Regresión Lineal Múltiple en Python

28 de Marzo de 2025

1 Introducción

La regresión lineal es un método estadístico fundamental que modela la relación entre una variable dependiente y una o más variables independientes (regresores). Un modelo que está conformado por una sola variable independiente se conoce como regresión lineal simple, mientras que un modelo con más de una variable independiente se conoce como regresión lineal múltiple.

El modelo asume una relación lineal múltiple entre las variables y se expresa mediante la ecuación:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon \tag{1}$$

donde:

- y es la variable dependiente.
- x_1, x_2 son las variables independientes.
- β_0 es el intercepto.
- β_1, β_2 son las pendientes.
- ϵ es el término de error.

2 Metodología

Para realizar el ejercicio de regresión lineal múltiple, se siguieron los siguientes pasos:

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score

[5]:
%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')

[6]:
#Lectura del csv
data = pd.read_csv("./articulos_ml.csv")
#Visualizacion de las dimenciones
data.shape
```

Figure 1: Importación de librerías y carga de datos.

2.1 Visualización General de los Datos de Entrada

Se eliminaron las variables categóricas del DataFrame y se visualizó la distribución de los datos. Se escogió como variable dependiente el número de compartidos, mientras que la cantidad de palabras y otros atributos fueron las variables regresoras.

```
# Visualizamos rápidamente las caraterísticas de entrada
data.drop(['Title', 'url', 'Elapsed days'], axis=1).hist()
plt.show()
```

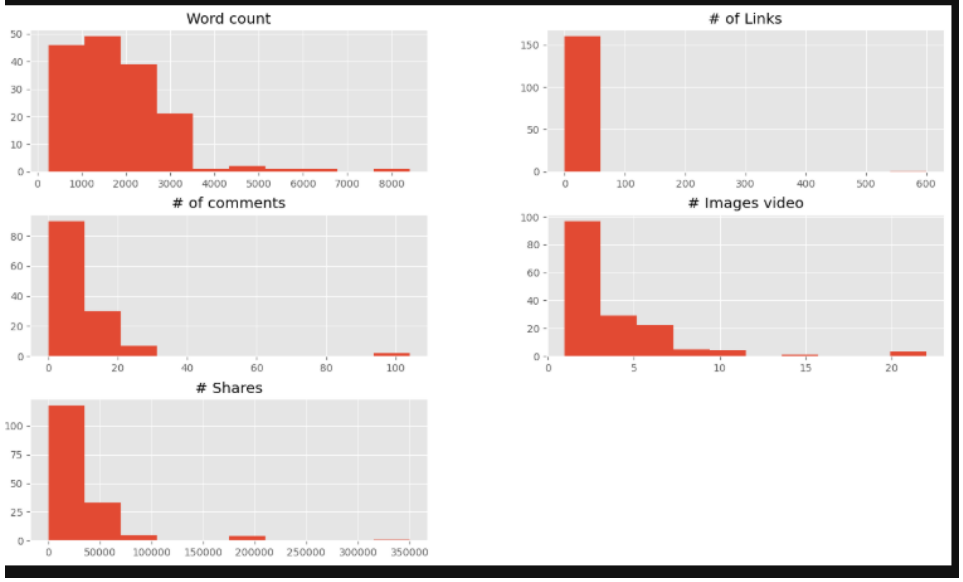


Figure 2: Visualización de datos.

2.2 Filtración de los Datos

Ya que escogimos nuestras variables dependiente e independientes, visualizamos su relación con una gráfica scatterplot. También establecimos un límite superior a los valores de los datos para evitar anomalías que puedan influir en el entrenamiento del modelo. Dado que no existe una clara relación lineal entre dos variables, agregamos una variable regresora adicional y así creamos un modelo de regresión lineal múltiple.

```
"""
Vamosa RECORTAR los datos en la zona donde se concentran más los punt
esto es en el eje X: entre 0 y 3.500 y en el eje Y: entre 0 y 80.000
"""

filtered_data = data[(data['Word count']<=3500)&(data['# Shares']<=80
colores=['orange','blue']
tamanios=[30,60]

