ESCEULA POLITECNICA NACIONAL

Materia:Métodos Numéricos

Nombre: Stiv Quishpe

Link repositorio

https://github.com/stiv001/Tarea-08.git

Ejercicios Unidad 03-C mínimos cuadrados

Conjunto de Ejercicios

1. Dados los datos:

x_i	4.0	4.2	4.5	4.7	5.1	5.5	5.9	6.3	6.8	7.1
y_i	102.56	130.11	113.18	142.05	167.53	195.14	224.87	256.73	299.50	326.72

a. Construya el polinomio por mínimos cuadrados de grado 1 y calcule el error.
b. Construya el polinomio por mínimos cuadrados de grado 2 y calcule el error.
c. Construya el polinomio por mínimos cuadrados de grado 3 y calcule el error.
d. Construya el polinomio por mínimos cuadrados de la forma be^{ax} y calcule el error.
e. Construya el polinomio por mínimos cuadrados de la forma bx^a y calcule el error.

Figure 1: image.png

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

x = np.array([4.0, 4.2, 4.5, 4.7, 5.1, 5.5, 5.9, 6.3, 6.8, 7.1])
y = np.array([102.56, 130.11, 113.18, 142.05, 167.53, 195.14, 224.87, 256.73, 299.50, 326.72
```

```
def calcular_error(y_real, y_pred):
    return np.mean((y_real - y_pred) ** 2)

coef_1 = np.polyfit(x, y, 1)
pol_1 = np.poly1d(coef_1)
y_pred_1 = pol_1(x)
error_1 = calcular_error(y, y_pred_1)

coef_2 = np.polyfit(x, y, 2)
pol_2 = np.poly1d(coef_2)
y_pred_2 = pol_2(x)
error_2 = calcular_error(y, y_pred_2)

coef_3 = np.polyfit(x, y, 3)
pol_3 = np.poly1d(coef_3)
y_pred_3 = pol_3(x)
error_3 = calcular_error(y, y_pred_3)
```

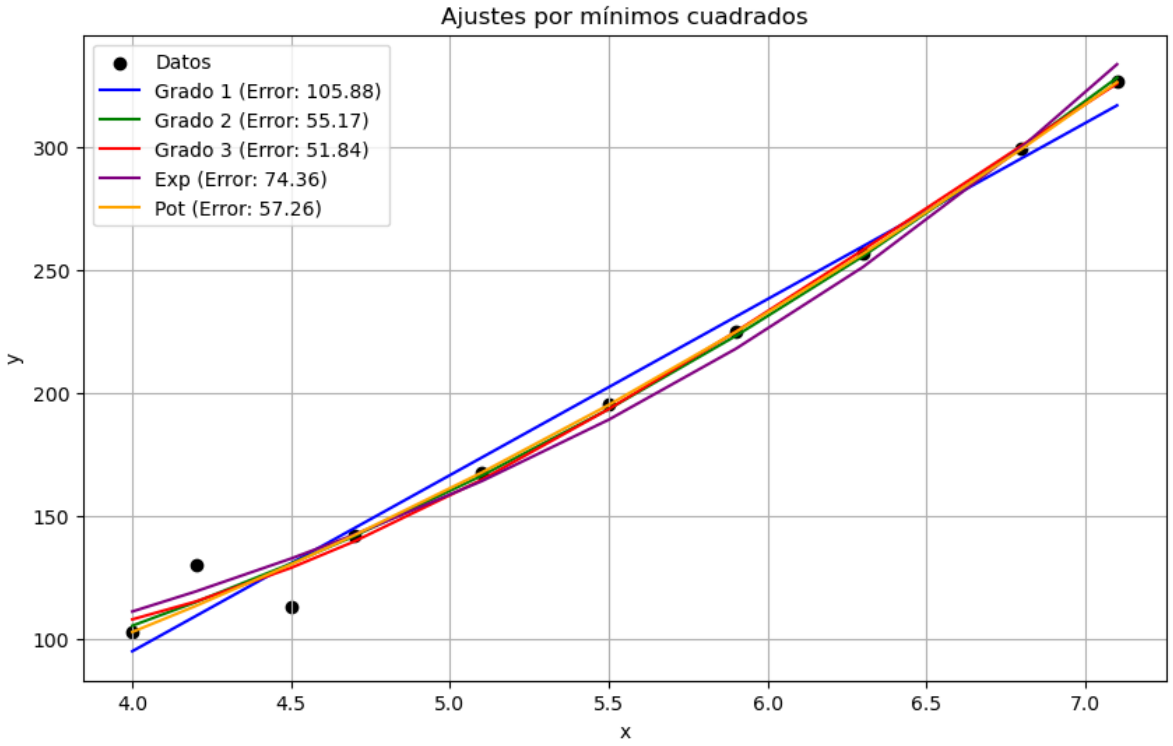
```
def modelo_exp(x, a, b):
    return b * np.exp(a * x)

