Codigo de Regresión Lineal

Manuel Montufar Galnares

Junio 2024

El presente trabajo es para mostrar el codigo utilizado para hacer una regresión lineal sin usar un frame work como sklearn o scipy, sino simples arreglos y operaciones que nos permitan predecir un valor.

La base de datos utilizada es un csv con información de varios vehiculos y contiene un seeling price que sera nuestra variable a predecir cabe aclarar que esta base ya habia sido en su mayoria limpiada para otro trabajo del semestre pasado, así que podré ahorrarme todo el trabajo del analisis exploratorio y limpieza de la base.

Dicho documento con la limpieza de la base tambien sera subido al repositorio de github en caso de que el presente profesor o lector de este documento desee verlo, sin más que añadir comencemos.

```
    """
␣␣␣␣Importacion␣de␣librerias
␣␣␣␣"""

    import polars as pl # Lectura y Transformacion de Datos
    import matplotlib.pyplot as plt # Visualizacion de Datos
    import seaborn as sns
    import numpy as np # Manejo de arreglos y operaciones de vectores
```

Comenzamos importando las librerias que utilizaremos para el trabajo.

```
    """
    Lectura de la base de datos
    """
␣␣␣␣#␣Establecemos␣la␣ruta␣de␣la␣base
␣␣␣␣ruta␣=␣r'C:\Users\Manuel␣Montufar\Documents\ProyectosManu\Escuela\
␣␣␣␣␣␣␣␣Concentracion\Clase␣Uresti\Implementacion\Vehiculos.csv'
␣␣␣␣#␣Definimos␣el␣data␣frame␣y␣leemos␣la␣base
␣␣␣␣df␣=␣pl.read_csv(f'{ruta}')
␣␣␣␣#Imprimimos␣el␣data␣frame␣y␣el␣tamanio␣de␣la␣base
␣␣␣␣print(df)
```

Hacemos la lectura de la base de datos con Polars y lo imprimimos para obtener el siguiente data frame:

shape: (558_837, 10)

year	make	model	body	…	color	interior	mmr	sellingprice
i64	str	str	str		str	str	i64	i64
2015	Kia	Sorento	SUV	…	White	black	20500	21500
2015	Kia	Sorento	SUV	…	White	beige	20800	21500
2014	BMW	3 Series	Sedan	…	Gray	black	31900	30000
2015	Volvo	S60	Sedan	…	White	black	27500	27750
2014	BMW	6 Series Gran Coupe	Sedan	…	Gray	black	66000	67000
…	…	…	…	…	…	…	…	…
2015	Kia	K900	Sedan	…	Silver	black	35300	33000
2012	Ram	2500	Crew Cab	…	White	black	30200	30800
2012	BMW	X5	SUV	…	Black	black	29800	34000
2015	Nissan	Altima	sedan	…	White	black	15100	11100
2014	Ford	F-150	SuperCrew	…	Gray	gray	29600	26700

Figura 1: Data Frame Crudo

```
    """
␣␣␣␣Analisis␣y␣Transformacion␣de␣Datos
␣␣␣␣"""
    # Esta base ya habia sido previamente limpiada para un proyecto anterior,
         adjuntare el EDA de dicho trabajo

    # Conteo de valores nulos por columna
    nulos = df.null_count()
    # Imprime el resultado
    print(f"El␣conteo␣de␣valores␣nulos␣es:␣{nulos}")

    # Encuentra la moda de la columna 'interior'
    moda_interior = df.select(pl.col("interior").mode()).item()
    print(f"La␣moda␣de␣la␣columna␣interior␣es:␣{moda_interior}")

    # Rellena los valores nulos en la columna 'interior' con la moda
    df = df.with_columns(
        pl.col("interior").fill_null(moda_interior)
    )

    # Verifica que ya no haya valores nulos
    nulos_post = df.null_count()
    print(f"Lo␣valores␣nulos␣despues␣de␣aplicar␣la␣moda␣son:␣{nulos_post}")

    # Obtiene los valores unicos por columna
    valores_unicos = {col: df[col].unique().to_list() for col in df.columns}

    # Imprime los valores unicos
    for col, valores in valores_unicos.items():
        print(f"Columna␣'{col}'␣tiene␣{len(valores)}␣valores␣unicos:
␣␣␣␣␣␣␣␣␣␣␣␣{valores[:10]}␣{'...'␣if␣len(valores)␣>␣10␣else␣''}")

    # Vemos estadisticas descriptivas
    print(df.describe())
```

Hacemos trasnformaciones de datos, checamos valores nulos y los rellenamos con la moda, contamos los valores unicos y vemos estadisticas descriptivas de las variables
Obtenemos el siguiente output:

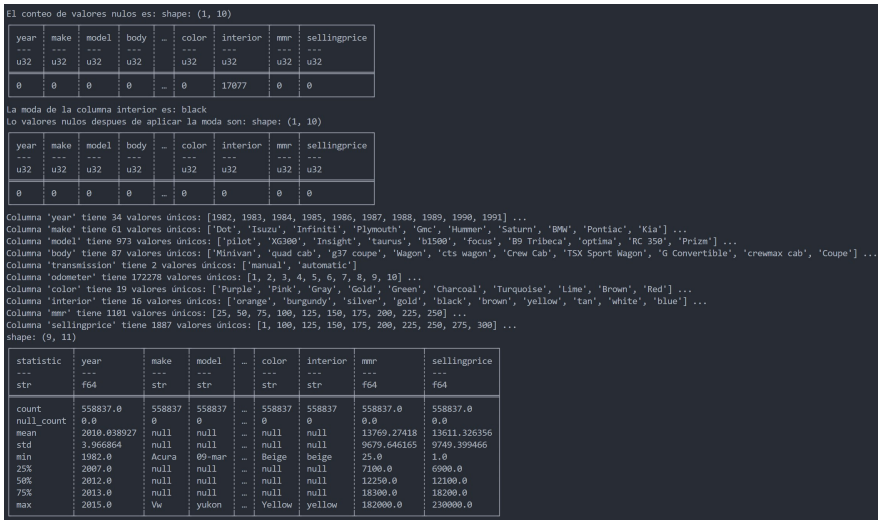

Figura 2: Transformación de Datos

```
    """
␣␣␣␣Identificacion␣de␣Outliers␣en␣las␣variables␣a␣elegir
␣␣␣␣"""
    # Selecciona las columnas relevantes
    df = df.select(["mmr", "sellingprice"])

    # Asignamos el mismo tipo de Dato
    mmr = df['mmr'].to_numpy().astype(float)
    sellingprice = df['sellingprice'].to_numpy().astype(float)

    # Funcion para calcular el IQR y detectar outliers
    def detectar_outliers(data):
        q1 = np.percentile(data, 25)
        q3 = np.percentile(data, 75)
        iqr = q3 - q1
        lower_bound = q1 - 1.5 * iqr
        upper_bound = q3 + 1.5 * iqr
        return (data < lower_bound) | (data > upper_bound), lower_bound, upper_bound

    # Identificar outliers
    outliers_mmr, lower_bound_mmr, upper_bound_mmr = detectar_outliers(mmr)
    outliers_sellingprice, lower_bound_sp, upper_bound_sp = detectar_outliers(sellingprice)

    # Imprimir resultados
    print(f"Numero␣de␣outliers␣en␣'mmr':␣{np.sum(outliers_mmr)}")
    print(f"Limites␣para␣outliers␣en␣'mmr':␣{lower_bound_mmr}␣(inferior)␣y␣{upper_bound_mmr}
␣␣␣␣␣␣␣␣␣␣(superior)")

    print(f"Numero␣de␣outliers␣en␣'sellingprice':␣{np.sum(outliers_sellingprice)}")
    print(f"Limites␣para␣outliers␣en␣'sellingprice':␣{lower_bound_sp}␣(inferior)
␣␣␣␣␣␣␣␣y␣{upper_bound_sp}␣(superior)")

    # Filtrar valores negativos en 'mmr'
    valores_negativos_mmr = df.filter(pl.col("mmr") < 0)
    print(f"Valores␣negativos␣en␣'mmr':␣{valores_negativos_mmr.shape[0]}")
    print(valores_negativos_mmr.head())

    # Filtrar valores negativos en 'sellingprice'
    valores_negativos_sellingprice = df.filter(pl.col("sellingprice") < 0)
    print(f"Valores␣negativos␣en␣'sellingprice':␣{valores_negativos_sellingprice.shape[0]}")
    print(valores_negativos_sellingprice.head())

    # Eliminar outliers basados en los limites
    df_limpio = df.filter(
        (pl.col("mmr") >= 0) & (pl.col("mmr") <= 35100) &
        (pl.col("sellingprice") >= 0) & (pl.col("sellingprice") <= 35150)
    )

    print(f"Numero␣de␣filas␣despues␣de␣eliminar␣outliers:␣{df_limpio.shape[0]}")
```

Aquí vereficamos outliers para nuestras variables asignadas ya que en la estadistica descriptiva, habia un cambio muy significativo entre el 75 % y el maximo cuartil, nos encargamos de transformar estos outliers en valores que puedan ser utilizados en la regresión y estandarizamos en base a los limites y nos regresa el número de filas totales despues de quitar y transformar outliers
Obetnemos el siguiente output

Figura 3: Transformación de Outliers

```
    """
␣␣␣␣Construccion␣de␣un␣modelo␣de␣regresion␣lineal␣sin␣framework
␣␣␣␣"""

    # Convertir a numpy arrays
    X = df["mmr"].to_numpy().reshape(-1, 1)  # Caracteristica
    y = df["sellingprice"].to_numpy()        # Variable objetivo


    # Calcular medias
    x_mean = X.mean()
    y_mean = y.mean()

    # Restar las medias
    X_centered = X - x_mean
    y_centered = y - y_mean

    # Calcular la pendiente (beta1) y la interseccion (beta0)
    numerador = (X_centered.flatten() * y_centered).sum()
    denominador = (X_centered.flatten() ** 2).sum()
    beta1 = numerador / denominador
    beta0 = y_mean - beta1 * x_mean

    # Realizar predicciones
    y_pred = beta0 + beta1 * X

    # Evaluar el modelo
    mse = ((y - y_pred.flatten()) ** 2).mean()
    r2 = 1 - (ss_residual / ss_total)

    # Imprimir resultados
    print(f"Error␣cuadratico␣medio␣(MSE):␣{mse}")
    print(f"Coeficiente␣de␣determinacion␣(R^2):␣{r2}")
```

