

[Taller 05] Mínimos cuadrados

Richard Tipantiza

Tabla de Contenidos

A) Interpole los puntos:	3
UTILICE:	3
B) Interpole el siguiente conjunto de datos:	4
Conjunto de datos 1	4
Conjunto de datos 2	6

REPOSITORIO

```
# Derivadas parciales para regresión lineal
# #####
def der_parcial_1(xs: list, ys: list) -> tuple[float, float, float]:
    """Retorna los coeficientes de la ecuación de la derivada parcial con respecto al parámetro
     $c_1 * a_1 + c_0 * a_0 = c_{ind}$ 

    ## Parameters

    ``xs``: lista de valores de x.

    ``ys``: lista de valores de y.

    ## Return

    ``c_1``: coeficiente del parámetro 1.

    ``c_0``: coeficiente del parámetro 0.

    ``c_ind``: coeficiente del término independiente.
```

```

"""

# coeficiente del término independiente
c_ind = sum(ys)

# coeficiente del parámetro 1
c_1 = sum(xs)

# coeficiente del parámetro 0
c_0 = len(xs)

return (c_1, c_0, c_ind)

def der_parcial_0(xs: list, ys: list) -> tuple[float, float, float]:
    """Retorna los coeficientes de la ecuación de la derivada parcial con respecto al parámetro 0.
    
$$c_1 * a_1 + c_0 * a_0 = c_{ind}$$


    ## Parameters

    ``xs``: lista de valores de x.

    ``ys``: lista de valores de y.

    ## Return

    ``c_1``: coeficiente del parámetro 1.

    ``c_0``: coeficiente del parámetro 0.

    ``c_ind``: coeficiente del término independiente.

    """
    c_1 = 0
    c_0 = 0
    c_ind = 0
    for xi, yi in zip(xs, ys):
        # coeficiente del término independiente
        c_ind += xi * yi

        # coeficiente del parámetro 1

```

```

        c_1 += xi * xi

        # coeficiente del parámetro 0
        c_0 += xi

    return (c_1, c_0, c_ind)

```

A) Interpole los puntos:

p1 = (5.4, 3.2) p2_i = (9.5, 0.7) p3 = (12.3, -3.6)

from src import ajustar_min_cuadrados

UTILICE:

ajustar_min_cuadrados([5.4, 9.5, 12.3], [3.2, 0.7, -3.6], [der_parcial_1, der_parcial_0])

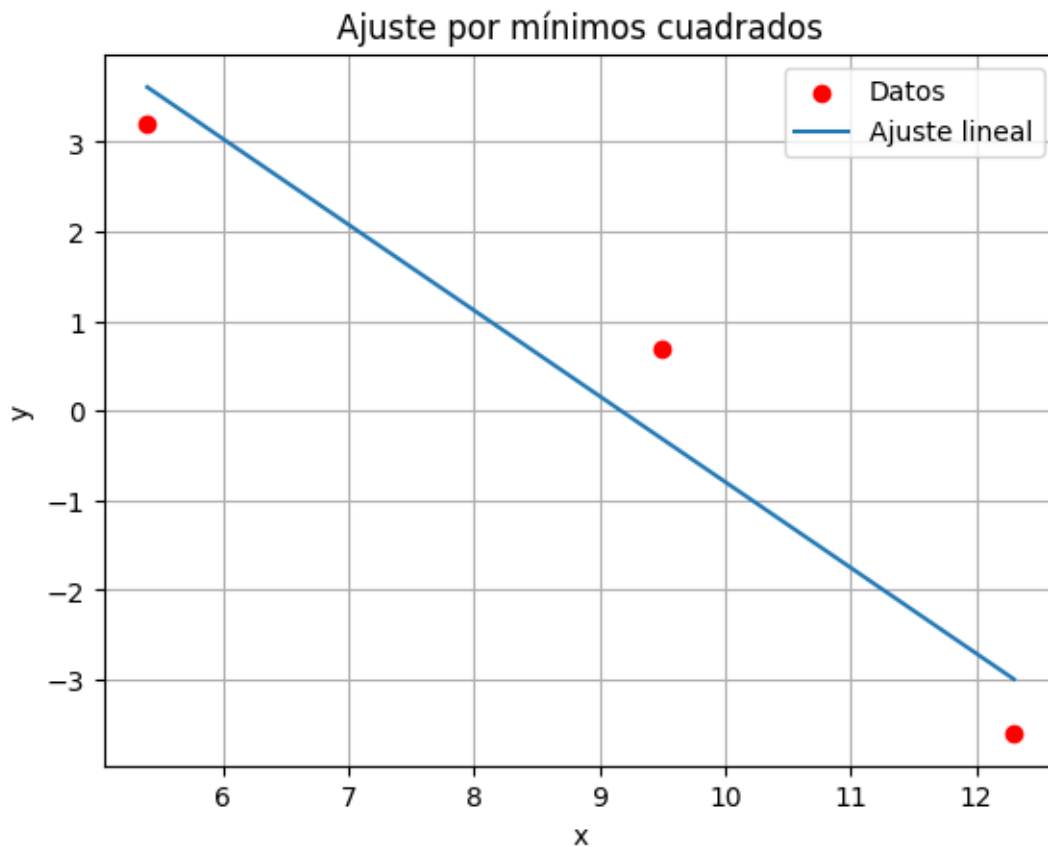
```

# graficar resultados
import matplotlib.pyplot as plt
import numpy as np

xs = np.array([5.4, 9.5, 12.3])
ys = np.array([3.2, 0.7, -3.6])
a1, a0 = ajustar_min_cuadrados([5.4, 9.5, 12.3], [3.2, 0.7, -3.6], [der_parcial_1, der_parcial_0])
plt.scatter(xs, ys, color='red', label='Datos')
plt.plot(xs, a1 * xs + a0, label='Ajuste lineal')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Ajuste por mínimos cuadrados')
plt.legend()
plt.grid()
plt.show()

