Tarea 8 - Mínimos Cuadrados

Nombre: Alexis Bautista
Fecha de entrega: 08 de enero de 2025
Paralelo: GR1CC
Enlace de GitHub: https://github.com/alexis-bautista/Tarea08-MN

Ejercicio 1

Dados los datos:

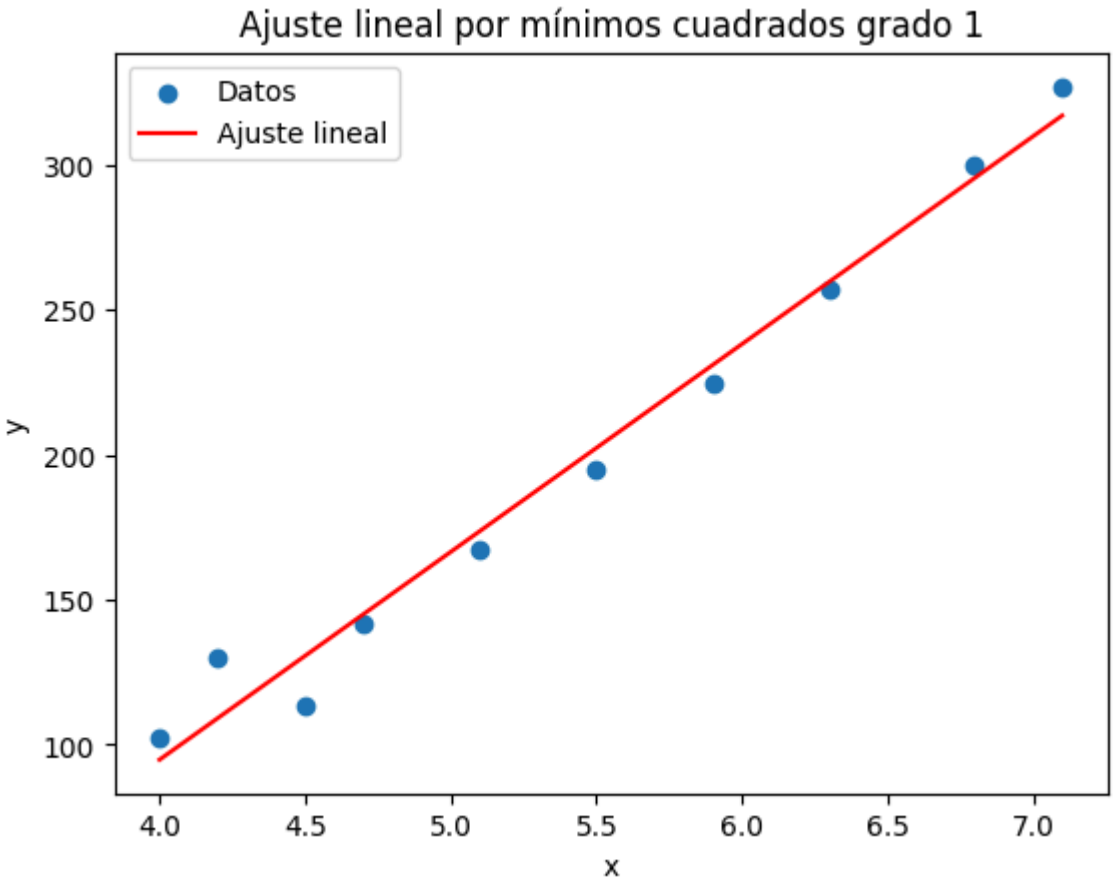
x_i	4.0	4.2	4.5	4.7	5.1	5.5	5.9	6.3	6.8	7.1
y_i	102.56	130.11	113.18	142.05	167.53	195.14	224.87	256.73	299.50	326.72

In [75]:
```
import numpy as np
import matplotlib.pyplot as plt

x_i = np.array ([4.00, 4.20, 4.50, 4.70, 5.10, 5.50, 5.90, 6.30, 6.80, 7.10])
y_i = np.array ([102.56, 130.11, 113.18, 142.05, 167.53, 195.14, 224.87, 256.73,
```

