Implementación_de_una_técnica_de_aprendizaje_máquir

José Carlos Sánchez Gómez A0174050

26 de agosto del 2024

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.preprocessing import StandardScaler

# from google.colab import drive
# drive.mount('/content/drive')
```

Separación de los datos en train y test, junto con la declaracion de las tetas y alpha (correción)

In [ ]:
```
data = pd.read_csv('../../..//Valhalla23.csv')
scaler = StandardScaler()
data[['Celsius']] = scaler.fit_transform(data[['Celsius']])

'''
Noté que usando los valores normales de Celsius, SDGRegressor me regresaba la tendenci
eran muy grandes, asi que decidí escalarlos, y se obtuv[].
'''
x_train, x_test, y_train, y_test = train_test_split(data[['Celsius']], data[['Valks']]

model = linear_model.SGDRegressor(max_iter = 100, tol = False, alpha= 0.0001)
#model = linear_model.LinearRegression()
model.fit(x_train, y_train)
```

```
C:\Users\jcsg6\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra
8p0\LocalCache\local-packages\Python312\site-packages\sklearn\utils\validation.py:133
9: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[ ]:
```
SGDRegressor
SGDRegressor(max_iter=100, tol=False)
```

Se seleccionaron estos hiperparametros de tal forma que el modelo se entrenara con 100 iteraciones, y no termine de entrenarse antes, el valor de alfa se selecciono con base a prueba y error buscando el que diera mejores resultados.

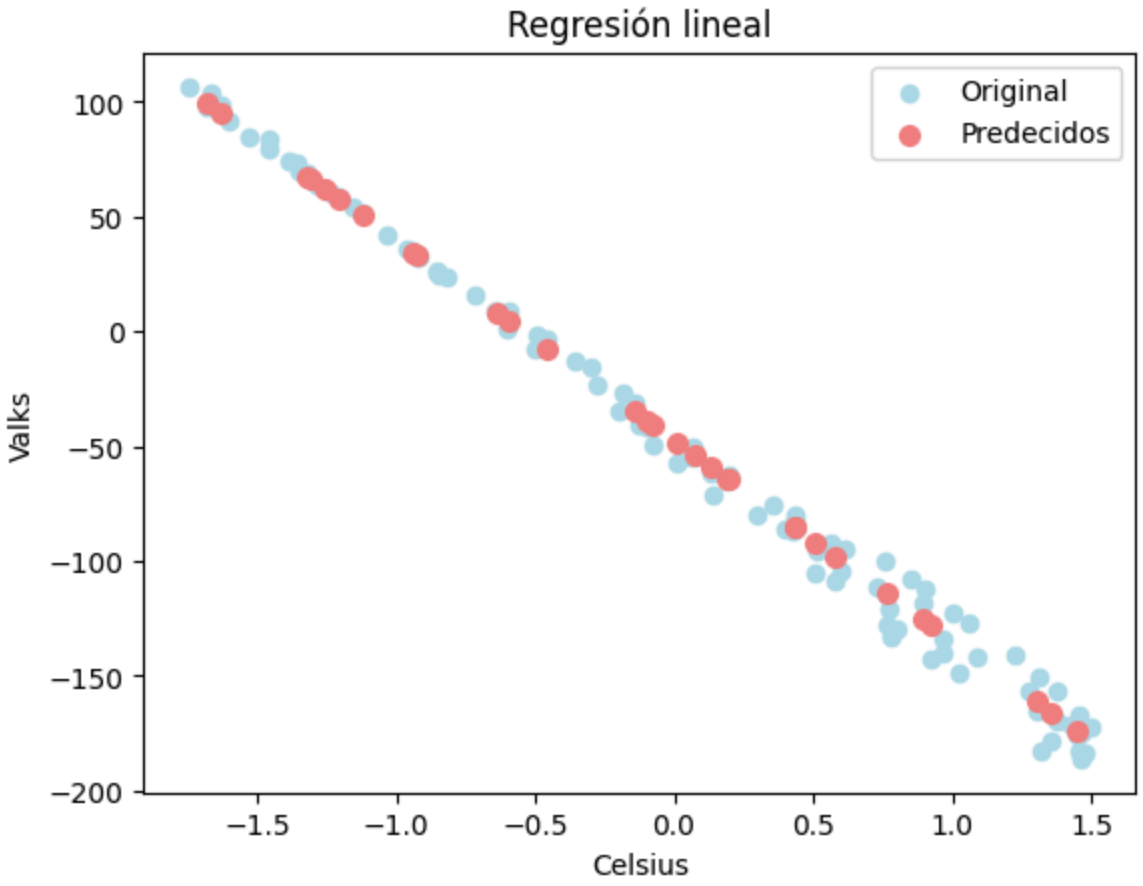
Entrenamiento del modelo con 100 iteraciones

In [ ]:
```
puntaje_entrenamiento = model.score(x_train, y_train)
puntaje_prueba = model.score(x_test, y_test)

print('Puntaje de entrenamiento:', puntaje_entrenamiento)
print('Puntaje de prueba:', puntaje_prueba)

predicciones = model.predict(x_test)
error_cuadratico_medio = metrics.mean_squared_error(y_test, predicciones)
print('Error cuadrático medio:', error_cuadratico_medio)
```

```
Puntaje de entrenamiento: 0.9942228761785792
Puntaje de prueba: 0.9939773251367556
Error cuadrático medio: 39.92649945923386
```

Comparando los resultados de los modelos usando los datos de test, contra los valores originales

In [ ]:
```
predicciones = model.predict(x_test)
plt.scatter(data['Celsius'], data['Valks'], color = "lightblue")
plt.scatter(x_test, predicciones, color='lightcoral', linewidth=2)
plt.xlabel('Celsius')
plt.ylabel('Valks')
plt.title('Regresión lineal')
plt.legend(["Original", "Predecidos"])
plt.show()
```



In [ ]:
```
!jupyter nbconvert --to html /content/drive/MyDrive/ColabNotebooks/Retroalimentacion_M
```