Template para uso de framework (scikit-learn)

En términos generales, debemos seguir los siguientes pasos:

1. Importar módulos
2. Cargar datos
3. Separar datos en subconjuntos
4. Entrenar el modelo
5. Analizar su desempeño
6. Usar el modelo para nuevas estimaciones (datos no vistos)

In [3]:
```
# Importar módulos
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.linear_model
```

In [4]:
```
# Cargar datos
datos = pd.read_csv('Valhalla23 copy.csv')

X = datos[['Celsius']]
Y = datos['Valks']

# Separar datos en subconjuntos (usando train_test_split)
x_train, x_test, y_train, y_test = sklearn.model_selection.train_test_split(
```

In [5]:
```
# Entrenar el modelo
# --- Crear objeto del modelo

model = sklearn.linear_model.SGDRegressor(penalty= None, max_iter= 10000000,

# --- Usar método fit para ajustar el modelo a los datos de entrenamiento
model.fit(x_train, y_train)
```

Out[5]:
```
SGDRegressor
SGDRegressor(eta0=1e-05, max_iter=10000000, penalty=None, tol=1e-0
6)
```

Parámetros seleccionados:

- Penalty: None, para tener el gradiente decendente sin regularización
- eta0 = 1e-5, se utiliza este learning rate ya que con uno más grande no se encuentra convergencia
- tol = 1e-6, se utiliza esta tolerancia para encontrar una buena solución

- max_iter = 1000000, para asegurar encontrar convergencia, con menos iteraciones es probable que no se encuentre

In [6]:
```
# Calcular el rendimiento del modelo
y_train_pred = model.predict(x_train)
print('R2 training set:', r2_score(y_train, y_train_pred))
print('MSE training set', mean_squared_error(y_train,y_train_pred))
```

```
R2 training set: 0.9915695954273287
MSE training set 62.86857896999853
```

In [7]:
```
# Usar el modelo para nuevas estimaciones (datos no vistos), usando método p

y_test_pred = model.predict(x_test)
print('R2 test set:', r2_score(y_test, y_test_pred))
print('MSE test set', mean_squared_error(y_test,y_test_pred))
```

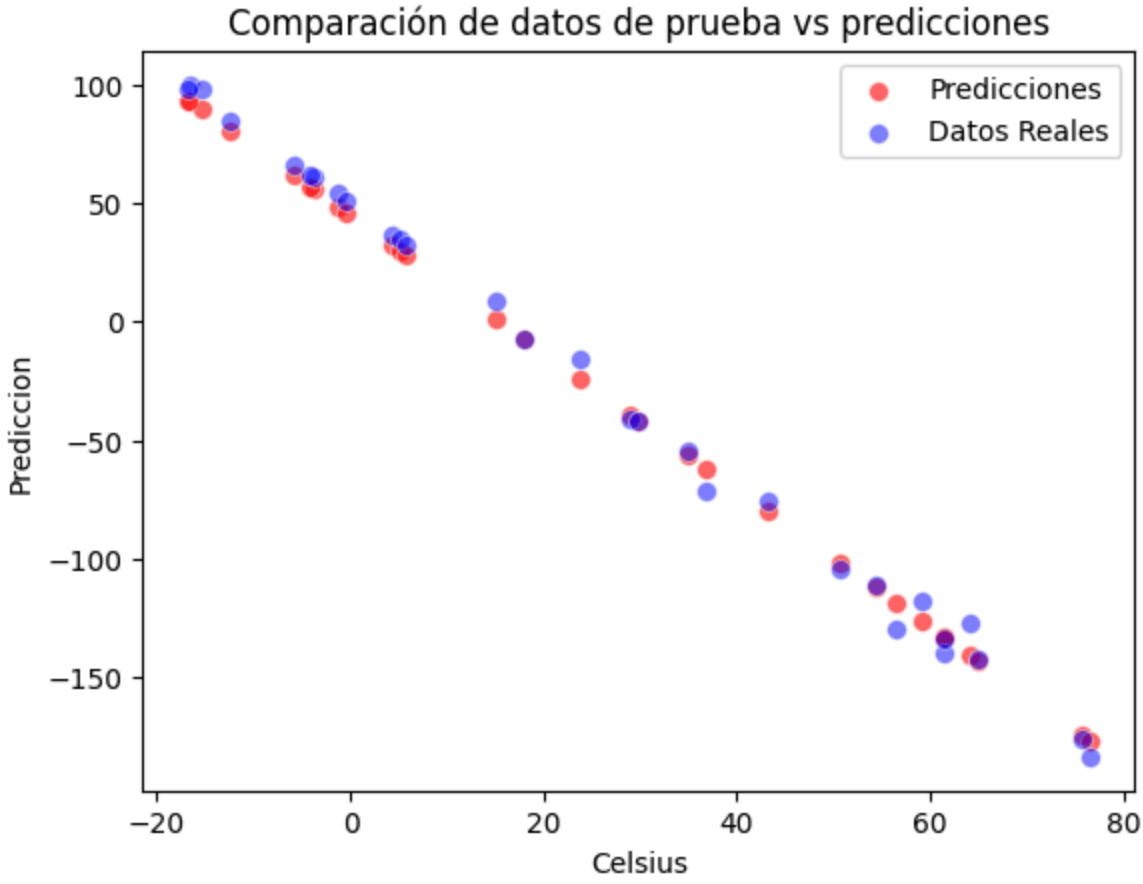
```
R2 test set: 0.9954147541889706
MSE test set 36.88598358773648
```