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

In [76]:
```
# Ajuste de polinomio de grado 1 (recta) por mínimos cuadrados
coeficientes = np.polyfit(x_i, y_i, 1)
p = np.poly1d(coeficientes)

# Calcular los valores ajustados
y_fit = p(x_i)

# Calcular el error cuadrático medio
error = np.mean((y_i - y_fit) ** 2)

print("Coeficientes del polinomio de grado 1:", coeficientes)
print("Polinomio: y =",  f"{coeficientes[0]:.2f}x + {coeficientes[1]:.2f}")
print("Error cuadrático medio:", error)

# Graficar los datos y el ajuste
plt.scatter(x_i, y_i, label='Datos')
plt.plot(x_i, y_fit, color='red', label='Ajuste lineal')
plt.title('Ajuste lineal por mínimos cuadrados grado 1')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

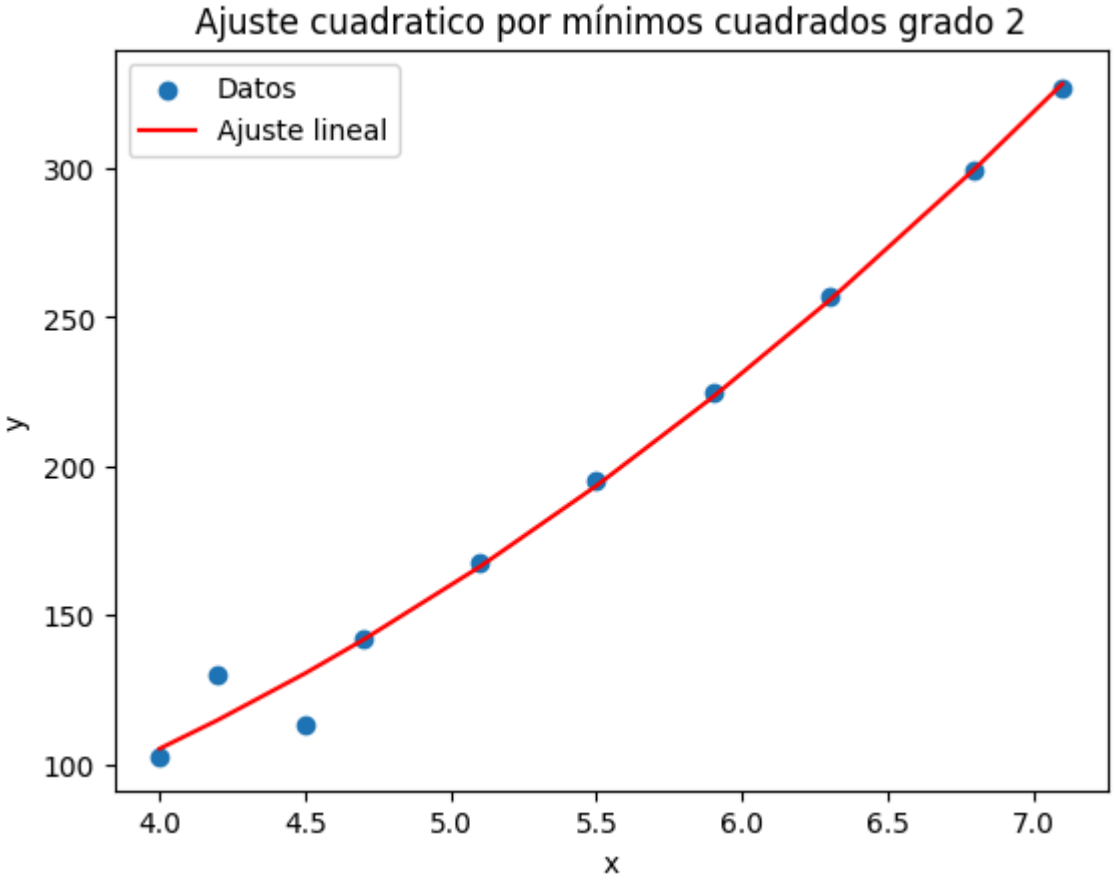
```
Coeficientes del polinomio de grado 1: [  71.61024372 -191.57241853]
Polinomio: y = 71.61x + -191.57
Error cuadrático medio: 105.88388862638904
```



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

In [77]:
```python
# Ajuste de polinomio de grado 2 por mínimos cuadrados
coeficientes = np.polyfit(x_i, y_i, 2)
p = np.poly1d(coeficientes)

