Programación para Analítica de Datos
Universidad Sergio Arboleda - Enero 2025

Taller 4
Docente : Guillermo De Mendoza

Total: 1 ejercicio

Implementación de Regresión Lineal usando la Fórmula Matricial en NumPy

Resultado esperado: Se espera como resultado del estudiante la implementación completa del algoritmo en un archivo Jupiter o .py

La regresión lineal es una técnica estadística utilizada para modelar la relación entre una variable dependiente “y” y una o más variables independientes “X”. En este taller, resolveremos el problema de regresión lineal simple utilizando la fórmula matricial de mínimos cuadrados:

$$\beta = (X^T X)^{-1} X^T y$$

Donde:

- X: es la matriz de características (también conocida como matriz de diseño).
- y: es el vector de valores observados (dependientes).
- β: es el vector de coeficientes (pendiente e intercepto) que queremos encontrar.

Tips:

-1 : invertir matriz

X^T : transpuesta de X

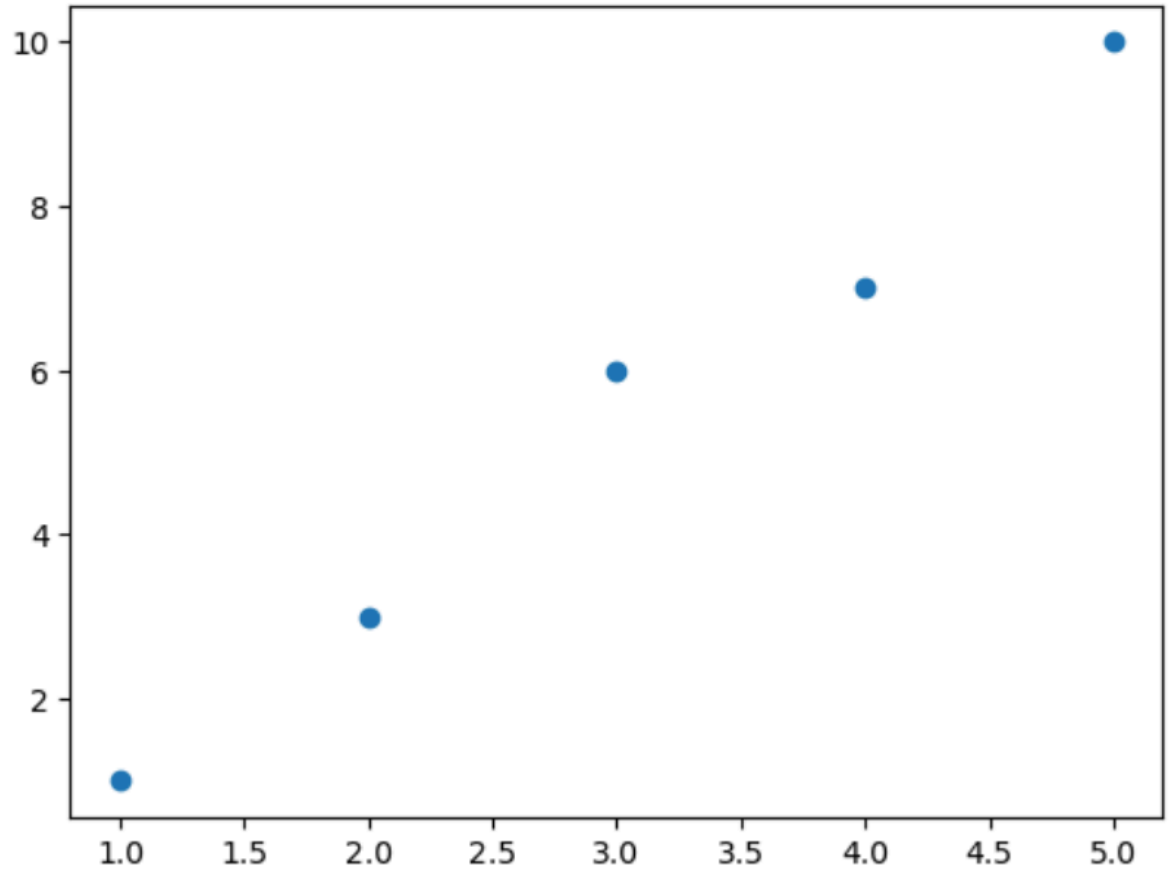
Datos:

```
x = np.array([1, 2, 3, 4, 5])
y = np.array([1, 3, 6, 7, 10])
```

```python
import numpy as np
import matplotlib.pyplot as plt

X = np.array([1, 2, 3, 4, 5])
y = np.array([1, 3, 6, 7, 10])

plt.scatter(X,y)
plt.show()
```



El objetivo es calcular el vector β, que representa los parámetros de la ecuación de la recta y=**m***x+**b**, donde **"m"** es la pendiente y **"b"** es el intercepto. Estos parámetros son obtenidos mediante el método de mínimos cuadrados, de modo que la recta resultante minimice el error cuadrático entre las predicciones del modelo y los puntos reales del conjunto de datos."

Donde **'x'** debe ser transformado a **'X'**, donde se incluye un valor de 1 al inicio de cada coordenada 'x'. Esto se hace para representar el término de intercepto, que corresponde al valor de la recta cuando x=0.

Para nuestro ejemplo los datos significaran:
x: horas de estudio
y: calificación

```python
import numpy as np
import matplotlib.pyplot as plt

# Datos de entrada (x) y salida (y)
x = np.array([1, 2, 3, 4, 5])
y = np.array([1, 3, 6, 7, 10])

# Crear la matriz de diseño X con una columna de 1's para el intercepto
X = np.vstack([np.ones_like(x), x]).T  # Agregar una columna de 1's para el intercepto

# Calcular los coeficientes beta usando la fórmula (X^T X)^-1 X^T y

beta = Aqui implementar la formula de BETA

# Los coeficientes beta representan [b, m] para la ecuación y = m * x + b
intercepto = beta[0]  # b (intercepto)
pendiente = beta[1]   # m (pendiente)

print(f"Coeficiente (intercepto) b: {intercepto}")
print(f"Pendiente m: {pendiente}")

# Predicciones con la recta de regresión
y_pred = np.dot( X , beta )  # Y = m * X + b

# Graficar los datos originales (puntos)
plt.scatter(x, y, color='blue', label='Datos', marker='o')

# Graficar la línea de regresión
plt.plot(x, y_pred, color='red', label=f'Regresión Lineal: y = {pendiente:.2f}x + {intercepto:.2f}', linewidth=2)

# Etiquetas y título
plt.xlabel('Horas de Estudio')
plt.ylabel('Calificación')
plt.title('Regresión Lineal: Horas de Estudio vs Calificación')

# Mostrar leyenda
plt.legend()

# Mostrar la gráfica
plt.grid(True)
plt.show()
```

```
Coeficiente (intercepto) b: -1.1999999999999955
Pendiente m: 2.200000000000001
```