Inteligencia Artificial
Act 10: Programando Regresión Lineal Múltiple en Python

1 Introducción

Mientras que la regresión lineal simple se basa en una sola variable de entrada y se relaciona con una variable de salida, la regresión lineal múltiple considera múltiples variables de entrada proporcionando un algoritmo más preciso.

El algoritmo de regresión lineal múltiple determina la relación entre múltiples variables de entrada (variables independientes) y una variable de salida (variable dependiente) mediante una ecuación lineal. La ecuación es: $y = b_0 + b_1x_1 + b_2x_2 + \ldots + b_nx_n$, donde:

- y es la variable dependiente, la variable que queremos predecir
- $x_1, x_2, \ldots, x_n$ son las variables independientes
- b_0 es la intersección
- $b_1, b_2, \ldots, b_n$ son las pendientes

El objetivo del algoritmo es encontrar la mejor ecuación de línea de ajuste que pueda predecir los valores en función de las variables independientes.

2 Metodología

Para llevar a cabo la actividad, se siguieron las indicaciones del libro *Aprende Machine Learning*, en la página 34. Primero, se creó una carpeta llamada "Regresión Lineal Múltiple" y se descargó el archivo de entrada CSV. En la carpeta se guardó el archivo CSV y se creó el archivo que va a contener el código Python, para realizar el código se hizo uso de Visual Studio Code.

Se esta utilizando el código realizado en la actividad pasada (Act 9: Programando Regresión Lineal en Python) con unas pequeñas modificaciones, lo que único que vamos a conservar serían los imports necesarios así como la carga y lectura de nuestro archivo CSV y nuestros datos ya filtrados.

```
# Imports necesarios
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

# cargamos los datos de entrada
data = pd.read_csv("./articulos_ml.csv")

# filtrar los datos
filtered_data = data[(data['Word count'] <= 3500) & (data['# Shares'] <=
    80000)]
```

En vez de usar una sola variable predictiva, se usarán dos variables predictivas, donde nuestra nueva variable será la suma de los enlaces, comentarios e imágenes.

```
# variable nueva = suma de los enlaces, comentarios e imagenes
suma = (filtered_data['# of Links'] + filtered_data['# of comments'].fillna(0)
     + filtered_data['# Images video'])
```

Preparamos los datos para empezar su entrenamiento, para esto definimos un nuevo dataframe, donde añadiremos las palabras contadas con nuestra nueva variable para después convertir el dataframe en un array, al final tenemos nuestra variable que usaremos como etiqueta de entrenamiento.

```
# preparacion de las variables antes del entrenamiento
dataX2 = pd.DataFrame()
dataX2['Word count'] = filtered_data['Word count']
dataX2['suma'] = suma
XY_train = np.array(dataX2)
z_train = filtered_data['# Shares'].values
```

Ahora nos pasamos a crear un objeto de regresión lineal con dos dimensiones, entrenamos el modelo y hacemos predicciones.

```
regr2 = linear_model.LinearRegression() # objeto de regresion lineal
regr2.fit(XY_train, z_train) # entrenar el modelo con 2 dimensiones
z_pred = regr2.predict(XY_train) # prediccion
```

Una vez entrenado el modelo, podemos imprimir los coeficientes, el error cuadrático medio y el puntaje de varianza, esto para poder interpretar el modelo y poder evaluar el rendimiento de nuestro modelo de regresión lineal múltiple.

```
print('Coefficients: ', regr2.coef_)
print("Mean squared error: %.2f" % mean_squared_error(z_train, z_pred))
print('Variance score: %.2f' % r2_score(z_train, z_pred))
```

Listing 1: Resultados del modelo

Para visualizar nuestro modelo, hay que usar un plano en 3 dimensiones, en nuestro caso, graficaremos nuestros puntos de entrenamiento en azul y los puntos predictivos en rojo.

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

# malla sobre la cual se graficara el plano
xx, yy = np.meshgrid(np.linspace(0, 3500, num=10), np.linspace(0, 60, num=10))

# calculo de los valores del plano
nuevoX = (regr2.coef_[0] * xx)
nuevoY = (regr2.coef_[1] * yy)
z = (nuevoX + nuevoY + regr2.intercept_)

ax.plot_surface(xx, yy, z, alpha=0.2, cmap='hot') # graficamos el plano
ax.scatter(XY_train[:, 0], XY_train[:, 1], z_train, c='blue',s=30) # azul los
    puntos de entrenamiento
ax.scatter(XY_train[:, 0], XY_train[:, 1], z_pred, c='red',s=40) # rojo los
    puntos predictivos
ax.view_init(elev=30., azim=65) # situamos la 'camara' para visualizar
```

Le agregamos títulos a los ejes y mostramos el plano.

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

plt.show()
```