# Calcular los valores ajustados
y_fit = p(x_i)

# Calcular el error cuadrático medio
error = np.mean((y_i - y_fit) ** 2)

print("Coeficientes del polinomio de grado 2:", coeficientes)
print("Polinomio: y =",  f"{coeficientes[0]:.2f}x^2 + {coeficientes[1]:.2f}x + {
print("Error cuadrático medio:", error)

# Graficar los datos y el ajuste
plt.scatter(x_i, y_i, label='Datos')
plt.plot(x_i, y_fit, color='red', label='Ajuste lineal')
plt.title('Ajuste cuadratico por mínimos cuadrados grado 2')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

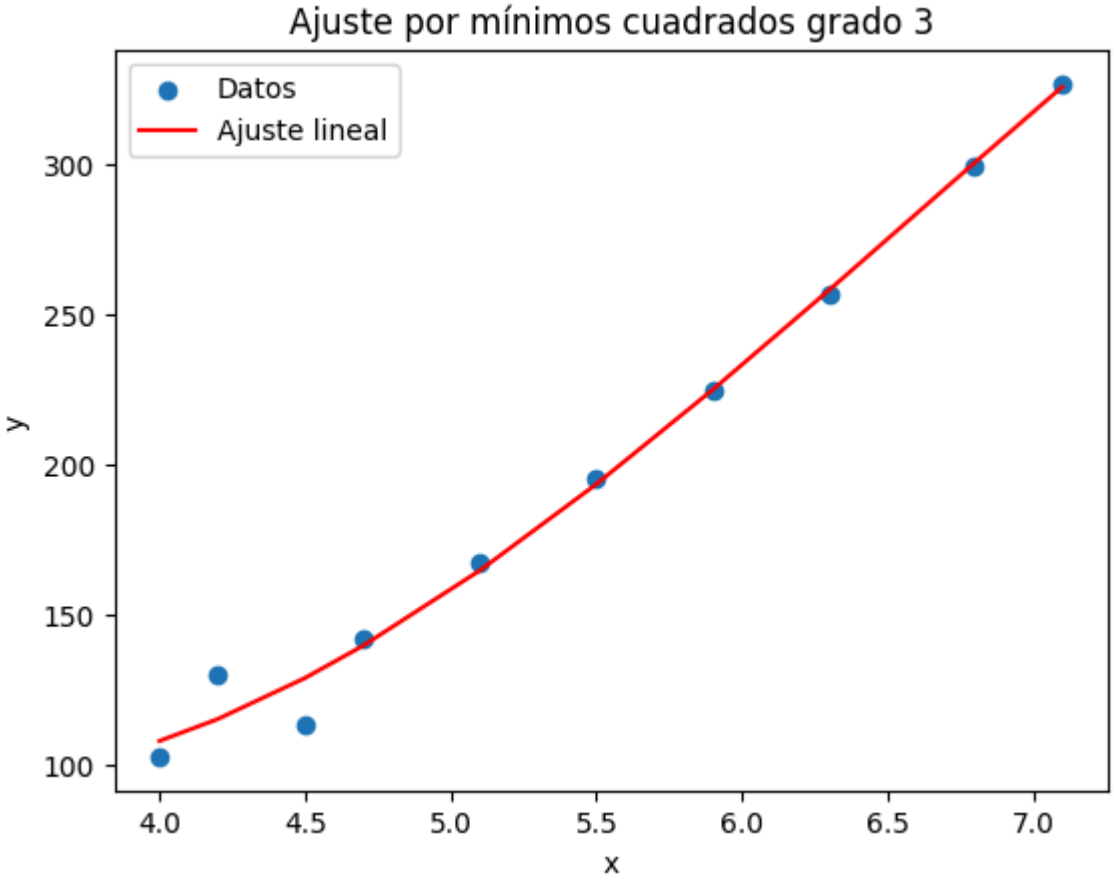
```
Coeficientes del polinomio de grado 2: [  8.21707232 -19.30860379  51.00078939]
Polinomio: y = 8.22x^2 + -19.31x + 51.00
Error cuadrático medio: 55.1656200117024
```



