Escuela Politécnica Nacional

Nombre: David Egas

Tarea 3: Minimos cuadrados

Funcion que nos ayudara a calcular el error:

```
import numpy as np
##Calcular el error
def error(ys, y_error):
    return np.mean((ys-y_error)**2)
```

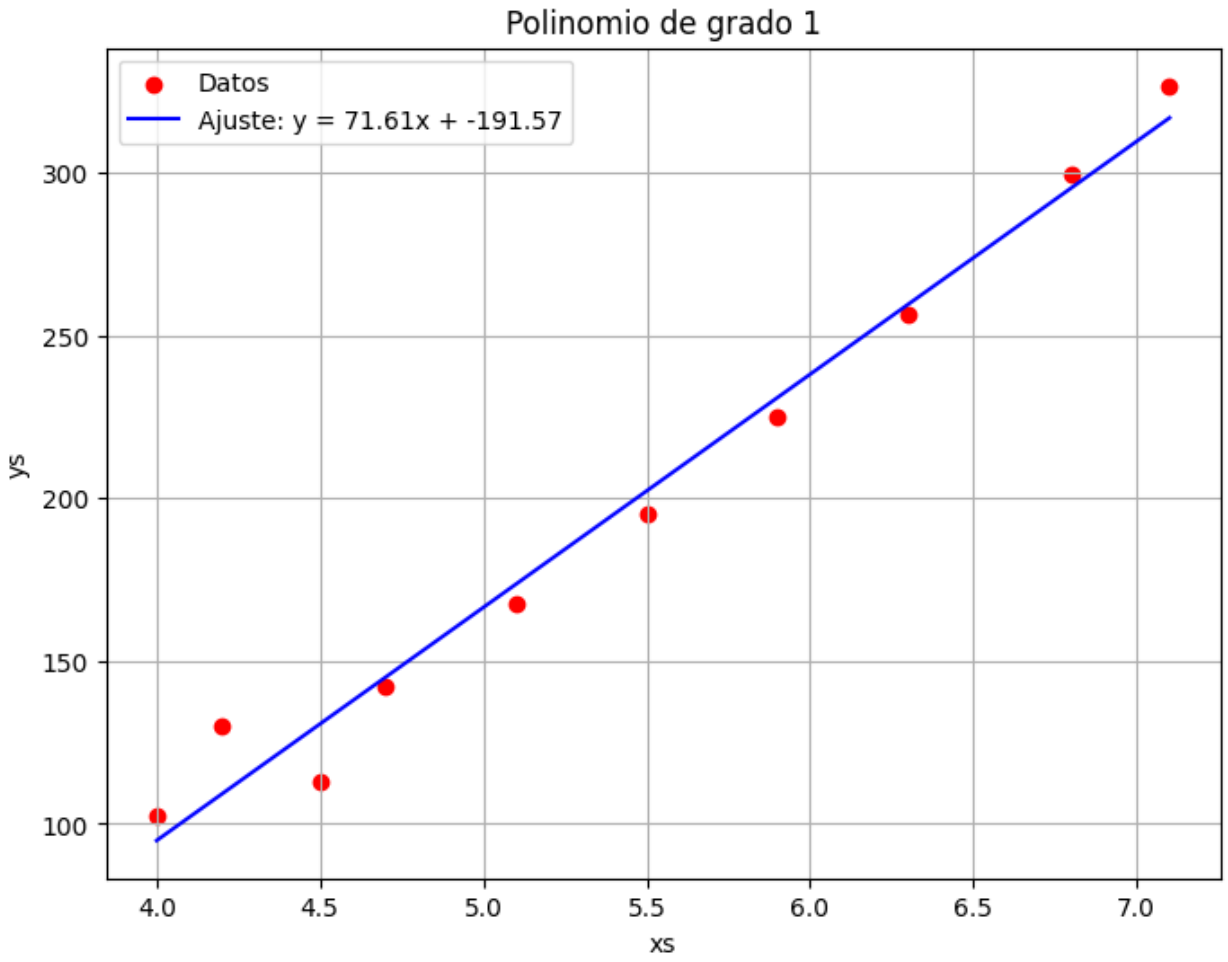
1. Dados los datos:

```
xs = [  4.0,  4.2,     4.5,    4.7,    5.1,    5.5,    5.9,    6.3,
6.8,    7.1]
ys = [102.56, 130.11, 113.18, 142.05, 167.53, 195.14, 224.87, 256.73,
299.50, 326.72]
```

a. Contruya el polinomio por minimos cuadrados de grado 1 y calcule el error.

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

# Ajuste polinómico de grado 1 (recta)
resultado = np.polyfit(xs, ys, 1)
m = resultado[0]
b = resultado[1]
y_error=np.polyval(resultado, xs)
error1=error(ys, y_error)

# Línea de ajuste
x_line = np.linspace(min(xs), max(xs), 100)  # Generar más puntos para
una línea más suave
y_line = m * x_line + b
# Crear la gráfica
plt.figure(figsize=(8, 6))
plt.scatter(xs, ys, label="Datos", color="red")  # Puntos de los datos
plt.plot(x_line, y_line, color="blue", label=f"Ajuste: y = {m:.2f}x +
{b:.2f}")  # Línea de ajuste

# Personalizar la gráfica
plt.xlabel("xs")
plt.ylabel("ys")
plt.title("Polinomio de grado 1")
plt.legend()  # Mostrar la leyenda
plt.grid(True)  # Añadir una cuadrícula
```

```
plt.show()
print(f"El polinomio resultante es: y = {round(m, 2)} X {round(b,
2)}")
print(f"El error total absoluto del modelo es : {error1}")
```

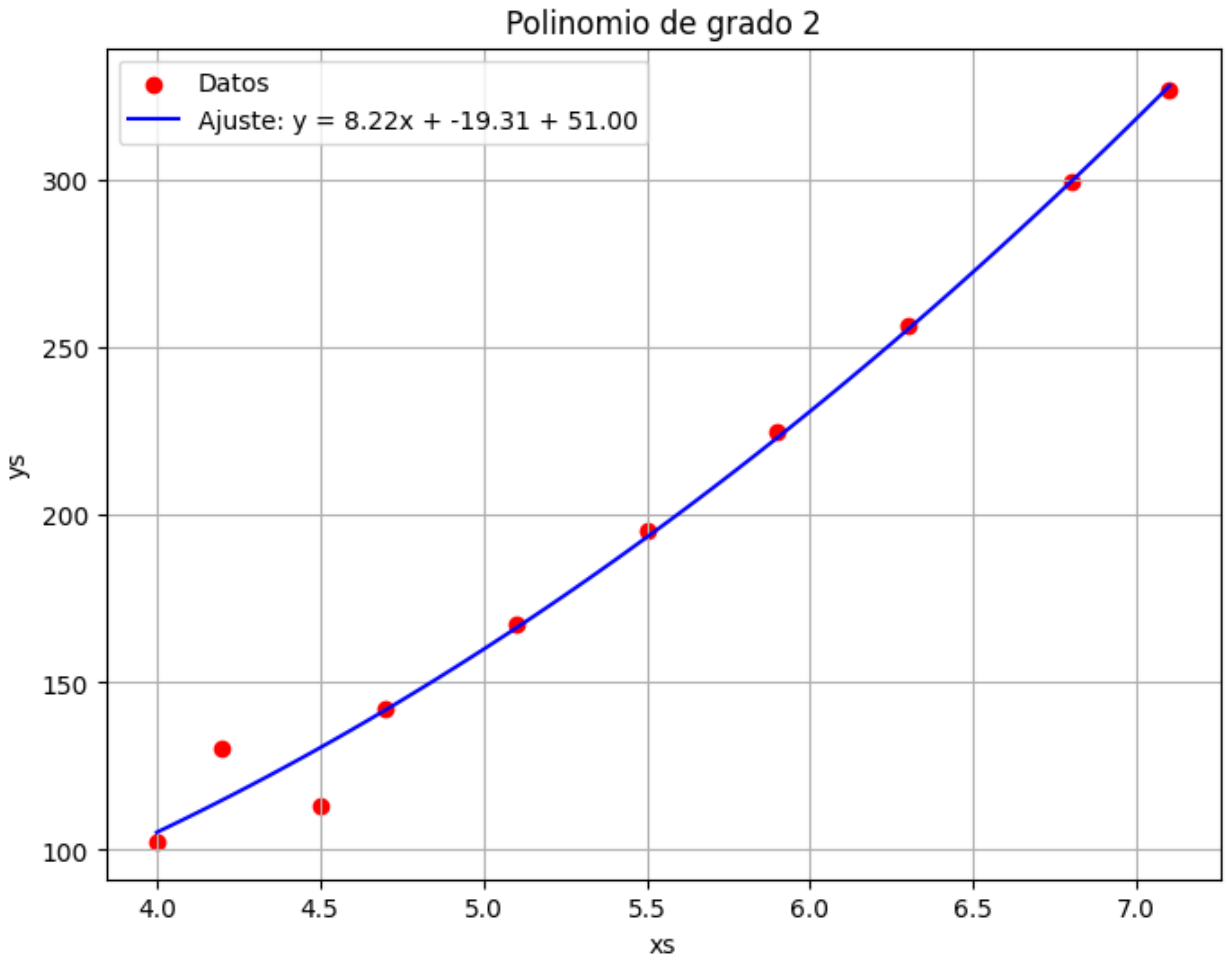



