Momento de Retroalimentación: Módulo 2 Implementación de una Técnica de Aprendizaje Máquina sin el Uso de un Framework

Héctor Hibran Tapia Fernández - A01661114

In [41]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Tecnológico de Monterrey

CLASIFICACIÓN
RETO

† Adapte el código de la regresión lineal desarrollado en clase para que el modelo entrenado corresponda con una regresión logística. Posteriormente, implemente un clasificador que estime si un estudiante aprueba o no el curso:

‡ Considerando solamente la columna 'Attendance'

‡ Considerando solamente la columna 'Homework'

† Calcule las métricas de desempeño. ¿Cuál es mejor? ¿Le ganan a la referencia?

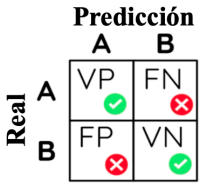
Attendance	Homework	Pass	Reference
80	75	yes	yes
65	70	no	no
95	85	yes	yes
95	100	yes	no
85	65	no	no
75	55	no	no
90	90	yes	yes
65	80	yes	no

$$accuracy = \frac{VP + VN}{VP + VN + FP + FN}$$

$$precision = \frac{VP}{VP + FP}$$

$$recall = \frac{VP}{VP + FN}$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$



In [42]:
```
attendance = np.array([80, 65, 95, 95, 85, 75, 90, 65])
homework = np.array([75, 70, 85, 100, 65, 55, 90, 80])
y = np.array([1, 0, 1, 1, 0, 0, 1, 1])  # 1 para "yes", 0 para "no"
```

In [43]:
```
theta_0 = 0 # de otra forma dan errores de convergencia, practica comun
theta_1 = 0 # de otra forma dan errores de convergencia, practica comun

alpha = 0.1 # Definimos el learning rate (alpha), garantizamos que el descenso
```

In [44]:
```
def sigmoide(z): # Función sigmoide
    return 1 / (1 + np.exp(-z)) # 1 / (1 + e^-z)

def hipotesis(x, theta_0, theta_1): # Función hipótesis
    return sigmoide(theta_0 + theta_1 * x) # h(x) = g(theta_0 + theta_1 * x)

def costo_logistica(x, y, theta_0, theta_1): # Función de costo
    n = len(y) # Número de muestras
    h = hipotesis(x, theta_0, theta_1) # Aplicamos la hipótesis
    return (-1/n) * sum(y * np.log(h) + (1 - y) * np.log(1 - h)) # -1/n * sum(
```

```
def delta_theta_0(x, y, theta_0, theta_1): # Derivada parcial de la función de
    n = len(y) # Número de muestras
    return (1/n) * sum(hipotesis(x, theta_0, theta_1) - y) # (1/n) * sum(h(x) -

def delta_theta_1(x, y, theta_0, theta_1): # Derivada parcial de la función de
    n = len(y) # Número de muestras
    return (1/n) * sum((hipotesis(x, theta_0, theta_1) - y) * x) # (1/n) * sum

def actualiza_thetas(theta_0, theta_1, alpha, delta_theta_0, delta_theta_1): #
    theta_0 = theta_0 - alpha * delta_theta_0
    theta_1 = theta_1 - alpha * delta_theta_1
    return theta_0, theta_1

def entrenando(x, y, theta_0, theta_1, alpha, iteraciones): # Entrenamos el mo
    for i in range(iteraciones):  # Iteramos sobre el número de iteraciones
        d_theta_0 = delta_theta_0(x, y, theta_0, theta_1) # Calculamos la deri
        d_theta_1 = delta_theta_1(x, y, theta_0, theta_1) # Calculamos la deri
        theta_0, theta_1 = actualiza_thetas(theta_0, theta_1, alpha, d_theta_0

        # Reducimos la alpha cada 1000 iteraciones, esto ayuda mucho con homew
        if (i + 1) % 1000 == 0: # Si el residuo de i + 1 entre 1000 es 0
            alpha *= 0.5 # Reducimos alpha a la mitad

    return theta_0, theta_1

def predice(x, theta_0, theta_1): # Función para predecir
    return hipotesis(x, theta_0, theta_1) >= 0.5 # Si la hipótesis es mayor o

def metricas(y_real, y_pred): # Función para calcular las métricas
    VP = sum((y_real == 1) & (y_pred == 1)) # Verdaderos positivos
    VN = sum((y_real == 0) & (y_pred == 0)) # Verdaderos negativos
    FP = sum((y_real == 0) & (y_pred == 1)) # Falsos positivos
    FN = sum((y_real == 1) & (y_pred == 0)) # Falsos negativos

    accuracy = (VP + VN) / len(y_real) # Verdaderos positivos + Verdaderos nega
    precision = VP / (VP + FP) if (VP + FP) != 0 else 0 # Verdaderos positivos
    recall = VP / (VP + FN) if (VP + FN) != 0 else 0 # Verdaderos positivos /
    F1 = 2 * (precision * recall) / (precision + recall) if (precision + recal

    return accuracy, precision, recall, F1
```

