Actividad 9 - Inteligencia Artificial
Programando Regresión Lineal en Python

**Nombre: Idalia Alejandra Delgado Moreno**
**Grupo: 31**

**Introducción**
**¿Qué es una regresión lineal**

La regresión lineal simple es una técnica que análisis de datos que predice el valor de datos desconocidos mediante el uso de otro valor de datos relacionado y conocido. Modela matemáticamente la variable desconocida o dependiente y la variable conocida o independiente como una ecuación lineal. Con Python, es un algoritmo de aprendizaje supervisado que se utiliza Machine Learning y estádistica. Se "dibuja una recta" que nos indica la tendencia de un conjunto de datos continuos. En estadísticas, regresión lineal es una aproximación para modelar una variable escalar dependiente "y" y una o más variables explicaticas nombradas con "X".

Fórmula de la recta:

$Y = mX + b$

Donde Y es el resultado, X es la variable, m la pendiente de la recta y b el punto donde se corta el eje Y en la gráfica cuando X=0.

---

**Metodolodgía**

Para realizar esta actividad, hay que seguir las intrucciones que vienen en el libro proporcionado por el profesor de la págia 34 a la 46 del pdf.

1. Hay que descargar el archivo csv que vienen en el link :artivculos.csv

2. Copias el siguiente código del libro a tu editor de código:

```
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
```

Si no tienes algunas librerías, habría que instalarlas.

**3.** Cargamos los datos de entrada (el documento que se instala en el paso 1) y checamos sus dimensiones:

```
data = pd.read_csv("./articulos_ml.csv")
data.shape
Output:
(161, 8)
```

**4. Estadísticas básicas y visualización:**
Vamos a ver algunas estadísticas y una visualización rápida.

```
data.describe()

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

**5. Filtrar y visualizar datos:**
Filtramos los datos para obtener una representación más clara.

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

colores = ['orange', 'blue']
tamanios = [30, 60]
f1 = filtered_data['Word count'].values
f2 = filtered_data['# Shares'].values

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()
```

**6. Regresión lineal y predicción:**
Entrenamos un modelo de regresión lineal con los datos filtrados.

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

regr = linear_model.LinearRegression()

# Entrenamos el modelo
regr.fit(X_train, y_train)
```

```
# Hacemos las predicciones
y_pred = regr.predict(X_train)

# Mostramos los coeficientes obtenidos
print('Coefficients: \n', regr.coef_)
print('Independent term: \n', regr.intercept_)

# Error cuadrado medio y puntaje de varianza
print("Mean squared error: %.2f" % mean_squared_error(y_train, y_pred))
print('Variance score: %.2f' % r2_score(y_train, y_pred))
```

**7. Graficamos la regresión lineal:**
Visualizamos los datos junto con la línea de regresión.

```
# Graficamos los datos y la regresión lineal
plt.scatter(X_train, y_train, color='blue', label="Datos reales")
plt.plot(X_train, y_pred, color='red', linewidth=2, label="Regresión lineal")
plt.xlabel("Word count")
plt.ylabel("# Shares")
plt.legend()
plt.show()
```

**8. Mejorando el modelo:**
Agregamos una nueva dimensión (suma de enlaces, comentarios e imágenes) para mejorar el modelo.

```
suma = (filtered_data["# of Links"] +
        filtered_data['# of comments'].fillna(0) +
        filtered_data['# Images video'])

# Creamos el nuevo DataFrame con dos características
dataX2 = pd.DataFrame()
dataX2["Word count"] = filtered_data["Word count"]
dataX2["suma"] = suma

# Convertimos a arrays de NumPy para el entrenamiento
XY_train = np.array(dataX2)
z_train = filtered_data['# Shares'].values

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

# Entrenamos el modelo con 2 dimensiones
regr2.fit(XY_train, z_train)

# Hacemos las predicciones
z_pred = regr2.predict(XY_train)
```

```
# Mostramos los coeficientes y evaluamos el modelo
print('Coefficients: \n', regr2.coef_)
print("Mean squared error: %.2f" % mean_squared_error(z_train, z_pred))
print('Variance score: %.2f' % r2_score(z_train, z_pred))
```

**9. Graficamos la regresión en 3D:**
Ahora visualizamos los resultados en un gráfico 3D.