```
El polinomio resultante es: y = 71.61 X -191.57
El error total absoluto del modelo es : 105.88388862638904
```

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

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

# Ajuste polinómico de grado 2 (recta)
resultado = np.polyfit(xs, ys, 2)
a = resultado[0]
b = resultado[1]
c= resultado[2]
y_error=np.polyval(resultado, xs)
```

```
error2=error(ys, y_error)

# Línea de ajuste
x_line = np.linspace(min(xs), max(xs), 100)  # Generar más puntos para
una línea más suave
y_line = a*x_line**2 + b*x_line + c

# Crear la gráfica
plt.figure(figsize=(8, 6))
plt.scatter(xs, ys, label="Datos", color="red")  # Puntos de los datos
plt.plot(x_line, y_line, color="blue", label=f"Ajuste: y = {a:.2f}x +
{b:.2f} + {c:.2f}")  # Línea de ajuste

# Personalizar la gráfica
plt.xlabel("xs")
plt.ylabel("ys")
plt.title("Polinomio de grado 2")
plt.legend()  # Mostrar la leyenda
plt.grid(True)  # Añadir una cuadrícula
plt.show()
print(f"El polinomio resultante es: y = {round(a, 2)} X**2 {round(b,
2)} + {round(c,2)}")
print(f"El error total absoluto del modelo es : {error2}")
```

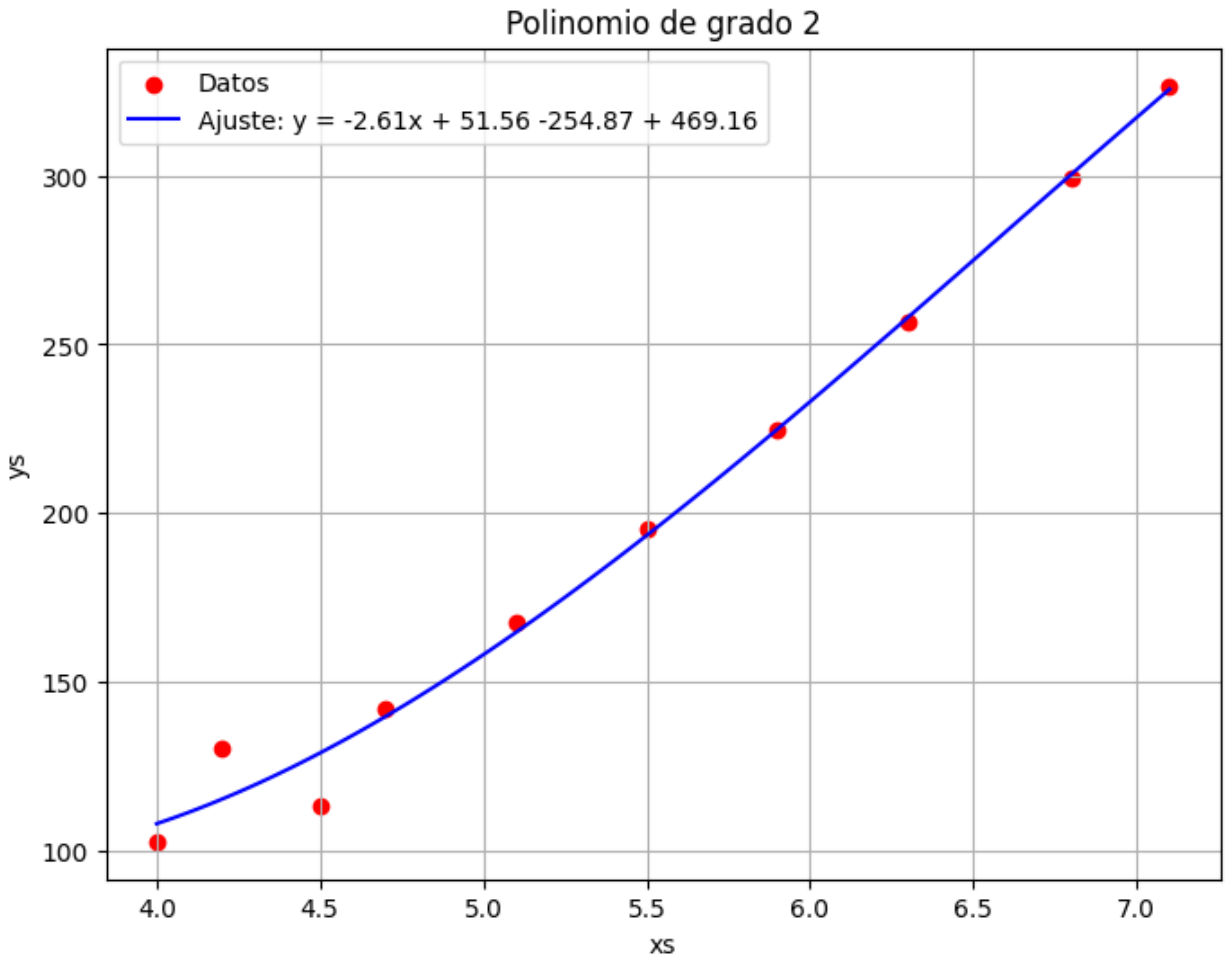



```
El polinomio resultante es: y = 8.22 X**2 -19.31 + 51.0
El error total absoluto del modelo es : 55.16562001170232
```

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

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

# Ajuste polinómico de grado 3 (recta)
resultado = np.polyfit(xs, ys, 3)
a = resultado[0]
b = resultado[1]
c= resultado[2]
d= resultado[3]
y_error=np.polyval(resultado, xs)
error3=error(ys, y_error)

# Línea de ajuste
x_line = np.linspace(min(xs), max(xs), 100)  # Generar más puntos para una línea más suave
```

```
y_line = a*x_line**3 + b*x_line**2 + c*x_line + d

# Crear la gráfica
plt.figure(figsize=(8, 6))
plt.scatter(xs, ys, label="Datos", color="red")  # Puntos de los datos
plt.plot(x_line, y_line, color="blue", label=f"Ajuste: y = {a:.2f}x +
{b:.2f} {c:.2f} + {d:.2f}")  # Línea de ajuste