f1=filtered_data['Word count'].values
f2=filtered_data['# Shares'].values
#Vamos a pintar en colores los puntos por debajo y por encima de la m
asignar=[]
for index, row in filtered_data.iterrows():
    if(row['Word count']>1808):
        asignar.append(colores[0])
    else:
        asignar.append(colores[1])
plt.scatter(f1, f2, c= asignar, s = tamanios[0])
plt.show()
```

Figure 3: Filtración de datos.

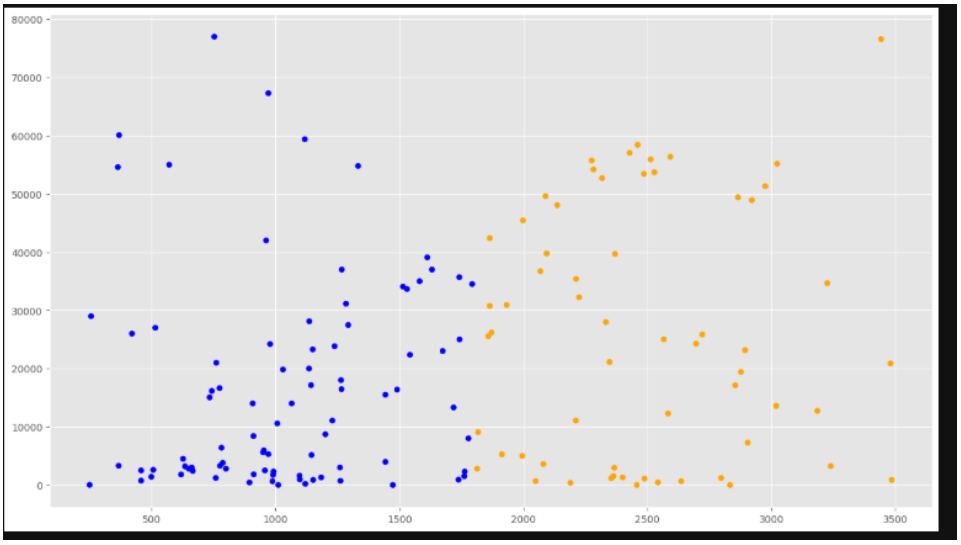


Figure 4: Gráfica de datos.

2.3 División de Datos en Conjuntos de Entrenamiento

Asignamos los valores de la columna "Word Count" como X_1, los valores de la columna "# Shares" como Y, y los valores de las columnas restantes como X_2.

```
suma = (filtered_data["# of Links"] + filtered_data["# of comments"].

dataX2 = pd.DataFrame()
dataX2["Word count"]= filtered_data["Word count"]
dataX2["suma"] = suma
XY_train = np.array(dataX2)
z_train = filtered_data["# Shares"].values
```

Figure 5: División de datos.

2.4 Ajuste del Modelo

Se crea el modelo de regresión lineal múltiple usando Scikit-Learn y se ajusta con los datos de entrenamiento. Además, generamos predicciones de Y con el modelo, las cuales deberían ser más precisas que en el modelo simple.

```
#Creamos un nuevo obkjeto de regresion lineal
regr2 = linear_model.LinearRegression()

#entrenamos el modelo con 2 dimenciones, obteniendo 2 coeficientes pa
regr2.fit(XY_train, z_train)

#Hacemos las predicciones
z_pred = regr2.predict(XY_train)
```

Figure 6: Ajuste del modelo.

2.5 Evaluación del Modelo

Una vez creado y ajustado el modelo, se imprimen los valores del intercepto, las pendientes, el error cuadrado medio y la varianza. Se observa que el error medio cuadrado sigue siendo grande, aunque ligeramente menor que el del modelo simple. Esto sugiere que la regresión lineal, ya sea simple o múltiple, podría no ser la mejor opción para explicar la relación entre los datos.

```
#Impresiones
print("Coeficientes: ", regr2.coef_)
print("MSE: %.2f" %mean_squared_error(z_train, z_pred))
print("Variance score: %.2f" %r2_score(z_train, z_pred))
```

