Tecnológico de Monterrey

REGRESIÓN LINEAL
RETO

† It has happened. Aliens have arrived. They hail from a planet called Valhalla-23, where the temperature is measured in Valks. These visitors tell you that they have come to solve Earth's global warming crisis*. They offer you a machine that will solve the problem, but they warn you:

1. The machine must be set up in Valks.
2. If you input a wrong temperature value, you may end up freezing or scorching the Earth.
3. No one knows how to transform between Celsius and Valks.

† You are tasked with finding a model for solving this problem, so you ask Humans and Valkians to collect temperature readings from several objects. The data are given in the *Valhalla23.csv* file.

Importar librerías y módulos

In [2]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
import pandas as pd
import numpy as np
```

In [3]:
```
# Leer el archivo CSV
df = pd.read_csv('valhalla23.csv')

df.head()
```

Out[3]:

	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

Separar datos en subconjuntos (usando train_test_split)

In [6]:
```
# Suponiendo que tu DataFrame se llama df
y = df['Valks']  # Asegúrate de que X sea un DataFrame de una sola columna
X = df[['Celsius']]  # y es una Serie

# Dividir los datos en subconjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
```

Entrenar el modelo

Crear objeto del modelo

In [7]:
```
# Crear el objeto del modelo
model = LinearRegression()
```

Usar método fit para ajustar el modelo a los datos de entrenamiento

In [8]:
```
# Ajustar el modelo a los datos de entrenamiento
model.fit(X_train, y_train)
```

Out[8]:
```
LinearRegression()
```

Analizar el desempeño

Opción 1: Usando método score del modelo

In [9]:
```
# Usar el método score para evaluar el modelo
train_score = model.score(X_train, y_train)
test_score = model.score(X_test, y_test)

print(f"Puntaje en el conjunto de entrenamiento: {train_score:.4f}")
print(f"Puntaje en el conjunto de prueba: {test_score:.4f}")
```

```
Puntaje en el conjunto de entrenamiento: 0.9930
Puntaje en el conjunto de prueba: 0.9976
```

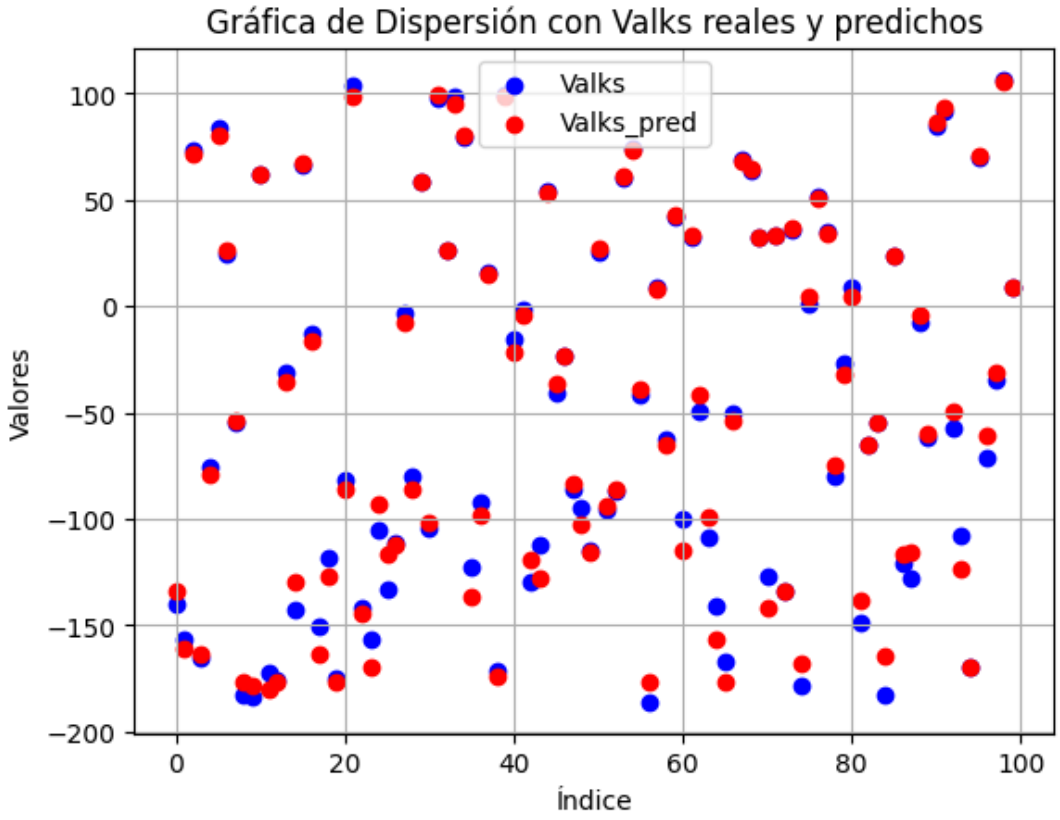
Opción 2: Haciendo predicciones y utilizando módulo de métricas

In [10]:
```
from sklearn import metrics

# Hacer predicciones en los datos de prueba
y_pred = model.predict(X_train)

# Calcular métricas de evaluación
mse = metrics.mean_squared_error(y_train, y_pred)
r2 = metrics.r2_score(y_train, y_pred)

print('Conjunto de Entrenamiento:')
print(f"Error Cuadrático Medio (MSE): {mse:.4f}")
print(f"Coeficiente de Determinación (R^2): {r2:.4f}")


# Hacer predicciones en los datos de prueba
y_pred = model.predict(X_test)

# Calcular métricas de evaluación
mse = metrics.mean_squared_error(y_test, y_pred)
r2 = metrics.r2_score(y_test, y_pred)

print('')
print('Conjunto de Prueba:')
print(f"Error Cuadrático Medio (MSE): {mse:.4f}")
print(f"Coeficiente de Determinación (R^2): {r2:.4f}")
```

```
Conjunto de Entrenamiento:
Error Cuadrático Medio (MSE): 50.4882
Coeficiente de Determinación (R^2): 0.9930

Conjunto de Prueba:
Error Cuadrático Medio (MSE): 20.1881
Coeficiente de Determinación (R^2): 0.9976
```

Visualización de valores predichos

In [14]:
```
import matplotlib.pyplot as plt

df['Valks_pred'] = model.predict(df[['Celsius']])

# Crear la gráfica de dispersión
plt.scatter(df.index, df['Valks'], label='Valks', color='blue')
plt.scatter(df.index, df['Valks_pred'], label='Valks_pred', color='red')
plt.xlabel('Índice')
plt.ylabel('Valores')
plt.title('Gráfica de Dispersión con Valks reales y predichos')
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]:
```
```