Regresion Lineal Multiple

Alain Lobato

31 de Marzo del 2025

1 Introduccion

La regresion lineal multiple es una extension de la regresion lineal simple, permitiendo analizar la relacion entre una variable dependiente y multiples variables predictoras. En lugar de ajustar una linea recta en un plano bidimensional, este modelo ajusta un hiperplano en un espacio de mayor dimension. La ecuacion ahora toma la forma:

$$Y = b + m_1X_1 + m_2X_2 + \ldots + m_nX_n \tag{1}$$

Añadir mas variables nos permite obtener predicciones mas complejas y, en muchos casos, mas precisas. Sin embargo, tambien aumenta la necesidad de interpretar correctamente los coeficientes y evitar problemas como la multicolinealidad.

2 Metodologia

Primero, clonamos nuestro repositorio de la materia en nuestra maquina y creamos una carpeta llamada `Multiple Linear Regression`. Luego, descargamos el archivo CSV con los datos y creamos el archivo `m_linear_regression.py` donde desarrollamos nuestro codigo en Visual Studio Code.

Iniciamos importando las librerias necesarias e instalando las dependencias en caso de ser necesario.

```python
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
```

Figure 1: Importacion de librerias.

Ahora, agregamos dos variables predictivas adicionales, lo que nos permitira graficar en 3D.

```
suma = (filtered_data["# of Links"] + filtered_data['# of comments'].fillna(0) + filtered_data['# Images video'])
dataX2 = pd.DataFrame()
dataX2["Word count"] = filtered_data["Word count"]
dataX2["suma"] = suma
XY_train = np.array(dataX2)
z_train = filtered_data['# Shares'].values
```

Figure 2: Agregacion de variables predictivas.

Creamos nuestro modelo de regresion lineal y lo entrenamos con los datos, imprimiendo los puntajes obtenidos.

```
# Creamos un nuevo objeto de Regresión Lineal
regr2 = linear_model.LinearRegression()
# Entrenamos el modelo, esta vez, con 2 dimensiones
# obtendremos 2 coeficientes, para graficar un plano
regr2.fit(XY_train, z_train)
# Hacemos la predicción con la que tendremos puntos sobre el plano hallado
z_pred = regr2.predict(XY_train)
# Los coeficientes
print('Coefficients: \n', regr2.coef_)
# Error cuadrático medio
print("Mean squared error: %.2f" % mean_squared_error(z_train, z_pred))
# Evaluamos el puntaje de varianza (siendo 1.0 el mejor posible)
print('Variance score: %.2f' % r2_score(z_train, z_pred))
```

Figure 3: Entrenamiento del modelo.

Los puntajes obtenidos se muestran a continuacion:

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

Figure 4: Puntajes de la regresion.

Graficamos los datos originales en azul y los proyectados sobre el plano de regresion en rojo en un espacio tridimensional.

```python
fig = plt.figure()
#ax = Axes3D(fig)
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 valores para z. Debemos sumar el punto de intercepion
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
ax.scatter(XY_train[:, 0], XY_train[:, 1], z_pred, c='red',s=40)

# con esto situamos la "camara" con la que visualizamos
ax.view_init(elev=30., azim=65)
ax.set_xlabel('Cantidad de Palabras')
ax.set_ylabel('Cantidad de Enlaces,Comentarios e Imagenes')
ax.set_zlabel('Compartido en Redes')
ax.set_title('Regresión Lineal con Múltiples Variables')
plt.show()
```

Figure 5: Visualizacion 3D de los datos y el modelo de regresion.

3 Resultados

Realizamos una prediccion para un post con 2000 palabras, 10 enlaces, 4 comentarios y 6 imagenes.

```python
z_Dosmil = regr2.predict([[2000, 10+4+6]])
print('Predicción:',int(z_Dosmil[0]))
```

Figure 6: Prediccion de shares.

```
Predicción: 20518
```

Figure 7: Resultados de la prediccion.

Podemos ver que la prediccion nos da un valor aproximado de 20518 compartidos, mejorando el resultado de la regresion lineal simple.

4 Conclusion

La regresion lineal multiple nos permite trabajar con mas de una variable predictora, haciendo que nuestras predicciones sean mas completas y menos susceptibles a sesgos. Aunque en este caso los resultados mejoraron, tambien nos damos cuenta de la importancia de elegir adecuadamente nuestras variables de entrada y de entender los coeficientes obtenidos. Como programadores, aprender y aplicar estos modelos nos da herramientas poderosas para el analisis de datos.

5 Referencias

Materiales de clase (2025). UANL.
Bagnato J. (2019). Aprende Machine Learning. Leanpub.