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

In [78]:
```
# Ajuste de polinomio de grado 3 por mínimos cuadrados
coeficientes = np.polyfit(x_i, y_i, 3)
p = np.poly1d(coeficientes)

# Calcular los valores ajustados
y_fit = p(x_i)

# Calcular el error cuadrático medio
error = np.mean((y_i - y_fit) ** 2)

print("Coeficientes del polinomio de grado 3:", coeficientes)
print("Polinomio: y =",  f"{coeficientes[0]:.2f}x^3 + {coeficientes[1]:.2f}x^2 +
print("Error cuadrático medio:", error)

# Graficar los datos y el ajuste
plt.scatter(x_i, y_i, label='Datos')
plt.plot(x_i, y_fit, color='red', label='Ajuste lineal')
plt.title('Ajuste por mínimos cuadrados grado 3')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

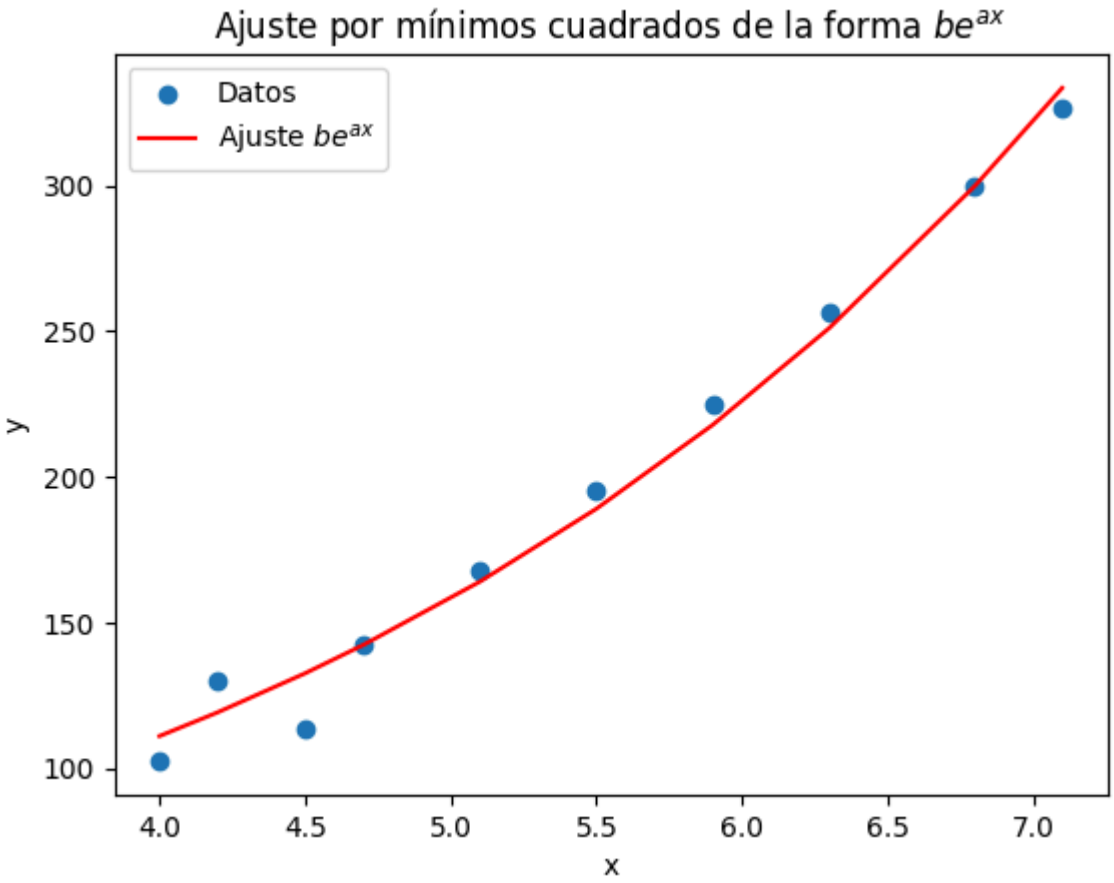
```
Coeficientes del polinomio de grado 3: [  -2.60683872   51.56095694 -254.87478338
469.16326528]
Polinomio: y = -2.61x^3 + 51.56x^2 + -254.87x + 469.16
Error cuadrático medio: 51.83830647403033
```



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

In [79]:
```python
from scipy.optimize import curve_fit

