |
| { -2, 2998, 1102, 9, 349 },  |       |       |       |       |
| { 2, 3221, 1424, 21, 382 },  |       |       |       |       |
| { 0, 3019, 1239, 16, 275 },  |       |       |       |       |
| { 0, 3022, 1285, 9, 303 },   |       |       |       |       |
| { 0, 3094, 1329, 11, 339 },  |       |       |       |       |
| { -2, 3009, 1210, 15, 536 }, |       |       |       |       |
| { 1, 3227, 1331, 21, 414 },  |       |       |       |       |
| { 2, 3308, 1368, 24, 282 },  |       |       |       |       |
| { 1, 3212, 1289, 17, 302 },  |       |       |       |       |
| { 3, 3381, 1444, 25, 253 },  |       |       |       |       |
| { 0, 3061, 1175, 12, 261 },  |       |       |       |       |
| { 3, 3478, 1317, 42, 259 },  |       |       |       |       |
| { 1, 3126, 1248, 11, 315 },  |       |       |       |       |
| { 3, 3468, 1508, 43, 286 },  |       |       |       |       |
| { 1, 3252, 1361, 26, 346 },  |       |       |       |       |
| { -2, 3052, 1186, 14, 443 }, |       |       |       |       |
| { 2, 3270, 1399, 24, 306 },  |       |       |       |       |
| { 2, 3198, 1299, 20, 367 },  |       |       |       |       |
| { -2, 2904, 1164, 6, 311 },  |       |       |       |       |
| { 2, 3247, 1277, 19, 375 },  |       |       |       |       |

Come applicazione pratica dell'algoritmo proposto in questo progetto, si vuole calcolare la regressione di  $y$  rispetto ai quattro predittori, verificando che essa coincida con il risultato riportato nel testo di riferimento (es. 6.31, pag. 241).

Ci aspettiamo, pertanto, che l'equazione di regressione sia:

$$y = -26.8 + 0.00767 x_2 + 0.00441 x_3 - 0.0237 x_4 - 0.00597 x_5$$

con un coefficiente di determinazione  $R^2 = 87.3\%$ .

A questo punto creiamo, all'interno del **MultipleLinearRegression**, una matrice **x** per i quattro predittori (ricordando che stiamo supponendo il vettore  $x_1$  come **fattore costante**) e un vettore **y** per la variabile dipendente.

Istanziamo un nuovo oggetto di nome **regression**, dando come parametri al costruttore **x** e **y**, così da poter richiamare il metodo **getBeta** per le variabili indipendenti e ottenere in output il risultato dell'algoritmo:

```
MultipleLinearRegression regression = new MultipleLinearRegression(x, y);

System.out.println("\n>>Regression equation is:");
System.out.println("y = "+ regression.getBeta(0) + " + "
    + regression.getBeta(1) + " x2 + " + regression.getBeta(2) + " x3 + "
    + regression.getBeta(3) + " x4 + " + regression.getBeta(4) + " x5\n");
System.out.println(">>R^2 = " + (regression.R2())*100 + " %\n");
```

Compilando ed eseguendo il programma otteniamo il seguente output:

```
root at kali in ~/Desktop/Multiple Linear Regression/src using
○ java -classpath Jama-1.0.3.jar:. MultipleLinearRegression

>>Regression equation is:
y = -26.773687306297692 + 0.007673512687412797 x2 + 0.004411876913283974 x3 +
-0.023671259620844372 x4 + -0.005970870339764395 x5

>>R^2 = 87.25662354555095 %

>>SSE = 13.23718229205895
>>SST = 103.875
```

Si nota facilmente che il risultato dell'algoritmo è simile a quello proposto dal testo di riferimento (a meno di qualche approssimazione), dimostrando che il programma **MultipleLinearRegression** funziona ed è affidabile.

Per concludere possiamo stimare gli **intervalli di confidenza** per l'inclinazione. Utilizziamo il metodo **confidenceInterval** che calcola tali intervalli utilizzando la formula espressa nel paragrafo dedicato agli intervalli di confidenza:

```
// get confidence interval
public static double[] confidenceInterval(double beta, double t, double S){
    double[] result = new double[2];
    double left = _leftInterval(beta, t, S), right = _rightInterval(beta, t, S);
    result[0] = (left < right) ? left : right;
    result[1] = (left > right) ? left : right;
    return result;
}
```

```
private static double _leftInterval(double beta, double t, double S){
    return beta - (t*S);
}

private static double _rightInterval(double beta, double t, double S) {
    return beta + (t * S);
}
```

Forniamo, quindi, in input le **deviazioni standard** e i **t-test** per i vari coefficienti:

```
double[] confidenceBeta0 = confidenceInterval(regression.beta(0), -4.97, 5.384);
double[] confidenceBeta1 = confidenceInterval(regression.beta(1), 3.97, 0.00193);
double[] confidenceBeta2 = confidenceInterval(regression.beta(2), 3.06, 0.00144);
double[] confidenceBeta3 = confidenceInterval(regression.beta(3), -0.92, 0.02574);
double[] confidenceBeta4 = confidenceInterval(regression.beta(4), -3.88, 0.00154);
```

Ed otteniamo i seguenti risultati:

```
>>Confidence intervals:
beta1: [-53.532167306297694 , -0.01520730629769318]
beta2: [1.141268741279653E-5 , 0.015335612687412799]
beta3: [5.476913283973539E-6 , 0.008818276913283974]
beta4: [-0.04735205962084437 , 9.540379155625889E-6]
beta5: [-0.011946070339764395 , 4.329660235604868E-6]
```

Si può facilmente notare che i vari **coefficienti di regressione** appartengono ai relativi **intervalli di confidenza** trovati mediante l'algoritmo.

## Conclusioni

Da quanto emerso dai risultati ottenuti, poiché il valore di  $R^2$  è abbastanza prossimo a 0.9, possiamo affermare che vi è una **forte correlazione** tra i dati. In altre parole, la qualità del vino di Bourdeaux è **ben spiegata** dalle quattro misurazioni meteorologiche del mese di aprile precedente la vendemmia.

## Referenze

- ❖ [Giuseppe Piazzi](#)
- ❖ [Carl Friedrich Gauss](#)
- ❖ [Metodo dei Minimi Quadrati](#)
- ❖ [Regressione Lineare Semplice](#)
- ❖ [Regressione Lineare Multipla](#)
- ❖ [Coefficiente di Correlazione](#)
- ❖ [Intervallo di Confidenza](#)
- ❖ [Multiple Linear Regression Java](#)
- ❖ [Multiple Linear Regression Docs](#)
- ❖ [JAMA Package Docs](#)
- ❖ [Libro da cui esempio](#)
- ❖ [Appunti Prof. O. Muscato \(Università di Catania\)](#)
- ❖ [Altro sulla Regressione Multipla \(Università di Padova\)](#)
- ❖ [Codice Completo su GitHub](#)