```

[12-15 17:17:41] [INFO] Se ajustarán 2 parámetros.



B) Interpole el siguiente conjunto de datos:

Conjunto de datos 1

```
xs = [ 3.38,  
0.35,  
2.07,  
-0.45,  
-0.55,  
-1.06,  
-2.77,  
3.08,  
-3.98,  
-5,  
-3.18,
```

```

-1.96,
2.37,
-1.66,
4.09]

ys = [-0.04,
1.28,
0.65,
2.18,
1.70,
1.96,
3.36,
0.18,
3.35,
4.16,
2.95,
2.34,
0.38,
1.75,
-1.06]

```

```

# UTILICE:
ajustar_min_cuadrados(xs, ys, [der_parcial_1, der_parcial_0])

```

[12-15 17:21:00] [INFO] Se ajustarán 2 parámetros.

```
array([-0.50323259,  1.49919762])
```

```

xs_np = np.array(xs)
ys_np = np.array(ys)

# Calculamos los coeficientes del ajuste
a1, a0 = ajustar_min_cuadrados(xs, ys, [der_parcial_1, der_parcial_0])
label_ajuste = f'Ajuste lineal: y = {a1:.4f}x + {a0:+.4f}'

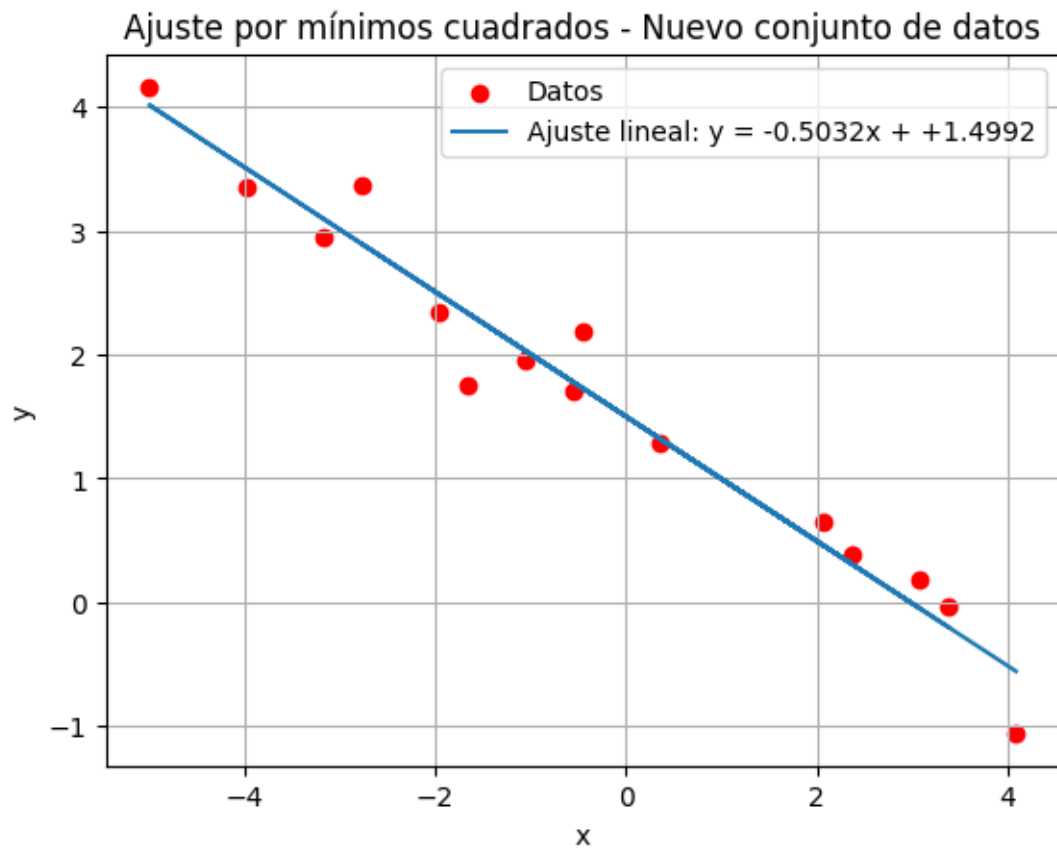
# Graficamos los puntos originales
plt.scatter(xs_np, ys_np, color='red', label='Datos')
plt.plot(xs_np, a1 * xs_np + a0, label=label_ajuste)

plt.xlabel('x')
plt.ylabel('y')