# Definir la función de ajuste
def func(x, a, b):
    return b * np.exp(a * x)

# Ajustar la curva
parametros, parametros_covarianza = curve_fit(func, x_i, y_i, p0=[1, 25.64])

# Calcular los valores ajustados
y_fit = func(x_i, parametros[0], parametros[1])

# Calcular el error cuadrático medio
error = np.mean((y_i - y_fit) ** 2)

print("Coeficientes del ajuste: a =", parametros[0], ", b =", parametros[1])
print("Error cuadrático medio:", error)

# Graficar los datos y el ajuste
plt.scatter(x_i, y_i, label='Datos')
plt.plot(x_i, y_fit, color='red', label='Ajuste $be^{ax}$')
plt.title('Ajuste por mínimos cuadrados de la forma $be^{ax}$')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

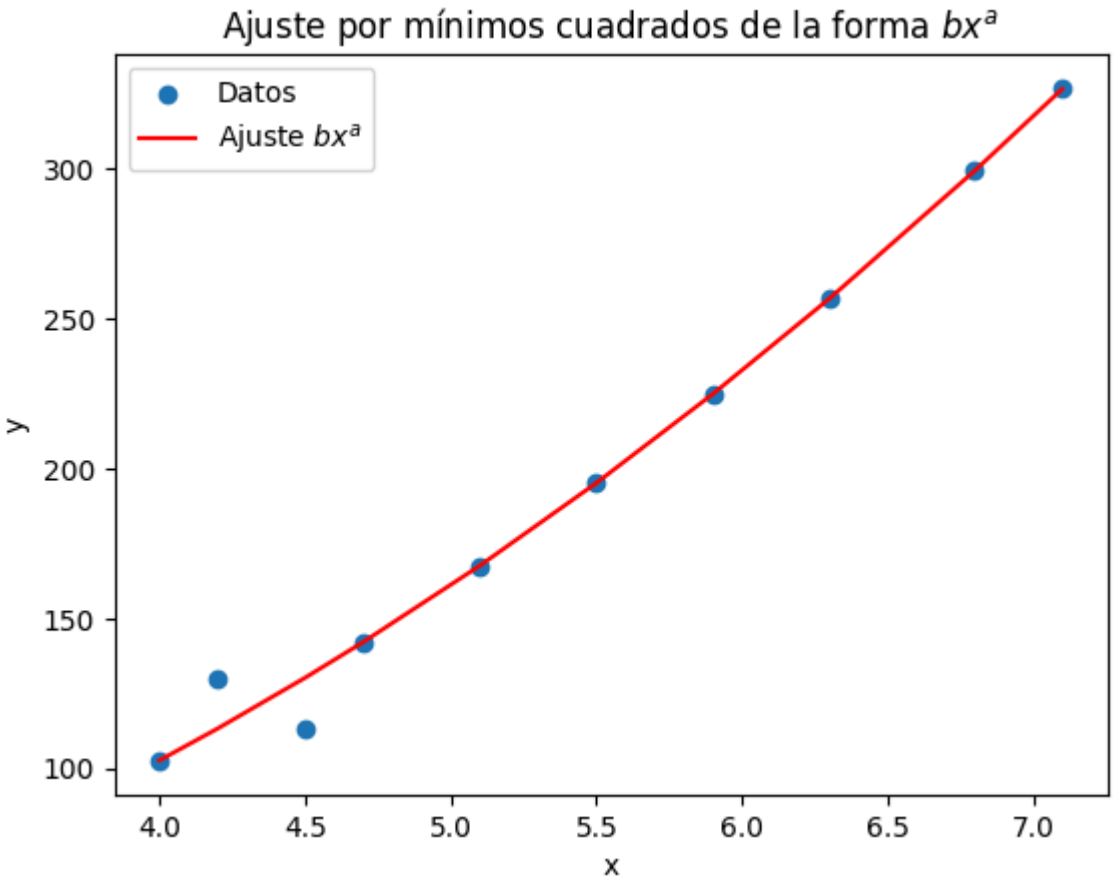
```
Coeficientes del ajuste: a = 0.3549271657971687 , b = 26.8407610314521
Error cuadrático medio: 74.36215913535702
```



e) Construya el polinomio por mínimos cuadrados de la forma bx^a y calcule el error.

