

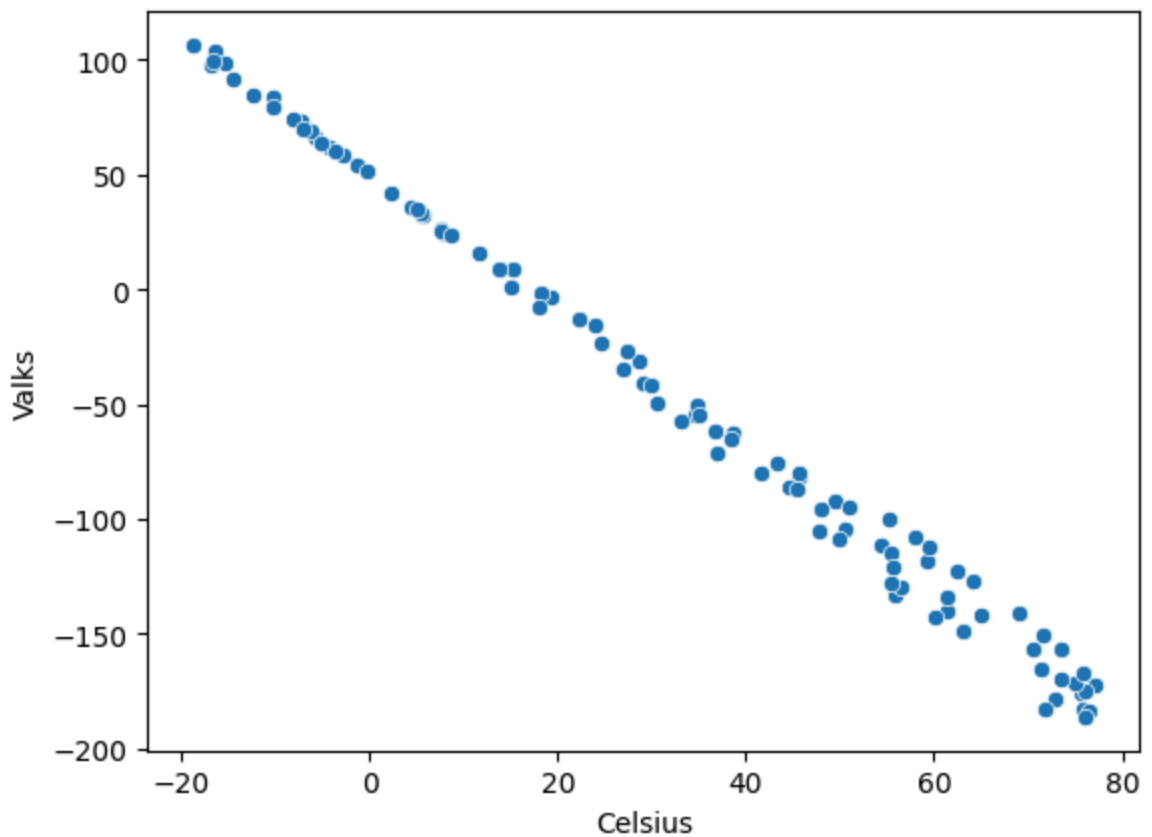
```
In [21]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [22]: datos = pd.read_csv('Valhalla23.csv')
datos.head()
```

```
Out[22]:
```

	Celsius	Valks
0	61.4720	-139.740
1	70.5790	-156.600
2	-7.3013	73.269
3	71.3380	-165.420
4	43.2360	-75.835

```
In [23]: sns.scatterplot(data = datos, x= 'Celsius', y = 'Valks')
plt.show;
```



```
In [24]: # Crear lista con los hiper-parámetros iniciales (thetas)
theta = [60, -10]
theta_nuevo = theta.copy()
# Cargar el valor del learning rate (alpha)
alpha = 0.0001
```

```
In [25]: # Crear función lambda para la función de hipótesis
prediccion = lambda b,m,x: b + m*x

# Calcular el total de muestras a partir de los datos (n)
n = len(datos)

# Separar datos en grupo de entrenamiento y de prueba
np.random.seed(30)

# Revolver el dataframe
datos_revueltos = datos.sample(frac=1).reset_index(drop=True)

# Índice para separar los datos
indice = int(n * 0.7)

# Separar en los dos grupos
training = datos_revueltos[:indice].copy()
test = datos_revueltos[indice:].copy()
```

```
In [26]: i = 1
converge1 = 1
converge2 = 1
n = len(training)
# Se entrena el modelo hasta que los valores de theta converjan o hasta que
while (converge1 > 1e-6 or converge2 > 1e-6) and i < 1000000:
    i = i + 1
    theta_nuevo[0] = theta[0] - alpha*(1/n)*np.sum(prediccion(theta[0], theta[1],
    theta_nuevo[1] = theta[1] - alpha*(1/n)*np.sum((prediccion(theta[0], theta[1],
    converge1 = abs(theta_nuevo[0] - theta[0])
    converge2 = abs(theta_nuevo[1] - theta[1])
    theta = theta_nuevo.copy()

print('Iteraciones:', i)
print('theta0 =', round(theta[0], 2))
print('theta1 =', round(theta[1], 2))
```

