Métodos Numericos

José Sarango

Tabla de Contenidos

Conjunto de ejercicios

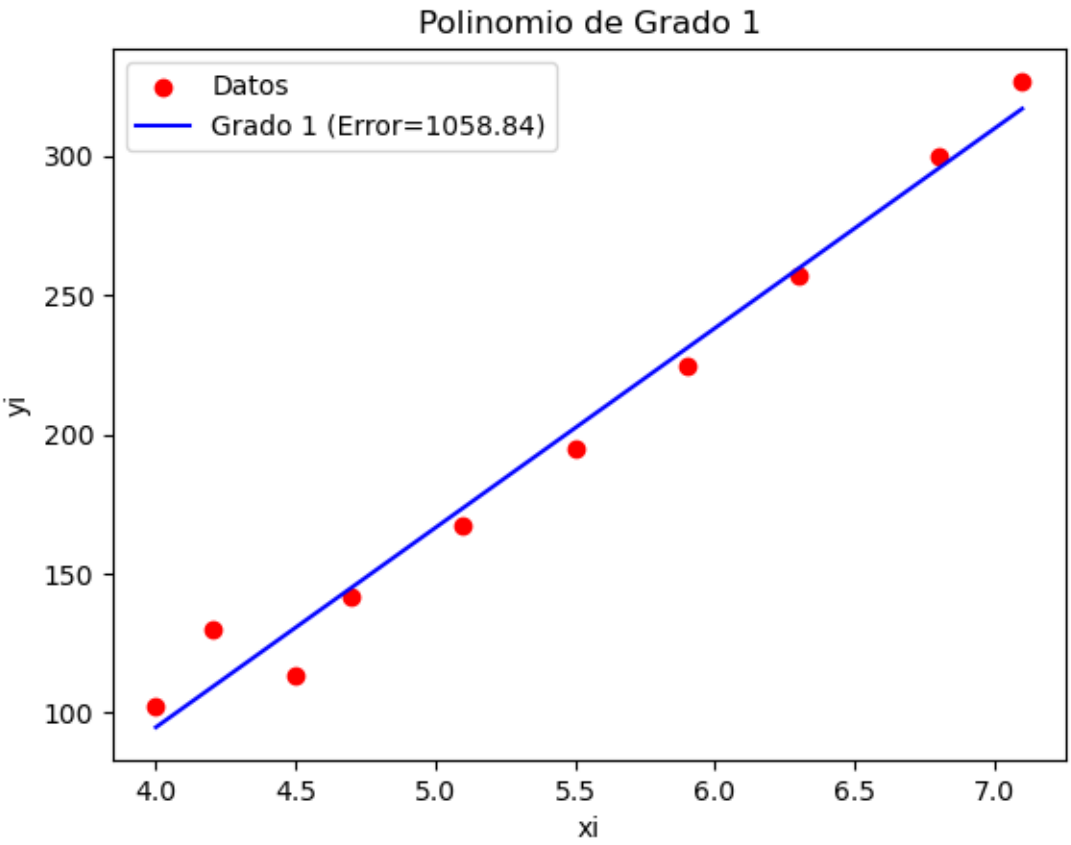
1. Dados los datos:

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

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

# Datos del ejercicio
xi = np.array([4.0, 4.2, 4.5, 4.7, 5.1, 5.5, 5.9, 6.3, 6.8, 7.1])
yi = np.array([102.56, 130.11, 113.18, 142.05, 167.53, 195.14, 224.87, 256.73, 299.50, 326.72
```

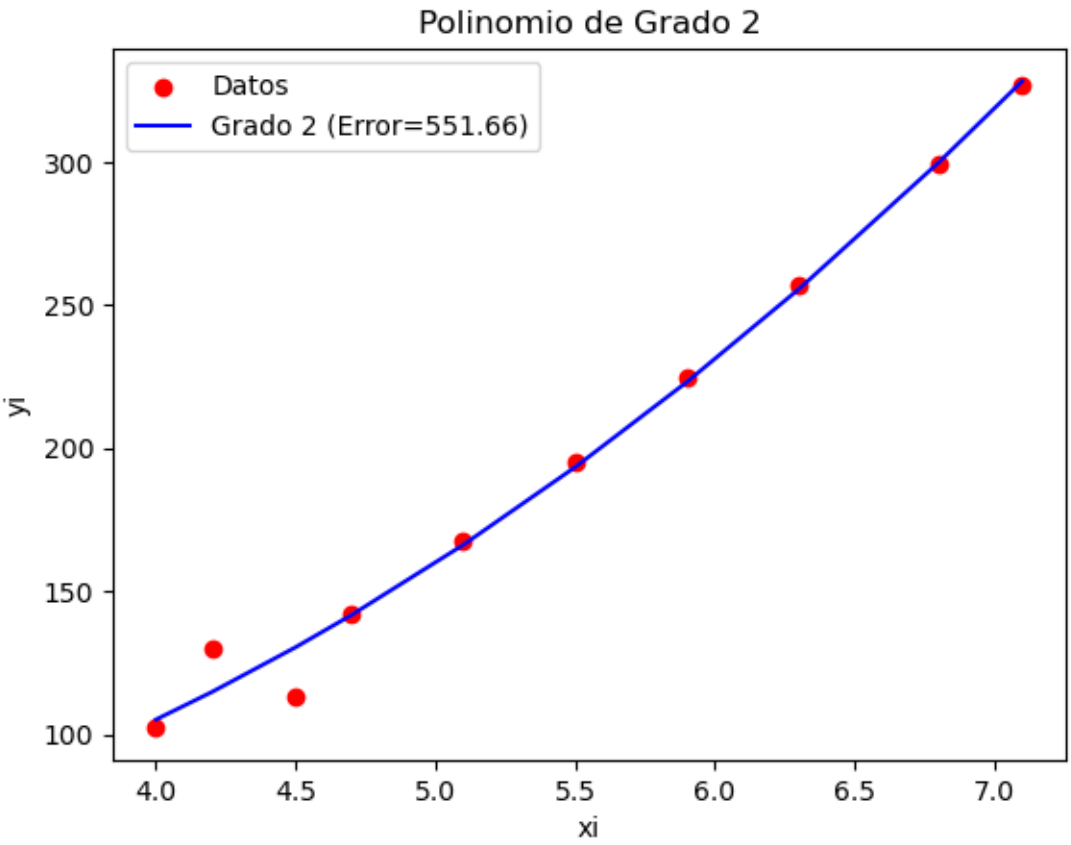
a. Construya el polinomio por mínimos cuadrados de grado 1 y calcule el error.

```python
coef1 = np.polyfit(xi, yi, 1) # Grado 1
poly1 = np.poly1d(coef1)
error1 = np.sum((poly1(xi) - yi) ** 2)