```

```
plt.title('Ajuste por mínimos cuadrados - Nuevo conjunto de datos')
plt.legend()
plt.grid()
plt.show()
```

[12-15 17:25:01] [INFO] Se ajustarán 2 parámetros.



Conjunto de datos 2

```
xs2=[ 3.38 ,0.35,  2.07, -0.45, -0.56, -1.06, -2.78,  3.08, -3.99, -5, -3.18, -1.97, 2.37, -
ys2=[ 15.65, -11.2, -4, 18.03,  7.94, 15.32, -4.4,  1.39, -11.92, -90.24, 6.92, 28.35,
```

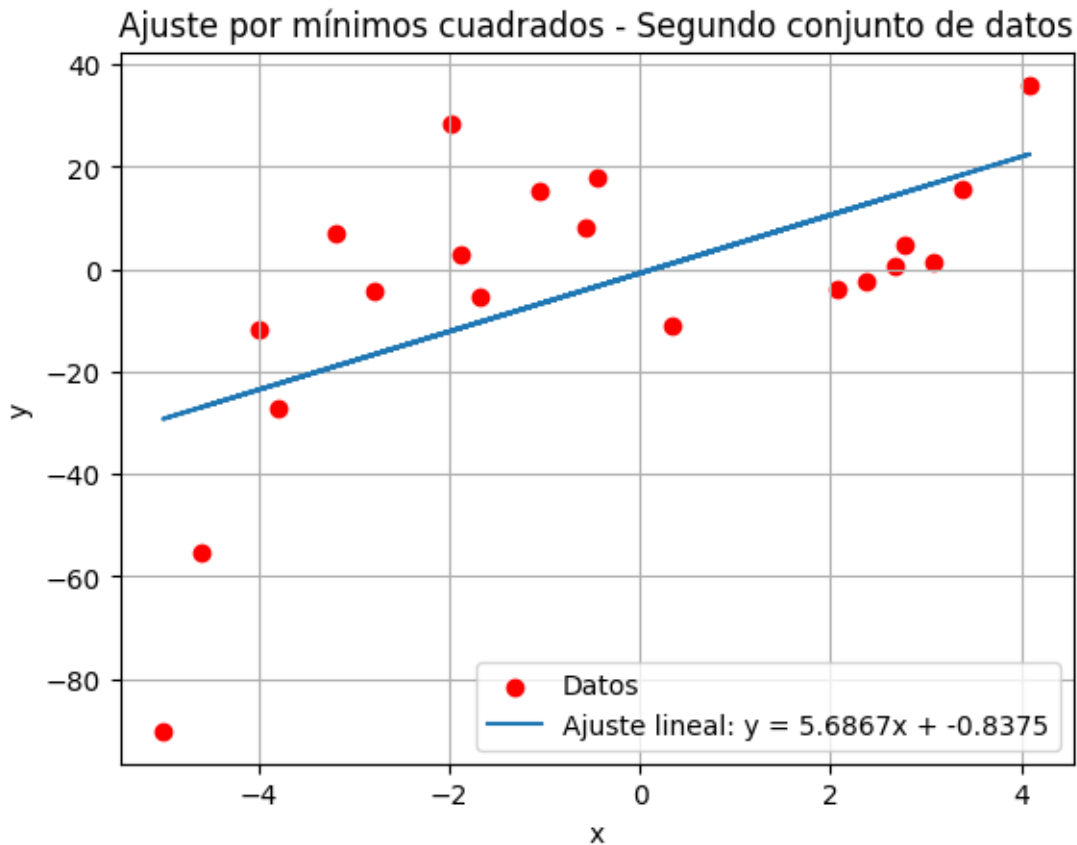
```
# UTILICE:  
ajustar_min_cuadrados(xs2, ys2, [der_parcial_1, der_parcial_0])
```

