Actividad 3 - Regresión Lineal Gradiente Descendente con SL

Héctor Hibran Tapia Fernández - A01661114

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.

*Valkians are not to be held responsible from any annihilation that derive from poor usage of their devices.

APRENDIZAJE DE MÁQUINA IVÁN MAURICIO AMAYA CONTRERAS 2

Tecnológico de Monterrey

SCIKIT-LEARN
ACTIVIDAD

† Teniendo en cuenta lo mostrado durante esta presentación, desarrolle un código de Python que implemente una regresión lineal para el dataset de Valhalla, usando Scikit-learn. Consulte el template disponible en Canvas para tener una guía paso a paso de lo que debe implementar.

† Grafique los datos suministrados y las predicciones obtenidas por su modelo. En su criterio, ¿Qué tan bueno es el modelo encontrado por Scikit-learn?

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)

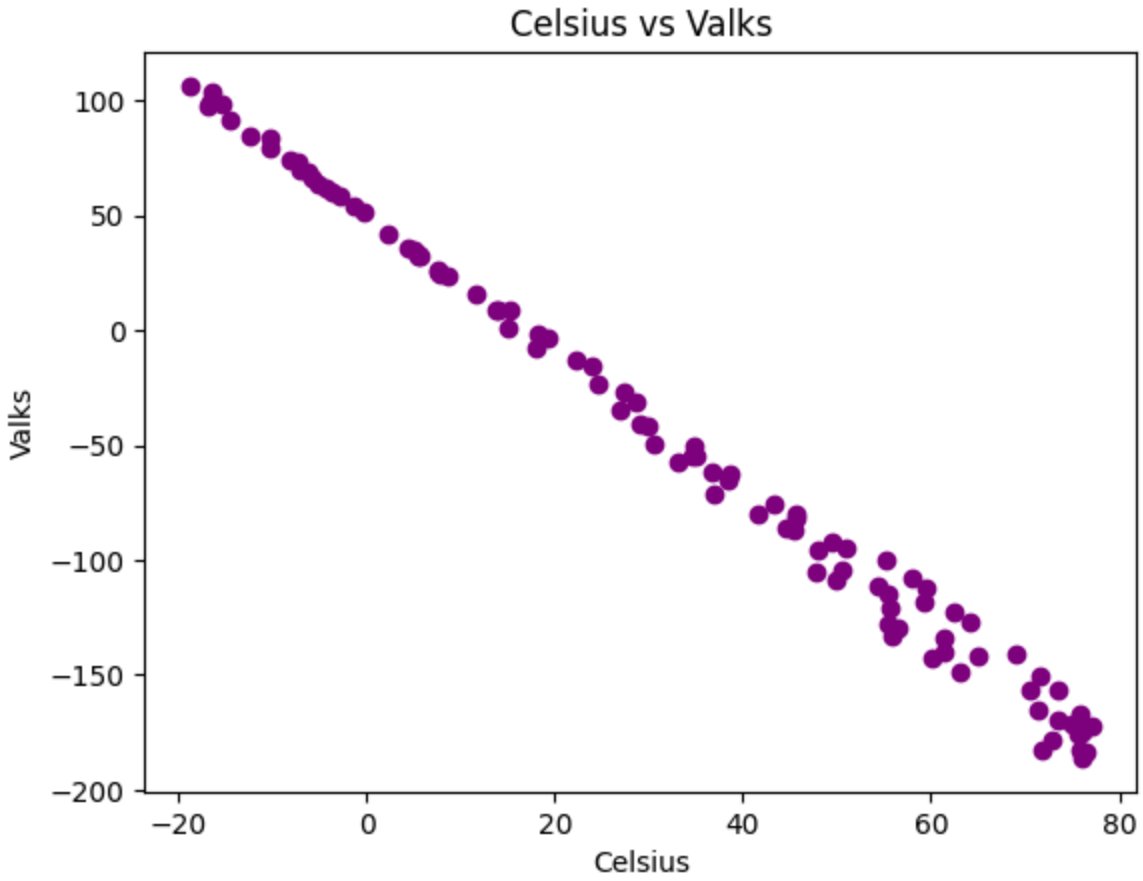
In [156…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

In [157…
```
df = pd.read_csv('./Valhalla23.csv')
df.head(10)
```

Out[157]:

	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
5	-10.2460	83.437
6	7.8498	24.680
7	34.6880	-55.108
8	75.7510	-182.820
9	76.4890	-183.460

In [158…
```
plt.scatter(df['Celsius'], df['Valks'], c = 'purple')
plt.title('Celsius vs Valks')
plt.xlabel('Celsius')
plt.ylabel('Valks')
plt.show()
```



