Ejercicios de regresión lineal - Artículos de ML

July 9, 2022

[3]:
```
#Empezar por los imports
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
```

[4]:
```
#Cargar mi dataset
data = pd.read_csv("./articulos_ml.csv")
print(data.shape)
```

```
(161, 8)
```

[5]:
```
#Cargar las primeras 5 renglones
data.head()
```

[5]:

	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				
3	Dbrain and the Blockchain of Artificial Intell…	NaN	1221	3				
4	Nasa finds entire solar system filled with eig…	NaN	2039	1				

```
2            0.0               1            10     42000
3            NaN               2            68    200000
4          104.0               4           131    200000
```

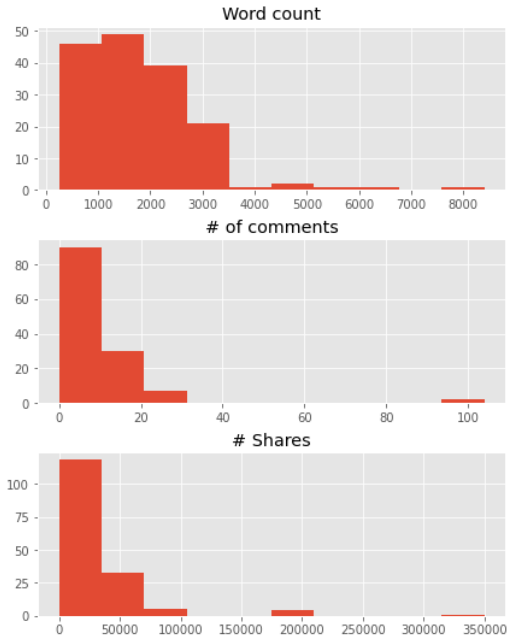
[6]:
```
#Estadística descriptiva de mis columnas números
data.describe()
```

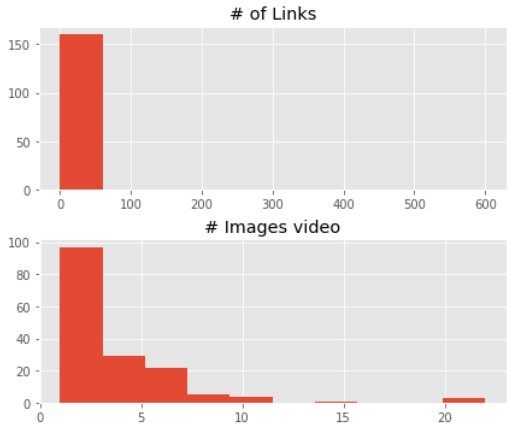
[6]:
```
         Word count  # of Links  # of comments  # Images video  Elapsed days  \
count    161.000000  161.000000     129.000000      161.000000    161.000000
mean    1808.260870    9.739130       8.782946        3.670807     98.124224
std     1141.919385   47.271625      13.142822        3.418290    114.337535
min      250.000000    0.000000       0.000000        1.000000      1.000000
25%      990.000000    3.000000       2.000000        1.000000     31.000000
50%     1674.000000    5.000000       6.000000        3.000000     62.000000
75%     2369.000000    7.000000      12.000000        5.000000    124.000000
max     8401.000000  600.000000     104.000000       22.000000   1002.000000

            # Shares
count     161.000000
mean    27948.347826
std     43408.006839
min         0.000000
25%      2800.000000
50%     16458.000000
75%     35691.000000
max    350000.000000
```

[8]:
```
#Visualización de datos de entrada
data.drop(['Title','url', 'Elapsed days'],1).hist()
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/ipykernel_launcher.py:2: FutureWarning: In a future version of pandas
all arguments of DataFrame.drop except for the argument 'labels' will be
keyword-only
```

[8]:
```
array([[<AxesSubplot:title={'center':'Word count'}>,
        <AxesSubplot:title={'center':'# of Links'}>],
       [<AxesSubplot:title={'center':'# of comments'}>,
        <AxesSubplot:title={'center':'# Images video'}>],
       [<AxesSubplot:title={'center':'# Shares'}>, <AxesSubplot:>]],
      dtype=object)
```





[9]:
```
#Filtrando los datos menor de 3500 en Word count y menor de 80000 en Shares
filtered_data = data[(data['Word count'] <= 3500) & (data['# Shares'] <= 80000)]
```

