Regresión Lineal con Python

Jared Sandoval

March 22, 2025

1 Introducción

La regresión lineal es un método estadístico que permite modelar la relación entre una variable dependiente y una o más variables independientes. Se basa en encontrar la recta que mejor se ajusta a los datos, minimizando el error cuadrático medio.

2 Metodología

Para este análisis, se usó el lenguaje Python y bibliotecas como `pandas`, `numpy`, `seaborn` y `scikit-learn`. A continuación, se presentan los pasos seguidos:

2.1 Carga de datos

Se cargó el conjunto de datos y se verificó su estructura:

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score

data = pd.read_csv("./articulos_ml.csv")
data.shape
data.head()
data.describe()
```

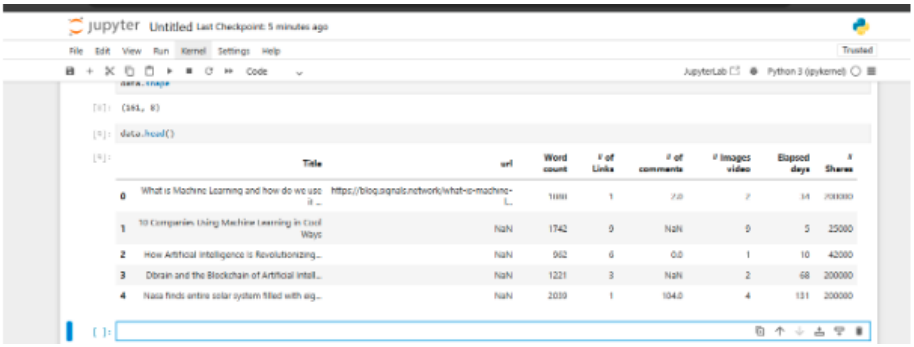

Figure 1: data.shape y data.head

Figure 2: Data.describe

2.2 Exploración de los datos

Se eliminaron columnas irrelevantes y se generó un histograma:

```
data.drop(['Title','url', 'Elapsed days'], axis=1).hist()
plt.show()
```

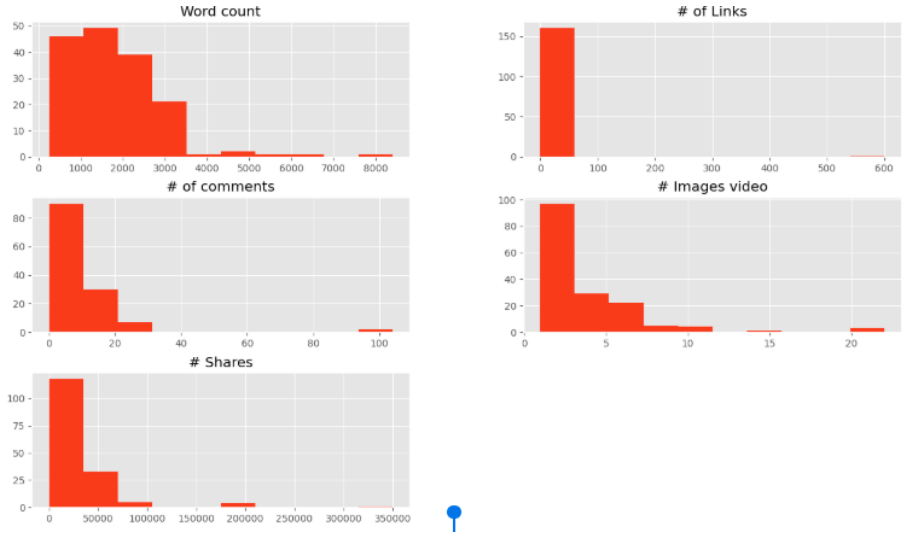

Figure 3: Histograma de los datos después de la limpieza

2.3 Filtrado de datos

Se filtraron los datos para reducir la variabilidad:

```
filtered_data = data[(data['Word count'] <= 3500) & (data['# Shares'] <= 80000)]
```

Se asignaron colores para la visualización:

```
colores=['orange','blue']
tamanios=[30,60]