popt_exp, _ = curve_fit(modelo_exp, x, y, maxfev=10000)
y_pred_exp = modelo_exp(x, *popt_exp)
error_exp = calcular_error(y, y_pred_exp)

def modelo_pot(x, a, b):
    return b * x**a
```

```
popt_pot, _ = curve_fit(modelo_pot, x, y, maxfev=10000)
y_pred_pot = modelo_pot(x, *popt_pot)
error_pot = calcular_error(y, y_pred_pot)

print("Polinomio de grado 1:\n", pol_1)
print("Error: ", error_1, "\n")

print("Polinomio de grado 2:\n", pol_2)
print("Error: ", error_2, "\n")

print("Polinomio de grado 3:\n", pol_3)
print("Error: ", error_3, "\n")
```

```
print("Polinomio de la forma b * exp(a * x):")
print(f"a = {popt_exp[0]:.4f}, b = {popt_exp[1]:.4f}")
print("Error: ", error_exp, "\n")

print("Polinomio de la forma b * x^a:")
print(f"a = {popt_pot[0]:.4f}, b = {popt_pot[1]:.4f}")
print("Error: ", error_pot, "\n")

plt.figure(figsize=(10, 6))
plt.scatter(x, y, label='Datos', color='black')
plt.plot(x, y_pred_1, label=f'Grado 1 (Error: {error_1:.2f})', color='blue')
plt.plot(x, y_pred_2, label=f'Grado 2 (Error: {error_2:.2f})', color='green')
plt.plot(x, y_pred_3, label=f'Grado 3 (Error: {error_3:.2f})', color='red')
plt.plot(x, y_pred_exp, label=f'Exp (Error: {error_exp:.2f})', color='purple')
plt.plot(x, y_pred_pot, label=f'Pot (Error: {error_pot:.2f})', color='orange')

plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid()
plt.title('Ajustes por mínimos cuadrados')
plt.show()
```

```
Polinomio de grado 1:

71.61 x - 191.6
Error:  105.88388862638897

Polinomio de grado 2:
        2
8.217 x - 19.31 x + 51
Error:  55.16562001170242

Polinomio de grado 3:
         3         2
-2.607 x + 51.56 x - 254.9 x + 469.2
Error:  51.83830647403022

Polinomio de la forma b * exp(a * x):
a = 0.3549, b = 26.8408
Error:  74.36215913505346
```

```
Polinomio de la forma b * x^a:
a = 2.0152, b = 6.2840
Error:  57.25817542056742
```



2. Repita el ejercicio 5 para los siguientes datos.

x_i	0.2	0.3	0.6	0.9	1.1	1.3	1.4	1.6
y_i	0.050446	0.098426	0.33277	0.72660	1.0972	1.5697	1.8487	2.5015

Figure 2: image.png

```python
x2 = np.array([0.2, 0.3, 0.6, 0.9, 1.1, 1.3, 1.4, 1.6])
y2 = np.array([0.050446, 0.098426, 0.33277, 0.72660, 1.0972, 1.5697, 1.8487, 2.5015])

coef1_2 = np.polyfit(x2, y2, 1)
y2_pred1 = np.polyval(coef1_2, x2)
err1_2 = calcular_error(y2, y2_pred1)

print("a) Polinomio grado 1:")
```

```
print(f"   Ecuación: y = {coef1_2[0]:.4f}x + {coef1_2[1]:.4f}")
print(f"   Error: {err1_2:.4f}\n")
```

```
a) Polinomio grado 1:
   Ecuación: y = 1.6655x + -0.5125
   Error: 0.0419
```

```
coef2_2 = np.polyfit(x2, y2, 2)
y2_pred2 = np.polyval(coef2_2, x2)
err2_2 = calcular_error(y2, y2_pred2)

print("b) Polinomio grado 2:")
print(f"   Ecuación: y = {coef2_2[0]:.4f}x² + {coef2_2[1]:.4f}x + {coef2_2[2]:.4f}")
print(f"   Error: {err2_2:.4f}\n")
```