Hacemos el modelo de regresión utilizando lo siguiente: Convertimos las variables que elegimos a arreglos de numpy, calculamos las medias y las restamos al arreglo calculamos la pendiente y la intersección el numerador lo obtenemos con el la suma del producto de X despues de aplica el metodo flatten que regresa un vector de 1 dimensión por y, despues para el denominador, sera la suma del vector X al cuadrado, despues obtenemos las pendientes con las formulas de pendiente. Realizamos las predicciones con la ecuacion. Obetenmos el error cuadratico medio y la r$\hat{2}$ para evaluar el modelo
El output:

```
Error cuadrático medio (MSE): 3085713.241441639
Coeficiente de determinación (R^2): 0.9675361060680938
```

Figura 4: Evaluación de Regresión

```
"""
Visualizacion de los datos
"""

# Configuracion de graficos
plt.figure(figsize=(10, 6))

# Graficar los puntos de datos reales
plt.scatter(X, y, color='blue', label='Datos reales')

# Graficar la linea de regresion
plt.plot(X, y_pred, color='red', linewidth=2, label='Linea de regresion')

# Etiquetas y titulo
plt.xlabel('Precio de Mercado (mmr)')
plt.ylabel('Precio de Venta (sellingprice)')
plt.title('Regresion Lineal: Precio de Mercado vs. Precio de Venta')
plt.legend()

# Mostrar el grafico
plt.show()
```

Creamos el grafico con los valores reales y la linea de regresión para visualizar los datos resultantes
Grafico:

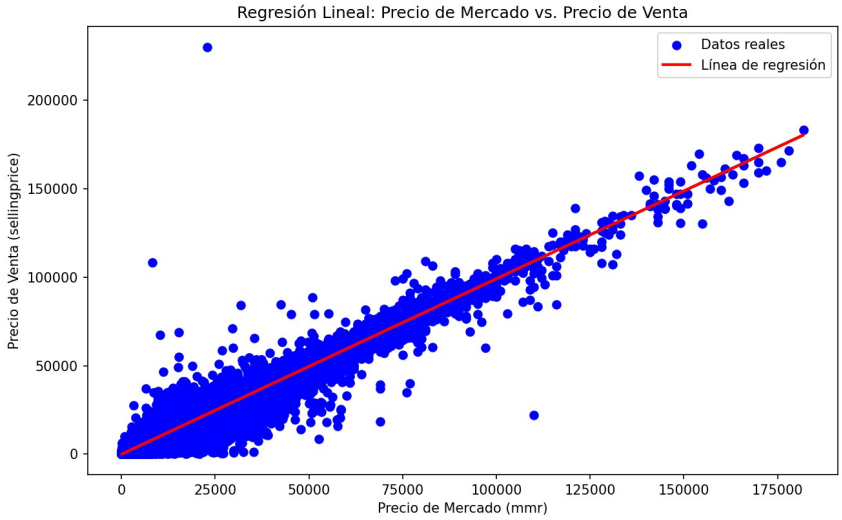

Figura 5: Grafico de Regresión

```
    """
    Prediccion
    """

␣␣␣␣#␣Definir␣un␣nuevo␣valor␣de␣mmr␣para␣la␣prediccion
␣␣␣␣nuevo_mmr␣=␣float(input('Ingresa␣el␣valor␣de␣mmr␣que␣consideres:␣'))
␣␣␣␣␣␣␣␣#␣Inputa␣este␣valor␣con␣el␣valor␣que␣deseas␣predecir

␣␣␣␣#␣Calcular␣el␣precio␣de␣venta␣predicho␣usando␣la␣formula␣de␣la␣regresion
␣␣␣␣precio_predicho␣=␣beta0␣+␣beta1␣*␣nuevo_mmr

␣␣␣␣#␣Imprimir␣el␣resultado
␣␣␣␣print(f"El precio de venta predicho para un mmr de {nuevo_mmr} es {precio_predicho:.2f}")
```

Finalmente hacemos una predicción de valor de venta para un mmr especifico con entrada de usuario Output:

```
Ingresa el valor de mmr que consideres: 17500
El precio de venta predicho para un mmr de 17500.0 es 17307.44
```

Figura 6: Predicción

Esto fue todo el código de regresión lineal, la evaluación del modelo pudo ser mejor, quizas utilizando sklearn podriamos obtener un mejor resultado, ya que cuando hice la practica con Mathematica los MSE eran más bajos, pero quizas puede ser un tema de estandarización. Fue una practica divertida para entender a fondo como funcionan algunos algoritmos clasicos en su forma más cruda y con las operaciones que hay por detras.