In [80]:
```
# Definir la función de ajuste
def func(x, a, b):
    return b * x ** a

# Ajustar la curva
parametros, parametros_covarianza = curve_fit(func, x_i, y_i, p0=[1, 25.64])

# Calcular los valores ajustados
y_fit = func(x_i, parametros[0], parametros[1])

# Calcular el error cuadrático medio
error = np.mean((y_i - y_fit) ** 2)

print("Coeficientes del ajuste: a =", parametros[0], ", b =", parametros[1])
print("Error cuadrático medio:", error)

# Graficar los datos y el ajuste
plt.scatter(x_i, y_i, label='Datos')
plt.plot(x_i, y_fit, color='red', label='Ajuste $bx^{a}$')
plt.title('Ajuste por mínimos cuadrados de la forma $bx^{a}$')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

```
Coeficientes del ajuste: a = 2.015190343070704 , b = 6.283966078899579
Error cuadrático medio: 57.25817542056747
```



Ejercicio 2

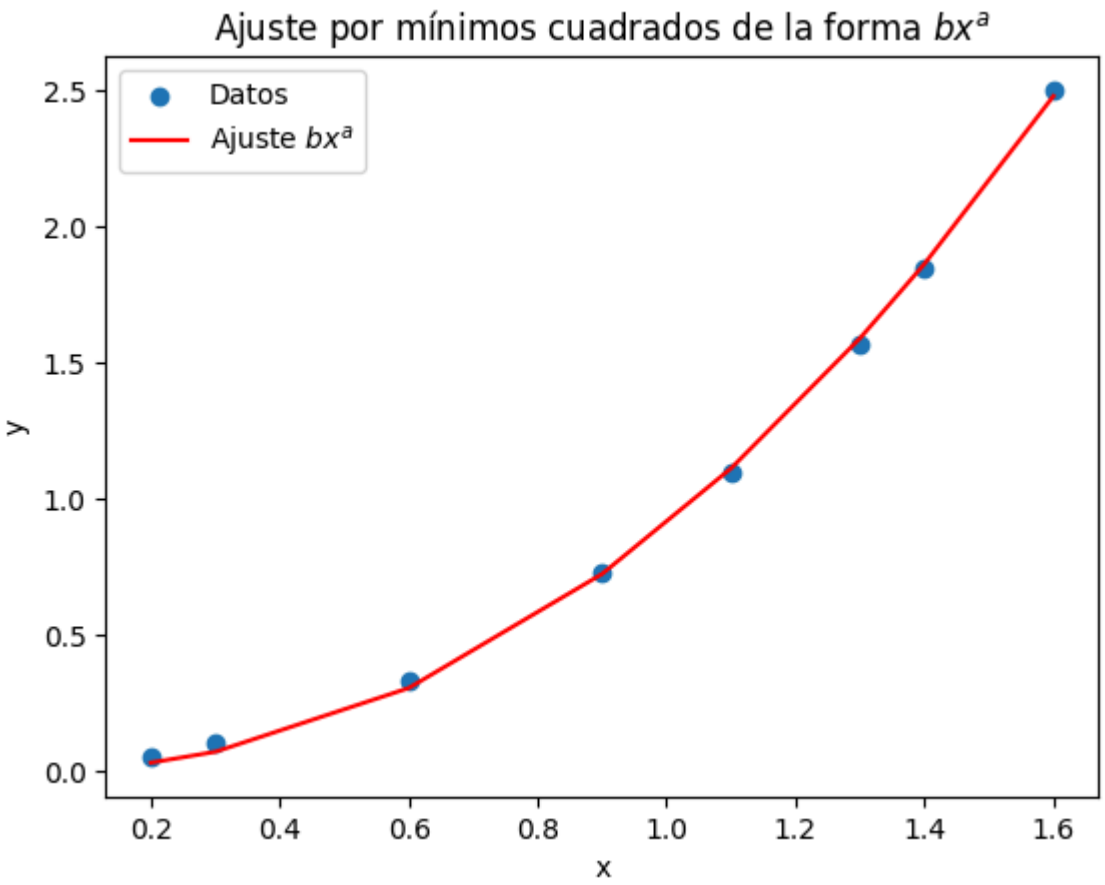
Repita el ejercicio 5 para los siguientes datos.

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

In [81]:
```python
# Datos
x_i = np.array([0.2, 0.3, 0.6, 0.9, 1.1, 1.3, 1.4, 1.6])
y_i = np.array([0.050446, 0.098426, 0.33277, 0.72660, 1.0972, 1.5697, 1.8487, 2.

