Regresión Lineal

Alain Lobato

31 de Marzo del 2025

1 Introducción

La regresión lineal es un método utilizado en el análisis de datos que permite predecir valores desconocidos en función de otros valores conocidos que están relacionados. Este método establece una ecuación lineal:

$$Y = mx + b \tag{1}$$

que representa la relación entre la variable dependiente (desconocida) y la variable independiente (conocida). Se centra en representar gráficamente la relación entre dos variables: x (variable independiente) y y (variable dependiente).

En el ámbito del Machine Learning, los algoritmos analizan grandes volúmenes de datos y determinan la ecuación de regresión lineal a partir de estos.

2 Metodología

Primero, clonamos nuestro repositorio de la materia en nuestra máquina y creamos una carpeta llamada "Linear Regression". Descargamos el archivo CSV para extraer los datos y creamos un archivo llamado `linear_regression.py` donde desarrollamos nuestro código usando Visual Studio Code.

Iniciamos importando las librerías necesarias, instalando las dependencias en caso de ser necesario:

```python
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
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

Figure 1: Importación de librerías

Ahora leemos el archivo CSV e imprimimos el tamaño de nuestra tabla, los primeros 5 registros y realizamos una descripción de los datos:

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

print(data.shape)
print(data.head())
print(data.describe())
```

Figure 2: Lectura del archivo CSV

```
[5 rows x 8 columns]
       Word count  # of Links  # of comments  # Images video  Elapsed days      # Shares
count   161.000000  161.000000     129.000000      161.000000    161.000000     161.000000
mean   1808.260870    9.739130       8.782946        3.670807     98.124224   27948.347826
std    1141.919385   47.271625      13.142822        3.418290    114.337535   43408.006839
min     250.000000    0.000000       0.000000        1.000000      1.000000       0.000000
25%     990.000000    3.000000       2.000000        1.000000     31.000000    2800.000000
50%    1674.000000    5.000000       6.000000        3.000000     62.000000   16458.000000
75%    2369.000000    7.000000      12.000000        5.000000    124.000000   35691.000000
max    8401.000000  600.000000     104.000000       22.000000   1002.000000  350000.000000
```

Figure 3: Descripción del DataFrame

Ahora eliminamos algunas variables irrelevantes y mostramos un histograma de los datos:

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

Figure 4: Eliminación de variables innecesarias

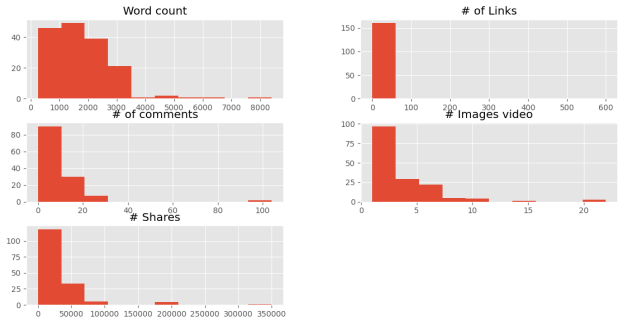

Figure 5: Histogramas generados

Luego, filtramos la data considerando solo registros con menos de 3500 palabras y 80000 compartidos. Pintamos de azul los puntos con menos de 1808 palabras y de naranja los que superan esta cantidad:

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

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 6: Filtrado de datos

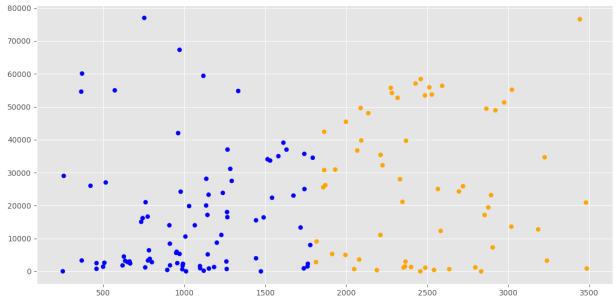

Figure 7: Gráfico de distribución de datos

Ahora, con `sklearn`, implementamos regresión lineal para predecir valores, entrenando primero nuestro modelo e imprimiendo los coeficientes obtenidos:

```
dataX =filtered_data[["Word count"]]
X_train = np.array(dataX)
y_train = filtered_data['# Shares'].values
 # Creamos la instancia de Regresión Linear
regr = linear_model.LinearRegression()
# Entrenamos el modelo
regr.fit(X_train, y_train)
y_pred = regr.predict(X_train)
# Coeficientes obtenidos
print('Coefficients: \n', regr.coef_)
# valor donde corta el eje Y
print('Independent term: \n', regr.intercept_)
# Error Cuadrado Medio
print("Mean squared error: %.2f" % mean_squared_error(y_train, y_pred))
# Puntaje de Varianza
print('Variance score: %.2f' % r2_score(y_train, y_pred))
```

Figure 8: Entrenamiento del modelo

Obtenemos los siguientes coeficientes:

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

Figure 9: Coeficientes obtenidos

Lo que nos da la siguiente función lineal:

$$y = 11200x + 5.69 \tag{2}$$

3 Resultados

Ahora que tenemos nuestro modelo, realizamos una predicción:

```
#Predicción en regresión lineal simple
y_Dosmil = regr.predict([[2000]])
print('Prediccion: ',int(y_Dosmil[0]))
```

Figure 10: Predicción con el modelo

El resultado obtenido es:

```
Prediccion:  22595
```

Figure 11: Resultado de la predicción

Podemos ver que al predecir para 2000 palabras, obtenemos que será compartido aproximadamente 22,595 veces.

4 Conclusión

El uso de la regresión lineal es bastante interesante. A pesar de ser un tema visto en estadística, es impresionante lo intuitivo que resulta cuando se aplica en programación. La facilidad con la que podemos implementar estos modelos nos permite analizar grandes volúmenes de datos de manera rápida y efectiva. Este ejercicio ayudó a entender mejor el funcionamiento del método y la utilidad que tiene en la investigación y desarrollo de distintas aplicaciones.

5 Referencias

Materiales de clase (2025). UANL.
Bagnato J. (2019). Aprende Machine Learning. Leanpub.