Nombre: Bryan Mauricio Morales

Tema: Mínimos cuadrados

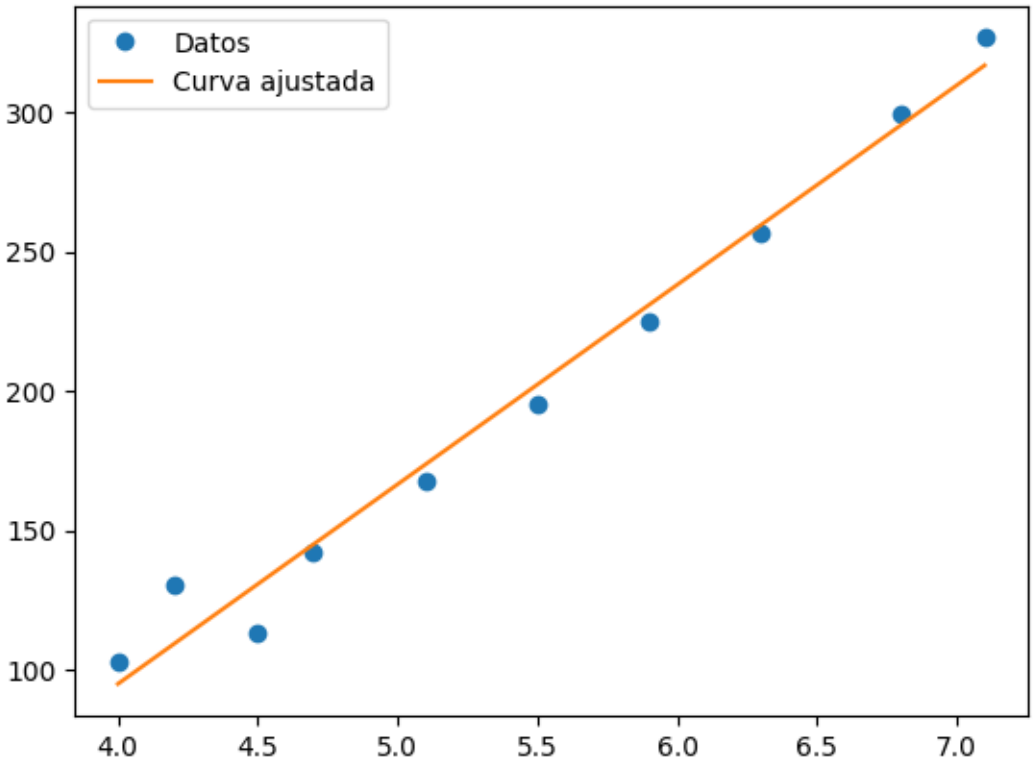
```
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

def linear(x, a, b):
    return a + b * x

def quadratic(x, a, b, c):
    return a + b * x + c * x**2

def cubic(x, a, b, c, d):
    return a + b * x + c * x**2 + d * x**3

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

def power(x, a, b):
    return b * x**a

def func_selector(func_type):
    if func_type == 'linear':
        return linear
    elif func_type == 'quadratic':
        return quadratic
    elif func_type == 'cubic':
        return cubic
    elif func_type == 'exponential':
        return exponential
    elif func_type == 'power':
        return power
    else:
        raise ValueError("Función no reconocida")

def func_representation(func_type, popt):
    if func_type == 'linear':
        return f"{popt[0]} + {popt[1]} * x"
    elif func_type == 'quadratic':
        return f"{popt[0]} + {popt[1]} * x + {popt[2]} * x^2"
    elif func_type == 'cubic':
```

```
        return f"{popt[0]} + {popt[1]} * x + {popt[2]} * x^2 + {popt[3]} * x^3"
    elif func_type == 'exponential':
        return f"{popt[1]} * exp({popt[0]} * x)"
    elif func_type == 'power':
        return f"{popt[1]} * x^{popt[0]}"
    else:
        raise ValueError("Función no reconocida")

def draw_fun (type, x, y):
    func_type = type
    func = func_selector(func_type)
    popt, pcov = curve_fit(func, x, y)
    print(func_representation(func_type=func_type, popt=popt))
    plt.plot(x, y, 'o', label='Datos')
    plt.plot(x, func(x, *popt), '-', label='Curva ajustada')
    plt.legend()
    plt.show()
```

