Regresión lineal múltiple para predicción de ventas de una compañía que invierte en publicidad.

Trejo Sergio *

* *Centro de Investigaciones en Optica, Leon, Gto (e-mail: sergiotrejo@cio.mx).*

Abstract: En este artículo se presenta un modelo de regresión lineal multivariable para la predicción de ventas de una compañía. Utilizamos la base de datos de gastos en publicidad por tres medios: televisión, radio y periódico, y 200 observaciones que utilizamos como entrada. El modelo alcanzó un $r^2 = 0.8972$.

Keywords: Regresión Lineal, Insurance cost, Predicciones.

1. INTRODUCCIÓN

En el mundo actual muchas personas requieren de un seguro médico como un medio de seguridad ante cualquier amenaza que pueda afectar a su salud, pero en Estados Unidos por parte tanto de empresas de seguro privado se han desarrollado métodos para calcular el costo de los seguros médicos de las personas ya que depende de cierta demografía predecir si una persona u otra utilizará servicios hospitalarios.

Uno de los mayores predictores es si una persona fuma ya que en algún momento desarrollará cáncer de pulmón inminentemente además si una persona tiene obesidad es muy probable que desarrolle enfermedades del corazón. En análisis cualitativos realizados se encontró que los seguros varían mucho del porcentaje de ingreso anual de la persona por lo que una persona de 50 años, con un ingreso de $500,000 dls anuales, fumador y en un plan plata es de $42,500 dls ya que el máximo que paga una persona es de 8.5% anual.

1.1 Regresión Lineal

La regresión lineal es una herramienta ampliamente utilizada para la realización de modelos ya que permite tener una comprensión más profunda de los fenómenos complejos e intenta realizar inferencias de datos que son complicados para un humano tratar de predecir Rodrigo (2020).

El modelo se enuncia de la siguiente hipótesis:

$$h_\theta = \theta_0 + \theta_1 x_{m,1} + \theta_2 x_{m,2} + ... + \theta_n x_{m,n} \tag{1}$$

donde h_θ es la variable dependiente. Los términos $x_{m,n}$ representan las n variables independientes o explicativas con m observaciones. Los coeficientes del modelo, θ_n son las n variables independientes calculados por el algoritmo de regresión, de modo que se minimicen los residuos.

⋆

La ecuación 1 puede representarse mediante el producto punto

$$h_\theta = X \cdot \Theta \tag{2}$$

Teniendo entonces que

$$\begin{bmatrix} h_0 \\ h_1 \\ . \\ . \\ h_m \end{bmatrix} = \begin{bmatrix} x_{1,0} & x_{1,1} & . & . & x_{1,n} \\ x_{2,0} & x_{2,1} & . & . & x_{2,n} \\ . & . & . & . & . \\ . & . & . & . & . \\ x_{m,0} & x_{m,1} & . & . & x_{m,n} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ \theta_n \end{bmatrix} \tag{3}$$

donde $x_{m,0} = 1$

Para entender si un modelo de regresión lineal tiene una buena aproximación se necesitan métricas para evaluarlo, la primera que se tratará en éste articulo es *Mean Squared Error (MSE)*, la cuál se define como el promedio de la diferencia entre los valores predichos y los observados, ésta diferencia se eleva al cuadrado. A continuación se muetra la fórmula para calcular el MSE de la regresión.

$$MSE = \frac{\sum_{i=1}^{m}(y_i - h_i)^2}{m} \tag{4}$$

Siendo y_i el valor observado, h_i el valor estimado y m el número de observaciones. El valor de ésta métrica se debe que al elevar al cuadrado hay un impacto significativo de los errores grandes y no penaliza en gran medida los errores pequeños.