# Personalizar la gráfica
plt.xlabel("xs")
plt.ylabel("ys")
plt.title("Polinomio de grado 2")
plt.legend()  # Mostrar la leyenda
plt.grid(True)  # Añadir una cuadrícula
plt.show()
print(f"El polinomio resultante es: y = {round(a, 2)} X**3 + {round(b,
2)}X**2 {round(c,2)}X + {round(d,2)}")
print(f"El error total absoluto del modelo es : {error3}")
```



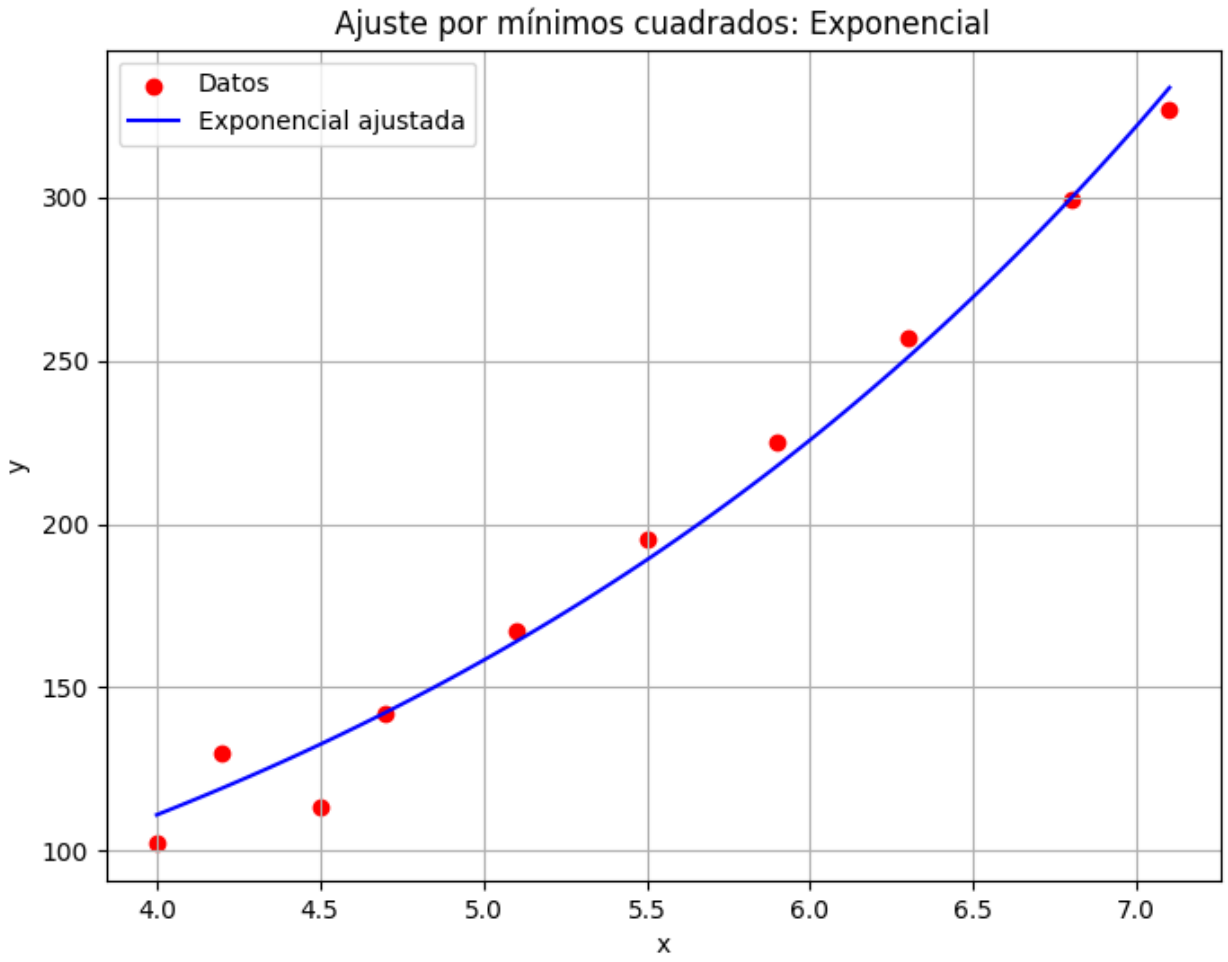
```
El polinomio resultante es: y = -2.61 X**3 + 51.56X**2 -254.87X + 469.16
El error total absoluto del modelo es : 51.83830647403006
```

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

Para este ejercicio se crea la funcion ajuste_minimos_cuadrados ya que los parametros de la funcion np.polifyt son para modelos lineales, no exponenciales . Por lo tanto, se necesita una funcion que ajuste un modelo exponencial los datos.

```
xs1 =np.array( [  4.0,  4.2,     4.5,    4.7,    5.1,    5.5,    5.9,    6.3,    6.8,    7.1])
ys1 = np.array([102.56, 130.11, 113.18, 142.05, 167.53, 195.14, 224.87, 256.73, 299.50, 326.72])

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

def ajuste_minimos_cuadrados(xs, ys, funcion, nombre_funcion):
    # Ajustar la función a los datos
    params, _ = curve_fit(funcion, xs, ys)
    # Generar los valores predichos con los parámetros ajustados
    y_pred = funcion(xs, *params)

    # Calcular el error absoluto promedio
    error_abt = np.mean(np.abs(ys - y_pred))  # Implementación del error absoluto promedio

    # Generar la línea ajustada
    x_line = np.linspace(min(xs), max(xs), 100)
    y_line = funcion(x_line, *params)  # params es igual a [a,b], *params = a,b

    # Graficar los datos y el ajuste
    plt.figure(figsize=(8, 6))
    plt.scatter(xs, ys, label="Datos", color="red")  # Puntos de datos
    plt.plot(x_line, y_line, color="blue", label=f"{nombre_funcion} ajustada")  # Línea ajustada
    plt.xlabel("x")
    plt.ylabel("y")
    plt.title(f"Ajuste por mínimos cuadrados: {nombre_funcion}")
    plt.legend()
    plt.grid(True)
    plt.show()
    return params, error_abt