In [8]:
```
x_test_flat = x_test.values.ravel()
y_test_flat = y_test.values
y_test_pred_flat = y_test_pred

datos_grafica = {
    'Celsius': x_test_flat,
    'Valks': y_test_flat,
    'Prediccion': y_test_pred_flat
}

test = pd.DataFrame(datos_grafica)

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('Comparación de datos de prueba vs predicciones')
plt.legend()
plt.show()
```

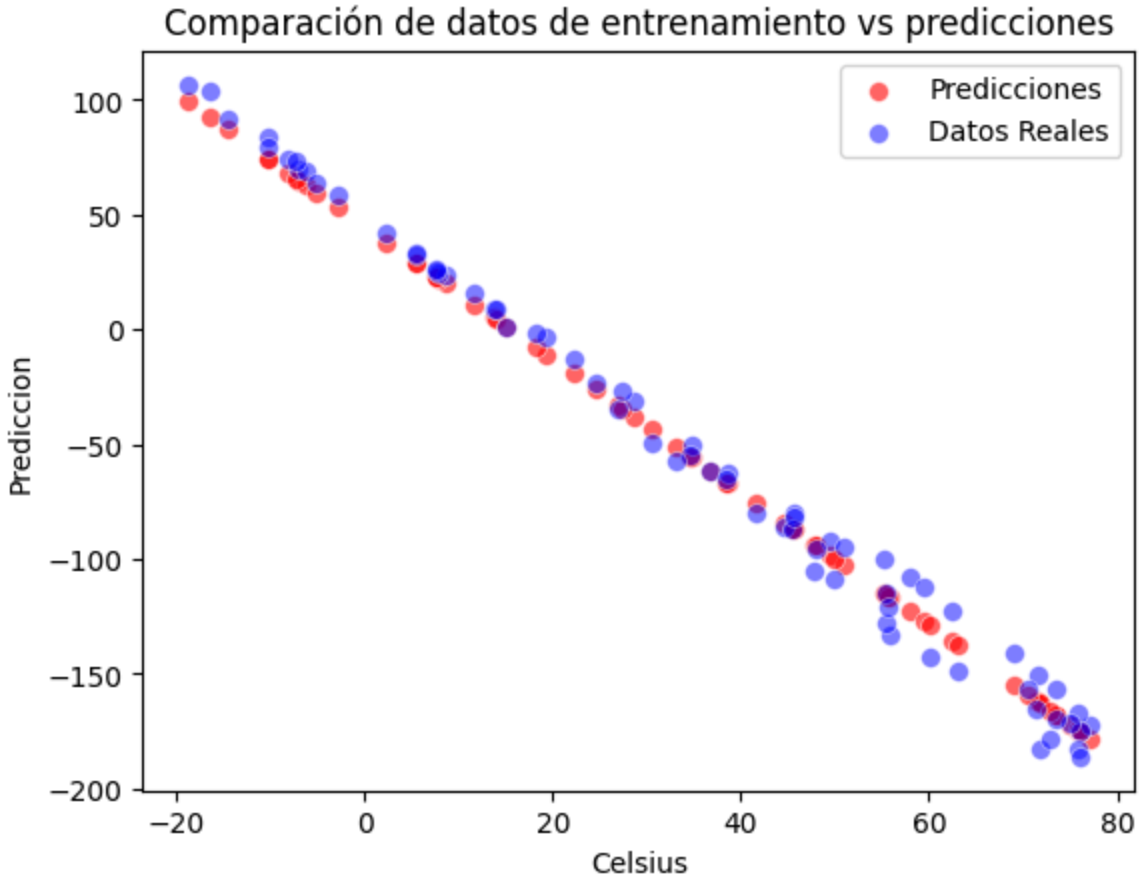


In [9]:
```
x_train_flat = x_train.values.ravel()
y_train_flat = y_train.values
y_train_pred_flat = y_train_pred

datos_grafica_2 = {
    'Celsius': x_train_flat,
    'Valks': y_train_flat,
    'Prediccion': y_train_pred_flat
}

train = pd.DataFrame(datos_grafica_2)

sns.scatterplot(data=train, x='Celsius', y='Prediccion', color='red', label=
sns.scatterplot(data=train, x='Celsius', y='Valks', color='blue', label='Dat
plt.title('Comparación de datos de entrenamiento vs predicciones')
plt.legend()
plt.show()
```


Comparación de datos de entrenamiento vs predicciones
Predicciones
Datos Reales
Prediccion
Celsius
100
50
0
−50
−100
−150
−200
−20
0
20
40
60
80