```
Coeficientes:  [   6.63216324 -483.40753769]
MSE: 352122816.48
Variance score: 0.11
```

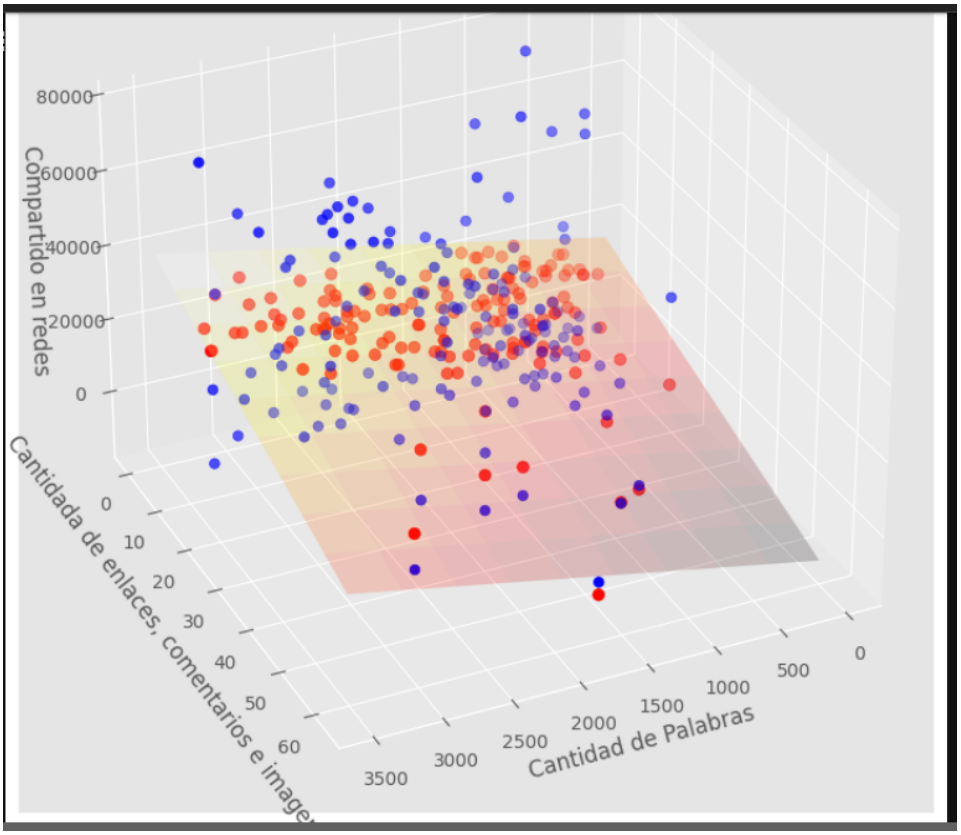
Figure 7: Evaluación del modelo.

3 Resultados

Para visualizar los resultados, creamos una gráfica 3D similar al scatterplot del modelo simple, observando el comportamiento de los datos en este modelo. Evaluamos también las predicciones del modelo con los mismos valores usados en la regresión simple.

```
#Graficamos
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

#Creamos una malla sobre la cual graficaremos el plano
xx, yy = np.meshgrid(np.linspace(0, 3500, num=10), np.linspace(0, 60, num=10))

#Calculamos los valores del plano para los puntos x e y
nuevoX = (regr2.coef_[0] * xx)
nuevoY = (regr2.coef_[1] * yy)

#Calculamos los correspondientes valores para z
z = (nuevoX + nuevoY + regr2.intercept_)

#Graficamos el plano
ax.plot_surface(xx, yy, z, alpha=0.2, cmap='hot')

#graficamos en azul los puntos en 3d
ax.scatter(XY_train[:, 0], XY_train[:, 1], z_train, c = 'blue', s = 30)

#graficamos en rojo los puntos que
ax.scatter(XY_train[:, 0], XY_train[:, 1], z_pred, c = 'red', s = 40)
```

```
#Situamos la camara para visualizar
ax.view_init(elev = 30, azim=65)

ax.set_xlabel('Cantidad de Palabras')
ax.set_ylabel('Cantidada de enlaces, comentarios e imagenes')
ax.set_zlabel('Compartido en redes')
ax.set_title('Regresion Lineal con multiples variables')
plt.show()
```



Para evaluar el desempeño del modelo, verificamos su predicción para el número de compartidos en redes cuando hay 2000 palabras en el post y una suma de 20 enlaces, comentarios e imágenes. Se observa que el modelo no acierta con exactitud a los valores esperados, pero su predicción se encuentra entre ellos. La relación entre las tres variables no parece ser estrictamente lineal, lo que sugiere que un modelo de regresión diferente podría ser más adecuado.

```
z_Dosmil = regr2.predict([[2000, 10+4+6]])
print(int(z_Dosmil))
```

```
20518
```

4 Conclusión

Los modelos de regresión lineal son herramientas estadísticas utilizadas para explicar la relación entre conjuntos de datos de manera lineal. Existen dos tipos principales:

- Regresión lineal simple: compuesta por una variable dependiente y una variable independiente o regresora.
- Regresión lineal múltiple: incluye más de una variable regresora.

En este caso, se intentó comprobar si el número de palabras y la suma de comentarios, enlaces e imágenes en un post tenían una relación lineal con el número de compartidos. Sin embargo, al graficar los datos y entrenar el modelo, se observó que no existía una correlación clara entre las variables.

El modelo no acertó con exactitud en sus predicciones, pero se aproximó a los valores esperados. Dado que los datos no parecen seguir un patrón lineal, sería recomendable probar con otros modelos de regresión, como la regresión logística o modelos no lineales, para mejorar las predicciones.