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

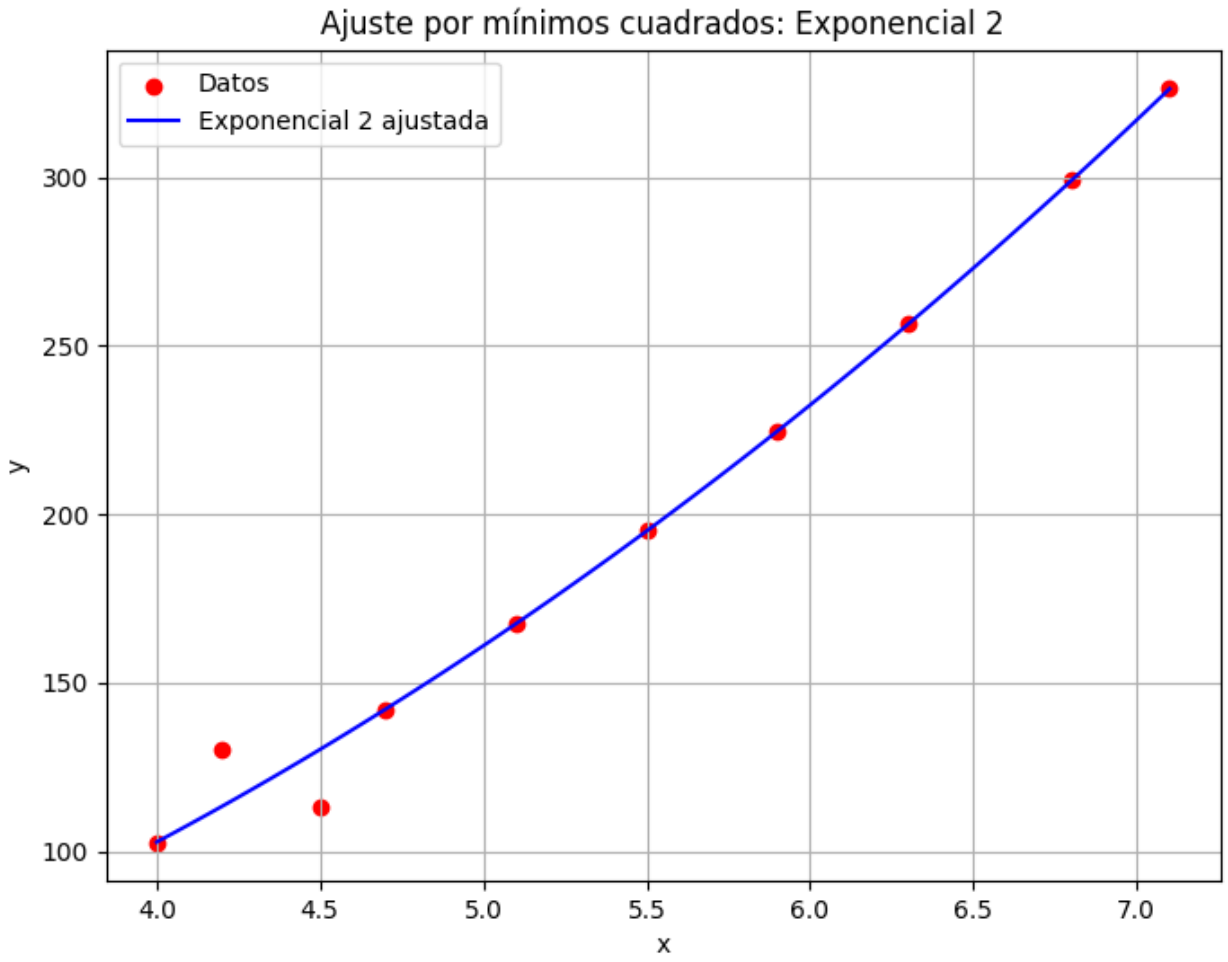
```
parametros,error =ajuste_minimos_cuadrados(xs1,ys1,
funcion_exp,nombre_funcion="Exponencial")
print(f"Funcion:
{round( parametros[1],2)}e^{round(parametros[0],2)}*X")
print (f"El error absoluto total es del modelo es: {error}")
```




```
Funcion: 26.84e^0.35*X
El error absoluto total es del modelo es: 6.847149185919551
```

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

```
def funcion_exp2(x,a,b):
    return b*x**a

parametros, error =ajuste_minimos_cuadrados(xs1,ys1,
funcion_exp2,nombre_funcion="Exponencial 2")
print(f"La funcion es:
{round(parametros[1],2)}X^{round(parametros[0],2)}")
print (f"El error absoluto total es del modelo es: {error}")
```




```
La funcion es: 6.28X^2.02
El error absoluto total es del modelo es: 3.5202098764748415
```

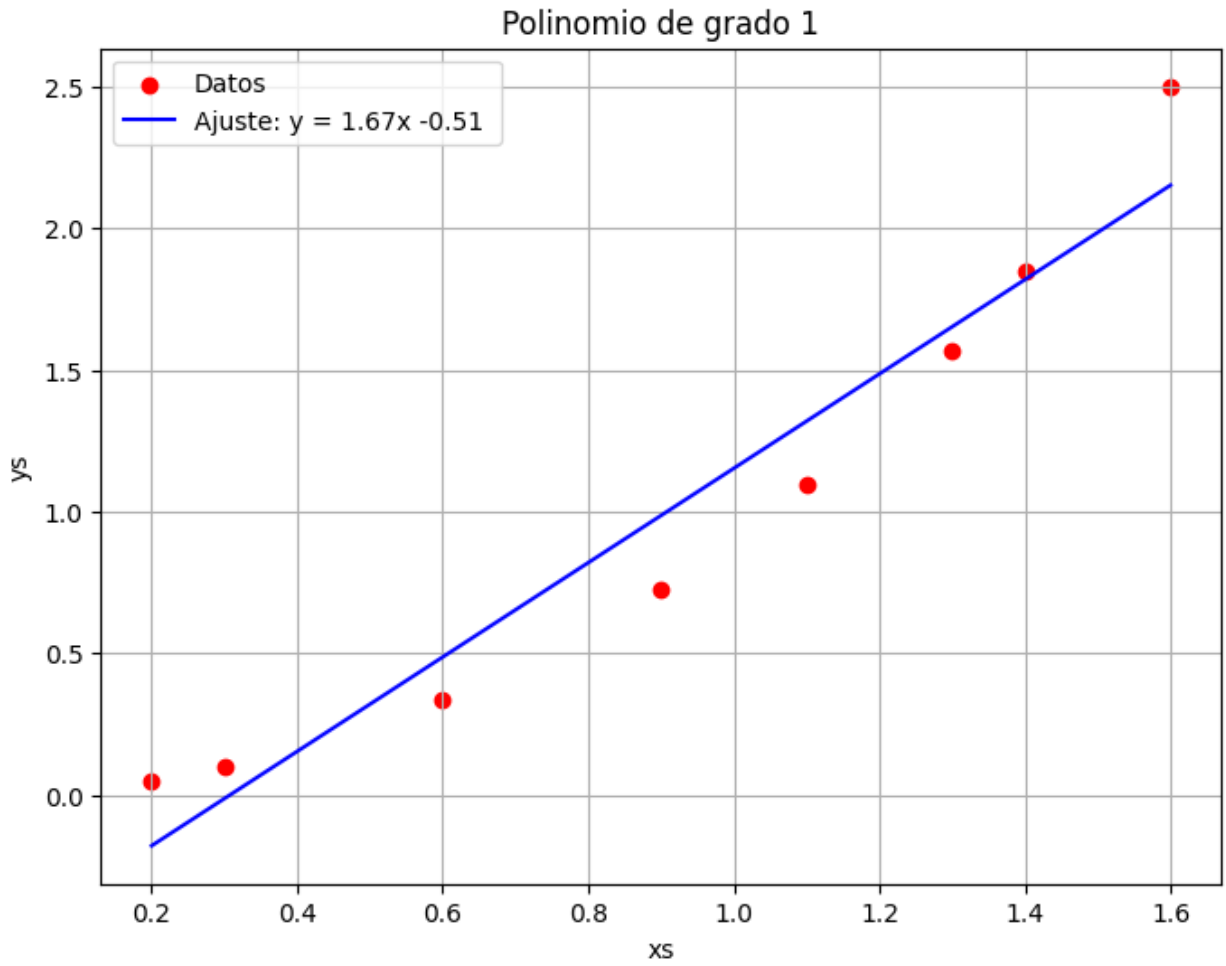
1. Repita el eejrcicio 5 con los siguientes datos:

```
xs2=[0.2,        0.3,      0.6,     0.9,    1.1,     1.3,    1.4,
1.6]
ys2=[0.050446, 0.098426, 0.33277, 0.72660, 1.0972, 1.5697, 1.8487,
2.5015]
```

a) Contruya el polinomio por minimos cuadrados de grado 1 y calcule el error.

