Regresión Lineal en Python

Carlos Oswaldo Gonzalez Garza

March 31, 2025

1 Introducción

La regresión lineal es un método estadístico utilizado para modelar la relación entre una variable dependiente y una o más variables independientes. La regresión lineal es un algoritmo de aprendizaje supervisado que se utiliza en Machine Learning y en estadística. En estadísticas, regresión lineal es una aproximación para modelar la relación entre una variable escalar dependiente "y" y una o mas variables explicativas nombradas con "X"

2 Metodología

Para este experimento, utilizamos Python y la librería scikit-learn para implementar un modelo de regresión lineal. Los pasos seguidos fueron:

1. Comenzamos importando las librerías que vamos a usar.

```
# Imports necesarios
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
plt.show()
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
```

2. Leemos el archivo csv y lo cargamos como un dataset de Pandas y vemos su tamaño.

```
#cargamos los datos de entrada
```

```
data = pd.read_csv("./articulos_ml.csv")
#veamos cuantas dimensiones y registros contiene
data.shape
#son 161 registros con 8 columnas. Veamos los primeros registros
data.head()
# Ahora veamos algunas estadísticas de nuestros datos
data.describe()
```

3. Intentaremos ver con nuestra relación lineal, si hay una correlación entre la cantidad de palabras del texto y la cantidad de Shares obtenidos.

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

4. Vamos a filtrar los datos de cantidad de palabras para quedarnos con los registros con menos de 3500 palabras y también con los que tengan Cantidad de compartidos menos a 80.000. Lo gratificaremos pintando en azul los puntos con menos de 1808 palabras y en naranja los que tengan más.

```
# Vamos a RECORTAR los datos en la zona donde se concentran más los puntos
# 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'] <= 80000)

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 media de C
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()
```

5. Vamos a crear nuestros datos de entrada por el momento sólo Word Count y como etiquetas los Shares. Creamos el objeto LinearRegression y lo hacemos encajar con el método fit(). Finalmente imprimimos los coeficientes y puntajes obtenidos.

```
# Asignamos nuestra variable de entrada X para entrenamiento y las etiquetas Y.
dataX =filtered_data[["Word count"]]
X_train = np.array(dataX)
y_train = filtered_data['# Shares'].values

# Creamos el objeto de Regresión Linear
regr = linear_model.LinearRegression()

# Entrenamos nuestro modelo
regr.fit(X_train, y_train)
# Hacemos las predicciones que en definitiva una línea (en este caso, al ser 2D
y_pred = regr.predict(X_train)

# Veamos los coeficienetes obtenidos, En nuestro caso, serán la Tangente
print('Coefficients: \n', regr.coef_)
# Este es el valor donde corta el eje Y (en X=0)
print('Independent term: \n', regr.intercept_)
# Error Cuadrado Medio
print("Mean squared error: %.2f" % mean_squared_error(y_train, y_pred))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('Variance score: %.2f' % r2_score(y_train, y_pred))
```

6. Por ultimop vamos a intentar probar nuestro algoritmo, suponiendo que quisiéramos predecir cuántos "compartir" obtendrá un articulo sobre ML de 2000 palabras

```
#Vamos a comprobar:
# Quiero predecir cuántos "Shares" voy a obtener por un artículo con 2.000 pala
# según nuestro modelo, hacemos:
y_Dosmil = regr.predict([[2000]])
print(int(y_Dosmil))
```

El código utilizado fue:

```
# Imports necesarios
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
```

```
plt.show()
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
#cargamos los datos de entrada
data = pd.read_csv("./articulos_ml.csv")
#veamos cuantas dimensiones y registros contiene
data.shape
#son 161 registros con 8 columnas. Veamos los primeros registros
data.head()
# Ahora veamos algunas estadísticas de nuestros datos
data.describe()
# Visualizamos rápidamente las caraterísticas de entrada
data.drop(['Title', 'url', 'Elapsed days'], axis=1).hist()
plt.show()
# Vamos a RECORTAR los datos en la zona donde se concentran más los puntos
# 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'] <= 80000)]

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 media de Cantidad c
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()
# Asignamos nuestra variable de entrada X para entrenamiento y las etiquetas Y.
dataX =filtered_data[["Word count"]]
X_train = np.array(dataX)
y_train = filtered_data['# Shares'].values

# Creamos el objeto de Regresión Linear
regr = linear_model.LinearRegression()
```

```
# Entrenamos nuestro modelo
regr.fit(X_train, y_train)
# Hacemos las predicciones que en definitiva una línea (en este caso, al ser 2D)
y_pred = regr.predict(X_train)

# Veamos los coeficienetes obtenidos, En nuestro caso, serán la Tangente
print('Coefficients: \n', regr.coef_)
# Este es el valor donde corta el eje Y (en X=0)
print('Independent term: \n', regr.intercept_)
# Error Cuadrado Medio
print("Mean squared error: %.2f" % mean_squared_error(y_train, y_pred))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('Variance score: %.2f' % r2_score(y_train, y_pred))
#Vamos a comprobar:
# Quiero predecir cuántos "Shares" voy a obtener por un artículo con 2.000 palabras,
# según nuestro modelo, hacemos:
y_Dosmil = regr.predict([[2000]])
print(int(y_Dosmil))
```

3 Resultados

Los coeficientes obtenidos fueron:

```
Coefficients: [5.69765366]
Independent term: 11200.303223074163
Mean squared error: 372888728.34
Variance score: 0.06
```

4 Conclusión

Implementamos un modelo de regresión lineal para analizar la relación entre la cantidad de palabras en un artículo y sus compartidos. Aunque se encontró una relación positiva, el bajo puntaje de varianza indica que el modelo no explica bien los datos. Esto sugiere que otros factores influyen en la cantidad de compartidos.