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	225.87	256.73	299.5	326.72

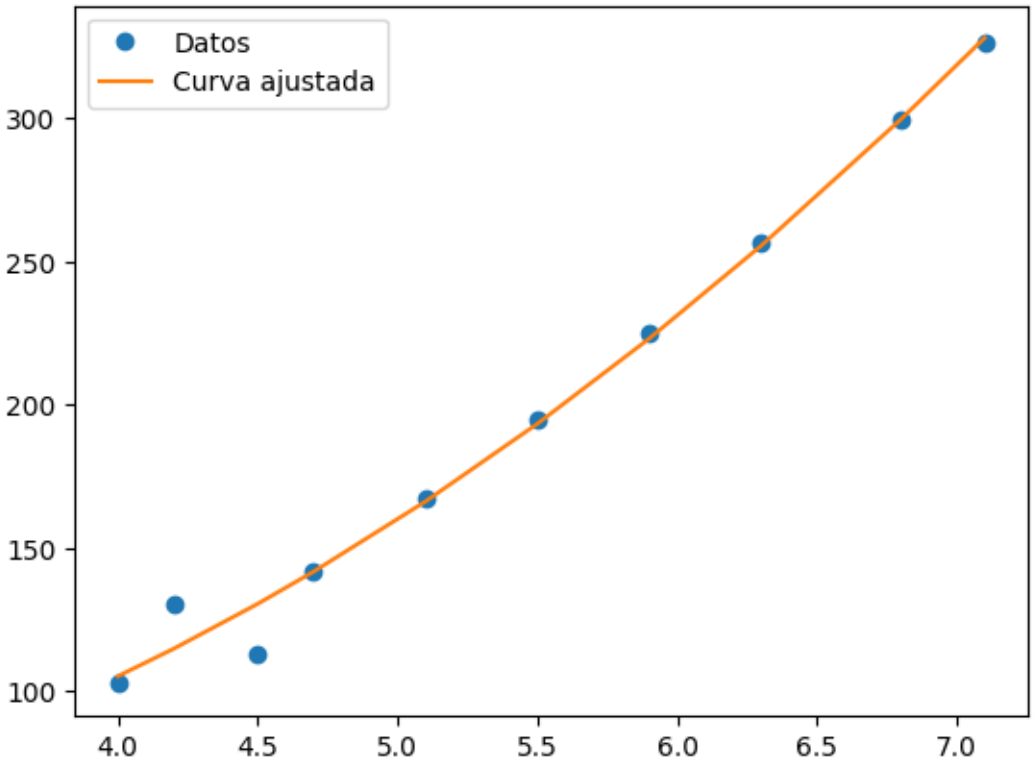
```
xdata = np.array([4.0, 4.2, 4.5, 4.7, 5.1, 5.5, 5.9, 6.3, 6.8, 7.1])
ydata = np.array([102.56, 130.11, 113.18, 142.05, 167.53, 195.14, 224.87, 256.73, 299.5, 326
func_type = 'linear'

draw_fun(type=func_type, x=xdata, y= ydata)
```

```
-191.57241823897462 + 71.61024366504328 * x
```