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

# Ajuste polinómico de grado 1 (recta)
resultado = np.polyfit(xs2, ys2, 1)
a = resultado[0]
b = resultado[1]
```

```
# Línea de ajuste
x_line = np.linspace(min(xs2), max(xs2), 100)  # Generar más puntos
para una línea más suave
y_line = a*x_line+b

# Crear la gráfica
plt.figure(figsize=(8, 6))
plt.scatter(xs2, ys2, label="Datos", color="red")  # Puntos de los
datos
plt.plot(x_line, y_line, color="blue", label=f"Ajuste: y = {a:.2f}x
{b:.2f} ")  # Línea de ajuste

# Personalizar la gráfica
plt.xlabel("xs")
plt.ylabel("ys")
plt.title("Polinomio de grado 1")
plt.legend()  # Mostrar la leyenda
plt.grid(True)  # Añadir una cuadrícula
plt.show()
print(f"El polinomio resultante es: y = {round(a, 2)} X + {round(b,
2)}")
```

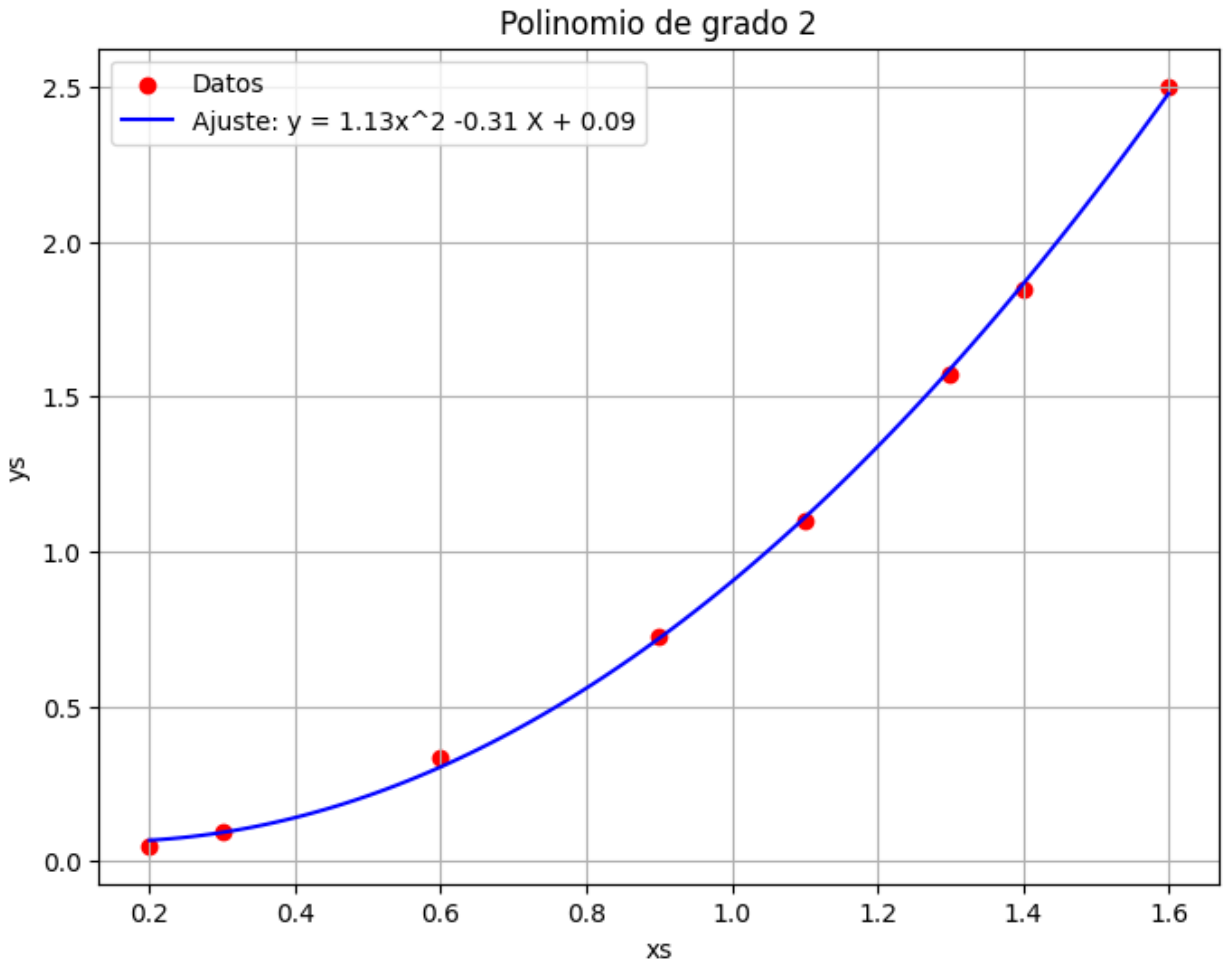



```
El polinomio resultante es: y = 1.67 X + -0.51
```

b) Contruya el polinomio por minimos cuadrados de grado 2 y calcule el error.

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

# Ajuste polinómico de grado 2 (recta)
resultado = np.polyfit(xs2, ys2, 2)
a = resultado[0]
b = resultado[1]
c= resultado[2]

# Línea de ajuste
x_line = np.linspace(min(xs2), max(xs2), 100)  # Generar más puntos para una línea más suave
y_line = a*x_line**2 + b*x_line + c

# Crear la gráfica
plt.figure(figsize=(8, 6))
```

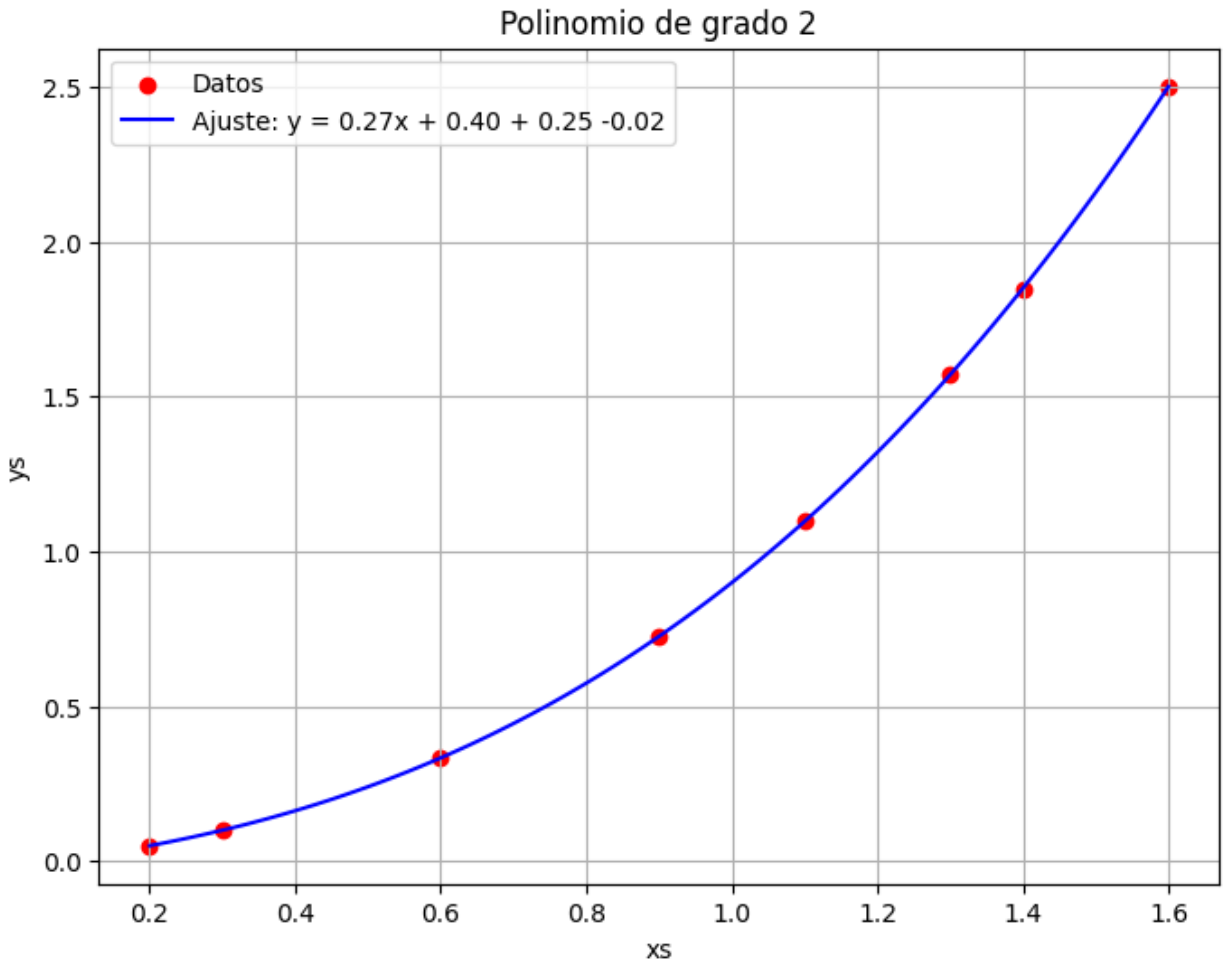
```
plt.scatter(xs2, ys2, label="Datos", color="red")  # Puntos de los
datos
plt.plot(x_line, y_line, color="blue", label=f"Ajuste: y = {a:.2f}x^2
{b:.2f} X + {c:.2f}")  # Línea de ajuste

# Personalizar la gráfica
plt.xlabel("xs")
plt.ylabel("ys")
plt.title("Polinomio de grado 2")
plt.legend()  # Mostrar la leyenda
plt.grid(True)  # Añadir una cuadrícula
plt.show()
print(f"El polinomio resultante es: y = {round(a, 2)} X**2 {round(b,
2)} + {round(c,2)}")
```




```
El polinomio resultante es: y = 1.13 X**2 -0.31 + 0.09
```

