Modelo de Regresión Lineal Simple 📉

Link para ver solo el codigo:

https://colab.research.google.com/drive/19KxYuE98ErUwXOjTssvql8ZqYYc3MQW0?authuser=2#scrollTo=pdx-quStkjFa

Conversión de Temperaturas 🌡️

Este análisis se enfoca en el uso de un modelo de regresión lineal para predecir los **valores numéricos** de la conversión entre temperaturas en grados Celsius a Fahrenheit. Teniendo como entrada el dato de los grados Celsius y la salida su equivalente en Fahrenheit. Este modelo aprende autónomamente a convertir temperaturas de Celsius a Fahrenheit, logrando predecir con precisión sin depender de la fórmula convencional:

"Grados Fahrenheit = (grados centígrados × 9/5) + 32"

El modelo logra predecir con precisión basándose en datos, evitando depender exclusivamente de dicha fórmula. Esto es posible gracias a su capacidad de interpretar patrones en los datos.

Librerías Utilizadas

Para este análisis, se emplearon las siguientes librerías: Pandas, Seaborn, Matplotlib y LinearRegression.

Concepto de Modelo de Regresión Lineal

La regresión lineal se parece a trazar una línea recta entre puntos en un gráfico. Esta línea ayuda a predecir cómo se comportarán otros puntos según la forma en que están dispuestos los puntos conocidos. En esencia, el modelo busca patrones en los datos pasados para hacer predicciones.

Por ejemplo imagina que alguien está vendiendo limonadas y quiere saber cuántas va a vender con respecto a la temperatura del día. Si esa persona se da cuenta de que cuando hace calor se venden más limonadas, puede utilizar la regresión lineal para trazar una línea que conecte esos puntos. Luego, si observa que el pronóstico del clima anuncia un día caluroso, esa línea le brindará una idea de cuántas limonadas podría vender basándose en lo observado en días anteriores. La regresión lineal ayuda a predecir situaciones sencillas utilizando la información que ya se tiene.

¿Cómo se encuentra la relación entre datos para crear una predicción?

Se utiliza el método de los mínimos cuadrados ordinarios o OLS (Ordinary Least Squares) para encontrar la relación entre datos y realizar predicciones. En este proceso, se manejan dos tipos de datos: la variable dependiente, que es la que se desea predecir, y la variable independiente, que se utiliza para realizar la predicción.

Importación de Bibliotecas

- **Pandas:** Es una herramienta fundamental para el procesamiento y gestión eficiente de datos en Python. Se utiliza para cargar, limpiar y transformar conjuntos de datos, preparándolos para el análisis.
- **Matplotlib:** Esta biblioteca es esencial para la generación de gráficos y la visualización de datos. Se emplea para crear visualizaciones que permiten explorar la relación entre las variables, presentar la línea de regresión.
- **Seaborn:** Complementaria a Matplotlib, Seaborn se utiliza para mejorar la estética y la presentación de las visualizaciones. Ayuda a crear gráficos estadísticos más atractivos y sofisticados, lo que facilita la comprensión de la relación entre las variables y los resultados del modelo.
- **LinearRegression:** Es una clase esencial que se emplea para llevar a cabo predicciones mediante regresión lineal. Esta clase, perteneciente a la biblioteca scikit-learn, permite entrenar modelos de regresión lineal, evaluar su desempeño y utilizarlos para realizar predicciones basadas en la relación lineal entre las variables de entrada y salida.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Carga y Exploración de Datos

Condiciones para que los datos sean adecuados para la construcción de un modelo.

- **Relación Lineal:** Asegúrese de que haya una conexión clara entre las dos datos que se está estudiando. Si un dato aumenta, el otro debería hacer algo similar.
- **Distribución Normal:** Revise de que los datos se vean como una colina en un gráfico, con la mayoría de los puntos cerca del medio y menos puntos en los extremos.
- **No Multicolinealidad:** Evite que los datos que se está estudiando estén muy relacionadas entre sí. Si dos datos son muy parecidas, es difícil decir cuál de las dos está afectando lo que se está investigando.
- **No Autocorrelación:** Asegúrese de que no haya patrones extraños en los datos que se repitan en el tiempo o en el orden en que los recopile. Esto podría confundir tus resultados.
- **Varianzas Iguales (Homocedasticidad):** Verifique que la diferencia entre lo que se predice el modelo y lo que realmente sucede sea más o menos la misma en todas las situaciones. No debería ser mucho más grande en un lugar que en otro.

