Regresión lineal multiple

Hiram Misael Cerda Casas
Matrícula: 1911548

3 de Marzo del 2025

1 Introducción

La regresión lineal múltiple es una extensión de la regresión lineal simple, donde el objetivo es modelar la relación entre una variable dependiente y varias variables independientes. Esta técnica permite predecir el valor de la variable dependiente a partir de múltiples características o predictores. El modelo de regresión lineal múltiple tiene la siguiente forma general:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \epsilon$$

donde:

- Y es la variable dependiente,
- $X_1, X_2, \ldots, X_n$ son las variables independientes o características,
- β_0 es la intersección o el valor de Y cuando todas las variables X son cero,
- $\beta_1, \beta_2, \ldots, \beta_n$ son los coeficientes que representan la influencia de cada variable independiente en Y,
- ϵ es el término de error que representa la variabilidad no explicada por el modelo.

La regresión lineal múltiple se utiliza en diversas áreas como economía, ciencia de datos, y ingeniería para predecir y entender relaciones complejas entre variables.

2 Metodología

En esta sección se describen los pasos seguidos para realizar el análisis de regresión lineal múltiple, con fragmentos de código que ilustran cómo se implementó el modelo.

2.1 Paso 1: Preparación de los datos

- Se recolectaron los datos necesarios para la regresión lineal múltiple. Los datos consisten en una variable dependiente Y y varias variables independientes $X_1, X_2, \ldots, X_n$.
- Los datos fueron limpiados y preprocesados para asegurar que no hubiera valores faltantes ni valores atípicos.

2.2 Paso 2: Implementación del modelo de regresión lineal múltiple

El siguiente fragmento de código en Python utiliza la librería `scikit-learn` para implementar el modelo de regresión lineal múltiple:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Datos de ejemplo
X = data[['X1', 'X2', 'X3']]  # Variables independientes
y = data['Y']  # Variable dependiente

# Dividir los datos en conjunto de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Crear el modelo de regresión lineal
model = LinearRegression()

# Ajustar el modelo a los datos de entrenamiento
model.fit(X_train, y_train)

# Predicciones en el conjunto de prueba
y_pred = model.predict(X_test)
```

2.3 Paso 3: Evaluación del modelo

Después de entrenar el modelo, se evalúa su desempeño utilizando métricas como el R^2, el error cuadrático medio (MSE) y otros indicadores de precisión.

```
from sklearn.metrics import mean_squared_error, r2_score

# Evaluar el modelo
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'MSE: {mse}')
print(f'R2: {r2}')
```

2.4 Visualización en un Mapa 3D

Para mejorar la comprensión visual de la relación entre las variables independientes y la variable dependiente, se creó una visualización en 3D utilizando el paquete `matplotlib` en Python. Esta visualización permite observar cómo se ajusta el modelo de regresión lineal múltiple sobre un plano que se extiende en un espacio tridimensional.

El siguiente fragmento de código muestra cómo se generó la visualización en 3D:

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Crea la figura
fig = plt.figure()
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 intersección
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 predichos
ax.scatter(XY_train[:, 0], XY_train[:, 1], z_pred, c='red', s=40)

# Configuración de la vista
ax.view_init(elev=30., azim=65)
ax.set_xlabel('Cantidad de Palabras')
ax.set_ylabel('Cantidad de Enlaces, Comentarios e Imágenes')
ax.set_zlabel('Compartido en Redes')
ax.set_title('Regresión Lineal con Múltiples Variables')

# Mostrar el gráfico
plt.show()
```

En este código:

- Se crea un gráfico en 3D utilizando `Axes3D` de `matplotlib`.
- Se genera una malla de puntos (`xx` y `yy`) para graficar el plano de regresión.
- Los puntos reales de entrenamiento se grafican en azul, mientras que los puntos predichos por el modelo se grafican en rojo.
- Se ajusta la vista 3D para facilitar la interpretación visual de los resultados.

Esta visualización facilita la comprensión de cómo el modelo de regresión lineal múltiple ajusta los datos en un espacio tridimensional y permite observar las predicciones comparadas con los datos reales.

3 Resultados

En esta sección se presentarán los resultados obtenidos del análisis de regresión lineal múltiple, como las métricas de evaluación y las predicciones del modelo.

- El error cuadrático medio (MSE) obtenido es: 352122816.48.
- El valor R^2 es: 0.11.
- Las predicciones realizadas por el modelo fueron: 20518 compartidas.

4 Conclusión

La regresión lineal múltiple es una herramienta poderosa para entender la relación entre varias variables y hacer predicciones. Su aplicación es amplia, desde la predicción de precios en economía hasta el análisis de factores que influyen en ciertos fenómenos científicos. Los resultados obtenidos a través de esta metodología permiten tomar decisiones informadas y realizar ajustes en los modelos predictivos.

La capacidad de incluir múltiples variables independientes permite capturar relaciones más complejas, lo que hace que la regresión lineal múltiple sea una técnica fundamental en el análisis de datos.