Regresión Lineal Multiple en Python

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 esta conformado por una sola variable independiene 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

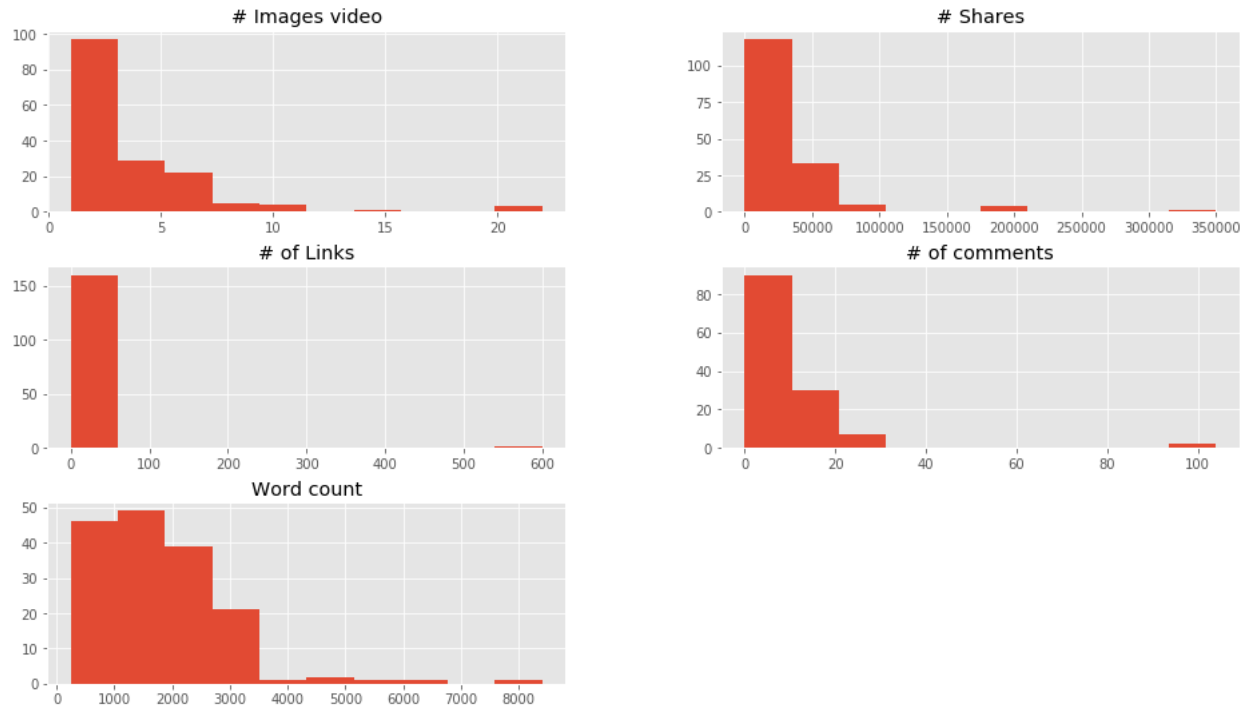
2 Metodología

Para realizar el ejercicio de regresión lineal múltiple, tuve que seguir los siguientes pasos: Visualización General de los datos de Entrada, Filtración de los datos, División de datos en conjuntos de entrenamiento, Ajuste del modelo y la Evaluación del Modelo.

```
#Importamos las librerias
import numpy as np
import pandas as pd
import seaborn as sns
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

#Creamos un dataframe
data = pd.read_csv("./articulos_ml.csv")
```

2.1 Visualización General de los datos de Entrada

En esta sección quitamos las variables categoricas del DataFrame y visualizamos la distribución de los datos.

