Regresión Lineal en Python

Aldo Hernández

Marzo 2025

1 Introducción

La regresión lineal es un algoritmo usado ampliamente en la estadística y el aprendizaje automático para hacer predicciones y mostrar tendencias a partir de un conjunto de datos, se basa en hacer una función lineal que indica la tendencia del conjunto y hacer predicciones a partir de ella.

La función se puede representar matemáticamente de la siguiente manera

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

donde β_i representa el coeficiente de la variable independiente x_i y x_i representa una variable explicativa i (nuestra etiqueta de interés).

La regresión lineal en el aprendizaje automático es un algoritmo que se puede clasificar como de aprendizaje supervisado ya que obtiene por sí mismo la función lineal que muestra la tendencia de nuestros datos.

2 Metodología

Primero, se importan los paquetes necesarios para la actividad y las configuraciones necesarias para ajustar el tamaño y estilo de los gráficos

Actividad9.py

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

Después, se leen los datos desde un archivo csv con pandas para procesarlo como un dataframe y eliminan las columnas que no son de interés

Actividad9.py

```python
data = pd.read_csv('./articulos_ml.csv')
data.drop(['Title', 'url', 'Elapsed days'], axis=1)
```

Como primer parte de la actividad, solo hay que filtrar los datos para que se ajusten a la mayoría de la información que tenemos; se usan dichas condiciones debido a que previamente se analizó el conjunto de datos mediante histogramas y se concluyó que la mayoría de datos se concentraba dentro de dichos filtros, con esto eliminamos posibles anomalías según nuestros datos. Luego le asignamos colores a la mitad sobre la media de palabras y a la mitad debajo de dicha media

Actividad9.py

```
filtered_data = data[(data['Word count'] ⩽ 3500) & data['# Shares'] ⩽ 80000]
colors = ['orange', 'blue']
size = [30, 60]
f1 = filtered_data['Word count'].values
f2 = filtered_data['# Shares'].values
assign = []
for index, row in filtered_data.iterrows():
    if row['Word count'] > 1808:
        assign.append(colors[0])
    else:
        assign.append(colors[1])
```

Continuando con la actividad, creamos un nuevo dataframe con nuestras dos etiquetas de interés: conteo de palabras y suma de enlaces, comentarios e imágenes o videos. Posteriormente creamos un modelo de regresión lineal múltiple que se ajuste a ambas etiquetas y nuestra salida (el número de compartidos) e imprimimos su información: los coeficientes lineales, el error cuadrado promedio como medida de eficacia y el puntaje de varianza.