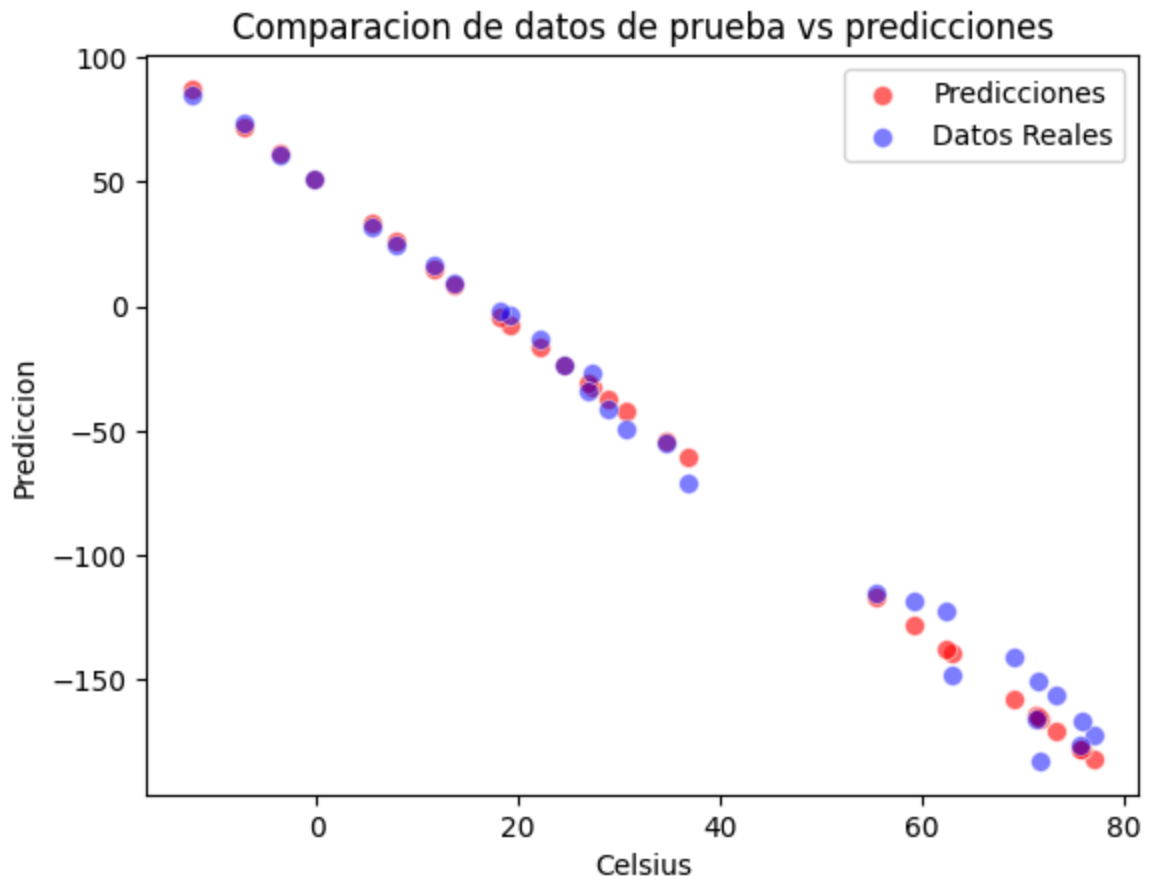
```
Iteraciones: 129011
theta0 = 49.98
theta1 = -3.0
```

```
In [27]: # Se añade una columna para tener las predicciones
test['Prediccion'] = prediccion(theta[0], theta[1], test['Celsius'])
training['Prediccion'] = prediccion(theta[0], theta[1], training['Celsius'])
test.head()
```