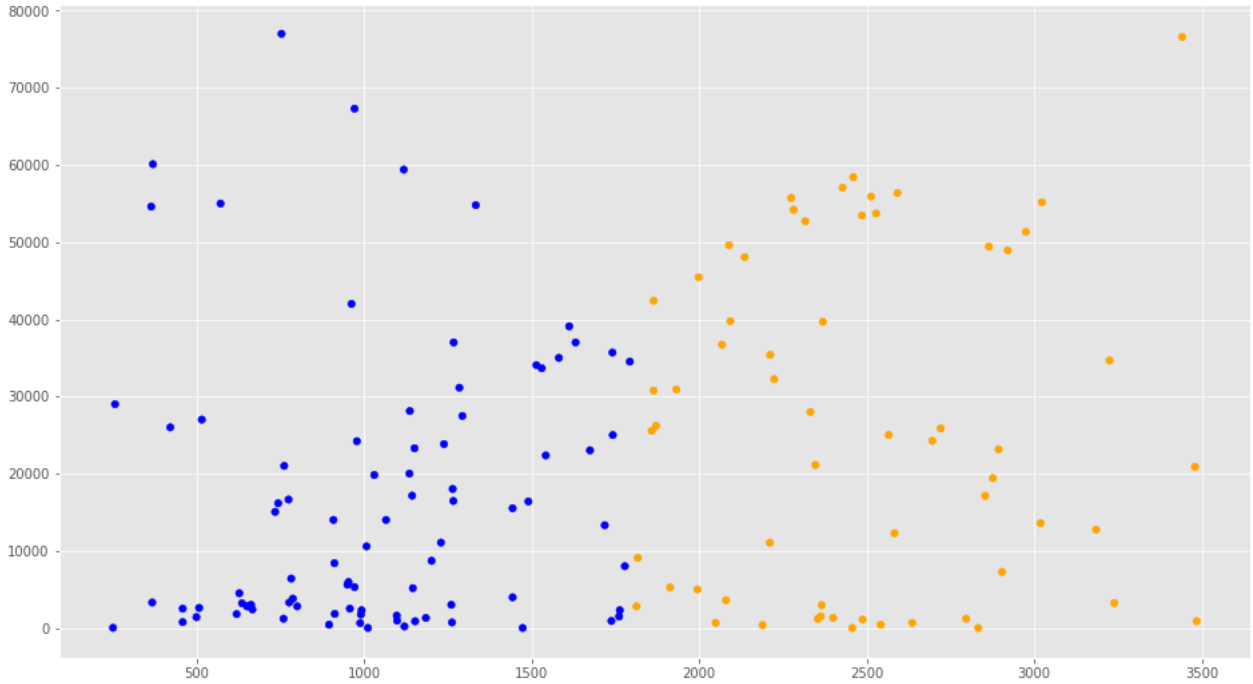


En este ejercicio, se escogió que la variable dependiente del modelo de regresión lineal simple fuera el número de compartidos, mientras que la variable regresora fuera la cantidad de palabras.

2.2 Filtración de los datos

Ya que escogimos nuestras variables dependiente e independiente, visualizamos su relación con una gráfica *scatterplot*. También establecemos un límite superior al valor de los datos, ya que como se puede apreciar en las gráficas anteriores la gran mayoría de los datos de estas dos variables se concentran en los valores más pequeños, aunque si tienen algunos datos que no siguen la distribución. Para evitar que estas anomalías influyan en el entrenamiento de nuestro modelo, no las vamos a contar en nuestro *dataset*.

```
#Recortamos los datos en la zona donde se encuentran más los puntos
filtered_data = data[(data['Word count'] <= 3500) & (data['# Shares'] <= 80000)]

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

# Vamos a pintar con distintos colores los puntos en ambos lados
#de la media de Cantidad de Palabras
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()
```



Como se puede ver, no existe una clara relación lineal entre estas dos variables, por lo que agregamos una variable regresora adicional y asi crear un modelo de regresión lineal multiple.

2.3 División de datos en conjuntos de entrenamiento

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

```
#Vamos a intentar mejorar el Modelo, con una dimensión más:
# Para poder graficar en 3D, haremos una variable nueva que será
#la suma de los enlaces, comentarios e imágenes
suma = (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"] = suma
XY_train = np.array(dataX2)
z_train = filtered_data['# Shares'].values
```

2.4 Ajuste del modelo

En esta parte, se crea el modelo de regresión lineal multiple usando `scikit-learn` y se ajusta usando los datos de entrenamiento que ya creamos. Además sacamos las predicciones de Y del modelo, las cuales deberían de ser más precisas que las del modelo simple.

```
# Creamos el objeto de Regresión Linear
regr2 = linear_model.LinearRegression()

# Entrenamos nuestro modelo
regr2.fit(XY_train, z_train)

# Hacemos las predicciones de la Y, que en nuestra gráfica 3D
# será representada con el eje z
z_pred = regr2.predict(XY_train)
```

2.5 Evaluación del modelo

Ya que tenemos el modelo creado y ajustado, podemos imprimir los valores del intercepto, la tangente, el error cuadrado medio y la varianza.

```
# Veamos los coeficientes obtenidos, En nuestro caso, serán la Tangente
print('Coefficients: \n', regr2.coef_)
# Error Cuadrado Medio
print("Mean squared error: %.2f" % mean_squared_error(z_train, z_pred))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('Variance score: %.2f' % r2_score(z_train, z_pred))
```