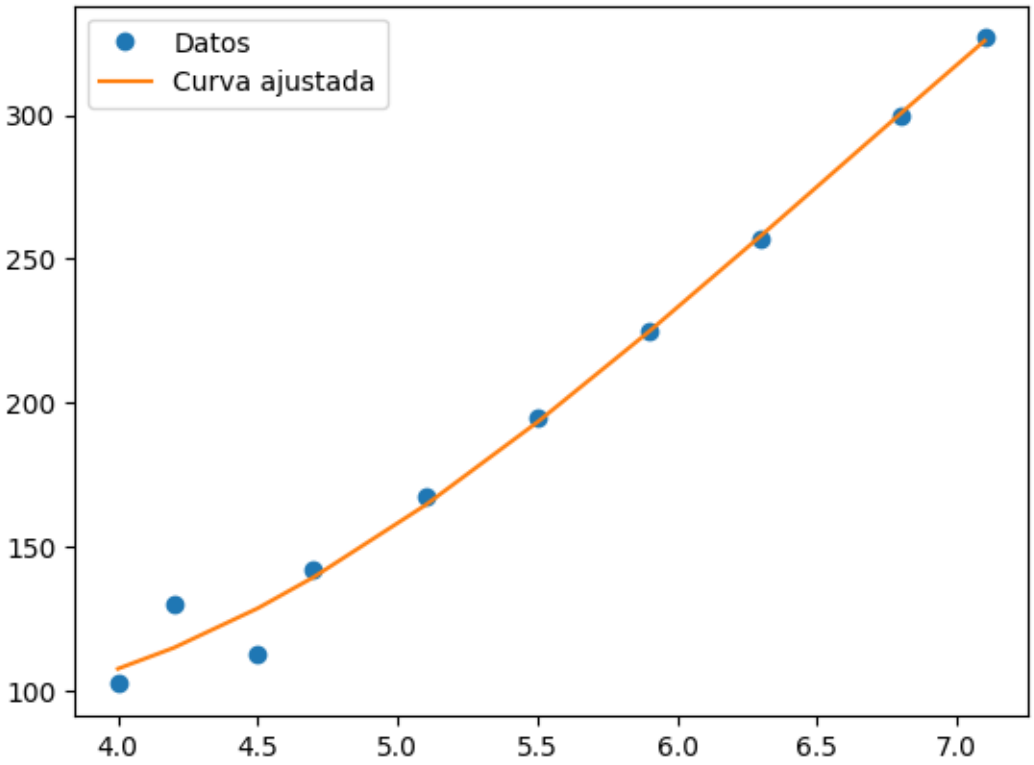

```
func_type = 'quadratic'

draw_fun(type=func_type, x=xdata, y= ydata)
```

```
51.00078817885329 + -19.30860335585865 * x + 8.217072282443924 * x^2
```



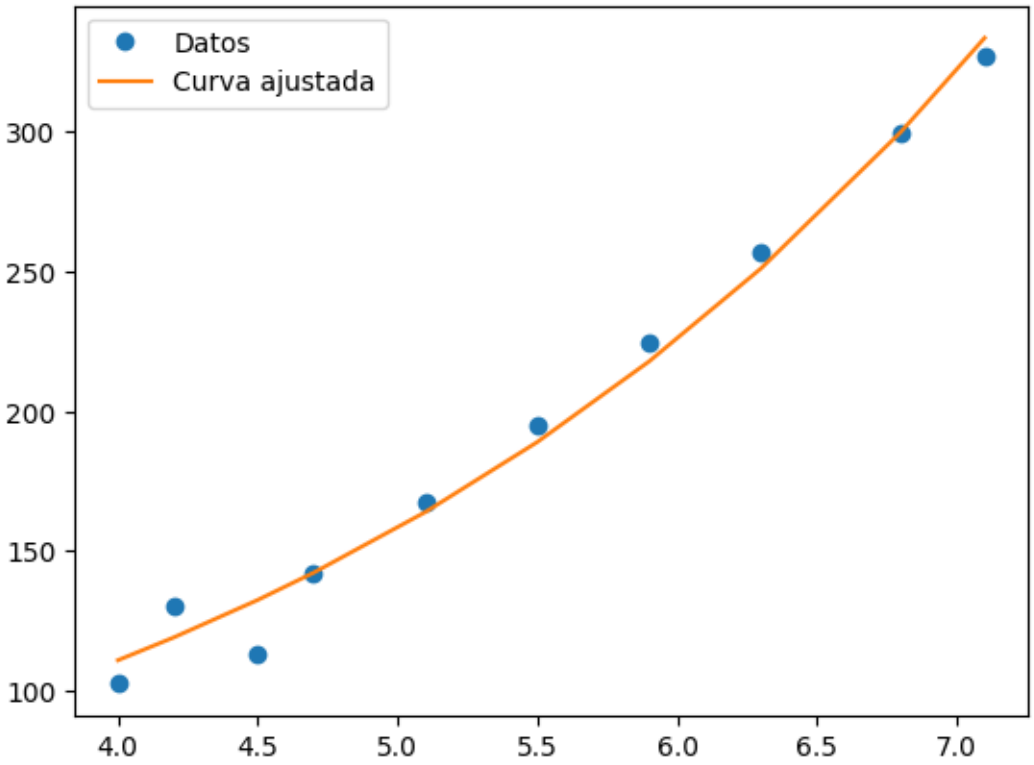

```
func_type = 'cubic'

draw_fun(type=func_type, x=xdata, y= ydata)
```

```
469.16550431978425 + -254.87604151357237 * x + 51.56118783626427 * x^2 + -2.606852575592616 *
```



