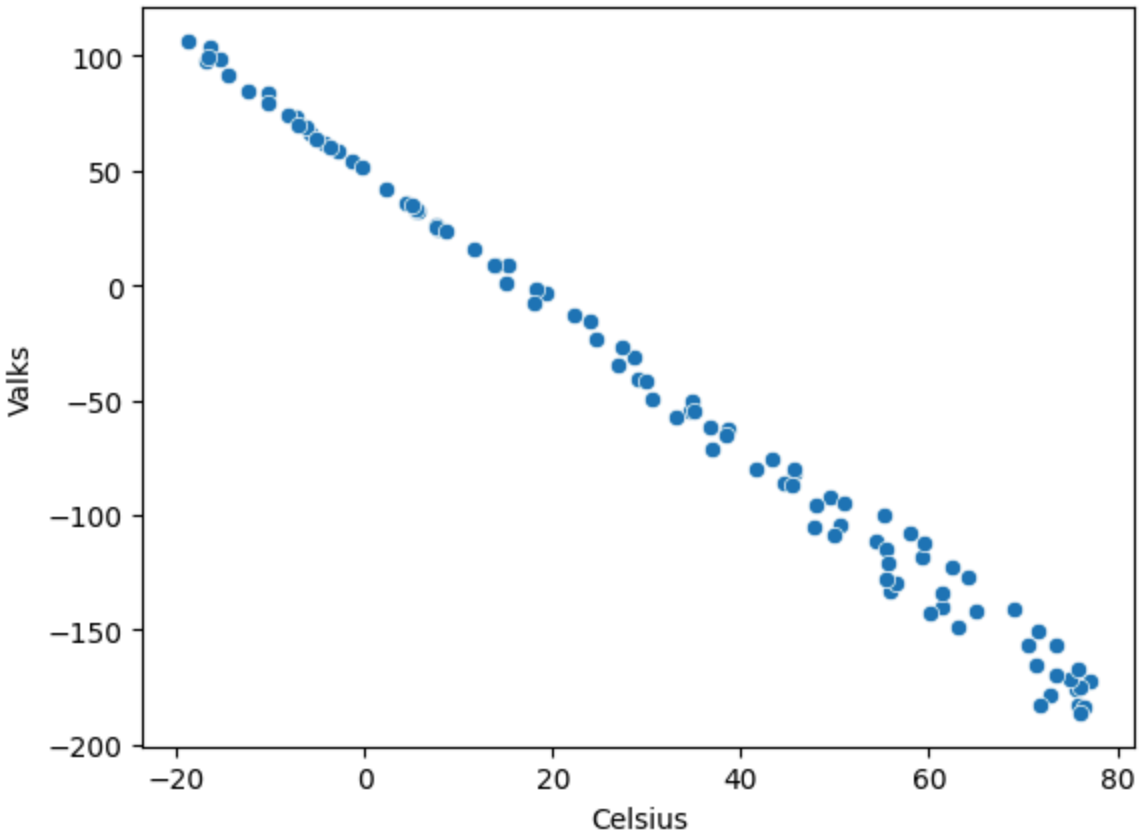
In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
datos = pd.read_csv('Valhalla23.csv')
datos.head()
```

Out[2]:

	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 [3]:
```
sns.scatterplot(data = datos, x= 'Celsius', y = 'Valks')
plt.show;
```



In [4]:
```
# Crear lista con los hiper-parámetros iniciales (thetas)
theta = [60,-10]
theta_nuevo = theta.copy()
# Cargar el valor del learning rate (alpha)
alpha = 1e-4
```

In [5]:
```
# 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(20)

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

# Indice 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 [6]:
```
i = 1
converge0 = 1
converge1 = 1
eps = 1e-6
n = len(training)
# Se entrena el modelo hasta que los valores de theta converjan o hasta que
while (converge0 > eps or converge1 > eps) and i < 1000000:
