

**UNIVERSIDAD MAYOR, REAL Y PONTIFICIA DE  
SAN FRANCISCO XAVIER DE CHUQUISACA  
FACULTAD DE TECNOLOGÍA**



**SIS-420**

**Carrera :** Ingeniería en Ciencias de la Computación

**Universitaria :** Lujan Renteria David Fernando

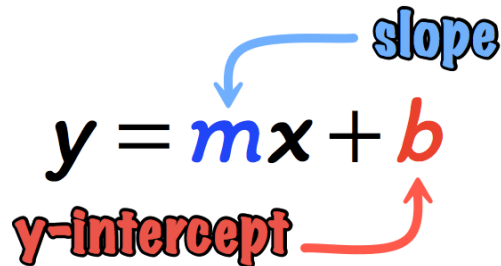
***Sucre – Bolivia***

***2023***

## Conceptos Básicos de Regresión Lineal

La **regresión lineal** es una técnica estadística utilizada para modelar la relación entre una variable dependiente (objetivo) y una o más variables independientes (predictoras). El objetivo es encontrar la mejor línea (o hiperplano en el caso de más de una variable) que minimice la distancia entre los datos reales y la línea ajustada.

La **fórmula general** para una regresión lineal simple es:

$$y = mx + b$$


- y es la variable dependiente (peso en nuestro caso),
- x es la variable independiente (estatura en nuestro caso),
- m es la pendiente de la línea,
- b es el intercepto con el eje y.

Para este ejemplo usaremos datos generados aleatoriamente de altura y peso con el siguiente código:

```
1
2 # Generamos 100 estaturas aleatorias entre 1.4m y 2.0m
3 estaturas = np.random.uniform(1.4, 2.0, 100)
4
5 pesos = [] # Lista para almacenar los pesos generados
6
7 # Bucle para generar pesos aleatorios controlados según la estatura
8 for estatura in estaturas:
9     # Calcular el peso mínimo y máximo usando el IMC saludable (18.5 a 24.9)
10    peso_min = 18.5 * (estatura ** 2) # Peso mínimo según IMC de 18.5
11    peso_max = 24.9 * (estatura ** 2) # Peso máximo según IMC de 24.9
12    # Generar un peso aleatorio entre el peso mínimo y máximo calculado
13    peso = np.random.uniform(peso_min, peso_max)
14    pesos.append(peso) # Añadir el peso a la lista de pesos
15
16 # Crear un DataFrame con los datos de estatura y peso
17 datos = pd.DataFrame({
18     'Estatura (m)': estaturas,
19     'Peso (kg)': pesos
20 })
21
```