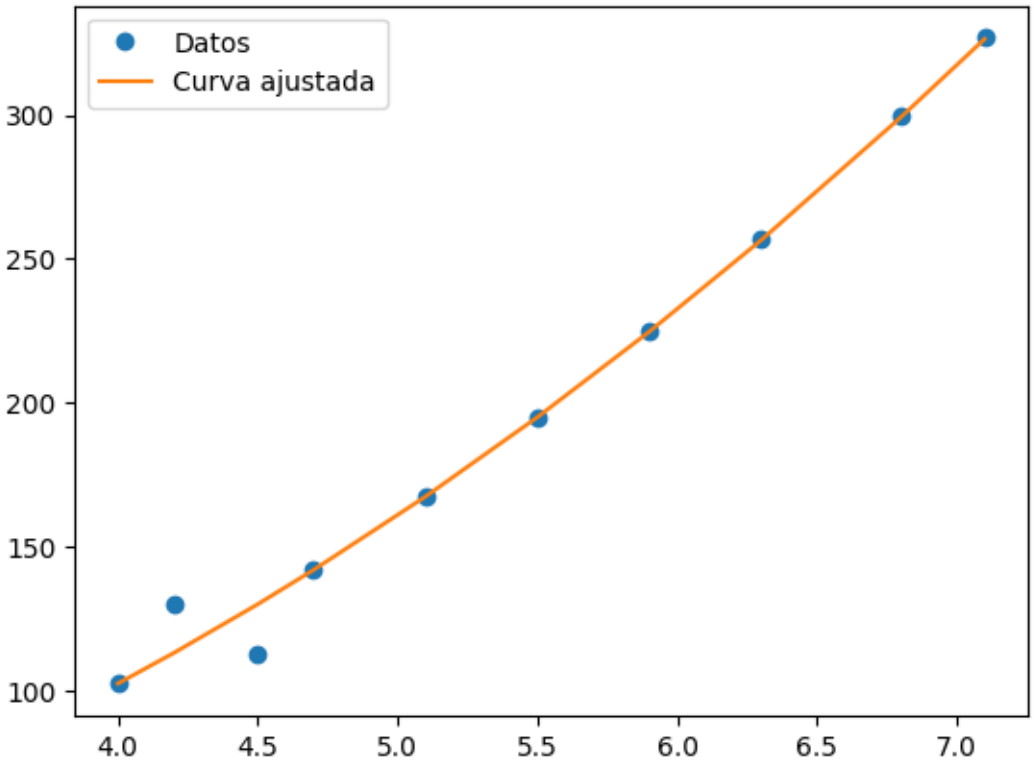

```
func_type = 'exponential'

draw_fun(type=func_type, x=xdata, y= ydata)
```

```
26.840771540960706 * exp(0.3549271021279226 * x)
```



