DEFINICIÓN DE REGRESIÓN LINEAL

Regresión Lineal Simple: Es un método estadístico que modela la relación entre una variable dependiente (resultado) y una variable independiente (predicción) a través de una línea recta. La ecuación general de la regresión lineal simple es:

$y=mx+b$

donde:

- y es la variable dependiente.
- m es la pendiente de la línea.
- x es la variable independiente.
- b es la intersección en el eje y.

Regresión Lineal Múltiple: Es una extensión de la regresión lineal simple que utiliza múltiples variables independientes para predecir una variable dependiente. La ecuación general de la regresión lineal múltiple es:

$y=b0+b1x1+b_2x_2+...+b_nx_n$

donde:

- y es la variable dependiente.
- b0 es la intersección en el eje y.
- b1,b2,...,bn son las pendientes de las variables independientes x1,x2,...,xn

Ejemplo de Regresión Lineal Simple

Supongamos que queremos predecir el precio de un producto en función de la cantidad de ventas.

```
# Importar las librerías necesarias
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Datos de ejemplo: cantidad de ventas (X) y precio del producto (y)
X = np.array([[1], [2], [3], [4], [5]])  # Cantidad de ventas
y = np.array([10, 15, 20, 25, 30])       # Precio del producto

# Crear el modelo de regresión lineal
model = LinearRegression()
model.fit(X, y)
```

```
# Hacer predicciones
y_pred = model.predict(X)

# Visualizar la línea de regresión
plt.scatter(X, y, color='blue', label='Datos Reales')  # Puntos de datos
plt.plot(X, y_pred, color='red', label='Línea de Regresión')  # Línea de
regresión
plt.title('Regresión Lineal Simple')
plt.xlabel('Cantidad de Ventas')
plt.ylabel('Precio del Producto')
plt.legend()
plt.show()

# Coeficientes
print(f'Intercepto: {model.intercept_}')
print(f'Pendiente: {model.coef_[0]}')
```

Ejercicios de Regresión Lineal Simple

1. **Ejercicio 1**: Dado un conjunto de datos que relaciona las horas de estudio con las calificaciones de los estudiantes, crea un modelo de regresión lineal simple y predice la calificación para 4.5 horas de estudio.
2. **Ejercicio 2**: Usa datos de la temperatura y el consumo de helados para construir un modelo de regresión lineal simple. Predice el consumo de helados a una temperatura de 30 grados.
3. **Ejercicio 3**: Crea un modelo de regresión lineal simple utilizando el número de seguidores de un influencer y las interacciones (likes, comentarios) en sus publicaciones. Predice las interacciones para un influencer con 20,000 seguidores.

Ejemplo de Regresión Lineal Múltiple

Supongamos que queremos predecir el precio de una casa en función de su tamaño en pies cuadrados, el número de habitaciones y la ubicación.

```
# Importar las librerías necesarias
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression

# Datos de ejemplo: tamaño (pies cuadrados), número de habitaciones y precio
(en miles)
X = np.array([[1500, 3], [1800, 4], [2400, 3], [3000, 5], [3500, 4]])  # Tamaño
y habitaciones
y = np.array([300, 400, 500, 600, 700])  # Precio en miles

# Crear el modelo de regresión lineal múltiple
model = LinearRegression()
model.fit(X, y)
```

```
# Hacer predicciones
y_pred = model.predict(X)

# Mostrar los resultados
print("Predicciones de precios:", y_pred)
```

Ejercicios de Regresión Lineal Múltiple

1. **Ejercicio 1**: Usa datos que relacionan el precio de automóviles con el número de cilindros, el tamaño del motor y la marca del automóvil. Crea un modelo de regresión lineal múltiple para predecir el precio de un automóvil.
2. **Ejercicio 2**: Dada la relación entre el rendimiento de los estudiantes (calificaciones) y el número de horas de estudio, el número de tareas completadas y la asistencia a clases, construye un modelo de regresión lineal múltiple para predecir las calificaciones.
3. **Ejercicio 3**: Modela el precio de las casas utilizando el tamaño en metros cuadrados, el número de baños y la antigüedad de la casa. Predice el precio de una casa con 120 metros cuadrados, 2 baños y 10 años de antigüedad.

Implementación con Datos de un Influencer

Ahora, vamos a crear un caso real donde se use la regresión lineal múltiple para predecir el número de vistas de un video de Germán Garmendia en función de diversas características de los videos.

Código para Google Colab

```
# Importar las librerías necesarias
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Generar datos aleatorios para videos de Germán Garmendia
np.random.seed(42)
n_videos = 100

# Duración del video (en minutos)
duraciones = np.random.randint(5, 21, n_videos)

# Número de "me gusta"
me_gusta = np.random.randint(100, 5001, n_videos)

# Número de comentarios
comentarios = np.random.randint(10, 501, n_videos)

# Número de suscriptores
suscriptores = np.random.randint(10000000, 15000000, n_videos)
```

```
# Generar el número de vistas (en millones), con una relación lineal
vistas = (0.5 * duraciones + 0.03 * me_gusta + 0.1 * comentarios +
          0.0005 * suscriptores + np.random.normal(0, 50, n_videos))

# Crear un DataFrame con los datos generados
data = pd.DataFrame({
    'Duracion': duraciones,
    'Me_Gusta': me_gusta,
    'Comentarios': comentarios,
    'Suscriptores': suscriptores,
    'Vistas': vistas
})

# Mostrar las primeras filas del DataFrame
print(data.head())

# Crear matriz de características (X) y variable objetivo (y)
X = data[['Duracion', 'Me_Gusta', 'Comentarios', 'Suscriptores']]
y = data['Vistas']

# Dividir los datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Crear y entrenar el modelo de regresión lineal
model = LinearRegression()
model.fit(X_train, y_train)

# Realizar predicciones
y_pred = model.predict(X_test)

# Mostrar resultados
print("Predicciones de vistas:", y_pred)
print("Valores reales de vistas:", y_test.values)

# Visualizar los resultados
plt.scatter(y_test, y_pred, color='blue')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red',
linestyle='--')
plt.xlabel('Vistas reales (millones)')
plt.ylabel('Predicciones de vistas (millones)')
plt.title('Predicción de Vistas de Videos de Germán')
plt.show()
```