Regresión lineal – Ejercicio con Python

Requerimientos para realizar el ejercicio

- Jupyter notebook con código Python.
- Librería SkLearn.
- Suite de Anaconda.

Descripción

A partir de las características de un artículo de Machine Learning intentaremos predecir, cuantas veces será compartido en Redes Sociales.

En este ejemplo cargaremos un archivo .csv[1] de entrada obtenido por webscraping que contiene diversas URLs a artículos sobre Machine Learning de algunos sitios muy importantes como Techcrunch o KDnuggets y como características de entrada -las columnas- tendremos:

- **Title**: Titulo del Artículo
- **url**: ruta al artículo
- **Word count**: la cantidad de palabras del artículo,
- **# of Links**: los enlaces externos que contiene,
- **# of comments:** cantidad de comentarios,
- **# Images video:** suma de imágenes (o videos),
- **Elapsed days:** la cantidad de días transcurridos (al momento de crear el archivo)
- **# Shares:** nuestra columna de salida que será la "cantidad de veces que se compartió el artículo".

Se realiza una primer predicción de **regresión lineal simple** (con una sola variable predictora) para poder graficar en 2 dimensiones (ejes X e Y) y luego un ejemplo de regresión Lineal Múltiple, en la que utilizaremos 3 dimensiones (X,Y,Z) y predicciones.

Procedimiento

Regresión lineal simple en Python (con 1 variable)

1. Importar las librerías que se utilizarán

```
# Imports necesarios
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
```

[1] El archivo .csv contiene mitad de datos reales, y otra mitad se generaron de forma manera aleatoria, por lo que las predicciones que se obtienen no serán reales.

2. Se lee el archivo csv y se cargamos como un dataset de Pandas.

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

3. Y visualiza su tamaño

```
data.shape
```

El resultado que devuelve es

(161, 8)

Lo que indican estos números, es que el archivo contiene 168 registros con 8 columnas

4. Se imprimen las primeras 5 columnas

```
data.head()
```

Resultado esperado

	Title	url	Word count	# of Links	# of comments	# Images video	Elapsed days	# Shares
0	What is Machine Learning and how do we use it ...	https://blog.signals.network/what-is-machine-l...	1888	1	2.0	2	34	200000
1	10 Companies Using Machine Learning in Cool Ways	NaN	1742	9	NaN	9	5	25000
2	How Artificial Intelligence Is Revolutionizing...	NaN	962	6	0.0	1	10	42000
3	Dbrain and the Blockchain of Artificial Intell...	NaN	1221	3	NaN	2	68	200000
4	Nasa finds entire solar system filled with eig...	NaN	2039	1	104.0	4	131	200000

Puedes observar algunos campos con valores NaN (nulos) por ejemplo algunas urls o en comentarios.

5. En el siguiente paso, se obtendrán algunos resultados de estadística básica, tomando como referencia los datos de entrada.

```
data.describe()
```

Resultado obtenido

	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

Algunos de los resultados podemos describir son los siguientes:

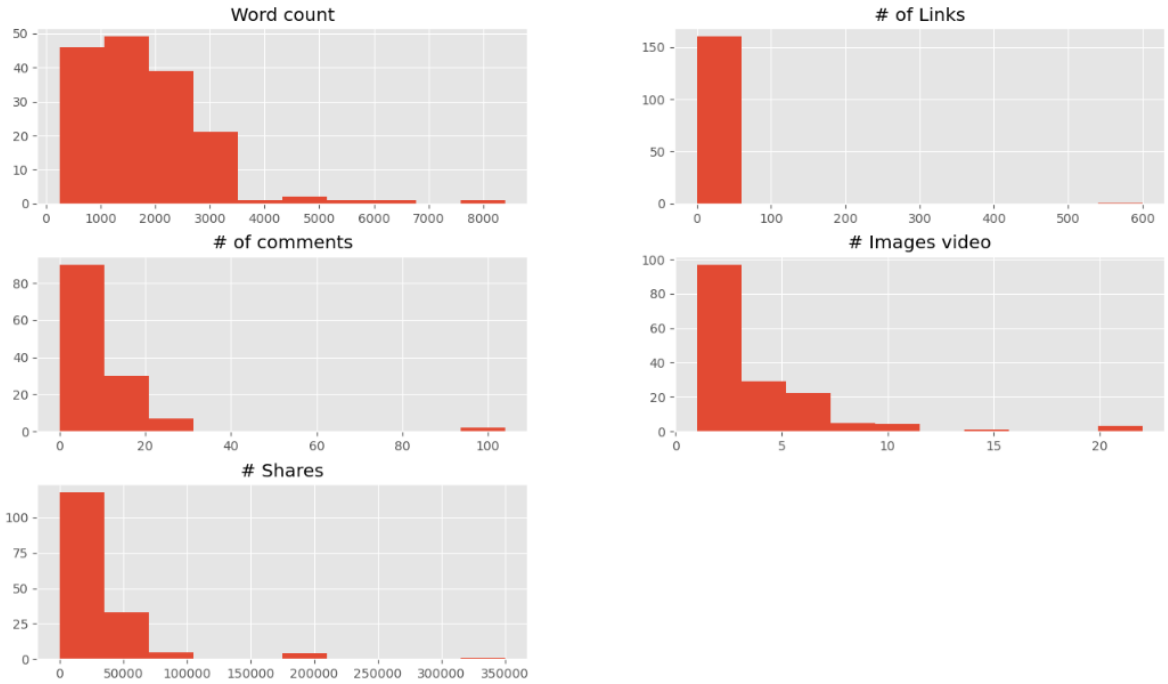
- La media de las palabras es de 1808.26.
- El artículo más corto tiene 250 palabras
- El artículo más extenso tiene 8401.

En este ejercicio se requiere conocer si existe una correlación entre la *cantidad de palabras* del texto y las cantidades de *Shares* obtenidos.

6. El siguiente bloque de código genera gráficas de los datos de entrada.

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

En las graficas podemos observar los valores en los que se concentran la mayoría de los registros.

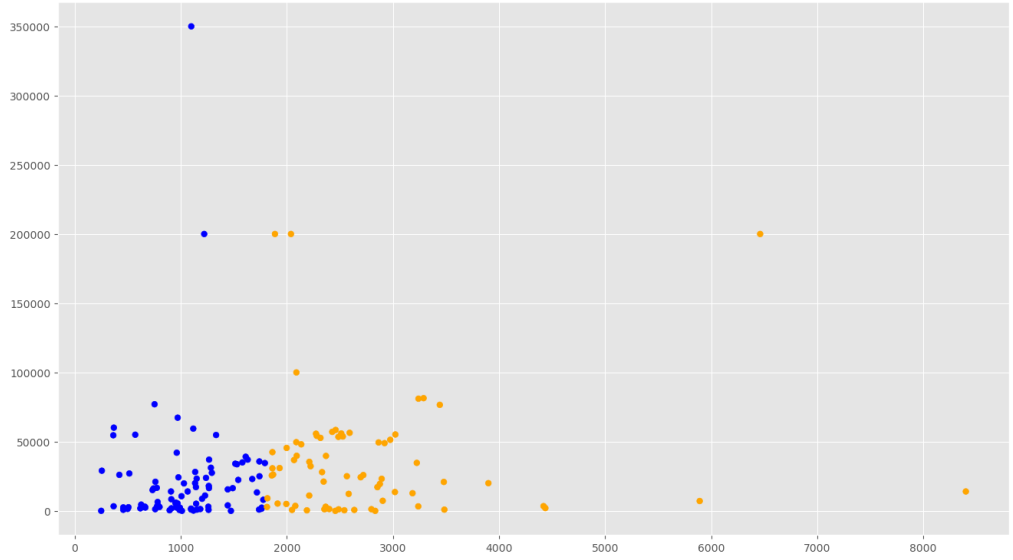


7. De todos los datos almacenados se filtrarán la cantidad de palabras para obtener los registros de menos de 3500 palabras y aquellos registros que tengan la cantidad de *Shares* menos a 80,000.

   Se graficará resaltando en color azul los puntos con menos de 1808 palabras, es decir, la media y en color naranja los que tengan más.