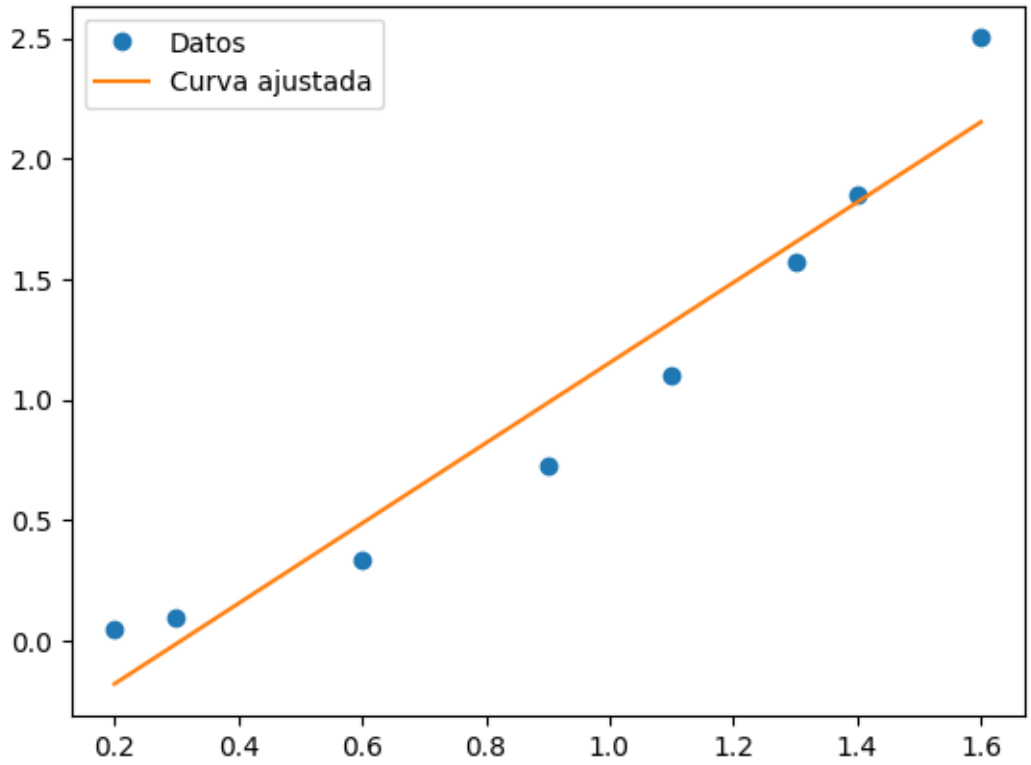

```
func_type = 'power'

draw_fun(type=func_type, x=xdata, y= ydata)
```

```
6.283966084740469 * x^2.015190342585292
```

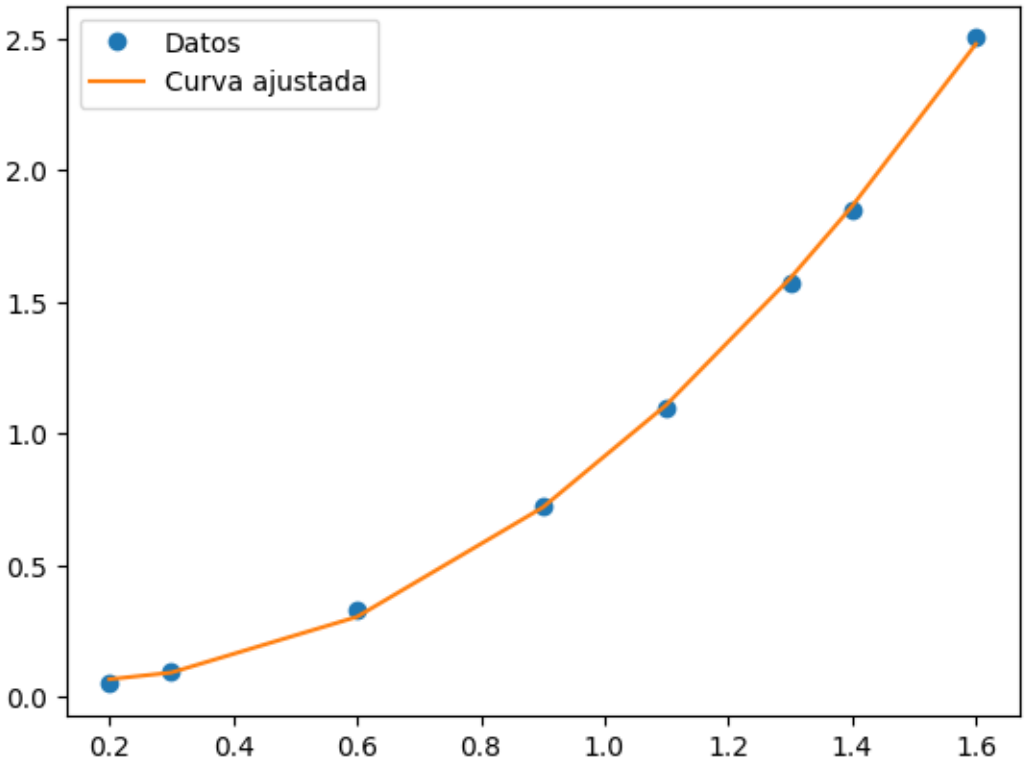


x_i	0.2	0.3	0.6	0.9	1.1	1.3	1.4	1.6
y_i	0.050446	0.098426	0.33277	0.72660	1.0972	1.5697	1.8487	2.5015

