

Crea un espacio dentro del repositorio de GitHub que creaste para el portafolio de implementación (por ejemplo, una carpeta para el módulo de ML con una subcarpeta para este entregable).

Selecciona uno de los dos primeros 'Challenge' vistos en clase (Week01_Challenge.pdf o Week02_Challenge1.pdf) y programa un algoritmo que permita resolver el problema. Dicho algoritmo debe ser uno de los algoritmos vistos en el módulo (o que tu profesor de módulo autorice), y no puedes usar ninguna biblioteca o framework de aprendizaje máquina, ni de estadística avanzada. Lo que se busca es que implementes manualmente el algoritmo, no que importes un algoritmo ya implementado.

```
In [62]: #Importaremos las librerías para poder leer el documento
import pandas as pd
import numpy as np

#Haremos uso del primer Challenge, el cual contiene una base de datos llamada
#paso, importaremos este dataset y le asignaremos una variable

v = pd.read_csv('Valhalla23.csv')

#Ahora asignaremos una variable para cada columna, de esta manera sera mas facil
X = v['Celsius']
y = v['Valks']
```

Divide el set de datos del problema en dos subconjuntos, uno para entrenamiento y otro para prueba. Entrena tu modelo sobre el primer subconjunto, y por un mínimo de 100 iteraciones. Selecciona valores para la tasa de aprendizaje y para los parámetros iniciales, según tu criterio.

```
In [63]: #Dividir los datos en conjuntos, uno de test y uno de train para facilitar s
mitad = len(X) // 2
x_train = X[:mitad]      # Primera mitad para entrenamiento
x_test = X[mitad:]       # Segunda mitad para prueba
y_train = y[:mitad]
y_test = y[mitad:]
```

```

#Asignamos valores para ambos tethas y alpha
tetha0 = 1
tetha1 = 1
alpha = 0.0001

#Asignamos una variable que indique el numero de iteraciones
num_iteraciones = 150000

#Declaramos la funcion de hipotesis
htetha = lambda tetha0, tetha1, x: tetha0 + (tetha1 * x)

#Calcular el total de muestras en el conjunto de prueba (n)
n = len(x_test)

#Creamos dos variables para poder ir almacenando las iteraciones
theta0_values = []
theta1_values = []
#Iterar el proceso de ajuste de theta0 y theta1
for i in range(num_iteraciones):
    #Calcular la hipótesis
    h0 = htetha(tetha0, tetha1, x_test)

    #Calcular los errores para theta0
    delta0 = np.subtract(h0, y_test)
    delta0t = np.sum(delta0)

    #Actualizar theta0
    tetha0 = tetha0 - alpha * (delta0t/n)

    #Calcular los errores para theta1
    deltax = np.multiply(delta0, x_test)
    deltaxt = np.sum(deltax)

    #Actualizar theta1
    tetha1 = tetha1 - alpha * (deltaxt/n)
    theta0_values.append(tetha0)
    theta1_values.append(tetha1)

print(tetha0)
print(tetha1)