plt.scatter(xi, yi, color='red', label='Datos')
plt.plot(xi, poly1(xi), label=f'Grado 1 (Error={error1:.2f})', color='blue')
plt.xlabel('xi')
plt.ylabel('yi')
plt.legend()
plt.title('Polinomio de Grado 1')
plt.show()
```



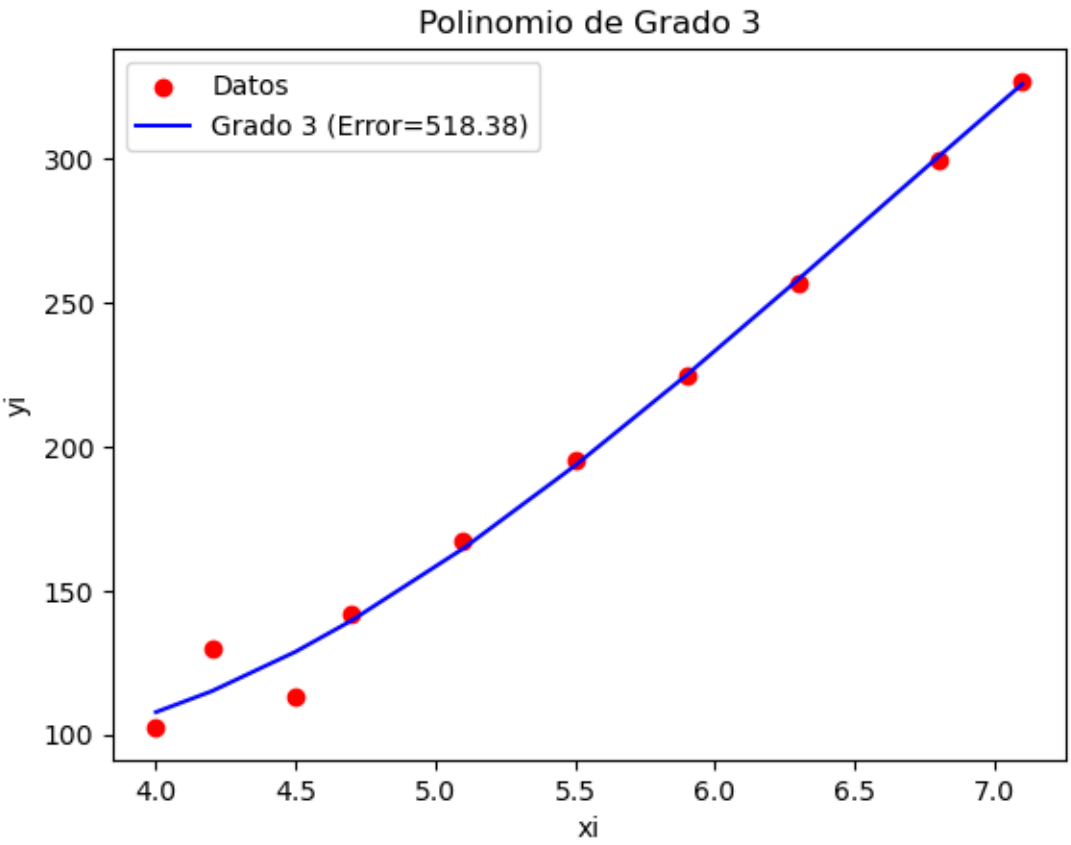
b. Construya el polinomio por mínimos cuadrados de grado 2 y calcule el error.

```
# Grado 2
coef2 = np.polyfit(xi, yi, 2)
poly2 = np.poly1d(coef2)
error2 = np.sum((poly2(xi) - yi) ** 2)

# Graficar
plt.scatter(xi, yi, color='red', label='Datos')
plt.plot(xi, poly2(xi), label=f'Grado 2 (Error={error2:.2f})', color='blue')
plt.xlabel('xi')
plt.ylabel('yi')
plt.legend()
plt.title('Polinomio de Grado 2')
plt.show()
```



c. Construya el polinomio por mínimos cuadrados de grado 3 y calcule el error.

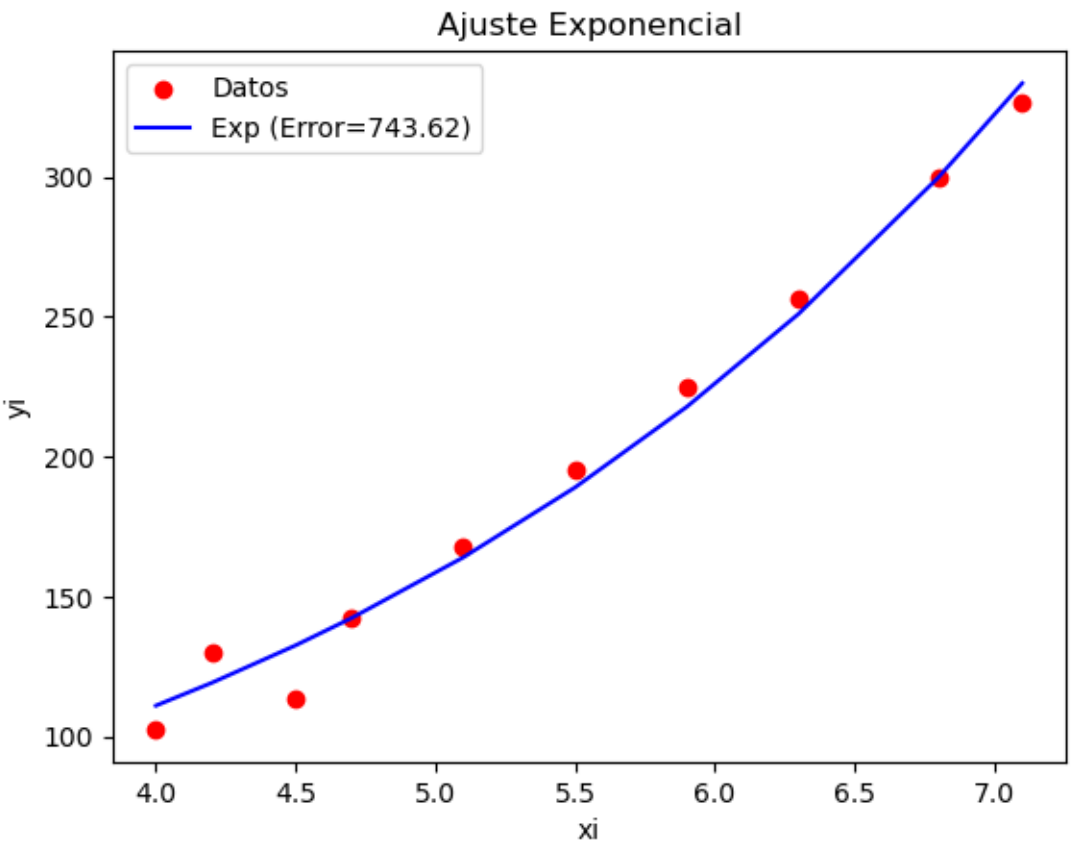
```
# Grado 3
coef3 = np.polyfit(xi, yi, 3)
poly3 = np.poly1d(coef3)
error3 = np.sum((poly3(xi) - yi) ** 2)