- **Calidad de los datos:** Use datos reales y no sintéticos. Evite duplicados para una mejor predicción.

Carga los datos desde el archivo CSV "celsius.csv" utilizando Pandas y se explora la información básica del DataFrame (tabla de base de datos).

```
datos = pd.read_csv("celsius.csv")
```

Devuelve información esencial sobre el DataFrame y sus columnas.

- **RangeIndex:** 7 entries, 0 to 6: Esto indica que el DataFrame tiene un índice que va desde 0 hasta 6, lo que significa que tiene 7 filas en total.
- **Data columns (total 2 columns):** Muestra que el DataFrame contiene un total de 2 columnas.
- **Column:** Aquí se enumeran las columnas presentes en el DataFrame, que son "celsius" y "fahrenheit".
- **Non-Null Count:** Indica que en ambas columnas ("celsius" y "fahrenheit") no hay valores nulos (missing values). Cada columna tiene 7 valores no nulos, lo que significa que todas las filas tienen datos en estas columnas.
- **Dtype:** Describe los tipos de datos de las columnas. La columna "celsius" contiene valores de tipo int64 (enteros de 64 bits) y la columna "fahrenheit" contiene valores de tipo float64 (números de coma flotante de 64 bits).
- **Memory Usage:** Estima la cantidad de memoria utilizada por el DataFrame para almacenar los datos. En este caso, el DataFrame utiliza aproximadamente 244.0 bytes de memoria.

```
datos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   celsius     7 non-null      int64  
 1   fahrenheit  7 non-null      float64
dtypes: float64(1), int64(1)
memory usage: 244.0 bytes
```

Muestra las primeras filas de los datos en el DataFrame

```
datos.head()
```

```
   celsius  fahrenheit
0      -40       -40.0
1      -10        14.0
```

```
2         0         32.0
3         8         46.4
4        15         59.0
```

