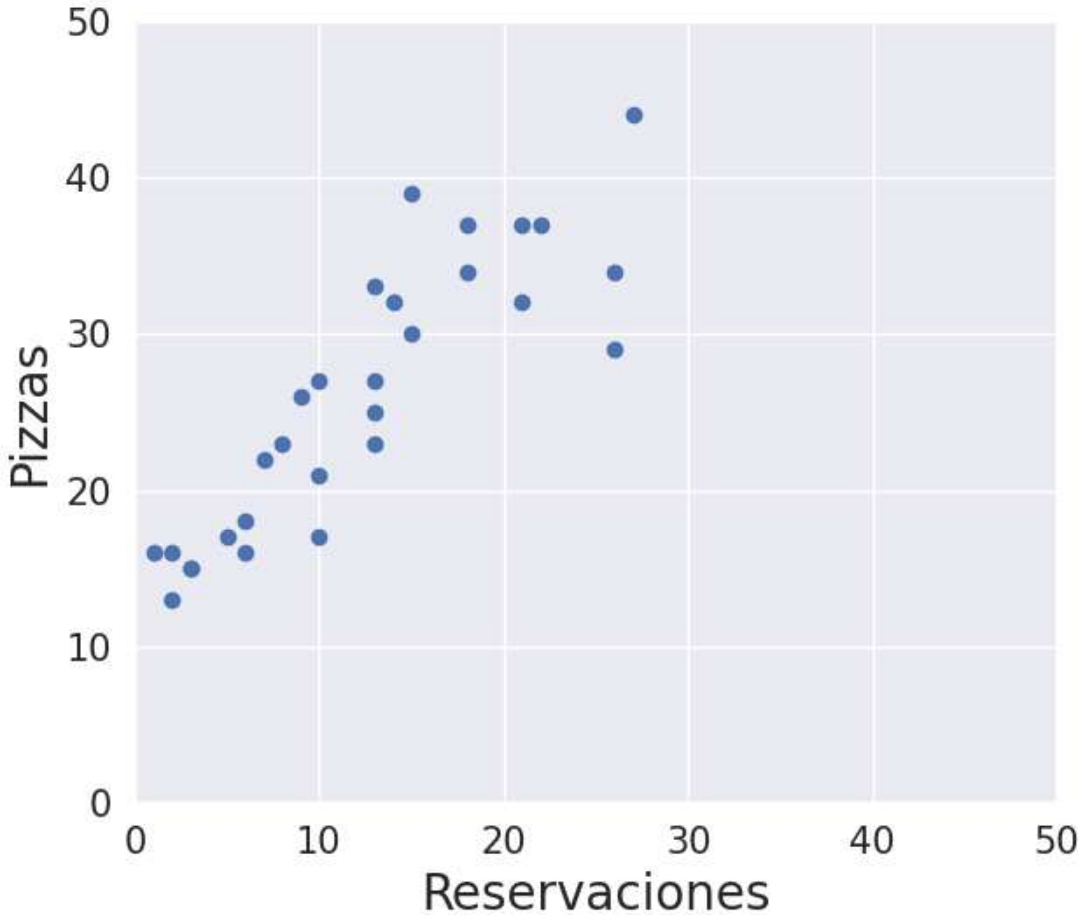
Tarea 15: Regresión lineal

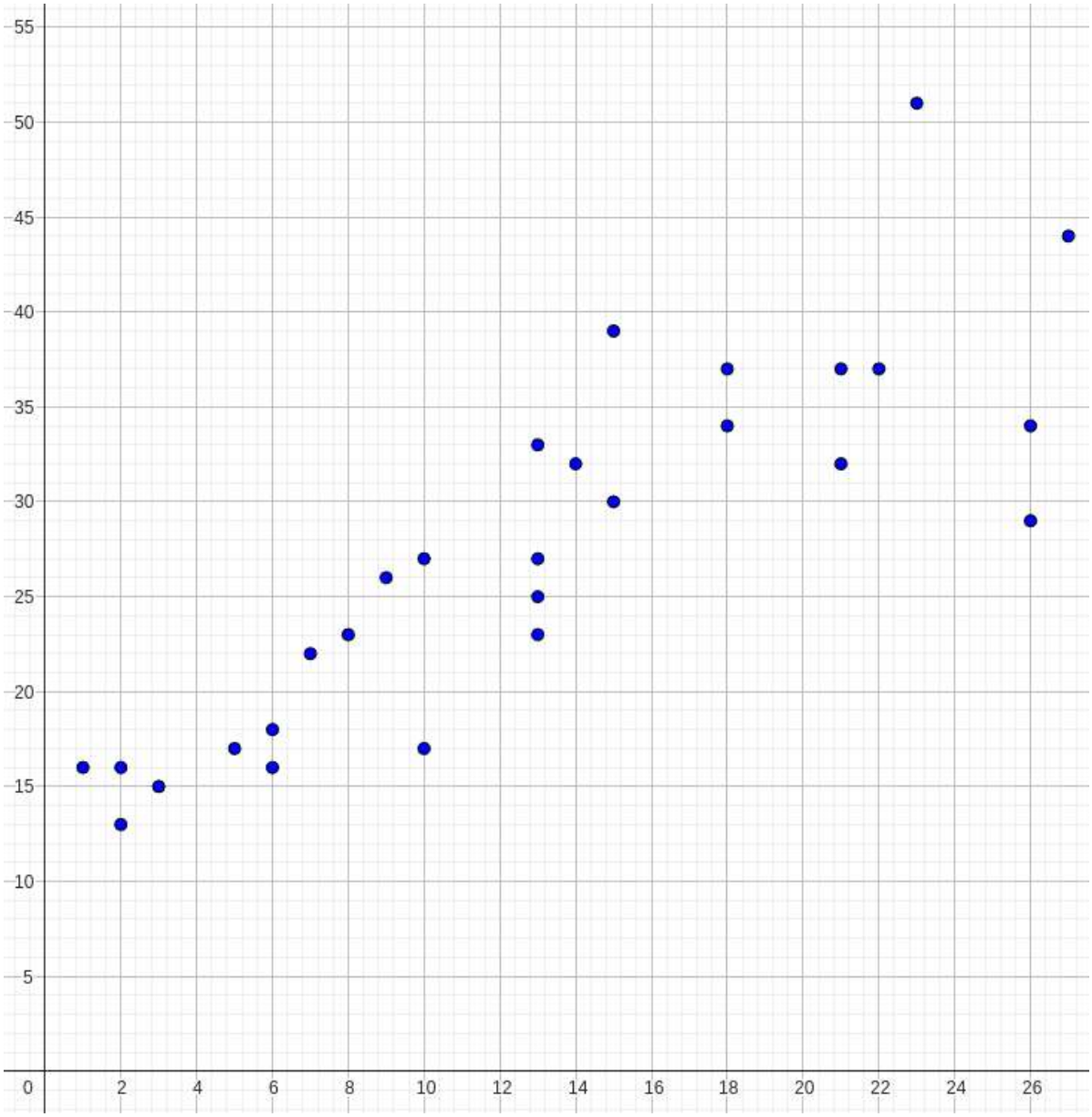
👤 Guillermo Daniel Mestas Alvarez

En esta tarea tenemos el problema planteado donde Roberto, dueño de una pizzería, necesita una forma más precisa de predecir la cantidad de pizzas que debe preparar cada noche. Actualmente, se basa en el número de reservas, pero sabe que esta relación no es perfecta, ya que muchos clientes no reservan o piden otros platos.

1. El primer paso fue analizar la relación entre las reservas y pizzas, la cual es que **a más cantidad de reservas habrá más pizzas**, teniendo eso en cuenta realizamos un gráfico de dispersión con los datos del archivo `pizza.txt`.



El anterior gráfico también lo graficamos en Geogebra, esto convirtiendo el archivo `txt` en uno de Excel, después copiamos los datos y lo pegamos en Geogebra, de esta manera graficando el gráfico de dispersión:



2. Después realizamos una función para poder predecir la cantidad de pizzas, esto mediante el **Aprendizaje supervisado**, el cual se basa en etiquetas, es decir, las reservas y pizzas donde tengo una cantidad de reservas, lo que me da como salida la cantidad de pizzas.

   Específicamente utilizamos la **Regresión lineal**, en la cual se formula es:

   $$y = mx + b$$

   Teniendo la fórmula creamos la función de `predecir`, pero no estamos considerando lo que sería el sesgo `b`:

```
X, Y = np.loadtxt('pizza.txt', skiprows=1, unpack=True)

def predecir(X, w):
    return (X * w)
```

3. Después la perdida, la cual se basa en:

$$error = predecir(x, w) - Y$$

```
def perdida(X, Y, w):
    return np.average((predecir(X, w) - Y)**2)
```

4. Continuamos con la función de entrenar:

```
def entrenar(X, Y, iteraciones, lr): # learning rate (lr)
    w = 0
    for i in range(iteraciones):
        perdida_actual = perdida(X, Y, w)
        print(f"Iteracion {i:4d} => Perdida: {perdida_actual
        if perdida(X, Y, w + lr) < perdida_actual:
            w += lr
        elif perdida(X, Y, w - lr) < perdida_actual:
            w -= lr
        else:
            return w

    raise Exception(f"No se puede convertir entre {iteracione
```

