Proyecto1

May 28, 2024

1 Proyecto 1 - Celsius a Farenheit

1.0.1 Importar las librerías

[ ]:
```
import tensorflow as tf
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
2024-05-28 22:07:21.887274: I tensorflow/core/platform/cpu_feature_guard.cc:210]
This TensorFlow binary is optimized to use available CPU instructions in
performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild
TensorFlow with the appropriate compiler flags.
```

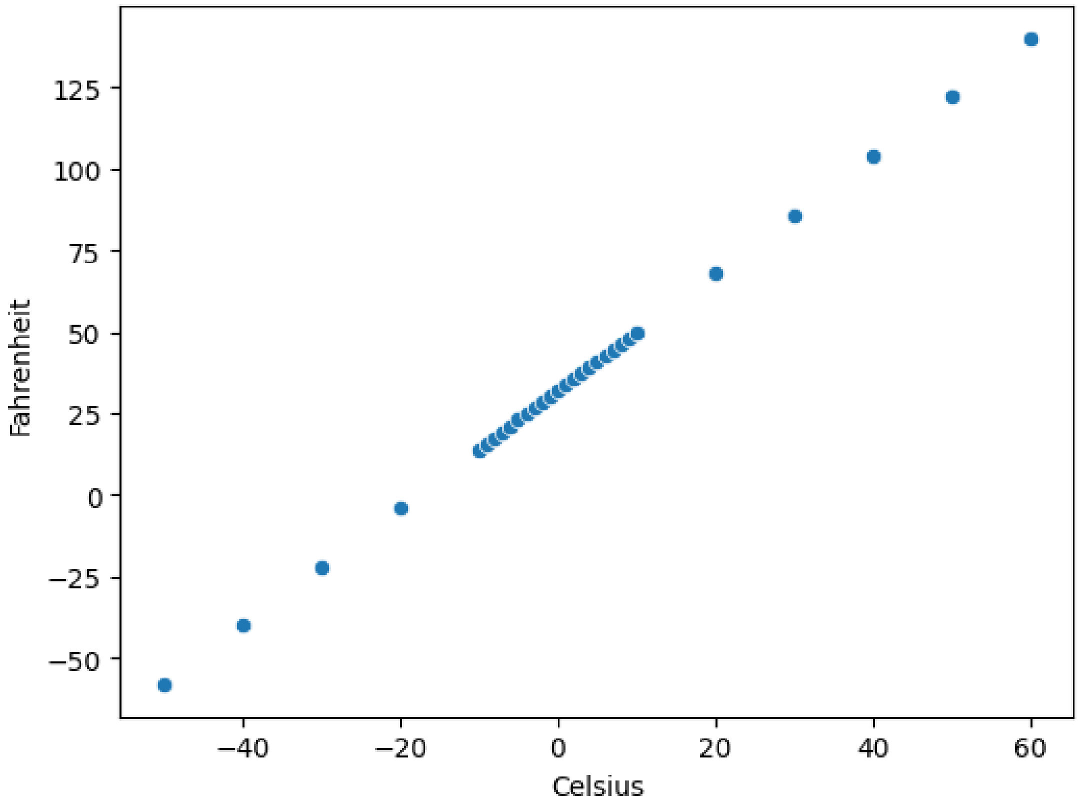
1.0.2 Importar el dataset

[ ]:
```
url = 'https://raw.githubusercontent.com/alejocampos1/AI-Projects/Projecto-1/
  ↪Proyecto%201/celsius_a_fahrenheit.csv'
temperature_df = pd.read_csv(url)
```

1.0.3 Visualización del dataset

[ ]:
```
#Se necesita especificar el eje que va a asignarse a cada columna
sns.scatterplot(x=temperature_df['Celsius'], y=temperature_df['Fahrenheit'])
```

[ ]:
```
<Axes: xlabel='Celsius', ylabel='Fahrenheit'>
```



1.0.4 Crear set de entrenamiento

```
[ ]: x_train = temperature_df['Celsius']
     y_train = temperature_df['Fahrenheit']
```