# Graficar
plt.scatter(xi, yi, color='red', label='Datos')
plt.plot(xi, poly3(xi), label=f'Grado 3 (Error={error3:.2f})', color='blue')
plt.xlabel('xi')
plt.ylabel('yi')
plt.legend()
plt.title('Polinomio de Grado 3')
plt.show()
```



d. Construya el polinomio por mínimos cuadrados de la forma be^{ax} y calcule el error.

```
# Función exponencial
def func_exp(x, a, b):
    return b * np.exp(a * x)

# Ajuste
popt_exp, _ = curve_fit(func_exp, xi, yi)
error_exp = np.sum((func_exp(xi, *popt_exp) - yi) ** 2)

# Graficar
plt.scatter(xi, yi, color='red', label='Datos')
plt.plot(xi, func_exp(xi, *popt_exp), label=f'Exp (Error={error_exp:.2f})', color='blue')
plt.xlabel('xi')
plt.ylabel('yi')
plt.legend()
plt.title('Ajuste Exponencial')
```

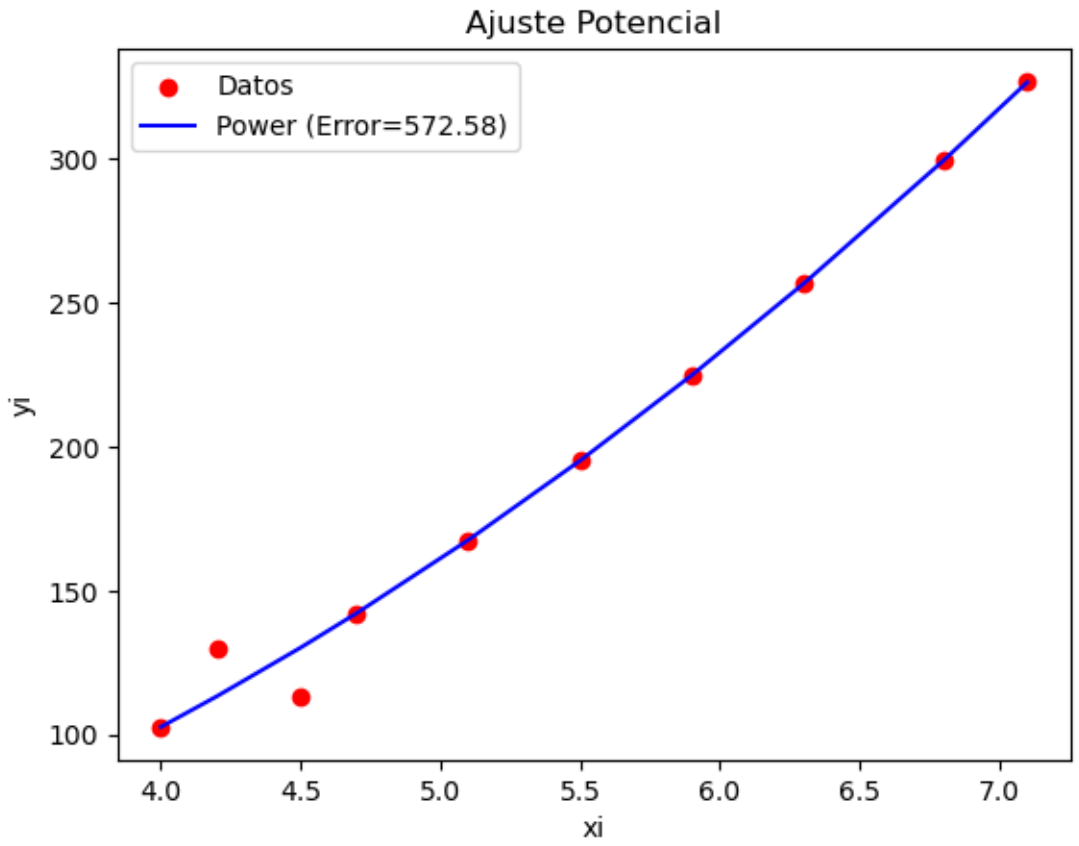
```
plt.show()
```



e. Construya el polinomio por mínimos cuadrados de la forma b^x y calcule el error.