5. Teniendo todo lo anterior solo faltaría ejecutar el entrenamiento para obtener la inclinación de la recta `w` y mostrarla en un gráfico:

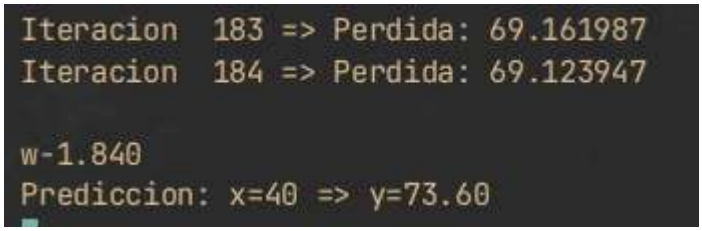
```
w = entrenar(X, Y, iteraciones=10000, lr=0.01)

print(f"\nw-{w:.3f}")
```

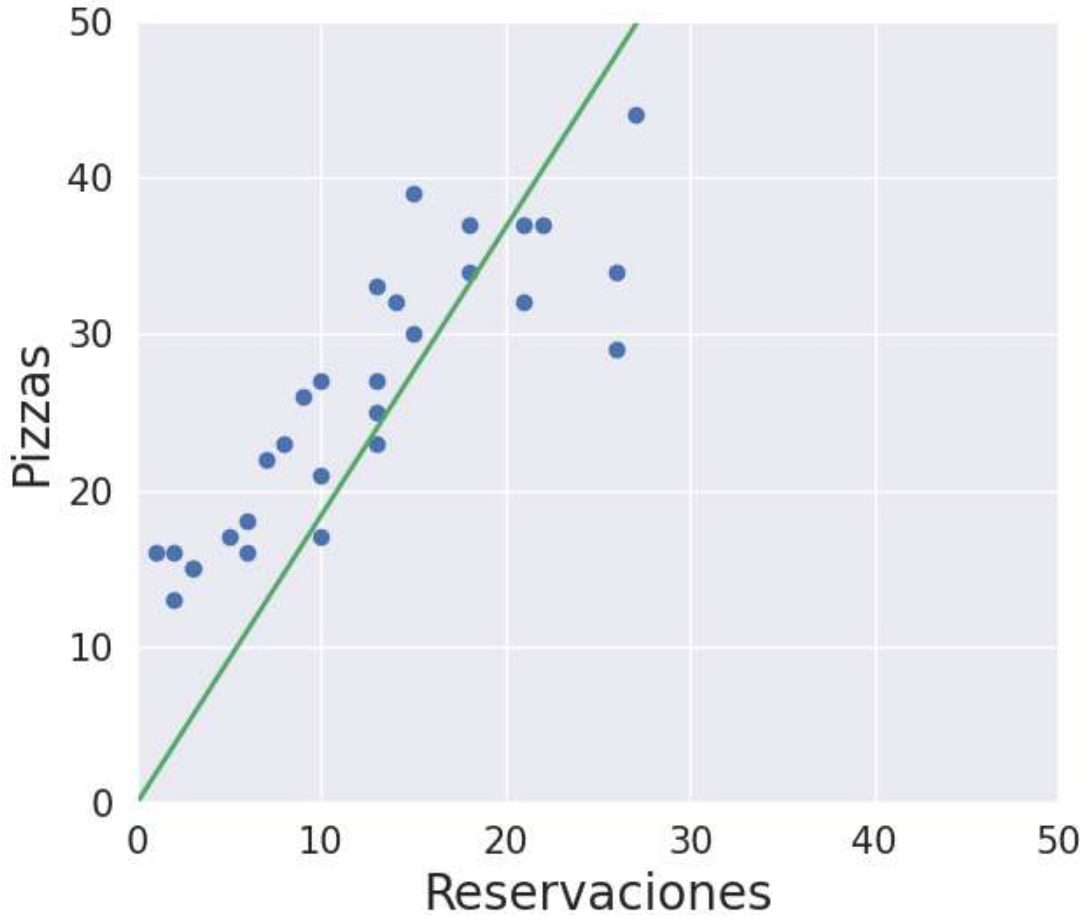
```
print(f"Prediccion: x={40} => y={predecir(40, w):.2f}")

sb.set()
plt.plot(X, Y, "bo")
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.xlabel('Reservaciones', fontsize=20)
plt.ylabel('Pizzas', fontsize=20)
x_edge, y_edge = 50, 50
plt.axis([0, x_edge, 0, y_edge])
plt.plot([0, x_edge], [0, predecir(x_edge, w)], linewidth=2.(
plt.show()
```

Primero en la consola veremos las iteraciones realizadas, la inclinación de la recta `w` que es: `1.840`, debajo está una predicción en la cual vemos que `x` que es la cantidad de reservas, teniendo como resultado que Roberto debería de realizar unas `73` pizzas en promedio `y`.