Listing 2: Visualización del modelo

Para terminar, probamos nuestro modelo prediciendo cuántos "Shares" se van a obtener por un artículo con 2000 palabras, 10 enlaces, 4 comentarios y 6 imágenes.

```
# predecir cuantos #Shares voy a obtener por un articulo con:
# 2000 palabras, 10 enlaces, 4 comentarios, 6 imagenes
z_Dosmil = regr2.predict([[2000, 10+4+6]])
print('Prediccion: ', int(z_Dosmil))
```

Listing 3: Prediccion del modelo

3 Resultados

A continuación, se van a presentar los resultados obtenidos a partir del código mostrado.

3.1 Resultados del modelo

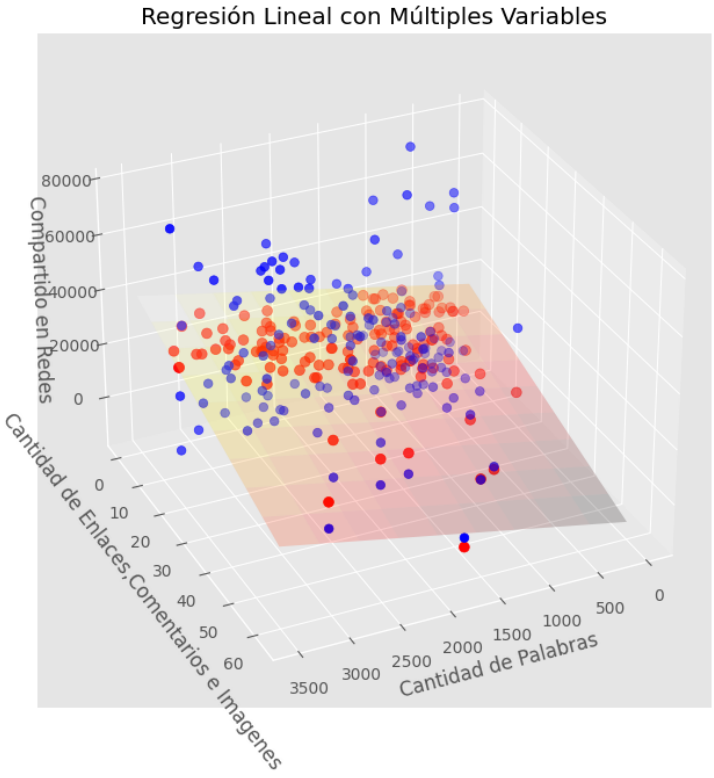
En el Listing 1 se evaluó el modelo, obteniendo los siguientes resultados:

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

Podemos notar que obtenemos, dos coeficientes, cada uno corresponde a nuestras dos variables predictivas. El error cuadrático medio obtenido sigue siendo considerablemente grande, sin embargo es mejor que con el modelo de regresión lineal simple y el puntaje de varianza es mejor por el doble aunque sigue siendo malo, ya que esta muy lejos del mejor valor posible, 1. Estos resultados nos dan a entender que nuestro algoritmo no será muy eficiente, pero sigue siendo mejor que el modelo de regresión simple.

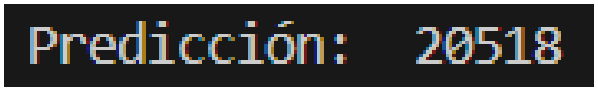
3.2 Visualización del modelo

El plano en 3D del modelo (Listing 2) queda de la siguiente manera:



3.3 Predicción del algoritmo

Finalmente, para el Listing 3 se probo el algoritmo para predecir cuantos "Shares" se obtendrán para un artículo de 2000 palabras y con 10 enlaces, 4 comentarios y 6 imágenes, el resultado fue el siguiente:



Entonces, dado nuestro algoritmo, se obtendrán 20,518 Shares para un artículo de 2000 palabras con 10 enlaces, 4 comentarios y 6 imágenes

4 Conclusión

En esta actividad, utilizamos un archivo CSV para entrenar un modelo de regresión lineal múltiple. Reutilizamos poquito código que usamos en la actividad pasada, así que con lo primero que empezamos fue con la creación de una nueva variable predictiva, y seguimos a preparar los datos para hacer uso de ellos en el entrenamiento. De igual manera, utilizamos el objeto de regresión lineal e imprimimos métricas de rendimiento para saber que tan efectivo fue nuestro modelo, lamentablemente el modelo sigue siendo malo, pero es mejor que el modelo de regresión lineal simple. No obstante, esto nos dice que deberiamos de utilizar más dimensiones así como encontrar datos de entrada mejores.