```
b) Polinomio grado 2:
   Ecuación: y = 1.1294x² + -0.3114x + 0.0851
   Error: 0.0003
```

```
coef3_2 = np.polyfit(x2, y2, 3)
y2_pred3 = np.polyval(coef3_2, x2)
err3_2 = calcular_error(y2, y2_pred3)
print("c) Polinomio grado 3:")
print(f"   Ecuación: y = {coef3_2[0]:.4f}x³ + {coef3_2[1]:.4f}x² + {coef3_2[2]:.4f}x + {coef3
print(f"   Error: {err3_2:.4f}\n")
```

```
c) Polinomio grado 3:
   Ecuación: y = 0.2662x³ + 0.4029x² + 0.2484x + -0.0184
   Error: 0.0000
```

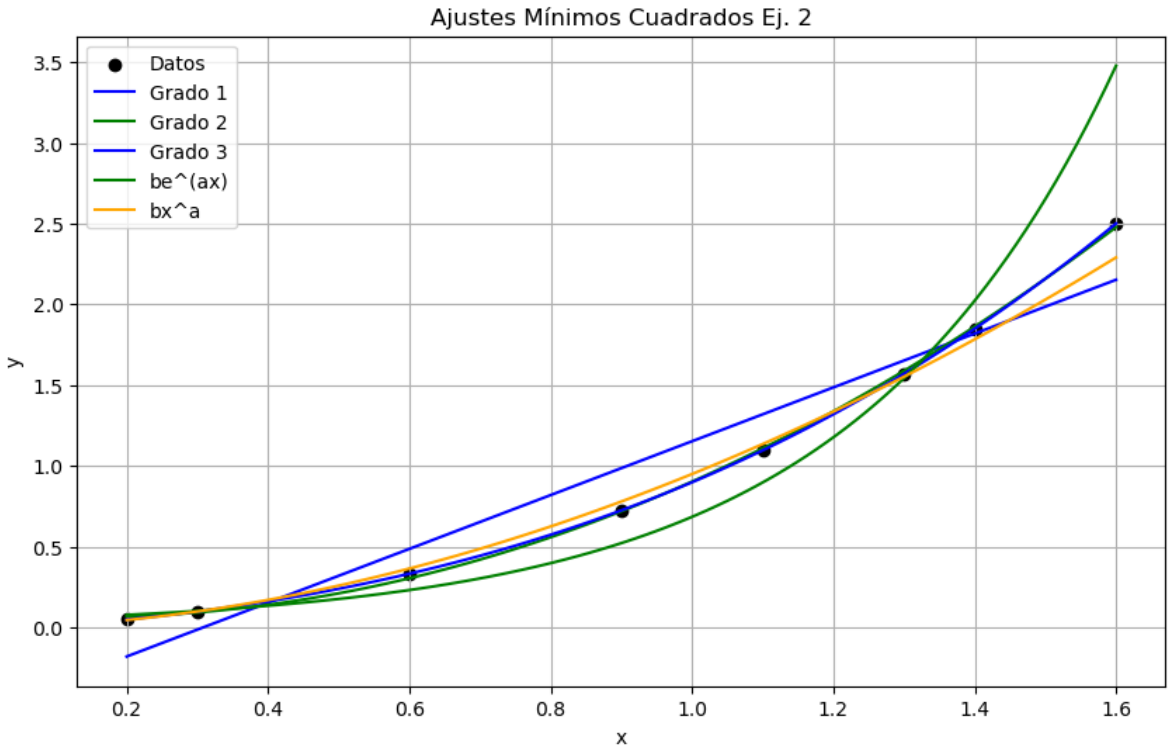
```
lny2 = np.log(y2)
coef_exp2 = np.polyfit(x2, lny2, 1)
a_exp2 = coef_exp2[0]
lnb_exp2 = coef_exp2[1]
b_exp2 = np.exp(lnb_exp2)
y2_pred_exp = b_exp2 * np.exp(a_exp2*x2)
err_exp2 = calcular_error(y2, y2_pred_exp)
print("d) Ajuste exponencial y = b e^{a x}:")
print(f"   a = {a_exp2:.4f}, b = {b_exp2:.4f}")
print(f"   Ecuación: y = {b_exp2:.4f} e^({a_exp2:.4f}x)")
print(f"   Error: {err_exp2:.4f}\n")
```