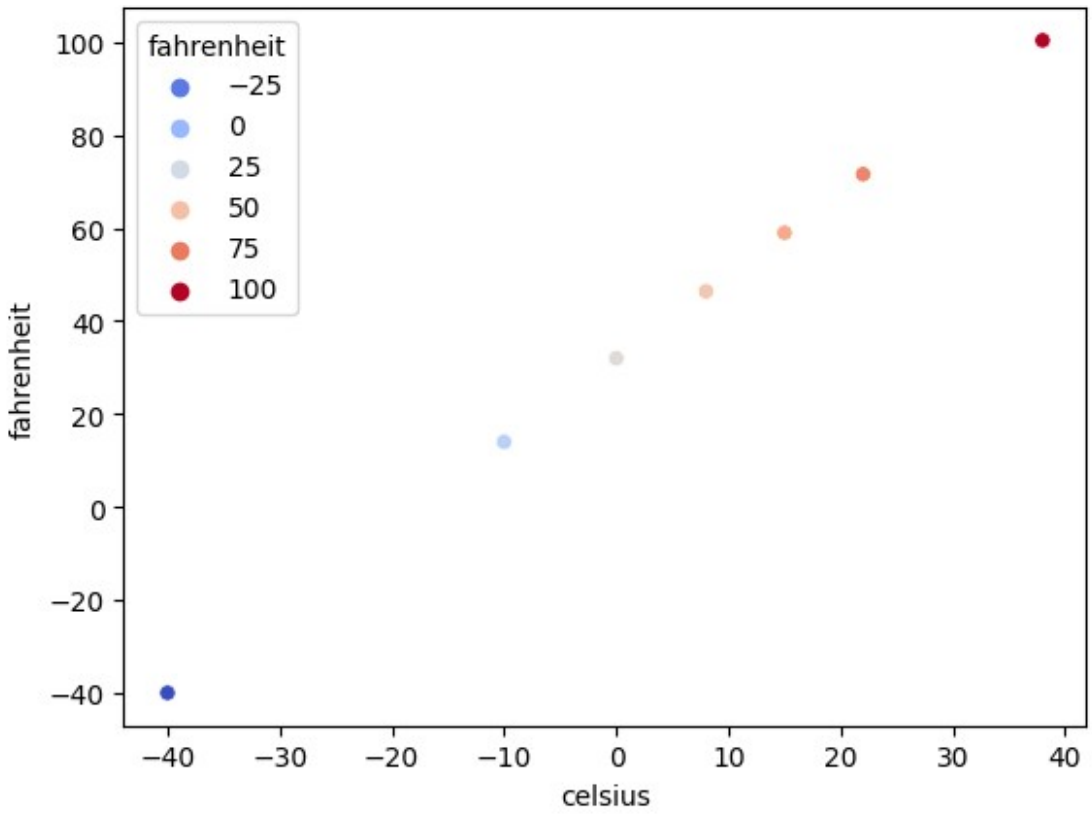
Gráfico de Dispersión

Gráfico de dispersión utiliza Seaborn para visualizar la relación entre las temperaturas en grados Celsius y Fahrenheit

Un gráfico de regresión lineal simple muestra cómo dos datos se relacionan. En el gráfico, en el eje **(X)** se coloca la variable que se usa para predecir, como los grados Celsius. En el eje **(Y)** se pone lo que se tratando de predecir, como los grados Fahrenheit. Si la línea en el gráfico sube, significa que cuando los grados Celsius suben, los grados Fahrenheit también suben. Eso se llama una relación positiva, donde un aumento en un dato predice un aumento en otro.

```
sns.scatterplot(x="celsius", y="fahrenheit", data=datos,
hue="fahrenheit",palette="coolwarm")
#vigilar los espacios entre los nombres de las columnas
("fahrenheit" != " fahrenheit")

# Mostrar el gráfico
plt.show()
```



Análisis de la Recta

La recta se ajusta a los datos ya que cuando estos están altamente correlacionados y siguen un patrón predecible, la recta se ajusta de forma precisa a la mayoría de los puntos, lo que permite al modelo predecir las temperaturas en Fahrenheit con alta precisión a partir de los grados Celsius.

grafica

Preparación de Datos

Se extraen las características (temperatura en Celsius) y las etiquetas (temperatura en Fahrenheit) del DataFrame para preparar los datos para el modelado.

La transformación de datos en arreglos bidimensionales es necesaria para preparar los datos para su uso en el modelo. Esto se hace porque el modelo requiere que los datos estén organizados en una forma específica: una matriz con filas y columnas.

```
#caracteristicas (X) - etiqueta (y)
#extracción del dataset
X = datos ["celsius"]
Y = datos ["fahrenheit"]

# Transformación de los datos para que estén en el formato adecuado
para el modelo
# Acomoda los datos en un arreglos [[],[],[],[],[],[]]

X_procesada = X.values.reshape(-1,1)
Y_procesada = Y.values.reshape(-1,1)
print("X - celsius:")
print(X_procesada)
print("Y - fahrenheit:")
print(Y_procesada)
```

```
X - celsius:
[[-40]
 [-10]
 [  0]
 [  8]
 [ 15]
 [ 22]
 [ 38]]
Y - fahrenheit:
[[-40. ]
 [ 14. ]
 [ 32. ]
 [ 46.4]
 [ 59. ]
 [ 71.6]
 [100.4]]
```

Creación y Entrenamiento del Modelo de Regresión Lineal

Se importa la clase LinearRegression de la biblioteca sklearn y se crea la instancia del modelo.

Luego, se ajusta el modelo a los datos de entrenamiento (temperaturas en Celsius y Fahrenheit).