```
# Función potencial
def func_power(x, a, b):
    return b * x**a

# Ajuste
popt_power, _ = curve_fit(func_power, xi, yi)
error_power = np.sum((func_power(xi, *popt_power) - yi) ** 2)

# Graficar
plt.scatter(xi, yi, color='red', label='Datos')
plt.plot(xi, func_power(xi, *popt_power), label=f'Power (Error={error_power:.2f})', color='bl
plt.xlabel('xi')
```

```
plt.ylabel('yi')
plt.legend()
plt.title('Ajuste Potencial')
plt.show()
```



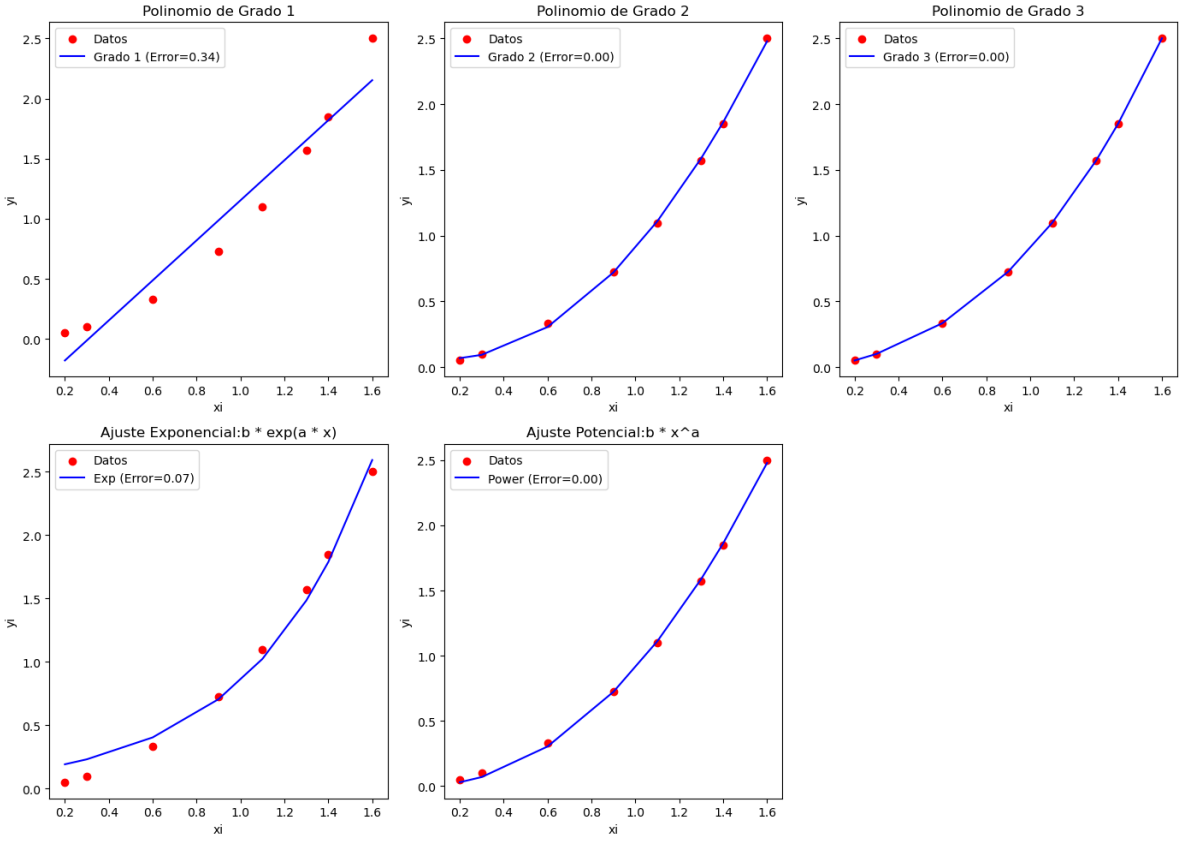
2. Repita el ejercicio 5 para los siguientes datos.

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

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