1.0.5 Crear modelo AI

```
[ ]: model = tf.keras.Sequential()
     model.add(tf.keras.layers.Dense(units=1, input_shape=[1]))
```

```
/opt/anaconda3/envs/proyecto_modelos/lib/python3.12/site-
packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
[ ]: model.summary()
```

```
Model: "sequential"



 Layer (type)                     Output Shape                  Param #
 dense (Dense)                    (None, 1)                        2


 Total params: 2 (8.00 B)

 Trainable params: 2 (8.00 B)

 Non-trainable params: 0 (0.00 B)
```

1.0.6 Compilado

[ ]:

```
model.compile(optimizer=tf.keras.optimizers.Adam(1.0),␣
 ↪loss='mean_squared_error')
```

1.0.7 Entrenando el modelo

[ ]:

```
epochs_history = model.fit(x_train, y_train, epochs=100)
```

```
Epoch 1/100
1/1               0s 479ms/step - loss:
4816.5850
Epoch 2/100
1/1               0s 24ms/step - loss:
2480.1663
Epoch 3/100
1/1               0s 25ms/step - loss:
1192.3315
Epoch 4/100
1/1               0s 22ms/step - loss:
840.8457
Epoch 5/100
1/1               0s 20ms/step - loss:
1108.7567
Epoch 6/100
1/1               0s 21ms/step - loss:
1521.4448
Epoch 7/100
1/1               0s 20ms/step - loss:
1719.6919
Epoch 8/100
1/1               0s 19ms/step - loss:
1619.1565
Epoch 9/100
1/1               0s 21ms/step - loss:
1314.1873
```

```
Epoch 10/100
1/1             0s 20ms/step - loss:
947.5693
Epoch 11/100
1/1             0s 20ms/step - loss:
641.2819
Epoch 12/100
1/1             0s 20ms/step - loss:
467.2536
Epoch 13/100
1/1             0s 20ms/step - loss:
437.1372
Epoch 14/100
1/1             0s 21ms/step - loss:
508.0764
Epoch 15/100
1/1             0s 20ms/step - loss:
606.7838
Epoch 16/100
1/1             0s 21ms/step - loss:
664.6744
Epoch 17/100
1/1             0s 21ms/step - loss:
645.7062
Epoch 18/100
1/1             0s 20ms/step - loss:
553.5973
Epoch 19/100
1/1             0s 20ms/step - loss:
420.8061
Epoch 20/100
1/1             0s 20ms/step - loss:
289.9388
Epoch 21/100
1/1             0s 20ms/step - loss:
196.3358
Epoch 22/100
1/1             0s 24ms/step - loss:
156.4730
Epoch 23/100
1/1             0s 22ms/step - loss:
164.4920
Epoch 24/100
1/1             0s 21ms/step - loss:
197.4441
Epoch 25/100
1/1             0s 21ms/step - loss:
226.9263
```

```
Epoch 26/100
1/1          0s 21ms/step - loss:
231.6635
Epoch 27/100
1/1          0s 20ms/step - loss:
205.1776
Epoch 28/100
1/1          0s 20ms/step - loss:
155.8637
Epoch 29/100
1/1          0s 20ms/step - loss:
100.9148
Epoch 30/100
1/1          0s 20ms/step - loss:
57.7771
Epoch 31/100
1/1          0s 20ms/step - loss:
36.8747
Epoch 32/100
1/1          0s 20ms/step - loss:
38.2097
Epoch 33/100
1/1          0s 20ms/step - loss:
52.7711
Epoch 34/100
1/1          0s 20ms/step - loss:
67.7118
Epoch 35/100
1/1          0s 22ms/step - loss:
72.5948
Epoch 36/100
1/1          0s 23ms/step - loss:
63.6834
Epoch 37/100
1/1          0s 20ms/step - loss:
44.5359
Epoch 38/100
1/1          0s 23ms/step - loss:
23.1931
Epoch 39/100
1/1          0s 22ms/step - loss:
7.7688
Epoch 40/100
1/1          0s 20ms/step - loss:
2.6609
Epoch 41/100
1/1          0s 21ms/step - loss:
7.0106
```

```
Epoch 42/100
1/1              0s 21ms/step - loss:
15.8462
Epoch 43/100
1/1              0s 20ms/step - loss:
23.0500
Epoch 44/100
1/1              0s 20ms/step - loss:
24.4774
Epoch 45/100
1/1              0s 20ms/step - loss:
19.6583
Epoch 46/100
1/1              0s 20ms/step - loss:
11.4308
Epoch 47/100
1/1              0s 21ms/step - loss:
3.9887
Epoch 48/100
1/1              0s 20ms/step - loss:
0.5459
Epoch 49/100
1/1              0s 20ms/step - loss:
1.8390
Epoch 50/100
1/1              0s 20ms/step - loss:
6.1124
Epoch 51/100
1/1              0s 20ms/step - loss:
10.3958
Epoch 52/100
1/1              0s 20ms/step - loss:
12.2356
Epoch 53/100
1/1              0s 20ms/step - loss:
10.9006
Epoch 54/100
1/1              0s 20ms/step - loss:
7.4914
Epoch 55/100
1/1              0s 20ms/step - loss:
4.0536
Epoch 56/100
1/1              0s 22ms/step - loss:
2.3167
Epoch 57/100
1/1              0s 24ms/step - loss:
2.7971
```

```
Epoch 58/100
1/1          0s 24ms/step - loss:
4.7020
Epoch 59/100
1/1          0s 26ms/step - loss:
6.5749
Epoch 60/100
1/1          0s 26ms/step - loss:
7.2164
Epoch 61/100
1/1          0s 25ms/step - loss:
6.3206
Epoch 62/100
1/1          0s 25ms/step - loss:
4.5060
Epoch 63/100
1/1          0s 25ms/step - loss:
2.8130
Epoch 64/100
1/1          0s 24ms/step - loss:
2.0361
Epoch 65/100
1/1          0s 25ms/step - loss:
2.3021
Epoch 66/100
1/1          0s 25ms/step - loss:
3.1052
Epoch 67/100
1/1          0s 24ms/step - loss:
3.7093
Epoch 68/100
1/1          0s 24ms/step - loss:
3.6258
Epoch 69/100
1/1          0s 22ms/step - loss:
2.8676
Epoch 70/100
1/1          0s 21ms/step - loss:
1.8606
Epoch 71/100
1/1          0s 21ms/step - loss:
1.1177
Epoch 72/100
1/1          0s 22ms/step - loss:
0.9117
Epoch 73/100
1/1          0s 22ms/step - loss:
1.1465
```

```
Epoch 74/100
1/1             0s 24ms/step - loss:
1.4783
Epoch 75/100
1/1             0s 25ms/step - loss:
1.5712
Epoch 76/100
1/1             0s 24ms/step - loss:
1.3080
Epoch 77/100
1/1             0s 23ms/step - loss:
0.8278
Epoch 78/100
1/1             0s 21ms/step - loss:
0.3943
Epoch 79/100
1/1             0s 24ms/step - loss:
0.2061
Epoch 80/100
1/1             0s 25ms/step - loss:
0.2779
Epoch 81/100
1/1             0s 48ms/step - loss:
0.4617
Epoch 82/100
1/1             0s 31ms/step - loss:
0.5696
Epoch 83/100
1/1             0s 23ms/step - loss:
0.5036
Epoch 84/100
1/1             0s 23ms/step - loss:
0.3046
Epoch 85/100
1/1             0s 24ms/step - loss:
0.1010
Epoch 86/100
1/1             0s 24ms/step - loss:
0.0074
Epoch 87/100
1/1             0s 22ms/step - loss:
0.0489
Epoch 88/100
1/1             0s 21ms/step - loss:
0.1583
Epoch 89/100
1/1             0s 21ms/step - loss:
0.2372
```

```
Epoch 90/100
1/1              0s 22ms/step - loss:
0.2278
Epoch 91/100
1/1              0s 21ms/step - loss:
0.1454
Epoch 92/100
1/1              0s 20ms/step - loss:
0.0549
Epoch 93/100
1/1              0s 20ms/step - loss:
0.0167
Epoch 94/100
1/1              0s 19ms/step - loss:
0.0451
Epoch 95/100
1/1              0s 19ms/step - loss:
0.1052
Epoch 96/100
1/1              0s 21ms/step - loss:
0.1463
Epoch 97/100
1/1              0s 23ms/step - loss:
0.1403
Epoch 98/100
1/1              0s 24ms/step - loss:
0.0975
Epoch 99/100
1/1              0s 24ms/step - loss:
0.0531
Epoch 100/100
1/1              0s 24ms/step - loss:
0.0368
```