In [159…
```
df.shape
```

Out[159]:
```
(100, 2)
```

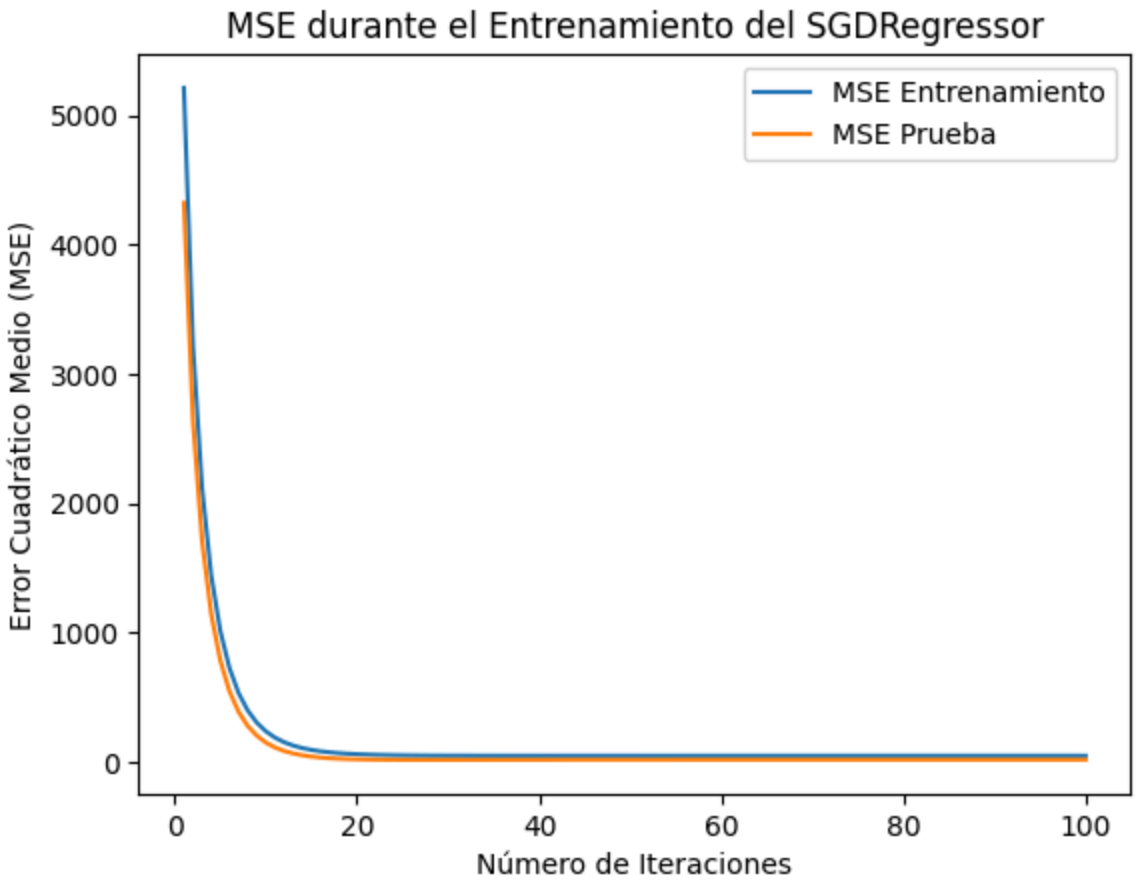
In [160…
```
X = df[['Celsius']]  # Característica
y = df['Valks']      # Variable objetivo

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, ran

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [161…
```
model = SGDRegressor(loss = 'squared_error', penalty = 'l2', alpha = 0.0001, m
```

- **loss = 'squared_error':** comúnmente utilizada para problemas de regresión
- **penalty = 'l2':** ayuda a prevenir el sobreajuste (overfitting) al reducir la complejidad del modelo
- **alpha = 0.0001:** determina la importancia de la penalización de los coeficientes en el cálculo de la pérdida total, valores bajos permiten al modelo ajustarse más estrechamente a los datos de entrenamiento
- **max_iter = 100:** requerimiento minimo para la entrega
- **tol = None:** N tiene condición de tolerancia para detener el entrenamiento, lo que el modelo se entrenará hasta completar el número máximo de iteraciones
- **random_state = 42:** Valor arbitrario elegido comúnmente para la semilla

In [162…
```
mse_train = []
mse_test = []

n_iter = 100

for _ in range(n_iter):
    model.partial_fit(X_train_scaled, y_train)

    y_train_pred = model.predict(X_train_scaled)
    y_test_pred = model.predict(X_test_scaled)