In [45]:
```
theta_0_attendance, theta_1_attendance = entrenando(attendance, y, theta_0, th
predicciones_attendance = predice(attendance, theta_0_attendance, theta_1_atte
metricas_attendance = metricas(y, predicciones_attendance) # Calculamos las mé
```

In [46]:
```
print("Resultados para Attendance:")
print("Accuracy: ", metricas_attendance[0])
print("Precision: ", metricas_attendance[1])
print("Recall: ", metricas_attendance[2])
print("F1 Score: ", metricas_attendance[3])
```

```
Resultados para Attendance:
Accuracy:  0.625
Precision:  0.6666666666666666
Recall:  0.8
F1 Score:  0.7272727272727272
```

In [47]:
```
costo_attendance = costo_logistica(attendance, y, theta_0_attendance, theta_1_a
print("Costo J para Attendance: ", costo_attendance)
```

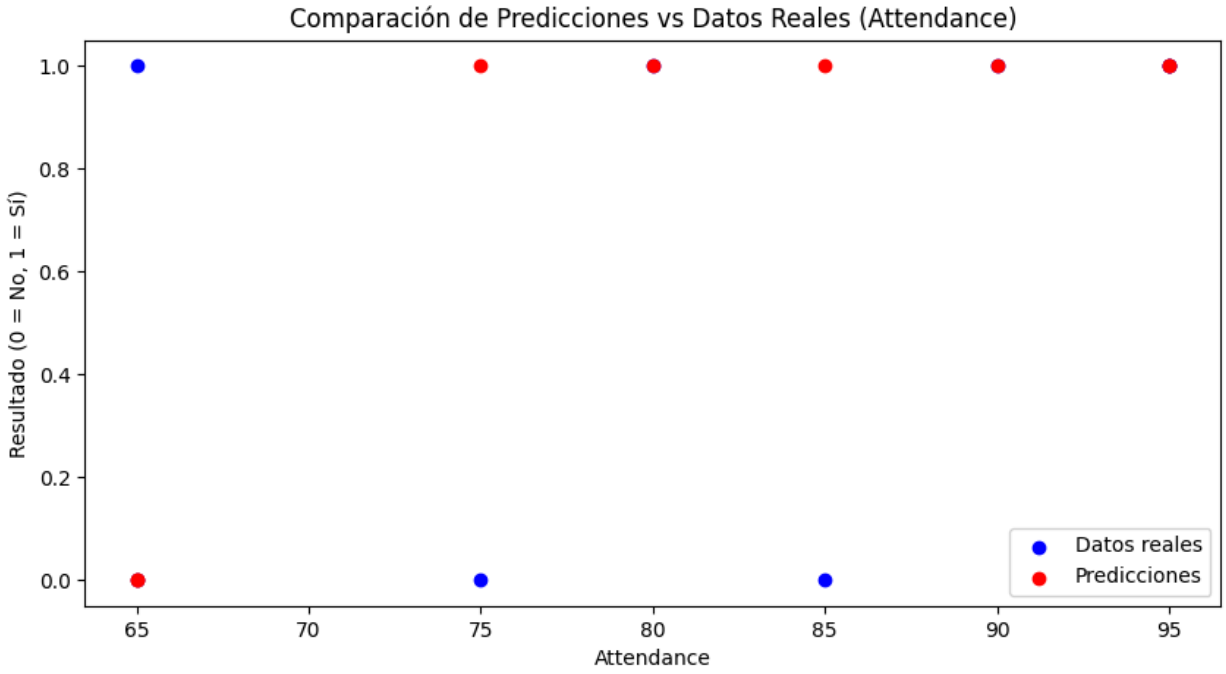
```
Costo J para Attendance:  0.5684548399733602
```

In [48]:
```
tabla_attendance = pd.DataFrame({'Attendance': attendance, 'Valor Real': y, 'P
tabla_attendance
```

Out[48]:

	Attendance	Valor Real	Predicción
0	80	1	1
1	65	0	0
2	95	1	1
3	95	1	1
4	85	0	1
5	75	0	1
6	90	1	1
7	65	1	0

In [49]:
```
plt.figure(figsize = (10, 5))
plt.scatter(attendance, y, color = 'blue', label = 'Datos reales')
plt.scatter(attendance, predicciones_attendance, color = 'red', label = 'Predi
plt.title("Comparación de Predicciones vs Datos Reales (Attendance)")
plt.xlabel("Attendance")
plt.ylabel("Resultado (0 = No, 1 = Sí)")
plt.legend()
plt.show()
```