c) Contruya el polinomio por minimos cuadrados de grado 3 y calcule el error.

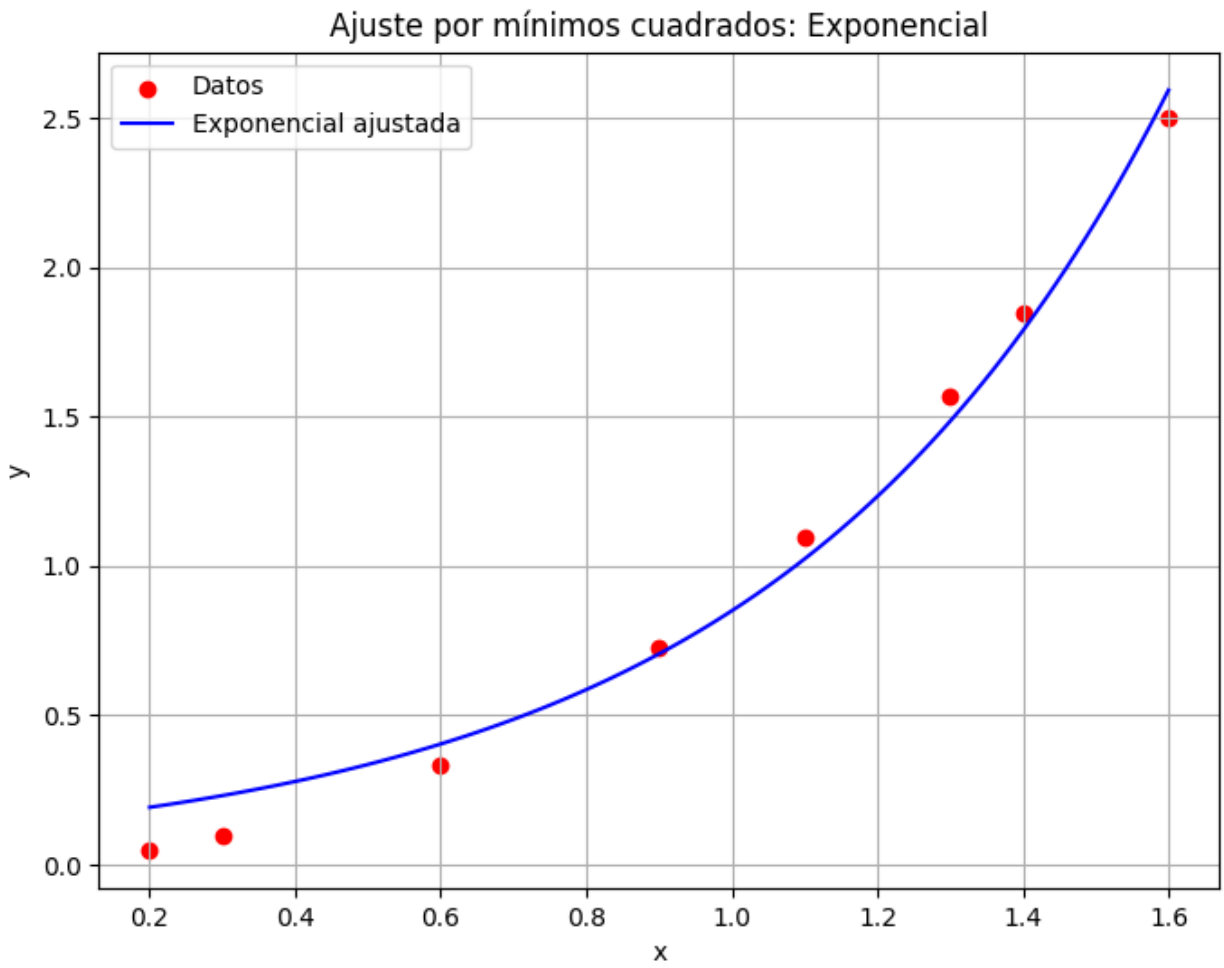
```python
import numpy as np
import matplotlib.pyplot as plt

# Ajuste polinómico de grado 3 (recta)
resultado = np.polyfit(xs2, ys2, 3)
a = resultado[0]
b = resultado[1]
c= resultado[2]
d= resultado[3]

# Línea de ajuste
x_line = np.linspace(min(xs2), max(xs2), 100)  # Generar más puntos
para una línea más suave
y_line = a*x_line**3 + b*x_line**2 + c*x_line + d

# Crear la gráfica
plt.figure(figsize=(8, 6))
plt.scatter(xs2, ys2, label="Datos", color="red")  # Puntos de los
datos
plt.plot(x_line, y_line, color="blue", label=f"Ajuste: y = {a:.2f}x +
{b:.2f} + {c:.2f} {d:.2f}")  # Línea de ajuste

# Personalizar la gráfica
plt.xlabel("xs")
plt.ylabel("ys")
plt.title("Polinomio de grado 2")
plt.legend()  # Mostrar la leyenda
plt.grid(True)  # Añadir una cuadrícula
plt.show()
print(f"El polinomio resultante es: y = {round(a, 2)} X**3 + {round(b,
2)}X**2 {round(c,2)}X + {round(d,2)}")
```




```
El polinomio resultante es: y = 0.27 X**3 + 0.4X**2 0.25X + -0.02
```