```
d) Ajuste exponencial y = b e^{a x}:
   a = 2.7073, b = 0.0457
   Ecuación: y = 0.0457 e^(2.7073x)
   Error: 0.1344
```

```
lnx2 = np.log(x2)
coef_pow2 = np.polyfit(lnx2, lny2, 1)
a_pow2 = coef_pow2[0]
lnb_pow2 = coef_pow2[1]
b_pow2 = np.exp(lnb_pow2)
y2_pred_pow = b_pow2*(x2**a_pow2)
err_pow2 = calcular_error(y2, y2_pred_pow)
print("e) Ajuste potencial y = b x^{a}:")
print(f"   a = {a_pow2:.4f}, b = {b_pow2:.4f}")
print(f"   Ecuación: y = {b_pow2:.4f} x^{a_pow2:.4f}")
print(f"   Error: {err_pow2:.4f}\n")
```

```
e) Ajuste potencial y = b x^{a}:
   a = 1.8720, b = 0.9502
   Ecuación: y = 0.9502 x^1.8720
   Error: 0.0068
```

```
plt.figure(figsize=(10,6))
plt.scatter(x2, y2, label='Datos', color='black')
x2_plot = np.linspace(min(x2), max(x2), 100)
plt.plot(x2_plot, np.polyval(coef1_2, x2_plot), label=f'Grado 1', color='b')
plt.plot(x2_plot, np.polyval(coef2_2, x2_plot), label=f'Grado 2', color='g')
plt.plot(x2_plot, np.polyval(coef3_2, x2_plot), label=f'Grado 3', color='b')
plt.plot(x2_plot, b_exp2*np.exp(a_exp2*x2_plot), label=f'be^(ax)', color='g')
plt.plot(x2_plot, b_pow2*(x2_plot**a_pow2), label=f'bx^a', color='orange')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Ajustes Mínimos Cuadrados Ejercicio 2')
plt.legend()
plt.grid(True)
plt.show()
```



3. La siguiente tabla muestra los promedios de puntos del colegio de 20 especialistas en matemáticas y ciencias computacionales, junto con las calificaciones que recibieron estos estudiantes en la parte de matemáticas de la prueba ACT (Programa de Pruebas de Colegios Americanos) mientras estaban en secundaria. Grafique estos datos y encuentre la ecuación de la recta por mínimos cuadrados para estos datos.

Puntuación ACT	Promedio de puntos	Puntuación ACT	Promedio de puntos
28	3.84	29	3.75
25	3.21	28	3.65
28	3.23	27	3.87
27	3.63	29	3.75
28	3.75	21	1.66
33	3.20	28	3.12
28	3.41	28	2.96
29	3.38	26	2.92
23	3.53	30	3.10
27	2.03	24	2.81

Figure 3: image.png

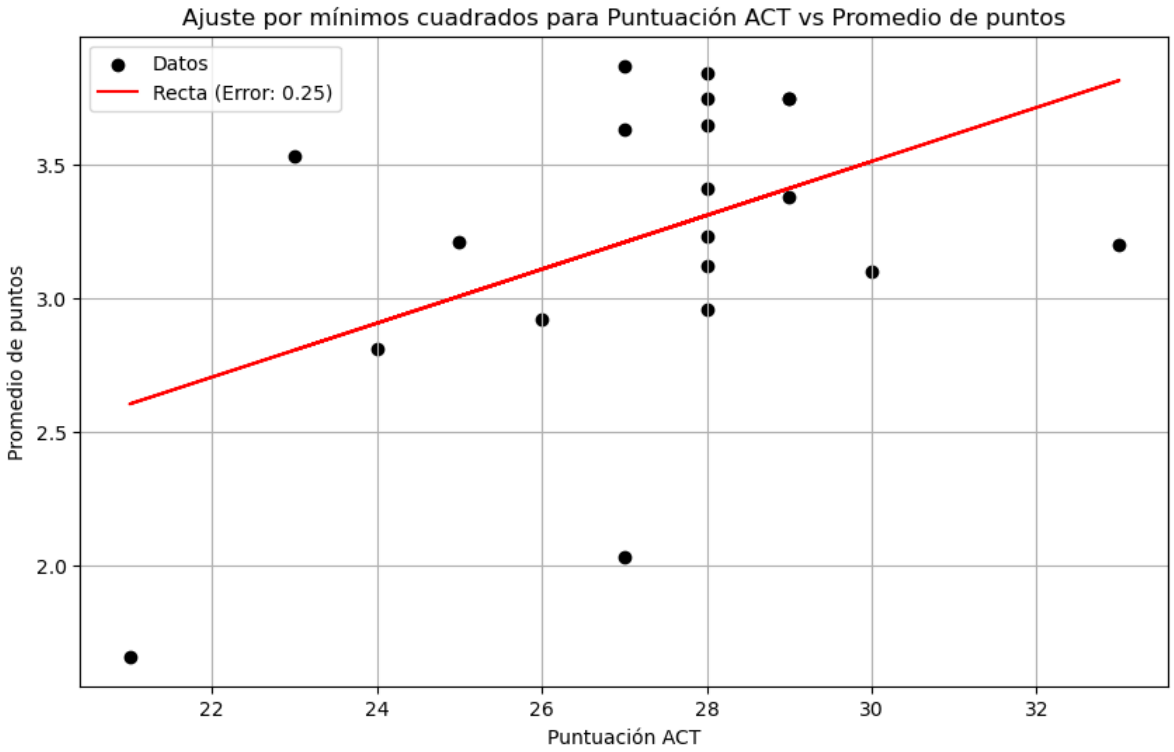
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
```

```
x = np.array([28, 25, 28, 27, 28, 33, 28, 29, 23, 27, 29, 28, 27, 29, 21, 28, 28, 26, 30, 24
y = np.array([3.84, 3.21, 3.23, 3.63, 3.75, 3.20, 3.41, 3.38, 3.53, 2.03, 3.75, 3.65, 3.87, 3
```

```
def calcular_error(y_real, y_pred):
    return np.mean((y_real - y_pred) ** 2)

coef_1 = np.polyfit(x, y, 1)
pol_1 = np.poly1d(coef_1)
y_pred_1 = pol_1(x)
error_1 = calcular_error(y, y_pred_1)
```

```
print("Polinomio de grado 1 (recta):")
print(pol_1)
print("Error:", error_1)

plt.figure(figsize=(10, 6))
plt.scatter(x, y, label='Datos', color='black')
plt.plot(x, y_pred_1, label=f'Recta (Error: {error_1:.2f})', color='red')
plt.xlabel('Puntuación ACT')
plt.ylabel('Promedio de puntos')
plt.legend()
plt.grid()
plt.title('Ajuste por mínimos cuadrados para Puntuación ACT vs Promedio de puntos')
plt.show()
```

```
Polinomio de grado 1 (recta):

0.1009 x + 0.4866
Error: 0.2524352808112324
```