1.2 Algoritmo gradiente descendente

La función del gradiente es esencial en el ámbito del Deep Learning ya que permite obtener una optimización de un modelo de predicción de cualquier tipo, para obtenerlo se pretende ajustar una función lineal 1 y evaluar mediante la función costo MSE 4.

$$J(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2 \tag{5}$$

el algoritmo Gradiente descendiente se repite hasta alcanzar converger. El gradiente de una función $J(\Theta)$, donde $\Theta = [\theta_0, \theta_1, .., \theta_n]$ se define como:

$$\nabla J(\Theta) = \left[\frac{\partial J}{\partial \theta_0}, \frac{\partial J}{\partial \theta_1}, .., \frac{\partial J}{\partial \theta_j}\right] \tag{6}$$

de modo que el algoritmo del gradiente descendente sería

$$\Theta := \Theta - \alpha \nabla J(\Theta) \tag{7}$$

Entonces calculamos la derivada parcial de la Ec. 5 con respecto a θ_0 y θ_1, obteniendo que

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_{i,0}, x_{i,1}) - y_i)x_{i,0} \tag{8}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_{i,0}, x_{i,1}) - y_i)x_{i,1} \tag{9}$$

Si se reescriben y generalizan de forma matricial las ecuaciones 8 y 9 obtenemos

$$\nabla J(\Theta) = \frac{1}{m}(h_\theta - Y) \cdot X^T = \frac{1}{m}(X \cdot \Theta - Y) \cdot X^T \tag{10}$$

Formando así el conjunto de ecuaciones para aproximar los parámetros Θ al conjunto de datos independientes contra los datos dependientes.

1.3 Normalización de datos

Si tranformamos los datos para que éstos tengan la propiedad de una distribución normal, aplicamos

$$X_{std} = \frac{X - \mu}{\sigma} \tag{11}$$

donde μ es la media de las observaciones de X y σ es la desviación estándar de las observaciones de X. De esta forma la distribución de X_{std} tendrá media cero y desviación estándar de 1.

1.4 Correlación de Pearson

La correlación de Pearson (r) es una medida estadística que evalúa la relación lineal, el rango que tiene ésta relación va de [-1, 1] donde 1 es una correlación perfecta, lo que significa que todos los datos están alineados entre las dos variables cuantitativas.

La fórmula matemática para calcular la correlación de Pearson entre dos variables X y Y es:

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2 \sum(Y_i - \bar{Y})^2}} \tag{12}$$

1.5 Descripción del dataset

El dataset que utilizamos describe las ventas de una compañía y la inversión hecha en publicidad por tres medios, por televisión, radio y periódico, con 200 observaciones. Una primer hipótesis que tenemos sobre los datos, es que las ventas incrementan cuando se invierte más es publicidad, y por tanto, candidatas a realizar regresión lineal. Se intengará realizar una regresión lineal de 3 variables, empleando un modelo multivariable generalizado, donde además se prueba realizar la regresión mediante el algoritmo gradiente descendente y normalizando los datos para obtener un entrenamiento más rápido

2. EXPERIMENTACIÓN

Para analizar nuestro dataset se imprimió el encabezado de la tabla de la base de datos mediante el atributo *.head()*. Obteniendo la tabla original se muestra en la Figura 1.

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

Fig. 1. Muestreo de nuestro dataset.

Para encontrar una relación de los valores se hizo un heatmap mostrado en la Figura 2 el cuál muestra las correlaciones de los valores de una forma gráfica y numérica, observamos los valores que tienen mayor correlación con la columna 'Sales' son TV, Radio y Newspaper, en orden de mayor a menos. Nuestra Y para el análisis será 'Sales' y 'TV','Radio' y 'Newspaper' siendo nuestras X.

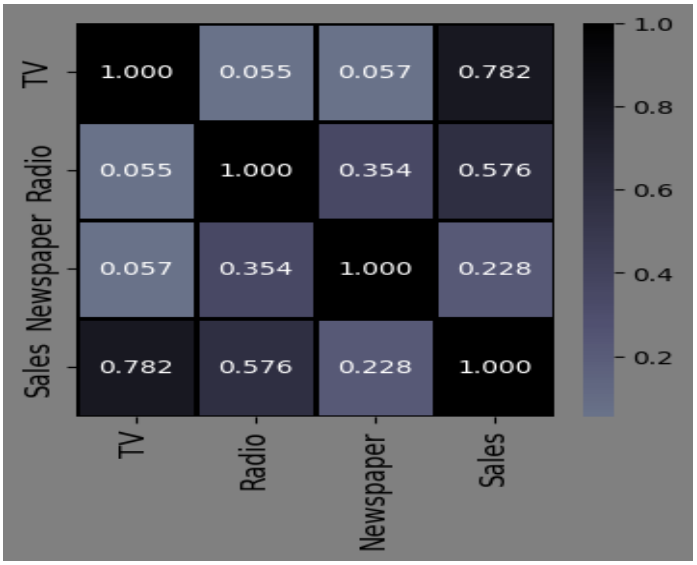

Fig. 2. Correlaciones de nuestro Dataset.