```
# Datos
xi = np.array([0.2, 0.3, 0.6, 0.9, 1.1, 1.3, 1.4, 1.6])
yi = np.array([0.050446, 0.098426, 0.33277, 0.72660, 1.0972, 1.5697, 1.8487, 2.5015])

# Polinomio de grado 1
coef1 = np.polyfit(xi, yi, 1)
poly1 = np.poly1d(coef1)
error1 = np.sum((poly1(xi) - yi) ** 2)

# Polinomio de grado 2
coef2 = np.polyfit(xi, yi, 2)
poly2 = np.poly1d(coef2)
error2 = np.sum((poly2(xi) - yi) ** 2)

# Polinomio de grado 3
coef3 = np.polyfit(xi, yi, 3)
poly3 = np.poly1d(coef3)
error3 = np.sum((poly3(xi) - yi) ** 2)

# Ajuste exponencial: y = b * exp(a * x)
def func_exp(x, a, b):
    return b * np.exp(a * x)

popt_exp, _ = curve_fit(func_exp, xi, yi)
error_exp = np.sum((func_exp(xi, *popt_exp) - yi) ** 2)

# Ajuste potencial: y = b * x^a
def func_power(x, a, b):
    return b * x**a

popt_power, _ = curve_fit(func_power, xi, yi)
error_power = np.sum((func_power(xi, *popt_power) - yi) ** 2)

# Graficar resultados
plt.figure(figsize=(14, 10))

# Polinomio de grado 1
plt.subplot(2, 3, 1)
plt.scatter(xi, yi, color='red', label='Datos')
plt.plot(xi, poly1(xi), label=f'Grado 1 (Error={error1:.2f})', color='blue')
plt.xlabel('xi')
plt.ylabel('yi')
```

```
plt.legend()
plt.title('Polinomio de Grado 1')

# Polinomio de grado 2
plt.subplot(2, 3, 2)
plt.scatter(xi, yi, color='red', label='Datos')
plt.plot(xi, poly2(xi), label=f'Grado 2 (Error={error2:.2f})', color='blue')
plt.xlabel('xi')
plt.ylabel('yi')
plt.legend()
plt.title('Polinomio de Grado 2')

# Polinomio de grado 3
plt.subplot(2, 3, 3)
plt.scatter(xi, yi, color='red', label='Datos')
plt.plot(xi, poly3(xi), label=f'Grado 3 (Error={error3:.2f})', color='blue')
plt.xlabel('xi')
plt.ylabel('yi')
plt.legend()
plt.title('Polinomio de Grado 3')

# Ajuste exponencial
plt.subplot(2, 3, 4)
plt.scatter(xi, yi, color='red', label='Datos')
plt.plot(xi, func_exp(xi, *popt_exp), label=f'Exp (Error={error_exp:.2f})', color='blue')
plt.xlabel('xi')
plt.ylabel('yi')
plt.legend()
plt.title('Ajuste Exponencial:b * exp(a * x)')