## Métodos Utilizados

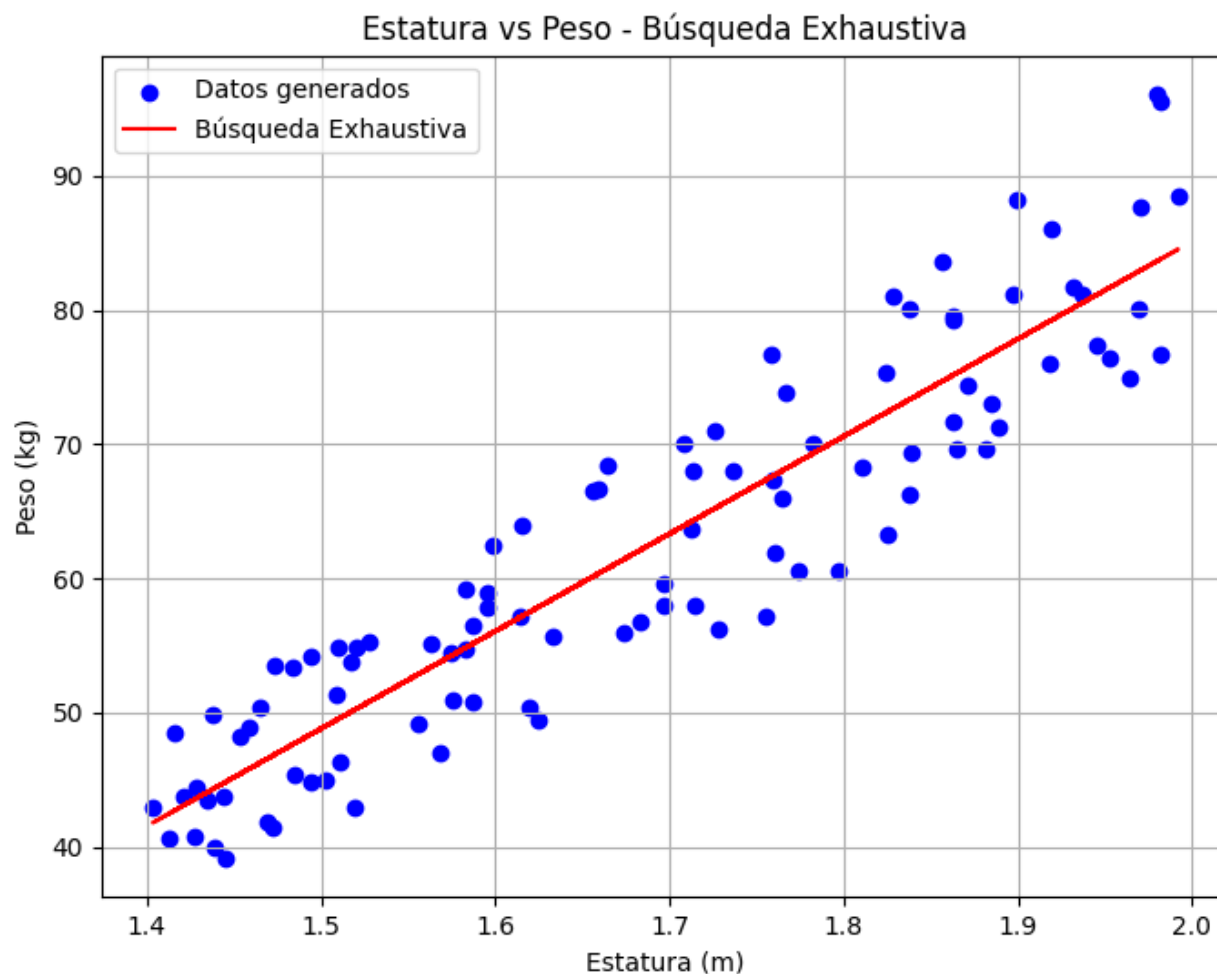
### 1. Búsqueda Exhaustiva

Utilizamos el método de búsqueda exhaustiva para encontrar los mejores parámetros de la recta de ajuste (mmm y bbb). Este método prueba todas las combinaciones posibles de pendiente e intercepto dentro de un rango definido para minimizar el error cuadrático medio (MSE). Este enfoque es exhaustivo pero puede ser lento debido al gran número de combinaciones probadas.

**Cálculo del MSE:**

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

```
1
2 # Método 1: Búsqueda Exhaustiva con Rango Adecuado
3 def busqueda_exhaustiva(x, y):
4     min_error = float('inf')
5     best_m = None
6     best_b = None
7     m_range = np.arange(50, 90, 0.1) # Pendiente variando de 50 a 90
8     b_range = np.arange(-70, -60, 0.1) # Intercepto variando de -70 a -60
9
10    for m_float in m_range:
11        for b_float in b_range:
12            error = np.sum((y - (m_float * x + b_float)) ** 2)
13            if error < min_error:
14                min_error = error
15                best_m = m_float
16                best_b = b_float
17    return best_m, best_b
18
19 x = datos['Estatura (m)']
20 y = datos['Peso (kg)']
21
22 best_m, best_b = busqueda_exhaustiva(x, y)
```