```

```

48.92545118184285
-2.989103471334676

```

Prueba tu implementación. Para ello, utiliza el modelo entrenado para hacer predecir las salidas del subconjunto de prueba, y compara contra los datos reales en una gráfica.

```
In [64]: #Importamos la libreria de matplotlib para poder hacer la grafica
```

```

import matplotlib.pyplot as plt

#Utilizamos el modelo entrenado para hacer predicciones
y_pred = htetha(tetha0, tetha1, x_test)

#Imprimir los valores finales de theta0 y theta1
print("Valor final de theta0:")
print(tetha0)
print("Valor final de theta1: ")
print(tetha1)

plt.figure(figsize=(10, 6))
plt.scatter(x_test, y_test, color='blue', label='Valores Reales', marker='o')
plt.scatter(x_test, y_pred, color='red', label='Predicciones', marker='o', a
plt.xlabel('Celsius')
plt.ylabel('Valks')
plt.title('Comparación')
plt.legend()
plt.grid(True)

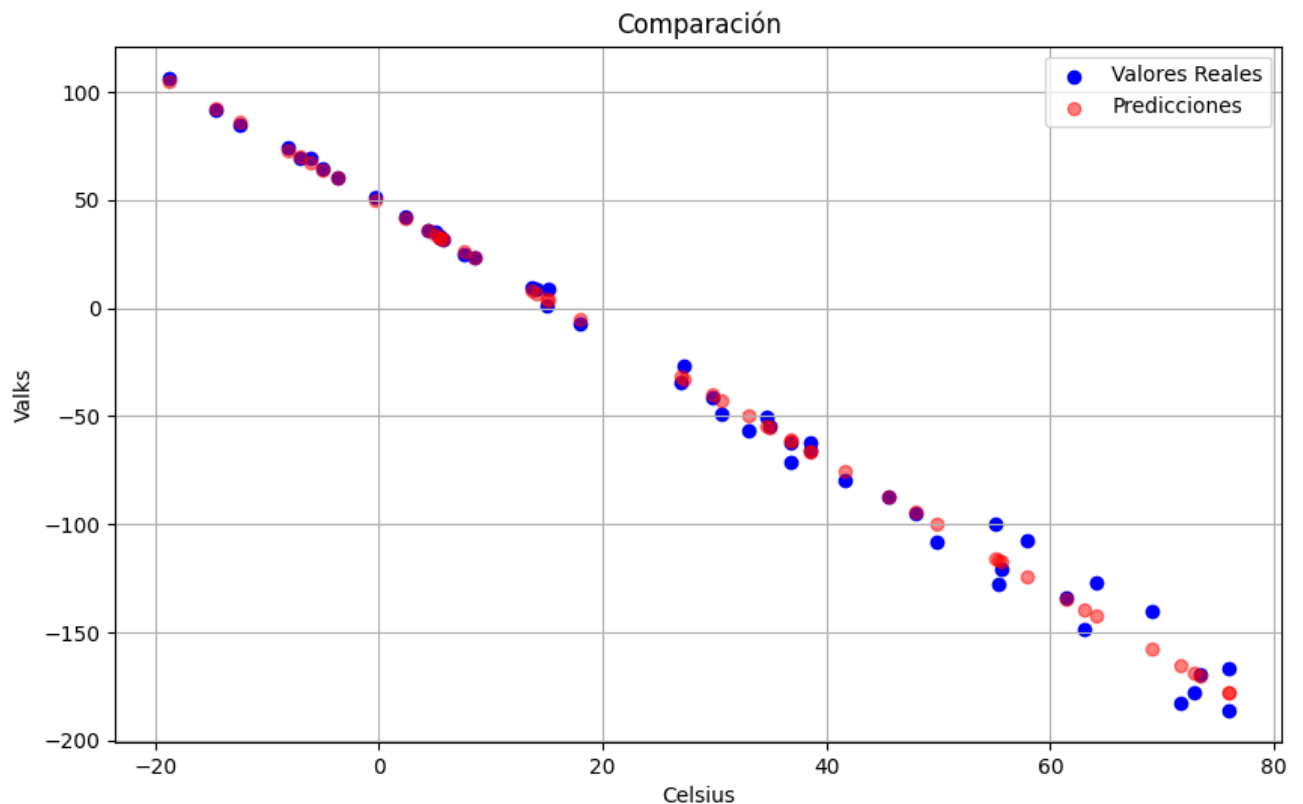
```

Valor final de theta0:

48.92545118184285

Valor final de theta1:

-2.989103471334676



Calcula el valor de la función de costo para el subconjunto de entrenamiento, y para el subconjunto de prueba.

```
In [65]: #Valores de test y train
y_pred_train = htetha(tetha0, tetha1, x_train)
y_pred_test = htetha(tetha0, tetha1, x_test)

#Calcular la funcion de costo para train
mse_train = np.mean((y_train - y_pred_train) ** 2)

#Calcular la función de costo para test
mse_test = np.mean((y_test - y_pred_test) ** 2)

print("Costo de entrenamiento:")
print(mse_train)

print("\nCosto de prueba:")
print(mse_test)
```

Costo de entrenamiento:
46.175964305611075

Costo de prueba:
45.23509334528928

Para facilitar la revisión, entrega dos archivos. El primero debe ser un Jupyter Notebook con todo el desarrollo (código comentado). El segundo debe ser un PDF del Jupyter Notebook. Para esto último, utiliza el comando nbconvert --to html para exportar el notebook a HTML y poder guardar el PDF más fácilmente (<https://github.com/jupyter/nbconvert>). Ten en cuenta que debes cargar tu directorio de Drive y dar la ruta al archivo, por lo que el comando completo sería:

```
!jupyter nbconvert --to html /content/drive/MyDrive/ColabNotebooks/archivo.ipynb
```

```
In [87]: !jupyter nbconvert --to html '/content/drive/MyDrive/ColabNotebooks/Varhalla
[NbConvertApp] Converting notebook /content/drive/MyDrive/ColabNotebooks/Varhalla.ipynb to html
[NbConvertApp] WARNING | Alternative text is missing on 1 image(s).
[NbConvertApp] Writing 349883 bytes to /content/drive/MyDrive/ColabNotebooks/Varhalla.html
```