1.0.8 Evaluación del modelo

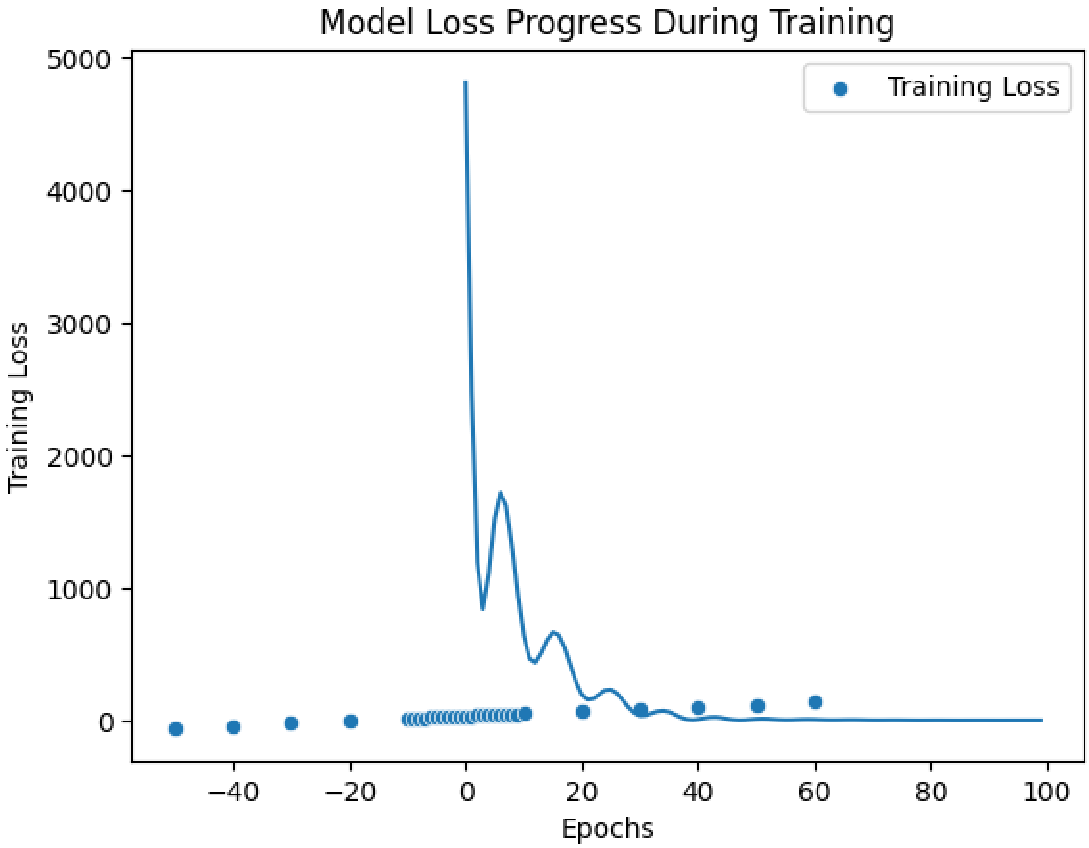
[ ]:

```
epochs_history.history.keys()

sns.scatterplot(x=temperature_df['Celsius'], y=temperature_df['Fahrenheit'])
plt.plot(epochs_history.history['loss'], )
plt.title('Model Loss Progress During Training')
plt.xlabel('Epochs')
plt.ylabel('Training Loss')
plt.legend(['Training Loss'])
```

[ ]:

```
<matplotlib.legend.Legend at 0x13370b080>
```



```
[ ]: model.get_weights()
```

```
[ ]: [array([[1.8064165]], dtype=float32), array([31.807835], dtype=float32)]
```

1.0.9 Predicciones

```
[ ]: temp_c = 0
     temp_c_array = np.array([[temp_c]], dtype=float)  # Convertir a una matriz 2D
     temp_f = model.predict(temp_c_array)
     print(temp_f)
```

```
1/1           0s 37ms/step
[[31.807835]]
```

```
[ ]: temp_f = 9/5 * temp_c + 32
     print(temp_f)
```

```
32.0
```