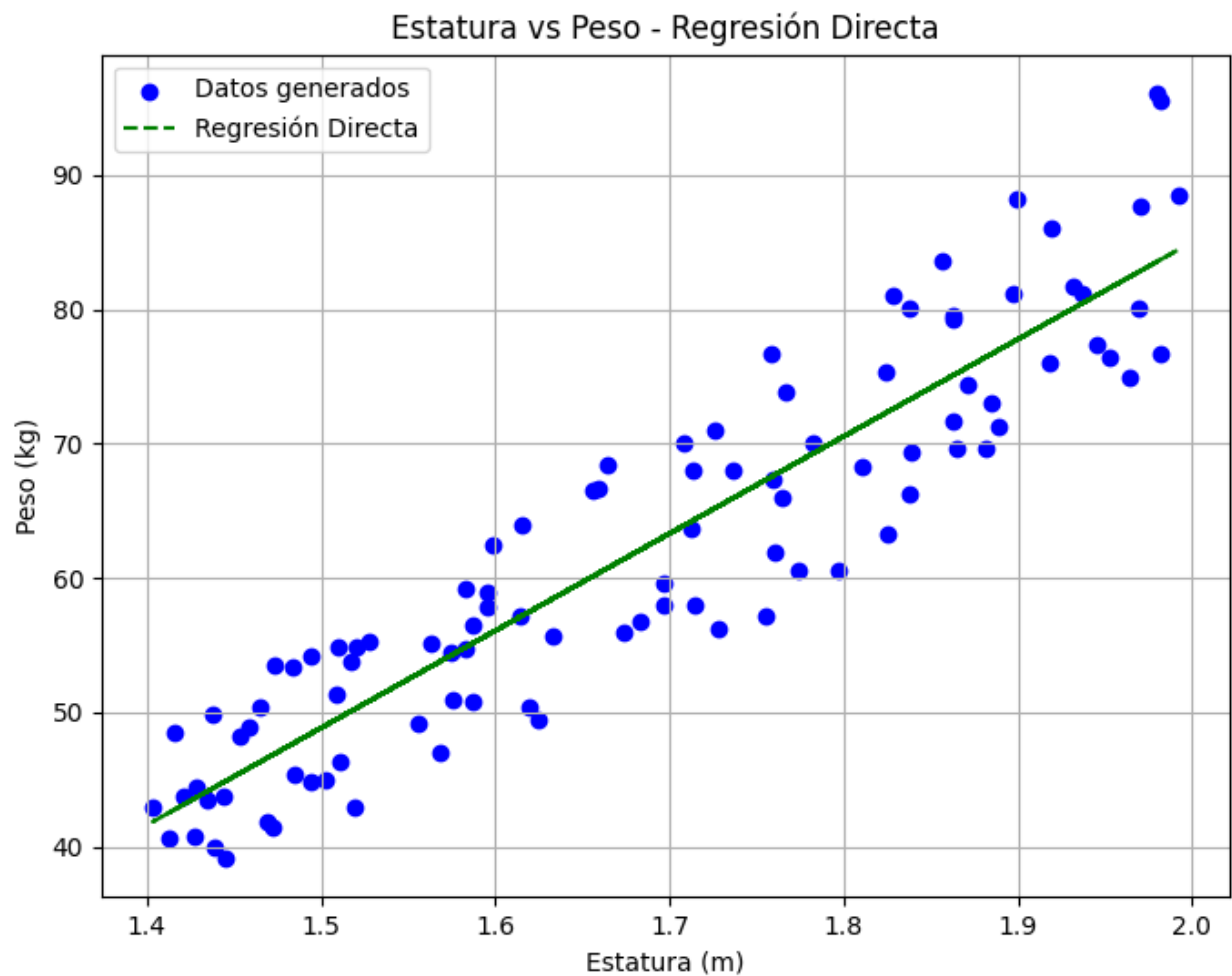
## 2. Regresión Lineal con Fórmulas Directas

Calculamos la pendiente (mmm) y el intercepto (bbb) usando fórmulas directas derivadas de la teoría de regresión lineal. Estas fórmulas se basan en los momentos estadísticos de los datos:

$$r = \frac{\sum((X - \bar{X})(Y - \bar{Y}))}{\sqrt{\sum(X - \bar{X})^2 \cdot \sum(Y - \bar{Y})^2}}$$



```
1 # Método 2: Regresión Lineal con Fórmulas Directas
2 m_directo = np.sum((x - np.mean(x)) * (y - np.mean(y))) / np.sum((x - np.mean(x)) ** 2)
3 b_directo = np.mean(y) - m_directo * np.mean(x)
4
```