Mostramos los datos empleando la librería pandas:

```
ax=data.plot(kind='scatter',x='TV',y='Sales', color='Green',
        s=15, alpha=.5,label='TV')
data.plot(kind='scatter',x='Radio',y='Sales', color='Blue',
        s=15, alpha=.5,label='Radio', ax=ax)
data.plot(kind='scatter',x='Newspaper', y='Sales',color='Red',
        s=15, alpha=.5,label='Newspaper', ax=ax)
```

Obteniendo la grafica

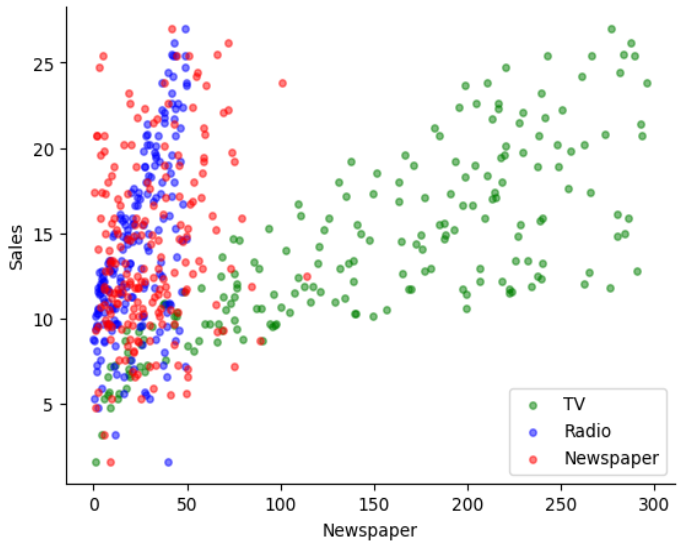

Fig. 3. Grafica de dispesión de la base de datos.

3. RESULTADOS Y DISCUSIÓN

3.1 Entrenamiento sin normalización de datos

El análisis exploratorio de datos realizado hasta ahora nos deja claro que la regresión lineal es aplicable para relacionar las ventas con la publicidad, y la dispersión de los datos 3 nos indica que numéricamente si hay un aumento de las ventas de manera proporcional con la publicidad , una pista ya sugerida por su correlación.

Creamos una variable de ventas estimada o hipótesis llamada h los cuales tienen que aproximarse a los de las observaciones de la base de datos Y. Inicializamos primeramente la variable θ con valores aleatorios y después creamos la función de hipótesis h a la cual le introducimos las observaciones de variables independientes junto con una columna de 1, $X = [\mathbf{1}, X]$ utilizando la función de Numpy hstack.

```
import numpy as np
columns_lst = list(data.columns)[1:4]
ones=np.ones((len(data['TV']),1))
print("ones="+str(ones.shape))
print('data='+str(data[columns_lst].shape))
theta = np.random.rand(4,1)
X = np.hstack((ones,data[columns_lst]))
h = h_m(X,theta)
Y = data['Sales']
Y = np.array(Y).reshape(200,1)

print('X=ones hstack data='+str(X.shape))
print('theta='+str(theta.shape))
print('h=X dot theta='+str(h.shape))
```

cuya salida es:

```
ones=(200, 1)
data= (200, 3)
X=ones hstack data= (200, 4)
theta=(4, 1)
h=X dot theta=(200, 1)
```

Se crearon una serie de funciones de python que generalizan las formulas de una sola variable a varias variables:

```
def h_m(X,theta):
    h = np.dot(X,theta)
    return h

def MSE_m(theta,x,y):
    block = h_m(x,theta) - y
    m = len(x)
    return np.dot(block.transpose(),block)/(2*m)

def R2(h,Y):
    return (1 - sum((h-Y)**2)/sum((Y-np.mean(Y))**2))

def normalizacion(data):
    mean = np.mean(data)
    std_dev = np.std(data)
    return  (data - mean) / std_dev
```