[10]:
```
filtered_data.head()
```

[10]:

	Title	url	Word count	# of Links	# of comments	# Images video	Elapsed days	# Shares
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
5	5 ways Data Science and Machine Learning impac…	NaN	761	0	NaN	1	14	21000
7	How Machine Learning can help Cryptocurrency T…	https://cryptovest.com/news/how-machine-learni…	753	3	0.0	1	78	77000
8	Tech companies should stop pretending AI won’t…	https://www.technologyreview.com/s/610298/tech…	1118	2	NaN	1	62	59400

[11]:
```
filtered_data.shape
```

[11]:
```
(148, 8)
```

[12]:
```
filtered_data.describe()
```

[12]:

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

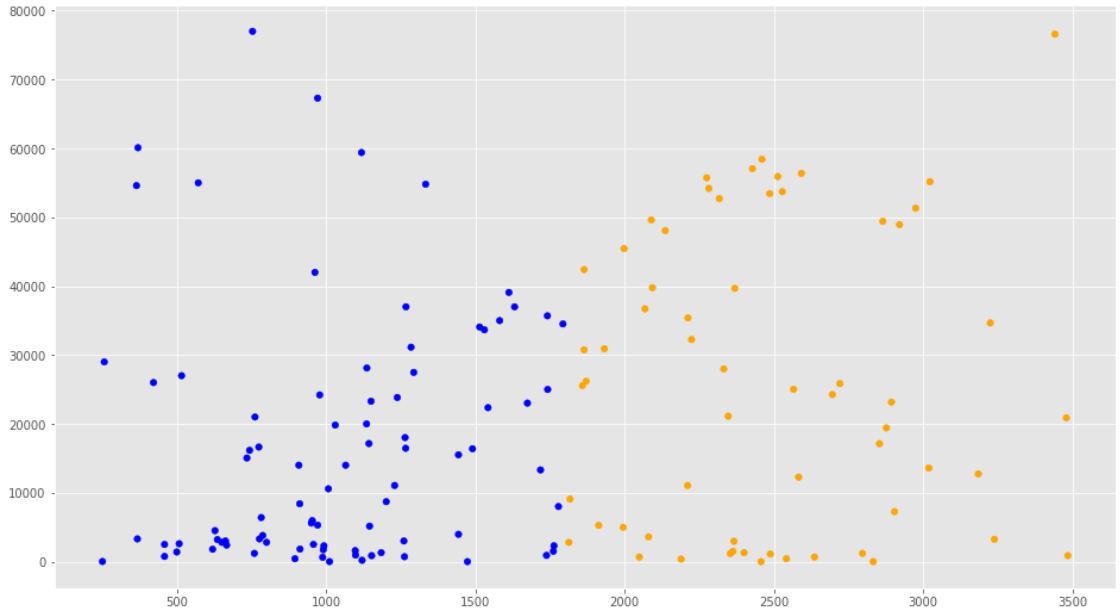
	# Shares
count	148.000000
mean	20545.648649
std	19933.865031
min	0.000000
25%	2750.000000
50%	15836.000000
75%	34177.500000
max	77000.000000

[13]:
```
#Variables para la gráfica:
colores=['orange','blue']
tamanios=[30,60]
```

[14]:
```
f1 = filtered_data['Word count'].values
f2 = filtered_data['# Shares'].values
```

[15]:
```
#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])
```

[15]:
```
<matplotlib.collections.PathCollection at 0x7f91ff70d668>
```



[16]:
```
#Vamos a crear nuestros datos de entrada por el momento sólo Word Count y como␣
↪etiquetas los # Shares.
dataX = filtered_data[["Word count"]]
X_train = np.array(dataX)
y_train = filtered_data['# Shares'].values
```

[17]:
```
X_train[:5]
```

[17]:
```
array([[1742],
       [ 962],
       [ 761],
       [ 753],
       [1118]])
```

[18]:
```
#Creacion del objeto
regresion = linear_model.LinearRegression()
```

[19]:
```
#Entreno el modelo
regresion.fit(X_train, y_train)
```

[19]:
```
LinearRegression()
```

[21]:
```
#Realizar predicciones
y_pred = regresion.predict(X_train)
```

[22]:
```
y_pred[:5]
```

[22]:
```
array([21125.61589425, 16681.44604148, 15536.21765635, 15490.63642709,
       17570.28001204])
```