# Definir la función de ajuste de la forma y = bx^a
def func(x, a, b):
    return b * x ** a

# Ajustar la curva
parametros, parametros_covarianza = curve_fit(func, x_i, y_i, p0=[1, 3.964])

# Calcular los valores ajustados
y_fit = func(x_i, parametros[0], parametros[1])

# Calcular el error cuadrático medio
error = np.mean((y_i - y_fit) ** 2)

print("Coeficientes del ajuste: a =", parametros[0], ", b =", parametros[1])
print(f"Error cuadrático medio:, {error:.4e}")

# Graficar los datos y el ajuste
plt.scatter(x_i, y_i, label='Datos')
```

```
plt.plot(x_i, y_fit, color='red', label='Ajuste $bx^{a}$')
plt.title('Ajuste por mínimos cuadrados de la forma $bx^{a}$')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

```
Coeficientes del ajuste: a = 2.142772912914175 , b = 0.9055224414487177
Error cuadrático medio:, 4.3594e-04
```



Ejercicio 3

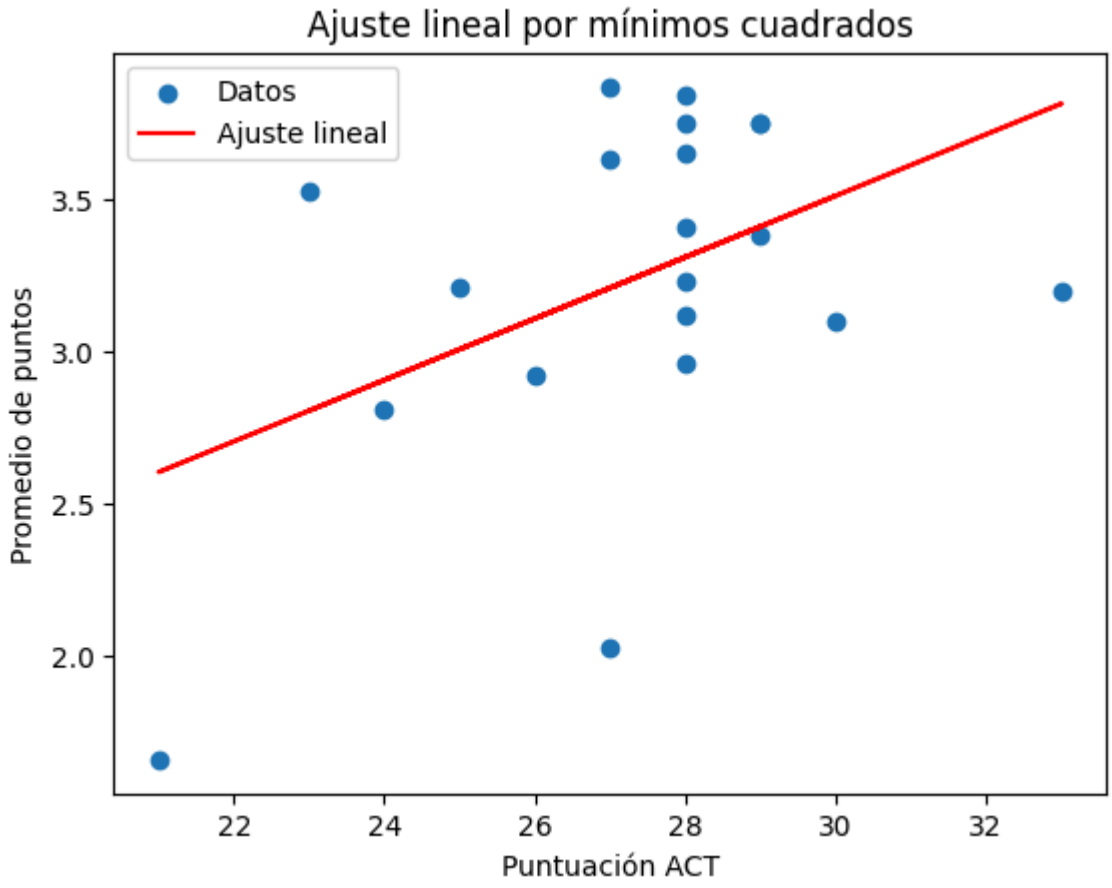
La siguiente tabla muestra los promedios de puntos del colegio de 20 especialistas en matemáticas y ciencias computacionales, junto con las calificaciones que recibieron estos estudiantes en la parte de matemáticas de la prueba ACT (Programa de Pruebas de Colegios Americanos)mientras estaban en secundaria.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