Después se emplean para graficar los datos de ventas con Y y h como eje x.

```
import matplotlib.pyplot as plt

costo = MSE_m(theta, X , Y)
print("Costo:"+str(float(costo)))
R_2 = R2(h,Y)
print("R-squared:"+str(R_2))

x=X[:,1:]
plt.scatter(X[:,1],Y,c='red',marker='.')
plt.scatter(X[:,2],Y,c='green',marker='.')
plt.scatter(X[:,3],Y,c='blue',marker='.')

plt.scatter(X[:,1],h,c='magenta',marker='.')
plt.scatter(X[:,2],h,c='yellow',marker='.')
plt.scatter(X[:,3],h,c='cyan',marker='.')

plt.show()
```

Con salida:

```
Costo: 2736.181867531726
R-squared: [-201.03852571]
```

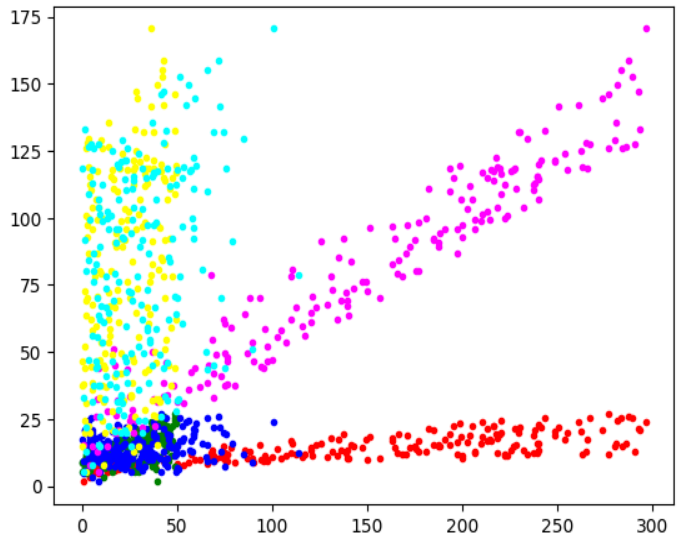

Fig. 4. Gráficas de dispersión de los datos de la base de datos comparados con la función hipótesis con parámetros inicializados aleatoriamente

En la Figura 4 se observa claramente con colores que las estimaciones tienen una mayor dispersión que las originales, además de que el parámetro R^2 es muy grande.

3.2 Entrenamiento

Al hacer una regresión lineal de todos los datos,

```
m = len(data['Sales'])
mse = MSE_m(theta, X , Y)

alpha = 5e-5
epochs = 500000;
```

```
list_theta = []
list_mse = []

list_theta.append(theta)
list_mse.append(float(mse))

for epoch in range(epochs):
    h = h_m(X,theta)
    partialJ = (1/m)*np.dot(X.transpose(),h-Y)
    theta = theta - alpha*partialJ
    mse = MSE_m(theta, X , Y)
    list_theta.append(theta)
    list_mse.append(float(mse))
print("list MSE first value: "+str(list_mse[0])+", last value: "+str(list_mse[-1]))
R_2 = R2(h,Y)
print("R-squared: "+str(R_2))
```

Con resultado:

```
list MSE first value: 2736.1818, last value: 1.3923
R-squared: [0.8971933]
```