In [50]:
```
theta_0_homework, theta_1_homework = entrenando(homework, y, theta_0, theta_1,
predicciones_homework = predice(homework, theta_0_homework, theta_1_homework) #
metricas_homework = metricas(y, predicciones_homework) # Calculamos las métrica
```

In [51]:
```
costo_homework = costo_logistica(homework, y, theta_0_homework, theta_1_homewo
print("Costo J para Homework: ", costo_homework)
```

```
Costo J para Homework:  0.20266200919692037
```

In [52]:
```
print("Resultados para Homework:")
print("Accuracy: ", metricas_homework[0])
print("Precision: ", metricas_homework[1])
print("Recall: ", metricas_homework[2])
print("F1 Score: ", metricas_homework[3])
```

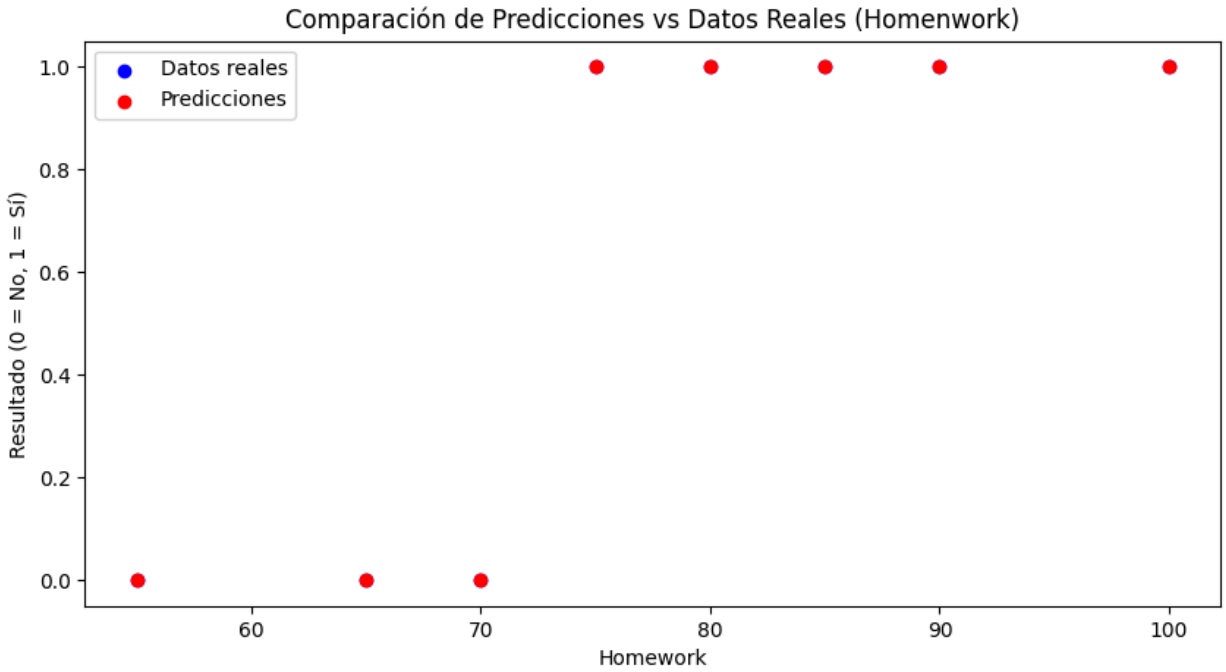
```
Resultados para Homework:
Accuracy:  1.0
Precision:  1.0
Recall:  1.0
F1 Score:  1.0
```

In [53]:
```
tabla_homework = pd.DataFrame({'Homework': homework, 'Valor Real': y, 'Predicc
tabla_homework
```

Out[53]:

	Homework	Valor Real	Predicción
0	75	1	1
1	70	0	0
2	85	1	1
3	100	1	1
4	65	0	0
5	55	0	0
6	90	1	1
7	80	1	1

In [54]:
```
plt.figure(figsize = (10, 5))
plt.scatter(homework, y, color = 'blue', label = 'Datos reales')
plt.scatter(homework, predicciones_homework, color = 'red', label = 'Prediccio
plt.title("Comparación de Predicciones vs Datos Reales (Homenwork)")
plt.xlabel("Homework")
plt.ylabel("Resultado (0 = No, 1 = Sí)")
plt.legend()
plt.show()
```


Comparación de Predicciones vs Datos Reales (Homenwork)
Datos reales
Predicciones
Resultado (0 = No, 1 = Si)
1.0
0.8
0.6
0.4
0.2
0.0
60
70
80
90
100
Homework