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

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

asignar=[]
for index, row in 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()
```

Salida esperada



8. Se realiza un acercamiento a 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)]

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 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()
```

9. Ahora se muestran como los valores cambian una vez filtrados, a continuación se muestra la tabla.

```
filtered_data.describe()
```

	Word count	# of Links	# of comments	# Images video	Elapsed days	# Shares
count	148.000000	148.000000	121.000000	148.000000	148.000000	148.000000
mean	1640.209459	5.743243	7.256198	3.331081	91.554054	20545.648649
std	821.975365	6.064418	6.346297	2.706476	91.143923	19933.865031
min	250.000000	0.000000	0.000000	1.000000	1.000000	0.000000
25%	971.000000	3.000000	2.000000	1.000000	28.750000	2750.000000
50%	1536.000000	5.000000	6.000000	3.000000	60.000000	15836.000000
75%	2335.750000	7.000000	11.000000	4.000000	110.500000	34177.500000
max	3485.000000	49.000000	30.000000	22.000000	349.000000	77000.000000

10. Regresión Lineal con Python y SKLearn

A continuación, se crean datos de entrada con *Word Count* y como etiquetas los *# Shares*.

Enel código se definen las variables de entrada X para el entrenamiento y las etiquetas Y.

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

En esta parte del código se crea el objeto *LinearRegression* y se realiza el entrenamiento con el método fit().

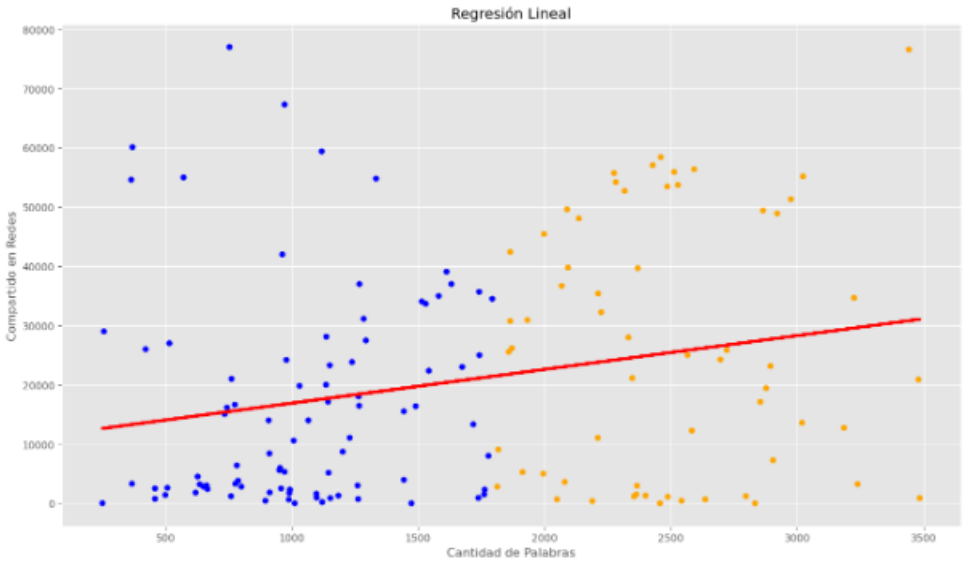
```
# Creamos el objeto de Regresión Linear
regr = linear_model.LinearRegression()

# 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))
```

Finalmente se imprimen los coeficientes y puntajes obtenidos.

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

11. La siguiente línea de código visualiza la recta obtenida.

```
plt.scatter(X_train[:,0], y_train,  c=asignar, s=tamanios[0])
plt.plot(X_train[:,0], y_pred, color='red', linewidth=3)

plt.xlabel('Cantidad de Palabras')
plt.ylabel('Compartido en Redes')
plt.title('Regresión Lineal')

plt.show()
```



12. En la siguiente línea se realiza la comprobación de los programado hasta ahora. Es decir, se va a predecir cuantos *Shares* voy a obtener por un artículo con 2000 palabras, según el modelo planteado.

```
y_Dosmil = regr.predict([[2000]])
print(int(y_Dosmil))
```

El resultado de esta predicción es el siguiente:

```
22595
```

13. Regresión Lineal Múltiple

El ejercicio que se realizó se va extender agregando más de una variable de entrada para el modelo.

Realizar esta modificación da mayor certeza al algoritmo de Machine Learning, pues de esta manera se logran obtener predicciones más complejas.

Ahora se utilizan dos variables predictivas, para poder graficar en 3D, aunque para mejorar la predicción se pueden utilizar más de dos entradas sin necesidad de utilizar gráficos.