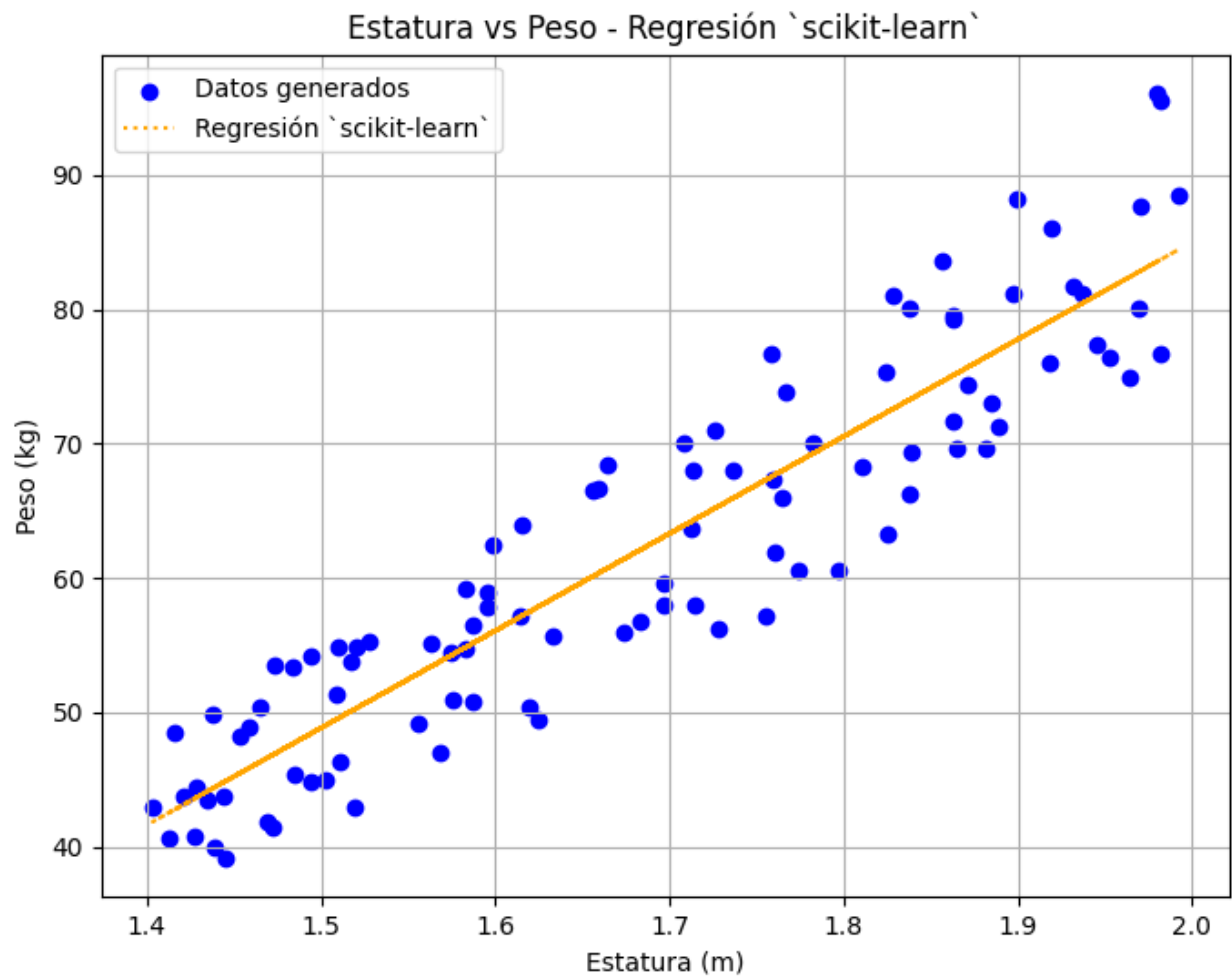


### 3. Regresión Lineal Usando scikit-learn

Utilizamos la biblioteca scikit-learn para realizar la regresión lineal. Esta biblioteca proporciona herramientas robustas para ajustar modelos y realizar predicciones.