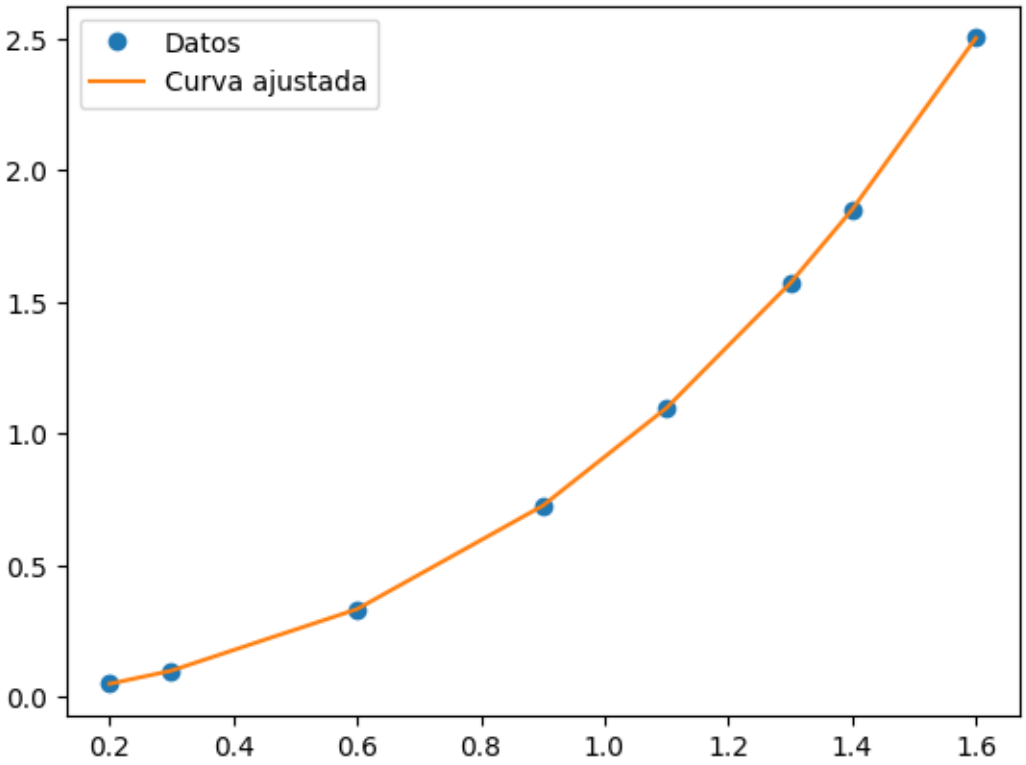
```
xdata = np.array([0.2,0.3,0.6,0.9,1.1,1.3,1.4,1.6])
ydata = np.array([0.050446,0.098426,0.33277,0.72660,1.0972,1.5697,1.8487,2.5015])
func_type = 'linear'

draw_fun(type=func_type, x=xdata, y= ydata)
```