Puntuación ACT	Promedio de puntos	Puntuación ACT	Promedio de puntos
28	3.41	28	2.96
29	3.38	26	2.92
23	3.53	30	3.10
27	2.03	24	2.81

In [82]:
```
# Datos
x_act = np.array([28, 25, 28, 27, 28, 33, 28, 29, 23, 27, 29, 28, 27, 29, 21, 28
y_prom = np.array([3.84, 3.21, 3.23, 3.63, 3.75, 3.2 , 3.41, 3.38, 3.53, 2.03, 3

# Ajuste de polinomio de grado 1 (recta) por mínimos cuadrados
coeficientes = np.polyfit(x_act, y_prom, 1)
p = np.poly1d(coeficientes)

# Calcular los valores ajustados
y_fit = p(x_act)

# Calcular el error cuadrático medio
error = np.mean((y_prom - y_fit) ** 2)

print("Coeficientes del polinomio de grado 1:", coeficientes)
print("Polinomio: y =",  f"{coeficientes[0]:.2f}x + {coeficientes[1]:.2f}")
print("Error cuadrático medio:", error)

# Graficar los datos y el ajuste
plt.scatter(x_act, y_prom, label='Datos')
plt.plot(x_act, y_fit, color='red', label='Ajuste lineal')
plt.title('Ajuste lineal por mínimos cuadrados')
plt.xlabel('Puntuación ACT')
plt.ylabel('Promedio de puntos')
plt.legend()
plt.show()
```

```
Coeficientes del polinomio de grado 1: [0.10085803 0.48657566]
Polinomio: y = 0.10x + 0.49
Error cuadrático medio: 0.2524352808112325
```



Ejrecicio 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 extrangero	1900 lb	9.6

In [83]:
```
# Datos
x_peso = np.array([4800, 3700, 3400, 2800, 1900])
y_porcentaje = np.array([3.1, 4.0, 5.2, 6.4, 9.6])

# Ajuste de polinomio de grado 1 (recta) por mínimos cuadrados
coeficientes = np.polyfit(x_peso, y_porcentaje, 1)
p = np.poly1d(coeficientes)

# Calcular los valores ajustados
y_fit = p(x_peso)
```

```python
# Calcular el error cuadrático medio
error = np.mean((y_porcentaje - y_fit) ** 2)

print("Coeficientes del polinomio de grado 1:", coeficientes)
print("Polinomio: y =",  f"{coeficientes[0]:.3e}x + {coeficientes[1]:.1e}")
print("Error cuadrático medio:", error)

# Graficar los datos y el ajuste
plt.scatter(x_peso, y_porcentaje, label='Datos')
plt.plot(x_peso, y_fit, color='red', label='Ajuste lineal')
plt.title('Ajuste lineal por mínimos cuadrados')
plt.xlabel('Peso Promedio')
plt.ylabel('Porcentaje de presentación')
plt.legend()
plt.show()
```

```
Coeficientes del polinomio de grado 1: [-2.25496975e-03  1.31464996e+01]
Polinomio: y = -2.255e-03x + 1.3e+01
Error cuadrático medio: 0.411827139152982
```