    mse_train.append(mean_squared_error(y_train, y_train_pred))
    mse_test.append(mean_squared_error(y_test, y_test_pred))
```

In [163…
```
plt.plot(range(1, n_iter + 1), mse_train, label = 'MSE Entrenamiento')
plt.plot(range(1, n_iter + 1), mse_test, label = 'MSE Prueba')
plt.xlabel('Número de Iteraciones')
plt.ylabel('Error Cuadrático Medio (MSE)')
plt.title('MSE durante el Entrenamiento del SGDRegressor')
plt.legend()
plt.show()
```

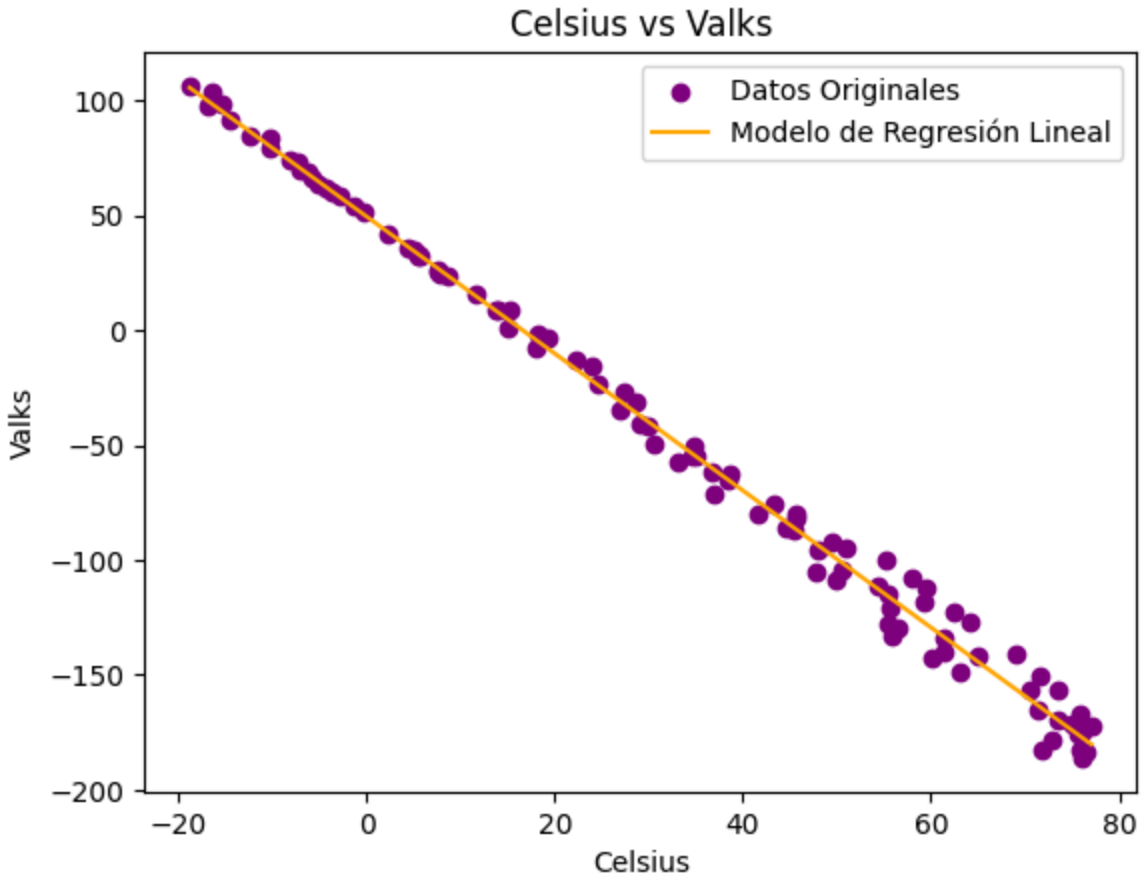


In [164…
```
y_pred = model.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

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

```
Error Cuadrático Medio (MSE): 20.22
Coeficiente de Determinación (R^2): 1.00
```

In [165…
```
temperaturas_celsius = np.linspace(df['Celsius'].min(), df['Celsius'].max(), 1(
temperaturas_celsius_scaled = scaler.transform(temperaturas_celsius)
predicciones_valks = model.predict(temperaturas_celsius_scaled)

plt.scatter(df['Celsius'], df['Valks'], color = 'purple', label = 'Datos Origi
plt.plot(temperaturas_celsius, predicciones_valks, color = 'orange', label = 'M
plt.title('Celsius vs Valks')
plt.xlabel('Celsius')
plt.ylabel('Valks')
plt.legend()
plt.show()
```

```
/Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/
sklearn/base.py:493: UserWarning: X does not have valid feature names, but Sta
ndardScaler was fitted with feature names
  warnings.warn(
```



In [166…
```
resultados = pd.DataFrame({'Valor Real (Valks)': y_test, 'Predicción (Valks)':
resultados = resultados.reset_index(drop = True)

resultados.head(10)
```

Out[166]:

	Valor Real (Valks)	Predicción (Valks)
0	-54.496	-54.726047
1	60.614	60.697071
2	-127.100	-141.492860
3	-40.934	-36.847909
4	54.276	53.463224
5	99.744	98.910846
6	-142.020	-144.000451
7	8.774	4.329056
8	61.973	62.187612
9	-139.740	-133.740499

In [167…
```
def predecir_valks(temperatura_celsius):
    temperatura_escalada = scaler.transform([[temperatura_celsius]])
    prediccion = model.predict(temperatura_escalada)
    return prediccion[0]

temperatura_usuario = float(input("Introduce la temperatura en Celsius: "))
```

In [168…
```
valks_predicho = predecir_valks(temperatura_usuario)
print(f"{temperatura_usuario} Celsius son {valks_predicho:.2f} Valks")
```

```
80.0 Celsius son -188.99 Valks
```

```
/Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/
sklearn/base.py:493: UserWarning: X does not have valid feature names, but Sta
ndardScaler was fitted with feature names
  warnings.warn(
```