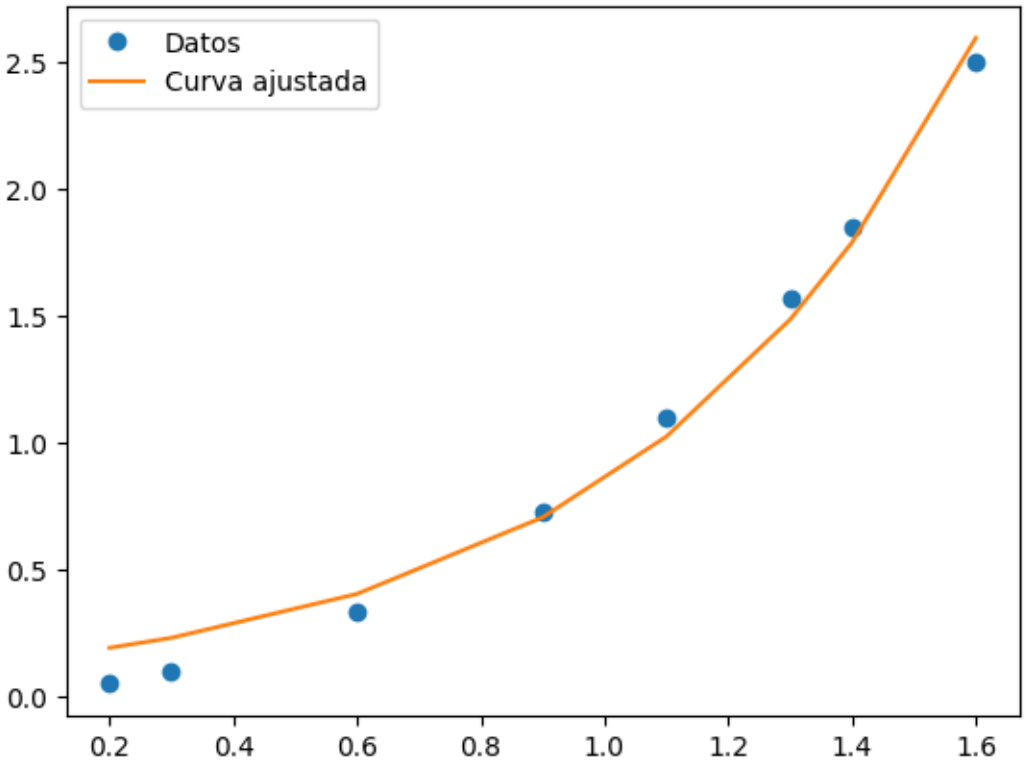

```
func_type = 'quadratic'

draw_fun(type=func_type, x=xdata, y= ydata)
```



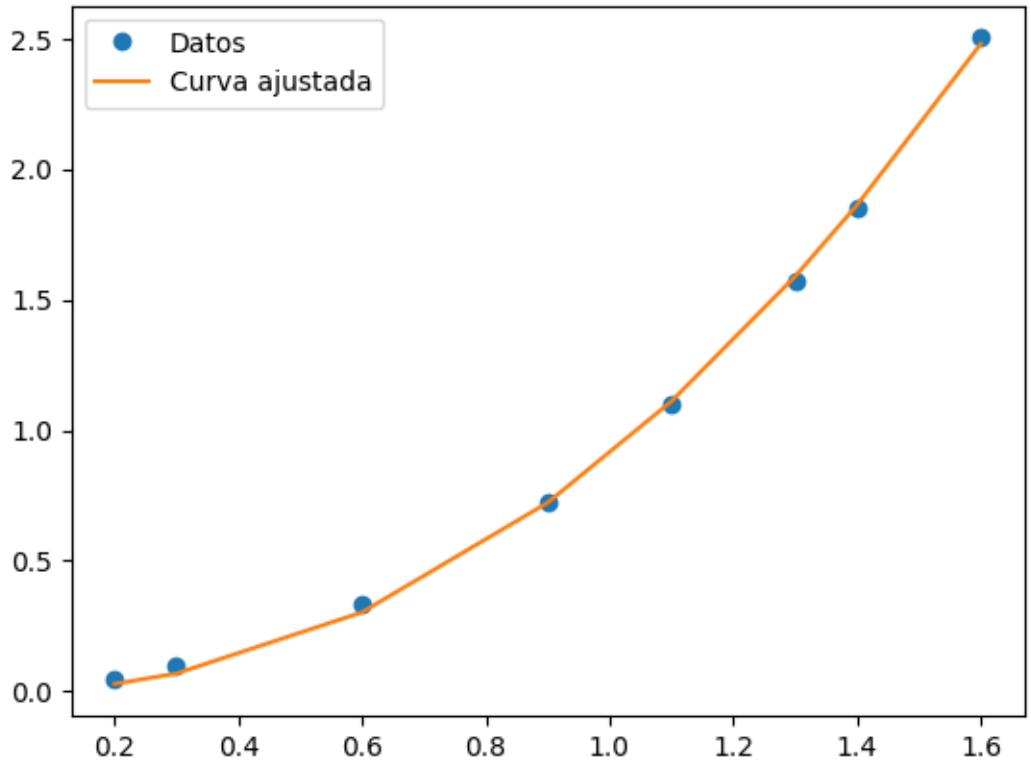

```
func_type = 'cubic'

draw_fun(type=func_type, x=xdata, y= ydata)
```




```
func_type = 'exponential'

draw_fun(type=func_type, x=xdata, y= ydata)
```