[12-15 17:30:03] [INFO] Se ajustarán 2 parámetros.

```
array([ 5.68666556, -0.83754722])
```

```
# graficar resultados  
import matplotlib.pyplot as plt  
import numpy as np  
xs2_np = np.array(xs2)  
ys2_np = np.array(ys2)  
a1, a0 = ajustar_min_cuadrados(xs2, ys2, [der_parcial_1, der_parcial_0])  
label_ajuste = f'Ajuste lineal: y = {a1:.4f}x + {a0:+.4f}'  
  
plt.scatter(xs2_np, ys2_np, color='red', label='Datos')  
plt.plot(xs2_np, a1 * xs2_np + a0, label=label_ajuste)  
plt.xlabel('x')  
plt.ylabel('y')  
plt.title('Ajuste por mínimos cuadrados - Segundo conjunto de datos')  
plt.legend()  
plt.grid()  
plt.show()
```

[12-15 17:31:04] [INFO] Se ajustarán 2 parámetros.



```
import numpy as np
def der_parcial_0(xs: list, ys: list) -> tuple[float, float, float, float, float]:
    xs_np = np.array(xs)
    ys_np = np.array(ys)

    c_ind = np.sum(ys_np * xs_np**3)
    c_a = np.sum(xs_np**6)
    c_b = np.sum(xs_np**5)
    c_c = np.sum(xs_np**4)
    c_d = np.sum(xs_np**3)

    return (c_a, c_b, c_c, c_d, c_ind)

def der_parcial_1(xs: list, ys: list) -> tuple[float, float, float, float, float]:
    xs_np = np.array(xs)
    ys_np = np.array(ys)
```

```

    c_ind = np.sum(ys_np * xs_np**2)

    c_a = np.sum(xs_np**5)
    c_b = np.sum(xs_np**4)
    c_c = np.sum(xs_np**3)
    c_d = np.sum(xs_np**2)

    return (c_a, c_b, c_c, c_d, c_ind)

def der_parcial_2(xs: list, ys: list) -> tuple[float, float, float, float, float]:

    xs_np = np.array(xs)
    ys_np = np.array(ys)

    c_ind = np.sum(ys_np * xs_np)
    c_a = np.sum(xs_np**4)
    c_b = np.sum(xs_np**3)
    c_c = np.sum(xs_np**2)
    c_d = np.sum(xs_np)

    return (c_a, c_b, c_c, c_d, c_ind)

def der_parcial_3(xs: list, ys: list) -> tuple[float, float, float, float, float]:

    c_ind = sum(ys)

    c_a = np.sum(np.array(xs)**3)
    c_b = np.sum(np.array(xs)**2)
    c_c = np.sum(xs)
    c_d = len(xs)

    return (c_a, c_b, c_c, c_d, c_ind)

from src import ajustar_min_cuadrados

xs2=[ 3.38 ,0.35,  2.07, -0.45, -0.56, -1.06, -2.78,  3.08, -3.99, -5, -3.18, -1.97, 2.37, -
ys2=[ 15.65, -11.2, -4, 18.03,  7.94,  15.32, -4.4,  1.39, -11.92, -90.24, 6.92,  28.35,

# UTILICE:
ajustar_min_cuadrados(xs2, ys2, [der_parcial_0, der_parcial_1, der_parcial_2, der_parcial_3])

```

[12-15 21:55:25] [INFO] Se ajustarán 4 parámetros.

```
array([ 1.04497792,  0.03132741, -8.88066397,  2.76228992])
```

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

a3, a2, a1, a0 = ajustar_min_cuadrados(xs2, ys2, [der_parcial_0, der_parcial_1, der_parcial_2])

xs2_np = np.array(xs2)
ys2_np = np.array(ys2)

x_fit = np.linspace(xs2_np.min(), xs2_np.max(), 100)
y_fit = a3 * x_fit**3 + a2 * x_fit**2 + a1 * x_fit + a0

plt.figure(figsize=(10, 6))

plt.scatter(xs2_np, ys2_np, color='red', label='Datos originales')
label_ajuste = f'Ajuste cúbico:  $y = \{a3:.4f\}x^3 + \{a2:+.4f\}x^2 + \{a1:+.4f\}x + \{a0:+.4f\}$ '
plt.plot(x_fit, y_fit, color='blue', label=label_ajuste)

plt.xlabel('x')
plt.ylabel('y')
plt.title('Ajuste de Polinomio Cúbico por Mínimos Cuadrados (Gráfico Corregido)')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```

[12-15 22:01:11] [INFO] Se ajustarán 4 parámetros.