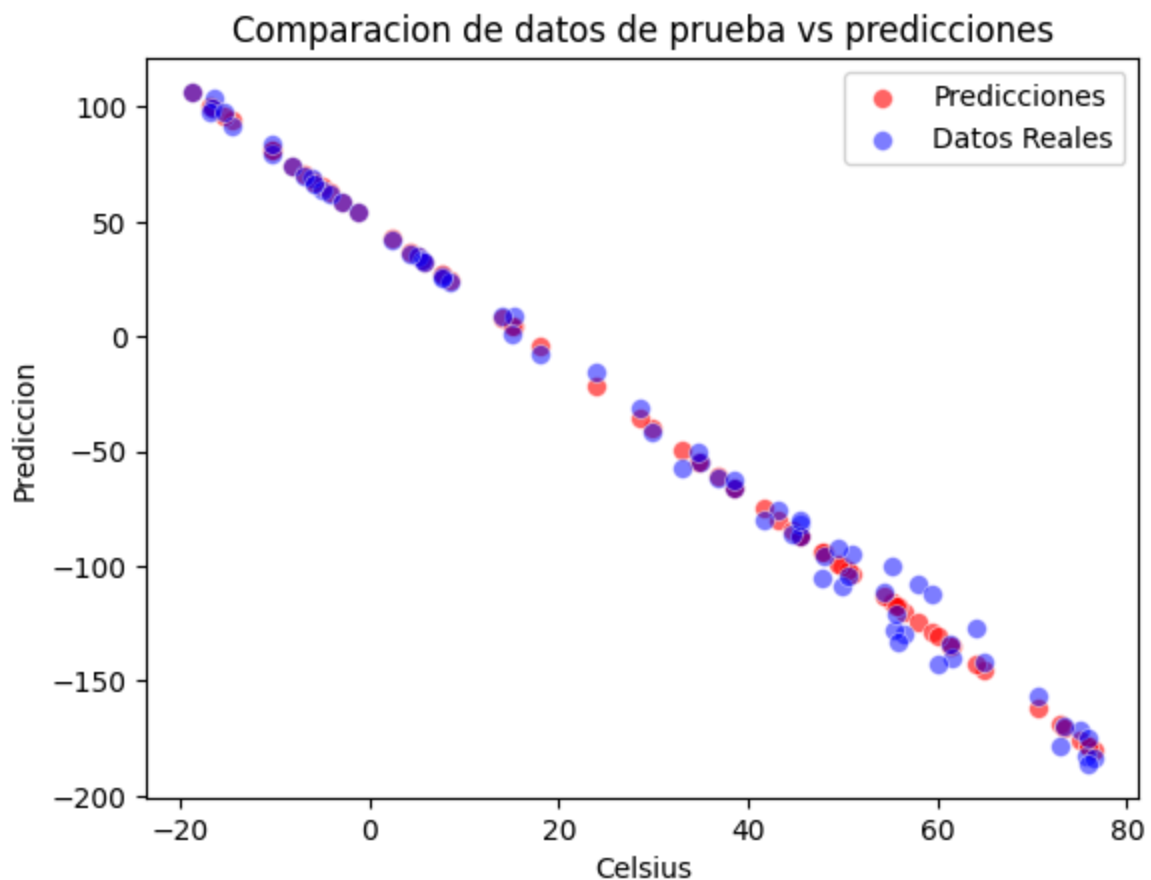
Out [27]:

	Celsius	Valks	Prediccion
70	77.059	-171.990	-181.524929
71	27.329	-27.032	-32.125690
72	69.090	-140.640	-157.584400
73	71.719	-182.570	-165.482461
74	-12.415	84.882	87.273533

```
In [28]: # Comparar los datos de prueba con sus predicciones
sns.scatterplot(data=test, x='Celsius', y='Prediccion', color='red', label='Prediccion')
sns.scatterplot(data=test, x='Celsius', y='Valks', color='blue', label='Datos Reales')
plt.title('Comparacion de datos de prueba vs predicciones')
plt.legend()
plt.show()
```



```
In [29]: # Comparar los datos de entrenamiento con sus predicciones
sns.scatterplot(data=training, x='Celsius', y='Prediccion', color='red', label='Prediccion')
sns.scatterplot(data=training, x='Celsius', y='Valks', color='blue', label='Datos Reales')
plt.title('Comparacion de datos de prueba vs predicciones')
plt.legend()
plt.show()
```



```
In [30]: # Se utiliza el error cuadratico medio para evaluar el rendimiento del modelo
mse_test = (1/n) * np.sum((test['Valks'] - test['Prediccion'])**2)
print('MSE prueba:', round(mse_test,4))

mse_training = (1/n) * np.sum((training['Valks'] - training['Prediccion'])**2)
print('MSE entrenamiento:', round(mse_training,4))
```

MSE prueba: 26.8303

MSE entrenamiento: 37.938