# Ajuste potencial
plt.subplot(2, 3, 5)
plt.scatter(xi, yi, color='red', label='Datos')
plt.plot(xi, func_power(xi, *popt_power), label=f'Power (Error={error_power:.2f})', color='bl
plt.xlabel('xi')
plt.ylabel('yi')
plt.legend()
plt.title('Ajuste Potencial:b * x^a')

plt.tight_layout()
plt.show()
```

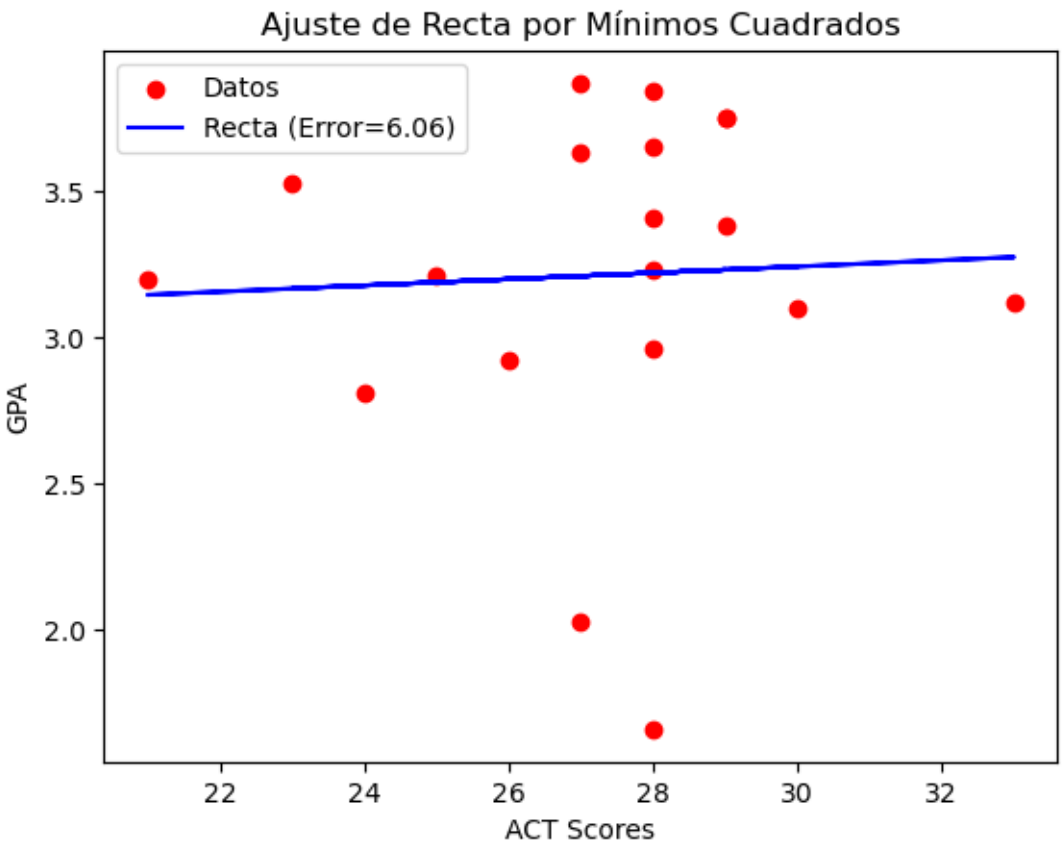

Polinomio de Grado 1
Datos
Grado 1 (Error=0.34)
Polinomio de Grado 2
Datos
Grado 2 (Error=0.00)
Polinomio de Grado 3
Datos
Grado 3 (Error=0.00)
Ajuste Exponencial:b * exp(a * x)
Datos
Exp (Error=0.07)
Ajuste Potencial:b * x^a
Datos
Power (Error=0.00)
xi
yi

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

```
act_scores = np.array([28, 29, 25, 28, 28, 27, 27, 29, 28, 21, 33, 28, 28, 29, 26, 23, 30, 2
gpa_scores = np.array([3.84, 3.75, 3.21, 3.65, 3.23, 3.63, 3.87, 3.75, 1.66, 3.20, 3.12, 3.4
```

```
# Ajuste de la recta
coef3 = np.polyfit(act_scores, gpa_scores, 1)
poly3 = np.poly1d(coef3)
error3 = np.sum((poly3(act_scores) - gpa_scores) ** 2)