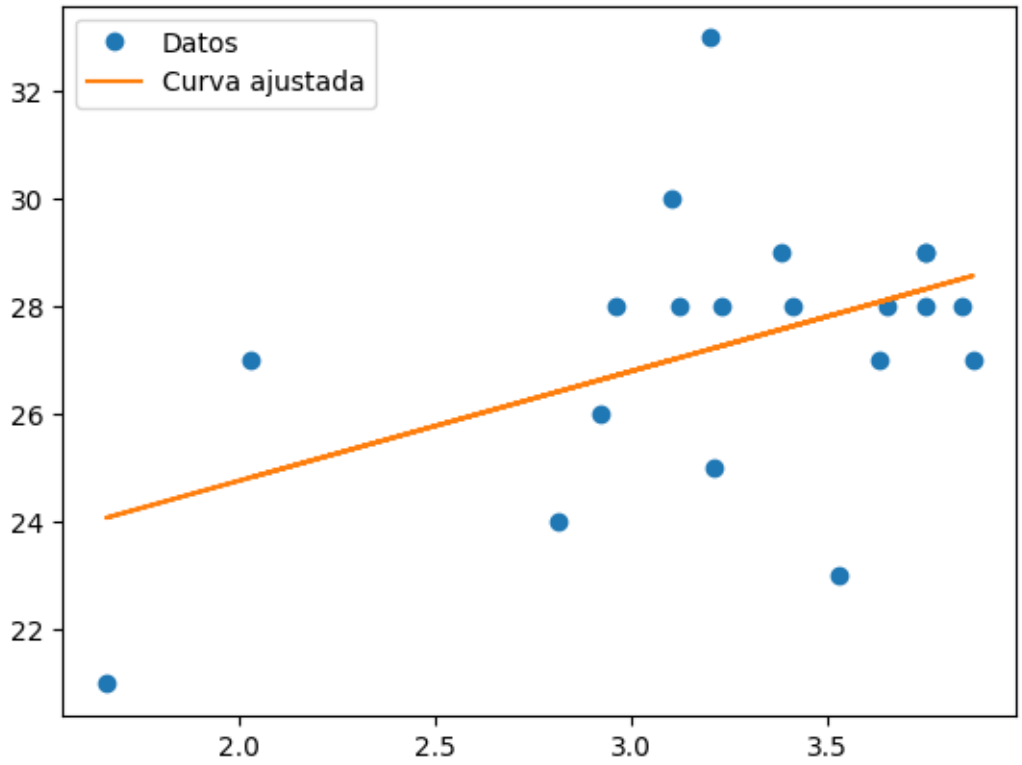

```
func_type = 'power'

draw_fun(type=func_type, x=xdata, y= ydata)
```



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

```
x_data = np.array([3.84, 3.21, 3.23, 3.63, 3.75, 3.2, 3.41, 3.38, 3.53, 2.03, 3.75, 3.65, 3.8
y_data = np.array([28, 25, 28, 27, 28, 33, 28, 29, 23, 27, 29, 28, 27, 29, 21, 28, 28, 26, 30
func_type = 'linear'

draw_fun(type=func_type, x=x_data, y= y_data)
```

```
20.705553156808858 + 2.035323099750353 * x
```



El siguiente conjunto de datos, presentado al Suncomité Antimomopolio 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 sería).

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

```
xdata = np.array([4800, 3700, 3400, 2800, 1900])
ydata = np.array([3.1, 4.0, 5.2, 6.4, 9.6])
func_type = 'power'

draw_fun(type=func_type, x=xdata, y= ydata)
```

```
75412.8806935697 * x^-1.1864962247787003
```