f1 = filtered_data['Word count'].values
f2 = filtered_data['# Shares'].values

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()
```

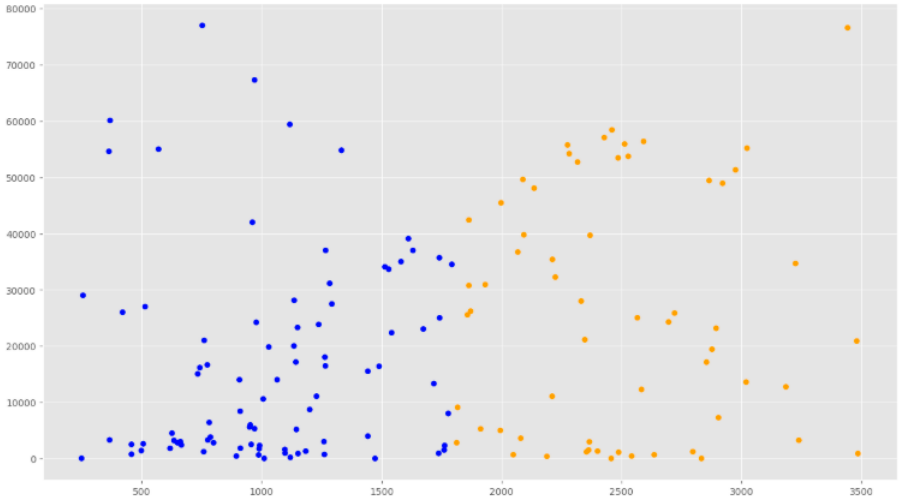

Figure 4: Scatterplot

2.4 Entrenamiento del modelo

Se prepararon las variables de entrada y salida y se entrenó el modelo de regresión lineal:

```
dataX = filtered_data[["Word count"]]
X_train = np.array(dataX)
y_train = filtered_data['# Shares'].values

regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)
```

2.5 Evaluación del modelo

Se calcularon los coeficientes, el error cuadrático medio y la varianza:

```
print('Coefficients: \n', regr.coef_)
print('Independent term: \n', regr.intercept_)
print("Mean squared error: %.2f" % mean_squared_error(y_train, y_pred))
print('Variance score: %.2f' % r2_score(y_train, y_pred))
```

```
Coefficients:
 [5.69765366]
Independent term:
 11200.30322307416
Mean squared error: 372888728.34
Variance score: 0.06
```

Figure 5: Coeficientes

2.6 Visualización de la recta de regresión

```
plt.scatter(X_train, y_train, color='blue', label='Datos reales')
plt.plot(X_train, y_pred, color='red', linewidth=2, label='Recta de regresión')
plt.xlabel("Word Count")
plt.ylabel("# Shares")
plt.title("Regresión Lineal - Word Count vs. Shares")
plt.legend()
plt.show()
```

[Espacio para imagen de la recta de regresión]

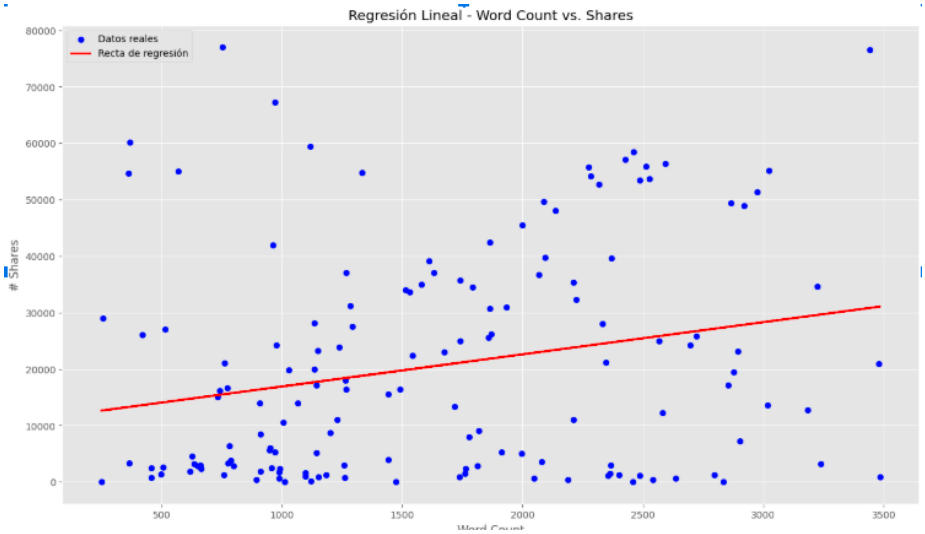

Figure 6: Regresion Lineal

3 Resultados

Se obtuvo la siguiente ecuación de regresión:

$$y = mX + b \tag{1}$$

Donde:

- Pendiente (m): `5.69765366`

- Intersección (b): `11200.303223074163`
- Error cuadrático medio: `372888728.34`
- Puntaje de varianza: `0.06`

4 Conclusión

Se observó que el modelo de regresión lineal ajustado no es muy bueno, ya que el error cuadrático medio es muy grande y el puntaje de varianza es bajo. Esto indica que la variable independiente "Word Count" no explica bien la variabilidad en la cantidad de "Shares".

Sin embargo, la implementación permitió entender los conceptos básicos de la regresión lineal, como se realizan usando bases de datos, y su aplicación en Python.