# Graficar
plt.scatter(act_scores, gpa_scores, color='red', label='Datos')
plt.plot(act_scores, poly3(act_scores), label=f'Recta (Error={error3:.2f})', color='blue')
plt.xlabel('ACT Scores')
plt.ylabel('GPA')
plt.legend()
plt.title('Ajuste de Recta por Mínimos Cuadrados')
plt.show()
```

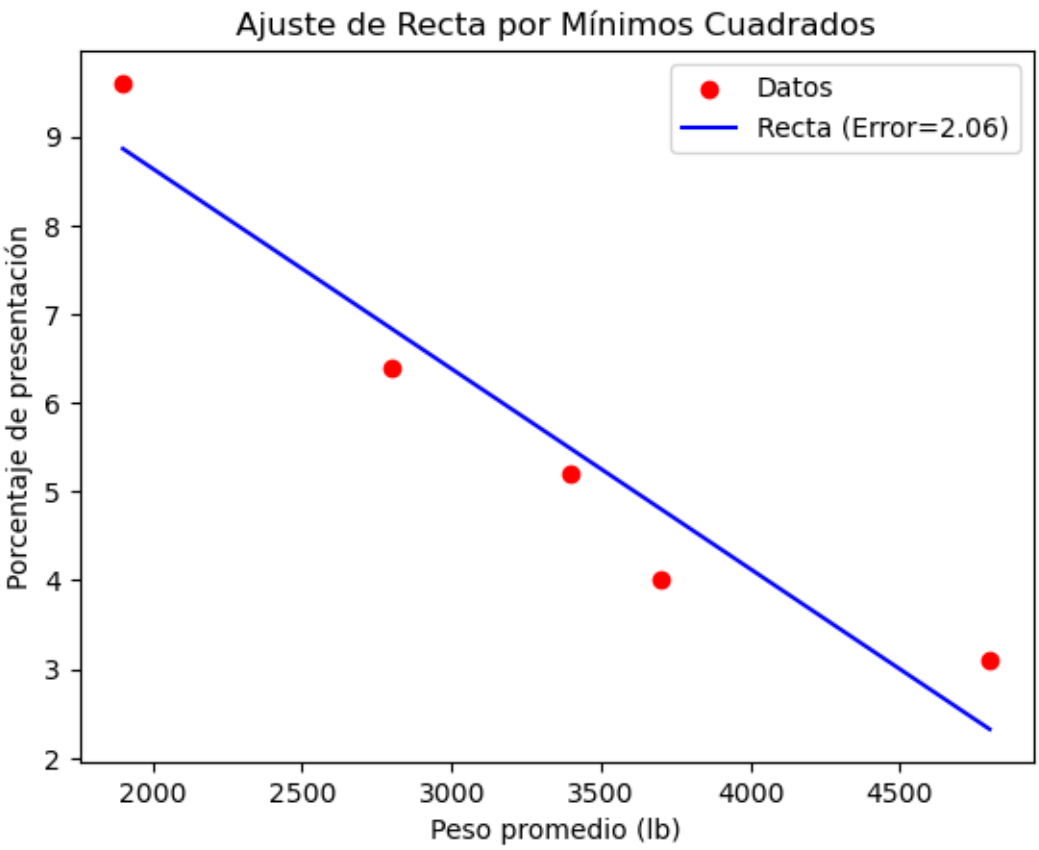

Ajuste de Recta por Mínimos Cuadrados
Datos
Recta (Error=6.06)
GPA
ACT Scores
3.5
3.0
2.5
2.0
22
24
26
28
30
32

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

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

```
# Ajuste de la recta
coef4 = np.polyfit(weight, percentage, 1)
poly4 = np.poly1d(coef4)
error4 = np.sum((poly4(weight) - percentage) ** 2)

# Graficar
plt.scatter(weight, percentage, color='red', label='Datos')
plt.plot(weight, poly4(weight), label=f'Recta (Error={error4:.2f})', color='blue')
plt.xlabel('Peso promedio (lb)')
plt.ylabel('Porcentaje de presentación')
plt.legend()
plt.title('Ajuste de Recta por Mínimos Cuadrados')
plt.show()
```



Link el repositorio: https://github.com/armando-2002/Metodos_Numericos.git