4. El siguiente conjunto de datos, presentado al Subcomité Antimonopolio del Senado, muestra las características comparativas de supervivencia durante un choque de automóviles de diferentes clases. Encuentre la recta por mínimos cuadrados que aproxima estos datos (la tabla muestra el porcentaje de vehículos que participaron en un accidente en los que la lesión más grave fue fatal o seria).

Tipo	Peso promedio	Porcentaje de presentación
1. Regular lujoso doméstico	4800 lb	3.1
2. Regular intermediario doméstico	3700 lb	4.0
3. Regular económico doméstico	3400 lb	5.2
4. Compacto doméstico	2800 lb	6.4
5. Compacto extranjero	1900 lb	9.6

Figure 4: image.png

```
x_peso = np.array([4800, 3700, 3400, 2800, 1900])
y_porcentaje = np.array([3.1, 4.0, 5.2, 6.4, 9.6])

coef_lin_4 = np.polyfit(x_peso, y_porcentaje, 1)
y_pred_4 = np.polyval(coef_lin_4, x_peso)
err_4 = calcular_error(y_porcentaje, y_pred_4)
```

```python
print("Ajuste lineal:")
print(f"   Ecuación: y = {coef_lin_4[0]:.4f}x + {coef_lin_4[1]:.4f}")
print(f"   Error: {err_4:.4f}\n")

plt.figure(figsize=(10,6))
plt.scatter(x_peso, y_porcentaje, label='Datos', color='black')
x_plot_4 = np.linspace(min(x_peso), max(x_peso), 100)
plt.plot(x_plot_4, np.polyval(coef_lin_4, x_plot_4), label='Ajuste lineal', color='r')
plt.xlabel('Peso (lb)')
plt.ylabel('Porcentaje de presentación')
plt.title('Ajuste Mínimos Cuadrados - Ejercicio 4')
plt.legend()
plt.grid(True)
plt.show()
```

```
Ajuste lineal:
   Ecuación: y = -0.0023x + 13.1465
   Error: 0.4118
```