[26]:
```
y_train[:5]
```

[26]:
```
array([25000, 42000, 21000, 77000, 59400])
```

[23]:
```
print('Coefficients: \n', regresion.coef_)
```

```
Coefficients:
 [5.69765366]
```

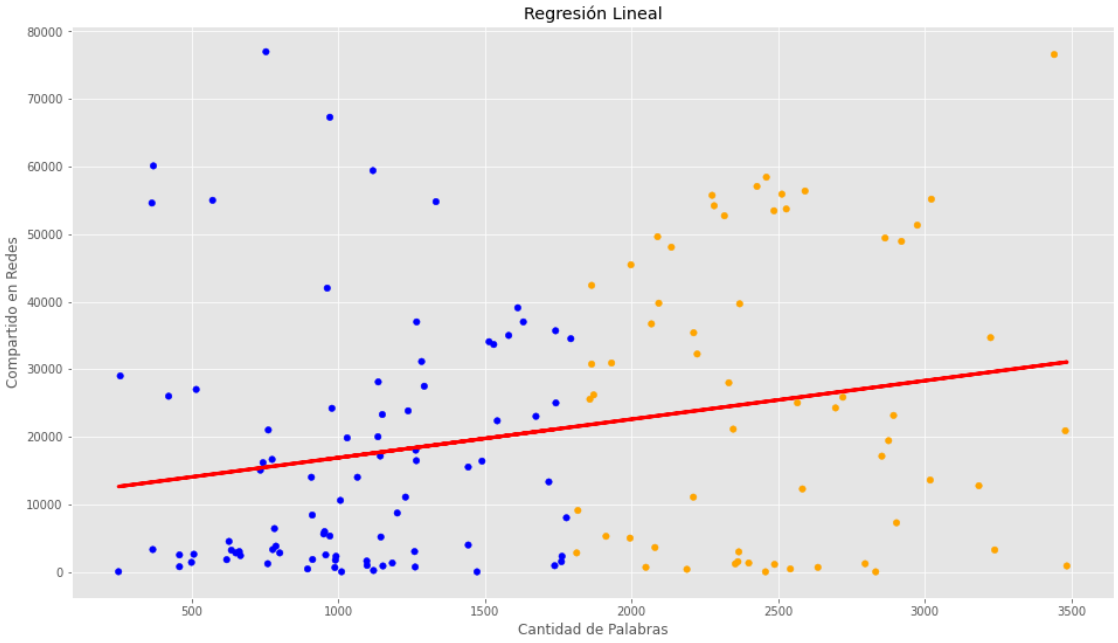
[25]:
```
print('Independent term: \n', regresion.intercept_)
```

```
Independent term:
 11200.30322307416
```

[27]:
```
#Pendiente:
#Coefficients: [5.69765366]
#Intersección de la recta:
#Independent term: 11200.303223074163
```

[28]:
```
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')
```

[28]:
```
Text(0.5, 1.0, 'Regresión Lineal')
```



[30]:
```
#Realizando una predicción para un articulo de 2000 palabras
y_Dosmil = regresion.predict([[2000]])
print(int(y_Dosmil))
```

```
22595
```

[31]:
```
#CODIGO PARA EL DE LAS VENTAS EJEMPLO EN CLASE
```

[32]:
```
X_train = np.array([[1],[2],[3],[4],[5],[6]])
y_train = np.array([7000,9000,5000,11000,10000,13000])
```

[33]:
```
#Creacion del objeto
regre2 = linear_model.LinearRegression()
```

[34]:
```
#Entreno el modelo
regre2.fit(X_train, y_train)
```

[34]:
```
LinearRegression()
```

[35]:
```
y_pred = regre2.predict(X_train)
```

[36]:
```
y_pred
```

[36]:
```
array([ 6380.95238095,  7495.23809524,  8609.52380952,  9723.80952381,
       10838.0952381 , 11952.38095238])
```

[37]:
```
regre2.predict([[7]])
```

[37]:
```
array([13066.66666667])
```

[39]:
```
#La pendiente
print('Coefficients: \n', regre2.coef_)
```

```
Coefficients:
 [1114.28571429]
```

[41]:
```
#Intersección de la linea
print('Independent term: \n', regre2.intercept_)
```

```
Independent term:
 5266.666666666666
```

[ ]:
```

```