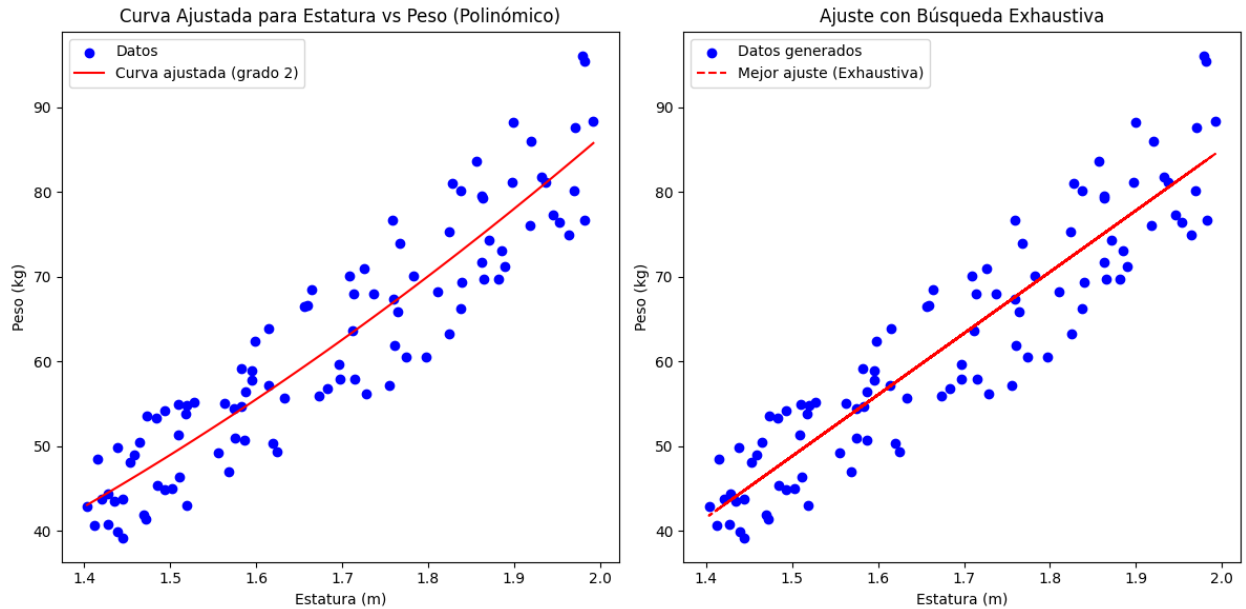


```
1 # Método 3: Regresión Lineal Usando `scikit-learn`  
2 modelo = LinearRegression()  
3 x_resaped = x.values.reshape(-1, 1) # `scikit-learn` requiere que x sea una matriz 2D  
4 modelo.fit(x_resaped, y)  
5 m_sklearn = modelo.coef_[0]  
6 b_sklearn = modelo.intercept_
```



### Visualización y Evaluación

- **Gráficas de Ajuste:** Creamos gráficos para visualizar los ajustes de los modelos:



- **Búsqueda Exhaustiva:** Muestra la línea ajustada utilizando la mejor combinación de  $m$  y  $b$  encontrados.
- **Regresión Directa:** Muestra el ajuste basado en fórmulas directas.
- **Regresión scikit-learn:** Muestra el ajuste usando la biblioteca scikit-learn.
- **Regresión Polinómica:** Muestra el ajuste de una curva cuadrática para comparar con el ajuste lineal.
- **Error Cuadrático Medio (MSE):** Calculamos el MSE para cada modelo para evaluar su precisión. El MSE nos da una idea de qué tan bien el modelo se ajusta a los datos reales.
- **Resultados Detallados:** Mostramos una tabla con una muestra de 20 datos que incluye las predicciones del modelo y los errores absolutos.

Muestra de Datos y Errores:				
	Estatura (m)	Peso (kg)	Predicción	Error
83	1.438135	49.875364	44.308602	5.566762
53	1.936896	81.155780	80.518679	0.637100
70	1.863347	79.289427	75.178982	4.110445
45	1.797513	60.537281	70.399471	9.862190
44	1.555268	49.158411	52.812456	3.654045
39	1.664091	68.452990	60.713043	7.739947
22	1.575287	50.958739	54.265821	3.307082
80	1.917862	76.075459	79.136785	3.061326
10	1.412351	40.601642	42.436661	1.835018
0	1.624724	49.365947	57.854968	8.489021
18	1.659167	66.652685	60.355525	6.297160