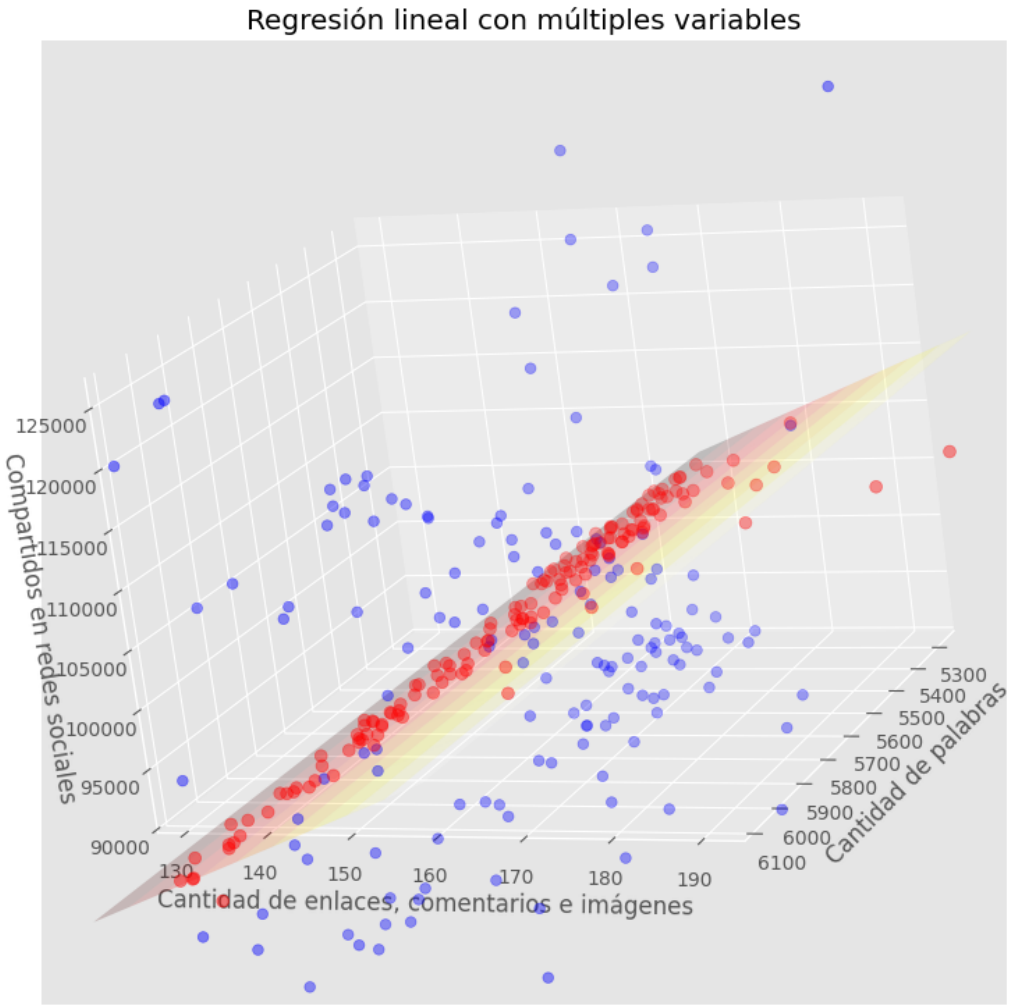
Actividad9.py

```
col_sum = (filtered_data['# of Links'] + filtered_data['# of comments'].fillna(0) +
 filtered_data['# Images video'])
dataX2 = pd.DataFrame()
dataX2['Word count'] = filtered_data['Word count']
dataX2['suma'] = col_sum
XY_train = np.array(dataX2)
z_train = filtered_data['# Shares'].values
regr2 = linear_model.LinearRegression()
regr2.fit(XY_train, z_train)
z_pred = regr2.predict(XY_train)
print(f'Coefficients: {regr2.coef_}. MSE: {round(mean_squared_error(z_train, z_pred), 2)}.
 Variance score: {round(r2_score(z_train, z_pred), 2)}.')
```

Finalmente, solo queda crear la gráfica en 3d; para ello, creamos una figura y una malla sobre la que representaremos nuestros datos, luego ajustamos nuestros vectores.

Actividad9.py

```
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
xx, yy = np.meshgrid( *xi: np.linspace( start: 0,  stop: 3500, num=10), np.linspace( start: 0,
  stop: 60, num=10))
nuevoX = (regr2.coef_[0] * xx)
nuevoY = (regr2.coef_[1] * yy)
z = (nuevoX + nuevoY + regr2.intercept_)
ax.plot_surface(xx, yy, z, alpha=0.2, cmap='hot')
ax.scatter(XY_train[:, 0], XY_train[:, 1], z_train, c='blue', s=30)
ax.scatter(XY_train[:, 0], XY_train[:, 1], z_pred, c='red', s=40)
ax.view_init(elev=30, azim=65)
ax.set_xlabel('Cantidad de palabras')
ax.set_ylabel('Cantidad de enlaces, comentarios e imágenes')
ax.set_zlabel('Compartidos en redes sociales')
ax.set_title('Regresión lineal con múltiples variables')
plt.show()
```



3 Resultados

A pesar de que según las métricas utilizadas sugieren que el modelo no es confiable, podemos hacer unas cuantas predicciones según la función generada

$$z = 21140.97 + 0.493x_1 + 289.3253x_2$$

Por ejemplo, si tenemos un artículo con 2000 palabras, 10 enlaces, 4 comentarios y 6 imágenes, deberiamos tener aproximadamente 27913 compartidos según la siguiente predicción:

$$z = 21140.97 + 0.493(2000) + 289.3253(10 + 4 + 6) \approx 27913$$

4 Conclusión

Este algoritmo es de gran utilidad para notar tendencias según nuestras características de interés, sin embargo, pierde mucho valor si no se ajusta bien a nuestros datos ya que puede llevarnos a predicciones erróneas. Es por esto que en mi opinión, la regresión lineal debería utilizarse sólo cuando se quiere ver la tendencia según ciertas características o si tenemos un conjunto demasiado grande de datos para evitar el underfitting.

Además, se puede ganar mucho valor de este algoritmo si se combina con alguna técnica de reducción de dimensionalidad como el SVD porque con ambos podemos comprender mejor el conjunto de datos que tenemos y nos puede mostrar mejor las tendencias usando menos etiquetas.