La primer variable es *Word count*, la segunda variable es la suma de 3 columnas de entrada: *# of Links*, *# of comments* y *# Images video*.

```
# 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
```

Una vez teniendo las dos variables de entrada en **XY_train** y la variable de salida pasa de ser "Y" a ser el eje "Z".

Se crea un nuevo objeto de Regresión lineal con SKLearn pero esta vez tendrá las dos dimensiones que entrenar: las que contiene XY_train. Al igual que antes, se imprimen los coeficientes y puntajes obtenidos:

```
# Creamos un nuevo objeto de Regresión Lineal
regr2 = linear_model.LinearRegression()

# Entrenamos el modelo, esta vez, con 2 dimensiones
# obtendremos 2 coeficientes, para graficar un plano
regr2.fit(XY_train, z_train)

# Hacemos la predicción con la que tendremos puntos sobre el plano hallado
z_pred = regr2.predict(XY_train)

# Los coeficientes
print('Coefficients: \n', regr2.coef_)
# Error cuadrático medio
print("Mean squared error: %.2f" % mean_squared_error(z_train, z_pred))
# Evaluamos el puntaje de varianza (siendo 1.0 el mejor posible)
print('Variance score: %.2f' % r2_score(z_train, z_pred))
```

La salida esperada es la siguiente

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

Los resultados obtenidos son dos coeficientes, cada uno correspondiente a nuestras 2 variables predictivas, pues ahora lo que graficamos no será una línea, si no, un plano en 3D.

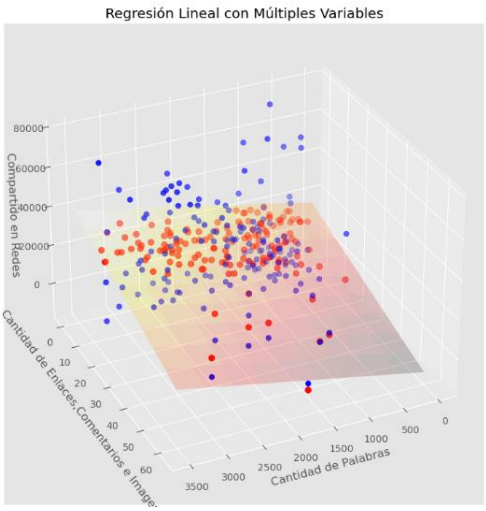
El error obtenido sigue siendo grande, aunque algo mejor que el anterior y el puntaje de Varianza mejora casi el doble del anterior, pero si se sigue lejos del 1 es malo.

14. Visualizar los resultados en una gráfica 3D

Se grafican los puntos de las características de entrada en color azul y los puntos proyectados en el plano en rojo. Recordemos que, en esta gráfica, el eje Z corresponde a la "altura" y representa la cantidad de Shares que se quieren predecir.

```
fig = plt.figure()
#ax = Axes3D(fig)
ax = fig.add_subplot(111, projection='3d')

# 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')

plt.show()
```

El resultado esperado es el siguiente:



Si se desea predecir cuántos *Shares* se obtienen por un artículo con 2000, con 10 enlaces, 4 comentarios y 6 imágenes, la línea de código a utilizar es la siguiente.

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

Resultado obtenido:

```
20518
```

Para restar los errores que se generan durante los cálculos, se puede agregar las siguientes líneas de código.

```
mejoraEnError = mean_squared_error(y_train, y_pred) - mean_squared_error(z_train, z_pred)
print(mejoraEnError)
```

```
20765911.860715985
```

También calculamos la mejora de la varianza

```
mejoraEnVarianza = r2_score(z_train, z_pred) - r2_score(y_train, y_pred)
print(mejoraEnVarianza)
```

```
0.052615337462582956
```

Aunque no parezca mucho, recordemos que el valor más alto que se puede obtener es 1.0

Finalmente, se realiza una mejora en la predicción de un artículo de 2.000 palabras, pues aunque disminuyen los "Shares" que se obtienen en el segundo modelo, lo más probables es que es un valor más cercano a la realidad.

```
diferenciaComparir = z_Dosmil - y_Dosmil
print(int(diferenciaComparir))
```

Referencia:
https://www.aprendemachinelearning.com/regresion-lineal-en-espanol-con-python/#more-5722