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

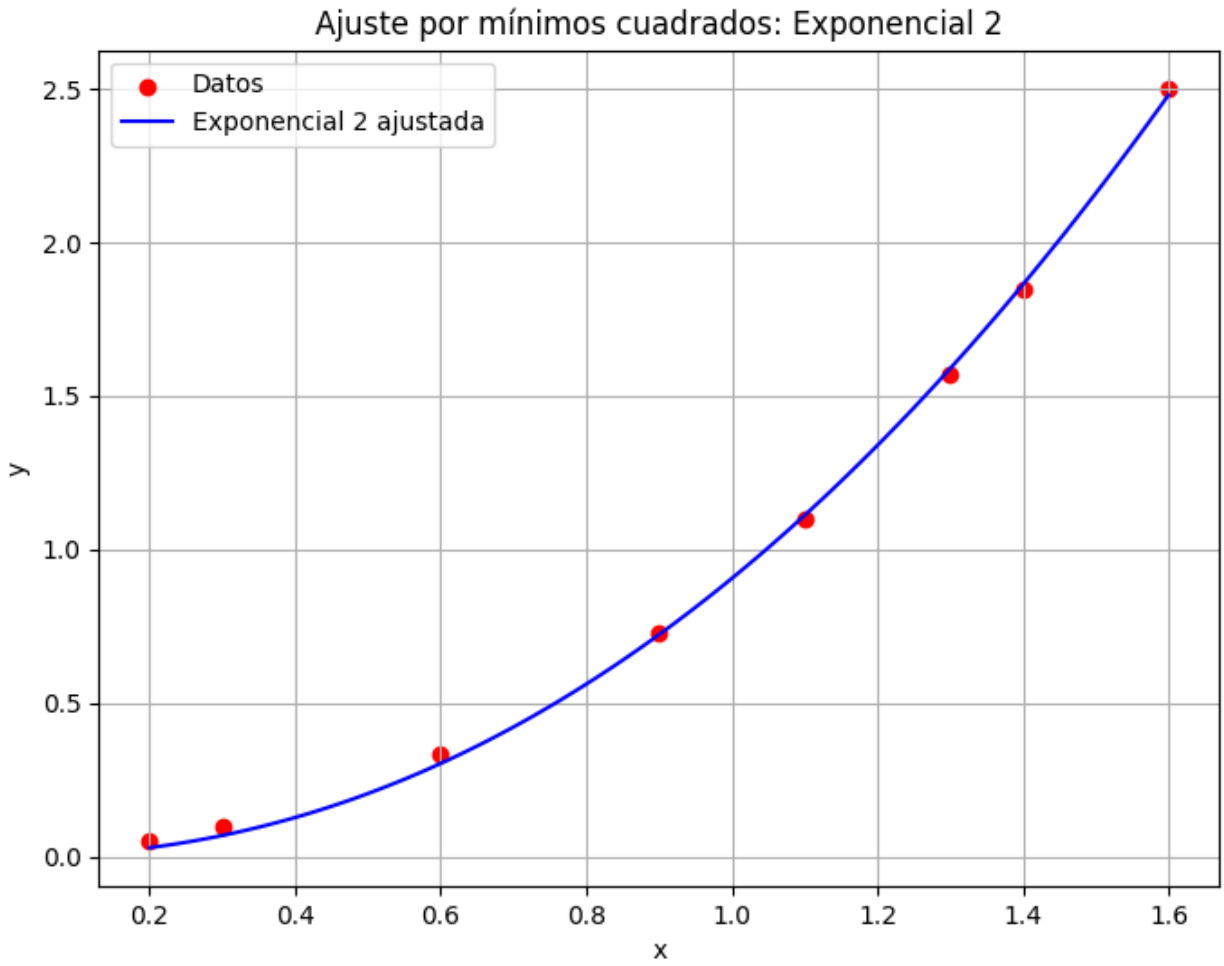
```
xs_2=np.array([0.2,        0.3,      0.6,     0.9,    1.1,     1.3,
1.4,   1.6])
ys_2=np.array([0.050446, 0.098426, 0.33277, 0.72660, 1.0972, 1.5697,
1.8487, 2.5015])

def funcion_exp3(x,a,b):
    return b*np.exp(a*x)

paremtros_2= ajuste_minimos_cuadrados(xs_2,ys_2,
funcion_exp3,nombre_funcion="Exponencial")
print(f"Funcion:
{round( paremtros_2[1],2)}e^{round(paremtros_2[0],2)}*X")
```




```
Funcion: 0.13e^1.86*X
```

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

```
def funcion_exp4(x,a,b):
    return b*x**a

parametros_3= ajuste_minimos_cuadrados(xs_2,ys_2,
funcion_exp4,nombre_funcion="Exponencial 2")
print(f"Funcion:
{round( parametros_3[1],2)}X^{round(parametros_3[0],2)}")
```