El método **fit()** enseña al modelo a entender cómo se relacionan las variables de entrada con la variable de salida, el modelo ajusta o "afina" sus configuraciones internas, conocidas como parámetros, para que haga predicciones precisas.

```
from sklearn.linear_model import LinearRegression

modelo = LinearRegression()

#entrenamiento
modelo.fit(X_procesada,Y_procesada)
```

```
LinearRegression()
```

Predicción de Temperatura

Ahora que el modelo está entrenado, puede ser usado para predecir la temperatura en Fahrenheit correspondiente a una temperatura en Celsius dada.

```
#dato de entrada
celsius = 123

prediccion = modelo.predict([[celsius]])

#dato de salida
print(f"{celsius} grados Celsius son {round(prediccion[0][0], 1)}
grados Fahrenheit")
```

```
123 grados Celsius son 253.4 grados Fahrenheit
```

Evaluación del Modelo

Se evalua el modelo con el coeficiente de determinación, también conocido como R^2(r cuadrado), es una medida que ayuda a evaluar cuán bien un modelo puede hacer predicciones basadas en datos. Se puede pensar en términos de un tiro al blanco: un R^2 cercano a 1 significa que las predicciones del modelo están muy cerca del objetivo, mientras que un R^2 cercano a 0 indica que las predicciones no se ajustan bien a los datos reales.

```
# Valor del coeficiente de determinación R²
coeficiente = modelo.score(X_procesada,Y_procesada)
print("R²: ",coeficiente)

#porcentaje de error
por_coeficiente = coeficiente * 100
```

```
print("Coeficiente de determinación :", round(por_coeficiente, 2),
"%")
```

```
R²:  1.0
Coeficiente de determinación : 100.0 %
```

Conclusión

El modelo de regresión lineal utilizado realizó predicciones precisas de la temperatura en Fahrenheit a partir de la temperatura en Celsius, incluso con un conjunto de datos pequeño. Esto se debe a su capacidad para capturar patrones con pocos ejemplos para el entrenamiento. El coeficiente de determinación del modelo fue de **1.0**, lo que indica un ajuste perfecto a los datos de entrenamiento.

Consideraciones y Limitaciones del Modelo de Regresión Lineal Simple

1.Linealidad: El modelo asume que las relaciones entre las variables son lineales, lo que puede ser una limitación si la relación es compleja.

2.Calidad de los Datos: La precisión del modelo depende de datos limpios y precisos, por lo que errores o datos incompletos pueden afectar su desempeño.

3.Independencia de Errores: Los errores en las predicciones deben ser independientes entre sí para que el modelo funcione correctamente.

4.Multicolinealidad: Correlaciones fuertes entre variables predictoras pueden dificultar la interpretación de su impacto en la variable objetivo.

5.Relevancia de las Variables: No todas las variables predictoras son igualmente importantes, y es necesario identificar las más relevantes.

6.Datos Numéricos: El modelo funciona mejor con datos numéricos en lugar de variables categóricas, y a veces es necesario convertirlas para su uso efectivo.

Estudio extra

1.Regresión Líneal Simple 📈 En Python 🐍 : https://www.youtube.com/watch?v=b7gOUbSmGIY&ab_channel=RafaGonzalezGouveia

2.REGRESIÓN LINEAL SIMPLE - TEORÍA | #13 Curso Machine Learning con Python : https://www.youtube.com/watch?v=5TcA5M5z4sA&ab_channel=AprendeIAconLigdiGonzalez

3.REGRESIÓN LINEAL SIMPLE - PRÁCTICA | #15 Curso Machine Learning con Python : https://www.youtube.com/watch?v=YLGhVBB5rGU&ab_channel=AprendeIAconLigdiGonzalez

4.Regresión Lineal con Python: https://www.youtube.com/watch?v=1CGbP0l0iqo&ab_channel=C%C3%B3digoM%C3%A1quina

5.Regresión Lineal y Mínimos Cuadrados Ordinarios | DotCSV :https://www.youtube.com/watch?v=k964_uNn3l0&ab_channel=DotCSV