```
fig = plt.figure(figsize=(16, 10))
ax = fig.add_subplot(111, projection='3d')

xx, yy = np.meshgrid(np.linspace(0, 3500, num=50), np.linspace(0, 60, num=50))

nuevoX = regr2.coef_[0] * xx
nuevoY = regr2.coef_[1] * yy
z = nuevoX + nuevoY + regr2.intercept_

ax.plot_surface(xx, yy, z, alpha=0.5, cmap='coolwarm', shade=True)

ax.scatter(XY_train[:, 0], XY_train[:, 1], z_train, c='blue', s=50, label="Datos reales"

z_pred = regr2.predict(XY_train)
ax.scatter(XY_train[:, 0], XY_train[:, 1], z_pred, c='red', s=55, label="Predicciones",

ax.view_init(elev=20, azim=45)

ax.grid(True)

ax.set_xlabel('Cantidad de Palabras', fontsize=14, labelpad=20)
ax.set_ylabel('Suma de Enlaces, Comentarios e Imágenes', fontsize=14, labelpad=20)
ax.set_zlabel('Compartido en Redes', fontsize=14, labelpad=20)
ax.set_title('Regresión Lineal con Múltiples Variables', fontsize=16, pad=25)

ax.legend(fontsize=14)

plt.show()
```

---

**Resultados:**
A continuación, se muestran los gráficos generados por el código:

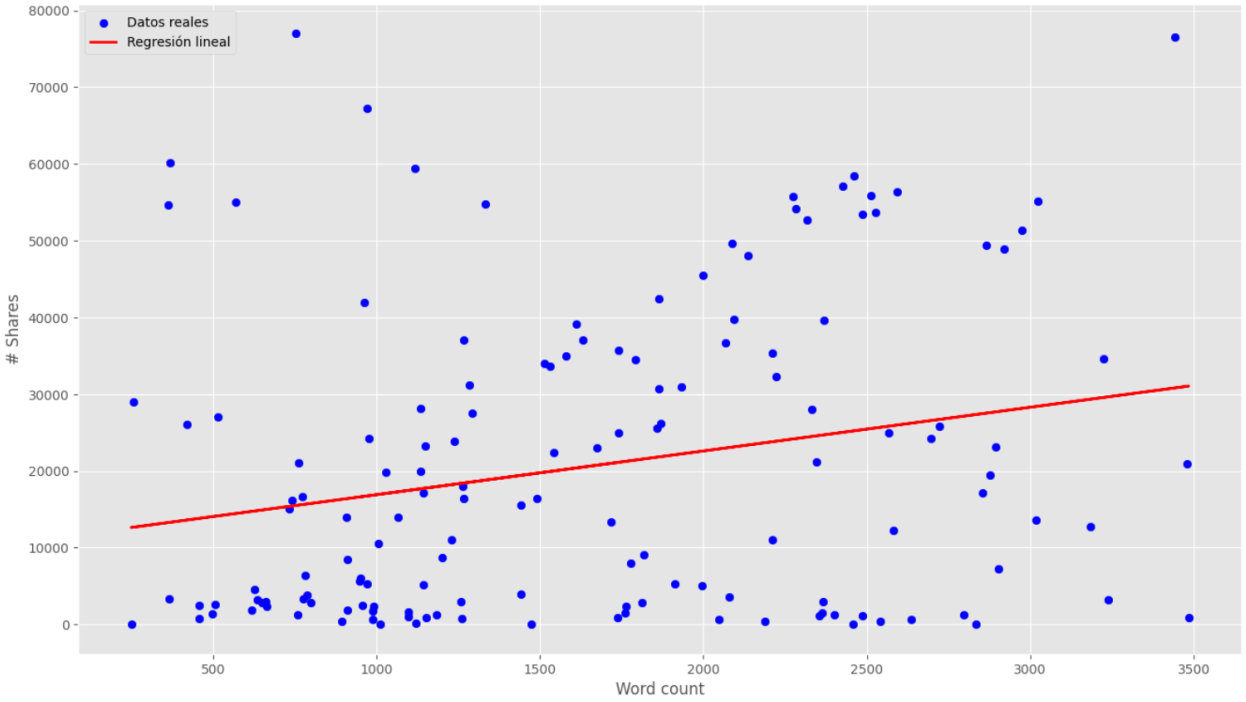


Figure 1: Gráfico de la regresión lineal simple.

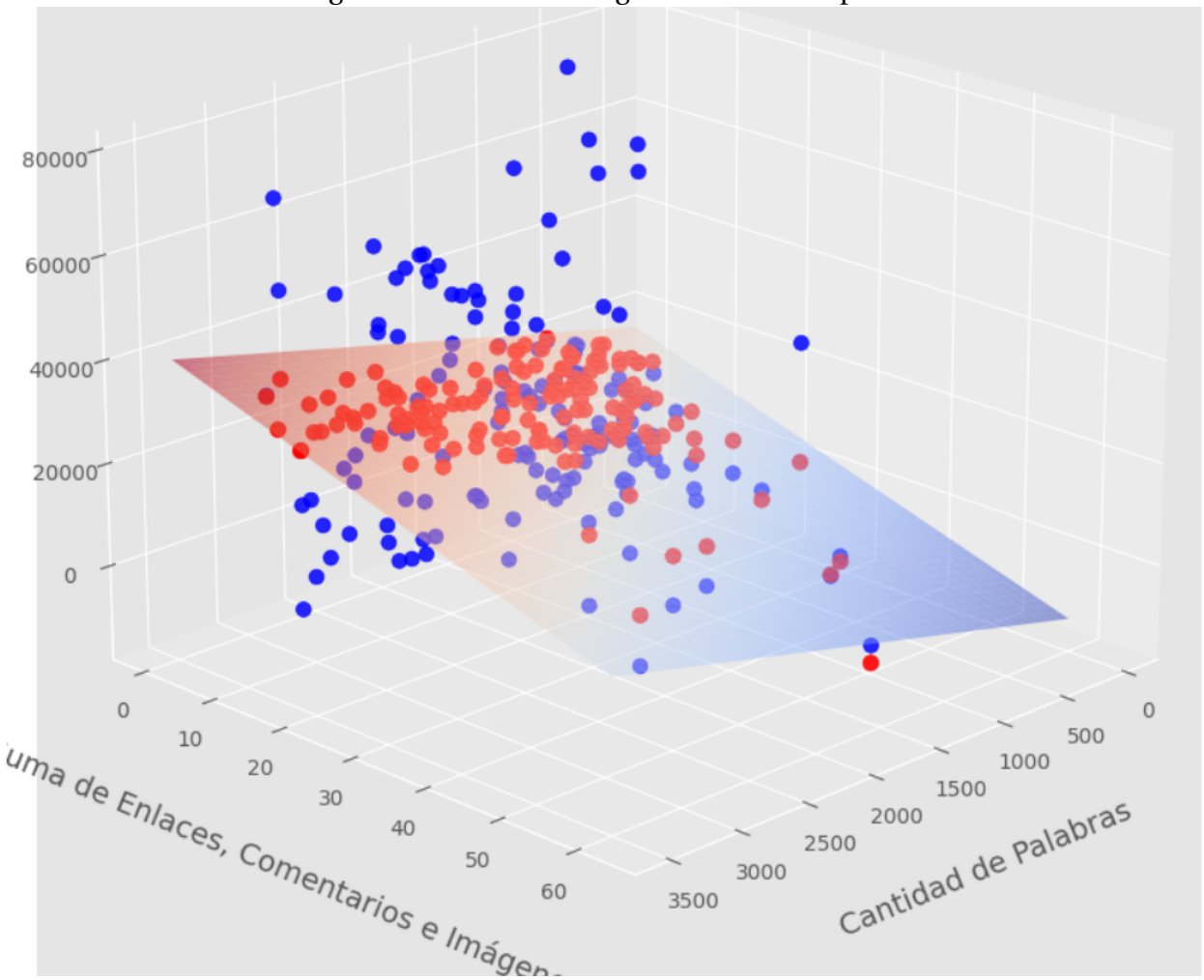


Figure 2: Gráfico 3D de la regresión lineal con múltiples variables.

**Conclusión:**
Durante esta actividad usamos la librería SKIlean para aplicar una regresión lineal al conjunto de datos que obtenidos del archivo articulos.csv y hacer predicciones con ellos. Utilizando las herramientas proporcionadas por SKlearn, como el modelo LinearRegression, pudimos entrenar un modelo predictivo sobre un conjunto de datos reales y evaluar su rendimiento de manera eficiente. También gracias a la librería, nos permitió visualizar los resultados de una forma clara, incluyendo la evaluación del error cuadrático medio y el puntuaje de varianza. En resumen, SKlearn fue fundamental para realizar este análisis, simplificando el proceso de aplicar técnicas de machine learning y mejorar las predicciones de manera efectiva.