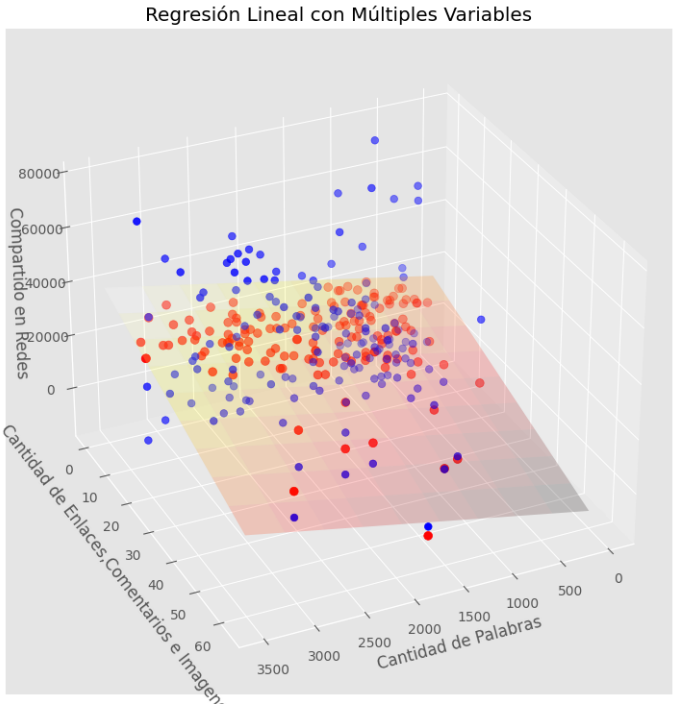
```
Coefficients:
 [   6.63216324 -483.40753769]
Mean squared error: 352122816.48
Variance score: 0.11
```

Hay que tomar en cuenta que el valor del error medio cuadrado es bastante grande aunque es ligeramente menor al error dado por el modelo lineal simple, por lo que podemos afirmar que un modelo de regresión lineal, ya sea simple o multiple, no es la mejor opción para explicar la relación entre los datos del *dataset*.

3 Resultados

Para visualizar los resultados, vamos a crear una gráfica 3D similar al *scatterplot* del modelo simple, para ver como se comportan los datos en este modelo. También podemos evaluar su desempeño checando las predicciones que hace con los mismos valores que se usaron para checar la precisión del modelo lineal simple.

```
fig = plt.figure()
ax = Axes3D(fig)

# 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. Debemos sumar el punto de intercepción
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)

# con esto situamos la "camara" con la que visualizamos
ax.view_init(elev=30., azim=65)

ax.set_xlabel('Cantidad de Palabras')
ax.set_ylabel('Cantidad de Enlaces,Comentarios e Imagenes')
ax.set_zlabel('Compartido en Redes')
ax.set_title('Regresión Lineal con Múltiples Variables')
```



Para evaluar el desempeño del modelo, vamos a checar su predicción del número de compartidos en redes cuando hay 2000 palabras en el *post* y una suma de 20 enlaces, comentarios e imagenes. En la gráfica 2D se puede apreciar que hay dos datos alrededor de las 2000 palabras: uno que esta por *y=5000* y otro que esta por *y=46000*. Estos dos valores son muy distintos, entonces verificamos que el modelo acierte en una de estas dos.

```
# Si quiero predecir cuántos "Shares" voy a obtener por un artículo con:
# 2000 palabras y con enlaces: 10, comentarios: 4, imagenes: 6
# según nuestro modelo, hacemos:

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

```
20518
```

El modelo no acerto a ninguno de los dos valores, pero su predicción esta a mediación de estos datos. Si volvemos a checar la gráfica podemos ver que la relación entre nuestras tres variables no parece que sea lineal, si fuera así, la mayoría de los datos estarían más acercados al plano. Para conseguir mejores resultados en un modelo de predicción para este *dataset*, sería mejor intentar con otros modelos de regresión como el logístico.

4 Conclusión

Los modelos de regresión lineal son herramientas estadísticas usadas para explicar la relación entre conjuntos de datos de manera lineal y de los cuales hay dos tipos: simple, que es cuando están compuestos por una variable dependiente y una variable independiente o regresora; y múltiple, que es cuando hay variables regresoras.

En este caso, se intentó comprobar que el número de palabras y la suma de comentarios, enlaces e imagenes del *post* tenía una relación lineal y causal con el número de compartidos. Al graficar los datos y entrenar el modelo, se puede ver que esto no es correcto. La forma de la gráfica no tiene una forma que indica una correlación entre las tres variables y el modelo refleja esto.

El modelo no acertó en su predicción a ninguno de los dos valores que se encuentran cerca de $x_1 = 2000$, pero su predicción está a mediación de estos datos. Para conseguir mejores resultados en un modelo de predicción para este *dataset*, sería mejor intentar con otros modelos de regresión como el logístico.