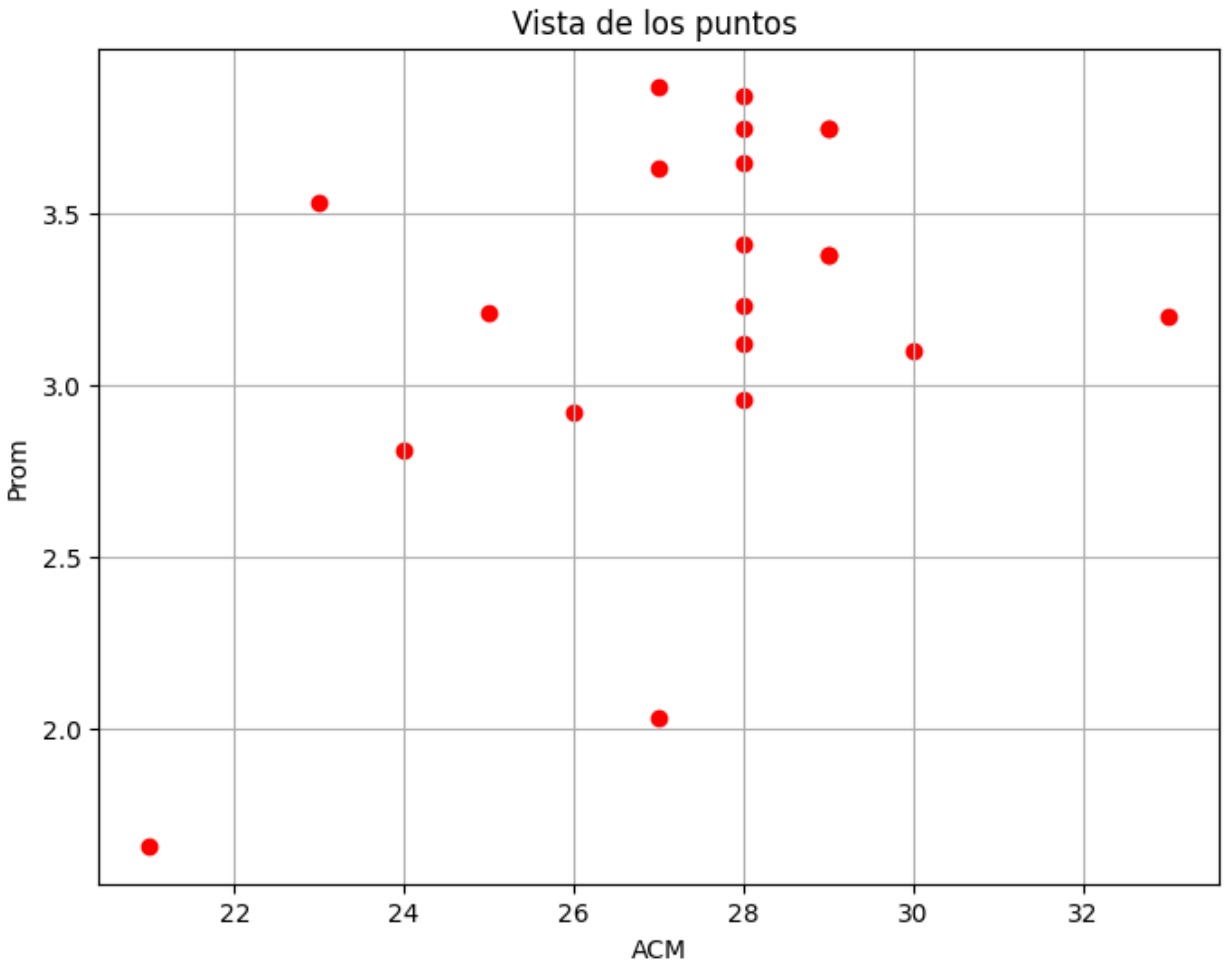

```
Funcion: 0.91X^2.14
```

1. 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

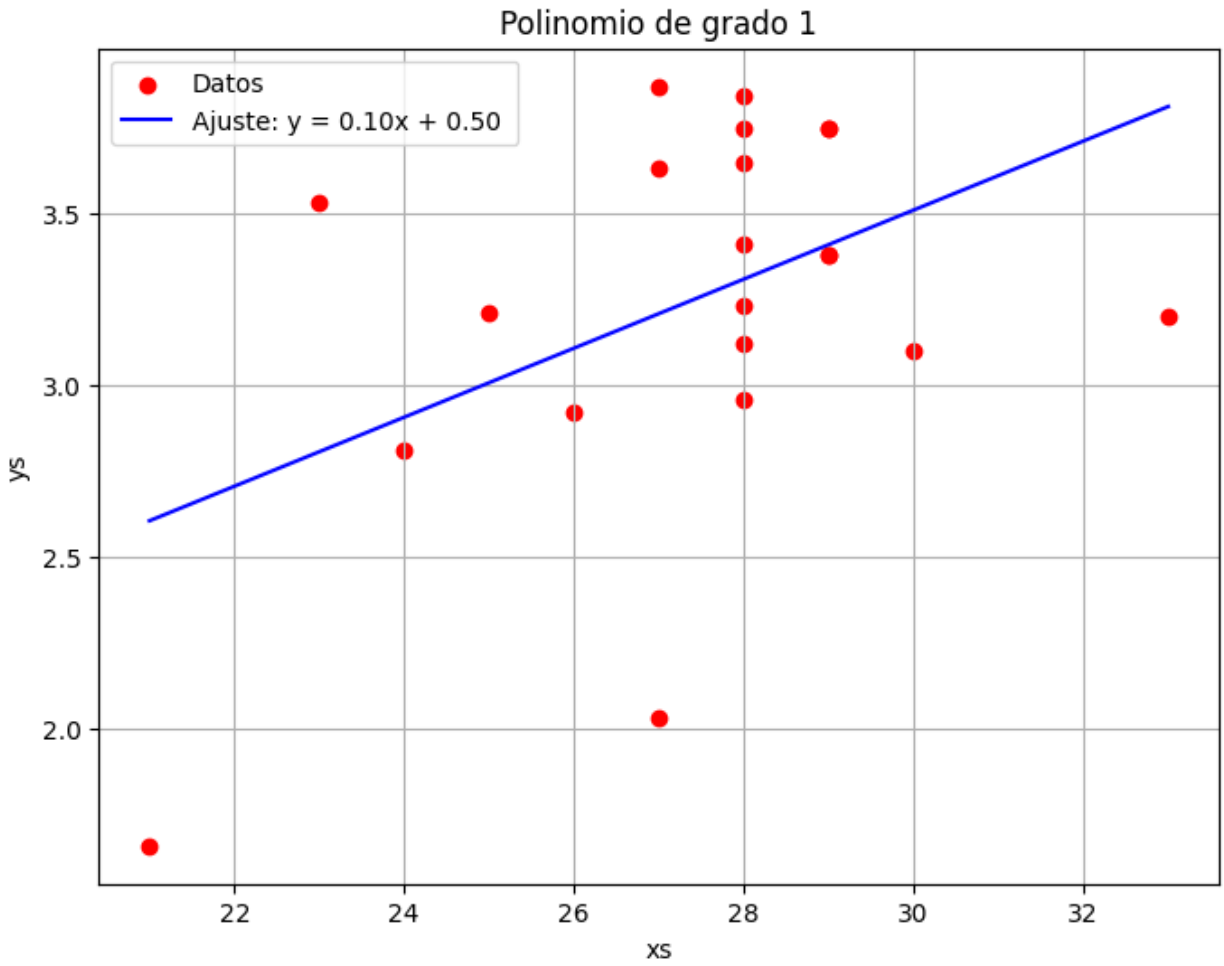
Se definen los puntos de la prueba de ACM "X" y sus promedios de puntos "Y" para ser analizados por minimos cuadrados.

```
ACM = [28,25,28,
       27,28,33,
       28,29,29,
       23,27,29,
       28,27,29,
       21,28,28,
       26,30,24]
P = [3.84,  3.21, 3.23,
```

```
      3.63, 3.75, 3.20,
      3.41, 3.38, 3.38,
      3.53, 2.03, 3.75,
      3.65, 3.87, 3.75,
      1.66, 3.12, 2.96,
      2.92, 3.10, 2.81]

import matplotlib.pyplot as plt

## Graficamos los puntos para observar la distribucion de los datos
plt.figure(figsize=(8, 6))
plt.scatter(ACM, P, label="ACM", color="red")  # Puntos de los datos
plt.title("Vista de los puntos ")
plt.xlabel("ACM")
plt.ylabel("Prom")
plt.grid(True)  # Añadir una cuadrícula
plt.show()

##Realizamos otro grafico con el ajuste de minimos cuadradfos con un
polinomio de grado 1
# Ajuste polinómico de grado 1 (recta)
resultado = np.polyfit(ACM, P, 1)
a = resultado[0]
b = resultado[1]
print(f"El polinomio resultante es: y = {round(a, 3)} X + {round(b,
3)}")
# Línea de ajuste
x_line = np.linspace(min(ACM), max(ACM), 100)  # Generar más puntos
para una línea más suave
y_line = a*x_line+b

# Crear la gráfica
plt.figure(figsize=(8, 6))
plt.scatter(ACM, P, label="Datos", color="red")  # Puntos de los datos
plt.plot(x_line, y_line, color="blue", label=f"Ajuste: y = {a:.2f}x +
{b:.2f} ")  # Línea de ajuste

# Personalizar la gráfica
plt.xlabel("xs")
plt.ylabel("ys")
plt.title("Polinomio de grado 1")
plt.legend()  # Mostrar la leyenda
plt.grid(True)  # Añadir una cuadrícula
plt.show()
```




```
El polinomio resultante es: y = 0.1 X + 0.496
```



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).

```
##Puntos
Pp= [4800, 3700, 3400, 2800, 1900] #Pp -> Peso Promedio
Pr= [3.1, 4.0, 5.2, 6.4, 9.6] # Pr-> % de representacion

##Analisis por minimos cuadrados:

# Ajuste polinómico de grado 1 (recta)
resultado = np.polyfit(Pp, Pr, 1)
a = resultado[0]
b = resultado[1]
print(f"El polinomio resultante es: y = {round(a, 8)} X + {round(b,
8)}")

x = np.linspace(Pp, Pr, 100)
```

```
y= a*x + b

plt.figure(figsize=(8, 6))
plt.scatter(Pp, Pr, label="Datos", color="red")
plt.plot(x, y, color="purple", label="Datos")
plt.grid(True)
plt.xlabel("Peso promedio (LB)")
plt.ylabel(" Porcentaje de representacion")
plt.title("Ajuste por minimos cuadrados ")
plt.show()

El polinomio resultante es: y = -0.00225497 X + 13.14649957
[-2.25496975e-03  1.31464996e+01]
```