En el siguiente gráfico se muestra la inclinación que se mostró antes, además de como este está interceptando la gran mayoría de puntos, dándonos a entender las predicciones son mas adecuadas.



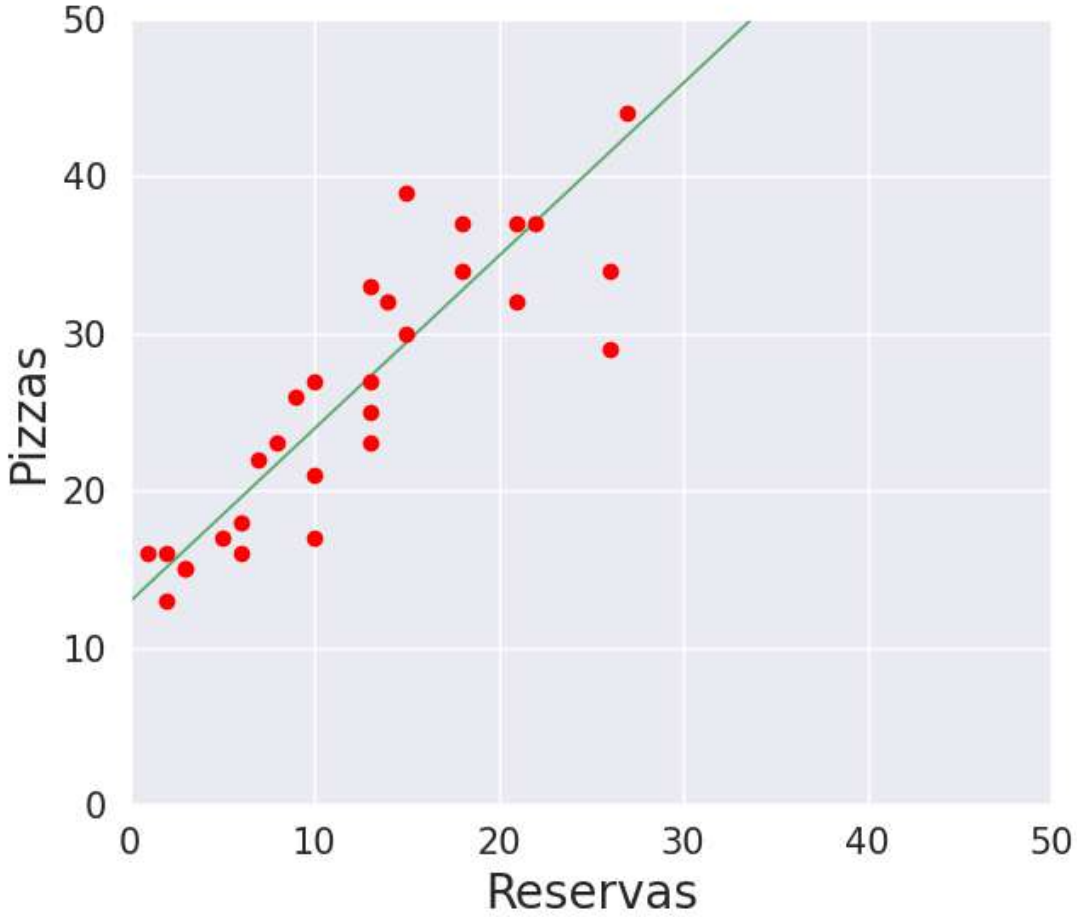
6. Por último tenemos que agregar ahora si el sesgo `b`, para ello modificamos las funciones anteriores agregando el parámetro `b`, además de igualar el valor de la inclinación que devuelve la función de `entrenar` a `b`:

```
b = 0  # Inicializamos el sesgo
w, b = entrenar(x, y, iteraciones=10000, lr=0.01, b=b)
```

Además, para Mostar `b` en el gráfico tenemos que cambiar:

```
plt.plot([0, x_edge], [b, predecir(x_edge, w, b)], linewidth=
```

De esta manera, teniendo el mejor `w` como `b`, pudiendo realizar mejores predicciones en cuanto a la cantidad de pizzas a realizar.


Pizzas
Reservas
0
10
20
30
40
50