  i = i + 1
  theta_nuevo[0] = theta[0] - alpha*(1/n)*np.sum(prediccion(theta[0], theta[
  theta_nuevo[1] = theta[1] - alpha*(1/n)*np.sum((prediccion(theta[0], theta
  converge0 = abs( 1 - theta_nuevo[0]/theta[0])
  converge1 = abs( 1 - theta_nuevo[1]/theta[1])
  theta = theta_nuevo.copy()

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

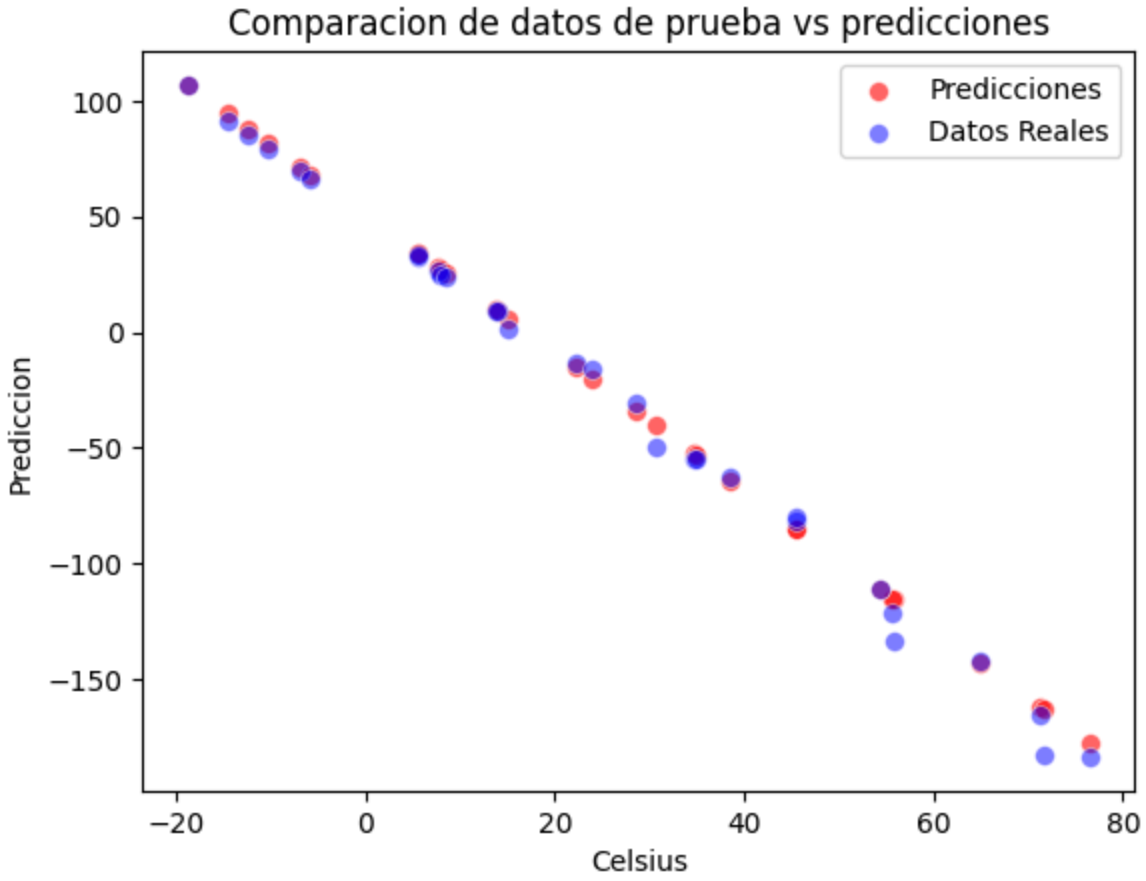
```
Iteraciones: 52308
theta0 = 50.83
theta1 = -2.98
```

In [7]:
```
# 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()
```

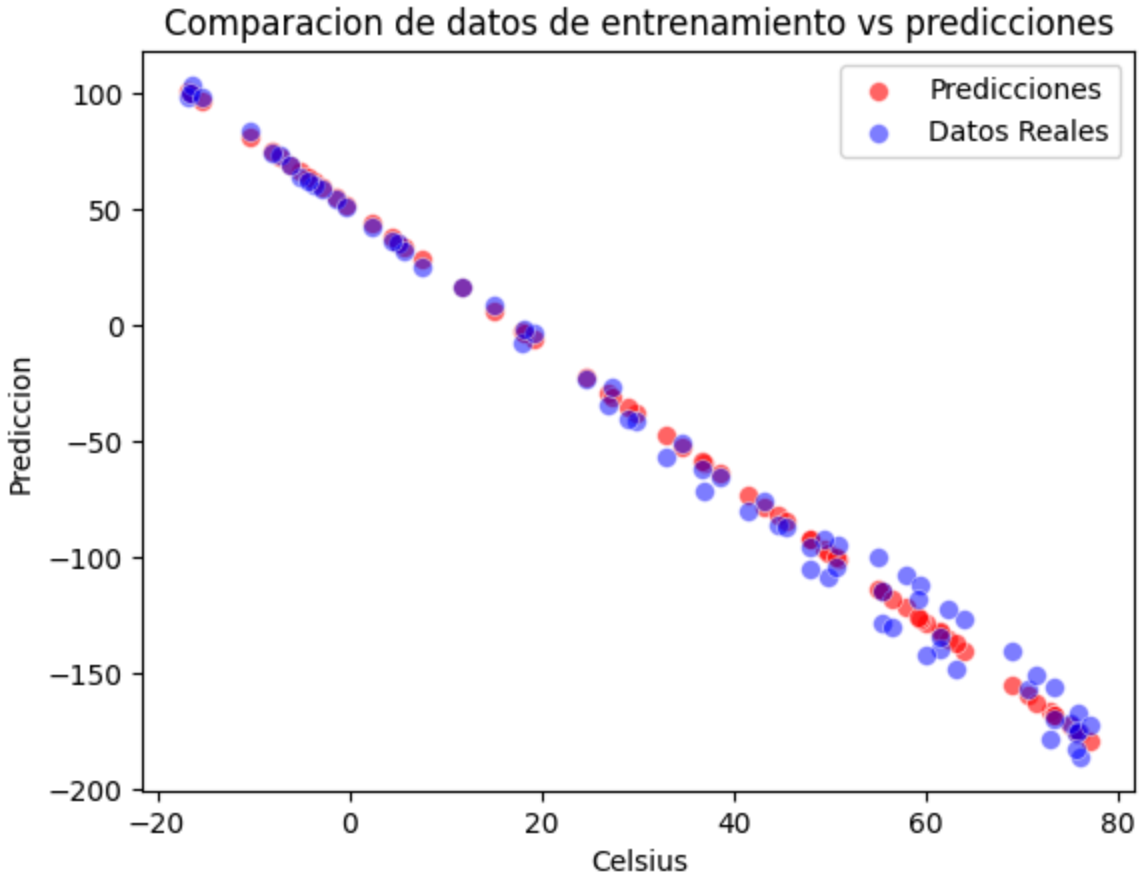
Out[7]:

	Celsius	Valks	Prediccion
70	7.6923	26.4480	27.904568
71	14.0390	8.7644	8.986369
72	5.5095	32.1980	34.411043
73	71.3380	-165.4200	-161.810091
74	55.7740	-133.3800	-115.417029

In [8]:
```
# Comparar los datos de prueba con sus predicciones
sns.scatterplot(data=test, x='Celsius', y='Prediccion', color='red', label='
sns.scatterplot(data=test, x='Celsius', y='Valks', color='blue', label='Datc
plt.title('Comparacion de datos de prueba vs predicciones')
plt.legend()
plt.show()
```



In [9]:
```
# Comparar los datos de entrenamiento con sus predicciones
sns.scatterplot(data=training, x='Celsius', y='Prediccion', color='red', lab
sns.scatterplot(data=training, x='Celsius', y='Valks', color='blue', label='
plt.title('Comparacion de datos de entrenamiento vs predicciones')
plt.legend()
plt.show()
```



In [10]:
```
# Se utiliza el error cuadratico medio para evaluar el rendimiento del model
mse_test = (1/len(test)) * np.sum((test['Valks'] - test['Prediccion'])**2)
print('MSE prueba:', round(mse_test,4))

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

```
MSE prueba: 34.409
MSE entrenamiento: 50.3266
```