donde observamos que el primer valor MSE era muy alto (2736.1818) y al final de las 500,000 épocas descendió a 1.3923. Por otro lado la R^2 ya se encuentra dentro del rango de 0-1 y se aproxima a 1. Se grafican a hora la dispersión de datos con el código:

```
from matplotlib import pyplot as plt
data_trained = pd.DataFrame(np.hstack([X,Y,h]),
    columns=['Unnamed: 0', 'TV', 'Radio', 'Newspaper',
    'Sales','Sales_model'])
print(data_trained.head())

ax=data_trained.plot(kind='scatter', x='TV',
    y='Sales', color='Green', s=15, alpha=.5,label='TV')
data_trained.plot(kind='scatter', x='TV',
    y='Sales_model', color='Magenta', s=15,
    alpha=.7,label='TV-modeled', ax=ax)
data_trained.plot(kind='scatter', x='Radio',
    y='Sales', color='Blue', s=15, alpha=.5,label='Radio',
    ax=ax)
data_trained.plot(kind='scatter', x='Radio',
    y='Sales_model', color='Yellow', s=15,
    alpha=.7,label='Radio-modeled', ax=ax)
data_trained.plot(kind='scatter', x='Newspaper',
    y='Sales',color='Red', s=15,
    alpha=.5,label='Newspaper', ax=ax)
data_trained.plot(kind='scatter', x='Newspaper',
    y='Sales_model',color='Cyan', s=15,
    alpha=.7,label='Newspaper-modeled', ax=ax)
```

Se visualiza entonces la Figura 5

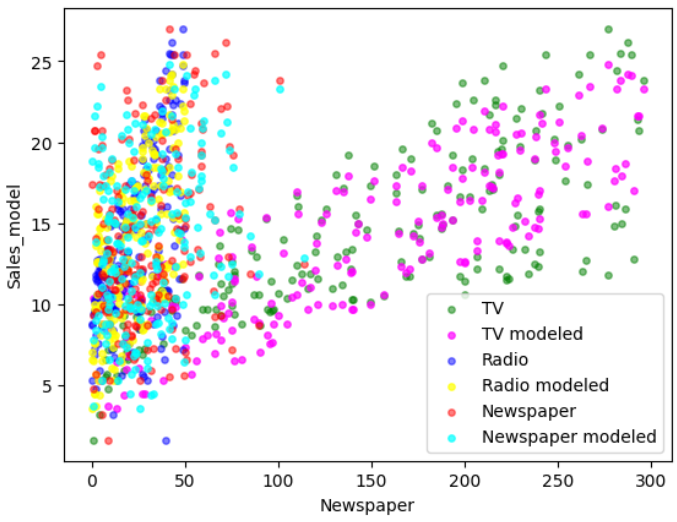

Fig. 5. Gráfica de dispersión de la base de datos comparado con la regresión lineal.

3.3 Regresión con estandarización de datos

La regresión lineal presentada anteriormente fueron con learning rates de $\alpha = 5 \times 10^{-5}$ y con 500, 000 épocas, esto requiere unos pocos segundos en completarse, y alcanza un R^2 de 0.8971933. Esperamos que con la normalización de datos obtener un mejor entrenamiento o igual, pero con un learning rate mayor y por tanto, con menor cantidad de épocas. Las funciones mencionadas anteriormente se volvió a declarar un vector-4 aleatorio Θ y se normalizó las variables independientes X a X_{std} mediante el siguiente código:

```
X_norm = normalizacion(X)
theta = np.random.rand(4,1) #numero de caracteristicas

h = h_m(X,theta)
costo = MSE_m(theta, X_norm , Y)
print("Costo: "+str(float(costo)))
R_2 = R2(h,Y)
print("R-squared: "+str(R_2))
```

Con resultado:

```
Costo: 104.9843005332167
R-squared: [-873.70546776]
```

Y empleando el mismo código de entrenamiento y con un learning rate $\alpha = 5 \times 10^{-1}$ y con sólo 1000 épocas se obtiene ahora un resultado de:

```
list MSE first value: 104.9843, last value: 1.392
R-squared: [0.89721064]
```

Visualmente la dispersión luce como la Figura 5, pero numéricamente es ligeramente mejor y 500 veces más rápido que el resultado obtenido sin normalizar la base de datos.

4. CONCLUSIONES

Debido a que ahora hay mas variables involucradas en la regresión, observé que la función de hipótesis dejó de ser una línea recta y se formó una nube de rectas, de modo que se cambió a dispersión. Se pudo aplicar la regresión lineal por gradiente descendente de forma satisfactoria, y mediante normalización se realizó mucho más rápido.

REFERENCES

Rodrigo, J.A. (2020). Regresión lineal con python. URL `https://cienciadedatos.net/documentos/py10-regresion-lineal-python`.